Exploring Brain Dynamics: An LSTM Approach To MEG Data Analysis

BERBER VAN DRUNEN (6396410), DEAN NEWAR (2755858), FREDERIEKE BLOM (6433294), JENS VAN DER WEIDE (2492520), JOEKE WOLTERBEEK (6798942), JOS KISJES (2675293)

In this paper, we explore brain activity using a Long Short-Term Memory (LSTM) model to learn patterns in neuroimaging technique Magnetoencephalography (MEG) data. MEG is used to capture brain wave activity by looking at the magnetic field on the skull. Machine learning has often been used to analyze EEG data, this research explores the less-researched MEG data. The sequential nature of MEG and the ability of LSTM models to capture long-term relations pointed us towards the usage of an LSTM model. The models were created separately for the intra-subject and cross-subject classification, to study the difference between the two approaches. While results could be improved upon, the final models perform better than the base models. Moreover, intra-subject classification has shown to be a more accurate approach for this kind of sequential data. Cross-subject classification remains challenging, mainly for complex brain wave patterns.

The code and relevant data are available at: https://github.com/berbervd/PRDL-assignment-2

1 INTRODUCTION

Magnetoencephalography (MEG) is a neuroimaging technique to measure brain activity based on the brain's magnetic field. Varying activity in the neuronal activity causes small but measurable temporal changes in the magnetic field (in the order of femtotesla). Such changes are picked up by the sensors on the scalp, processed and finally used to infer the subjects' actions, such as resting or solving mathematical problems.

Recently, machine learning techniques have seen an increase in popularity for their efficient and accurate classification of tasks using sequential data, including the inference of cognitive information from brain data. Previous lines of research emphasized employing ML models using data from electroencephalography (EEG) (e.g. Gupta et al. (2022)). This research focuses on the counterpart, MEG data. This data is used to classify the subjects' mental actions using deep learning methods.

For the task, we have chosen to employ a neural network (NN) architecture. Specifically, a recurrent neural network (RNN) with Long Short-Term Memory (LSTM), for its capabilities to address complexities in brain wave data analysis. This choice was made for the structural advantages of LSTMs for sequential data. The MEG data gathered for this research is of a sequential nature. As found by Pham (2021), the effectiveness and accuracy of LSTM models in classifying MEG data, is high, whilst also saving time for data learning, and being cost-effective.

XXXX-XXXX/2024/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The question we aim to answer in this research is the following: can an LSTM model accurately classify mental tasks based on time-series MEG data, and is there a difference in model performance when classifying intra-subject and cross-subject data?

The structure of the paper is as follows: First, we look at existing literature on classifying brain activity. Then, we highlight LSTMs and their various properties. Later in the methods section, we show our research set-up, the two classifications, and our choices of hyperparameters and LSTM adaptions. In the results, we show the optimal found parameters for the models and their performance results. Lastly, we further analyze our results by comparing the models for the intra and cross-data sets, discuss the strengths and limitations of our research and highlight interesting research opportunities.

2 RELEVANT LITERATURE

NN architectures have been successfully employed to learn patterns in data from brain activity in previous studies. One study employed various combinations of convolutional, recurrent and attention layers to classify motor/cognitive tasks based on the MEG data (Abdellaoui et al., 2020). While this architecture shows promising results, the focus was on the effectiveness of attention layers. Another study used a convolutional neural network (CNN) to classify different motor tasks using EEG data (Lun et al., 2020).

Several studies leveraged the capabilities of LSTM models to analyse M/EEG data. One study employed an LSTM network to localize the neural sources of EEG signals (Hecker et al. (2022)). Similarly, Dinh et al. (2021) employed an LSTM model to estimate the source of MEG signals in the brain. Finally, Kumar et al. (2019) used an LSTM network to classify motor tasks using EEG data. While these studies focused mainly on either localization problems or the integration of spatial aspects in classification, they all mention that LSTM-like architectures are well-suited for temporal data. Finally, nearly all models used in previous literature were enriched with fully-connected dense layers to increase model complexity.

The problem – classifying various tasks based on temporal MEG data – and proposed solution – using an LSTM-like network – that this research addresses has thus received limited attention in previous literature.

2.1 LSTM preliminaries

This section aims to provide a brief background on LSTM architectures and, as such, provide a theoretical justification for why this is a well-suited architecture for the classification task at hand.

Long Short-Term Memory models, as originally introduced by Hochreiter and Schmidhuber (1997), differ from regular fully connected layers by replacing single neurons with 'cells'. The main functionality of these cells is to retain or discard information from previous inputs to the network (i.e. data from earlier time steps). As such, the cell network can capture long or short-term temporal relations between the data. As will be discussed in 3.1, the data at hand is a time series: several matrices corresponding to different tasks, each matrix consisting of 248 information channels in the rows and time-steps in the columns. To classify such sequential data, it is important to capture the temporal relations: brain activity at the end of the task might depend on the activity at the beginning of the task.

The structure of the data underlines the importance of temporal context, rather than spatial context (i.e. relations between sensors/information channels). The latter (and short-term temporal contexts) can be captured by a CNN, but we have decided to focus on just the temporal dimension of the data.

3 METHODOLOGY

3.1 Data description

In this section, the systematic approach employed to preprocess, analyze, and model the data will be outlined. The data itself consists of multiple files. Each file is one scan where the participant was performing one of the four tasks: resting, math & reading task, motor task, or working memory task. The results of each scan are a matrix with a shape of 248x35624, where the rows highlight the sensors used, and the columns are the total time steps in one scan. This gives the data a time-series character: data collected over a certain time period, in this case, 17.5 seconds, and the entries are in sequential order. This allows for the extraction of temporal features that capture the dynamic changes in brain activity associated with each task. Finally, it should be noted that the data is balanced: all labels occur equally.

The objective of this study is to compare 'intra' and 'cross' conditions, all pre-divided into train and test sets already. Intra is data gathered from one subject: 32 train samples and 8 test samples. Cross is data gathered from multiple subjects: 64 train samples and 48 test samples. Importantly, the test samples are different subjects than the train data in the cross condition.

3.2 Data preparation

In this section, the approach to preprocessing the data will be outlined. First, the data is normalized. We opted for time-wise z-score normalisation. With this, the normalization of the data is done 'per sensor' (i.e. per row), which ensures a distribution with a mean of 0 and a standard deviation of 1 in the temporal dimension. There is no need for any normalization across the different sensors since we are only looking at temporal relations in the data.

Next, the data is downsampled. With this technique, the model can be trained faster, and while it may result in some information loss due to the scaling down of the dataset, the benefits of expected improvements in training time and memory management outweigh these negatives. Our team has opted for a systematic sampling approach: a subset of the dataset is selected at regular intervals. This method was selected because it still enables a representative and efficient utilization of the data. The interval rate selected is 2, thus halving the dataset. Since labels were not registered automatically to each data matrix, they had to be extracted from its file name, done by matching the file name to a predefined task and mapping it to a numerical value for the training (and mapped back later on).

Since the datasets were ordered and could potentially induce bias in the learning process, the files were shuffled before they were extracted. The files were already split into training and testing sets, however for the cross data there were three test sets, which is why we used one as a validation set and merged the remaining two sets as one test set.

3.3 Experimental setup

The experimental setup is as follows: first, we designed a 'base' LSTM model by tuning the hyperparameters to find an optimal configuration. This model will give an idea of how a simple LSTM architecture will perform. To further improve, we have extended the hyperparameter search by allowing dense layers after the LSTM layers. As described in section 2, this is a common practice to add more complexity to the model. Finally, ten different models will be trained with the best hyperparameters to review the variance in training. All of the aforementioned steps will be performed for both the cross and intra-datasets separately.

3.3.1 Model implementation.

The entire implementation of the different LSTM models was done using TensorFlow/Keras. Stacking various layers in the Sequential model gives us a flexible way to build and adjust different models throughout the experiment. The model compiles using the Adam optimization algorithm with a variable learning rate, which will be tuned later on. A Categorical Cross entropy loss function for calculating the loss fits the multi-class classification problem at hand. For training, standard parameters were initially adopted for both epochs and batch sizes, 50 and 8, respectively. We allowed for an 'early stop' callback, which is used to reduce the risk of overfitting. It terminates the training process if there is no substantial loss improvement throughout 5 epochs.

The training of the smaller models was done locally on a CPU. For bigger models / multiple runs, we used Google Colab's TPUs to speed up the training.

3.3.2 Model Tuning.

The next step is to find the right configuration of the models. We determined the various hyperparameters in the different models using a random search, which includes not only the determination of the number of layers but also the fine-tuning of several hyperparameters. The Random search tuner employs ten trials, each trial having three executions for proper exploration.

The tuning process consists of two main components: architecture tuning and hyperparameter tuning. The architecture tuning consists of finding the right combination and number of layers in both the base and the improved model. For the former, this means finding the right number of LSTM layers, while for the latter, this means finding both the right number of LSTM layers and dense layers, as well as the optimal number of units per layer. Hyperparameter tuning is carried out to further optimize the performance of the different model architectures. The tuned hyperparameters in this study include the learning rate (selected from [0.01, 0.001, 0.0001]), LSTM dropout rates (ranging from 0.1 to 0.5), and batch size during training. The learning rate and batch size affect the speed of the learning process, while the dropout rate mitigates overfitting.

3.3.3 Evaluation.

Evaluation is carried out by comparing the performance of the improved models with the best hyperparameters. To further investigate stochastic elements in the training process, we trained ten different models with the same best-performing hyper-parameters on both the intra and the cross-datasets separately. Additionally, the precision, recall, and f1-scores of the best-performing models are calculated to compare performance on the cross and intra-data sets.

A final note on the comparison of intra and cross is in place. We opted for different models across both conditions to achieve the best possible performance. Since both data conditions are significantly different, we thought it justified to tailor the models to the data, instead of using a fixed architecture. This not only allows for the best performance, but the results of the model architecture give indirect insight into the complexity needed to capture patterns in the different data conditions. Importantly, the experimental setting that was held fixed, was the search space for the models.

4 RESULTS

In this section, the results of the methods described in the previous chapters will be discussed. First, the results of the intra-subject classification will be discussed, by providing the results from the base model briefly, before continuing with a more in-depth analysis of the improved model. The same is done for the cross condition. Finally, the classification results of both conditions are compared.

4.1 Intra

4.1.1 Model results. The base model random search yielded these parameters as optimal three LSTM layers with units of 128, 16, and 128. A learning rate of 0.001 was used, and a dropout of 0.2.

After fine-tuning the hyperparameters using RandomSearch, we identified the optimal configuration for the LSTM model. This set of hyperparameters includes three LSTM layers, with 128 units for the first layer, 16 units for the second, and 64 units for the third, as can be seen in table 1. The best dropout rate was found to be 0.3, which helps prevent overfitting. Then, three dense layers are employed, with 32 units in the first layer, 16 units in the second, 128 in the third, and 32 units in the fourth layer. After finding these parameters, ten training runs were executed to find the optimal weights and give a better assessment of the model's performance, considering the variability in the training process.

Layer (type)	Param #	Hyperparameters
LSTM	193024	units=128, dropout=0.3
LSTM	9280	units=16
LSTM	20736	units=64
Dense	2080	units=32
Dense	528	units=16
Dense	2176	units=128
Dense	516	units=4, activation=softmax

Table 1. The layers and parameters of the improved intra-classification LSTM model. It contains three LSTM layers, three dense layers and one dense output layer.

Layer (type)	Param #	Hyperparameters
LSTM	35968	units=32
LSTM	8320	units=32
LSTM	82432	units=128
Dense	16512	units=128
Dense	16512	units=128
Dense	16512	units=128
Dense	16512	units=128
Dense	4128	units=32
Dense	132	units=4, activation=softmax

Table 2. The layers and parameters of the improved cross classification LSTM model. It contains three LSTM layers, five dense layers and one dense output layer.

4.1.2 Performance results . The results of the best model's performance are listed in the confusion matrix in Figure 1. We see that the model was able to predict five out of the eight test files correctly, resulting in a test accuracy of 62.5% and a train accuracy of 43.7%. If we compare the results to the base model's test accuracy of 37.5%, then the updated model is an improvement for this classification task. There is thus some benefit from increasing the complexity of the model and adding dense layers for the intra-condition.

From the confusion matrix, it becomes clear that the model was able to correctly predict the resting task and motor task, but got it completely wrong for the story & math task, and also made a mistake for the working memory task. It turns out that the current model is not all too well equipped to generalize the data even within one subject. This might be due to the model's architecture, but the limited data samples could also be a factor. However, further tweaking of the architecture can provide promising results.

4.2 Cross

4.2.1 Model results. The base model random search yielded these parameters as optimal: three LSTM layers with units of 256, 64, 128, and 128. A learning rate of 0.0001 was used.

The improved cross-subject classification model had a more complex structure, with three LSTM layers and five dense layers. Optimal hyperparameters were identified through random search. They included 32 units for LSTM layers, with a dropout rate of 0.2. The kernel regularizers were put at 0.001 for the LSTM layers, as well as the dense layers. The first three dense layers had 128 units each, while the remaining two layers had 32 units. The full overview of the architecture of this model can be seen in table 2. Throughout the model, a consistent learning rate of 0.001 was maintained. The development of the improved structure led to better performance.

4.2.2 Performance results. The performance of our base model showed low precision and recall across all classes. The accuracy was 28%. The model had trouble identifying the Math & Story class, which it was completely unable to classify correctly.

The improved cross-subject classification model gave significantly better results, as can be seen in the confusion matrix in Figure 1. The average training accuracy of the improved model is 58%. However, the average test accuracy is still low at 29%, with the best model test

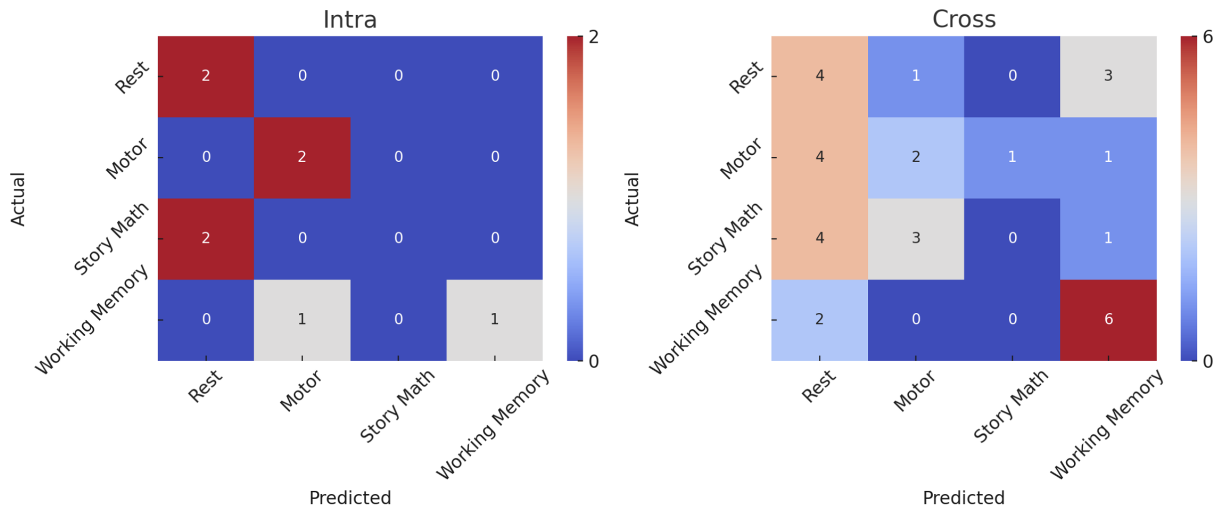

Fig. 1. Confusion Matrices. On the intra (left) data set, it is notable that the model has trouble predicting the story/math and the working memory tasks. On the cross (right) data set, the corresponding model accurately predicts working memory and often misclassifies other tasks as rest

accuracy hitting 38%. The confusion matrix indicates a persistent difficulty in identifying the Math & Story class with accuracy, once again being completely unable to classify it correctly. The model did however show an increased performance on the Working Memory class, classifying 6 instances out of 8 of this class correctly. This suggests a better fit for this class in particular. The predictions for the Rest class were scattered almost uniformly over the true labels, signifying that predictions for this class were made seemingly at random. This may mean that no good fit was found for this class.

From these test results, combined with the fact that the train accuracy is 65.6%, we can conclude that the cross model overfitted. This is likely due to the high complexity of the model. Attempts to mitigate this, the regularization and the dropout, have not proven effective enough.

4.3 Comparing intra and cross

In this section, we perform a comparing analysis of both the base models and the improved LSTM models for cross- and intra-classification tasks before proceeding with their evaluation. We chose not to perform a statistical comparison between these models. This decision is based on the notion that the data and final model structures are different for all variants; however, they share a common underlying framework. Therefore, we chose to evaluate them by comparing their performance metrics with relevant background information.

As mentioned earlier, the base models, which consist only of LSTM layers, yield respective accuracies of 37.5% for intra- classification and 28% for cross-classification. Both accuracy rates just overcome what would have been achieved with a baseline model, which would have yielded an accuracy of 25% (due to the presence of four different classes). It is, however, for the base model remarkable that the accuracy of the intra-classification model exceeds that of the cross-classification model. A plausible explanation for this divergence is the nature of the data: intra-classification uses the same subjects in both the training and test data, while cross-classification uses different subjects and could be less likely to capture subject specific patterns in the data.

Before turning to a detailed analysis of the model differences and possible underlying reasons, let us first look at the results of the improved models for both classification methods. Table 3 shows the performance rates of both models. It is immediately clear that the accuracy of both improved models has improved over their respective base models. The integration of dense layers after the LSTM layers has thus proven beneficial, justifying the characterization of these models as "improved." The accuracies have increased notably (from 37.5% to 62.5% for intra and from 28% to 37.5% for cross), but they are still relatively low.

Looking at the other performance metrics, it becomes clear that predicting the story-math task is challenging for both models, as shown in figure 3. This observation is further supported by the confusion matrix in figure 1, where it can be seen that the label story and math were predicted only once in the cross-classification task and not at all in the intra-classification task. This is remarkable given the balanced nature of the dataset. The low score for the story and math task may be due to the complexity of the classification or to the possibility that there are fewer distinguishing features available for the model to learn from. Looking at the confusion matrix, figure 1, there seems to be a similarity in the frequency of predicting the rest state in both models, which as a prediction occurs most often. This may be because the rest state overlaps with all other states, leading to more frequent predictions.

To continue with the comparison of the models themselves, they are based on the same principles (LSTM layers, dense layers and an output layer with four units). Both models contain three LSTM layers, however, the cross-classification model contains two additional dense layers due to the tuning of the model. A possible explanation for this addition of layers could be the complexity of the cross-classification data. However, on the other hand, it could also lead to a higher risk of overfitting the model. Overfitting is indeed evident in the case of cross-classification, as noted above in section 4.2. This is supported if we examine the training and testing accuracies of both models, as shown in Figure 2. In Figure 2 (b) for cross-classification, we see that in all ten runs, the training accuracy is consistently greater than the testing accuracy. This is a clear indication of overfitting the model. It is, nevertheless, noteworthy that the test accuracy remains relatively stable in all runs compared to the accuracies observed in the intra-classification task.

The over-fitting of the cross-classification model results in the difficulty of generalization and this is further enhanced by the fact that the data used for cross-classification comes from different individuals for the training, validation and testing. Therefore, the low performance of the model raises doubts about its ability to predict data from new subjects. Intra-classification, on the other hand, may also face challenges because the model is trained and tested only on data from a single individual. As a result, performance on new, previously unseen data subjects may also be at risk.

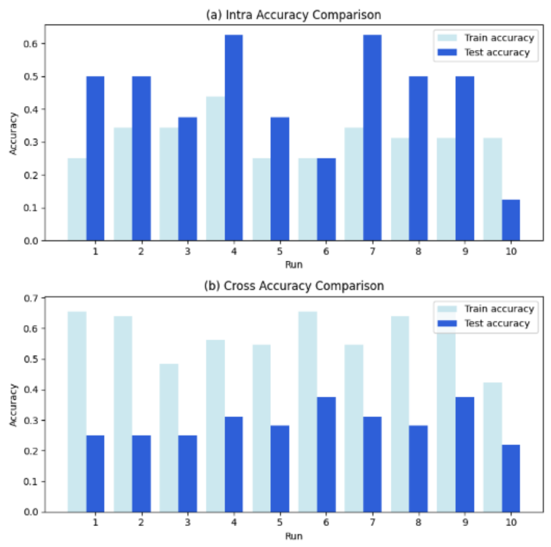

Fig. 2. Accuracy comparison for training and testing across ten runs for both (a) Intra-accuracy comparison and (b) cross-accuracy comparison. The accuracy rates vary across the runs, indicating fluctuations in model performance from run to run, especially in (a). In (b) the train accuracy is consistently higher than the test accuracy for each run, depicted by the lighter bars (train) versus the darker bars (test).

5 LIMITATIONS

This section provides a brief overview of the main limits of this study in terms of data, model architecture and experimental design. It provides suggestions for future work. The amount of data is a notable restrictive factor on the performance of the models. There were limited samples for each model to train on, especially for the intra condition.

While the samples proved limited, the amount of data was high. Training and running multiple big models was computationally costly, restricting the search space for the model configurations. In the future, the data could be downsampled further or more powerful hardware can be used for efficient training.

	INTRA			CROSS		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Rest	0.50	1.00	0.67	0.29	0.50	0.36
Motor	0.67	1.00	0.80	0.33	0.25	0.29
Story Math	0.00	0.00	0.00	0.00	0.00	0.00
Working Memory	1.00	0.50	0.67	0.55	0.75	0.63
Accuracy		62.5%			37.5%	

Table 3. This table shows the performance metrics (precision, recall, F1-scores, and accuracy) for each model under investigation. The accuracy is shown per model (intra or cross), whereas the other performance metrics are shown per predictable task. We notice that the accuracy of the intra-model is higher compared to that of cross-classification. In addition, what stands out is the performance around the story math prediction task, since it scores 0.00 in both models. Overall, the precision, the number of true positive predictions, is higher with intra compared to cross.

Finally, the low performance of the LSTM architecture could form an argument to find other neural network solutions for this problem. Classifying the MEG data might, for example, benefit more from architectures that are less temporally oriented and more spatially oriented. Future studies can further investigate a CNN architecture, which can combine these dimensions and has already proved promising results.

6 CONCLUSION

To summarize, this study analyzes the classification of tasks in intra- and cross-subject MEG data using LSTM networks, to reveal the differences between these approaches. While cross-subject classification is still difficult, with the average accuracy not exceeding 29%, this study shows some results in intra-subject classification, with 62.5% average accuracy compared to a base model of 37.5%. We observed particularly high contrast between the approaches in the prediction accuracy of the math and story class, pointing towards the superiority of the intra-subject approach for classifying more complex brain wave patterns. The performance of improved LSTM models compared to the base models shows the value of enriching LSTM models with other types, like dense layers. While the results of this study may be improved, this study does reveal the challenges in MEG data interpretation and contributes to advancing the exploration of LSTM's potential in understanding brain dynamics through machine learning.

REFERENCES

Abdellaoui, I. A., Fernández, J. G., Sahinli, C., & Mehrkanoon, S. (2020). Deep brain state classification of meg data. *ArXiv*, *abs/2007.00897*. https://api.semanticscholar.org/CorpusID:220301687

Dinh, C., Samuelsson, J. G., Hunold, A., Hämäläinen, M. S., & Khan, S. (2021). Contextual meg and eeg source estimates using spatiotemporal lstm networks [This article is part of the Research Topic "Advanced Imaging Methods in Neuroscience."]. *Frontiers in Neuroscience*, *15*, 552666. https://doi.org/10.3389/fnins.2021.552666

Gupta, A., Kumar, D., Verma, H., Tanveer, M., Javier, A. P., Lin, C.-T., & Prasad, M. (2022). Recognition of multi-cognitive tasks from eeg signals using emd methods - neural computing and applications. *SpringerLink*. https://link.springer.com/article/10.1007/s00521-022-07425-9#citeas

Hecker, L., Maschke, M., Rupprecht, R., Tebartz van Elst, L., & Kornmeier, J. (2022). Evaluation of long-short term memory networks for m/eeg source imaging with simulated and real eeg data. *bioRxiv*. https://doi.org/10.1101/2022.04.13.488148

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kumar, S., Sharma, A., & Tsunoda, T. (2019). Brain wave classification using long short-term memory network based optical predictor. *Sci Rep*, *9*, 9153. https://doi.org/10.1038/s41598-019-45605-1

Lun, X., Zhenglin, Y., Tao, C., Fang, W., & Yimin, H. (2020). A simplified CNN classification method for MI-EEG via the electrode pairs signals. *Frontiers in Human Neuroscience*, *14*. https://doi.org/10.3389/fnhum.2020.00338

Pham, T. D. (2021). Time–frequency time–space lstm for robust classification of physiological signals. *Scientific Reports*, *11*(1). https://doi.org/10.1038/s41598-021-86432-7