ECE C147 Final Report

Muhan Zhang UID: 405510018
Xuyang Zhou UID: 405571728
Yubo Zhang UID: 605816942
Yuxuan Qi UID: 305583072

Abstract

We investigated the application of various neural network architectures to EEG data classification, focusing on Vanilla CNN, ResNet, Hybrid CNN+RNN, and Vision Transformer models. Our result suggests that models trained exclusively on data from one subject exhibited varying performance, with some models showing susceptibility to overfitting on subject-specific data. Furthermore, longer time ranges led to improved accuracies, while FT surrogate augmentation mitigated overfitting.

1. Introduction

In this project, we performed experiments on the EEG dataset [3] using four neural network architectures, Vanilla CNN, ResNet, Hybrid CNN + RNN, and Vision Transformer. In addition to the three default questions, we also investigated how techniques such as FT surrogate augmentation on the EEG dataset could impact the models' training process and performance.

1.1. Data Preparation and Training

Prior to training, label smoothing is applied to the dataset. We've also experimented with Fourier-transform (FT) Surrogates, a data augmentation technique proposed by Schwabedal et al. [6] [8]. This technique involved randomizing the phase of Fourier coefficients of the original EEG data [7]. For every trial in training data, each channel has a 0.5 probability of being transformed, and the phase of each transformed channel in the same trial is perturbed equally. The technique is demonstrated in Appendix C. We evaluated the performance and observed differences when training with versus without FT Surrogate augmentation.

In training, we used Adam as the optimizer and Cross Entropy as the loss functions. We also employed an learning rate scheduler for fine-tuning the model in later epochs. Moreover, an early-stopping routine (adapted from [2]) was employed to reduce overfitting by cutting off the training once validation loss stopped improving.

1.2. Vanilla CNN

We started with vanilla CNN networks. Despite the differences between EEG data and images, we believed that by adapting CNNs to process EEG signals, we could use their ability to preserve spatial relationships in images to capture the temporal relationships in EEG data.

We utilized two convolution blocks (convolution - batch norm - ELU - avg pool - dropout), convolving in both the time and channel dimensions. We experimented with different hyperparameters and found that a large initial kernel size in the time dimension performs well. Inspired by Depthwise Convolution in EEGNet proposed by Lawhern et al. [4], the first convolution block in our vanilla CNN has two convolution operations performed separately over the time and channel dimension. In this way, we reduced the number of parameters by an order of magnitude compared to one two-dimensional kernel. The specific architecture is shown in Appendix B.1.

1.3. ResNet

In addition to vanilla CNN, we also studied ResNet, a modified version of CNN. The ResNet architecture contains an initial convolution block, whose output is passed through two residual blocks. In the end, an average pooling layer, a dropout layer, and a fully-connected layer are applied.

Each residual block has a main path with two 1D convolution blocks and a skip connection that bypasses these layers, directly forwarding input to the block's end. Skip connection typically performs an identity mapping of the input. If the input and output dimensions differ, skip connection performs an additional convolution on the input to align the dimensions [9]. The main path's output is summed with the input, then undergoes activation through an ELU layer, and finally passes through a dropout layer. The detailed structure is depicted in Appendix B.2. ResNet architectures with one, three, and four residual blocks had been tested before. It was found that architecture with one residue block underfit, while architectures with more than two residue blocks tend to overfit. The model's code used [5] as a reference.

1.4. Hybrid CNN + RNN

In addition to pure CNN networks, we also studied hybrid architecture combining convolution blocks and Long

Short-term Memory (LSTM) layers. We believe that integrating RNN into CNN enables the network to better capture inter-channel nuances and connections.

The architecture is shown in Appendix B.3. At a high level, four 1D convolution blocks are applied across the temporal dimension of the input. The output is passed into two LSTM layers which treat the output channels as the time-series dimension. At the end of the network, a fully-connected layer with dropout is applied before output. Since the structure of this architecture is highly flexible, we used the optimizing framework "Optuna" [1] to tune its hyperparameters. Specifically, the best optimal number of CNN/LSTM layers, kernel sizes, layer sizes, activation functions, learning rates, and weight decay were first roughly tuned by Optuna before manual fine-tuning.

1.5. Vision Transformer (ViT)

While convolutional networks leverage spacial and cross-channel convolution, vanilla transformer architectures have also demonstrated their potency in image classification tasks. Therefore, We believe that applying ViT to temporal EEG data could achieve great performance.

Our architecture is inspired by ViT [10] and ViT2EEG [11], which leveraged transformer encoder blocks for prediction. A vanilla ViT model [12] (referred to as Vanilla Transformer) segments images into smaller patches of pixels. For EEG data, we plan to adapt this mechanism to partition the input data along the time dimension to generate patches.

Our Vision Transformer model (Appendix B.4) has two (convolution - batch norm - ELU) blocks followed by (max pool - dropout) before transformer encoder layers. The first convolution is performed on the time dimension and the second convolution is performed on the channel dimension. Notably, data from the output of the convolution layers are divided along its width (equivalent to the time dimension of the original data) into patches, where positional embedding and class tokens are applied.

2. Result

2.1. Experiment on Raw Data of All Subjects

To compare performance, we evaluated the models on validation and testing data. The testing accuracies for Vanilla CNN, ResNet, Hybrid CNN + RNN, and Vision Transformer on raw data of all subjects are 0.720, 0.702, 0.736, and 0.673, respectively (Table 1).

Analyzing respective training histories (Figure 3 Left Column), we observe that CNN converges the fastest, while Vision Transformer requires many epochs that gradually improve the performance. All models exhibit overfitting on training data, with training accuracies increasing while validation accuracies fluctuating at lower plateaus. We terminated training when validation accuracies plateaued.

While all models reach around 0.7 accuracies, no models can achieve above 0.8 testing accuracies despite various attempts to adjust hyperparameters. We suspect that the types of architectures used by our models might have intrinsic limitations that prevent higher testing accuracies.

2.2. Experiment on Subject 0 Data

We tested model performance on Subject 0 when trained only with Subject 0 data. The training and validation accuracy and the loss of all models are shown in Figure 1. Vanilla CNN and Hybrid CNN + RNN overfit on Subject 0 data, as shown by the continuous improvements in training accuracy and loss and no improvements in testing accuracy and loss. In comparison, ResNet and Vision Transformer have better validation accuracy of 0.6 to 0.7.

The testing accuracies for Vanilla CNN, ResNet, Hybrid CNN + RNN, and Vision Transformer on Subject 0 are 0.260, 0.700, 0.320, and 0.540, respectively. ResNet has the best testing performance, followed by Vision Transformer. The other two models have low testing accuracies due to overfitting.

In addition, models specifically trained on Subject 0 have low testing accuracies when applied to the entire dataset. In contrast, models trained on all subjects can be applied to individual subjects with reasonably good performance. For instance, ResNet trained on all subjects achieves 0.78 accuracy on Subject 0, while ResNet trained on Subject 0 only achieves 0.35 accuracy on all data.

2.3. Experiment on Different Time Duration

In addition to using all 1000 slots of the time series, we also tested model performance when trained on data with reduced time ranges (50, 100, 150, 200, 250, 500, 750, and 1000 time slots). The validation and test accuracies of all models as a function of the data's time ranges are shown in Figure 2.

In general, accuracies increase as the models use data of longer time ranges. On data consisting of fewer time slots (e.g. 50 time slots), all models exhibit low accuracies ranging from 0.4 to 0.45. Nevertheless, when training data consists of more than 200 time slots, the accuracies of the models become stable and close to their optimal performance when data contains all 1000 time slots.

2.4. Experiment on FT Surrogate Augmented Data

We trained each model separately on raw and augmented data and applied augmentation exclusively on the training set to prevent potential leakage into the validation and testing sets. The testing accuracies for Vanilla CNN, ResNet, Hybrid CNN + RNN, and Vision Transformer on FT aug-

mented data are 0.711, 0.634, 0.673, and 0.607, respectively.

As shown in Figure 3, the training and testing metrics (accuracy and loss) of all models trained on the augmented dataset exhibit closer alignment. Additionally, models trained on augmented datasets exhibit a more obvious zigzag pattern during optimization. However, using augmented data did not increase the final validation accuracy of our models, even after we attempted to fine-tune the hyperparameters specifically for the augmented dataset.

3. Discussion

3.1. Comparing different architectures

All four models trained on the entire dataset achieved good testing accuracies of around 70%. Among them, Hybrid CNN + RNN has the highest testing accuracy on raw data of all subjects, reaching an accuracy of 0.736, the highest among all models trained. Vanilla CNN performs the best on FT surrogate augmented data, achieving a testing accuracy of 0.711. Vision Transformer has the lowest testing accuracy for both raw data and augmented data; nevertheless, its accuracies still reach about 0.6. In general, the performances of the four models do not vary significantly.

We have also trained vanilla RNN and vanilla transformer models. Despite significant effort in hyperparameter optimizations, these models produced less than satisfactory results (Table 1). This has motivated us to develop combined CNN-RNN and CNN-Transformer models for better capturing cross-channel relations and reducing the dimensionality of data.

We discovered that all of our best-performing models integrated some form of CNN. This aligns with our intuition that CNNs can be used to capture temporal information in EEG data. We suspect that for the classification task of this project, EEG data displays local temporal dependencies that are better suited for CNNs, in contrast with long-term temporal correlations that require RNNs.

3.2. Comparing one and all subject data

Vanilla CNN and Hybrid CNN + RNN perform significantly better on all subject data than one subject data because they overfit on one subject data. Vision Transformer has a testing accuracy of 0.54 on one subject data and 0.67 on all subject data. ResNet has a testing accuracy of 0.70 for both one subject data and all subject data. ResNet has a high testing accuracy for one subject data possibly due to the high weight decay value of 0.15 for performing L-2 regularization (the typical value is 0.001 for training other models), which encourages simpler weights. In general, if a model does not overfit on one subject data, its performances on one subject data and all subject data are similar.

The curves of training accuracy and validation accuracy on one subject data are generally smoother (contain fewer zig-zags) than the curves of those trained all subjects data. This observation is reasonable since the noise in data from one subject is expected to be smaller than the noise in data across different subjects, each with distinct EEG data characteristics.

3.3. Comparing different time series duration

Experiment results suggest an increase in performance as the number of time slots in training data increases. This observation is consistent with our expected result. Data with short time ranges does not provide enough information along the time dimension for the models to capture the correct correlation. In addition, using shorter time ranges, which reduces the input data size, leads to overfitting, hence negatively impacting the performance.

Moreover, when data contains more than 200 time slots, the classification accuracies of all models increase to reasonable values and are close to their final test accuracies. We can thus conclude that at least 200 time slots are required to obtain good test accuracies for all models. Interestingly, as we further increase the number of time slots used, all models' test accuracies experience slight zig-zags, with Visual Transformer having the greatest fluctuations in accuracy of about 5%. We suspect such zig-zags are caused by noise in the original dataset.

3.4. Comparing raw and FT surrogate augmented data

The aim of FT surrogate augmentation is to preserve important information like frequency-bands and energy trends while relying less on specific waveform behavior in the original training set. Randomization in our augmentation process generated surrogates with new shapes in the time domain, effectively producing a larger dataset. We hypothesized that augmentation could help us train more complex and better-performing models that are less prone to overfitting.

The result displays a closer alignment of test and validation accuracy after applying FT surrogate augmentation. The increasing occurrences of zig-zags on the accuracy and loss curves of the augmented set means a more complicated and challenging training loss topology for the model to navigate through, which we attribute to noise introduced by augmentation. These observations agrees with our hypothesis that overfitting can be mitigated.

However, no significant improvement in model performance is observed on the augmented training set, even after we increased the complexity of our architecture. Indeed, testing accuracies for all four models decreased after applying FT surrogate augmentation. We suspect that the data augmentation process adds too much noise, which loses information within the training data and leads to underfitting.

References

[1] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July 25). Optuna: A next-generation hyperparameter optimization framework. arXiv.org. https://arxiv.org/abs/1907.10902 2

[2] Bjarten. (n.d.). Bjarten/early-stopping-pytorch: Early stopping for pytorch. GitHub. https://github.com/Bjarten/early-stopping-pytorch 1

[3] Clemens Brunner, Robert Leeb, Gernot Müller-Putz, January 17, 2024, "BCI Competition 2008–Graz data set A", IEEE Dataport, doi: https://dx.doi.org/10.21227/katb-zv89. 1

[4] Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., & Lance, B. J. (2018, May 16). EEGNet: A compact convolutional network for EEG-based brain-computer interfaces. arXiv.org. https://arxiv.org/abs/1611.08024 1

[5] Nouman. (2022, June 21). Writing resnet from scratch in Pytorch. Paperspace Blog. https://blog.paperspace.com/writing-resnet-from-scratch-in-pytorch/ 1

[6] Rommel, C., Paillard, J., Moreau, T., & Gramfort, A. (2022, November 15). Data Augmentation for learning predictive models on EEG: A systematic comparison. arXiv.org. https://arxiv.org/abs/2206.14483 1

[7] Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., & Ball, T. (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. Human Brain Mapping, 38(11), 5391–5420. https://doi.org/10.1002/hbm.23730 1

[8] Schwabedal, J. T. C., Snyder, J. C., Cakmak, A., Nemati, S., & Clifford, G. D. (2019, January 28). Addressing class imbalance in classification problems of noisy signals by using Fourier transform surrogates. arXiv.org. https://arxiv.org/abs/1806.08675 1

[9] Shorten, C. (2019, May 15). Introduction to resnets. Medium. https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4 1

[10] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 2

[11] Yang, R., & Modesitt, E. (2023). ViT2EEG: Leveraging Hybrid Pretrained Vision Transformers for EEG Data. arXiv preprint arXiv:2308.00454 2

[12] Pulfer, B. (2023, December 19). Vision Transformers from Scratch (PyTorch): A step-by-step guide. Medium. https://medium.com/@brianpulfer/vision-transformers-from-scratch-pytorch-a-step-by-step-guide-96c3313c2e0c 2

Appendix A. Experiment Results

Table 1. Model Performance Comparison on Raw Data

Model	Validation Accuracy	Test Accuracy
Vanilla CNN	0.7308	0.7201
ResNet	0.7160	0.7020
Hybrid CNN + RNN	0.7456	0.7359
Vision Transformer	0.7012	0.6727
Vanilla RNN	0.4260	0.4221
Vanilla Transformer	0.6213	0.5395

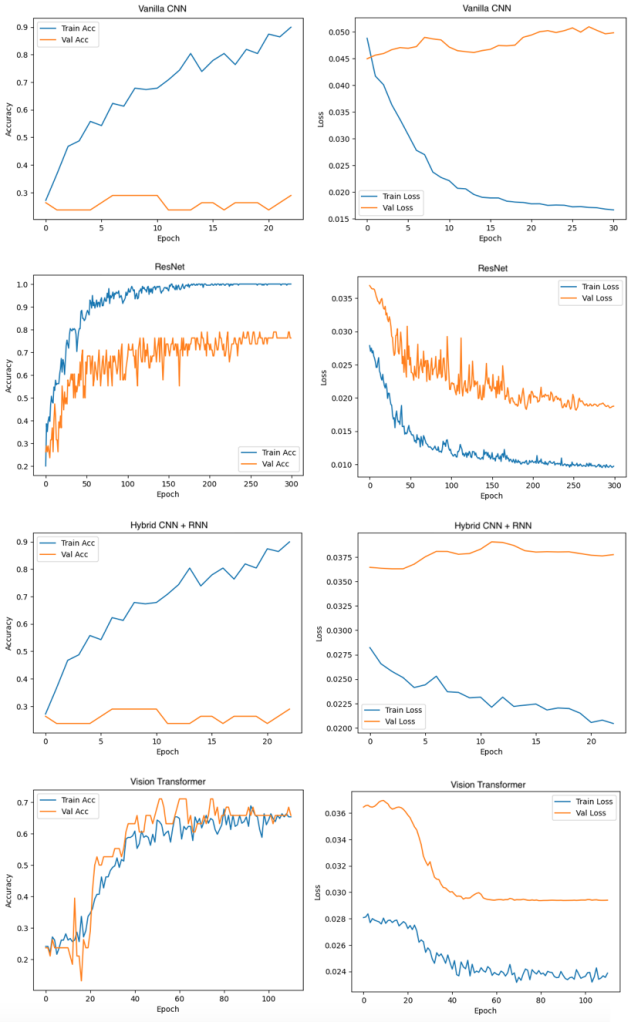


Figure 1. Validation and test accuracy (left) and loss (right) of all models on Subject 0

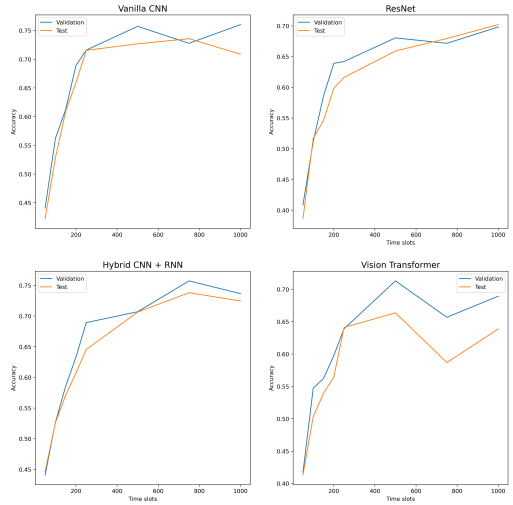


Figure 2. Validation and test accuracy of all models on different time durations

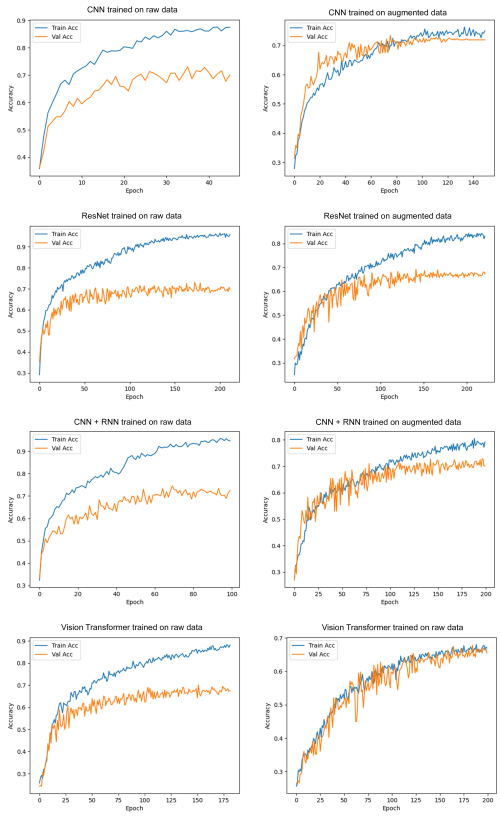


Figure 3. Validation and test accuracy on raw data (left) and validation and test accuracy on augmented data (right) of all models

Appendix B. Model Architectures

B.1. Vanilla CNN

Index	Layer	Params	Output Shape
0	(input)	-	(N, 22, 1000)
1-1	Reshape	-	(N, 1, 22, 1000)
1-2	Conv2D	channels = 40 kernel size = (1, 64) padding = (0, 32)	(N, 40, 22, 1001)
1-3	BatchNorm2D	-	(N, 40, 22, 1001)
1-4	Conv2D	channels = 80 kernel size = (22, 1) groups = 8	(N, 80, 1, 1001)
1-5	BatchNorm2D	-	(N, 80, 1, 1001)
1-6	Reshape	-	(N, 80, 1001)
1-7	ELU	-	(N, 80, 1001)
1-8	AvgPool1D	kernal size = 4 stride = 4	(N, 80, 250)
1-9	Dropout	p = 0.25	(N, 80, 250)
2-1	Conv1D	channels = 24 kernel size = 16 padding = 8 groups = 8	(N, 24, 251)
2-2	BatchNorm1D	-	(N, 24, 251)
2-3	ELU	-	(N, 24, 251)
2-4	AvgPool1D	kernal size = 8 stride = 8 padding = 0	(N, 24, 31)
2-5	Dropout	p = 0.25	(N, 24, 31)
2-6	Flatten	-	(N, 744)
2-7	Linear	n = 4	(N, 4)
3	Softmax	-	(N, 4)

B.2. ResNet

Index	Layer	Params	Output Shape
0	(input)	-	(N, 22, 1000)
1-1	Conv1D	channels = 50 kernel size = 70 padding = 35	(N, 50, 1001)
1-2	BatchNorm1D	-	(N, 50, 1001)
1-3	ELU	-	(N, 50, 1001)
1-4	MaxPool1D	kernal size = 3 stride = 2 padding = 1	(N, 50, 501)
1-5	Dropout	p = 0.3	(N, 50, 501)
2-1	Conv1D	channels = 22 kernel size = 3 stride = 2 padding = 1	(N, 22, 251)
2-2	BatchNorm1D	-	(N, 22, 251)
2-3	Conv1D	channels = 22 kernel size = 3 padding = 1	(N, 22, 251)
2-4	BatchNorm1D	-	(N, 22, 251)
2-5	ResidualAdd	skip connection = (2-5-1, 2-5-2)	(N, 22, 251)
2-5-1	Conv1D	channels = 22 kernel size = 1 stride = 2	(N, 22, 251)
2-5-2	BatchNorm1D	-	(N, 22, 251)
2-6	ELU	-	(N, 22, 251)
2-7	Dropout	p = 0.3	(N, 22, 251)
3-1	Conv1D	channels = 24 kernel size = 3 stride = 2 padding = 1	(N, 24, 126)
3-2	BatchNorm1D	-	(N, 24, 126)
3-3	Conv1D	channels = 24 kernel size = 3 padding = 1	(N, 24, 126)
3-4	BatchNorm1D	-	(N, 24, 126)
3-5	ResidualAdd	skip connection = (3-5-1, 3-5-2)	(N, 24, 126)
3-5-1	Conv1D	channels = 24 kernel size = 1 stride = 2	(N, 24, 126)
3-5-2	BatchNorm1D	-	(N, 24, 126)
3-6	ELU	-	(N, 24, 126)
3-7	Dropout	p = 0.3	(N, 24, 126)
4-1	AvgPool1D	channels = 24	(N, 24, 12)
4-2	Flatten	-	(N, 288)
4-3	Dropout	p = 0.3	(N, 288)
4-4	Linear	n = 4	(N, 4)
5	Softmax	-	(N, 4)

Note: To match the dimension of the input and the main path's output, the skip connection performs an additional downsampling convolution to the input. The notations (2-5-1, 2-5-2) and (3-5-1, 3-5-2) refer to the downsampling convolutions on the skip connection.

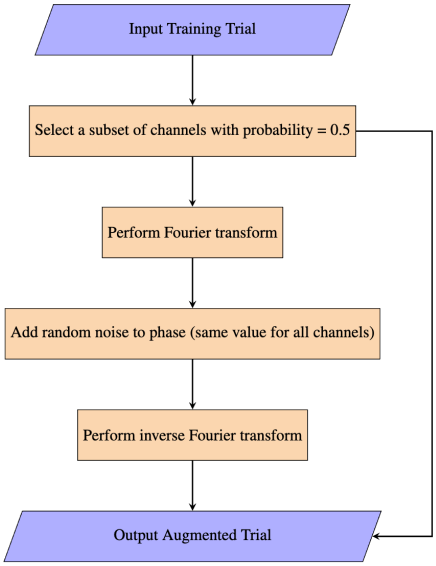
B.3. Hybrid CNN + RNN

Index	Layer	Params	Output Shape
0	(input)	-	(N, 22, 1000)
1-1	Conv1D	channels = 32 kernel size = 8 padding = 4	(N, 32, 1001)
1-2	ELU	-	(N, 32, 1001)
1-3	BatchNorm1D	-	(N, 32, 1001)
1-4	MaxPool1D	kernal size = 2	(N, 32, 500)
1-5	Dropout	p = 0.5	(N, 32, 500)
2-1	Conv1D	channels = 64 kernel size = 10 padding = 5	(N, 64, 501)
2-2	ELU	-	(N, 64, 501)
2-3	BatchNorm1D	-	(N, 64, 501)
2-4	MaxPool1D	kernal size = 2	(N, 64, 250)
2-5	Dropout	p = 0.5	(N, 64, 250)
3-1	Conv1D	channels = 128 kernel size = 12 padding = 6	(N, 128, 251)
3-2	ELU	-	(N, 128, 251)
3-3	BatchNorm1D	-	(N, 128, 251)
3-4	MaxPool1D	kernal size = 2	(N, 128, 125)
3-5	Dropout	p = 0.5	(N, 128, 125)
4-1	Conv1D	channels = 256 kernel size = 14 padding = 7	(N, 256, 126)
4-2	ELU	-	(N, 256, 126)
4-3	BatchNorm1D	-	(N, 256, 126)
4-4	MaxPool1D	kernal size = 2	(N, 256, 63)
4-5	Dropout	p = 0.5	(N, 256, 63)
5	LSTM	hidden size = 150	(N, 256, 150)
6	LSTM	hidden size = 75	(N, 256, 75)
7-1	Flatten	-	(N, 19200)
7-2	Linear	n = 150	(N, 150)
7-3	BatchNorm1D	-	(N, 150)
7-4	ReLU	-	(N, 150)
7-3	Dropout	p = 0.5	(N, 150)
7-3	Linear	n = 4	(N, 4)
8	Softmax	-	(N, 4)

B.4. Vision Transformer (ViT)

Index	Layer	Params	Output Shape
0	(input)	-	(N, 22, 1000)
1-1	Reshape	-	(N, 1, 22, 1000)
1-2	Conv2D	channels = 20 kernel size = (1, 64) padding = (0, 32)	(N, 20, 22, 1001)
1-3	BatchNorm2D	-	(N, 20, 22, 1001)
1-4	ELU	-	(N, 20, 22, 1001)
1-5	Conv2D	channels = 40 kernel size = (22, 1) padding = 0 groups = 10	(N, 40, 1, 1001)
1-6	BatchNorm2D	-	(N, 40, 1, 1001)
1-7	ELU	-	(N, 40, 1, 1001)
1-8	AvgPool1D	kernal size = 4 stride = 4 padding = 0	(N, 40, 1, 250)
1-9	Dropout	p = 0.3	(N, 40, 1, 250)
1-10	Reshape	-	(N, 40, 250)
2-1	Patchify	-	(N, 125, 64)
2-2	Linear	-	(N, 125, 64)
2-3	Class Token	-	(N, 126, 64)
2-4	Positional Embedding	-	(N, 126, 64)
2-5	Transformer EncoderLayer	channels = 24 heads = 8 hidden dim = 64 feed-forward dim = 64 dropout p = 0.3	(N, 64)
3-1	Linear	-	(N, 4)
3-2	Softmax	-	(N, 4)

Appendix C. FT Surrogate Augmentation



We utilized `FTSurrogate` and `AugmentedDataLoader` from the open-source library Braindecode to implement these augmentations.

The augmentation is performed as the model is training, and surrogate data is generated in training time, so the data is different for every epoch.