EEG Signal Epileptic Seizure Detection with Deep Learning Sequence Processing Units

Paulo Victor Correia (2167525) ,
Ecole Polytechnique de Montréal, Montreal, Canada
paulo-victor.queiroz-correia@polymtl.ca

Abstract

Epileptic seizures affect the quality of life of thousands of people globally. To give quality of life to the people who suffer from them, scientists developed diverse techniques to mitigate the effects of the people on their patients. A few of those techniques revolve around processing Electroencephalogram (EEG) signals to extract features and apply them to deep learning models. In this paper, we propose comparing different deep-learning architectures to detect seizures on EEG signals on the UCI EEG dataset. We are going to use different approaches to self-attention transformers, long short-term memory (LSTM) recurrent networks, and 1D convolutional neural networks, and compare the performance of those architectures for seizure detection. Our results indicate that only self-attention transformer architectures perform worse than architectures with convolutional or recurrent layers, even though transformers are more computationally efficient. Hence, our results indicate that mixing convolutional and recurrent layers are still state-of-the-art and outperform the current methods for time series classification with self-attention transformers.

1 Introduction

Epilepsy and seizure disorders affect around 70 million people worldwide. It is rather a symptom complex than a disorder with a single cause and effect, which comes from a strong genetic predisposition and have multiple risk factors that affect the quality of life of people who suffer from it [5]. The symptoms of epilepsy include loss of consciousness and awareness, temporary confusion and even uncontrollable movements of the body caused by ordered tiny electrical impulses from the brain that causes spasms in different parts of the body [10]. Researchers estimate that children with seizures are at a higher risk for mental health, development and physical comorbidities, which increases the necessity for specialized services and care coordination from diverse sectors of society [4]. Therefore, it's important and urgent to develop methods to quickly and accurately analyze evidence of seizures before they cause much damage to the patient's life, thus helping them improve their quality of life.

One of the ways to identify and study epileptic seizures is through Electroencephalogram (EEG) signal analysis. EEG consists of electrical signals produced by certain areas of the brain. Even though they are more superficial, recording EEG signals is far less invasive than other methods. Also, they provide the most efficient way to localize epileptogenic zones [3]. We can model EEG signals as time series and apply top-notch methods to detect epileptic seizures such as deep learning models.

Deep learning networks have a solid potential to extract useful features for epileptic seizure detection. They have a high abstraction factor that can be useful for multiple data structures. Many researchers use convolutional networks and recurrent neural network models to process time series structures such as EEG and other signals [8; 10; 12]. These deep learning architectures can automatically extract features from time series and already achieve 95% accuracy on the EEG epileptic seizure detection task.

However, convolutional and recurrent networks do not scale decently on larger models in terms of computational efficiency. It isn't easy to parallelize recurrent neural networks because they need to process the data sequentially [7]. Thus they would demand a lot of time and resources to train compared to other structures. Also, recurrent networks also struggle to learn on long sequences, due to the gradient vanishing/exploding. Therefore, researchers have proposed novel architectures like the Long Short-Term Memory recurrent neural network [1] and Transformers [6].

The transformers however are more computationally efficient than convolutional and LSTM networks [7]. They can be parallelized and thus scale better for larger models, especially on language models where they were originally designed for [6]. However, the current state of the art for transformers in time series classification is in an early stage [2] and there are still few projects on transformers and time series classification like [2; 11]. But we do not have a lot of papers using transformers for EEG classification.

Therefore, in this paper, we propose to compare different deep learning architectures to compare the performance of time series classifiers on the EEG seizure detection task. More importantly, we want to compare the performance of transformers in time series classification, analyzing if cur-

rently developed transformer models can keep up with the state of art convolutional and recurrent neural networks. We are going to use the UCI EEG dataset [9] as our source of data. Also, we are going to use the following models: one-dimensional convolution plus LSTM (1D-CNN + LSTM) [10], Fully Convolutional Network (FCN), Gated Transformer Network (GatedTransformerNet) [2], Stacked Encoder Transformer with unsupervised and supervised training, and a multilayer perceptron (MLP) network as the baseline.

The rest of the paper is as follows: Section 2 discusses the background of the paper with the specialized layers for sequence processing; Section 3 describes the dataset used; Section 4 presents a brief description of the deep learning models used; Section 5 describes the experimental setup of the models' training; Section 6 presents and discussed the results of our models; and finally Section 7 draws the conclusions of the present work as well as the future work.

2 Background

We are going to introduce the specific neural network architectures we used in this project. These architectures are Self-Attention Transformers, 1D-Convolutional Layers and LSTM architecture.

Transformers

The transformer architecture proposed in 2017 takes advantage of the attention mechanisms to produce a novel neural network architecture for sequential processing [6]. This novel architecture completely dispenses convolutional and recurrent operations, turning into a more computationally efficient architecture to process sequences as it allows a deeper parallelization [7].

First, the transformers project the input into query, key and value sets using a linear perceptron layer. The attention mechanism then performs the scaled dot-product and takes the projected sets of key and query pairs to an output. And then, we perform a matrix multiplication of this output with the set of values previously projected. Equation 1 describes the attention operation, which takes the product of queries and keys normalized with the squared root of the number of dimensions, apply this to a softmax operation and finally multiplied by the value set.

$$\text{Attention}(Q, K, V) = \textit{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

The transformers split the attention operation into multiple heads before the attention operation, which are equally sized splits of the query, key and value. Then, they concatenate the outputs of the different heads and pass through a linear layer. For the sake of simplicity, we will use only the encoder architecture of the transformer, displayed in Figure 1.

Figure 1 displays the information flow that the inputs pass through a transformer encoder layer. First, it passes through an embedding layer, but we skip them since our inputs are raw time series. However, positional encoding is still necessary since they make the transformer aware of the positions of the samples in the sequence. Equations 2 and 3 describe how the

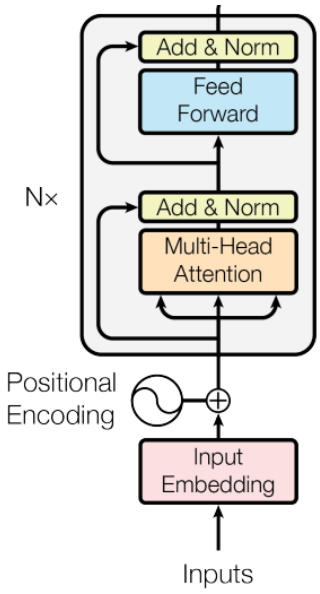

Figure 1: Transformer encoder layer. Source: [6]

fixed positional encoding works for even and odd positions of the sequence.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \tag{2}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \tag{3}$$

After that we pass the encoded input through the described attention mechanism, add the attention output to the input and normalize the result of the addition. Then we pass through a linear perceptron network, add and normalize again. We can stack as many encoder layers as we want to form a deep neural transformer.

LSTM

Researchers created the Long Short-Term Memory Networks to solve a recurring problem of the Recurrent Neural Networks called gradient vanishing or exploding. This gradient problem consists of RNNs that accumulate gradients over time approaching either 0 or 1 which diminishes the model's capability to learn in longer sequences. The LSTM solves the gradient vanishing problem by inserting a perceptron structure named gate that has the capability to forget and select what features it will learn and pass forward to future time steps.

Figure 2 displays the architecture of an LSTM cell. It contains the forget gate, the input gate and the output gate. The cell state in Equation 8 corresponds to a state that will be shared along all time steps, which retains the information sharing and is updated each time step. The hidden state in Equation 9 ponders the final cell state of the current time step with the output gate to determine the actual output of the cell, undermining the gradient problems in previous recurrent neural networks. The input gate and the cell gate in Equations 4 and 5 determine what information will be added to the current cell state based on the previous hidden state and

the current input. The forget gate in Equation 6 ponders the current input with the hidden state from the last time step to select what information from previous time steps will be forgotten. Finally, the output gate in Equation 7 determines the information passed through the next time steps by pondering the previous hidden state and the current input.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (4)$$

$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \quad (5)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (6)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (7)$$

$$c_t = f_t * c_{t-1} + i_t * g_t \quad (8)$$

$$h_t = o_t * tanh(c_t) \quad (9)$$

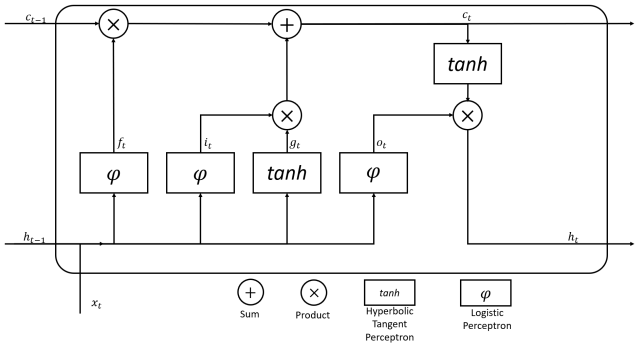

Figure 2: LSTM cell blueprint.

Convolutional Networks

Convolutional Networks have a strong known baseline for extracting features in 2D images. They perform the convolution operation on the input, which works by highlighting useful features for the classification task [8; 10]. Therefore, we use 1D Convolutional networks to filter the univariate EEG signals searching for epileptic seizure samples.

By stacking one-dimensional convolutional networks, we extract even more high-level features from our samples with an even higher discriminating capability for epileptic seizure recognition. Also, it creates more channels on the univariate signals, which can be useful when passing through other architectures.

3 Materials

We extracted the EEG signal from the UCI repository [9]. It consists of univariate EEG time series signals with 178 samples in each one extracted from 500 different patients, labelled in five distinct classes:

1. Epileptic seizure signal
2. Normal condition: patient opening their eyes
3. Normal condition: patient closing their eyes
4. Normal condition: signal extracted from healthy brain areas
5. Normal condition: signal extracted from unhealthy brain areas with tumours

In total, we have 11,500 sequences to classify, equally distributed for all five classes with 2300 time series in each class. To simplify, we are transforming this into a binary classification problem for either an EEG signal displaying a seizure or normal behaviour. Figure 3 display samples extracted from the dataset. According to this figure, the seizure EEG signals in red have greater amplitudes, while the other signal classes look more stationary.

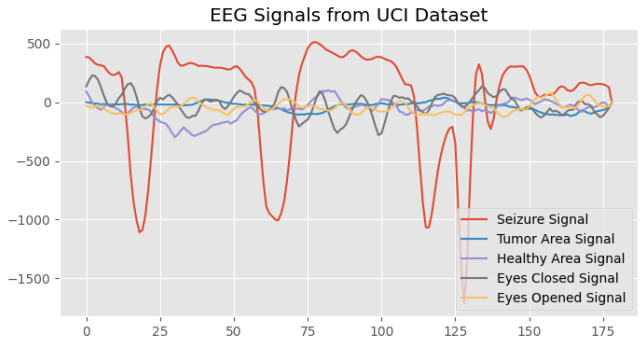

Figure 3: EEG signals for all five classes extracted from the UCI dataset.

4 Models

In this section, we are going to make a brief presentation of the models used in this project, as well as training specificities if there are any.

4.1 Multilayer Perceptron Network

The Multilayer Perceptron Network takes the EEG signal input and passes forward all of its layers. The main disadvantage is that it does not keep track of the order of the samples of the signal nor exploits this sequence.

The MLP architecture used in this project follows the work of [8] beginning with a layer with 500 perceptrons followed by a dropout regularization layer and a ReLU activation function. We then repeat this two more times totalizing a 3 layered MLP model. The output layer consists of a single perceptron activated with a sigmoid function for binary classification.

4.2 FCN

The Fully Convolutional Network uses multiple layers of 1D convolutive operations to extract features from the time series. We also use the proposed architecture in [8], which begins with a convolutive layer with 128 output channels, padding of 1 and a kernel size of 3, followed by a Batch Normalization layer and ReLU activation function. Then for the second layer, we have 256 output channels with padding equal to 1 and a kernel size of 3, also followed by a Batch Normalization and a ReLU activation function. The last convolutive layer has 128 output channels though and the same configurations as the previous layers.

For the final classification layers, we take the output from the last convolutive layer and pass it through a global average pooling. This layer takes the average of the batch and the time series length and fixes the number of channels from the last convolutive layer. This results in a flattened vector with the size of the number of output channels from the last convolution layer, which is 128. The output layer consists of a single perceptron activated with a sigmoid function.

4.3 LSTM

The LSTM model simply encodes the input time series until the last time step. After passing through the whole sequence, we use the last time step output and pass through a single perceptron activated with a sigmoid function for classification.

The architecture of this model follows the same LSTM structure used in [10]. The LSTM then contains 2 layers with 64 neurons in each layer. We then use the output from the last time step on the final classification layer.

4.4 1D-CNN + LSTM

The hybrid architecture with 1D-CNN and LSTM proposed by [10] first takes the sequence input and passes through a convolutive network with 64 output channels, a kernel size of 3 and followed by a max pooling layer with a window size of 2 and stride of 2. Next, we have three convolutive layers with 128, 512 and 1024 output channels respectively, with all layers activated with ReLU function.

We then pass the convolutive layers' output through a linear perceptron network with 256 neurons and pass it forward to an LSTM network. The LSTM network contains 2 layers with 64 neurons in each, with a dropout of 0.2 to avoid quick overfitting.

The final classification layer consists of using the last time step encoded by the LSTM to feed a deep MLP network. The MLP consists of 3 layers with 256, 128 and 64 neurons in each respectively, all activated with ReLU activation function. The last classification layer is a single neuron activated with a sigmoid function. Figure 4 displays a schematic for the hybrid network.

4.5 Gated Transformer Network

The Gated Transformer Network consists of a 2-tower style transformer network. Each tower consists of transformer encoder layers with different structural configurations: the channel-wise encoder and the step-wise encoder [2]. Figure 5 displays the architecture of the gated transformer network.

The Channel-wise encoder projects each time step sample with an embedding layer. This layer consists of only a perceptron layer that takes the input and projects into a higher dimensional space. Then it passes the embedded input through 2 layers of encoder layers.

However, the Step-wise encoder also projects the input time series but it also adds a positional encoding to the layers. Also, this tower performs masked multi-head attention operation, so we add the padding mask to this tower while passing the inputs.

After that, we perform a gating operation with the outputs of both encoders to concatenate their results. The gating operation first concatenates the results of each encoder. Then, it

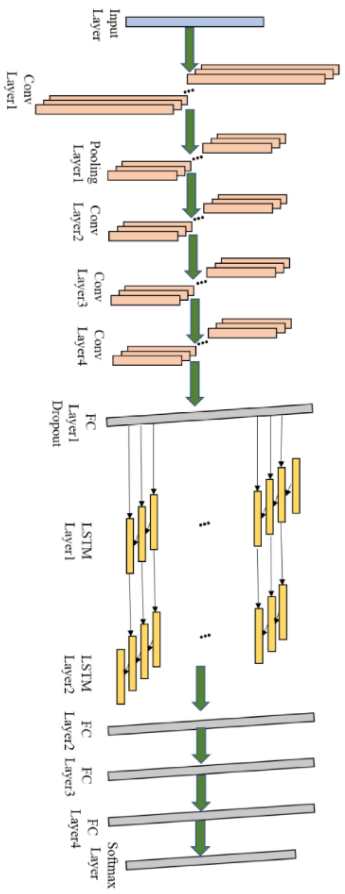

Figure 4: The structure of the 1D-CNN + LSTM model. Source: [8]

projects the concatenation with a perceptron layer of only 2 neurons. This weights the importance of each tower to the final prediction by activating these results with a *softmax* function.

Finally, it multiplies these weights by the encoded output of each tower and concatenates again the weighted encoded value to an output classification layer. The output classification layer consists of a 3-layered MLP model with 256, 128 and 64 neurons each, activated with ReLU activation function and regularized by a dropout layer with a rate of 0.2. We perform the final classification with a single perceptron activated with a sigmoid function.

4.6 Transformer Encoder Classifier

The transformer encoder consists of a single encoder block with 2 transformer layers. Proposed by [12], this architecture projects an input sequence with a perceptron layer. After that, it passes the encoded input through a transformer encoder and a classification output layer.

The differential in this work consists of performing an unsupervised pre-training to better initialize the weights. The model tries to replicate the inputs in the unsupervised pre-training. This makes the model have a better initialization a accelerates the supervised training for classification.

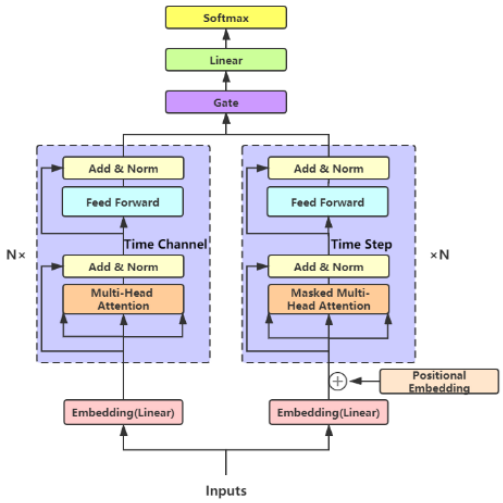

Figure 5: Gated Transformer Network with Step-wise and Channel-wise encoders.

5 Experimental Setup

The preprocessing of our data consists in first splitting the available data into 70% for training the model, 15% to validate the models and 15% to test the models. Finally, we normalized all the time series individually to have a mean of 0 and a variance of 1 to serve as the input of our models.

All models in our project used the Adam optimizer with a learning rate of 0.0001 and 75 epochs of training. We also used a learning rate scheduler which divided the learning rate by 10 if there was no improvement in the validation loss value. The loss function on the supervised training consisted of binary cross-entropy loss with logits since we had a binary classification task at hand. Further, we applied different weights to each sample according to its rarity because we had an imbalanced classification problem, where for each epileptic seizure sample we had four normal samples. Therefore we applied a position weight of 4 in the positive class.

The unsupervised training we used on the model with a single stacked transformer encoder had the mean squared error as the loss function. We also used the Adam optimizer but with a higher learning rate of 0.001 and only 30 epochs. We intended to have a better initialization of the weights on the model to accelerate the training and even obtain better results.

6 Results and Discussion

Figure 6 displays the training loss function over the 75 epochs of training for all of the six model types analyzed. Most of the models during training converged with the given experimental setup. However, MLP had the worst training loss convergence and would not improve with further training. Also, the FCN model had a margin to minimize even further the loss function. Finally, all of the other models had a decent training convergence with values close to zero.

Figure 7 though displays the validation loss results. Despite the optimal convergence during training, most of the models struggled to converge with the validation data set. The

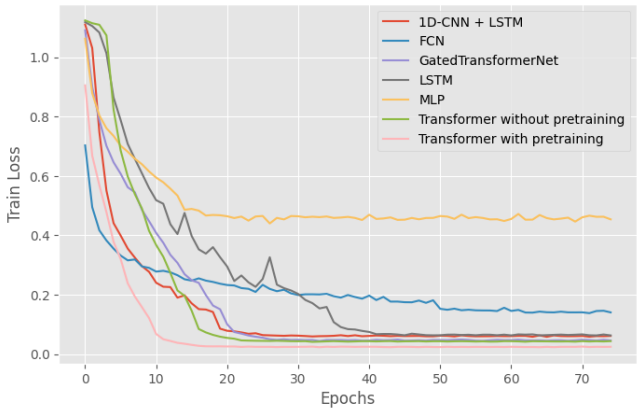

Figure 6: Training loss over the 75 epochs of training.

models with minimal validation loss were the LSTM, the 1D-CNN+LSTM and the FCN model, with the latter achieving the minimal loss with the given experimental setup. All of the other models had a validation loss above 0.5. Therefore, this is strong evidence that the architectures with neither convolutive nor recurrent layers overfitted the EEG time series classification problem.

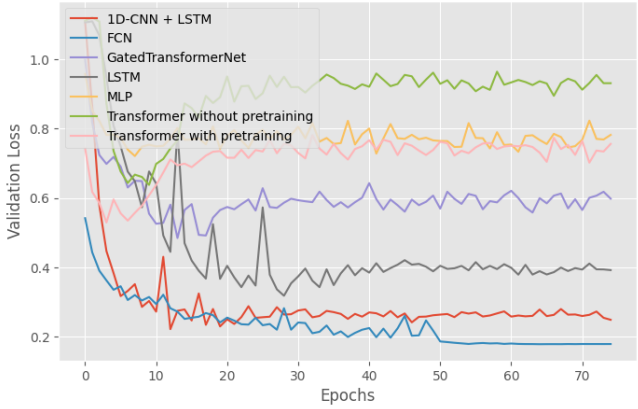

Figure 7: Validation loss over the 75 epochs of training.

And then we have Figure 8 displaying the validation accuracy over the training process. Concerning the validation loss observations, we confirm that the models with convolutive and recurrent layers also outperformed MLP and transformer architectures for time series classification. The transformers were only better than the MLP architecture which does not take into account the order of the samples in each time series, whereas the transformers do.

Finally, Table 1 displays the classification report of all the models in the metrics of accuracy, precision, recall and F1-Score for the test set. These results confirm to us that combining convolutional and recurrent networks outperformed all of the proposed models, with the highest metrics assigned to this model exactly. Also, the GatedTransformerNet outperforms the other transformers' models.

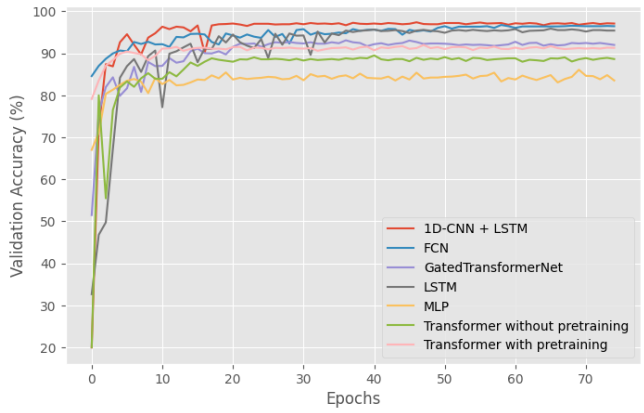

Figure 8: Accuracy evolution over the 75 epochs of training.

Model	Accuracy	Precision	Recall	F1-Score
1D-CNN + LSTM	**97.1**	**92.8**	92.8	**0.928**
FCN	96.9	88.8	**96.8**	0.926
LSTM	96.1	89.2	91.9	0.902
MLP	83.8	57.5	73.0	0.644
GatedTransformerNet	92.4	80.7	81.4	0.811
Transformer pretrained	91.2	79.4	75.9	0.776
Transformer without pretraining	88.3	72.0	68.4	0.701

Table 1: Result metrics for the test dataset.

7 Conclusion

In this work, we explored different architectures to perform binary time series classification for epileptic seizure detection. We used state-of-the-art models with convolutional and recurrent layers for EEG time series classification and compared them to distinct Transformer architectures and other baselines. And after performing several experiments, we could not confirm that Transformer architectures could keep up with state-of-the-art models for the EEG problem. However, we drew some conclusions from the results obtained.

Transformers however have a surprisingly suboptimal performance when compared to LSTM or 1D-CNN, as [7] discusses but in language models. Our results strongly indicate that current transformer models also have a sub-optimal performance over the other layers for time series classification. Thus, it needs further study on optimal architectures for this task if one desires to use more efficient transformers.

Combining convolutional and recurrent layers for time series classification is the state of the art for EEG time series classification. Our results demonstrate that these layers still outperform most of the novel architectures proposed to process time series. However, their scalability and feed-forward speed still struggles when compared to transformers. Hence, we must take this into account when building sensitive classification models such as EEG signal classifiers that can be coupled to devices attached to a person.

Transformers overfit compared to the other sequence processing models. Since transformers cannot extract features from sequences as good as convolutional layers, the given features for attentional operations were not good enough for generalization purposes. Thus, a good approach to transformers in future works would be coupling convolutive layers before the transformers and feeding them with features that allow the model to generalize better.

Pretraining Transformers improved significantly the performance. The single transformer encoder methodology with pretraining significantly outperformed the same model without pretraining. Thus, pretraining has a significant impact on transformers used for EEG time series classification. The next step would be performing the unsupervised training on the two-encoder GatedTransformerNet since it outperformed the single-encoder model.

References

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song. Gated transformer networks for multivariate time series classification. 3 2021.

[3] S. Noachtar and J. Rémi. The role of eeg in epilepsy: a critical review. *Epilepsy & Behavior*, 15(1):22–33, 2009.

[4] S. A. Russ, K. Larson, and N. Halfon. A national profile of childhood epilepsy and seizure disorder. *Pediatrics*, 129(2):256–264, 2012.

[5] R. D. Thijs, R. Surges, T. J. O'Brien, and J. W. Sander. Epilepsy in adults. *The Lancet*, 393(10172):689–701, 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[7] C. Wang, M. Li, and A. J. Smola. Language models with transformers. *arXiv preprint arXiv:1904.09408*, 2019.

[8] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. *Proceedings of the International Joint Conference on Neural Networks*, 2017-May:1578–1585, 2017.

[9] E. Wu, Qiuyi Fokoue. Epileptic Seizure Recognition. UCI Machine Learning Repository, 2017. DOI: 10.24432/C5G308.

[10] G. Xu, T. Ren, Y. Chen, and W. Che. A One-Dimensional CNN-LSTM Model for Epileptic Seizure Recognition Using EEG Signal Analysis. *Frontiers in Neuroscience*, 14(December):1–9, 2020.

[11] Y. Yuan and L. Lin. Self-supervised pretraining of transformers for satellite image time series classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:474–487, 2020.

[12] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff. A Transformer-based Framework for Multivariate Time Series Representation Learning. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2114–2124, 2021.