Diffusion Modelling approaches to EEG-based Auditory Attention Decoding

David Rannaleet

Victor Gunnarsson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6227
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2024

Acknowledgements

We would like to express our sincerest gratitude to our thesis supervisors, Emina Alickovic, Martin Skoglund and Bo Bernhardsson, for their invaluable guidance and insights throughout this project. Their expertise in many topics included in our thesis helped us immensely with the direction of our work. In addition to this, they also provided valuable sanity checking of our results, and have given us valuable feedback to the formulation of the report itself. Without their continued support and input at times when we were stuck, this thesis would not have been possible.

We would also like to thank the Department of Automatic Control at LTH for providing us with office spaces and equipment that we used during the course of the project.

Some of this project's computations were made possible by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Chalmers University of Technology partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Abstract

Machine learning models can analyze physiological data, such as electroencephalography (EEG), for various classification tasks. One such task is Auditory Attention Decoding (AAD), aimed at identifying the sound a person is actively attending to, offering significant benefits for users of hearing aids. However, EEG data often exhibits a low signal-to-noise ratio, and its collection is often expensive, cumbersome and requires trained specialists. Additionally, gathering EEG data over prolonged periods of time presents challenges. The resulting scarcity of available EEG data to train models can be alleviated by using generative models, which generate new examples from the data they were trained on.

Diffusion Probabilistic Models (DPMs) have in recent years emerged as the state-of-the-art of generative models within the image domain, showing widespread success in models such as Stable Diffusion and DALL-E. This work investigates whether this success can extend to the domain of multichannel time series data, specifically EEG data. The diffusion models were trained on 1-second EEG data segments and were used as a data augmentation tool for 3 different classification tasks, including AAD. Our findings indicate that diffusion models can effectively generate realistic EEG data, supported by both a visual comparison and a measure of Jensen-Shannon divergence to the real EEG data distribution. In addition to this, a significant improvement in mean performance was achieved in our Locus of Attention (LoA) task, where we classify between a test subject attending to a left or right speaker. Here an approximate classification accuracy of **71%** was achieved compared to our baseline of **70.4%**.

Sammanfattning

Maskininlärningsmodeller kan analysera fysiologisk data, t.ex. elektroencefalografi (EEG), för rad olika klassificeringsproblem. Ett sådan problem är Auditory Attention Decoding (AAD), som syftar till att identifiera det ljud som en person aktivt ägnar sig åt, vilket kan ge betydande fördelar för användare av hörapparater. EEG data har dock ofta ett lågt signal-brusförhållande, och insamlingen är ofta dyr, besvärlig och kräver utbildade specialister. Dessutom är det svårt att samla in EEG data under längre tidsperioder. Den resulterande bristen på tillgänglig EEG data för att träna modeller kan lindras genom att använda generativa modeller, som genererar nya exempel från den data som de tränades på.

Diffusion Probabilistic Models (DPM) har under de senaste åren framkommit som den mest framgångsrika generativa modellen inom bilddomänen och har visat stor framgång i modeller som Stable Diffusion och DALL-E. I detta arbete undersöks om denna framgång kan utvidgas till domänen för flerkanalig tidsseriedata, specifikt EEG data. Vi tränade flera diffusionsmodeller på 1-sekunders segment av EEG data och använde dem som ett dataförstärkningsverktyg för 3 olika klassificeringsproblem, inklusive AAD. Våra resultat visar att diffusionsmodeller effektivt kan generera realistisk EEG data, vilket stöds av både en visuell jämförelse och ett mått på Jensen-Shannon divergens till den verkliga EEG-datadistributionen. Utöver detta uppnådde vi en betydande förbättring av den genomsnittliga prestandan för vår Locus of Attention (LoA) uppgift, där vi klassifiercerar mellan att ett testsubjekt aktivt ägnar sig åt en höger eller vänster högtalare. Där uppnådde vi ett medelvärde i klassificeringsnoggrannhet på **71%** jämfört med vår baslinje på **70.4%**.

Abbreviations

AAD	Auditory Attention Decoding
AUC	Area Under the Curve
CNN	Convolutional Neural Network
DDIM	Denoising Diffusion Implicit Model
DDPM	Denoising Diffusion Probabilistic Model
DPM	Diffusion Probabilistic Model
ECoG	Electrocorticography
EEG	Electroencephalography
ELU	Exponential Linear Unit
GAN	Generative Adversarial Network
KL	Kullback–Leibler
LoA	Locus of Attention
MEG	Magnetoencephalography
MSE	Mean Squared Error
NN	Neural Network
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SD	Standard Deviation
SiLU	Sigmoid-Linear Unit
SPL	Sound Pressure Level
STFT	Short-Time Fourier Transform
VAE	Variational Autoencoder
VQ-VAE	Vector Quantised-Variational AutoEncoder

Contents

1

Introduction

In recent decades, hearing aids have increasingly become a crucial assistant for people with hearing loss, aiding in their ability to carry out everyday tasks that require auditory attention. Many of these tasks occur in environments where multiple audio sources compete for attention, as often is the case at various social events. At such events, the occurrence of background conversations that interfere with a primary conversation of interest are common. Traditional hearing loss compensation strategies, such as amplification of all incoming sounds, fall short in these scenarios as the amplification of unwanted background noise often leads to such discomfort that users will forgo the use of a hearing aid entirely. This highlights a need for more complex strategies that can differentiate and enhance relevant sounds while suppressing background noise.

1.1 Background & Context

The problem of selectively attending to a speaker of interest amidst other competing speakers has been recognized for decades and was named the cocktail party problem by Colin Cherry in 1953 [Cherry, 1953]. Recognizing the complexity of the cocktail party problem, hearing aid manufacturers have become increasingly focused on developing signal-processing technology designed to alleviate this issue.

Modern hearing aids have made significant advances in reducing discomfort produced by background noise through the use of sophisticated signal processing algorithms. These algorithms attempt to discern and amplify relevant sounds while suppressing background noise. However, their effectiveness is inherently limited by the inability of the hearing aid to decode a user's audio attention. While individuals with normal hearing can naturally filter out sound sources the listener is not attending to, this is a significant challenge for those with hearing impairments, which is well-documented in the literature [Marrone et al., 2008].

Recent studies have shown the possibility of using electroencephalography (EEG) to non-intrusively extract attention-related information [O'sullivan et al., 2015; Mirkovic et al., 2016; Alickovic et al., 2019; Geirnaert et al., 2021; Crosse et al., 2021; Alickovic et al., 2020; Alickovic et al., 2021; Puffay et al., 2023]. In contrast to other methods such as Electrocorticography (ECoG), EEG is a non-invasive

procedure that indirectly captures information from the firing of neurons within the brain, offering a temporal resolution in the millisecond range [Gevins et al., 1999], making it ideal for real-time applications.

Auditory attention decoding (AAD) can utilize EEG data to decode the listener's attention, i.e., identify and track attended sound in environments with competing sound sources. Recent research has focused on developing frameworks for non-linear approaches to AAD, specifically emphasizing stimulus (i.e., speech) reconstruction, as seen in [Taillez et al., 2020], or directly decoding the direction of attention of the listener, termed Locus of Attention (LoA) [Vandecappelle et al., 2021; Wilroth et al., 2023].

According to Geirnaerts et al.'s review, while linear models excel with larger time windows, non-linear models tend to outperform linear models when using shorter time windows. They speculate that non-linear models achieving limited generalization across multiple datasets could be attributed to the limited size of training dataset available for AAD, suggesting a need for larger and more diverse datasets. [Geirnaert et al., 2021]

One note of importance is that the data used for studying AAD is often recorded with participants of normal hearing and in controlled listening environments. Two publicly published datasets use normal hearing participants attending one of two sound streams coming from equipped in-ear headphones, not fully simulating a cocktail party environment [Das et al., 2020].

The dataset used in this work attempts to more closely simulate a cocktail party environment from the perspective of a listener with hearing impairment. The dataset has participants being exposed to background babble noise, while also trying to attend to one of two primary audio sources. Differences and the availability of datasets for AAD make comparing published models difficult, and a standard dataset for testing AAD models has not yet been introduced [Geirnaert et al., 2021]. As such, comparing results from different papers needs to be done with caution.

1.2 Purpose & Project Objectives

This thesis aims to investigate the viability of utilizing Diffusion Probabilistic Models (DPMs) for the synthetic generation of EEG data and to evaluate the potential value of such synthesized data for augmenting existing EEG datasets. The motivation for this study originates from prior successful work in EEG data synthesis using Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) [Sun and Mou, 2023]. Given the proven competitiveness of DPMs in image synthesis when compared to GANs and VAEs [Bond-Taylor et al., 2022; Dhariwal and Nichol, 2021], there is interest in their potential applicability in the EEG domain. The specific objectives arising from this primary aim include:

1. Investigate the viability of generating EEG data using DPMs traditionally trained on image data.

As a first step, we aim to assess whether DPMs, traditionally designed and utilized for image data, are suitable for EEG data. In this case, we represent a multichannel EEG-signal as a grayscale 2D image.

2. Evaluate the quality of the synthetically generated EEG data through both visual inspection and objective metrics.

If generating EEG data proves viable, the subsequent goal is to assess the quality of the synthetic data and its sensitivity to the loss function used during training. This assessment will involve visual inspection as well as objective metrics to evaluate the quality of the generated EEG data.

3. Quantify the benefits of using generated EEG data for data augmentation in three different classification tasks.

Lastly, to align with the initial purpose, this objective involves exploring the performance of synthetically generated EEG data in data augmentation. This evaluation includes three different classification tasks characterized by different class ratios and variations in the amount of added synthetic data. These tasks are as follows:

a. A Locus-of-Attention (LoA) task with balanced classes.

A classification model classifies EEG data as either belonging to a “Left” or “Right” label. Here, an equal amount of segments are allocated to each label, resulting in a balanced dataset.

b. A Passive vs. Active listening task with imbalanced classes.

c. A Passive vs. Active listening task with artificially balanced classes.

The same underlying EEG data segments, which correspond to “Passive” or “Active” listening during an experiment, are used for these tasks. The natural class imbalance is reflected in the first sub-task. The final task addresses the class imbalance through oversampling, aiming to evaluate the effect of balancing on model performance.

1.3 Delimitations

There are some delimitations for the project, which are to be viewed in connection to the stated objectives.

- We limit ourselves to using existing diffusion frameworks and models designed and primarily utilized for image data.

As the main goal is to investigate the feasibility of using DPMs to generate EEG data rather than finding the optimal approach, we restrict ourselves to using existing diffusion models. Instead, the focus lies on modifying only the loss function to improve performance.

- We only consider one classification model with fixed parameters for evaluating the performance of using diffusion generated EEG data for data augmentation.

The use of only one classification model with fixed parameters was necessitated by the limited processing time available. Conducting hundreds of tests for different configurations of the classifier model when adding generated EEG data would have been impractical within the time constraints of the thesis.

- We only consider one EEG dataset for all tasks.

We constrained ourselves to a single EEG dataset for all tasks. As our primary aim was to investigate the viability of generating realistic EEG data using DPMs, the inclusion of multiple datasets becomes more relevant once initial viability is confirmed.

2

Theory

This chapter goes over the foundational theories underpinning our project, providing a comprehensive overview of the most important concepts. This includes the structure of EEG data, AAD, and DPMs. Later in Chapter 3, we will introduce specific changes or adaptations made to some of these for the project.

2.1 Electroencephalogram

First recorded from the human scalp by Hans Berger in 1924, the EEG represents one relatively non-invasive way to record brain activity. This was made possible by earlier discoveries of measurable electrical activity within the brain in the late 1800s by Richard Caton [Silva, 2010]. Modern EEG recordings are usually done with an array of electrodes placed at specific locations on the scalp. The locations differ depending on the number of measurement electrodes and the system used, but an often used one is known as the International 10-20 system [Klem et al., 1999], which is illustrated in Figure 2.1. While EEG is one of several methods used for measuring brain activity, the non-invasive nature makes it possible to record in a wide range of everyday settings.

2.2 Auditory Attention Decoding

AAD refers to the task of decoding (i.e., identifying) a speaker that a listener is attending to in a multi-talker environment. Research has extensively shown that the information required to decode this attention can be found by measuring cortical activity [Mesgarani and Chang, 2012; Ding and Simon, 2012; O'Sullivan et al., 2015]. The cortical activity can be measured in different forms such as by ECoG [Mesgarani and Chang, 2012; O'Sullivan et al., 2019], Magnetoencephalography (MEG) [Puvvada and Simon, 2017], or EEG [O'Sullivan et al., 2015; Alickovic et al., 2019].

The measurements taken can then be used in various algorithms that perform the actual decoding of attention. Specifically, in the context of LoA, the task involves determining the direction towards which the attended speaker is located. This could be either "Left" or "Right" for example.

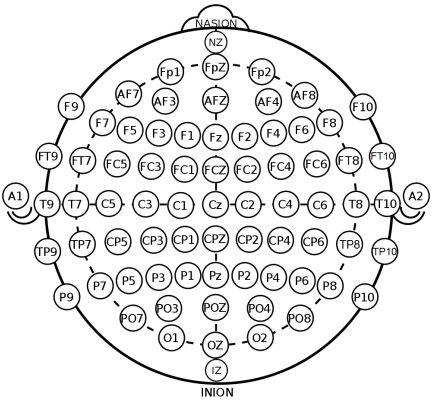


Figure 2.1 A representation of the International 10–20 system electrode locations used to record EEG data on a top-down view of the human scalp. The top of the image corresponds to the front of the head.

Research indicates that non-linear models, such as neural networks, are particularly effective for decoding attention in short time windows [Taillez et al., 2020; Ciccarelli et al., 2019], often surpassing classical linear methods in performance.

2.3 Denoising Diffusion Probabilistic Models (DDPMs)

Diffusion models are a class of generative models that have recently seen widespread success in image generation tasks. Text-to-image models, which generate images from a given prompt, such as DALL-E 2 [Ramesh et al., 2022], often use the Denoising Diffusion Probabilistic (DDPM) version of diffusion models.

In diffusion models, noise is iteratively added to data in small steps, known as forward diffusion, to disrupt the underlying distribution. This is illustrated in figure 2.2. The task of the model is to learn and predict the noise added at each iteration, referred to as time steps. Essentially, the model aims to learn to restore the data distribution disrupted by the forward process through a reverse diffusion process. This modeling approach, introduced in [Sohl-Dickstein et al., 2015], addresses the tradeoff problem encountered by machine learning models between tractability and flexibility. They use a Markov chain to define the model, where each step in the Markov chain is a small transformation of the data that is easy to calculate and evaluate. The full chain of the model converts the unknown distribution of a dataset to a known distribution, making the model able to adapt to different types of data inputs. Recent research has demonstrated that this method of generating data can generate images of higher quality compared to GANs [Dhariwal and Nichol, 2021].

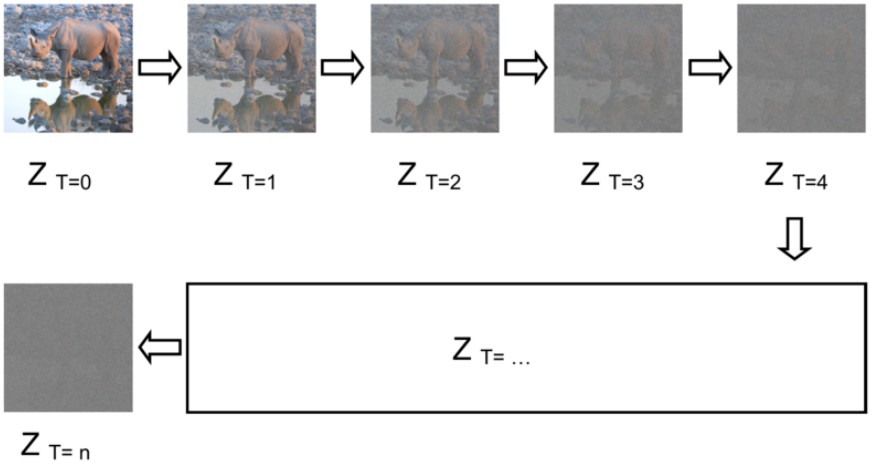


Figure 2.2 Example of the forward diffusion process, where an image gets progressively more disrupted by the added noise, by [MrAlanKoh, 2022]

Diffusion Model

As described in the original paper on diffusion models [Sohl-Dickstein et al., 2015], the diffusion process is modeled as a Markov chain. The forward diffusion process gradually adds noise to the data by stepping forward in the Markov chain by multiplication with the Markov transition kernel $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. The full forward process is defined as:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.1}$$

where $t = 0$ represents the original data distribution $q(\mathbf{x}_0)$, and $t = T$, the final instance of the chain represents the converted data distribution $q(\mathbf{x}_t)$. The distribution the data is converted to can be any distribution as long as it is tractable, however, many implementations of the diffusion model use the Gaussian distribution [Ho et al., 2020; Nichol and Dhariwal, 2021; Dhariwal and Nichol, 2021]. Thus, the Markov transition kernel, or in this case, diffusion kernel as described in [Sohl-Dickstein et al., 2015] is expressed as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I). \tag{2.2}$$

Here, β_t represents the variance of the noise introduced at time t. While β_t can be learned by parameterization, by setting them to constants shown in [Ho et al., 2020], determined by some schedule, the forward process will not contain any learnable parameters. Another notable property of the diffusion model, as highlighted in [Ho et al., 2020], is the ability to perform the forward step for any arbitrary time step t in a single step, as it is otherwise necessary to repeatedly apply Eq. (2.2) up to time

step t in the Markov chain. This is done by introducing α and its cumulative product as $\bar{\alpha}$, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha_t} = \prod_{s=1}^{t} \alpha_s$, making it possible to represent the forward diffusion kernel Eq. (2.2) in closed form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)I). \tag{2.3}$$

This expression can be rewritten with the parameterization trick seen in [Kingma and Welling, 2022] to be a linear combination of the mean, standard deviation, and noise sampled from a Gaussian distribution $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$, expressed as:

$$\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon. \tag{2.4}$$

If the reverse translations for the Markov chain are known, it will be possible to sample from the Gaussian distribution and, through this reverse process, generate data with the same distribution as $q(\mathbf{x}_0)$. Starting at the tractable distribution $p(\mathbf{x}_T)$, the full reverse process is defined by [Sohl-Dickstein et al., 2015] as:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.5}$$

where $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is the reverse Markov transition kernel. Given a small size of the variance β_t, the reverse Markov transition kernel will be of the same functional form as the forward process:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)). \tag{2.6}$$

Here, the mean $\mu_\theta(\mathbf{x}_t, t)$ and covariance $\Sigma_\theta(\mathbf{x}_t, t)$ are unknown parameters, which are estimated by a neural network. However, in [Ho et al., 2020], instead of approximating $\Sigma_\theta(\mathbf{x}_t, t)$ with a neural network, it is set to be time-dependent constants $\sigma_t^2 I$ defined by the values of β_t. The probability of obtaining the original distribution $q(\mathbf{x}_0)$, also known as the model probability, from the Gaussian distribution $p(\mathbf{x}_T)$ is modeled by [Sohl-Dickstein et al., 2015] as:

$$p_\theta(\mathbf{x}_0) = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right], \tag{2.7}$$

which in turn is used to define the loss function for training the model to estimate the parameters. In [Ho et al., 2020], the loss function is defined as the negative loss likelihood of the model probability. Training using this loss is done by optimizing the evidence lower bound (ELBO):

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}_0)} \left[-\log p_\theta(\mathbf{x}_0) \right] \leq \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] = \\ \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] =: L. \end{aligned} \tag{2.8}$$

By further reduction, this equation can be written using Kullback–Leibler divergence (KL-divergence), which reduces the variance in the loss:

$$L = \mathbb{E}_{q(\mathbf{x}_{0:T})}[D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) + \sum_{t>1}^{T} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)], \quad (2.9)$$

where D_{KL} is the KL-divergence. However, as the forward process variances β_t are set by a scheduler and have no learnable parameters, the term $\mathbb{E}_{q(\mathbf{x}_{0:T})}[D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))$ of the loss function is a constant and can be ignored, as neither $q(\mathbf{x}_T|\mathbf{x}_0)$ nor $p(\mathbf{x}_T)$ contains variables. The term $\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)$, which models the step $t = 1$ to $t = 0$ in the reverse diffusion process, is in [Ho et al., 2020] approximated using a neural network when training. This essentially means that while generating data using the reverse diffusion process, otherwise known as sampling, no noise is added at time step $t = 0$. The last term of the loss function $\sum_{t>1}^{T} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t))$, compares with KL divergence the probability distribution of the forward process kernel in reverse $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, with the reverse process kernel $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which both have known, time-dependent constants as variances. Therefore, the KL-divergence of the probabilities can be simplified to:

$$L_{t-1} = \mathbb{E}_q[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2] + C \quad (2.10)$$

resulting in the Mean Squared Error (MSE) loss between the means, where $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ is the known and calculateable mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $\mu_\theta(\mathbf{x}_t, t)$ is the mean of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which needs to be approximated using a model. The constant C contains the variances and other constant terms and can therefore be ignored. However, further simplifications by [Ho et al., 2020] using Eq. (2.4), the mean $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ can be rewritten in terms of added noise ϵ. Repameterization of $\mu_\theta(\mathbf{x}_t, t)$ using this knowledge leads to a simplified loss function that predicts the added noise ϵ:

$$L = \mathbb{E}_{t,\mathbf{x}_0,\epsilon}[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2]. \quad (2.11)$$

Where ϵ_θ is the predicted noise given the sample $\mathbf{x}_t$. Note that the scaling term is removed, as [Ho et al., 2020] found that this gave better results. Training the diffusion model now amounts to optimizing this simplified loss function. When a model has been trained, the sampling using the reverse process is described by:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z} \quad (2.12)$$

where $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, I)$. This formula is derived using the same method as in Eq. (2.4) to rewrite the Gaussian distribution function of the reverse process as a linear equation.

3

Method

3.1 Dataset

The dataset used in this project to train the neural network classifier and the diffusion models was provided by Eriksholm Research Centre, part of Oticon A/S. This dataset has been previously analyzed using different analysis methods [Alickovic et al., 2021; Andersen et al., 2021]. In particular, [Alickovic et al., 2021] showed the benefits of using noise reduction schemes in hearing aids, improving the neural representation of target and masker speech while attenuating undesired background noise. The full setup of the methods for the data acquisition can be read at [Alickovic et al., 2021]. This section will highlight some of the important parts most relevant to the thesis, e.g., the EEG data and the general idea of the setup.

Participants

The study involved 24 male and 10 female Danish-speaking participants between the ages of 21 and 84, with an average age of 64.2 and a standard deviation of 13.6. All participants were experienced hearing aid users, had no documented background of neurological disorders, dyslexia, or diabetes, and had normal or corrected-to-normal vision. The participants had symmetrical sensorineural hearing loss, with an average of 4-frequency pure-tone audiometry average of 47.5 dB hearing loss [Alickovic et al., 2021].

Experiments Setup

Both the hearing aid fitting for each participant and the signal processing algorithms used are outlined in [Alickovic et al., 2020] for the noise reduction scheme ON/OFF. [Andersen et al., 2021] used similar signal processing algorithms. Recordings of the EEG were done using the BioSemi Active Two recording system (Amsterdam, Netherlands). Each participant was equipped with 64 electrodes using the international 10-20 system layout placement over the scalp, with two additional electrodes placed on the mastoid for reference. Sampling of the electroencephalography data was done at 1,024 Hz. The experiment was performed in a sound-proofed booth, with the participant sitting in the center. The room was equipped with six speakers, arranged in a circular arrangement, with a screen in front of the participant. The

loudspeakers were positioned at ±30°, ±112.5°, ±157.5° from the perception of the front-facing participant, see A in Figure 3.1. The two loudspeakers in front were used to broadcast the speech stimuli, the target, and the masker sound, using news clips with neutral content to avoid emotional responses. The news clips included both male and female newscasters. The remainder of the loudspeakers introduced background noise to the environment, simulating 16 (4 × 4) people in conversation. The sound stream coming from the loudspeakers in front was normalized to the root-mean-square intensity and set to 73 dB Sound Pressure Level (SPL). The loudspeakers presenting the background noise were set at 64 dB SPL, resulting in a total of 70 dB SPL background noise, which gives a difference of 3 dB between foreground and background noise.

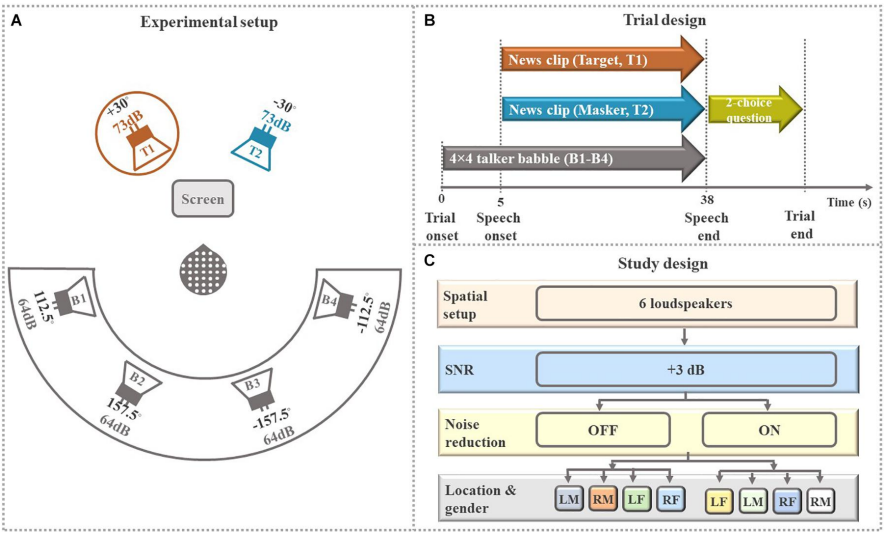


Figure 3.1 Figure taken from [Alickovic et al., 2021]: A. Shows the experiment setup used when recording the EEG. A participant is seated in the middle and prompted to listen to one speaker, ignoring the other. Multi-babel simulating background noise from 16 different people is played from the rear speakers. B. Shows the timeline of each trial procedure, namely when each sound starts playing. C. Shows the setup and division of the data, 20 trials are done for both noise reduction schemes, where each trial is divided into sections of "Left Male", "Right Male", "Left Female", and "Right Female".

During each trial of the experiment, participants were presented with background noise for five seconds prior to the speech stimuli. They were instructed to attend to either the left or right speaker while ignoring the other speaker, with the background noise playing from the rear loudspeakers. After 38 seconds, including the initial 5 seconds of background noise, participants were asked to answer a two-choice question about the content of the attended speech.

In total, 84 trials were conducted. Four trials were used to familiarize each participant with the task and were excluded from the analysis, leaving a total of 80 trials per subject analyzed in the project. Each trial consisted of a male and female talker from the front speakers. The trials were divided into four sessions, with each session using a different hearing aid setting and comprising 20 trials. Two different noise reduction algorithms were used, either activated or deactivated, resulting in a 2x2 design. Within each session, the 20 trials were divided into 4 blocks of 5 consecutive trials. Each block maintained the same direction and gender for the target speaker but used a randomized selection of sound files. For each block, a new combination of direction and gender was selected that had not been chosen previously.

Data pre-processing

The dataset was already pre-processed before its use in this project. This was primarily done to remove noisy and irrelevant components, such as artifacts arising from eye movement, muscle movement, or heartbeats. The preprocessing steps included:

- Bandpass filtering between 0.5 Hz and 70 Hz using an FIR Hamming window.
- Notch filtering with bandwidths 49-51 Hz, removing power line noise.
- Downsampling from 1024 Hz to 256 Hz
- Removal of noisy channels via visual inspection, where interpolation was used to replace the removed data.
- Independent component analysis to remove additional artifacts.

Note that from the original 34 subjects in the dataset, only 31 of these made it through all pre-processing steps. Missing subjects were removed due to persistent artifacts, alongside similar issues.

Usage

Since the dataset simulates a cocktail party situation, the diffusion models should aim to produce realistic EEG responses for AAD. In an everyday environment, a short response time (<1 second) when performing AAD is desirable, and for this reason, the EEG data is segmented into 1-second chunks for our models.

Our Preprocessing

All the project's models were implemented in Python, using the PyTorch library as a base. Since the dataset was originally in the MATLAB .mat format, it was converted to the more python accessible HDF5 format [The HDF Group, 2023], which was selected due to the large amount of data that needed to be grouped and labeled. Interfacing with the HDF5 format was done with the h5py package [Collette, 2014].

Additionally, the Python library MNE [Gramfort et al., 2013] was used to load and process raw EEG data, as well as segment and label it. These are the steps performed during the construction of the HDF5 dataset:

1. Mastoid reference channels are dropped (2 channels), leaving 64 channels.
2. The trials for each subject are split into "Passive" (0-5 seconds) and "Active" (5-38 seconds) listening sections.
3. "Passive" and "Active" listening sections are labeled depending on the trial conditions accordingly:
    - "Passive" Section $\rightarrow$ 0
    - Attending to "Male Left" $\rightarrow$ 1
    - Attending to "Male Right" $\rightarrow$ 2
    - Attending to "Female Left" $\rightarrow$ 3
    - Attending to "Female Right" $\rightarrow$ 4
4. Trials are labeled according to the noise reduction algorithm used during each trial.

This results in a dataset organized in a hierarchy with the noise reduction algorithm used being at the top level, followed by the trial subject, and finally split into two inner datasets. These two inner datasets are of the array form (Trial, Channels, Samples), with trials ranging from 1 to 20, channels being fixed at 64, and the number of samples depending on whether it is the "Passive" or "Active" listening section. This inner split into two datasets was done for performance reasons, since it allows us to easily skip reading the labels for some tasks.

During training, additional transformations were performed on the constructed dataset, which will be covered in the relevant task training section.

Segmented into 1-second samples, the full data set consists of 91960 samples, divided into 12100 passive listening samples and 79860 active listening samples. The active listening labeled data is evenly split with 39930 left labels and 39930 right labels, used for the LoA task.

Dataset split. The dataset was divided into three sets when training the models: Training set, test set, and validation set. The training set was used to estimate the model's parameters, while the test set was used for evaluating the model during training. The validation set could be used to evaluate a final version of the model when training on both the train and test sets.

The division of the dataset is done differently depending on the task. For the Active vs. Passive task, the first 12 trials for each noise reduction scheme were assigned to the training set, the trials 13-16 to the test set, and the remaining trials 17-20 to the validation set. This was done for all subjects. In the case of the LoA

task, trials were grouped in sets of 5, all with the same "Left" or "Right" label. In these sets the split was done by allocating the first three to the training set, the fourth to the validation set, and the fifth to the test set repeatedly. This was done to make sure each set had a similar distribution of "Left" and "Right" examples. This resulted in a 60/20/20 split:

- Training set trials [1, 2, 5, 6, 7, 10, 11, 12, 15, 16, 17]
- Test set trials [4, 9, 14, 19]
- Validation set trials [3, 8, 13, 18]

Due to a late discovered programming error, trial 20 was not used here. Since this is not a large part of missing data, we do not believe it had a major impact on the later results.

Normalization. To normalize the data, standardization was performed using the mean and standard deviation (SD) of the training data.

$$\mathbf{x} := \frac{\mathbf{x} - \mathbb{E}(\mathbf{x})}{\sqrt{Var(\mathbf{x})}} \tag{3.1}$$

The mean and variance are calculated with reduction on all dimensions, in this case, our EEG channels. As the training set is divided differently for the tasks, the mean and SD for both training set splits were calculated. Standardization is used for numerical stability during the training process while keeping the form of the data intact.

3.2 Diffusion Model

The implementation of the diffusion models can be divided into three main blocks:

1. Inner model
2. Scheduler
3. Data generation

Inner Model

In diffusion models, an inner network implements the estimation of parameters required for sampling. The model takes a noisy sample as input, together with the corresponding time step, and outputs a prediction. Different prediction types exist, which will be mentioned in the training section, however, all prediction types estimate the mean $\mu_\theta(\mathbf{x}_t, t)$ in the reverse diffusion kernel $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ shown in Eq. (2.6). The type of inner model used in this project is a version of the original UNet

structure implemented in [Ronneberger et al., 2015], which is a model frequently used for image domain diffusion. The implementation of UNet is taken from the HuggingFace *diffusers* library, which is built upon the more known *transformers* library [Wolf et al., 2020], which has open-source code for a wide range of machine learning models.

The UNet model utilizes the structure of an encoder and decoder, downsampling the input data to a lower dimension which inputs to a small network aptly named bottleneck, before upsampling the output back to the original input size. The input to UNet is in the form of (N, Channel/Depth, Height, Width), which in our case becomes (N, 1, 64, 256). N here represents the number of examples in one batch of input. As the EEG data is processed as an image, the height represents the channels of the EEG data and the width of the time axis. Channel/Depth represents the number of color channels when used with images, since we only have one value for each EEG channel at a specific time, we leave it as 1. The implemented UNet uses a base of six convolution blocks each for the encoder and decoder, where the number of filters used, in order are $128 \rightarrow 128 \rightarrow 256 \rightarrow 256 \rightarrow 512 \rightarrow 512$. This is reversed for the decoder part. For each step in the encoder blocks, the feature maps are downsampled with two, e.g., the input data of size (64, 256) is downsampled to (32, 128), etc. Downsampling is done using convolution and is used in all encoder blocks except the last one. In the same fashion, the decoder upsamples the feature maps by a factor of two using convolution except for the last block, which does not use upsampling. The output layer consists of a convolution with one filter to reduce the amount of channels to the original size. All convolutions in the network use a 3×3 kernel, and the activation function used is the Sigmoid-Linear Unit (SiLU) activation function. As UNet does not have any knowledge of the current time steps in the diffusion model, it is added as an additional input to the model with sinusoidal positional embeddings in the same way used in the original DDPM [Ho et al., 2020].

The basic model structure can be seen in Figure 3.2, which shows how each block in the encoder is connected to the corresponding block in the decoder with a skip connection. This implementation is similar to the one found at `https://huggingface.co/docs/diffusers/tutorials/basic_training`. The complete list of layers can be found in Listing 6.1 in the Appendix.

Scheduler

The scheduler has two primary tasks in the implementation of the diffusion model. The first is to add noise to the data during the training loop, while the second is to sample the data for the generative process. The choice of scheduler determines how these two tasks are implemented, as each scheduler uses different equations, each with its advantages and disadvantages. This project uses Denoising Diffusion Implicit Models (DDIMs) when implementing the scheduler [Song et al., 2022]. The DDIM scheduler adds noise to the data in the same way DDPM does during training, which is shown in Eq. (2.4). However, during sampling DDIM leads to a

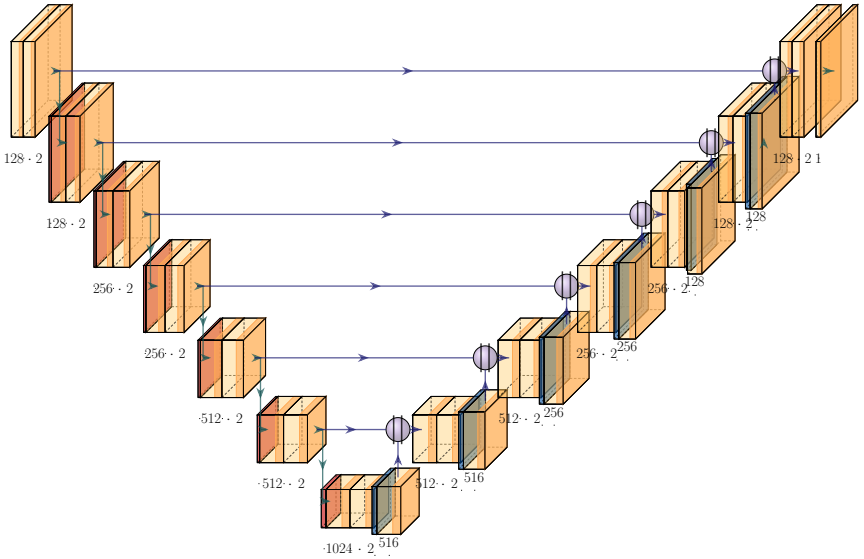


Figure 3.2 The image shows the structure of the UNet used as the inner model used for approximating the parameters of the reverse process. The boxes represent one convolution layer. The output features are provided under the boxes, where ·2 indicates the number of convolutional layers.

faster generative process compared to DDPM, as it does not require sequential time steps to produce a new sample. This is possible due to a redefinition of the forward process done by [Song et al., 2022] to instead represent a family of non-Markovian forward distributions indexed by σ:

$$q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_t, \mathbf{x}_0) = q_\sigma(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \tag{3.2}$$

where $q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_{t-1}}}, \sigma_t^2 I\right)$. This family of forward distributions now depends on the variances σ_t and is conditioned on both the previous step $\mathbf{x}_{t-1}$ and the initial data $\mathbf{x}_0$. This differs from the Markovian forward process in Eq. (2.1) that is only conditioned on the previous step $\mathbf{x}_{t-1}$. However, [Song et al., 2022] proves that training can still be done using the same MSE loss function between the added noise ϵ and predicted noise ϵ_θ seen in Eq. (2.11). The training loop of DDIM is identical to that of DDPM as it models the same parameter ϵ_θ and uses the same forward diffusion for adding noise. Since the reverse diffusion process is determined by the forward diffusion process, it is possible to use another forward diffusion q_σ defined on a subset of the latent variables $\mathbf{x}_{\tau_1}, \ldots, \mathbf{x}_{\tau_S}$ during sampling, thereby reducing the number of steps from T to S. Sampling using

this algorithm is defined by [Song et al., 2022] as:

$$\mathbf{x}_{\tau_{i-1}}(\eta) = \sqrt{\bar{\alpha}_{\tau_{i-1}}}\left(\frac{\mathbf{x}_{\tau_i} - \sqrt{1-\bar{\alpha}_{\tau_i}}\epsilon_\theta^{(\tau_i)}(\mathbf{x}_{\tau_i})}{\sqrt{\bar{\alpha}_{\tau_i}}}\right) + \sqrt{1-\bar{\alpha}_{\tau_{i-1}} - \sigma_{\tau_i}(\eta)^2}\epsilon_\theta^{(\tau_i)}(\mathbf{x}_{\tau_i}) + \sigma_{\tau_i}(\eta)\epsilon_i \quad (3.3)$$

where $\sigma_{\tau_i}(\eta) = \sqrt{\frac{1-\bar{\alpha}_{\tau_{i-1}}}{1-\bar{\alpha}_{\tau_i}}}\sqrt{1-\frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{i-1}}}}$ and η is a hyperparameter that sets the stochasticity of the process. This is the sampling function that is used in the implementation. Note that this reverse process is similar to Eq. (2.12) in that it uses the same prediction of ϵ_θ from the trained model.

The variances (β) of the diffusion model are set when initializing the DDIM schedule using a β scheduler. Our implementation uses the squared cos scheduler proposed by [Nichol and Dhariwal, 2021]. This β scheduler results in noise being added more slowly in the noise scheduler. Each β_t is now set to the value of:

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}} \quad (3.4)$$

where

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2. \quad (3.5)$$

The s is a small number added to prevent β_t to be too small when $t = 0$. The max value of β is limited to 0.999.

An estimation for $\mathbf{x}_0 \sim \left(\frac{\mathbf{x}_{\tau_i} - \sqrt{1-\bar{\alpha}_{\tau_i}}\epsilon_\theta^{(\tau_i)}(\mathbf{x}_{\tau_i})}{\sqrt{\bar{\alpha}_{\tau_i}}}\right)$, shown in Eq. (3.3) is done for each step in the sample process. The estimation is bound to the same interval as the normalized data to ensure that the sampling process uses an $\mathbf{x}_0$ estimation within normalized values. This is done with dynamic thresholding in our model, proposed by [Saharia et al., 2022], which helps prevent saturation of samples produced. Dynamic thresholding is done by calculating the quantile of the absolute value of the predicted $\mathbf{x}_0$ at time t. This is done to get the edge value s, with the percentile used as a hyperparameter. The values are then thresholded using the interval $[-s, s]$ and divided by s to result in a boundary of $[-1, 1]$. This method avoids saturating the values near the boundary of the data, which occurs in static thresholding. However, as the EEG dataset is not normalized to [-1, 1], and instead standardized, the code is modified to not divide by s, giving the sample the bound of $[-s, s]$. While the data is not bounded, the values outside [-5,5] after standardization contain most-to-only outliers. The percentile used for dynamic thresholding was experimentally determined to fit most of the data while excluding outliers.

Spectral Loss

The MSE loss used does not capture the spectral properties of the dataset, as the data is expressed in the time domain. Therefore, an additional loss function to complement the MSE loss was introduced which considers these spectral components of the data, named spectral loss. It is defined as:

$$L_{spec} = \frac{1}{N}\| |STFT(\mathbf{x})| - |STFT(\hat{\mathbf{x}})| \|_2^2 \tag{3.6}$$

where $\mathbf{x}$ is the data, $\hat{\mathbf{x}}$ is the predicted data and N is the amount of elements in $\mathbf{x}$. It is inspired by the spectral loss used as a reconstruction loss of a VQ-VAE in [Dhariwal et al., 2020] to consider mid and high frequencies.

Magnitudes of the MSE and Short-Time Fourier Transform (STFT) losses are taken into consideration by normalizing the STFT loss to the interval [0, 1]. As the loss has no set max and min values and will never be negative, the function:

$$\frac{1+x}{2(1+|x|)} - 0.5 \tag{3.7}$$

is used for scaling the data. In contrast to the MSE loss function that uses the noise, data, or v-prediction depending on the prediction type, the loss that uses STFT needs the prediction from the model to be in the original data format. In the case of the model predicting noise, the noisy images would be denoised with the predicted noise using the scheduler in reverse. The transformation of the data from the time domain to the frequency domain is done using PyTorch's `torch.STFT` function. This function uses the discrete STFT which is then clamped so that the output is never lower than 1e−8 for numerical stability. The resolution in frequency against the time of the short-Fourier transformed data depends on three arguments: The size of the FFT, the length of the window used, and how far the window jumps for each calculation. This causes the loss to depend on set hyperparameters, resulting in vastly different loss. To reduce this dependency, the mean of multiple resolutions of the STFT is taken, preventing overfitting to one STFT representation and allows the model to consider different time-frequency characteristics of the data [Yamamoto et al., 2020]. The full implementation of the losses is a modified version of the code available from [Steinmetz and Reiss, 2020].

Training

Prediction type. The standard for diffusion models is to predict the noise (ϵ) added to the sample. This can be changed to predicting the sample ($\mathbf{x}$) itself or using a combination of both, called v-prediction introduced by [Salimans and Ho, 2022]. They define the new prediction target $\mathbf{v}$ as:

$$\mathbf{v} \equiv \alpha_t \epsilon - \sigma_t \mathbf{x} \tag{3.8}$$

meaning the prediction target becomes a weighted sum of the added noise and the data. The math for the diffusion has small differences in these cases, however, the loss

will still be the MSE of the target and prediction, albeit with the data or v-prediction used instead of the noise.

Training loop. During the training of the model, the amount of steps used in the diffusion process is set to $T = 1000$ similar to [Sohl-Dickstein et al., 2015], [Ho et al., 2020]. The input data is a one-second segment from the recorded EEG data with the shape [N,1,64,256], where N is the number of examples in a batch. The N was varied depending on which platform was used to train the model. The final diffusion models were trained with N set to 64.

For a batch of the training data, noise is sampled from a standard normal distribution with the same dimension as the input data ([N,1,64,256]), and a random time step t is chosen for each data sample in the batch. Every time step t determines how much noise progression should be made for that image. Noise is then added to each EEG data in the batch with the noise scheduler using the time steps t and sampled noise. These noisy images are fed to the model, which tries to predict the noise added to each image, the original image, or v-prediction depending on the set prediction target. The MSE loss is calculated using this prediction and ground truth. If spectral loss is used then this will be calculated with the original image and the predicted one, which in case the model predicts noise is extracted by reversing the addition of noise on the images using the predicted noise. The full loss would then be a linear combination of the MSE loss and spectral loss:

$$L = L_{MSE} + L_{\text{spec}}. \tag{3.9}$$

The model parameters are updated using the default values of the PyTorch AdamW optimizer, with the learning rate set to 1e-4.

Several models with different loss functions and prediction targets were tested. The full combinations of tests can be seen in Table 3.1.

Table 3.1 The table shows every combination of prediction type and loss function tested for the diffusion model. The L_{MSE} is the MSE loss, L_{spec} is the Spectral loss. In the target column, $\mathbf{x}$ means that the loss function uses the EEG data itself as the target, while ϵ means that the target is the noise added to the image. $\mathbf{v}$ from Eq. (3.8) is the weighted combination of both.

Model label	Loss function	Target
Epsilon	L_{MSE}	ϵ_t
V-pred	L_{MSE}	$\mathbf{v}_t$
Spectral	$L_{MSE} + L_{\text{spec}}$	$\mathbf{x}_t$

The shared hyperparameters for training the models are seen in Table 3.2. For all trained models, a cosine learning rate with warmup was used. This learning rate scheduler is tested in [He et al., 2018], showing a small improvement in performance for their tested models. The idea is to use a smaller learning rate in the beginning (in this case 0), and linearly increase it to the maximum value, where the incline

depends on the number of warmup steps used. This ensures stability in the learning process, as the starting parameters set initially in the model may have values far from the solution, where having a high learning rate may introduce instability. When the initial learning rate is reached, the values decay according to a cosine function. The learning rate η_t is computed at batch T as:

$$\eta_t = \frac{1}{2}\left(1 + \cos\frac{t\pi}{T}\right)\eta \tag{3.10}$$

where η is the learning rate set as a hyperparameter. The implementation uses only half a cosine cycle, which means the learning rate only decays and never increases. The amount of warmup steps used is 500.

Algorithm 1 Training

1: The target $\mathbf{z}$ is set to one of $\mathbf{x}, \epsilon, \mathbf{v}$
2: **for** $\mathbf{x}_0$ in Batch **do**
3: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $t \sim \text{Uniform}(1, \ldots, T)$
5: $noisy\_images \leftarrow \texttt{DDIMScheduler}(\mathbf{x}_0, \epsilon, t)$
6: $\hat{\mathbf{z}} \leftarrow \texttt{UNet}(noisy\_images, t)$
7: $L \leftarrow Loss\_function(\hat{\mathbf{z}}, \mathbf{z})$
8: Update `UNet` weights using AdamW optimizer with L
9: **end for**

Table 3.2 The standard parameters used when training the diffusion models

Hyperparameter	Value
Time steps	1000
Batch size	64
Learning rate	1e-4
Learning rate scheduler	Cosine with warmup and decay

Data generation

The synthetic data is generated using the *diffusers'* DDIM pipeline, which uses the noise scheduler and streamlines the full reverse diffusion process. Normal distributed noise, with the same shape as the EEG data, is sampled and input with the time step T to the trained neural network. The neural network's prediction, the Normal distributed noise, and the time step are input to the scheduler which denoise the image one step. The partly denoised image is then used as input to the model together with time step $T - 1$ and the resulting prediction is used with the scheduler to denoise the partly denoised image. This is done for a set amount of sampled time steps from the $T = 1000$ time steps used during training.

Since the diffusion models are not given any information about the underlying sample label, a separate model is trained for each label. In the LoA task, 25000 samples of EEG data are generated for each label. This is changed to 45000 samples for the Passive vs. Active listening task due to the larger training set. The generated data for the different tasks are saved in separate files.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: Subsample time steps $\tau_S \subset [T, \ldots, 1]$
3: **for** $t = \tau_S$ **do**
4: $\quad \hat{\mathbf{z}} \leftarrow \texttt{UNet}(\mathbf{x}_t, t)$ ▷ Predict the target $\hat{\mathbf{z}}$
5: $\quad \mathbf{x}_{t-1} \leftarrow \texttt{DDIMScheduler}(\hat{\mathbf{z}}, \mathbf{x}_t, t)$ ▷ Compute previous image $\mathbf{x}_{t-1}$
6: **end for**
7: **return** $\mathbf{x}_0$

The default diffusion pipelines implemented in the *diffusers* library clamp the generated data to [-1,1]. Due to the use of standardization, this was not desired for our synthetic EEG data. To circumvent this, a custom pipeline was made, using the original implementation, but skipping the clamping step.

Evaluation

For evaluating the data quantitatively, Jensen-Shannon divergence was used, which is a measure of similarity between two probability distributions. It is a symmetric version of the KL-divergence which uses the average of its output using both distributions as reference and is defined as:

$$D_{JS}(P\|Q) = \frac{1}{2} \cdot D_{KL}(P\|\frac{P+Q}{2}) + \frac{1}{2} \cdot D_{KL}(Q\|\frac{Q+P}{2}). \tag{3.11}$$

Where P and Q are distributions. The distributions of the EEG dataset and the generated data are difficult to calculate due to the interdependency of the data points, as such, the distributions were approximated. This was done using histograms, which removes the temporal and spatial information of the data. When calculating the histograms, 15000 random samples are drawn from the real and generated EEG data for each label. The histograms are calculated for each channel using 200 bins, using of a range of [-10,10]. The resulting histograms are then used in calculating the Jensen-Shannon divergence. The P and Q in Eq. (3.11) are therefore histograms for each channel of the data. The overall Jensen-Shannon Divergence score is the mean score calculated over all channels. For inspecting the distribution of the data, the channel distributions are pairwise overlaid in a plot, where each pair forms the corresponding real and synthetic distribution.

3.3 Classifier Network

For the classification tasks, LoA and Passive vs. Active listening, the classifier EEGNeX from [Chen et al., 2024] was used. The original paper's implementation was done in TensorFlow, which had to be re-implemented in PyTorch for use in this project. EEGNeX was chosen as the classifier model due to the improved performance on a wide range of classification tasks compared to previous convolutional neural network (CNN) models used for EEG data, e.g EEGNet [Lawhern et al., 2018; Chen et al., 2024].

Model structure

The CNN structure used is a PyTorch implementation of EEGNeX which uses 5 convolutional blocks and a fully connected layer. The full setup can be seen in Figure 3.3. This CNN uses the Exponential Linear Unit (ELU) activation function instead of the more common Rectified Linear Unit (ReLU) and includes dropout in two layers to prevent overfitting along with average pooling. The most noticeable difference compared to a regular CNN is that the middle layer utilizes depth-wise convolution. This means that convolution is done on each channel separately, improving the extraction of spatial features in the EEG data [Chen et al., 2024]. We leave out the final softmax activation layer in our implementation due to the use of PyTorch.

The original version of EEGNex by [Chen et al., 2024] uses a depth of 2 for the depth-wise convolution, while our implementation uses 4 instead, which adds to the number of filters used. The training is done using a batch size of 64, running for 100 epochs, with a fixed learning rate of 5e-4. To prevent overfitting on the training set, the model with the lowest loss on the test set across all epochs is saved at the end of training.

Evaluation

The main evaluation metric used for the classifier tasks is accuracy. Accuracy measures the ratio of correctly classified samples out of the total number of samples classified. This metric is one of the main evaluation methods used in papers comparing LoA models. However, on an imbalanced dataset, classification accuracy can often give a misguided view of the models' performance. As an example, in an 80/20 class split, a model could classify everything as the dominant class to achieve an accuracy of 80%, while not having learned anything about either class. In our Passive vs. Active listening task, wrong classifications of the Passive listening labels have a much smaller effect on the accuracy. Therefore, an additional metric in the form of the F1-score is taken into consideration for the unbalanced case. The F1-score is defined as the harmonic mean of recall and precision:

$$\mathrm{F1} = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.12}$$

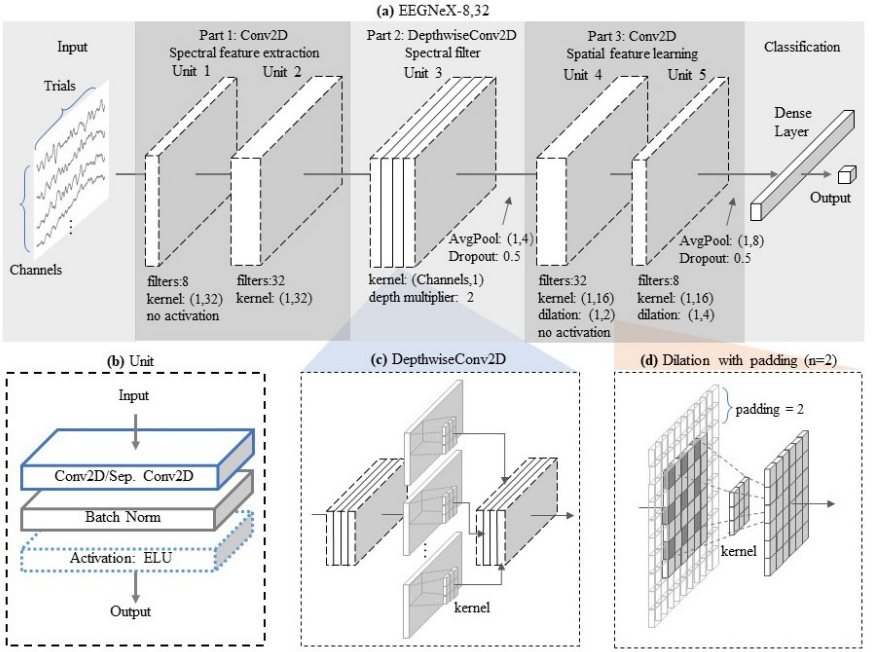


Figure 3.3 Figure 4 taken from [Chen et al., 2024] showing the EEGNeX-8,32 architecture. (a) Shows an overview of the general structure, (b) Gives a view of a single unit, (c) is an illustration of the depthwise convolution operation and (d) illustrates the effect of padding.

where recall and precision are defined as:

$$recall = \frac{TP}{TP + FN} \quad precision = \frac{TP}{TP + FP}. \tag{3.13}$$

The metrics of recall and precision provide insight into the model's performance from different perspectives. True Positives (TP) represent instances correctly identified as belonging to the positive class, while False Negatives (FN) are instances that belong to the positive class but are incorrectly classified as negative. Conversely, False Positives (FP) refer to instances that are wrongly identified as the positive class. The 'positive' designation is relative and can refer to either class when calculating precision and recall, with each class taking a turn as the 'positive' in separate calculations. Specifically, "recall describes how large a proportion of the positive data points are correctly predicted as positive" and "precision describes what the ratio of true positive points are among the ones predicted as positive" [Lindholm, 2022]. In the context of amplifying and attenuating sound in a hearing aid, such as when using the models in the Passive vs. Active listening task, precision could arguably be more crucial than recall. Since amplifying noise (Passive) could be considered worse for the user in contrast to a failure in amplification for attended

sound (Active). However, since this could be argued in both directions, we settled on using the F1-score, giving an equal weight to both metrics.

As F1-scores are calculated for the classes separately, their weighted mean is taken as the final score. By weighting the F1-score based on the occurrences of each class, the F1-score of the dominant class impacts the mean F1-score more than the other class by the fraction of the distribution.

The last measurement, supporting accuracy and the F1-score, is the AUC-ROC score. This score is obtained by calculating the Area Under the Curve (AUC) of the Receiver Operating Characteristics (ROC) curve. This gives a measure of the overall model for using different decision thresholds, where the one used in our model is 0.5. The ROC curve is plotted for recall against the false positive rate (FP/N where N is the total amount of the negative class) [Lindholm, 2022]. For the Passive vs. Active listening task, the Active class is considered the positive, which becomes the Right class in the LoA task.

Pipeline. Testing of the classification model was done using identical parameters and adding the synthetic data generated by the different diffusion models listed in Table 3.1. The model was trained with synthetic data constituting a predetermined percentage of the training set, and the performance was assessed using the evaluation metrics combined with prior knowledge about the quality of the generated data by the Jensen-Shannon divergence.

To accommodate model variance, we trained 20 iterations of both the baseline classifier models and those supplemented with synthetic data. This approach allowed us to calculate confidence intervals for our metrics and estimate the mean performance of the models. We tested augmenting the training data by 15%, 30%, 60%, and 100% with the generated synthetic data, to evaluate how the ratio of synthetic data would affect performance.

Testing for the Active vs. Passive listening tasks used both the inherently unbalanced dataset and a version adjusted for balance through oversampling of the Passive class. We employed a sliding window approach with a window length of 256 (equivalent to a 1-second sample) and a stride of 32, yielding an approximately eightfold increase in Passive class samples to balance the dataset.

Classifier training adhered to a data split ratio of 60% for training, 20% for testing, and 20% for validation. The models were trained on the training data set and evaluated on the test set. While models were trained and assessed using the training and test sets, a final model could be trained on the combined training and test sets and then evaluated on the validation set. However, due to the time constraints of the thesis, this was not performed.

As the diffusion models produce samples using normal distributed noise, another prospect to examine would be adding similar noise to the original EEG samples. Therefore, an additional 20 models of the classifier having 15% added noise of the total training set were trained. This is referred to as the noise addition model, which provides a comparison to a simpler data augmentation method that a degenerate

diffusion model might mimic.

4

Results

The results are split into two different subsections: Diffusion and Classifier. The Diffusion section mainly presents how well the generated synthetic EEG data matches the training set, based on the loss function used during training. The impact on classification performances is not considered here. Some results are also presented, which are considered training set "agnostic", meaning the underlying label is not the main focus. Instead, the focus is on general EEG synthesis.

In the Classifier section, the performance of the classifiers on the LoA task and the two different Passive vs. Active listening tasks is presented. The main focus here is to determine if there was a significant improvement when using generated EEG data for data augmentation, contrasting it with a baseline model as well as with a simple noise addition approach.

4.1 Diffusion

The first objective metric assessing the performance of our diffusion models in generating synthetic EEG data is presented in Tables 4.1 and 4.2. The underlying data label appears to have a minimal effect on the divergence, with similar values across both task labels. The Epsilon and Spectral models exhibit comparable divergences, particularly for the Passive vs. Active listening labels, with a slight difference for the LoA labels. Notably, the v-prediction model shows the highest divergence for both sets of labels, except for the Passive listening label.

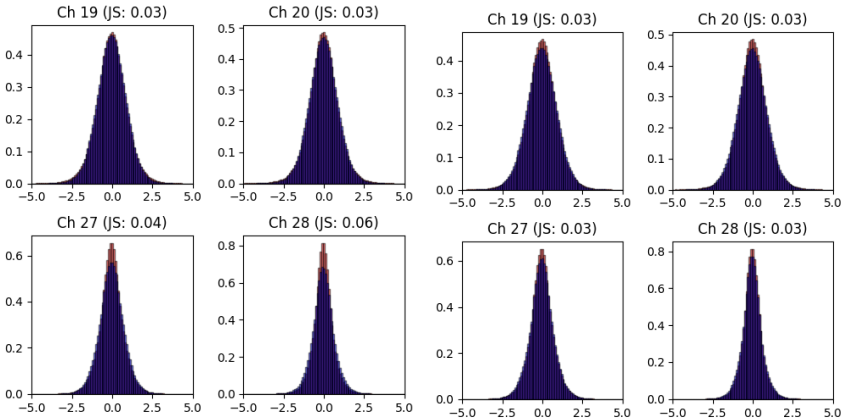
Table 4.1 Jensen–Shannon divergences for the LoA task. Lower values indicate a better fit to the cross-channel value distribution of the real normalized data.

Model	Label	Jensen–Shannon divergence
Epsilon	Left	0.026
	Right	0.028
V-pred	Left	0.064
	Right	0.072
Spectral	Left	0.042
	Right	0.047

Table 4.2 Jensen–Shannon divergences for the Passive vs. Active listening task. Lower values indicate a better fit to the cross-channel value distribution of the real normalized data.

Model	Label	Jensen–Shannon divergence
Epsilon	Active	0.032
	Passive	0.042
V-pred	Active	0.067
	Passive	0.037
Spectral	Active	0.030
	Passive	0.037

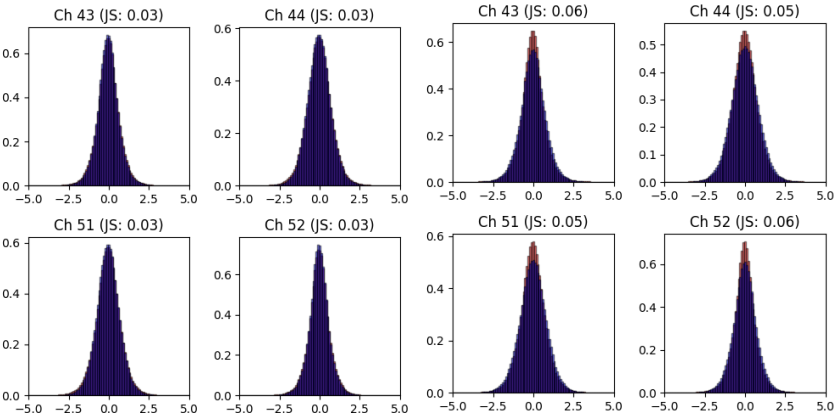
A visual comparison of the Jensen-Shannon divergences is provided in Figures 4.1 and 4.2. Figure 4.1 shows the difference between diffusion models, while Figure 4.2 shows the difference based on the underlying label. For most channels and values, the distributions exhibit a close match, making it visually challenging to see the difference. Most of the differences can be seen around 0, where the real EEG data distributions show more occurrences. Overall, the differences in the divergence between models or labels are minimal. Additional plots of the Jensen-Shannon divergence showing all channels are available in the Appendix.



(a) Spectral diffusion model **(b)** Epsilon diffusion model

Figure 4.1 Jensen–Shannon divergences for a selection of individual channels for two different diffusion models. Red histogram bars represent the real data distribution, blue the synthetic. The largest difference is visible around 0.

A more direct visual comparison between real EEG and synthetic EEG can be seen in Figures 4.3 and 4.4. A randomly picked 1-second segment of real EEG data is compared with synthetically generated EEG data. Overall, the synthetic data closely resembles real EEG data, matching longer frequencies and showing channel



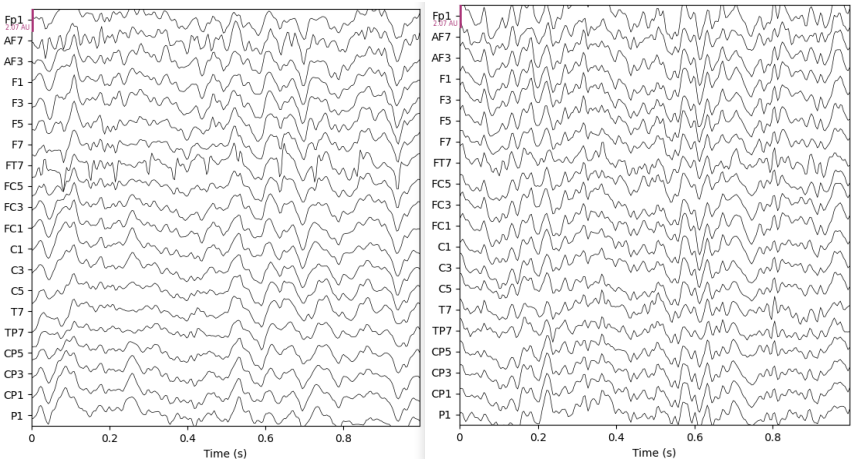
(a) Epsilon model (Active) **(b)** Epsilon model (Passive)

Figure 4.2 Jensen–Shannon divergences for a selection of individual channels for Passive and Active listening classes. Red histogram bars represent the real data distribution (for the class), blue the synthetic. The largest difference is visible around 0.

correspondence. Values around 0 are somewhat less well represented, although it might be challenging to observe this in a single sample. A 2D representation of the same type of comparison is illustrated in Figure 4.3. This is achieved by scaling the EEG data across channels between -1 and 1 before applying a colormap, with -1 shown as dark blue, 0 as white, and 1 as dark red. In this representation, it is easier to see patterns across channels, as well as the comparative amplitudes of the channel signal. It is subtle, but here we can also see a slight tendency of the real EEG data to have more values close to 0, as well as a smoother gradient between high and low values.

(a) Real EEG data **(b)** Synthetic EEG data

Figure 4.3 2D image plot comparing real and synthetic EEG. 1 second of EEG data is treated as an image with channels corresponding to each horizontal line of pixels. Values are mapped to colors using the 'seismic' colormap from `matplotlib`.



(a) Real EEG data **(b)** Synthetic EEG data

Figure 4.4 Comparison between a segment of real EEG data and synthetic EEG data generated with diffusion. A random selection of 1 second of data is plotted using MNE with a subset of all 64 channels shown. Real EEG data is normalized in the same way as when training the diffusion model, as such the units are arbitrary.

4.2 Classifier

In this section, we will go over the 3 different classification tasks and the respective value of adding synthetic EEG data from the different diffusion models. These are the LoA task and the two Passive vs. Active listening tasks.

Locus of Attention

In Figure 4.5 we can see a comparison of the classification accuracy between the baseline, the noise addition model and the different diffusion models. Most notable is the performance of the noise addition model compared to the baseline, where we see an approximate 3% drop in the mean accuracy. In contrast, most diffusion models are on par with the baseline, with some performing slightly above and below the baseline.

To see if any model has a significant difference in classification accuracy compared to the baseline, an independent Welch's t-test is performed. The results of this can be seen in Table 4.3. In the table we can see that only 3 models significantly differ from the baseline:

- Noise addition
- Epsilon with 60% added data

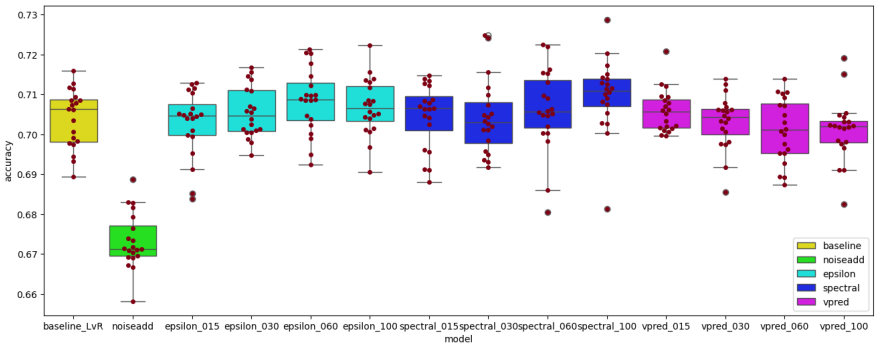


Figure 4.5 A box-plot comparison of classification accuracy results for the different classifier models in the LoA task, where each dot represents a single model run result. Outliers are represented as circled dots outside the confidence intervals.

- Spectral with 100% added data

The significance of the noise addition can also be seen in Figure 4.5 and from it we know it imparts a negative effect on the accuracy.

Table 4.3 P-values for an independent t-test between the baseline and the different models at different ratios of generated data added (n = 20). $\ll$ **0.01** signify values less than 10^{-4}.

Model	Generated data	P-Value (Acc)
Epsilon (LoA)	0.15	0.599
	0.3	0.491
	0.6	**0.079**
	1.0	0.219
Spectral (LoA)	0.15	0.808
	0.3	0.908
	0.6	0.454
	1.0	**0.035**
V-pred (LoA)	0.15	0.363
	0.3	0.720
	0.6	0.243
	1.0	0.238
Noise Addition	0.15	$\ll$ **0.01**

For the other two significant models their estimated improvement compared to the baseline can be seen in Table 4.4. Since the confidence intervals significantly overlap it is not clear one is better than the other for this sample size. We can however note that the confidence interval for the classification accuracy mean is entirely positive for the spectral model.

Table 4.4 Mean improvement (n = 20) for models with significant P-values (< 0.10).

Model	Generated data	Mean Improvement	95% CI
Epsilon (LoA)	0.6	0.44%	-0.02% to 0.90%
Spectral (LoA)	1.0	0.58%	0.18% to 0.98%

Exact mean values for classification accuracy, the F1-score and AUC can be seen in Table 4.5. Accuracy for all models except noise addition is around 70-71%, with spectral at 100% added data achieving the highest at almost 71%. We can note that all three metrics have good correspondence, which is expected with a 50-50 class split.

Table 4.5 Mean statistics (n = 20) for the LoA task, best results for each augmented model marked in bold.

Model	Generated data	Accuracy (%)	F1-score	AUC
Baseline	0	70.40	0.7033	0.7815
Epsilon	0.15	70.27	0.7019	0.7818
	0.30	70.55	0.7048	0.7834
	0.60	**70.84**	**0.7074**	**0.7877**
	1.0	70.69	0.7060	0.7858
V-pred	0.15	**70.58**	**0.7054**	**0.7819**
	0.3	70.32	0.7027	0.7818
	0.6	70.11	0.7002	0.7790
	1.0	70.11	0.7001	0.7793
Spectral	0.15	70.46	0.7041	0.7827
	0.3	70.43	0.7033	0.7846
	0.6	70.62	0.7054	0.7870
	1.0	**70.98**	**0.7088**	**0.7908**
Noise	0.15	67.33	0.6727	0.7445

Passive vs. Active listening

The Passive vs. Active listening task is a bit more complex with an unequal class split of 5-33. For this reason we consider the weighted average F1-score first in Figure 4.6 and then the classification accuracy in Figure 4.7

Figure 4.6 shows that none of the models significantly improved compared to the baseline, with many having a slight negative impact on the F1-score. Of note is that it might appear that many models show a much higher variance for the F1-score compared to the baseline. However, the scale for the score is quite slim, only being plotted between 0.83-0.86.

In Figure 4.7 we can see that classification accuracy becomes worse for most models compared to the baseline, with noise addition being the only one comparable

in performance. We can note that a classifier only predicting Active listening here would achieve a classification accuracy of just under 87%, which all models surpass.

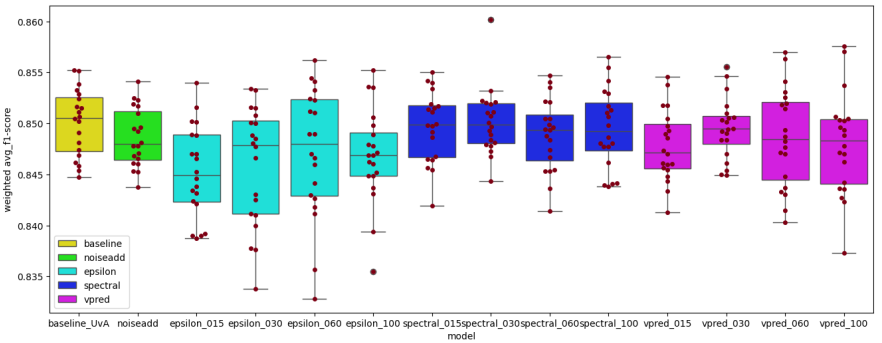


Figure 4.6 A box-plot comparison of F1-score values for the different classifier models for the Passive vs. Active listening task. Each dot corresponds to a single model run result. Outliers are represented as circled dots outside the confidence intervals.

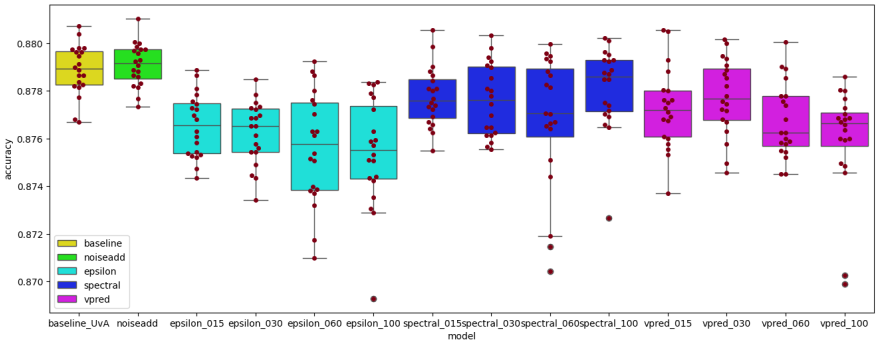


Figure 4.7 A box-plot comparison of classification accuracy results for the different classifier models in the Passive vs. Active listening task. Each dot corresponds to a single model run result. Outliers are represented as circled dots outside the confidence intervals.

Table 4.6 shows that several models indeed significantly differ from the baseline, which is also visually clear in Figures 4.6 and 4.7. Unfortunately from the box-plot we can infer that this is a negative effect. This is further seen in Table 4.7, where only noise addition has a higher mean accuracy compared to the baseline. The F1-scores and AUC scores also don't see any significant positive improvement compared to the baseline, including the noise addition model.

Table 4.6 P-values for an independent t-test between the baseline (Passive vs. Active listening Task) and the different models at different ratios of generated data added (n = 20). $\ll$ **0.01** signify values less than 10^{-4}.

Model	Generated data	P-Value (F1)	P-Value (Acc)
Epsilon	0.15	$\ll$ **0.01**	$\ll$ **0.01**
	0.30	**0.008**	$\ll$ **0.01**
	0.60	**0.078**	$\ll$ **0.01**
	1.0	**0.015**	$\ll$ **0.01**
Spectral	0.15	0.713	**0.004**
	0.30	0.960	**0.004**
	0.60	0.272	**0.004**
	1.0	0.576	0.104
V-pred	0.15	**0.038**	**0.001**
	0.30	0.556	**0.011**
	0.60	0.275	$\ll$ **0.01**
	1.0	0.122	$\ll$ **0.01**
Noise Addition	0.15	0.137	0.497

Table 4.7 Mean statistics (n = 20) for the statistics for the Passive vs. Active listening task, best results for each augmented model marked in bold.

Model	Generated data	Accuracy (%)	F1-score	AUC
Baseline	0	87.89	0.8501	0.7915
Epsilon	0.15	**87.65**	0.8452	0.7854
	0.3	87.62	0.8459	**0.7870**
	0.6	87.56	**0.8472**	0.7858
	1	87.54	0.8468	0.7860
V-pred	0.15	87.73	0.8478	0.7901
	0.3	**87.77**	**0.8495**	**0.7935**
	0.6	87.68	0.8486	0.7902
	1	87.60	0.8480	0.7909
Spectral	0.15	**87.78**	0.8497	**0.7896**
	0.3	87.76	**0.8502**	**0.7896**
	0.6	87.67	0.8489	0.7855
	1	87.81	0.8495	0.7860
Noise	0.15	87.91	0.8486	0.7884

Passive vs. Active (Balanced)

For the Passive vs. Active listening (Balanced) task, we oversample the Passive listening class by using overlapping time windows on the EEG data. This achieves a balanced 50-50 class split for our class labels.

Figure 4.8 shows that an increase in generated data seems to be correlated with a

decrease in overall classification accuracy across all diffusion models. Variance also increases for most models, except for the v-prediction model at 15% generated data added. Visually, it is clear that none of the models improved classification accuracy compared to the baseline. Noise addition is comparable here to the baseline, since there is still a significant overlap of confidence intervals.

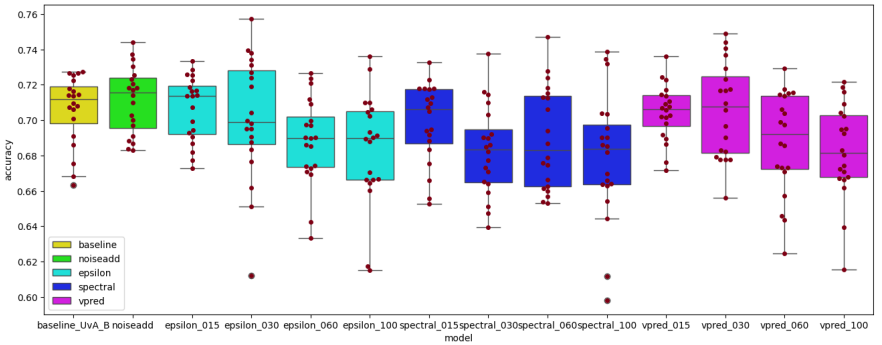


Figure 4.8 Box-plot comparison of accuracy between the different classifier models for the Passive vs. Active listening (Balanced) task where each dot represents a single model run result. Outliers are represented as circled dots outside the confidence intervals.

Looking at Table 4.8 we can see that an increase in generated data added generally increases the significance of being different from the baseline. In this case, we know it has a negative effect on the classification accuracy from Figure 4.8. We can also confirm that the noise addition model is not significantly different from the baseline.

Finally, in Table 4.9 we can see that while both the noise addition model and the epsilon model at 15% added generated data show a higher mean classification accuracy compare to the baseline, we know from the t-tests in Table 4.8 that they cannot be considered significant. Since the rest of the models do not show an increase in the mean classification accuracy compared to the baseline, their significance point to a negative effect on the mean.

Model	Generated data	P-Value (Acc)
Epsilon (PvA)	0.15	0.929
	0.3	0.596
	0.6	**0.016**
	1.0	**0.012**
Spectral (PvA)	0.15	0.283
	0.3	**0.002**
	0.6	**0.041**
	1.0	**0.006**
V-pred (PvA)	0.15	0.775
	0.3	0.944
	0.6	**0.022**
	1.0	**0.003**
Noise Addition	0.15	0.392

Table 4.8 P-values for an independent t-test between the baseline (Passive vs. Active listening (Balanced) task) and the different models at different ratios of generated data added (n = 20)

Model	Generated data	Accuracy (%)	F1-score	AUC
Baseline	0	70.60	0.7516	0.7920
V-pred	0.15	70.44	0.7503	0.7857
	0.3	**70.55**	**0.7508**	0.7858
	0.6	68.73	0.7366	**0.7870**
	1.0	68.26	0.7329	0.7854
Epsilon	0.15	**70.65**	**0.7519**	0.7837
	0.3	70.12	0.7473	**0.7849**
	0.6	68.82	0.7374	0.7817
	1.0	68.38	0.7337	0.7816
Spectral	0.15	**69.87**	**0.7457**	**0.7859**
	0.3	68.27	0.7330	0.7855
	0.6	68.94	0.7383	**0.7859**
	1.0	67.88	0.7295	0.7850
Noise	0.15	71.12	0.7557	0.7907

Table 4.9 Mean statistics (n = 20) for the statistics for the Passive vs. Active (Balanced) task, best results for each augmented model marked in bold

5

Discussion

The Discussion and Conclusion sections will interpret the significance of thesis results, and assess the extent to which the objectives of the thesis were accomplished. At the end, we will suggest potential improvements, as well as outline future research directions to build upon the findings of this project.

5.1 Diffusion Models

The results demonstrate that the diffusion models tested demonstrate a strong capability to learn to generate synthetic EEG data. This assertion is supported by the Jensen-Shannon divergence presented in Tables 4.1 and 4.2, which indicate a close match between the distribution of values in synthetic and real data. In addition, since the Jensen-Shannon divergence does not consider any temporal relation between data points, Figures 4.4 and 4.3 provide visual evidence that the synthetic data also closely resembles the real data temporally, suggesting that the models have effectively captured many frequency components of the EEG data.

However, as noted in the results, the diffusion models encounter challenges in achieving a matching distribution for values around 0. The reasons for this are not entirely clear, but may be related to the inherent train-test mismatch of the sampling process for the diffusion models. As discussed in [Saharia et al., 2022], while their dynamic thresholding approach mitigates the issue of oversaturation for images, it does not fully resolve the train-test mismatch. Since we adopted the dynamic thresholding to work for values beyond the described normalization range of $[-1, 1]$, it is plausible that this modification could contribute to the slight tendency towards extreme values observed in the generated synthetic data.

Data Duplication

Another aspect that we have not looked into is the degree of duplicated EEG data being produced by the diffusion models. Duplication of training data has been a large topic of discussion in the image generation domain, with [Somepalli et al., 2023] showing that this is an issue for many diffusion models. Relevant to this work are the results that the number of training examples directly impact the possibility of duplication occurring, which is a risk for the Passive listening class. This issue

is further compounded by the fact that many performance metrics, such as the MSE loss, will by design favor models that replicate data.

The effect this has had on the results is hard to interpret, since we have no measure of the degree of replication for the implemented models. We could speculate that a larger degree of replication could cause a classifier to overfit faster on the training set, due to the addition of duplicated examples in each epoch. This could then lead to a loss in generalization performance, since duplicated examples will increase the training set bias. Whether this has happened for the models tested is worthy of further investigation.

Differences Between Models

Another area of interest is how the metrics change between the different diffusion models. As mentioned in the results, which can also be seen in Tables 4.1 and 4.2, v-prediction showed the highest divergence, with the two other targets being much more comparable.

One potential reason why v-prediction had a harder time fitting the underlying distribution could be the mismatch when compared to the original paper's training setup. In the paper introducing v-prediction [Salimans and Ho, 2022], they primarily use it for their teacher-student model. Although they do some testing using v-prediction for standard single-model diffusion training, the metrics they report are only interpretable for the image domain.

Epsilon and spectral models are quite similar in terms of Jensen-Shannon divergence. This makes sense since the spectral model in theory only adds additional losses for mismatches in frequency components, while the Jensen-Shannon divergence does not take these into account.

Differences Between Class Labels

We can also consider the differences between labeled EEG data. In Tables 4.1 and 4.2 we can see that the Jensen-Shannon divergence changes little between labels, with one exception for the v-prediction model trained on the Passive label.

Why v-prediction performs much better on the Passive label is not clear. The v-prediction model might require more training to reach a similar level of convergence compared to the other models. This observation, coupled with the fact that the Passive label constitutes the smallest subset of the EEG data, may suggest that v-prediction only had sufficient training time to converge on this label.

As to why the labels in general appear to have little impact on the divergence, most likely this is because more general EEG-characteristics make up the majority of the value distribution. This could imply that the differences between labels are not primarily found in the value distribution and instead appear in the frequency domain.

5.2 Classification Tasks

From the results, we observe an increase in classification accuracy for two different diffusion models on the LoA task. The epsilon model with 60% added data and the spectral model with 100% added data, seen in Table 4.4. The spectral model achieved the largest mean improvement, with an increase of **0.58%** over the baseline accuracy of **70.4%**. This leaves the best model just under a **71%** classification accuracy on the LoA task for a 1-second time-window, with values up to **71.4%** being inside the confidence interval. This indicates that the diffusion models likely generated EEG data with novel information about the underlying labels not previously learned by the baseline model.

Looking at the trend for epsilon in Figure 4.5, the models seem to improve with higher ratios of added data up to a certain point. The same cannot be said for both the v-prediction and spectral models, where v-prediction models trend downwards and the spectral models see a slight decrease at 30% added data. However, it is clear that a significant difference for this effect cannot be seen or claimed for the amount of runs we have, but it is something that could be investigated in future work.

Impact of Class Split

The diffusion models struggled to achieve any increase in performance on the Active vs. Passive listening tasks. This can be seen in both Tables 4.7 and 4.9, where most diffusion models were on par with the baseline or performed worse. This holds for both the imbalanced case and the balanced case achieved through oversampling of the minority class. This shows that the underlying class split in this case seems important to achieve any performance increases by using diffusion generated EEG data.

Even when considering the weighted average F1-score, which takes into account the recall and precision of both classes, no improvements in performance compared to the baseline can be seen. This coupled with a decrease in accuracy suggests that the performance decrease across models was not offset by a performance gain for the opposite class.

One possible cause is that there was not enough of the Passive listening EEG data for the diffusion model to learn the features of Passive listening. In addition to this, as mentioned in Section 5.1, data duplication is a possible risk that is more likely to occur on a smaller subset of the data.

The way generated EEG data was added could also be suboptimal for the imbalanced case, as we base the amount added on the underlying number of class examples in the training set. This results in a much smaller number of generated Passive listening examples being added compared to the Active listening examples. It might be the case that another scheme, such as adding more synthetic examples for the minority class, might be more optimal.

Comparison to Noise Addition

The introduction of a noise addition model as a comparison to the diffusion models was important to make sure the diffusion models were not comparable to the addition of noisy data.

From the results, it is clear that the diffusion models differ from the noise addition model, especially in the LoA task. As seen in Figure 4.5, noise addition was the only model with a decrease in classification accuracy compared to the baseline. This strongly suggests that the diffusion models differ from the noise addition method, and are not comparable to adding noise to the original data.

In the Passive vs. Active listening tasks, the noise addition model seems more comparable in performance compared to the diffusion models. However, we do not see any significant improvements compared to the baseline for both the noise addition and diffusion models. For this reason, it is no longer of interest whether the noise addition is comparable to the diffusion models since neither performs above the baseline.

There is an exception to the noise addition model being comparable to the diffusion models in the imbalanced Passive vs. Active listening task. In Figure 4.7, the noise addition model is the only model comparable to the baseline, with the diffusion models all performing worse. It is likely that the noise addition model caused the classifier to become slightly more biased towards the majority class Active listening, which would have had a minor effect on the accuracy due to the class imbalance. The diffusion models here instead seem to influence the classification of both classes similarly, causing a decrease in the overall accuracy. We can also observe the weighted average F1-score comparison in Figure 4.6 again shows comparable performance between the noise addition model and diffusion models, with none performing above the baseline.

Statistical Significance

As mentioned in Chapter 3, 20 models of each type presented in the Results were trained to enable a statistical comparison against the baseline. These are not a large number of samples, and it is possible some significant findings would change if more runs had been performed. In addition to this, we only used a pairwise independent Welch's t-test between the baseline and the augmented models, instead of using a more rigorous method.

In the case that the explored diffusion augmented models happen to not be significantly different from the baseline, this would nonetheless imply that the generated data is comparable to the original. This follows from the fact that the classifier must be classifying the generated synthetic data with at least the same accuracy as the original, otherwise a decrease in performance would occur. Therefore, the generated samples must contain features of the labels that are identifiable to the same degree as the original data. A more in-depth statistical analysis could be explored in future work.

6

Conclusion

Throughout this project, we have successfully implemented and evaluated the performance of multiple diffusion models in generating synthetic EEG data. The synthetic data, as verified by our selected metrics, demonstrates a high degree of similarity to real EEG data. This conclusion is supported by both the matching value distributions across each EEG channel, as quantified by Jensen-Shannon divergence, and visual comparisons with real EEG data.

Additionally, our findings suggest the potential of incorporating diffusion generated EEG data in data augmentation for LoA tasks. Particularly, augmenting our classifier model with an additional 100% synthetic EEG data generated by the spectral diffusion model led to a significant improvement in classification accuracy. Specifically, an average increase of **0.58%** was observed, improving the classification accuracy to approximately **71%** from a baseline of **70.4%** for the LoA task, based on 1-second EEG time-windows.

6.1 Connection to Project Objectives

As outlined in the Introduction, our thesis project aimed to achieve the following objectives:

1. Investigate the viability of generating EEG data using DPMs traditionally trained on image data.
2. Evaluate the quality of the synthetically generated EEG data through both visual inspection and objective metrics.
3. Quantify the benefits of using generated EEG data for data augmentation in three different classification tasks.

The first objective has been met through the successful implementation and training of multiple diffusion models for EEG data synthesis. A high degree of similarity between the generated synthetic EEG data and real EEG data was observed, as shown by our selected metrics. These outcomes not only underscore the potential of DPMs in synthesizing EEG data but also address our second objective by affirming the realistic qualities of the produced synthetic data.

While the application of diffusion-generated EEG data in the three classification tasks yielded mixed outcomes, the results from the balanced LoA task stand out. For the imbalanced task, no significant improvements were observed in average classification accuracy or F1-score, even with the implementation of oversampling to balance the class label split. However, the notable success in the LoA task demonstrates the value of diffusion-generated synthetic EEG data as a data augmentation tool, particularly in scenarios where class balance can be achieved.

The implications of these findings are especially relevant to the advancement of neuro-steered hearing aids. By effectively enhancing classification accuracy in the LoA task through the use of synthetic EEG data, we lay foundations for future developments in hearing aid technologies that employ AAD algorithms.

6.2 Future Work

Building upon our findings in this project, several areas for future work emerge. This section will go over a few areas we find particularly relevant and in need of further research. This could be within the area of AAD, aimed at improving future hearing aids, or any number of sub-fields within the study of EEG.

Different inner diffusion model

As outlined in our Delimitations, we chose not to implement an inner denoising model specifically tailored for multichannel time series data. Instead, we used a variant of the UNet neural network architecture, which is standard for most modern diffusion models working in the image domain. However, optimizing the inner model to effectively differentiate noise from the distinctive features of the training data, adopting it to excel with EEG data, or multichannel time series data in general, could lead to significant improvements in generation quality.

Conditional EEG-generation

Training our diffusion models separately for each different class label reduced the overall availability of training data for each model and required substantial processing power. An alternative approach would involve designing a conditional DPM, capable of incorporating additional information of interest. This information could be class labels, text, audio or any other relevant feature to condition the EEG generation process.

During the initial stages of the project, attempts to train a conditional model on our EEG data proved unsuccessful, leading us to abandon this in favor of the unconditional approach. However, we recognize this as a notable area for future improvements, particularly in terms of reducing processing costs when training models.

Continuous data generation

Given that our models were trained to specifically generate 1-second segments of EEG data, adapting them to tasks that requiring longer time windows poses a challenge. One potential solution could involve generating continuous data instead of discrete segments. Such an approach would expand the usability of synthesized EEG data across a broader range of tasks that operate on varying time scales. Furthermore, it could by design improve the temporal consistency of the generated data, as each new sample would be conditionally linked to the previous one.

Bibliography

Alickovic, E., T. Lunner, F. Gustafsson, and L. Ljung (2019). "A tutorial on auditory attention identification methods." *Frontiers in Neuroscience* **13**. ISSN: 1662-453X.

Alickovic, E., T. Lunner, D. Wendt, L. Fiedler, R. Hietkamp, E. H. N. Ng, and C. Graversen (2020). "Neural representation enhanced for speech and reduced for background noise with a hearing aid noise reduction scheme during a selective attention task". *Frontiers in Neuroscience* **14**. ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00846. URL: https://www.frontiersin.org/articles/10.3389/fnins.2020.00846.

Alickovic, E., E. H. N. Ng, L. Fiedler, S. Santurette, H. Innes-Brown, and C. Graversen (2021). "Effects of hearing aid noise reduction on early and late cortical representations of competing talkers in noise". *Frontiers in Neuroscience* **15**. ISSN: 1662-453X. DOI: 10.3389/fnins.2021.636060. URL: https://www.frontiersin.org/articles/10.3389/fnins.2021.636060.

Andersen, A. H., S. Santurette, M. S. Pedersen, E. Alickovic, L. Fiedler, J. Jensen, and T. Behrens (2021). "Creating clarity in noisy environments by using deep learning in hearing aids." *Seminars in Hearing* **42**:3, pp. 260–281. ISSN: 07340451.

Bond-Taylor, S., A. Leach, Y. Long, and C. G. Willcocks (2022). "Deep generative modelling: a comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**:11, pp. 7327–7347. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3116668.

Chen, X., X. Teng, H. Chen, Y. Pan, and P. Geyer (2024). "Toward reliable signals decoding for electroencephalogram: a benchmark study to EEGNeX". *Biomedical Signal Processing and Control* **87**, p. 105475. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2023.105475. URL: https://www.sciencedirect.com/science/article/pii/S1746809423009084.

Cherry, E. C. (1953). "Some experiments on the recognition of speech, with one and with two ears". *The Journal of the acoustical society of America* **25**:5, pp. 975–979.

Ciccarelli, G., M. Nolan, J. Perricone, P. Calamia, S. Haro, T. Quatieri, C. Smalt, J. O'Sullivan, and N. Mesgarani (2019). "Comparison of two-talker attention decoding from EEG with nonlinear neural networks and linear methods." *Scientific Reports* **9**:1. ISSN: 20452322.

Collette, A. (2014). *Hdf5 for python*. URL: `https://docs.h5py.org/en/stable/#` (visited on 2023-02-04).

Crosse, M. J., N. J. Zuk, G. M. Di Liberto, A. R. Nidiffer, S. Molholm, and E. C. Lalor (2021). "Linear modeling of neurophysiological responses to speech and other continuous stimuli: methodological considerations for applied research". *Frontiers in neuroscience* **15**, p. 705621.

Das, N., T. Francart, and A. Bertrand (2020). *Auditory attention detection dataset kuleuven*. Version 1.1.0. This research work was carried out at the ESAT and ExpORL Laboratories of KU Leuven, in the frame of KU Leuven Special Research Fund BOF/ STG-14-005, OT/14/119 and C14/16/057. The work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 637424). Zenodo. DOI: `10.5281/zenodo.3997352`. URL: `https://doi.org/10.5281/zenodo.3997352`.

Dhariwal, P., H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever (2020). *Jukebox: a generative model for music*. arXiv: `2005.00341 [eess.AS]`.

Dhariwal, P. and A. Nichol (2021). "Diffusion models beat gans on image synthesis."

Ding, N. and J. Z. Simon (2012). "Emergence of neural encoding of auditory objects while listening to competing speakers." *Proceedings of the National Academy of Sciences of the United States of America* **109**:29, pp. 11854–11859. ISSN: 00278424.

Geirnaert, S., S. Vandecappelle, E. Alickovic, A. de Cheveigne, E. Lalor, B. T. Meyer, S. Miran, T. Francart, and A. Bertrand (2021). "Electroencephalography-based auditory attention decoding: toward neurosteered hearing devices." *IEEE Signal Processing Magazine* **38**:4, pp. 89–102. ISSN: 10535888.

Gevins, A., M. E. Smith, L. K. McEvoy, H. Leong, and J. Le (1999). "Electroencephalographic imaging of higher brain function". *Philosophical Transactions: Biological Sciences* **354**:1387, pp. 1125–1133. ISSN: 09628436. URL: `http://www.jstor.org/stable/56995` (visited on 2024-02-09).

Gramfort, A., M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. S. Hämäläinen (2013). "MEG and EEG data analysis with MNE-Python". *Frontiers in Neuroscience* **7**:267, pp. 1–13. DOI: `10.3389/fnins.2013.00267`.

He, T., Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li (2018). *Bag of tricks for image classification with convolutional neural networks*. arXiv: `1812.01187 [cs.CV]`.

Ho, J., A. Jain, and P. Abbeel (2020). "Denoising diffusion probabilistic models". In: Larochelle, H. et al. (Eds.). *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 6840–6851. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Kingma, D. P. and M. Welling (2022). *Auto-encoding variational bayes*. arXiv: 1312.6114 [stat.ML].

Klem, G. H., H. Lüders, H. H. Jasper, and C. E. Elger (1999). "The ten-twenty electrode system of the international federation. the international federation of clinical neurophysiology." *Electroencephalography and clinical neurophysiology. Supplement* **52**, pp. 3–6.

Lawhern, V. J., A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance (2018). "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". *Journal of Neural Engineering* **15**:5, p. 056013. ISSN: 1741-2552. DOI: 10.1088/1741-2552/aace8c. URL: http://dx.doi.org/10.1088/1741-2552/aace8c.

Lindholm, A. (2022). *Machine learning. a first course for engineers and scientists.* Cambridge University Press. ISBN: 9781108843607.

Marrone, N., C. Mason, and J. Kidd G (2008). "Evaluating the benefit of hearing aids in solving the cocktail party problem." *Trends in Amplification* **12**:4, pp. 300–315. ISSN: 1084-7138.

Mesgarani, N. and E. F. Chang (2012). "Selective cortical representation of attended speaker in multi-talker speech perception." *Nature* **485**:7397, pp. 233–236. ISSN: 00280836.

Mirkovic, B., M. G. Bleichner, M. De Vos, and S. Debener (2016). "Target speaker detection with concealed EEG around the ear". *Frontiers in neuroscience* **10**, p. 349.

MrAlanKoh (2022). *Example of forward diffusion models*. This file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/. URL: https://commons.wikimedia.org/wiki/File:Example_of_Forward_Diffusion_Models.png.

Nichol, A. and P. Dhariwal (2021). "Improved denoising diffusion probabilistic models". *CoRR* **abs/2102.09672**. arXiv: 2102.09672. URL: https://arxiv.org/abs/2102.09672.

O'Sullivan, J., A. Power, E. Lalor, N. Mesgarani, S. Rajaram, B. Shinn-Cunningham, J. Foxe, M. Slaney, and S. Shamma (2015). "Attentional selection in a cocktail party environment can be decoded from single-trial eeg." *Cerebral Cortex* **25**:7, pp. 1697-1706 –1706. ISSN: 14602199.

O'sullivan, J. A., A. J. Power, N. Mesgarani, S. Rajaram, J. J. Foxe, B. G. Shinn-Cunningham, M. Slaney, S. A. Shamma, and E. C. Lalor (2015). "Attentional selection in a cocktail party environment can be decoded from single-trial EEG". *Cerebral cortex* **25**:7, pp. 1697–1706.

O'Sullivan, J., J. Herrero, E. Smith, C. Schevon, G. M. McKhann, S. A. Sheth, A. D. Mehta, and N. Mesgarani (2019). "Hierarchical encoding of attended auditory objects in multi-talker speech perception." *Neuron* **104**:6, pp. 1195–1209. ISSN: 0896-6273.

Puffay, C., B. Accou, L. Bollens, M. J. Monesi, J. Vanthornhout, H. V. hamme, and T. Francart (2023). "Relating EEG to continuous speech using deep neural networks: a review". *Journal of Neural Engineering* **20**:4, p. 041003. DOI: 10.1088/1741-2552/ace73f. URL: https://dx.doi.org/10.1088/1741-2552/ace73f.

Puvvada, K. and J. Simon (2017). "Cortical representations of speech in a multitalker auditory scene." *Journal of Neuroscience* **37**:38, pp. 9189-9196 –9196. ISSN: 15292401.

Ramesh, A., P. Dhariwal, A. Nichol, C. Chu, and M. Chen (2022). *Hierarchical text-conditional image generation with clip latents*. arXiv: 2204.06125 [cs.CV].

Ronneberger, O., P. Fischer, and T. Brox (2015). *U-net: convolutional networks for biomedical image segmentation*. arXiv: 1505.04597 [cs.CV].

Saharia, C., W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi (2022). *Photorealistic text-to-image diffusion models with deep language understanding*. arXiv: 2205.11487 [cs.CV].

Salimans, T. and J. Ho (2022). "Progressive distillation for fast sampling of diffusion models". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=TIdIXIpzhoI.

Silva, F. L. da (2010). "EEG: origin and measurement". In: Mulert, C. et al. (Eds.). *EEG - fMRI: Physiological Basis, Technique, and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 19–38. ISBN: 978-3-540-87919-0. DOI: 10.1007/978-3-540-87919-0_2. URL: https://doi.org/10.1007/978-3-540-87919-0_2.

Sohl-Dickstein, J., E. Weiss, N. Maheswaranathan, and S. Ganguli (2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR, pp. 2256–2265.

Somepalli, G., V. Singla, M. Goldblum, J. Geiping, and T. Goldstein (2023). "Diffusion art or digital forgery? investigating data replication in diffusion models". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6048–6058. DOI: 10.1109/CVPR52729.2023.00586.

Song, J., C. Meng, and S. Ermon (2022). *Denoising diffusion implicit models*. arXiv: 2010.02502 [cs.LG].

Steinmetz, C. J. and J. D. Reiss (2020). "Auraloss: Audio focused loss functions in PyTorch". In: *Digital Music Research Network One-day Workshop (DMRN+15)*.

Sun, C. and C. Mou (2023). "Survey on the research direction of EEG-based signal processing". *Frontiers in Neuroscience* **17**. ISSN: 1662-453X. DOI: 10.3389/fnins.2023.1203059. URL: https://www.frontiersin.org/articles/10.3389/fnins.2023.1203059.

Taillez, T., B. Kollmeier, and B. T. Meyer (2020). "Machine learning for decoding listeners' attention from electroencephalography evoked by continuous speech." *European Journal of Neuroscience* **51**:5, pp. 1234–1241. ISSN: 0953816X.

The HDF Group (2023). *The hdf5 library & file format*. URL: https://www.hdfgroup.org/solutions/hdf5/ (visited on 2023-02-04).

Vandecappelle, S., L. Deckers, N. Das, A. H. Ansari, A. Bertrand, and T. Francart (2021). "EEG-based detection of the locus of auditory attention with convolutional neural networks". *eLife* **10**. Ed. by B. G. Shinn-Cunningham, J. O'Sullivan, and A. Dimitrijevic, e56481. ISSN: 2050-084X. DOI: 10.7554/eLife.56481. URL: https://doi.org/10.7554/eLife.56481.

Wilroth, J., B. Bernhardsson, F. Heskebeck, M. A. Skoglund, C. Bergeling, and E. Alickovic (2023). "Improving EEG-based decoding of the locus of auditory attention through domain adaptation". *Journal of Neural Engineering* **20**:6, p. 066022.

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush (2020). *Huggingface's transformers: state-of-the-art natural language processing*. arXiv: 1910.03771 [cs.CL].

Yamamoto, R., E. Song, and J.-M. Kim (2020). *Parallel wavegan: a fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram*. arXiv: 1910.11480 [eess.AS].

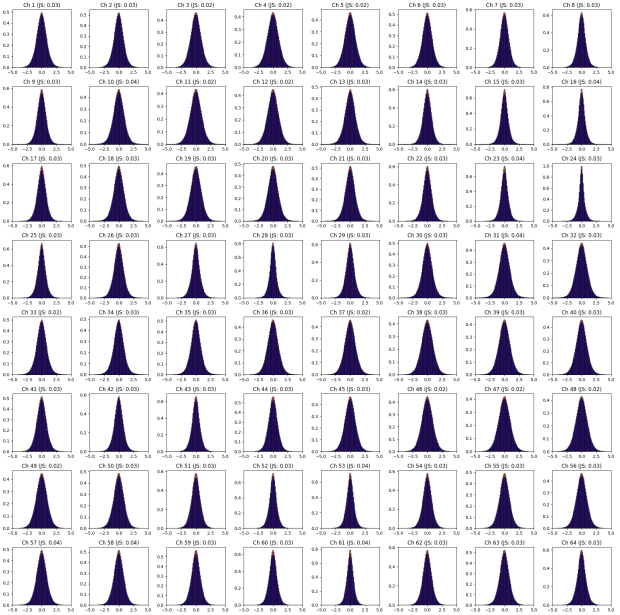
Appendices

Listing 6.1 Full list of layers for the used *diffusers'* UNet implementation when summarized with the python library `torchinfo`. An input with batch size of 1 is used here.

```
==========================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================
UNet2DModel                                             [1, 1, 64, 256]           --
├─Timesteps: 1-1                                        [1, 128]                  --
├─TimestepEmbedding: 1-2                                [1, 512]                  --
│    └─Linear: 2-1                                      [1, 512]                  66,048
│    └─SiLU: 2-2                                        [1, 512]                  --
│    └─Linear: 2-3                                      [1, 512]                  262,656
├─Conv2d: 1-3                                           [1, 128, 64, 256]         1,280
├─ModuleList: 1-4                                       --                        --
│    └─DownBlock2D: 2-4                                 [1, 128, 32, 128]         --
│    │    └─ModuleList: 3-1                             --                        --
│    │    │    └─ResnetBlock2D: 4-1                     [1, 128, 64, 256]         361,344
│    │    │    └─ResnetBlock2D: 4-2                     [1, 128, 64, 256]         361,344
│    │    └─ModuleList: 3-2                             --                        --
│    │    │    └─Downsample2D: 4-3                      [1, 128, 32, 128]         147,584
│    └─DownBlock2D: 2-5                                 [1, 128, 16, 64]          --
│    │    └─ModuleList: 3-3                             --                        --
│    │    │    └─ResnetBlock2D: 4-4                     [1, 128, 32, 128]         361,344
│    │    │    └─ResnetBlock2D: 4-5                     [1, 128, 32, 128]         361,344
│    │    └─ModuleList: 3-4                             --                        --
│    │    │    └─Downsample2D: 4-6                      [1, 128, 16, 64]          147,584
│    └─DownBlock2D: 2-6                                 [1, 256, 8, 32]           --
│    │    └─ModuleList: 3-5                             --                        --
│    │    │    └─ResnetBlock2D: 4-7                     [1, 256, 16, 64]          1,050,368
│    │    │    └─ResnetBlock2D: 4-8                     [1, 256, 16, 64]          1,312,512
│    │    └─ModuleList: 3-6                             --                        --
│    │    │    └─Downsample2D: 4-9                      [1, 256, 8, 32]           590,080
│    └─DownBlock2D: 2-7                                 [1, 256, 4, 16]           --
│    │    └─ModuleList: 3-7                             --                        --
│    │    │    └─ResnetBlock2D: 4-10                    [1, 256, 8, 32]           1,312,512
│    │    │    └─ResnetBlock2D: 4-11                    [1, 256, 8, 32]           1,312,512
│    │    └─ModuleList: 3-8                             --                        --
│    │    │    └─Downsample2D: 4-12                     [1, 256, 4, 16]           590,080
│    └─AttnDownBlock2D: 2-8                             [1, 512, 2, 8]            --
│    │    └─ModuleList: 3-11                            --                        (recursive)
│    │    │    └─ResnetBlock2D: 4-13                    [1, 512, 4, 16]           3,935,744
│    │    └─ModuleList: 3-12                            --                        (recursive)
│    │    │    └─Attention: 4-14                        [1, 512, 4, 16]           1,051,648
│    │    └─ModuleList: 3-11                            --                        (recursive)
│    │    │    └─ResnetBlock2D: 4-15                    [1, 512, 4, 16]           4,984,320
│    │    └─ModuleList: 3-12                            --                        (recursive)
│    │    │    └─Attention: 4-16                        [1, 512, 4, 16]           1,051,648
│    │    └─ModuleList: 3-13                            --                        --
│    │    │    └─Downsample2D: 4-17                     [1, 512, 2, 8]            2,359,808
│    └─DownBlock2D: 2-9                                 [1, 512, 2, 8]            --
│    │    └─ModuleList: 3-14                            --                        --
│    │    │    └─ResnetBlock2D: 4-18                    [1, 512, 2, 8]            4,984,320
│    │    │    └─ResnetBlock2D: 4-19                    [1, 512, 2, 8]            4,984,320
├─UNetMidBlock2D: 1-5                                   [1, 512, 2, 8]            --
│    └─ModuleList: 2-12                                 --                        (recursive)
│    │    └─ResnetBlock2D: 3-15                         [1, 512, 2, 8]            --
│    │    │    └─GroupNorm: 4-20                        [1, 512, 2, 8]            1,024
│    │    │    └─SiLU: 4-21                             [1, 512, 2, 8]            --
│    │    │    └─LoRACompatibleConv: 4-22               [1, 512, 2, 8]            2,359,808
│    │    │    └─SiLU: 4-23                             [1, 512]                  --
│    │    │    └─LoRACompatibleLinear: 4-24             [1, 512]                  262,656
│    │    │    └─GroupNorm: 4-25                        [1, 512, 2, 8]            1,024
│    │    │    └─SiLU: 4-26                             [1, 512, 2, 8]            --
│    │    │    └─Dropout: 4-27                          [1, 512, 2, 8]            --
│    │    │    └─LoRACompatibleConv: 4-28               [1, 512, 2, 8]            2,359,808
│    └─ModuleList: 2-11                                 --                        --
│    │    └─Attention: 3-16                             [1, 512, 2, 8]            --
│    │    │    └─GroupNorm: 4-29                        [1, 512, 16]              1,024
│    │    │    └─LoRACompatibleLinear: 4-30             [1, 16, 512]              262,656
│    │    │    └─LoRACompatibleLinear: 4-31             [1, 16, 512]              262,656
│    │    │    └─LoRACompatibleLinear: 4-32             [1, 16, 512]              262,656
│    │    │    └─ModuleList: 4-33                       --                        262,656
│    └─ModuleList: 2-12                                 --                        (recursive)
│    │    └─ResnetBlock2D: 3-17                         [1, 512, 2, 8]            --
│    │    │    └─GroupNorm: 4-34                        [1, 512, 2, 8]            1,024
│    │    │    └─SiLU: 4-35                             [1, 512, 2, 8]            --
│    │    │    └─LoRACompatibleConv: 4-36               [1, 512, 2, 8]            2,359,808
│    │    │    └─SiLU: 4-37                             [1, 512]                  --
│    │    │    └─LoRACompatibleLinear: 4-38             [1, 512]                  262,656
│    │    │    └─GroupNorm: 4-39                        [1, 512, 2, 8]            1,024
│    │    │    └─SiLU: 4-40                             [1, 512, 2, 8]            --
│    │    │    └─Dropout: 4-41                          [1, 512, 2, 8]            --
│    │    │    └─LoRACompatibleConv: 4-42               [1, 512, 2, 8]            2,359,808
├─ModuleList: 1-6                                       --                        --
│    └─UpBlock2D: 2-13                                  [1, 512, 4, 16]           --
│    │    └─ModuleList: 3-18                            --                        --
│    │    │    └─ResnetBlock2D: 4-43                    [1, 512, 2, 8]            7,869,440
│    │    │    └─ResnetBlock2D: 4-44                    [1, 512, 2, 8]            7,869,440
│    │    │    └─ResnetBlock2D: 4-45                    [1, 512, 2, 8]            7,869,440
│    │    └─ModuleList: 3-19                            --                        --
│    │    │    └─Upsample2D: 4-46                       [1, 512, 4, 16]           2,359,808
│    └─AttnUpBlock2D: 2-14                              [1, 512, 8, 32]           --
│    │    └─ModuleList: 3-24                            --                        (recursive)
│    │    │    └─ResnetBlock2D: 4-47                    [1, 512, 4, 16]           7,869,440
│    │    └─ModuleList: 3-25                            --                        (recursive)
```

```
│    │    │    └─Attention: 4—48                  [1, 512, 4, 16]           1,051,648
│    │    └─ModuleList: 3—24                      ──                        (recursive)
│    │    │    └─ResnetBlock2D: 4—49              [1, 512, 4, 16]           7,869,440
│    │    └─ModuleList: 3—25                      ──                        (recursive)
│    │    │    └─Attention: 4—50                  [1, 512, 4, 16]           1,051,648
│    │    └─ModuleList: 3—24                      ──                        (recursive)
│    │    │    └─ResnetBlock2D: 4—51              [1, 512, 4, 16]           6,558,208
│    │    └─ModuleList: 3—25                      ──                        (recursive)
│    │    │    └─Attention: 4—52                  [1, 512, 4, 16]           1,051,648
│    │    └─ModuleList: 3—26                      ──                        ──
│    │    │    └─Upsample2D: 4—53                 [1, 512, 8, 32]           2,359,808
│    └─UpBlock2D: 2—15                            [1, 256, 16, 64]          ──
│    │    └─ModuleList: 3—27                      ──                        ──
│    │    │    └─ResnetBlock2D: 4—54              [1, 256, 8, 32]           2,690,048
│    │    │    └─ResnetBlock2D: 4—55              [1, 256, 8, 32]           2,034,176
│    │    │    └─ResnetBlock2D: 4—56              [1, 256, 8, 32]           2,034,176
│    │    └─ModuleList: 3—28                      ──                        ──
│    │    │    └─Upsample2D: 4—57                 [1, 256, 16, 64]          590,080
│    └─UpBlock2D: 2—16                            [1, 256, 32, 128]         ──
│    │    └─ModuleList: 3—29                      ──                        ──
│    │    │    └─ResnetBlock2D: 4—58              [1, 256, 16, 64]          2,034,176
│    │    │    └─ResnetBlock2D: 4—59              [1, 256, 16, 64]          2,034,176
│    │    │    └─ResnetBlock2D: 4—60              [1, 256, 16, 64]          1,706,240
│    │    └─ModuleList: 3—30                      ──                        ──
│    │    │    └─Upsample2D: 4—61                 [1, 256, 32, 128]         590,080
│    └─UpBlock2D: 2—17                            [1, 128, 64, 256]         ──
│    │    └─ModuleList: 3—31                      ──                        ──
│    │    │    └─ResnetBlock2D: 4—62              [1, 128, 32, 128]         706,048
│    │    │    └─ResnetBlock2D: 4—63              [1, 128, 32, 128]         541,952
│    │    │    └─ResnetBlock2D: 4—64              [1, 128, 32, 128]         541,952
│    │    └─ModuleList: 3—32                      ──                        ──
│    │    │    └─Upsample2D: 4—65                 [1, 128, 64, 256]         147,584
│    └─UpBlock2D: 2—18                            [1, 128, 64, 256]         ──
│    │    └─ModuleList: 3—33                      ──                        ──
│    │    │    └─ResnetBlock2D: 4—66              [1, 128, 64, 256]         541,952
│    │    │    └─ResnetBlock2D: 4—67              [1, 128, 64, 256]         541,952
│    │    │    └─ResnetBlock2D: 4—68              [1, 128, 64, 256]         541,952
├─GroupNorm: 1—7                                  [1, 128, 64, 256]         256
├─SiLU: 1—8                                       [1, 128, 64, 256]         ──
├─Conv2d: 1—9                                     [1, 1, 64, 256]           1,153
==========================================================================================
```

(a) Left class



(b) Right class

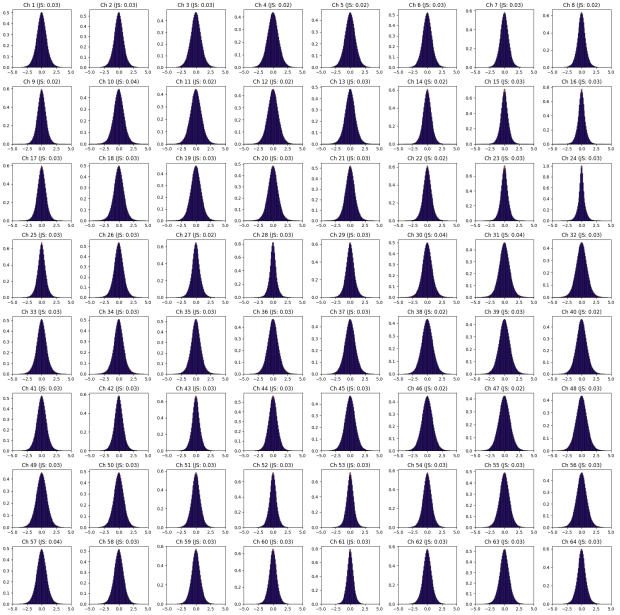


Figure 6.1 Epsilon Jensen-Shannon divergence graphs for the epsilon model

All implemented models have a large set of parameters to use. The tables show the parameters that have been changed, the others are the standard parameters set by the implementation in the *diffusers'* library.

Hyperparameter	Value
Batch size	64
Learning rate	5e-4
Learning rate scheduler	Constant
Overlap percent	0.25
Normalize with train	True

Table 6.1 Standard parameters used for the classification tasks

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
March 2024

Document Number
TFRT-6227

Author(s)
David Rannaleet
Victor Gunnarsson

Supervisor
Emina Alickovic, Eriksholm Research Centre
Martin Skoglund, Eriksholm Research Centre
Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden
Pontus Giselsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Diffusion Modelling approaches to EEG-based Auditory Attention Decoding

Abstract

Machine learning models can analyze physiological data, such as electroencephalography (EEG), for various classification tasks. One such task is Auditory Attention Decoding (AAD), aimed at identifying the sound a person is actively attending to, offering significant benefits for users of hearing aids. However, EEG data often exhibits a low signal-to-noise ratio, and its collection is often expensive, cumbersome and requires trained specialists. Additionally, gathering EEG data over prolonged periods of time presents challenges. The resulting scarcity of available EEG data to train models can be alleviated by using generative models, which generate new examples from the data they were trained on.

Diffusion Probabilistic Models (DPMs) have in recent years emerged as the state-of-the-art of generative models within the image domain, showing widespread success in models such as Stable Diffusion and DALL-E. This work investigates whether this success can extend to the domain of multichannel time series data, specifically EEG data. The diffusion models were trained on 1-second EEG data segments and were used as a data augmentation tool for 3 different classification tasks, including AAD. Our findings indicate that diffusion models can effectively generate realistic EEG data, supported by both a visual comparison and a measure of Jensen-Shannon divergence to the real EEG data distribution. In addition to this, a significant improvement in mean performance was achieved in our Locus of Attention (LoA) task, where we classify between a test subject attending to a left or right speaker. Here an approximate classification accuracy of 71% was achieved compared to our baseline of 70.4%.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-42

Recipient's notes

Security classification

http://www.control.lth.se/publications/