ECE247 Project: CNN vs. RNN for Classifying EEG Data

Aki Kono
UCLA
Los Angeles, CA 90095
akisaitoh@ucla.edu

Bryan J Lee
UCLA
Los Angeles, CA 90095
bryanae@ucla.edu

Abstract

Given electroencephalography (EEG) data and their labels, the paper compares two approaches for building a best classifier for the data. In the first approach, a deep 2D CNN was combined with a shallow 3D CNN to extract spatiotemporal features of the data. Data augmentation before training the combined classifier was proven to be effective and increased test accuracy by 13%. In the second approach, an autoencoder was optimized with respect to its latent dimension and the data was encoded to reduce their spacial dimensions and improve test accuracy. The processed data was used to train a combination of a CNN and RNN classifier. In conclusion, the model with purely CNN structure resulted in higher test accuracy of 78% compared to 67% of the CNN-RNN structure.

1. Introduction

This project for ECE247 at UCLA compares two classifiers with different EEG data preparation techniques and neural network structures. In the first approach, a 2D-3D CNN model is trained with augmented data. The deep 2D CNN extracts features of hierarchical time-series data whereas the shallow 3D CNN extracts 22 spatial features. In the second approach, the data was encoded so that the the spacial dimensions of the data was reduced. The motivation for encoding data was to speed up the training process and, possibly, improve the test accuracy of the model by feeding relevant spatial information to it.

2. EEG data analysis and data augmentation

EEG data is considered spatiotemporal, noisy, subject-specific and task-specific. Feature extraction of such high dimensional multivariate data is challenging especially when small-sized dataset is available. Variability in EEG data and effective augmentation methods are examined.

Frequency band	Frequency	IMF equivalent
Gamma (γ)	>35 Hz	IMF1
Beta (β)	12-35 Hz	IMF2
Alpha (α)	8-12 Hz	IMF2
Theta (θ)	4-8 Hz	IMF3
Delta (δ)	0.5-4 Hz	IMF4

Table 1: Five Basic Brain Waves and IMFs

2.1. EEG data analysis

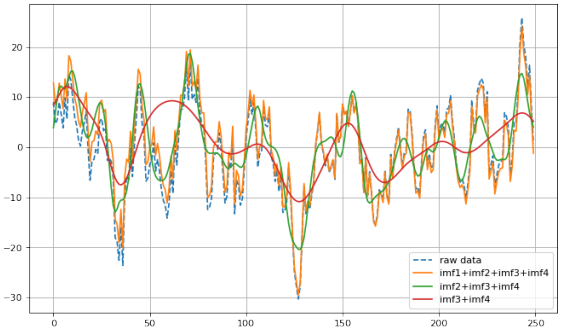
A simple 2D CNN classifier demonstrated in [6] is trained with different subset of data to correlate the model accuracy and different features of data. The prediction accuracy is optimized for each dataset and summarized in Figure 4. Strong correlation is observed with the EEG data for $t = [0, 500)$ where t is the timestpes. Also, data for $t = [500, 750)$ was kept as noise in the data. Indeed, higher accuracy was achieved with $t = [0, 750)$ as compared to $t = [0, 500)$.

To examine frequency correlation, the original time series data is decomposed into 4 different time series data using the iterated masking empirical mode decomposition (itEMD) [2]. Figure 1a shows the decomposed time series data, or the intrinsic mode functions (IMFs), ranging from faster dynamic waveform (IMF1) to slower dynamic waveform (IMF4). The frequency band of each IMF is approximately categorized into five basic brain waves as shown in Table 1.

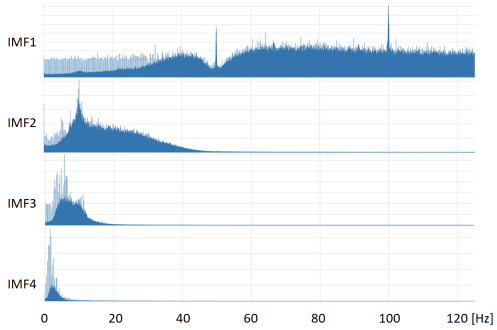
Figure 4 shows that the higher accuracy is achieved with IMF2 signal (Beta and Alpha ranges) than with other signals IMFs. Performance variability is also observed across individual subjects where the lowest accuracy is 43% (subject 2) and the highest accuracy is 74% (subject 9).

2.2. Data augmentation

Data augmentation compensates small data size and subject dependent noise. The Small sample size of the EEG data, 2115 trials, was a major obstacle to train deeper neu-



(a) Example of EMD of a time-series EEG data and its IMFs



(b) FFT on all IMFs obtained from original EEG dataset.

Figure 1: Visualization of EMD on EEG dataset.

ral network model.

The process of augmenting data is as follows. First, separate data into 4 classes. Second, segment the data into 3 time frames $t = [0, 250), t = [250, 500)$, and $t = [500, 750)$ where $t = [750, 1000)$ is removed. Third, randomly select one data segment from each of 3 time frames and concatenate them along the time axis. The shape of this newly created data is $(1, 22, 750)$, 22 channel time-series data from $t = 0$ to $t = 750$.

The motivation is to combine data from different subjects so that classifier become more invariant to subject-specific artifacts. The same approach with successful implementation was found in [3]. The total number of augmented data is increased until classifier accuracy converges. The optimal ratio of raw data to augmented data was 1:4. Augmentation increased test accuracy from 63% to 76%.

When preparing K-folds cross validation dataset, augmented data is separated into 5 folds such that each fold contains approximately balanced distribution of each distinct sub-group, i.e. 4 classes from each 9 subject resulting in the total of 36 distinct composite groups. Distributing data evenly helped to increase testing accuracy by 1%.

Note: Because dataset is augmented before separated into train and validation sets, our validation dataset contained the segments of data used to train the model, resulted in high validation accuracy close to 99%. Followup training was conducted, and 2D-3D CNN model was able to achieve 77.2% test accuracy without validation data.

3. Approach I: 2D-3D CNN Model

2D-3D CNN model is designed to supplement spatial feature extraction of the 2D CNN model introduced by [6]. EEG data is expected to have spatial correlation. However, no significant correlation was found when the 2D CNN model was trained with right or left channels only. To further investigate spatial correlation, 3D CNN layer is added parallel to the 2D CNN model. The 2D-3D CNN architecture is tabulated in Table 4. Input data is rearranged into 2D spatial matrix as shown in Figure 5(a). Similar approach can be found in [5] and [2]. Max pooling layer is first applied along the time axis to downsample the data from 250Hz to 62.5Hz, then 3D convolution layer is applied. Deep structure with more than two 3D convolutional layers didn't improve the accuracy. Kernel size of 3x3xt with different timestep t also did not perform well.

To generalize the model, batch size is reduced from 64 to 16, dropout is increased from 0.5 to 0.525, label smoothing is applied with $\epsilon = 0.2$, and the testing accuracy increased by 2%, resulted in 78% highest accuracy.

Figure 5(b) visualize the weights of 3D filters. No significant spatial and temporal feature was observed.

4. Approach II: CNN-RNN Model with Data Encoding

This second approach processes the EEG data using an encoder to reduce the spacial dimension and then populate additional data using the data augmentation method in approach I. Then the processed data is used to train a combination of CNN and RNN models. The flow chart in Fig. 2 gives an overview of the approach.

4.1. Data Encoding

We reduced the spacial dimensions of the data using a simple encoder so that only relevant spacial information are fed to the model. As shown in Table 3 (Appendix B), the autoencoder consists of an encoding affine layer with weight W_1 and bias b_1 and a decoding affine layer with W_2 and b_2. The loss function $\mathcal{L}$ is a mean-squared error defined below [1].

$$\mathcal{L} \triangleq \|\hat{x} - x\|^2 \tag{1}$$

$$z = A_1 x + b_1 \qquad \hat{x} = A_2 z + b_2 \tag{2}$$

where $A_1 \in \mathbb{R}^{22\times d}$, $b_1 \in \mathbb{R}^d$, $A_2 \in \mathbb{R}^{d\times 22}$, $b_2 \in \mathbb{R}^{22}$. The latent dimension d is a hyperparameter to be optimized for a neural network. The optimization process is visualized in Fig. 2

Two observations were made regarding the use of the autoencoder. First, using an encoder improved performance in

terms of the loss and validation accuracy of the CNN-RNN model for latent dimensions greater than or equal to 10 (See Fig. 3). The highest test accuracy was for $d = 10, 16$. Second, there was not much difference in performance for latent dimensions between 10 and 22. For the optimal latent dimension, we picked 10 since it was the smallest spacial dimension allowed without sacrificing performance of the neural network.

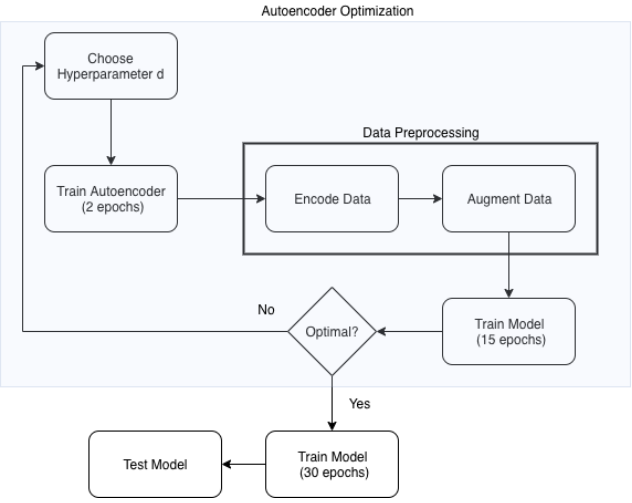


Figure 2: Flow chart of approach II

4.2. CNN-RNN Model Structure

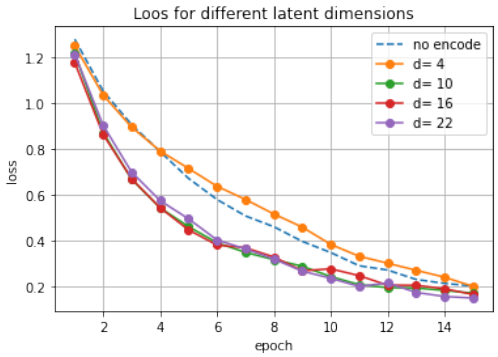
The CNN-RNN structure in Table 5 (Appendix C) was initially motivated by the failure of a purely RNN model to produce satisfying test accuracy–best test accuracy was around 25%. The added CNN structure helped increase test accuracy to above 60%.

5. Result

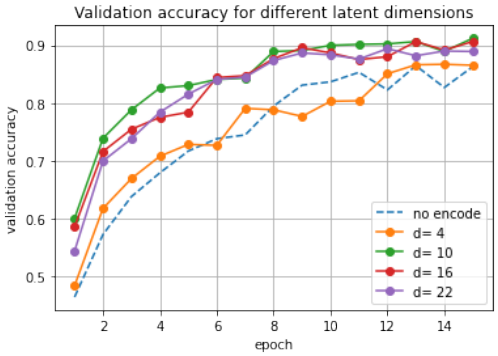
Table 2 compares the two approaches described above. The best test accuracy of the CNN model was 78%. With the optimal latent dimension $d = 10$ for the encoder, the best test accuracy of the CNN-RNN model was 67%. Note that the test data for this model had to be encoded to fit the input size of the model.

6. Discussion

The accuracy of 2D-3D CNN model was increased 13% with simple data augmentation to help the model to generalize. However, some studies [7] and [4] showed high correlation with spatial feature which our model was not able to extract from EEG data. Autoencoder analysis suggested spatial dimension can be reduced almost half while keeping relevant features. This limitation in 2D-3D CNN model may be overcome by reducing the model complexity.



(a) Loss



(b) Validation accuracy

Figure 3: Comparison for different latent dimensions d

	Approach I	Approach II
Classifier model	2D-3D CNN	CNN-RNN
Data encoding	No	Yes
Data augmentation	Yes	Yes
Time-steps used	[0:750]	[0:500]
Subjects	All	All
Test accuracy	$\sim 78\%$	$\sim 67\%$

Table 2: Comparison of the two models

Comparing the two models, it could be infer that a deep CNN structure along time-axis is essential for a successful classification. In the second approach, encoding the spatial information and implementing the shallow CNN followed by the RNN was not as capable as the deep CNN model of the first approach. Therefore, we suggest a deep CNN architecture for classifying EEG data with sparse spatial features.

References

[1] F. Chollet. Building autoencoders in keras. 2016.

[2] M. S. Fabus, A. J. Quinn, C. E. Warnaby, and M. W. Woolrich. Automatic decomposition of electrophysiological data into distinct nonsinusoidal oscillatory modes. *Journal of Neurophysiology*, 126(5):1670–1684, 2021. PMID: 34614377.

[3] C. He, J. Liu, Y. Zhu, and W. Du. Data augmentation for deep neural networks model in eeg classification task: A review. *Frontiers in Human Neuroscience*, 15, 2021.

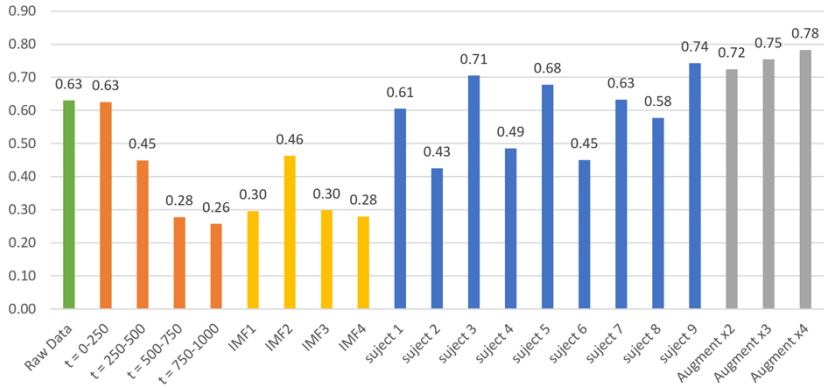
[4] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 15(5):056013, jul 2018.

[5] T. Liu and D. Yang. A three-branch 3d convolutional neural network for eeg-based different hand movement stages classification, May 2021.

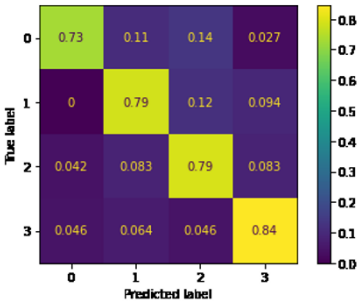
[6] T. Monsoor. Jupyter notebooks – discussion 8 notebook.ipynb, discussion 9 notebook.ipynb, 2022.

[7] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG. *CoRR*, abs/1703.05051, 2017.

Appendix A. Performance

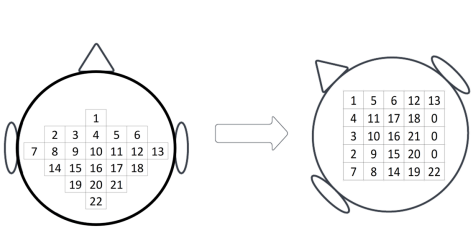


(a) Prediction accuracies evaluated on various sub-groups of original dataset. The same CNN model is trained and optimized for each sub-group. For the three different trials on augmented data, each dataset contains raw data and additional augmented data that is twice, three times, or four times larger in size than the original raw dataset.

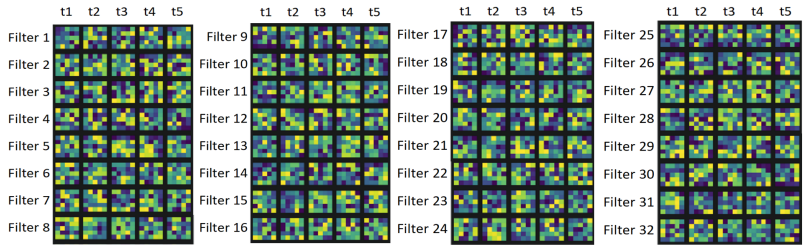


(b) Confusion matrix of predictions from CNN model that achieved 78.78% accuracy. Label 0: left hand; Label 1: right hand; Label 2: foot; and Label 3: Tongue.

Figure 4: Performance of the CNN model evaluated for various sub-groups of dataset. Normalized confusion matrix is shown for Augment x4 dataset that achieved 78.8% testing accuracy.



(a) Left: The spatial arrangement of the 22 EEG data channels. Right: Approximate rearrangement of 22 channels into 2D matrix with additional three zero padding on the right column. 3D convolutional layer is applied after the rearrangement.



(b) Weight visualization of 32 filters of the first 3D convolutional layer. The rearranged data shown in (a) is first max-pooled along the time axis. Then, 3D filters were applied to extract spatiotemporal features. The shape of the each filater is 5x5x5.

Figure 5: Data rearrangement for 3D convolutional layer and weight visualization of resulted filters.

Appendix B. Autoencoder (AE) Architecture and Training Parameters

Layer	Output Shape	Units	Activation
Input	(N,22)	-	-
Dense	(N,d)	d	none
Dense	(N,22)	22	none

(AE Total params: 742)

	2D-3D CNN	AE	CNN-RNN
epochs	300	5	30
optimizer	adam	adam	adam
batch size	16	N/A	64
loss	crossentropy	MSE	crossentropy

Table 3: AE architecture (left) and training parameters (right)

Appendix C. Model Architecture

Layer	Output	Units	Filter	Activation	Other
Input	(N,750,1,22)				
Conv2D	(N,750,1,25)	5525	(10,1)	elu	stride=1
MaxPooling2D	(N,250,1,25)		(3,1)		stride=1
BN,Dropout	(N,250,1,25)	100			drop=0.525
Conv2D	(N,250,1,50)	12550	(10,1)	elu	stride=1
MaxPooling	(N,84,1,50)		(3,1)		stride=1
BN,Dropout	(N,84,1,50)	200			drop=0.525
Conv2D	(N,84,1,100)	50100	(10,1)	elu	stride=1
MaxPooling2D	(N,28,1,100)		(3,1)		stride=1
BN,Dropout	(N,28,1,100)	400			drop=0.525
Conv2D	(N,28,1,200)	200200	(10,1)	elu	stride=1
MaxPooling	(N,10,1,200)		(3,1)		stride=1
BN,Dropout	(N, 10, 1, 200)	800			drop=0.525
Flatten,Dense2D	(None, 4)	8004		elu	
Input	(N,5,5,750,1)				
MaxPooling3D	(N,5,5,187,1)		(1,1,4)		stride=4
Conv3D	(N,5,5,187,32)	4032	(5,5,5)	elu	stride=1
MaxPooling3D	(N,5,5,62,32)		(1,1,3)		stride=3
BN,Dropout	(N,5,5,62,32)	128			drop=0.525
Flatten,Dropout,Dense3D	(None, 4)	198404		elu	drop=0.525
Concat(Dense2D,Dense3D),Dense	(None, 4)	36		softmax	

(Total params: 480,479, trainable params: 479,665, non-trainable params: 814)

Table 4: 2D-3D CNN architecture

Layer	Output Shape	Units	Filter size	Activation	Other
Input	(N,750,1,16)				
Conv2D	(N,750,1,50)	50	(10,1)	elu	pad=0, stride=1
MaxPooling2D	(N,250,1,50)		(3,1)		pad=0, stride=3
BatchNormalization	(N,250,1,50)				
Dropout	(N,250,1,50)				drop=0.5
Conv2D	(N,250,1,100)	100	(10,1)	elu	pad=0, stride=1
MaxPooling	(N,84,1,100)		(3,1)		pad=0, stride=3
BatchNormalization	(N, 84, 1, 100)				
Dropout	(N,84,1,100)				drop=0.5
Flatten	(N,8400)				
Dense	(None,100)	100		none	
Reshape	(None,100,1)				
LSTM	(None, 50)	50		tanh	recurrent activation=sigmoid
Dense	(None, 4)	4		softmax	

(Total params: 909,454, trainable params: 909,154, non-trainable params: 300)

Table 5: CNN-RNN architecture