Deep convolution neural network for attention decoding in multi-channel EEG with conditional variational autoencoder for data augmentation

M. Asjid Tanveer

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6194
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 2023

Abstract

Objectives: This project aims to develop a deep learning-based attention decoding system that can distinguish between noise and speech in noise and also identify the direction of attended speech from the brain data recorded with electroencephalography (EEG) instruments. Two deep convolutional neural network (DCNN) models will be designed: (1) one DCNN model capable of classifying incoming segments of sound as speech or speech in background noise, and (2) one DCNN model identifying the direction (left vs. right) of incoming attended speech. In addition, two conditional variational autoencoders (CVAEs) will be trained to generate artificial data for data augmentation, with the goal of improving the performance of the final models by learning from a latent space of training data to generate unique data for each respective class.

Design: The proposed methods will be tested on a data set of 32 participants who performed an auditory attention task. Participants were instructed to attend to one of two talkers in the front and ignore the talker on the other side and background noise behind them, while high-density EEG was recorded. The EEG data consists of 66 channels in total, and all channels will be used in this study.

Main Results: The DCNN models achieved accuracy (ACC) of 69.9%, 84.9%, and area under the curve (AUC) scores of 77.5%, 92.3% on the two tasks mentioned in the objectives. With augmented data from the CVAE model, the performance of the DCNN models improved to ACC of 70.5%, 86.6% and AUC of 78.3%, 93.6%, respectively. The time window used for the EEG data was 1 second, enabling the models to work in real-time situations. The CVAE model was able to generate data for the given classes effectively, with generated data from the test data latent space showing promising results.

Conclusion: The findings of this study demonstrate the high capability of the proposed DCNN models in accurately detecting the direction of incoming speech and differentiating between noise and speech-in-noise, even with a small time window of just 1 second using multi-channel EEG data. Moreover, the results highlight the success of the CVAE model as a valuable tool for data augmentation, generating synthetic data that closely approximates the latent space information of the training

data. This suggests the potential of CVAE for improving the performance of deep learning models in EEG-based attention decoding tasks.

Acknowledgements

Acknowledgements are an important part of any thesis, as they provide the author with an opportunity to express their gratitude to those who have supported and contributed to their work.

With this in mind, I would like to take this opportunity to express my heartfelt thanks to everyone who has made my master's thesis possible. First and foremost, I would like to thank my thesis advisors Emina Alickovic and Martin Skoglund, for their unwavering support, guidance, and expertise throughout the entire process.

I am also grateful to my university supervisor Bo Bernhardsson, whose teachings and resources have been invaluable to me during my graduate studies.

Additionally, I would like to acknowledge the support of my family, who have always been there for me, providing me with the encouragement and motivation I needed to pursue my academic goals.

Finally, I would like to thank my friends and colleagues, who have supported me with their kind words and encouragement throughout this journey.

Thank you all for your support and contributions.

Contents

1

Introduction

The human brain is a complex and sophisticated structure, capable of performing a vast array of tasks including perception, reasoning, memory, and motor control. The brain's ability to solve complex problems is a result of its intricate network of neurons that process and transmit information. Electroencephalography (EEG) signals, which measure the electrical activity generated by neurons in the brain, can provide valuable insights into the brain's problem-solving processes.

In recent years, the field of deep learning has made remarkable progress in developing algorithms that can mimic the brain's ability to process and analyze information. Deep learning models, such as artificial neural networks, are capable of analyzing large amounts of data and making predictions or decisions based on that data. In the field of EEG signal processing, deep learning models have been used to analyze brain activity patterns to classify different types of mental tasks and states, such as attention, memory recall, and sleep. These models can also be used to predict future EEG signals, which can provide a non-invasive way to study the brain's activity and its relationship to various mental and physiological processes.

In normal conditions when engaging in speech with individuals, in a somewhat noisy environment our brain has the capacity to inhibit outside noises and focus on the intended speech. For instance in a restaurant setting, when surrounded by noise from other individuals we automatically tend to speak louder in an attempt to maintain the effective signal-to-noise ratio, typically causing what is known as the "Lombard effect" where all individuals at the restaurant attempt to do the same [Peelle and Wingfield, 2022], however fortunately the mammalian brain has evolved to extract important signals of information despite the noisy environment. Such a task is simple for individuals with normal hearing, however for individuals with hearing impairment it is a daunting task [Utoomprurkporn et al., 2020; Löhler et al., 2019]. To counter this problem we intend to work on creating deep learning based neural network models capable of detecting speech from background noise, to help individuals pay attention to intended speech [Soroush et al., 2021; Dash et al., 2020].

Deep learning is a subfield of machine learning that has been widely used for various EEG tasks in recent years. EEG is a non-invasive technique for measuring

the electrical activity of the brain, and it has been widely used in fields such as neuroscience, clinical neuroscience, and brain-computer interfaces (BCIs) [Craik et al., 2019].

Deep learning algorithms, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Autoencoders, have been used to address various EEG tasks [Zeng et al., 2018; Dose et al., 2018; Dai et al., 2019]. Some of these tasks are seen as follows:

- EEG-based classification: Deep learning algorithms have been used to classify EEG signals into different classes, such as sleep stages, seizure detection, or movement intention.
- EEG-based decoding: Deep learning algorithms have been used to decode EEG signals into continuous variables, such as arm movements, speech signals, or facial expressions.
- EEG-based feature extraction: Deep learning algorithms have been used to extract meaningful features from EEG signals, which can be used as input to other EEG-based tasks, such as classification or decoding.
- EEG-based representation learning: Deep learning algorithms have been used to learn low-dimensional representations of EEG signals, which can capture the underlying structure and patterns of EEG signals.

Overall, deep learning has shown promising results for EEG tasks and has the potential to significantly advance our understanding of the brain and its functions. However, more research is needed to fully realize the potential of deep learning for EEG tasks and to address challenges such as small sample size, high variability, and limited interpretability of deep learning models. [Roy et al., 2019] produced a review of 154 such research articles focusing on application of deep learning on EEG tasks has mentioned these advantages, and also recommended the use of methods to increase the data available for deep learning models to fully take advantage of their capacity to learn from larger data sets.

This thesis will investigate the use of deep learning architectures to detect and distinguish between background noise and intended speech for individuals with hearing impairment with data augmentation for improving the performance of aforementioned deep learning architectures. The architecture used in the project is based on a Deep Convolution Neural Network (DCNN) for classification task. Oticon A/S has provided a data set containing sound-evoked EEG measurements and corresponding audio recordings to train the neural network. The data set was collected from intended speakers in a noisy environment, the details of this data set will be discussed in the subsequent paragraph. As mentioned earlier, deep learning models require extensive data sets to be trained effectively. Therefore this thesis will work on data augmentation techniques to augment the data set with artificially generated data to further enhance the DCNN's performance.

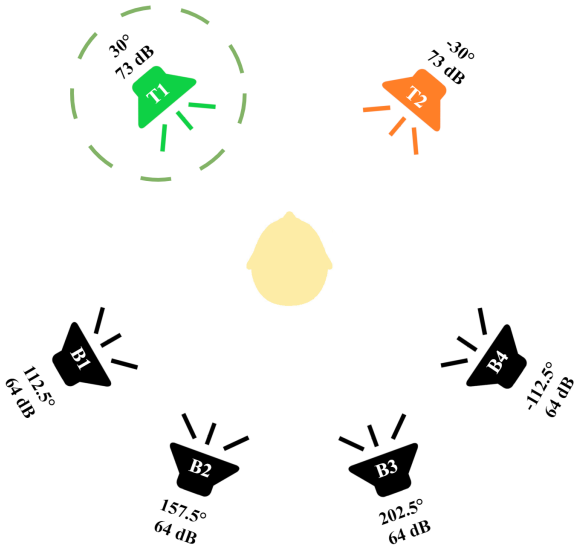


Figure 1.1 Visualisation of the experimental setup used for gathering data, based on [Alickovic et al., 2021]. The subject seated in the center with four background babble noise speakers, is told to be attentive to incoming speech from one of the forward speakers.

The data set comprises measurements obtained from 32 participants tested under carefully controlled experimental conditions that emulated a cocktail-party scenario. The participants had electrodes attached to their scalp and were situated in a room with four loudspeakers positioned equidistantly in the back and two loudspeakers in front, separated by an angle of $\pm 30^{\circ}$. The frontal loudspeakers played distinct news clips, one spoken by a male and the other by a female with similar speech patterns, while the back loudspeakers played the background noise. The participants were instructed to focus on a specific front loudspeaker, denoted as T1 in Figure 1.1, and EEG measurements were recorded.

The Deep learning model used in this study will be inspired from previous architectures used for such tasks (i.e., tasks oriented towards single/multi-channel EEG data). The task will be essentially to classify between noise and intended speech, comparing the performance to previous such methods with a hope to improve on previous performance. In addition, the thesis will aim to use small time windows instead of large time windows for this task, intended for better use in real applications. The model will be trained and tested on both non-augmented and augmented data sets to see performance improvement (if any).

Data augmentation for this thesis will employ the use of Conditional Variational AutoEncoders (CVAEs) due to their stability and success for augmenting EEG data [Luo et al., 2020; Bethge et al., 2022]. Since DCNNs require increasingly large data

set, **and** data that is a good representative of the original data distribution, CVAE is a strong pick due to its capacity to generate data from the same distribution as the original data set **while** creating new data distinct from the original training data.

The following sections in this thesis will discuss the topics mentioned above in details: **Chapter 2** will discuss how sound and EEG are measured, in addition to how sound and EEG are perceived by the human brain. **Chapter 3** will discuss data collection and pre-processing applied to the data set such as filtering to remove unwanted information. **Chapter 4** will discuss the Deep learning architecture, the parameters and optimizer used along with the theory and mathematical reasoning for each. **Chapter 5** will discuss Conditional Variational AutoEncoders (CVAEs), the model architecture along with parameters and how its implemented. **Chapter 6 and 7** will discuss the results, showcasing the performance achieved by the DCNN along with the performance achieved by the CVAE, in addition to performance achieved by the DCNN after data augmentation. **Chapter 8** will providing concluding remarks for this thesis.

2

Auditory attention decoding

The challenge of focusing on a single sound stream of interest while suppressing unwanted sounds in noisy environments is referred to as the cocktail party problem [Cherry, 1953]. The problem of decoding (i.e., identifying) a specific sound source of interest in a complex auditory environment from brain activity recorded with EEG is called EEG-based auditory attention decoding (AAD) [O'sullivan et al., 2015]. A multitude of EEG-based AAD algorithms, which linearly map the sound stimuli to the EEG, have been proposed in recent years to decode attention in multi-talker environments (see e.g., [O'sullivan et al., 2015; Wong et al., 2018; Cheveigné et al., 2018; Alickovic et al., 2019; Geirnaert et al., 2021]). However, the non-linear nature of the brain adds a further layer of complexity to the problem. Given the non-trivial nature of AAD, it is beneficial to gain a more comprehensive understanding of the underlying brain mechanisms by exploiting more complex non-linear models.

2.1 Auditory attention theory

The brain's ability to process sound can be broken down into four distinct stages in the auditory domain [Aroudi et al., 2016]. The first stage involves detecting the sound, followed by intentional and attentional hearing in the second stage. The third stage focuses on extracting meaning and information from the sound, while the fourth stage involves communication, which refers to the interactive and bidirectional exchange of meaning and information.

For the cocktail party problem, the most significant processes are intentional and attentional hearing, as well as the extraction of meaning and information. Objective assessment of these processes is crucial, particularly in measuring attention and listening effort when listening to speech in noisy environments.

The primary system utilized by the brain in sound processing is known as the auditory cortex, which is located within the temporal lobe. When sound enters the ear, it is processed to identify individual words, primarily occurring in the left temporal lobe. The processed phonemes, which represent the smallest units that differentiate between words, are continually compared against all known words until only one

possibility remains. The superior and middle temporal lobes then gather the lexical information associated with the chosen word, resulting in a comprehension of the spoken word.

In [Brodbeck et al., 2018] the authors conducted a study on how the brain processes sound, which resulted in a general structure of speech perception. The process involves identifying the audio stream, transforming it into linguistic information, and utilizing that information to access abstract word representation. The study indicates that the various stages of speech perception are primarily localized in the superior and middle temporal lobes, specifically around the auditory cortex. The response latency relative to the phonetic onset was approximately 110-120 ms, indicating that phonetic information is almost immediately used for lexical processing.

The process of identifying a spoken word involves all candidate words competing for recognition until only one remains. For instance, if the sequence "no" is spoken, both "noble" and "notable" are candidates for recognition. When the next part of the sequence, such as "b," is added, "notable" is no longer a valid candidate and thus discarded. Not all plausible candidates are treated equally, as word frequency plays a role in how favorable the word is compared to other candidates. The results indicate that the left temporal lobe mainly processes word recognition from audio, and the study also analyzed the results in a cocktail party problem scenario. The findings suggest that for the attended sound, the results align with previous research, while for the unattended sound, no processing is done to recognize the words in the audio. This finding suggests that only one speech stream can be correctly processed at a time [Brodbeck et al., 2018].

3

Data set

High-quality training data is essential to guarantee optimal performance of neural networks. The data set used in this thesis has been provided by Oticon A/S and has been previously published using a different analysis approach in [Alickovic et al., 2021]. The ultimate goal of this project is to train neural networks that may then be used in hearing aids, where the attended sound can be detected in online fashion with a minimal (if any) noticeable delay. To ensure effectiveness, a training data should reflect real-world scenarios and encompass diverse dynamics. Additionally, the neural network should be able to function effectively with the other features available in hearing aids.

3.1 Data acquisition

The experiment involved 32 native Danish speakers (24 males) aged from 21 to 84 years, with a mean age of 64.2 years and standard deviation of 13.6 years. All participant were experienced hearing aid users and had normal or corrected-to-normal vision, with no history of neurological disorders, dyslexia, or diabetes mellitus. The participants had mild to moderately severe symmetrical sensorineural hearing loss, with an average of 47.5 dB hearing loss on a 4-frequency pure-tone audiometry test. All participants were fitted with two identical hearing aids, (Oticon Opn S 1^{TM} mini-Receiver-in-the-ear), which had implemented noise reduction algorithms proposed in [Alickovic et al., 2021; Andersen et al., 2021].

Along with the hearing aids, the participants were equipped with electrodes to measure their EEG using a BioSemi Active Two recording system (Amsterdam, Netherlands). The recording system had a sampling frequency of 1024 Hz, and a total of 64 active electrodes were placed on the scalp according to the international 10-20 system. Two additional electrodes were included, namely an active electrode for common mode sensing and a passive electrode for driven right leg, to function as reference electrodes [Alickovic et al., 2021]. Additionally, two more electrodes were placed over the mastoids. To ensure optimal measurement quality, the elec-

trodes were adjusted by adding more conductive gel until the absolute voltage of the electrode was below 50 mV.

After the necessary preparations, the participants were seated in a chair in the centre of a sound proof room with six loud speakers positioned at $\pm 30°, \pm 112.5°,$ and $\pm$ 157.5°. This setup can be seen in Figure 1.1. During the test, news clips of neutral content read by a male and female talker were played through two front-facing loudspeakers (T1 and T2), while four other loudspeakers (B1-B4) positioned behind the participant played background babble noise from 16 talkers in order to increase task complexity. Long audio gaps were reduced to 200 ms, and recordings were normalized to the same root mean squared intensity. Speech stimuli were played at 73 dB sound pressure level (SPL) and 4-talker babbles were played at 64 dB SPL each, resulting in a total background noise level of 70 dB SPL.

A total of 84 trials were conducted, with 4 trials used to familiarize participants with the task and the remaining 80 trials used for testing. At the end of each trial the participant was asked a two-choice question related to the content of the attended speech. The study involved 80 trials divided into four blocks lasting approximately 20 minutes each. The blocks were conducted under different listening conditions: noise reduction algorithm 1 OFF, noise reduction algorithm 1 ON, noise reduction algorithm 2 OFF, and noise reduction algorithm 2 ON. The trial design can be seen in Figure 3.1.

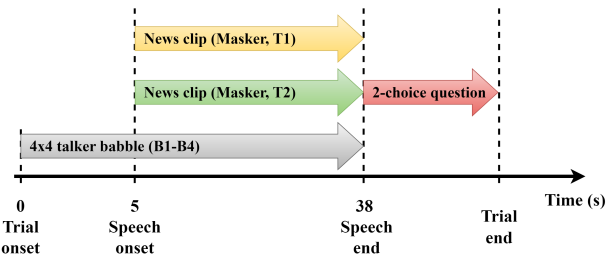


Figure 3.1 Experiment procedure, based on [Alickovic et al., 2021] with a total of 84 trials for each subject.

3.2 Data pre-processing

As EEG data being extracted measures all the brain activity, it is important to pre-process the data to remove any unnecessary artifacts and noise.

The EEG signals were segmented from -15 to 58 seconds with respect to the onset of the target and masker talker. Additionally, ten seconds of EEG signals before and after any stimuli were used as buffer zones to filter out edge artifacts. The average of the two reference mastoid channels was used to reference the EEG signals.

A digital bandpass filter was applied between 0.5-70 Hz, and an additional filter between 49-51 Hz was applied to eliminate power line noise. The filters were applied forward and backward using the filtfilt function in MATLAB to avoid phase shifts or delays. The EEG signals were downsampled to 256 Hz to reduce processing time, and visually corrupted channels (average of 2.2 channels removed, SD = 2.3) were removed. The nearest neighbor method in the Fieldtrip toolbox was used to interpolate the data from the surrounding clean EEG channels to replace the removed channels. Artifacts in the signals resulting from heartbeats, blinking, single-channel noise, etc., were removed with a denoising and independent component analysis, and components related to unwanted artifacts were manually removed. On average, 14.6 (SD = 4.6) components were removed, and one participant with excessively noisy data was excluded from further study. One block for one participant was also excluded due to technical problems [Alickovic et al., 2021].

In addition to the data preprocessing as mentioned earlier, we need to preprocess the data in a format that is ideal for deep learning architectures. For this purpose we sample the data into one second 256 samples time windows. The data available to us essentially consists of three classes: (1) background babble noise, (2) attend to the speaker on the right (3) attend to the speaker on the left. For the two latter cases (2 and 3) the 33 seconds of data as shown in Figure 3.1 is sampled into **non-overlapping** one second windows. However, due to the limited amount of data available for background babble noise, we use **overlapping** windows to extend the samples available for this class. Overlapping is a method used for data augmentation, although many do not explicitly consider this data augmentation. The authors in [O'Shea et al., 2017; Ullah et al., 2018] explicitly use overlapping windows as a data augmentation technique, while [Kwak et al., 2017] compared different overlapping windows showing that using a greater overlap (or smaller shift) resulting in more samples improved overall performance. The sampling methodology is shown in Figure 3.2 for one channel including only the background babble and speech duration. As shown, the data for background babble uses overlapping windows (with

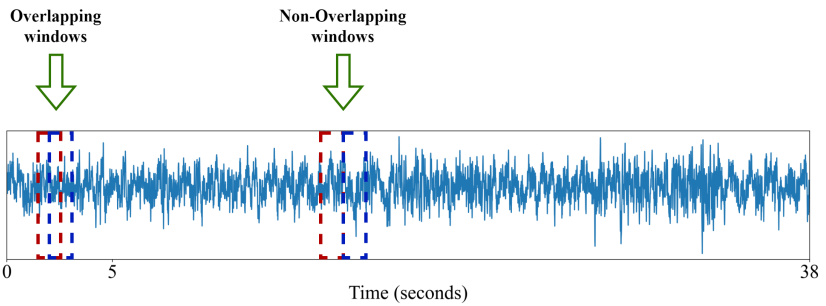


Figure 3.2 Sampling of EEG data for deep learning model.

0.875 seconds overlap) while the data for speech (left or right) uses non-overlapping windows.

Data normalization is important for deep learning as it improves neural network performance and stability during training. Deep learning architectures tend to perform better when data is normalized within a certain range, usually (0 to 1) or (-1 to 1) values. Normalization is the process of transforming the input data so that it has a standard scale and range. This process makes the data more uniform and easier for the neural network to learn from. In other words, normalization ensures that the features are on the same scale and have similar variances, which helps the model to converge faster and to avoid getting stuck in local optima. Moreover, normalization also prevents exploding or vanishing gradients, a common issue when training deep neural networks. Unnormalized input features can cause some features to dominate, leading to weight and bias instability, which may slow down or prevent the model from converging. For EEG data the range (-1 to 1) is used for data normalization. The following equation, gives a simple way to normalize a one second time window of EEG sample within this range.

$$x_{\text{normalized}} = 2\frac{\mathbf{x} - \min(x)}{\max(x) - \min(x)} - 1, \tag{3.1}$$

where **x** refers to the one second time window extracted from the EEG signal, including all channels. Hence each value in the sample is normalized using the maximum and minimum value across all *channels* within each sample.

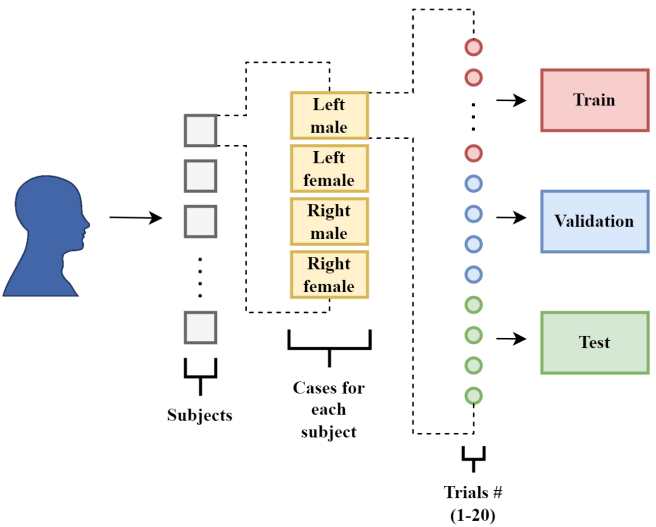


Figure 3.3 Data split for DCNN training incorporating all (four) cases across each subject, with 20 trials across each case.

Deep learning usually makes use of three training splits (i.e., training, validation and test sets). In Figure 3.3, we illustrate how the data was split. Each subject within our data consists of four cases: (1) left male speaker, (2) left female speaker, (3) right male speaker and (4) right female speaker, with each case having 20 trials in total. The trials from each case are divided into the three respective training-validation-test splits, with trial # 1-12 used as training data, trial # 13-16 used as validation data and trial # 17-20 used as testing data. After the data is partitioned, it is pre-processed and then sampled into one second time windows.

4

Deep convolution neural network

The brain's highly non-linear structure suggests that non-linear algorithms are necessary to address problems pertaining to the brain. Therefore, utilizing deep convolutional neural networks (DCNNs) seems like a reasonable approach to tackle these problems. For this purpose, we propose the use of DCNN for identifying which direction attended speech is coming from, and additionally identifying whether the perceived sound is background noise or attended speech. EEGNet [Lawhern et al., 2018] has proven to be an effective model for EEG related tasks. In this thesis, a model inspired by EEGNet will be worked on, albeit with some modifications. Some previous work [Blinowska and Malinowski, 1991; Acharya et al., 2005; Hernández et al., 1995] has shown the effectiveness of non-linear models in EEG related tasks, particularly in determining the locus of attention. In the forthcoming chapter, the theoretical foundations of DCNNs, the preferred loss function, and optimizer for this thesis will be outlined, followed by a discussion of the network's implementation.

4.1 Theory

DCNNs have gained significant attention and popularity in the field of computer vision due to their ability to extract useful features from raw image data [Krizhevsky et al., 2017]. DCNNs are composed of multiple layers, each of which consists of a set of learnable filters that perform convolutions on the input data. These filters are designed to detect various visual features in the image, such as edges, corners, and textures [LeCun et al., 2015]. The success of DCNNs in computer vision tasks can be attributed to their ability to learn complex features hierarchically, starting from simple features at lower layers and gradually combining them into more abstract representations at higher layers [Simonyan and Zisserman, 2014]. One of the key challenges in training DCNNs is the problem of vanishing gradients, which occurs when the gradients become very small as they propagate backwards through

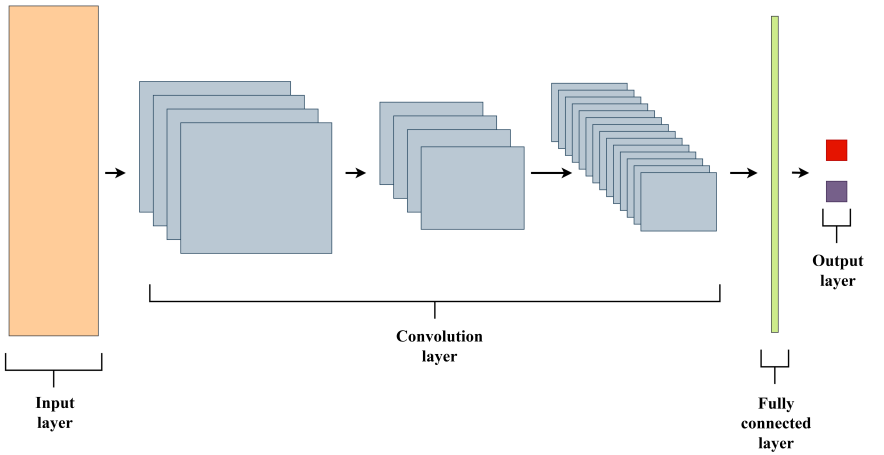


Figure 4.1 Deep convolution neural network

the network. This can be addressed using techniques such as weight initialization, batch normalization, and skip connections [He et al., 2016]. DCNNs have achieved state-of-the-art results in a wide range of computer vision tasks, including image classification, object detection, and semantic segmentation [Long et al., 2015].

Figure 4.1 illustrates a common model architecture for DCNNs, which includes an input layer followed by convolution layers designed to extract features from the input data. The convolution layers may be accompanied by batch normalization, dropout, pooling layers, and activation functions. After the feature extraction layers, the model includes fully connected layers which can also be combined with the aforementioned layers. The model concludes with an output layer that produces the logits for various classes.

Convolution layers consist of kernels that perform convolution operation on the input data with given kernel dimensions. The following equation outlines this function:

$$y_{i,j} = \sigma\left(\sum_{m=1}^{M}\sum_{n=1}^{N} w_{m,n}x_{i+m-1,j+n-1} + b\right), \tag{4.1}$$

where $y_{i,j}$ is the output activation at position (i,j), $\sigma(\cdot)$ is the activation function, $w_{m,n}$ is the weight at position (m,n) of the filter and (M,N) describe the number of weights, $x_{i+m-1,j+n-1}$ is the input activation at position $(i+m-1, j+n-1)$ where (i,j) are the input position of the data, and b is the bias term.

Additionally, we have batch normalization layers typically following the convolution layers. The basic idea of batch normalization is to normalize the inputs of a layer in a way that produces data with zero mean and unit variance, which can help

to stabilize the training process and improve the overall performance of the model. Specifically, batch normalization applies the following transformation to the input:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}, \tag{4.2}$$

where x is the input, μ is the mean of the batch, σ^2 is the variance of the batch, and ε is a small constant added for numerical stability. The transformed input $\hat{x}$ is then scaled and shifted by learnable parameters γ and β, respectively:

$$y = \gamma\hat{x} + \beta, \tag{4.3}$$

where y is the output of the batch normalization layer. By normalizing the activations of a layer, batch normalization can help to alleviate the problem of internal covariate shift, which is the tendency of the distribution of activations to shift during training as the parameters of the network are updated. This can help to speed up the training process and improve the generalization performance of the model.

Dropout layers are also introduced into the training model, by randomly dropping out some of the units in a layer. Dropout can help to prevent overfitting and improve the generalization performance of the model. It can also help to encourage the network to learn more robust and transferable features. Specifically, dropout applies the following transformation to the input:

$$\tilde{x} = x \cdot m, \tag{4.4}$$

where x is the input, and m is a binary mask with values drawn from a Bernoulli distribution with probability p of being 1. For example during training, the value of p can be set to 0.5, which means that each unit has a 50% chance of being dropped out. The transformed input $\tilde{x}$ is then scaled by a factor of $\frac{1}{1-p}$ during training, and left unchanged during inference. This is done to ensure that the expected value of the output is the same during training and inference.

Additionally, deep learning models contain pooling layers used to reduce the spatial dimensionality of feature maps produced by convolutional layers, while retaining important information and preserving spatial invariance. The most common types of pooling layers are max pooling and average pooling. Max pooling takes the maximum value within a sliding window and uses it as the output value for that region. Average pooling takes the average value within a sliding window and uses it as the output value. The following equation shows how pooling works:

$$y(i,j,k) = \frac{1}{st}\sum x(i+s, j+t, k), \tag{4.5}$$

where s and t are the size of the sliding window. Following the pooling layers the

models typically will have activation functions. Activation functions are a crucial part of deep learning models as they introduce non-linearity into the network, allowing it to model complex relationships between input and output data. There are several commonly used activation functions, including ReLU (Rectified Linear Unit) and tanh (hyperbolic tangent). The ReLU activation function along with its variant Leaky ReLU is used in the original paper for EEGNet [Lawhern et al., 2018], however in this study a new activation function known as Mish [Misra, 2019] was tested. The Mish activation function is a smooth and continuous function that is mathematically defined as:

$$\text{Mish}(x) = x \cdot \tanh(\text{softplus}(x)), \tag{4.6}$$

where $\text{softplus}(x) = \log(1 + e^x)$ is a smooth approximation of the rectified linear unit (ReLU) function. The Mish function is differentiable and has a derivative given by:

$$\text{Mish}'(x) = \tanh(\text{softplus}(x)) + x \cdot \text{sech}^2(\text{softplus}(x)) \cdot \frac{e^x}{1 + e^x}, \tag{4.7}$$

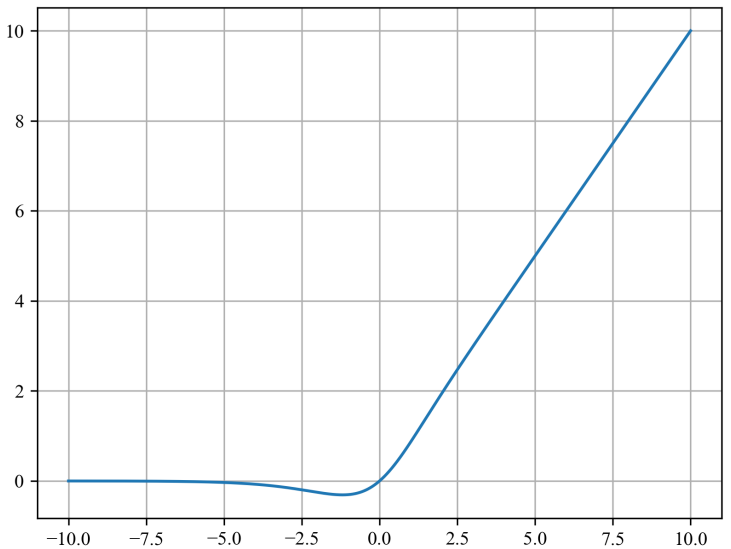


Figure 4.2 Mish activation function used in DCNN model architecture.

where $\text{sech}(x) = \frac{1}{\cosh(x)}$ is the hyperbolic secant function. The Mish activation function is shown in Figure 4.2, as we can see the activation function allows gradients to flow in the negative direction as well. This is in contrast to some other activation functions such as ReLU, which can suffer from the "dying ReLU" problem, where the gradient becomes zero for negative inputs, effectively killing the neuron and preventing further learning.

Compared to other commonly used activation functions such as ReLU, sigmoid, and tanh, Mish has been reported to offer several advantages. One major advantage of Mish is its smoothness, which makes it easier to optimize using gradient-based optimization algorithms. In addition, it has been shown to provide better performance in image classification tasks and object detection tasks, as well as in language modeling tasks.

Cross entropy loss

Cross entropy loss is a commonly used loss function in machine learning, particularly in classification tasks. It measures the difference between the predicted probability distribution and the true probability distribution of the classes. The cross entropy loss function is defined as:

$$\mathscr{L}\wr\int\int = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}), \tag{4.8}$$

where N is the number of samples, C is the number of classes, y_{ij} is the binary indicator (0 or 1) for whether sample i belongs to class j, and p_{ij} is the predicted probability of sample i belonging to class j.

The loss is calculated by taking the negative logarithm of the predicted probability of the correct class for each sample, and then averaging over all samples. The use of the logarithm function ensures that the loss penalizes predictions that are both confident and wrong more heavily than those that are less confident.

In practice, the cross entropy loss function is commonly used in combination with a softmax activation function on the final layer of a neural network, which ensures that the predicted probabilities sum to one. The predicted probabilities are then compared to the true labels using the cross entropy loss function. In case of Pytorch this is done automatically by the loss function, as such the final layer of the model **does not** use softmax activation function.

Adam optimizer

For this study we preferred the use of Adam optimizer ***with weight decay*** due to its popularity and typically high performance. The Adam optimizer with weight decay is a variant of the standard Adam optimizer that includes a weight decay term in the update rule. Weight decay is a regularization technique that penalizes large weights

in the model, which can help prevent overfitting. The update rule for Adam with weight decay is given by:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon}\hat{m}_t - \lambda\,\eta\,\theta_t, \tag{4.9}$$

where λ is the weight decay coefficient, which controls the strength of the regularization.

The weight decay term penalizes large weights by adding a small proportional negative value to the update for each weight. This encourages the weights to stay small and close to zero, which can help prevent overfitting and improve generalization.

To implement Adam with weight decay in practice, we simply add the weight decay term to the standard Adam update rule. The hyperparameter λ can be chosen by cross-validation or by using a rule of thumb such as weight decay coefficients in the range of 0.0001 to 0.1. The following algorithm displays how Adam optimizer is implemented with weight decay:

Algorithm 1 Adam Optimization Algorithm with Weight Decay

Initialize parameters θ
Initialize first and second moment variables, m_0 and v_0
Initialize hyperparameters: η (learning rate), β_1, β_2, ε, λ (weight decay coefficient)
for $t = 1, 2, \ldots, T$ **do**
    Get mini-batch of data samples and corresponding labels: $(X^{(t)}, Y^{(t)})$
    Compute gradient: $g_t \leftarrow \nabla_\theta \mathscr{L}(X^{(t)}, Y^{(t)}, \theta)$
    Update biased first moment estimate: $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
    Update biased second moment estimate: $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
    Compute bias-corrected first moment estimate: $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
    Compute bias-corrected second moment estimate: $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
    Update parameters with weight decay: $\theta \leftarrow (1-\lambda\,\eta)\theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}$
end for
return θ

4.2 Implementation

The primary objective of this thesis is to identify the direction of speech based on two options, namely left or right, as well as to distinguish between noise and speech-in-noise from the brain activity recorded with EEG. This section introduces the model architecture, which takes inspiration from EEGNet [Lawhern et al., 2018]. The proposed method involves developing a neural network for a classification task

that identifies whether the sound being attended to originates from the left or the right side, and the system can be expanded to cover additional directions. This technique is known as locus of attention (LoA) classification and has been previously investigated with promising results in [Geirnaert et al., 2021; Su et al., 2022; Vandecappelle et al., 2021]. Instead of classifying which reconstructed sound stream resembles the original sound stream the most, the method focuses on classifying the direction the attended sound.

Figure 4.3 displays the model architecture used for a classification task, which involves extracting features from 1 second time samples obtained from EEG signals. The first step of the model is to extract temporal features from each channel individually by applying a 1D convolution kernel. This allows the model to capture channel-specific features, which are then normalized using batch normalization to prevent overfitting. The first layer of the model does not include any activation function, following the EEGNet architecture described in [Lawhern et al., 2018].

The next step of the model is to extract spatial features across all channels, with a fixed kernel size of 1 along the temporal dimension. This allows the model to extract features across all channels by analyzing the signals. Batch normalization and average pooling are applied to downscale the data by a factor of 4.

The model then proceeds to extract features from the single-dimensional data, followed by batch normalization and average pooling to further downscale the data. Both the former and current layers use the Mish activation function on their outputs.

Finally, the data is flattened and passed through fully connected layers, which returns the output logits for the two classes of interest.

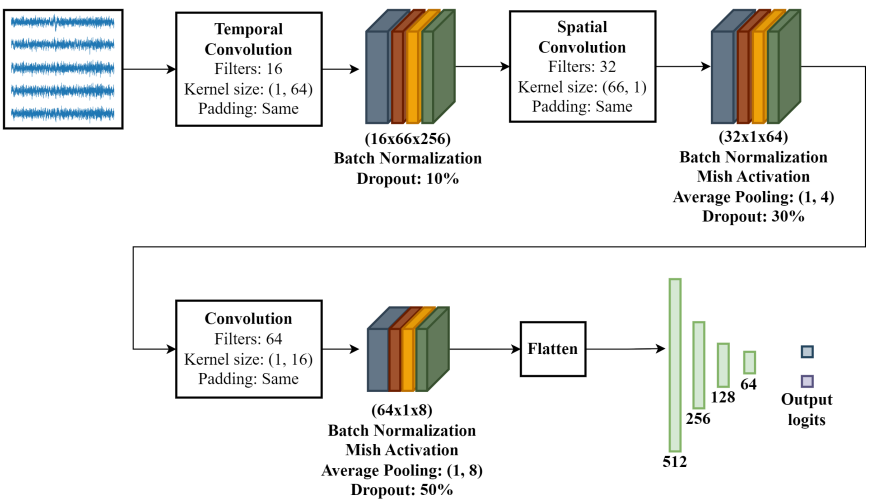


Figure 4.3 Deep learning architecture for the classification tasks. The colored boxes represent the outputs of convolutional layers, with additional operations specified below each box.

The thesis focuses on two task classification i.e., distinguishing between noise vs. speech-in-noise, and distinguishing between Left vs. Right attended speech. As such two models are trained separately for each task, resulting ultimately in a three class classification system. The two models are trained using the same architecture and hyper parameters. The subsequent algorithm outlines the primary steps involved in the training and evaluation of the DCNN model, providing a more detailed account of the training methodology. It is worth noting that since pytorch is used as a training environment, the algorithm is presented in a corresponding format.

Algorithm 2 DCNN model training training and validation algorithm

Initialize optimizer: $Adam\ (optimizer)$
Initialize DCNN model: $Model$
Initialize loss function: $Cross\ entropy\ loss\ (criterion)$
Initialize parameters: $\eta : 0.0005$, $batch\ size : 64, Epochs : 50$
Initiate Model training:
for $epoch = 1, 2, \ldots, Epochs$ **do**
    for $t = 1, 2, \ldots, T_{training}$ **do**
        Get mini-batch of data samples and corresponding labels: $(X^{(t)}, Y^{(t)})$
        Get model output: output = Model($X^{(t)}$)
        Calculate loss using output and labels: loss = criterion(output, $Y^{(t)}$)
        Back propagate through model using loss: loss.backward()
        Perform parameter update based on current gradient: criterion.step()
    end for
    Initiate Model Validation:
    for $t = 1, 2, \ldots, T_{validation}$ **do**
        Get mini-batch of data samples and corresponding labels: $(X^{(t)}, Y^{(t)})$
        Get model output: output = Model($X^{(t)}$)
        Calculate loss using output and labels: loss = criterion(output, $Y^{(t)}$)
        Store validation loss for saving model: $loss_{validation}$ = loss.item()
    end for
    Saving model based on current validation loss:
if $loss_{validation}$ <= $loss_{best}$:
    Save model
    Update best loss: $loss_{best} = loss_{validation}$
end for

5

Conditional variational autoencoder and Data augmentation

Medical practices involving EEG data often have a limited amount of data available, as is the case with the current study. In such situations, it can be challenging to collect additional data due to practical constraints. To address this issue, it is crucial to use a generative network that can create artificial data that closely resemble the original data. By doing so, we can increase the size of the data set and potentially improve the performance of the analysis. To prevent the deep learning model from being too closely fit to the training data and unable to generalize to new data from different subjects or trials, it is important to acquire more data. In order to address this problem, the thesis proposes using a Conditional Variational AutoEncoder (CVAE) [Mu and Chen, 2022; Liu et al., 2022] to augment the existing training data by generating synthetic data that has a distribution similar to that of the original training data.

5.1 Theory

Variational Autoencoders (VAEs) are a type of data generative neural network that learn a low-dimensional representation of high-dimensional data by leveraging the probabilistic nature of latent variables. CVAEs are a type of VAE that allows for the conditional generation of data based on some input information [Sohn et al., 2015]. The CVAE framework extends the traditional VAE architecture by conditioning the input data with some auxiliary information. In the CVAE framework, both the input data and the auxiliary information (e.g., class labels or attributes) are used to generate new data samples that are similar to the input data and conditioned on the auxiliary information.

The training process of CVAEs involves maximizing the evidence lower bound (ELBO) objective, which is defined as:

$$ELBO = E[\log p(x|z,y)] - \mathrm{KL}[q(z|x,y)||p(z|y)], \tag{5.1}$$

where the variable x represents the input data, y represents auxiliary information (in this case, labels), z denotes the latent variable. The conditional likelihood of the data given the latent variable and auxiliary information is denoted by $p(x|z,y)$, while the approximate posterior distribution of the latent variable given the input data and auxiliary information is denoted by $q(z|x,y)$. The prior distribution of the latent variable is represented by $p(z|y)$.

The first term of the ELBO objective, $E[\log p(x|z,y)]$, encourages the generated samples to be similar to the input data and conditioned on the auxiliary information. The second term, $\mathrm{KL}[q(z|x,y)||p(z|y)]$, penalizes the approximate posterior distribution of the latent variable from deviating too much from the prior distribution. This ensures that the latent variables are well-structured and informative.

The encoder network, $q(z|x,y)$, maps the input data and the auxiliary information to a distribution over the latent variable. This is typically parameterized as a Gaussian distribution, with mean and variance vectors that depend on both the input data and the auxiliary information. The decoder network, $p(x|z,y)$, maps the latent variable and the auxiliary information to a distribution over the data space. This is also typically parameterized as a Gaussian distribution, with mean and variance vectors that depend on both the latent variable and the auxiliary information.

During training, the CVAE samples a latent variable from the approximate posterior distribution, $z \sim q(z|x,y)$, and then generates a new data sample by sampling from the decoder distribution, $x \sim p(x|z,y)$. The ELBO objective is then optimized using gradients to update the parameters of the encoder and decoder networks. As such the model learns from the available training data how to reconstruct data samples using the latent representation.

CVAEs have been successfully applied in various domains, including image generation, text generation, and speech synthesis [Kingma and Welling, 2013; Mirza and Osindero, 2014; Bowman et al., 2015; Hsu et al., 2017]. For example, CVAEs have been used for image generation tasks such as generating new images based on class labels or attributes, as well as for text generation tasks such as generating new sentences based on a given prompt or topic. CVAEs have also been used for speech synthesis tasks, where they can generate realistic speech samples by conditioning on speaker identities or emotions.

As mentioned earlier the CVAE is composed of two main components: the encoder and the decoder networks. The encoder maps the input data to a lower-dimensional latent space, while the decoder takes this latent space representation to reconstruct the original data. To enable the generation of label-specific data, CVAEs additionally use labels to guide the model's training. The general framework of the

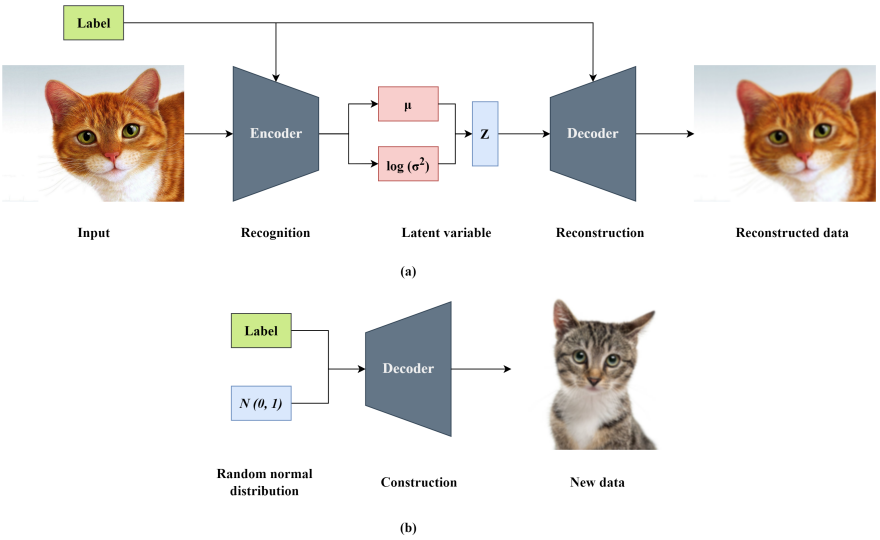


Figure 5.1 Displaying: (a) Conditional variational autoencoder training (b) Data generation using random noise.

CVAE is illustrated in Figure 5.1 (a), where the input data comprises both the training images and their corresponding labels. The encoder network uses this data to output the latent variables of a specific dimension. These latent variables are then fed into the decoder network, together with the training labels, to reconstruct the original images. The loss is computed based on the reconstructed images and their comparison with the original data. Figure 5.1 (b) shows the process of generating new data using a trained decoder network. The model takes as input a random noise distribution of the same dimension as the latent space, along with the labels for the specific class, resulting in an output image that is generated based on the specified labels.

5.2 Implementation

The secondary objective of this thesis was to generate artificial data to further enhance the total data available for our binary classification models. As deep learning models are highly dependent on the amount and uniqueness of the data set available, generating additional data is imperative. This is done to help the model generalize better and avoid overfitting on a smaller data set. The proposed architecture in this thesis for CVAE is shown in Figure 5.2. The following two sub-sections will explain the **encoder** and **decoder** part of the CVAE.

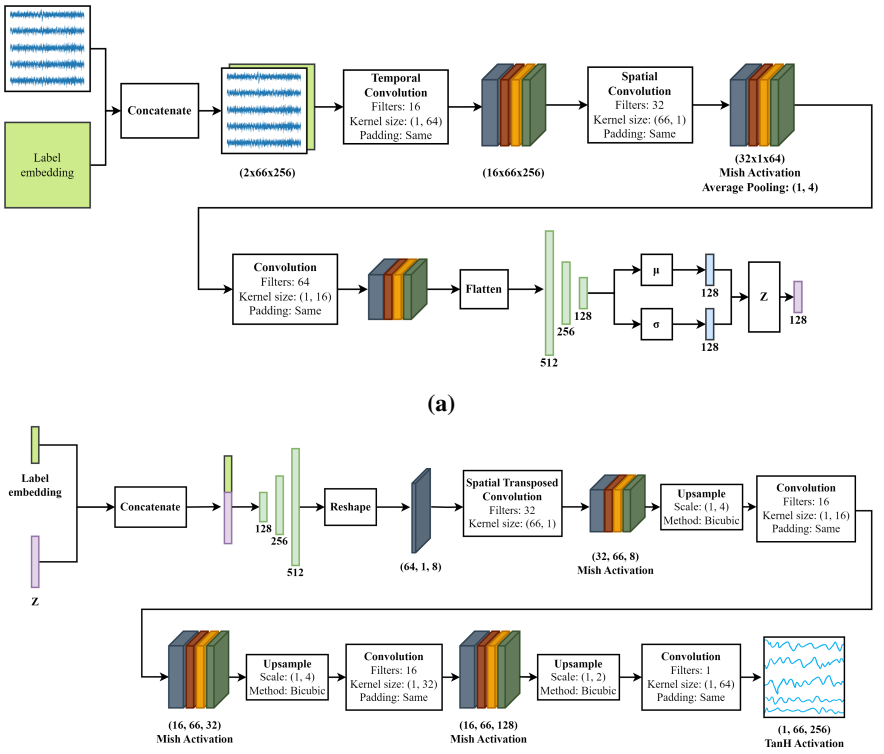


Figure 5.2 (a) Encoder network for CVAE model. (b) Decoder network for CVAE model.

Encoder

The encoder model starts off by creating an embedding for incoming labels, this is done so that we can associate each training sample with its respective label. The training samples are then concatenated with their respective label embedding and passed through the first layer of the encoder network. The model mimics the DCNN model used earlier for classification, as in both cases we are trying to extract *features* to represent the data.

It is worth noting that the encoder network does not involve the use of batch normalization layers. Batch normalization (BN) is used to standardize the inputs of each layer. It helps to reduce the internal covariate shift, which is the phenomenon of the distribution of the layer inputs changing as the network trains.

The reason why CVAE does not use batch normalization is that it can interfere with the probabilistic nature of the model. The standardization of the inputs introduced by batch normalization can constrain the learned distribution of latent variables, which can lead to a loss of information in the encoding process.

The final fully connected layer of the encoder consists of two outputs, the mean and variance of the encoded latent variables: The encoder network takes an input data point and produces two vectors that represent the mean and variance of the probability distribution over the latent variables. However these values in of themselves are not directly used in the decoder network, rather we sample a latent variable from the mean and variance variables. This is done by using a method known as the *reparameterization trick*.

The re-parameterization trick involves sampling a value ε from a standard normal distribution N(0,1), and then transforming it into a sample from the learned probability distribution over the latent variables. This transformation is done using the following equation:

$$z = \mu + \varepsilon\sigma. \tag{5.2}$$

In this equation:

- μ is the mean of the learned probability distribution over the latent variables, output by the encoder network.
- σ is the standard deviation (square root of the variance) of the learned probability distribution over the latent variables, output by the encoder network.
- ε is a random sample drawn from a standard normal distribution with mean 0 and variance 1.

By using this reparameterization trick, we can ensure that the back propagation algorithm can compute gradients for the encoder network, since the random variable epsilon is independent of the parameters of the encoder network. The resulting latent variable $\boldsymbol{Z}$ (shown in Figure 5.2) can then be fed into the decoder network to generate a reconstructed data point.

Decoder

The decoder starts off by having incoming information from the encoder (i.e., the latent variable $\mathbf{Z}$) with auxiliary information (i.e., the associated labels) for each batch of data. The labels are converted into a vector embedding representative of each label, and concatenated with the latent variable. The decoder model then passes this concatenated vector through a fully connected layer which is an inverted form of the output fully connected layer from the encoder, essentially the decoder will mimic the encoder model, in the opposite direction.

The output of the fully connected layer is reshaped into a 2D structure, followed by a *spatial transposed convolution* which is used to append all the channels into the 2D structure. This is followed by a series of *upsampling* and *convolution* layers. The upsampling layers are introduced to increase the dimensionality in the temporal

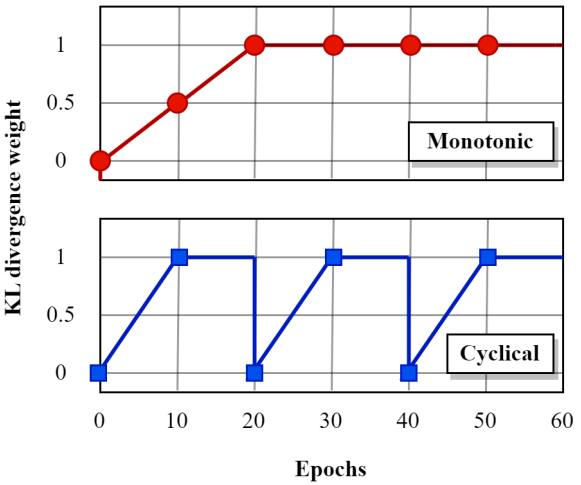


Figure 5.3 Comparison between traditional KL constant schedule and cyclical annealing schedule.

dimension of each channel in the 2D structure, with convolution layers added to extract features in an attempt to *restructure* the upsampled information into the proper EEG information. Upsampling is preferred over transposed convolution [Pons et al., 2021] as the former is known to add tonal artifacts which can show up as a noisy appearance within the reconstructed image.

By using the latent variables with the auxiliary information the decoder is capable of regenerating the original data. It is trained to minimize the reconstruction error and the KL divergence, and it is used to generate new samples by sampling from the learned latent distribution and varying the conditional variables.

Training strategy

During the training of a Conditional Variational Autoencoder (CVAE), the Kullback-Leibler (KL) divergence term in the loss function is utilized to regulate the latent space to match a prior distribution. However, sometimes, this term can dominate the loss function excessively, leading to the model ignoring the conditional information. To resolve this issue, a cyclical KL divergence can be employed in the loss function.

In Figure 5.3, we can compare the traditional monotonic schedule with the proposed cyclical annealing schedule. The traditional method involves gradually increasing the weight of the KL divergence term up to a specified number of training iterations or epochs, after which the weight remains constant for the rest of the training process. The cyclical schedule can be defined in several ways, with a recent study [Fu et al., 2019] suggesting the usage of a cyclical annealing schedule. The

proposed cyclical annealing schedule gradually increases the weight of the KL divergence term up to one and then keeps it constant for a certain number of epochs before decreasing it back down to zero and repeating the cycle. This approach allows the model to focus on the reconstruction loss initially and then gradually regularize the latent space, which balances the reconstruction loss and the KL divergence loss and prevents the problem of KL vanishing.

The model training algorithm is shown in Algorithm 3. As we can see, we train the model for a total of 500 epochs. This applies to both generating data for (left vs. right) speech for attention decoding *and* (noise vs. speech-in-noise) for differentiating between noise and speech. For generating new data the decoder network from the final trained model is used, with input: normal random Gaussian noise and corresponding labels as auxiliary information for generating images for the specific labels.

Algorithm 3 Conditional Variational Autoencoder training algorithm

Initialize: Training data X, conditional labels Y
Initialize: Encoder network $q_\theta(\mu, \sigma|x, y)$, Decoder network $p_\phi(x|z, y)$
Initialize: Adam optimizer with learning rate $\alpha : 0.0005$, number of epochs $T : 500$
for $t = 1$ to T **do**
    for each (x, y) in X, Y **do**
        Compute the mean μ and standard deviation σ of the input data using the encoder network: $\mu, \sigma = q_\theta(x, y)$
        Sample a latent variable z from $N(\mu, \sigma^2 I)$
        Decode the latent variable z and conditional label y using the decoder network: $\hat{x} = p_\phi(z, y)$
        Compute the reconstruction loss: $L_{\text{rec}} = \frac{1}{2}|x - \hat{x}|^2$
        Compute the KL-divergence loss: $L_{\text{KL}} = -\frac{1}{2}\sum_{j=1}^{Z}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$
        Compute the total loss: $L = L_{\text{rec}} + L_{\text{KL}}$
        Compute the gradients of the total loss with respect to the parameters θ and ϕ: $\nabla_{\theta,\phi} L$
        Update the parameters using the Adam optimizer: $\theta, \phi = \text{Adam}(\nabla_{\theta,\phi} L, \theta, \phi)$
    end for
end for

6

Results

This chapter presents the results from the implementation of DCNN and CVAE. The first section will give a short introduction into the methods for evaluating the final trained model performance(s), the second section will display the training graphs for the models and discuss the given graphs, the third section will display results achieved by the trained models using the evaluation criteria and finally the last section will display the CVAE's ability to generate data.

6.1 Evaluation criteria

To evaluate the performances of the models, four evaluation criteria are considered: accuracy (ACC), area under curve (AUC) score, sensitivity, and specificity. These metrics are commonly used to assess the effectiveness of classification models. Accuracy is determined by the ratio of correctly predicted samples to the total number of samples, which can be expressed as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6.1}$$

Here, TP represents the number of true positives, TN represents the number of true negatives, FP represents the number of false positives, and FN represents the number of false negatives. AUC score is calculated as the area under the Receiver Operating Characteristic (ROC) curve, which is a plot of true positive rate (TPR) against the false positive rate (FPR) at various threshold values:

$$AUC = \int_0^1 TPR(FPR^{-1}) dFPR. \tag{6.2}$$

Sensitivity is a measure of the proportion of true positives predicted by the model over the total number of actual positive samples and is calculated as:

$$Sensitivity = \frac{TP}{TP + FN}. \tag{6.3}$$

Similarly, specificity is a measure of the proportion of true negatives predicted by the model over the total number of actual negative samples and is expressed as:

$$Specificity = \frac{TN}{TN + FP}. \tag{6.4}$$

6.2 Model training graphs

Both DCNN models, designed for noise vs. speech-in-noise and attended left vs. attended right speech, are subjected to uniform training for a period of 50 epochs. The model training parameters and hyperparameters remain consistent between the two models, indicating that the divergence between them arises solely from the data utilized and its potential influence on the model training. For the first model (noise vs. speech-in-noise) the training graph, explaining both accuracy and loss, are seen in Figure 6.1. The x-axes of the graphs store results equally spaced 6 times per epoch, as such resulting in a total of 300 steps. As we can see the model starts to overfit somewhere around 100 steps, and after that the model validation loss slowly increases. The final models saved during training were the models that performed the *best* on the validation data, i.e., the model that gave the lowest validation loss and highest validation accuracy on the validation data.

For the second model (Attended Left vs. Attended Right Speech) the training graph for both accuracy and loss can be seen in Figure 6.2. From the graph it is evident that the model *so far* did not begin to overfit on the training data, however the model performance on the validation data does slow down considerably and comes to somewhat of a halt at a relatively higher validation loss compared to the training loss. Despite the noticeable increase in validation accuracy, the rise in loss implies a lack of confidence in the model's ability to predict accurately on the validation data. Once again, the best model, i.e., the one with the lowest validation loss and highest validation accuracy, was saved and later used for evaluation.

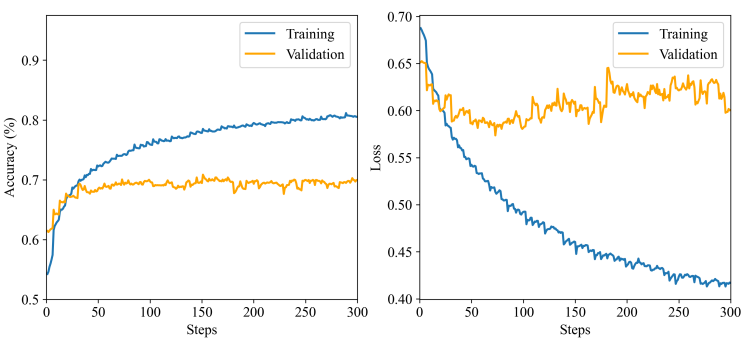


Figure 6.1 Noise vs. Speech-in-noise: Training/Validation graphs for Accuracy and Loss.

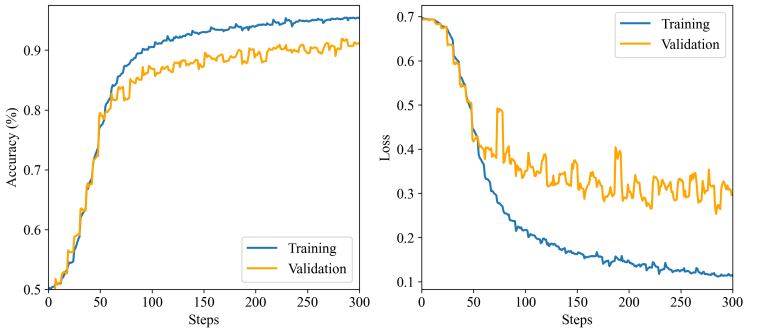
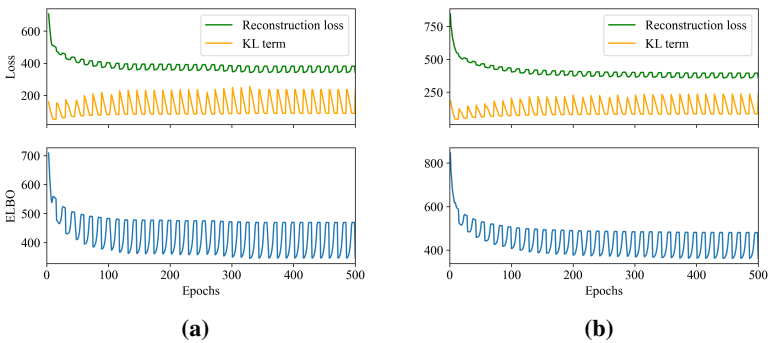


Figure 6.2 Left vs. Right attended speech: Training/Validation graphs for Accuracy and Loss.

Based on the graphs, it is evident that the two models exhibit varying performance despite sharing identical model structure and hyperparameters, which can be attributed to their respective tasks and available data set. The difference in their ability to generalize, specifically in the tasks of Noise versus Speech and Left versus Right Speech, is likely the root cause.

Figure 6.3 presents the training graphs for the two cases: Noise vs. Speech-in-noise (left graph) and Attended Left vs. Attended Right Speech (right graph) for CVAE training. The graphs display the reconstruction loss (or error) in conjunction with the KL term from the ELBO equation for both cases.

It is evident that the reconstruction error for both cases initially starts at relatively high loss values, but eventually converges to a minimum and oscillates around that minimum. On the other hand, the KL term increases from a low value and then continues to oscillate around that maximum.



(a) **(b)**

Figure 6.3 ELBO and loss terms for both data: (a) Noise vs. Speech-in-noise (b) Left vs. Right attended speech.

This behavior of the loss terms can be attributed primarily to the change in the KL constant applied to the KL term in the ELBO loss equation. As the constant increases, the contribution of the KL term also increases, resulting in an overall increase in the ELBO loss, as well as the reconstruction error. This trend is also reflected in the ELBO loss shown for both cases, where it starts at a relatively high value and eventually stabilizes at a minimum, oscillating around that value.

6.3 Performance measures

Figure 6.4 presents the performance of individual subjects in both original and augmented versions of the models for the classification tasks. It is important to note that the test data constitutes 20% of the total data, maintaining the 60-20-20(%) ratio commonly used in deep learning applications for train-validation-test. Mean accuracy for each task, across each model, is shown as dashed lines. The average results for both the original and augmented versions of the Left vs. Right attended speech model demonstrate superior performance compared to the Noise vs. Speech-in-noise model. Notably, the augmented versions of both models exhibit better performance, with a more significant improvement observed in the Left vs. Right attended speech model. Furthermore, some subjects show substantial improvements, such as subjects #19 and #26 in the Left vs. Right attended speech model, with enhancements ranging from 20-35% while other subjects show minor improvements.

The average performance of the two models on their respective test data are presented in Table 6.1. The Noise vs. Speech-in-noise model demonstrates an accuracy of approximately 70%, while the Left vs. Right attended speech model achieves an accuracy of around 85%. Both models perform relatively well with the latter achieving 90%+ AUC score. It is noteworthy that for the former, specificity indi-

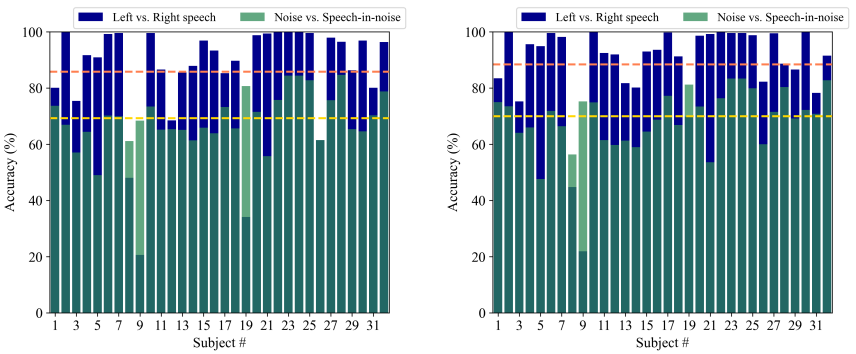


Figure 6.4 Subject wise classification results across models for both (a) Original models (b) Augmented models. Dashed lines represent the average accuracies for both models.

Model name	ACC (%)	AUC (%)	Sensitivity (%)	Specificity (%)
Noise vs. Speech-in-noise	70.0	77.5	65.5	74.4
Left vs. Right attended speech	**84.9**	**92.4**	**80.2**	**89.4**

Table 6.1 Model performance for both models on available test data.

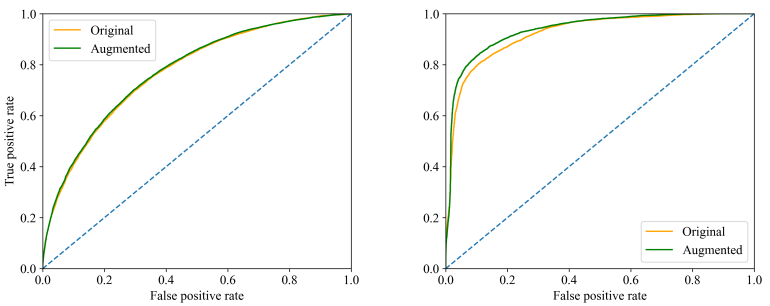
Model name	ACC (%)	AUC (%)	Sensitivity (%)	Specificity (%)
Noise vs. Speech-in-noise	70.5	78.3	68.2	72.8
Left vs. Right attended speech	**86.6**	**93.7**	**80.5**	**92.5**

Table 6.2 Model performance for both augmented models on available test data.

cates how accurately the model predicts the speech-in-noise samples, while for the latter, specificity represents the model's ability to predict the right side speech samples.

Furthermore, Table 6.2 presents the outcomes of both models after training on additional augmented data. The results indicate that both models have better performance across all evaluation criteria when trained on augmented data for the given test data set. The Noise vs. Speech-in-noise model showed a slight improvement from 70.0% to 70.5%, while the Left vs. Right attended speech model demonstrated a more significant increase in performance from 84.9% to 86.6%. The reason for the lack of substantial performance improvement in the former model is not due to the lack of data, but because the model has already learned enough to generalize well on unseen test data. In contrast, the latter model's slightly better performance shows that the scarcity of data is one of the factors limiting the model's maximum potential performance.

Figure 6.5 displays the ROC curves, which provide insights into the performance of a binary classification model. The ROC curves show the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at different classifica-



(a) **(b)**

Figure 6.5 ROC curve for both models with augmented data: (a) Noise vs. Speech-in-noise (b) Left vs. Right attended speech.

tion thresholds. The dotted lines in the plot represent the worst possible results (i.e., random guessing). From Figure 6.5.(a), it can be observed that the model performs similarly with and without augmented data, with the augmented version slightly outperforming the original model. However, in Figure 6.5.(b), the augmented version of the model exhibits noticeable improvement over the original model.

6.4 CVAE reconstructed data

The Conditional Variational Autoencoder (CVAE) was trained with two objectives: (a) generating data for Noise vs. Speech-in-noise, and (b) generating speech for a specific direction. The performance of both models in generating multi-channel EEG data was found to be similar, as shown in Figure 6.3 which displays the training graphs and results. To visually test the CVAE, the test data set used previously was normalized and divided into one-second time windows, which were then passed through the CVAE model for the specific task. The reconstructed data was considered as the output of the CVAE and compared with the original data that the model aims to reconstruct. Figure 6.6 presents the results for the Noise vs. Speech-in-noise CVAE model.

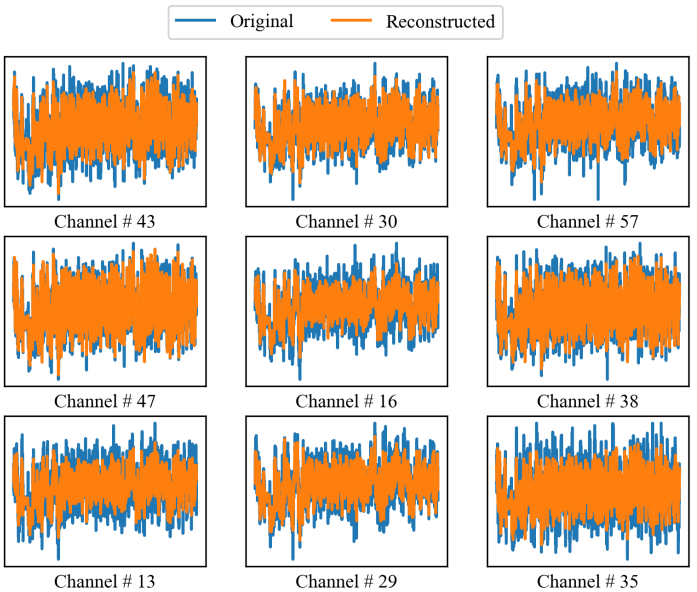


Figure 6.6 Comparison of Original vs. Reconstructed EEG signals using CVAE model across randomly extracted channels for an entire trial duration (38 seconds).

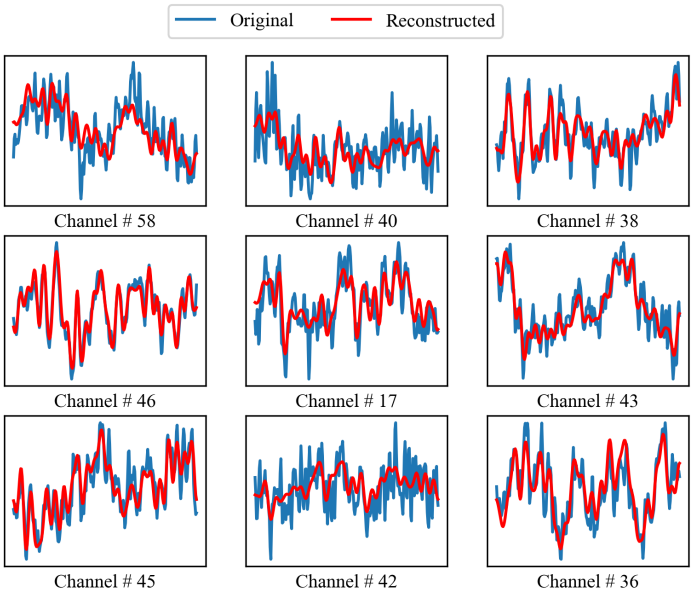


Figure 6.7 Comparison of Original vs. Reconstructed EEG signals using CVAE model across randomly extracted channels for one second samples of EEG signals.

Figure 6.6 shows the reconstructed data for randomly selected channels across the 66 available channels, overlapped with the original channels, to demonstrate the similarity between the original and reconstructed data. As observed from the figure, the reconstructed data aligns well with the original data, indicating a noticeable similarity. To further analyze the results, the 38 seconds of data was sampled into randomly distributed one-second time windows across all channels, as shown in Figure 6.7. The figure displays the reconstructed samples for one-second windows overlapped with the original one-second samples, with the respective channels labeled. From the figure, it can be deduced that most of the channels are reconstructed well with noticeable similarity to the original data, although some channels show slight deviations especially concerning the high-frequency details in the signals.

7

Discussion

In this chapter, the findings from Chapter 6 are analyzed and discussed. The possible reasons behind the specific results obtained by the DCNN models on their respective data sets are explored, including the ability of CVAE models to generate novel and distinct data for further training of the DCNN models. Additionally, a comparison of the current results with existing literature is provided.

7.1 Deep convolution neural network

Despite the small time windows (one second) used in this study, both DCNN models perform relatively well, yielding decent results. However, the accuracy for distinguishing between Noise and Speech-in-noise remains around 70%, despite higher training accuracies observed in the training graphs. This suggests that the models may be overfitting and failing to generalize from the available training data. Although augmented data from CVAE models was utilized to compensate for this, the performance did not show significant improvement, indicating that data scarcity may not be the primary reason for the lack of generalization beyond 70%. It is likely that the current model's inability to learn better features for distinguishing between Noise and Speech-in-noise is the underlying issue. Future research could explore better pre-processing methodologies to enhance feature extraction by the deep learning models. Additionally, investigating different types of deep learning models, such as adjusting model architecture parameters or incorporating more complex structures, may be beneficial in extracting better features for this task.

The results for distinguishing between Left and Right Speech demonstrate significantly better accuracy of approximately 85%, indicating potential for real-world production use. To explore the impact of increased data on model performance, data augmentation was performed using CVAE, resulting in an accuracy of approximately 87%. This suggests that data scarcity may be one of the reasons why the models fail to learn beyond this point. However, it is important to consider other factors such as hyperparameters, loss function, and model architecture that could also

Study	Classification	EEG	Group	Method	Time window (s)	Accuracy
Mirkovic et al. [Mirkovic et al., 2015]	SR	2–8 Hz	NHS	LR	60	88.0%
Das et al. [Das et al., 2016]	SR	1–9 Hz	NHS	LR	30	76.0%, 87.2 %
Mirkovic et al. [Mirkovic et al., 2016]	SR	2–8 Hz	NHS	LR	60	69.3%, 84.8%
Fuglsang et al. [Fuglsang et al., 2017]	SR	1–8 Hz	NHS	LR	40-50	80.0-90.0%
Ciccarelli et al. [Ciccarelli et al., 2019]	SR	2–32 Hz	NHS	LR & DNN	10	87.0%
Taillez et al. [Taillez et al., 2020]	SR	1-32 Hz	NHS	DNN	2	67.8%
Vandecappelle et al. [Vandecappelle et al., 2021]	LoA	1–32 Hz	NHS	CNN	1-2	81.0%
Su et al. [Su et al., 2022]	LoA	1–32 Hz	NHS	CNN	1	71.9%, 90.1%
Current work	LoA	**0.5–70 Hz**	HIS	**DCNN**	**1**	**86.6%**

Table 7.1 Comparison of EEG-based auditory attention decoding performance between our study and previous literature, including stimulus reconstruction (**SR**) and locus of attention (**LoA**) models for normal (**NHS**) and hearing-impaired (**HIS**) subjects using linear regression (**LR**), deep neural network (**DNN**) and (deep) convolutional neural networks ((**D)CNN**).

contribute to the final results. The use of one-second time windows in the model suggests its potential for real-time information processing and production use.

The utilization of small time windows and testing on unseen trial subjects is a noteworthy aspect of this study. The fact that the model performs well despite having no prior information on the given trials highlights its capacity to adapt to new data. Even when unseen portions of the same trials are used for validation and testing, some level of information sharing among training and testing sets may still occur. However, the current study focuses primarily on unseen trials for both validation and test sets, providing confidence in the model's performance on real-world unseen data.

The comparison of the proposed methodology's performance with that of previous studies is presented in Table 7.1. The results presented in [Mirkovic et al., 2015] indicate that the proposed method achieved an accuracy of 88.02%, which is promising. However, it should be noted that this result was obtained using a 60-second time window, which may not be practical in real-time scenarios. Similarly, [Fuglsang et al., 2017] reported high accuracies ranging from 80-90% on 40-50 second time windows, which may also not be practical in real-world applications.

In contrast, the study conducted by [Das et al., 2016] reported a decoding accuracy of 76.0% for non-subject-specific decoders and 87.2% for subject-specific decoders. It is important to note that while subject-specific decoders achieved higher accuracy, they may not be practical in real-world applications as subject specific decoders would be a hindrance to product scalability. Additionally, [Mirkovic et al., 2016] proposed two methods of decoding EEG (cap-EEG and ear-EEG) with electrode placement around the cap or ear of the subject, respectively. The study reported an average accuracy of 84.8% for cap-EEG and 69.3% for ear-EEG, with the latter suggesting that unobtrusive miniaturized electrodes placed around the ear are sufficient for successful decoding of the attended speaker in two-speaker scenarios. However, the proposed methodology outperforms [Mirkovic et al., 2016] with higher accuracies and smaller time windows.

According to the studies conducted by [Taillez et al., 2020; Vandecappelle et

al., 2021], deep learning architectures were utilized for attention decoding, and achieved accuracies of 67.8% and 81.0%, respectively, using 2-second time windows. The latter study also experimented with a range of 1-2 second time windows. However, in comparison, the proposed study achieved higher accuracy (86.6%) by utilizing a deep learning approach with a 1-second time window for attention decoding. The proposed method by [Su et al., 2022] employed two publicly available data sets [Fuglsang et al., 2018; Das et al., 2019], which resulted in average accuracies of 71.9% and 90.1%, respectively. Although the study showed promising results, particularly on the latter data set, it struggled to achieve strong accuracy results on the former data set.

7.2 Conditional variational autoencoder

The CVAE models have demonstrated the ability to produce highly similar images when tested on previously unseen data. This suggests that CVAE models have potential as a tool for generating new data to enhance multi-channel EEG tasks. Both the original and augmented models have exhibited significant improvements in performance in situations where limited or unique data has hindered model performance. However, while the CVAE models capture the general characteristics of EEG channels, they struggle to reproduce the minute details of EEG signals, as evidenced by the generated channels having a *smoother* appearance than the original ones. Nevertheless, the model's performance has improved despite this limitation, highlighting the generalizing capabilities of DCNN models. Further research can be conducted to address this shortcoming of the CVAE models and generate data that accounts for the subtle changes within EEG samples.

8

Conclusion

The objective of this chapter is to present conclusions, analyze the findings, and discuss the final results achieved in this thesis. The conclusions drawn in this chapter are derived from the findings presented in Chapter 6, which focus on the utilization of DCNN and CVAE. In brief, the results indicate that CVAE is capable of generating multi-channel EEG data that is relatively realistic, while the DCNN yields favorable outcomes when applied to 1-second time windows. Furthermore, incorporating generated data into the training process enhances the performance of the DCNN models.

8.1 Deep convolution neural network classification

The results obtained from the two DCNN networks demonstrate promising outcomes, particularly with the latter model (Left vs. Right attended speech) achieving significant accuracy (84.9% ACC and 92.4% AUC). Moreover, the utilization of 1-second time windows during training indicates potential for real-time implementation. However, the former model (Noise vs. Speech-in-noise) tends to exhibit overfitting on the training data, struggling to generalize effectively to unseen trial data. This limitation may stem from the model's difficulty in identifying effective features that can distinguish between the two states, as evidenced by the relatively similar results even with data augmentation. Nevertheless, the augmented data appears to significantly improve the performance of the latter model (Left vs. Right attended speech), with accuracy and AUC increasing to 86.6% and 93.7%, respectively, underscoring the effectiveness of training with augmented data. In conclusion, DCNN models prove to be valuable tools in decoding attention-related features in multi-channel EEG data.

8.2 Conditional variational autoencoder augmentation

The CVAE model is trained using a latent space that represents the original training data, incorporating auxiliary information in the form of labels to generate data

specific to each label (e.g., Left vs. Right attended speech or Noise vs. Speech-in-noise). The generated data on unseen test trials, as demonstrated in Chapter 6, clearly illustrates the model's capability in generating samples that closely resemble the given input image. Notably, the augmented data used in training is constructed from unique examples generated by using Gaussian normal noise, ensuring that each generated sample is distinct. Subsequently, the trained DCNN models exhibit improvement after data augmentation, indicating the effectiveness of augmented data in enhancing model learning and generalization. In conclusion, CVAE models serve as a valuable tool for data augmentation in the context of multi-channel EEG data.

Bibliography

Acharya, R., O. Faust, N. Kannathal, T. Chua, and S. Laxminarayan (2005). "Non-linear analysis of EEG signals at various sleep stages". *Computer methods and programs in biomedicine* **80**:1, pp. 37–45.

Alickovic, E., T. Lunner, F. Gustafsson, and L. Ljung (2019). "A tutorial on auditory attention identification methods". *Frontiers in neuroscience*, p. 153.

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**, p. 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". In: *Seminars in Hearing*. Vol. 42. 03. Thieme Medical Publishers, Inc., pp. 260–281.

Aroudi, A., B. Mirkovic, M. De Vos, and S. Doclo (2016). "Auditory attention decoding with EEG recordings using noisy acoustic reference signals". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 694–698.

Bethge, D., P. Hallgarten, T. Grosse-Puppendahl, M. Kari, L. L. Chuang, O. Özdenizci, and A. Schmidt (2022). "EEG2Vec: learning affective EEG representations via variational autoencoders". In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, pp. 3150–3157.

Blinowska, K. J. and M. Malinowski (1991). "Non-linear and linear forecasting of the EEG time series". *Biological cybernetics* **66**:2, pp. 159–165.

Bowman, S. R., L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio (2015). "Generating sentences from a continuous space". *arXiv preprint arXiv:1511.06349*.

Brodbeck, C., L. E. Hong, and J. Z. Simon (2018). "Rapid transformation from auditory to linguistic representations of continuous speech". *Current Biology* **28**:24, pp. 3976–3983.

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.

Cheveigné, A. de, D. D. Wong, G. M. Di Liberto, J. Hjortkjær, M. Slaney, and E. Lalor (2018). "Decoding the auditory brain with canonical component analysis". *NeuroImage* **172**, pp. 206–216.

Ciccarelli, G., M. Nolan, J. Perricone, P. T. Calamia, S. Haro, J. O'sullivan, N. Mesgarani, T. F. Quatieri, and C. J. Smalt (2019). "Comparison of two-talker attention decoding from EEG with nonlinear neural networks and linear methods". *Scientific reports* **9**:1, p. 11538.

Craik, A., Y. He, and J. L. Contreras-Vidal (2019). "Deep learning for electroencephalogram (EEG) classification tasks: a review". *Journal of neural engineering* **16**:3, p. 031001.

Dai, M., D. Zheng, R. Na, S. Wang, and S. Zhang (2019). "EEG classification of motor imagery using a novel deep learning framework". *Sensors* **19**:3, p. 551.

Das, N., W. Biesmans, A. Bertrand, and T. Francart (2016). "The effect of head-related filtering and ear-specific decoding bias on auditory attention detection". *Journal of neural engineering* **13**:5, p. 056014.

Das, N., T. Francart, and A. Bertrand (2019). "Auditory attention detection dataset kuleuven". *Zenodo*.

Dash, D., P. Ferrari, S. Dutta, and J. Wang (2020). "Neurovad: real-time voice activity detection from non-invasive neuromagnetic signals". *Sensors* **20**:8, p. 2248.

Dose, H., J. S. Møller, S. Puthusserypady, and H. K. Iversen (2018). "A deep learning MI-EEG classification model for bcis." In: *EUSIPCO*, pp. 1676–1679.

Fu, H., C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin (2019). "Cyclical annealing schedule: a simple approach to mitigating kl vanishing". *arXiv preprint arXiv:1903.10145*.

Fuglsang, S. A., D. D. Wong, and J. Hjortkjær (2018). "Eeg and audio dataset for auditory attention decoding". *Zenodo*.

Fuglsang, S. A., T. Dau, and J. Hjortkjær (2017). "Noise-robust cortical tracking of attended speech in real-world acoustic scenes". *Neuroimage* **156**, pp. 435–444.

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.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hernández, J. L., J. Valdés, R. Biscay, J. C. Jiménez, and P. Valdés (1995). "EEG predictability: adequacy of non-linear forecasting methods". *International journal of bio-medical computing* **38**:3, pp. 197–206.

Hsu, W.-N., Y. Zhang, and J. Glass (2017). "Unsupervised learning of disentangled and interpretable representations from sequential data". *Advances in neural information processing systems* **30**.

Kingma, D. P. and M. Welling (2013). "Auto-encoding variational bayes". *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2017). "Imagenet classification with deep convolutional neural networks". *Communications of the ACM* **60**:6, pp. 84–90.

Kwak, N.-S., K.-R. Müller, and S.-W. Lee (2017). "A convolutional neural network for steady state visual evoked potential classification under ambulatory environment". *PloS one* **12**:2, e0172578.

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.

LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep learning". *nature* **521**:7553, pp. 436–444.

Liu, Y., H. Jiang, Y. Wang, Z. Wu, and S. Liu (2022). "A conditional variational autoencoding generative adversarial networks with self-modulation for rolling bearing fault diagnosis". *Measurement* **192**, p. 110888.

Löhler, J., M. Cebulla, W. Shehata-Dieler, S. Volkenstein, C. Völter, and L. E. Walther (2019). "Hearing impairment in old age: detection, treatment, and associated risks". *Deutsches Aerzteblatt International* **116**:17, p. 301.

Long, J., E. Shelhamer, and T. Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Luo, Y., L.-Z. Zhu, Z.-Y. Wan, and B.-L. Lu (2020). "Data augmentation for enhancing EEG-based emotion recognition with deep generative models". *Journal of Neural Engineering* **17**:5, p. 056021.

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.

Mirkovic, B., S. Debener, M. Jaeger, and M. De Vos (2015). "Decoding the attended speech stream with multi-channel eeg: implications for online, daily-life applications". *Journal of neural engineering* **12**:4, p. 046007.

Mirza, M. and S. Osindero (2014). "Conditional generative adversarial nets". *arXiv preprint arXiv:1411.1784*.

Misra, D. (2019). "Mish: a self regularized non-monotonic activation function". *arXiv preprint arXiv:1908.08681*.

Mu, G. and J. Chen (2022). "Developing a conditional variational autoencoder to guide spectral data augmentation for calibration modeling". *IEEE Transactions on Instrumentation and Measurement* **71**, pp. 1–8.

O'Shea, A., G. Lightbody, G. Boylan, and A. Temko (2017). "Neonatal seizure detection using convolutional neural networks". In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, pp. 1–6.

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.

Peelle, J. and A. Wingfield (2022). "How our brains make sense of noisy speech". *Acoustics Today* **18**:3, pp. 40–48.

Pons, J., S. Pascual, G. Cengarle, and J. Serrà (2021). "Upsampling artifacts in neural audio synthesis". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3005–3009.

Roy, Y., H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert (2019). "Deep learning-based electroencephalography analysis: a systematic review". *Journal of neural engineering* **16**:5, p. 051001.

Simonyan, K. and A. Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". *arXiv preprint arXiv:1409.1556*.

Sohn, K., H. Lee, and X. Yan (2015). "Learning structured output representation using deep conditional generative models". *Advances in neural information processing systems* **28**.

Soroush, P. Z., M. Angrick, J. Shih, T. Schultz, and D. J. Krusienski (2021). "Speech activity detection from stereotactic EEG". In: *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, pp. 3402–3407.

Su, E., S. Cai, L. Xie, H. Li, and T. Schultz (2022). "Stanet: a spatiotemporal attention network for decoding auditory spatial attention from EEG". *IEEE Transactions on Biomedical Engineering* **69**:7, pp. 2233–2242.

Taillez, T. de, 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.

Ullah, I., M. Hussain, H. Aboalsamh, et al. (2018). "An automated system for epilepsy detection using EEG brain signals based on deep learning approach". *Expert Systems with Applications* **107**, pp. 61–71.

Utoomprurkporn, N., K. Woodall, J. Stott, S. G. Costafreda, and D. E. Bamiou (2020). "Hearing-impaired population performance and the effect of hearing interventions on montreal cognitive assessment (moca): systematic review and meta-analysis". *International Journal of Geriatric Psychiatry* **35**:9, pp. 962–971.

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**, e56481.

Wong, D. D., S. A. Fuglsang, J. Hjortkjær, E. Ceolini, M. Slaney, and A. De Cheveigne (2018). "A comparison of regularization methods in forward and backward models for auditory attention decoding". *Frontiers in neuroscience* **12**, p. 531.

Zeng, H., C. Yang, G. Dai, F. Qin, J. Zhang, and W. Kong (2018). "EEG classification of driver mental states by deep learning". *Cognitive neurodynamics* **12**, pp. 597–606.

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

Document name
MASTER'S THESIS

Date of issue
May 2023

Document Number
TFRT-6194

Author(s)
M. Asjid Tanveer

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

Title and subtitle

Deep convolution neural network for attention decoding in multi-channel EEG with conditional variational autoencoder for data augmentation

Abstract

Objectives: This project aims to develop a deep learning-based attention decoding system that can distinguish between noise and speech in noise and also identify the direction of attended speech from the brain data recorded with electroencephalography (EEG) instruments. Two deep convolutional neural network (DCNN) models will be designed: (1) one DCNN model capable of classifying incoming segments of sound as speech or speech in background noise, and (2) one DCNN model identifying the direction (left vs. right) of incoming attended speech. In addition, two conditional variational autoencoders (CVAEs) will be trained to generate artificial data for data augmentation, with the goal of improving the performance of the final models by learning from a latent space of training data to generate unique data for each respective class.

Design: The proposed methods will be tested on a data set of 32 participants who performed an auditory attention task. Participants were instructed to attend to one of two talkers in the front and ignore the talker on the other side and background noise behind them, while high-density EEG was recorded. The EEG data consists of 66 channels in total, and all channels will be used in this study.

Main Results: The DCNN models achieved accuracy (ACC) of 69.9%, 84.9%, and area under the curve (AUC) scores of 77.5%, 92.3% on the two tasks mentioned in the objectives. With augmented data from the CVAE model, the performance of the DCNN models improved to ACC of 70.5%, 86.6% and AUC of 78.3%, 93.6%, respectively. The time window used for the EEG data was 1 second, enabling the models to work in real-time situations. The CVAE model was able to generate data for the given classes effectively, with generated data from the test data latent space showing promising results.

Conclusion: The findings of this study demonstrate the high capability of the proposed DCNN models in accurately detecting the direction of incoming speech and differentiating between noise and speech-in-noise, even with a small time window of just 1 second using multi-channel EEG data. Moreover, the results highlight the success of the CVAE model as a valuable tool for data augmentation, generating synthetic data that closely approximates the latent space information of the training data. This suggests the potential of CVAE for improving the performance of deep learning models in EEG-based attention decoding tasks.

Keywords

Classification system and/or index terms (if any)		
Supplementary bibliographical information		
ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-51	*Recipient's notes*
Security classification		

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