



LUND UNIVERSITY

Machine Learning for Perception and Localization: Efficient and Invariant Methods

Berg, Axel

2024

Document Version:

Publisher's PDF, also known as Version of record

[Link to publication](#)

Citation for published version (APA):

Berg, A. (2024). *Machine Learning for Perception and Localization: Efficient and Invariant Methods*. [Doctoral Thesis (compilation), Centre for Mathematical Sciences]. Centre for Mathematical Sciences, Lund University.

Total number of authors:

1

General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Machine Learning for Perception and Localization

Efficient and Invariant Methods

AXEL BERG



Lund University
Faculty of Engineering
Centre for Mathematical Sciences
Mathematics



Machine Learning for Perception and Localization

Efficient and Invariant Methods

Machine Learning for Perception and Localization

Efficient and Invariant Methods

by Axel Berg



LUND
UNIVERSITY

Thesis for the degree of Doctor of Philosophy in Engineering

Thesis advisors:

Assoc. Prof. Magnus Oskarsson, Dr. Chuteng Zhou, Prof. Kalle Åström

Faculty opponent:

Assoc. Prof. Jesper Rindom Jensen
Aalborg University

To be presented, with the permission of the Faculty of Engineering of Lund University, for public criticism in Hörmandersalen (MH:H) at the Centre for Mathematical Sciences on the 7:th of February 2025 at 13:15.

Organization LUND UNIVERSITY Centre for Mathematical Sciences Box 118 SE-221 00 LUND Sweden		Document name Doctoral Thesis	
Author(s) Axel Berg		Date of presentation 2025-02-07	
		Sponsoring organization Arm, WASP	
Title and subtitle Machine Learning for Perception and Localization: Efficient and Invariant Methods			
Abstract <p>This thesis covers a set of methods related to machine perception and localization, which are two important building blocks of artificial intelligence. In Paper I, we explore the concept of regression via classification (RvC), which is often used for perception tasks where the target variable is either ordinal or when the distance metric of the target space is not well-suited as an objective function. However, it is not clear how the discretization of the target variable ought to be done. To this end, we introduce the concept of label diversity and propose a new loss function based on concepts from ensemble learning that can be used for both ordinal and continuous targets.</p> <p>Papers II and III deal with applying the concept of self-attention to different data domains. In Paper II we focus on point clouds, which are modeled as unordered sets in 3D space. Although applying self-attention to sets is straightforward, we find that this mechanism in itself is not enough to improve feature learning. Instead, we propose a hierarchical approach inspired by graph neural networks, where self-attention is applied to both patches of points, and to points within the patches. This results in improved predictive performance and reduced computational cost, while preserving invariance to permutations of points in the set.</p> <p>In Paper III we explore the use of self-attention for auditory perception. Using a simple Transformer architecture, we achieve state-of-the-art performance for speech classification. However, deploying speech recognition models in real world scenarios often involves making trade-offs between predictive performance and computational costs. In Paper IV, we therefore explore floating point quantization of neural networks in the context of federated learning and propose a new method that allows training to be performed on low-precision hardware. More specifically, we propose a method for quantization aware training and server-to-device communication in 8-bit floating point. This allows for a significant reduction in the amount of data that needs to be communicated during the training process. Building upon the results in Paper III, we also show that our Transformer-based model can be quantized and trained in a realistic federated speech recognition setup and still achieve good performance.</p> <p>Papers V, VI and VII also deal with auditory perception, but from the localization point of view. This involves processing signals from microphone arrays and extracting spatial cues that enable the system to infer the location of the sound source. One such cue is the time difference of arrival (TDOA), which is estimated by correlating signals from different pairs of microphones. However, measuring TDOA in adverse acoustical conditions is difficult, which motivates the use of machine learning for this task. In Paper V, we propose a learning-based extension of a classical method for TDOA estimation that improves prediction accuracy, while simultaneously preserving some of the properties of the classical method. This is achieved by using a model architecture that is equivariant to time shifts together with an RvC training objective.</p> <p>TDOA estimates are often used as input to sound source localization (SSL) systems. In Paper VI, we extend the method from Paper V to predict TDOAs from multiple overlapping sound sources and show that this is a good pre-training task for extracting correlation features to an SSL system, with improved localization performance compared to popular handcrafted input features.</p> <p>In Paper VII, we instead focus on a single sound source, but with variable number of microphones in the array. Most machine learning methods for SSL are trained using a specific microphone array setup and will not work if a microphone is turned off or moved to a different position. We solve this problem by modeling pairs of audio recordings and microphone coordinates as nodes in a multi-modal graph. This enables the use of an attention-based autoencoder model that infers the location of the sound source using both microphone coordinates, i.e. a set of points in 3D space, and audio features, while preserving invariance to permutations of microphones. Furthermore, we address variants of the problem where data is partially missing, such as signals from a microphone at an unknown location.</p>			
Keywords neural networks; deep learning; machine perception; ordinal regression; shape recognition; transformer; audio classification; quantization; sound source localization			
Classification system and/or index terms (if any)			
Supplementary bibliographical information		Language English	
ISSN and key title Doctoral Thesis in Mathematical Sciences 1404-0034		ISBN 978-91-8104-314-3 (print) 978-91-8104-315-0 (pdf)	
Recipient's notes	Number of pages 260	Price	
	Security classification		

I, the undersigned, being the copyright owner of the abstract of the above-mentioned dissertation, hereby grant to all reference sources the permission to publish and disseminate the abstract of the above-mentioned dissertation.

Signature _____

Date 2024-12-16 _____

Machine Learning for Perception and Localization

Efficient and Invariant Methods

by Axel Berg



LUND
UNIVERSITY

Cover illustration: waveform of a recording
of the author uttering the sentence
“Machine Learning for Perception and Localization”.

pp. i-91	©	2024, Axel Berg
Paper I	©	2021, IEEE
Paper II	©	2022, IEEE
Paper III	©	2021, ISCA
Paper IV	©	2024, The Authors
Paper V	©	2022, ISCA
Paper VI	©	2024, The Authors (CC BY 4.0)
Paper VII	©	2024, IEEE

Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

Doctoral Thesis in Mathematical Sciences 2024:5
ISSN: 1404-0034
ISBN: 978-91-8104-314-3 (print)
ISBN: 978-91-8104-315-0 (electronic)
LUTFMA-1088-2024

Printed in Sweden by Media-Tryck, Lund University, Lund 2024



Media-Tryck is a Nordic Swan Ecolabel
certified provider of printed material.
Read more about our environmental
work at www.mediatryck.lu.se

MADE IN SWEDEN 

*Dedicated to my parents
Karin and Roger*

Abstract

This thesis covers a set of methods related to machine perception and localization, which are two important building blocks of artificial intelligence. In Paper I, we explore the concept of regression via classification (RvC), which is often used for perception tasks where the target variable is either ordinal or when the distance metric of the target space is not well-suited as an objective function. However, it is not clear how the discretization of the target variable ought to be done. To this end, we introduce the concept of label diversity and propose a new loss function based on concepts from ensemble learning that can be used for both ordinal and continuous targets.

Papers II and III deal with applying the concept of self-attention to different data domains. In Paper II we focus on point clouds, which are modeled as unordered sets in 3D space. Although applying self-attention to sets is straightforward, we find that this mechanism in itself is not enough to improve feature learning. Instead, we propose a hierarchical approach inspired by graph neural networks, where self-attention is applied to both patches of points, and to points within the patches. This results in improved predictive performance and reduced computational cost, while preserving invariance to permutations of points in the set.

In Paper III we explore the use of self-attention for auditory perception. Using a simple Transformer architecture, we achieve state-of-the-art performance for speech classification. However, deploying speech recognition models in real world scenarios often involves making trade-offs between predictive performance and computational costs. In Paper IV, we therefore explore floating point quantization of neural networks in the context of federated learning and propose a new method that allows training to be performed on low-precision hardware. More specifically, we propose a method for quantization aware training and server-to-device communication in 8-bit floating point. This allows for a significant reduction in the amount of data that needs to be communicated during the training process. Building upon the results in Paper III, we also show that our Transformer-based model can be quantized and trained in a realistic federated speech recognition setup and still achieve good performance.

Papers V, VI and VII also deal with auditory perception, but from the localization point of view. This involves processing signals from microphone arrays and extracting spatial cues that enable the system to infer the location of the sound source. One such cue is the time difference of arrival (TDOA), which is estimated by correlating signals from different pairs of microphones. However, measuring TDOA in adverse acoustical conditions is difficult, which motivates the use of machine learning for this task. In Paper V, we propose a learning-based extension of a classical method for TDOA estimation that improves prediction accuracy, while simultaneously preserving some of the properties of the classical

method. This is achieved by using a model architecture that is equivariant to time shifts together with an RvC training objective.

TDOA estimates are often used as input to sound source localization (SSL) systems. In Paper VI, we extend the method from Paper V to predict TDOA's from multiple overlapping sound sources and show that this is a good pre-training task for extracting correlation features to an SSL system, with improved localization performance compared to popular handcrafted input features.

In Paper VII, we instead focus on a single sound source, but with variable number of microphones in the array. Most machine learning methods for SSL are trained using a specific microphone array setup and will not work if a microphone is turned off or moved to a different position. We solve this problem by modeling pairs of audio recordings and microphone coordinates as nodes in a multi-modal graph. This enables the use of an attention-based autoencoder model that infers the location of the sound source using both microphone coordinates, i.e. a set of points in 3D space, and audio features, while preserving invariance to permutations of microphones. Furthermore, we address variants of the problem where data is partially missing, such as signals from a microphone at an unknown location.

List of Publications

The contents of this thesis is based on the following publications:

- Paper I **Deep Ordinal Regression with Label Diversity**
Axel Berg, Magnus Oskarsson, Mark O'Connor
Proc. 2020 25th International Conference on Pattern Recognition (ICPR),
2021, pp. 2740-2747
- Paper II **Points to Patches: Enabling the Use of Self-Attention for 3D Shape Recognition**
Axel Berg, Magnus Oskarsson, Mark O'Connor
Proc. 2022 26th International Conference on Pattern Recognition (ICPR),
2022, pp. 528-534
- Paper III **Keyword Transformer: A Self-Attention Model for Keyword Spotting**
Axel Berg, Mark O'Connor, Miguel Tairum Cruz
Proc. Interspeech 2021, pp. 4249-4253
- Paper IV **Towards Federated Learning with on-device Training and Communication in 8-bit Floating Point**
Bokun Wang, Axel Berg, Durmus Alp Emre Acar, Chuteng Zhou
Submitted. A shorter version of this paper was presented at FedKDD: International Joint Workshop on Federated Learning for Data Mining and Graph Analytics, 2024.
- Paper V **Extending GCC-PHAT using Shift Equivariant Neural Networks**
Axel Berg, Mark O'Connor, Kalle Åström, Magnus Oskarsson
Proc. Interspeech 2022, pp. 1791-1795
- Paper VI **Learning Multi-Target TDOA Features for Sound Event Localization and Detection**
Axel Berg, Johanna Engman, Jens Gulin, Kalle Åström, Magnus Oskarsson
Proc. Detection and Classification of Acoustic Scenes and Events 2024 Workshop (DCASE2024), pp. 16-20
- Paper VII **wav2pos: Sound Source Localization using Masked Autoencoders**
Axel Berg, Jens Gulin, Mark O'Connor, Chuteng Zhou, Kalle Åström, Magnus Oskarsson
Proc. 2024 14th International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-8

This thesis is an extension of my Licentiate thesis *Applications of Diversity and the Self-Attention Mechanism in Neural Networks* [15], which includes Papers I – III.

Subsidiary Publications

I have also made contributions to the following publication, which is not considered to be a part of the thesis.

The LU System for DCASE 2024 Sound Event Localization and Detection Challenge

Axel Berg, Johanna Engman, Jens Gulin, Kalle Åström, Magnus Oskarsson
DCASE2024 Challenge. Technical Report.

Paper VI is an extension of this work.

Acknowledgements

I would like to give sincere thanks to my main supervisor Magnus Oskarsson for all the advice and helpful discussions during the course of my studies. Likewise, I would like to thank my industrial supervisor at Arm, Chu Zhou, and my academic co-supervisor Kalle Åström. I am also grateful to Mark O'Connor, who was my industrial supervisor at Arm during the first years of my studies. This dissertation would not have been possible without the guidance from all of you.

The contents of this thesis is to a large extent the result of collaborative work, for which I am very grateful. I would therefore like to give a special thanks to my other co-authors Jens Gulin, Johanna Engman, Bokun Wang, Alp Acar and Miguel Tairum-Cruz, for working together with me on my research projects. In addition, I would like to thank everyone in the computer vision and machine learning group (CVML) at the Centre for Mathematical Sciences for all the discussions and feedback, and also for providing a fun and stimulating research environment. A special thanks to my office mate Martin Trimmel for providing me with good company during my studies.

The opportunity for me pursue doctoral studies had not been possible without the support from many people at Arm, both in Lund and abroad. In particular, I would like to thank Fredrik Nordström, Johan Grönqvist, Paul Whatmough, Magnus Midholt and Fredrik Knutsson. I would also like to acknowledge the WASP community and in particular the members of the geometric deep learning cluster for valuable meetings and discussions.

Finally, I would like to express my gratitude to my family, and especially my wife Anna, for supporting me in pursuing my studies.

Axel Berg, November 2024

Funding

This work was financially supported by Arm and by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg (KAW) Foundation. Computational infrastructure was provided by Arm and by the KAW Foundation at the National Supercomputer Centre in Sweden. Travel costs were partially supported by the Royal Physiographic Society in Lund.

Popular summary

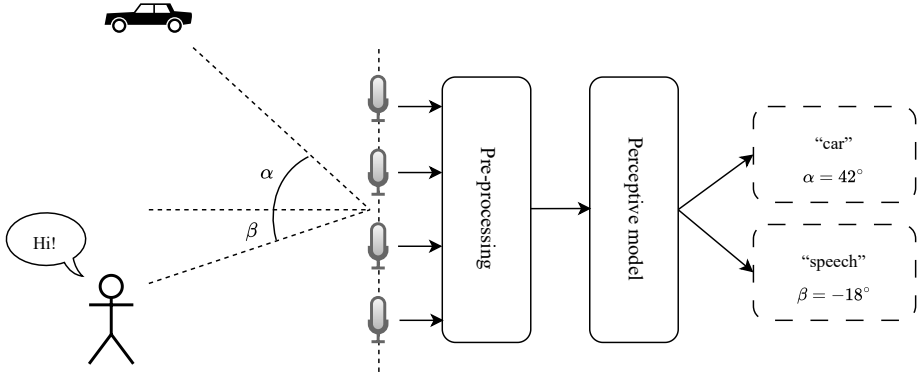
Perception is an important building block in artificial intelligence (AI) systems which allows them to process sensory information from the outside world. The “senses” of an AI system are the sensors that convert physical phenomena into digital signals, e.g. cameras, microphones and antennas. The perception consists of interpreting these signals by extracting the relevant information, such as the semantic content of the images captured by the camera or the words in a speech recording captured by the microphones. Depending on the use case of the system, the information can be used in different ways. For example, an autonomous robot might use perception in order to navigate through its surroundings. In other use cases, perceptive AI systems can assist human decision making, such as in advanced driver-assistance systems used in many modern vehicles.

Today, most state of the art perceptive methods are built using machine learning, i.e. statistical algorithms, or “models”, that learn representations of the signals by training on interpreting large amounts of data. In this thesis, a set of new machine learning methods for perception tasks are introduced, with focus on auditory perception. Designing such methods includes training models for classifying different types of sound recordings.

In some scenarios, it is necessary to also be able to localize the sound events in 3D space. Sound source localization has several important applications within the fields of robotics and navigation, and there are also military applications. For this task, it is typically necessary to work with multiple audio recordings from a synchronized microphone array. This is analogous to how humans benefit from having two ears when localizing sounds. Training a localization model for this task therefore involves processing multichannel audio and extracting spatial cues that allows the model to infer the locations of the sound events.

When designing machine learning models for perception, there are several evaluation criteria that ought to be considered. Perhaps most importantly, the inferences made by the system ought to be as accurate as possible. However, just as is the case in human perception, there will always be a small probability of making incorrect inferences. For example, the system might not be able to understand speech with poor pronunciation. Other factors, such as background noise and interference can also make it difficult for the system to understand the data. Therefore it is important to design the training data for the perceptive model in such a way that it contains all the scenarios required to be managed by the system.

When using AI systems in real-world scenarios it is also necessary to consider the limited hardware capabilities of the machine on which the model is being deployed. In recent years, there has been an accelerating trend for models to become more complex and have larger number of trainable parameters. This increases both the memory footprint of the model and possibly also the time required to perform inferences. In this thesis, the perceptive models are therefore analyzed from a computational perspective as well, and solutions for



Conceptual overview of an auditory perceptive system that simultaneously classifies and localizes multiple audio events. Audio signals are recorded by a microphone array and after some pre-processing, they are sent to the perceptive model which infers information about the events and their location. In this thesis, we propose learning-based methods for solving these types of problems.

reducing model complexity are proposed.

In the illustration shown above, we show a simplified example of one of the perceptive tasks considered in the thesis. Here, the goal is to train a model that can detect, classify, localize and track different sound events over time. Most approaches to this problem rely on a pre-processing stage that extracts two types of hand-crafted audio features: 1) spectral features that are good for distinguishing different types of sounds and 2) spatial features that contain information about the location of the sounds source. However, these spatial features do not deal adequately with overlapping sounds and noisy environments, leading to incorrect inferences. To overcome this, a new spatial feature that can be trained to adapt to different environments and learn to separate overlapping sound events is introduced in this thesis. We test this method on real-world audio recordings and the results show that our learning-based approach to feature extraction yields better localization performance than the hand-crafted one.

Machine learning methods have proven to be very successful at solving problems related to perception due to their ability to model reality by learning from data. Nevertheless, the performance of the methods do not only depend on the quality of the data, but also the structure of the models and how the learning procedure is implemented. How should we design the models in order efficiently extract relevant information from the data? One answer is that the model design ought to be adapted to the underlying structures, or symmetries, of the inputs and outputs to the model, which depend not only upon the data, but also on the particular task the model is being trained for. Continue reading the thesis in order to find out how symmetries can be exploited in order to improve the performance of perceptive models.

Populärvetenskaplig sammanfattning

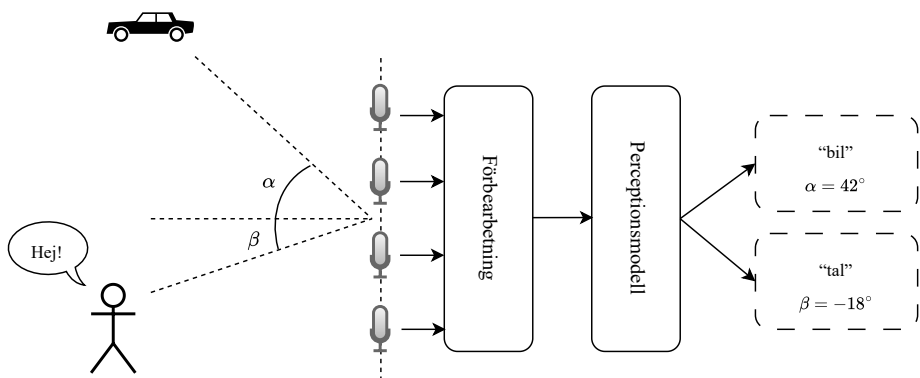
Perception är en viktig komponent i system baserade på artificiell intelligens (AI) som möjliggör bearbetning av signaler från olika sensorer. AI-systemets *sinnen* utgörs av sensorer som omvandlar fysikaliska fenomen till digitala signaler, till exempel kameror, mikrofoner och antenner. Perceptionen består av att tolka dessa signaler genom att extrahera relevant information, så som det semantiska innehållet i kamerabilder eller orden i en talinspelning från mikrofoner. Beroende på systemets användningsområde kan informationen användas på olika sätt. Om systemet används för att styra en autonom robot, kan perceptionen exempelvis användas för att möjliggöra navigering i omgivningen. Perceptiva AI-system kan också användas för att assistera människors beslutsfattande, vilket är vanligt förekommande i bland annat förarassistanssystem.

Numera konstrueras de flesta perceptiva metoder med hjälp av maskininlärning, som bygger på statistiska *modeller* som kan lära sig representationer av signaler genom att träna på stora mängder data. I denna avhandling introduceras flera nya maskininlärningsmetoder, med fokus på bearbetning av ljudsignaler. Dessa metoder innefattar träning av modeller för att klassificera olika typer av ljud från mikrofoninspelningar.

I vissa scenarier kan det vara nödvändigt att även kunna lokalisera ljudkällor i systemets omgivning. Ljudlokalisering har flera viktiga tillämpningar inom bland annat robotik och navigering, och det finns även militära tillämpningar. För att åstadkomma noggrann lokalisering är det nödvändigt att använda en uppsättning med flera mikrofoner, precis som att vår egen förmåga att lokalisera ljud till stor del är beroende av att vi har två öron. Att träna en modell för lokalisering innebär således att behandla en uppsättning signaler och extrahera spatial information som gör det möjligt att uppskatta ljudkällans position.

Det finns ett flertal olika sätt att utvärdera prestandan för perceptiva maskininlärningsmodeller som bör beaktas i designprocessen. Ett viktigt kriterium är att slutledningsförmågan är tillräckligt bra för att systemet ska kunna användas i verklighetstroga användningsområden. Därför är det nödvändigt att träna modellen på en datamängd som innehåller även svåra och ovanliga fall. För en taligenkänningsmodell kan detta innebära exempelvis inspelningar med otydligt uttal eller mycket bakgrundsbrus.

För att ett AI-system skall vara användbart i realistiska scenarier, är det även nödvändigt att kunna implementera det på hårdvara med begränsad beräkningskapacitet. De senaste åren har det emellertid funnits en trend där dessa system blivit mer och mer beräkningsintensiva, vilket till stor del beror på att antalet modellparametrar ökat. Ett viktigt fokusområde i avhandlingen är därför att designa och utvärdera metoder även utifrån ett komplexitetsperspektiv.



Konceptuell överblick av ett perceptionssystem som samtidigt klassificerar och lokaliserar flera ljudhändelser. Inspelningarna från mikrofonerna förbehandlas och bearbetas därefter av en perceptiv modell som klassificerar ljudhändelserna och uppskattar deras position. I denna avhandling introducerar vi inlärningsbaserade metoder för att lösa denna typ av problem.

Illustrationen ovan visar ett förenklat exempel på ett av problemen som behandlas i avhandlingen. Problemet består i att träna en modell som kan detektera, klassificera, lokalisera och spåra olika typer av ljud. Det vanligaste sättet att angripa detta problem är att först använda sig av ett förbearbetningssteg som extraherar två typer av ljudrepresentationer: 1) en spektral representation som är bra för att särskilja olika slags ljud och 2) en spatial representation som innehåller information om ljudkällans position. Den spatiala representationen är emellertid ofta inte tillräckligt bra för att kunna hantera överlappande ljud och bakgrundsbrus, vilket kan leda till felaktiga slutledningar. I denna avhandling introducerar vi därför en metod för att extrahera en adaptiv representation utifrån träningsdata. Vi utvärderar denna representation på realistiska ljudinspelningar och demonstrerar bättre lokaliseringsförmåga jämfört med andra spatiala representationer.

Maskininlärningsmodeller har visat sig vara bra på att lösa perceptionsrelaterade problem på grund av deras förmåga att extrahera representationer av verkligheten från träningsdata. Likväl är deras förmåga också beroende på hur modellernas struktur och hur träningen utformas. Hur bör vi designa dessa modeller för att på bästa sätt extrahera relevant information från en datamängd? Ett möjligt svar på denna fråga är att utnyttja de underliggande symmetrierna som bestämmer förhållandet mellan indata och utdata för det specifika problemet. Läs gärna resten av avhandlingen för att få reda på hur symmetrier kan användas för att förbättra perceptiva modellers prestanda för en mängd olika problem.

List of Abbreviations

k -NN	k -nearest neighbour
ACCDOA	Activity-coupled Cartesian DOA
ADAM	Adaptive moment estimation
ADPIT	Auxiliary duplicating permutation invariant training
ASR	Automatic speech recognition
BN	Batch normalization
CE	Cross entropy
CNN	Convolutional neural network
DCT	Discrete cosine transform
DOA	Direction of arrival
FLOPS	Floating point operations per second
FMR	Feature matching recall
FP	Floating point
GCC-PHAT	Generalized cross correlation with phase transform
GNN	Graph neural network
LN	Layer normalization
MAE	Mean absolute error
MAP	Maximum a posteriori probability
MFCC	Mel-frequency cepstrum coefficients
mIoU	mean intersection-over-union
ML	Maximum likelihood
MLP	Multi-layer perceptron
MSA	Multi-head self-attention
MSE	Mean squared error

NGCC-PHAT	Neural generalized cross correlation with phase transform
PIT	Permutation invariant training
QAT	Quantization aware training
RANSAC	Random sample consensus
RDE	Relative distance error
RNN	Recurrent neural network
RvC	Regression via classification
SA	Self-attention
SALSA	Spatial cue-augmented log-spectrogram
SELD	Sound event localization and detection
SGD	Stochastic gradient descent
SLAM	Simultaneous localization and mapping
SNR	Signal-to-noise ratio
SSL	Sound source localization
TDOA	Time-difference of arrival
UQ	Unbiased quantization

Contents

Abstract	ix
List of Publications	xi
Acknowledgements	xiii
Funding	xiv
Popular summary	xv
Populärvetenskaplig sammanfattning	xvii
List of Abbreviations	xix
Background and Research Context	1
1 Introduction	3
1.1 Motivation and Research Objectives	4
1.2 Outline of the Thesis	5
2 Artificial Neural Networks and Deep Learning	7
2.1 Supervised Learning	8
2.2 Training a Neural Network	10
2.3 Federated Learning	12
2.4 Number Formats for Neural Network Training and Inference	14
3 Diversity and Ordinal Regression	19
3.1 Regression Ensembles	20
3.2 The Bias-Variance Decomposition	21
3.3 Label Binning	23
3.4 Regression via Classification	26
3.5 Ordinal Regression	29
3.6 Label Diversity	29
4 Neural Network Architectures and Input-Output Symmetries	33
4.1 Permutation Symmetry and Learning on Sets	34
4.2 Permutation Invariant Training	36
4.3 Relation Networks	37
4.4 Learning on Graphs	38
4.5 Translational Symmetry and Convolutional Neural Networks	39
4.6 Self-Attention and Transformers	40
5 Audio Recognition and Sound Source Localization	47

5.1	Audio Feature Descriptors	47
5.2	Speech Recognition and Keyword Spotting	54
5.3	Sound Source Localization	56
6	Summary of Contributions	69
6.1	Paper Contributions	69
6.2	Conclusions and Outlook	75
	References	77
Scientific Publications		91
I	Deep Ordinal Regression with Label Diversity	95
1	Introduction	95
2	Related Work	97
2.1	Methods	97
2.2	Relevant Applications	98
3	Proposed Method	100
3.1	Label Diversity by Overlapping Bins	100
3.2	Backpropagation	100
3.3	Inference	101
4	An Illustrative Example	103
5	Experiments	104
5.1	Age Estimation	105
5.2	Head Pose Estimation	106
5.3	Historical Image Dating	107
6	Conclusion	108
	References	109
II	Points to Patches: Enabling the Use of Self-Attention for 3D Shape Recognition	115
1	Introduction	115
2	Related Work	116
3	Method	118
4	Experiments	121
4.1	Shape Classification	121
4.2	Ablation Study	123
4.3	Feature Matching on 3DMatch	125
5	Conclusions	126
	References	127
III	Keyword Transformer: A Self-Attention Model for Keyword Spotting	133
1	Introduction	133
2	Related Work	135
2.1	Keyword Spotting	135
2.2	Self-Attention and the Vision Transformer	135

3	The Keyword Transformer	136
3.1	Model Architecture	136
3.2	Knowledge Distillation	137
4	Experiments	138
4.1	Keyword Spotting on Google Speech Commands	138
4.2	Ablation Studies	139
4.3	Attention Visualization	140
4.4	Latency Measurements	141
5	Conclusion	142
	References	142
IV	Towards Federated Learning with on-device Training and Communication in 8-bit Floating Point	149
1	Introduction	149
2	Related Work	151
2.1	Federated Learning with Quantized Communication	151
2.2	FP8 Quantization for Neural Networks	152
3	Method	153
3.1	Preliminaries	153
3.2	Floating Point Representation	153
3.3	On-Device Quantization-Aware Training	154
3.4	Unbiased Quantized Communication	155
3.5	Server-Side Optimization (SERVEROPTIMIZE)	157
3.6	Overall algorithm	158
4	Convergence analysis and theoretical motivations	159
5	Experiments and Ablation Studies	160
5.1	Datasets and models	160
5.2	Mixed Precision Quantization Implementation	162
5.3	Results	162
5.4	Ablation studies	164
6	Conclusions and Future Work	165
	References	166
	Appendix	170
A	Quantization Function	170
B	Convergence Analysis of Quantization-aware Training (QAT)	170
C	Convergence Analysis of FP8FedAvg-UQ	173
C.1	Lemmas on the Stochastic Quantization for Model Communication	174
C.2	Lemma on a Single Communication Round	176
C.3	Proof of the Main Theorem	182
V	Extending GCC-PHAT using Shift Equivariant Neural Networks	187
1	Introduction	187

2	Method	188
3	Experiments	192
4	Conclusions	195
5	Acknowledgments	195
	References	195
VI Learning MultiTarget TDOA Features for Sound Event Localization and De-		
	tection	201
1	Introduction	201
2	Method	203
	2.1 Background	203
	2.2 Permutation Invariant Training for TDOA Estimation	204
3	Experimental Setup	205
	3.1 Using TDOA Features for SELD	205
	3.2 Dataset and Model Training	207
4	Results	208
5	Conclusions	210
	References	210
VII wav2pos: Sound Source Localization using Masked Autoencoders		217
1	Introduction	217
2	Method	219
3	Experimental Results	223
4	Conclusions and Future Work	227
	References	227
	Appendix	232

Background and Research Context

1 Introduction

Machine perception and localization are two important aspects of artificial intelligence (AI). In this context, *perception* refers to the ability of an AI system to process and interpret sensory information, for example images and audio. As of today, most methods for machine perception are built using *machine learning*, where the perceptive system is able to approximate input-output relations by learning from a collection of data. The widespread use of machine learning has accelerated the capabilities of AI systems over the last couple of decades and the research area is now moving forward at a quick pace. In particular, the use of *artificial neural networks* has played an important part in this trend, and they are a central component in all the methods presented in this thesis.

In order for a perceptive system to be useful in a real-world application it has to fulfill certain criteria in terms of both *accuracy* and *efficiency*. The inferences should be accurate, because otherwise the system cannot be relied on for decision making. However, they should also be efficient in terms of the computational intensity, memory requirements and communication bandwidth. For example, consider an automatic speech recognition (ASR) system that provides automatic captions for an online teleconference application. Such a system is only useful if it can operate in real-time, which puts restrictions on the complexity of the system in relation to the hardware on which it is being deployed. Furthermore, a model that can run in real-time on a server or desktop computer may be slower on an edge device, for example a mobile phone. Running a system on such a device may require optimizing the structure of the perceptive system in order to reduce the computational workload.

In terms of performance evaluation, a reasonable expectation is for such a system to be at least as good as a human transcriber. But how can we compare the performance of an ASR system to a human? A common measure is the word error rate (WER). In 2016, a group of researchers first designed a system with human-level performance (5.8 % WER) on a dataset consisting of recorded telephone conversations [4]. However, the speakers were native English speakers, and subsequent research has shown that performance of ASR systems drop significantly when evaluated on non-native speakers [12]. Human-level performance has also been claimed for other perceptive tasks, including machine translation [101], image classification [59] and 3D object detection [44].

In addition to perception, some systems might be required to perform *localization*. This could involve localizing external objects in the vicinity of the object, or self-localization in relation to a known map of the environment. When both the location and the map are unknown, these need to be estimated jointly by exploiting movements, a problem known as simultaneous localization and mapping (SLAM).

SLAM systems can be used in autonomous systems, where navigation is performed using

the constructed map, or as assistants to human navigation, for example in advanced driver-assistance systems (ADAS). This is another example where efficiency is especially important, since navigation becomes difficult without real-time updates to the map. Furthermore, ADAS systems need to satisfy very stringent safety requirements. In some use cases, the system might also be required to process and interpret multiple data streams simultaneously, which requires more computational capabilities. For example, many SLAM methods for ADAS depend on inputs from both cameras and LiDAR [25].

In other scenarios, SLAM can also be performed using sound measurements from a single [71] or multiple [43] microphones. Using acoustic measurements for localization can be useful as a cheaper alternative to using cameras and they can also work in poor lighting conditions where camera sensors fail to operate, given that there are some active sound sources in the surroundings. Other applications of acoustic localization include smart teleconference systems. For example, when applying ASR to multichannel audio recordings from a microphone array, performance can be improved if the speech can be localized in space and separated from other interfering sound events by exploiting spatial diversity from the array [45, 24].

When designing methods for perception and localization, it is often useful to consider the intrinsic properties of input-output relations. For example, when evaluating a localization system, we are not interested in the order in which the detected events are arranged, only how accurate and efficient the system is at detecting and localizing the events. In other words, the evaluation method should be *invariant* to reordering of events. In some scenarios, the setup of the sensor array might be reordered, and hence we want to design a model that is invariant to reordering of the input. Such properties can either be learned implicitly from the data, or be explicitly built into the neural network architecture. In this thesis, we emphasize the latter approach while designing methods for perception and localization.

1.1 Motivation and Research Objectives

This thesis deals with problems related to both perception and localization, in a variety of different contexts and problem setups. For each task, we present learning-based methods that deal with different aspects of solving the problem, such as accuracy, computational efficiency and invariance to input and output transformations. Although the scope of this thesis is rather broad and the contributions span several different research areas within the topics of perception and localization, they are centered around common themes and motivations. Here, we briefly state some research questions that we have tried to answer during the course of this research project.

[RQ1] What objective functions are suitable for perception and localization? More specifically, we examine different objective functions for problems where the target variable

can be regarded as either continuous or categorical. In addition, we investigate different ways that regression problems can be modeled as ordinal classification problems.

- [RQ2] How can the Transformer architecture be applied to different data modalities, such as audio and point clouds?** In recent years, the Transformer has become the most widely adopted neural network architecture for many categories of problems. First proposed in 2017 [131], it quickly became popular in the natural language processing research community and it has contributed greatly towards the development and success of large language models. However, it was not until the Vision Transformer [40] was proposed in 2020 that it became popular in the computer vision community. One of our research objectives is to leverage the potential of adopting Transformers in other domains and compare their performance with more traditional architectures.
- [RQ3] Can these models be used without excessive computational footprint?** Transformers are known for achieving high benchmark scores on many tasks, but often this is achieved by increasing the number of parameters compared to other models, which is undesirable in many use cases. Can we achieve the same model performance without requiring more parameters and computations?
- [RQ4] How can machine learning be applied to sound source localization problems?** Although some aspects of sound source localization using machine learning have been widely explored, there are other scenarios where classical methods are still widely used. This is a great research opportunity, since deep learning has been very successful in other aspects of audio processing. Localization performance is often limited by adverse acoustic conditions in the environment, which is something that adaptive methods can potentially alleviate. We therefore investigate properties of the classical localization methods and try to develop machine learning methods that can replace components of the classical methods in order to improve performance.
- [RQ5] How can intrinsic properties of perception and localization problems, such as invariance and equivariance, be exploited to design better models for perception and localization?** Choosing the neural network architecture is an important step in designing perceptive systems. We investigate different choices in representations of the inputs and outputs, as well as the networks internal representations. We also explore how concepts from geometric deep learning, such as invariance and equivariance, can be exploited in the design process of machine learning models.

1.2 Outline of the Thesis

Section 2 briefly covers some of the basic concepts related to neural networks and supervised learning that lay the foundation for the rest of the thesis. In Section 3, we cover the background of some specific problems related to the choice of objective function for

regression and classification problems. We then proceed to discuss some neural network architectures and their properties in Section 4. In Section 5 we discuss audio processing and sound source localization from a machine learning perspective. Finally, in Section 6 we highlight the main contributions of the thesis and present the main results of the included publications.

2 Artificial Neural Networks and Deep Learning

Feature extraction is an important step in pattern recognition that deals with finding useful representations of data, for example images or audio. These representations can then be used to solve downstream tasks. Sometimes it is possible to design hand-crafted feature extractors based on human intuitions [33], however in recent years they have to a large extent fallen out of favor and been replaced by machine learning models that learn to extract features based on patterns in large data collections. In recent years, the use of neural networks as feature extractors has accelerated progress in many fields. In this section, we provide a brief introduction to some important concepts in deep learning.

An artificial neural network consists of a set of neurons, with corresponding weights and activation functions, that form a computational graph. A network $f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y}$, parameterized by a set of parameters θ , defines a mapping from the input space \mathcal{X} to the output space \mathcal{Y} . The simplest form of neural network is the perceptron [107], which was originally inspired by neurons in the brain. It maps an input vector $\mathbf{x} = [x^{(1)}, \dots, x^{(d)}]^T$ to an output y , using a weighed sum

$$y = \sigma(\mathbf{x}^T \mathbf{w} + b). \quad (1)$$

Here σ is a non-linear function, also known as the activation function. The perceptron is parameterized by the weight vector $\mathbf{w} = [w^{(1)}, \dots, w^{(d)}]^T$ and the bias b , which can be modified in order for the perceptron to compute different functions.

By combining multiple layers of perceptrons, we get a directed computational graph which is referred to as a multi-layer perceptron (MLP) [108], as shown in Figure 3. The neurons in-between the input and the output layer are “hidden” layers and their outputs store intermediate values that are propagated forward through the network. By increasing the number of hidden layers, we arrive at what is commonly referred to as a deep artificial neural network, although there is no strict definition on where to draw the line between shallow and deep networks.

Under certain conditions, MLPs are universal function approximators, in the sense that they can approximate any continuous function with arbitrary precision. In theory, a single hidden layer with sufficient width is enough to achieve this property [31]. A similar result can be shown for or a single neuron per layer with residual connections and sufficient depth [79], but in practice most networks are designed by finding a good trade-off between width and depth [53].

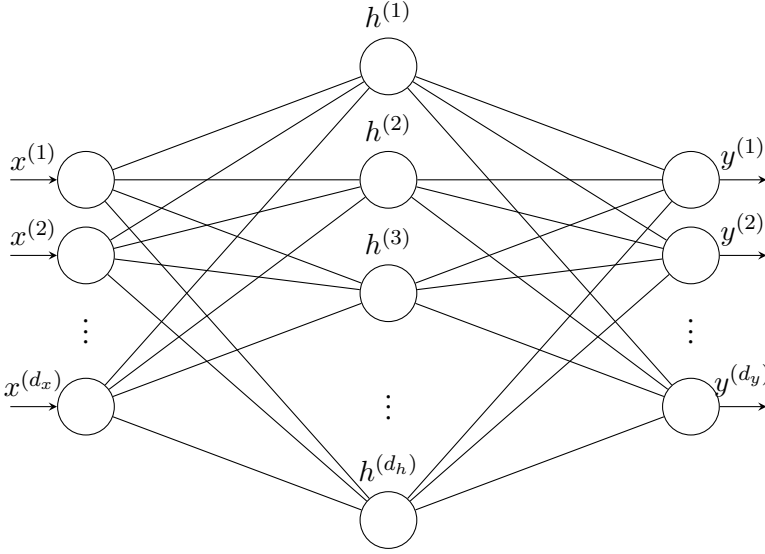


Figure 3: An MLP with a single hidden layer. The inputs $x^{(1)}, \dots, x^{(d_x)}$ are first propagated to the hidden layer, where activations $h^{(1)}, \dots, h^{(d_h)}$ are computed. These are then propagated to the final layer, where the output $y^{(1)}, \dots, y^{(d_y)}$ of the neural network is obtained.

2.1 Supervised Learning

When performing function approximation using a neural network, a learning rule is necessary to update the parameters of the network. In the context of supervised learning, this is done by minimizing a loss function, which depends on the problem type. More specifically, given a set of input and output pairs $\mathcal{S} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$, we want to update the parameters of our network such that we minimize some objective function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$, which often is referred to as the loss function. Under the assumption that the data samples are independent and identically distributed (i.i.d.), we seek to optimize the parameters $\boldsymbol{\theta}$ such that the likelihood $\log p_{\boldsymbol{\theta}}(t_n | \mathbf{x}_n)$ of the observed data is maximized. This is equivalent to minimizing the negative log-likelihood:

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} \prod_{n=1}^N p_{\boldsymbol{\theta}}(t_n | \mathbf{x}_n) = \arg \min_{\boldsymbol{\theta}} - \sum_{n=1}^N \log p_{\boldsymbol{\theta}}(t_n | \mathbf{x}_n). \quad (2)$$

This is known as the maximum likelihood (ML) estimate and it can take various forms depending on how we choose to model the data distribution. For example, assume that the underlying process generating the data can be modeled as $t_n = f_{\boldsymbol{\theta}}(\mathbf{x}_n) + \epsilon_n$, where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. random variables sampled from a Gaussian distribution and $f_{\boldsymbol{\theta}}$ is our neural network model. Our ML estimate in (2) then simplifies to

$$\begin{aligned}
\boldsymbol{\theta}_{\text{ML}} &= \arg \min_{\boldsymbol{\theta}} - \sum_{n=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(t_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2}{2\sigma^2}\right) \\
&= \arg \min_{\boldsymbol{\theta}} \frac{N}{2} \log 2\pi\sigma^2 + \sum_{n=1}^N \frac{1}{2\sigma^2} (t_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2.
\end{aligned} \tag{3}$$

Ignoring the constants, we find that maximizing the likelihood corresponds to minimizing the squared error between the network predictions and the ground truth data labels. In this case it is therefore natural to use the mean squared error as loss function:

$$\mathcal{L}_{\text{MSE}}(\mathbf{x}, t; \boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N (t_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n))^2. \tag{4}$$

When choosing the neural network architecture for this problem, which is a form of non-linear regression, the last layer of the network has a single output neuron such that the predictions $y_n = f_{\boldsymbol{\theta}}(\mathbf{x}_n) \in \mathbb{R}$. Furthermore, the regression problem can be easily extended to higher dimensions.

In other scenarios, networks are used for multi-way classification problems where the targets $t_n \in \{1, \dots, K\}$ belong to a set of discrete categories. The network architecture then has to be modified to use K output neurons, where each neuron models the probability $p_{\boldsymbol{\theta}}(k|\mathbf{x}_n)$ of \mathbf{x}_n being an instance of class k . In order to generate a probability distribution over classes, the output of the last hidden layer \mathbf{h} is typically normalized using a softmax function as

$$\text{softmax}(\mathbf{h})^{(k)} = \frac{e^{h^{(k)}}}{\sum_{j=1}^K e^{h^{(j)}}}. \tag{5}$$

This guarantees that the predicted probabilities are non-negative and sum to 1. While there are other functions with these properties, the softmax is most commonly used because it has several desirable properties, such as invariance to additive scalars. Furthermore, the softmax can be interpreted as a smooth differentiable approximation of the $\arg \max$ function [53].

Although the performance of a classification model is usually evaluated using its accuracy, i.e. the fraction of correctly classified data samples, this objective is difficult to optimize in practice. Therefore it is more common to use a loss function that measures the similarity between the predicted distribution and the ground truth distribution $q(k|\mathbf{x}_n)$, such as the cross-entropy (CE), which for an individual data sample is defined as

$$H(q, p_{\theta}) = - \sum_{k=1}^K q(k|\mathbf{x}_n) \log p_{\theta}(k|\mathbf{x}_n). \quad (6)$$

Since the data points belong to a set of discrete classes, we can use a one-hot encoding to model the ground truth distribution:

$$q(k|\mathbf{x}_n) = \begin{cases} 1, & t_n = k \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Note that the choice of $q(k|\mathbf{x}_n)$ need not be a one-hot distribution. For example, label smoothing [123], which is a form of network regularization, uses a smoothing parameter α that penalizes over-confident predictions

$$q(k|\mathbf{x}_n) = \begin{cases} 1 - \alpha, & t_n = k \\ \frac{\alpha}{K-1}, & \text{otherwise.} \end{cases} \quad (8)$$

For ordinal regression problems, where the classes can be ranked on an ordinal scale, there are other several other possible label encodings, which are further discussed in Section 3.

There exists an intimate connection between the CE loss and the ML estimation technique. Using the one-hot encoding in 7, we can calculate the average cross-entropy across the entire data set as

$$\mathcal{L}_{\text{CE}}(\mathbf{x}, t; \theta) = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K q(k|\mathbf{x}_n) \log p_{\theta}(k|\mathbf{x}_n) = -\frac{1}{N} \sum_{n=1}^N \log p_{\theta}(t_n|\mathbf{x}_n), \quad (9)$$

where $p_{\theta}(t_n|\mathbf{x}_n)$ here should be interpreted as the likelihood of the parameters θ given the observations (\mathbf{x}_n, t_n) . By comparing this with equation (2), we can conclude that under these assumptions, minimizing the average cross entropy is in fact equivalent to minimizing the average negative log-likelihood.

2.2 Training a Neural Network

Training a neural network is an optimization problem where the goal is to minimize the expected loss $\mathbb{E}_{(\mathbf{x}_n, t_n) \sim \mathcal{S}}[\mathcal{L}(\mathbf{x}_n, t_n; \theta)]$ for the entire data set. Ideally, if the network is able

to recognize patterns in the training data, and consequently minimize the loss function, it will be able to infer the same patterns on data not seen during training. An assumption is that the unseen data is drawn from the same underlying distribution as the training data. For example, if a neural network has been trained to classify images of cats and dogs, it will hopefully generalize to unseen examples of cats and dogs. However, if the training set only contains dogs of specific races, e.g. Labrador Retrievers, the classification performance might not generalize well to images of Chihuahuas.

Another common problem in neural network training is poor generalization due to memorization of training examples, also referred to as overfitting. This can occur when the parameter space is too large, which in practice often means that the network is too deep (has too many layers) or too wide (each layer has too many parameters). Therefore, an obvious way to regularize a network is to make it smaller, but in practice it has been shown that over-parameterized networks perform better even on simpler tasks [144, 90]. Instead, different regularization strategies are thus applied during training, which can be done for example by modifying the loss function to penalize memorization of training examples. For instance, weight decay (WD) can be applied by adding an extra term to the loss function which penalizes the l^2 -norm of the parameters:

$$\mathcal{L}_{\text{WD}}(\mathbf{x}, t; \boldsymbol{\theta}) = \mathcal{L}(\mathbf{x}, t; \boldsymbol{\theta}) + \frac{\lambda}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}. \quad (10)$$

Here, $\lambda > 0$ is a hyperparameter that determines the regularization strength. The intuition behind weight decay is that it encourages the optimizer to find a “simple” solution where the weights are close to 0. Furthermore, it can be shown that adding weight decay in the loss function is equivalent to setting a Gaussian prior on $\boldsymbol{\theta}$ [46]. Regularization can also be achieved by randomly deleting activations in the network during training. This technique is known as Dropout [120] and applying it results in a regularization penalty similar to that of weight decay [11].

Another common regularization technique is to use data augmentation in order to artificially expand the training set by feeding the network multiple augmented copies of the same data samples. For example, when learning to classify images of dogs and cats, we can use random scaling, rotations and cropping of the images. This also forces the network to be approximately invariant to those transformations, since the labels are not being changed (a rotated cat is still a cat). In some cases, it may also be desirable to make the network *exactly* invariant by restricting the architecture of the network itself. This topic is further explored in Section 4.

In practice, neural networks are optimized using various forms of gradient descent. The optimization procedure consists of 1) estimating the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ of the loss function with respect to the network parameters and 2) updating the parameters in the negative

direction of the gradient. In its simplest form, the learning rule is given by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}, \quad (11)$$

where η is the learning rate. The update is applied repeatedly until convergence is reached, which usually requires iterating over the training data multiple times. Since the neural network is essentially a computational graph, the gradient with respect to individual scalar parameter of the network can be found using backpropagation, which involves applying the chain rule recursively.

In practice, it is not computationally efficient (or even feasible) to compute the expected gradient using all training samples, since only a subset is needed to get an estimate of the gradient. The most common learning rule is batched stochastic gradient descent (SGD), where for each update the gradient is estimated using a randomly sampled subset of the training data. More specifically, each update uses a minibatch $\mathcal{B} \subset \mathcal{S}$ of training data and then estimates the gradient as

$$\nabla_{\boldsymbol{\theta}} \mathcal{L} \approx \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x}_n, \ell_n \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}_n, y_n; \boldsymbol{\theta}). \quad (12)$$

In recent years, more advanced learning rules that use gradient momentum in order to adaptively adjust the learning rate, such as Adam [69], have been proposed in order to reach faster convergence. The success of SGD-based optimization, together with the fast evolution of modern computer hardware that allow for efficient acceleration of both training and inference on computational graphs, has made deep neural networks a ubiquitous tool for machine learning research, and also feasible to deploy in many commercial applications. In this section we have only made a brief introduction to this topic and the reader is referred to [53] for a more rigorous treatment of basic deep learning concepts.

2.3 Federated Learning

In some scenarios, it can be difficult to access a centralized dataset to train on. For example in medical applications, it is not uncommon for hospitals to store large amounts of medical records for their patients, but due to the sensitive nature of the data, it cannot always be shared with researchers outside the hospital without complicated legal procedures. This makes it difficult to combine data from different hospitals and use it for model training.

Suppose that instead of centralizing data from different hospitals and training a single model, we could train one model for each local dataset and then combine the local models into a centralized model. This is the main idea behind federated learning. At the start of

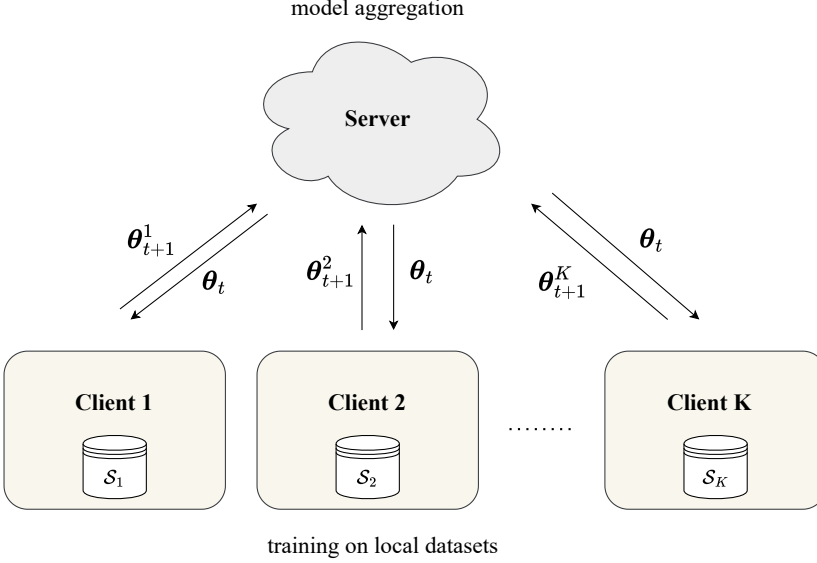


Figure 4: Conceptual overview of federated learning. For each training round t , the server distributes its model parameters θ_t to the clients, and each client then performs training on its local dataset \mathcal{S}_k . At the end of the training round, the updated parameters θ_{t+1}^k are sent back to the server and aggregated to a central model, before being distributed again in the next round. In the general setting, not all clients are required to participate in every training round.

each training round t , a subset of clients $\mathcal{P}_t \subset \{1, \dots, K\}$ receive a copy of the same model weights θ_t and then perform local training using a certain number of local gradient updates computed from their own local disjoint datasets $\{\mathcal{S}_k\}_{k \in \mathcal{P}_t}$, which results in a set of different local models $\{\theta_{t+1}^k\}_{k \in \mathcal{P}_t}$. At the end of the training round, the weights are sent to the server and aggregated to a centralized model. The new centralized model is then transmitted back to the clients at the start of the next round $t + 1$, that perform another round of training, and so on. Figure 4 shows an illustration of the learning process.

The simplest form of server aggregation is to use a weighted average [86] of the client model weights

$$\theta_{t+1} \leftarrow \frac{|\mathcal{S}_k|}{|\mathcal{S}|} \sum_{k \in \mathcal{P}_t} \theta_{t+1}^k, \quad (13)$$

where \mathcal{S}_k is the local dataset of the k :th client. In a scenario where all the local datasets are independent and identically distributed, and each client only takes a single gradient step, the federated average reduces to a form of distributed SGD. However, when more local training steps are performed by each client, the local models will diverge from each other, which makes the training dynamics quite different from centralized training. In

order to alleviate this problem, more sophisticated training strategies have been developed [66, 78, 5] that prevent client drift by adding regularization to the local training objectives or by modifying the aggregation method.

There are many practical applications of federated learning, with perhaps the most well known example being Google’s mobile keyboard prediction model [58]. When users type on their smartphone, the keyboard provides next-word predictions from a model that has been trained on user data. Due to the ethical concerns of storing user inputs on a central server, each smartphone instead trains their own local model and these are then combined to form a central model with better predictive performance than the individual models. The same technique is also used to some extent for training speech models in voice assistants [2]. Furthermore, ensuring that the trained models do not leak information about the training data is essential, which has lead to the development of differentially private federated learning methods [87].

2.4 Number Formats for Neural Network Training and Inference

Training large neural networks requires significant resources and hardware capabilities, which has lead to the development of hardware accelerators for general matrix multiplications (GEMM). Specialized hardware, such as graphics processing units (GPUs) and neural processing units (NPU), rely on parallelism across multiple cores in order to speed up training. This is done by utilizing single-instruction, multiple data (SIMD) execution which allows for increased total throughput compared to processing data sequentially. For example, consider the MLP in Figure 3. Here, each value in the hidden layer can be computed independently using dot-products between the input vector and the corresponding weight vector, followed by an element-wise activation function. These operations can thus be executed in parallel by exploiting multiple computational cores.

The increasing computational resources available to researchers has been a driving factor towards designing larger and larger neural networks. However, an increasing demand for machine learning applications everywhere has also lead to an opposite research trend, with the goal of making models available on consumer devices, such as laptops, smartphones or even microcontrollers [146]. Part of this research involves designing neural networks under hardware constraints that makes it feasible for them to run on devices with limited hardware capabilities. Another aspect is how to optimize the designs of widely available models. This includes system optimizations, where the computations are made efficiently implemented for specific hardware capabilities, as well as model compression, which involves re-designing parts of the model. Whereas system optimizations preserve the computational structure of the model, model compression can lead to a degradation of the predictive performance. Methods for model compression include quantization, pruning and knowledge distillation [96]. Here we give a brief overview of neural network quantization using different number

formats that are popular for deep learning applications.

The computations inside a neural network are typically done using one of two number formats: floating point (FP), typically using 32, 16 [21], or 8 [135] bits, or fixed point integer using 8 bits (INT8) [64] or fewer [138]. Whereas the former can be used for both training and inference, the latter is typically only used for inference, since training requires a higher dynamic range in order to capture variability in the gradients during backpropagation. A set of signed floating point numbers $\mathbb{F} \subset \mathbb{R}$ consists of numbers of the form

$$x = (-1)^s 2^{p-b} \sum_{i=0}^m d_i 2^{-i}, \quad s, d_i \in \{0, 1\}, \quad p \in \{0, 1, \dots, 2^e - 1\}. \quad (14)$$

Here, s is the sign bit, m is the number of mantissa bits, e is the number of exponent bits and b is the exponent bias. Given a fixed number of bits available, they can be assigned to either the exponent part, which yields a higher dynamic range, or the mantissa, which gives higher precision. The exponent bias can be used to scale all the numbers into the desired range. A common floating point format is IEEE single precision, which uses 1 sign bit, 8 bits for the exponent and 23 for the mantissa [3].

For “normal” numbers, d_0 is always set to 1. The special case where $p = 0$ is reserved for “subnormal” numbers by setting $d_0 = 0$, which allows for an exact representation of 0. Also note that we can use the same framework to represent the integer format by using a single exponent bit.

In practice, the same number format does not need to be used in all operations. In a neural network accelerator, different number formats can be used for storing weights and activations in memory, computing the arithmetic operations and accumulating the results. This technique is also known as mixed precision training [88]. For example, it is common to use FP16 arithmetic combined with FP32 accumulators, which allows for improved training speed without sacrificing performance. Furthermore, different network layers might require different levels of numerical precision, which implies that finding the best trade-off between efficiency and performance might require using different number formats for different layers.

Quantization of neural networks typically follows one of two paradigms: post-training quantization (PTQ) and quantization-aware training (QAT). PTQ can be applied to any pre-trained model by quantizing the weights and activations of the network into a low-precision format. For example, when using INT quantization, each scalar value needs to be scaled using a linear transformation and then rounded to the nearest integer. Inference can then be run using dynamic or static quantization. For dynamic quantization, the transformations are calculated on the fly for each example using the dynamic range of the inputs to the network and all intermediate activations. For static quantization, this trans-

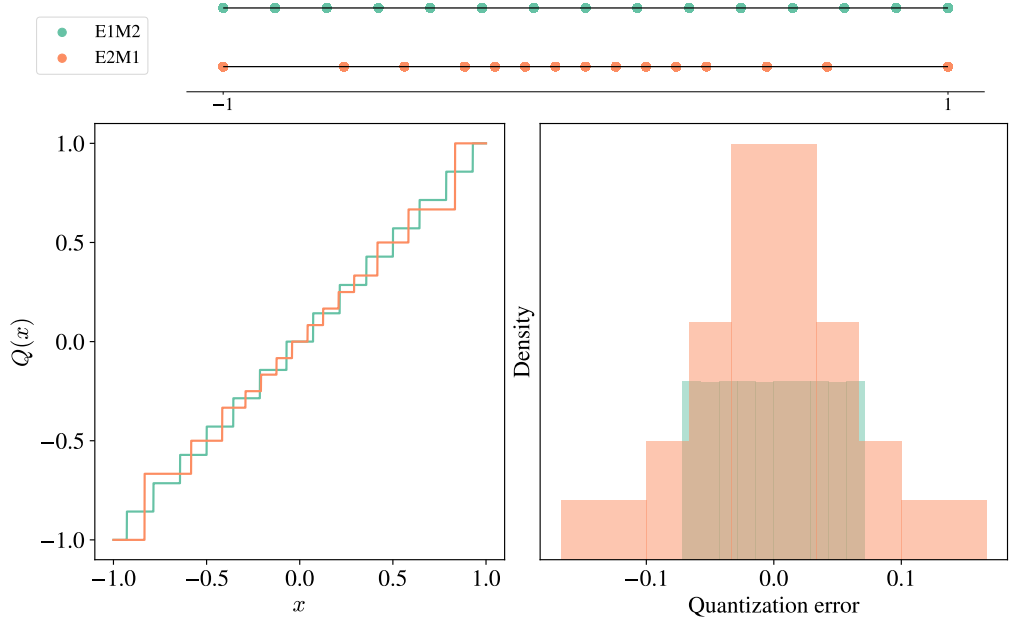


Figure 5: Visualization of the quantization gridpoints (top), the corresponding quantization functions (left) and the error distribution for two different 4-bit floating point number formats using E exponent bits and M mantissa bits and 1 sign bit. Since the grids are symmetric around 0, the total number of gridpoints is $2^4 - 1 = 15$. Here, the exponent bias has been adapted to achieve a dynamic range between -1 and 1, and the error distribution is based on inputs uniformly distributed in this interval. Note that the E1M2 format corresponds to uniform quantization, i.e. INT4, whereas the E2M1 yields lower quantization error for numbers close to 0 and larger quantization error for numbers close to ± 1 .

formation needs to be computed beforehand using a calibration dataset. In either case, due to the loss of precision caused by the weight and activation quantization, PTQ typically results in a loss of the models predictive performance. To what extent the performance of a neural network is affected by quantization is also architecture-dependent and different quantization techniques might be required for different network architectures [89].

QAT is a training technique that tries to mitigate the performance drop caused by PTQ by introducing quantization errors into the training process. This is done by applying a quantization operator before and after each layer in the network. For each scalar value $x^{(i)}$ in the vector \mathbf{x} , applying the quantization operator can be written as

$$Q(x^{(i)}) = \begin{cases} x_{\min}, & x^{(i)} \leq x_{\min}, \\ s_i \left\lfloor \frac{x^{(i)}}{s_i} \right\rfloor, & x_{\min} < x^{(i)} < x_{\max}, \\ x_{\max}, & x^{(i)} \geq x_{\max}, \end{cases} \quad (15)$$

where s_i is a scale-factor and $\lfloor \cdot \rfloor$ denotes rounding to the nearest integer. Furthermore, values that fall outside the range $[x_{\min}, x_{\max}]$ of the smallest and largest representable num-

bers in the quantized format are clipped. For integer quantization, the quantization grid is uniform, which implies that the scale factor is the same for all elements in the vector. A natural choice that leaves all elements unclipped would be $s_i = \max_i |x^{(i)}| \forall i$. Floating point quantization, on the other hand, corresponds to uniform quantization with m bits in intervals of consecutive powers of two $[2^{p-b}, 2^{p-b+1})$. The scale factor that leads to uniform quantization in this interval is therefore given by $\log_2 s_i = \lfloor \log_2 |x^{(i)}| \rfloor - m$. Figure 5 shows examples of two different quantization functions.

When performing QAT, each layer in the network receives its own scale factors that are treated as extra learnable parameters. After training, the scale factors are frozen and can be used to map the weights and activations to the right dynamic ranges before converting the model into the quantized number format. For both integer and floating point formats, QAT typically results in lower performance drop compared to PTQ [130]. Furthermore, the quantization noise introduced during QAT can act as a regularizer similar to weight decay [9]. Another aspect of QAT is that gradients need to be backpropagated through the quantization operator. Since this function has zero gradients at all points where it is differentiable, this is done by instead using the straight-through estimator (STE) [14], which is given by $\frac{\partial \lfloor x \rfloor}{\partial x} = 1$.

Finally, it is worth noting that quantizing neural networks can bring other benefits that are not directly related to running the actual computations. In the previous section, we briefly introduced the federated learning concept, in which model parameters need to be sent back and forth between a server and multiple clients during the training phase. This can potentially result in large communication costs and reducing the amount of communication is therefore an important aspect of training efficiency. Quantizing model parameters before the communication step can thus lead to large savings in the amount of transmitted data. In Paper IV we further explore the synergy effects that can be leveraged when using low-precision number formats for training and communication in a federated learning context.

3 Diversity and Ordinal Regression

In this section, we will cover the notion of diversity in the context of statistical learning theory, which serves as background material for Paper I. Here, diversity refers to differences of properties within a collection of models, data sets or predictions. Hence, a diverse set of models can mean either models with different architectures or different values of their parameters. Likewise, a diverse data set contains a wide variety of samples that are not biased towards a particular setting. For example, a non-diverse data set with images of cats and dogs might only contain instances of particular breeds, or only images captured indoors, whereas a diverse data set should capture enough variety in both of these aspects. Obviously, it is not trivial to quantify diversity in this aspect, since the notion is inherently subjective and depends on the context. However, a diversity in predictions is easier to quantify, since we can measure the variance of the predictions. Therefore a set of predictions can be considered diverse if they have a variance larger than zero.

An analogy for diverse predictions can be made with juries used for grading in sport competitions, for example gymnastics and figure skating, where the final score is assigned to each competitor by averaging scores assigned by individual jurors. This mitigates the influence of biased jurors and is therefore considered more fair than a single juror, a phenomenon sometimes referred to as the wisdom of the crowd. In order to make the final score robust to outlier judges, it is common to use a trimmed average, where e.g. the lowest and highest scores are not included.

Diversity is also used to improve fairness in legal systems. In countries where juries are used in judicial courts, a convicting verdict can only be reached if a majority of jurors are in agreement (often, a supermajority or even unanimity is required). Furthermore, during the jury selection process, it is often considered desirable to have diversity within the jury with respect to e.g. income, gender and occupation, which is similar to the notion of model diversity.

In wireless communication, diversity can be exploited in order to mitigate the influence of random processes. For example, when a mobile phone has poor signal reception, the cell tower may transmit a repeated version of the signal, which can then be averaged in the receiver. The white noise generated in the receiver antenna will then tend to zero as the number of repetitions increase. In some applications, multiple receiver or (and) transmitter antennas are used as well, which increases the probability of at least one antenna pair having good reception [137].

Diversity in machine learning is closely related to ensemble learning, in which a set of model predictions are combined in order to improve predictive performance compared to a single model on a given task. An old saying goes “two heads are always better than one”, and as we shall prove in this section, the error of an ensemble average is always smaller than the average

error of the individual ensemble members, as long as there is enough diversity among the ensemble members. This makes it possible to combine a set of weak models, with predictive performance only slightly better than random guessing, into a strong ensemble model with good predictive performance, a method known as boosting [48].

3.1 Regression Ensembles

For a given problem, we might have several models f_{θ}^m , $m = 1, \dots, M$, that we wish to combine in order to create an ensemble model. For a regression problem with predictions $y_m = f_{\theta}^m(x)$, the simplest form of combination is to use a linear combination of the individual predictions

$$y_{ens} = \sum_{m=1}^M w_m y_m, \quad (16)$$

where the set of weights $\{w_m\}_{m=1}^M$ are typically restricted to be non-negative and sum to 1, which is also referred to as a convex linear combination. The simplest form of linear combination is to assign equal weight to each individual predictor, i.e. $w_m = 1/M, \forall m$, but it is also possible to optimize them to minimize the prediction error on a validation data set [97].

Given that we have calculated an ensemble average of our predictions, we would like to understand the benefit of this operation. Fortunately, there exists an easy way to decompose the ensemble error, which allows us to quantify the error directly in terms of the errors of the individual predictors.

Theorem 3.1. The Ambiguity Decomposition (Krogh and Vedelsby [72]) Consider an ensemble of $m = 1, \dots, M$ models and the convex linear ensemble average computed as in (16). The quadratic error between the ensemble average y_{ens} and the target t then satisfies

$$(y_{ens} - t)^2 = \sum_{m=1}^M w_m (y_m - t)^2 - \sum_{m=1}^M w_m (y_m - y_{ens})^2. \quad (17)$$

Proof: The result can easily be shown by manipulation of terms:

$$\begin{aligned}
\sum_{m=1}^M w_m (y_m - t)^2 &= \sum_{m=1}^M w_m (y_m - y_{ens} + y_{ens} - t)^2 \\
&= \sum_{m=1}^M w_m [(y_m - y_{ens})^2 + (y_{ens} - t)^2 + 2(y_m - y_{ens})(y_{ens} - t)] \\
&= (y_{ens} - t)^2 + \sum_{m=1}^M w_m (y_m - y_{ens})^2,
\end{aligned} \tag{18}$$

where we have used the fact that $y_{ens} = \sum_m w_m y_m$ and $\sum_m w_m = 1$. The final result is obtained by rearranging the terms. \square

The ambiguity decomposition states that the error of the ensemble average is less than or equal to the average error of the individual predictions of the ensemble members. In order for the ensemble error to be small, the first term in (17) should be minimized, which requires that the individual predictions are accurate. However, the second term, which grows with increased variance within the ensemble predictions, should be large. Therefore there is a trade-off to be made between accurate and diverse ensemble members.

3.2 The Bias-Variance Decomposition

In Section 2, we briefly mentioned the concept of overfitting, which occurs when a model explains the training data well, but fails to generalize to previously unseen examples in a held-out test set. Here we shall define this in more precise terms.

Assume that we are given a dataset of input-output pairs $\mathcal{S} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^d$ and $t_n \in \mathbb{R}$, which can be modeled as $t_n = f(\mathbf{x}_n) + \epsilon_n$, where ϵ_n are i.i.d. random variables with mean $\mathbb{E}[\epsilon_n] = 0$ and variance $\text{Var}[\epsilon_n] = \mathbb{E}[\epsilon_n^2] - \mathbb{E}[\epsilon_n]^2 = \sigma^2$. Furthermore, assume that we have a trained model f_{θ} , and we are interested in decomposing the expected prediction errors (also known as the prediction risk), which we define as

$$R(f_{\theta}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{S}} \left[(f(\mathbf{x}) - f_{\theta}(\mathbf{x}))^2 \right]. \tag{19}$$

The bias-variance decomposition then tells us that the expected error can be decomposed as [49]

$$R(f_{\theta}) = (\text{Bias}_{\mathcal{S}}[f_{\theta}])^2 + \text{Var}_{\mathcal{S}}[f_{\theta}] + \sigma^2, \tag{20}$$

where

$$\text{Bias}_S[f_\theta] = \mathbb{E}_{\mathbf{x} \sim \mathcal{S}} [f_\theta(\mathbf{x}) - f(\mathbf{x})], \quad (21)$$

$$\text{Var}_S[f_\theta] = \mathbb{E}_{\mathbf{x} \sim \mathcal{S}} \left[(\mathbb{E}_{\mathbf{x} \sim \mathcal{S}} [f_\theta(\mathbf{x})] - f_\theta(\mathbf{x}))^2 \right]. \quad (22)$$

The bias-variance decomposition tells us that the expected prediction errors can be decomposed into three terms. The first term is the bias of the model, which tells us how well the model fits the training data. The second term is the variance, which tells us how sensitive predictions are to the noise in the training data, and the third term is the irreducible error. According to the decomposition, we can in theory trade-off bias and variance by making design choices in our model, which might be beneficial in some scenarios. For example, the weight decay introduced in equation (10) acts as a regularization term that reduces overfitting. In terms of the bias-variance decomposition, weight decay reduces the variance of the model at the cost of an increased bias. On the other hand, too much regularization can result in underfitting. According to this interpretation, a task for the machine learning practitioner is then to find the sweet-spot, i.e. the amount of regularization that makes the best trade-off.

The irreducible error σ^2 does not depend on how the predictive model is constructed. For example, consider the problem of age estimation from a single image of a person's face. Is it possible to construct a model with zero prediction error? The answer is clearly no, since the age of a person cannot be inferred solely on the basis of a photograph. For this problem, it therefore makes sense to model the error which cannot be explained by the content of the image as an additive noise term in the underlying model.

We also note that the bias-variance decomposition looks similar to the ambiguity decomposition in equation (17). In fact, one can show that for an ensemble of M models with zero covariance, the ensemble average has the same bias as the average individual ensemble member, but achieves a variance reduction by a factor of $1/M$ [19]. However, if ensemble predictions have positive covariance, this will increase the variance of the ensemble. This has led to the development of techniques for negative correlation learning, where ensemble members are explicitly trained to have negative covariance [80].

The conventional interpretation of the bias-variance trade-off tells us that our models should not have too many parameters, since this will lead to overfitting the training data. However, the success of deep learning seems to contradict this, since many deep neural networks with millions of parameters generalize better than shallower networks with fewer parameters, even on small datasets. This has led researchers to question the validity of the supposed trade-off in the high-parameter regime of deep networks and proposed a different interpretation, where variance decreases as the number of parameters increase beyond a certain

threshold [13, 91, 140]. Note that the decomposition of the expected prediction error itself is still valid, but how the variance and bias are affected by the model size and architecture is non-trivial.

3.3 Label Binning

For some problems, the labels exhibit a certain pattern where they can be grouped into clusters. Again, consider the problem of age estimation from a single image, where the input \mathbf{x} is an image of a human face and the label t is the age of the person in the image. Firstly, the choice of objective function will depend on how the age is represented. Perhaps the most natural choice is to model the domain of the labels as $t \in \mathbb{R}^+$, i.e. the age is a positive real number. However, in some scenarios the labeled examples in the training data only contains the year of birth of each person. Therefore, another natural choice would be to use a set of non-negative integers, e.g. $\tilde{t} \in \{0, 1, \dots, 100\}$. The rounding error can then be modeled as uniformly distributed noise.

In general, it is hard to say which label representation is better, as the effect of grouping labels depends on the properties of both the model and the dataset. However, in order to highlight possible benefits of label binning, we hereby construct a toy example where it can be shown to yield variance reduction compared to using ordinary least squares minimization.

Assume a linear regression model $\mathbf{T} = \mathbf{X}\boldsymbol{\theta}$, where $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times d}$ is a rank- d feature matrix with $N \geq d$, $\mathbf{T} = [t_1, \dots, t_N]^T \in \mathbb{R}^N$ are the labels and $\boldsymbol{\theta} \in \mathbb{R}^d$ are the linear model parameters. Suppose that we are given the feature matrix and a set of corrupted labels $\tilde{\mathbf{T}} = \mathbf{T} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is i.i.d. zero-mean noise with covariance $\sigma^2 \mathbf{I}$. Furthermore, assume that the noise is bounded, such that the addition of the noise does not change the order of the labels, i.e. if $t_1 < t_2 < \dots < t_N$, then it also holds that $t_1 + \epsilon_1 < t_2 + \epsilon_2 < \dots < t_N + \epsilon_N$. In practice this means that we can sort the labels and the label noise does not change the outcome of the sorting.

The ordinary least squares estimate of $\boldsymbol{\theta}$ is given by $\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{T}$ and the predictions (on the training data) are given by $\mathbf{Y} = \mathbf{L} \mathbf{T}$, where $\mathbf{L} = \mathbf{X} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is a projection matrix. Assume now that we divide the training data into K separate clusters $\{\mathcal{D}_k\}_{k=1}^K$, or “bins”, where each bin contains N' samples. We can then assign new labels \tilde{t}_n to each training example by averaging the labels in each cluster, i.e. $\tilde{t}_n = \frac{1}{N'} \sum_{n' \in \mathcal{D}_k} t_{n'}$ for all $n \in \mathcal{D}_k$. In this hypothetical example, we can show that there is a benefit to performing the label binning before fitting our model, because it leads to a reduction in variance.

Given some predictions \mathbf{Y} , we define the variance as

$$\text{Var}(\mathbf{Y}) = \text{tr}(\text{cov}(\mathbf{Y}, \mathbf{Y})) = \sum_{n=1}^N \text{Var}(y_n), \quad (23)$$

where tr and cov denote the matrix trace and covariance respectively. With this definition, we can show the following

Lemma 3.2 (Variance of a linear model with label binning). Given the binned labels \mathbf{T}_{bin} , the variance of the re-projections \mathbf{Y}_{bin} are bounded as $\text{Var}(\mathbf{Y}_{\text{bin}}) \leq \sigma^2 \min(d, K)$.

Proof: Assuming that the labels are sorted, we can write the re-projections of the binned regression model as $\mathbf{Y}_{\text{bin}} = \mathbf{L}_{\text{bin}} \tilde{\mathbf{T}}$ where $\mathbf{L}_{\text{bin}} = \mathbf{L}\mathbf{B}$ and \mathbf{B} is a matrix that performs the binning operation on the labels. Since this consists of averaging N' labels in each cluster, we can write \mathbf{B} as a block-diagonal matrix with K blocks on the diagonal and zeros in all other entries, i.e.

$$\mathbf{B} = \frac{1}{N'} \begin{pmatrix} \mathbf{J}_{N'} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{J}_{N'} \end{pmatrix}, \quad (24)$$

where $\mathbf{J}_{N'}$ is an $N' \times N'$ matrix containing all ones. Then we have

$$\text{Var}(\mathbf{Y}_{\text{bin}}) = \text{tr}(\text{cov}(\mathbf{L}_{\text{bin}} \tilde{\mathbf{T}}, \mathbf{L}_{\text{bin}} \tilde{\mathbf{T}})) = \sigma^2 \text{tr}(\text{cov}(\mathbf{L}_{\text{bin}}, \mathbf{L}_{\text{bin}})) = \sigma^2 \|\mathbf{L}_{\text{bin}}\|_F^2, \quad (25)$$

where $\|\cdot\|_F$ is the Frobenius norm and we have used the fact that $\text{cov}(\tilde{\mathbf{T}}, \tilde{\mathbf{T}}) = \sigma^2 \mathbf{I}$. The norm can be bounded as

$$\|\mathbf{L}_{\text{bin}}\|_F^2 \leq \text{rank}(\mathbf{L}_{\text{bin}}) \|\mathbf{L}_{\text{bin}}\|_2^2 \leq \text{rank}(\mathbf{L}\mathbf{B}) \|\mathbf{L}\|_2^2 \|\mathbf{B}\|_2^2, \quad (26)$$

where $\|\cdot\|_2$ is the induced 2-norm. Since \mathbf{L} is a projection matrix, it satisfies $\text{rank}(\mathbf{L}) = \text{rank}(\mathbf{X}) = d$ and $\|\mathbf{L}\|_2 = 1$. It is easy to verify that $\text{rank}(\mathbf{B}) = K$ and $\|\mathbf{B}\|_2 = 1$. Furthermore, $\text{rank}(\mathbf{L}\mathbf{B}) \leq \min(\text{rank}(\mathbf{L}), \text{rank}(\mathbf{B}))$, which directly gives the desired result. \square

Note that the variance of ordinary least squares regression $\text{Var}(\mathbf{Y}) = \sigma^2 d$ is obtained for $K = N$, which implies a bin size of $N' = 1$ samples. The insight from this result is that if the labels appear to be clustered, we can achieve a model that is more robust to the label noise by binning the labels. However, in contrast to ordinary least squares regression, the model will not be unbiased, since according to the Gauss-Markov theorem ordinary least squares has lowest variance of all unbiased estimators [126].

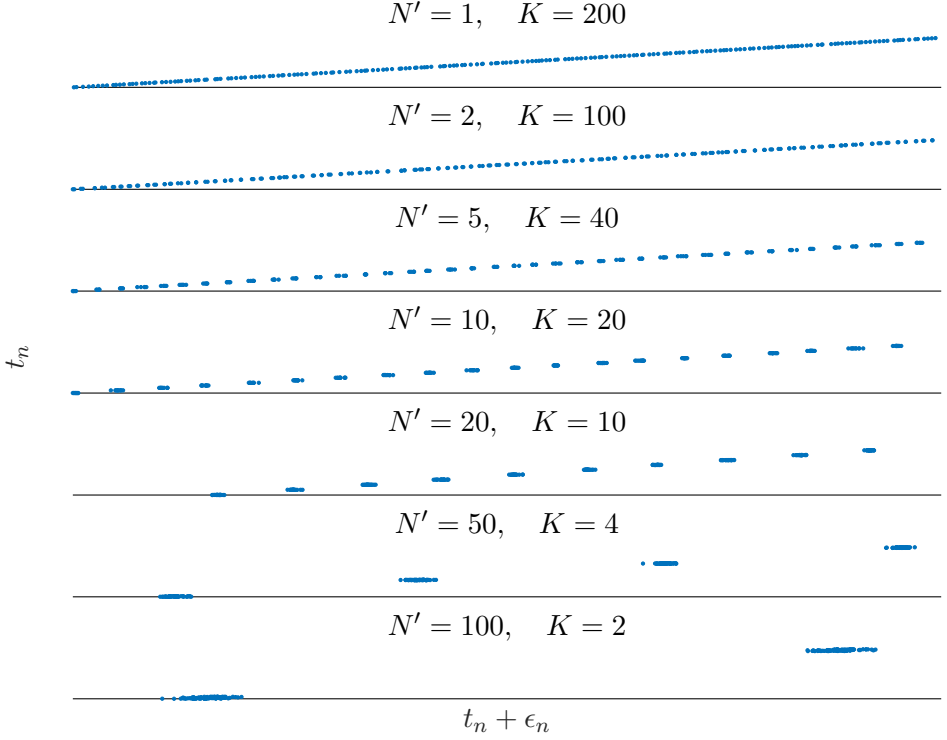


Figure 6: Examples of several datasets with clustered labels, highlighting the transition from continuous targets to something that looks more like categorical targets. Here, N' indicates the number of samples in each cluster and K is the total number of clusters. When performing label binning, each label is replaced by the mean value of all labels in the same cluster.

In order to visualize the bias-variance trade-off for our example, we can estimate the two terms using Monte-Carlo simulation of our example. Figure 6 shows sets of label pairs with and without noise generated from a rank $d = 100$ linear model with $N = 200$ samples. In this toy example, each dataset has been generated in a way such that the labels can naturally be divided into K bins. For each dataset, we fit two linear models to the noisy labels: ordinary least squares and least squares with label binning. In order to estimate the variance and bias of the two models, we then repeat the process 500 times and calculate the sample variance and bias. In each iteration, the features \mathbf{X} and labels \mathbf{Y} are constant, but the noise term ϵ is re-sampled from the same Gaussian distribution.

Figure 7 shows how the MSE of the binned regression model, decomposed into the variance and bias parts, compared to the ordinary least squares MSE. As the number of samples per bin N' increases, the re-projection variance decreases proportional to $1/N'$. However, as N' grows larger, the bias also increases and eventually the error reduction due to binning starts decreasing.

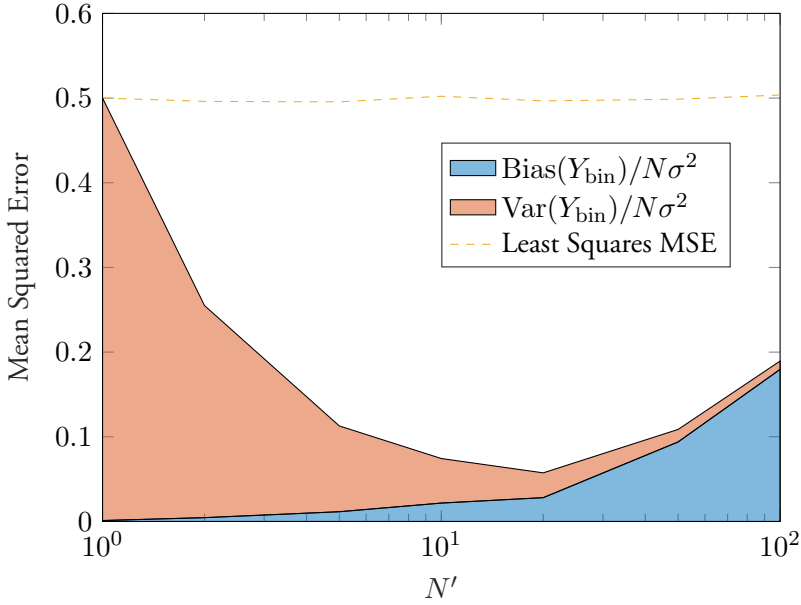


Figure 7: Bias and variance of least squares regression with label binning compared to the ordinary least squares without binning.

3.4 Regression via Classification

When using label binning, MSE minimization might not be the best option, since the labels now appear to belong to a set of discrete classes. Figure 6 shows that as the number of clusters decreases, the labels transition into something that looks more like categorical targets. When designing a model, we therefore need to decide on whether to model the labels as continuous points on the real number line, or categorical targets belonging to different classes.

The difference between the choice of representation is subtle, but it has consequences for both the objective function and the model architecture when designing for example a neural network. In terms of the objective function, the first choice leads to a regression problem where the network has a single output that directly predicts target variable. The second choice leads to a classification problem, where the network outputs a probability score that models the likelihood of the input belonging to a certain class. Is it possible to say which of these two choices is better in terms of the final predictive performance of the network?

For certain problems, using direct regression (or regression with label binning) might be undesirable due to difficulties in finding a suitable loss function. Consider for example the problem of pose estimation, where the target variable $t \in [0, 2\pi)$ is an angle of rotation.

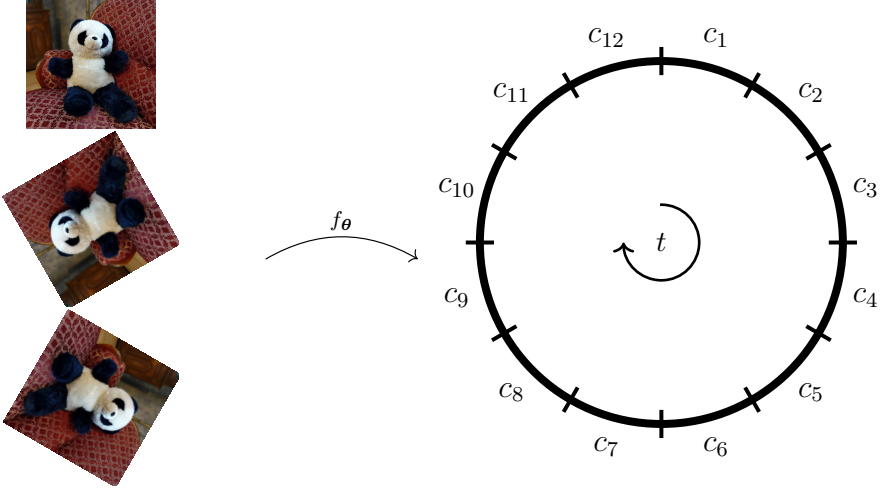


Figure 8: An example of how to perform RvC for image rotation prediction. Here, each continuous angle $t \in [0, 2\pi)$ is mapped to one of $K = 12$ categorical labels using intervals of equal width on the unit circle. The model then tries to classify each image in the correct bin, instead of predicting the angle directly.

If we use the standard MSE loss function as in (4), it will suffer from discontinuity at $t = k2\pi$, $k \in \mathbb{Z}$. For example, if the target is $t = 2\pi - \epsilon$ for some small value of ϵ , and the model predicts $y = \epsilon$, the error modulo 2π is 2ϵ . However, the squared error is $(2\pi - 2\epsilon)^2$, so using the MSE loss is not suitable for this problem.

We could try to use a periodic loss function with several minima at $k2\pi$, e.g. $\sin^2((y - t)/2)$, but having several local minima makes optimization more difficult. This could be solved for example by letting the model predict a pair of coordinates on the unit circle, instead of directly predicting the pose angle, but this would require a normalization of the outputs that makes optimization difficult. Here we will instead focus another method known as regression via classification (RvC) [127].

The main idea of RvC is to consider the target as a categorical variable, which makes it possible to use classification methods for predictions. However, it is not always obvious how the categorical classes ought to be defined for a given problem. The simplest solution is to create bins of equal width that span the entire domain of the target. In the case of pose estimation where the target variable is an angle, we can for instance divide it into K equally wide bins, forming a set of categorical variables $\{c_k\}_{k=1}^K$, where each class corresponds to an interval, such that $c_k = [\frac{k-1}{K}2\pi, \frac{k}{K}2\pi)$. Now, we can define a mapping from the continuous targets to the K classes by simply assigning class k to all targets that belong to the interval c_k , as shown in Figure 8.

When selecting the number of intervals K that spans the target domain, there is a trade-off to be made between the discretization error and the number of training examples in each

class. For example, if we assume that our training data consists of N data points we can let $K = N$ and have one training example per class. This will make it difficult for the model to generalize to unseen data points, since it will not learn to group nearby angles in the same category. On the other extreme end, we could let $K = 2$, such that $c_1 = [0, \pi)$ and $c_2 = [\pi, 2\pi)$, which corresponds to a binary classification problem where the pose is classified into two segments of the unit circle. Obviously, this is not a good choice, because even if the classifier can achieve high accuracy, we want to be able to infer poses with higher precision than this. Therefore we can conclude that the optimal number of bins lies somewhere in between these extremes, although there is no general rule that can be inferred on what the optimal value for K ought to be.

When training the model, we assign each training data point to one of the classes and create labels using e.g. a one-hot distribution

$$q(c_k) = \begin{cases} 1, & t \in c_k, \\ 0, & \text{otherwise.} \end{cases} \quad (27)$$

The model is trained to match the labels by predicting a probability distribution $p(t_n \in c_k | \mathbf{x}_n)$ that matches the labels. For brevity, we will from here forth denote the predicted probability simply as $p(c_k | \mathbf{x})$.

Model optimization can be done using the same procedure as for an ordinary classification problem, using e.g. the cross-entropy loss function

$$\mathcal{L} = - \sum_{k=1}^K q(c_k) \log p(c_k | \mathbf{x}). \quad (28)$$

The RvC approach might seem counter intuitive at first, but it effectively solves the problem of discontinuous loss functions, since predictions on either side of the cutoff angle $t = 2\pi$ will be penalized equally. However, the price that we pay for using a categorical label representation is that the loss function is now agnostic to the magnitude of the error, because it does not take into account which classes are nearby. For example, using the categorical labels in Figure 8, if the true angle lies in c_1 , the loss will be the same if the model predicts c_2 or c_7 , although c_2 is clearly a better prediction.

From this it becomes obvious that the RvC method has thrown away useful information about the labels, namely 1) the ordinal relationship between the classes and 2) the distance metric used to compute the prediction errors, which in this case is the shortest path along the unit circle. However, there are ways to exploit this information even in the context of RvC. One solution, proposed by Diaz et al. [36], is to use a soft label representation

$$p(c_k|\mathbf{x}) = \frac{e^{-\phi(\gamma(c_k),t)}}{\sum_{i=1}^K e^{-\phi(\gamma(c_k),t)}}, \quad (29)$$

where ϕ is a distance metric and $\gamma(c_k)$ is the midpoint of the interval. In other words, the probability assigned to each class is inversely proportional to the distance between the class and the true class. In practice, this will penalize nearby class predictions less than far away predictions according to the chosen distance metric.

3.5 Ordinal Regression

Now let us consider a problem domain where a natural distance metric does not exist. Consider for example the problem of ranking, where the objective is to classify objects into ordinal categories $t \in \{1, \dots, K\}$ and the classes can be ordered, but there doesn't exist a well-defined distance between classes. An example of such a problem would be to predict survey responses on a Likert scale, i.e. from "strongly disagree" to "strongly agree", based on some features of the respondent. This setting is referred to as ordinal regression and several methods have been proposed for solving it.

Frank et al. [47] proposed to treat K -class ranking problem as $K - 1$ separate binary classification problems by noting that instead of predicting "what is the rank of x ?", asking "is the rank of x greater than k ", for $k = 1, \dots, K - 1$. In a machine learning setting, a model can then be trained to predict $p(t > k|x)$ for all relevant values of k , and the ranking probabilities can then be recovered by noting that

$$\begin{aligned} p(t = 1|\mathbf{x}) &= 1 - p(t > 1|\mathbf{x}), \\ p(t = k|\mathbf{x}) &= p(t > k|\mathbf{x}) - p(t > k - 1|\mathbf{x}), \quad k = 2, \dots, K - 1, \\ p(t = K|\mathbf{x}) &= p(t > K - 1|\mathbf{x}). \end{aligned} \quad (30)$$

Although this method is simple, it does not automatically guarantee that the predictions are rank-consistent, i.e. that $p(t > 1|\mathbf{x}) \geq p(t > 2|\mathbf{x}) \geq \dots \geq p(t > K - 1|\mathbf{x})$, but methods have later been developed to enforce rank-consistency [77]. Nevertheless, training a model by summing the individual losses of each binary classifier will result in an overall loss that penalizes errors based on their magnitude on the ordinal scale.

3.6 Label Diversity

In Paper I, we consider an alternative approach to RvC and ordinal regression that exploits diversity, but in the labels rather than in the data. This approach is inspired by the fact that

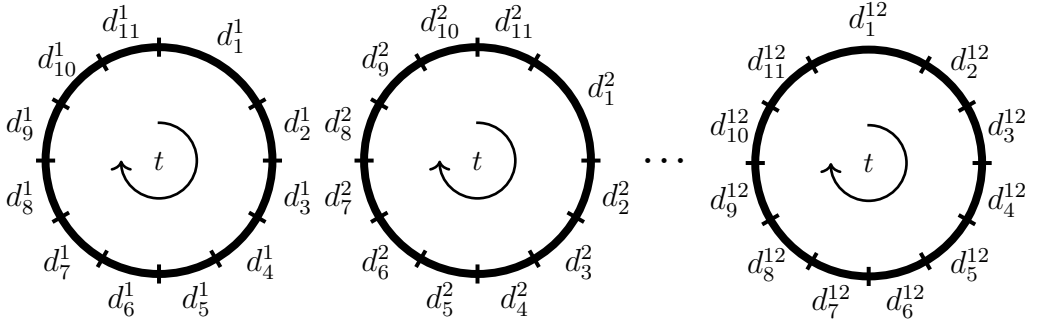


Figure 9: An example of how to create diverse sets of intervals by combining nearby classes. In this example, there are $M = 12$ different sets of labels, each containing $L = 11$ intervals that span the unit circle.

there are many possible ways to create categorical labels from continuous targets and that there is no general method for knowing a priori which categorical representation will be most suitable for a particular task. Going back to the example of pose estimation in Figure 8, we note that there are several arbitrary design choices that have to be made dividing the unit circle into intervals. For instance, how do we choose the number of intervals K ? Should the intervals have equal length and should there be overlap between them?

From ensemble theory we know that combining different model predictions, where each model has been trained on a different label representation, will yield a smaller prediction error than the average error from the individual models. Therefore, without knowing which representation is most suitable, we can create a diverse set of representations and use an ensemble of models to combine the predictions on the different representations into a final prediction. For example, we could train M different models using different values of K , and combine the the model predictions using an ensemble average.

The method of label diversity extends ordinary multi-class classification and ordinal regression problems, since different labels can be grouped in arbitrary ways. For example, a K -way classification problem can be expanded to $M = K$ different L -way classification problems, where $L = K - R + 1$ by grouping R consecutive classes

$$\begin{aligned}
 d_1^1 &= c_1 \cup c_2 \cup \dots \cup c_R, & d_2^1 &= c_{R+1}, & \dots, & d_L^1 &= c_K, \\
 d_1^2 &= c_2 \cup c_3 \cup \dots \cup c_{R+1}, & d_2^2 &= c_{R+2}, & \dots, & d_L^2 &= c_1, \\
 & & & & \vdots & & \\
 d_1^K &= c_K \cup c_1 \cup \dots \cup c_{R-1}, & d_2^K &= c_R, & \dots, & d_L^K &= c_{K-1}.
 \end{aligned} \tag{31}$$

For example, by combining $R = 2$ nearby classes in our discretization of the unit circle, we can create $M = 12$ different 11-way classification problems, as shown in Figure 9. Each

individual predictor can then be trained to output a probability $p_m(d_l^m|\mathbf{x})$, using labels $q(d_l^m|\mathbf{x})$ obtained as in (27). Note that the way labels are combined in (31) is simply an illustrative example, since in general we don't need to restrict L to be the same for each of the M classifiers. In other words, each circle in Figure 9 could be divided into a different number of intervals.

Before combining the individual predictions, we first need to marginalize over d_l^m to recover distributions $p_m(c_k|\mathbf{x})$ over the original class intervals c_k for each individual predictor m . We note that $p(c_k|d_l^m, \mathbf{x})$ only depends on the overlap between c_k and d_l^m . For example, using our sets of labels in Figure 8 and 9, we can infer that $p_1(c_1|\mathbf{x}) = \frac{1}{2} \cdot p_1(d_1^1|\mathbf{x})$. More generally we have that

$$p(c_k|d_l^m, \mathbf{x}) = \frac{||d_l^m \cap c_k||}{||d_l^m||} = \begin{cases} \frac{1}{R}, & d_l^m \cap c_k \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases} \quad (32)$$

This allows us to calculate the posterior likelihoods as

$$p_m(c_k|\mathbf{x}) = \sum_{l=1}^L p(c_k|d_l^m, \mathbf{x}) p_m(d_l^m|\mathbf{x}), \quad m = 1, \dots, M. \quad (33)$$

Now we combine the predicted likelihoods of the ensemble using a convex combination:

$$p(c_k|\mathbf{x}) = \sum_{m=1}^M w_m p_m(c_k|\mathbf{x}). \quad (34)$$

If each classifier is trained by minimizing the cross-entropy loss, then the sum of the individual losses will incorporate the ordinal ranking of the targets. This can be seen by noting that nearby angles will fall into the same intervals for some classifiers, and different intervals for others. For example, a pair of very small angles ϵ and $-\epsilon$ will be mapped to d_{11}^2 and d_{10}^2 respectively, but they will both be mapped to $d_1^{1,2}$. Hence the total loss will incorporate the fact that these angles are close, but not equal, and therefore the method of label diversity can be regarded as a form of ordinal regression.

For regression problems where the target is continuous, the method of label diversity can be generalized even more. For example, we can consider generating M sets of intervals $\{\mathcal{D}^m\}_{m=1}^M$ where $\mathcal{D}^m = \{d_l^m\}_{l=1}^{L_m}$ are arbitrary discretizations of the unit circle, with the restriction that the intervals in each set are non-overlapping. For any particular angle, we can then directly map it to an interval in each set \mathcal{D}^m . Since this mapping will obviously not be injective, we will still have to deal with any discretization error.

There are several ways that the predicted likelihoods can be used to form an estimate y_m of the angle. One possibility is to use the midpoint $\gamma(d_l^m)$ of the interval with the highest likelihood, i.e. $y_m = \gamma(d_{l^*}^m)$, where $l^* = \arg \max_l p(d_l^m | \mathbf{x})$. Another option is to use the expected value as $y_m = \sum_l \gamma(d_l^m) p(d_l^m | \mathbf{x})$. The predictions of the ensemble members can then be combined using a convex combination as in (16). From the ambiguity decomposition, we can then establish that the ensemble error will be smaller than the average error of the ensemble members, since a non-zero variance is induced by the different discretizations. If the discretization errors are small, but independent, then the ensemble average will mitigate the effects of discretization. Likewise, there will be a trade-off between the discretization error of each member and the number of members required in the ensemble for good accuracy.

There remain several questions on how to implement label diversity in practice, as well as investigating in which types of problems this is most useful. We refer to Paper I for examples of applying label diversity on real problems and performance comparisons with standard regression and RvC.

4 Neural Network Architectures and Input-Output Symmetries

An important aspect of neural network architecture design is to incorporate geometric priors by making the networks invariant (or equivariant) to certain transformations. As was briefly mentioned in Section 2.2, this can be regarded as a form of regularization, since it restricts the search space of optimization to be limited to a smaller set of functions that obey the imposed restrictions. In practice, this can be done by designing the computational graph of the network such that the invariance is satisfied. For example, for a particular problem, such as classifying cats and dogs, where we have two augmented versions of the same image \mathbf{x} and $\tilde{\mathbf{x}}$, we can strive to design our network such that $f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})$, regardless of the particular value of $\boldsymbol{\theta}$. In general, we say that a transformation that does not change the underlying property of the object is a symmetry of that object. For example, in the context of image classification, a rotated and translated cat, is still a cat.

In order to formalize the notion of invariant neural networks, it is useful to borrow some concepts from group theory. Here we will briefly state some useful definitions and we refer to [18] for a more in-depth treatment of the relationship between group theory and deep learning. Given a symmetry group \mathfrak{G} with group actions $\mathfrak{g} \in \mathfrak{G}$, we define the corresponding group representations as $\rho : \mathfrak{G} \rightarrow \text{GL}_n(\mathbb{R})$, where $\text{GL}_n(\mathbb{R})$ is the general linear group consisting of invertible $n \times n$ matrices with elements in \mathbb{R} . Here, the matrix dimension n depends on the feature space \mathcal{X} on which the representations are acting. We now have the following definition:

Definition 4.1. A function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is invariant with respect to \mathfrak{G} if $f(\rho(\mathfrak{g})x) = f(x)$ for all $x \in \mathcal{X}$ and $\mathfrak{g} \in \mathfrak{G}$. Similarly, if $\mathcal{X} = \mathcal{Y}$, f is said to be equivariant with respect to \mathfrak{G} if $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$.

For example, consider the simple case where $\mathcal{X} = \mathbb{R}^N$ and \mathfrak{G} is the group of isotropic scaling. The group actions \mathfrak{g} are then different scalings of the input vector, and the group representations $\rho(\mathfrak{g})$ are scalar matrices $c\mathbf{I}$, where $c > 0$ and \mathbf{I} is the $N \times N$ identity matrix. A scale-invariant function should then satisfy $f(c\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^N$. This can be achieved, for example, by letting $f(\mathbf{x}) = g(\mathbf{x}/\|\mathbf{x}\|)$ for some function g . Similarly, a scale-equivariant function should satisfy $f(c\mathbf{x}) = cf(\mathbf{x})$, which holds for all linear maps.

A common method used for designing neural networks that are invariant with respect to symmetry groups is to use compositions of invariant and equivariant layers, which can be realized by modifying the computational graph [30]. For example, if a neural network can be decomposed into two layers as $f_{\boldsymbol{\theta}} = f_{\boldsymbol{\theta}_1}^{\text{inv}} \circ f_{\boldsymbol{\theta}_2}^{\text{equiv}}$, where $f_{\boldsymbol{\theta}_1}^{\text{inv}}$ is \mathfrak{G} -invariant and $f_{\boldsymbol{\theta}_2}^{\text{equiv}}$ is \mathfrak{G} -equivariant, then obviously $f_{\boldsymbol{\theta}}$ is going to be \mathfrak{G} -invariant. In general, if we can decompose a deep neural network containing D layers as $f_{\boldsymbol{\theta}} = f_{\boldsymbol{\theta}_D}^D \circ \dots \circ f_{\boldsymbol{\theta}_1}^1$, where layers $1, \dots, D-1$ are equivariant, then the entire network will be invariant (equivariant) if layer

D is invariant (equivariant).

Going back to the MLP example in Figure 3, we note that the function computed by this network will in general not exhibit any type of invariance, which has inspired the development of more advanced network architecture. In this section we will take a closer look at different network architectures and their relation to input-output symmetries.

4.1 Permutation Symmetry and Learning on Sets

We will now investigate how group invariance can be applied to design network operating on unordered sets and graphs. Let us first consider the problem of function approximation on sets consisting of vector-valued elements $\{\mathbf{x}_u\}_{u=1}^N$, where $\mathbf{x}_u \in \mathbb{R}^d$. (Here, the subscript u denotes a vector-valued set element of a single data point, not the u :th member of the data set as in the previous sections). Without loss of generality, we can stack the set elements in a feature matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T$ where each row corresponds to an element of the set, but the ordering of the rows is arbitrary. The task of the network is then to output either a global feature of the entire set or element-wise features, or both. Learning on sets has several interesting applications, including 3D shape recognition from point clouds and text retrieval [143]. In contrast to pixels on a grid, the set elements need not have a well-defined ordering, and therefore our function approximator should yield the same prediction regardless of how the rows of \mathbf{X} are arranged. More precisely, we can define permutation invariance in the following way:

Definition 4.2. A function $f : \mathbb{R}^{N \times d} \rightarrow \mathcal{Y}$ is invariant with respect to the permutation group Σ_N if $f(\mathbf{P}\mathbf{X}) = f(\mathbf{X})$ for all $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $N \times N$ permutation matrices \mathbf{P} . Similarly, if $\mathcal{Y} = \mathbb{R}^{N \times d'}$, f is said to be equivariant with respect to Σ_N if $f(\mathbf{P}\mathbf{X}) = \mathbf{P}f(\mathbf{X})$.

Designing a permutation-invariant neural network puts severe restrictions on what types of layers can be used, but fortunately we can use our recipe of combining equivariant and invariant layers in order to achieve this. In general, we can describe the set of permutation invariant functions using a decomposition of two functions. Let $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^M$ be a function that acts on individual set elements \mathbf{x}_u , $u \in \{1, \dots, N\}$, i.e. the rows of \mathbf{X} , and let $\phi : \mathbb{R}^M \rightarrow \mathcal{Y}$. A permutation equivariant function can then be realized as

$$f(\mathbf{X}) = \phi \left(\bigoplus_u \psi(\mathbf{x}_u) \right), \quad (35)$$

where \bigoplus is any permutation-invariant aggregation operation, for example summation, in which case we say that this is a sum-decomposition of f via the latent space \mathbb{R}^M . This

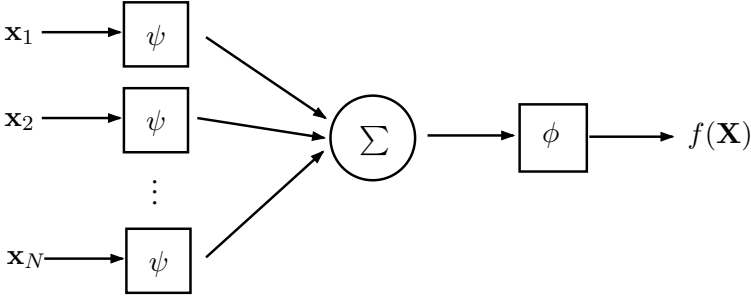


Figure 10: An illustration of the Deep Sets [143] architecture. Each set element is processed by the same neural network ψ and the outputs are aggregated using summation. The aggregated output is then fed into another neural network ϕ that predicts a global property of the set. Note that summation can be replaced by any other permutation-invariant aggregation.

suggests that we could design an invariant neural network in this way by equipping ψ and ϕ with their own set of layers and learnable parameters, a method which is known as Deep Sets [143] and is illustrated in Figure 10. A similar method, using the max-decomposition as aggregation function, has also been proposed [102].

A natural question to ask is then if this design will satisfy universal approximation and, if so, are there any restrictions on the dimension of the latent space M ? Although, there is no definitive answer under general conditions, some important results exist if we restrict ourselves to scalar sets ($d = 1$) and continuous functions with codomain $\mathcal{Y} = \mathbb{R}$.

Theorem 4.3. Wagstaff et al. (2019) [133]. Let $M > N$. Then there exist permutation invariant continuous functions $f : \mathbb{R}^M \rightarrow \mathbb{R}$ which are not sum-decomposable via \mathbb{R}^N .

The theorem gives us a necessary condition on the latent space in order to make sure that all functions can be represented, although clearly some functions exist that can be represented with a smaller latent space. In particular, the minimum size of the latent space depends on the number of elements in the set, such that larger input sets require a larger latent space. Furthermore, the following theorem proves that a latent space of size N is not only necessary, but also sufficient:

Theorem 4.4. Wagstaff et al. (2019) [133]. Let $f : \mathbb{R}^M \rightarrow \mathbb{R}$ be continuous. Then f is permutation-invariant if and only if it is continuously sum-decomposable via \mathbb{R}^M .

The reader is referred to [133] for proofs of theorems 4.3 and 4.4, which together have the following important implication: the Deep Sets architecture is guaranteed to have universal continuous function representation on sets of size N if and only if the dimension of the latent space is at least N . Although these theorems deal with universal representation, they have also been adapted for universal approximation with the same conclusions [134]. It should also be noted that this does not imply that the function approximation can always

be practically implemented using a neural network, since the choice ψ and ϕ together with the optimization procedure might result in a network that in practice does not converge to the true function. Nevertheless, it gives a strong suggestion that the latent space of the network should grow with the number of elements in the input set.

4.2 Permutation Invariant Training

For some problems, we are more interested in invariance with respect to permutations of the output rather than the input. Suppose that we have a model that outputs a set of K predictions $\{y_k\}_{k=1}^K = f(\mathbf{x})$, where the ordering of the predictions is unimportant. A useful technique for achieving this is permutation invariant training (PIT), which was first proposed for the task of speech separation [142, 141], also known as the cocktail-party problem. For this task, the input is an audio recording that consists of multiple components as $\mathbf{x} = \sum_k \mathbf{x}_k$, where each component represents a single speaker. The goal is then to train a model that separates each component in a unique output channel. When evaluating the speaker separation performance, we do not know a-priori which output should be matched to which speaker. Therefore, we need to construct a loss function that is capable of finding the best speaker permutation that matches the outputs.

Let $\text{Perm}(\{1, \dots, K\})$ denote the set of all possible permutations of the label ordering $\{1, \dots, K\}$. For each permutation α , we can then evaluate the loss between the set of model predictions $\{y_k\}_{k=1}^K$ and permuted targets $\{t_{\alpha(k)}\}_{k=1}^K$. In order to achieve permutation invariance during training, we can aggregate over all the outputs and then take the minimum over all permutations as

$$\mathcal{L} = \min_{\alpha \in \text{Perm}(\{1, \dots, K\})} \sum_k \ell(y_k, t_{\alpha(k)}), \quad (36)$$

where $\ell(\cdot, \cdot)$ is the pairwise loss that we want to minimize, for example the squared error.

A problem with PIT is that it scales poorly with the number of outputs, since $K!$ individual loss components need to be evaluated when solving the assignment problem. However, some researchers have recently proposed methods using faster techniques than brute-force computation of all permutations, such as the Hungarian algorithm which achieves $\mathcal{O}(K^3)$ complexity [41]. By relaxing the problem to approximate the best permutation, it is also possible to achieve further speedups using Sinkhorn iteration [124].

Another aspect of PIT that if the number of speakers K is not fixed, the method needs to be extended to handle an arbitrary number of outputs. This can be handled by recursively separating each speaker until a stopping criterion is reached [125]. Another method that uses auxiliary duplication is discussed in Section 5.3.

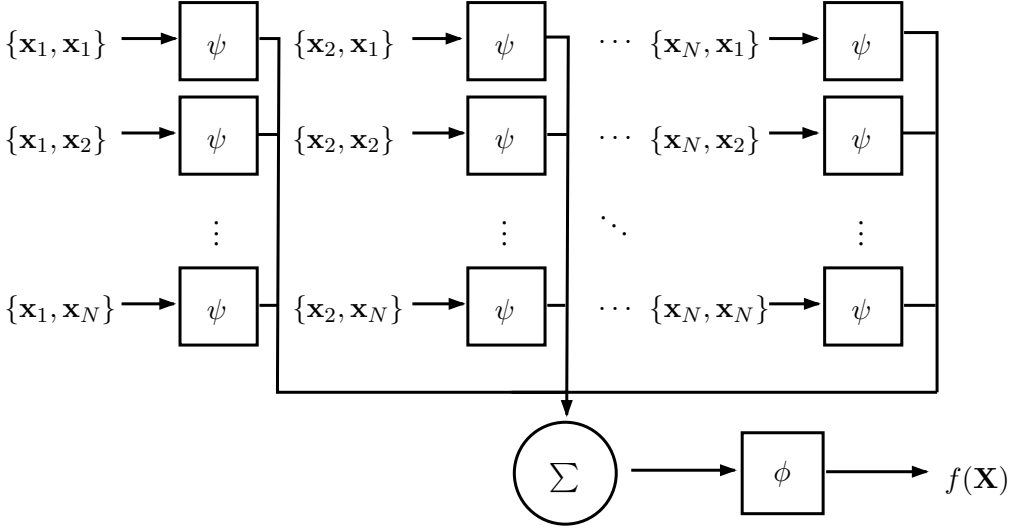


Figure 11: An illustration of the relation network [113] architecture. This can be viewed as an extension of Deep Sets, where relation features for all N^2 pair-wise combinations of set elements are computed before aggregation. In practice, some relations, for example pairing set elements with themselves, can be omitted.

4.3 Relation Networks

In certain applications, neural networks are used to reason about the relationship between set elements. For example, consider a set of images depicting various objects and suppose that we want to infer how many duplicates there are in the set. In order to do this, we can for example compare each image in the set to every other image and infer whether they are duplicates or not. When comparing the images, we are inferring a *relation* between them, which tells us something about their similarity. After inferring all relations, we need to perform an aggregation over all N^2 pair-wise relations and compute the number of duplicates. Combining these two operations (computing relations and aggregating them) can be viewed as computing a function over the the set of images.

A relation network (RN), first coined by Santoro et al. [113], is defined as

$$f(\mathbf{X}) = \phi \left(\bigoplus_{u,v} \psi(\mathbf{x}_u, \mathbf{x}_v) \right), \quad (37)$$

where the relations are typically computed for all N^2 pairwise combinations, although it might not be necessary to compare set elements to themselves. Note that in general $\psi(\mathbf{x}_u, \mathbf{x}_v) \neq \psi(\mathbf{x}_v, \mathbf{x}_u)$, which implies that both relations need to be computed in order for the whole RN to be permutation invariant, as shown in Figure 11. As we shall see in Section 5, RN's have nice properties that allow them to be applied to problems where

pairwise features are used as input.

4.4 Learning on Graphs

Learning on sets can be viewed as a special case of learning on graphs. Consider the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertices $\mathcal{V} = \{u\}_{u=1}^N$ and directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each vertex u of the graph can then be assigned certain features using row-vectors \mathbf{x}_u and the edges model interaction between the vertices. A graph neural network (GNN) can then learn to extract features from the graph and predict properties of the vertices and edges, as well as the entire-graph. Applications of GNNs include modelling of social networks, where vertices are members of the network with their associated features and edges represent interactions between members. Another example is drug discovery, where graphs are used to model interactions between atoms in a molecule [18].

In this context, learning on sets is a special case of learning on graphs, where the set of edges is empty, i.e. $\mathcal{E} = \emptyset$. However, if the set of edges is non-empty, we can use the adjacency matrix \mathbf{A} , which has elements of the form

$$a_{u,v} = \begin{cases} 1, & (u, v) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \quad (38)$$

For the sake of brevity, we only consider undirected graphs where $\mathbf{A} = \mathbf{A}^T$. Using this notation, a GNN can in general be described by a function that takes both the vertex features \mathbf{X} and the adjacency matrix \mathbf{A} as input. Similarly as for sets, we require permutation invariance for GNNs, since in general it is not possible to order the vertices of the graph in a well-defined manner. Since a permutation of the rows of \mathbf{X} implies a permutation of both the rows and columns of \mathbf{A} , we can define the notion of permutation invariance for functions on graphs as follows:

Definition 4.5. Let \mathcal{A} be the set of adjacency matrices of all the corresponding graphs of cardinality $N = |\mathcal{V}|$. A function $f : \mathbb{R}^{N \times d} \times \mathcal{A} \rightarrow \mathcal{Y}$ is invariant with respect to the permutation group Σ_N if $f(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^T) = f(\mathbf{X}, \mathbf{A})$ for all $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $N \times N$ permutation matrices \mathbf{P} . Similarly, if $\mathcal{Y} = \mathbb{R}^{N \times d'}$, f is said to be equivariant with respect to Σ_N if $f(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^T) = \mathbf{P}f(\mathbf{X}, \mathbf{A})$.

GNNs are often realized using various forms of message-passing. This means that, in each layer l of the network, the features $h^l(x_u)$ of each vertex u are updated with messages from their neighbouring vertices $\mathcal{N}_u = \{v : (u, v) \in \mathcal{E}\}$. Letting $h^0(\mathbf{x}_u) = \mathbf{x}_u$, this can be written as

$$h^l(\mathbf{x}_u) = \phi \left(h^{l-1}(\mathbf{x}_u), \bigoplus_{v \in \mathcal{N}_u} \psi(h^{l-1}(\mathbf{x}_u), h^{l-1}(\mathbf{x}_v)) \right), \quad l = 1, \dots, L. \quad (39)$$

Note that we require permutation invariance with respect to the ordering of the vertices in \mathcal{N}_u . The message passing is then repeated at each layer and if the graph is connected, i.e. there exists a path between every pair of vertices, then information can propagate between all vertices, given that there are sufficiently many layers in the network. If the goal is to predict a global feature of the graph, we can of course aggregate over all vertices in the graph in the last layer L as

$$f(\mathbf{X}) = \varphi \left(\bigoplus_u h^L(\mathbf{x}_u) \right). \quad (40)$$

Here we can make the observation that Deep Sets is a special form of GNN, because if $\mathcal{E} = \emptyset$ then each layer (39) simplifies to an update for each individual set element, without any interaction between different elements of the set. Furthermore, RNs can also be viewed as a form of GNN for complete graphs, i.e. $\mathcal{E} = \mathcal{V} \times \mathcal{V}$.

4.5 Translational Symmetry and Convolutional Neural Networks

When the input domain is a grid, which is common when the input domain consists of e.g. 2D images or 1D audio signals, it can be beneficial to work with translational symmetry. For example, when performing image classification, we want our classification function to be invariant to the location of the object inside the image. In other use cases, we might have multiple objects in the same image and the goal is to output a heatmap of detections for a particular class of objects. For this scenario, the function should be equivariant to the object locations.

For brevity, we here give the definition of translational symmetry on 1D grids. Such a grid can be viewed as a directed graph, where the connectivity is defined as

$$a_{u,v} = \begin{cases} 1, & (v - u) \bmod N = 1 \\ 0, & \text{otherwise.} \end{cases} \quad (41)$$

In other words, each node u has a directed edge to its neighbor $u + 1$. In order to deal with edge effects at the start ($u = 1$) and end ($u = N$) of the grid, these nodes should be connected. Given a signal \mathbf{X} on this grid, we can then define a translation of \mathbf{X} as circulant shifts of the rows of \mathbf{X} . More specifically, shifting \mathbf{X} with Δ grid-points upwards

can be written as \mathbf{SX} , where \mathbf{S} is an $N \times N$ circulant shift matrix with elements $s_{uv} = \delta_{u, (v+\Delta) \bmod N}$. We can now give the following definition:

Definition 4.6. A function $f : \mathbb{R}^{N \times d} \rightarrow \mathcal{Y}$ is equivariant with respect to the group of 1D translations T_N if $f(\mathbf{SX}) = f(\mathbf{X})$ for all inputs $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $N \times N$ circulant shift matrices \mathbf{S} . Similarly, if $\mathcal{Y} = \mathbb{R}^{N \times d'}$, f is said to be translation equivariant with respect to T_N if $f(\mathbf{SX}) = \mathbf{S}f(\mathbf{X})$.

It is clear that all functions of the form $f(\mathbf{X}) = \mathbf{CX}$, where \mathbf{C} is a circulant matrix, are shift equivariant. This operation is equivalent to a circular convolution, and therefore CNNs can be implemented as MLPs with circulant matrices. In practice, many CNNs that are used in practice use convolutions with other types of padding at the grid edges, which results in only approximate equivariance [68]. It is also common to use strided pooling layers, which perform a downsampling operation on the feature maps, that breaks equivariance [23].

4.6 Self-Attention and Transformers

Self-attention is a special form of message-passing where the messages passed from v to u only depend on the features of v , but each message is weighted using a scalar attention weight

$$h^l(\mathbf{x}_u) = \phi \left(h^{l-1}(\mathbf{x}_u), \bigoplus_{v \in \mathcal{N}_u} \alpha(h^{l-1}(\mathbf{x}_u), h^{l-1}(\mathbf{x}_v)) \psi(h^{l-1}(\mathbf{x}_v)) \right). \quad (42)$$

The attention weight $\alpha(h^{l-1}(\mathbf{x}_u), h^{l-1}(\mathbf{x}_v))$ can be interpreted as a form of “soft” adjacency, indicating the strength of the connection between vertices u and v . Although there are many ways to implement self-attention, the most popular method is to calculate the attention weights using the scaled dot product of linear projections of the features. Let $\mathbf{q}_u = \mathbf{x}_u \mathbf{W}_q$ and $\mathbf{k}_v = \mathbf{x}_v \mathbf{W}_k$ denote the queries and keys, where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d \times d_h}$ are learnable parameters of the model. The raw attention scores are calculated as

$$s_{uv} = \frac{\mathbf{q}_u \mathbf{k}_v^T}{\sqrt{d_h}}, \quad (43)$$

and the scaled dot-product attention weights are then given by

$$\alpha(\mathbf{x}_u, \mathbf{x}_v) = \frac{e^{s_{uv}}}{\sum_{v' \in \mathcal{N}_u} e^{s_{uv'}}}. \quad (44)$$

Note that due to the softmax normalization, we have that $\alpha(\mathbf{x}_u, \mathbf{x}_v) > 0$ and $\sum_{v \in \mathcal{N}_u} \alpha(\mathbf{x}_u, \mathbf{x}_v) = 1$. The message generating function in (42) is here simply defined as another linear projection $\psi(\mathbf{x}_v) = \mathbf{x}_v \mathbf{W}_v$ where $\mathbf{W}_v \in \mathbb{R}^{d \times d_h}$.

The scaled-dot product attention was originally proposed in [131] in order to improve natural language processing (NLP) models. In the context of NLP, self-attention is applied to sequences of tokens, which allows the model to exploit complex word-to-word interactions when processing the text. The connection between self-attention and GNNs here is subtle, but when self-attention is applied to the entire sequence, this can be regarded as modelling the sequence as a complete graph, i.e. $\mathcal{N}_u = \mathcal{V}$, $\forall u$. Using the complete graph allows us to write the self-attention weights in matrix form as

$$\tilde{\mathbf{A}}(\mathbf{X}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_h}} \right), \quad (45)$$

where $\mathbf{Q} = \mathbf{X}\mathbf{W}_q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_k$ and the softmax function is applied on each row individually. With this notation it follows that $\tilde{\mathbf{A}}_{uv} = \alpha(\mathbf{x}_u, \mathbf{x}_v)$, and we can define the self-attention (SA) operator as

$$\text{SA}(\mathbf{X}) = \tilde{\mathbf{A}}(\mathbf{X})\mathbf{V}, \quad (46)$$

where $\mathbf{V} = \mathbf{X}\mathbf{W}_v$, $\mathbf{W}_v \in \mathbb{R}^{d \times d_h}$. In order to make the operation even more expressive, it can be performed several times in parallel to for the multi-headed self-attention (MSA) operator

$$\text{MSA}(\mathbf{X}) = [\text{SA}_1(\mathbf{X}), \text{SA}_2(\mathbf{X}), \dots, \text{SA}_h(\mathbf{X})]\mathbf{W}_P, \quad (47)$$

where $\mathbf{W}_P \in \mathbb{R}^{hd_h \times d}$. GNNs based on multi-headed self-attention are known as Transformers.

Position and Modality Encodings

Transformers have become the key component in large language models (LLMs), most notably the series of Generative Pre-trained Transformer (GPT) [20, 6] models. However, since the MSA operation is permutation invariant by design, Transformers are not able to distinguish between sequences of words where words have been re-ordered. Therefore, language models use some form of internal positional encoding that contains information about the position of the word in the sequence. In practice, this is often done by either modifying the self-attention module [122] or adding a position embedding to elements of

the input features before the Transformer layers, which yields a sequence of modified input $\{\tilde{\mathbf{x}}_u\}_{u=1}^N$ as $\tilde{\mathbf{x}}_u = \mathbf{x}_u + \mathbf{p}_u$.

Several variants of positional encoding have been proposed. The first Transformer networks used a hard-coded positional encoding that consists of sinusoidal waves with frequency that depends on the absolute position of each element in the sequence [131]. Others have proposed learnable embeddings [40], that are updated as other learnable parameters of the network during training. Both hard-coded and learnable position encodings can encode either absolute or relative [32] position of the sequence elements, and the latter can be used in order to achieve translational equivariance. Relative position encoding is also more straightforward to extend to arbitrary sequence lengths, since it does not require specifying a maximum sequence length. Conditional position encodings have also been suggested [29], where \mathbf{p}_u is a function of \mathbf{x}_u and its neighbors.

Properties other than position of input elements can also be encoded by adding different encodings. Suppose for example, that the input consists of two different types of modalities: a and b . These could be for example images and their respective captions, which we can group into feature pairs $\{\mathbf{x}_u^a, \mathbf{x}_u^b\}$. Transformers can then be used to learn representations of images and text in a shared latent space, but in order for the model to distinguish between different input modalities, we need to encode them by adding modality encodings \mathbf{q}^a and \mathbf{q}^b [50]. Together with position encoding, the inputs to the model then take the form

$$\begin{aligned}\tilde{\mathbf{x}}_u^a &= \mathbf{x}_u + \mathbf{p}_u + \mathbf{q}^a \\ \tilde{\mathbf{x}}_u^b &= \mathbf{x}_u + \mathbf{p}_u + \mathbf{q}^b\end{aligned} \quad u = 1, \dots, N. \quad (48)$$

Other ways to deal with multi-modality include summation and concatenation of input features, as well as special cross-modality attention layers [139].

Learnable Input Tokens

An important feature of the Transformer architecture is its ability to deal with variable sequence lengths. This enables Transformers to process e.g. texts with different number of words. Another advantage is that it allows for prediction of multiple outputs by appending the input sequence with special elements, or “tokens”. Suppose that we have a sequence of N elements, and we want to predict the next element in the sequence. We can then append a special “mask” token \mathbf{x}_{mask} at the end of the input sequence [35], such that the input (before adding position encoding) becomes $\{\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}_{\text{mask}}\}$. Similarly, we can predict multiple elements by just appending more mask tokens. Typically, the mask token is a learnable parameter of the network, similar to the position encoding. Information is then propagated from the sequence elements to the mask tokens via the self-attention

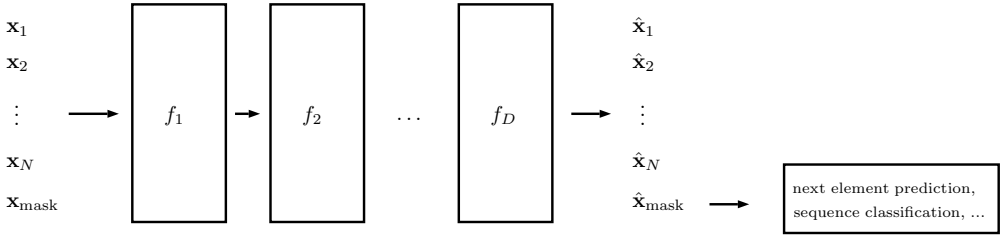


Figure 12: An illustration of the principle of mask token insertion. Before the first layer of the model, a mask token is appended at the end of the input sequence. The entire sequence is then passed through series of Transformer layers $f = f^D \circ \dots \circ f^1$, where information is transferred to the mask token via self-attention. At the final layer, the updated mask token can be used to form predictions about the input sequence.

mechanism in each Transformer layer. An illustration of this concept is shown in Figure 12. Here, the mask token is appended before the first transformer layer, but of course it could just as well be appended in-between two consecutive layers.

Inserting special tokens can also be used for summarizing the contents input sequence. For example, if we want to classify the entire sequence, we can append a token that is trained to encode the class of the inputs. This is equivalent to inserting a mask token, but sometimes it is referred to as a “class token” instead. Predicting missing elements in a sequence is also useful as a technique for unsupervised learning, where elements are randomly masked out from the input during training and replaced with mask tokens, a technique known as masked autoencoders [60].

Transformers for Image Recognition

So far, we have shown that the self-attention mechanism is one of many possible message-passing functions that can be derived from permutation invariance on sets or graphs, even though it was not originally proposed in this context. We shall now move on to discuss some problems where it is not obvious that self-attention is better than alternative methods, nor trivial to adapt to the particular setting.

When designing deep networks, the family of architectures considered usually depends on the input domain \mathcal{X} . For example, the problem of image classification has the input domain of RGB images of a particular resolution, i.e. $\mathcal{X} = \mathbb{R}^{h \times w \times 3}$. Since the domain is a three-dimensional grid of pixels, we can use convolutions and pooling with local receptive fields in order to design networks that are either invariant or equivariant with respect to the location of objects in the picture. The representation of an image on a two-dimensional grid is intuitive, in the sense that it corresponds well with the way that we as humans perceive them with our eyes. However, this does not necessarily imply that this particular representation is optimal for neural network learning. For example, the image can just as well be represented

on the one-dimensional grid \mathbb{R}^{3hw} by stacking the pixel values in a vector. Furthermore, an image can also be represented as an unordered set $\{\mathbf{x}_u\}_{u=1}^{hw}$, where each element $\mathbf{x}_u = [r^{(u)}, g^{(u)}, b^{(u)}, p_x^{(u)}, p_y^{(u)}]^T$ stores information about the color intensity and coordinates of the pixels. From a human perspective, they appear to be counter intuitive representations, but from a deep learning perspective, it is not obvious if they are better or worse suited for learning.

More generally, we can describe images, and many other types of data, as graphs. If we extract square patches from an image with resolution (P, P) , then we can represent each patch \mathbf{x}_u on the domain \mathbb{R}^{CP^2} , yielding a total of $N = hw/P^2$ patches. We can then construct a graph $\mathcal{G} = (\{\mathbf{x}_u\}_{u=1}^N, \mathcal{E})$ by defining the set of edges \mathcal{E} . There are several ways to do this, and the edges of the graph will be redundant if we also use positional encodings for each patch. One possibility is to define edges between neighbouring patches in the image. Another possibility is to assign edges based on the semantic contents of the patches, i.e. let patches with similar objects be connected, but it is not clear how the semantic similarity would be defined. Finally, it is also possible to let the graph be complete and let all patches have edges between them, and then *learn* the strength of the connections, which is exactly what a Transformer does.

Recent work in computer vision [40, 129] suggests that the patch-based representation of images might be better suited for deep learning than a grid. By treating an image like a complete graph of small patches, and applying self-attention to infer the strength of the connections, Transformer-based networks can achieve equal or stronger performance in image classification tasks compared to convolutional networks. This is in many ways remarkable, since the Transformer-based network is not inherently translation invariant, and was not originally designed for solving problems in computer vision.

Transformers for Point Cloud Recognition

An important application of GNNs is point cloud processing. Point clouds are frequently encountered in computer graphics, but can also be obtained from measurements collected by 3D scanners, such as radar or LiDAR. Common applications include simultaneous localization and mapping (SLAM), which can be used for indoor navigation, and driver assistants in vehicles. In both scenarios, point clouds are used for creating maps of the environment. In order to create global maps, point clouds from different measurements need to be registered in a common frame of reference. This requires local point features that are resilient to outliers, since the measured data often contain noise and artifacts. The features can be based on purely geometrical properties, but with the advent of deep learning it is also possible to extract semantic features that describe the contents of the observations. Furthermore, such features can also be used for scene segmentation and object detection, which is used in self-driving cars for detecting e.g. lanes, signs and other vehicles.

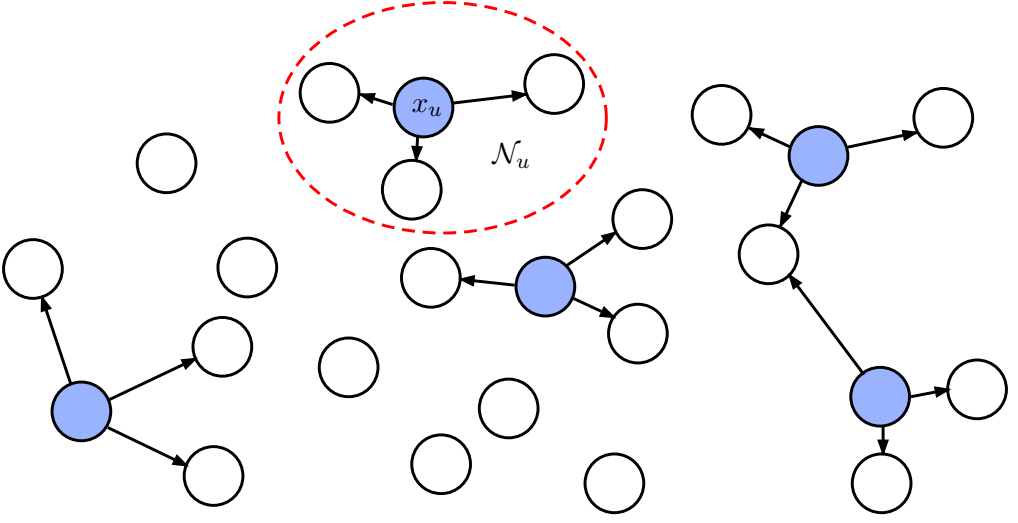


Figure 13: An example of a sparse k -NN graph created from a point cloud. Each anchor point (shown in blue), is connected with a directed edge from itself to its neighbours.

A point cloud is an unordered set $\{\mathbf{x}_u\}_{u=1}^N$, where $\mathbf{x}_u = [p_x^{(u)}, p_y^{(u)}, p_z^{(u)}]^T$ are the 3D Cartesian coordinates for an individual point. Deep networks for point cloud processing can be designed to extract global features that can be used for downstream tasks such as classification and retrieval, but also point-wise features that can be used for segmentation. Pioneering works in the field include PointNet [102] and Deep Sets [143], which used permutation invariant aggregation over the entire point cloud. It was also shown that the network satisfies universal approximation for a large enough dimension of the latent space, a result which is closely related to Theorems 4.3 and 4.4. Nevertheless, networks that treat the point cloud as an unordered set come with some limitations, namely that they do not allow interactions of neighbouring points. This motivates the introduction of GNNs, which can capture local interaction more effectively.

In order to model the point cloud as a graph, we can use the k -nearest neighbours (k -NN) of x_u . Specifically, the k -NN graph of the point cloud is defined by edges between \mathbf{x}_u and \mathbf{x}_v if \mathbf{x}_v is one of \mathbf{x}_u 's k -nearest neighbours in terms of Euclidean distance. We can then use GNNs with message-passing from all the neighbours of each point to the point itself. Other forms of local connectivity is also possible, for example by allowing edges from \mathbf{x}_u to all points within a certain distance. These ideas were first explored in [103] and [136], where the latter also considered dynamic graphs that are updated in various stages of the network.

We may also choose to model point clouds using complete graphs with message passing between all points using self-attention, which was first proposed in [75]. However, the self-attention matrix $\tilde{\mathbf{A}}$ in (45) for N points will have size $N \times N$, which implies that

the computational and memory requirements of the network will grow as $\mathcal{O}(N^2)$. For large inputs (point clouds often have thousands of points), using the complete graph will therefore not be computationally efficient.

Furthermore, it is not obvious that self-attention is useful when applied on individual point features, since a single point contains no semantic information. A natural extension is therefore to consider attention between patches of points, since a collection of neighbouring points can describe objects or parts of objects. One way to do this is to sample a sparse set of anchor points and create patches by aggregating over their neighbours, as shown in Figure 13. Applying attention between these patches would both decrease the computational complexity and capture more semantically meaningful connections between, compared to using the complete graph. These ideas are explored in more detail in Paper II, where they are also verified by extensive experiments on both real and synthetic data.

5 Audio Recognition and Sound Source Localization

Hearing is an important part of human perception that allows us to detect, understand and localize sound in our environment. Sounds perceived by humans through our hearing sense arises from rapid periodic changes in air pressure which are picked up from the tympanic membrane (ear drum) via the pinna and auditory canal (see Figure 14). The vibrations induced in the ear drum are then transmitted via the ossicles—the three small bones malleus, incus and stapes—to the cochlea, where they are further transmitted in fluid canals. At the end of the cochlea, hair cells convert the vibrations to electric signals in the nervous system that eventually make their way to the brain [99]. We will not delve into the details of human hearing, but the basic structure presented here serves as a useful blueprint for machine perception of sound.

The “ears” of a computer audition (CA) system are the microphones, that convert the sound waves to electric signals. Similar to the ear drum, they typically use a sensitive membrane that vibrates as air pressure changes. The vibrations can then be used to induce voltage changes in an electric circuit by connecting it to e.g. the plates in a capacitor or the magnet in an induction coil. In order to be able to further process the signal in a computer, it is then sampled using an analog-to-digital converter. Although this necessarily leads to some distortion of the signal due to aliasing and quantization, it is not a problem in most practical applications, given that a sufficiently large bit depth and sampling frequency is used. Since human hearing is limited to frequencies approximately between 20 Hz and 20 kHz, the Nyquist-Shannon sampling theorem tells us that a sample rate of 40 kHz suffices for purposes mimicking human auditory perception [117]. Indeed, 44.1 kHz is a commonly used sampling frequency, for example in CDs and DVDs where it is used together with a bit depth of 16 bits [109].

5.1 Audio Feature Descriptors

We can represent a digital audio signals of length N as a real-valued vector in \mathbb{R}^N . When performing perceptive tasks, one option is to use this directly as input to a neural network and train it to learn better internal representations. However, for many applications, it is common to use other input representations of the signal. Here we will briefly introduce the signal spectrogram, cepstrum and the mel-scale representations, which are commonly used for audio recognition tasks, as well as techniques for learning features from raw audio waveforms.

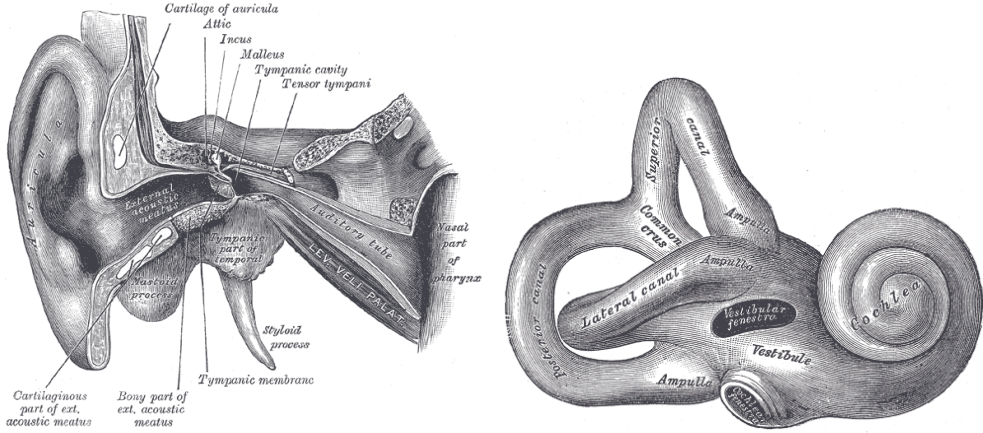


Figure 14: Anatomy of the human external and middle (left) and inner (right) ear. The shape of the outer ear filters the sound such that the spectral response is changed depending on the direction of arrival of the sound. The eardrum (tympanic membrane) transmits the pressure changes in the ear via the ossicles to the cochlear fluid. Hair cells inside the cochlea then convert the fluid vibration to electric signals which can be interpreted by the brain. Illustration borrowed from Henry Gray's *Anatomy of the Human Body* [54].

Spectrograms, Ceptograms and the Mel-frequency Scale

Let $\mathcal{F} : \mathbb{R}^N \rightarrow \mathbb{C}^N$ denote the discrete Fourier transform (DFT) of a real-valued signal. The DFT pairs $\mathbf{X} = \mathcal{F}(\mathbf{x})$ is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi kn/N} \Leftrightarrow x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi kn/N}, \quad (49)$$

for time samples $n = 0, \dots, N-1$ and frequencies $k = 0, \dots, N-1$. Since \mathbf{x} is real-valued, it follows that \mathbf{X} is even symmetric, i.e. $X[k] = X^*[-k \bmod N]$, where $*$ denotes the complex conjugate. The square of the absolute value of the transformed signal $|X[k]|^2$ is often referred to as the *power spectrum*, or simply the spectrum, of \mathbf{x} .

An important property of the DFT is that shifting the signal in the time domain corresponds to multiplying the signal with a linear phase in the frequency domain. Specifically, if we let $(\mathbf{x})_\tau$ denote a circular shift of $\tau \in \mathbb{Z}$ samples, then

$$\mathcal{F}((\mathbf{x})_\tau)[k] = X[k] e^{-i2\pi k\tau/N}. \quad (50)$$

A consequence of this is that the spectrum is invariant to circular shifts of the signal in the time domain. This makes the spectrum a useful feature when we are only interested in the frequency contents in the signal and do not care about “where” in the signal these

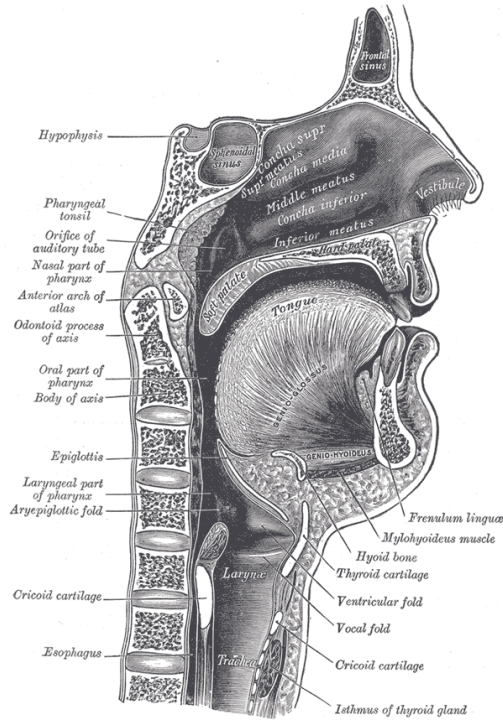


Figure 15: Anatomy of the human vocal tract. Sounds are produced by the vocal cords, also known as vocal folds, and their vibration frequency determines the pitch of the sound. The as the sound proceeds through the pharynx, its characteristics are changed and this process gives rise to a formant structure that depends on the position of the jaw and tongue. Roughly, this process can be described as linear filtering. Illustration borrowed from Henry Gray's *Anatomy of the Human Body* [54].

frequencies are active. Suppose, for instance, we want to train a neural network to detect the presence of a sound with a particular frequency pattern in audio signals that otherwise only contain silence. Using the spectrum as the input feature would automatically make the outcome of the prediction invariant to where in the signal the pattern occurs. Hence, by using the power spectrum, we can achieve invariance without using an invariant network architecture, e.g. a CNN with max-pooling.

For some types of audio signals, for example in music or human speech, the frequencies are associated with multiple overtones. Given a fundamental frequency f_0 , the overtones are the multiples $f_m = m f_0$, where m is a positive integer. The energy in the overtones usually decays with higher frequency. Fundamental frequency detection, also known as pitch estimation, is an important problem related to music and speech applications [28].

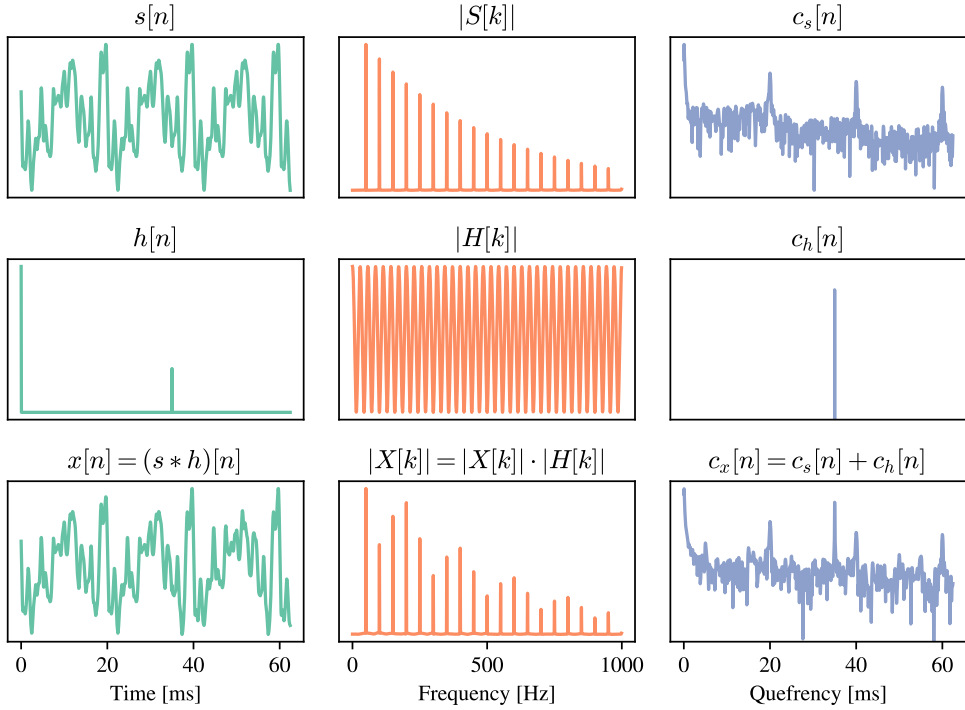


Figure 16: Illustrative example of the additive properties of the signal cepstrum. Here, a signal sampled at 16 kHz with a fundamental frequency of $f_0 = 50$ Hz is convolved with an impulse response consisting of a direct path and an attenuated echo arriving with a delay of $\tau = 35$ ms. The spectrum of the received signal therefore contains two periodic components that can be resolved in the cepstrum. The first peak at quefrency $20 \text{ ms} = 1 / (50 \text{ Hz})$ corresponds to the fundamental frequency and the second one at 35 ms corresponds to the time delay of the echo.

A feature which is sometimes used for speech analysis is the so called *cepstrum* [93] of a signal, which is defined as

$$\mathbf{c}_x = \left| \mathcal{F}^{-1} \left\{ \log |\mathcal{F}\{\mathbf{x}\}|^2 \right\} \right|^2. \quad (51)$$

In order to gain intuition for the cepstrum as a feature, we can give a toy example of a signal consisting of a fundamental frequency f_0 and M overtones. In the time domain, we can write this as $s[n] = \sum_{m=0}^M \alpha^m e^{2\pi i m f_0 n}$, where $0 < \alpha < 1$ is the decay factor of the overtones. For simplicity we have assumed a complex-valued signal, although the following derivations can be made for real signals as well. The spectrum of the signal is given by $S[k] = \sum_{m=1}^M \alpha^m \delta[k - m f_0]$. This is a “pulse train”, with equidistantly spaced pulses f_0 units apart. In other words, the spectrum of the signal is periodic with a frequency of f_0 .

Suppose now that the original signal is recorded in an environment where there is an

attenuated echo of the signal arriving at the microphone slightly after the direct path. The impulse response from the sound source to the microphone can then be written as $h[n] = \delta[n] + \beta\delta[n - \tau]$, with $0 < \beta < 1$, and the recorded signal is the convolution between the original sound and impulse response, i.e. $x[n] = (s * h)[n]$. Due to the properties of the DFT, the spectrum of the received signal can be written as a product: $|X[k]|^2 = |S[k]|^2 |H[k]|^2$. Since $H[k]$ is periodic with a period of τ , this will result in a spectrum that is periodical in both f_0 and τ . Separating these two components can be difficult using spectral analysis, which motivates the use of the cepstrum.

From Equation (51), we can see that the cepstrum is defined as the absolute value squared of the inverse DFT of the logarithm of the spectrum. Note that taking the logarithm of the spectrum results in the two periodic components of the spectrum becoming additive rather than multiplicative. Taking the inverse DFT then yields a “spectrum of the spectrum”, which has peaks at the *quefrequencies* corresponding to f_0 and τ . Consequently, cepstral analysis allows us to resolve both frequencies and echos in a single feature. See Figure 16 for an illustration of this example.

In general, the cepstrum can be used for separating signals from other types of transfer functions, such as distortions induced by the recording equipment, which makes it a useful feature for audio analysis. Another form of filtering also takes place in the human vocal tract. As sounds are produced in the vocal cords, they pass through the vocal tract before exiting the mouth, as shown in Figure 15. This process can be modeled as a form of convolutional filtering, where the impulse response of the vocal tract gives rise to a periodic formant structure corresponding to harmonics excited by the resonance. Cepstral analysis therefore provides a good tool for finding the pitch, i.e. fundamental frequency, in human speech [92].

So far, we have dealt with spectral representations on a linear scale. However, human perception of frequencies is not linear, since it is easier for us to distinguish different frequencies at the lower end of the hearing spectrum than the in the higher. Therefore, perceptual scales of frequencies are often used, which are typically logarithmic. The most common one is the *mel* scale [121], for which the mapping is given by

$$f_{\text{mel}} = 2595 \log_{10} \left(1 + \frac{f_{\text{Hz}}}{700} \right). \quad (52)$$

Using the mel-scale allows us to compute the mel spectrum, log-mel spectrum and the mel cepstrum (or “mel frequency cepstral coefficients”, often abbreviated as MFCC), which are all widely used features, in particular for speech recognition. The conversion from Hz to mel scale is in practice done by applying a filter bank with triangular kernels to the spectrum. Therefore, the number of filters (and coefficients for the MFCC) is a hyperparameter that needs to be chosen. For audio classification, between 20 and 80 filters are typically

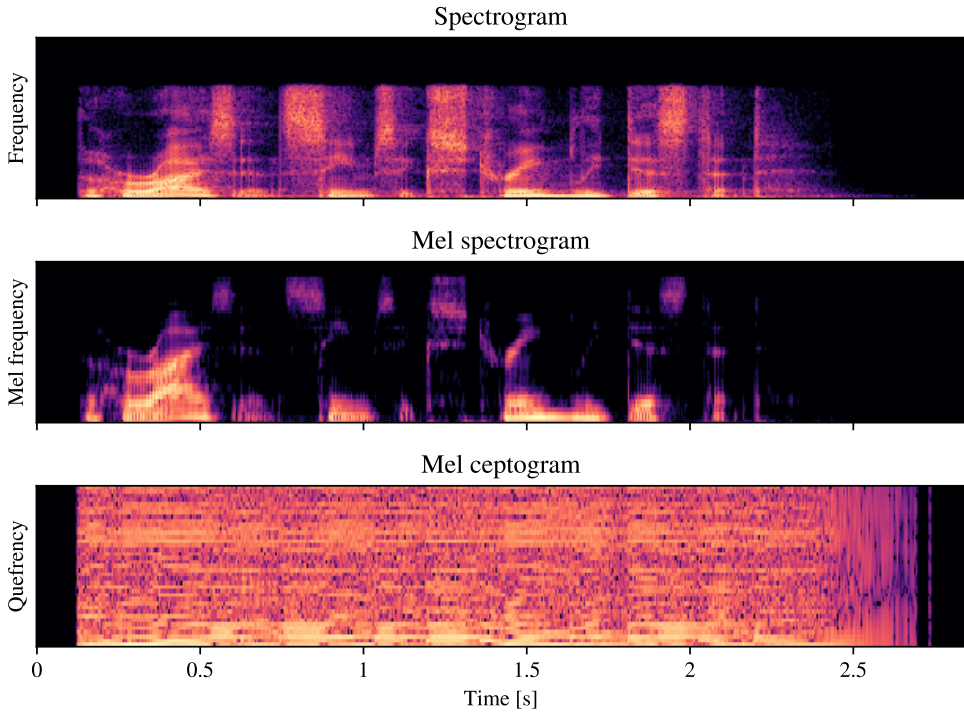


Figure 17: Different representations of an audio recording of a speaker uttering the sentence “the horizon seems extremely distant”. Can you spot where the letter “s” is being pronounced?

used, where fewer filters often lead to more robust performance [128]. In terms of the number of MFCC coefficients, the range is similar, although for vowel classification five coefficients is descriptive enough [67].

When the patterns in the signals vary over time, the spectrum or cepstrum are not suitable as features for long signals. In human speech, for instance, the frequency pattern changes over time as different syllables are pronounced. Clearly, the order in which the syllables occur is also important for understanding speech, yet this information is not captured by the signal spectrum.

A better alternative is to use the *spectrogram* (or *ceptogram*) of the signal, which consists of multiple power spectrums (or cepstrums), where each feature is calculated over a short time window where the frequency content can be regarded as stationary. For speech, it is common to use a window length between 20 and 40 ms [128]. Recent work has also suggested to use a learnable window length that is dynamically adjusted based on the training data [84]. Figure 17 shows illustrations of different time-varying representations, where the warping between linear and mel-scale frequency can be seen.

Learnable Feature Descriptors

Although the mel-scale representation of frequencies provides a good heuristic for human perception, this does not automatically imply that it is the best representation. Therefore, a natural extension of this idea is to try and learn a better representation that is well-suited for a particular task. A simple way to do this is to use convolutional filters, using e.g. a CNN, that learns to adapt to the data and extract useful features. Several works [94, 112, 63] have shown that this approach is able to match the performance of that when using log-mel or MFCC input features for speech recognition tasks. However, one downside to this approach is that due to the lack of inductive bias it requires a large enough dataset for the filters to learn meaningful features. Furthermore, it requires a potentially large number of parameters compared to computing hand-crafted features such as the MFCC.

In order to learn features more efficiently, some works have proposed a strategy of learning features by adding constraints to the filters [111, 116]. Notably, Ravanelli and Bengio proposed the SincNet architecture [104], in which the filters in the first layer of the CNN is replaced by learnable bandpass filters. In time domain, we can write these filters as

$$g[n] = 2f_2 \text{sinc}(2\pi f_2 n) - 2f_1 \text{sinc}(2\pi f_1 n), \quad (53)$$

which in the frequency domain becomes

$$G[k] = \text{rect}\left(\frac{k}{2f_2}\right) - \text{rect}\left(\frac{k}{2f_1}\right), \quad (54)$$

where $\text{rect}(\cdot)$ is the rectangular function and f_1 and f_2 are the learnable cutoff frequencies of the bandpass, with $f_2 > f_1 > 0$. During training, the cutoff frequencies are updated and adapted to find the important frequency ranges in the training data. This implies that there are only two parameters per filter, regardless of the filter length, which makes them less expensive compared to regular convolutional layers. Furthermore, we can insert inductive bias into the model by choosing a clever initialization of the cutoff frequencies, such as the ones corresponding to the mel-scale.

For the tasks of speaker identification and verification, it was shown that SincNet features yielded lower error rates compared to using a fixed filterbank, regular CNN filters or MFCC features [104]. Even more, the learned frequency response of the bandpass filters showed visible peaks corresponding to the pitch and first and second formants for English vowels. However, it is difficult to say which types of audio features are “best” in general, since this might be task specific, and also depend on the amount of training data available. For small datasets we can expect hand-crafted features to have an advantage. Some studies

have compared using hand-crafted and learnable filterbanks and handcrafted features for the task of keyword spotting, [82, 83] and found no significant difference in performance, although learnable filter banks were found to be more noise robust. When it comes to using different hand-crafted features, log-mel spectrograms and MFCC give roughly the same performance [105, 42]. One study used neural architecture search and found that using MFCC features was able to achieve better accuracy than SincNet features, but at the cost of higher parameter count [98]. In the next section, we will explore keyword spotting more in depth and compare some of the different approaches that can be used for this task.

5.2 Speech Recognition and Keyword Spotting

An important use-case of the audio feature descriptors described in the previous section is automatic speech recognition (ASR). Deep learning is well suited for this problem, since it is difficult to derive hand-crafted heuristics that transcribe the semantic contents of the speech signal. By leveraging large amounts of training data together with deep neural networks, ASR has now become a standard feature of our daily lives, for example in smart voice assistants.

An example of a smart assistant pipeline is shown in Figure 18. The first step in the process is wake-word detection, which is often done in an always on manner, that triggers the device to analyze the audio further. The second step is keyword spotting, where the goal is to detect the presence of keywords from a small dictionary. Examples of keywords could be commands like “play” or “pause” in the context of listening to music. If no keyword is detected, then ASR is triggered, which is sometimes done on a remote server if on-device computational resources are limited. There are several advantages of performing each step locally on-device, including reduced latency. Data privacy can also be of concern and local audio processing reduces the amount of potentially sensitive data that needs to be transmitted to the server.

In Paper III, we focus on the problem of keyword spotting, which is an audio classification task where the goal is to classify short snippets of speech as one of the keywords in the dictionary or “silence” or “unknown”. As previously mentioned, most state-of-the-art keyword spotting methods do not operate on the raw audio input, but on pre-processed audio representation such as mel-spectrograms or MFCC, as shown in Figure 19.

Let $\mathbf{c}_n \in \mathbb{R}^d$ denote the d cepstral coefficients of the n :th time window. By stacking all time windows in a matrix as $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_T]^T \in \mathbb{R}^{T \times d}$, we get a mel-ceptogram, which resembles an image on a grid, with one grid axes for time and one for mel-scale quefrequency. The most common network architecture for keyword spotting has been CNNs [81]. Since the ceptogram can be viewed as a $T \times d$ image, architectures from computer vision can be directly applied to this type of input. Variants of this architecture that have been explored

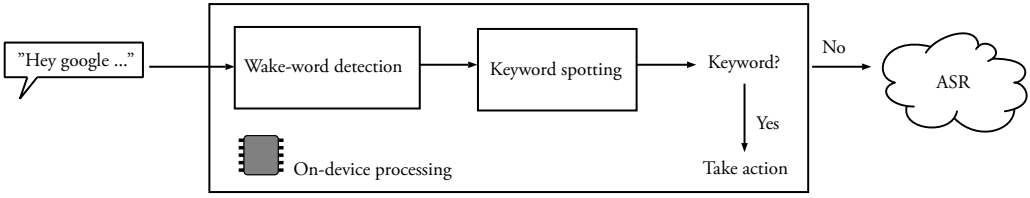


Figure 18: An example speech processing pipeline for smart assistants. Wake-word detection and keyword spotting are performed on-device. If no keyword is detected, the audio data is sent to a server where ASR is performed.

include depth-wise separable [145], temporal [27] and recurrent [34] CNNs.

However, given the surprising success of Transformers in computer vision, an interesting question is whether this architecture also works well for audio processing. In the same way that an image can be partitioned into patches, this can also be done to the spectrogram. Unlike for the case of point clouds, the ordering of the time slots matter, so positional encoding is necessary in order for the Transformer to exploit this information. In Paper III we introduce the Keyword Transformer (KWT) and demonstrate superior classification performance to CNNs, while being more efficient in terms of latency.

Concurrent work introduced the Audio Spectrogram Transformer (AST) [51], expanding the idea of Transformers for speech classification to longer audio sequences and other types of audio. Subsequent variants of AST [52, 10] include elements of self-supervised learning by reconstructing masked patches in the spectrogram, i.e. using AST as a form of masked autoencoder. This highlights another benefit of modeling audio as patches, since it allows for exploiting training techniques that are well-suited for Transformers.

Subsequent work has developed further optimizations of the KWT models. Jelčicová and Verhelst showed that 80 % of the multiply-accumulate operations in KWT can be removed while maintaining the overall classification accuracy of the model [65]. This is achieved by noting that there exists a redundancy in the input MFCC tokens, because many of them carry similar information. By pruning tokens based on a similarity threshold, only a sparse set of tokens are fed through the self-attention and MLP layers. Furthermore, this method does not require any additional training of the model. Al-Qawlaq et al. proposed KWT-Tiny [7], which only uses 0.3 % of the original parameter count by using fewer cepstral coefficients, smaller embedding dimension and INT8 quantization. However, the model was designed for only binary classification of two different keywords and yielded an accuracy loss of 10 percentage points.

Training speech recognition models in a centralized setting could potentially require collecting large amounts of speech recordings from users. An alternative would be to use a distributed training method in order to avoid collecting private data. In Paper IV we therefore explore training the KWT in a federated learning setting. Federated keyword spotting was first explored in [76] and the setup can be simulated by splitting the dataset based on

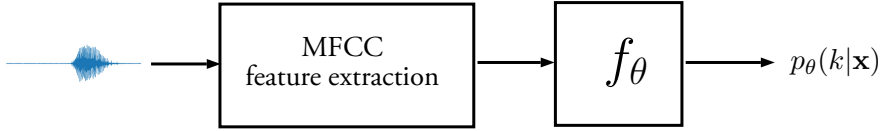


Figure 19: A keyword spotting pipeline. The raw audio waveform is first processed by extracting the MFCC spectrogram, which is then fed into a neural network that predicts a probability distribution over the different keywords.

speaker identity, as would be the case in a realistic scenario. Furthermore, it would involve training on low-end edge devices, and we therefore combine federated learning with low-precision training. The results indicate that training KWT in FP8 rather than FP32 yields a small drop in accuracy, but with the benefit of lower communication costs and potential energy savings.

5.3 Sound Source Localization

Auditory perception does not only involve interpreting the contents of audio, but also physically localizing the audio source in 3D space. Just as humans benefit from having two ears when localizing sound sources, most sound source localization methods rely on recordings from multiple microphones. Although localization using a single microphone is possible by exploiting echoes in a room with known geometry [95] or by using microphones with direction-dependent filtering [114], we focus on multichannel recordings in this thesis.

Classical methods for sound source localization (SSL) include trilateration and multilateration, both of which depend on measuring distances between the sound source and microphones. If the sound source and microphones are synchronized, and the time at which the sound event occurred is known, the distances can be measured by estimating the time of flight, i.e. the time it takes for the sound to travel from the sound source to the microphone. Trilateration is then the process of computing the sound source coordinate, given the distance measurements and the known microphone array geometry. Solving the trilateration problem amounts to finding the intersection of spheres and closed form solutions exist in the minimal case using three microphones [74].

However, in most SSL scenarios we do not know at which point in time the sound was transmitted, and the distances cannot be measured directly. Instead, we can resort to measuring differences between distances by estimating the time difference of arrival (TDOA) between pairs of microphones. Multilateration then refers to the process of computing the sound source coordinate given the distance difference measurements and the known microphone locations, which corresponds to finding the intersection between hyperboloids. Closed form solutions to the multilateration problem therefore exist in the minimal case with four microphones. Since the solutions to the minimal trilateration and multilateration problems are not unique, additional measurements can be used to form over-determined

systems that can be solved in the least squares sense. For the general 3D localization problem with non-colinear microphone coordinates, at least four microphones are required for a unique least squares solution to the trilateration problem, and five for multilateration [39].

If the microphone locations are also unknown and we want to localize them simultaneously, the problem can still be solved up to a coordinate transformation by exploiting movements of the sound source. Techniques like multidimensional scaling can then be used to solve for sound source and microphone locations jointly [74].

If the distance between the microphones in the array is small compared to the distance to the sound source, the incident sound can be approximated as planar waveforms. In this setup it is appropriate to estimate the direction of arrival (DOA), since movements towards or away from the array are difficult to distinguish from TDOA measurements. A common application of DOA estimation is spatial filtering, which allows sound to be amplified or attenuated based on its location [73].

In recent years, SSL methods based on deep learning have become more popular. Although classical methods such as multilateration perform well in many scenarios, their localization performance is limited by the accuracy of the TDOA estimates. If these were error-free, then the sound source location can be recovered exactly. However, in adverse acoustic conditions, noise and reverberation can lead to unreliable TDOA estimates, which then lead to poor localization performance. The benefits of using deep learning methods for SSL is therefore that it allows for dealing with such acoustic conditions, which can be hard to model explicitly. Deep learning methods can be trained to predict the sound source coordinates (or DOA) directly, or indirectly by predicting e.g. the TDOAs [57].

Paper V, VI and VII all deal with methods for SSL. The contributions in the papers are related to two different aspects of the problem:

1. Learning feature descriptors, which can be used as input to a localization method. In Paper V, we develop a method for improving TDOA estimation for pairs of microphones. These estimates can then be used as input features to a localization method, e.g. a multilateration algorithm or a machine learning method, which we demonstrate in Paper VI and VII.
2. Learning to predict the sound source location, given the microphone coordinates and some set of input features from microphone recordings. This problem is dealt with in Paper VI (using a small tetrahedral microphone array) and Paper VII (using a general ad-hoc microphone setup).

Room Acoustics

Suppose there is a single sound source located at $\mathbf{s} \in \mathbb{R}^3$ and a set of M microphones located at $\mathbf{r}_i \in \mathbb{R}^3$, $i = 1, \dots, M$. The received signal at microphone i can then be modeled as

$$x_i[n] = (h_{\mathbf{s}, \mathbf{r}_i} * u)[n] + w_i[n], \quad (55)$$

where u is the sound emitted by the source and $h_{\mathbf{s}, \mathbf{r}_i}$ is the impulse response from the source to the i :th microphone and w_i is i.i.d. white additive noise. The impulse response depends on the length of the path from source to the microphone, as well as indirect paths that depend on the room geometry and materials. For training SSL methods, it is common to train on synthetic data, which requires realistic simulation of sound wave propagation. With the model in Equation (55), this can be done by simulating the impulse response for a given room geometry with known material properties. A frequently used method is the so called image source method [8], which exploits the fact that an indirect path caused by a reflection on a surface can be modeled as a mirrored “image” source on the opposite side of the surface. Second order reflections then create new mirrored sources, and so on. Due to the exponential growth in the number of images, higher order reflections need to be truncated at some point for tractability.

Given the set of source images $\mathcal{V}_r(\mathbf{s})$ and the absorption coefficient $\alpha \in [0, 1]$ of the wall materials, the impulse response can be generated using Green’s function as

$$h_{\mathbf{s}, \mathbf{r}_i}[n] = \sum_{\tilde{\mathbf{s}} \in \mathcal{V}_r(\mathbf{s})} \frac{(1 - \alpha)^{\text{gen}(\tilde{\mathbf{s}})}}{4\pi \|\mathbf{r}_i - \tilde{\mathbf{s}}\|} \delta_{\text{LP}} \left(n - F_s \frac{\|\mathbf{r}_i - \tilde{\mathbf{s}}\|}{c} \right), \quad (56)$$

where $\text{gen}(\tilde{\mathbf{s}})$ is the reflection order of the image source $\tilde{\mathbf{s}}$ and δ_{LP} is a windowed sinc function (an ideal low-pass filter) that samples the ideal impulse response. Several software implementations of the image source method exist that allows for simulating acoustics in indoor environments, where the room geometry and materials can be specified [115, 38]. This is useful for both training and evaluating SSL models when real recordings are not available. It is also possible to measure impulse responses in a physical room using frequency sweeps [85]. The recordings can then be used to simulate realistic propagation together with some database of sound events [106].

In order to understand how an SSL method is affected by the room acoustics, we need a way to quantify how reverberant a given room is. A common metric for this is the reverberation time t_{60} , which is defined as the time it takes for the sound pressure level to be reduced

by 60 dB. The reverberation time depends on the room volume and wall materials. An approximation is given by Sabine's formula [110]

$$t_{60} = 0.161 \frac{V}{\alpha S}, \quad (57)$$

where V is the total volume of the room, S is the total surface area of the walls and α is the absorption coefficient. In reality, the reverberation time is frequency dependent and also depends on the geometry of the room, but for our purposes we can use this formula when evaluating SSL methods. Note that we have assumed that the absorption is the same for all walls in the room, which allows us to invert the formula and compute the absorption coefficient needed to produce a room with a given reverberation time.

Time Delay Estimation

As previously mentioned, classical methods such as multilateration depend on TDOA measurements from microphone pairs. The TDOA τ_{ij} (as measured in samples) for the microphone pair (i, j) is defined as

$$\tau_{ij} = \lfloor \frac{F_s}{c} (||\mathbf{s} - \mathbf{r}_i||_2 - ||\mathbf{s} - \mathbf{r}_j||_2) \rfloor, \quad (58)$$

where F_s is the sampling rate and c is the speed of sound. For the sake of simplicity, let's consider an example with two microphones where there is no noise or reverberation and the received signals only differ in terms of amplitude and time delay. We can then write the two signals as

$$\begin{aligned} x_1[n] &= a_1 u[n - t_1], \\ x_2[n] &= a_2 u[n - t_2], \end{aligned} \quad (59)$$

where $t_1 - t_2 = \tau_{12}$ is the TDOA and $a_1, a_2 > 0$. A useful tool for TDOA estimation is the cross-correlation function, which we here define for a real-valued function as

$$(x_1 \star x_2)[m] = \sum_{n=0}^{N-1} x_1[n] x_2[(n - m) \bmod N]. \quad (60)$$

Note that this corresponds to a circular convolution between $x_1[n]$ and the time-reversed $x_2[-n \bmod N]$. The convolution theorem states that circular convolution in the time

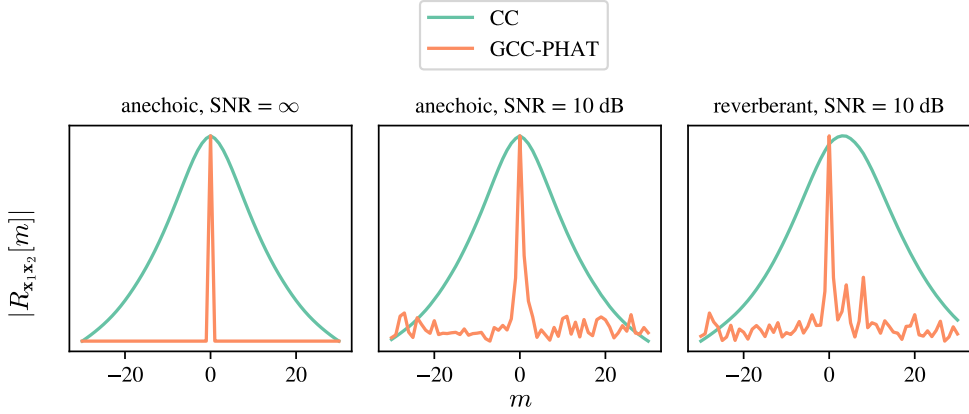


Figure 20: Illustration of the cross correlation (CC) with and without PHAT filtering in three different setups, using the signal from Figure 16. In the leftmost figure, the two signals are identical and noise-free. The CC therefore corresponds to the autocorrelation of the transmitted signal, with a peak centered at a delay of 0 samples. Similarly, GCC-PHAT is a unit pulse centered the same delay. In the center figure, additive white noise has been added to both signals, with a signal-to-noise ratio (SNR) of 10 dB. In the rightmost figure, two echoes with delays of 4 and 8 samples have been added. This causes smearing of the CC, resulting in a peak that is not centered at the correct delay. In contrast, the GCC-PHAT resolves the echoes as two additional peaks with lower amplitude.

domain corresponds to element-wise multiplication in the frequency domain, i.e. $\mathcal{F}(\mathbf{x}_1 \star \mathbf{x}_2) = \mathcal{F}(\mathbf{x}_1) \odot \mathcal{F}(\mathbf{x}_2)$. We can use this to compute the cross-correlation as

$$(\mathbf{x}_1 \star \mathbf{x}_2) = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}_1) \odot \mathcal{F}(\mathbf{x}_2)^*), \quad (61)$$

where we have used the fact that time reversal corresponds to conjugation in the frequency domain for real-valued signals. Equivalently, we can thus write the cross-correlation as

$$(x_1 \star x_2)[m] = \frac{1}{N} \sum_{k=0}^{N-1} X_1[k] X_2^*[k] e^{\frac{i2\pi km}{N}}. \quad (62)$$

The TDOA can then be found as the delay which maximizes the cross-correlation. However, the peak in the cross-correlation is not always sharp, since its shape depends on the autocorrelation function of the transmitted signal. In scenarios where reflections are present in the signal, this can cause multiple peaks in the cross-correlation to overlap an “smear” the peak of the direct path. A well-established method for dealing with this problem is to introduce a pre-filtering of the signals, which can be done directly in the frequency domain. This method is known as the generalized cross-correlation (GCC) [70], which is given by

$$R_{\mathbf{x}_1\mathbf{x}_2}[m] = \frac{1}{N} \sum_{k=0}^{N-1} W_{\mathbf{x}_1\mathbf{x}_2}[k] X_1[k] X_2^*[k] e^{\frac{i2\pi km}{N}}, \quad (63)$$

where $W_{\mathbf{x}_1\mathbf{x}_2}$ is a linear filter. If the spectrum of the signal and noise are known a priori, one can construct the maximum likelihood filter, but in practice we often do not have access to this information. The most widely adopted GCC uses the phase transform (PHAT) filter, which is given by

$$W_{\mathbf{x}_1\mathbf{x}_2}^{\text{PHAT}}[k] = \frac{1}{|X_1[k] X_2^*[k]|}. \quad (64)$$

The PHAT filter weights the contribution to the phase of each frequency in the cross-correlation by normalization with the spectral amplitude. Experiments have also shown that in practice, the PHAT filter yields better predictions than using the ML filter [17]. A potential negative side-effect of the PHAT filter is that for frequencies with zero power, the phase is undefined. Division by zero can be avoided with by adding a small constant to the denominator, but it will still result in an incorrect phase estimate contributing to the total sum.

In order to see why the GCC-PHAT generates sharper peaks than the standard cross-correlation, consider again the simplified model in Equation (59). Here, the GCC-PHAT becomes

$$\begin{aligned} R_{\mathbf{x}_1\mathbf{x}_2}[m] &= \frac{1}{N} \sum_{k=0}^{N-1} \frac{a_1 U[k] e^{-\frac{i2\pi kt_1}{N}} a_2 U^*[k] e^{\frac{i2\pi kt_2}{N}}}{\left| a_1 U[k] e^{-\frac{i2\pi kt_1}{N}} a_2 U^*[k] e^{\frac{i2\pi kt_2}{N}} \right|} e^{\frac{i2\pi km}{N}} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{i2\pi(m-(t_1-t_2))k}{N}} = \delta_{t_1-t_2}[m] \end{aligned} \quad (65)$$

The peak is a unit pulse centered at the TDOA for the microphone pair and does not depend on the autocorrelation of the transmitted signal, which makes it is easier to resolve peaks from multiple paths (see Figure 20). Note that the result only holds exactly for circular signal delays, which is not realistic. In practice, even in perfect noise-free acoustic conditions, there will be parts of the two signals at the beginning and end of the recording that do not overlap. However, if the time delay is small compared to the signal length, the peak can still be clearly resolved.

In Paper V, we expand on the idea of pre-filtering the signals by using *learnable* filters. This allows the filters to adapt to a particular type of signal as well as removing additive

noise. Furthermore, we use multiple non-linear filters, compute multiple correlations for each filtered signal pair and then combine the result into a single output. However, we want to make sure that applying the filters does not change or remove information about the TDOA, which is what we ultimately want to recover. From Section 4.5, we know that this information is preserved for CNN's with circular convolutions, and we can apply the following lemma for cross-correlations:

Lemma 5.1. Let $f : \mathbb{R}^N \rightarrow \mathbb{R}^{N \times d}$ and $g : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^N$ be translation-equivariant neural networks. Then the neural cross-correlation resulting from applying f and g before and after the cross-correlation is equivariant to the difference in circular translations $t_1, t_2 \in \mathbb{Z}$ of the two input signals, i.e.

$$g(f((\mathbf{x}_1)_{t_1}) \star f((\mathbf{x}_2)_{t_2})) = (g(f(\mathbf{x}_1) \star f(\mathbf{x}_2)))_{t_1 - t_2}. \quad (66)$$

Proof. The lemma follows directly from the composition of equivariant functions and the definition of the cross-correlation.

When the standard cross-correlation is exchanged for GCC-PHAT, the same property holds, since the frequency weighting-function is invariant to translations and delays. Hence, we can construct a neural GCC-PHAT (NGCC-PHAT), by letting f and g be two CNNs. Furthermore, we can use the SincNet architecture previously discussed, in order to learn to extract features in the relevant frequency bands. Training of the weights can then be done by predicting TDOAs between pairs of microphones and backpropagating through the GCC-PHAT operation¹. This idea is demonstrated in Paper V, showing that this improves TDOA prediction accuracy compared to the classic GCC-PHAT. In Paper VII we also demonstrate improved localization performance using a multilateration algorithm with TDOA estimates from NGCC-PHAT. In Paper VI, we extend the method to multiple sound events, which we will discuss in the next section.

Sound Event Localization and Detection

For small microphone arrays with known geometry, GCC-PHAT correlations can be mapped directly to the DOA by using a steered-response power (SRP). Using M microphones, it is calculated by summing the contributions of each microphone pair at delays corresponding to different source locations $\mathbf{s} \in \mathbb{R}^3$ as

¹Although the PHAT filter is not everywhere complex-differentiable, backpropagation is still possible by using Wirtinger derivatives [1].

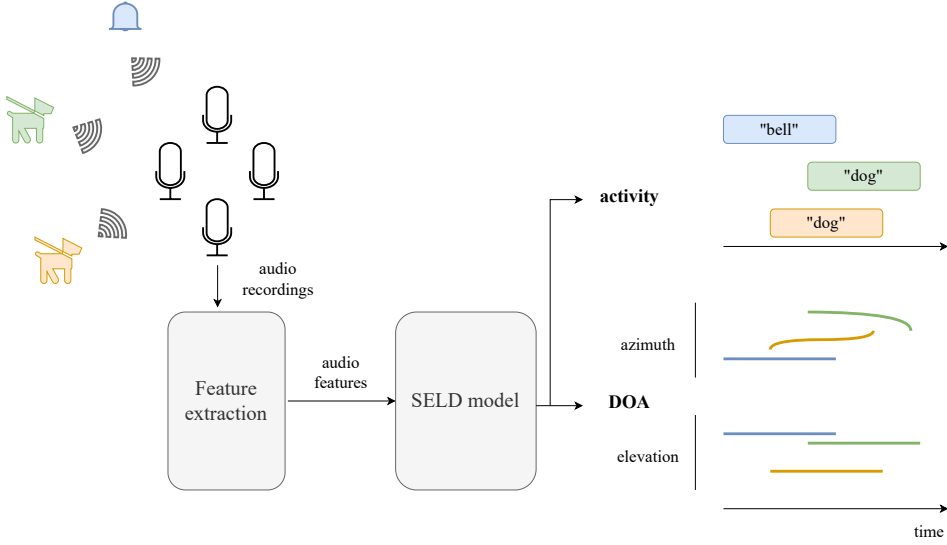


Figure 21: Overview of the SELD task considered in Paper VI. Given a stationary microphone array, the task consists of 1) detecting and classifying active, possibly overlapping, sound events and 2) localizing them in terms of the DOA. Both the detection and localization are done on a frame-by-frame basis in order to perform tracking of events over time. The task can also be extended to include distance estimation.

$$P(\mathbf{s}) = \sum_{i=1}^M \sum_{j=1}^M R_{\mathbf{x}_i \mathbf{x}_j} [\tau_{ij}(\mathbf{s})]. \quad (67)$$

An estimate of the DOA is then given by the angle to the coordinate $\hat{\mathbf{s}} = \arg \max_{\mathbf{s}} P(\mathbf{s})$ that maximizes the SRP over some set of candidate coordinate points. When training neural networks for DOA prediction, it is common to use the SRP [37, 26] or individual GCC-PHAT features [100] as input to the network.

In certain applications, it might also be necessary to simultaneously localize multiple events. This requires developing some heuristics for handling overlapping events in time and how to distinguish them. The sound event localization and detection (SELD) problem is an extended variant of the SSL problem, where the task is to detect, classify, localize and track multiple sound events over time, as shown in Figure 21. Since GCC-PHAT carries little information about spectral characteristics of a particular sound, it is common to combine GCC-PHAT with log-mel spectrograms or MFCC features as input to SELD models for improved detection and classification performance. When events other than speech are present, some studies have indicated that using log-mel spectrograms gives better classification performance than using MFCC [22, 16]. Some studies have also explored SELD using raw waveforms as input [61, 62].

In order to localize multiple sound events, a SELD model needs to be able to separate events and output a prediction for each event detected. In section 4.2 we introduced permutation invariant training (PIT), which is a common training technique for handling the permutations of different events. Given a maximum of K events belonging to one of C possible classes, the model uses CK separate outputs, where each output predicts the DOA for a single active event. Inactive events can either be handled using a separate binary classification output, or by modeling the DOA as an activity-coupled Cartesian DOA (ACCDOA) vector where the magnitude is either 0 or 1 depending on whether the event is active or not [118]. When more than one, but less than K events are active for a single class, it is beneficial to duplicate the target label of the active event, since it allows each output to be trained to predict the same output as if there was only a single active event [119]. This technique is known as auxiliary duplicating PIT (ADPIT).

In Paper VI, we experiment with using NGCC-PHAT features as input to a SELD model. In order to be able to train it to perform TDOA prediction on a dataset with multiple overlapping events, we incorporate ADPIT during the training phase. Experiments show that NGCC-PHAT features together with log-mel spectrograms provides significantly better performance compared to using regular GCC-PHAT features.

Localization using Ad-Hoc Arrays

For many SSL problems, we can use audio descriptors as input to a neural network and trained it to predict the sound source location. For example, Vera-Diaz et al. proposed to predict the sound source location as a function of the audio recordings using an end-to-end CNN model [132]. However, this only works as long as the microphones are stationary and the network can learn the mapping from signals to locations from the data. If the microphones can be moved into a different configuration, then such a model would not be able to learn, since the mapping depends on the configuration. Hence, a more general model for SSL needs to also use the microphone locations as input in this setting, as illustrated in Figure 22.

A model using both audio and microphone coordinates as input was proposed in [56], but the method assumes that the number of microphones in the array does not change. Still, in the event where a microphone is turned off or is removed from the array for some reason, we would ideally still want to be able to make predictions. Furthermore, the proposed CNN model is also not permutation invariant, which implies that the predictions will change if the microphones are re-ordered. We can therefore make the case that an SSL model ought to be able to handle a variable number of microphones during both training and inference, while also being permutation invariant, which is not the case for many deep learning approaches to SSL. From Section 4 we know that graph neural networks such as Transformers are both permutation invariant and support variable number of inputs, and

here we shall see how we can model the SSL problem using a graph structure.

Let $\mathbf{r}_m \in \mathbb{R}^3, m = 1, \dots, M$ denote the Cartesian coordinates of a distributed microphone array and consider the fully connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of microphones with vertices $\mathcal{V} = \{m\}_{m=1}^M$ and edges $\mathcal{E} = \mathcal{V} \times \mathcal{V}$. The TDOA features $\{\mathbf{R}_{ij}\}_{i,j=1}^M$ (for example GCC-PHAT) can be viewed as a set of pairwise *relations* on \mathcal{G} . Hence, we can use the relation network architecture from Section 4.3. But we also need to incorporate the microphone coordinates into the relations. Then we can aggregate over all relations in order to get a relation network that predicts the sound source coordinate. This idea was first proposed by Grinstein et al. [55], where the model structure was formulated as

$$\hat{\mathbf{s}} = \phi \left(\sum_{i \neq j} \psi(\mathbf{r}_i, \mathbf{r}_j, \mathbf{R}_{ij}) \right), \quad (68)$$

where ψ and ϕ are typically implemented using MLPs. The relation network architecture has several nice properties which makes it suitable for localization using distributed microphone arrays. Importantly, the number of input microphones is variable, since we can easily remove vertices from the graph without changing the network architecture. Furthermore, the output is invariant to the ordering of the microphones, which implies that if two microphones switch position, the prediction will not be affected.

In Paper VII, we try to make the idea of SSL from a graph structure more general. In particular, we design a model that in addition to standard SSL, can also handle the following variations to the problem:

- 1) A subset of the microphones have unknown location.
- 2) A subset of the signals are missing.
- 3) Combinations of 1) and 2).
- 4) The transmitted signal is known. This corresponds to the trilateration problem where we can measure the time of flight by exploiting the known signal.

The relation network structure on its own is not able to handle these scenarios, since if for example \mathbf{r}_i is known, but we don't know \mathbf{r}_j , we cannot evaluate the expression in Equation (68).

In order to design a model that can be used in all of these setups, we resort to the masked autoencoder method that we introduced in Section 4.6. Using this method allows us to model unknown coordinates using mask tokens. First, we define two new sets of graph

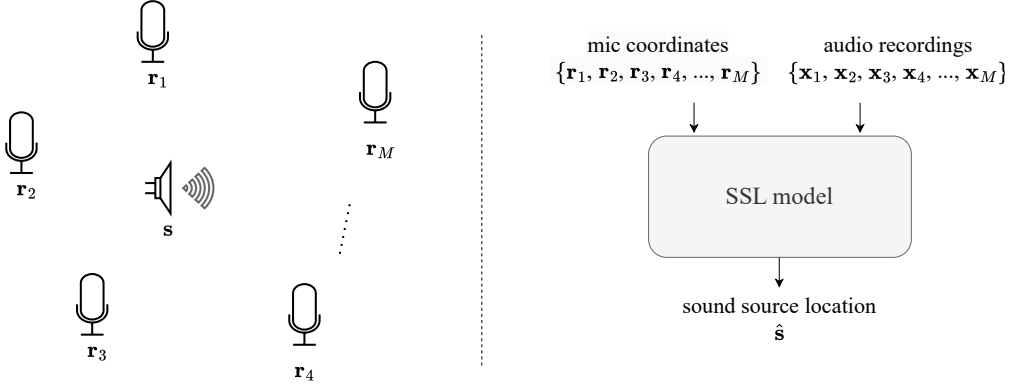


Figure 22: Overview of the SSL task considered in Paper VII. Given the the coordinates and audio recording from a distributed ad-hoc microphone array, the task is to localize a single sound source. Note that the microphone coordinates are needed as input to the system, since the microphone positions are not necessarily stationary. In this example, five microphones are used, but the system ought to handle an arbitrary number of microphones. The task can be extended to localize microphones with unknown coordinates as well.

vertices $\mathcal{S} = \{m : \mathbf{x}_m \text{ known}\}$ and $\mathcal{R} = \{m : \mathbf{r}_m \text{ known}\}$, that correspond to the sets of known recorded audio signals and coordinates respectively. We can then learn a function that takes the set of known signals and coordinates as input and predicts the sound source location as

$$\hat{\mathbf{s}} = f(\{\mathbf{x}_m\}_{m \in \mathcal{S}}, \{\mathbf{r}_m\}_{m \in \mathcal{R}}), \quad (69)$$

where f can be implemented using a Transformer architecture. The missing coordinates and recordings are added to the set of features inside the Transformer using mask tokens. And in the scenario where the transmitted audio signal is known, it can simply be added to the set of known input signals. In Paper VII, we also experiment with predicting multiple outputs, which it to predict the unknown microphone coordinates using the mask tokens.

Note that the set of microphone coordinates is essentially a 3D point cloud and we know that Transformers are well suited for dealing with this data type. However, compared to large point clouds from e.g. a LiDAR scan that contain thousands of points, the number of microphones is much smaller, so the computational complexity of the self-attention mechanism will not be a problem. When it comes to audio, we do not want to use spectrograms or other features that do not preserve the relative time delays between different signals, thus removing important spatial information. Therefore, we use raw waveforms, with the possibility of adding TDOA features separately.




In order for the Transformer to distinguish between coordinates and audio tokens, a modality encoding can be added to the inputs as in Equation (48). Note that we should not add

positional encoding to the inputs, since this would break invariance with respect to permutations of the microphone ordering. However, we still must preserve the information about which microphone coordinate corresponds to which audio recording. This can be done by doing message passing between the two types of tokens, which does not break permutation invariance.




Masking can also be used as a technique to evaluate a similar relation network as the one in Equation (68). This requires replacing unknown coordinates with mask tokens. In Paper VII, we do this using both GCC-PHAT and NGCC-PHAT as TDOA features. Notably, we find that the localization performance of the Transformer-based autoencoder model is increased when combining it with TDOA features from a relation network compared to when using only raw waveform as inputs. Although this result could be due to other factors, such as limited training data, it shows that GCC-PHAT and similar features have a strong inductive bias for localization tasks.

6 Summary of Contributions

Table 1: Summary of data modalities, scientific concepts and author contributions in the included papers.

Modalities:  - images,  - audio,  - point coordinates.

Author contributions:  : main contribution,  : significant contribution,  : minor contribution

Modalities				Author contributions			
Publication				Main Concepts	Ideas	Experiments	Writing
Paper I	✓			RvC, label diversity	●	●	●
Paper II			✓	Transformers, perm. symmetry	●	●	●
Paper III		✓		Transformers	⦿	⦿	⦿
Paper IV	✓	✓		quantization, model diversity	⦿	⦿	⦿
Paper V		✓		RvC, transl. symmetry	●	●	●
Paper VI		✓		RvC, perm. symmetry	●	●	●
Paper VII		✓	✓	Transformers, perm. symmetry	⦿	●	●

We have now covered the necessary background theory for the included papers. In this section, we briefly summarize the main contributions of each paper and the author contributions. A summary of the different data modalities, scientific concepts and author contributions in the included papers is shown in Table 1.

Although each paper is a standalone contribution, there is overlap both in terms of the tasks considered and the methods used, and the contribution of the thesis is therefore best understood if they are read in order. A conceptual overview of the how the papers are related is shown in Figure 23.

6.1 Paper Contributions

Paper I Deep Ordinal Regression with Label Diversity

Axel Berg, Magnus Oskarsson, Mark O'Connor

Proc. 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 2740-2747

Preprint: <https://arxiv.org/abs/2006.15864>

Code: <https://github.com/axeber01/dold>

Scientific contributions: This paper deals with applying the notion of label diversity to a wide range of problems in computer vision, where the data labels are both continuous and ordinal. By exploiting the possibility to combine different discretizations of continuous labels, and combinations of ordinal labels, we propose several strategies to induce label

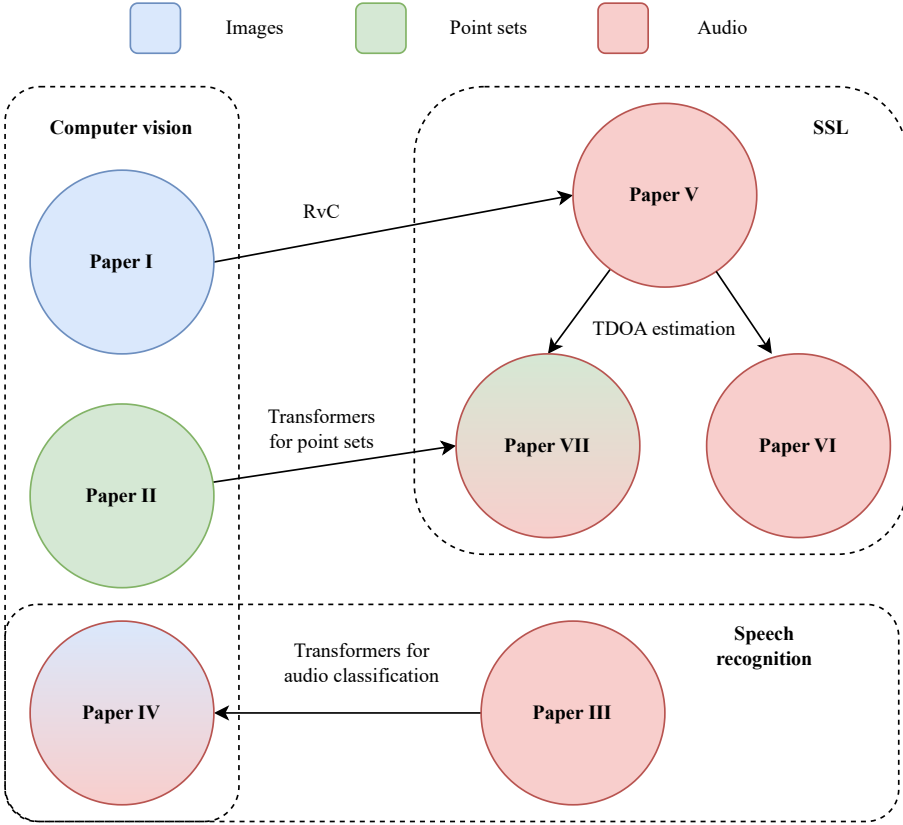


Figure 23: Conceptual overview of the included papers. Colors indicate different data modalities and arrows indicate how the papers relate to prior work.

diversity in practice. Furthermore, we provide a method for implementing label diversity with minimal computational overhead that can be used in conjunction with standard neural network feature extractors. The method is based on using multiple prediction heads, one for each label representation. During the training phase, each head learns to classify the data into a specific set of labels, and during inference the predictions are combined in an ensemble-like fashion. From the ambiguity decomposition in Equation (17), we know that the prediction error of the ensemble average is guaranteed to be smaller than the average individual prediction error. By thorough experiments we also show that label diversity can reduce the prediction errors compared to standard methods like regression and RvC.

Author contributions: AB came up with the idea after some help from the other authors, developed the theory and implemented the experiments. The paper was written by AB,

with input from the other authors.

Paper II Points to Patches: Enabling the Use of Self-Attention for 3D Shape Recognition

Axel Berg, Magnus Oskarsson, Mark O'Connor

Proc. 2022 26th International Conference on Pattern Recognition (ICPR), 2022, pp. 528-534

Preprint: <https://arxiv.org/abs/2204.03957>

Code: <https://github.com/axeber01/point-tnt>

Scientific contributions: We apply Transformers to point cloud processing, where the main problem is to reduce the quadratic complexity of the self-attention operator. We do this by using a hierarchical Transformer that applies self-attention locally and globally in a two-stage process. Experiments show that this not only reduces the computational footprint, but also makes the features of the Transformer better suited for downstream tasks such as classification and segmentation. Finally, we also show that the proposed method can be used to improve feature matching between point clouds, which is commonly used in SLAM and other applications.

Author contributions: AB came up with the idea after some discussions with the other authors. The implementation and all experiments were done by AB. The paper was written by AB, with input from the other authors.

Paper III Keyword Transformer: A Self-Attention Model for Keyword Spotting

Axel Berg, Mark O'Connor, Miguel Tairum Cruz

Proc. Interspeech 2021, pp. 4249-4253

Preprint: <https://arxiv.org/abs/2104.00769>

Code:

<https://github.com/ARM-software/keyword-transformer>

Scientific contributions: Inspired by the success of Transformer models for natural language processing and computer vision, we investigate the use of this architecture for audio classification. We propose the first Transformer-based keyword spotting method and achieves state-of-the-art results on common benchmarks. By ablation studies we highlight the benefit of applying temporal self-attention to the MFCC input features, and we show that the model learns to attend to the time slots that are most important for classifica-

tion. In addition, we measure latency on a mobile phone and show that Transformers are competitive in this regard as well.

Author contributions: MOC came up with the idea of applying Transformers to keyword spotting and AB implemented the model. AB implemented most of the experiments, except for knowledge distillation, which was done by MOC, and latency measurements, which was done by MTC. The paper was written jointly by the three authors.

Paper IV Towards Federated Learning with on-device Training and Communication in 8-bit Floating Point

Bokun Wang, Axel Berg, Durmus Alp Emre Acar, Chuteng Zhou
Submitted. A shorter version of this paper was presented at Fed-KDD: International Joint Workshop on Federated Learning for Data Mining and Graph Analytics, 2024.
Preprint: <https://arxiv.org/abs/2407.02610>

Scientific contributions: Training models in a federated setting comes with two major challenges, both of which we address in this paper: 1) limited hardware capabilities of the clients and 2) high communication costs due to server-client communication. This paper explores federated learning in 8 bit floating point, which is a training format expected to become widely adopted by the industry. We emulate low-precision hardware computation using quantization-aware training and show that this only has a small impact on model performance. Furthermore, we investigate the impact of quantized model weights in the communication step, and our experiments show that for a given model performance, this results in significant reduction of communicated data compared to using full 32-bit precision. We also provide the theoretical convergence rate and motivate the use of stochastic quantization. Finally, we introduce a server-side optimization method that is able to recover some of the performance loss induced by weight quantization. Experiments are done both using CNNs for image classification and the Transformer model from Paper III, which is trained in a realistic federated keyword spotting setup.

Author contributions: BW developed and implemented the main method, including FP8 QAT training, quantization of neural network layers, federated training setup and server-side optimization. BW did the experiments on the image classification task, and AB did the experiments for keyword spotting. AA and CZ developed the convergence analysis, with input from the other authors. The paper was written jointly by all authors.

Paper V Extending GCC-PHAT using Shift Equivariant Neural Networks

Axel Berg, Mark O'Connor, Kalle Åström, Magnus Oskarsson

Proc. Interspeech 2022, pp. 1791-1795

Preprint: <https://arxiv.org/abs/2208.04654>Code: <https://github.com/axeber01/ngcc>

Scientific contributions: Many methods for SSL depends on accurate TDOA estimation, yet the standard GCC-PHAT method is still commonly used for SSL. The main contribution of this paper is a simple yet very effective way to improve the TDOA estimation accuracy of GCC-PHAT. This is done by applying of trainable convolutional filters before and after the GCC-PHAT operation and we refer to this method as Neural GCC-PHAT (NGCC-PHAT). Due to the translation equivariance of convolutions, there are some guarantees of TDOA recovery in ideal conditions, even though the method is data driven. The filters can be trained end-to-end for TDOA estimation using RvC. Experiments on simulated data with a single sound source show improved performance compared to standard GCC-PHAT.

Author contributions: KÅ proposed the idea of using machine learning for TDOA estimation. The goal was initially to develop a method that replaces GCC-PHAT estimates in a localization pipeline. AB proposed exploiting the translation equivariance of CNNs to filter the signals independently before computing the GCC-PHAT. AB implemented the method, created the simulated dataset and performed model training and evaluation. AB wrote most of the paper with input from the other authors.

Paper VI Learning Multi-Target TDOA Features for Sound Event Localization and Detection

Axel Berg, Johanna Engman, Jens Gulin, Kalle Åström, Magnus Oskarsson

Proc. Detection and Classification of Acoustic Scenes and Events 2024 Workshop (DCASE2024), pp. 16-20

Preprint: <https://arxiv.org/abs/2408.17166>Code: <https://github.com/axeber01/ngcc-seld>

Scientific contributions: This paper extends the method from Paper V to multiple overlapping sound sources using permutation invariant training. We show that the learned NGCC-PHAT features can be used as input to an existing SSL method with improved performance compared to other commonly used input features. The method is evaluated

on a widely adopted benchmark as part of the DCASE challenge.

Author contributions: AB proposed the idea of using permutation-invariant training in combination with the NGCC-PHAT method from Paper V in order to extract better features as input for SELD models. The method implementation and evaluation was done by AB. The paper was written by AB and JG, with input from the other authors.

Paper VII **wav2pos: Sound Source Localization using Masked Autoencoders**

Axel Berg, Jens Gulin, Mark O'Connor, Chuteng Zhou, Kalle Åström, Magnus Oskarsson

Proc. 2024 14th International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-8

Preprint: <https://arxiv.org/abs/2408.15771>

Code: <https://github.com/axeber01/wav2pos>

Scientific contributions: In Paper VI we considered SSL with a fixed microphone array. Here, we instead consider another version of the SSL task where the number of microphones, or their coordinates, are not necessarily known at inference time and therefore needs to be used as input to the model, as in the case of classical multilateration. We propose an autoencoder based method that can handle a variety of such SSL tasks. Furthermore, it can exploit audio from a microphone at an unknown location or from the source itself. Another contribution from this paper is that we evaluate the TDOA estimates from NGCC-PHAT with a classical multilateration method, which extends the results from Paper V.

Author contributions: MOC proposed the idea of using a masked autoencoder that could learn to solve the localization problem with varying number of microphones. AB refined this idea and drew inspiration from other multimodal autoencoders with modality embeddings. AB proposed the idea of pair-wise positional encoding to preserve permutation equivariance.

The model implementation, data set generation and all experiments were done by AB. KÅ provided an implementation of the multilateration method and AB evaluated it using TDOA estimates from GCC-PHAT and NGCC-PHAT. The implementations for the other methods used for comparisons were retrieved online from the official code repositories and AB trained and evaluated them using the same setup as the proposed method. The paper was written by AB, JG and MO, with input from the other authors.

6.2 Conclusions and Outlook

The outcome of this thesis can be summarized by revisiting the research questions in Section 1.1. In Paper I, we addressed [RQ1] by proposing an new objective function that exploits label diversity. We found that this is a suitable objective for several problems in computer vision. Although RvC was used for TDOA estimation in Paper V, we did not experiment with the method of label diversity for this problem, which we leave for future work. It would also be interesting to apply label diversity to DOA prediction, which has a similar output domain as the pose estimation problem considered in Paper I.

[RQ2] and [RQ3] are dealt with in Papers II, III and IV, where we investigate efficient implementations of Transformer models for point cloud analysis and speech recognition, respectively. The combined findings of these studies indicate that Transformers can be trained to achieve high classification accuracy for both of these data modalities, while still being computationally efficient. Further efficiency improvements can be obtained by training in low-precision number formats, such as FP8.

Papers V, VI and VII all deal with various aspects of [RQ4] (machine learning for SSL). Notably, we show that a learning-based method can achieve good TDOA prediction performance and that the learned representations can be used as input to either a classical multilateration method, or to a neural network, for solving the SSL problem. Another important finding is that predicting the sound source location using only raw audio waveforms as input to a neural network is difficult, since the localization performance of the autoencoder model in Paper VII was significantly improved when TDOA features were used as input to the model. Hence, we conclude that using features with a strong inductive bias, such as TDOA features, yields better localization performance (at least when the dataset is not sufficiently large).

In terms of [RQ5], we have dealt with invariance to several types of transformations in the different methods: invariance with respect to input permutations (Paper II and VII) and output permutations (Paper VI), as well as invariance with respect to relative translations of signals (Paper V). For the SSL problem, an interesting future research direction would be to also consider input-output equivariance with respect to other transformations of the microphone coordinates, e.g. reflections, rotations and translations.

Other aspects of future work could involve combining different aspects of the methods in the thesis in order to construct more advanced system for machine perception. For example, by combining the contributions from speech recognition (Paper III), low-precision training (Paper IV) and SSL (Papers V-VII), it would be possible to develop a system that can perform sound event detection, localization speech recognition and localization, while running on a low-end device. One interesting application could be augmented reality, where these methods are also combined with methods for visual perception (as in Paper

I) or LiDAR data (as in Paper II). Combining different types data in order to construct multimodal AI systems has been a strong research trend in recent years and the capabilities of such systems are improving each year. We therefore anticipate that future work will further develop the findings of this thesis and incorporate them into larger scale AI systems for perception and localization.

References

- [1] Autograd mechanics. PyTorch 2.4 documentation. <https://pytorch.org/docs/stable/notes/autograd.html#complex-autograd-doc>. [Accessed 27-09-2024].
- [2] Federated Learning with Formal Differential Privacy Guarantees (blog post). <https://research.google/blog/federated-learning-with-formal-differential-privacy-guarantees/>. [Accessed 06-09-2024].
- [3] Ieee standard for floating-point arithmetic. Technical Report IEEE Std 754-2008, IEEE Computer Society, 2008.
- [4] Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016.
- [5] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.
- [6] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [7] A. Al-Qawlaq, D. John, et al. Kwt-tiny: Risc-v accelerated, embedded keyword spotting transformer. *arXiv preprint arXiv:2407.16026*, 2024.
- [8] J. B. Allen and D. A. Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950, 1979.
- [9] M. AskariHemmat, R. A. Hemmat, A. Hoffman, I. Lazarevich, E. Saboori, O. Mastrogiuseppe, S. Sah, Y. Savaria, and J.-P. David. Qreg: On regularization effects of quantization. *arXiv preprint arXiv:2206.12372*, 2022.
- [10] A. Baade, P. Peng, and D. Harwath. Mae-ast: Masked autoencoding audio spectrogram transformer. In *Interspeech 2022*, pages 2438–2442, 2022. doi: 10.21437/Interspeech.2022-10961.
- [11] P. Baldi and P. J. Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26, 2013.
- [12] I. Beaver. Is ai at human parity yet? a case study on speech recognition. *AI Magazine*, 43(4):386–389, 2022.

- [13] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.
- [14] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [15] A. Berg. *Applications of Diversity and the Self-Attention Mechanism in Neural Networks*. Licentiate thesis, Lund University, 2022.
- [16] A. Berg, J. Engman, J. Gulin, K. Åström, and M. Oskarsson. The LU System for DCASE 2024 Sound Event Localization and Detection Challenge. Technical report, DCASE2024 Challenge, June 2024.
- [17] M. S. Brandstein and H. F. Silverman. A robust method for speech signal time-delay estimation in reverberant rooms. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 375–378. IEEE, 1997.
- [18] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.
- [19] G. Brown. *Diversity in neural network ensembles*. PhD thesis, University of Birmingham, 2004.
- [20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020. URL <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>.
- [21] N. Burgess, J. Milanovic, N. Stephens, K. Monachopoulos, and D. Mansell. Bfloat16 processing for neural networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pages 88–91, 2019. doi: 10.1109/ARITH.2019.00022.
- [22] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *2015 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2015.
- [23] A. Chaman and I. Dokmanić. Truly shift-invariant convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3773–3783, 2021.

- [24] S. E. Chazan, H. Hammer, G. Hazan, J. Goldberger, and S. Gannot. Multi-microphone speaker separation based on deep doa estimation. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019.
- [25] M. Chghaf, S. Rodriguez, and A. E. Ouardi. Camera, lidar and multi-modal slam systems for autonomous ground vehicles: a survey. *Journal of Intelligent & Robotic Systems*, 105(1):2, 2022.
- [26] J.-H. Cho and J.-H. Chang. SR-SRP: Super-Resolution based SRP-PHAT for Sound Source Localization and Tracking. In *Proc. INTERSPEECH 2023*, pages 3769–3773, 2023. doi: 10.21437/Interspeech.2023-2369.
- [27] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha. Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices. *Proc. Interspeech 2019*, pages 3372–3376, 2019.
- [28] M. G. Christensen, P. Stoica, A. Jakobsson, and S. H. Jensen. Multi-pitch estimation. *Signal Processing*, 88(4):972–983, 2008.
- [29] X. Chu, Z. Tian, B. Zhang, X. Wang, and C. Shen. Conditional positional encodings for vision transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=3KWnuT-R1bh>.
- [30] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.
- [31] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [32] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.
- [33] J. G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A*, 2(7):1160–1169, 1985.
- [34] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929*, 2018.
- [35] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics, 2019.

- [36] R. Diaz and A. Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4738–4747, 2019.
- [37] D. Diaz-Guerra, A. Miguel, and J. R. Beltran. Robust Sound Source Tracking using SRP-PHAT and 3D Convolutional Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:300–311, 2020.
- [38] D. Diaz-Guerra, A. Miguel, and J. R. Beltran. gpurir: A python library for room impulse response simulation with gpu acceleration. *Multimedia Tools and Applications*, 80(4):5653–5671, 2021.
- [39] J. Díez-González, R. Álvarez, L. Sánchez-González, L. Fernández-Robles, H. Pérez, and M. Castejón-Limas. 3d tdoa problem solution with four receiving nodes. *Sensors*, 19(13):2892, 2019.
- [40] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.
- [41] S. Dovrat, E. Nachmani, and L. Wolf. Many-speakers single channel speech separation with optimal permutation training. In *Interspeech 2021*, pages 3890–3894, 2021.
- [42] I. L. Espejo, Z.-H. Tan, and J. Jensen. An experimental study on light speech features for small-footprint keyword spotting. In *IberSPEECH 2022*, 2022.
- [43] C. Evers, A. H. Moore, and P. A. Naylor. Acoustic simultaneous localization and mapping (a-slam) of a moving microphone array and its surrounding speakers. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE, 2016.
- [44] L. Fan, Y. Yang, Y. Mao, F. Wang, Y. Chen, N. Wang, and Z. Zhang. Once detected, never lost: Surpassing human performance in offline lidar based 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19820–19829, 2023.
- [45] J. Feldmaier, M. Rothbucher, and K. Diepold. Sound localization and separation for teleconferencing systems. Technical report, Lehrstuhl für Datenverarbeitung, 2014.
- [46] M. A. Figueiredo. Adaptive sparseness using jeffreys prior. In *NIPS*, pages 697–704, 2001.

- [47] E. Frank and M. Hall. A simple approach to ordinal classification. In *European conference on machine learning*, pages 145–156. Springer, 2001.
- [48] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, pages 148–156, 1996.
- [49] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.
- [50] X. Geng, H. Liu, L. Lee, D. Schuurmans, S. Levine, and P. Abbeel. Multimodal Masked Autoencoders Learn Transferable Representations. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022.
- [51] Y. Gong, Y.-A. Chung, and J. Glass. Ast: Audio spectrogram transformer. In *Interspeech 2021*, pages 571–575, 2021. doi: 10.21437/Interspeech.2021-698.
- [52] Y. Gong, C.-I. Lai, Y.-A. Chung, and J. Glass. Ssast: Self-supervised audio spectrogram transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10699–10709, 2022.
- [53] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [54] H. Gray and W. H. Lewis. *Anatomy of the human body*. Philadelphia, Lea and Febiger, 1918. URL <https://www.biodiversitylibrary.org/item/60234>.
- [55] E. Grinstein, M. Brookes, and P. A. Naylor. Graph Neural Networks for Sound Source Localization on Distributed Microphone Networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [56] E. Grinstein, V. W. Neo, and P. A. Naylor. Dual Input Neural Networks for Positional Sound Source Localization. *EURASIP Journal on Audio, Speech, and Music Processing*, 2023(1):32, 2023.
- [57] P.-A. Grumiaux, S. Kitić, L. Girin, and A. Guérin. A Survey of Sound Source Localization with Deep Learning mMethods. *The Journal of the Acoustical Society of America*, 152(1):107–151, 2022.
- [58] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

- [59] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [60] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are Scalable Vision Learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
- [61] Y. He and A. Markham. Sounddoa: Learn sound source direction of arrival and semantics from sound raw waveforms. In *Interspeech*, pages 2408–2412, 2022.
- [62] Y. He and A. Markham. SoundSynp: Sound Source Detection from Raw Waveforms with Multi-Scale Synperiodic Filterbanks. In *International Conference on Artificial Intelligence and Statistics*, pages 9010–9023. PMLR, 2023.
- [63] Y. Hoshen, R. J. Weiss, and K. W. Wilson. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4624–4628. IEEE, 2015.
- [64] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.
- [65] Z. Jelčicová and M. Verhelst. Delta keyword transformer: Bringing transformers to the edge through dynamically pruned multi-head self-attention. In *tinyML Research Symposium’22*, 2022.
- [66] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.
- [67] T. Kathiresan, D. Maurer, and V. Dellwo. Highly spectrally undersampled vowels can be classified by machines without supervision. *The Journal of the Acoustical Society of America*, 146(1):EL1–EL7, 2019.
- [68] O. S. Kayhan and J. C. v. Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020.
- [69] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1412.6980>.

- [70] C. Knapp and G. Carter. The Generalized Correlation Method for Estimation of Time Delay. *IEEE transactions on acoustics, speech, and signal processing*, 24(4):320–327, 1976.
- [71] M. Kreković, I. Dokmanić, and M. Vetterli. Echoslam: Simultaneous localization and mapping with acoustic echoes. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11–15. Ieee, 2016.
- [72] A. Krogh, J. Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238, 1995.
- [73] K. Kumatani, J. McDonough, and B. Raj. Microphone array processing for distant speech recognition: From close-talking microphones to far-field sensors. *IEEE Signal Processing Magazine*, 29(6):127–140, 2012.
- [74] M. Larsson. *Localization using Distance Geometry: Minimal Solvers and Robust Methods for Sensor Network Self-Calibration*. PhD thesis, Lund University, 2022.
- [75] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.
- [76] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6341–6345. IEEE, 2019.
- [77] L. Li and H.-T. Lin. Ordinal regression by extended binary classification. In *Advances in neural information processing systems*, pages 865–872, 2007.
- [78] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.
- [79] H. Lin and S. Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *Advances in neural information processing systems*, 31, 2018.
- [80] Y. Liu. *Negative correlation learning and evolutionary neural network ensembles*. PhD thesis, University College, The University of New South Wales, 1998.
- [81] I. López-Espejo, Z.-H. Tan, J. H. Hansen, and J. Jensen. Deep spoken keyword spotting: An overview. *IEEE Access*, 10:4169–4199, 2021.
- [82] I. López-Espejo, Z.-H. Tan, and J. Jensen. Exploring filterbank learning for keyword spotting. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 331–335. IEEE, 2021.

- [83] I. López-Espejo, R. C. Shekar, Z.-H. Tan, J. Jensen, and J. H. Hansen. Filterbank learning for noise-robust small-footprint keyword spotting. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [84] J. Martinsson and M. Sandsten. Dmel: The differentiable log-mel spectrogram as a trainable layer in neural networks. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5005–5009. IEEE, 2024.
- [85] T. McKenzie, L. McCormack, and C. Hold. Dataset of spatial room impulse responses in a variable acoustics room for six degrees-of-freedom rendering and analysis. *arXiv preprint arXiv:2111.11882*, 2021.
- [86] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [87] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.
- [88] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.
- [89] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.
- [90] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.
- [91] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*.
- [92] A. M. Noll. Cepstrum pitch determination. *The journal of the acoustical society of America*, 41(2):293–309, 1967.
- [93] A. V. Oppenheim and R. W. Schaffer. From frequency to quefrency: A history of the cepstrum. *IEEE signal processing Magazine*, 21(5):95–106, 2004.

- [94] D. Palaz, M. Magimai-Doss, and R. Collobert. Analysis of cnn-based speech recognition system using raw speech as input. In *Interspeech 2015*, pages 11–15, 2015. doi: 10.21437/Interspeech.2015-3.
- [95] R. Parhizkar, I. Dokmanić, and M. Vetterli. Single-channel indoor microphone localization. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1434–1438. IEEE, 2014.
- [96] Y. Park, K. Budhathoki, L. Chen, J. M. Kübler, J. Huang, M. Kleindessner, J. Huan, V. Cevher, Y. Wang, and G. Karypis. Inference optimization of foundation models on ai accelerators. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6605–6615, 2024.
- [97] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In *How we learn; How we remember: Toward an understanding of brain and neural systems: Selected papers of Leon N Cooper*, pages 342–358. World Scientific, 1995.
- [98] D. Peter, W. Roth, and F. Pernkopf. End-to-end keyword spotting using neural architecture search and quantization. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3423–3427. IEEE, 2022.
- [99] C. J. Plack. *The sense of hearing*. Routledge, 2018.
- [100] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen. Overview and evaluation of sound event localization and detection in dcase 2019. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:684–698, 2020.
- [101] M. Popel, M. Tomkova, J. Tomek, Ł. Kaiser, J. Uszkoreit, O. Bojar, and Z. Žabokrtský. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature communications*, 11(1):1–15, 2020.
- [102] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [103] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.
- [104] M. Ravanelli and Y. Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.

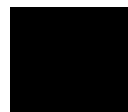
- [105] A. Riviello and J.-P. David. Binary speech features for keyword spotting tasks. In *INTERSPEECH*, pages 3460–3464, 2019.
- [106] I. R. Roman, C. Ick, S. Ding, A. S. Roman, B. McFee, and J. P. Bello. Spatial scaper: a library to simulate and augment soundscapes for sound event localization and detection in realistic rooms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024.
- [107] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [108] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.
- [109] F. Rumsey. *Digital Audio Recording Formats and Editing Principles*, pages 703–729. Springer New York, New York, NY, 2008. ISBN 978-0-387-30441-0. doi: 10.1007/978-0-387-30441-0_36. URL https://doi.org/10.1007/978-0-387-30441-0_36.
- [110] W. Sabine. Reverberation. *The American Architect*, 1900.
- [111] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, and B. Ramabhadran. Learning filter banks within a deep neural network framework. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 297–302. IEEE, 2013.
- [112] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals. Learning the speech front-end with raw waveform cldnns. In *Interspeech 2015*, pages 1–5, 2015. doi: 10.21437/Interspeech.2015-1.
- [113] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.
- [114] A. Saxena and A. Y. Ng. Learning sound location from a single microphone. In *2009 IEEE International Conference on Robotics and Automation*, pages 1737–1742. IEEE, 2009.
- [115] R. Scheibler, E. Bezzam, and I. Dokmanić. Pyroomacoustics: A Python Package for Audio Room Simulation and Array Processing Algorithms. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 351–355. IEEE, 2018.
- [116] H. Seki, K. Yamamoto, and S. Nakagawa. A deep neural network integrated with filterbank learning for speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5480–5484. IEEE, 2017.

- [117] C. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. doi: 10.1109/JRPROC.1949.232969.
- [118] K. Shimada, Y. Koyama, N. Takahashi, S. Takahashi, and Y. Mitsufuji. Accdoa: Activity-coupled cartesian direction of arrival representation for sound event localization and detection. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 915–919. IEEE, 2021.
- [119] K. Shimada, Y. Koyama, S. Takahashi, N. Takahashi, E. Tsunoo, and Y. Mitsufuji. Multi-ACCDOA: Localizing and detecting overlapping sounds from the same class with auxiliary duplicating permutation invariant training. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 316–320. IEEE, 2022.
- [120] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [121] S. S. Stevens, J. Volkman, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3): 185–190, 1937.
- [122] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- [123] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- [124] H. Tachibana. Towards listening to 10 people simultaneously: An efficient permutation invariant training of audio source separation using sinkhorn’s algorithm. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 491–495. IEEE, 2021.
- [125] N. Takahashi, S. Parthasaarathy, N. Goswami, and Y. Mitsufuji. Recursive speech separation for unknown number of speakers. In *Interspeech 2019*, pages 1348–1352, 2019. doi: 10.21437/Interspeech.2019-1550.
- [126] H. Theil. Linear algebra and matrix methods in econometrics. *Handbook of econometrics*, 1983.
- [127] L. Torgo and J. Gama. Regression using classification algorithms. *Intelligent Data Analysis*, 1(4):275–292, 1997.

- [128] W. Toussaint, A. Mathur, A. Y. Ding, and F. Kawsar. Characterising the role of pre-processing parameters in audio-based embedded machine learning. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 439–445, 2021.
- [129] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.
- [130] M. van Baalen, A. Kuzmin, S. S. Nair, Y. Ren, E. Mahurin, C. Patel, S. Subramanian, S. Lee, M. Nagel, J. Soriaga, et al. Fp8 versus int8 for efficient deep learning inference. *arXiv preprint arXiv:2303.17951*, 2023.
- [131] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. *Advances in neural information processing systems*, 30, 2017.
- [132] J. M. Vera-Diaz, D. Pizarro, and J. Macias-Guarasa. Towards End-to-End Acoustic Localization using Deep Learning: From Audio Signals to Source Position Coordinates. *Sensors*, 18(10):3418, 2018.
- [133] E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, and M. A. Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pages 6487–6494. PMLR, 2019.
- [134] E. Wagstaff, F. B. Fuchs, M. Engelcke, M. A. Osborne, and I. Posner. Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151): 1–56, 2022.
- [135] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.
- [136] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.
- [137] J. H. Winters, J. Salz, and R. D. Gitlin. The impact of antenna diversity on the capacity of wireless communication systems. *IEEE transactions on Communications*, 42(234):1740–1751, 1994.
- [138] H. Xi, C. Li, J. Chen, and J. Zhu. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168, 2023.

- [139] P. Xu, X. Zhu, and D. A. Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12113–12132, 2023.
- [140] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*, pages 10767–10777. PMLR, 2020.
- [141] D. Yu, X. Chang, and Y. Qian. Recognizing multi-talker speech with permutation invariant training. In *Interspeech 2017*, pages 2456–2460, 2017. doi: 10.21437/Interspeech.2017-305.
- [142] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245. IEEE, 2017.
- [143] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3394–3404, 2017.
- [144] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.
- [145] Y. Zhang, N. Suda, L. Lai, and V. Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.
- [146] Y. Zhu, M. Mattina, and P. Whatmough. Mobile machine learning hardware at arm: A systems-on-chip (soc) perspective. *SysML 2018*.

Scientific Publications



Deep Ordinal Regression with Label Diversity

AXEL BERG^{1,2}, MAGNUS OSKARSSON², MARK O’CONNOR¹

¹Arm Research, ²Centre for Mathematical Sciences, Lund University

Abstract: Regression via classification (RvC) is a common method used for regression problems in deep learning, where the target variable belongs to a set of continuous values. By discretizing the target into a set of non-overlapping classes, it has been shown that training a classifier can improve neural network accuracy compared to using a standard regression approach. However, it is not clear how the set of discrete classes should be chosen and how it affects the overall solution. In this work, we propose that using several discrete data representations simultaneously can improve neural network learning compared to a single representation. Our approach is end-to-end differentiable and can be added as a simple extension to conventional learning methods, such as deep neural networks. We test our method on three challenging tasks and show that our method reduces the prediction error compared to a baseline RvC approach while maintaining a similar model complexity.

1 Introduction

Choosing the right objective function is a crucial part of successfully training an accurate and generalized regression model, for example a deep neural network. Among the standard objective functions are the mean squared error (MSE, or L^2 loss), the mean absolute error (MAE, or L^1 -loss) and hybrid variants such as the Huber loss. Much attention has also been given to deriving problem-specific objective functions that incorporate certain aspects of the target variable, such as modularity and norm constraints for geometric regression. It is also possible to treat a continuous dependent variable as belonging to a finite number of discrete classes, although this necessarily comes at the expense of introducing a discretization error. Such approaches are known as regression via classification (RvC) and they are frequently used in tasks where a regression loss would at first seem more natural [31, 30, 33, 29, 1, 27].

Ordinal regression techniques can be applied to classification problems where the dependent variable exhibits a relative ordering. Techniques for ordinal regression can be applied to RvC problems in order to preserve the ordinal structure of the labels, and recent work has shown that this method can be used to improve the accuracy in several regression problems, such as age estimation [24] [3], image ranking and depth estimation [9].

One problem with the RvC approach is the ambiguity in how the discrete classes should be created from the distribution of the dependent variable. The standard approach is to create bins of equal width covering the target output range. For skewed distributions one can

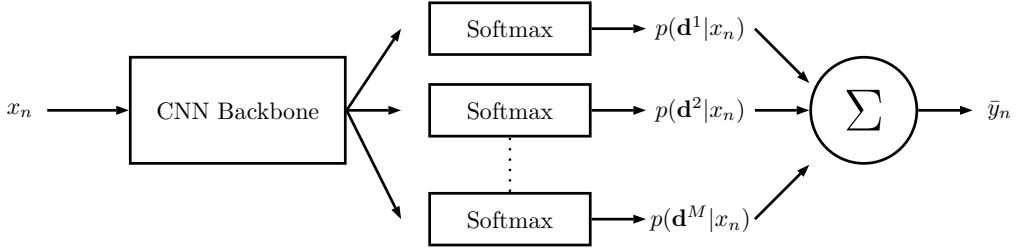


Figure 1: An illustration of the neural network architecture with multiple output heads. At inference time the expected values of the different distributions are combined using an ensemble average.

also apply the method of equal frequency, where the bins are created from the cumulative distribution function of the target, such that each bin contains the same number of training examples. Regardless of the method, the number of bins must be selected and optimized for the given task, which raises the question of what the optimal number of bins is for a given problem. If the bin-width is too small, this can result in few training examples in each class, but if it is too large, the discretization error can become a limiting factor.

The ambiguity in how to bin a continuous variable leads to a diverse range of possibilities in how to represent the target values — a fact that can be exploited. From previous research it is well known that a diverse ensemble of individual predictors can be combined in order to reduce the overall prediction error. Such approaches often involves training multiple regressors where the diversity is ensured by either data augmentation or model selection. This leads to increased overhead at both training and inference time [26]. However, with the use of label diversity and deep neural networks, one can create a multi-output predictor that enforces diversity without extra computational complexity.

Contribution: In this paper, we show that a collection of different binning variants of the target values can be used to improve prediction accuracy without increasing the computational complexity compared to a standard classifier. We do so by training a deep multi-output convolutional neural network (CNN) to classify training examples in multiple overlapping bins simultaneously. By doing so, we can effectively take into account the ordinal structure of the regression problem, while also making use of the diversity of the different possible representations of the target variable. We demonstrate our method on a number of different tasks and show competitive results compared to current state-of-the-art methods.

2 Related Work

2.1 Methods

Ordinal Regression

Ordinal regression, or ranking learning, is used for problems where the target variable exhibits a relative ordering on an arbitrary scale, e.g. categories such as "bad", "good" and "very good". When performing vanilla RvC, the ordinal information contained in the target values is lost, but this can be resolved by using ordinal regression techniques. A common variant is to use extended binary classification, where a single multi-class classification problem is reduced to a set of several binary classification problems [20]. With the advance of deep learning in recent years, ordinal regression has been used successfully for several tasks, including monocular depth estimation [12], age estimation [24], head pose estimation [17], medical diagnosis [21] and historical image dating [22].

Ensemble Learning

The fact that a set of individual regressors or classifiers can be combined into an ensemble in order to reduce the overall prediction accuracy of a model is frequently exploited in machine learning research [26]. A key notion in ensemble learning is the bias-variance-covariance decomposition, which says that in order for an ensemble to reduce the prediction error, there has to be some variance in the predictions of the ensemble members. Therefore, the aim should be to create ensembles that consist of accurate but diverse predictors.

Diversity can be created in several ways, most commonly through methods like bagging [5] and boosting [11], which rely on different forms of data diversity. However, using such method comes at the cost of extra model complexity and training time. Recently, Zhang et al. [35] proposed a framework for training an ensemble of networks using negative correlation learning, where the high level features are learned via parameter sharing. This reduces the overhead, while keeping the benefits of a diverse ensemble. In general, a multi-output neural network can be used to form an ensemble, which we exploit in our proposed method.

2.2 Relevant Applications

Age Estimation

Deep learning methods have been used successfully for age estimation, where the task consists of predicting the age of a person given a single RGB image of the person's face. Depending on the implementation, a person's age can be considered either as a continuous variable on the positive real numbers, or as a class belonging to a set of discrete positive integers. Rothe et al. [27] first highlighted the use of end-to-end training of CNNs for age estimation from a single image. The authors noted that classification yielded better results than direct regression and since then, several new methods using ordinal regression techniques have been published.

Agustsson et al. [2] performed end-to-end piecewise linear regression by assigning each regressor to an anchor point. Others have observed that it is easier for a human to distinguish differences in age between two persons, rather than their absolute age and used this as a design principle for ordinal regression [36, 24]. Alternative methods have focused on various soft encodings of the age over classes, where the elements of the probability vector are proportional to the distance from the true class [13, 34, 9]. In this way both the ordinal and metric information can be effectively encoded in the labels. Furthermore, it has also been shown that forcing the output of the classifier to be rank consistent over ages can improve the overall accuracy [3, 7].

Head Pose Estimation

Head pose estimation is the task of predicting the pose of a human head with three degrees of freedom, given an image and possibly depth information [23]. There are several ways to represent the pose, including three rotation angles (pitch, yaw and roll) with respect to a set of principal axes, a 3×3 rotation matrix or a single quaternion. During the past years, several CNN-based head pose estimators trained on specific loss functions have been proposed. Chang et al. [8] combined direct head pose regression with facial landmark detection and used it for facial alignment. Ruiz et al. [28] used a multi-loss CNN and showed that using a balanced hybrid variant of regression and classification yielded improvements over previous methods. Ordinal methods have also been used, such as in [17], where a combined ranking and MSE regression loss is used in conjunction with a quaternion representation of the head pose. The results indicated that training the CNN to regress the three angles while simultaneously solving several binary ranking problems improved the prediction accuracy compared to a standard regression or classification baseline.

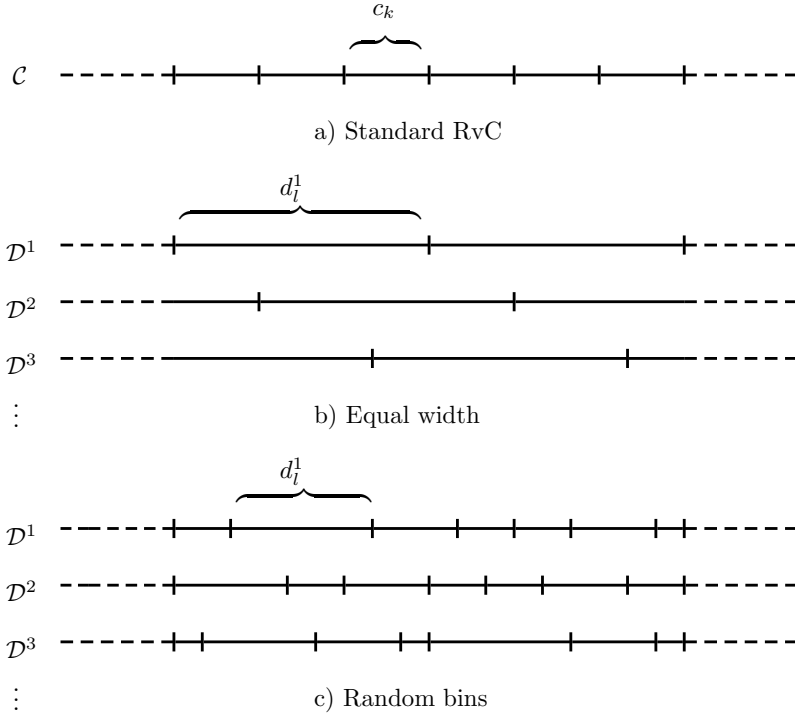


Figure 2: Examples of how the sets of class intervals \mathcal{D}^m can be constructed using the different methods.

Historical Image Dating

Palermo et al. [25] first introduced the task of automatically estimating the historical time in which a color photograph was taken using machine learning techniques. The authors noted that there are several features of the color imaging process that are typical to the era in which the images were taken, such as hue, saturation and color histogram. They then used a support vector machine to classify the images into different decades and showed that this method vastly outperformed untrained humans in terms of classification accuracy. Ginosar et al. [14] used American high school yearbooks to train a deep neural network for the same task, but in this case the extracted features were also dependent on the image content, e.g. facial attributes and hairstyle. Recently, ordinal regression techniques have also been applied successfully to the task of image dating [22, 4].

3 Proposed Method

3.1 Label Diversity by Overlapping Bins

Let $x_n \in \mathbb{R}^p$ denote the n^{th} input (independent variable) and let $t_n \in \mathbb{R}$ be the corresponding target value (dependent value) for $n = 1, \dots, N$. Now let $\mathcal{C} = \{c_k\}_{k=1}^K$ be a set of non-overlapping intervals on the real line \mathbb{R} , as shown in Figure 2a, such that $\cup_{k=1}^K c_k$ covers the samples t_n for all n . The standard RvC approach would now be to map each target value t_n to a unique class c_k and train a classifier to predict the posterior probability over classes $p(c_k|x_n)$.

In order to create label diversity, we instead introduce M new sets of class intervals $\mathcal{D}^m = \{d_l^m\}_{l=1}^{L_m}$ for $m = 1, \dots, M$, such that $\cup_{l=1}^{L_m} d_l^m = \cup_{k=1}^K c_k$, $\forall m$. By doing so, we have created several new discretizations that all cover the support of the target value, but in different ways. Here we do not provide an answer to exactly how these discretizations ought to be chosen, since this in itself is an optimization problem where the solution most likely depends on the problem domain, but instead we focus on the two following possibilities, where we assume that $L_m = L$, $\forall m$, such that each discretization contains equally many classes:

1. Assuming that we do not want to increase the complexity of the training algorithm compared to the standard RvC approach, we fix the total number of classes such that $ML = K$. Then we create M discretizations, each containing L equally wide bins, such that the overlap between d_l^m and d_l^{m+1} is fixed. An illustration of this approach is shown in Figure 2b. We refer to this method as "equal width".
2. In order to maximize diversity between different discretizations, for each $m = 1, \dots, M$, we randomly sample $L < K$ classes (with replacement) from \mathcal{C} and let these classes be the centers of the new bins in \mathcal{D}^m , such that target values that do not belong to any of the chosen classes are assigned to the nearest neighbor in the sample. An illustration of this approach is shown in Figure 2c. We refer to this method as "randomized bins".

If the target is multi-dimensional, the same methods can be applied by creating classes for each dimension individually.

3.2 Backpropagation

The proposed methods can be used together with a neural network by replacing the last fully connected layer and activation with M fully connected layers of size L in parallel,

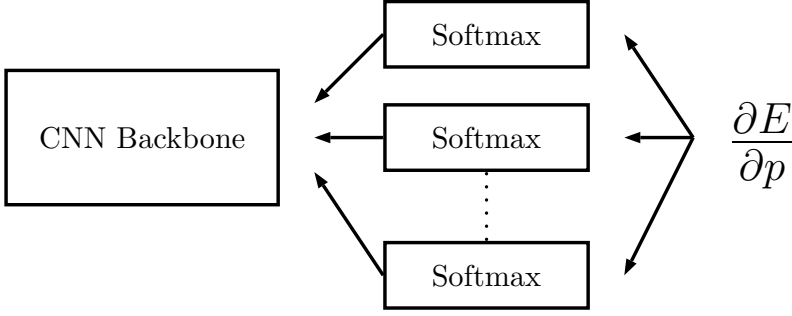


Figure 3: During training, the loss is backpropagated through the different softmax heads to the previous layers.

where each layer has its own softmax activation. The network is trained by minimizing the sum of the negative cross-entropies between the individual classifier and the targets over each mini-batch of size N_b . The loss function then becomes

$$E = - \sum_{n=1}^{N_b} \sum_{m=1}^M \sum_{l=1}^{L_m} q_n(d_l^m) \log p(d_l^m | x_n), \quad (1)$$

where $q_n(d_l^m)$ is a one-hot encoding of the binned target value, such that the only non-zero element is the one where the bin overlaps the true target:

$$q_n(d_l^m) = \begin{cases} 1, & t_n \in d_l^m \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The predictions $p(d_l^m | x_n)$ are computed using the M individual softmax heads of size L , and the loss is then back-propagated through the network by differentiating with respect to the predictions as shown in Figure 3.

In order to see how the loss function incorporates the ordinal relationship between the targets, we can again consider Figure 2. In order to make a correct prediction, each softmax head must output a high probability for the correct class in each discretization. If the output probabilities are correct for only a subset of the M different discretizations (which implies that the prediction is slightly off) then this will be penalized by an increased loss.

3.3 Inference

At inference time, the output posterior should be evaluated and converted to a point estimate of the target value by a hard decision. For standard RvC methods, this is typically done by either taking the expected value of the output distribution over classes or using the maximum-a-posteriori estimator. Therefore, we propose two similar methods to perform inference.

For problems where the target value t_n belongs to the real line, we do this by computing the expected value over each estimated posterior distribution as

$$\hat{y}_{m,n} = \sum_{l=1}^{L_m} w_l^m p(d_l^m | x_n), \quad (3)$$

where w_l^m denotes the mean value of the bin d_l^m . This gives us M different estimates of the target, which are then combined using a weighted average. Here we only consider uniform weighting of the individual estimates, i.e.

$$\bar{y}_n = \frac{1}{M} \sum_{m=1}^M \hat{y}_{m,n}. \quad (4)$$

This is a form of ensemble average, and we note that we can decompose an individual error made by the ensemble using the ambiguity decomposition [6] as

$$(\bar{y}_n - t_n)^2 = \frac{1}{M} \sum_{m=1}^M (\hat{y}_{n,m} - t_n)^2 - \frac{1}{M} \sum_{m=1}^M (\hat{y}_{n,m} - \bar{y}_n)^2, \quad (5)$$

which shows that the ensemble error is less than the average error of the individual estimates if there is enough variance within the ensemble. In this case, the non-zero variance is guaranteed by the different weights associated with each expected value in equation (3).

On the other hand, if the target belongs to a set of ordinal classes where we cannot define a useful distance metric, it is more suitable to use the MAP estimate. We do this by marginalizing over d_l^m in order to estimate the average posterior over the original classes c_k as

$$p(c_k | x_n) = \frac{1}{M} \sum_{m=1}^M \sum_{l=1}^{L_m} p(c_k | d_l^m, x_n) p(d_l^m | x_n), \quad (6)$$

where we assume that the conditional probability $p(c_k | d_l^m, x_n)$ is independent of the input x_n , i.e.

$$p(c_k | d_l^m, x_n) = \frac{||d_l^m \cap c_k||}{||d_l^m||}. \quad (7)$$

Then we compute the MAP estimate as

$$k^* = \arg \max_k p(c_k | x_n). \quad (8)$$

This shows that our method can be used both for true regression problems where we can measure distances between target values and for classification problems where an ordering of targets exists, but without a well-defined distance. In section V we show experiments for tasks of both the first and second kind.

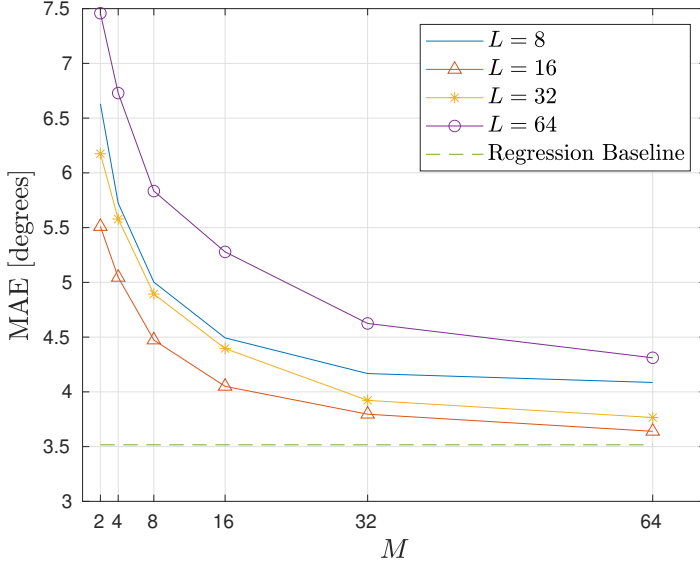


Figure 4: The MAE on the rotated MNIST datasets using the randomized bins method for different combinations of L and M

4 An Illustrative Example

In order to demonstrate the ensemble-like effect of our method, we train a shallow CNN on the task of predicting the rotation angle of digits from the MNIST dataset of handwritten digits [19]. The dataset consists of 5,000 training images and 5,000 test images, where a digit is rotated by an integer drawn uniformly in the interval $t_n \in [-45^\circ, 45^\circ]$. We implemented label diversity using the randomized bins method for all combinations of $M \in [2, 4, 16, 8, 32, 64]$ and $L \in [8, 16, 32, 64]$. We then trained a five-layer CNN, with M softmax heads, each containing L output units, to predict the one-hot encoded labels of the rotation angles. At inference time, we used equation (4) for prediction and evaluated the MAE on the test set. For comparison, we also trained a regression baseline by using the same shallow CNN with the MSE loss, but where the softmax heads were replaced by a single output unit with a linear activation. The training process was repeated for 10 random initializations of the network and the MAE was averaged over the 10 different trials.

The results of the experiment is presented in Figure 4. Here we clearly see the ensemble-like effect of our method, where the error decreases as the number of softmax heads M is increased. This is expected from the error decomposition in equation (5). Additionally, we note that increasing the number of output units L , which leads to a decreased discretization error caused by the binning of the target, does not necessarily imply a decrease in prediction error. In this experiment, $L = 16$ yielded the smallest MAE for all values of M . This agrees

with previous findings, namely that too few bins leads to a large discretization error, but too many can lead to poor convergence [9, 12]. In general, the optimal number of bins depends on the specific task and finding it therefore requires an extensive parameter search.

5 Experiments






GT: 30	GT: 28	GT: 55	GT: 37	GT: 33
				
Predictions:	Predictions:	Predictions:	Predictions:	Predictions:
$\hat{y}_1 = 30.17$	$\hat{y}_1 = 28.04$	$\hat{y}_1 = 54.68$	$\hat{y}_1 = 30.37$	$\hat{y}_1 = 43.96$
$\hat{y}_2 = 30.38$	$\hat{y}_2 = 28.5$	$\hat{y}_2 = 54.61$	$\hat{y}_2 = 30.79$	$\hat{y}_2 = 44.06$
$\hat{y}_3 = 29.81$	$\hat{y}_3 = 28.54$	$\hat{y}_3 = 54.82$	$\hat{y}_3 = 30.16$	$\hat{y}_3 = 43.83$
$\hat{y}_4 = 29.88$	$\hat{y}_4 = 27.33$	$\hat{y}_4 = 54.87$	$\hat{y}_4 = 30.14$	$\hat{y}_4 = 43.66$
$\hat{y}_5 = 29.8$	$\hat{y}_5 = 27.77$	$\hat{y}_5 = 55.17$	$\hat{y}_5 = 30.3$	$\hat{y}_5 = 43.64$
Average:	Average:	Average:	Average:	Average:
$\bar{y} = 30.01$	$\bar{y} = 28.03$	$\bar{y} = 54.83$	$\bar{y} = 30.35$	$\bar{y} = 43.83$

Figure 5: A subset of the images in UTKFace dataset [37] with ground truth (GT) labels and predictions using the equal width method.

We compare our method with direct regression and classification baselines, as well as current state of the art methods on three challenging datasets. In order to make fair comparisons of the methods, we use the same baseline architecture for all experiments. More specifically, we use a pre-trained ResNet50 [16] and replace the final fully connected layer with one fully connected layer of size 2048, a ReLU activation, and then a method-specific layer. For the direct regression approach (referred to as "Direct"), we use a fully connected layer of size 1 with a linear activation and train it by minimizing the MSE. For the RvC approach, we use one fully connected layer of size K with a single softmax activation and train it using the cross-entropy loss function. For our own method we use M fully connected layers of size L , with an individual softmax activation function on each layer, and train it by minimizing the sum of the individual cross-entropies for each discretization as in equation (1).

For all datasets, we train the network for 30 epochs using the ADAM optimizer [18] with a mini-batch size of 32, a learning rate of 0.0005 that is decreased by a factor of 0.1 every 10th epoch, and an L^2 -regularization factor of 0.001 on the weights. For data augmentation, we use random horizontal flipping of the images and apply a uniformly distributed random

translation and scaling between $[-20, 20]$ pixels and $[0.7, 1.4]$ respectively. All results are averaged over 10 trials with different random initializations of the last fully connected layers. The experiments were implemented in Matlab and the network training was done using an NVIDIA Titan V graphics card. The source code for our experiments is available at <https://github.com/axeber01/dold>.

5.1 Age Estimation

Table 1: Mean average error in years for the different methods on the UTKFace [37] test set.

Method	CORAL [7]	DCTD [15]	Direct (Ours)	RvC (Ours)	Equal Width (Ours)	Randomized Bins (Ours)
MAE	5.47 ± 0.01	4.65 ± 0.02	4.60 ± 0.02	4.71 ± 0.03	4.58 ± 0.03	4.55 ± 0.04

For age estimation, we test our method on the UTKFace dataset [37], which provides a large collection of images with human subjects labeled with ground truth ages. More specifically, we use the same train-test split of the data as in [7] and [15], where the ages are in the set of integers $t_n \in [21, 60]$. The subset contains 13,144 training images and 3,287 test images. Furthermore, the images have been cropped such that only the faces of the subjects are visible, as is shown in the examples in Figure 5.

For the RvC baseline, we simply use one class per age group in years, such that $\mathcal{C} = \{21, 22, \dots, 60\}$. We compare the baseline with two implementations of the proposed method: equal-width overlapping bins and randomized bins. For the equal width approach we let $L = 8$ and $M = 5$, such that $LM = 40$, which means that the network complexity similar to the complexity of the RvC baseline. In practice this means that the first output head will classify images into age categories of $\mathcal{D}^1 = \{21\text{--}25, 26\text{--}30, \dots, 56\text{--}60\}$, while the second output head will have $\mathcal{D}^2 = \{21, 22\text{--}26, 27\text{--}31, \dots, 57\text{--}60\}$ and so on.

For the randomized bins approach, we let $L = 10$ and $M = 30$ and draw a new sample of randomized bins for each run. This approach leads to a increased number of output heads compared to the RvC baseline, but the increase in model complexity is still small in relation to the size of the CNN backbone.

The results are shown and compared to current state-of-the art methods in Table 1, where we have evaluated the mean average error (MAE) of the different methods on the test set. The results are averaged over the trials with different random seeds and presented with the corresponding standard deviation. Of the two baselines, the direct method performs best. Since this is perhaps the most natural design choice, it should not come as a surprise, although other papers have reported contrary results on age estimation tasks [27]. Both the equal width and randomized bins approach improve over the RvC baseline significantly, and they yield a small improvement over the direct method, which show that for this task, our proposed methods are more effective than the baseline methods. Furthermore, we

reduce the average error compared to current state of the art by 2%. We hypothesize that this error reduction is due to the diverse representation of the target values, which also incorporates the ordinal relationship between the age categories.

5.2 Head Pose Estimation

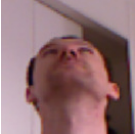
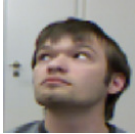
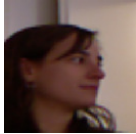
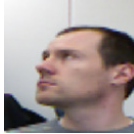
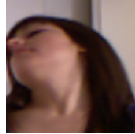
GT: Yaw: 4.13 Pitch: 57.79 Roll: 0.64	GT: Yaw: 20.98 Pitch: 25.88 Roll: 9.46	GT: Yaw: -48.44 Pitch: 17.11 Roll: 2.19	GT: Yaw: 43.64 Pitch: 24.55 Roll: 11.48	GT: Yaw: 47.24 Pitch: 49.74 Roll: -8.5
				
Predictions: Yaw: 4.09 Pitch: 57.96 Roll: 0.6	Predictions: Yaw: 20.73 Pitch: 26.11 Roll: 9.59	Predictions: Yaw: -48.05 Pitch: 16.06 Roll: 3.98	Predictions: Yaw: 49.05 Pitch: 28.21 Roll: 17.29	Predictions: Yaw: 31 Pitch: 31.08 Roll: -1.59

Figure 6: A subset of the images in BIWI dataset [10] with ground truth (GT) labels and predictions using the equal width method.

The BIWI dataset [10] consists of 24 video sequences of 20 subjects recorded in a controlled environment and each frame is labeled with the corresponding head pose of the subject. We use the train-test split defined as protocol 2 in [32], where 16 videos are used for training and 8 for testing. In total, the training set consists of 10,063 images and the test set of 5,065 images. The pose is represented using the yaw, pitch and roll angles of the head, where each angle is approximately distributed in the range $t_n \in [-75^\circ, 75^\circ]$.

Following [15], we use the same approach as for head pose estimation, but with small modifications needed to get a three dimensional output from the network. For the direct regression approach, we simply replace the last fully connected layer of size 1 by three fully connected layers of the same size. For the RvC, we use three chains of fully connected layers at the end, one for each angle, with a corresponding softmax head. We discretize each angle using 1 degree per bin, such that $\mathcal{C} = \{-75, -74, \dots, 75\}$.

For the equal width approach, we let $M = 3$ and $L = 50$, such that $LM = 150$. Again, this gives a similar network complexity as the RvC baseline. Hence each softmax head has 3 degrees per bin, i.e. $\mathcal{D}^1 = \{(-75) - (-73), \dots, 72 - 75\}$, $\mathcal{D}^2 = \{-75, (-74) - (-71), \dots, 73 - 75\}$ and similarly for \mathcal{D}^3 . For the randomized bins approach, we let $L = 20$ and $M = 30$ and make a new sample of randomized bins for each run.

The results for the BIWI dataset are shown in Table 2, where we have evaluated the MAE for the three different angles for each method. On average, direct regression performs best, while the randomized bins method is best at predicting the yaw angle. However, both equal width and randomized bins outperform the standard RvC approach. Additionally, our direct method reduces the current state of the art average error by 16 %, which shows that a carefully tuned regression baseline can outperform more sophisticated methods on this problem.

Table 2: Mean average error in degrees for the different methods on the Biwi [10] test set.

Method	Yang et al. [32]	DCTD [15]	Direct (Ours)	RvC (Ours)	Equal Width (Ours)	Randomized Bins (Ours)
Yaw	2.89	2.67 ± 0.08	2.64 ± 0.16	2.85 ± 0.12	2.62 ± 0.11	2.52 ± 0.06
Pitch	4.29	3.61 ± 0.12	2.75 ± 0.05	3.22 ± 0.09	3.12 ± 0.08	3.01 ± 0.10
Roll	3.60	2.75 ± 0.10	2.24 ± 0.07	2.48 ± 0.08	2.33 ± 0.09	2.34 ± 0.10
Average	3.60	3.01 ± 0.07	2.54 ± 0.05	2.85 ± 0.08	2.69 ± 0.04	2.63 ± 0.04

5.3 Historical Image Dating

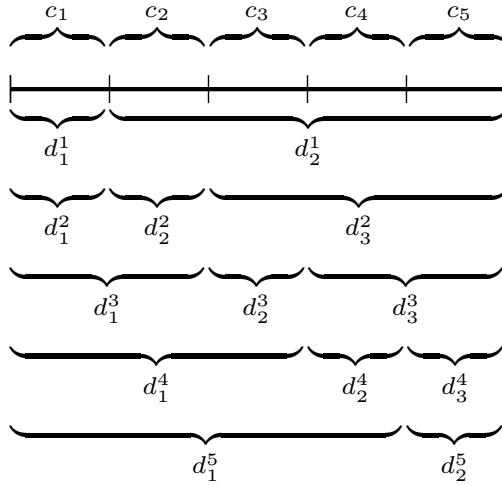


Figure 7: The sets of overlapping classes used for label diversity on the historical image dating task.

In order to test our method on a small dataset with a small number of ordinal classes, we ran experiments on the Historical Color Images (HCI) dataset [25]. The dataset consists of 1,375 color images from five decades, spanning from the 1930s to the 1970s. For evaluation, we use Monte Carlo random sampling with an 80/20 train-test split for each decade, drawn randomly at each iteration.

For this dataset, the target value can then be considered as belonging to one of five ordinal classes $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5\}$, where each class corresponds to one of the five decades. Likewise, the number of possibilities for creating our new sets of bins is limited, so the

methods of overlapping and randomized bins are not suitable. We instead create 5 new sets $\{\mathcal{D}^m\}_{m=1}^5$ as shown in Figure 7 and refer to this method simply as "Label Diversity". Although it is possible to define a distance metric between classes as the distance in decades, this is unsuitable, since the year 1939 is closer to 1940 than it is to 1949, but the distances in decades are the same. We therefore treat this as an ordinal classification problem and use the MAP estimate in equation (8) for inference.

We evaluate the methods in terms of classification accuracy and MAE. A sample of correct and incorrect predictions are shown in Figure 8. The results are shown in Table 3 and we conclude that using label diversity improves both accuracy and MAE over the regression baseline. Label diversity also decreases MAE compared to the RvC baseline, which we claim is due to the exploitation of the ranking between classes. Furthermore, our method improves the accuracy by one percentage point compared to the current state of the art method [4].

Table 3: Accuracy and mean average error in decades for the different methods on the HCI dataset [25].

Method	CNNOr [22]	PN [4]	ELB [4]	Direct (Ours)	RvC (Ours)	Label Diversity (Ours)
Acc (%)	41.56	56.67 \pm 2.30	54.90 \pm 2.40	46.9 \pm 1.7	57.1 \pm 1.9	57.7 \pm 2.0
MAE	1.04	0.67 \pm 0.05	0.63 \pm 0.04	0.76 \pm 0.01	0.72 \pm 0.04	0.67 \pm 0.03

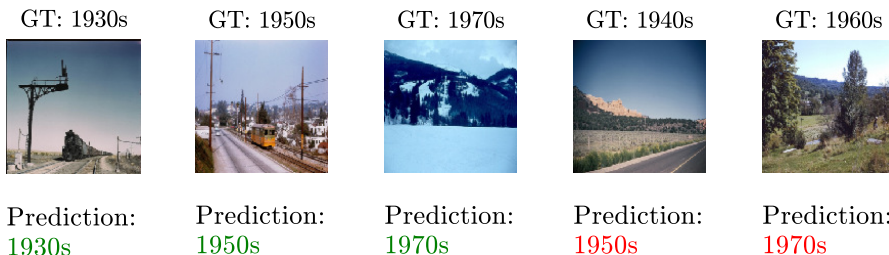


Figure 8: A subset of the images in HCI dataset [25] with ground truth (GT) labels and predictions using the label diversity method.

6 Conclusion

In this work, we have shown that employing a series of different discrete representations of the target values, it is possible to improve the predictive performance of a deep neural network, compared to when using a single such representation. For some problems, it can also outperform direct regression. We note that our methods yield the strongest improvement compared to the regression baseline on historical image dating, where the target belongs to a small set of ordinal classes. For head pose estimation, where the target is continuous, the improvement was either small or negligible, but there is a significant improvement over

the RvC baseline. For age estimation, where the target belongs to a large set of ordered classes, there is an improvement over both the regression and RvC method. Nevertheless, our method consistently improves over the RvC baseline for all methods. Hence our conclusion is that, if it is suitable to approach a regression problem via classification, then using several diverse representations can improve performance.

This opens up a wide range of options when it comes to selecting the representations, since there are many ways to create different discrete binnings of a continuous target value. As we have also shown, the number of discretizations and the number of bins for each discretization will have an impact on the prediction performance, but since these choices also affect the training convergence, it is difficult to select them for a given problem without extensive parameter search. In future research we will continue to investigate these questions and how diversity in label representations can be exploited in other ways.

References

- [1] A. Abbas and A. Zisserman. A geometric approach to obtain a bird’s eye view from an image. *arXiv preprint arXiv:1905.02231*, 2019.
- [2] E. Agustsson, R. Timofte, and L. Van Gool. Anchored regression networks applied to age estimation and super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1643–1652, 2017.
- [3] C. Beckham and C. Pal. Unimodal probability distributions for deep ordinal classification. *arXiv preprint arXiv:1705.05278*, 2017.
- [4] S. Belharbi, I. B. Ayed, L. McCaffrey, and E. Granger. Deep ordinal classification with inequality constraints. *arXiv preprint arXiv:1911.10720*, 2019.
- [5] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- [6] G. Brown and J. L. Wyatt. The use of the ambiguity decomposition in neural network ensemble learning methods. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 67–74, 2003.
- [7] W. Cao, V. Mirjalili, and S. Raschka. Rank-consistent ordinal regression for neural networks. *arXiv preprint arXiv:1901.07884*, 2019.
- [8] F.-J. Chang, A. Tuan Tran, T. Hassner, I. Masi, R. Nevatia, and G. Medioni. Faceposenet: Making a case for landmark-free face alignment. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1599–1608, 2017.
- [9] R. Diaz and A. Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4738–4747, 2019.

- [10] G. Fanelli, T. Weise, J. Gall, and L. V. Gool. Real time head pose estimation from consumer depth cameras. In *33rd Annual Symposium of the German Association for Pattern Recognition (DAGM'11)*, September 2011.
- [11] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. Citeseer, 1996.
- [12] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.
- [13] B.-B. Gao, C. Xing, C.-W. Xie, J. Wu, and X. Geng. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838, 2017.
- [14] S. Ginosar, K. Rakelly, S. Sachs, B. Yin, and A. A. Efros. A century of portraits: A visual historical record of american high school yearbooks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–7, 2015.
- [15] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. Dctd: Deep conditional target densities for accurate regression. *arXiv preprint arXiv:1909.12297*, 2019.
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [17] H.-W. Hsu, T.-Y. Wu, S. Wan, W. H. Wong, and C.-Y. Lee. Quatnet: Quaternion-based head pose estimation with multiregression loss. *IEEE Transactions on Multimedia*, 21(4):1035–1046, 2018.
- [18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1412.6980>.
- [19] Y. LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>.
- [20] L. Li and H.-T. Lin. Ordinal regression by extended binary classification. In *Advances in neural information processing systems*, pages 865–872, 2007.
- [21] X. Liu, Y. Zou, Y. Song, C. Yang, J. You, and B. K Vijaya Kumar. Ordinal regression with neuron stick-breaking for medical diagnosis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.
- [22] Y. Liu, A. W.-K. Kong, and C. K. Goh. Deep ordinal regression based on data relationship for small datasets.

-
- [23] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):607–626, 2008.
 - [24] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4920–4928, 2016.
 - [25] F. Palermo, J. Hays, and A. A. Efros. Dating historical color images. In *ECCV (6)*, pages 499–512, 2012.
 - [26] Y. Ren, L. Zhang, and P. N. Suganthan. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational intelligence magazine*, 11(1):41–53, 2016.
 - [27] R. Rothe, R. Timofte, and L. Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 10–15, 2015.
 - [28] N. Ruiz, E. Chong, and J. M. Rehg. Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2074–2083, 2018.
 - [29] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.
 - [30] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015.
 - [31] S. Workman, M. Zhai, and N. Jacobs. Horizon lines in the wild. In *British Machine Vision Conference (BMVC)*, 2016.
 - [32] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang. Fsa-net: Learning fine-grained structure aggregation for head pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1087–1096, 2019.
 - [33] B. Zeisl and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *European conference on computer vision*, pages 468–484. Springer, 2014.
 - [34] X. Zeng, C. Ding, Y. Wen, and D. Tao. Soft-ranking label encoding for robust facial age estimation. *arXiv preprint arXiv:1906.03625*, 2019.

- [35] L. Zhang, Z. Shi, M.-M. Cheng, Y. Liu, J.-W. Bian, J. T. Zhou, G. Zheng, and Z. Zeng. Robust regression via deep negative correlation learning. *arXiv preprint arXiv:1908.09066*, 2019.
- [36] Y. Zhang, L. Liu, C. Li, and C. C. Loy. Quantifying facial age by posterior of age comparisons. *arXiv preprint arXiv:1708.09687*, 2017.
- [37] Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

Paper II



Points to Patches: Enabling the Use of Self-Attention for 3D Shape Recognition

AXEL BERG^{1,2}, MAGNUS OSKARSSON², MARK O’CONNOR¹

¹Arm Research, ²Centre for Mathematical Sciences, Lund University

Abstract: While the Transformer architecture has become ubiquitous in the machine learning field, its adaptation to 3D shape recognition is non-trivial. Due to its quadratic computational complexity, the self-attention operator quickly becomes inefficient as the set of input points grows larger. Furthermore, we find that the attention mechanism struggles to find useful connections between individual points on a global scale. In order to alleviate these problems, we propose a two-stage Point Transformer-in-Transformer (Point-TnT) approach which combines local and global attention mechanisms, enabling both individual points and patches of points to attend to each other effectively. Experiments on shape classification show that such an approach provides more useful features for downstream tasks than the baseline Transformer, while also being more computationally efficient. In addition, we also extend our method to feature matching for scene reconstruction, showing that it can be used in conjunction with existing scene reconstruction pipelines.

1 Introduction

Due to the unordered nature of 3D point clouds, applying neural networks for shape recognition requires the use of permutation-equivariant architectures. The Transformer, first introduced by Vaswani et al. [33] for the task of natural language processing, is an example of such an architecture and given its recent success in the fields of image classification [17, 31], object detection [3], video analysis [25], speech recognition [10, 5, 21, 2], and more, its application to 3D point clouds is a natural step of exploration. While some attempts to adopt the Transformer architecture for this task have been made [19, 40, 11], they all suffer from different weaknesses, such as a reduced receptive field and high computational cost. Inspired by the Transformer-in-Transformer architecture for image processing [13], we propose a method that addresses both of these problems by using a two-stage attention mechanism which is able to learn more descriptive features while also lowering the number of required computations.

In summary, our main contributions are

1. We propose a two-stage Transformer architecture that combines attention mechanisms on a local and global scale. By sampling a sparse set of anchor points we create

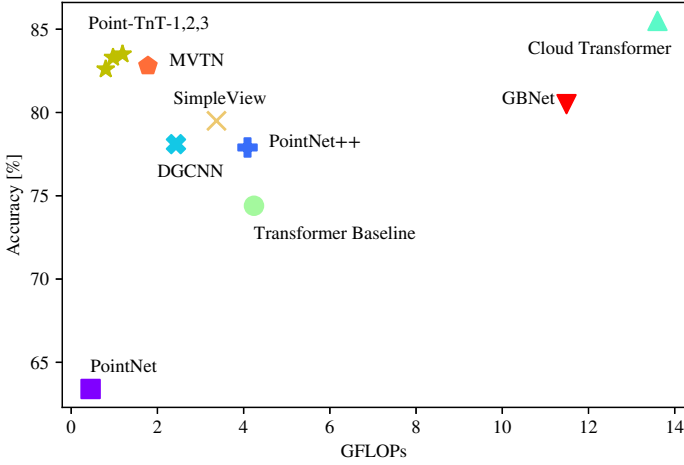


Figure 1: Classification accuracy and GFLOPs on the ScanObjectNN dataset [32]. Our proposed Point-TnT networks provides the best trade-off between accuracy and fast inference when compared to previous methods, and significantly improves performance compared to a Transformer baseline.

patches of local features. Self-attention can then be applied both on points within the patches and on the patches themselves.

2. Our experiments show that this approach gives significant uplifts compared to applying self-attention on the entire set of points, while also reducing the computational complexity.
3. We show that our proposed architecture achieves competitive results on 3D shape classification benchmarks, while requiring less floating point operations (FLOPs) compared to other methods, and that it can be used for improving 3D feature matching.

2 Related Work

Shape Recognition Extracting useful features from 3D point cloud shapes requires special considerations. In contrast to images, where the pixels are naturally ordered on a grid structure, a point cloud is typically represented as an unordered set of data points, each containing three-dimensional coordinates and possibly other feature channels, such as normal vectors and color information. This prevents the direct use of classical deep learning techniques like multi-layer perceptrons (MLPs) and convolutional neural networks (CNNs)

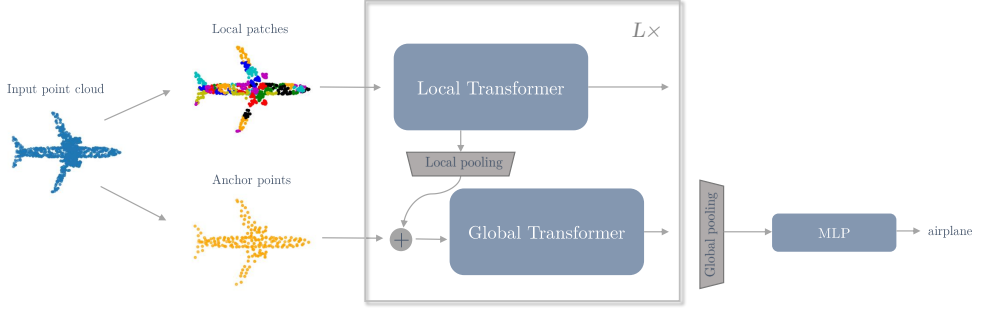


Figure 2: Overview of our proposed Point Transformer-in-Transformer method. Given an input point cloud, we sample a sparse set of anchor points and then create local patches around each anchor using the k -nearest neighbour graph. The anchors and patches are then fed into a sequential network of local and global Transformers, each with its own self-attention mechanism. At the final layer, global and local features can be extracted and used for downstream tasks, such as 3D shape classification.

when combining features from different points, since their predictions depend on the ordering of the input features. In recent years, several paradigms have been developed to work around this problem and they can roughly be divided into three categories: volumetric, multi-view and set based approaches.

The volumetric approach, as proposed in [36, 23], aims to map the points onto a three-dimensional occupancy grid of voxels, which allows for the use of CNNs for feature extraction. However, this approach suffers from poor scaling behaviour with the grid-resolution, making it intractable for processing of large point cloud scenes. This problem can be alleviated to some extent by using sparse convolutions [6], but will inevitably suffer from loss of geometric detail when points are mapped to an occupancy grid. In contrast, the multi-view approach instead maps the point clouds onto multiple two-dimensional grids, each capturing a different view of the scene. This method, first proposed in [30], has later been refined to use adaptive views that are learned at training time [12]. Others have proposed combining the multi-view approach with graph convolutional networks (GCN) in order to aggregate views over nearby view positions [35].

A family of neural networks termed Deep Sets, which allows for feature extraction on unordered sets without mapping them to any grid-like structure, was introduced by Zaheer et al. [38]. Such networks can be realized by weight sharing across all input points and the use of a symmetric aggregation function, e.g. the max or mean values of the sets, which enables extraction of permutation-equivariant point features. Concurrent work by Qi et al. [27] introduced PointNet, where this architecture was combined with a Spatial Transformer network [16] (not to be confused with the attention-based Transformer), that can learn to align point clouds in a common frame of reference [27]. This approach has later been enhanced by exploiting geometric properties of the point cloud, such as the nearest neighbour graph [28, 34], which enables aggregation not only over the entire point cloud,

but also over local neighbourhoods of points.

Self-Attention and the Transformer The Transformer architecture [33] belongs to the family of permutation-equivariant neural networks and is therefore a natural extension of the Deep Sets architecture. By allowing all elements of the input set to attend to each other, the Transformer is able to learn interactions between elements, which greatly enhances the network ability to learn complex interactions. It was first applied to point clouds by Lee et al. [19], who noted that this quickly becomes infeasible as the computational requirement grows quadratically with number of input points. They addressed this problem by introducing the concept of induced set attention by allowing the points to attend to a set of learnable parameters instead of themselves.

Zhao et al. [40] proposed a modified Transformer architecture, which only computes local attention between a point and its nearest neighbours. However, this approach removes the main benefit of self-attention, namely that it enables a global receptive field in early layers of the network. In contrast, Guo et al. [11] introduced offset attention with a neighbour embedding module in order to perform attention between groups of points, where each group is represented by a local feature. This method omits local attention entirely and for segmentation tasks it also suffers from quadratic complexity in the number of input points. Finally, a third approach has been explored in [4], where self-attention is computed both between points and channels within points, but only at a late stage in the network and only for the particular task of areal LiDAR image segmentation.

Recently, Transformers have also shown great success in the domain of computer vision in the form of the Vision Transformer [17, 31], where an image is regarded as a set of local patches, which can then be processed using a Transformer with minimal modifications. Han et al. [13] proposed the Transformer-in-Transformer architecture, which extends the baseline vision Transformer with pixel-wise attention within patches. Our approach is inspired by this work, in the sense that we use two branches of Transformers, one for local and one for global attention. This enables a global receptive field early in the network, while reducing the computational requirement of the self-attention operation compared to a baseline Transformer implementation.

3 Method

Preliminaries Let $X \in \mathbb{R}^{N \times d}$ be a matrix representation of a set, containing N features in d -dimensional space. Furthermore, following the definitions introduced in [33], let $Q = X_l W_Q$, $K = X_l W_K$ and $V = X_l W_V$ denote the queries, keys and values respectively, where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_h}$ are learnable parameters and d_h is the attention-

head dimension. We then define the self-attention (SA) operator as

$$\text{SA}(X) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V, \quad (1)$$

where the softmax function is applied on each row individually. We note that the SA operation is permutation-equivariant, since for any permutation π over the rows of X , we have that $\text{SA}(\pi X) = \pi \text{SA}(X)$. This property is especially useful if X represents a collection of unordered points, which is common in many 3D learning scenarios.

By performing multiple SA operations, where each operator has its own learnable set of weights, in parallel and concatenating the results column-wise, we can define the multi-headed SA operator (MSA) as

$$\text{MSA}(X) = [\text{SA}_1(X); \text{SA}_2(X); \dots; \text{SA}_h(X)]W_P, \quad (2)$$

where $;$ denotes column-wise concatenation, $W_P \in \mathbb{R}^{hd_h \times d}$ is another learnable parameter and h is the number of attention heads. Using the above notation, we can define the Transformer layer T_θ as

$$\tilde{X} = \text{MSA}(\text{LN}(X)) + X, \quad (3)$$

$$T_\theta(X) = \text{MLP}(\text{LN}(\tilde{X})) + \tilde{X}, \quad (4)$$

where MLP denotes a multi-layer perceptron with a single hidden layer and GELU activation [15], LN is the LayerNorm [1] operation and θ is the collection of parameters for the particular layer. Here we use the pre-norm [14] variant of the Transformer, where LayerNorm is applied before the MSA and MLP operations.

In order to aggregate features in a permutation-invariant fashion, we follow [34] and compute the maximum and average over all features and concatenate the results column-wise. This can be compactly expressed as

$$\alpha(X) = [\max_i X_i; \frac{1}{N} \sum_i X_i], \quad i = 1, \dots, N, \quad (5)$$

where X_i are the rows of X .

Point Transformer-in-Transformer Now we are ready to define our main architecture. Let $\mathcal{X} = \{x_i\}_{i=1}^N$ be a set of N points in three-dimensional space. A naive application of the Transformer architecture would be to apply self-attention between all points, such that

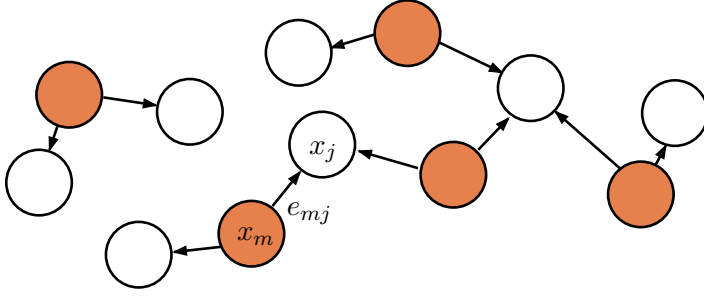


Figure 3: An example of the sparse subgraph \mathcal{H} , using $M = 5$ anchor points (shown in orange) and $k = 2$ nearest neighbours.

each point in the set can attend to every other point. Here we instead investigate how the attention mechanism can be applied within and between local patches of points, effectively splitting the self-attention operator into two different branches. An overview of our method can be seen in Figure 2.

First, we compute the k -nearest neighbour (k -NN) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertices $\mathcal{V} = \{i\}_{i=1}^N$ and directed edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ from vertex i to j if x_j is one of x_i 's k -nearest neighbours. In order to create patches, we then sample a sparse set of M anchor points and aggregate features from their neighbours. We do this by considering the subgraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}')$ of \mathcal{G} , which has edges from i to j if $i \in \mathcal{V}' \wedge (i, j) \in \mathcal{E}$, where $\mathcal{V}' \subset \mathcal{V}$ and $|\mathcal{V}'| = M < N$. In practice, we create \mathcal{H} by sampling a subset of points using farthest point sampling [8], as shown in Figure 3, which guarantees that the vertices in \mathcal{V}' are evenly distributed across the whole point cloud. From now on, we will refer to the set $\mathcal{V} = \{x_m\}_{m \in \mathcal{V}'}$ as the anchor points.

Since local geometry is best represented in a local frame of reference [34], we then extract edge features for each anchor point as $E^m = [e_{m1}, \dots, e_{mk}]^T$, where

$$e_{mj} = x_j - x_m, \quad (m, j) \in \mathcal{E}'. \quad (6)$$

Without loss of generality, we can stack the anchor points into a matrix $Y = [x_1, \dots, x_M]^T \in \mathbb{R}^{M \times 3}$, where the order of the anchor points is not important. These are then, together with their corresponding edge features, projected to higher dimensions as $Y_0 = YW_Y$ and $E_0^m = E^mW_E$, where $W_Y \in \mathbb{R}^{3 \times d_Y}$ and $W_E \in \mathbb{R}^{3 \times d_E}$ respectively.

The anchor and edge features are then processed using two branches of L sequential Transformer blocks. The local branch computes self-attention between edge features, which enables interaction between edges within the neighbourhoods of the anchor points:

$$E_l^m = T_{\theta_l}^{\text{local}}(E_{l-1}^m), \quad l = 1, \dots, L. \quad (7)$$

After each local Transformer layer, the neighbourhood features are aggregated and concatenated into a new matrix as $E_l = [\alpha(E_l^1), \dots, \alpha(E_l^M)] \in \mathbb{R}^{M \times 2d_E}$, which is then added to each anchor point using another linear projection:

$$\tilde{Y}_{l-1} = Y_{l-1} + E_l W_l, \quad l = 1, \dots, L, \quad (8)$$

where $W_l \in \mathbb{R}^{2d_E \times d_Y}$. Now, Y_l contains feature descriptors for the local patches around each anchor point. The global branch then computes attention between the patches, which enables interaction on a global scale:

$$Y_l = T_{\theta_l}^{\text{global}}(\tilde{Y}_{l-1}), \quad l = 1, \dots, L. \quad (9)$$

In order to exploit intermediate feature representations, the anchor point features from each layer are concatenated, combined using a single-layer MLP and aggregated in order to form a single global feature for the entire point cloud:

$$Z = \alpha(\text{MLP}([Y_1; \dots; Y_L])), \quad (10)$$

such that $Z \in \mathbb{R}^{2d_f}$, where d_f is the embedding dimension of the global feature. The global feature is then used for downstream tasks, using e.g. another MLP for classification. Alluding to the Transformer-in-Transformer for images [13], we refer to our proposed method as Point Transformer-in-Transformer (Point-TnT).

Computational Analysis Whereas a naive transformer implementation requires $\mathcal{O}(N^2)$ computations for the SA operator, the complexity is reduced significantly by splitting the attention into local and global branches. For our method, the complexity of the local and global transformers are $\mathcal{O}(Mk^2)$ and $\mathcal{O}(M^2)$ respectively, resulting in a reduced number of self-attention operations compared to a naive Transformer implementation as long as $Mk^2 + M^2 < N^2$, a condition which can easily be satisfied by limiting the number of neighbours and anchor points.

4 Experiments

4.1 Shape Classification

We train and evaluate our model on the ScanObjectNN dataset [32], which contains 14,510 real-world 3D objects in 15 categories, obtained from scans of indoor environments. This dataset is particularly challenging, since the point clouds contain cluttered backgrounds

and partial occlusions. We use the hardest version of the dataset (PB_T50_RS), which has been altered using random perturbations, and the official 80/20 train/test split. This dataset also comes with per-point labels that can be used to segment the point cloud into the object and background categories. While some methods use the segmentation masks during training in order to learn to discard points belonging to the background, we do not exploit this information, since it requires additional computational overhead.

In the default setting, we use $M = 192$ anchor points and $k = 20$ neighbours. The embedding dimensions are chosen as $d_Y = 192$ and $d_E = 32$ for the anchor points and edges respectively, and we use a global feature embedding dimension of $d_f = 1024$ and a total of $L = 4$ sequential Transformer blocks. Following [31], we let $d_Y/h = 64$ and consequently use $h = 3$ attention heads for both Transformer branches. As in [34], the final MLP used for classification contains two hidden layers of size 512 and 256 respectively, with batch normalization and dropout applied in each layer. We train our network using the standard cross-entropy loss function for 500 epochs, with a batch size of 32 using the AdamW optimizer [22] with a weight decay of 0.1 and a cosine learning rate schedule starting at 0.001. For data augmentation, we use RSMix [18] in addition to random anisotropic scaling and shifting. For each experiment, we train our model three times and report a 95 % confidence interval for the mean accuracy¹.

We compare our method to a naive approach that applies self-attention directly on all points, which corresponds to using all points as anchors and no neighbours, thereby discarding the local branch of the network, and refer to this method simply as Baseline. When comparing to previously published methods, we use two different evaluation protocols. Protocol 1 uses the model at the last training epoch for evaluation on the test set, whereas Protocol 2 evaluates the model on the test set each training epoch and reports the best obtained test accuracy. Although Protocol 2 clearly exploits information from the test set, we present results using both protocols for better side-by-side comparisons with other methods. For further discussion of different evaluation protocols, we refer to [9].

We compare the overall accuracy and average per-class accuracy to previously published methods in Table 1. Our method achieves an accuracy on par with state-of-the-art results, and the best result among the methods that do not exploit the background segmentation masks during training, outperforming all multi-view methods. Furthermore, it can be observed that using a sparse set of anchor points in conjunction with aggregation over neighbouring points performs significantly better than the baseline approach, which is not competitive on this task. This shows that global point-wise attention is not enough to capture semantic properties of objects, and that it is necessary to consider local geometric properties in order to efficiently utilize the attention mechanism.

For completeness, we also evaluate our method on the ModelNet40 dataset [36], which

¹Code is available at <https://github.com/axeber01/point-tnt>

Table 1: Classification results on ScanObjectNN [32]. Results obtained using Protocol 2 are marked by *.

Method	input	overall acc.	mean acc.
PointNet [27]	1024p	68.2	63.4
SpiderCNN [37]	1024p	73.7	69.8
PointNet++ [27]	1024p	77.9	75.4
DGCNN [34]	1024p	78.1	73.6
PointCNN [20]	1024p	78.5	75.1
BGA-PointNet++ [32]	mask, 1024p	80.2	77.5
BGA-DGCNN [32]	mask, 1024p,	79.7	75.7
Cloud Transformer [24]	mask, 2048p	85.5*	83.1*
SimpleView [9]	6 views	79.5	-
GBNet [29]	1024p	80.5*	77.8*
MVTN [12]	12 views	82.8*	-
Baseline (ours)	1024p	74.4 ± 1.3	69.6 ± 1.5
		$75.4 \pm 0.4^*$	$71.3 \pm 1.1^*$
Point-TnT (ours)	1024p	83.5 ± 0.1	81.0 ± 1.3
		$84.6 \pm 0.5^*$	$82.6 \pm 1.2^*$
Point-TnT (ours)	2048p	84.2 ± 0.9	81.8 ± 0.9
		$85.0 \pm 0.9^*$	$83.0 \pm 0.8^*$

contains 9,843 synthetic shapes for training and 2,468 for testing, in 40 different categories. We obtain 92.6 ± 0.2 % and 93.2 ± 0.2 % accuracy using Protocol 1 and 2 respectively, which is competitive with other methods.

We visualize the learned attention patterns from the different attention heads, as shown in Figure 4, using our proposed method and the naive baseline. It can be seen that when using a sparse set of anchor points, using their neighbours to form patches, the attention mechanism effectively learns to segment the point cloud into semantically meaningful parts. This allows the network to connect different parts of the point cloud when computing the global feature. However, when using the baseline approach, the attention map is not semantically meaningful. This illustrates the importance of neighbourhood context when computing self-attention, which is not used in the naive baseline approach.

4.2 Ablation Study

Model scaling In order to highlight the trade-off between model size, computational complexity and classification accuracy, we ablate the number of attention heads (and scale the embedding dimension d_Y accordingly) for our method. In Table 2 it can be seen that accuracy increases with the model size and that the number of FLOPs is low compared to the

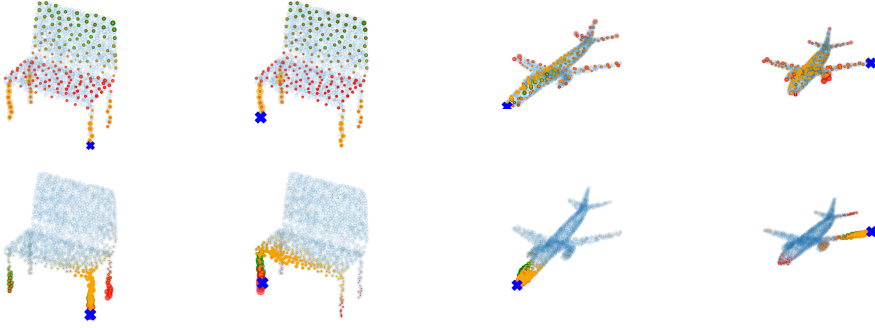


Figure 4: Visualization of the learned attention patterns in the final Transformer layer. The blue cross indicates the point for which the attention map is computed and the red, green and yellow dots highlight the attention for the three different heads, where the size of the points are scaled proportionally to the magnitude of the attention weights. Top: our proposed method using a sparse set of anchor points. Bottom: baseline method, using all points as anchors.

baseline, even for similar model sizes. As shown in Section 3, splitting the self-attention operator into two branches reduces the number of computations significantly, which in practice leads to a reduced number of FLOPs per forward-pass through the network.

In Figure 1, we illustrate the trade-off between FLOPs and classification accuracy for different methods, highlighting the competitive trade-off our proposed method. This result is in agreement with recent trends in computer vision, where Transformers are shown to be more computationally efficient compared to other architectures [13].

Attention mechanisms In order to demonstrate the effectiveness of the self-attention mechanisms, we ablate both the global and local attention modules in the Transformer blocks of the network by disabling the corresponding MSA operations. For the baseline method, there is only a single (global) attention mechanism which can be ablated. The results, shown in Table 3, verify that for our proposed method, both modules are indeed useful and increase the overall accuracy. Furthermore, they are additive in the sense that they can be used interchangeably or in combination, in order to trade-off accuracy for computation and model size. The tighter bound on the accuracy when using global attention also suggests that it helps stabilize training. The baseline method also benefits from attention to some extent, but adding it to the network does not yield the same improvement as simply splitting the network into two branches.

Number of anchors and neighbours Finally, we investigate the effect of varying the number of anchor points and neighbours, as shown in Table 4. As expected, a relatively large number of anchors is required in order to accurately represent the global shape of the point cloud. However, concerning the number of nearest neighbours, it seems that 10 is sufficient for good accuracy, but increasing it further reduces variance between different training

runs.

Table 2: Model size, computational complexity and accuracy on ScanObjectNN for the baseline and our proposed method with different number of attention heads.

Method	#params	GFLOPs	acc.
Baseline	3.8M	4.33	74.4 ± 1.3
Point-TnT-1	1.7M	0.80	82.6 ± 0.4
Point-TnT-2	2.6M	0.97	83.3 ± 0.1
Point-TnT-3	3.9M	1.19	83.5 ± 0.1

Table 3: Accuracy on ScanObjectNN with and without different attention mechanisms for the baseline and our proposed method.

Method	local att.	global att.	acc.
Baseline		\times	67.1 ± 0.7
		\checkmark	74.4 ± 1.3
Point-TnT	\times	\times	79.8 ± 0.8
	\checkmark	\times	80.4 ± 1.8
	\times	\checkmark	82.9 ± 0.1
	\checkmark	\checkmark	83.5 ± 0.1

Table 4: Accuracy on ScanObjectNN for different number of anchor points M and nearest neighbours k .

	anchors			neighbours		
M/k	12	48	192	5	10	20
acc.	74.2 ± 0.7	82.5 ± 0.8	83.5 ± 0.1	79.7 ± 1.6	83.5 ± 1.5	83.5 ± 0.1

4.3 Feature Matching on 3DMatch

In order to demonstrate how Point-TnT can be applied to a real-world scene reconstruction scenario, we consider the problem of feature matching on the 3DMatch dataset [39], which consists of 62 indoor scenes collected from RGB-D measurements. The goal of the feature matching task is to generate descriptors of local patches of the scenes which can be used for matching scans that have significant overlap. More specifically, given a pair of point cloud scenes $(\mathcal{X}, \mathcal{X}')$ with at least 30 % overlap, we find the corresponding points by extracting local features and matching them using nearest neighbour search. It then becomes possible to register the two scenes by estimating a rigid transformation using e.g. RANSAC.

We use the official split, with 54 scenes for training and 8 for testing, and the same pre-processing setup as in DIP [26], with the exception of the feature extraction network. Whereas the original implementation uses a Spatial Transformer for initial alignment of the point clouds and then a simple PointNet for feature extraction, we remove the Spatial Transformer entirely and replace the PointNet with our Point-TnT model. During

training, we sample local patches consisting of $N = 256$ points from the overlapping regions, and train by minimizing the hardest contrastive loss [7]. We train our network for 10 epochs using the AdamW optimizer [22] with a weight decay of 0.1 and a cosine learning rate schedule starting at 0.0001, without any data augmentation. Since each local patch processed by the network consists of only 256 points, we modify the parameters of our method to use $M = 48$ and $k = 10$ anchor points and neighbours respectively. For fair comparison to DIP, we use the same feature dimension (32) at the final layer as the original implementation.

During evaluation, we randomly sample 5000 points from each fragment and report the feature matching recall (FMR), i.e. the fraction of successful alignments with an inlier ratio of at least 5 %, where an inlier pair is defined by being less than 10 cm apart. In order to match corresponding patches, we use the global feature pairs (Z, Z') , where Z and Z' are calculated using (10) for the two patches, and perform mutual nearest neighbour search in feature space. We refer to the original DIP authors’ publication for a complete description of the experimental settings and evaluation protocol [26].

The results shown in Table 5 suggest that Point-TnT finds more descriptive features than the original DIP implementation and reduces the number of unsuccessful matches by 38 %, without requiring any initial alignment, which shows that our method can be used for improving real-world scene reconstruction pipelines.

Table 5: Feature matching recall and standard deviation on 3DMatch.

Method	FMR	std
DIP [26]	0.948	0.046
DIP + Point-TnT	0.968	0.031

5 Conclusions

In this work, we have explored the limitations and advantages of the Transformer architecture for 3D shape recognition. We have shown that naively applying self-attention to all points in a point cloud is both computationally inefficient and not very useful for learning descriptive feature representations. However, when applied within and between local patches of points, representation ability is drastically improved. This result is in agreement with recent works in image classification [17, 13], where the Transformer architecture has shown to work better on local image patches rather than individual pixels. It also makes feature extraction more computationally tractable, which creates new opportunities for applying attention mechanisms to edge use cases that rely on unstructured 3D data, such as simultaneous localization and mapping (SLAM), where computational resources are lim-

ited. In future work, we will consider integrating and extending our method for mobile SLAM pipelines.

References

- [1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] A. Berg, M. O'Connor, and M. T. Cruz. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In *Proc. Interspeech 2021*, pages 4249–4253, 2021. doi: 10.21437/Interspeech.2021-1286.
- [3] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.
- [4] L. Chen, Z. Xu, Y. Fu, H. Huang, S. Wang, and H. Li. Dapnet: A double self-attention convolutional network for segmentation of point clouds. *arXiv preprint arXiv:2004.08596*, 2020.
- [5] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li. Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5904–5908. IEEE, 2021.
- [6] C. Choy, J. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.
- [7] C. Choy, J. Park, and V. Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8958–8966, 2019.
- [8] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.
- [9] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng. Revisiting point cloud shape classification with a simple and effective baseline. *International Conference on Machine Learning*, 2021.
- [10] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented Transformer for Speech Recognition. *Proc. Interspeech 2020*, pages 5036–5040, 2020.

- [11] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.
- [12] A. Hamdi, S. Giancola, and B. Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021.
- [13] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.
- [16] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015.
- [17] A. Kolesnikov, A. Dosovitskiy, D. Weissenborn, G. Heigold, J. Uszkoreit, L. Beyer, M. Minderer, M. Dehghani, N. Houlsby, S. Gelly, T. Unterthiner, and X. Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.
- [18] D. Lee, J. Lee, J. Lee, H. Lee, M. Lee, S. Woo, and S. Lee. Regularization strategy for point cloud via rigidly mixed sample. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15900–15909, 2021.
- [19] J. Lee, Y. Lee, J. Kim, A. Kosiolek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.
- [20] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.
- [21] A. T. Liu, S.-W. Li, and H.-y. Lee. Tera: Self-supervised learning of transformer encoder representation for speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2351–2366, 2021.
- [22] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.
- [23] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

-
- [24] K. Mazur and V. Lempitsky. Cloud transformers: A universal approach to point cloud processing tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10715–10724, 2021.
 - [25] D. Neimark, O. Bar, M. Zohar, and D. Asselmann. Video Transformer Network. *arXiv preprint arXiv:2102.00719*, 2021.
 - [26] F. Poiesi and D. Boscaini. Distinctive 3d local deep descriptors. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5720–5727. IEEE, 2021.
 - [27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
 - [28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.
 - [29] S. Qiu, S. Anwar, and N. Barnes. Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 2021.
 - [30] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.
 - [31] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.
 - [32] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1588–1597, 2019.
 - [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. *Advances in neural information processing systems*, 30, 2017.
 - [34] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5): 1–12, 2019.
 - [35] X. Wei, R. Yu, and J. Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859, 2020.

- [36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.
- [37] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.
- [38] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3394–3404, 2017.
- [39] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811, 2017.
- [40] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

Paper III

Keyword Transformer: A Self-Attention Model for Keyword Spotting

AXEL BERG^{1,2,*}, MARK O’CONNOR^{1,*}, MIGUEL TAIRUM CRUZ¹

¹Arm Research, ²Centre for Mathematical Sciences, Lund University

Abstract: The Transformer architecture has been successful across many domains, including natural language processing, computer vision and speech recognition. In keyword spotting, self-attention has primarily been used on top of convolutional or recurrent encoders. We investigate a range of ways to adapt the Transformer architecture to keyword spotting and introduce the Keyword Transformer (KWT), a fully self-attentional architecture that exceeds state-of-the-art performance across multiple tasks without any pre-training or additional data. Surprisingly, this simple architecture outperforms more complex models that mix convolutional, recurrent and attentive layers. KWT can be used as a drop-in replacement for these models, setting two new benchmark records on the Google Speech Commands dataset with 98.6% and 97.7% accuracy on the 12 and 35-command tasks respectively.

1 Introduction

Recent works in machine learning show that the Transformer architecture, first introduced by Vaswani et al. [28], is competitive not only in language processing, but also in e.g. image classification, [12, 27, 34], image colorization [18], object detection [7], automatic speech recognition [14, 8, 20], video classification [22] and multi-agent spatiotemporal modeling [5]. This can be seen in the light of a broader trend, where a single neural network architecture generalizes across many domains of data and tasks.

Attention mechanisms have also been explored for keyword spotting [11, 25], but only as an extension to other architectures, such as convolutional or recurrent neural networks.

Inspired by the strength of the simple Vision Transformer (ViT) model [12] in computer vision and by the techniques that improves its data-efficiency [27], we propose an adaptation of this architecture for keyword spotting and find that it matches or outperforms existing models on the much smaller Google Speech Commands dataset [32] without additional data.

We summarize our main contributions as follows:

1. An investigation into the application of the Transformer architecture to keyword

*Equal contribution.

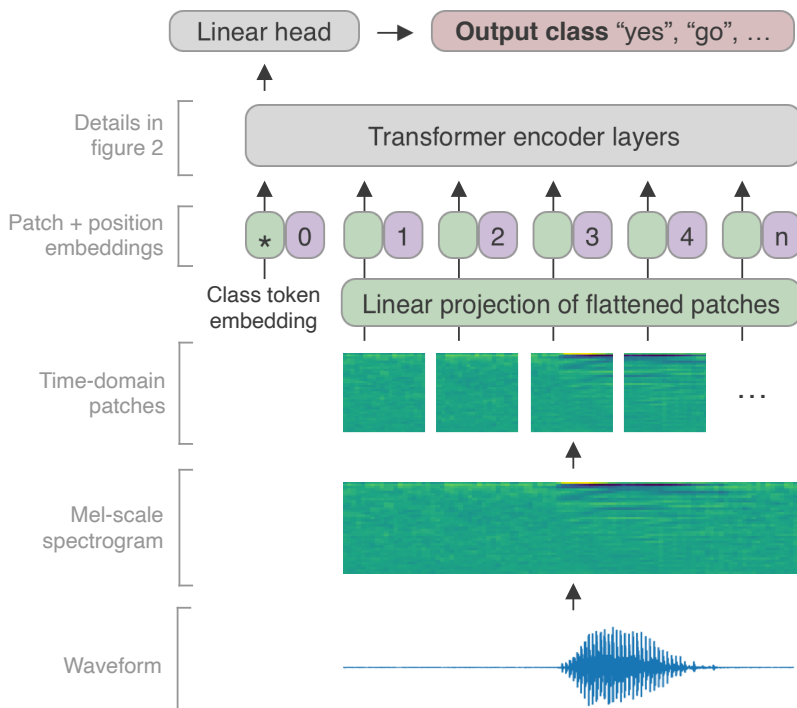


Figure 1: The Keyword Transformer architecture. Audio is preprocessed into a mel-scale spectrogram, which is partitioned into non-overlapping patches in the time domain. Together with a learned class token, these form the input tokens for a multi-layer Transformer encoder. As with ViT [12], a learned position embedding is added to each token. The output of the class token is passed through a linear head and used to make the final class prediction.

spotting, finding that applying self-attention is more effective in the time domain than in the frequency domain.

2. We introduce the Keyword Transformer, as illustrated in Figure 1, a fully self-attentional architecture inspired by ViT [12] that can be used as a drop-in replacement for existing keyword spotting models and visualize the effect of the learned attention masks and positional embeddings.
3. An evaluation of this model across several tasks using the Google Speech Commands dataset with comparisons to state-of-the-art convolutional, recurrent and attention-based models.
4. An analysis of model latency on a mobile phone, showing that the Keyword Transformer is competitive in edge use cases.

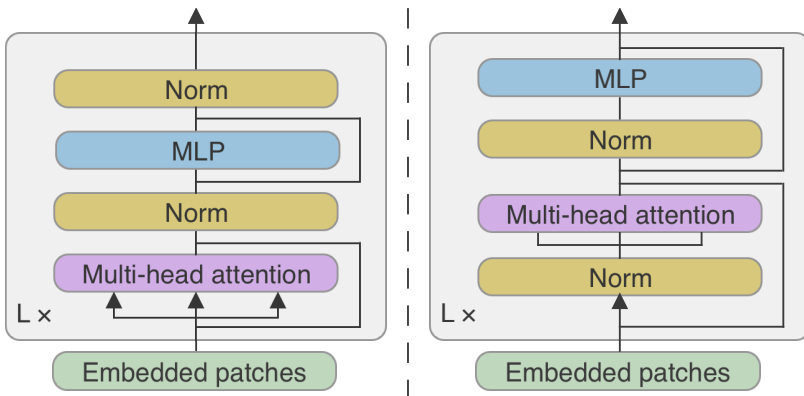


Figure 2: The PostNorm (left) and PreNorm (right) Transformer encoder architectures. KWT uses a PostNorm encoder.

2 Related Work

2.1 Keyword Spotting

Keyword spotting is used to detect specific words from a stream of audio, typically in a low-power always-on setting such as smart speakers and mobile phones. To achieve this, audio is processed locally on the device. In addition to detecting target words, classifiers may also distinguish between “silence” and “unknown” for words or sounds that are not in the target list.

In recent years, machine learning techniques, such as deep (DNN), convolutional (CNN), recurrent (RNN) and Hybrid-Tree [13] neural networks, have proven to be useful for keyword spotting. These networks are typically used with a pre-processing pipeline that extracts the mel-frequency cepstrum coefficients (MFCC) [10]. Zhang et al. [35] investigated several small-scale network architectures and identified depthwise-separable CNN (DS-CNN) as providing the best classification/accuracy tradeoff for memory footprint and computational resources. Other works have improved upon this result using synthesized data [19], temporal convolutions [9, 21], and self-attention [11]. Recently Rybakov et al. [25] achieved a new state of the art result on Google Speech Commands using MHAtt-RNN, a non-streaming CNN, RNN and multi-headed (MH) self-attention model.

2.2 Self-Attention and the Vision Transformer

Dosovitskiy et al. introduced the Vision Transformer (ViT) [12] and showed that Transformers can learn high-level image features by computing self-attention between different image patches. This simple approach outperformed CNNs but required pre-training on

large datasets. Tournou et al. [27] improved data efficiency using strong augmentation, careful hyperparameter tuning and token-based distillation.

While Transformers have been explored for wake word detection [31] and voice triggering [4], to the best of our knowledge fully-attentional models based on the Transformer architecture have not been investigated for keyword spotting. Our approach is inspired by ViT, in the sense that we use patches of the audio spectrogram as input and closely follows [27] to understand how generally this technique applies to new domains. We restrict ourselves to a non-streaming setting in this work, noting that others have previously investigated extensions of Transformers to a streaming setting [31, 8].

3 The Keyword Transformer

3.1 Model Architecture

Let $X \in \mathbb{R}^{T \times F}$ denote the output of the MFCC spectrogram, with time windows $t = 1, \dots, T$ and frequencies $f = 1, \dots, F$. The spectrogram is first mapped to a higher dimension d , using a linear projection matrix $W_0 \in \mathbb{R}^{F \times d}$ in the frequency domain. In order to learn a global feature that represents the whole spectrogram, a learnable class embedding $X_{\text{class}} \in \mathbb{R}^{1 \times d}$ is concatenated with the input in the time-domain. Then a learnable positional embedding matrix $X_{\text{pos}} \in \mathbb{R}^{(T+1) \times d}$ is added, such that the input representation fed into the Transformer encoder is given by

$$X_0 = [X_{\text{class}}; XW_0] + X_{\text{pos}} \quad (1)$$

The projected frequency-domain features are then fed into a sequential Transformer encoder consisting of L multi-head attention (MSA) and multi-layer perceptron (MLP) blocks. In the l :th Transformer block, queries, keys and values are calculated as $Q = X_l W_Q$, $K = X_l W_K$ and $V = X_l W_V$ respectively, where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_h}$ and d_h is the dimensionality of each attention-head. The self attention (SA) is calculated as

$$\text{SA}(X_l) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V \quad (2)$$

The MSA operation is obtained by linearly projecting the concatenated output, using another matrix $W_P \in \mathbb{R}^{kd_h \times d}$, from the k attention heads.

$$\text{MSA}(X_l) = [\text{SA}_1(X_l); \text{SA}_2(X_l); \dots; \text{SA}_k(X_l)]W_P \quad (3)$$

In our default setting, we use the PostNorm [28] Transformer architecture as shown in Figure 2, where the Layernorm (LN) [6] is applied after the MSA and MLP blocks, in

contrast to the PreNorm [15] variant, where LN is applied first. This decision is discussed further in the ablation section. As is typical for Transformers, we use GELU [16] activations in all MLP blocks.

In summary, the output of the l :th Transformer block is given by

$$\tilde{X}_l = \text{LN}(\text{MSA}(X_{l-1}) + X_{l-1}), \quad l = 1, \dots, L \quad (4)$$

$$X_l = \text{LN}(\text{MLP}(\tilde{X}_l) + \tilde{X}_l), \quad l = 1, \dots, L \quad (5)$$

At the output layer, the class embedding is fed into a linear classifier. Our approach treats time windows in a manner analogous to the handling of image patches in ViT. Whereas in ViT, the self-attention is computed over image patches, the attention mechanism here takes place in the time-domain, such that different time windows will attend to each other in order to form a global representation in the class embedding.

The model size can be adjusted by tuning the parameters of the Transformer. Following [27], we fix the number of sequential Transformer encoder blocks to 12, and let $d/k = 64$, where d is the embedding dimension and k is the number of attention heads. By varying the number of heads as $k = 1, 2, 3$, we end up with three different models as shown in Table 1.

Table 1: Model parameters for the KWT architecture.

Model	dim	mlp-dim	heads	layers	# parameters
KWT-1	64	256	1	12	607K
KWT-2	128	512	2	12	2,394K
KWT-3	192	768	3	12	5,361K

3.2 Knowledge Distillation

As introduced by Hinton et al. [17], knowledge distillation uses a pre-trained teacher’s predictions to provide an auxiliary loss to the student model being trained. Touvron et al. [27], introduced a distillation token, finding this benefits Transformers in the small data regime. This method adds a learned distillation token to the input. At the output layer this distillation token is fed into a linear classifier and trained using hard (one-hot) labels predicted by the teacher.

Let Z_{sc} be the logits of the student class token, Z_{sd} be the logits of the student distillation token and Z_t be the logits of the teacher model. The overall loss becomes

$$\mathcal{L} = \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_{\text{sc}}), y) + \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_{\text{sd}}), y_t), \quad (6)$$

where $y_t = \operatorname{argmax}(Z_t)$ are the hard decision of the teacher, y are the ground-truth labels, ψ is the softmax function and \mathcal{L}_{CE} is the cross-entropy loss. At inference time the class and distillation token predictions are averaged to produce a single prediction. Note that unlike Noisy Student [33], the teacher receives the same augmentation of the input as the student, effectively correcting labels made invalid by very strong augmentation. In all experiments, we use MHAtt-RNN as a teacher and denote distillation models with KWT \mathfrak{M} .

4 Experiments

4.1 Keyword Spotting on Google Speech Commands

We provide experimental results on the Google Speech Commands dataset V1 and V2 [32]. Both datasets consist of 1 second long audio snippets, sampled at 16 kHz, containing utterances of short keywords recorded in natural environments. V1 of the dataset contains 65,000 snippets of 30 different words, whereas V2 contains 105,000 snippets of 35 different words. The 12-label classification task uses 10 words: "up", "down", "left", "right", "yes", "no", "on", "off", "go", and "stop", in addition to "silence" and "unknown", where instances of the latter is taken from the remaining words in the dataset, whereas the 35-label task uses all available words. We use the same 80:10:10 train/validation/test split as [32, 35, 25] for side-by-side comparisons. We adhere as closely as possible to the evaluation criteria of [25], and for each experiment, we train the model three times with different random initializations.

As our intention is to explore the extent to which results using Transformers from other domains transfer to keyword spotting, we follow the choices and hyperparameters from [27] as closely as possible, with the notable exception that we found increasing weight decay from 0.05 to 0.1 to be important. Furthermore, we use the same data pre-processing and augmentation policy as in [25], which consists of random time shifts, resampling, background noise, as well as augmenting the MFCC features using SpecAugment [24]. We train our models over the same number of total input examples as MHAtt-RNN (12M) to allow a fair comparison. For clarity, the hyperparameters used in all experiments are reported in Table 2.

The results are shown in Table 3, where for our own results, we report a 95% confidence interval for the mean accuracy over all three model evaluations. Our best models match or surpass the previous state-of-the-art accuracies, with significant improvements on both the 12-label and 35-label V2-datasets. In general, Transformers tend to benefit more from large amounts of data, which could explain why KWT does not outperform MHAtt-RNN on the smaller V1-dataset. Nevertheless, we also note that knowledge distillation is effective in improving the accuracy of KWT in most scenarios.

Table 2: Hyperparameters used in all experiments.

Training		Pre-processing	
Training steps	23,000	Time window length	30 ms
Batch size	512	Time window stride	10 ms
Optimizer	AdamW	#DCT Features	40
Learning rate	0.001	Data augmentation	
Schedule	Cosine	Time shift [ms]	[-100, 100]
Warmup epochs	10	Resampling	[0.85, 1.15]
Regularization		Background vol.	0.1
Weight decay	0.1	#Time masks	2
Label smoothing	0.1	Time mask size	[0,25]
Dropout	0	#Frequency masks	2
		Frequency mask size	[0,7]

Table 3: Accuracy on Speech Commands V1 [2] and V2 [3].

Model	V1-12	V2-12	V2-35
DS-CNN [35]	95.4		
TC-ResNet [9]	96.6		
Att-RNN [11]	95.6	96.9	93.9
MatchBoxNet [21]	97.48 \pm 0.11	97.6	
Embed + Head [19]		97.7	
MHAtt-RNN [25]	97.2	98.0	
Res15 [29]		98.0	96.4
MHAtt-RNN (Ours)	97.50 \pm 0.29	98.36 \pm 0.13	97.27 \pm 0.02
KWT-3 (Ours)	97.24 \pm 0.24	98.54 \pm 0.17	97.51 \pm 0.14
KWT-2 (Ours)	97.36 \pm 0.20	98.21 \pm 0.06	97.53 \pm 0.07
KWT-1 (Ours)	97.05 \pm 0.23	97.72 \pm 0.01	96.85 \pm 0.07
KWT-3 $\text{\textcircled{A}}$ (Ours)	97.49 \pm 0.15	98.56 \pm 0.07	97.69 \pm 0.09
KWT-2 $\text{\textcircled{A}}$ (Ours)	97.27 \pm 0.08	98.43 \pm 0.08	97.74 \pm 0.03
KWT-1 $\text{\textcircled{A}}$ (Ours)	97.26 \pm 0.18	98.08 \pm 0.10	96.95 \pm 0.14

4.2 Ablation Studies

We investigate different approaches to self-attention by varying the shapes of the MFCC spectrogram patches that are fed into the Transformer. Using our default hyperparameters, the spectrogram consists of 98 time windows, containing 40 mel-scale frequencies. Our baseline uses time-domain attention, but we also investigate frequency-domain attention and intermediate steps where rectangular patches are used. We find time-domain attention to perform best, as shown in Figure 3. This is in agreement with previous findings that temporal convolutions work well for keyword spotting [9], since the first projection layer of our model can be interpreted as a temporal convolution with kernel size (40, 1) and stride 1 in the time-domain.

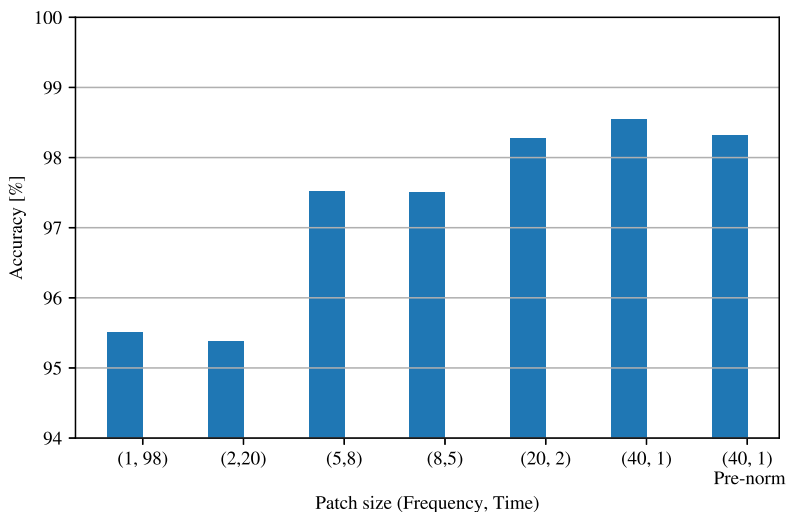


Figure 3: Accuracy on Speech Commands V2-12 using KWT-3 with different patch sizes.

We also investigate the use of PreNorm and PostNorm and found that the latter improves performance for keyword spotting in our experiments. This is contrary to previous findings on other tasks [23], where PreNorm has been shown to yield better results and we encourage further work to explore the role of normalization in Transformers across different domains.

4.3 Attention Visualization

In order to examine which parts of the audio signal the model attends to, we propagate the attention weights of each Transformer layer from the input to the class token by averaging the attention weights over all heads. This produces a set of attention weights for each time window of the input signal. Figure 4 shows the attention mask overlayed on the waveform of four different utterances. It can be seen that the model is able to pay attention to the important parts of the input while effectively suppressing background noise.

We also study the position embeddings of the final model by analyzing their cosine similarity, as shown in Figure 5. Nearby position embeddings exhibit a high degree of similarity and distant embeddings are almost orthogonal. This pattern is less emphasized for time windows near the start and the beginning of the audio snippets. We hypothesize that this is either because words are typically in the middle of each snippet and therefore relative position is more important there, or because the audio content at the start and end is less distinguishable.

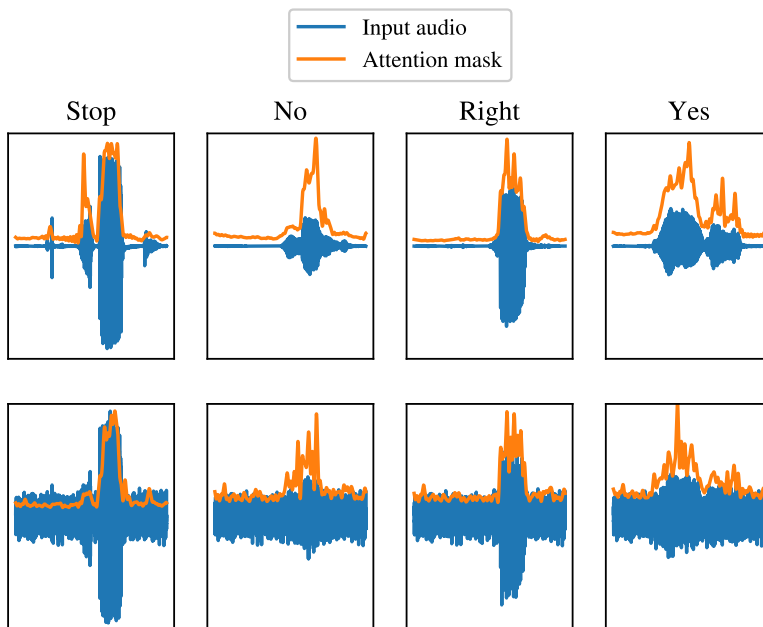


Figure 4: The learned attention mask, propagated from the input to the class token, overlaid on four different audio snippets, without (top) and with (bottom) background noise.

4.4 Latency Measurements

We converted our KWT models, DS-CNN (with stride) [35], TC-ResNet [9] and MHAtt-RNN [25] to Tensorflow (TF) Lite format to measure inference latency on a OnePlus 6 mobile device based on the Snapdragon 845 (4x Arm Cortex-A75, 4x Arm Cortex-A55) and report accuracy figures for the Google Speech Commands V2 with 12 labels and 35 labels [3, 25]. The TFLite Benchmark tool [1] is used to measure latency, defined by the processing time of a single one-second input. For each model, we do 10 warmup runs followed by 100 inference runs, capturing the average latency on a single thread.

In Figure 6 we observe that Transformer-based models are competitive with the existing state-of-the-art despite being designed with no regard to latency. There is a broad body of research on optimizing Transformer models — of particular note is the replacement of layer normalization and activations in [26] that decreases latency by a factor of three. Our findings here suggest many of these results could be leveraged in the keyword spotting domain to extend the practicality of these models.

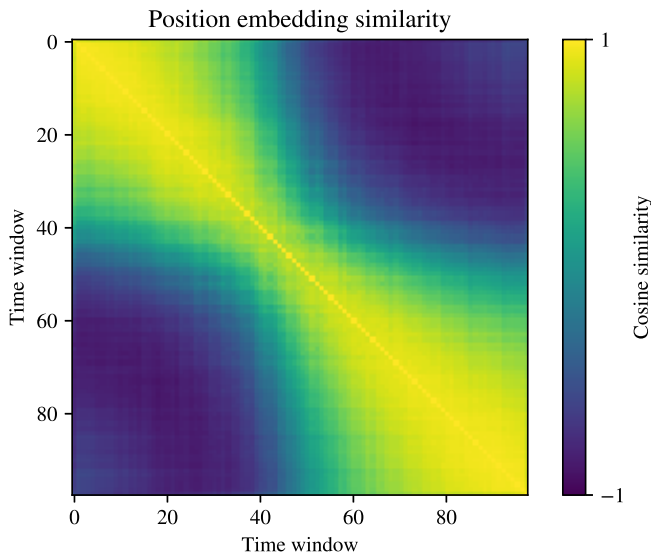


Figure 5: Cosine similarities of the learned position embeddings of KWT.

5 Conclusion

In this paper we explore the direct application of Transformers to keyword spotting, using a standard architecture and a principled approach to converting the audio input into tokens.

In doing so we introduce KWT, a fully-attentional model that matches or exceeds the state-of-the-art over a range of keyword spotting tasks with real-world latency that remains competitive with previous work.

These surprising results suggest that Transformer research in other domains offers a rich avenue for future exploration in this space. In particular we note that Transformers benefit from large-scale pre-training [12], have seen 5.5x latency reduction through model compression [26] and up to 4059x energy reduction through sparsity and hardware codesign [30]. Such improvements would make a meaningful impact on keyword spotting applications and we encourage future research in this area.

References

- [1] Tflite model benchmark tool. URL <https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/tools/benchmark>.

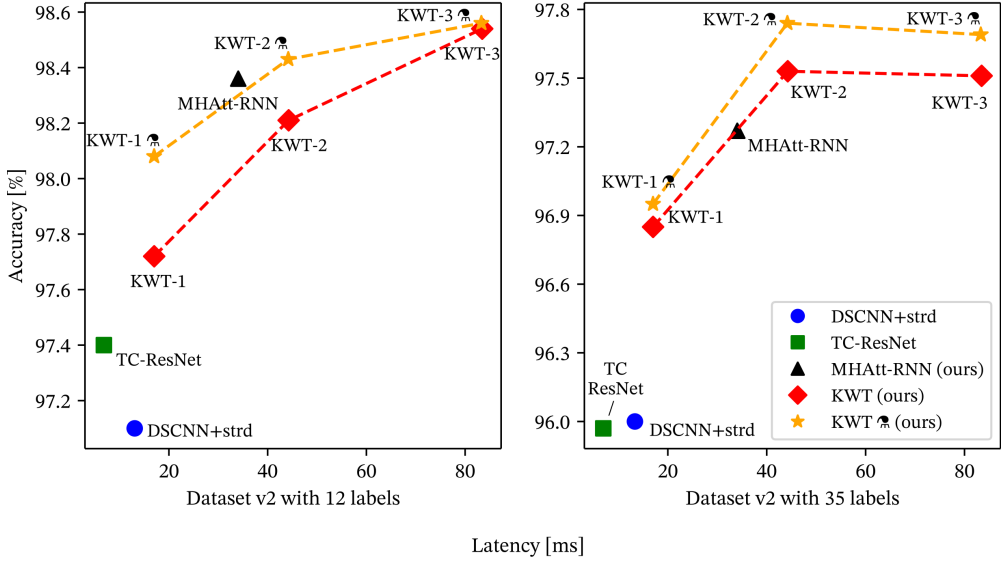


Figure 6: Latency and accuracy of processing a one-second input, on a single thread on a mobile phone.

- [2] Speech commands dataset v1., . URL http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz.
- [3] Speech commands dataset v2., . URL https://storage.googleapis.com/download.tensorflow.org/data/speechcommands_v0.02.tar.gz.
- [4] S. Adya, V. Garg, S. Sigtia, P. Simha, and C. Dhir. Hybrid transformer/ctc networks for hardware efficient voice triggering. *Proc. Interspeech 2020*, pages 3351–3355, 2020.
- [5] M. A. Alcorn and A. Nguyen. baller2vec: A Multi-Entity Transformer For Multi-Agent Spatiotemporal Modeling. *arXiv preprint arXiv:1609.03675*, 2021.
- [6] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.
- [8] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li. Developing Real-time Streaming Transformer Transducer for Speech Recognition on Large-scale Dataset. *arXiv preprint arXiv:arXiv:2010.11395*, 2020.
- [9] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha. Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices. *Proc. Interspeech 2019*, pages 3372–3376, 2019.

- [10] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.
- [11] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929*, 2018.
- [12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [13] D. Gope, G. Dasika, and M. Mattina. Ternary hybrid neural-tree networks for highly constrained iot applications. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 190–200, 2019. URL <https://proceedings.mlsys.org/paper/2019/file/a97da629b098b75c294dffdc3e463904-Paper.pdf>.
- [14] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented Transformer for Speech Recognition. *Proc. Interspeech 2020*, pages 5036–5040, 2020.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [16] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.
- [17] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL <http://arxiv.org/abs/1503.02531>.
- [18] M. Kumar, D. Weissenborn, and N. Kalchbrenner. Colorization Transformer. *arXiv preprint arXiv:2102.04432*, 2021.
- [19] J. Lin, K. Kilgour, D. Roblek, and M. Sharifi. Training keyword spotters with limited and synthesized speech data. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7474–7478. IEEE, 2020.
- [20] A. T. Liu, S.-W. Li, and H.-y. Lee. Tera: Self-supervised learning of transformer encoder representation for speech. *arXiv preprint arXiv:2007.06028*, 2020.
- [21] S. Majumdar and B. Ginsburg. MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition. *Proc. Interspeech 2020*, pages 3356–3360, 2020.

-
- [22] D. Neimark, O. Bar, M. Zohar, and D. Asselmann. Video Transformer Network. *arXiv preprint arXiv:2102.00719*, 2021.
 - [23] T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.
 - [24] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *Proc. Interspeech 2019*, pages 2613–2617, 2019.
 - [25] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo. Streaming Keyword Spotting on Mobile Devices. *Proc. Interspeech 2020*, pages 2277–2281, 2020.
 - [26] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. *arXiv preprint arXiv:2004.02984*, 2020.
 - [27] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.
 - [28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. *Advances in neural information processing systems*, 30, 2017.
 - [29] R. Vygon and N. Mikhaylovskiy. Learning Efficient Representations for Keyword Spotting with Triplet Loss. *arXiv preprint arXiv:2101.04792*, 2021.
 - [30] H. Wang, Z. Zhang, and S. Han. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. *arXiv preprint arXiv:2012.09852*, 2021.
 - [31] Y. Wang, H. Lv, D. Povey, L. Xie, and S. Khudanpur. Wake word detection with streaming transformers. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5864–5868. IEEE, 2021.
 - [32] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.
 - [33] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-Training With Noisy Student Improves ImageNet Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
 - [34] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. Tay, J. Feng, and S. Yan. Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet. *arXiv preprint arXiv:2101.11986*, 2021.

- [35] Y. Zhang, N. Suda, L. Lai, and V. Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.

Paper IV

Submitted. A shorter version of this paper was presented at FedKDD: International Joint Workshop on Federated Learning for Data Mining and Graph Analytics, 2024. Copyright 2024, the authors.

Towards Federated Learning with on-device Training and Communication in 8-bit Floating Point

BOKUN WANG¹, AXEL BERG^{2,3}, DURMUS ALP EMRE ACAR², CHUTENG ZHOU²

¹Texas A&M University, ²Arm, ³Lund University

Abstract: Recent work has shown that 8-bit floating point (FP8) can be used for efficiently training neural networks with reduced computational cost compared to training in FP32/FP16. In this work, we investigate the use of FP8 training in a federated learning context. This approach brings not only the usual benefits of FP8 which are desirable for on-device training at the edge, but also reduces client-server communication costs due to significant weight compression. We present a novel method for combining FP8 client training while maintaining a global FP32 server model and provide convergence analysis. Experiments with various machine learning models and datasets show that our method consistently yields communication reductions of at least 2.9x across a variety of tasks and models compared to an FP32 baseline to achieve the same trained model accuracy.

1 Introduction

A large amount of data is generated daily on personal smartphones and other devices at the edge. This data is very valuable for training machine learning models to provide services such as better voice recognition [18] or text completion [8]. However, the local data often carries sensitive personal information which needs to be protected for privacy reasons. Furthermore, communication of local data between billions of devices and data centers is expected to occupy lots of network bandwidth and transmission is costly in terms of power consumption, which is a primary concern for edge devices running on batteries.

Despite these constraints, it is still possible to train a model without having to transmit this local data using federated learning [23]. In federated learning, each local device performs training locally with their local data and update their local models. When it comes to communication, the central server receives local models from a subset of devices. The central server then aggregates these local models and transmits a new global model back to those devices for a model update. In this way, no local data is ever exposed during communication and the global model can learn from local data as communication goes on.

Since its inception, new techniques around federated learning have been proposed to reduce communication cost. The local models, albeit smaller than the local data, are still expensive to transmit via wireless communication and will be taxing on local devices' battery life if performed very frequently. One method to reduce communication cost is to quantize the

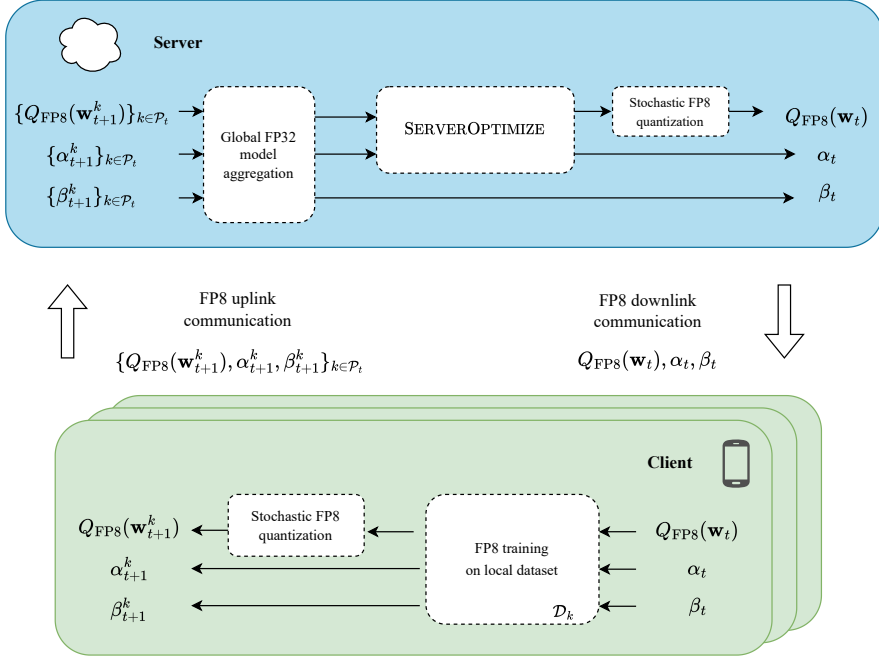


Figure 1: Overview of the proposed federated learning with local FP8 on-device training and weight quantization in both uplink and downlink communication. For time step t , the server sends quantized FP8 global model weights $Q_{FP8}(\mathbf{w}_t)$ and range parameters α_t, β_t for the weights and activations respectively. Each client k that participates in the training round then performs FP8 local training and sends back updated weight and range parameters. At the server, new global weights are obtained using global aggregation and an additional optimization step.

models before the communication occurs, and several works have shown that this can be done with limited reduction of model accuracy [7, 11, 26].

In this paper, we focus on the use of a new type of short floating point number format which has not yet been explored for federated learning: 8-bit floating point (FP8). An 8-bit floating point number is only $\frac{1}{4}$ of the length of a 32-bit floating point number. Therefore, it has smaller representation power and lower precision than full 32-bit floating point number format, but it offers great savings in terms of model storage and memory access. Computation can be greatly accelerated with the FP8 format because of significantly less bit-wise operations required compared to FP32/FP16. Application of FP8 number format to deep learning model training and inference is in a nascent stage but is widely expected to have fast growth.

Being a very efficient training datatype, FP8 is a good candidate for on-device training at the edge and the wide industrial support [24] behind it points to wide-spread real-world applications. A future scenario where edge devices can perform efficient on-device training

with native hardware support introduces a new class of federated learning problems. It also increases device heterogeneity in a federated learning setting, where participating devices and servers may have different levels of hardware support for FP8.

In this work, we introduce an implementation of federated learning with on-device FP8 training, which learns from quantized models effectively while being efficient in its communication and computing cost. A high-level overview of this method is shown in Figure 1. We summarize our main contributions as follows:

1. A novel method for combining local FP8 client training with an FP32 server model. The local FP8 training is simulated by quantization-aware training (QAT) and all communication between the server and the clients is done in FP8. Furthermore, we provide an additional optimization procedure for weight aggregation on the server that alleviates potential performance drops caused by quantization, without affecting communication costs.
2. We provide convergence analysis and motivation for the use of stochastic quantization for communication, but deterministic quantization for QAT. These design choices are further supported by experimental ablation studies.
3. We proved experiments on image and audio classification benchmarks, showing minimal loss in performance while obtaining large reductions in communication costs for both convolutional and transformer-based models.

2 Related Work

2.1 Federated Learning with Quantized Communication

Quantizing model weights is an effective way of reducing the communication cost of federated learning. For example, quantizing weights from 32 to 8 bits immediately reduces the number of communicated bits by 75% per training round. However, in practice the same functional performance is often not reached by the quantized model, because quantization can cause slower convergence of the training process. As a consequence, the communication reduction obtained to yield a particular performance threshold is often far from the ideal.

An important reason for the slower convergence obtained with quantized communication is that quantization of the model weights will introduce a bias term, which makes the server model a biased estimate of the average client model. In order to alleviate this problem, the use of stochastic rounding has been proposed [39, 6]. Hence, when aggregating the client

models at the server, the stochastic rounding errors tend to zero as the number of clients grows large. This method has also been shown to improve the learning process from a privacy perspective [38]. Similar work, He et al. [10], has shown that stochastic rounding in conjunction with non-linear quantization can reduce the number of communication rounds to reach convergence even more. Nevertheless, there is a limited amount of research on how to effectively combine quantized communication with low-precision client training.

Yoon et al. [37] investigated a similar setup as ours, i.e. low-precision local training with quantized communication, but only for uniform quantization schemes, such as INT4 and INT8. This leads to sub-optimal performance compared to full-precision training, since many neural networks are known to be sensitive to integer quantization, especially gradients during training are thought to require a bigger dynamic range [16]. For this reason, the industry is coalescing around FP8 for efficient training hardware [24]. This motivates an investigation of local training in low-precision floating point, which provides better training dynamics at the same communication cost.

2.2 FP8 Quantization for Neural Networks

While integer representations, have been widely adopted for neural network quantization for efficient inference, the use of FP8 remains relatively new in comparison and has been more focused on model training. A few recent works [34, 32, 29] have proposed centralized neural network training in FP8 with promising results. However, for some networks, a single FP8 representation was found to be insufficient for retaining accuracy in certain operations. Notably, the backwards pass through the network typically requires higher dynamic range than the forward pass. To this end, a binary interchange FP8 format that uses both E4M3 (4-bit exponent and 3-bit mantissa) and E5M2 (5-bit exponent and 2-bit mantissa) representations has been proposed, which allows for minimal accuracy drops compared to FP16 across a wide range of network architectures [24]. Concurrent work, Kuzmin et al. [16], proposed a similar solution, where the exponent bias is flexible and updated for each tensor during training, which allows for maintaining different dynamic ranges in different parts of the network.

An important factor in neural network quantization that is often overlooked, is that not all model architectures are equally sensitive to quantization, since the distribution of weights and activations are often architecture-dependent. As a consequence, a quantization scheme that works well for e.g. convolutional networks might not be suitable for attention-based models. Shen et al. [31] provided a thorough investigation of different quantization schemes for models on a wide variety of applications and found that FP8 quantization, with a combination of E4M3 and E3M4 in particular, was least detrimental to model performance. Similar results were found by Nikolic et al. [27] by dynamically adjusting the number of exponent and mantissa bits during training, which lead to large reductions in both memory

footprint and energy consumption. This makes FP8 training particularly interesting in a federated training setup, where the available hardware resources are often limited.

3 Method

3.1 Preliminaries

Consider the federated learning problem, where K clients update their local models by training on disjoint local datasets $\{\mathcal{D}_k\}_{k=1}^K$. Each client minimizes their own local objective functions $F_k(\mathbf{w}, Q, \alpha, \beta) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_k} [l(\mathbf{w}; \mathbf{x}, y, Q, \alpha, \beta)]$, where \mathbf{w} are the model parameters, Q is a quantization operator and l is the loss function. Furthermore, α and β are the per-tensor clipping values used for maintaining the dynamic range when quantizing weights and activations respectively. Henceforth, we denote quantized weights based on range α as $Q(\mathbf{w}; \alpha)$. In practice, each layer of the network has its own unique clipping parameters for both the weights and activations, but we omit this in our notation for readability.

We consider a modification of the standard federated averaging (FedAvg) [23] with quantized weights, where the objective is to find

$$\begin{aligned} \min_{\mathbf{w}} \quad & F(\mathbf{w}, Q, \alpha, \beta), \\ F(\mathbf{w}, Q, \alpha, \beta) = \quad & \sum_{k=1}^K \frac{n_k}{n} F_k(\mathbf{w}, Q, \alpha, \beta), \end{aligned} \tag{1}$$

where $n_k = |\mathcal{D}_k|$ is the number of training samples on the k :th device and $n = \sum_k n_k$ is the total number of training examples.

3.2 Floating Point Representation

A floating point number x with e exponent bits and m mantissa bits can be written as

$$x = (-1)^s 2^{p-b} \left(1 + \frac{d_1}{2} + \frac{d_2}{2^2} + \dots + \frac{d_m}{2^m} \right), \tag{2}$$

where $s \in \{0, 1\}$ is the sign bit, $d_i \in \{0, 1\}$ is the mantissa, $p \in \{0, 1, \dots, 2^e - 1\}$ is the exponent and b is the exponent bias. In addition, we assume that $p = 0$ is reserved for subnormal numbers, which allows an exact representation of 0 and other special values.

Compared to integer quantization, the quantization grid for floating point numbers is not uniform. Increasing the number of exponent bits results in a higher dynamic range, whereas

increasing the number of mantissa bits increases the precision. Therefore, give a fixed budget on the number of bits to allocate, there is a trade-off to be made between the two.

3.3 On-Device Quantization-Aware Training

Depending on the hardware support, on-device local training can be performed in native FP8 or using quantization-aware training (QAT), or a mix of the two. Native FP8 training is supported by the latest AI hardware in data centers such as Nvidia’s H100/H200 series of GPUs. There is significant industry support behind FP8 and it is only a matter of time for FP8 hardware support to arrive on edge devices.

For research purposes, we here resort to QAT, and follow the FP8 QAT method described in Kuzmin et al. [16], using per-tensor quantization for both model weights and activations, with one signed bit, and $m = 3$ and $e = 4$ bits for the mantissa and exponent respectively, as well as a flexible exponent bias that depends on learnable clipping parameters. QAT with both weights and activations quantization is a good way of simulating native FP8 training on supported hardware with low precision arithmetic. In our setting, we are not simulating the effect of gradient quantization which happens on FP8-enabled hardware. However, previous work, Kuzmin et al. [16], has shown that it is a good approximation to ignore its effect.

Let $\mathbf{x} = (x_1, \dots, x_d)^T \in \mathbb{R}^d$ denote an FP32 input tensor and $Q_{\text{det}} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ the FP8 deterministic quantization operator with a clipping parameter α , whose element-wise outputs are given by

$$Q_{\text{det}}(x_i; \alpha) = \begin{cases} -\alpha, & x_i \leq -\alpha, \\ s_i \left\lfloor \frac{x_i}{s_i} \right\rfloor, & \\ \alpha, & x_i \geq \alpha, \end{cases} \quad (3)$$

where $\lfloor \cdot \rfloor$ denotes rounding to the nearest integer. Here, $s_i = 2^{p_i}$ is a scale factor that is applied before and after rounding. The scale factor can be computed from p_i using

$$p_i = \begin{cases} \lfloor \log_2 |x_i| + b \rfloor - b - m, & \lfloor \log_2 |x_i| + b \rfloor > 1 \\ 1 - b - m, & \text{otherwise,} \end{cases} \quad (4)$$

where the exponent bias depends on the clipping value α as $b = 2^e - \log_2 \alpha + \log_2(2 - 2^{-m}) - 1$. At training time, α is first initialized using the maximum absolute value of each weight range, and then treated as a learnable parameter that is updated during training. Furthermore, the gradients of the non-differentiable rounding operators are computed using the straight-through estimator (STE) [3], i.e. $\frac{\partial \lfloor x_i \rfloor}{\partial x_i} = 1$, with a key exception being $\log_2 |x_i|$, which is treated as a constant following a similar approach as in Kuzmin et al.

Algorithm 1 LOCALUPDATE

Input: $\mathbf{w}, \alpha, \beta, Q, \mathcal{D}$, number of minibatches P

- 1: Set $\mathbf{w}_1 = \mathbf{w}, \alpha_1 = \alpha, \beta_1 = \beta$
 - 2: **for** $p = 1, \dots, P$ **do**
 - 3: Sample minibatch \mathcal{B}_p randomly from \mathcal{D}
 - 4: Do forward pass with Q on the minibatch \mathcal{B}_p
 - 5: Do backwards pass using STE and update $\mathbf{w}_{p+1}, \alpha_{p+1}, \beta_{p+1}$
 - 6: **end for**
 - 7: Return $\mathbf{w}_{P+1}, \alpha_{P+1}, \beta_{P+1}$
-

[16]. Activations are quantized using the same procedure, but with a separate clipping value β . A summary of the local on-device training procedure is given in Algorithm 1.

3.4 Unbiased Quantized Communication

When applying FP8 QAT to a federated learning scenario, an important aspect is the ability to reduce communication overhead by transferring weights between clients and the server using only 8 bits per scalar value. On client devices with hardware for FP8 mixed-precision training support, a copy of high-precision master weights [33] is present as in our QAT setup. Therefore, at the end of each communication round, the participating clients need to perform a hard reset of their master weights to the de-quantized FP8 values on a quantization grid. This approach allows for cost reduction in both the uplink and downlink communication.

At each communication round t , active clients \mathcal{P}_t will send the FP8-quantized weights to the central server together with the clipping parameters. However, to form an unbiased estimate of the average client weight, we need a different quantization operator. We therefore introduce stochastic quantization as

$$Q_{\text{rand}}(x_i; \alpha) = s_i \begin{cases} \left\lceil \frac{x_i}{s_i} \right\rceil & \kappa \leq \frac{x_i}{s_i} - \left\lfloor \frac{x_i}{s_i} \right\rfloor \\ \left\lfloor \frac{x_i}{s_i} \right\rfloor & \text{otherwise,} \end{cases} \quad (5)$$

for $-\alpha \leq x_i \leq \alpha$ and where κ is a Bernoulli random variable that takes the values 0 and 1 with equal probability. It is straightforward to verify that this randomized quantization is unbiased from a statistics point of view, i.e. for $-\alpha \leq x_i \leq \alpha$ we have $\mathbb{E}[Q_{\text{rand}}(x_i; \alpha)] = x_i$, while the deterministic quantization introduced earlier is biased¹.

¹The unbiasedness of stochastic quantization assumes a finite domain of the input variable, such that no clipping occurs. While clipping may occur for some values in practice, this does not affect the majority of the weights or activations. In order to simplify theoretical analysis, the assumption of no clipping is often made in the literature, see for example [19].

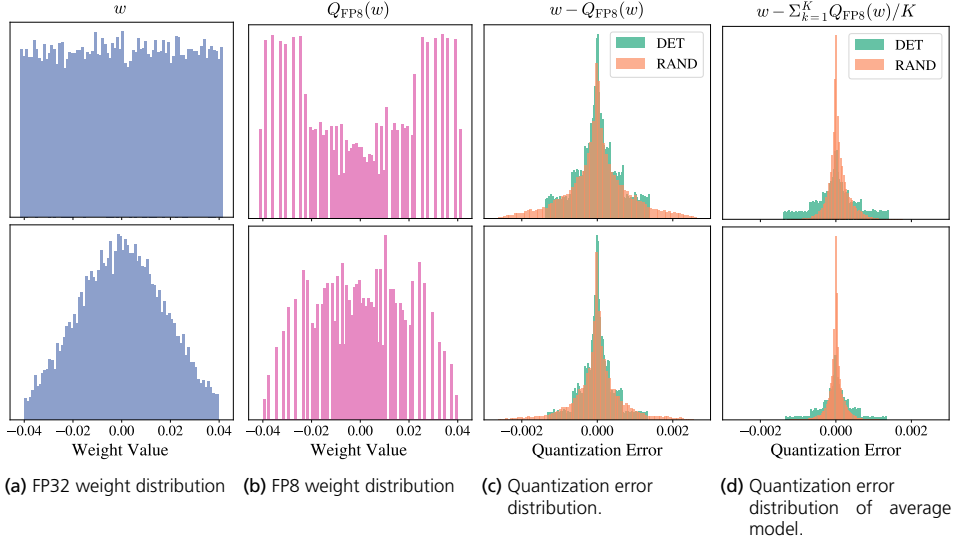


Figure 2: Examples of FP8 quantized weights using the E4M3 format and the corresponding quantization errors for different weight distributions. The range α of the quantization is calculated such that the entire distribution fits within the dynamic range of the FP8 representation. In general, deterministic quantization yields the lowest quantization error for any distribution. However, when weights are quantized independently on several devices (here we use $K = 10$) and averaged, stochastic quantization yields smaller errors. Top row: uniform weight distribution, which is a common initialization for convolutional layers. Bottom row: truncated normal distribution, which is commonly used to initialize the linear layers in Transformer models.

The quantized weights are then aggregated at the server using a weighted federated average as

$$\mathbf{w}_{t+1} \leftarrow \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} Q_{\text{rand}}(\mathbf{w}_{t+1}^k; \alpha_{t+1}^k), \quad (6)$$

where $m_t = \sum_{k' \in \mathcal{P}_t} n_{k'}$. The clipping values are also aggregated, but without quantization, since their contribution to the communication overhead is small relative to the weights. The aggregated weights are then quantized again to FP8 and transmitted to the next set of active clients with a new set of quantization parameters.

An illustrative example of weight quantization is shown in Figure 2a-d, where different weight distributions and the corresponding quantization errors are shown for both deterministic and stochastic quantization. For a given scalar weight, deterministic quantization always results in a lower quantization error, but for communication purposes we are more interested in the quantization error of the aggregated server weights. As can be seen in Figure 2d, stochastic quantization yields a lower error for the average model. For illustration purposes, we have here assumed that the weights are identical on each device, which is not the case in practice. Nevertheless, with stochastic quantization, the average quantized model will be an unbiased estimate of the average unquantized model.

Algorithm 2 SERVEROPTIMIZE

Input: $\{\mathbf{w}_t^k, \alpha_t^k, n_k\}_{k \in \mathcal{P}_t}, Q$

- 1: Compute $m_t = \sum_k n_k$
 - 2: Using gradient descent, update the weights as
$$\mathbf{w}_{t+1} \leftarrow \arg \min_{\mathbf{w}} \sum_k \frac{n_k}{m_t} \|Q(\mathbf{w}; \alpha_{t+1}) - Q(\mathbf{w}_t^k; \alpha_t^k)\|_2^2$$
 - 3: Set $S \leftarrow [\min_k \alpha_t^k, \max_k \alpha_t^k]$
 - 4: Using grid search, update the range parameters as
$$\alpha_{t+1} \leftarrow \arg \min_{\alpha \in S} \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \|Q(\mathbf{w}_{t+1}; \alpha) - Q(\mathbf{w}_t^k; \alpha_t^k)\|_2^2$$
 - 5: Return $\mathbf{w}_{t+1}, \alpha_{t+1}$
-

3.5 Server-Side Optimization (SERVEROPTIMIZE)

The standard federated average of the weights in the un-quantized scenario corresponds to the minimization of weighted mean squared error (MSE) between the server parameters and the client parameters. However, when the server parameters are quantized before transmission to the clients, this property no longer holds. We therefore propose a modification to the server-side model aggregation, where the MSE is explicitly minimized. This can be done without communication overhead since all computations are done on the server. Another advantage of this approach is that it leverages the computing power of the server to do more optimization, since the server typically possesses more computing power than a client device.

At time step t , we perform model/parameters aggregation to obtain $\mathbf{w}_{t+1}, \alpha_{t+1}$ for the next communication round by minimizing the following mean-squared error (MSE).

$$\mathbf{w}_{t+1}, \alpha_{t+1} = \arg \min_{\mathbf{w}, \alpha} \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \|Q_{\text{rand}}(\mathbf{w}; \alpha) - Q_{\text{rand}}(\mathbf{w}_t^k; \alpha_t^k)\|_2^2.$$

Note that when there is no communication quantization, the closed-form solution to SERVEROPTIMIZE is the federated average $\mathbf{w}_{t+1} = \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \mathbf{w}_t^k$. Since the problem above has no closed-form solution for the quantized communication case, we use the *alternating minimization* approach to optimize \mathbf{w} and α . First, we optimize the model weights \mathbf{w} , while fixing α to $\alpha_{t+1} = \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \alpha_t^k$. This is done by minimizing the MSE as

$$\mathbf{w}_{t+1} = \arg \min_{\mathbf{w}} \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \|Q_{\text{rand}}(\mathbf{w}; \alpha_{t+1}) - Q_{\text{rand}}(\mathbf{w}_t^k; \alpha_t^k)\|_2^2. \quad (7)$$

This minimization can be done using gradient descent with a fixed number of iterations

Next, we aim to optimize α while fixing \mathbf{w} to \mathbf{w}_{t+1} . However, minimizing the MSE with respect to α using gradient descent would require access to $\frac{\partial s_i}{\partial \alpha}$, which is non-differentiable

Algorithm 3 FP8FedAvg-UQ, FP8FedAvg-UQ+

Input: $\mathbf{w}_1, \alpha_1, \beta_1, Q_{\text{det}}, Q_{\text{rand}}$

- 1: **for** $t = 1, \dots, T$ **do**
- 2: Sample a set $\mathcal{P}_t \in [K]$ of P active devices
- 3: **for** each client $k \in \mathcal{P}_t$ **do**
- 4: Receive $Q_{\text{rand}}(\mathbf{w}_t; \alpha_t), \alpha_t, \beta_t$ from server
- 5: $\{\mathbf{w}_{t+1}^k, \alpha_{t+1}^k, \beta_{t+1}^k\} \leftarrow \text{LOCALUPDATE}(\mathbf{w}_t, Q_{\text{det}}; \alpha_t, \beta_t, \mathcal{D}_k)$
- 6: Send $Q_{\text{rand}}(\mathbf{w}_{t+1}^k; \alpha_{t+1}^k), \alpha_{t+1}^k, \beta_{t+1}^k$ to server
- 7: **end for**
- 8: Compute $m_t = \sum_{k \in \mathcal{P}_t} n_k$
- 9: Compute $\beta_{t+1} \leftarrow \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \beta_{t+1}^k$
- 10: Compute $\mathbf{w}_{t+1} \leftarrow \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} Q_{\text{rand}}(\mathbf{w}_{t+1}^k; \alpha_{t+1}^k)$
- 11: Compute $\alpha_{t+1} \leftarrow \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \alpha_{t+1}^k$
- 12: $\{\mathbf{w}_{t+1}, \alpha_{t+1}\} \leftarrow \text{SERVEROPTIMIZE}(\{\mathbf{w}_{t+1}^k, \alpha_{t+1}^k, n_k\}_{k \in \mathcal{P}_t}, Q_{\text{rand}})$
- 13: **end for**
- 14: Evaluate on $\mathbf{w}_{T+1}, \alpha_{T+1}, \beta_{T+1}$

at multiple points and therefore highly numerically unstable. We instead perform a grid search over a fixed set of clipping values uniformly distributed in $S = [\min_{k \in \mathcal{P}_t} \alpha_t^k, \max_{k \in \mathcal{P}_t} \alpha_t^k]$ as

$$\alpha_{t+1} = \arg \min_{\alpha \in S} \sum_{k \in \mathcal{P}_t} \frac{n_k}{m_t} \|Q_{\text{rand}}(\mathbf{w}_{t+1}; \alpha) - Q_{\text{rand}}(\mathbf{w}_t^k; \alpha_t^k)\|_2^2. \quad (8)$$

The SERVEROPTIMIZE routine is given in Algorithm 2, which takes place completely on the server and can be used as an optional step in order to improve aggregation of the model weights.

3.6 Overall algorithm

A summary of our proposed FP8 FedAvg with unbiased communication (FP8FedAvg-UQ) method is presented in Algorithm 3, where the optional server-optimization step (UQ+) corresponds to replacing the quantized federated averaging of the weight and range parameters with our two-step MSE minimization optimization in Equations (7) and (8). It is important to note that our method involves two different quantization operators. On-device QAT uses a deterministic and biased quantizer, while the communication part adopts its stochastic counterpart which is unbiased. In the next section, we will give a convergence

analysis result for FP8FedAvg-UQ and show that these design choices are well-motivated from a theoretical perspective.

4 Convergence analysis and theoretical motivations

We briefly state our main convergence theorem here and refer to the Appendix for formal assumption definitions and proof. Please note that we make the simplifying assumption to only consider weight quantization in our proof, which is standard for this type of theoretical analysis.

Theorem 4.1 (Convergence of FP8FedAvg-UQ). *For convex and L -smooth federated losses in (1) with G -bounded unbiased stochastic gradients using an FP8 deterministic quantization method during training and an FP8 unbiased quantization method with bounded scales for model communication, the objective gap $\mathbb{E} [F(Q_{\text{rand}}(\mathbf{w}_\tau)) - F(\mathbf{w}_*)]$ decreases at a rate of*

$$O\left(\underbrace{\frac{\Delta^2 + G^2 U}{\sqrt{TU}}}_{\tau_1} + \underbrace{\frac{UG^2 L}{T}}_{\tau_2} + \underbrace{\frac{GU^{2.5} S \sqrt{d} L}{\sqrt{T}}}_{\tau_3} + \underbrace{S \sqrt{d} G}_{\tau_3}\right),$$

where $\Delta = \|\mathbf{w}_1 - \mathbf{w}_*\|_2$ is the initial and optimal model difference, τ is uniformly sampled from $\{1, 2, \dots, T\}$, T is the number of rounds, U is the total number of updates done in each round, the quantization scales s_i are uniformly bounded by S , \mathbf{w}_1 is the initial model, and \mathbf{w}_* is an optimal solution of (1).

Proof Structure. The proof builds on upper bounding a drift quantity similar to the one defined in Karimireddy et al. [13] as²,

$$V_t = \frac{1}{KU} \sum_{k \in [K]} \sum_{u \in [U]} E \left\| Q_{\text{rand}}(\mathbf{w}_t) - Q_{\text{det}}(\mathbf{w}_{t,u}^k) \right\|_2^2.$$

Note that if local models diverge, we get a higher V_t . If there is no quantization and local models converge, we get $V_t = 0$. We focus on V_t and the server model \mathbf{w} in a single communication round and give the following Lemma as,

Lemma 1. *For a setting that satisfies the assumptions listed in Theorem 4.1, we have,*

$$V_t \leq 18U^3 S \sqrt{d} G \eta_t + 9U^2 \eta_t^2 G^2, \quad (9)$$

$$E \|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 - E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2 \leq +\eta_t L U V_t + 2S \sqrt{d} G U \eta_t + \eta_t^2 U^2 G^2 - 2U \eta_t E (F(Q_{\text{rand}}(\mathbf{w}_t)) - F(\mathbf{w}_*)) \quad (10)$$

²See Appendix C.2 on $\mathbf{w}_{t,u}^k$ definition for inactive devices.

where η_t is the learning rate. By combining the equations with proper coefficients, we get a telescoping sum on the difference between the server model and the optimal model, as well as the accumulation of the loss with respect to the optimal model. This leads to Theorem 4.1 and we provide the full proof in Appendix C.

Remark 1. \mathcal{T}_1 is a term similar to SGD convergence where it decreases with $O\left(\frac{1}{\sqrt{T}}\right)$ and depends on the bound on the second moment of stochastic gradient G , the smoothness L , as well as the ℓ_2 -distance between the initial model w_1 and the optimal solution w_* .

Remark 2. \mathcal{T}_2 and \mathcal{T}_3 are terms that arise due to quantization. Due to (4) and the definition of S , the terms \mathcal{T}_2 and \mathcal{T}_3 exponentially decay when the number of mantissa bits m increases, i.e., $\mathcal{T}_2 \propto 2^{-m}$, $\mathcal{T}_3 \propto 2^{-m}$.

Remark 3. We emphasize that unbiased quantization during communication is crucial. In the case of biased communication, the convergence proof does not hold and one can construct even diverging cases [5] for FedAvg. To ensure convergence for biased communication, we need more sophisticated techniques such as error feedback [30].

Remark 4. Deterministic quantization is used during training. We bound the norm of QAT quantization error in the proof. Since deterministic quantization has a smaller error norm than stochastic one, this motivates us to use deterministic quantization during training.

As we shall see in the next section, we observe strictly worse results if we use stochastic quantization during training or deterministic quantization during model transmission in our experiments, which aligns with the remarks above.

5 Experiments and Ablation Studies

We test our proposed method on two different tasks: image classification and keyword spotting. For each task, we use two different models and perform experiment using both an i.i.d. and a non-i.i.d. dataset split across clients. In the i.i.d. setup, the dataset is split randomly across the set of clients. In the non-i.i.d. setup, we simulate a more realistic heterogeneity across clients which is specific to each dataset.

5.1 Datasets and models

Image classification. For image classification, we use the CIFAR10 and CIFAR100 datasets [14], which consist of 60 000 examples of 32x32 color images divided into 10 and 100 different classes respectively. For this task, we adopt two different convolutional networks: 1)

LeNet [17], which has 800K parameters, and ResNet18 [9], with 11M parameters. For the ResNet model, we replace batch normalization after the convolutional layers with Group-Norm [36], since this is known to work better in a federated setting with skewed data distributions [12].

In the (independent and identical and distributed) i.i.d. scenario, we use $K = 100$ clients, a participation rate of $C = 0.1$ in each round and train for $R = 1000$ rounds with a local batch size of $B = 50$, where each client trains for 5 local epochs. In the non-i.i.d. image classification setting we sample the local datasets from a Dirichlet distribution with a concentration parameter of 0.3. Furthermore, we use SGD as the local optimizer with a constant learning rate of 0.1, weight decay of 0.001 and random cropping and horizontal flipping for data augmentation.

Keyword spotting. In order to apply our method to a more realistic federated learning task and more advanced model architectures, we resort to keyword spotting, where the task is to classify short snippets of audio as words in a small dictionary. For this, we use the Google SpeechCommands v2 dataset, which consists of 105 000 one-second recordings belonging to one of 35 classes [35]. Examples of classes in the dictionary include short words like “go”, “yes” and “on”.

For the keyword spotting task, we train two different models: MatchboxNet3x1x64 [22], which uses 1D time-channel separable convolutions with skip-connections, as well as the Keyword Transformer (KWT-1) [4] model, which consists of sequential transformer layers with time-domain self-attention. These two models have 350K and 450K parameters respectively and they both use mel-frequency cepstral coefficients (MFCCs) as input features.

A known problem when training transformers is that SGD optimization requires many iterations and may fail to converge to a good solution [15]. Due to the relatively small number of local updates in a training round, SGD is therefore not a suitable optimization method for transformers in this setup. To achieve better training convergence, we instead use the momentum-based AdamW [21] as local optimizer for the clients, where the states of the local optimizer for each active client are reset at the start of each communication round. Furthermore, we use an initial learning rate of 0.001 with a cosine decay scheduler and a weight decay of 0.1. For data augmentation, we apply SpecAugment [28] to the MFCCs during training.

When training in the i.i.d. scenario, we use the same setup for keyword spotting as for image classification. However, since the SpeechCommands datasets provides speaker identity as well, we can simulate a more realistic heterogeneity for the non-i.i.d. dataset split. We therefore use the setup proposed in [20], such that all recordings from a given speaker are assigned to a single client. This results in a total of $K = 2112$ clients. In order to get a similar number of total training steps as in the i.i.d scenario, we reduce the participation rate to $C = 0.01$, the number of rounds to $R = 500$ and the number of local gradient

updates to 50.

5.2 Mixed Precision Quantization Implementation

Since most of the bandwidth during the communication phase between the server and clients is used for transmitting the model weights in the fully connected and convolutional layers, we focus on learning FP8 representations for these. Hence, we simulate a mixed-precision training scenario where these parameters are trained in FP8, whereas bias parameters and normalization layer parameters (GroupNorm for convolutional networks and LayerNorm for transformers) are trained in FP32, since initial experiments showed these to be sensitive to quantization. Note that this only results in a negligible impact on the client-server communication, since these parameters account for less than 2 % of the total parameter count in the models.

An illustration of how mixed-precision training can be simulated using QAT is shown in Figure 3. The diagram illustrates the necessary operations for quantizing most layers in the networks we use. A notable exception is the self-attention layers in the transformer model. In these layers, we perform calculations of the keys, queries and values, as well as calculation of the dot-product attention scores in FP8. Softmax normalization of the attention scores is done in FP32, similar to how it is done for other activation functions.

For all experiments, we have used $m = 3$ and $e = 4$ bits for the mantissa and exponent respectively. In addition, when performing server-side optimization of weight aggregation, we use 5 gradient descent steps when optimizing for the quantized weights in Equation (7), where the learning rate was selected using grid search in $\{0.01, 0.1, 1\}$, and 50 grid points when optimizing the range values in Equation (8).

5.3 Results

The results are shown in Table 3, where we present the final centralized test accuracy and standard deviation across three random seeds, as well as the average communication gain compared to an FP32 baseline. In order to compare communication costs, we do not pick a common accuracy threshold for all methods, but instead calculate the gains individually for each method as the reduction in communicated bytes compared to FP32 training at the maximum accuracy reached by both FP32 and FP8.

In Table 1 it can be seen that for most datasets and methods, FP8-FedAvg-UQ achieves similar test accuracy as the FP32 baseline, which results in communication gains of 4.2x on average, and for some experiments larger than 9x. Note that even though FP8 quantization sometimes results in a small accuracy drop, a large communication gain is still possible due

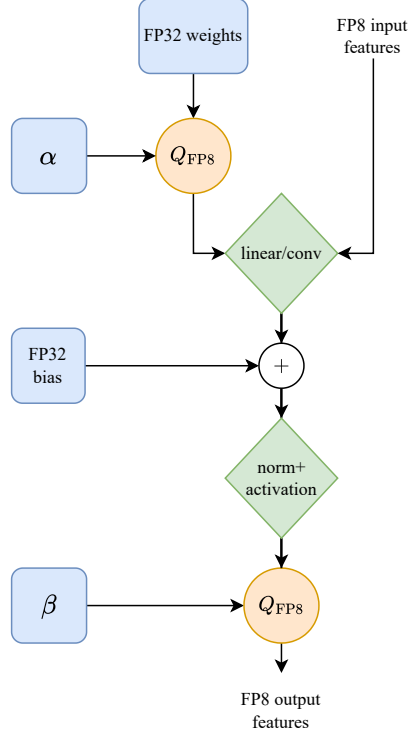


Figure 3: Illustration of the QAT method for a linear/convolutional layer. FP32 master weights are quantized using the range parameters α , before being applied to the already quantized input. We then apply addition of bias, an optional normalization layer and the activation function in FP32, before quantizing to FP8 again using the range parameters β . In a real application, these operations could be performed using mixed-precision hardware.

to less data being transferred in each communication round. However, in certain cases, for example when applying FP8 quantization to LeNet on CIFAR100, accuracy increases significantly compared to the FP32 baseline. For these experiments, we observed that FP8 quantization to some extent prevented overfitting to the local client datasets, and thereby acted as a regularizer. This effect of quantization has been observed in other studies as well [2].

Table 1 further shows results for i.i.d. and a more realistic non-i.i.d. settings. Here, we see an expected accuracy drop when moving from the i.i.d. to the non-i.i.d. setting, yet our method shows gains in both scenarios. We note that our method could potentially be combined with more advanced federated learning algorithms in heterogenous settings to improve accuracy levels in the non-i.i.d. setting as well.

Finally, we note that the server-side optimization in most scenarios yields additional performance improvements, with an average gain of 4.5x. The impact of this additional optimization is most notable when the FP8 quantization results in an accuracy drop com-

Table 1: Final test accuracy and communication gain compared to FP32 FedAvg for our proposed methods on CIFAR10/CIFAR100 and Google SpeechCommands.

Model	Setting	FP32 FedAvg	FP8 FedAvg-UQ	FP8 FedAvg-UQ+
CIFAR10				
LeNet	i.i.d.	$82.1 \pm 0.1 / 1\times$	$82.0 \pm 0.1 / 4.1\times$	$82.2 \pm 0.3 / 4.7\times$
	Dir(0.3)	$77.1 \pm 0.4 / 1\times$	$77.3 \pm 0.9 / 3.9\times$	$77.7 \pm 0.5 / 3.9\times$
ResNet18	i.i.d.	$92.0 \pm 0.1 / 1\times$	$91.1 \pm 0.2 / 3.4\times$	$92.0 \pm 0.1 / 3.9\times$
	Dir(0.3)	$85.5 \pm 0.5 / 1\times$	$87.4 \pm 0.7 / 5.2\times$	$87.5 \pm 0.5 / 5.2\times$
CIFAR100				
LeNet	i.i.d.	$43.0 \pm 0.3 / 1\times$	$44.8 \pm 0.4 / 6.0\times$	$44.9 \pm 0.5 / 6.0\times$
	Dir(0.3)	$38.3 \pm 0.7 / 1\times$	$41.1 \pm 0.3 / 9.1\times$	$41.3 \pm 0.7 / 9.5\times$
ResNet18	i.i.d.	$64.6 \pm 0.3 / 1\times$	$64.0 \pm 0.2 / 3.5\times$	$64.6 \pm 0.1 / 4.0\times$
	Dir(0.3)	$56.1 \pm 0.7 / 1\times$	$55.4 \pm 0.6 / 3.6\times$	$55.5 \pm 0.6 / 3.6\times$
SpeechCommands				
MatchboxNet	i.i.d.	$91.5 \pm 0.3 / 1\times$	$90.0 \pm 0.4 / 3.5\times$	$90.8 \pm 0.4 / 3.4\times$
	speaker-id	$79.6 \pm 0.7 / 1\times$	$75.4 \pm 0.6 / 3.1\times$	$77.0 \pm 1.3 / 3.3\times$
KWT-1	i.i.d.	$91.4 \pm 0.4 / 1\times$	$89.2 \pm 0.3 / 2.3\times$	$90.7 \pm 0.2 / 2.9\times$
	speaker-id	$83.2 \pm 0.2 / 1\times$	$79.6 \pm 0.5 / 2.9\times$	$82.4 \pm 0.8 / 3.7\times$
Average gain		$1\times$	$4.2\times$	$4.5\times$

pared to FP32. On the contrary, when there is no accuracy loss, the improvement due to server optimization is smaller. Overall, the quantized communication in combination with server-side optimization results in communication gains greater than $2.9\times$ compared to FP32 across all tasks and models.

5.4 Ablation studies

Next, we ablate the use of deterministic and stochastic quantization in order to validate our design choices. Table 2 shows the server test accuracy when using the two different quantization methods for the on-device QAT training. In general, we observe that performing local QAT does not severely impact the accuracy of the global model. In some cases, it can even increase test accuracy, which is the case for LeNet on CIFAR100, which we attribute to the quantization noise acting as a regularizer during training. Overall, deterministic quantization works slightly better, which can also be understood intuitively, since in each forward pass through the network, deterministic quantization results in a smaller quantization error. We refer to Appendix B for more details about QAT convergence.

In Table 3, we ablate the effect of deterministic and stochastic quantization in server-device communication, and it is clear that stochastic quantization results in both higher accuracy and gain. This is in agreement with Remark 3, which shows that stochastic quantization

Table 2: Final test accuracy on CIFAR10 and CIFAR100 (i.i.d.) for deterministic/stochastic QAT without quantized communication compared to FP32.

Model	FP32 FedAvg	det. QAT	rand. QAT
CIFAR10			
LeNet	82.1 ± 0.1	82.1 ± 0.2	82.0 ± 0.2
ResNet18	92.0 ± 0.1	91.9 ± 0.2	91.8 ± 0.3
CIFAR100			
LeNet	43.0 ± 0.3	44.4 ± 0.5	43.7 ± 0.6
ResNet18	64.6 ± 0.3	64.5 ± 0.1	63.5 ± 0.5

Table 3: Final test accuracy on CIFAR10 and CIFAR100 (i.i.d.) for different quantized communication (CQ) methods with deterministic QAT compared to FP32.

Model	FP32 FedAvg	det. CQ	rand. CQ
CIFAR10			
LeNet	82.1 ± 0.1	80.1 ± 0.1	82.0 ± 0.1
ResNet18	92.0 ± 0.1	91.1 ± 0.1	91.7 ± 0.2
CIFAR100			
LeNet	43.0 ± 0.3	38.0 ± 0.4	44.8 ± 0.4
ResNet18	64.6 ± 0.3	62.5 ± 0.9	64.0 ± 0.2

is important for the convergence of our algorithm. In addition, we show the server test accuracy as a function of communication cost for different methods in Figure 4. Here we can clearly see the gain arising from quantized communication, as well as the benefits of stochastic quantization and server-side optimization.

6 Conclusions and Future Work

In this work, we show that on-device FP8 QAT training combined with quantized communication can be integrated into a federated learning setting with a well-designed algorithm. Our results show that this can be done with minimal drop in model predictive performance, while obtaining large savings in terms of communication cost. This opens up a wide range of possibilities in terms of exploiting client heterogeneity. For example, it allows for combining devices with different computational capabilities, which could involve training with different levels of precision in different clients. The design of our algorithm is well-motivated by theory. We show results for various different models and datasets, and ablation studies to validate design choices.

Furthermore, our paper has important practical implications for machine learning hardware

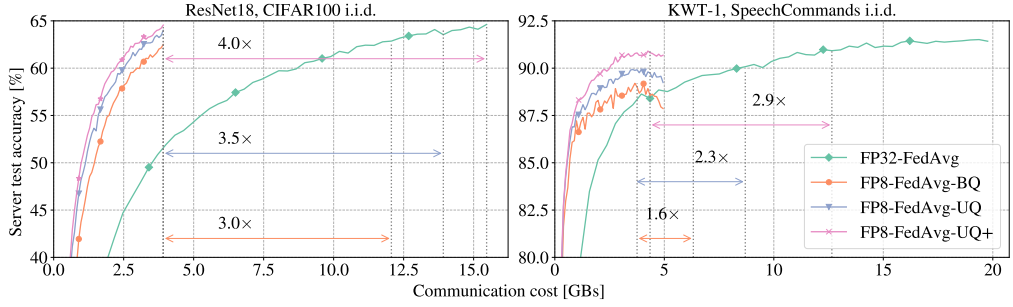


Figure 4: Server test accuracy versus communication cost for FP32 FedAvg, and FP8 QAT with biased (BQ)/unbiased (UQ) quantization for communication, and server-side optimization (UQ+). For each method, the communication gain compared to FP32 has been highlighted with arrows.

and system design, our results suggest that both deterministic and stochastic modes need to be supported by hardware FP8 quantizer for better training convergence in a distributed or federated setting.

Finally, since the use of low-precision number formats is orthogonal to the optimization method itself, our proposed method may be extended beyond standard federated averaging. FP8 local training may therefore be combined with more advanced optimization techniques that deal with problems such as client drift. We leave this as future work and hope our work will inspire the wider research community to explore different combinations of reduced precision floating point number formats in a federated learning setting.

References

- [1] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.
- [2] M. AskariHemmat, R. A. Hemmat, A. Hoffman, I. Lazarevich, E. Saboori, O. Mastrogiorgio, S. Sah, Y. Savaria, and J.-P. David. Qreg: On regularization effects of quantization. *arXiv preprint arXiv:2206.12372*, 2022.
- [3] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [4] A. Berg, M. O’Connor, and M. T. Cruz. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In *Proc. Interspeech 2021*, pages 4249–4253, 2021. doi: 10.21437/Interspeech.2021-1286.

-
- [5] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276):1–50, 2023.
 - [6] P. S. Bouzinis, P. D. Diamantoulakis, and G. K. Karagiannidis. Wireless quantized federated learning: a joint computation and communication design. *IEEE Transactions on Communications*, 2023.
 - [7] K. Gupta, M. Fournarakis, M. Reisser, C. Louizos, and M. Nagel. Quantization robust federated learning for efficient inference on heterogeneous devices. *TMLR*, 2023.
 - [8] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.
 - [9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
 - [10] Y. He, H.-P. Wang, M. Zenk, and M. Fritz. Cossqd: Communication-efficient federated learning with a simple cosine-based quantization. *arXiv preprint arXiv:2012.08241*, 2020.
 - [11] R. Hönig, Y. Zhao, and R. Mullins. Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning. In *International Conference on Machine Learning*, pages 8852–8866. PMLR, 2022.
 - [12] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.
 - [13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.
 - [14] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
 - [15] F. Kunstner, J. Chen, J. W. Lavington, and M. Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=a65YK0cqH8g>.
 - [16] A. Kuzmin, M. Van Baalen, Y. Ren, M. Nagel, J. Peters, and T. Blankevoort. Fp8 quantization: The power of the exponent. *Advances in Neural Information Processing Systems*, 35:14651–14662, 2022.

- [17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [18] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6341–6345. IEEE, 2019.
- [19] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein. Training quantized nets: A deeper understanding. *Advances in Neural Information Processing Systems*, 30, 2017.
- [20] X.-C. Li, J.-L. Tang, S. Song, B. Li, Y. Li, Y. Shao, L. Gan, and D.-C. Zhan. Avoid Overfitting User Specific Information in Federated Keyword Spotting. In *Proc. Interspeech 2022*, pages 3869–3873, 2022.
- [21] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.
- [22] S. Majumdar and B. Ginsburg. MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition. In *Proc. Interspeech 2020*, pages 3356–3360, 2020.
- [23] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [24] P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu, et al. Fp8 formats for deep learning. *arXiv:2209.05433*, 2022.
- [25] Y. Nesterov et al. *Lectures on convex optimization*, volume 137. Springer.
- [26] R. Ni, Y. Xiao, P. Meadowlark, O. Rybakov, T. Goldstein, A. T. Suresh, I. L. Moreno, M. Chen, and R. Mathews. Fedagt: Accurate quantized training with federated learning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6100–6104. IEEE, 2024.
- [27] M. Nikolic, E. Torres Sanchez, J. Wang, A. Hadi Zadeh, M. Mahmoud, A. Abdelhadi, K. Ibrahim, and A. Moshovos. Schrodinger’s fp training neural networks with dynamic floating-point containers. *Proceedings of Machine Learning and Systems*, 6: 60–73, 2024.
- [28] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Proc. Interspeech 2019*, pages 2613–2617, 2019.

-
- [29] H. Peng, K. Wu, Y. Wei, G. Zhao, Y. Yang, Z. Liu, Y. Xiong, Z. Yang, B. Ni, J. Hu, et al. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023.
- [30] P. Richtárik, I. Sokolov, and I. Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34:4384–4396, 2021.
- [31] H. Shen, N. Mellempudi, X. He, Q. Gao, C. Wang, and M. Wang. Efficient post-training quantization with fp8 formats. *Proceedings of Machine Learning and Systems*, 6:483–498, 2024.
- [32] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. *NeurIPS*, 32, 2019.
- [33] A. Vishwanath. Mixed-precision training techniques using tensor cores for deep learning. <https://developer.nvidia.com/blog/video-mixed-precision-techniques-tensor-cores-deep-learning/>, 2019. Accessed: 2019-01-30.
- [34] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.
- [35] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.
- [36] Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [37] J. Yoon, G. Park, W. Jeong, and S. J. Hwang. Bitwidth heterogeneous federated learning with progressive weight dequantization. In *International Conference on Machine Learning*, pages 25552–25565. PMLR, 2022.
- [38] Y. Youn, Z. Hu, J. Ziani, and J. Abernethy. Randomized quantization is all you need for differential privacy in federated learning. In *Federated Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities*, 2023.
- [39] S. Zheng, C. Shen, and X. Chen. Design and analysis of uplink and downlink communications for federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2150–2167, 2020.

Appendix

For FedAvg convergence proof, our analysis builds on [13, 19, 1]. [13, 1] focus on debiasing the local losses in a standard non-quantized federated learning setting. Differently, we show convergence using quantization aware training in federated learning. We can further extend our analysis to use the sophisticated debiasing methods for better heterogeneity control. Li et al. [19] proves convergence of different quantization aware training schemes in a centralized non-federated setting. Differently, we give convergence of quantization aware training in a distributed federated learning setting. Additionally, we give a proof for more general non-uniform quantization grids such as FP8, which is different from the uniform quantization consideration in [19].

A Quantization Function

Definition A.1 (Quantization). For an unquantized number x , we define the quantization of x as

$$Q_{\text{rand}}(x) = s \begin{cases} \left\lceil \frac{x}{s} \right\rceil & p \leq \frac{x}{s} - \left\lfloor \frac{x}{s} \right\rfloor \\ \left\lfloor \frac{x}{s} \right\rfloor & \text{otherwise,} \end{cases}, \quad Q_{\text{det}}(x) = s \left\lfloor \frac{x}{s} \right\rfloor,$$

where $p \in [0, 1]$ is a random variable.

We omit the parameter of quantization for the sake of simplicity in the notation. We overload the notation and define quantization of a vector $\mathbf{x} \in \mathbb{R}^d$ as the element-wise quantization of the vector, $Q(\mathbf{x}) = [Q([\mathbf{x}]_1), Q([\mathbf{x}]_2), \dots, Q([\mathbf{x}]_d)]^T$

Let's define the quantization error.

Definition A.2 (Quantization Error). Let $r_Q(\mathbf{w}) = Q(\mathbf{w}) - \mathbf{w}$.

Note that if $Er_Q(\mathbf{w}) = \mathbf{0}$, we have an unbiased quantization as in our model transmission where expectation is over the randomness of the quantization.

Note that we simplified the definition by ignoring the quantization based learnable parameters such as α and β in our proof. Hence, we redefine them here.

B Convergence Analysis of Quantization-aware Training (QAT)

As a warmup, we provide the convergence analysis of QAT training on a single machine, similar to the one in [19].

We want to find a *quantized* model that solves $\min_{\mathbf{w} \in \mathbb{R}^d} F(\mathbf{w})$. We start with an unquantized model as \mathbf{w}_1 and use QAT training as $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla F(Q(\mathbf{w}_t); \xi_t)$ where η_t is learning rate and ξ_t controls randomness of SGD at iterate t . Let us define the best model as $\mathbf{w}_* = \arg \min_{\mathbf{w}} F(\mathbf{w})$.

Our analysis is based on the following assumptions on the objective function F .

Assumption 1 (Convexity). We assume that F is differentiable and convex, i.e.,

$$-\langle \nabla F(\mathbf{x}), \mathbf{x} - \mathbf{y} \rangle \leq -F(\mathbf{x}) + F(\mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y}.$$

Assumption 2 (Bounded Unbiased Gradients). We assume the gradients are unbiased and bounded.

$$E_{\xi}[\nabla F(\mathbf{x}; \xi)] = \nabla F(\mathbf{x}), \quad E_{\xi} \|\nabla F(\mathbf{x}; \xi)\|_2^2 \leq G^2 \quad \forall \mathbf{x}.$$

where ξ defines the randomness due to stochastic gradient estimator. The algorithm draws $\nabla F(\mathbf{x}; \xi)$ instead of $\nabla F(\mathbf{x})$.

Assumption 3 (Bounded Quantization Scales). We assume the scales s_i are uniformly upper bounded during the training by a constant S .

Next, we provide an upper bound on the quantization error.

Lemma 2. If assumption 3 holds, we have,

$$E \|r_Q(\mathbf{w})\|_2 \leq \sqrt{d}S$$

Proof. Each dimension of $r_Q(\mathbf{w})$ can be at max S . We have d dimensions. Hence, $E \|r_Q(\mathbf{w})\|_2 \leq \sqrt{d}S$ \square

We can then prove the following lemma on the t -th iteration of QAT training.

Lemma 3 (QAT step update). If assumptions 1, 2, and 3 hold and $\eta_t = \frac{1}{\sqrt{T}}$, we have,

$$E [F(Q(\mathbf{w}_t)) - F(\mathbf{w}_*)] \leq \frac{\sqrt{T}}{2} [-E \|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 + E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2] + G\sqrt{d}S + \frac{1}{2\sqrt{T}}G^2$$

Proof. Based on the update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla F(Q(\mathbf{w}_t); \xi_t)$ in the t -th iteration of QAT

training,

$$\begin{aligned}
E\|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 &= E\|\mathbf{w}_t - \eta_t \nabla F(Q(\mathbf{w}_t); \xi_t) - \mathbf{w}_*\|_2^2 \\
&= E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E\langle \nabla F(Q(\mathbf{w}_t); \xi_t), \mathbf{w}_t - \mathbf{w}_* \rangle + \eta_t^2 E\|\nabla F(Q(\mathbf{w}_t); \xi_t)\|_2^2 \\
&\leq E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E\langle \nabla F(Q(\mathbf{w}_t); \xi_t), \mathbf{w}_t - \mathbf{w}_* \rangle + \eta_t^2 G^2 \\
&= E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E\langle \nabla F(Q(\mathbf{w}_t)), \mathbf{w}_t - \mathbf{w}_* \rangle + \eta_t^2 G^2 \\
&= E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E\langle \nabla F(Q(\mathbf{w}_t)), Q(\mathbf{w}_t) - \mathbf{w}_* \rangle \\
&\quad - 2\eta_t E\langle \nabla F(Q(\mathbf{w}_t)), \mathbf{w}_t - Q(\mathbf{w}_t) \rangle + \eta_t^2 G^2 \\
&\leq E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E(F(Q(\mathbf{w}_t)) - F(\mathbf{w}_*)) \\
&\quad - 2\eta_t E\langle \nabla F(Q(\mathbf{w}_t)), \mathbf{w}_t - Q(\mathbf{w}_t) \rangle + \eta_t^2 G^2 \\
&\leq E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E(F(Q(\mathbf{w}_t)) - F(\mathbf{w}_*)) \\
&\quad + \eta_t E\|\nabla F(Q(\mathbf{w}_t))\|_2 \|r(\mathbf{w}_t)\|_2 + \eta_t^2 G^2 \\
&\leq E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E(F(Q(\mathbf{w}_t)) - F(\mathbf{w}_*)) + 2\eta_t G E\|r(\mathbf{w}_t)\|_2 + \eta_t^2 G^2 \\
&\leq E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2\eta_t E(F(Q(\mathbf{w}_t)) - F(\mathbf{w}_*)) + 2\eta_t G \sqrt{d}S + \eta_t^2 G^2
\end{aligned}$$

where A.2, A.1, $\langle \mathbf{x}, \mathbf{y} \rangle \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$, A.2, and Lemma 2 are used respectively in the inequalities. We use the fact that gradients are unbiased as well, A.2. Let $\eta_t = \frac{1}{\sqrt{T}}$. Note that the same rate can be obtained with a decreasing learning rate scheme with a couple extra steps. Rearranging the terms and dividing with the learning rate give the Lemma. \square

By the telescoping sum of Lemma 3 over all iterations $t = 1, \dots, T$, we can prove the convergence of QAT training.

Theorem B.1 (QAT Convergence). *For a convex function with bounded unbiased stochastic gradients using a quantization method with bounded scales, we have*

$$E[F(Q(\mathbf{w}_\tau)) - F(\mathbf{w}_*)] = O\left(\frac{1}{\sqrt{T}} (\|\mathbf{w}_1 - \mathbf{w}_*\|_2^2 + G^2) + G\sqrt{d}S\right)$$

where τ is a random variable that takes values in $\{1, 2, \dots, T\}$ with equal probability³, T is the number of iterations, \mathbf{w}_1 is the initial model and \mathbf{w}_* is the optimal model $\mathbf{w}_* \in \arg \min_{\mathbf{w}} F(\mathbf{w})$, and the remaining constants are defined in the assumptions.

³We could further derive a model bound using Jensen on LHS since the function is convex. This allows us to avoid introducing another random variable τ , and would give LHS as $E\left[F\left(\frac{1}{T} \sum_{t=1}^T Q(\mathbf{w}_t)\right) - F(\mathbf{w}_*)\right]$. However the model, $\frac{1}{T} \sum_{t=1}^T Q(\mathbf{w}_t)$, is not necessarily a quantized model. Since we are interested in quantized model performance, we further need to argue that the quantization error of the averaged model is small and that would not change the rate. To avoid these extra steps, we introduced another random variable, τ , for the sake of simplicity in the proof.

Proof. If we average Lemma 3 for all iterations we get,

$$\begin{aligned}
E \left[\left[\frac{1}{T} \sum_{t=1}^T F(Q(\mathbf{w}_t)) \right] - F(\mathbf{w}_*) \right] &\leq \frac{1}{2\sqrt{T}} [-E\|\mathbf{w}_{T+1} - \mathbf{w}_*\|_2^2 + E\|\mathbf{w}_1 - \mathbf{w}_*\|_2^2] \\
&\quad + G\sqrt{d}S + \frac{1}{2\sqrt{T}}G^2 \\
&\leq \frac{1}{2\sqrt{T}}\|\mathbf{w}_1 - \mathbf{w}_*\|_2^2 + G\sqrt{d}S + \frac{1}{2\sqrt{T}}G^2
\end{aligned}$$

Note that LHS is the same if we choose $Q(\mathbf{w}_t)$ at random from all iterations with equal probability. \square

Remark 5. Note that the proof uses a bound on the quantization error in the form of Lemma 2. Deterministic quantization would have a smaller bound on the norm of the quantization error, $E\|r_Q(w)\|_2$, compared to the stochastic quantization. This motivates the use of deterministic quantization during the training phase.

Remark 6. LHS of the convergence rate in Theorem B.1 has two terms. First term decays with $O\left(\frac{1}{\sqrt{T}}\right)$ which is similar to the SGD rate. The second term is a constant. This constant term accounts for irreducible loss due to quantization.

C Convergence Analysis of FP8FedAvg-UQ

We note that F_k is the local loss at device $k \in [K]$ and F is the average of local functions, i.e $F(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K F_k(\mathbf{w})$. We assume the number of data points in each device is the same so that F is a non-weighted average of local functions for the sake of simplicity. We note that results can be adjusted easily for non-equal dataset size cases. We denote \mathbf{w}^* as the optimal model of the global loss, i.e $\arg \min F(\mathbf{w})$. For simplicity, we consider the balanced clients $n_k = \frac{n}{K}$ in our proof. However, the proof can be extended to the general imbalanced case similar to [23].

Assumption 4 (Smoothness). We assume the functions are L smooth.

$$\|\nabla F_k(\mathbf{x}) - \nabla F_k(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2 \quad \forall \mathbf{x}, \mathbf{y}, k.$$

Property 1. If we have smooth and convex functions, as in [13, 25, 1], for all $\mathbf{w}, \mathbf{x}, \mathbf{y}$,

$$-\langle \nabla F_k(\mathbf{w}), \mathbf{y} - \mathbf{x} \rangle \leq -F_k(\mathbf{y}) + F_k(\mathbf{x}) + \frac{L}{2} \|\mathbf{y} - \mathbf{w}\|_2^2.$$

C.1 Lemmas on the Stochastic Quantization for Model Communication

Lemma 4. Stochastic quantization is unbiased, i.e.,

$$Er_{Q_{\text{rand}}}(\mathbf{x}) = \mathbf{0}.$$

Proof. It follows directly from the definition as,

$$\begin{aligned} Er_{Q_{\text{rand}}}(\mathbf{x}) &= EQ_{\text{rand}}(\mathbf{x}) - \mathbf{x} \\ &= s \left(\frac{\mathbf{x}}{s} - \left\lfloor \frac{\mathbf{x}}{s} \right\rfloor \right) \odot \left(\left\lfloor \frac{\mathbf{x}}{s} \right\rfloor + 1 \right) + s \left(1 - \frac{\mathbf{x}}{s} + \left\lfloor \frac{\mathbf{x}}{s} \right\rfloor \right) \odot \left\lfloor \frac{\mathbf{x}}{s} \right\rfloor - \mathbf{x} = \mathbf{0} \end{aligned}$$

where \odot denotes the element-wise product. \square

Lemma 5. Let Q_{rand} be the stochastic unbiased quantization satisfying assumption 3. Then we have,

$$E \|r_{Q_{\text{rand}}}(\mathbf{x})\|_2^2 \leq S \|\mathbf{x}\|_1 \leq S\sqrt{d} \|\mathbf{x}\|_2$$

Proof. Let's start with a scalar case and we extend it to a vector case.

$$\begin{aligned} E |r_{Q_{\text{rand}}}(x)|^2 &= s^2 \left(\frac{x}{s} - \left\lfloor \frac{x}{s} \right\rfloor \right) \left(\left\lfloor \frac{x}{s} \right\rfloor + 1 - \frac{x}{s} \right)^2 + s^2 \left(1 - \frac{x}{s} + \left\lfloor \frac{x}{s} \right\rfloor \right) \left(\left\lfloor \frac{x}{s} \right\rfloor - \frac{x}{s} \right)^2 \\ &= s^2 \left(\frac{x}{s} - \left\lfloor \frac{x}{s} \right\rfloor \right) \left(1 + \left\lfloor \frac{x}{s} \right\rfloor - \frac{x}{s} \right) \leq s^2 \min \left(\frac{x}{s} - \left\lfloor \frac{x}{s} \right\rfloor, 1 - \frac{x}{s} + \left\lfloor \frac{x}{s} \right\rfloor \right) \\ &\leq s^2 \left| \frac{x}{s} \right| \leq S|x| \end{aligned}$$

where inequalities follow from the fact that $\frac{x}{s} - \left\lfloor \frac{x}{s} \right\rfloor \leq 1$.

We can add the scalar variances to bound a vector variance as,

$$E \|r_{Q_{\text{rand}}}(\mathbf{x})\|_2^2 = \sum_{i \in [d]} E \|r_{Q_{\text{rand}}}([\mathbf{x}]_i)\|_2^2 \leq S \sum_{i \in [d]} |[\mathbf{x}]_i| = S \|\mathbf{x}\|_1 \leq S\sqrt{d} \|\mathbf{x}\|_2$$

where we use Cauchy–Schwarz inequality in the last step. \square

Lemma 6 (Quantization Error Decomposition). Let assumption 3 holds. Both uniform and FP8 quantization satisfies,

$$E |r_Q(Q(x) + y)|^2 \leq S|y|.$$

For d dimensional vectors we get,

$$E \|r_Q(Q(\mathbf{x}) + \mathbf{y})\|_2^2 \leq S\sqrt{d} \|\mathbf{y}\|_2.$$

Proof. We give a proof for scalar case. Vector version comes from upper bounding scalar case using Cauchy-Schwarz. Note that variance is higher for randomized quantization so let's prove the bound for Q_{rand} . Due to symmetry, we can assume $Q_{\text{rand}}(x) \geq 0$.

Let's define grid points as g_i where $g_0 = 0$ and $g_{i+1} > g_i$. Note that due to quantization definitions, we have $g_i \% (g_{i+1} - g_i) = 0$, .i.e $\exists k \in \mathbb{Z}^+$ such that $g_i = k(g_{i+1} - g_i)$. Furthermore, we have a finer resolution close to 0, .i.e $g_{i+1} - g_i \geq g_i - g_{i-1}$.

We extensively use a step in Lemma 5 as,

$$E |r_{Q_{\text{rand}}}(z)|^2 = s^2 q_z (1 - q_z) \leq s^2 \min(q_z, 1 - q_z) \leq s^2 \left\lfloor \frac{z}{s} \right\rfloor = s|z|$$

where $q_z = \frac{z}{s} - \left\lfloor \frac{z}{s} \right\rfloor$. We use this relation by plugging in $z = Q_{\text{rand}}(x) + y$ and investigating $q_{Q_{\text{rand}}(x)+y}$.

Since $Q_{\text{rand}}(x)$ is already quantized, $\exists i \geq 0$ such that $Q_{\text{rand}}(x) = g_i$. Let $g_{j+1} > Q_{\text{rand}}(x) + y \geq g_j$.

Let $y = \delta + g_j - g_i$. Then we have $g_{j+1} > \delta + g_j \geq g_j \implies g_{j+1} - g_j > \delta \geq 0$. We also know $g_{j+1} - g_i > y \geq g_j - g_i$.

We have, by definition,

$$q_{Q_{\text{rand}}(x)+y} = \frac{g_j + \delta}{g_{j+1} - g_j} - \left\lfloor \frac{g_j + \delta}{g_{j+1} - g_j} \right\rfloor = \frac{\delta}{g_{j+1} - g_j} - \left\lfloor \frac{\delta}{g_{j+1} - g_j} \right\rfloor = q_\delta$$

since g_j is a multiple $g_{j+1} - g_j$.

Let's look at different cases.

Case $i \leq j$

Note that $g_j - g_i \geq 0$ so that $|y| = |\delta + g_j - g_i| \geq |\delta|$. Then, we have,

$$\begin{aligned} E |r_{Q_{\text{rand}}}(Q(x) + y)|^2 &\leq (g_{j+1} - g_j)^2 \min(q_\delta, 1 - q_\delta) \\ &\leq (g_{j+1} - g_j) |\delta| \leq S|\delta| \leq S|y| \quad \square. \end{aligned}$$

Case $i > j + 1$

Note that $g_{j+1} - g_i < 0$ and y is negative. Let's look at magnitude of y and δ .

$$0 > g_{j+1} - g_i > y \geq g_j - g_i \implies |y| > g_i - g_{j+1} \geq g_{j+2} - g_{j+1}.$$

We already know that $g_{j+1} - g_j > \delta \geq 0$. Then we have,

$$|y| > g_{j+2} - g_{j+1} \geq g_{j+1} - g_j > \delta$$

Since $|y| > |\delta|$, we get,

$$\begin{aligned} E \left| r_{Q_{\text{rand}}} (Q(x) + y) \right|^2 &\leq (g_{j+1} - g_j)^2 \min(q_\delta, 1 - q_\delta) \leq (g_{j+1} - g_j) |\delta| \\ &\leq S|\delta| \leq S|y| \quad \square. \end{aligned}$$

Case $i = j + 1$

We have $y = \delta + g_j - g_i = \delta - (g_{j+1} - g_j)$. Let's look at q_δ as,

$$\begin{aligned} q_\delta &= \frac{\delta}{g_{j+1} - g_j} - \left\lfloor \frac{\delta}{g_{j+1} - g_j} \right\rfloor = \frac{\delta - (g_{j+1} - g_j)}{g_{j+1} - g_j} - \left\lfloor \frac{\delta - (g_{j+1} - g_j)}{g_{j+1} - g_j} \right\rfloor \\ &= \frac{y}{g_{j+1} - g_j} - \left\lfloor \frac{y}{g_{j+1} - g_j} \right\rfloor = q_y \end{aligned}$$

Then we have,

$$\begin{aligned} E \left| r_{Q_{\text{rand}}} (Q(x) + y) \right|^2 &\leq (g_{j+1} - g_j)^2 \min(q_\delta, 1 - q_\delta) \\ &= (g_{j+1} - g_j)^2 \min(q_y, 1 - q_y) \leq (g_{j+1} - g_j) |y| \leq S|y|. \end{aligned}$$

□

Please note that the above proof holds for any quantization scheme of which the grid is symmetric with respect to zero and the bin size increases monotonically going from zero to plus or minus infinity. The FP8 quantization obviously satisfies this condition.

C.2 Lemma on a Single Communication Round

We define some useful quantities. For simplicity in the proof, let us define auxiliary models as,

$$\mathbf{v}_{t,u+1}^k = \mathbf{v}_{t,u}^k - \eta_t \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \quad \forall u \in [U], \quad \mathbf{v}_{t,1}^k = Q_{\text{rand}}(\mathbf{w}_t)$$

where U is the total number of local updates per communication round per device. Furthermore, we can unroll the recursion as,

$$\begin{aligned}\mathbf{v}_{t,U+1}^k &= \mathbf{v}_{t,U}^k - \eta_t \nabla F_k \left(Q_{\det} \left(\mathbf{v}_{t,U}^k \right); \xi_{t,U}^k \right) \\ &= \mathbf{v}_{t,UE-1}^k - \eta_t \nabla F_k \left(Q_{\det} \left(\mathbf{v}_{t,U-1}^k \right); \xi_{t,U-1}^k \right) - \eta_t \nabla F_k \left(Q_{\det} \left(\mathbf{v}_{t,U}^k \right); \xi_{t,U}^k \right) \\ &= \dots = Q_{\text{rand}}(\mathbf{w}_t) - \eta_t \sum_{u \in [U]} \nabla F_k \left(Q_{\det} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right)\end{aligned}$$

It is clear to see that $\mathbf{w}_{t+1}^k = \mathbf{v}_{t,U+1}^k$ for active devices. Let's define inactive device $\mathbf{w}_{t+1}^k = \mathbf{v}_{t,U+1}^k$ as well. Note that this is just for notation and the algorithm is unchanged. Because if k is not active we do not use \mathbf{w}_{t+1}^k in our algorithm. Let us define a drift quantity similar to [13].

$$V_t = \frac{1}{KU} \sum_{k \in [K]} \sum_{u \in [U]} E \left\| Q_{\text{rand}}(\mathbf{w}_t) - Q_{\det} \left(\mathbf{v}_{t,u}^k \right) \right\|_2^2. \quad (11)$$

Note that if local models diverge, we get a higher V_t . We can obtain the following lemma for a single communication round of the FP8FedAvg-UQ algorithm.

Lemma 7. If assumptions 1, 2, 3, 4 hold and we use an unbiased quantization for model transmission, we have,

$$\begin{aligned}E \left\| \mathbf{w}_{t+1} - \mathbf{w}_* \right\|_2^2 &\leq E \left\| \mathbf{w}_t - \mathbf{w}_* \right\|_2^2 - 2U\eta_t E \left(F(Q_{\text{rand}}(\mathbf{w}_t)) - F(\mathbf{w}_*) \right) \\ &\quad + \eta_t LUV_t + 2S\sqrt{d}GU\eta_t + \eta_t^2 U^2 G^2\end{aligned} \quad (12)$$

$$V_t \leq 18U^3 S\sqrt{d}G\eta_t + 9U^2 \eta_t^2 G^2 \quad (13)$$

Proof. First, we prove Eq. 12. Due to the model-to-server communication and the model aggregation on the server in the t -th round, we have

$$\begin{aligned}E \left\| \mathbf{w}_{t+1} - \mathbf{w}_* \right\|_2^2 &= E \left\| \frac{1}{P} \sum_{k \in \mathcal{P}_t} Q_{\text{rand}} \left(\mathbf{w}_{t+1}^k \right) - \mathbf{w}_* \right\|_2^2 \\ &\leq \frac{1}{P} E \sum_{k \in \mathcal{P}_t} \left\| Q_{\text{rand}} \left(\mathbf{w}_{t+1}^k \right) - \mathbf{w}_* \right\|_2^2 = \frac{1}{K} \sum_{k \in [K]} E \left\| Q_{\text{rand}} \left(\mathbf{w}_{t+1}^k \right) - \mathbf{w}_* \right\|_2^2\end{aligned}$$

where we use definition of \mathbf{w}_{t+1} and triangular inequality ($\left\| \sum_{n \in [N]} a_n \right\|^2 \leq N \sum_{n \in [N]} \|a_n\|^2$). Lastly, we use the fact that active devices are sampled uniformly at random so that each

device has an activation probability of $\frac{P}{K}$. Let's continue as

$$\begin{aligned}
E \|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 &\leq \frac{1}{K} \sum_{k \in [K]} E \left\| Q_{\text{rand}} \left(\mathbf{w}_{t+1}^k \right) - \mathbf{w}_* \right\|_2^2 \\
&= \frac{1}{K} \sum_{k \in [K]} E \left\| r_{Q_{\text{rand}}} \left(\mathbf{w}_{t+1}^k \right) + \mathbf{w}_{t+1}^k - \mathbf{w}_* \right\|_2^2 \\
&= \frac{1}{K} \left(\sum_{k \in [K]} E \left\| r_{Q_{\text{rand}}} \left(\mathbf{w}_{t+1}^k \right) \right\|_2^2 + 2E \left\langle r_{Q_{\text{rand}}} \left(\mathbf{w}_{t+1}^k \right), \mathbf{w}_{t+1}^k - \mathbf{w}_* \right\rangle + E \left\| \mathbf{w}_{t+1}^k - \mathbf{w}_* \right\|_2^2 \right) \\
&= \frac{1}{K} \left(\sum_{k \in [K]} E \left\| r_{Q_{\text{rand}}} \left(\mathbf{w}_{t+1}^k \right) \right\|_2^2 + E \left\| \mathbf{w}_{t+1}^k - \mathbf{w}_* \right\|_2^2 \right)
\end{aligned}$$

where we use the fact that Q_{rand} is an unbiased quantizer. Let's bound $E \|\mathbf{w}_{t+1}^k - \mathbf{w}_*\|^2$ as

$$\begin{aligned}
E \left\| \mathbf{w}_{t+1}^k - \mathbf{w}_* \right\|_2^2 &= E \left\| Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_* - \eta_t \sum_{u \in [U]} \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \right\|_2^2 \\
&= E \left\| Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_* \right\|_2^2 - 2\eta_t \sum_{u \in [U]} E \left\langle Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_*, \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \right\rangle \\
&\quad + \eta_t^2 E \left\| \sum_{u \in [U]} \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \right\|_2^2 \\
&\leq E \left\| Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_* \right\|_2^2 - 2\eta_t \sum_{u \in [U]} E \left\langle Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_*, \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \right\rangle \\
&\quad + \eta_t^2 U^2 G^2 \\
&= E \left\| Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_* \right\|_2^2 - 2\eta_t \sum_{u \in [U]} E \left\langle Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_*, \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right) \right) \right\rangle \\
&\quad + \eta_t^2 U^2 G^2 \\
&\leq E \left\| Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_* \right\|_2^2 \\
&\quad + 2\eta_t \left(\sum_{u \in [U]} E \left[-F_k \left(Q_{\text{rand}} (\mathbf{w}_t) \right) + F_k (\mathbf{w}_*) \right] + \frac{L}{2} \left\| Q_{\text{rand}} (\mathbf{w}_t) - Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right) \right\|_2^2 \right) \\
&\quad + \eta_t^2 U^2 G^2 \\
&= E \left\| Q_{\text{rand}} (\mathbf{w}_t) - \mathbf{w}_* \right\|_2^2 - 2U\eta_t E \left(F_k \left(Q_{\text{rand}} (\mathbf{w}_t) \right) - F_k (\mathbf{w}_*) \right)
\end{aligned}$$

$$+ \eta_t L \sum_{u \in [U]} \left\| Q_{\text{rand}}(\mathbf{w}_t) - Q_{\text{det}}(\mathbf{v}_{t,u}^k) \right\|_2^2 + \eta_t^2 U^2 G^2$$

where we use the fact that gradients are bounded, $\nabla F_k(Q_{\text{det}}(\mathbf{v}_{t,u}^k); \xi_{t,u}^k)$ is an unbiased gradient estimate and property 1. We further restate $E \|Q_{\text{rand}}(\mathbf{w}_t) - \mathbf{w}_*\|_2^2$ as,

$$\begin{aligned} E \|Q_{\text{rand}}(\mathbf{w}_t) - \mathbf{w}_*\|_2^2 &= E \|r_{Q_{\text{rand}}}(\mathbf{w}_t) + \mathbf{w}_t - \mathbf{w}_*\|_2^2 \\ &= E \|r_{Q_{\text{rand}}}(\mathbf{w}_t)\|_2^2 + 2E \langle r_{Q_{\text{rand}}}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_* \rangle + E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2 \\ &= E \|r_{Q_{\text{rand}}}(\mathbf{w}_t)\|_2^2 + E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2 \end{aligned}$$

where we use the fact that Q_{rand} is an unbiased quantizer. Then, we have,

$$\begin{aligned} &E \left\| \mathbf{w}_{t+1}^k - \mathbf{w}_* \right\|_2^2 \\ &\leq E \|Q_{\text{rand}}(\mathbf{w}_t) - \mathbf{w}_*\|_2^2 - 2U\eta_t E (F_k(Q_{\text{rand}}(\mathbf{w}_t)) - F_k(\mathbf{w}_*)) \\ &\quad + \eta_t L \sum_{u \in [U]} \left\| Q_{\text{rand}}(\mathbf{w}_t) - Q_{\text{det}}(\mathbf{v}_{t,u}^k) \right\|_2^2 + \eta_t^2 U^2 G^2 \\ &= E \|r_{Q_{\text{rand}}}(\mathbf{w}_t)\|_2^2 + E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2U\eta_t E (F_k(Q_{\text{rand}}(\mathbf{w}_t)) - F_k(\mathbf{w}_*)) \\ &\quad + \eta_t L \sum_{u \in [U]} \left\| Q_{\text{rand}}(\mathbf{w}_t) - Q_{\text{det}}(\mathbf{v}_{t,u}^k) \right\|_2^2 + \eta_t^2 U^2 G^2 \end{aligned}$$

Using Lemma 6 we have,

$$\begin{aligned} E \left\| r_{Q_{\text{rand}}}(\mathbf{w}_{t+1}^k) \right\|_2^2 &= E \left\| r_{Q_{\text{rand}}} \left(Q_{\text{rand}}(\mathbf{w}_t) - \eta_t \sum_{u \in [U]} \nabla F_k(Q_{\text{det}}(\mathbf{v}_{t,u}^k); \xi_{t,u}^k) \right) \right\|_2^2 \\ &\leq S\sqrt{d}E \left\| \eta_t \sum_{u \in [U]} \nabla F_k(Q_{\text{det}}(\mathbf{v}_{t,u}^k); \xi_{t,u}^k) \right\|_2 \leq S\sqrt{d}GU\eta_t \end{aligned}$$

where U is the number of local iterates. Finally, we can upper bound RHS as,

$$\begin{aligned} E \|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 &\leq \frac{1}{K} \left(\sum_{k \in [K]} E \|r_{Q_{\text{rand}}}(\mathbf{w}_{t+1}^k)\|_2^2 + E \|\mathbf{w}_{t+1}^k - \mathbf{w}_*\|_2^2 \right) \\ &\leq E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2U\eta_t E (F(Q_{\text{rand}}(\mathbf{w}_t)) - F(\mathbf{w}_*)) \\ &\quad + \frac{\eta_t L}{K} \sum_{k \in [K]} \sum_{u \in [U]} \left\| Q_{\text{rand}}(\mathbf{w}_t) - Q_{\text{det}}(\mathbf{v}_{t,u}^k) \right\|_2^2 + 2S\sqrt{d}GU\eta_t + \eta_t^2 U^2 G^2 \\ &= E \|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2U\eta_t E (F(Q_{\text{rand}}(\mathbf{w}_t)) - F(\mathbf{w}_*)) + \eta_t LUV_t + 2S\sqrt{d}GU\eta_t + \eta_t^2 U^2 G^2 \end{aligned}$$

This completes Eq. 12's proof.

Remark 7. Note that we extensively use unbiasedness of stochastic quantization via $E \langle \cdot, r_{Q_{\text{rand}}}(\mathbf{w}) \rangle = \mathbf{0}$. Otherwise, we need to upper bound this term. There exists cases where a biased resetting diverges [5]. Hence, stochastic quantization is needed for convergence.

Next, we prove Eq. 13 for upper bounding the drift V_t in round t defined in (13).

$$\begin{aligned}
 & E \left\| Q_{\text{det}} \left(\mathbf{v}_{t,u+1}^k \right) - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &= E \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) + \mathbf{v}_{t,u+1}^k - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &= E \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) + \mathbf{v}_{t,u}^k - \eta_t \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &= E \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) - r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u}^k \right) - \eta_t \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) + Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &\leq \frac{U}{U-1} E \left\| Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &\quad + U E \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) - r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u}^k \right) - \eta_t \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \right\|_2^2 \\
 &\leq \frac{U}{U-1} E \left\| Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &\quad + 3UE \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) \right\|_2^2 + 3UE \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u}^k \right) \right\|_2^2 + 3U\eta_t^2 E \left\| \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right); \xi_{t,u}^k \right) \right\|_2^2 \\
 &\leq \frac{U}{U-1} E \left\| Q_{\text{det}} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}}(\mathbf{w}_t) \right\|_2^2 \\
 &\quad + 3UE \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) \right\|_2^2 + 3UE \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u}^k \right) \right\|_2^2 + 3U\eta_t^2 G^2
 \end{aligned}$$

where we use $\|\mathbf{x} + \mathbf{y}\|_2^2 \leq (1 + \frac{1}{A}) \|\mathbf{x}\|_2^2 + (A+1) \|\mathbf{y}\|_2^2$, triangular inequality and bound on the gradients.

Let's bound $E \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) \right\|_2^2$ using Lemma 6 as,

$$\begin{aligned}
 E \left\| r_{Q_{\text{det}}} \left(\mathbf{v}_{t,u+1}^k \right) \right\|_2^2 &= E \left\| r_{Q_{\text{det}}} \left(Q_{\text{rand}}(\mathbf{w}_t) - \eta_t \sum_{s \in [u]} \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,s}^k \right); \xi_{t,s}^k \right) \right) \right\|_2^2 \\
 &\leq S\sqrt{d}E \left\| \eta_t \sum_{s \in [u]} \nabla F_k \left(Q_{\text{det}} \left(\mathbf{v}_{t,s}^k \right); \xi_{t,s}^k \right) \right\|_2 \leq S\sqrt{d}G u \eta_t
 \end{aligned}$$

This leads to

$$\begin{aligned}
& E \left\| Q_{\det} \left(\mathbf{v}_{t,u+1}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 \\
& \leq \frac{U}{U-1} E \left\| Q_{\det} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 + 3UE \left\| r_{Q_{\det}} \left(\mathbf{v}_{t,u+1}^k \right) \right\|_2^2 \\
& \quad + 3UE \left\| r_{Q_{\det}} \left(\mathbf{v}_{t,u}^k \right) \right\|_2^2 + 3U\eta_t^2 G^2 \\
& \leq \frac{U}{U-1} E \left\| Q_{\det} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 + 6U^2 S \sqrt{d} G \eta_t + 3U\eta_t^2 G^2
\end{aligned}$$

Let's unroll the recursion noting that $Q_{\det}(\mathbf{v}_{t,1}^k) = Q_{\text{rand}}(\mathbf{w}_t)$,

$$\begin{aligned}
& E \left\| Q_{\det} \left(\mathbf{v}_{t,u+1}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 \\
& \leq \frac{U}{U-1} E \left\| Q_{\det} \left(\mathbf{v}_{t,u}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 + 6U^2 S \sqrt{d} G \eta_t + 3U\eta_t^2 G^2 \\
& \leq \left(\frac{U}{U-1} \right)^2 E \left\| Q_{\det} \left(\mathbf{v}_{t,u-1}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 \\
& \quad + \left(6U^2 S \sqrt{d} G \eta_t + 3U\eta_t^2 G^2 \right) \left(1 + \frac{U}{U-1} \right) \\
& \dots \\
& \leq \left(6U^2 S \sqrt{d} G \eta_t + 3U\eta_t^2 G^2 \right) \left(1 + \frac{U}{U-1} + \dots + \left(\frac{U}{U-1} \right)^{u-1} \right)
\end{aligned}$$

Let's bound the second term in the RHS as,

$$\begin{aligned}
1 + \frac{U}{U-1} + \dots + \left(\frac{U}{U-1} \right)^{u-1} & \leq u \left(\frac{U}{U-1} \right)^{u-1} = u \left(1 + \frac{1}{U-1} \right)^{u-1} \\
& \leq U \left(1 + \frac{1}{U-1} \right)^{U-1} \leq Ue \leq 3U
\end{aligned}$$

Hence we get

$$E \left\| Q_{\det} \left(\mathbf{v}_{t,u+1}^k \right) - Q_{\text{rand}} \left(\mathbf{w}_t \right) \right\|_2^2 \leq 18U^3 S \sqrt{d} G \eta_t + 9U^2 \eta_t^2 G^2 \quad (14)$$

Note that we inherently assume $U > 1$ in order to have a coefficient as $\frac{U}{U-1}$. Assume $U = 1$. Then we have, $V_t = 0$ by definition and Eq. 14 holds. If we average Eq. 14 over U and K we get Eq. 13 as, $V_t \leq 18U^3 S \sqrt{d} G \eta_t + 9U^2 \eta_t^2 G^2$.

□

C.3 Proof of the Main Theorem

Now, we are ready to present the main theorem on the convergence of the proposed FP8FedAvg-UQ algorithm.

Theorem C.1 (FP8FedAvg-UQ Convergence). **For convex and smooth federated losses with bounded unbiased stochastic gradients using a quantization method with bounded scales during training and an unbiased quantization with bounded scales for model transfer, we have,**

$$\begin{aligned} & E[F(Q(\mathbf{w}_\tau)) - F(\mathbf{w}_*)] \\ &= O\left(\frac{1}{\sqrt{TU}}\|\mathbf{w}_1 - \mathbf{w}_*\|_2^2 + \frac{1}{T}UG^2L + \frac{1}{\sqrt{T}}G\sqrt{U}\left(G + U^2S\sqrt{d}L\right) + S\sqrt{d}G\right) \end{aligned}$$

where τ is a random variable that takes values in $\{1, 2, \dots, T\}$ with equal probability, T is the number of rounds, U is the total number of updates done in each round, the quantization scales s_i are uniformly bounded by S , \mathbf{w}_1 is the initial model, and \mathbf{w}_* is an optimal solution of (1).

Combining Eq. 12 and $\eta_t LU$ times Eq. 13 gives,

$$\begin{aligned} E\|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 &\leq E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 - 2U\eta_t E(F(Q_{\text{rand}}(\mathbf{w}_t)) - F(\mathbf{w}_*)) \\ &\quad + 2S\sqrt{d}GU\eta_t + \eta_t^2 U^2 G^2 + 18U^4 S\sqrt{d}G\eta_t^2 L + 9U^3 \eta_t^3 G^2 L \end{aligned}$$

Rearranging the terms and dividing both sides with $2U\eta_t$ gives,

$$\begin{aligned} E[F(Q_{\text{rand}}(\mathbf{w}_t)) - F(\mathbf{w}_*)] &\leq \frac{1}{2U\eta_t} \left(-E\|\mathbf{w}_{t+1} - \mathbf{w}_*\|_2^2 + E\|\mathbf{w}_t - \mathbf{w}_*\|_2^2 \right) \\ &\quad + S\sqrt{d}G + \frac{1}{2}\eta_t UG^2 + 9U^3 S\sqrt{d}G\eta_t L + \frac{9}{2}U^2 \eta_t^2 G^2 L \end{aligned}$$

Let $\eta_t = \frac{1}{\sqrt{UT}}$. Note that we can get the same rate with a decreasing learning rate as well. Let's average the inequality over t as,

$$\begin{aligned}
& E \left[\left[\frac{1}{T} \sum_{t=1}^T F(Q_{\text{rand}}(\mathbf{w}_t)) \right] - F(\mathbf{w}_*) \right] \\
& \leq \frac{1}{2\sqrt{TU}} [-E\|\mathbf{w}_{T+1} - \mathbf{w}_*\|_2^2 + E\|\mathbf{w}_1 - \mathbf{w}_*\|_2^2] + \frac{1}{T} \frac{9}{2} U G^2 L \\
& \quad + \frac{1}{\sqrt{T}} G \sqrt{U} \left(\frac{1}{2} G + 9U^2 S \sqrt{d} L \right) + S \sqrt{d} G \\
& \leq \frac{1}{2\sqrt{TU}} \|\mathbf{w}_1 - \mathbf{w}_*\|_2^2 + \frac{1}{T} \frac{9}{2} U G^2 L + \frac{1}{\sqrt{T}} G \sqrt{U} \left(\frac{1}{2} G + 9U^2 S \sqrt{d} L \right) + S \sqrt{d} G \\
& = O \left(\frac{1}{\sqrt{TU}} \|\mathbf{w}_1 - \mathbf{w}_*\|_2^2 + \frac{1}{T} U G^2 L + \frac{1}{\sqrt{T}} G \sqrt{U} \left(G + U^2 S \sqrt{d} L \right) + S \sqrt{d} G \right) \quad \square
\end{aligned}$$

Paper v



Reprinted from Proc. InterSpeech, 2022, pp. 1791-1795, © 2022, with permission from ISCA. Minor typos in equation (1) and (9) that appeared in the published version of this paper have been corrected in this reprint. Also, the term "mean average error" has been corrected to "mean absolute error".

Extending GCC-PHAT using Shift Equivariant Neural Networks

AXEL BERG^{1,2}, MARK O’CONNOR³, KALLE ÅSTRÖM², MAGNUS OSKARSSON²

¹Arm, ²Centre for Mathematical Sciences, Lund University, ³Tenstorrent

Abstract: Speaker localization using microphone arrays depends on accurate time delay estimation techniques. For decades, methods based on the generalized cross correlation with phase transform (GCC-PHAT) have been widely adopted for this purpose. Recently, the GCC-PHAT has also been used to provide input features to neural networks in order to remove the effects of noise and reverberation, but at the cost of losing theoretical guarantees in noise-free conditions. We propose a novel approach to extending the GCC-PHAT, where the received signals are filtered using a shift equivariant neural network that preserves the timing information contained in the signals. By extensive experiments we show that our model consistently reduces the error of the GCC-PHAT in adverse environments, with guarantees of exact time delay recovery in ideal conditions.

1 Introduction

Time delay estimation (TDE) is an essential component in many applications involving acoustic localization, including sound source tracking [7], robotics [14] and self-calibration [3]. In a typical setup, time delays are estimated by analyzing the signals received by a set of synchronized microphones with known positions. The transmitted waveform, which is assumed to have originated from a single sound source, is considered unknown, as is the time at which it was transmitted. Therefore, the time of travel from the source to the microphones cannot be obtained directly, but instead the time difference of arrival (TDOA) is measured by correlating the received signals. The set of TDOA measurements from a microphone array can then be used to compute the signal direction of arrival (DOA) or the sound source position using multilateration [2].

The generalized cross-correlation (GCC) has been the most widely adopted method for TDOA estimation for many decades. In particular, the phase transform (GCC-PHAT) [13] filter is commonly used in many acoustic scenarios, due to its fast implementation and robustness in adverse environments. However, with the recent advent of deep learning, a wide variety of methods for sound source localization estimation have been developed without the use of cross-correlations, instead processing only the raw waveforms or spectrograms of the signals [8]. Furthermore, several of these methods do not explicitly estimate the TDOAs, but instead train the models to directly predict the DOA [4, 16] or sound source coordinates [24].

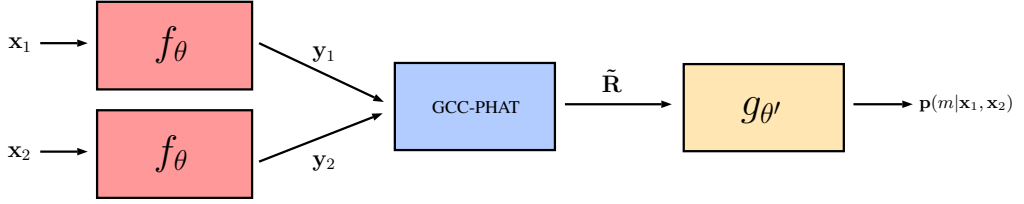


Figure 1: An illustration of our proposed method. The input signals are first filtered using the same neural network f_{θ} , then correlated using GCC-PHAT, whose outputs are mapped to a probability distribution over time delays by another neural network $g_{\theta'}$.

Other works have instead attempted to combine GCCs with neural networks [27, 9]. Comanducci et al. [6] used an autoencoder network on the outputs of a frequency sliding GCC [5] in order to de-noise the correlations. Wang et al. [25] instead used a neural network for predicting a speech mask that can be interpreted as a learnable frequency-selective linear filter. The speech mask is then applied together with the PHAT when correlating the received signals. Notably, Salvati et al. [21] proposed computing multiple GCCs, each with its own weighted transfer function, and processing the outputs using a convolutional neural network (CNN) that predicts the TDOA for the two signals. Although this method reduces the average error, it struggles to make accurate prediction within a few samples, which is required for high-precision localization.

We propose a novel method for TDOA estimation by filtering the raw waveforms using a neural network before computing the GCC-PHAT. The network can then be trained to exploit patterns in the data, e.g. the acoustic properties of human speech, in order to remove the effects of noise and reverberation. Furthermore, by using a shift equivariant CNN (SE-CNN), the network can learn to find useful representations while preserving the timing information contained in the signals.

2 Method

Prerequisites. Consider a reverberant three-dimensional room with two microphones positioned at $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{R}^3$ and a single sound source positioned at $\mathbf{r}_s \in \mathbb{R}^3$ emitting an unknown acoustic signal \mathbf{s} . Assuming a time-window of N samples, the received signals $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^N$ at the two microphones can be written as

$$\begin{aligned} x_1[n] &= (h_1 * s)[n] + w_1[n], \\ x_2[n] &= (h_2 * s)[n] + w_2[n], \end{aligned} \tag{1}$$

for $n = 0, \dots, N - 1$. Here $h_1[n], h_2[n]$ and $w_1[n], w_2[n]$ are the channel impulse responses from the source to the microphones and additive white noise respectively. Taking

the discrete Fourier transform (DFT) of both sides of (1) yields

$$\begin{aligned} X_1[k] &= H_1[k]S[k] + W_1[k], \\ X_2[k] &= H_2[k]S[k] + W_2[k], \end{aligned} \quad (2)$$

for $k = 0, \dots, N - 1$. With this notation, the GCC is defined as

$$R[m] = \frac{1}{N} \sum_{k=0}^{N-1} \phi[k] X_1[k] X_2^*[k] e^{\frac{i2\pi km}{N}}, \quad (3)$$

for $m = -\tau_{\max}, \dots, \tau_{\max}$. The maximal delay is typically taken to be $\tau_{\max} = \lceil \|\mathbf{r}_1 - \mathbf{r}_2\| F_s / c \rceil$ where c is the speed of sound and F_s is the sample rate. Furthermore, $\phi[k]$ is a weighting function. In particular, the PHAT [13] weighting function is given by $\phi[k] = 1/|X_1[k]X_2^*[k]|$. The estimated time delay is then obtained as

$$\hat{\tau} = \arg \max_m R[m]. \quad (4)$$

The PHAT can be regarded as a weighting function that places equal importance on all frequencies in the signal spectrum and only considers the phase of received signals. In an anechoic noise-free environment, the GCC-PHAT outputs a unit impulse centered at the correct time-delay. In the presence of echoes, the PHAT filter attenuates the interference to some extent by limiting the smearing of the correlation, which makes it possible to recover the line of sight component. However, the PHAT also introduces errors from frequency components outside the signal spectrum, which motivates the introduction of a learnable filter that can suppress noise and interference.

Extending GCC-PHAT. The main idea of the proposed method is to apply a non-linear filter function on the received signals in order to remove effects of reverberation and noise, which can be realized by using a neural network. However, in the process of doing so, the network needs to preserve the timing information stored in the signals, since this is what we seek to recover when estimating the TDOA. Hence, we seek to employ a network architecture that is explicitly designed for this purpose, and the following definition is then useful.

Definition 2.1. Let \mathbf{x} be a signal and $(\mathbf{x})_\tau$ denote a column-wise circular shift of \mathbf{x} . Then $f : \mathbb{R}^N \rightarrow \mathbb{R}^{N \times L}$ is said to be equivariant with respect to time shifts if for any time lag $\tau \in \mathbb{Z}$ we have that $f((\mathbf{x})_\tau) = (f(\mathbf{x}))_\tau$.

Although CNNs are generally approximately shift-equivariant, not all implementations satisfy this property exactly, due to edge effects. We therefore employ circular padding of the signals before each convolutional layer in the network in order to preserve the timing

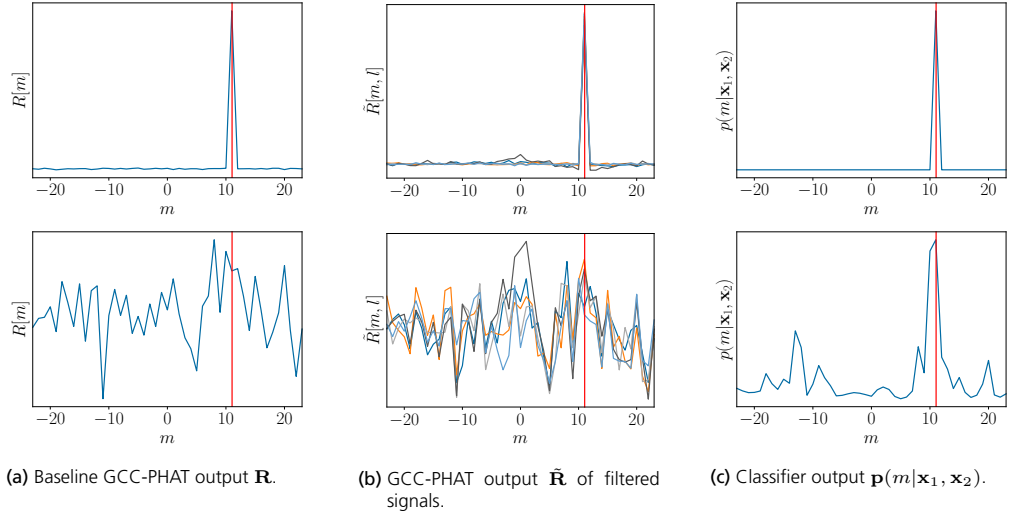


Figure 2: Examples of the baseline GCC-PHAT output \mathbf{R} and $\tilde{\mathbf{R}}$ for the unfiltered speech signals $\mathbf{x}_1, \mathbf{x}_2$ and the filtered signals $\mathbf{y}_1, \mathbf{y}_2$ respectively, where the true delay is marked with a red line. For better visualization, only the first five columns of $\tilde{\mathbf{R}}$ are shown. In an ideal environment, the correlations are identical up round-off errors. However, in an adverse environment not all correlations exhibit a clear peak, but the classifier is able to recover the correct delay by combining them into a single probability distribution $\mathbf{p}(m|\mathbf{x}_1, \mathbf{x}_2)$. Top row: ideal noise-free environment. Bottom row: noisy environment.

information completely. For further discussion on convolutions and shift equivariance, we refer to [11].

An overview of the proposed method is illustrated in Figure 1. Let $f_{\theta} : \mathbb{R}^N \rightarrow \mathbb{R}^{N \times L}$ denote the SE-CNN parameterized by a set of learnable weights θ . The network receives input signals of length N and outputs L new signals of the same length

$$\begin{aligned} \mathbf{y}_1 &= f_{\theta}(\mathbf{x}_1), \\ \mathbf{y}_2 &= f_{\theta}(\mathbf{x}_2), \end{aligned} \quad (5)$$

where $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^{N \times L}$, such that each column of the outputs represent a filtered signal. By letting $L > 1$, the network can learn to apply different non-linear filters that capture different properties of the transmitted signals. For each of the L signal components, we then apply the GCC-PHAT individually as

$$\tilde{R}[m, l] = \frac{1}{N} \sum_{k=0}^{N-1} \frac{Y_1[k, l] Y_2^*[k, l]}{|Y_1[k, l] Y_2^*[k, l]|} e^{\frac{i 2 \pi k m}{N}}, \quad (6)$$

where $\mathbf{Y}_1, \mathbf{Y}_2$ are the column-wise DFTs of $\mathbf{y}_1, \mathbf{y}_2$ and $\tilde{\mathbf{R}} \in \mathbb{R}^{(2\tau_{\max}+1) \times L}$. Now if $\mathbf{x}_1 = (\mathbf{x}_2)_{\tau}$, then it follows from the shift equivariance of the network that $\mathbf{y}_1 = (\mathbf{y}_2)_{\tau}$, or equivalently that $Y_1[k, l] = Y_2[k, l] e^{-\frac{i 2 \pi k \tau}{N}}$. Evaluating the GCC-PHAT of the two

filtered signals yields

$$\begin{aligned}\tilde{R}[m, l] &= \frac{1}{N} \sum_{k=0}^{N-1} \frac{Y_1[k, l] Y_1^*[k, l] e^{-\frac{i2\pi k\tau}{N}}}{|Y_1[k, l] Y_1^*[k, l] e^{-\frac{i2\pi k\tau}{N}}|} e^{\frac{i2\pi km}{N}} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{i2\pi k(m-\tau)}{N}} = \delta_\tau[m],\end{aligned}\tag{7}$$

where δ_τ is a unit pulse centered at time τ . This shows that in an ideal anechoic noise-free environment, applying f_θ to the signals will not prevent the GCC-PHAT from recovering the timing perfectly, as can be seen in Figure 2b. However, in a reverberant and noisy environment, the network is trained to learn to remove these effects and output signal representations that are equal up to a circular time shift.

In order to output a probability distribution over time shifts, the GCC-PHAT output is fed into another network $g_{\theta'} : \mathbb{R}^{(2\tau_{\max}+1) \times L} \rightarrow \mathbb{R}^{(2\tau_{\max}+1)}$, with its own set of parameters θ' that combines the L different correlations and applies softmax normalization in the final layer. Consequently, the final predictions are obtained as

$$\mathbf{p}(m|\mathbf{x}_1, \mathbf{x}_2) = g_{\theta'}(\tilde{\mathbf{R}}),\tag{8}$$

such that $\mathbf{p}(m|\mathbf{x}_1, \mathbf{x}_2)$ contains the probabilities for each time delay $m = -\tau_{\max}, \dots, \tau_{\max}$ considered in the correlation. As can be seen in Figure 2c, the trained classifier is able to combine a set of noisy correlations into an accurate prediction.

The two networks f_θ and $g_{\theta'}$ can be trained jointly by minimizing the cross-entropy (CE) loss function, which for a single training example becomes

$$L(\mathbf{x}_1, \mathbf{x}_2) = - \sum_{m=-\tau_{\max}}^{\tau_{\max}} \delta_\tau[m] \log p(m|\mathbf{x}_1, \mathbf{x}_2),\tag{9}$$

where $\tau = \text{round}((\|\mathbf{r}_1 - \mathbf{r}_s\| - \|\mathbf{r}_2 - \mathbf{r}_s\|)F_s/c)$ is the true time delay rounded to the nearest sample. This approach can be regarded as a form of regression-via-classification (RvC), where the model tries to classify the set of correlations into the correct time delay. In contrast, a regression-based approach, such as the one proposed in [21], tries to estimate τ directly and the model is trained to minimize the mean squared error (MSE). In Section 3 we compare the two methods experimentally in order to justify our RvC approach.

Network architecture. In order to efficiently process the raw waveforms, we use the SincNet [20] architecture, which consists of a series of parallel band-pass filters with learnable cutoff frequencies, in the first layer of f_θ . Specifically, we use 128 filters of length 1023. The following layers consist of regular convolutions with filter lengths of 11, 9 and 7, each with $L = 128$ output channels. Similarly, $g_{\theta'}$ consists of 4 convolutional layers with filters of

length 11, 9, 7 and 5, all of which has 128 output channels, except for the last layer that has a single output channel which models the log-probabilities for each time delay. In both networks, we use BatchNorm [10] and LeakyReLU [15] activations in each layer.

3 Experiments

We perform a series of simulated experiments in order to evaluate our method and compare it to baselines. In order to simulate realistic sound propagation, we use Pyroomacoustics [22], which enables modeling of reverberant indoor environments based on the image source method [1]. The audio signals were collected from the LibriSpeech dataset [18], which contains speech recordings from read audiobooks in English, sampled at $F_s = 16$ kHz. We split the data based on speakers, such that 40 speakers were used for training, 3 for validation and 3 for testing. For each recording we first remove silent parts using a voice activity detector and then extract a 2 second long snippet from each recording. This results in 1892 snippets for training (corresponding to roughly one hour of audio), 188 for validation and 216 for testing. During training, we randomly sample a frame of $N = 2048$ samples for each snippet, while during testing we evaluate on each of 15 non-overlapping windows, for a total of $216 \cdot 15 = 3240$ time delay estimates.

Network training was done inside a simulated room of dimension $7 \times 5 \times 3$ m, with microphones placed roughly in the middle of the room at $\mathbf{r}_1 = [3.5, 2.25, 1.5]^T$ and $\mathbf{r}_2 = [3.5, 2.75, 1.5]^T$ m from the origin. This setup results in a maximum delay of $\tau_{\max} = 23$ samples. The source positions \mathbf{r}_s were sampled from randomly from a uniform distribution over the entire room for each training sample. Furthermore, random reverberation times T_{60} and signal-to-noise ratios (SNR) were uniformly sampled in the ranges $[0.2, 1.0]$ s and $[0, 30]$ dB respectively. We use the Adam optimizer [12] with a batch size of 32, a learning rate of 0.001 with a cosine decay schedule and train the network for 30 epochs.¹

For comparison, we implement the PGCC-PHAT method following the description in [21]. The method uses a CNN that takes several differently weighted GCC-PHAT correlations as input and combines them into a single time delay prediction. Each correlation has a different filter $\phi_\beta[k] = 1/|X_1[k]X_2^*[k]|^\beta$, for $\beta \in \{0, 0.1, \dots, 1\}$. In contrast to our method, the network is trained to minimize the MSE loss.

The trained models were evaluated in a different room with dimensions $6 \times 4 \times 2.5$ m and with the microphones placed at $\mathbf{r}_1 = [3, 1.75, 1.25]^T$ and $\mathbf{r}_2 = [3, 2.25, 1.25]^T$ m respectively, and the source positions were again sampled randomly across the whole room. Each recording was evaluated for SNRs $\in \{0, 6, 12, 18, 24, 30\}$ dB and reverberation times $T_{60} \in \{0.2, 0.4, 0.6, 0.8, 1\}$ s.

¹Code available at: <https://github.com/axeber01/ngcc>

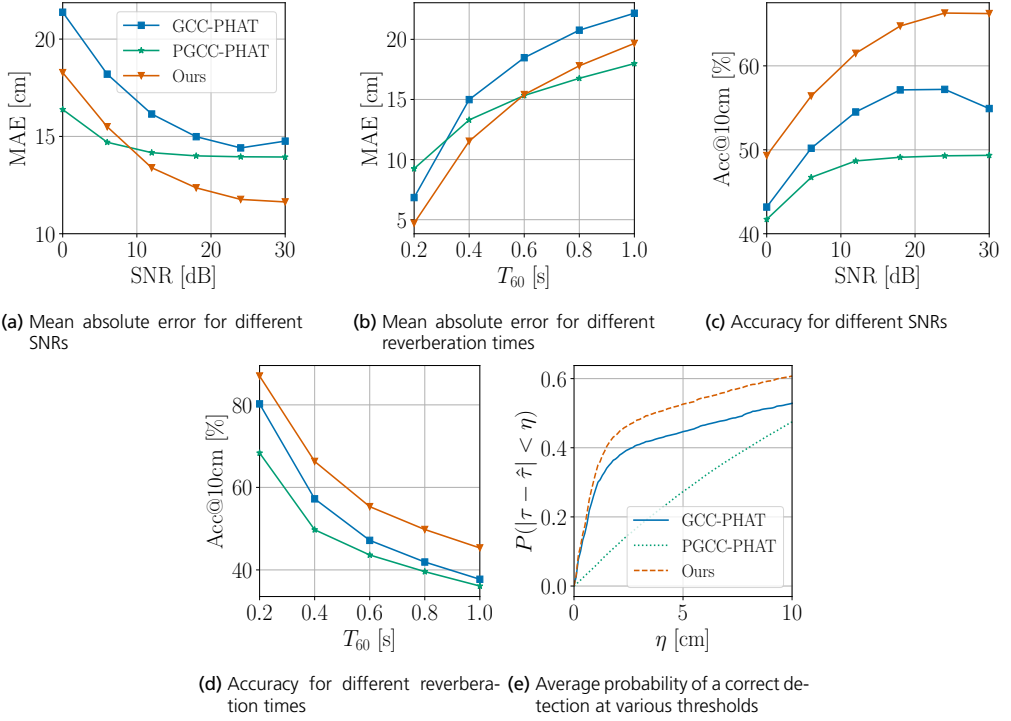


Figure 3: Performance of the different methods in a room of size $6 \times 4 \times 2.5$ m.

The results are presented in Figure 3, where the mean absolute error (MAE) and accuracy is presented for different SNRs and reverberation times. All TDOA errors have been converted to their corresponding distance errors and the accuracy should be interpreted as the probability $P(|\tau - \hat{\tau}| < \eta)$, where $\eta = 10$ cm, since this is a typical level of average precision that can be achieved by an acoustic localization system [17, 26]. For completeness, we show results for lower thresholds in Figure 3e as well. Our method achieves the highest accuracy in all conditions and the lowest MAE for conditions with high SNR or low reverberation time, consistently outperforming the GCC-PHAT baseline. A comparison of error distributions for the different methods in a high SNR environment can be seen in Figure 4. Since PGCC-PHAT has been trained to minimize the MSE, its error distribution has a smaller tail but fails to make accurate predictions within a few centimeters, which is necessary for real-world indoor localization. The large number of predictions at $\hat{\tau} = 0$ is due to imperfect audio pre-processing, which results in some time windows containing long periods of silence.

In order to disentangle the influence of the CE and MSE loss functions, we ablate the two by changing only the last layer of the networks. In Table 1 it can be seen that the higher accuracy of our method cannot be attributed solely to the CE loss, since it is more accurate

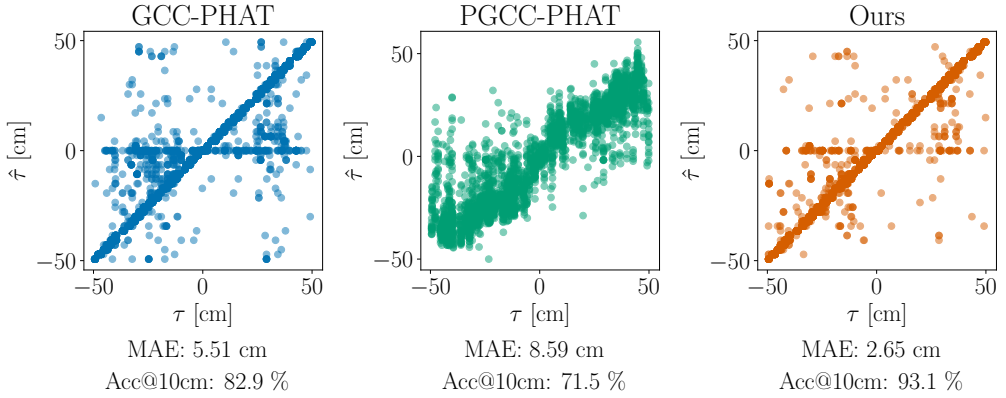


Figure 4: Scatter plots of ground truth and predicted time delays for reverberation time $T_{60} = 0.2$ s and SNR = 30 dB.

than PGCC-PHAT when trained with the same loss function. However, our method is not as effective when trained with the MSE loss. We attribute this to the fact that in contrast to PGCC-PHAT, the entries of our correlation matrix $\tilde{\mathbf{R}}$ do not exhibit any time-smearing, which causes the correlation peaks to be uniformly distributed for incorrect detections. Nevertheless, the RvC approach is preferable for both methods when accuracy is most important. Additionally, the smaller number of parameters in our model makes it more feasible for use in efficient real-time inference on small devices.

In order to demonstrate the effectiveness of using multiple GCC-PHAT correlations, we ablate the number of channels L in the last layer of our feature extractor f_{θ} , leaving the other layers unchanged. In Table 2, it can be seen that adding more channels yields better performance. Moreover, replacing the SincNet layer with a standard learnable convolution with the same filter length results in a performance drop, which motivates the use of learnable bandpass filters in the first layer.

Table 1: Results using different loss functions for reverberation time $T_{60} = 0.2$ s, averaged over all SNRs.

Model	GCC-PHAT	PGCC-PHAT		Ours	
#params	0	11.5M		0.9M	
Loss	-	MSE	CE	MSE	CE
RMSE [cm]	15.21	10.87	13.61	12.44	13.24
MAE [cm]	6.84	6.59	5.39	8.38	5.13
Acc@10cm	80.2	80.6	85.7	71.9	86.5

Table 2: Results for our method using different number of correlation channels L for reverberation time $T_{60} = 0.2$ s, averaged over all SNRs.

#channels L	1	8	32	128	128 w/o SincNet
MAE [cm]	5.59	5.35	5.23	5.13	5.38
Acc@10cm	84.4	85.6	85.8	86.5	85.4

4 Conclusions

We have demonstrated that our proposed method is able to consistently improve detection accuracy over the baseline GCC-PHAT and PGCC-PHAT. Furthermore, as the signal strength increases relative to noise and echoes, in the limit our method is guaranteed to recover the time delay within sample accuracy. Although incorrect detections sometimes results in large errors, this is of less practical importance, since robust localization methods are designed to handle a large fraction of incorrect detections by removing outliers in the measurements [2, 19, 23]. Moreover, it can be used as a drop-in replacement for GCC-PHAT, and the outputs from the SE-CNN can be re-used when considering more than two microphones. We therefore conclude that our method would be a suitable alternative for time delay estimation in real-world speaker localization scenarios.

In future work, we will consider integrating our method into a full sound source localization system. This requires tracking delays over time, as well as considering the geometry of the microphones and sound sources. Here we see further potential of applying machine learning methods in order to tackle the localization problem with an end-to-end approach.

5 Acknowledgments

This work was partially supported by ELIIT and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

References

- [1] J. B. Allen and D. A. Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950, 1979.
- [2] K. Åström, M. Larsson, G. Flood, and M. Oskarsson. Extension of Time-Difference-of-Arrival Self Calibration Solutions using Robust Multilateration. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 870–874. IEEE, 2021.

- [3] S. Burgess, Y. Kuang, and K. Åström. Toa sensor network self-calibration for receiver and transmitter spaces with difference in dimension. *Signal Processing*, 107:33–42, 2015.
- [4] S. Chakrabarty and E. A. Habets. Broadband doa estimation using convolutional neural networks trained with noise signals. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 136–140. IEEE, 2017.
- [5] M. Cobos, F. Antonacci, L. Comanducci, and A. Sarti. Frequency-sliding generalized cross-correlation: A sub-band time delay estimation approach. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1270–1281, 2020.
- [6] L. Comanducci, M. Cobos, F. Antonacci, and A. Sarti. Time difference of arrival estimation from frequency-sliding generalized cross-correlations using convolutional neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2020.
- [7] D. Diaz-Guerra, A. Miguel, and J. R. Beltran. Robust Sound Source Tracking using SRP-PHAT and 3D Convolutional Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:300–311, 2020.
- [8] P.-A. Grumiaux, S. Kitić, L. Girin, and A. Guérin. A survey of sound source localization with deep learning methods. *arXiv preprint arXiv:2109.03465*, 2021.
- [9] W. He, P. Motlicek, and J.-M. Odobez. Deep neural networks for multiple speaker detection and localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 74–79. IEEE, 2018.
- [10] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [11] O. S. Kayhan and J. C. v. Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020.
- [12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
- [13] C. Knapp and G. Carter. The Generalized Correlation Method for Estimation of Time Delay. *IEEE transactions on acoustics, speech, and signal processing*, 24(4):320–327, 1976.
- [14] X. Li, L. Girin, F. Badeig, and R. Horaud. Reverberant sound localization with a robot head based on direct-path relative transfer function. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2819–2826. IEEE, 2016.

-
- [15] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.
- [16] T. N. T. Nguyen, N. K. Nguyen, H. Phan, L. Pham, K. Ooi, D. L. Jones, and W.-S. Gan. A general network architecture for sound event localization and detection using transfer learning and recurrent neural network. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 935–939. IEEE, 2021.
- [17] M. Omologo and P. Svaizer. Use of the crosspower-spectrum phase in acoustic event location. *IEEE Transactions on Speech and Audio Processing*, 5(3):288–292, 1997.
- [18] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An ASR Corpus Based on Public Domain Audio Books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.
- [19] A. Plinge, F. Jacob, R. Haeb-Umbach, and G. A. Fink. Acoustic microphone geometry calibration: An overview and experimental evaluation of state-of-the-art algorithms. *IEEE Signal Processing Magazine*, 33(4):14–29, 2016.
- [20] M. Ravanelli and Y. Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.
- [21] D. Salvati, C. Drioli, and G. L. Foresti. Time Delay Estimation for Speaker Localization Using CNN-Based Parametrized GCC-PHAT Features. In *Proc. Interspeech 2021*, pages 1479–1483, 2021. doi: 10.21437/Interspeech.2021-988.
- [22] R. Scheibler, E. Bezzam, and I. Dokmanić. Pyroomacoustics: A Python Package for Audio Room Simulation and Array Processing Algorithms. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 351–355. IEEE, 2018.
- [23] J. Velasco, D. Pizarro, J. Macias-Guarasa, and A. Asaei. Tdoa matrices: Algebraic properties and their application to robust denoising with missing data. *IEEE Transactions on signal processing*, 64(20):5242–5254, 2016.
- [24] J. M. Vera-Diaz, D. Pizarro, and J. Macias-Guarasa. Towards End-to-End Acoustic Localization using Deep Learning: From Audio Signals to Source Position Coordinates. *Sensors*, 18(10):3418, 2018.
- [25] J. Wang, X. Qian, Z. Pan, M. Zhang, and H. Li. Gcc-phat with speech-oriented attention for robotic sound source localization. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5876–5883. IEEE, 2021.

- [26] L. Wang, N. Kitaoka, and S. Nakagawa. Robust distant speaker recognition based on position-dependent cmn by combining speaker-specific gmm with speaker-adapted hmm. *Speech communication*, 49(6):501–513, 2007.
- [27] X. Xiao, S. Zhao, X. Zhong, D. L. Jones, E. S. Chng, and H. Li. A learning-based approach to direction of arrival estimation in noisy and reverberant environments. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2814–2818. IEEE, 2015.

Paper VI

Proc. Detection and Classification of Acoustic Scenes and Events 2024 Workshop (DCASE2024), pp. 16-20.
This work is licensed under a Creative Commons Attribution 4.0 International License.

Learning Multi-Target TDOA Features for Sound Event Localization and Detection

AXEL BERG^{1,2}, JOHANNA ENGMAN¹, JENS GULIN^{1,3},
KARL ÅSTRÖM¹, MAGNUS OSKARSSON¹,

¹Computer Vision and Machine Learning, Centre for Mathematical Sciences,
Lund University, Sweden
²Arm, Lund, Sweden
³Sony Europe B.V., Lund, Sweden

Abstract: Sound event localization and detection (SELD) systems using audio recordings from a microphone array rely on spatial cues for determining the location of sound events. As a consequence, the localization performance of such systems is to a large extent determined by the quality of the audio features that are used as inputs to the system. We propose a new feature, based on neural generalized cross-correlations with phase-transform (NGCC-PHAT), that learns audio representations suitable for localization. Using permutation invariant training for the time-difference of arrival (TDOA) estimation problem enables NGCC-PHAT to learn TDOA features for multiple overlapping sound events. These features can be used as a drop-in replacement for GCC-PHAT inputs to a SELD-network. We test our method on the STARSS23 dataset and demonstrate improved localization performance compared to using standard GCC-PHAT or SALSA-Lite input features.

1 Introduction

The sound event localization and detection (SELD) task consists of classifying different types of acoustic events, while simultaneously localizing them in 3D space. The DCASE SELD Challenge [1] provides first order ambisonics (FOA) recordings and signals captured from a microphone array (MIC). In recent years, most systems submitted to the challenge have utilized the former format, whereas the latter has been less explored. In this work, we therefore focus on how to better exploit information in the MIC recordings by learning to extract better features.

Generalized cross-correlations with phase transform (GCC-PHAT) [8] combined with

This work was partially supported by the strategic research project ELLIIT and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg (KAW) Foundation. Model training was enabled by the Berzelius resource provided by the KAW Foundation at the National Supercomputer Centre in Sweden.

Code: <https://github.com/axeber01/ngcc-seld/>

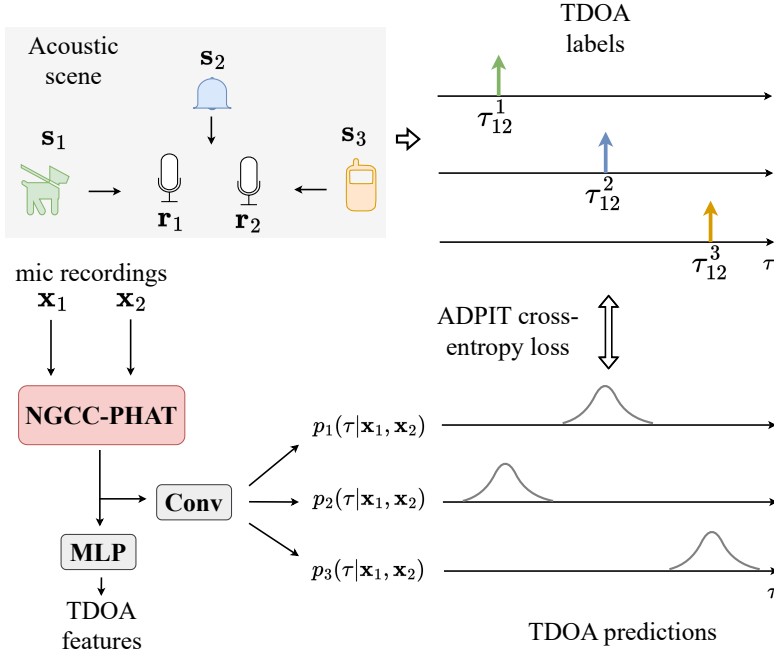


Figure 1: Overview of our pre-training strategy with $K = 3$ tracks. Given a set of sound events, we train a neural GCC-PHAT to predict the TDOA of each event. When the number of sound events is less than K , auxiliary duplication of the labels is used. In this illustration, only two microphones are shown for brevity.

spectral audio features is the basis for most SELD methods for microphone arrays. The spectral features contain important cues on what type of sound event is active, whereas the purpose of GCC-PHAT is to extract the time-differences of arrival (TDOA) for pairs of microphones. The TDOA measurements can then be mapped to direction-of-arrival (DOA) estimates, given the geometry of the array. However, GCC-PHAT is known to be sensitive to noise and reverberation [4]. GCC-PHAT can also fail to separate TDOAs for overlapping events, since two events at different locations can have the same TDOA for a given microphone pair, which yields only one correlation peak.

To improve separation of overlapping events, Xu et al. [26] proposed a beamforming approach, where phase differences from the cross-power spectrum are used as input features. Similarly, Cheng et al. [5] showed that localization performance can be improved by first filtering the audio signals using a sound source separation network before performing feature extraction. Several works [23, 7] have also proposed end-to-end localization from raw audio signals. The most widely adopted input feature is however the spatial cue-augmented log-spectrogram (SALSA) [12] and variants thereof (SALSA-Lite) [13], that combine directional cues with spectral cues in a single feature. This is done by calculating the principal eigenvector of the spatial covariance matrix for the different frequencies in the spectrogram.

Although some recent works [25, 19, 2] have approached TDOA estimation using learning-based methods, there is a lack of research in how to combine this with the SELD task. Berg et al. [2] proposed using a shift-equivariant neural GCC-PHAT (NGCC-PHAT) network. However, this method, as it was originally proposed, only supports single-source TDOA estimation and was not evaluated in a real-world localization scenario.

In this work, we describe how NGCC-PHAT can be trained to extract TDOA features for multiple sound sources. We show that such features can be learnt by employing permutation invariant training, which allows for prediction of TDOAs for multiple overlapping sound events. Furthermore, we show that these features can be used with an existing SELD-pipeline on a real-world dataset, for better performance compared to using traditional input features. The material presented in this work is an extension of our DCASE 2024 challenge submission [3].

2 Method

2.1 Background

Consider an acoustic scene, as shown in Figure 1, with M microphones located at positions $\mathbf{r}_m \in \mathbb{R}^3$ for $m = 1, \dots, M$. Furthermore, let $\mathbf{s}_p \in \mathbb{R}^3$, $p = 1, \dots, P$ denote the locations of the active sound events. For a given time frame, each microphone records a signal x_i , which is composed of the sum of active events as

$$x_i[n] = \sum_{p=1}^P (h_{p,i} * u_p)[n] + w_i[n], \quad n = 1, \dots, N, \quad (1)$$

where u_p is the p :th active event, $h_{p,i}$ is the room impulse response from the p :th event to the i :th microphone, w_i is additive noise and N is the number of samples. Furthermore, we define the TDOA for microphone pair (i, j) and the p :th event as

$$\tau_{ij}^p = \lfloor \frac{F_s}{c} (||\mathbf{s}_p - \mathbf{r}_i||_2 - ||\mathbf{s}_p - \mathbf{r}_j||_2) \rfloor, \quad (2)$$

where F_s is the sampling rate, c is the speed of sound and $\lfloor \cdot \rfloor$ denotes rounding to the nearest integer.

The GCC-PHAT is defined as

$$R_{ij}[\tau] = \frac{1}{N} \sum_{k=0}^{N-1} \frac{X_i[k]X_j^*[k]}{|X_i[k]X_j^*[k]|} e^{\frac{j2\pi k\tau}{N}}, \quad (3)$$

where (X_i, X_j) are the discrete Fourier transforms of (x_i, x_j) . The feature is calculated for time delays $\tau = -\tau_{\max}, \dots, \tau_{\max}$, where $\tau_{\max} = \max_{i,j} \lfloor \|\mathbf{r}_i - \mathbf{r}_j\|_2 F_s / c \rfloor$ is the largest possible TDOA for any pair of microphones. In an anechoic and noise-free environment with a single sound event u_p , this results in $R_{ij}[\tau] = \delta_{\tau_{ij}^p}[\tau]$, where

$$\delta_{\tau_{ij}^p}[\tau] = \begin{cases} 1, & \tau = \tau_{ij}^p, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

In practice, GCC-PHAT will often yield incorrect TDOA estimates due to noise and reverberation. In the case of multiple overlapping sound events, the different events may interfere and result in difficulties resolving peaks in their signal correlations.

NGCC-PHAT attempts to alleviate this problem by filtering the input signals using a learnable filter bank with L convolutional filters, before computing GCC-PHAT features R_{ij}^l , $l = 1, \dots, L$ for each channel in the signals independently. In theory, such a filter bank can perform source separation so that different channels in the NGCC-PHAT correspond to TDOAs for different sound events. Note that for an ideal filter bank that perfectly separates the p :th sound event to the l :th channel, we would have $R_{ij}^l[\tau] = \delta_{\tau_{ij}^p}[\tau]$ in an anechoic and noise-free environment, due to the shift-equivariance of the convolutional filters.

2.2 Permutation Invariant Training for TDOA Estimation

We extend NGCC-PHAT to predict time delays for multiple events in a single time frame using auxiliary duplicating permutation invariant training (ADPIT) [20], by creating separate target labels for each active sound event. This is done by training a classifier network to predict the TDOA of all active events for all pairs of microphones by treating it as a multinomial classification problem. The L correlation features are first processed using another series of convolutional layers with C output channels. These are then projected to K different output tracks which are assigned to the different events. The last layer of the NGCC-PHAT network therefore outputs probability distributions $p_k(\tau | \mathbf{x}_i, \mathbf{x}_j)$ for $k = 1, \dots, K$ over the set of integer delays $\tau \in \{-\tau_{\max}, \dots, \tau_{\max}\}$, as illustrated in Figure 1.

With K as the number of tracks, assume for now that there are also $P = K$ active events. Furthermore, let $\text{Perm}([K])$ denote the set of permutations of the events $\{1, \dots, K\}$. For a single microphone pair (i, j) and an event arrangement $\alpha \in \text{Perm}([K])$, the loss is

calculated using the average cross-entropy over all output tracks as

$$l_\alpha(\mathbf{x}_i, \mathbf{x}_j) = -\frac{1}{K} \sum_{k=1}^K \sum_{\tau=-\tau_{\max}}^{\tau_{\max}} \delta_{\tau_{ij}^{\alpha(k)}}[\tau] \log p_k(\tau | \mathbf{x}_i, \mathbf{x}_j). \quad (5)$$

Due to the ambiguity in assigning different output tracks to different events, we calculate the loss for all possible permutations of the events and use the minimum. The loss is then averaged over all $M(M-1)/2$ microphone pairs, giving the total loss

$$\mathcal{L} = \frac{2}{M(M-1)} \sum_{\substack{i,j=1 \\ i < j}}^M \min_{\alpha \in \text{Perm}([K])} l_\alpha(\mathbf{x}_i, \mathbf{x}_j). \quad (6)$$

Note that this loss function is class-agnostic, since the output tracks are not assigned class-wise. The main purpose of the TDOA features are therefore to provide better features for localization when combined with spectral features that are suitable for classification.

When the assumption $P = K$ does not hold, the formal implication is that α needs to cover another set of event arrangements. Our approach is equivalently to transform each such case into subcases where the assumption holds. Time frames with no active events ($P = 0$) are discarded in the loss calculation, since no TDOA label can be assigned. When $1 \leq P \leq K-1$, we perform auxiliary duplication of events following the method in [20], which makes the loss invariant to both permutations and which events that are duplicated. Furthermore, in the case of $K < P$, it is possible to compute the loss for all subsets of K events from P and use the minimum.

3 Experimental Setup

3.1 Using TDOA Features for SELD

In order to show the benefits of better TDOA features for SELD, we demonstrate how they can be used in conjunction with a SELD-system. This involves two training phases: 1) pre-training of the NGCC-PHAT network for TDOA prediction and 2) training the SELD-network using the TDOA features as input. The NGCC-PHAT network operates on raw audio signals and consists of four convolutional layers, the first being a SincNet [17] layer, and the remaining three use filters of length 11, 9, and 7 respectively. Here, each convolutional layer has $L = 32$ channels and together form the filter bank mentioned in Section 2.1, which is applied independently to audio from the different microphones. GCC-PHAT features are computed channel-wise for each microphone pair, and the features are then processed by another four convolutional layers, where the final layer has $C = 16$ output channels.

The maximum delay used is chosen for compatibility with the setup in the STARSS23 dataset [21], which uses a tetrahedral array with $M = 4$ microphones. The diameter of the array is 8.4 cm, which corresponds to a maximum TDOA of $\tau_{\max} = 6$ delays at a sampling rate of $F_s = 24$ kHz. In total, the TDOA features therefore have shape $[C, M(M - 1)/2, 2\tau_{\max} + 1] = [16, 6, 13]$.

During pre-training for TDOA-prediction, the 16 channels are then mapped by a convolutional layer to $K = 3$ output tracks. Although the maximum polyphony in a single time frame in the dataset is five, we use $K = 3$ tracks since the computational complexity of permutation invariant training scales as $\mathcal{O}(K!)$ and more than three simultaneous events are rare. When more than three events are active, for pre-training we randomly select labels for three events and discard the rest.

When training the SELD-network, we extract the TDOA input features for longer audio signals by windowing the NGCC-PHAT computation without overlap. We use an input duration of 5 second audio inputs, which corresponds to $T = 250$ TDOA features when using a window length of 20 ms. Since the TDOA features are designed to be class-agnostic, we combine them with spectral features for the same time-frame in order to better distinguish between different types of event. For this we use log mel-spectrograms (MS) with $F = 64$ spectral features for each recording.

When merging the spectral features with the TDOA features, we first concatenate the 16 channels for the 6 microphone pairs of the TDOA features, and use a multi-layer perceptron to map the 13 time-delays to 64 dimensions. The TDOA features are then reshaped and concatenated with the M spectral features channel-wise, as shown in Figure 2, resulting in a combined feature size of $[CM(M - 1)/2 + M, T, F] = [100, 250, 64]$.

The combined feature is passed through a small convolutional network with 64 output channels with pooling over the time and spectral dimensions. Here we use two pooling variants that determine the size of the input features to the SELD-network: 1) pooling over 5 time windows and 4 frequencies, which produces features of size $[64, 50, 16]$, or 2) pooling over 5 time windows and no pooling over frequencies, which results in features of size $[64, 50, 64]$. We call the resulting network variants *Small* and *Large* for this reason.

For SELD-training, we use a CST-Former [22] network that consists of Transformer blocks, where each block contains three self-attention modules: temporal attention, spectral attention and channel attention with unfolded local embedding. We use the default configuration with two blocks, each with eight attention heads, and refer to [22] for more details about this architecture.

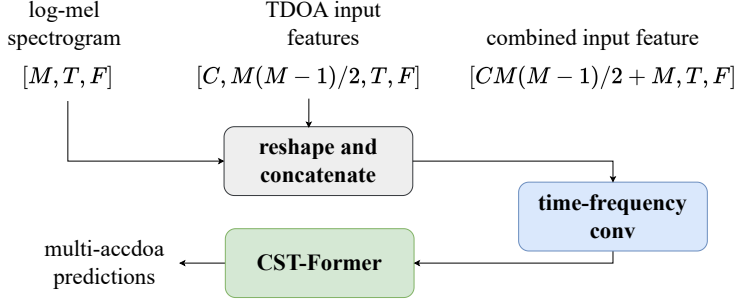


Figure 2: Illustration of how TDOA features are used together with log mel-spectrograms as input to the CST-Former network.

3.2 Dataset and Model Training

We train all our models on a mixture of real spatial audio recordings and simulated recordings. The real recordings are from the STARSS23 [21] audio-only dev-train dataset, which consists of about 3.5 hours of multi-channel audio recordings. The dataset has up to 5 simultaneous events from 13 different classes. For data augmentation, we use channel-swapping [24], which expands the dataset by a factor of 8 by swapping the input channels and corresponding DOA labels in different combinations.

The simulated data is provided as a part of the DCASE 2024 challenge [9] and consists of 20 hours of synthesized recordings, where the audio is taken from the FSD50K [6] dataset. In addition, we generate an additional 2 hours of synthesized recordings using Spatial Scaper [18] with impulse responses from the TAU [16] and METU [14] databases. This additional data contains sounds from classes that occur rarely in STARSS23, namely “bell”, “clapping”, “doorCupboard”, “footsteps”, “knock” and “telephone”. The total amount of training data is about 50 hours.

The NGCC-PHAT network was trained for one epoch with a constant learning rate of 0.001, after which the weights were frozen. The CST-Former network was then trained for 300 epochs using the AdamW optimizer [11] with a batch size of 64, a cosine learning rate schedule starting at 0.001 and weight decay of 0.05. The mean squared error was used as loss function with labels in the Multi-ACCDOA [20] format, with distances included as proposed in [10]. In order to penalize errors in predicted distance relative to the proximity of the sound events, we scale loss-terms for the distance error with the reciprocal of the ground truth distance.

Evaluations were done using the DCASE 2024 SELD challenge metrics [1, 15]. This includes the location dependent F-score F_{LD} , the DOA error $DOAE$ and the relative distance error RDE , which is the distance error divided by the ground truth distance to the event. Each metric is calculated class-wise and then macro-averaged across all classes. Fur-

Table 1: Macro-averaged test results on STARSS23 [21] dev-test.

Input feature	$F_{LD} \uparrow$	$DOAE \downarrow$	$RDE \downarrow$	#params
CST-Former Small				
GCC + MS	15.7 ± 1.0	27.7 ± 2.1	0.78 ± 0.02	550K
SALSA-Lite	24.6 ± 2.0	27.0 ± 1.2	0.41 ± 0.02	530K
NGCC + MS	26.0 ± 2.0	25.8 ± 2.3	0.42 ± 0.01	663K
CST-Former Large				
GCC + MS	14.2 ± 1.1	28.4 ± 1.9	0.84 ± 0.03	1.37M
SALSA-Lite	26.1 ± 1.0	26.4 ± 3.6	0.42 ± 0.02	1.35M
NGCC + MS	28.2 ± 2.8	23.2 ± 1.8	0.50 ± 0.02	1.49M

thermore, the location dependent F-score only counts predicted events as true positives if they are correctly classified and localized, such that predictions with $DOAE$ larger than $T_{DOA} = 20^\circ$ or RDE larger than $T_{RD} = 1$ are counted as false positives. We focus on evaluating the performance of our method compared to that of other commonly used input features with the same SELD-network, and do not compare to other (e.g. FOA-based) state-of-the-art methods.

4 Results

Our main results are presented in Table 1, where we compare our method to GCC with MS and to SALSA-Lite. Our method performs better in terms of F_{LD} and $DOAE$, for both the Small and Large variant of the network, although SALSA-Lite has the lowest RDE for the Large variant. When increasing the model size, the results improve for both SALSA-Lite and NGCC, but not for GCC. Since GCC features are less informative, the increase in model size results in overfitting. The same can be said for the increase in RDE when using NGCC + MS, since the TDOA features from both GCC and NGCC mostly contain angular cues, but less information about spatial distance. Note that GCC + MS and NGCC + MS use exactly the same CST-Former architecture, so the extra parameter count when using NGCC comes from the pre-trained feature extractor. When using SALSA-Lite, the pooling operations in the convolutional layers were adjusted in order to achieve a similar model size.

Table 2: Ablations of the number input channels used in the TDOA input features for CST-Former Small.

C	$F_{LD} \uparrow$	$DOAE \downarrow$	$RDE \downarrow$	#params
1	24.4 ± 2.3	29.7 ± 3.3	0.44 ± 0.08	608K
4	24.2 ± 0.8	23.2 ± 2.5	0.46 ± 0.01	619K
16	26.0 ± 2.0	25.8 ± 2.3	0.42 ± 0.01	663K

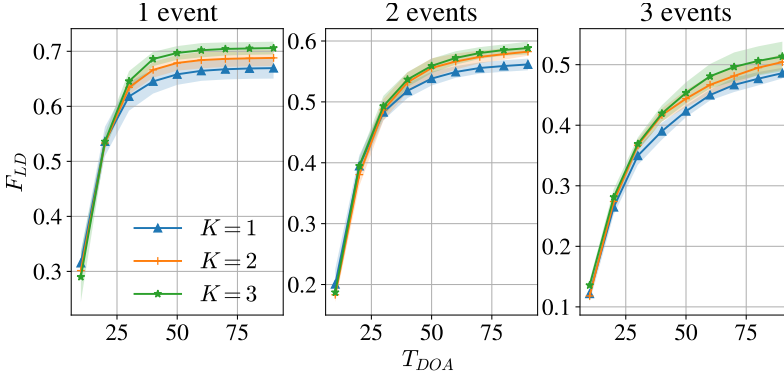


Figure 3: Micro-averaged F-score as a function of the angular threshold T_{DOA} using different number of output tracks K during TDOA pre-training. Evaluation was done using CST-Former Small.

In order to verify the importance of using more than one input channel for TDOA features, we ablate the number of channels C in the NGCC-PHAT network. The results are shown in Table 2, where it can be seen that increasing the number of channels from 1 to 16 increases performance in terms of all metrics. This agrees with the intuition that using more than one input channels enables the pre-training to better separate spatial cues from different events. Furthermore, the cost for increasing the number channels in terms of the increase in model parameters is relatively small.

We also ablate the number of tracks K used for TDOA-prediction during pre-training, and present the location dependent F-score for values of T_{DOA} in Figure 3. Due to the sensitivity of the macro-averaged F-score to incorrect predictions for rare classes in the test data, we instead use the micro-averaged statistic. At the default 20° threshold, the effect of increasing the number of tracks is small, but asymptotically it is clear that using $K = 3$ tracks increases the F-score regardless of how many events are active. Note that the number of tracks only affects the complexity in the pre-training stage of NGCC-PHAT, and not the overall parameter count of the final model, since all C channels are used as input to the network, and the mapping to K tracks can be discarded.

Finally, we show examples of TDOA predictions in Figure 4. When the TDOAs of the events are well-separated, the different tracks yield different peaks at approximately the correct time delays. However, for the microphone pairs where events are tightly spaced,

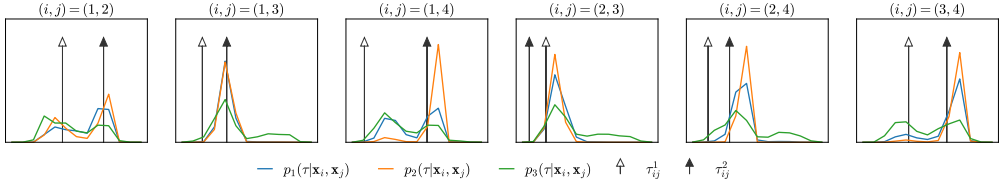


Figure 4: An example of the TDOA predictions $p_k(\tau|\mathbf{x}_i, \mathbf{x}_j)$ from the pre-trained NGCC-PHAT network using $K = 3$ output tracks. Predictions are shown for all six microphone combinations (i, j) at a single time frame with two events and ground truth TDOAs τ_{ij}^1 and τ_{ij}^2 .

the predictions fail to separate the different TDOAs.

5 Conclusions

In this work we proposed an input feature based on NGCC-PHAT and showed its usefulness as input to a SELD-network. Permutation invariant training for the TDOA estimation problem enabled NGCC-PHAT to learn TDOA features for multiple overlapping sound events, and improved SELD performance compared to using GCC-PHAT or SALSA-Lite input features.

These results indicate that our NGCC-PHAT pre-training for TDOA classification provides a good feature extractor for the SELD task. Intuitively, better TDOA prediction in the feature extractor ought to yield better SELD results, but further studies are needed to validate this. Evaluating TDOA prediction performance would however involve new methodology, such as heuristics for peak selection from the output tracks, as well as selecting useful evaluation metrics. The downstream network could be resilient to some type of information our current loss function aims to suppress. In addition, a source-wise or class-wise TDOA format could be beneficial. We therefore anticipate future work to explore other pre-training options and end-to-end training.

Focusing on the feature extractor, we made minimal effort to address the other challenges of the dataset. We leave for future work to incorporate known techniques, such as class balancing, additional data augmentation, temporal filtering and ensemble voting.

References

- [1] Audio and Audiovisual Sound Event Localization and Detection with Source Distance Estimation.
<https://dcase.community/challenge2024/task-audio-and->

audiovisual-sound-event-localization-and-detection-with-source-distance-estimation, 2024. [Accessed 2024-07-03].

- [2] A. Berg, M. O'Connor, K. Åström, and M. Oskarsson. Extending GCC-PHAT using Shift Equivariant Neural Networks. In *Proc. Interspeech 2022*, pages 1791–1795, 2022. doi: 10.21437/Interspeech.2022-524.
- [3] A. Berg, J. Engman, J. Gulin, K. Åström, and M. Oskarsson. The LU System for DCASE 2024 Sound Event Localization and Detection Challenge. Technical report, DCASE2024 Challenge, June 2024.
- [4] B. Champagne, S. Bédard, and A. Stéphenne. Performance of time-delay estimation in the presence of room reverberation. *IEEE Transactions on Speech and Audio Processing*, 4(2):148–152, 1996.
- [5] S. Cheng, J. Du, Q. Wang, Y. Jiang, Z. Nian, S. Niu, C.-H. Lee, Y. Gao, and W. Zhang. Improving sound event localization and detection with class-dependent sound separation for real-world scenarios. In *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 2068–2073. IEEE, 2023.
- [6] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra. FSD50K: An open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021.
- [7] Y. HE and A. Markham. SoundDoA: Learn Sound Source Direction of Arrival and Semantics from Sound Raw Waveforms. In *Proc. Interspeech 2022*, pages 2408–2412, 2022. doi: 10.21437/Interspeech.2022-378.
- [8] C. Knapp and G. Carter. The Generalized Correlation Method for Estimation of Time Delay. *IEEE transactions on acoustics, speech, and signal processing*, 24(4):320–327, 1976.
- [9] D. A. Krause and A. Politis. [DCASE2024 Task 3] Synthetic SELD mixtures for baseline training, Apr. 2024. URL <https://doi.org/10.5281/zenodo.10932241>.
- [10] D. A. Krause, A. Politis, and A. Mesaros. Sound event detection and localization with distance estimation. *arXiv preprint arXiv:2403.11827*, 2024.
- [11] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [12] T. N. T. Nguyen, D. L. Jones, K. N. Watcharasupat, H. Phan, and W.-S. Gan. SALSA-Lite: A Fast and Effective Feature for Polyphonic Sound Event Localization and Detection with Microphone Arrays. In *ICASSP 2022-2022 IEEE International*

- Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 716–720. IEEE, 2022.
- [13] T. N. T. Nguyen, D. L. Jones, K. N. Watcharasupat, H. Phan, and W.-S. Gan. SALSA-Lite: A Fast and Effective Feature for Polyphonic Sound Event Localization and Detection with Microphone Arrays. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 716–720. IEEE, 2022.
 - [14] O. Olgun and H. Hacıhabiboglu. METU SPARG Eigenmike em32 Acoustic Impulse Response Dataset v0.1.0, Apr. 2019. URL <https://doi.org/10.5281/zenodo.2635758>.
 - [15] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, and T. Virtanen. Overview and evaluation of sound event localization and detection in dcase 2019. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:684–698, 2020.
 - [16] A. Politis, S. Adavanne, and T. Virtanen. TAU Spatial Room Impulse Response Database (TAU-SRIR DB), Apr. 2022. URL <https://doi.org/10.5281/zenodo.6408611>.
 - [17] M. Ravanelli and Y. Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.
 - [18] I. R. Roman, C. Ick, S. Ding, A. S. Roman, B. McFee, and J. P. Bello. Spatial Sca- per: a library to simulate and augment soundscapes for sound event localization and detection in realistic rooms. *arXiv preprint arXiv:2401.12238*, 2024.
 - [19] D. Salvati, C. Drioli, and G. L. Foresti. Time Delay Estimation for Speaker Local- ization Using CNN-Based Parametrized GCC-PHAT Features. In *Interspeech*, pages 1479–1483, 2021.
 - [20] K. Shimada, Y. Koyama, S. Takahashi, N. Takahashi, E. Tsunoo, and Y. Mitsufuji. Multi-ACCDOA: Localizing and detecting overlapping sounds from the same class with auxiliary duplicating permutation invariant training. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 316–320. IEEE, 2022.
 - [21] K. Shimada, A. Politis, P. Sudarsanam, D. A. Krause, K. Uchida, S. Adavanne, A. Hakala, Y. Koyama, N. Takahashi, S. Takahashi, T. Virtanen, and Y. Mitsufuji. STARSS23: An audio-visual dataset of spatial recordings of real scenes with spatiotemporal annotations of sound events. In *Advances in Neural Information Pro- cessing Systems*, volume 36, pages 72931–72957. Curran Associates, Inc., 2023.

-
- [22] Y. Shul and J.-W. Choi. CST-Former: Transformer with channel-spectro-temporal attention for sound event localization and detection. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8686–8690. IEEE, 2024.
- [23] H. Sundar, W. Wang, M. Sun, and C. Wang. Raw waveform based end-to-end deep convolutional network for spatial localization of multiple acoustic sources. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4642–4646, 2020. doi: 10.1109/ICASSP40776.2020.9054090.
- [24] Q. Wang, J. Du, H.-X. Wu, J. Pan, F. Ma, and C.-H. Lee. A four-stage data augmentation approach to resnet-conformer based acoustic modeling for sound event localization and detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1251–1264, 2023.
- [25] X. Xiao, S. Zhao, X. Zhong, D. L. Jones, E. S. Chng, and H. Li. A learning-based approach to direction of arrival estimation in noisy and reverberant environments. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2814–2818. IEEE, 2015.
- [26] W. Xue, Y. Tong, C. Zhang, G. Ding, X. He, and B. Zhou. Sound Event Localization and Detection Based on Multiple DOA Beamforming and Multi-Task Learning. In *Proc. Interspeech 2020*, pages 5091–5095, 2020. doi: 10.21437/Interspeech.2020-2759.

Paper VII

Reprinted from Proc. 2024 14th International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-8, © 2024, with permission from IEEE

wav2pos: Sound Source Localization using Masked Autoencoders

AXEL BERG^{1,2}, JENS GULIN³, MARK O’CONNOR⁴, CHUTENG ZHOU²,
KALLE ÅSTRÖM¹, MAGNUS OSKARSSON¹

¹Centre for Mathematical Sciences, Lund University, ²Arm, ³Sony, ⁴Tenstorrent

Abstract: We present a novel approach to the 3D sound source localization task for distributed ad-hoc microphone arrays by formulating it as a set-to-set regression problem. By training a multi-modal masked autoencoder model that operates on audio recordings and microphone coordinates, we show that such a formulation allows for accurate localization of the sound source, by reconstructing coordinates masked in the input. Our approach is flexible in the sense that a single model can be used with an arbitrary number of microphones, even when a subset of audio recordings and microphone coordinates are missing. We test our method on simulated and real-world recordings of music and speech in indoor environments, and demonstrate competitive performance compared to both classical and other learning based localization methods.

1 Introduction

Mapping, positioning and localization are key enabling technologies for a wide range of applications. Thanks to its global coverage and scalability, global navigation satellite systems (GNSS) have become the de-facto standard for outdoor localization. However, for localization in urban areas, indoor environments and underground, as well as in safety-critical applications, GNSS technology cannot deliver the accuracy, reliability and coverage needed. Many sensor modalities and setups can be used to address these issues. In this paper we focus our attention on sound source localization (SSL), which is the task of determining the location of one or several sound sources using recordings from a microphone array.

Depending on the setup, the sound source position can be estimated in several ways. For fixed equidistant microphones with small physical spacing, localization is typically performed by estimating the direction of arrival (DOA) using e.g. steered-response power with phase transform (SRP-PHAT) [10, 7], spectrograms [39, 30] or raw waveforms [19] as input features.

This work was partially supported by the strategic research project ELLIIT and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg (KAW) Foundation. Model training was enabled by the Berzelius resource provided by the KAW Foundation at the National Supercomputer Centre in Sweden. We thank Martin Larsson and Erik Tegler for assistance with data formatting. We also thank Malte Larsson and Gabrielle Flood for their feedback on the paper.

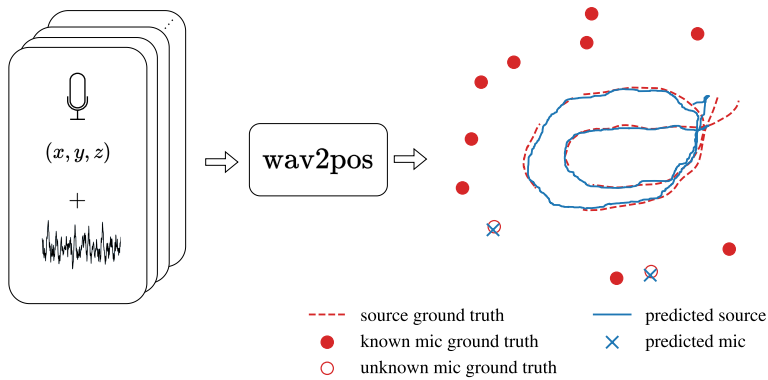


Figure 1: Method overview: **wav2pos** can simultaneously localize a moving sound source and several microphones given audio recordings and microphone coordinates on a frame-by-frame basis. Here, predictions on the **music3** recording from the LuViRa dataset [40] are shown (viewed from above), where a moving median filter has been applied to predictions for better visualization.

When the microphone positions are distributed around the sound source in an ad-hoc fashion, it is possible to estimate the 3D location of the sound source with respect to some coordinate system, given the microphone positions. Depending on whether the sound source is time-synchronized with the microphones or not this is known as trilateration or multilateration, respectively.

Classical methods. For trilateration, there is a large body of previous work. The minimal amount of data needed is when the number of distance measurements equals the spatial dimension, and this problem has a closed-form solution [36, 28, 8]. For the over-determined problem, finding the maximum likelihood (ML) estimate given Gaussian noise in the measurements is a nonlinear, non-smooth and non-convex problem. Unlike the minimal case, there is no closed-form solution. A number of iterative methods, with various convergence guaranties, exist [27, 4, 23, 34]. Simplifications to the ML problem, include relaxations by minimizing the error in the squared distance measurements [25, 41, 3, 6]. Various heuristics can be used to arrive at a linear formulation, see [35] and references therein. For the multilateration problem, classical methods rely on estimating the time-differences of arrival (TDOA) between pairs of microphones using pairwise feature extractors, where the most common one is the generalized cross-correlation with phase transform (GCC-PHAT) [24]. TDOA measurements can then be used to perform multilateration, where the sound source location is obtained by solving a system of equations using e.g. the least squares method or a minimal solver [1].

Learning based methods. Learning based methods have been used extensively for TDOA [32, 5, 31] and DOA [10, 7, 39, 30, 19] estimation. However, there is a lack of research on learning based methods for localization using distributed microphone arrays. Vera-Diaz

et al. [38] used a convolutional neural network to directly regress the source coordinates, but only for a fixed microphone array. Grinstein et al. [16] proposed a dual-input neural network for end-to-end SSL with spatial coordinates in 2D, where both audio signals and microphone coordinates are used as input. This allows the model to be trained in setups with ad-hoc arrays where the microphone locations are not fixed. However, the network is limited to using a fixed number of inputs, which prevents the user from adding or removing microphones to the array at inference time. Similarly, in [15] a graph neural network is proposed for the same task that works with variable number of microphones by aggregating over GCC-PHAT features, which are used as inputs to the network. Similar architectures have also been proposed in [14, 11], where a transformer [37] architecture is used for localization on a 2D grid. However, these methods fail to scale to three dimensions since the discretization of large spaces becomes infeasible. For further reading about prior work, we refer the reader to the extensive survey published in [17].

Main contributions. In this work, we present a novel method for single-source 3D SSL that directly predicts the sound source coordinates using an ad-hoc distributed microphone array. Inspired by the success of masked autoencoders in natural language processing [9], computer vision [18], audio processing [21] and combinations thereof [12], we formulate the SSL problem as a multi-modal set-to-set regression problem, which allows our method to localize not only the sound source, but also solve a variety of similar problems where audio or locations are missing for some microphones, as shown in Figure 1.

2 Method

Problem setup. Consider M microphones with ad-hoc coordinates $\mathbf{r}_m \in \mathbb{R}^3$, $m = 1, \dots, M$ and *one* audio source located at $\mathbf{r}_0 \in \mathbb{R}^3$. For a given time slot of N samples, the source emits a signal $\mathbf{s}_0 \in \mathbb{R}^N$ and each microphone receives a delayed and noisy copy $\mathbf{s}_m \in \mathbb{R}^N$, $m = 1, \dots, M$, of the signal that depends on the impulse response h_m from the source position to each microphone,

$$s_m[n] = (h_m * s_0)[n] + w_m[n], \quad n = 1, \dots, N, \quad (1)$$

where w_m can be approximated as i.i.d. Gaussian noise and $(*)$ denotes the convolution operator. The window length N is assumed to be small enough for the sound source to be modeled as stationary. The SSL task is to recover the audio source location \mathbf{r}_0 , given the recordings of each microphone and their known locations. In the more general setup, the task can be extended to predict some unknown microphone locations as well. Note that we only consider a single sound event for each prediction, and finding the trajectory of a moving sound source thus amounts to making predictions on a frame-by-frame basis. In the general, full calibration problem, it is not possible to estimate arbitrary microphone positions from a single sound event using only distance measurements. However, in our

simplified problem there is a limited number of possible microphone positions in the training data. Furthermore, other spatial cues such as the acoustic features of the environment can be learnt by the model, which makes the problem setup tractable.

Masked autoencoders for localization. The main idea of our method is to consider the SSL problem as function approximation over the set $\{\mathbf{s}_m, \mathbf{r}_m\}_{m=0}^M$ of audio signals and locations. This allows for exploiting redundancy in the data by *masking*, where missing inputs are filled in by the model. In this context, masking can be used both as a training strategy that enables it to perform localization using different microphone array setups, and to predict missing microphone coordinates. Note that while masked autoencoders are often trained in a self-supervised manner, our method is fully supervised, but random masking is used during training in order to increase robustness to missing inputs.

Let $\mathcal{S} = \{m : \mathbf{s}_m \text{ not masked}\}$ and $\mathcal{R} = \{m : \mathbf{r}_m \text{ not masked}\}$ denote the set of non-masked recorded audio signals and coordinates respectively, with set sizes $K_{\mathcal{S}} = |\mathcal{S}|$ and $K_{\mathcal{R}} = |\mathcal{R}|$. Using the set of non-masked inputs, we seek to learn a function f_{θ} that outputs predictions $\hat{\mathbf{s}}_m, \hat{\mathbf{r}}_m$ for the complete set:

$$\{\hat{\mathbf{s}}_m, \hat{\mathbf{r}}_m\}_{m=0}^M = f_{\theta}(\{\mathbf{s}_{m_s}\}_{m_s \in \mathcal{S}}, \{\mathbf{r}_{m_r}\}_{m_r \in \mathcal{R}}). \quad (2)$$

The function approximation model consists of an encoder, which operates on the non-masked subset of inputs, and a decoder that forms predictions on the entire set. Both the encoder and decoder consist of sequential transformer blocks that process audio and coordinate tokens jointly, as shown in Figure 2. In other words, both audio and coordinate tokens are treated as elements of the same unordered set by each transformer block.

We train our method using mean squared error loss on the sound source coordinate, the masked microphone coordinates and the non-masked audio. Reconstructing masked audio patches requires learning impulse responses across the room, which is out of the scope of this work, and initial experiments showed poor performance for that task. For audio prediction, we therefore restrict the loss to the non-masked audio tokens, since this allows the model to learn to perform audio de-noising. Thus, the total loss becomes

$$L = \frac{\lambda_{\text{audio}}}{K_{\mathcal{S}}} \sum_{m \in \mathcal{S}} \|\hat{\mathbf{s}}_m - \mathbf{s}_m\|_2^2 + \lambda_{\text{source}} \|\hat{\mathbf{r}}_0 - \mathbf{r}_0\|_2^2 + \frac{\lambda_{\text{mic}}}{M - K_{\mathcal{R}}} \sum_{m \notin \mathcal{R}, m \geq 1} \|\hat{\mathbf{r}}_m - \mathbf{r}_m\|_2^2, \quad (3)$$

where $\lambda_{\text{audio}}, \lambda_{\text{source}}, \lambda_{\text{mic}}$ are hyperparameters that balance the contribution of the source localization error, microphone localization error and audio reconstruction error, respectively. We will now proceed to describe the method in more detail.

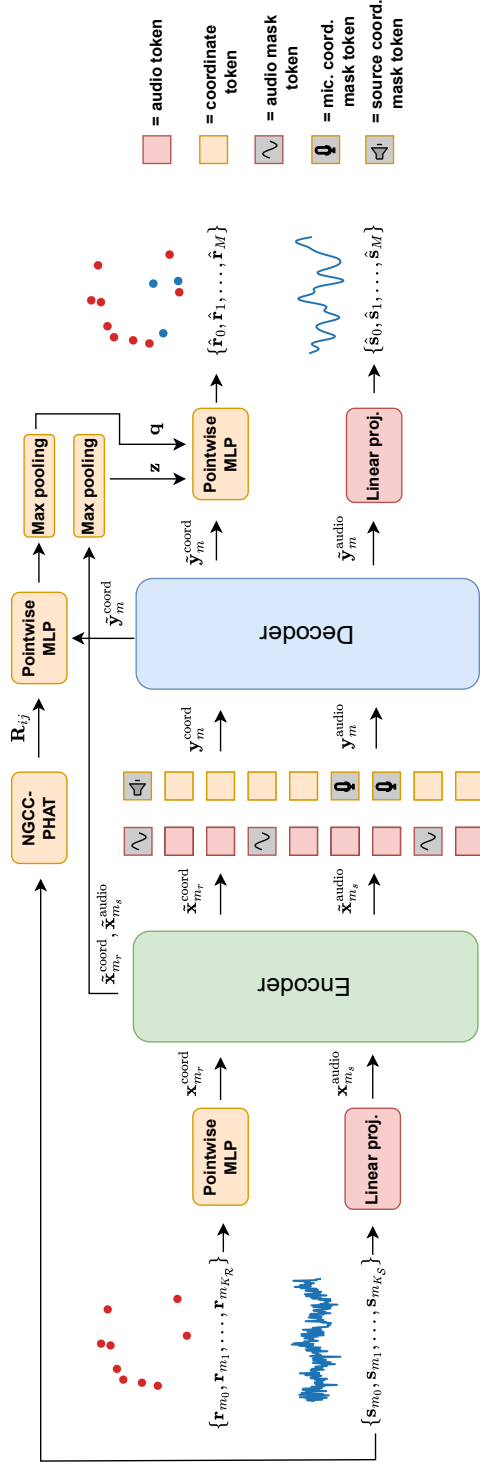


Figure 2: High-level illustration of the proposed wav2pos method. Modality embedding (added before encoder and decoder) and pairwise positional encoding (added before decoder) are omitted for brevity. The mask tokens are not generated by the encoder, but appended as learnable tokens in the input sequence to the decoder.

Feature embedding. Audio snippets are first processed individually for each microphone using a linear projection in order to form input tokens $\mathbf{x}_{m_s}^{\text{audio}} = \mathbf{W}_{\text{enc}} \mathbf{s}_{m_s}$, where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{d \times N}$ and d is the embedding dimension. Similarly, the microphone coordinates are projected to the same embedding space using a point-wise MLP: $\mathbb{R}^3 \rightarrow \mathbb{R}^d$ with one hidden layer and batch normalization [22] that computes the coordinate tokens $\mathbf{x}_{m_r}^{\text{coord}}$. In order to let the model distinguish between the two modalities of tokens, we follow [12] and add learnable modality embeddings $\mathbf{v}_{\text{enc}}^{\text{audio}}, \mathbf{v}_{\text{enc}}^{\text{coord}} \in \mathbb{R}^d$ to each token according to its modality. The audio and coordinate features are then processed jointly by a series of D sequential transformer encoder blocks with layer normalization [2] and GELU activations [20].

After the final encoder layer, two types of learnable mask tokens $\mathbf{u}^{\text{audio}}, \mathbf{u}^{\text{coord}} \in \mathbb{R}^d$ are appended to the output set for each input that was masked out from the input. Additionally a special mask token $\mathbf{u}^{\text{source}} \in \mathbb{R}^d$ for the sound source coordinate is appended, and new modality embeddings for the decoder $\mathbf{v}_{\text{dec}}^{\text{audio}}, \mathbf{v}_{\text{dec}}^{\text{coord}} \in \mathbb{R}^d$ are added to all tokens.

Pairwise positional encoding. Since the source coordinate prediction should be invariant to permutations of the microphone order (and all other outputs should be equivariant), we do not add any form of positional encoding in the usual sense that encodes the relative or absolute ordering of tokens, as is typically done when using transformers for sequence modeling. However, the decoder still needs to be informed about which audio snippet corresponds to which microphone location and vice versa, or whether the corresponding audio/coordinate token was masked. Therefore, we propose a pairwise message-passing embedding that communicates information between tokens originating from the same microphone. The messages are computed using two separate functions $\gamma^{a \rightarrow c}, \gamma^{c \rightarrow a} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ that are implemented as MLPs with a single hidden layer. In total, the inputs $\mathbf{y}_m^{\text{audio}}, \mathbf{y}_m^{\text{coord}}$ to the decoder are given by

$$\begin{aligned} \mathbf{y}_m^{\text{audio}} &= \mathbf{t}_m^{\text{audio}} + \mathbf{v}_{\text{dec}}^{\text{audio}} + \gamma^{c \rightarrow a}(\mathbf{t}_m^{\text{coord}}), \\ \mathbf{y}_m^{\text{coord}} &= \mathbf{t}_m^{\text{coord}} + \mathbf{v}_{\text{dec}}^{\text{coord}} + \gamma^{a \rightarrow c}(\mathbf{t}_m^{\text{audio}}), \end{aligned} \quad (4)$$

for $m = 0, \dots, M$ and where

$$\mathbf{t}_m^{\text{audio}} = \begin{cases} \tilde{\mathbf{x}}_m^{\text{audio}}, & m \in \mathcal{S} \\ \mathbf{u}^{\text{audio}}, & m \notin \mathcal{S} \end{cases} \quad \mathbf{t}_m^{\text{coord}} = \begin{cases} \tilde{\mathbf{x}}_m^{\text{coord}}, & m \in \mathcal{R} \\ \mathbf{u}^{\text{source}}, & m = 0 \\ \mathbf{u}^{\text{coord}}, & m \notin \mathcal{R} \end{cases}, \quad (5)$$

and $\tilde{\mathbf{x}}_m^{\text{audio}}, \tilde{\mathbf{x}}_m^{\text{coord}}$ are the encoder outputs. The tokens are then passed through the decoder, which, similarly to the encoder, consists of D sequential transformer layers.

At the output of the decoder, features $\tilde{\mathbf{y}}_m^{\text{audio}}, \tilde{\mathbf{y}}_m^{\text{coord}}$ are collected for all audio sequences and coordinates. Audio reconstructions are formed by using a simple linear projection as $\hat{\mathbf{s}}_m = \mathbf{W}_{\text{dec}} \tilde{\mathbf{y}}_m^{\text{audio}}$ for $m \in \mathcal{S}$, where $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{N \times d}$.

Time delay feature module. For the coordinate predictions, we use additional information from previous layers by computing a global feature $\mathbf{z} \in \mathbb{R}^d$ by max-pooling over all encoder outputs as $\mathbf{z} = \max_{m_s \in \mathcal{S}, m_r \in \mathcal{R}} (\tilde{\mathbf{x}}_{m_s}^{\text{audio}}, \tilde{\mathbf{x}}_{m_r}^{\text{coord}})$. Similarly to the method proposed in [15], we also use TDOA features $\mathbf{R}_{ij} \in \mathbb{R}^{2\tau+1}$, which we obtain from a pre-trained NGCC-PHAT [5] for all non-masked pairs of audio inputs, where τ is determined by the maximum possible delay between microphones. We then combine each time-delay feature with coordinate features from the two corresponding microphones, and pool over all $M(M-1)$ microphone pairs in order to form a global feature which contains information about all TDOA measurements as

$$\mathbf{q} = \max_{i,j \in \mathcal{S}, i \neq j} \varphi \left(\mathbf{R}_{ij}, \tilde{\mathbf{y}}_i^{\text{coord}}, \tilde{\mathbf{y}}_j^{\text{coord}} \right), \quad (6)$$

where $\varphi : \mathbb{R}^{2(\tau+d)+1} \rightarrow \mathbb{R}^d$ is a single hidden-layer MLP. In order to form the final predictions, we use two separate MLPs $\psi_{\text{source}}, \psi_{\text{mic}} : \mathbb{R}^{3d} \rightarrow \mathbb{R}^3$ that produce coordinate predictions for the sound source and microphones as

$$\begin{aligned} \hat{\mathbf{r}}_0 &= \psi_{\text{source}}(\tilde{\mathbf{y}}_0^{\text{coord}}, \mathbf{z}, \mathbf{q}), \\ \hat{\mathbf{r}}_m &= \psi_{\text{mic}}(\tilde{\mathbf{y}}_m^{\text{coord}}, \mathbf{z}, \mathbf{q}), \quad m = 1, \dots, M. \end{aligned} \quad (7)$$

Masking strategy. When training the model, different masking strategies can be used depending on the use case. If the number of microphones is known to be fixed at inference time, we only mask out the audio and coordinates of the sound source, i.e. $\mathcal{R} = \mathcal{S} = \{1, \dots, M\}$. When the number of microphones available is not fixed, we instead randomly mask a subset of both the audio snippets and coordinates in order to make the model more robust to using a variable number of microphones. A unique solution to the multilateration problem requires four TDOA measurements from five microphones [13], and therefore we always restrict masking such that $|\mathcal{S} \cap \mathcal{R}| \geq 5$. We also require that not both audio and coordinates from the same microphone are masked.

3 Experimental Results

Real indoor recordings. We evaluate our method on the LuViRA [40] audio-only dataset, which contains eight real-world recordings, about one minute long, of music or speech in an indoor environment. Each recording is captured by 11 stationary and synchronized microphones and the speaker location ground truth is given by a motion capture system. An additional 12:th microphone is placed next to the speaker, and can be used as stand-in for the source audio \mathbf{s}_0 . We evaluate on the `music3` and `speech3` recordings, and use the remaining three music and three speech recordings for training. In order to improve model generalization to unseen source locations, the dataset is also expanded with simulated

Table 1: Model properties and localization performance on the LuViRA [40] **music3** and **speech3** trajectories using all 11 microphones. Input types refers to: 1 - GCC-PHAT, 2 - NGCC-PHAT, 3 - raw audio waveforms.

Setup	Method	Input	var. #mics	perm. inv.	#params	music3		speech3	
						MAE [cm] ↓	acc@30cm ↑	MAE [cm] ↓	acc@30cm ↑
1 _a	Multilat [1]	1	✓	✓	-	38.8 ± 2.5	72.5 ± 1.6	72.8 ± 4.4	55.7 ± 2.1
	Multilat* [1, 5]	2	✓	✓	0.9M	16.3 ± 1.6	94.7 ± 0.8	34.9 ± 3.2	84.9 ± 1.6
	DI-NN [16]	3	✗	✗	3.6M	26.0 ± 0.8	73.0 ± 0.2	44.7 ± 1.7	45.9 ± 2.3
	GNN [15]	1	✓	✗	2.2M	17.0 ± 0.7	90.7 ± 1.0	31.9 ± 1.6	71.2 ± 2.0
	wav2pos	2+3	✓	✓	10.5M	14.2 ± 0.5	95.4 ± 0.7	23.6 ± 1.2	81.6 ± 1.7

recordings, where the sound source is randomly sampled in a room of size $7 \times 8 \times 2$ m, with microphones placed in the same positions as in the real recordings. Simulations are done using Pyroomacoustics [33], where in each time frame, we randomly sample a source position and a reverberation time t_{60} in the range $(0, 0.4)$ and use audio captured from the 12:th microphone as input to the simulation. The total amount of training data is therefore approximately $2 \times (3 + 3) = 12$ minutes of audio recordings.

We initialize all network layers, as well as mask tokens and modality embeddings, from a Gaussian distribution $\mathcal{N}(0, 0.02)$, then train for 500 epochs using the AdamW optimizer [26] with a batch size of 256, a learning rate of 0.0005 and weight decay of 0.1. In all experiments we use $\lambda_{\text{source}} = \lambda_{\text{mic}} = 1.0$ and $\lambda_{\text{audio}} = 0.1$, an embedding dimension $d = 256$, $D = 4$ transformer layers and a signal length of $N = 2048$ at a sample rate of 16 kHz. For TDOA features, we use NGCC-PHAT [5] by pre-training it on the same dataset. For data augmentation we use additive Gaussian noise and random time shifts of the audio, uniformly sampled in $[-0.05, 0.05]$ s. The same time shift is applied to all microphones in order to preserve relative time differences, and the speaker is assumed to be stationary within this time period. Silent periods are excluded (for both training and inference) by thresholding the signal power. The audio reconstruction loss is computed on the non-masked inputs without noise, which enables the model to perform de-noising.

We compare our method to a robust multilateration method [1], where TDOAs are estimated using GCC-PHAT or a pre-trained NGCC-PHAT. We also extend the dual input neural network (DI-NN) [16] and graph neural network (GNN) [15] methods to 3D localization, and train them on the same dataset using the MSE loss, but with the hyperparameters proposed in the corresponding publications. Localization errors are truncated at 3 m, since the traditional multilateration method sometimes yields very large errors or fails to converge.

The results are shown in Table 1, along with the input type used by each method, whether they support a variable number of microphones, if they are invariant to permutations of the microphone order and the number of learnable parameters. Evaluation is done assuming all microphone locations are known, which we denote Setup 1_a. The mean absolute error (MAE) and accuracy are evaluated using a 95 % confidence interval by bootstrapping. The

Table 2: Sound source localization MAE [cm] on the `speech3` trajectory using different number of microphones and setups. Multilat* fails to converge for 5 microphones.

Setup	Method	$M = 5$	$M = 7$	$M = 9$
1_a	Multilat	244.9 ± 4.8	133.1 ± 5.7	94.3 ± 5.0
	Multilat*	N/A	105.6 ± 6.1	56.7 ± 4.5
	DI-NN $_M$	94.9 ± 2.6	76.1 ± 2.0	58.5 ± 1.6
	GNN $_{\mathcal{M}}$	80.5 ± 2.2	53.1 ± 1.9	41.1 ± 1.7
	wav2pos $_{\mathcal{M}}$	66.8 ± 2.0	38.8 ± 1.7	28.4 ± 1.4
1_b	wav2pos $_{\mathcal{M}}$	42.3 ± 1.4	26.1 ± 1.0	20.4 ± 1.0
2_a	wav2pos $_{\mathcal{M}}$	47.2 ± 1.8	32.2 ± 1.4	25.3 ± 1.2
2_b	wav2pos $_{\mathcal{M}}$	33.0 ± 1.2	23.3 ± 1.0	19.6 ± 1.0

results indicate that our method consistently has the lowest MAE for both music and speech recordings. Although multilateration with NGCC-PHAT achieves similar accuracy at the 30 cm threshold, our method has a shorter tail in the error distribution and therefore yields a lower MAE.

Table 3: Microphone localization MAE [cm] over all unknown microphone locations, on the `speech3` trajectory using different numbers of known microphone locations.

Setup	Method	$M = 7$	$M = 8$	$M = 9$
2_a	wav2pos $_{\mathcal{M}}$	182.8 ± 1.7	93.1 ± 1.9	36.8 ± 1.9
2_b	wav2pos $_{\mathcal{M}}$	181.7 ± 1.7	90.4 ± 1.8	34.8 ± 1.6

Evaluation with missing inputs. In order to test our method in different problem setups, we also train it using random masking. During training, we randomly leave between 8 and 11 coordinates and audio snippets, such that at least 5 microphones are in both sets. Since the sound source audio s_0 might be known in some scenarios, we mask this token with 50 % probability, and denote our method trained with random masking as wav2pos $_{\mathcal{M}}$. At inference time, unknown microphone locations are masked and, if the corresponding sound recordings are not masked, the coordinates can be predicted by the decoder.

Since DI-NN does not support a variable number of microphones, we train separate models DI-NN $_M$ for each scenario where M microphones are used. The masked version of GNN is trained by randomly sampling a subset containing between 5 and 11 of microphones, and we denote this method GNN $_{\mathcal{M}}$. However, unlike our proposed method, these models cannot predict microphone locations, or exploit the sound source audio.

The results using different number of known microphone coordinates are shown in Table 2, where it can be seen that our method is consistently more robust when performing localization using a subset of the microphones. In addition, our method can also be evaluated

Table 4: Sound source localization MAE [cm] on the `speech3` trajectory using subsets of microphones and network modules.

Setup	Method	$M = 7$	$M = 11$
1_a	wav2pos baseline	148.8 ± 2.4	144.3 ± 3.1
	+pairwise pos-enc.	129.7 ± 2.9	60.1 ± 2.5
	+random masking	96.7 ± 3.0	81.2 ± 3.0
	+max-pooling	92.8 ± 2.9	73.4 ± 3.8
	+TDOA feat. (GCC-PHAT) (NGCC-PHAT)	57.9 ± 1.8 38.8 ± 1.7	32.9 ± 1.3 22.7 ± 1.1

in the scenario where the sound s_0 emitted from the source is given (and hence audio from the 12:th microphone is not masked), but its location unknown (Setup 1_b). Evaluating the same model in this setup shows that it can exploit the additional data to improve localization performance. Furthermore, our method can exploit audio from microphones in unknown locations, denoted as Setup 2_a (unknown source audio) and 2_b (known source audio), where audio from all microphones are used as input, but their coordinates are masked (except for the M known). This improves the localization performance and also allows for localization of the microphones themselves. Table 3 shows that this is possible for a small number of unknown microphone locations, but the errors become very large when less than 8 microphone locations are known.

Ablation study. Next, we ablate the different components of our method in Table 4. Notably, the pairwise positional encoding is crucial for the method to work, since it allows the transformers to connect audio and coordinates from the same microphone. The ablation also shows that our random masking strategy significantly improves robustness to missing microphones. Providing TDOA features as an additional input drastically reduces the localization error, and we note that using a pre-trained NGCC-PHAT yields significantly better results compared to regular GCC-PHAT inputs.

Simulated environment. So far, we have only considered recordings with microphones in a limited number of possible locations. In order to validate our approach in scenarios with flexible microphone locations and changes in signal to noise ratio (SNR) and reverberation times, we perform additional experiments in a controlled simulated environment¹. We use the same setup 1_a as for LuViRA, but microphone locations are randomly sampled across each of the walls, floor and ceiling, for a total of six microphones. We use recordings from the LibriSpeech dataset [29] and create a 20 000/2 000 train/test split based on speaker-ids. Results are shown in Figures 3a-d, where performance is evaluated over a range of SNRs and reverberation times. Notably, our proposed method outperforms DI-NN and GNN in all scenarios. GNN, which relies on GCC-PHAT as input, performs poorly as reverberation increases, which highlights the necessity of using better feature extractors for good performance in such conditions. In addition, Figure 3e shows the signal de-noising

¹Code: <https://github.com/axeber01/wav2pos/>

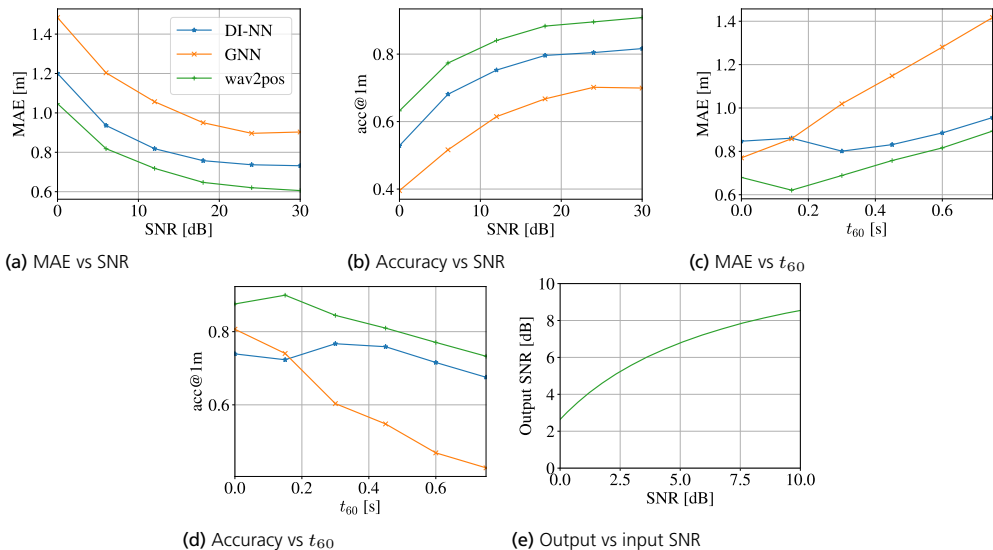


Figure 3: Results on the simulated dataset under varying noise and reverberation conditions.

performance of our method, yielding positive gains roughly in the range of 0 to 7 dB. However, de-noising performance is limited by the embedding dimension of the encoder and decoder, making it difficult to reconstruct high-frequency content.

4 Conclusions and Future Work

In this work, we have proposed a general SSL method that can be used in a wide range of problem scenarios. We conjecture that our method can also be further extended to localizing multiple sound sources. This is possible due to the flexibility of masked autoencoders, where additional inputs or outputs can be added seamlessly. It is also possible to consider the full self-calibration problem where no microphone locations are known, but this requires processing longer sequences of moving sound sources. We leave this as future work and hope that it can inspire the wider research community to create more challenging localization datasets and tasks.

References

- [1] K. Åström, M. Larsson, G. Flood, and M. Oskarsson. Extension of Time-Difference-of-Arrival Self Calibration Solutions using Robust Multilateration. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 870–874. IEEE, 2021.

- [2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [3] A. Beck, P. Stoica, and J. Li. Exact and Approximate Solutions of Source Localization Problems. *IEEE Trans. on Signal Processing*, 56(5):1770–1778, May 2008. ISSN 1941-0476. doi: 10.1109/TSP.2007.909342.
- [4] A. Beck, M. Teboulle, and Z. Chikishev. Iterative Minimization Schemes for Solving the Single Source Localization Problem. *SIAM Journal on Optimization*, 19(3):1397–1416, Jan. 2008. ISSN 1052-6234, 1095-7189. doi: 10.1137/070698014.
- [5] A. Berg, M. O’Connor, K. Åström, and M. Oskarsson. Extending GCC-PHAT using Shift Equivariant Neural Networks. In *Proc. Interspeech 2022*, pages 1791–1795, 2022. doi: 10.21437/Interspeech.2022-524.
- [6] K. W. Cheung, H. C. So, W. Ma, and Y. T. Chan. Least squares algorithms for time-of-arrival-based mobile location. *IEEE Trans. on Signal Processing*, 52(4):1121–1130, Apr. 2004. ISSN 1941-0476. doi: 10.1109/TSP.2004.823465.
- [7] J.-H. Cho and J.-H. Chang. SR-SRP: Super-Resolution based SRP-PHAT for Sound Source Localization and Tracking. In *Proc. INTERSPEECH 2023*, pages 3769–3773, 2023. doi: 10.21437/Interspeech.2023-2369.
- [8] I. D. Coope. Reliable computation of the points of intersection of n spheres in R^n . *ANZIAM Journal*, 42:C461–C477, Dec. 2000. ISSN 1445-8810. doi: 10.21914/anziamj.v42i0.608.
- [9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [10] D. Diaz-Guerra, A. Miguel, and J. R. Beltran. Robust Sound Source Tracking using SRP-PHAT and 3D Convolutional Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:300–311, 2020.
- [11] L. Feng, Y. Gong, and X.-L. Zhang. Soft Label Coding for end-to-end Sound Source Localization with ad-hoc Microphone Arrays. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [12] X. Geng, H. Liu, L. Lee, D. Schuurmans, S. Levine, and P. Abbeel. Multimodal Masked Autoencoders Learn Transferable Representations. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022.

-
- [13] M. D. Gillette and H. F. Silverman. A Linear Closed-Form Algorithm for Source Localization from Time-Differences of Arrival. *IEEE Signal Processing Letters*, 15:1–4, 2008.
- [14] Y. Gong, S. Liu, and X.-L. Zhang. End-to-end Two-Dimensional Sound Source Localization with Ad-Hoc Microphone Arrays. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1944–1949. IEEE, 2022.
- [15] E. Grinstein, M. Brookes, and P. A. Naylor. Graph Neural Networks for Sound Source Localization on Distributed Microphone Networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [16] E. Grinstein, V. W. Neo, and P. A. Naylor. Dual Input Neural Networks for Positional Sound Source Localization. *EURASIP Journal on Audio, Speech, and Music Processing*, 2023(1):32, 2023.
- [17] P.-A. Grumiaux, S. Kitić, L. Girin, and A. Guérin. A Survey of Sound Source Localization with Deep Learning mMethods. *The Journal of the Acoustical Society of America*, 152(1):107–151, 2022.
- [18] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are Scalable Vision Learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
- [19] Y. HE and A. Markham. SoundDoA: Learn Sound Source Direction of Arrival and Semantics from Sound Raw Waveforms. In *Proc. Interspeech 2022*, pages 2408–2412, 2022. doi: 10.21437/Interspeech.2022-378.
- [20] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.
- [21] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer. Masked Autoencoders that Listen. *Advances in Neural Information Processing Systems*, 35:28708–28720, 2022.
- [22] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [23] R. Jyothi and P. Babu. SOLVIT: A Reference-Free Source Localization Technique Using Majorization Minimization. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, 28:2661–2673, 2020. ISSN 2329-9304. doi: 10.1109/TASLP.2020.3021500.

- [24] C. Knapp and G. Carter. The Generalized Correlation Method for Estimation of Time Delay. *IEEE transactions on acoustics, speech, and signal processing*, 24(4):320–327, 1976.
- [25] M. Larsson, V. Larsson, K. Åström, and M. Oskarsson. Optimal Trilateration Is an Eigenvalue Problem. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5586–5590, May 2019. doi: 10.1109/ICASSP.2019.8683355.
- [26] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [27] D. R. Luke, S. Sabach, M. Teboulle, and K. Zatlaway. A simple globally convergent algorithm for the nonsmooth nonconvex single source localization problem. *Journal of Global Optimization*, 69(4):889–909, Dec. 2017. ISSN 1573-2916. doi: 10.1007/s10898-017-0545-6.
- [28] D. E. Manolakis. Efficient solution and performance analysis of 3-D position estimation by trilateration. *IEEE Trans. on Aerospace and Electronic Systems*, 32(4):1239–1248, Oct. 1996. ISSN 1557-9603. doi: 10.1109/7.543845.
- [29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An ASR Corpus Based on Public Domain Audio Books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.
- [30] W. Phokhinnan, N. Obin, and S. Argentieri. Binaural Sound Localization in Noisy Environments Using Frequency-Based Audio Vision Transformer (FAViT). In *Proc. INTERSPEECH 2023*, pages 3704–3708, 2023. doi: 10.21437/Interspeech.2023-2015.
- [31] A. Raina and V. Arora. SyncNet: Correlating Objective for Time Delay Estimation in Audio Signals. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [32] D. Salvati, C. Drioli, and G. L. Foresti. Time Delay Estimation for Speaker Localization Using CNN-Based Parametrized GCC-PHAT Features. In *Interspeech*, pages 1479–1483, 2021.
- [33] R. Scheibler, E. Bezzam, and I. Dokmanić. Pyroomacoustics: A Python Package for Audio Room Simulation and Array Processing Algorithms. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 351–355. IEEE, 2018.

-
- [34] N. Sirola. Closed-form algorithms in mobile positioning: Myths and misconceptions. In *Navigation and Communication 2010 7th Workshop on Positioning*, pages 38–44, Mar. 2010. doi: 10.1109/WPNC.2010.5653789.
- [35] P. Stoica and J. Li. Lecture notes-source localization from range-difference measurements. *IEEE Signal Processing Magazine*, 23(6):63–66, 2006.
- [36] F. Thomas and L. Ros. Revisiting trilateration for robot localization. *IEEE Trans. on Robotics*, 21(1):93–101, Feb. 2005. ISSN 1941-0468. doi: 10.1109/TRO.2004.833793.
- [37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. *Advances in neural information processing systems*, 30, 2017.
- [38] J. M. Vera-Diaz, D. Pizarro, and J. Macias-Guarasa. Towards End-to-End Acoustic Localization using Deep Learning: From Audio Signals to Source Position Coordinates. *Sensors*, 18(10):3418, 2018.
- [39] Y. Wang, B. Yang, and X. Li. FN-SSL: Full-Band and Narrow-Band Fusion for Sound Source Localization. In *Proc. INTERSPEECH 2023*, pages 3779–3783, 2023. doi: 10.21437/Interspeech.2023-714.
- [40] I. Yaman, G. Tian, M. Larsson, P. Persson, M. Sandra, A. Dürr, E. Tegler, N. Challa, H. Garde, F. Tufvesson, et al. The LuViRA Dataset: Measurement Description. *arXiv preprint arXiv:2302.05309*, 2023.
- [41] Y. Zhou. A closed-form algorithm for the least-squares trilateration problem. *Robotica*, 29(3):375–389, May 2011. ISSN 1469-8668, 0263-5747. doi: 10.1017/S0263574710000196.

Appendix

Details on model training. The implementations of DI-NN² and GNN³ available online are extended from two to three dimensions. We consider using GNN with spatial likelihood functions infeasible in large 3D environments, thus we use GNN with GCC-PHAT inputs only. We also tried pre-trained NGCC-PHAT features as input to GNN, but the training then failed to converge.

For M microphones GNN uses $M(M - 1)/2$ TDOA features by only computing \mathbf{R}_{ij} and not \mathbf{R}_{ji} , which speeds up computation time, but breaks permutation invariance. In contrast, wav2pos uses all $M(M - 1)$ features in order to preserve this property. This also results in slightly better localization performance. Computation time is saved by noting that $\mathbf{R}_{ij}[t] = \mathbf{R}_{ji}[-t]$ for any time delay t , a property that holds for both GCC-PHAT and NGCC-PHAT.

NGCC-PHAT is trained using an available implementation⁴ with slight modifications. At pre-training time, two random microphones are picked for each training example and the TDOA is estimated using classification in the range of integers $-\tau, \dots, \tau$ using the cross-entropy loss. The maximum possible time delay between two microphones can be calculated by considering the distance between the two most separated microphones. For the LuViRA dataset, this results in a maximum delay of $\tau = 314$ samples. At inference time, TDOA estimates are computed for all pairs of microphones.

The multilateration method runs in Matlab using open source code⁵. We modify the code to not consider the full self-calibration problem, but only the sound source localization with known microphone positions. This method uses a RANSAC loop that tests each of the four strongest peaks in the GCC-PHAT feature per microphone pair.. When adopting the method to use NGCC-PHAT, we only consider a single peak, since NGCC-PHAT is trained to estimate a single TDOA.

Model training is done in Pytorch on a single NVIDIA A100 GPU. Training and inference times are shown in Table 5. We do not report time for multilateration, as there is no training step and the inference code was not optimized for speed. The wav2pos network without TDOA features corresponds to the Table 1 max-pooling step. We note that computing TDOA features using NGCC-PHAT increases the execution time of our method considerably, since these are computed sequentially for all 55 pairwise microphone combinations. This could be parallelized in order to speed up execution time.

²https://github.com/egrinstein/di_nn/

³https://github.com/egrinstein/gnn_ssl/

⁴<https://github.com/axeber01/ngcc/>

⁵<https://github.com/kalleastrom/StructureFromSound2/>

Table 5: Training and inference times for setup 1_a with $M = 11$ microphones, measured on a single A100 GPU. Training was done for 500 epochs on the LuViRA dataset.

Method	Training [h]	Inference [ms]
DI-NN	29.5	0.2
GNN	34.7	1.0
wav2pos w/o TDOA feat.	27.8	0.2
wav2pos	64.3	5.9

Dataset details. Figure 4 shows the 3D layout of the microphones in the LuViRA dataset, which are identical for all eight recordings. In addition, the microphone locations in simulated dataset are visualized in Figure 5. Table 6 provides the train/test split used for the simulated dataset.

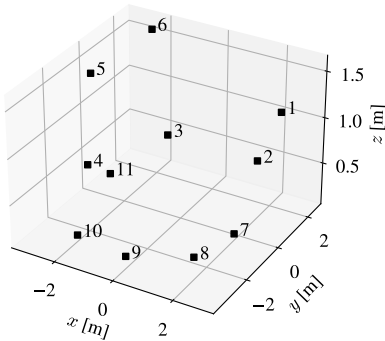


Figure 4: Microphone locations in the LuViRA dataset.

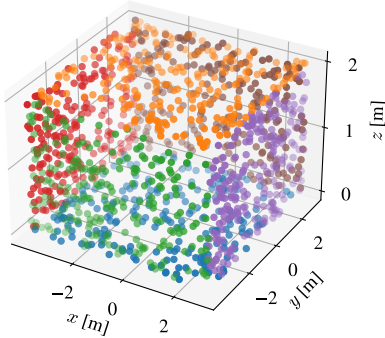


Figure 5: Microphone distribution in the simulated dataset. Each dot shows a microphone for a single training example. The color shows which of the six faces of the room it belongs to.

Table 6: Speaker-ids used in the simulated dataset. Recordings shorter than one second are discarded.

Dataset	Test speaker-ids	Train speaker-ids
Librispeech test-clean	61, 121, 237	all other 43 speakers

Additional results on LuViRA dataset. We provide additional results for all methods on several splits of the LuViRA dataset in Table 7. The train/test splits are constructed such that six tracks are used for training, e.g. `music1-3` and `speech1-3`, and two are used for testing, e.g. `music4` and `speech4`. Out of the eight tracks, `music2` and `music4` are the only ones with significant height variation in the source trajectory. The corresponding cumulative error distributions are shown in Figure 6 and additional visualizations of model predictions are shown in Figure 7. It can be seen that although the multilateration methods are often very accurate, they have a long tail in the error distributions due to outliers, whereas the learning-based methods do not suffer from this problem.

Table 7: Mean absolute error [cm] on the LuViRA dataset using setup 1_a and different test splits.

Method	<code>music1</code>	<code>music2</code>	<code>music3</code>	<code>music4</code>	<code>speech1</code>	<code>speech2</code>	<code>speech3</code>	<code>speech4</code>
Multilat	67.2 ± 3.2	48.6 ± 2.3	38.8 ± 2.5	28.1 ± 1.6	80.9 ± 3.2	90.4 ± 3.5	72.8 ± 4.4	124 ± 3.8
Multilat*	32.9 ± 2.3	20.7 ± 1.5	16.3 ± 1.6	9.5 ± 0.8	20.7 ± 1.7	18.9 ± 1.6	34.9 ± 3.2	41.9 ± 2.5
DI-NN	54.0 ± 1.9	64.9 ± 1.5	26.0 ± 0.8	47.1 ± 1.2	29.6 ± 0.9	22.5 ± 0.6	44.7 ± 1.7	43.2 ± 1.5
GNN	42.3 ± 1.8	74.9 ± 2.1	17.0 ± 0.7	33.6 ± 0.8	21.6 ± 0.9	19.2 ± 0.6	31.9 ± 1.6	35.1 ± 1.3
wav2pos	22.7 ± 1.0	28.1 ± 0.8	14.2 ± 0.5	19.9 ± 0.4	13.1 ± 0.6	11.7 ± 0.4	23.6 ± 1.2	24.8 ± 0.9

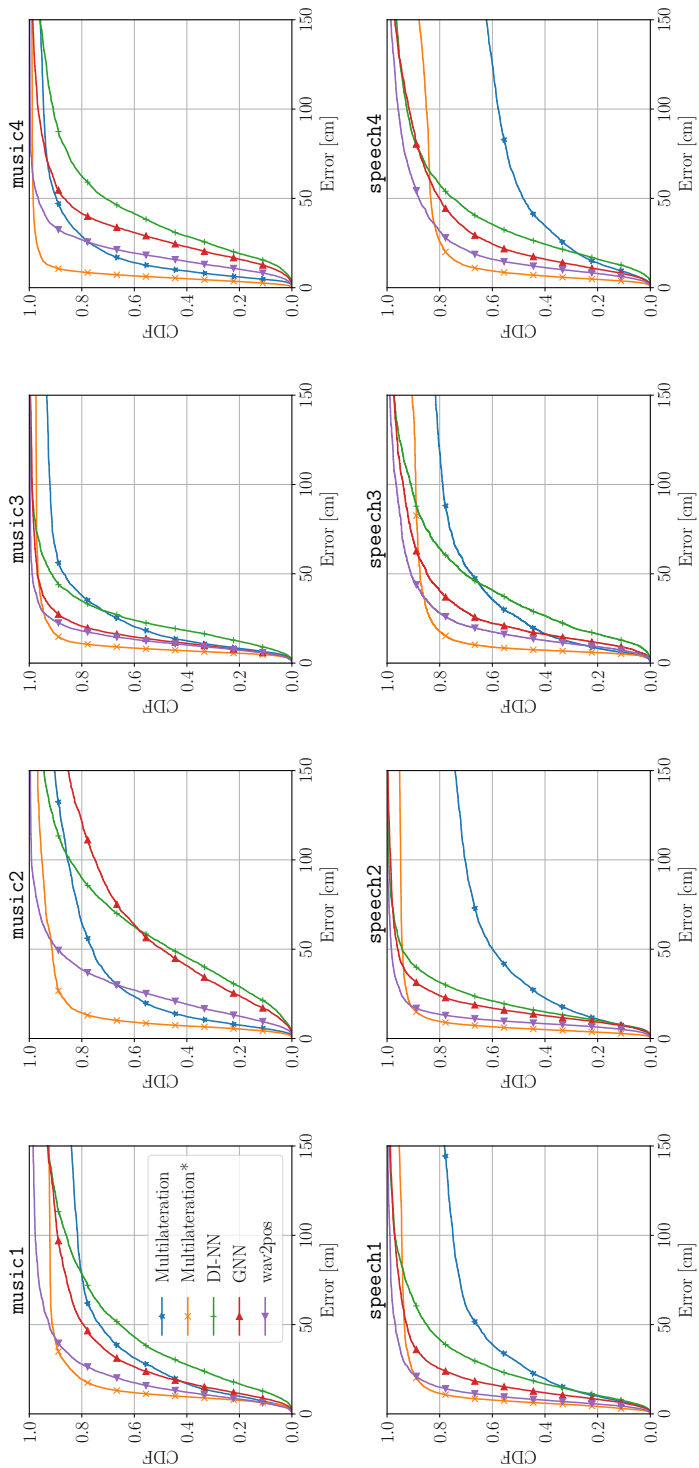


Figure 6: Cumulative error distributions on the LuVIRA dataset using setup 1_a and different test splits.

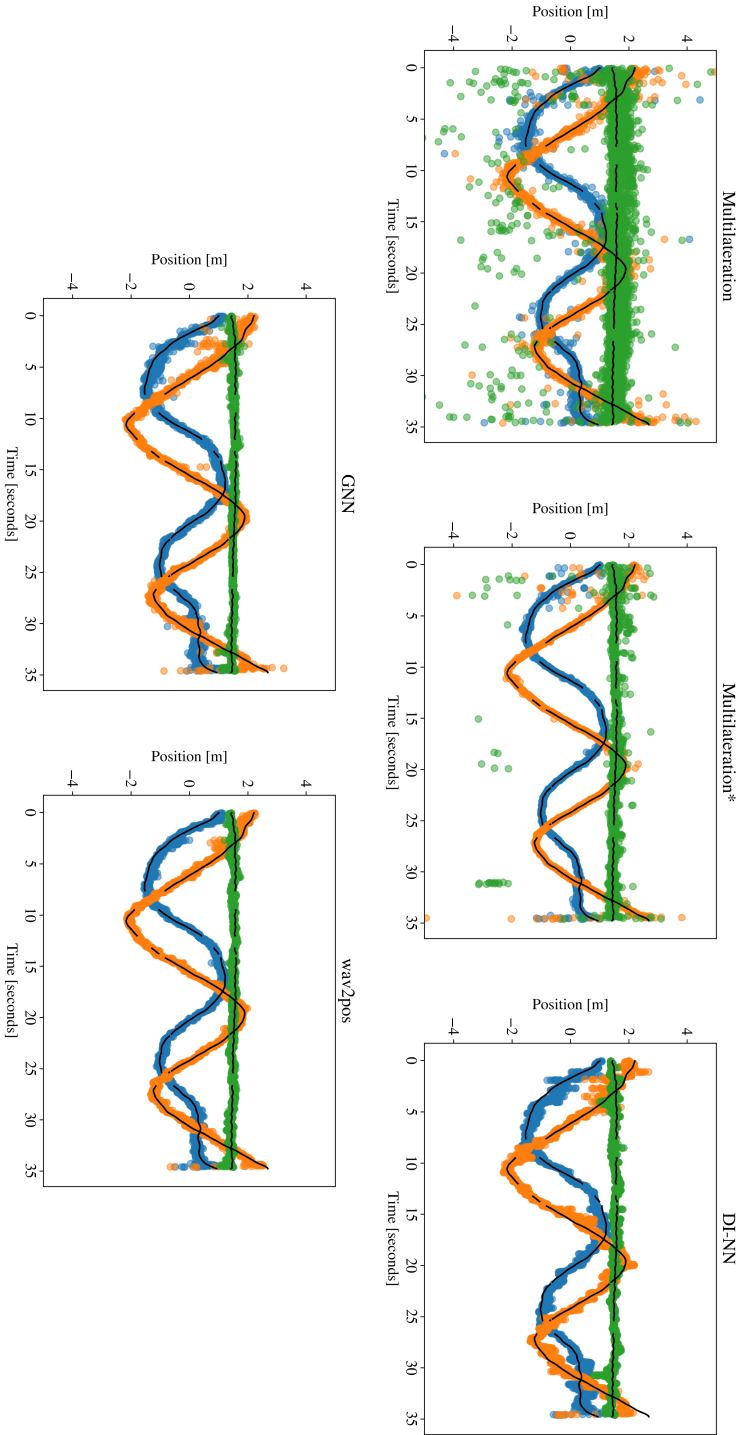


Figure 7: Visualizations of the predictions on the `mustc3` track, where each coordinate prediction is shown separately (x : blue, y : orange, z : green). Ground truth coordinates are traced in black (some timestamps are missing ground truth).