Exploration of Pattern Recognition Methods for Motor Imagery EEG Signal with Convolutional Neural Network Approach

Hanina N Zahra, Hasballah Zakaria, and Beni R Hermanto

School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, West Java 40132, Indonesia

haninanz@gmail.com, fahala@gmail.com, benirio@gmail.com

Abstract. As an application of EEG, Motor Imagery based Brain-Computer Interface (MI BCI) plays a significant role in assisting patients with disability to communicate with their environment. MI BCI could now be realized through various methods such as machine learning. Many attempts using different machine learning approaches as MI BCI applications have been done with every one of them yielding various results. While some attempts managed to achieve agreeable results, some still failed. This failure may be caused by the separation of the feature extraction and classification steps, as this may lead to the loss of information which in turn causes lower classification accuracy. This problem can be solved by integrating feature extraction and classification by harnessing a classification algorithm that processed the input data as a whole until it produces the prediction, hence the use of convolutional neural network (CNN) approach which is known for its versatility in processing and classifying data all in one go. In this study, the CNN exploration involved a task to classify 5 different classes of fingers' imaginary movement (thumb, index, middle, ring, and pinky) based on the processed raw signal provided. The CNN performance was observed for both non-augmented and augmented data with the data augmentation techniques used include sliding window, noise addition, and the combination of those two methods. From these experiments, the results show that the CNN model managed to achieve an averaged accuracy of 47%, meanwhile with the help of augmentation techniques of sliding window, noise addition, and the combined methods, the model achieved even higher averaged accuracy of 57,1%, 47,2%, and 57,5% respectively.

1. Introduction

Essentially, Brain-Computer Interface is a system that allows humans' interactions with their surroundings without the help of peripheral nerves and muscle [1]. The system recognizes a certain set of patterns brain signals and translates them into actions supposedly intended by the user [2]. BCI implementation usually consists of these five stages; signal acquisition, preprocessing or signal enhancement, feature extraction, classification, and control interface. While each stage has its own importance, feature extraction and classification stages handles the "translation" task of BCI jointly, hence these two stages need more consideration on how they are designed.

As written above, most of the existing BCI studies treat feature extraction and classification as two separated stages altogether but apparently, it is indicated that by separating feature extraction from classification, some of the crucial information contained in the raw data is lost which may lead to a

rather low classification accuracy [3, 4, 5]. Lee et al. [6] showed in their study that their CNN designed to process and classify raw data all at once managed to achieve much higher performance than another approach that uses Filter Bank Common Spatial Pattern (FBCSP) as the feature extraction method and regularized Linear Discriminant Analysis (LDA) for the classification. The CNN managed to obtain 88% for 3-class MI classification and 82% for the 5-class, meanwhile the FBCSP+LDA approach only yielded 44% for both paradigms. Another downside to the separation is the long steps required in most feature extraction techniques. While longer steps that yields higher performance is quite an acceptable trade off, the very same performance that could be obtained in a much shorter time is preferable. For these considerations, an approach that could incorporates both feature extraction and classification in one stage is more preferred. CNN is one of most prominent deep learning approach that is designed for end-to-end learning purpose and able to exploit hierarchical structure in natural signals [7], which is suitable for BCI implementation with integration in feature extraction and classification stages. Previous existing studies that incorporates CNN as their approach also show that with appropriate architecture design, CNN is capable of classifying MI tasks quite accurately [6, 7, 8].

The purpose of this study is to explore further about how CNN operates on MI EEG signal and optimize it further through data augmentation techniques to achieve even better classification performance on the dataset developed by Kaya et al. [9], more specifically on the 5F (5 Fingers) paradigm dataset.

2. Materials

2.1. Data Description

The open access dataset used in this study was developed with the 5F paradigm. This dataset contains 19 data, each from different sessions, recorded from 8 subjects whose age ranges from 20 to 35 years old. All session recorded in this paradigm persists for around an hour yet the number of events in each session differ to each other.

Some of the data in this paradigm were acquired in 200 Hz sampling rate, while the others in 1 kHz. This was meant for inspecting higher frequency [9]. For the sake of having more data at hand, in this study all the data with 1 kHz sampling rate were down-sampled to 200 Hz.

2.2. Data Acquisition

All EEG data were acquired using an EEG-1200 JE-921A EEG system. The EEG-1200 itself provides EEG signal measurements up to 38 input channels, but during this acquisition process only 19 input channels used. These 19 electrodes are configured according to the 10/20 international configuration. For the recordings, no electromagnetic shielding or artifact control was attempted for the reason that the BCI data processing sub-system should be able to cope with the pollution [9]. The 10/20 montage was also slightly modified; 19 standard 10/20 EEG leads, 2 ground leads placed at the earbuds, and 1 bipolar lead for data synchronization.

The sessions were organized as sequences of 15 minutes long BCI interaction segments separated by 2 minutes break. Each session was initialized by a relaxation period for 2.5 minutes to make sure the participants were acclimatized to the experiment's conditions. During the BCI interaction segments, graphical user interface (GUI) was placed in front of the subjects, displaying an open palm. Numbers represented from 1 to 5 will popped out above the fingers which signifies cues for the subjects as well as the movement imagery to be performed. These cues appear at the start of each trial and will remain for 1 second. The participants were expected to perform the movement imagery accordingly and remain passive until the next cue.

3. Methods

3.1. Preprocessing

In this study, the preprocessing stage only involves filtering, creating EEG epochs from the data, and lastly down-sampling for signals with 1 kHz sampling rate. For filtering, the filter used was a low-pass Finite Impulse Response (FIR) filter with pass-band frequency of 0-40 Hz. Some of the considerations for this are since the mu (8-13 Hz) and beta (20-30 Hz) rhythm has power modulation of MI signal [10], clearly they must be included for the classification purposes. Aside from that, finger movement which is the focus of the classification also induces gamma oscillation, specifically in 36-40 Hz frequency range [11]. Additionally, 0-5 Hz contains movement-related potentials (MRP) which has proven to contribute to high classification performance [10]. Blackman was used as the window to apply the filter since it has wider transition band, stable application under high frequency, and low attenuation only second to Hann window [12].

After filter has been applied, the synchronization channel was set aside, leaving only 19 EEG channels and 2 ground channels remain in the signal. The signals then were cut into shorter signals with the duration of 0.85 s starting from the onset of the stimuli. According to Mishchenko et al. [10], this epoch length gave the highest classification performance for this dataset. Once EEG epochs have been obtained, all epochs with 1 kHz sampling rate was down-sampled to make sure that every epoch has the same length in terms of number of samples. Once the epochs with 1 kHz sampling rate has been resampled, all epochs should have the length of 170 data samples.

3.2. Data Augmentation

In general, there are two different experiments held in this study; one that involves data augmentation and one that does not. For the experiment without augmentation, the epochs obtained from preprocessing stages were immediately divided into train and test set with a ratio of 9:1 respectively. As for the experiments that incorporates data augmentation, for each method used in this study, the order of when the data as a whole would be divided into train and test set slightly differ to each other for reasons that will be explained below.

3.2.1. Sliding Window Method

Sliding window is one of a few data augmentation techniques that yields better results for EEG classification purposes [13]. As the name implies, sliding window works by applying a unit square window of length l to the real signal to be multiplied element-wise, resulting in a shorter signal with the same length l that contains some parts of the original signal. The window is then shifted towards the end of the original signal, skipping one or a few data samples. The number of data samples skipped during this shifting process is known as stride. This cycle of multiplying and shifting repeats itself until the window reaches the end of the signal, and as a result, multiple shorter signals were obtained from one single long signal.

Due to the way sliding window works to augment data, the number of data obtained from this technique may be very limited. There are a few things to consider before implementing sliding window, such as the window length must be long enough to retain most of the original signal information yet short enough to maximize the amount of data that can be obtained. The length of stride is also important factor; a long stride length may help a deep learning model better in generalizing the data given since the similarity between each new epoch may diminishes with longer stride, but this also reduces the amount of data resulted from augmentation.

In this study, we use 0.75 s or 150 data samples for the window length and 0.05 s or 10 data samples for stride length. Note that for this experiment, dividing data into train and test set can only be done once the data has been augmented. Augmenting data on the train set only will undoubtedly cause an error once training starts since the CNN model used in this study is inflexible in terms of input data shape.

3.2.2. Noise Addition Method

Same as the sliding window technique from before, noise addition is one of data augmentation techniques that yields better EEG classification performance especially in deep learning approaches. Compared to sliding window, noise addition is relatively more straightforward; simply add noise to the original signal and the result will be considered as new data. Some noises that are often used for this augmentation method are Poisson, Gaussian, and Salt and Pepper [13], but according to Wang et al. [14] for EEG signals, local noise such as Poisson and Salt and Pepper could damage the intrinsic features of the signal hence these two are not recommended.

Gaussian noise has three parameters to be accounted for; mean (μ), standard deviation (σ), and magnification factor (*m*). Unlike sliding window, noise addition excels in terms of the amount of data that could be added to the original dataset since it is technically possible to use various values for the parameters as long as the noise's addition to the signal does not cause the signal-to-noise ratio (SNR) to drop below 0. Most studies that utilized this augmentation method used zero mean and magnification factor of 1, only varying the value for standard deviation [13]. For that reason, the mean and magnification factor for this experiment were set to the same value as well. 0.5 and 0.6 were used as the standard deviations of the Gaussian noise.

Specifically for noise addition experiment, augmentation is only applied to the train set, which means dividing the dataset took place before augmentation as opposed to the sliding window experiment.

3.2.3. Combined Method

In this study, combined method referred to application of both sliding window and noise addition to the same dataset jointly. The parameters for the implementation of this combined method are the same and the same parameter values are used in this experiment as well. To implement this method, firstly sliding window must be applied to the dataset before splitting them into train and test set. Again, this is to make sure that every single data for both training and validating has the same length. Once train set has been obtained, noise addition can be applied to it.

Figure 1. shows general steps of augmentation for the three methods used in this study. Note that data for both sliding window and combined methods has different length compared to the original signal, which means the CNN model built for these two experiments and noise addition experiment would have different input shapes. Nevertheless, the CNN architecture and hyperparameter configurations for all experiments are one and the same to narrow the scope of analysis.

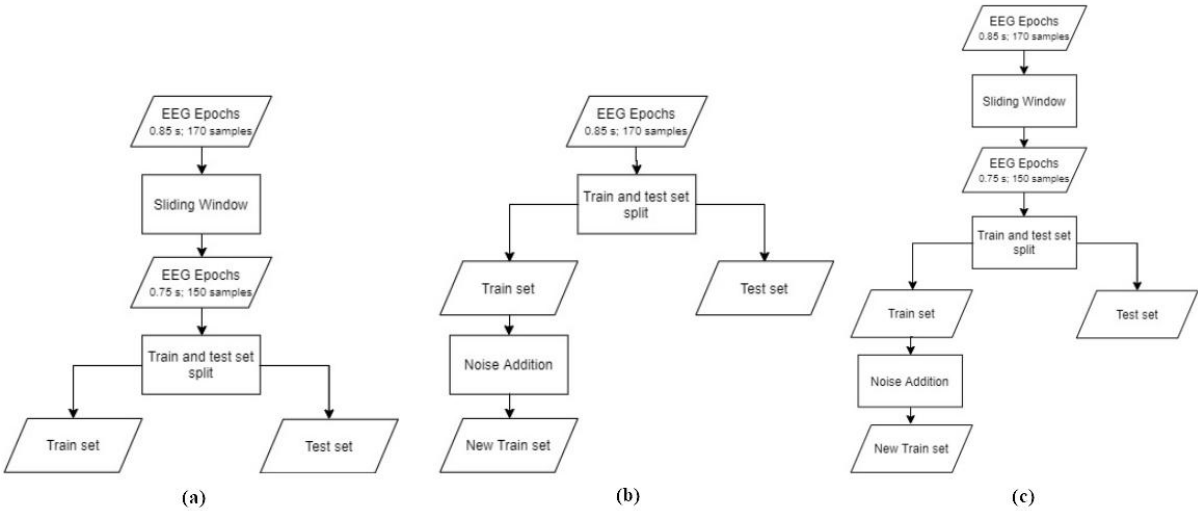


Figure 1. Flow charts describing the order of augmentation and data splitting for (a) sliding window, (b) noise addition, (c) combined method experiments.

3.3. Building CNN Model and Classification

The CNN architecture used in this study was adapted from CNN designed by Dose et al. [8] with a few changes intended as improvement. The general architecture for CNN is shown below in Figure 2.

As shown in the figure, the CNN consists of 2 convolutional layers, one average pooling and flatten layer, and lastly 2 fully connected layers. The first convolutional layer is designed to perform convolution along the time axis while the second one performs on the EEG channel axis. This is inspired from how FBCSP work where bandpass and CSP spatial filter are applied to the signal [7]. The average pooling layer is supposedly analogous of the log-variance computation in FBCSP although in practice it is simplified to a certain extent. In a sense, this CNN architecture is quite similar to applying FBCSP to EEG signal then classifying it using conventional machine learning approach.

Although the data inputted to the CNN are not entirely raw anymore due to the preprocessing stage implemented before, they could still be considered as "raw" in comparison to the MI EEG conventional classification method where the data are entirely transformed that they are no longer data of time domain. These processed signals are shaped in (number of EEG channels, data length, 1) format, where the EEG channels are treated as data row and the length in data samples are considered as data column. The addition of 1 in the end is merely data dimension expansion since the CNN used in this study only accepts 3 dimensional data. The EEG channels are set to follow this order from the first to the last row; Fp1, Fp2, F3, F4, C3, C4, P3, P4, O1, O2, A1, A2, F7, F8, T3, T4, T5, T6, Fz, Cz, and lastly Pz. To give a clearer overview about input data, the following Figure 3 gives quite a close approximate illustration on it.

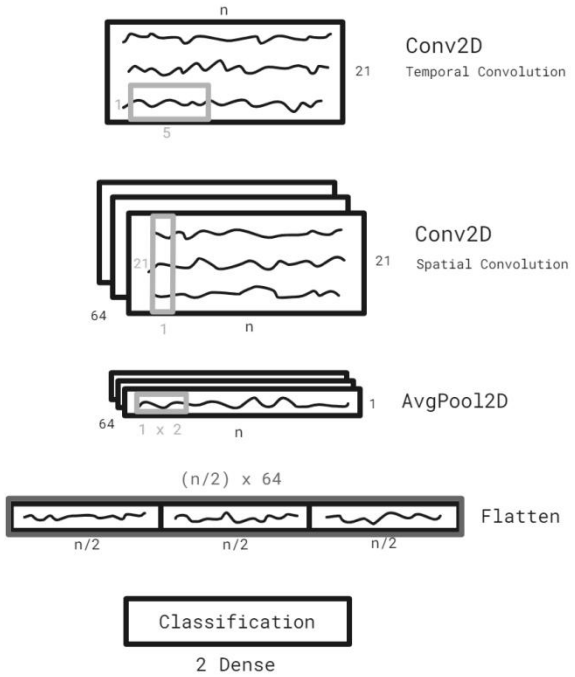


Figure 2. General CNN architecture for all experiments. The notation *n* signifies the length of input signal that differs depending on the ongoing experiment.

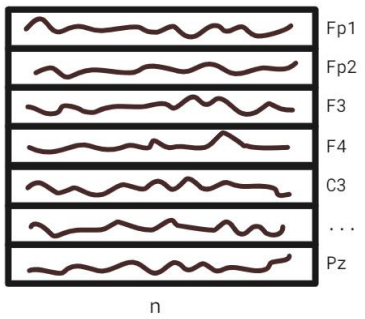


Figure 3. Input data format for all experiments. Length *n* varies depending on the ongoing experiment.

The convolutional layers are set to have 64 filters each instead of 40 as in Dose et al. since more filters could help improve classification accuracy in exchange for longer training time [15]. 64 is chosen since it is considered as a good trade off between the two. Aside from number of filters, the convolutional layers in this study use the same configuration as Dose et al. The first layer uses padding

during the convolution process to maintain data length before inputting it to the next convolutional layer [8]. The second layer uses no padding hence the data shape shrunk from 21 to 1 with signals from all channels at one time got convoluted. The next average pooling layer does not use padding either since it would not provide much benefit in this case. The pooling kernel for averaging is set to (1, 2) with stride of 2, and this results in shortened data length with 0.5 scaling.

After the feature maps have been flattened, these flattened transformed EEG data are inputted to the first dense or fully connected layer. This layer consisted of 32 units of neurons and is activated using Exponential Linear Unit (ELU) instead of the usual Rectified Linear Unit (ReLU). ELU has more advantage due to its mathematical formulation compared to ReLU in terms of preventing dead weights [16]. Instead of rectifying all negative values, ELU attenuates it according to this formula below [17].

$$\varphi(\boldsymbol{x}) = \begin{cases} x^i, x^i > 0 \\ \alpha\,(\exp(x^i) - 1), x^i \leq 0 \end{cases} \quad (1)$$

The last dense layer contained 5 units that represent each class in the dataset. Since this task involved belongs to multi-class classification, Softmax is employed as the activation function. The following Table 2 contains the summary of the CNN architecture and its parameter configuration explained in this section. Again, n signifies the input data length, which means it could be either 150 or 170. As explained before in the data augmentation subsection, input data length may differ depending on whether data augmentation was applied or not and method used for it.

Classification is separated according to the subjects considering that EEG signal may have different characteristics for each different person. While the whole dataset itself has almost 18 thousand of EEG epochs, separating them per subject made the train set shrunk in size quite drastically, hence the use of augmentation in this study. Although this problem is already addressed as augmentation has been implemented, another problem at hand is that for most subjects the data distribution for each class is uneven. For this problem, stratified k-fold cross validation, an improvement of the conventional cross validation that takes the data distribution in each class into consideration [18], is employed instead of the conventional cross validation. Stratified cross validation tries to maintain the data proportion for each class as the whole dataset is split, which is why this evaluation method is recommended especially for dataset with imbalanced data distribution such as this one. In this study, the k defined as 10 hence the 9:1 train and test set ratio. Number of training epochs used varies depending on each experiment, but still, training hyperparameters such as learning rate and batch size are set to the same value. The optimizer used in this study is Adam, learning rate is set to 1 x 10-5, whereas batch size is set to 16.

Table 1. Evaluation Metrics Used.

Name of Metric	Definition	Calculation
Accuracy	Ratio between all correct predictions and all predictions made	$a = \frac{TP + TN}{TP + FP + TN + FN}$
Precision	Ratio between true positive predictions and all positive predictions	$p = \frac{TP}{TP + FP}$
Recall/Specificity	Ratio between true positive predictions and all correct predictions	$r = \frac{TP}{TP + TN}$

Right after the training for one fold of cross validation wraps up, the model's performance is evaluated on the test set to see how well the CNN model could generalize what it learns from the train set to the test set. Some metrics used for this are accuracy, precision, and recall/specificity, complemented with confusion matrix to further analyze how well could the model make a prediction. While accuracy is quite self-explanatory, precision and recall are chosen since we want to specifically

see the model's performance for true positive predictions. The formulation for these evaluation metrics could be seen on Table 1. TP stands for true positive, TN for true negative, FP for false positive, and lastly FN for false negative.

Table 2. CNN Specifications and Its Hyperparameter Configurations.

Layer name	Input shape	Hyperparameters	Output shape
Conv2D	(21, *n*, 1)	• Kernel size: (1, 5) • Number of filters: 64 • Padding: 'same' • Activation function: ReLU • Kernel initializer: Glorot Uniform, seed 42	(21, *n*, 64)
Conv2D	(21, *n*, 64)	• Kernel size: (21, 1) • Number of filters: 64 • Padding: 'valid' • Activation function: ReLU • Kernel initializer: Glorot Uniform, seed 42	(1, *n*, 64)
AveragePooling2D	(1, *n*, 64)	• Pool size: (1, 2) • Stride: 2 (default value)	(1, *n*/2, 64)
Flatten	(1, *n*/2, 64)	-	*n*/2 x 64
Dense	*n*/2 x 64	• Number of units: 32 • Activation function: ELU • Kernel initializer: Glorot Uniform, seed 24	32
Dense	32	• Number of units: 5 • Activation function: Softmax • Kernel initializer: Glorot Uniform, seed 24	5

4. Result and Discussion

4.1. No Augmentation Experiment

Initially for this experiment, the number of training epochs was set to 30. As seen in Figure 4. below, the model took around 25 training epochs to reach 100% training accuracy, but from the decreasing validation accuracy that started after 15 training epochs, it was highly possible that overfitting had occurred rather early in this experiment. This problem could be solved by using smaller training epochs for future training, hence for the stratified cross validation, 25 was used instead of 15. While the graph shows that 25 is still too large to prevent overfitting, note that this couldn't be generalize to all subjects since every subject has different amount of data. For training epochs smaller than 25, most training processes have not reached convergence yet which is why 25 is chosen.

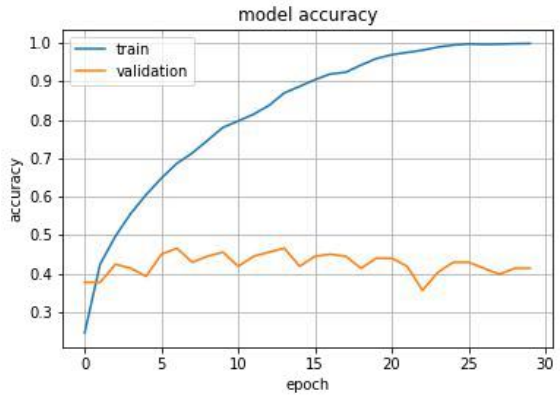


Figure 4. Training progress for no-augmentation experiment on subject G. The epoch on the *x* axis refers to training epochs or training repetition, not EEG epochs.

The result from stratified cross validation shows that this CNN model managed to achieve an averaged accuracy and recall of 47% with 9.4% standard deviation and averaged precision of 47.4% with 9.6% standard deviation. For comparison, Kaya et al. [9] who developed this dataset validated it using Support Vector Machine (SVM), in which the features were Fourier Transform Amplitude (FTA) of the EEG signals. The SVM yielded an averaged accuracy of 43% with 10% standard deviation. While there is indeed improvement from Kaya's SVM, it is not quite a significant one. Nevertheless, this shows that the CNN architecture design is relatively on the right track and the CNN model managed to do its task slightly better than conventional methods.

4.2. Augmentation Experiments

The sliding window experiment used augmented dataset whose size is thrice the original, but for initial testing, 30 training epochs was used to analyze the training progress for this experiment. The result was shown in Figure 5., and apparently there was quite significant improvement in validation accuracy compared to the previous no-augmentation experiment. In contrast, the trend line of training accuracy was slightly worse, but this indicated that the CNN model was not in overfitting state. For this reason, the stratified cross validation used the same training epochs.

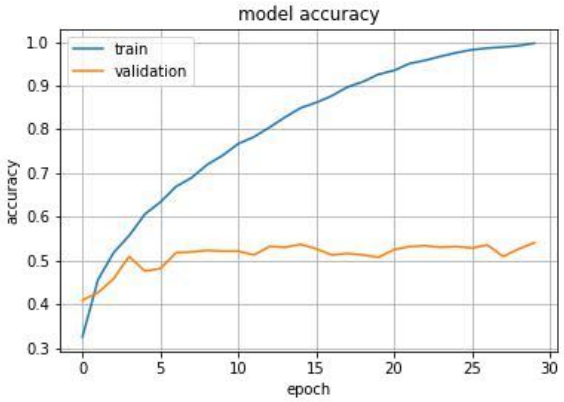


Figure 5. Training progress for sliding window experiment on subject G.

Stratified cross validation yielded result that more or less similar to what is shown on Figure 5.; both averaged accuracy and recall reached 57.1% with 7.9% standard deviation, while averaged precision is slightly higher with 57.8% and 7.8% standard deviation. This shows that sliding window technique provides quite a tremendous contribution in helping the CNN model generalize the train set into the test set. One possible explanation for this is with how sliding window works, it manages to frame a particular finger's MRP which is one EEG feature in time domain that plays a big role in both imaginary and executed movement [19, 20] in various position, therefore helping CNN's ability to generalize data.

For the noise addition experiment, 30 training epochs is also used for initial testing. Interestingly enough, the training progress in Figure 6. shows opposite result relative to that of sliding window. The CNN model only needed around 10 training epochs to reach 100% training accuracy, yet there was not much improvement on the validation accuracy compared to the model's performance without the help of augmentation. For the implementation of stratified cross validation, the number of training epochs used was reduced to 10.

Stratified cross validation's result more or less reflected what is shown on Figure 6. The CNN model only managed to achieve 47.2% of averaged accuracy and recall with 9.9% standard deviation. On the other side, the averaged precision achieved was slightly higher; 48.6% with 9.7% standard deviation. This implies that noise addition technique helps the model better in scoring less false positive predictions, although with lower accuracy scored, this improvement may not have that much of significance.

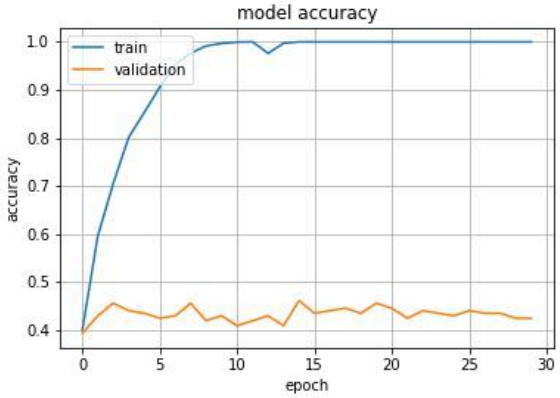


Figure 6. Training progress for noise addition experiment on subject G.

One possible reason for fast learning process that occurred during the noise addition experiment is that although the amount of training data had multiplied thrice its actual amount, the new data resulted from augmentation did not really count as new data that provides new information to the CNN, but rather a repetition of information that has been learned from the original data. Considering that noise addition simply alters input signals' amplitude without doing anything to MRP positions, this is a quite plausible explanation.

One of the reason for the addition of combine augmentation method is to solve the two previous augmentation methods' problem. Sliding window helps the model performs it task more accurately, but it requires a relatively large number of training epochs. On the other hand, noise addition is rather subpar in terms of model performance, but by applying it together with sliding window, the number of training epochs required may be cut down to lesser training epochs.

Since the amount of data resulted from this combined method is much bigger than the previous ones--nine times the original amount; the number of training epochs is reduced to 10. Figure 7. shows the training progress for this experiment. As expected, the training accuracy increased much faster compared to the sliding window experiment with the validation accuracy also shows higher result if not similar. For stratified cross validation, the number of training epochs used would not be changed.

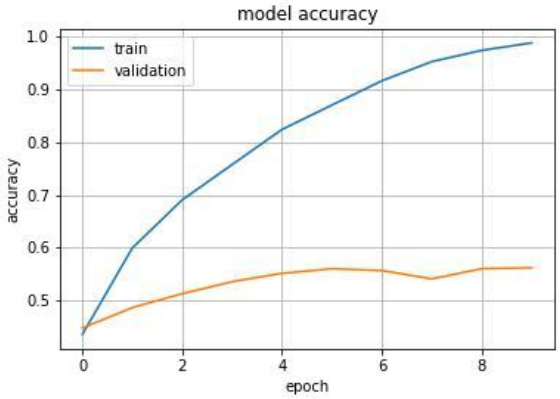


Figure 7. Training progress for noise addition experiment on subject G.

The averaged accuracy and recall obtained from stratified cross validation managed to achieve 57.5% with 7.9% standard deviation; only 0.4% higher than sliding window's result. Meanwhile the averaged precision was exactly the same although there was a slight increase in standard deviation; 57.8% and 7.9%. While the performance indeed was similar to sliding window, it seemed that the higher precision found in noise addition did not extend to this combined method experiment.

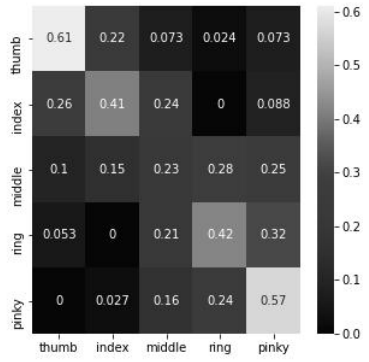
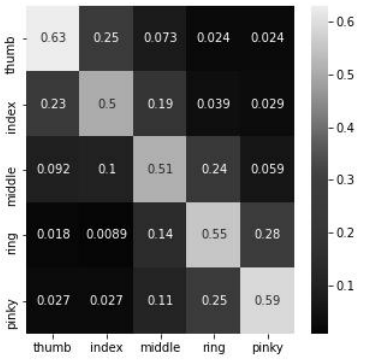
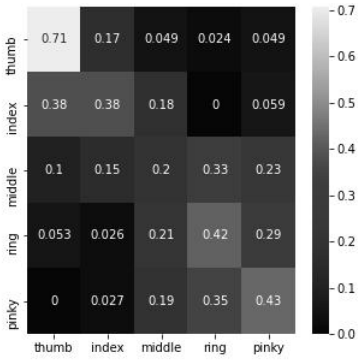
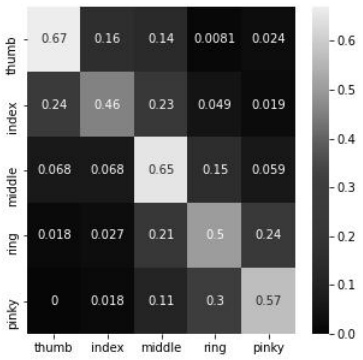
As a summary, Table 3. contains the result of all experiments. Table 4. contains the confusion matrices taken from subject G for further analysis purpose.

Table 3. Classification Results of All Experiments.

Experiment	Evaluation Metrics					
	Accuracy (%)		Precision (%)		Recall (%)	
	Avg	Std. Dev	Avg	Std. Dev	Avg	Std. Dev
No Augment	47	9.4	47.4	9.6	47	9.4
Sliding Window	57.1	7.9	57.8	7.8	57.1	7.9
Noise Addition	47.2	9.9	48.6	9.7	47.2	9.9
Combined	57.5	7.9	57.8	7.9	57.5	7.9

Although the confusion matrices included here are from subject G only, these matrices could represent the classification result of all subjects in general. From the matrices, it could be seen that for no augment and noise addition experiments, the CNN model managed to avoid misclassification at all for some classes. Meanwhile for the sliding window and combined method, the number of predictions proven to be true indeed was higher, but misclassification is generally unavoidable. It could be infer from this fact that originally the CNN model is better at scoring true negatives rather than true positive which is proven by the result shown from no augmentation and noise addition experiments. Unfortunately, scoring more true positives is more desirable in this study. Sliding window in general could help the CNN model earn more true positive predictions although in return the number of true negative predictions decreased.

Table 4. Classification Results of All Experiments.

No Augment	Sliding Window
	
Noise Addition	**Combined Method**
	

5. Conclusions

In general, the CNN model used in this study has performed better compared to the conventional machine learning approach used in the previous study. Though there is not much significant improvement specifically in the no augmentation experiment, the result aligned with the statement that separating feature extraction and classification leads to lower classification accuracy. Besides, the CNN architecture especially the feature extraction part was basically a simplified version of FBCSP which is known for its excellent result in MI EEG classification. Further improvement on this part may lead to even better performance.

Aside from that, deep learning approaches are known to work better with larger dataset, which is unfortunately quite lacking in this study. The various augmentation experiments have shown that indeed with more data provided to the CNN model, the model performance increased quite tremendously. Of all the augmentation techniques employed, sliding window is the most appropriate technique for this dataset as it helps the model to reach 57.1% averaged accuracy. Combining sliding window with noise addition does increase the averaged accuracy to 57.5%, although more importantly it cut down the number of training epochs necessary to train the CNN properly.

References

[1] Nicolas-Alonso L F and Gomez-Gil J 2012 Brain-computer interfaces, a review *Sensors* **12** 1211-79

[2] Wolpaw J R, Birbaumer N, McFarland D J, Pfurtscheller G and Vaughan T M 2002 Brain-computer interfaces for communication and control *Clinical Neurophysiology* **112** 767-91

[3] Zebari R R, Abdulazeez A M, Zeebaree D Q, Zebari D A and Saeed J N 2020 A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction *Journal of Applied Science and Technology Trends* **1** 56-70

[4] Tang Z, Chou L and Sun S 2016 Single-trial EEG classification of motor imagery using deep convolutional neural network *Optik* **130** 11-18

[5] Cireşan D, Meier U and Schmidhuber J 2012 Multi-column deep neural networks for image classification *IEEE Conf. On Computer Vision and Pattern Recognition* 3642-49

[6] Lee B, Jeong J, Shim K and Lee S 2020 Classification of high-dimensional motor imagery tasks based on an end-to-end role assigned convolutional neural network *arXiv* 2002.00210

[7] Schirrmeister R T, Springenberg J T, Fiederer L J F, Glasstetter M, Eggensperger K, Tangermann M, Hutter F, Burgard W and Ball T 2017 Deep learning with convolutional neural networks for EEG decoding and visualization *Human Brain Mapping* **38** 5391-420

[8] Dose H, Møller J S, Iversen H K and Puthusserypady S 2018 An end-to-end deep learning approach to MI-EEG signal classification for BCIs *Expert Systems with Applications* **114** 532-42

[9] Kaya M, Binli M K, Ozbay E, Yanar H and Mishchenko Y 2018 A large electroencephalographic motor imagery dataset for electroencephalographic brain computer interfaces *Sci Data* **5** 10.1038/sdata.2018.211

[10] Mishchenko Y, Kaya M, Ozbay E and Yanar H 2019 Developing a three to six-state EEG-based brain-computer interface for a virtual, **66**, 977 -87

[11] Pfurtscheller G and Lopes da Silva F H 1999 Event-related EEG/MEG synchronization and desynchronization: basic principles *Clinical Neurophysiology* **110**, 1842-57

[12] Diana N E, Kalsum U, Sabiq A, Jatmiko W and Mursanto P 2016 Comparing windowing methods on finite impulse response (FIR) filter algorithm in electroencephalography (EEG) data processing *Journal of Theoretical and Applied Information Technology* **88** 558–67

[13] Lashgari E, Liang D and Maoz U 2020 Data augmentation for deep learning-based electroencephalography *Journal of Neuroscience Methods* **346** 10.1016/j.jneumeth.2020.108885

[14] Wang F, Zhong S, Peng J, Jiang J and Liu Y 2018 Data augmentation for EEG-based emotion recognition with deep convolutional neural networks *MMM* 82-93 10.1007/978-3-319-73600-6_8
[15] Ahmed W S and Karim A A 2020 The impact of filter size and number of filters on classification accuracy in CNN, *International Conference on Computer Science and Software Engineering* 88-93 10.1109/CSASE48920.2020.9142089
[16] Nwankpa C E, Ijomah W, Gachagan A and Marshall S 2018 Activation functions: comparison of trends in practice and research for deep learning *arXiv* 1811.03378v1
[17] Clevert D, Unterthiner T and Hochreiter S 2016 Fast and accurate deep network learning by exponential linear units (ELUs) *arXiv* 1511.07289v5
[18] Kohavi R 1995 A study of cross validation and bootstrap for accuracy estimation and model selection, *International Joint Conference on Artificial Intelligence* 1137–43
[19] Cunnington R, Iansen R, Bradshaw J L and Phillips J G 1996 Movement-related potentials associated with movement preparation and motor imagery *Exp Brain Res* **111** 429-36
[20] Cunnington R., Iansen R, Johnson K A and Bradshaw J L 1997 Movement-related potentials in Parkinson's disease: motor imagery and movement preparation *Brain* **120** 1339-53