Pre-Seizure Classification Efficacy with Convolutional Neural Networks, Long-Short Term Networks, and Transformers with Inter-cranial EEG Data

Douglas Schonholtz
Northeastern University, Boston, Masssachusetts, 02115
schonholtz.d@northeastern.edu
Code can be found at: https://github.com/dschonholtz/MLProject

Abstract

State of the art research for EEG data currently published suggests that bi-directional LSTMs used in conjunction with Deep Convolutional Neural Networks (DCNNs) perform the best for classifying pre-seizure EEG data. (Daoud and Bayoumi 2019) (Cherian and Kanaga 2022) In currently unpublished work, the Wireless Networks and Embedded Systems (WiNES) Lab's work of one dimensional CNN's with channel and time averaging seems to have created a new state of the art algorithm with 1D CNN The algorithms and Sigmoid Focal Loss. I compare a previous version of that CNN with a sigmoid focal loss function to a large transformer encoder model, a CNN model into an LSTM model and a CNN model that is fed into a transformer. I find that the CNN into the transformer and the large transformer model perform extremely well and are comparable with existing iEEE state of the art algorithms and approaches(SoTA).

Introduction

Epilepsy is a serious affliction effecting an approximate 50 million people in the world. (Organization) Only about 25% of those cases are preventable with medication.(Organization) The remaining cases must either be be treated with vagus nerve stimulation, (MayoClinic) or by simply warning the patient that seizure is likely to be oncoming. Since vagus nerve stimulation has been shown to benefit from only stimulating the neurons before a seizure is likely to occur, and that the invasive brain surgery involved is not always an option, it is often useful in epilepsy cases to be able to trigger vagus nerve stimulation with a machine learning model that predicts an oncoming seizure, or to warn the patient that a seizure is very likely to occur over the next hour. Therefore, a machine learning model is capable of helping all of those parties who cannot be treated with medication. The WiNES research lab has what appears to be a SoTA model for ECG and are working on a version of the same model for iEEG data, but I wanted to see if I could get comparable results with various other algorithms.

Background

Background on CNNs, Bi-directional LSTMs, and Transformers

In this section, we provide a brief background on Convolutional Neural Networks (CNNs), Bi-directional Long Short-Term Memory (Bi-LSTM) networks, and Transformers. These three types of neural network architectures are the foundation for many state-of-the-art machine learning models and are leveraged in some of the models in this paper for seizure classification.

Convolutional Neural Networks (1D)

In this section, we focus on one-dimensional Convolutional Neural Networks (1D CNNs), which are a variant of the traditional CNNs designed to process sequential data. 1D CNNs have been successfully applied in various domains, such as time-series analysis, natural language processing, and signal processing. We will discuss the key components of 1D CNNs, including convolutional layers with varying kernel sizes, stride, and max pooling layers.

1D Convolutional Layer The core component of a 1D CNN is the convolutional layer, which applies a series of learnable filters (also known as kernels) to the input data. Each filter is designed to learn and detect specific features or patterns in the data. In a 1D convolutional layer, the input is a sequence of length L, and the filter has a size of k. The convolution operation is defined as:

$$(X * K)i = \sum_{n=0}^{k-1} X_{i+n} \cdot K_n \tag{1}$$

where X is the input sequence, K is the filter, and i is the position index.

Stride The stride parameter determines the step size by which the filter moves across the input sequence during the convolution operation. A stride of 1 means that the filter moves one step at a time, while a larger stride results in a smaller output size and reduced computational complexity. The output length O after applying a 1D convolutional layer with stride s can be computed as:

$$O = \left\lfloor \frac{L-k}{s} \right\rfloor + 1 \tag{2}$$

Where L is the original input length, k is the kernel size, and s is the stride.

Varying Kernel Sizes Using multiple filters with different kernel sizes allows the model to learn and detect features at different scales. For instance, smaller kernel sizes can capture fine-grained, local patterns, while larger kernel sizes can learn more abstract, global features. In practice, incorporating filters of varying sizes can improve the model's ability to capture diverse patterns and generalize to unseen data.

Max Pooling Layer Max pooling is a downsampling operation commonly used in CNNs to reduce the spatial dimensions of the input while preserving important features. In a 1D max pooling layer, the input sequence is divided into non-overlapping segments of size p, and the maximum value from each segment is retained in the output.

$$\text{MaxPool}(X)i = \max_{n=0}^{p-1} X_{i \cdot p+n} \tag{3}$$

where X is the input sequence, p is the pooling size, and i is the index of the output sequence.

Max pooling layers are typically placed after one or more convolutional layers in a CNN architecture. They help reduce the number of parameters in the model, control overfitting, and increase the receptive field of the subsequent layers, enabling the model to learn more complex patterns in the data.

In summary, 1D CNNs are a powerful tool for processing sequential data by learning local patterns through convolutional layers with varying kernel sizes and reducing spatial dimensions using max pooling layers. The combination of these components enables the model to capture diverse features and generalize well to unseen data.

Bi-directional Long Short-Term Memory (Bi-LSTM) Networks Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to learn long-range dependencies in sequential data. The key component in an LSTM is the memory cell, which consists of an input gate i_t, a forget gate f_t, an output gate o_t, and a cell state c_t. The LSTM cell can be described by the following equations:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{4}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{5}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{6}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{8}$$
$$h_t = o_t \odot \tanh(c_t) \tag{9}$$

where σ represents the sigmoid activation function, $\odot$ denotes element-wise multiplication, and W_{ij} and b_{ij} are the weight matrices and biases, respectively.

Bi-directional LSTMs (Bi-LSTMs) consist of two LSTMs that process the input sequence in both forward and backward directions. The hidden states of the forward LSTM $h_t^{\rightarrow}$ and the backward LSTM $h_t^{\leftarrow}$ are concatenated to form the final output:

$$h_t = [h_t^{\rightarrow}; h_t^{\leftarrow}] \tag{10}$$

Bi-LSTMs are particularly effective in natural language processing tasks, as they can capture information from both past and future context.

Transformers Transformers are a class of neural network architectures introduced by (Vaswani et al. 2017) that rely on self-attention mechanisms to process input data. The key component of the Transformer is the scaled dot-product attention, which can be described as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{11}$$

where Q, K, and V represent the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors. The attention mechanism allows the model to weigh the importance of different input elements relative to each other. In the Transformer architecture, multi-head attention is used to allow the model to focus on different aspects of the input:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{12}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, and W_i^Q, W_i^K, W_i^V, and W^O are learnable weight matrices.

The Transformer consists of a stack of encoder and decoder layers. The encoder layer contains a multi-head self-attention mechanism followed by position-wise feed-forward networks, while the decoder layer contains an additional multi-head attention mechanism that attends to the encoder output. Both encoder and decoder layers employ residual connections and layer normalization.

One of the key innovations in the Transformer architecture is the use of positional encoding to provide information about the relative positions of input elements, as the self-attention mechanism itself is permutation invariant. The positional encoding is added to the input embeddings before they are processed by the Transformer:

$$\text{PE}(pos, 2i) = \sin(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{13}$$
$$\text{PE}(pos, 2i+1) = \cos(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{14}$$

where pos represents the position in the sequence, i is the dimension, and d is the model's input dimension.

Transformers have become the foundation for many state-of-the-art models in natural language processing, such as BERT (Devlin et al. 2019), and T5 (Raffel et al. 2019), due to their ability to model complex patterns and dependencies in the input data. They also have been able to be used for classification problems as well for the same properties such

as is shown by the large library of classification models on HuggingFace.(hug)

Sensitivity, Specificity, ROC, and AUC

In this section, we discuss several important metrics used to evaluate the performance of binary classification models, namely sensitivity, specificity, Receiver Operating Characteristic (ROC) curves, and the Area Under the ROC Curve (AUC).

Sensitivity and Specificity Sensitivity, also known as true positive rate (TPR) or recall, measures the proportion of positive instances correctly identified by the classifier. It is defined as:

$$\text{Sensitivity} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Negatives (FN)}} \tag{15}$$

Specificity, also known as true negative rate (TNR), measures the proportion of negative instances correctly identified by the classifier. It is defined as:

$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN) + False Positives (FP)}} \tag{16}$$

Sensitivity and specificity are important metrics for understanding the performance of a classifier, as they provide insights into the model's ability to correctly identify positive and negative instances separately. However, these metrics do not convey the trade-offs between true positive and false positive rates, which can be crucial for assessing the overall performance of a classifier.

Receiver Operating Characteristic (ROC) Curve The Receiver Operating Characteristic (ROC) curve is a graphical representation that illustrates the trade-offs between the true positive rate (sensitivity) and the false positive rate (1 - specificity) as the discrimination threshold of a classifier is varied. The ROC curve is created by plotting the TPR against the FPR for different threshold values:

$$\text{ROC} = (\text{FPR}(\tau), \text{TPR}(\tau)) \mid \tau \in [0, 1] \tag{17}$$

where τ is the discrimination threshold.

An ideal classifier would have a ROC curve that passes through the top-left corner of the plot, indicating a perfect balance between sensitivity and specificity. In practice, the ROC curve of a classifier provides insights into the trade-offs between true positive and false positive rates, and can be used to identify an optimal threshold for a specific application.

Sigmoid Focal Loss

In this section, we discuss the sigmoid focal loss, a loss function designed to handle class imbalance in object detection and classification tasks. The focal loss was introduced by (Lin et al. 2017) as an alternative to the standard cross-entropy loss, with the aim of focusing model training on hard examples and down-weighting the easy, well-classified examples.

Standard Cross-Entropy Loss To provide context, let us first review the standard cross-entropy loss for binary classification. Given the ground truth label $y \in 0, 1$ and the predicted probability $\hat{p}$, the cross-entropy loss is defined as:

$$\text{CE}(y, \hat{p}) = -y \log(\hat{p}) - (1 - y) \log(1 - \hat{p}) \tag{18}$$

Although this loss function is widely used, it has some limitations when dealing with class imbalance. In such cases, the model may become biased towards the majority class, resulting in poor performance on the minority class.

Area Under the ROC Curve (AUC) The Area Under the ROC Curve (AUC) is a scalar metric that quantifies the overall performance of a binary classifier. AUC is calculated as the integral of the ROC curve:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}), d\text{FPR} \tag{19}$$

The AUC value ranges from 0 to 1, with an AUC of 1 indicating perfect classification performance, while an AUC of 0.5 suggests that the classifier is no better than random chance. The AUC has several desirable properties, including being invariant to class distribution and classification thresholds. Additionally, it can be interpreted as the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance by the classifier. In many cases, the AUC is a useful single-number summary of classifier performance and is commonly used to compare different models.

Sigmoid Focal Loss The sigmoid focal loss is designed to address class imbalance with a parameter α while also downweighting the loss from easy samples and more heavily weighting loss from harder samples with a parameter γ

$$\text{FL}(y, \hat{p}) = -\alpha y(1-\hat{p})^\gamma \log(\hat{p}) - (1-\alpha)(1-y)\hat{p}^\gamma \log(1-\hat{p}) \tag{20}$$

where $\alpha \in [0, 1]$ is a balancing factor that controls the trade-off between the importance of positive and negative examples, and $\gamma \geq 0$ is the focusing parameter that controls the down-weighting of easy examples. When $\gamma = 0$, the focal loss is equivalent to the standard cross-entropy loss.

The focal loss can be interpreted as a dynamically scaled cross-entropy loss, where the scaling factor $(1-\hat{p})^\gamma$ for positive examples and $\hat{p}^\gamma$ for negative examples modulates the contribution of each example to the total loss. As a result, the model becomes more robust to class imbalance and better able to learn from hard examples.

Related Work

In previous work at the WiNES Research Lab(Uvaydov et al. 2022) they found they could get fairly good results by using a stacked 1D CNN with cross entropy loss. However, the accuracy and false positive rates were still much lower than other works, specifically those using a CNN to downsample the iEEG sample from 1024 Hz and then processing that through an LSTM. Several works, show that this appears to

be the best performing model architecture(Daoud and Bayoumi 2019)(Cherian and Kanaga 2022). This appears to be because the model learns the time series relationship in the data that LSTMs are best at, while still being parameter efficient due to the CNN downsampling. However, in currently unpublished work, the WiNES lab(Johari) has gotten comparable or better results by using a new loss function Sigmoid Focal Loss (Lin et al. 2017) and a new CNN. Presumably, it would therefore be possible to use the advantage previous works have found when using bi-directional LSTMs or other time series focused algorithms to process the data after it has been downsampled with a CNN.

As discussed in background, another algorithm that is well known for doing time series classification and relationship discovery is the transformer architecture. To our knowledge, there is no related work attempting to use transformers. Therefore, it seems probable that by using a sigmoid focal loss function with a CNN into an LSTM or transformer it would likely be possible to get a new SOTA for seizure classification.

Methodology

Dataset and Preprocessing

The dataset used is the existing dataset the WiNES(Johari) lab has curated from the epilepsiae(Duempelmann) dataset. In it, the total data of hundreds of hours from 27 different patients is split into equal sample size portions of pre-siez, data 1 hour before a seizure and non-seiz data that isn't a seizure and is either after a seizure or at least one hour before a seizure. That data is then broken up into 4 second 1024 Hz samples. Those samples are then downsampled to 256 Hz with an iir low pass filter for all channels. Where each channel is a separate electrode that was collecting eeg data. That entire dataset is then saved to an H5 file. The data therefore is of shape, channel size x 1024, representing a single 4 second multi-channel pre-seiz or non-seiz sample.

In this work, I built a dataloader that processes each of those h5 files. The dataloader breaks up the associated channels and then makes each of the samples randomly queryable such that channels are not necessarily next to each other.

Algorithms

To compare multiple algorithms, and to make sure we could reproduce existing previous works I implemented 4 algorithms and compared it with a fifth, the currently unpublished model the WiNES lab has internally that performs extremely well on ECG and iEEG data.

The first network architecture can be found in Figure 1.

This is the OneDCNN Architecture. It is based on an old architecture the WiNES lab had been using. I built it out with the same architecture but using sigmoid focal loss. The intention being to see how much the lab's new CNN architecture vs the sigmoid focal loss function contributed to their latest model's success.

The second architecture is a naive transformer architecture. The thought was that large sequence classification models with transformers often are just large stacked encoder layers. Therefore, it would be interesting to see what happened if you attempted to classify seizures with 6 x 1024 transformers with 4 attention heads each with 2048 feed forward layers. This model does in fact perform well, but it is still not SoTA and the total model parameter count is 50,678,210 parameters. Therefore, it is completely incompatible with most of the FPGA's the lab is looking at for loading the model onto, as it seems the upper limit on parameters is roughly on the order of one million parameters assuming the model can be pruned very aggressively.

The third architecture is based off of the CNN architecture the WiNES lab is using internally and hasn't published yet. It can be found in Figure 2.

I did not train and run this algorithm directly, instead I replaced the 64 x 64 linear layer with two transformer layers or two LSTM layers. The transformer layers had 64 inputs, 64 feed forward values and 8 attention heads with dropout of .1. The LSTM layers had an input layer size of 64 and a hidden size of 64 as well. They were not bi-directional. I thought that they were, but did not realize until after I had collected results, that although much of the hyper parameter tuning I had done was with bi-directional layers, I had forgotten to set the bi-directional flag on the final training run.

Hyper Parameter Tuning

Before settling on the above architectures, I attempted to find a variety of other architectures that may be better, and tuned several hyper parameters. To do this, I chose the patient with the least samples 543,576 samples, and then attempted to find the highest accuracy possible in as few epochs as possible 1-3. Each epoch took 15 minutes to run so this was a highly time consuming process. Ultimately, I found that my CNN networks, could not beat Ali's, the SoTA architecture in Figure 2, so I adopted the top layers of his architecture and then replaced the bottom layers with LSTMs transformers and that did appear to be the best performing architecture I tried. Use the same technique, I optimized batch size, and size of various parameters in the transformer.

On top of this, the different architectures were very sensitive to different learning rates. So I used a learning rate finder (Smith 2017) to find the optimal learning rate. I did this by checking the produced chart of gradients across various learning rates and then chose the learning rate that produced the steepest gradient where it looked like annealing the learning rate later would still produce further learning. The final learning rates I settled on were 6.51519e-06 for the CNN into LSTM and CNN into transformer models. I chose .0002 for the 1D CNN model. And I chose .0001 for the naive transformer model.

Post Processing and Result Collection

To collect results I built a result collection pipeline that ran at the end of training. Once the model had converged, I saved the model checkpoint and then I ran all of the pre-seiz data and non-seiz data through the trained model.

Although normally, it would be preferable to plot the loss and then to pick the model checkpoint from the epoch at the elbow in the loss plot, I found the optimal number of epochs when doing hyper parameter tuning on the patient

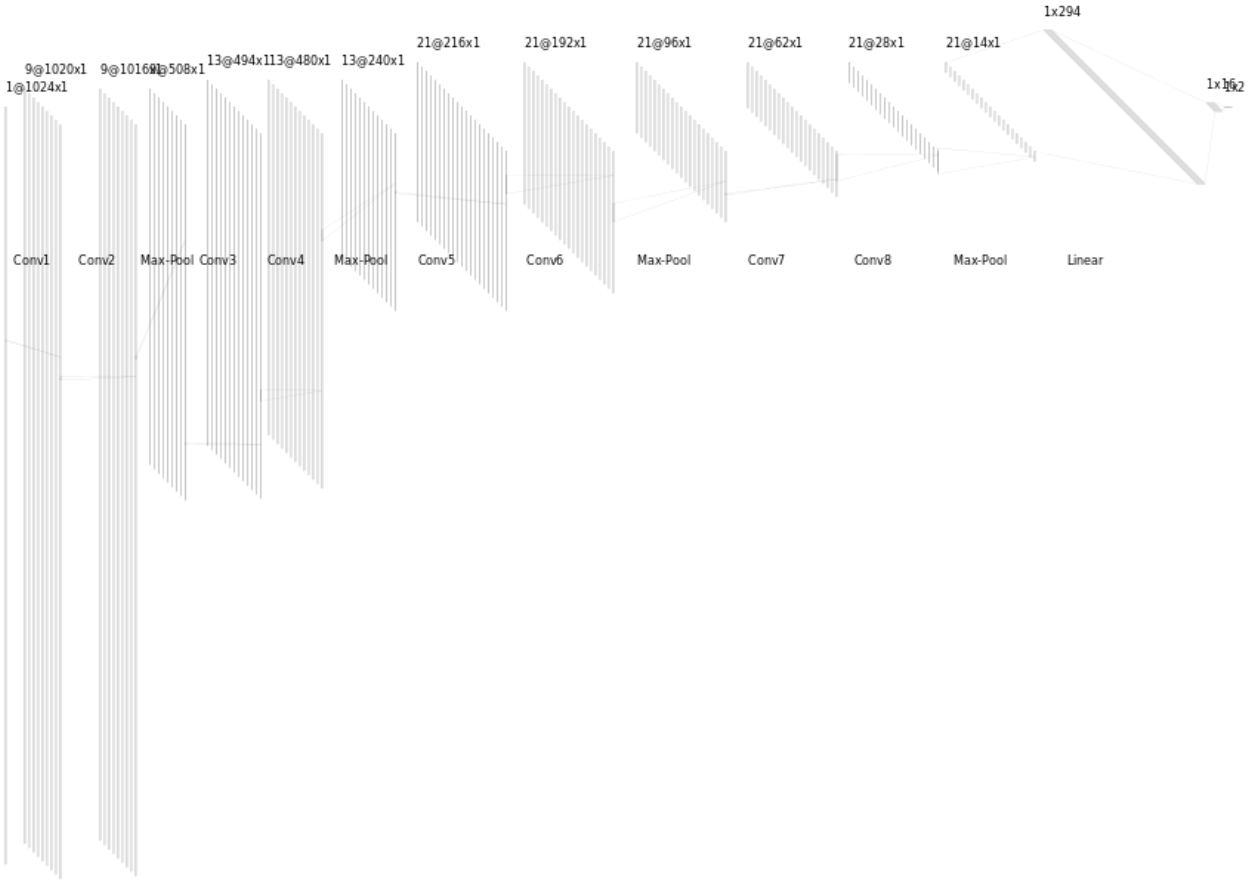


Figure 1: One D CNN (Mine) Based on the CNN architecture the WiNES lab used to use.

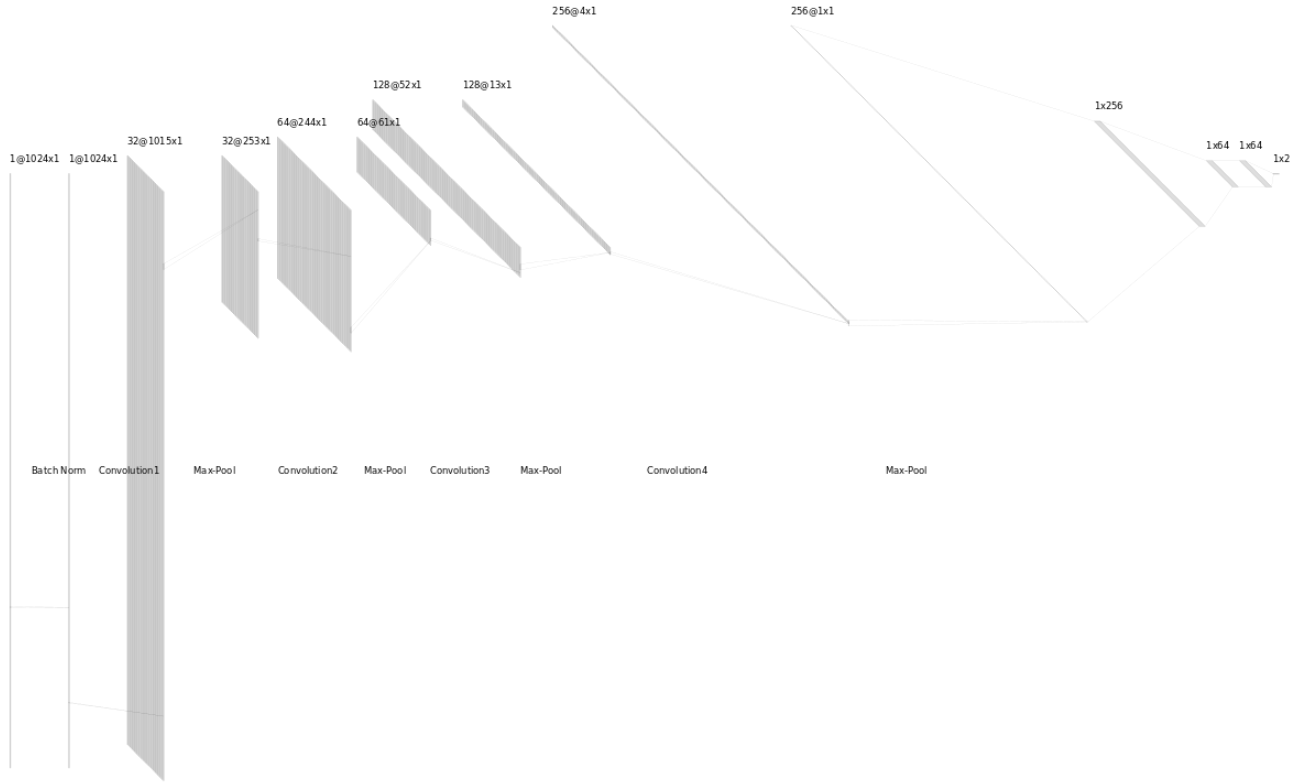


Figure 2: The model architecture the WiNES lab is currently testing and the basis for the CNN into LSTM and CNN into transformer architectures.

with the smallest number of samples. That combined with the fact that I annealed the learning rate if the validation set no longer improved by a factor of 10 meant the last epoch for all observed cases was the best performing model.

I saved the data output to a JSON file as a 0 or 1 depending on if the classification accuracy was greater than or less than .5. This ended up being a mistake. It meant that I could not do AUC on the raw data because ROC curve expects the probabilities of each of the predictions, and it likely removed non-trivial data for when doing later post processing on the data.(Channel voting/Time voting) This is an improvement that will be discussed in Future Work.

On top of the mistake of saving the values as 0s and 1s, I also made a mistake of saving these zero and ones into lists labeled non and pre for the representative classes. But I had a mistake in separating the total labels, I did not account for channel sizes properly. Luckily, the total number of samples in each class was equal plus or minus a single sample and I had saved the samples in order. Therefore, I re-combined the saved data and then split the data at the mid-point accounting for channel sizes and then plotted all of my results based on this newly split saved set of results. This does mean that there is likely one set of points at the margin between the classes that is assigned incorrectly. However, considering there are hundreds of thousands if not millions of samples for each patient, I decided this was acceptable as long as it was documented.

The next step was to do channel voting and time voting on these results(Uvaydov et al. 2022). The premise here is that although you do inference on each channel individually, it is possible to get a better result by averaging the result for any single sample across the results of all channels to get the actual probability of a seizure at the time. Then you can average across n channel voted samples in the time domain to get a time voted likelihood to predict with even higher accuracy the likelihood of a seizure in the next hour.

Experimental Results

The first set of results that is the most important is raw accuracy, sensitivity and specificity. If the model cannot predict with high accuracy whether or not a single sample is indicative of a seizure while minimizing false positives and false negatives, then the user will be notified excessively they will have seizure when they in fact will not or they will not be notified before they have seizure.

You can see in Figure 1, that some patients for the 1D CNN were classified no better than random and the same occurred for the CNN to LSTM. I do not know why this is, and when I examined the training, that specific model for that specific patient never learned anything, it always returned 50 percent accuracy. I do not know why. Otherwise, the accuracy's for all patients and models and patients was between 60 and 100 percent.

We then channel vote across the 64 to 256 channels, where this variable number of channels is based on the number of electrodes used at time of data recording. We can see in Figure 4 that this greatly improves results. The large transformer and CNN into transformer architectures appear to be the best performing models.

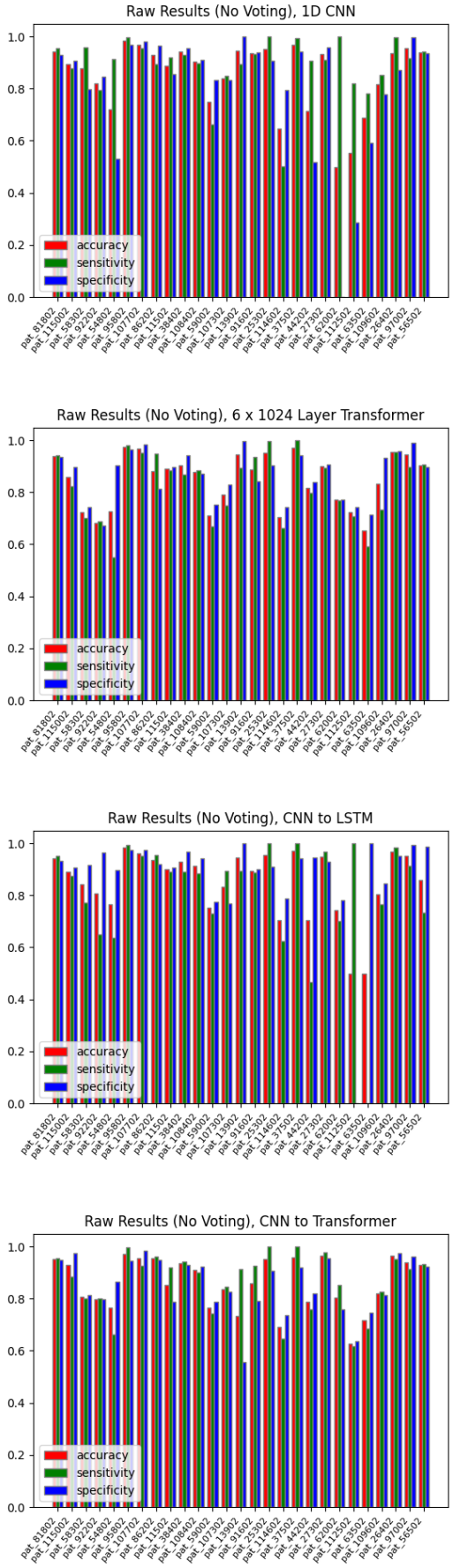
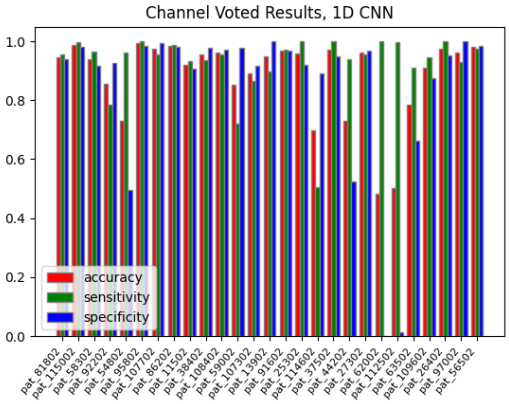
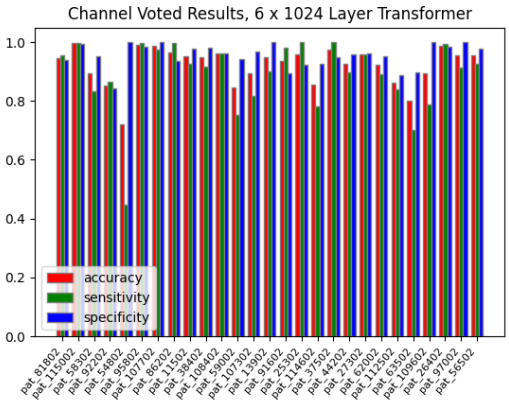
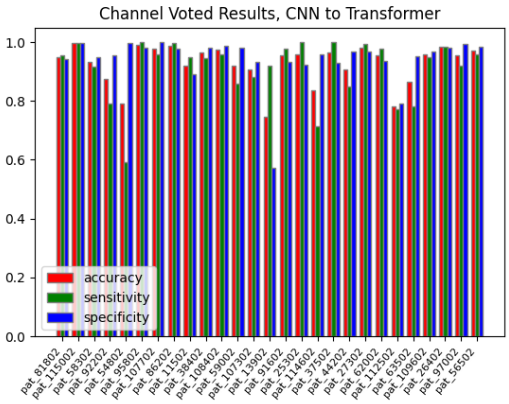


Figure 3: Accuracy, Sensitivity, and Specificity Before Channel or Time Voting







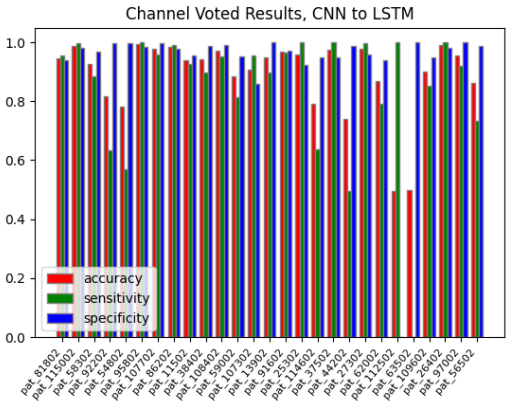
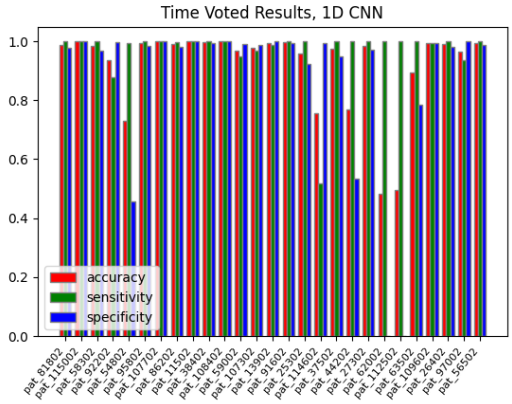
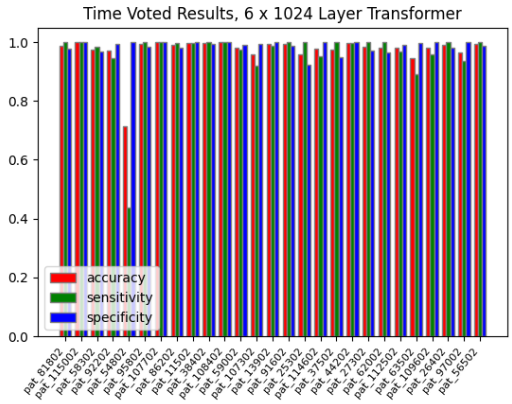
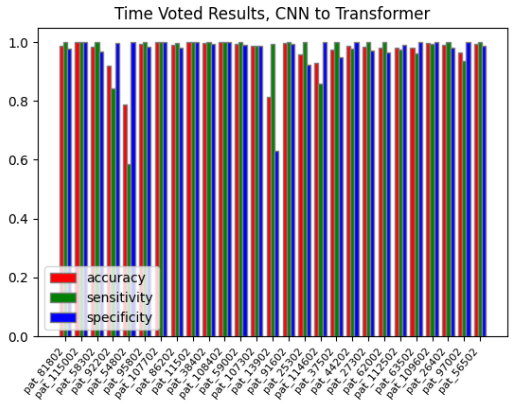


Figure 4: Accuracy, Sensitivity, and Specificity with Channel Voting







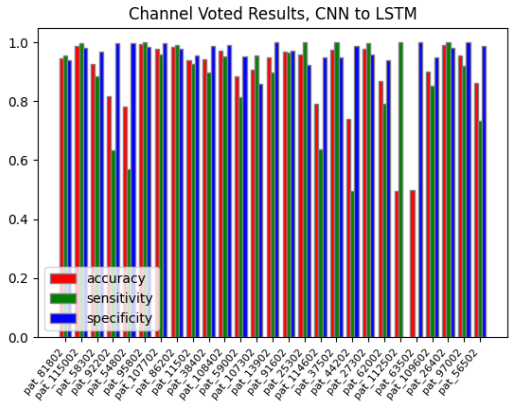


Figure 5: Accuracy, Sensitivity, and Specificity With Channel And Time Voting

Then shown in Figure 5 we do time voting. With the large transformer and CNN into Transformer this gives us accuracy, sensitivity and specificity of greater than 97 percent for nearly all patients for both the large transformer and the CNN into Transformer model.

Then in figure 6 we show channel voted ROC curves. We see that channel voting provides a fairly good classifier, but is not good enough for the failure intolerant environment of seizure detection.

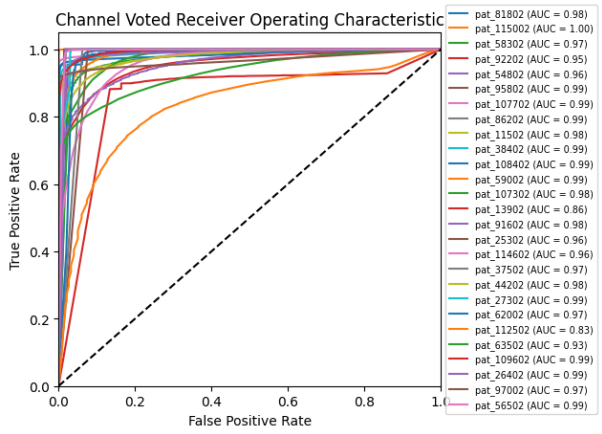


Figure 6: Channel Voted ROC Curves CNN into Transformer

We can then look at the impact of the time voting on the ROC curves as well in Figure 7 and Figure 8. As described in the background section, the ROC curve is nearly in the top left hand corner of the chart for the time voted results for nearly all patients indicating near perfect classification for both the large naive transformer and the CNN into transformer model.

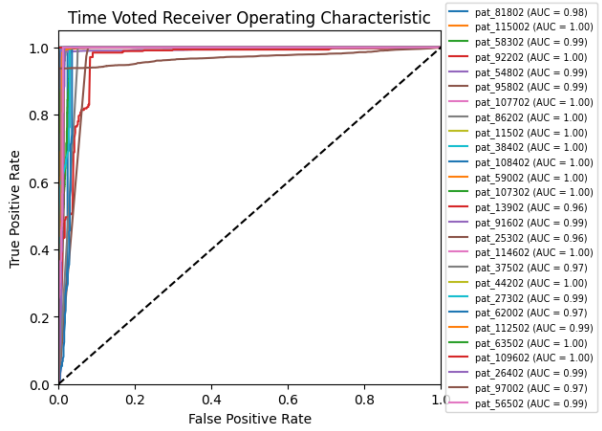


Figure 7: TimeVoted ROC Curves for CNN into Transformer

We can further verify that, by looking at the AUC of all patients in figures 8 and 9. With average AUCs for the time voted patients of .99 this suggests we have a near perfect

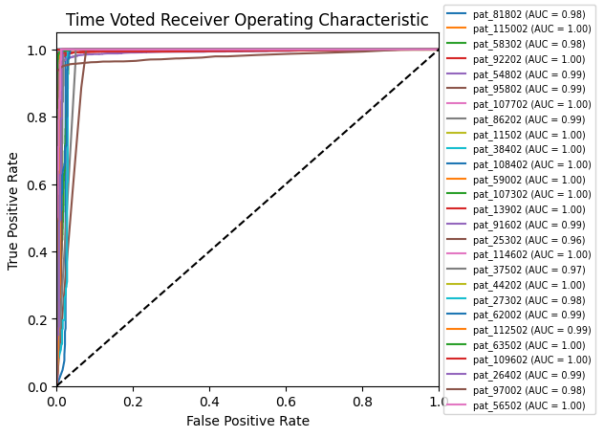


Figure 8: Time Voted ROC Curves for Naive Transformer

classifier for both the naive 6 x 1024 transformer and the CNN into Transformer models.

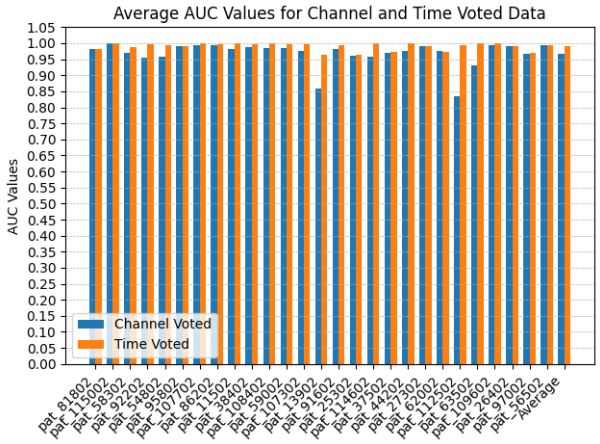


Figure 9: AUC for Channel and Time voted CNN into Transformer Model. AVG Channel Voted AUC: .97 AVG Time Voted AUC .99

One other metric that is minimized is false positives. Ideally, the patient would never get a false alarm, but would still always be notified of a seizure. In figure 11, we can see a break down of hourly false positives for channel voted data. We can see that there are a few patients who have a false positive on every negative sample in the 1D CNN and for one patient in the CNN into LSTM. However, the naive transformer and CNN into transformer both perform much better than that.

In figure 12 we can see that the naive transformer has 7.5 false positives per hour in the worst case. The CNN into Transformer has one outlier, but the other patients have a similar number of false positives/hour. The number of false positives/hour for the CNN into Transformer after time voting is 2.71. With some further smoothing on the device monitoring the neural network this may be productizable. The corresponding median for all patients is 1.02.

Table 1: Comparison of All Models Across Metrics.

Model	Time Voted FP / Hour	AVG Time Voted AUC	AVG Sensitivity Time Voted	Avg Specificity Time Voted	AVG Accuracy Time Voted	Parameter Count
1D CNN (Mine)	24.89	97	**96.05**	86.16	91.05	**58,342**
6 x 1024 Transformer	3.06	**99**	94.26	98.29	**96.24**	50,678,210
CNN to Transformer	5.18	**99**	95.14	97.12	96.10	523,108
CNN to LSTM	9.24	95	88.53	94.86	91.62	514,020
1D CNN (WiNES Lab)	**.37**	**99**	91	**99**	95	451,620

Notes: Columns with the Time Voted designation are time voted with 5 samples and channel voted. FP/Hour refers to false positives per hour

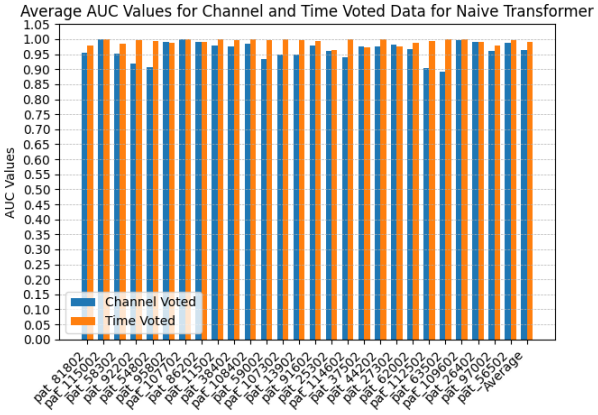


Figure 10: AUC for Channel and Time Voted 6 x 1024 Transformer Model AVG Channel Voted AUC .96 AVG Time Voted AUC: .99

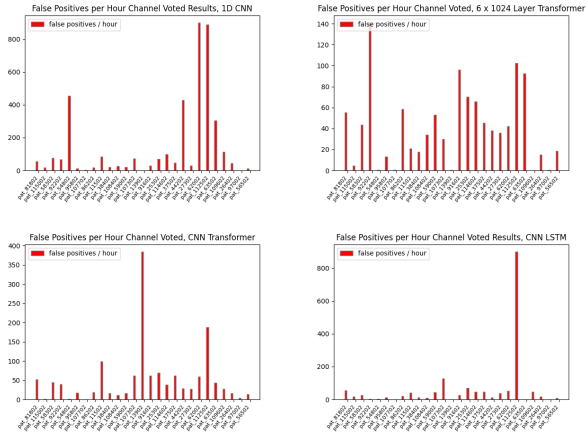


Figure 11: Channel Voted False Positive Rates / Hour

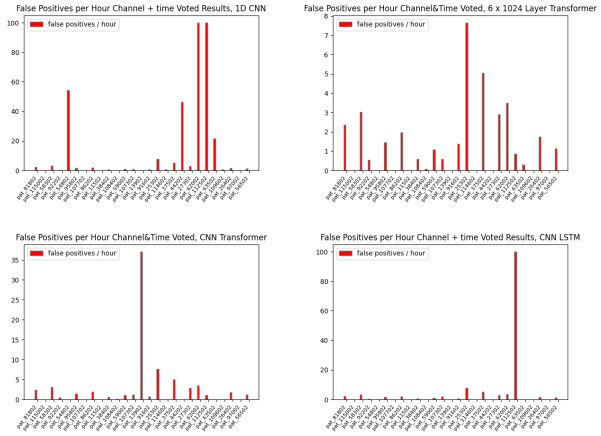


Figure 12: Channel And Time Voted False Positive Rates / Hour

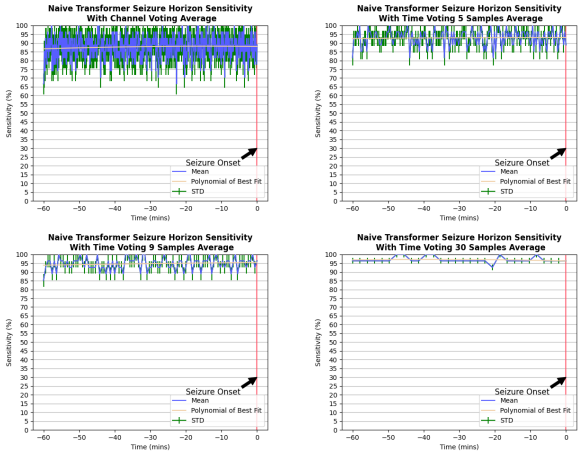


Figure 13: Sensitivity an Hour before Seizure Onset for the Naive Transformer

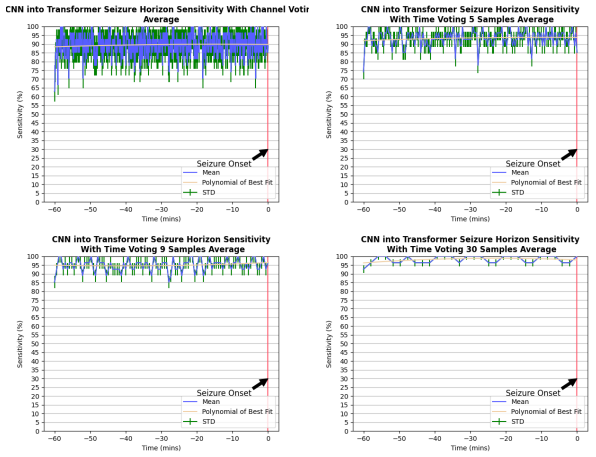


Figure 14: Sensitivity an Hour before Seizure Onset for the CNN into Transformer Model

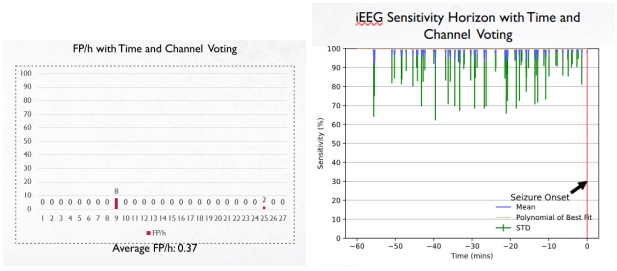


Figure 15: False positive rates for all patients(Left) and Sensitivity Horizon for one hour of pre-seiz data before seizure onset averaged across all patients. (Right) This is not necessarily the same hour of data in the other sensitivity plots.

The final set of result plots is sensitivity an hour before seizure onset. In it you can consistently watch sensitivity rise throughout the hour as the model becomes more confident. In Figure 13, we see the Naive Transformer model's sensitivity rise from 87 percent to 89 percent throughout the hour. We also can see right at the start of the period the model thinks the likelihood of it being pre-seizure is only 60 percent which makes sense as that sample would be seconds away from being classified as non-seizure data.

As time sampling is applied we can see watch the best fit line rise form the initial 87 percent to 96 percent.

We can then directly compare these results to the CNN into Transformer model in Figure 14. Where the initial best fit line, starts at 88 percent and slowly rises to 97 percent as time voting is applied. We also can see that the 30 time sampled approach has a higher 100 percent accuracy rating for the CNN into Transformer model which may mean it is more practical than the other model which struggles to break 96 percent sensitivity.

We can now directly compare all of our different metrics to see which model is the best. We do this in Table 1, where we compare False Positives / Hour, time voted AUC, sensitivity, specificity, accuracy and parameter count.

You can see that each model has aspects of it that perform the best. The 1D CNN has the highest sensitivity and lowest parameter count by nearly an order of magnitude. The large transformer model has the highest overall accuracy.

The CNN to transformer model has the second highest accuracy and sensitivity while still being relatively parameter efficient and usable unlike the large transformer model. The CNN to LSTM is not the best at anything, but if it is switch to bi-directional LSTMs it may be. The existing 1D CNN model that Wines lab has has the best false positive rate per hour and best corresponding specificity, while still having the second least number of parameters.

We can also compare the false positive plots from the WiNES lab model to our models via the false positive rates and sensitivity seizure horizon plots in Figure 15. Presumably, the average sensitivity horizon value would be closer to the average sensitivity of 91 percent for the WiNES model for other pre-seizure samplings.

Ultimately, the extremely low false positive rates from the WiNES lab model, likely makes it the most practical model, as it will still always warn the user of incoming seizures while extremely rarely warning them incorrectly.

Future Work

There are several opportunities for future work. The first is result collection. In this work, all results were collected as binary saved in a json file. It would be far better to collect it as the original probability so that channel voting and time voting have more accurate numbers to average. It also would be good to save that data correctly and/or do inference with a model check point rather than adjusting the data that was saved in incorrect buckets resulting in one sample being marked incorrectly for every patient as was discussed in Methodology-Post Processing and Result Collection.

Hyper parameter tuning could definitely be further improved upon. Hyper parameters were only fine tuned on a single patient which may result in some of the poor performance I found on some specific patients.

The dataset used is only approximately 1/16 of the total epilepsiae dataset. This is to create equal sample sets to make binary classification easier. However, sigmoid focal loss can explicitly handle varying class sample sizes as discussed in the background section. Because of this we are working on a PostgreSQL based system of loading individual pre and non seiz data as binary into a database. That way all of the data can be queried randomly or in order at scale. This is necessary to avoid out of memory errors with the full 8 terrabyte dataset.

Additionally, in this work, bi-direcitonal LSTMs were not used, and uni-directional LSTMs were used instead. Considering the past SoTA used bi-directional LSTMs it may be worth it to train a model using bi-directional LSTMs.

Finally, it potentially would be possible to skip some of the voting mechanics by pushing multiple channels or multiple time samples through a higher dimensional neural network. Potentially, the model would be able to find higher dimensional relationships across samples or channels which would greatly boost accuracy.

Conclusion

Ultimately, we found that several of these models had different best performing characteristics, but that the model the WiNES lab currently has internally, is likely the best due to the fact that it successfully minimizes false positives / hour better than any other model, even if it's average accuracy and sensitivity do not always outperform other models.

References

Cherian, R., and Kanaga, E. G. 2022. Theoretical and methodological analysis of eeg based seizure detection and prediction: An exhaustive review. *Journal of Neuroscience Methods* 369:109483.

Daoud, H., and Bayoumi, M. A. 2019. Efficient epileptic seizure prediction based on deep learning. *IEEE Transactions on Biomedical Circuits and Systems* 13(5):804–813.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Duempelmann, M. The european epilepsy database.

Models.

Johari, P. Wireless networks and embedded systems labratory.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection.

MayoClinic. Vagus nerve stimulation.

Organization, W. H. Epilepsy.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR* abs/1910.10683.

Smith, L. N. 2017. Cyclical learning rates for training neural networks.

Uvaydov, D.; Guida, R.; Johari, P.; Restuccia, F.; and Melodia, T. 2022. Aieeg: Personalized seizure prediction through partially-reconfigurable deep neural networks. In *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 77–88.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need.

Appendix

See additional results plots here.

Figure 16 shows a relationship of sample size and accuracy. This shows a slight negative relationship between the number of samples in the dataset for a specific patient and it's output accuracy. The R^2 value on this relationship is higher for the more accurate models. This suggests that despite controlling the number of epochs with hyper parameter tuning it is possible these models are slightly overfit, and may not generalize well in the future.

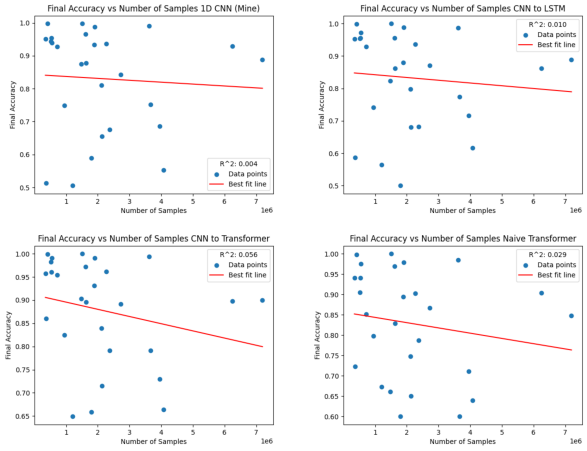


Figure 16: Plot of number of samples per patient and the final model accuracy for that patient.

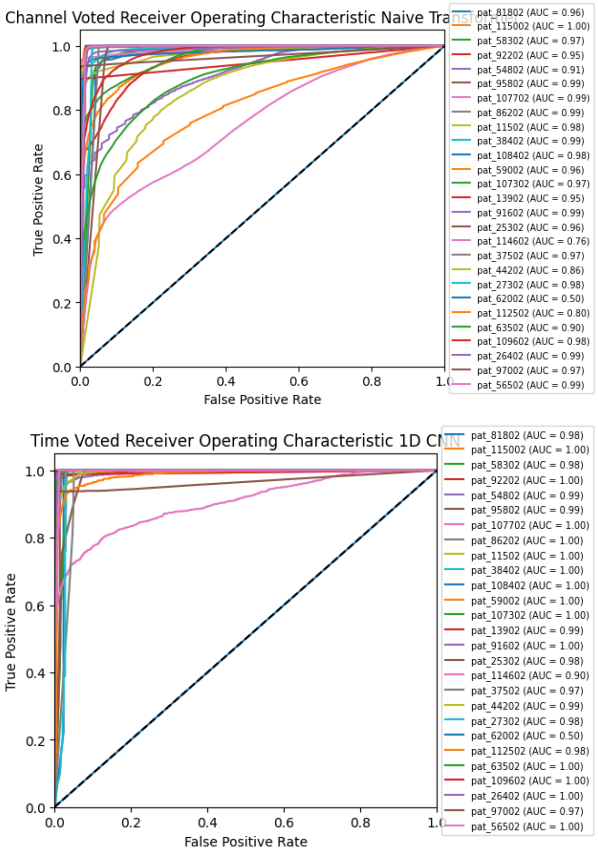


Figure 17: ROC for Channel and Time voted 1D CNN(Mine)

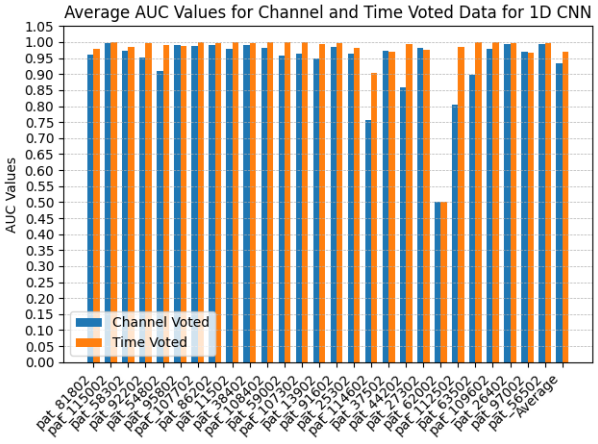


Figure 18: AUC for Channel and Time voted 1D CNN (Mine)

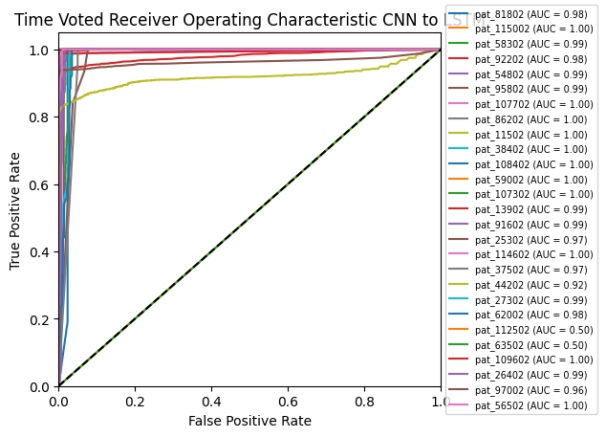


Figure 19: ROC for Channel and Time voted CNN to LSTM

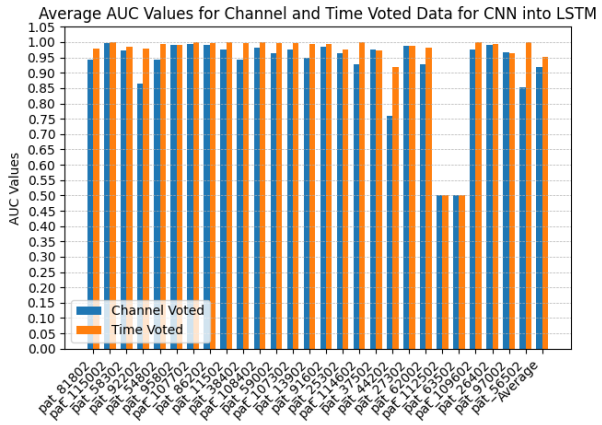


Figure 20: AUC for Channel and Time voted CNN into LSTM (Mine)

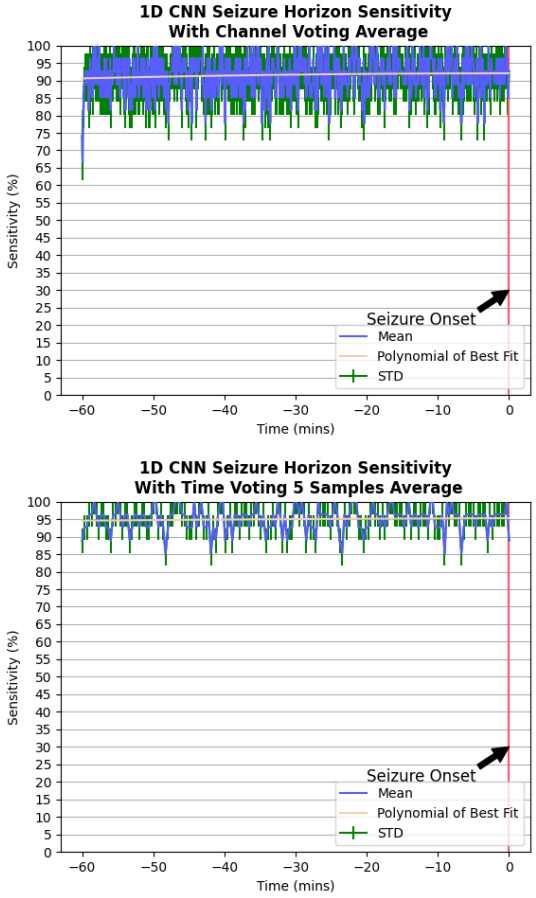


Figure 21: Sensitivity Plot One Hour Pre-Seiz for the 1D CNN (Mine)

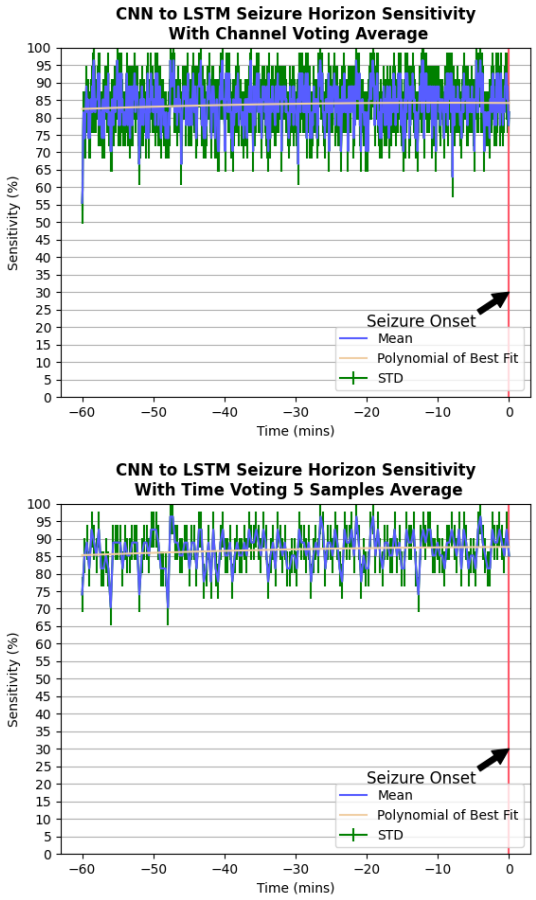


Figure 22: Sensitivity Plot One Hour Pre-Seiz for the CNN into LSTM model