Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Study About the Size and Quality of the EEG Dataset

Semester Thesis

Florian Lerch

flerch@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Ard Kastrati, Damian Pascual, Martyna Plomecka
Prof. Dr. Roger Wattenhofer

May 27, 2021

Acknowledgements

I would like to thank my supervisors Ard Kastrati, Damian Pascual and Martyna Plomecka for guiding me throughout this project and giving me valuable feedback and ideas. I'm very grateful for the opportunity to collaborate with them and learned a lot during this semester project.

Abstract

This thesis contributes to a project that aims to develop an electroencephalography (EEG) based eye-tracker. Using a large dataset of simultaneously recorded EEG and infrared video-based eye-tracking, we demonstrate how its size and structure impacts the predictions and how it relates to the complexity of our deep learning models.
When shuffling all samples at the beginning, using only 40% of the available training data allowed the InceptionTime model to obtain a remarkable accuracy (93.1%) of the left/right saccade prediction, after hyperparameter tuning. In contrast, when all samples of a particular subject are either in the training or validation set, at least 80% of the available training data is required for the same model to reliably predict left/right saccade classification with a similar accuracy (93.7%). Moreover, we show that when training the models within subjects, the EEGNet model performs best, with an accuracy between 80% and 90% for most subjects.

Contents

Chapter 1

Introduction

Our eyes are an important sense for extracting visual information from our environment [1]. In cognitive science and psychology, gaze information is often considered for analyzing the attentional focus, cognitive control and decision making [2]. Given the trend of applying more naturalistic experimental paradigms such as movie watching to validate research findings, this will become an ever more important measure.

Eye-tracking systems are known to be expensive and requiring a complex setup, which excluded many neuroscience labs from collecting gaze information in the past. Recently, researchers have been able to successfully estimate gaze directions from functional Magnetic Resonance Imaging (fMRI) time series data using machine learning algorithms [3]. However, acquiring fMRI data is costly too and additionally does not provide temporal resolution at the level where the cognition actually occurs.
In contrast, electroencephalography (EEG) is a much cheaper and safe method that directly measures the electrical activity of the brain. Typically, these brain signals contain a lot of information, but extracting and understanding this information is not trivial. Moreover, such signals can differ substantially from one subject to another subject and hence, it is difficult to build deep learning models that generalize well. Although machine learning techniques proved to be successful and are an important part of several EEG-based research and application areas [4], deep learning for EEG time series classification still lags behind image recognition, especially in experimental studies and architectural designs [5].

In order to address those issues, this thesis builds upon the work of Kastrati and Plomecka, et al. in [6]. In addition to using their recorded EEG dataset and deep learning models for predicting the saccade's direction (left and right), this thesis investigates how the size and the structure of this dataset impacts the accuracy of such models. In particular, it explores how different numbers of subjects impact the accuracy, which can serve as a reference point for future studies when new datasets with other participants need to be recorded. Furthermore, it examines how the accuracy changes when the models are trained across or within subjects, which brings the difference between the EEG signals of these subjects

into context. Especially the across subjects analysis shows how well a trained model is able to generalize to completely new subjects. In contrast, training the models within subjects serves as an indicator of the quality of a participant's data. Finally, it investigates how all that relates to the complexity of the deep neural networks with respect to the number of parameters, runtime, depth of the architecture, etc.

This thesis includes four chapters: besides the introductory Chapter 1, Chapter 2 covers the structure of the dataset and used deep learning models. Chapter 3 presents the results of the dataset analysis and Chapter 4 concludes this thesis by discussing the work and giving a brief outlook.

Chapter 2

Dataset and Models

This chapter presents the dataset and used deep learning models.
In Section 2.1, we describe how the data was acquired in order to understand where it comes from, which measurements were taken and how they are structured.
Section 2.2 outlines the process of preparing this dataset to be able to use it in a benchmark with machine learning techniques. Moreover, some statistics about the particular dataset, which was then used for the study in this thesis, are provided as well.
Section 2.3 explains the benchmark task. Furthermore, the different deep neural networks used in this study are illustrated to give an overview of the architectures and hyperparameters.
For further information about the data acquisition and preprocessing, please refer to [6].

2.1 Data Acquisition

The dataset examined in this study originates from existing simultaneously recorded EEG and infrared video-based eye-tracking data of 364 participants. In the acquisition setup, the EEG system was composed of 129 electrodes with a sampling rate of 500 Hz (i.e. one measurement per 2 ms). The infrared video-based eye-tracking data was used to provide a ground truth information for the later analysis.
The pro- and antisaccade paradigm was the main task, based on the internationally standardized protocol for antisaccade testing developed by Antoniades et al. [7]. In short, the participants were asked to focus on the center of the screen. Subsequently, the cue (i.e. dot) appeared on the left or right hand-side. During the prosaccade task, they needed to focus their gaze on the cue as fast as possible. Conversely, during the antisaccade task, they were asked to perform a saccade in the opposite side of the cue. In both tasks, the cue was shown for 1 second, and as soon as it disappeared, the participants shifted their focus back to the center of the screen. Executing one such pro- or antisaccade is what we

Event	mean	std	Total
Saccades	1'094	326	403'678
Fixations	1'090	326	402'257
Blinks	175	121	64'701

Table 2.1: Overview of the dataset.

will refer to as a trial in the following.
The two tasks were executed in blocks of the following order: 1. Prosaccade, 2. Antisaccade, 3. Antisaccade, 4. Antisaccade, 5. Prosaccade. A prosaccade block was made up of 60 trials, while an antisaccade block consisted of 40 trials. The trials in each block were balanced and randomized, such that in half of the trials the cue appeared on the left side and in the other half on the right side.

For each saccade, several parameters were measured: besides the latency and duration, also the saccade's amplitude, maximum velocity, as well as start and end position with respect to the x- and y-coordinates. In addition to the saccade itself, i.e. the direction of the gaze, fixations and blinks were also determined. A fixation is when the participant focused a stimulus, i.e. the cue, and a blink is when the eye was rapidly opened and closed. For each fixation, the latency, duration, average position in x- and y-coordinates and the average pupil size were measured, while for each blink, only the latency and duration.
This resulted in a very rich dataset, consisting of millions of measurements. An overview of the size of this dataset together with the mean and standard deviation of these events among all participants is provided in Table 2.1.

2.2 Dataset Preparation

The dataset collected from the simultaneous EEG and eye-tracking measurements in the pro- and antisaccade paradigm is very large. In order to use this data for a benchmark in machine learning, it is required to prepare the dataset in the right shape given a certain benchmark task. For instance, one could be interested in the fixations instead of the saccades, such that the recorded EEG data precisely at the time steps where these specific events occur needs to be collected.
Part of the work in this thesis was contributing to such a framework for data preparation. The goal of this framework is to be a tool to increase the quality of the dataset in the sense that other students, researchers and engineers can easily use it for their own benchmarks. Additionally, it should be suitable for the preparation of newly recorded datasets in the future.
In particular, the configurability and various design decisions of the framework were improved so that the user can specify the desired blocks (pro- or antisaccade tasks) to cut from the measured data and apply different filters, e.g. to exclude trials where the saccade comes too soon, too late or when its amplitude is below

a certain threshold.
This will support further development of a benchmark repository for saccade's direction. All code is available on our website[1].

The dataset study in this thesis focused on the prosaccade task. Hence, after preparing the data and filtering out all error trials, e.g. where the saccade happened in the wrong direction, the dataset comprised 357 participants, which resulted in a total of 36'109 samples. Each sample consisted of 129×500 data points, corresponding to the 129 electrodes of the EEG system and the sampling rate of 500 Hz. Based on the ground truth information from the infrared video-based eye-tracking data, each sample was mapped either to 0 (for looking left) or 1 (for looking right). Table 2.2 provides an overview of this dataset, in particular showing the mean and standard deviation of the amount of samples per subject. Furthermore, one observes that the dataset is relatively balanced, i.e. the number of zeros and ones (for looking left and right, respectively) does not differ much.

mean	std	# of zeros	# of ones	Total
101	21	17'378	18'731	36'109

Table 2.2: Overview of the dataset.

2.3 Classification of Gaze Direction

The benchmark task in this study was the classification of gaze direction. Essentially, we had our prepared dataset with the EEG measurements as an input to our machine learning models. Then, based on this EEG data, we tried to predict the saccade's direction, i.e. whether a subject looked left or right, which involved two classes as outputs of our neural networks. Eventually, the predictions were compared to the ground truth mapping such that the model could learn and we could gain insights about the performance, e.g. through the accuracy.

For the classification of the gaze direction, 80% of the samples in the dataset were used for training and 20% for validation. The structural details of the training and validation sets for the various analyses are described in the relevant sections of Chapter 3.

In this study, three different state-of-the-art deep neural network (DNN) architectures were deployed: InceptionTime [5], EEGNet [8], and Pyramidal Convolutional Neural Network (PyramidalCNN) [9]. As outlined in [6], all models were trained in an ensemble composed of 5 identical architectures. The major votes out of 5 classifications from each ensemble were used for the final classification. The values of the tuned hyperparameters of the mentioned architectures are specified in Chapter 3. All models were trained on the TIK Slurm cluster at

[1] https://gitlab.ethz.ch/kard/dl-project

ETH Zürich[2], in Tensorflow with the Keras API [10].
The following subsections serve as an overview of the different DNN architectures used in this study.

2.3.1 InceptionTime

This subsection about the InceptionTime network is based on the work published in [5].
InceptionTime is a novel architecture for time series classification. By default, it contains two residual blocks, each comprising three Inception modules, as depicted in Figure 2.1. After these residual blocks is a Global Average Pooling (GAP) layer to average the output time series over the whole time dimension. Finally, there is a fully-connected layer with the same number of output neurons as there are classes in the dataset.

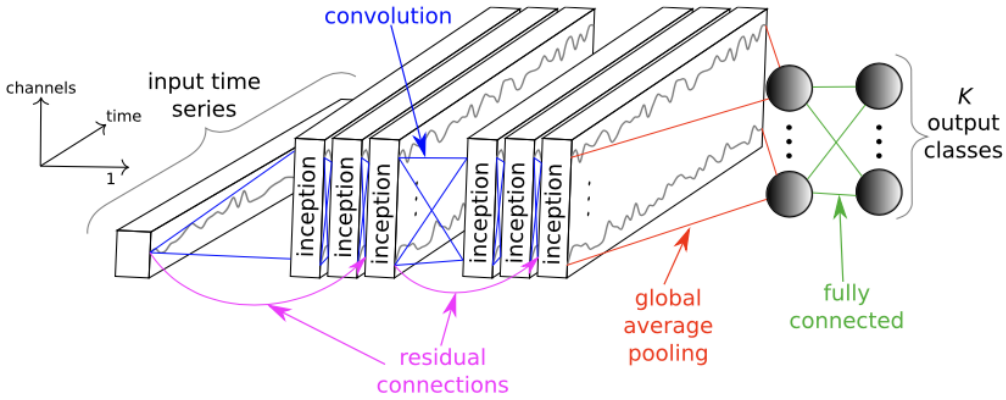

Figure 2.1: Architecture of the InceptionTime network, taken from [5].

In the Inception module mentioned above, the first component is the bottleneck layer, which considerably reduces the amount of dimensions of the input time series. Then, multiple filters of varying lengths are applied on this output time series of the bottleneck layer. This is illustrated in Figure 2.2 with convolutions of length 10, 20 and 40. In parallel, a MaxPooling operation is employed on the initial input time series followed again by a bottleneck layer, which takes the maximum value of its given window, with the goal of making the model invariant to small perturbations. Lastly, the final output is the concatenation of all outputs of the parallel convolutions and MaxPooling, as observable also in Figure 2.2.

[2] https://computing.ee.ethz.ch/Services/SLURM-tik

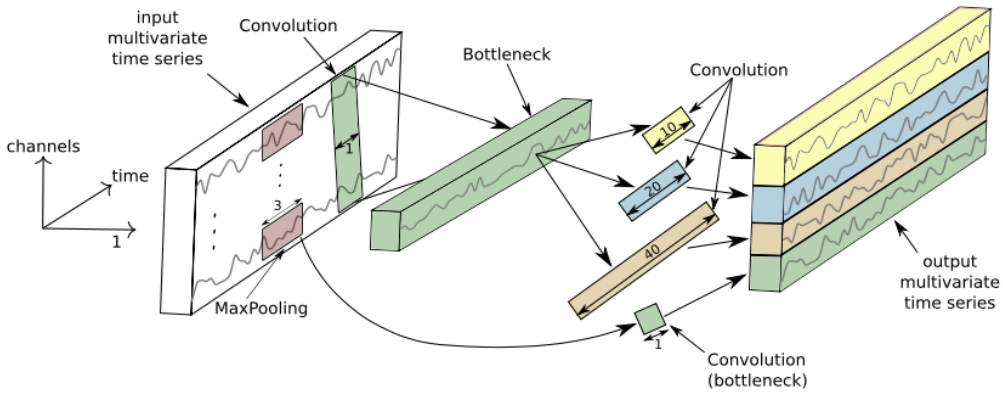

Figure 2.2: Inception module with a bottleneck size of 1 and kernel sizes 10, 20 and 40, taken from [5].

The following gives an overview of the hyperparameters of the InceptionTime network, which were tuned in this study:

Bottleneck size mainly reduces the number of parameters in the network for shorter runtime.

Depth determines the number of layers in the architecture. In principle, deeper networks are more accurate for longer time series.

Kernel size determines the length of the various filters applied in the Inception module. A longer kernel is able to capture longer patterns in the data, which often improves the accuracy.

Number of filters relates to the filters applied on the output of the bottleneck layer in the Inception module. Using too many filters substantially increases the complexity of the network.

2.3.2 EEGNet

This subsection about EEGNet is following the work published in [8].
The architecture of EEGNet is depicted in Figure 2.3 and is composed of several convolutions: The first one is a temporal convolution to learn frequency filters. Afterwards, it uses a depthwise convolution in order to learn frequency-specific spatial filters. Finally, a separable convolution combines a depthwise convolution to learn a temporal summary for each feature map with a pointwise convolution to learn how to optimally mix the feature maps together.

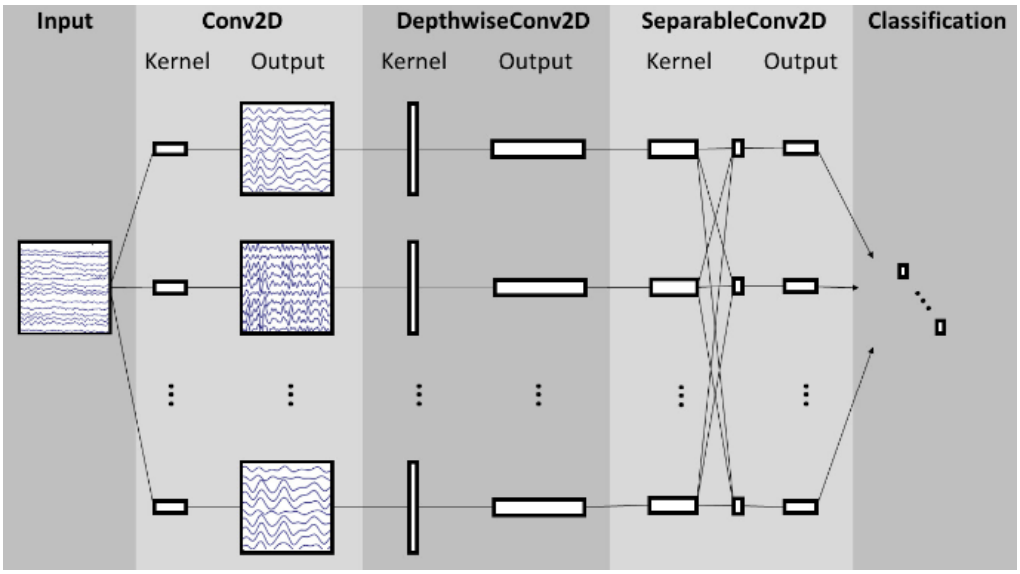

Figure 2.3: EEGNet architecture with depth D = 1, taken from [8].

These are the hyperparameters of EEGNet, which were tuned in this study:

Number of temporal filters (F1) relates to the first convolution illustrated in Figure 2.3.

Kernel length determines the length of the temporal kernel.

Depth parameter (D) regulates the number of spatial filters to learn for each feature map.

Dropout rate shuts down a fraction of the neurons, which is a regularization technique to avoid overfitting.

Number of pointwise filters (F2) affects the separable convolution. Its purpose is to reduce the number of parameters to fit and to decouple the relationship of the feature maps.

2.3.3 PyramidalCNN

The Pyramidal Convolutional Neural Network (PyramidalCNN) [9] is based on a traditional CNN model, but implemented in a pyramidal structure, i.e. for each depth, the number of filters is increased. Each layer is composed of a simple convolution with batch normalization and a ReLU activation, followed by MaxPooling.

The following gives an overview of its hyperparameters, which were tuned in this study:

Depth determines the number of layers in the architecture. It has a great influence on the number of parameters in the network due to the pyramidal approach.

Kernel size determines the length of the filters applied in the convolution.

Number of filters corresponds to the amount of filters used in the first layer. In each subsequent layer, the number of filters used equals the specified value multiplied with the current depth.

Chapter 3

Results

This chapter describes our various analyses about the size and quality of the EEG dataset and presents the results.
In Section 3.1, we explore how different numbers of subjects impact the accuracy, which we call "across subjects v1 analysis" in the following. This can serve as a reference point for future studies when new datasets with other participants need to be recorded. We investigated the InceptionTime network [5], as it performed best in [6], and implemented a way to specify the percentage of training data to be actually used for training. Finally, we also showed how this relates to the complexity of the deep neural networks with respect to the number of parameters, runtime, depth of the architecture, etc.
In Section 3.2, we examine how the accuracy changes when the models are trained across subjects such that all samples of a particular subject are either in the training or validation set, but not in both. This, what we will call "across subjects v2 analysis", shows how well a trained model is able to generalize to completely new subjects. For comparison, we again used the InceptionTime model and studied the same amounts of training data.
In Section 3.3, we trained and predicted only with one subject at a time, which we refer to as "within subjects analysis" in the following. This serves as an indicator of the quality of a participant's data. Since there were a lot less samples available, we used simpler models, in particular EEGNet [8] and PyramidalCNN [9]. Additionally, we also analyzed simple classifiers from scikit-learn [11].

3.1 Across Subjects v1 Analysis

In the across subjects v1 analysis, the dataset was shuffled before splitting it into a training and validation set. This means we trained and predicted with multiple subjects and it didn't matter if there were samples from the same subject in both the training and validation set.

An overview of the different configurations of the InceptionTime model is provided in Table 3.1. The hyperparameters of the model for lower amounts of data, i.e. for 20-60% of training data, were tuned with Keras Tuner [12]

% of data	Depth	# of parameters	Runtime [min]	Accuracy	Loss
0.2	12	577'841	150	0.882	0.300
0.4	12	577'841	160	0.931	0.192
0.6	12	577'841	270	0.941	0.164
0.8	30	2'150'721	410	0.940	0.165
1.0	40	1'343'921	460	0.941	0.158
1.0	12	395'425	160	0.948	0.137

Table 3.1: Overview of the different configurations of the InceptionTime model.

using random search on 40% of the data. For larger amounts of data, i.e. 80% and above, manual hyperparameter tuning was done investigating also deeper networks. There, we discovered that a depth of 50 didn't provide increased stability compared to depths between 30 and 40, while the kernel size and the number of filters shouldn't be set below 32 and above 32, respectively. Using a bottleneck size greater than 16 didn't really affect the accuracy, but had a large increasing impact on the number of parameters and the runtime, as described in [5]. Finally, 100% of training data was also examined on the network suggested by Kastrati and Plomecka, et al. in [6], which had very similar hyperparameters as the networks used for lower amounts of data.

The result with the corresponding hyperparameters for 20% training data is depicted in Figure 3.1. While the achieved accuracy (88.2%) is already pretty stable and reaches a large value after very few epochs (below 10), one observes that using 40% training data on the same model results in a substantially higher accuracy (93.1%), greater stability and faster increase in accuracy, as shown in Figure 3.2. In contrast, the run with 60% training data on the same model, shown in Figure 3.3, has a much smaller increase in accuracy (94.1%) and similar overall stability.

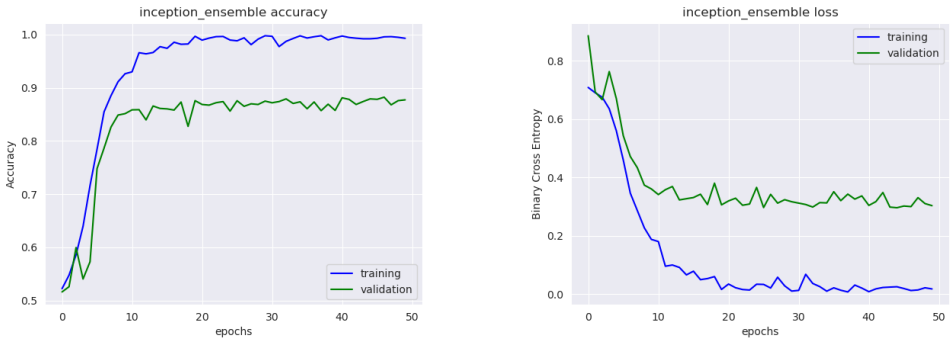

Figure 3.1: InceptionTime model with 20% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

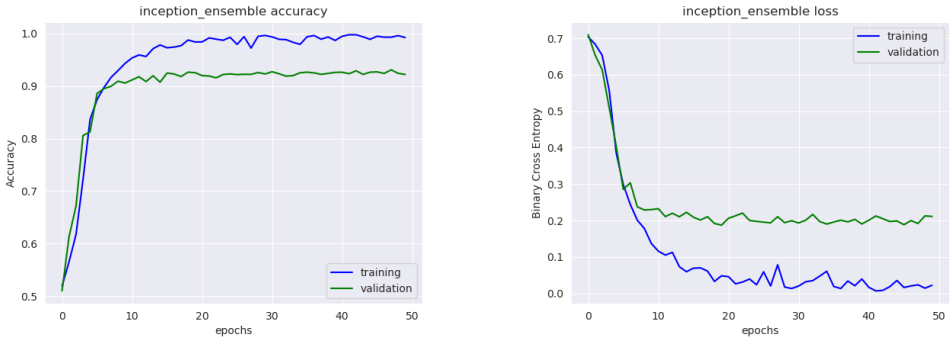

Figure 3.2: InceptionTime model with 40% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

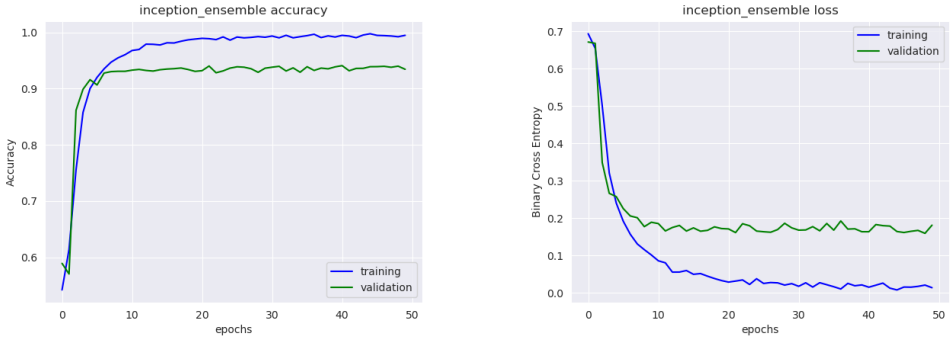

Figure 3.3: InceptionTime model with 60% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

Using 80% and 100% training data on much deeper networks, as depicted in Figure 3.4 and 3.5, respectively, exhibits very similar results in terms of accuracy and stability compared to the trainings with 40% and 60% of data. Finally, using 100% training data on a model with depth 12, shown in Figure 3.6, outperforms all previous configurations with a remarkable accuracy of 94.8%, nearly matching the one obtained in [6].

Therefore, we conclude that for 80% of training data and above, using a deeper model provided good stability, but also using a model with shorter depth similar to the runs with less data provided equally good results, while having fewer hyperparameters and much shorter runtime. Another main insight from these results is that 40% of training data sufficed for good predictions, as we already reached an accuracy of 93% and above, which didn't increase substantially for larger amounts of data.

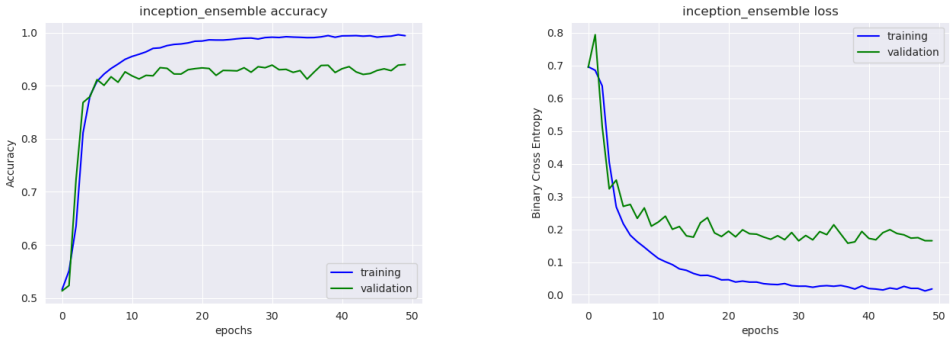

Figure 3.4: InceptionTime model with 80% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=30, bottleneck_size=32, batch_size=64.

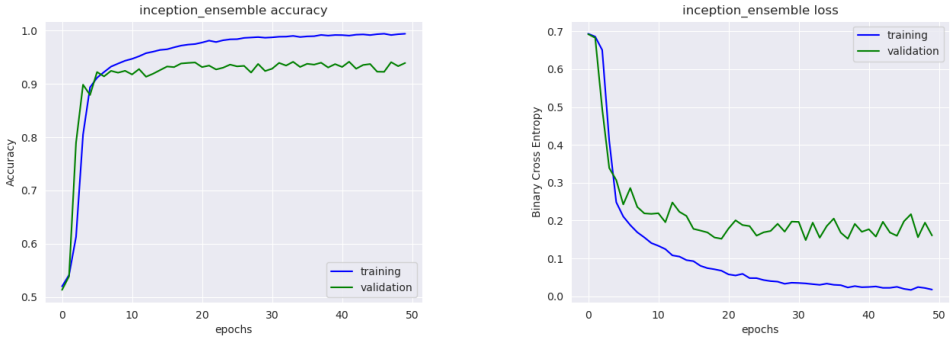

Figure 3.5: InceptionTime model with 100% training data and use_residual=True, kernel_size=32, nb_filters=16, depth=40, bottleneck_size=32, batch_size=64.

As a side note, a very similar network to the ones above with a depth of 12 was run on a downsampled dataset, which had only 1/4 many data points. The results with the corresponding hyperparameters tuned again with Keras Tuner on 40% of the data are shown in Appendix A.1. Overall, we noticed that using only 40% of this data sufficed for good predictions as well. Using 80% and more data resulted in instabilities of the network, which were most likely due to the network being tuned on only 40% of the data and thus having a too simple architecture for very large amounts of training data. This led to the analysis of deeper networks as described above with manual hyperparameter tuning, since the Keras Tuner only favors accuracy but not stability.

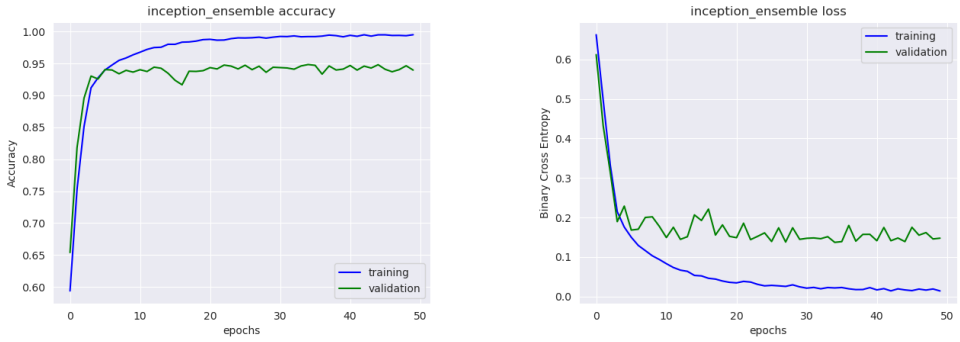

Figure 3.6: InceptionTime model with 100% training data and use_residual=True, kernel_size=64, nb_filters=16, depth=12, bottleneck_size=16, batch_size=64.

3.2 Across Subjects v2 Analysis

In the across subjects v2 analysis, the dataset was divided into two subsets in a ratio of approximately 80:20, where all trials of a subject were either in the training or validation set, but not in both. This ensured that all data in the validation set was from completely new subjects which were not used for training before.

An overview of the different configurations of the InceptionTime model is provided in Table 3.2, where nearly the same models as in Section 3.1 were analyzed.

% of data	Depth	# of parameters	Runtime [min]	Accuracy	Loss
0.2	12	577'841	110	0.872	0.313
0.4	12	577'841	105	0.901	0.356
0.6	12	577'841	140	0.921	0.234
0.8	30	2'150'721	680	0.937	0.159
0.8	12	395'425	130	0.937	0.166
1.0	40	728'481	385	0.942	0.157
1.0	12	395'425	145	0.943	0.141

Table 3.2: Overview of the different configurations of the InceptionTime model.

The result with the corresponding hyperparameters for 20% training data is depicted in Figure 3.7. Overall, it is very similar to the same configuration from the across subjects v1 analysis (Figure 3.1), but has a slightly lower accuracy (87.2%) and stability and requires a bit more epochs to reach a good accuracy. Surprisingly, using 40% training data resulted in a very unstable network, shown in Figure 3.8. Conversely, the run with 60% training data, depicted in Figure

3.9, again exhibits a more stable behavior, though still worse compared to the one in the across subjects v1 analysis (Figure 3.3).

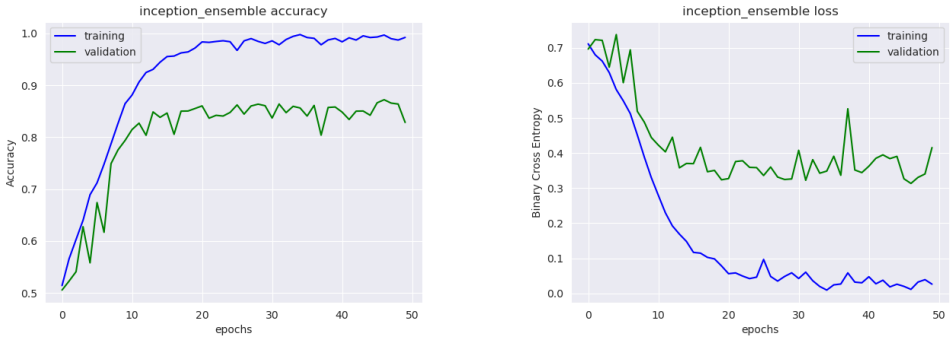

Figure 3.7: InceptionTime model with 20% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

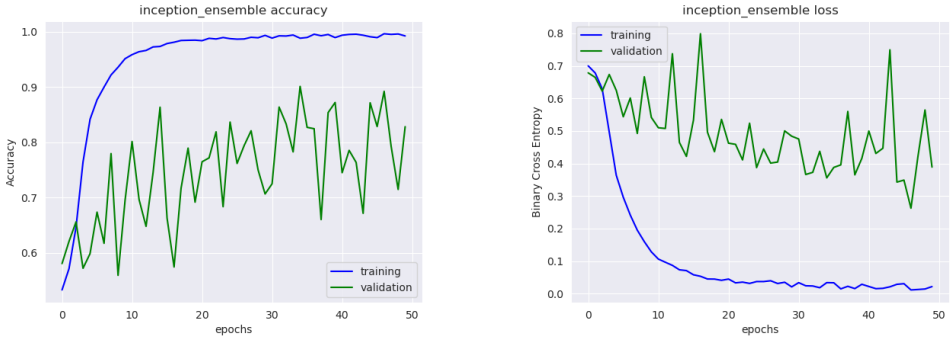

Figure 3.8: InceptionTime model with 40% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

Using 80% and 100% training data on both smaller networks with depth 12 and deeper ones with depth 30 and 40 resulted in good accuracy and stability with negligible differences between these configurations, which are presented in Figures 3.10, 3.11, 3.12 and 3.13. In comparison to the runs in Section 3.1, they have a similar performance in terms of accuracy and stability. Again, we noticed that using 100% training data on a model with depth 12 gave the best accuracy (94.3%).

One observes that more training subjects are required compared to the across subjects v1 analysis such that the model is able to generalize well to completely new subjects. In particular, having at least 80% of the training data provided satisfying results, while for 60% and below, the network was too unstable. Again as seen in the across subjects v1 analysis, using much deeper networks with

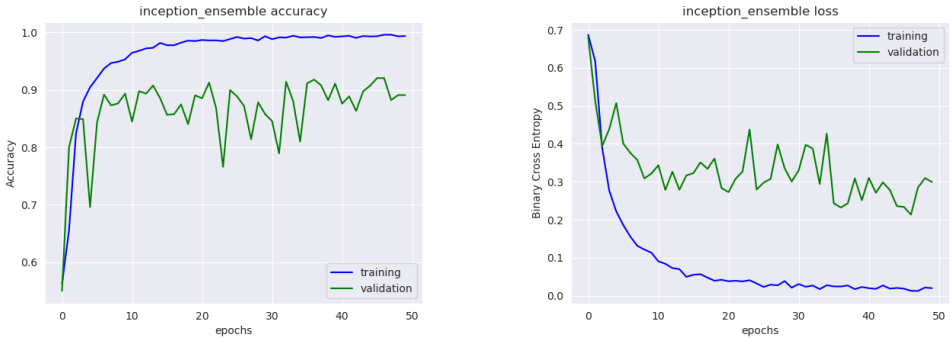

Figure 3.9: InceptionTime model with 60% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

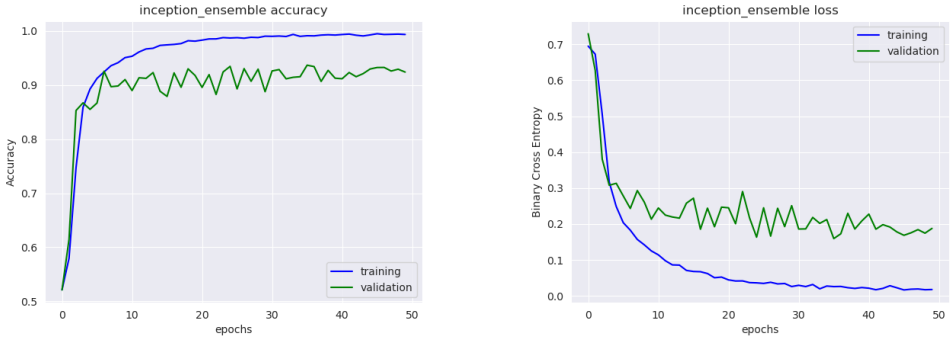

Figure 3.10: InceptionTime model with 80% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=30, bottleneck_size=32, batch_size=64.

depths between 30 and 40 did not provide a reasonable benefit compared to shorter networks with depth 12. However, the discrepancy in stability between using 20% and 40% training data required further research as outlined below. My guess was that these 20% training subjects were particularly well suited for the given validation set. In other words, with 40% data we may have included subjects with significantly differing EEG signals compared to the ones in the validation set.

To investigate the discrepancy in the stability between the runs with 20% and 40% of training data, two approaches were tested.

The first one was to change the order of shuffling and cutting the dataset. Up until now, the training data was first shuffled and then cut according to the specified percentage, e.g. 20%. This first case could be interpreted as still having a lot of different subjects but fewer trials per subject. The second, newly

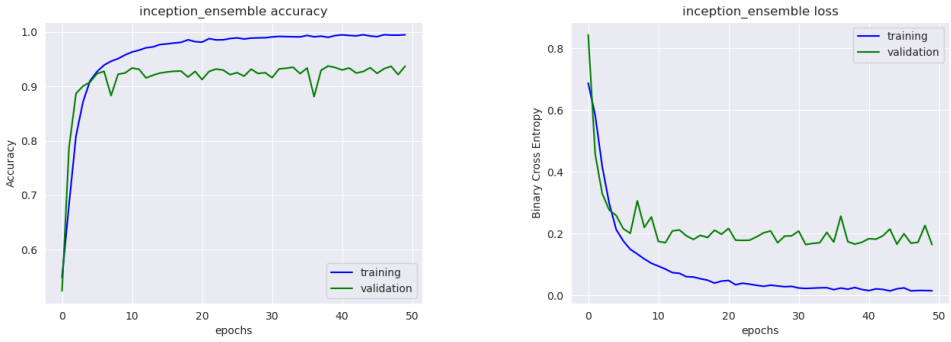

Figure 3.11: InceptionTime model with 80% training data and use_residual=True, kernel_size=64, nb_filters=16, depth=12, bottleneck_size=16, batch_size=64.

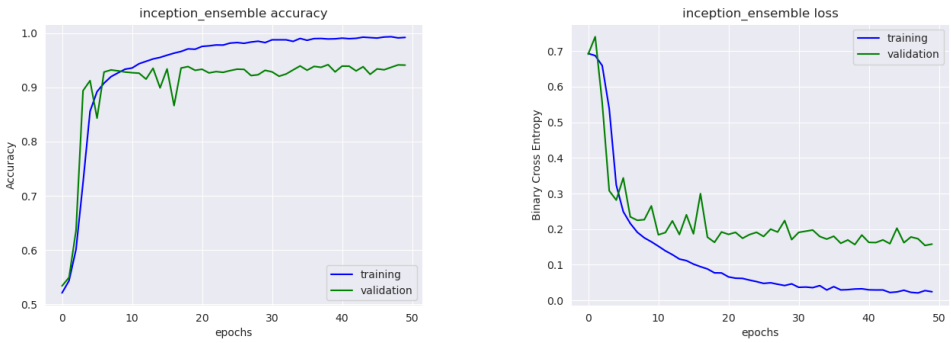

Figure 3.12: InceptionTime model with 100% training data and use_residual=True, kernel_size=32, nb_filters=16, depth=40, bottleneck_size=16, batch_size=64.

investigated case was first cutting the training data and afterwards shuffling it, which could be interpreted as having fewer subjects but still a lot of trials per subject. The results of this second case for 20%, 60% and 80% of training data with the same hyperparameters as used before in the first case are depicted in Figures 3.14, 3.15 and 3.16. Interestingly, they seem to have a qualitatively similar behavior compared to the first case, i.e. using 20% and 80% training data exhibited stable behavior with good accuracy, while using 40% data was very unstable. The fact that it did not matter much whether the samples came from few or many subjects indicates that our models are able to generalize well.

The second approach was to change the validation set. Up until now, the last 20% of the dataset was always used as a validation set. Therefore, we tried using the first 20% of the dataset as a validation set. The results of these runs for the same amounts of training data and the same hyperparameters are depicted in

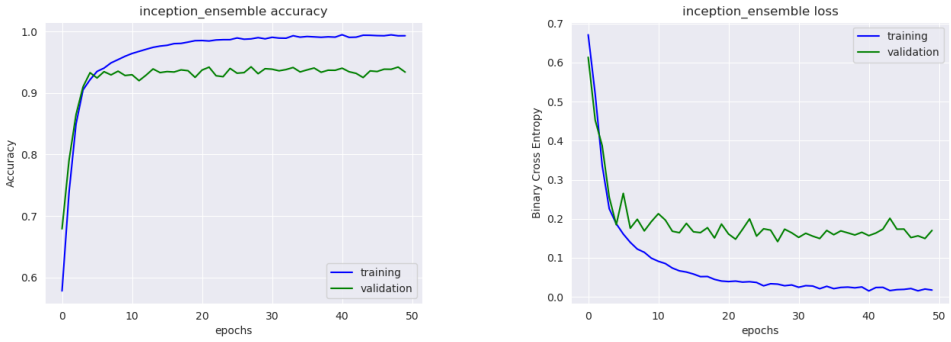

Figure 3.13: InceptionTime model with 100% training data and use_residual=True, kernel_size=64, nb_filters=16, depth=12, bottleneck_size=16, batch_size=64.

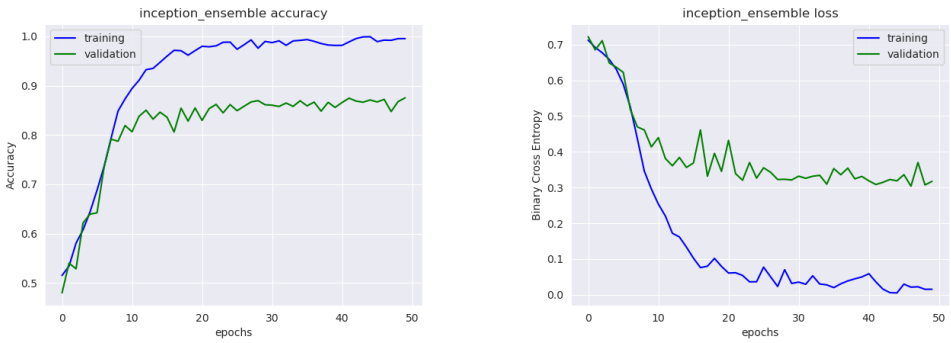

Figure 3.14: InceptionTime model with 20% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

Figures 3.17, 3.18 and 3.19. One observes that the chosen validation set made a substantial difference in accuracy, especially for lower amounts of data. In particular, note that the run with 40% training data (Figure 3.18) was now very stable. Hence, we concluded that with few training data, one could be lucky and get good accuracy, but with more training data it is safer that the model generalizes well to new data and thus gets good accuracy.

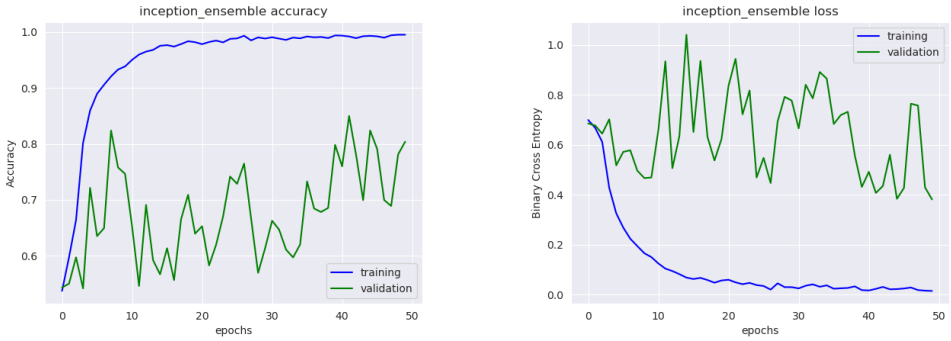

Figure 3.15: InceptionTime model with 40% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

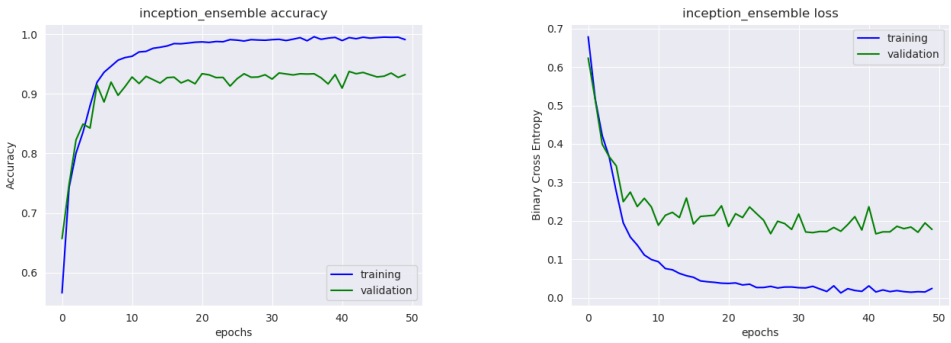

Figure 3.16: InceptionTime model with 80% training data and use_residual=True, kernel_size=64, nb_filters=16, depth=12, bottleneck_size=16, batch_size=64.

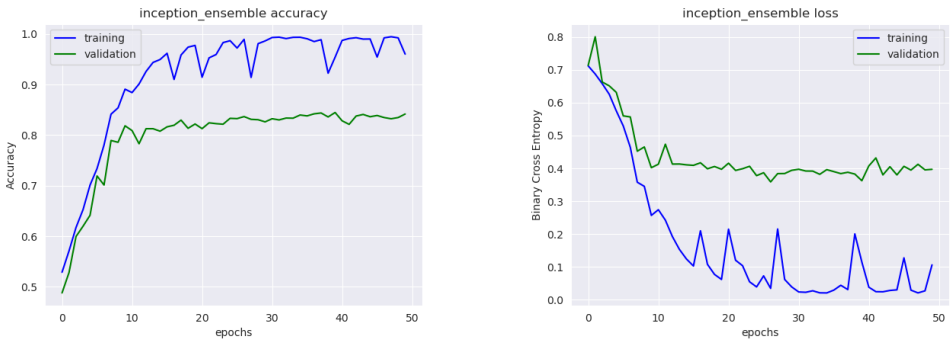

Figure 3.17: InceptionTime model with 20% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

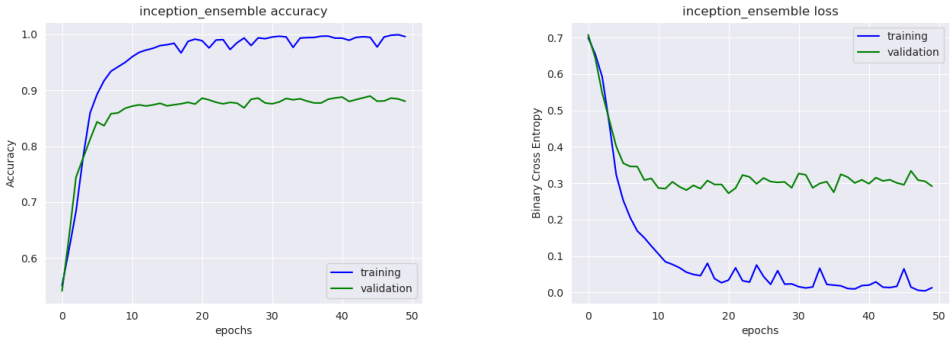

Figure 3.18: InceptionTime model with 40% training data and use_residual=True, kernel_size=40, nb_filters=32, depth=12, bottleneck_size=16, batch_size=64.

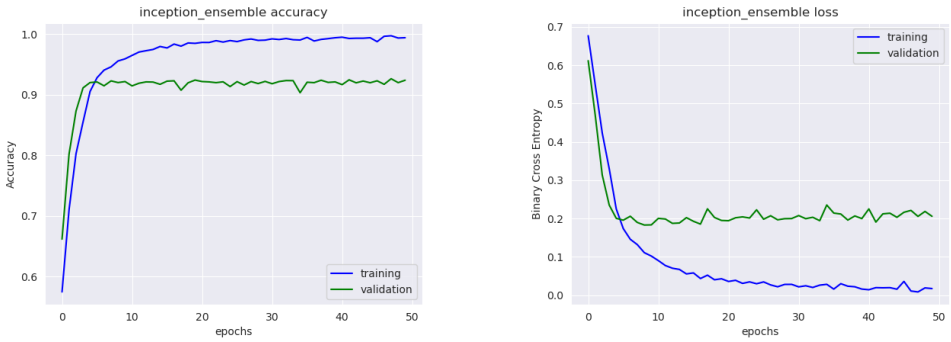

Figure 3.19: InceptionTime model with 80% training data and use_residual=True, kernel_size=64, nb_filters=16, depth=12, bottleneck_size=16, batch_size=64.

3.3 Within Subjects Analysis

In the within subjects analysis, only one subject was used for both training and validation. This showed the quality of a particular subject (in a deep learning sense). Since there was just one subject for training and validation, much smaller models were used.

First, we tried out EEGNet with manual hyperparameter tuning. The results of some exemplary subjects together with the configured hyperparameters are depicted in Figures 3.20 and 3.21.

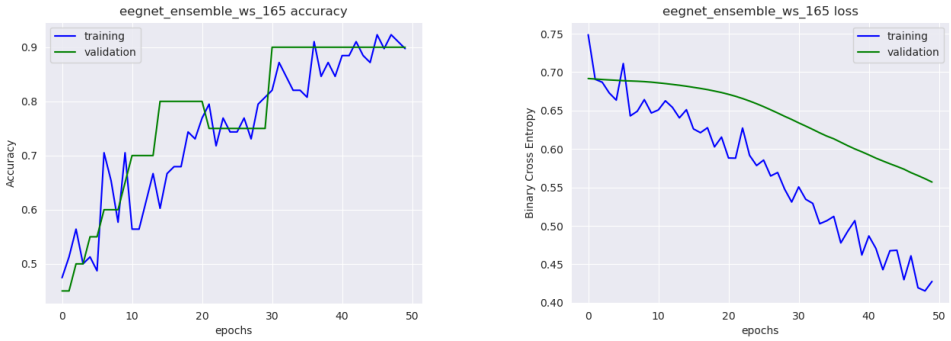

Figure 3.20: EEGNet model with subject ID 165 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

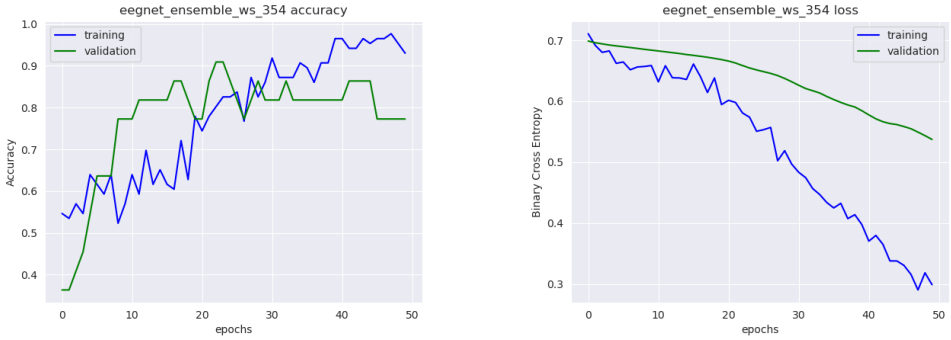

Figure 3.21: EEGNet model with subject ID 354 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

Second, PyramidalCNN was also tested, where we tried depths of around 4, in order to allow the network to learn more abstract features. Consequently, we fixed the kernel size in the first layer to just 1, such that the number of parameters did not increase too much, while leaving the kernel sizes of the other layers according to the specified hyperparameter. Some exemplary results and

hyperparameters of PyramidalCNN are shown in Figures 3.22 and 3.23. Results of additional subjects of the two models are provided in Appendices A.2 and A.3, respectively.

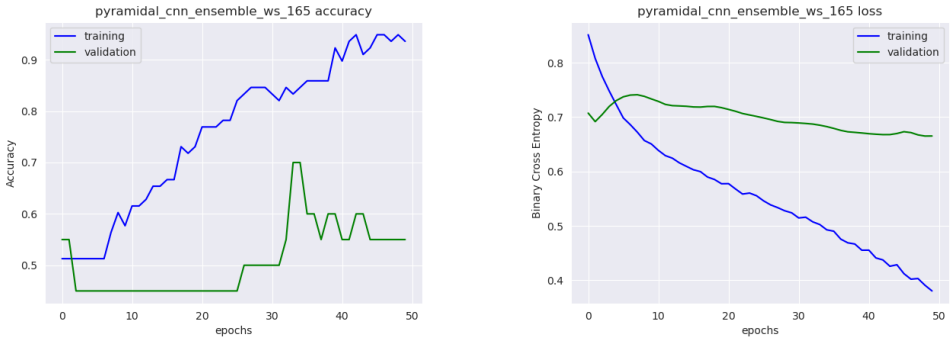

Figure 3.22: PyramidalCNN model with subject ID 165 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.

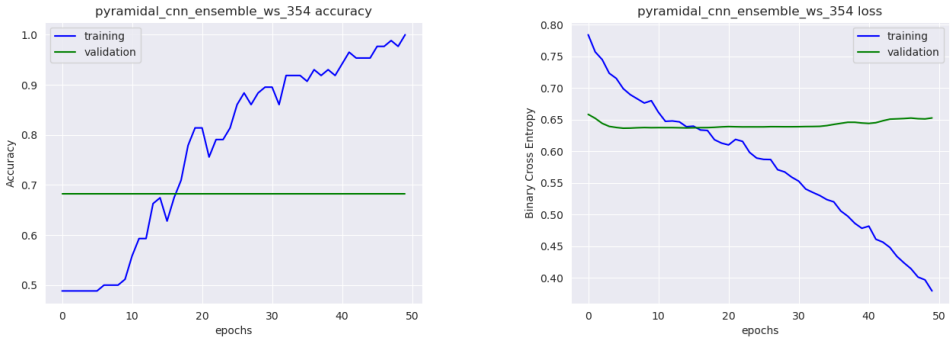

Figure 3.23: PyramidalCNN model with subject ID 354 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.

We concluded that EEGNet performed much better than PyramidalCNN in the within subjects analysis, reaching an accuracy of 80% to 90% for most subjects. Especially, one observes that in the runs with EEGNet, there was no large difference in accuracy between the training and validation sets, i.e. the model was actually learning. This was according to our expectations, as the EEGNet model was created precisely for EEG data.

Moreover, it is evident that these runs had much larger steps in accuracy compared to the runs in the across subjects analysis, notably for the validation set. This was due to only analyzing one subject per run, e.g. if a particular subject had only 20 trials in its validation set, each correct prediction raised the accuracy by $\frac{1}{20} = 5\%$.

Furthermore, we also ran simpler classifiers from scikit-learn [11]. In particular, RandomForestClassifier[1] and LinearSVC[2] were tested, where we determined the mean and the standard deviation of the scores among all subjects. The results of these two classifiers with the configured parameters are shown in Table 3.3 and 3.4, respectively. Note that they were not able to outperform the previous runs with EEGNet.

Estimators	Max depth	Score mean	Score std	Runtime [sec]
30	20	0.627	0.146	5247
15	20	0.604	0.135	4950
30	10	0.627	0.146	5723
15	10	0.604	0.135	4947
60	20	0.646	0.145	5284
30	40	0.627	0.146	4936
60	40	0.646	0.145	5016

Table 3.3: Overview of the different configurations of the RandomForestClassifier.

C	Score mean	Score std	Runtime [sec]
0.1	0.697	0.144	5046
1	0.697	0.144	5056
10	0.697	0.144	4866

Table 3.4: Overview of the different configurations of the LinearSVC with tol=1e-4 and max_iter=500.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Chapter 4

Conclusion

The findings of the across subjects v1 analysis suggest that using 40% of the available training data, i.e. around 11'500 samples, already suffices for allowing the InceptionTime model to make very good left/right gaze predictions with an accuracy above 93%. Using 2.5 times as many samples, i.e. 100% of the training data, only leads to a marginal increase of less than 2% in accuracy.
Furthermore, we discovered that a model with depth 12, as proposed in [6], offers a proper balance between accuracy and complexity, and that even deeper networks do not provide a reasonable advantage.

In the across subjects v2 analysis, the insights suggest that the choice of the validation set can have a large impact on the accuracy and stability of the network when training with lower amounts of data. Generally, it seems that using at least 80% of the available training data, i.e. around 23'000 samples, is needed to reliably predict left/right gazes of completely new subjects.
Moreover, analogously to the across subjects v1 analysis, we observed that depths of 12 also suffice for this setting.
Finally, we noticed that changing the order of shuffling and cutting the training set to a desired percentage had a negligible influence on the validation accuracy and loss, which hints at a good generalizability of our models.

The findings of the within subjects analysis indicate that the EEGNet model seems best suited for this task, reaching an accuracy between 80% and 90% for most subjects. In comparison, PyramidalCNN and also simpler classifiers like Random Forest and Support Vector Machines performed worse. Feature extraction could potentially improve the accuracy of these simple classifiers. However, it was not applied in this study, since it can be very difficult and often requires understanding of the EEG data.

As new datasets are currently being recorded to extend the classification of saccade's direction beyond left and right, this thesis will provide a reference point for assessing the structure and quality of such new datasets in the future. The contributions to the framework for data preparation will allow other students, researchers and engineers to extend it and implement new filters tailored to their benchmarks and datasets.

In the longer-term, the project will result in an open-source toolbox, empowering EEG research and clinical institutions worldwide to objectively, reliably and time-efficiently assess gaze positions.

Bibliography

[1] E. Grant and M. Spivey, "Eye movements and problem solving: Guiding attention guides thought," *Psychological science*, vol. 14, pp. 462–6, 10 2003.

[2] M. X. Cohen, C. E. Elger, and C. Ranganath, "Reward expectation modulates feedback-related negativity and EEG spectra," *NeuroImage*, vol. 35, no. 2, pp. 968–978, 2007. [Online]. Available: https://doi.org/10.1016/j.neuroimage.2006.11.056

[3] J. Son, L. Ai, R. Lim, T. Xu, S. Colcombe, A. Franco, J. Cloud, S. Laconte, J. Lisinski, A. Klein, C. Craddock, and M. Milham, "Evaluating fmri-based estimation of eye gaze during naturalistic viewing," *Cerebral cortex (New York, N.Y. : 1991)*, vol. 30, 10 2019.

[4] Y. Roy, H. J. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: a systematic review," *CoRR*, vol. abs/1901.05498, 2019. [Online]. Available: http://arxiv.org/abs/1901.05498

[5] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Min. Knowl. Discov.*, vol. 34, no. 6, pp. 1936–1962, 2020. [Online]. Available: https://doi.org/10.1007/s10618-020-00710-y

[6] A. Kastrati, M. B. Plomecka, N. Langer, and R. Wattenhofer, "Using Deep Learning to Classify Saccade Direction from Brain Activity," in *Proceedings of the 13th ACM Symposium on Eye Tracking Research and Applications (ETRA), Short Papers, Virtual Event*, May 2021.

[7] C. Antoniades, U. Ettinger, B. Gaymard, I. Gilchrist, A. Kristjánsson, C. Kennard, R. Leigh, I. Noorani, P. Pouget, N. Smyrnis, A. Tarnowski, D. Zee, and R. Carpenter, "An internationally standardised antisaccade protocol," *Vision research*, vol. 84, March 2013.

[8] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: A compact convolutional network for eeg-based brain-computer interfaces," *CoRR*, vol. abs/1611.08024, 2016. [Online]. Available: http://arxiv.org/abs/1611.08024

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017. [Online]. Available: http://doi.acm.org/10.1145/3065386

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, K. Keeton and T. Roscoe, Eds. USENIX Association, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2078195

[12] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Keras Tuner," https://github.com/keras-team/keras-tuner, 2019.

Appendix A

Additional Results

A.1 Results from Downsampled Dataset

% of data	Accuracy	Loss
0.2	0.864	0.362
0.4	0.912	0.228
0.6	0.927	0.210
0.8	0.914	0.219
1.0	0.926	0.197

Table A.1: Overview of the different configurations of the InceptionTime model.

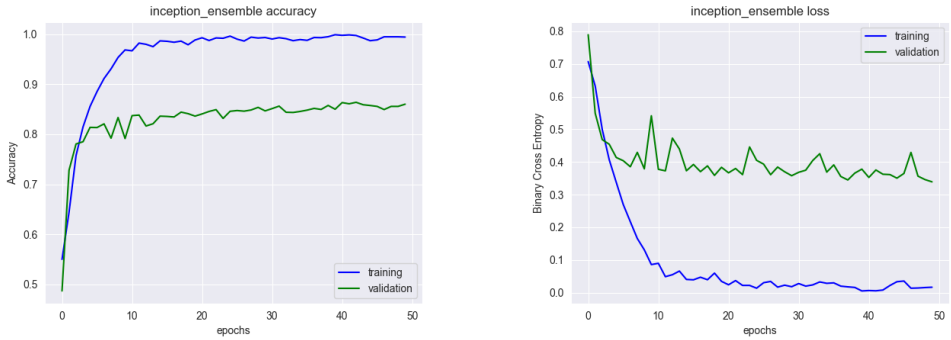

Figure A.1: InceptionTime model with 20% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=12, bottleneck_size=32, batch_size=64.

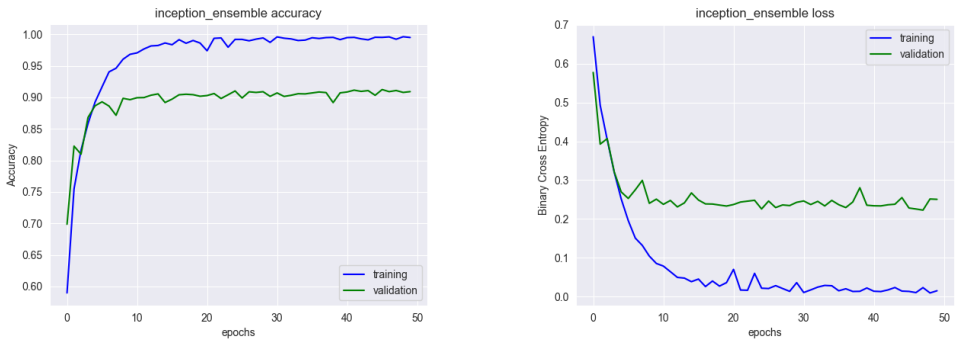

Figure A.2: InceptionTime model with 40% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=12, bottleneck_size=32, batch_size=64.

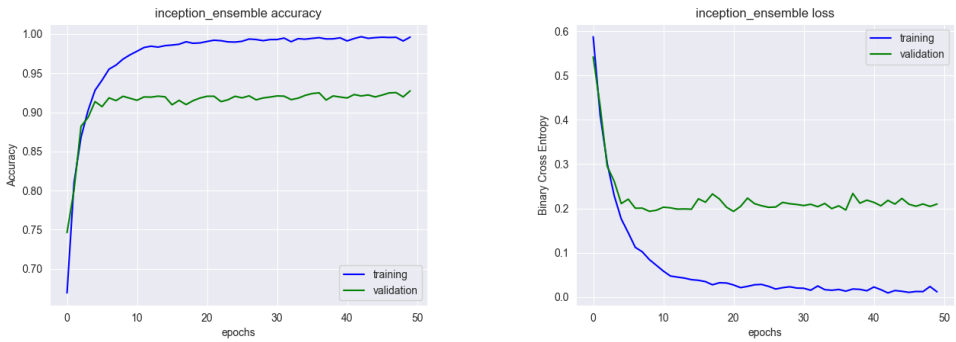

Figure A.3: InceptionTime model with 60% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=12, bottleneck_size=32, batch_size=64.

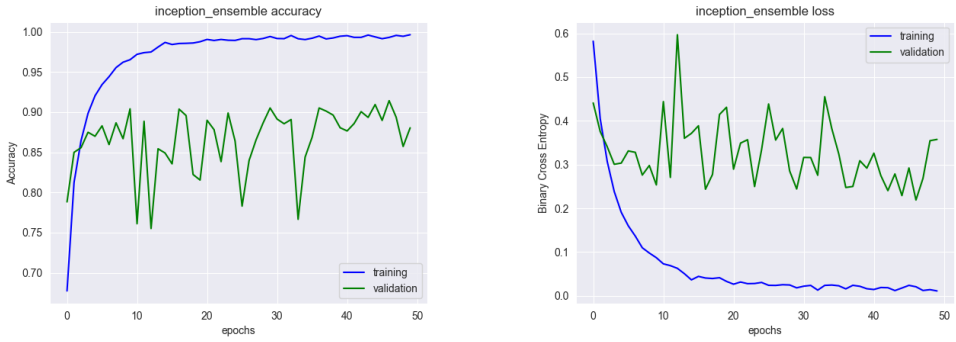

Figure A.4: InceptionTime model with 80% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=12, bottleneck_size=32, batch_size=64.

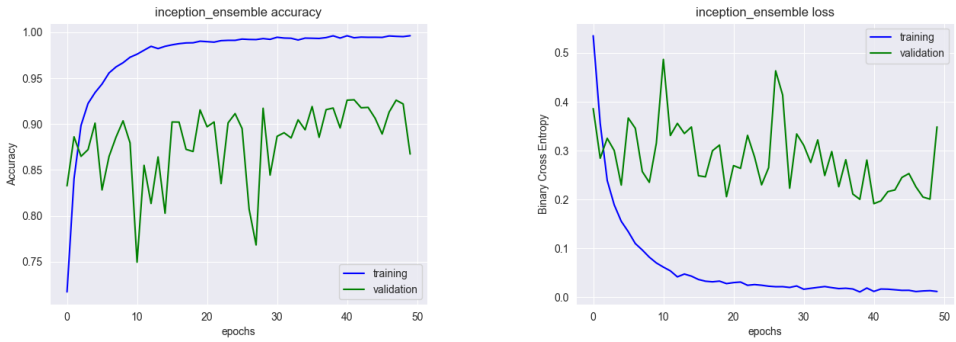

Figure A.5: InceptionTime model with 100% training data and use_residual=True, kernel_size=32, nb_filters=32, depth=12, bottleneck_size=32, batch_size=64.

A.2 Results from EEGNet

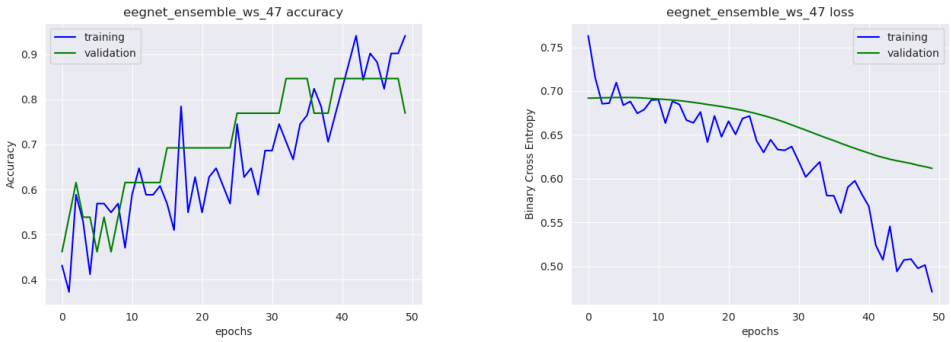

Figure A.6: EEGNet model with subject ID 47 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

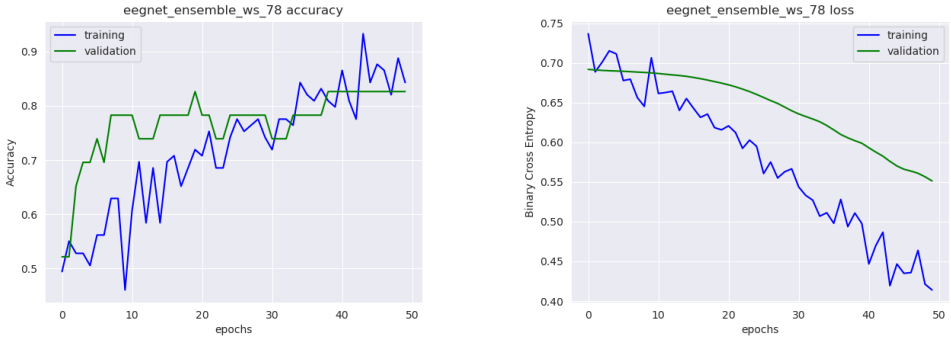

Figure A.7: EEGNet model with subject ID 78 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

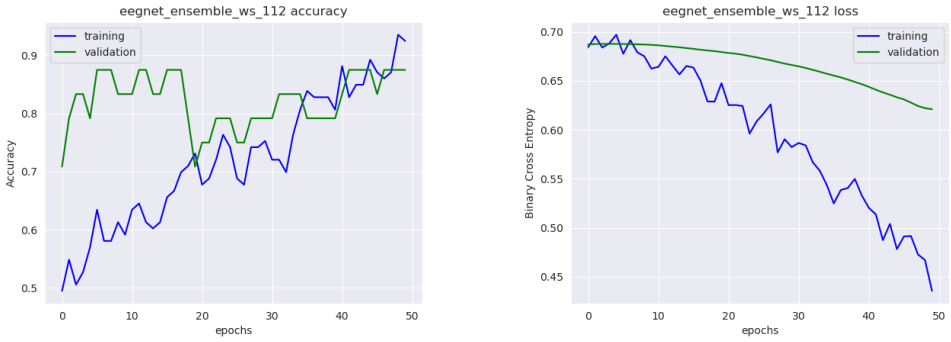

Figure A.8: EEGNet model with subject ID 112 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

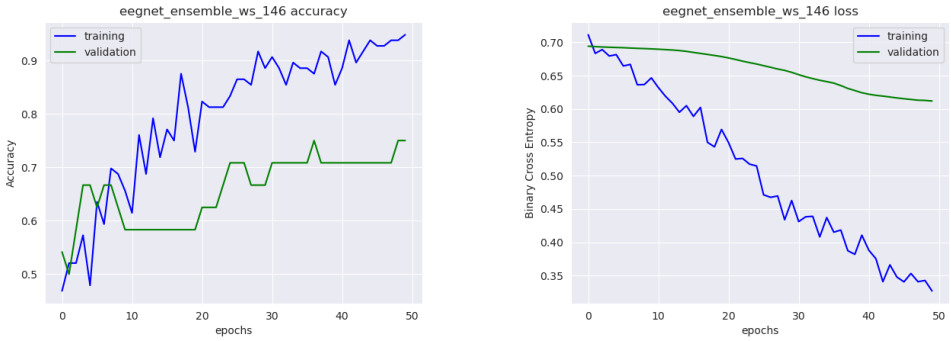

Figure A.9: EEGNet model with subject ID 146 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

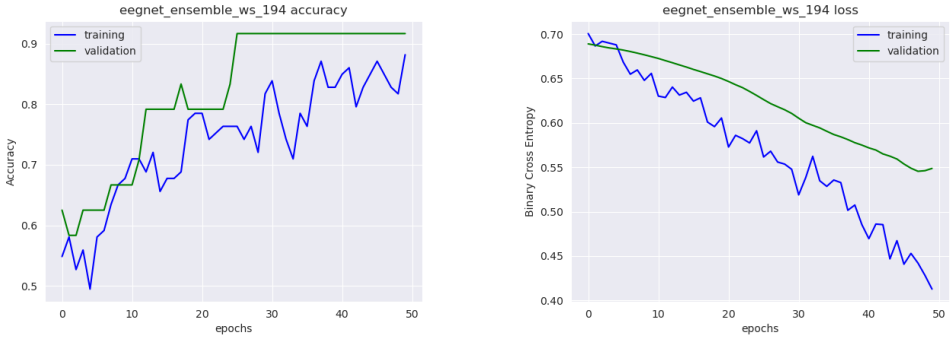

Figure A.10: EEGNet model with subject ID 194 and kernLength=16, F1=4, D=1, F2=8, dropoutRate=0.5, normRate=0.5.

A.3 Results from PyramidalCNN

Figure A.11: PyramidalCNN model with subject ID 47 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.

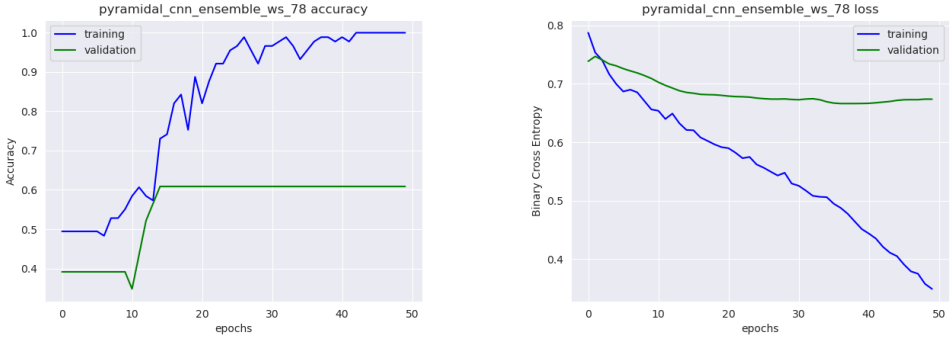

Figure A.12: PyramidalCNN model with subject ID 78 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.

Figure A.13: PyramidalCNN model with subject ID 112 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.

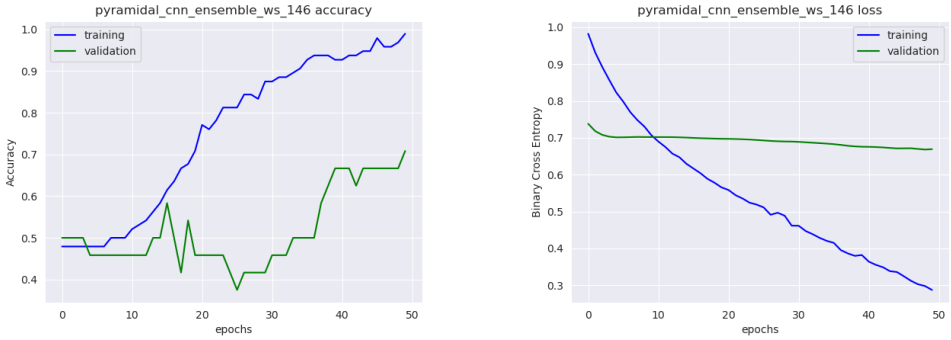

Figure A.14: PyramidalCNN model with subject ID 146 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.

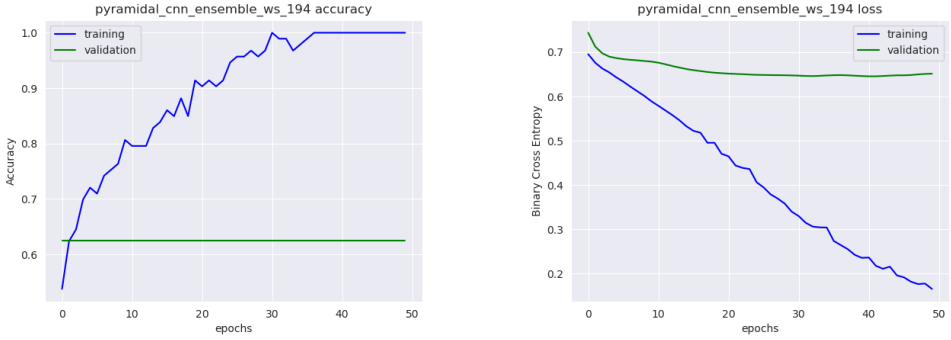

Figure A.15: PyramidalCNN model with subject ID 194 and kernel_size=8, nb_filters=4, depth=4, batch_size=64.