**Optimized EEG Dataset Study with Hybrid Deep Learning Stategies: CNN Architectures Combined with LSTM and Transformer Architectures**

Elizabeth Babbitt - [ebabbitt@ucla.edu](mailto:ebabbitt@ucla.edu)

University of California - Los Angeles

Julia Wang - juliawang03@ucla.edu

University of California - Los Angeles

Daniel Yang - danielyang32@ucla.edu

University of California - Los Angeles

# **1. Abstract**

*This paper shows how our project created several neural networks to decode EEG data from the emg2qwerty dataset. We utilized a Convolution Neural Network (CNN), Long Short Term Memory Network (LSTM), Recurrent Neural Network (RNN), and Transformer. Then we sought to further execute hybrid architectures of CNN+LSTM and CNN+Transformer, analyzing varying layer complexities. All algorithms clearly trained and validated better with increased epoch. Our top performer for the hybrid architectures was the CNN+LSTM model (21.74 test/CER) as the CNN provides a great encoder with a 128 hidden layer size. Our results demonstrate that the vanilla LSTM model (19.213 test/CER) marginally outperforms the CNN+LSTM hybrid model. Clearly by comparison, our data trending demonstrates how the other basic architectures (Transformer, RNN) did not nearly perform as well as the hybrid models (CNN+Transformers), which resulted in an optimal test/CER of 26.778 at 2 layers. Note that RNN models were tested based on the assumption that previous outputs would affect future outputs. From this contrast in performance, it is clear that the lack of performance for the RNN model was likely not due to a flaw in our assumptions, seeing as how maintaining memory of previous input/output data is also a key feature of LSTMs. Rather, the lack of performance in the RNN is more likely due to flaws in the RNN architecture that have since been solved by successor architectures like the LSTM. Many models containing transformer layers were also tested due to its widespread adoption in many multimodal generative AI applications. However, the performance of the pure transformer encoder model was considerably lacking.*

# 2. Introduction

Transformative analytical tools used in EEG studies have applied machine learning to reveal imperative information pertaining to neural classification. EEG analysis has profound application in neuroscience, neural engineering, as well as commercial applications. Primarily characterized by the number of convolutional layers and the type of end classifier, Convolutional neural networks (CNN) serve as an ideal starting point for the prevalent architecture designs of EEG classification tasks [1]. So in this project, we succeeded in performing end-to-end deep learning neural network methods to anticipate keystrokes given sEMG signal sequences.

We aim to address the challenge of reducing test Character Error Rate (CER) in sEMG-based typing for a single user by exploring and modifying various neural network architectures. The projected output character sequence can potentially consist of substituted, inserted, or deleted characters. Note that the labeled sequences consist of characters typed at arbitrary timestamp. So since the Connectionist Temporal Classification (CTC) loss aims to address classification with unsegmented sequential data, we further measured how well the predicted sequences match the targets [5].



Figure 1 - CER (Character Error Rate) Fractional Equivalent: “S” = number of character substitutions, “D” = number of character deletions, “I” = number of character insertions, “N” = total number of characters

Building upon established ASR-inspired models, we systematically evaluated several architecture variants—including standalone RNN, LSTM, Transformer, and hybrid models that combine CNN layers with LSTM or Transformer modules—to capture the nuanced temporal dynamics of sEMG signals. RNNs (with a specified number of recurrent layers) were one of the first autoregressive architectures, with each RNN feeding previous outputs as inputs for the next iteration. The task of converting EEG signals to text output is essentially an NLP task except previous signals are used to predict the next output instead of previous tokens [6]. Furthermore, Long Short-Term Memory (LSTM) models are excellent for recurrent models with inputs as sequences. LSTM networks are a specific type of RNN that include additional gating mechanisms—input, output, and forget gates—that help the network selectively retain or discard information over time, making them more effective for many sequence tasks than a basic RNN with simple hidden state updates. Applying LSTM models overcomes the challenge of long-term dependency in RNN, because of the reality that long character sequences can be hard to learn from basic RNN, which is trained by back-propagation through time (leading to vanishing/exploding gradient) [2]. Moreover, since Transformers are an extremely powerful and versatile architecture used in almost every modern machine learning application,we aimed to gain insight into attempts using the architecture to solve this particular sequence modeling problem [8].

By rigorously comparing these architectures, we strived to demonstrate that personalized model tuning can significantly improve decoding accuracy. The hybrid architectures, which leverage the feature extraction power of CNNs alongside the sequential modeling capabilities of LSTM and Transformer layers, showed promise in achieving lower test CER, ultimately enhancing the usability of sEMG-based typing interfaces in personalized applications. Thus, we strived to leverage the CNN models containing transformer and LSTM in order to observe the effect of adjusting the number of layers and hyperparameters in order to reach our highest test accuracy across all the model data [3]. Initially, LSTM models can standalone, but extremely long sequences can deteriorate the algorithm.

Fortunately, previous studies have decoded EEG signals with the hybrid CNN+LSTM model (shown in the figure below), supporting the idea of eating a CNN and LSTM hybrid model can allow the CNN to extract spatial features while the LSTM can maintain the sequential patterns and perform great with dynamic data. The CNN “effectively extract[ed] the features” while the LSTM may require further training across hidden layer optimum (may need fine tuning) can lead to better generalization [4]. By optimizing these architecture mechanisms towards character-level modeling tasks, our goal was to identify a configuration that not only minimizes CER but also adapts effectively to the idiosyncratic patterns of an individual’s sEMG profile.

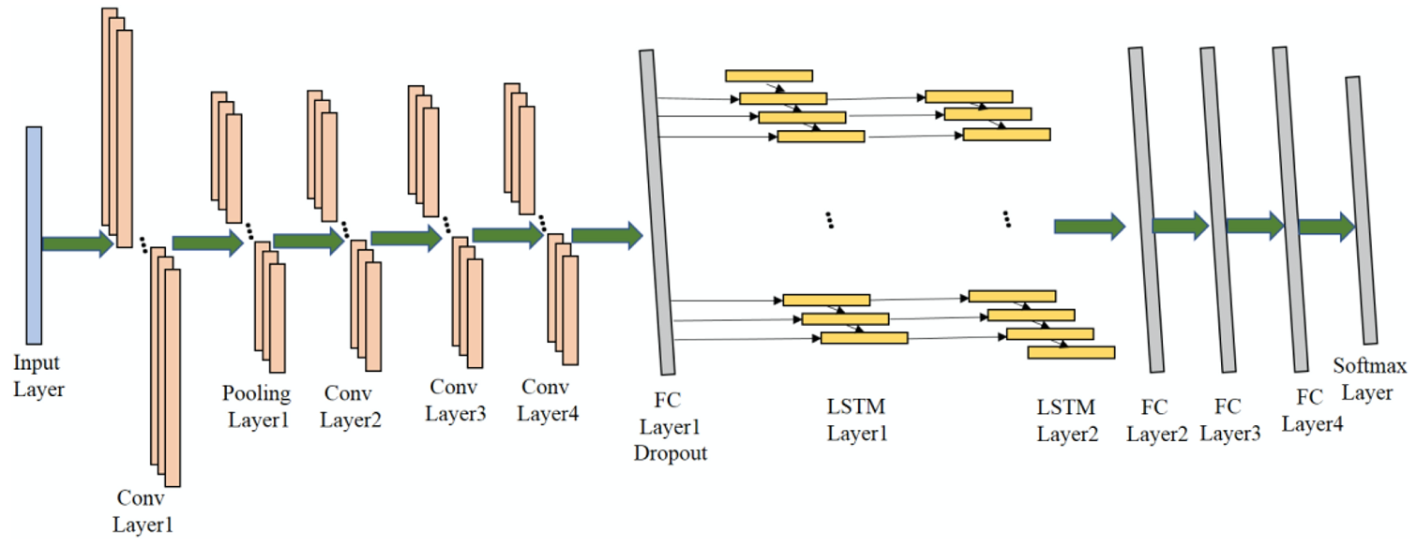


Figure 2: EEG Study with Hybrid CNN + LSTM Model [4]

# **3. Methods**

## 3.0 Data Preprocessing + Augmentation

We did not deviate from the techniques used in the baseline experiment. The baseline preprocessing pipeline converts raw sEMG signals into log-scaled spectral features computed over a custom frequency scale with tailored cutoffs, and applies per-channel 2D batch normalization to normalize the spectrogram statistics. In addition to this, the model employs several data augmentation techniques: SpecAugment is used to apply time- and frequency-masked distortions to the spectral features, electrode rotation augmentation shifts the channels by −1, 0, or +1 positions, and temporal alignment jitter randomly offsets the alignment between the left and right sEMG time series by up to 60 ms. [ADD FOOTNOTE/ CITE PAPER]

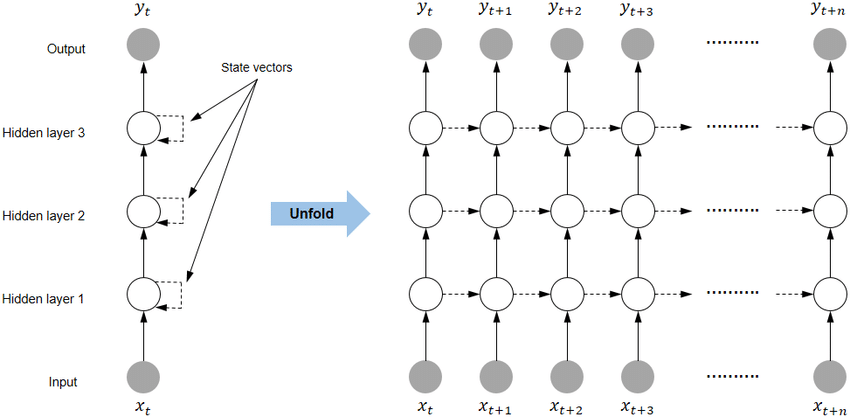
## 3.1 Model Architectures

### 3.1.0 Baseline TDS-CNN

For our baseline TDS-CNN experiments, we replicated the original setup in every respect—using the same model architecture, hyperparameters, data preprocessing, and augmentation techniques—as described in the paper, with one exception: we reduced the maximum number of training epochs from 150 to 50 due to computational resource limitations. As such, for consistency across all experiments, we set the maximum training epochs to 50 for all architecture comparisons. This standardization ensures that any differences in performance are attributable to the model architectures themselves rather than variations in training duration, and reflects our adjustment due to computational resource constraints.

### 3.1.1. Basic RNN

For the RNN runs, an RNN of 128 hidden parameters is connected to a TDS Fully Connected Block, which consists of a linear layer, Relu, then another linear layer. After the TDS Fully Connected Block is another linear layer. During training, the Adam optimizer was used with an initial learning rate of 0.001. The number of layers in the RNN was initially 4. An illustration of a deep RNN is shown in the figure below. Increased layer count was not tested due to limits on RAM. A single layer RNN was also tested to confirm whether the stall in learning at CER 100.0 was due to vanishing gradients, but this did not appear to be the case.



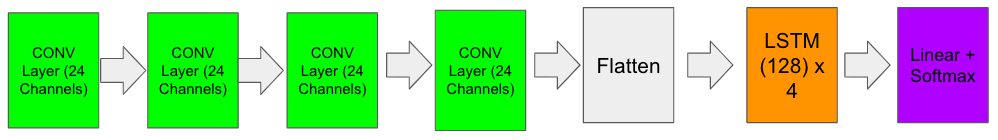
[caption] Figure X. Structure of a deep 3 layer RNN. [7]

### 3.1.2. Basic LSTM:

The basic LSTM model, implemented in the TDSLSTMEncoder, first passes the input through a bidirectional LSTM with 4 layers, each having a hidden size of 128 per direction—resulting in an output feature dimension of 256. This LSTM output is then processed by a fully connected residual block (TDSFullyConnectedBlock) that maintains the 256-dimensional space, before a final linear layer maps the features back to the original dimension. This architecture ensures that temporal dependencies are captured in both directions while preserving the input dimensionality for subsequent processing. As in the baseline CNN, we utilized a batch size of 32, the Adam optimizer with 0.01 learning rate, and a CTCGreedyDecoder for transcription. Training is carried out for 50 epochs on a single GPU with 4 worker processes.

### 3.1.3. CNN + LSTM**:**.

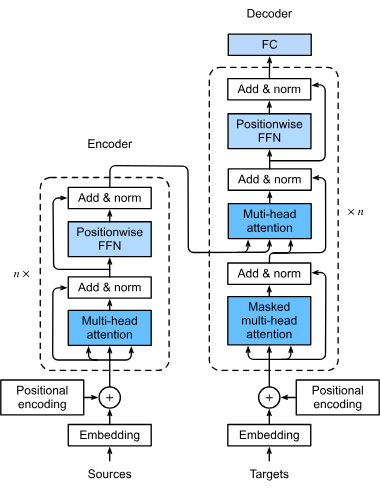
We construct a hybrid CNN + LSTM architecture that begins with 4 convolutional layers—each configured with 24 channels, a kernel width of 32, and appropriate padding (1800 for the past context and 200 for the future context). After flattening the convolutional output, the network proceeds through 4 LSTM layers. To identify the optimal configuration, we experimented with LSTM hidden sizes of 64, 128, and 256, training each variant for 30 epochs. The model with 128 hidden neurons per LSTM layer yielded the best performance in terms of CER (see section 4.3 for details). Based on these results, we then ran the final experiment using the 128 hidden size configuration for 50 epochs. A final linear layer with softmax activation produced the output probabilities, and the model was optimized using the Adam optimizer with a learning rate of 0.001. All experiments were run on a GPU with a DataLoader configuration utilizing 4 workers.



[caption] Figure X. CNN + LSTM block diagram

### 3.1.4. Basic Transformer

We also tested the pure Transformer architecture with a model setup of a single Transformer encoder layer placed right after the MLP flattening layer. Since the transformer encoder has non-linearity in the attention layer and a feed forward layer afterwards, an additional linear layer should be unnecessary. The Adam optimizer and 0.001 learning rate was used to train the model, same as the other models.



[caption] Figure X. Transformer architecture, depicting both the encoding and decoding components. [8]

### 

### 3.1.5. CNN + Transformer

In our hybrid CNN + Transformer architecture, we first process the raw EMG features through a CNN block that consists of spectrogram normalization, a multi-band rotation invariant MLP (with a hidden layer of 384 units), and a TDS convolutional encoder built from 4 sequential pairs of convolutional and fully connected blocks (each convolutional layer using 24 channels with a kernel width of 32 and appropriate padding). The CNN block outputs a flattened feature representation that is then enhanced with positional encoding before being fed into a Transformer.

Initially, we experimented with a Transformer having 4 encoder and decoder layers with 0.1 dropout, and we ran this configuration for 50 epochs; however, this setup overfit the training data (see section 4.5 for details). Consequently, we reduced the number of layers—testing configurations with 2 layers and 1 layer—and increased the dropout to 0.3 to improve generalization. Due to computational resource constraints, these latter models were only run for 30 epochs. Finally, a linear layer projects the transformed features to the target class space, with a log softmax activation applied for probability estimation. Training was conducted using the Adam optimizer with a learning rate of 0.001, and the overall data loading and training settings (batch size, window length, padding, etc.) remained consistent with our other experiments.

## 3.2. Algorithm Performance

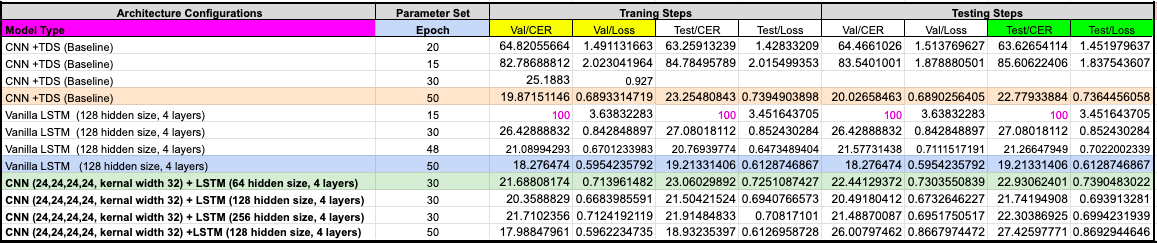


Table 1 - Summarized Performance of any CNN, LSTM, CNN+LSTM Algorithms Tested:

Light Orange = top performance for CNN+TDS baseline; Light Blue = top performance for vanilla LSTM; Light Green = top performance for CNN+LSTM

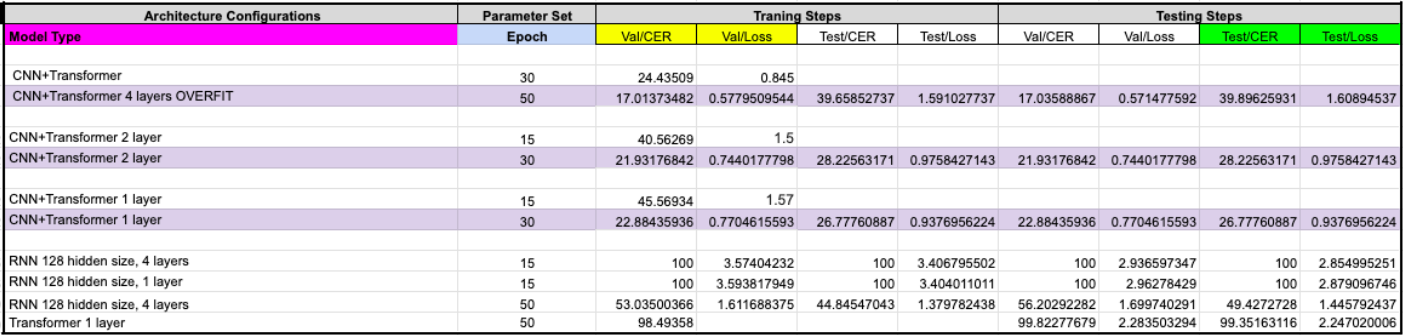


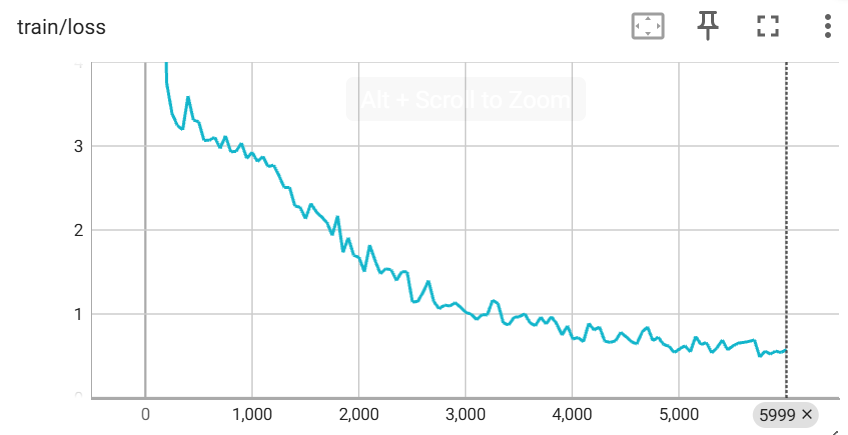
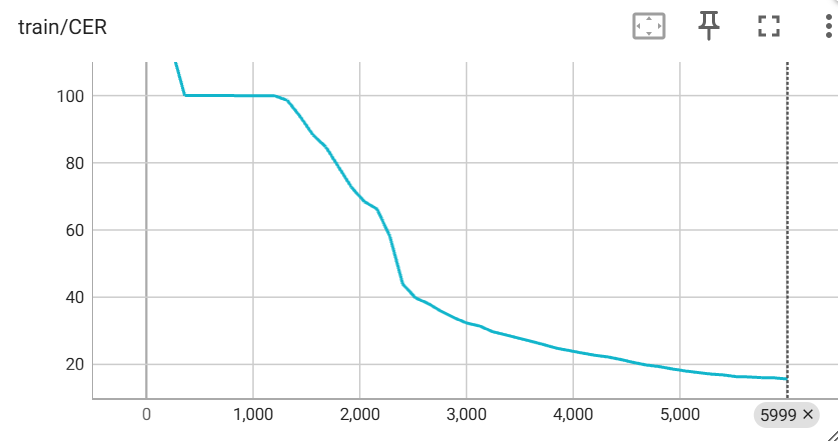
Table 2 - Summarized Performance of Basic RNN, Transformer, and Hybrid CNN+Transformer: Overall Top Performer = Light Purple

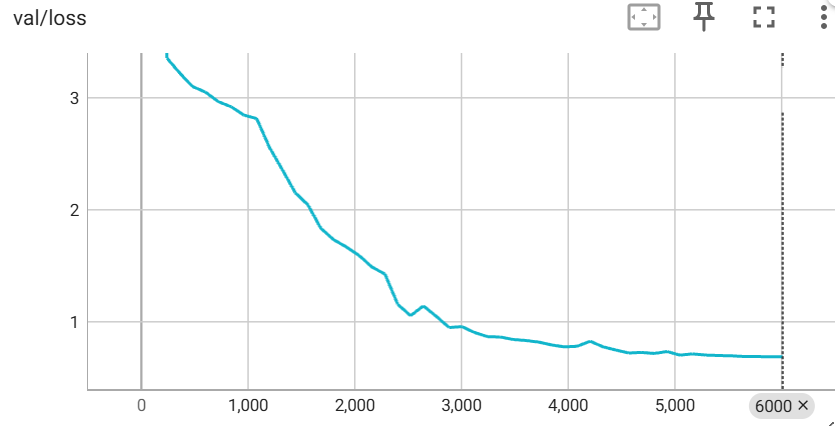
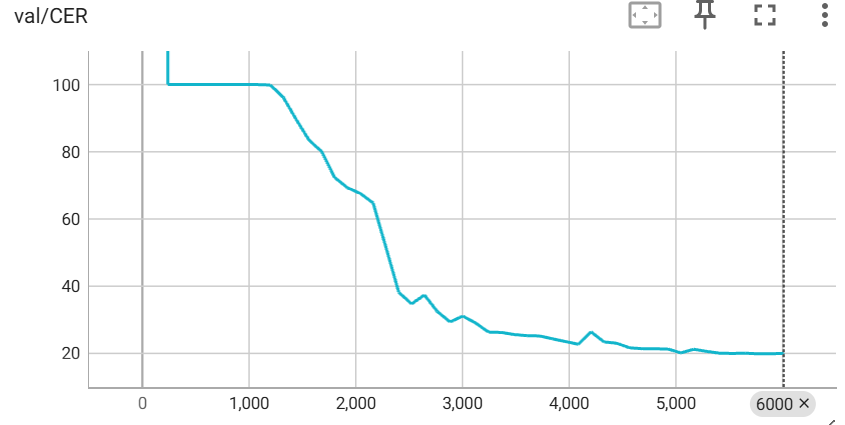
# **4. Results**

## 4.0 Baseline TDS-CNN

The results for the baseline Time Depth Separable CNN (TDS-CNN) model are shown in Figure X. Due to computational constraints, we reduced the maximum number of training epochs from the original 150 to 50 for our single-user experiments, while keeping all other hyperparameters unchanged. After training for 50 epochs, the baseline TDS-CNN—comprising 4 convolutional layers, a flattening operation, and a final linear layer with softmax activation—achieved a test CER of 22.78 and a test loss of 0.74. As illustrated in the graphs, the validation CER and loss decreased significantly over the training epochs, demonstrating improved generalization performance with extended training.

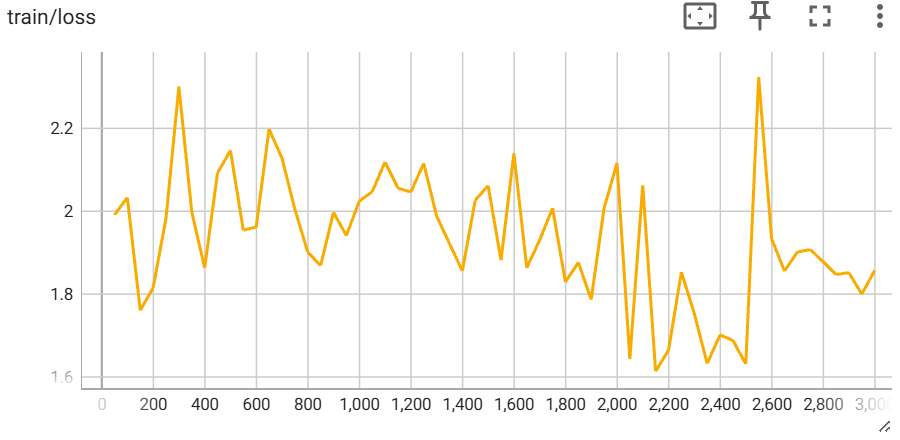
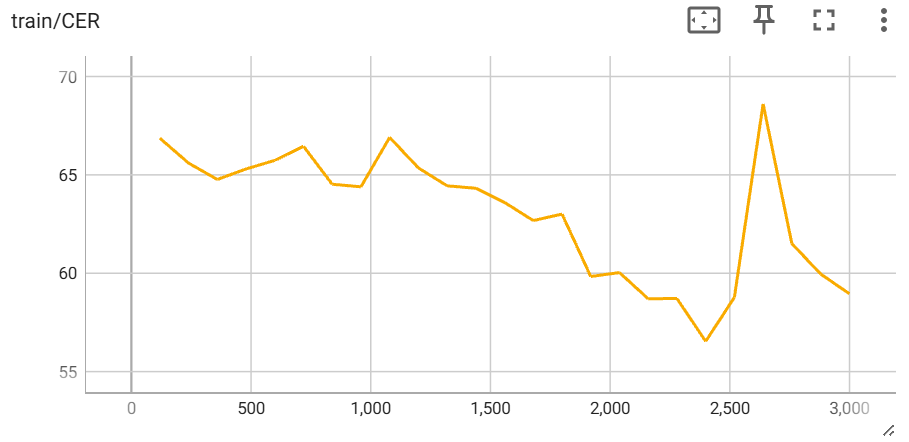
The training and validation losses follow a similarly steep initial decline, but display some minor oscillations as training progresses. These fluctuations are typical with optimizers like Adam, which can cause the loss to bounce slightly around local minima. The overall downward trend, however, indicates consistent progress in both training and validation sets, reinforcing that the network is effectively reducing error without overfitting too severely.

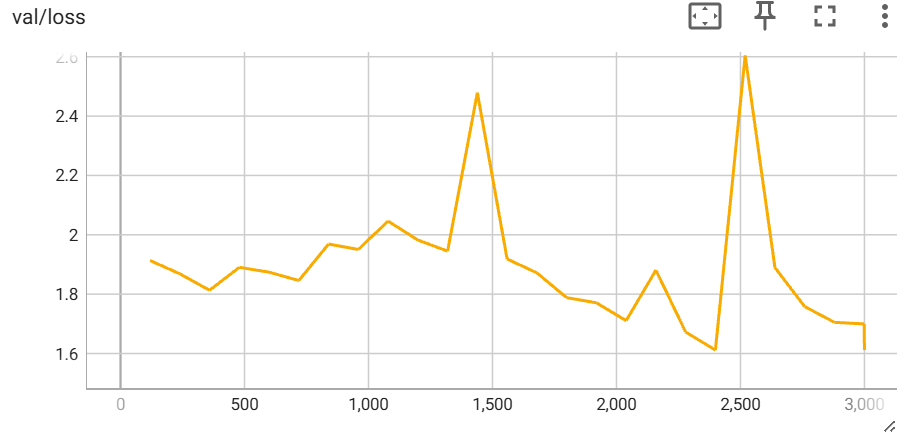
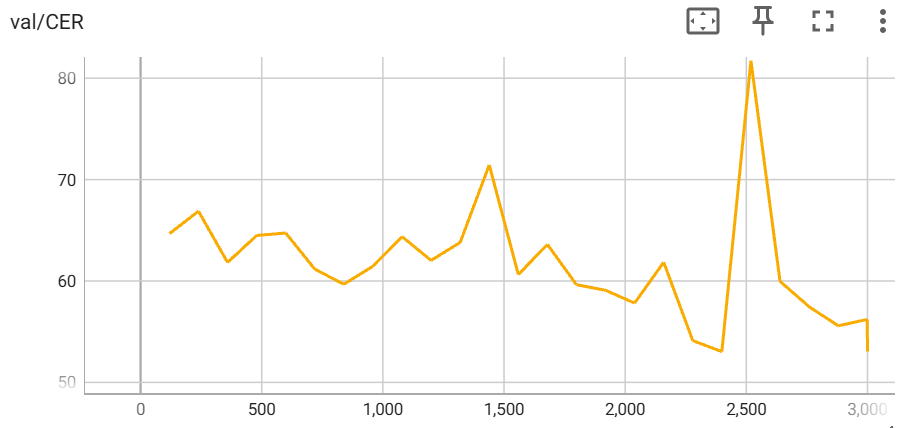




[caption] Figure X: Learning curves for the TDS-CNN baseline model showing training metrics (top) and validation metrics (bottom) for 50 epochs.

## 4.1. Basic RNN

****

****

[caption] Figure X. Training and validation metrics for the basic RNN model of the second half of the 50 epoch training run starting from the 25th epoch.

Recurrent Neural Networks (RNNs) are a machine learning architecture that allows for previous outputs to serve as inputs for the next time step. Since the EEG signals depict typing activity, our hypothesis was that past predictions for keypresses are correlated to future keypresses. Therefore, RNNs should perform relatively well. However, this architecture empirically performed very poorly.

Testing an RNN of hidden size 128 with 4 layers for 15 epochs caused the CER loss to be stuck at 100 for most of the initial epochs. Our initial assumption of the cause was either overfitting or vanishing gradient due to the RNN being too large. Therefore, we ran another training run with a smaller model with only 1 layer instead of 4. However, the issue still persisted with the CER loss being stuck at 100. Increasing the training epochs on the 4 layer RNN from 15 to 50 allowed the validation CER loss to decrease to 56.20, however this performance is much worse than the baseline TDS-CNN model which trained for 50 epochs and achieved a validation CER loss of 20.03. Surprisingly, the test CER loss was lower than the validation CER loss for the 4 layer RNN, with the test CER loss being 49.43. This could indicate a better generalization ability than the baseline model, but learning could also plateau much earlier than the baseline CNN due to inherent problems with the RNN model such as the vanishing and exploding gradient issue arising from backpropagation through time.

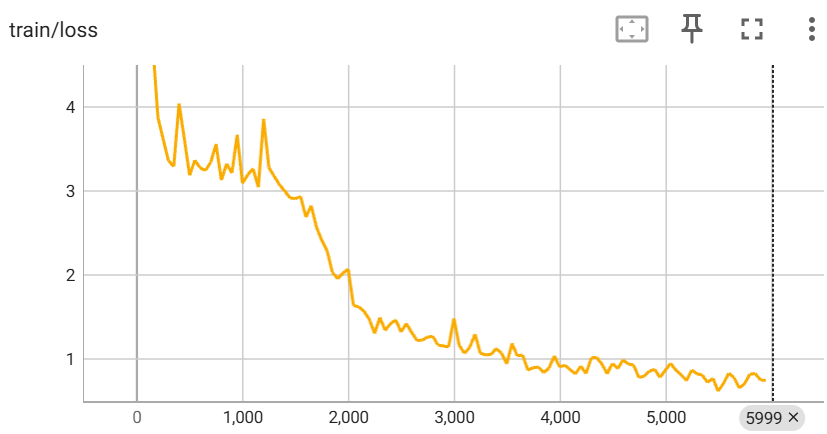
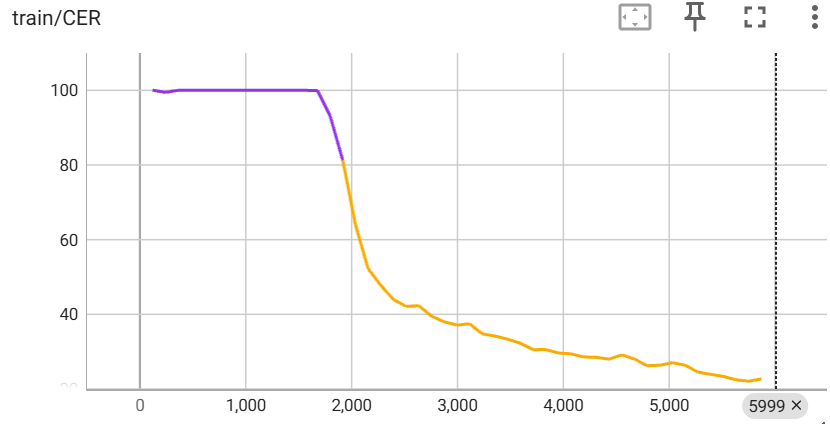
Next, we will examine a specific type of RNN that is designed to better capture long-term dependencies and mitigate the vanishing or exploding gradient problems common in vanilla RNNs.

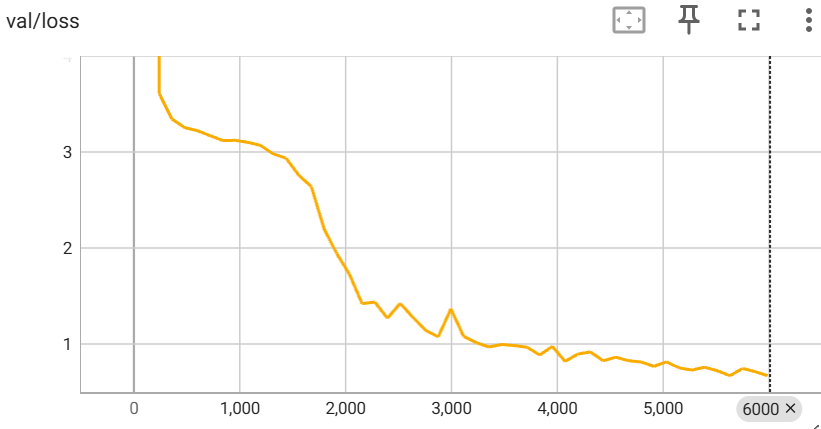
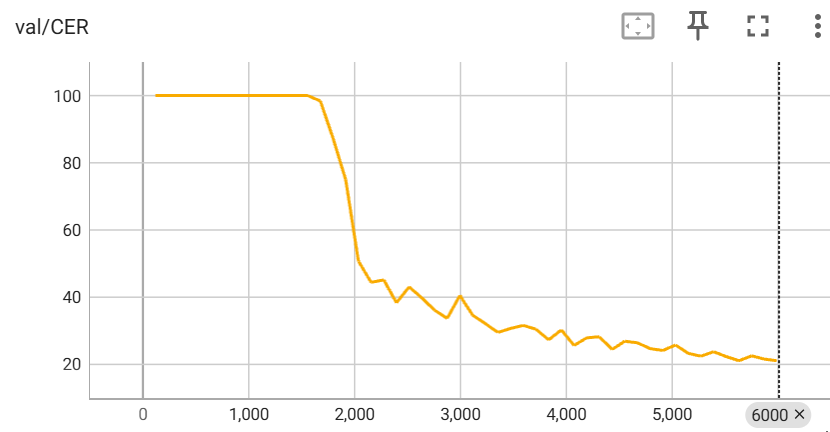
## 4.2. Basic LSTM

Our basic LSTM implementation consists of a bidirectional LSTM with 4 layers, each having a hidden size of 128 per direction, in addition to the other components that remain the same. This results in a test CER of 19.21 and a test loss of 0.61.

Over the course of 50 epochs, both training and validation metrics continue trending downward, with the validation CER showing no clear sign of plateauing by the end of training—suggesting that the model could likely benefit from additional epochs if computational resources were not constrained. Compared to the baseline TDS-CNN, the LSTM yields a lower test CER (19.21 vs. 22.78) and a lower test loss (0.61 vs. 0.74). Moreover, when contrasted with the extremely poor performance of the vanilla RNN, which recorded a test CER of 49.43, the LSTM’s superior handling of long-range dependencies via its gating mechanisms is evident, making it more suitable for the complex temporal patterns present in sEMG-based typing.

This improvement can be attributed to the LSTM’s ability to capture longer-range temporal dependencies, which is particularly useful in sEMG data where signals for successive keystrokes overlap and co-articulate. By learning these extended temporal patterns, the LSTM can better align EMG segments with the corresponding characters, thereby enhancing predictive accuracy.





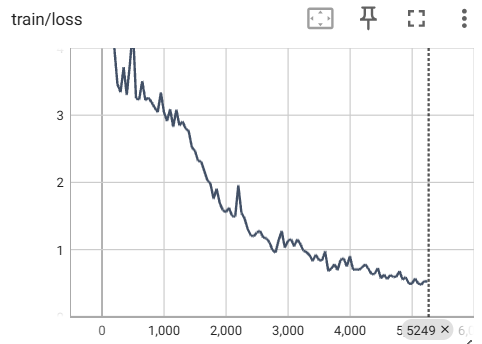
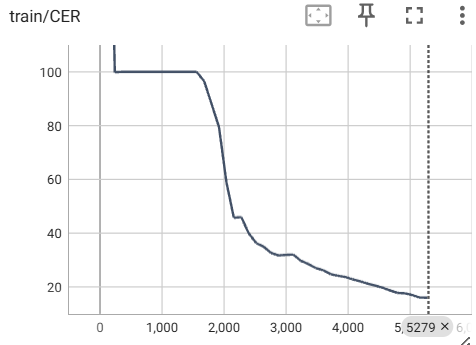
[caption] Figure X: Learning curves for the basic LSTM model showing training metrics (top) and validation metrics (bottom) for the first 48 epochs.

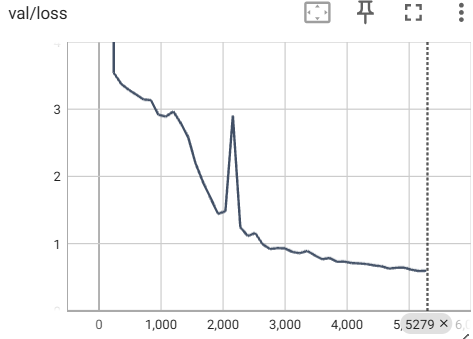
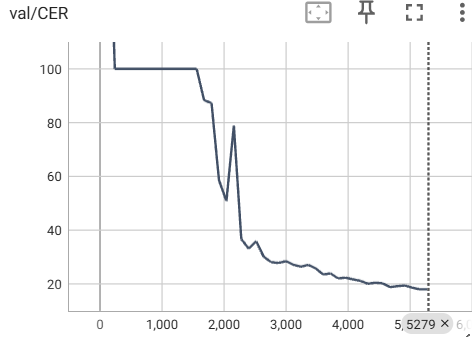
## 4.3. CNN + LSTM

As shown in Table 1, the optimal CNN + LSTM model with 4 convolutional layers, a flatten layer, 4 LSTM layers, and a linear layer with softmax activation results in a low test/CER of 27.43. Notice in the corresponding block diagram shown in Figure X in section 3.1.3, we used a flatten layer prior to the LSTM for this hybrid architecture. In addition, we utilized various hidden layer sizes of 64, 128, 256 - all at 30 epoch illustrated below (with a realistic run time of approximately 1 hour to each train and test).

* For hidden layer size 64: training Val/CER of 21.688, training Val/loss of .714, test/CER of 22.931, and test/loss of 0.739.
* For hidden layer size 128: training Val/CER of 20.359, training Val/loss of .668, test/CER of 21.742, and test/loss of 0.694.
* For hidden layer size 256: training Val/CER of 21.710, training Val/loss of .712, test/CER of 22.304, and test/loss of 0.699.

Thus, hidden layer size of 128 was the most favorable optimum between simplicity and complexity, since it had the lowest CER and loss data for both training and test cases. It is also confirmed that training and test CER trends are in sync with each other, as expected. So once it was understood that employing a hidden layer size of 128 was an ideal sweet spot at 30 epoch, giving us the lowest test/CER of 21.742, we increased the epoch to 50 for that hybrid CNN + LSTM configuration. This then resulted in the improved training Val/CER of 17.988 and Val/loss of 0.596. However, we observed that the test CER and test loss increased to 27.42 and 0.87 respectively. This took approximately 3 hours total between training and test and demonstrated overfit data. As observed below in Figure X (assessing 43 epoch), all training and validation parameters overall decrease, stabilizing to a better performance with increased iterations.

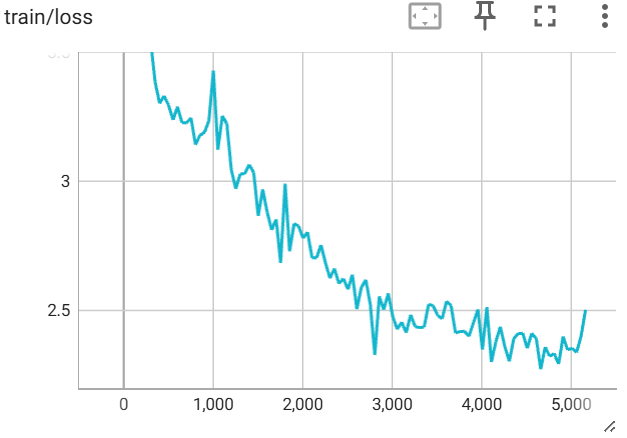
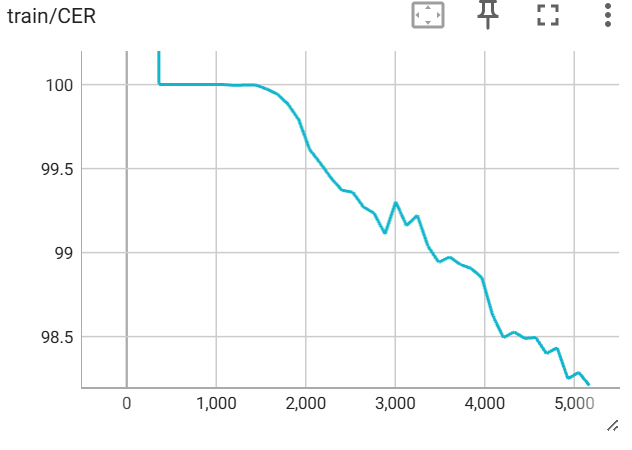


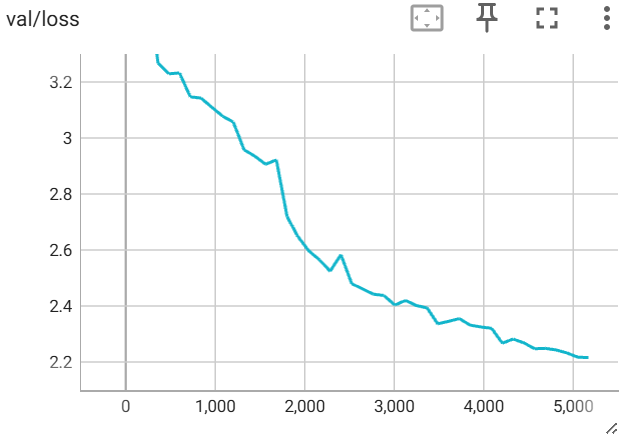
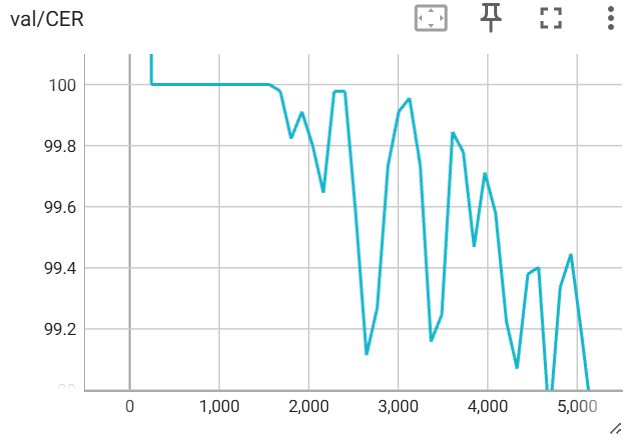


[caption] Figure X. Training and validation metrics for the 4 layer CNN + LSTM model with hidden size 128 up to the 43rd epoch.

## 

## 4.4. Basic Transformer





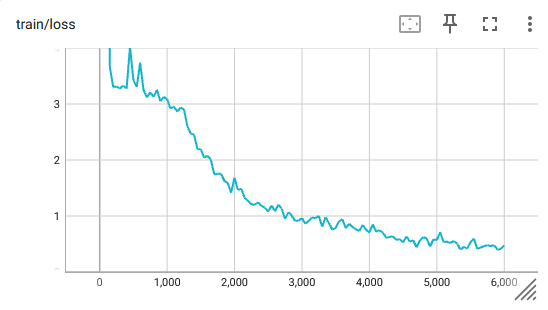
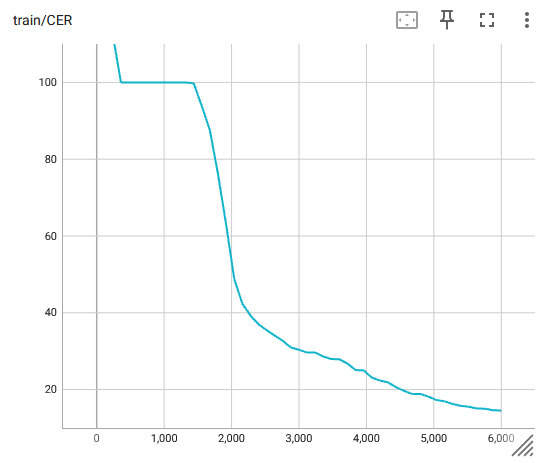
[caption] Figure X. Training and validation metrics of the single layer transformer architecture for 50 epochs.

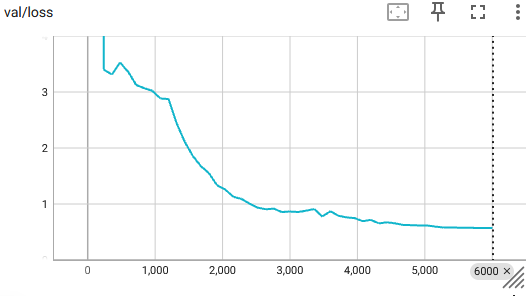
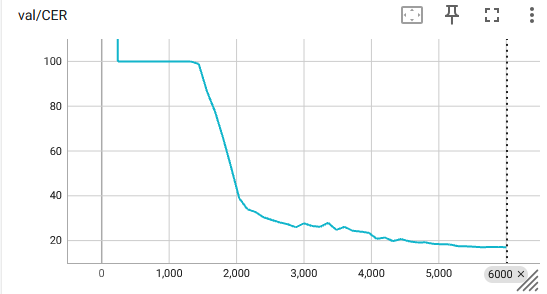
A small 1 layer transformer encoder layer was used to replace the TDS-CNN in the baseline model and subsequently trained for 50 epochs.Initially, the transformer faced the same issue as the basic RNN and LSTM model, where the CER loss would start out high at epoch 0 and then plateau at 100 for many epochs. Since the epochs were set to 50, the transformer encoder could train past the plateau and the training loss continued to be reduced, ultimately reaching a validation CER of 98.89. From the figures, it is apparent that performance of the transformer is much worse than the RNN after a similar number of training epochs despite the transformer being a more robust model than the RNN. This is likely due to the increased computational requirements to train transformers. Since the model is much more complex than RNNs, transformers likely require much more training than RNNs and LSTMs which are much smaller in comparison.

However, the increased training requirements of transformers likely results in better generalization performance and performance plateau, visible in the transformer’s relatively smooth and stable loss minimization curve compared to the RNNs. Although our transformer’s loss was much higher than the RNNs, other machine learning applications have shown the scalability and high performance of this model which makes it highly likely that increasing the training epochs of all our tested models will show the superiority of the transformer. However, we don’t currently have the data to support that claim in the context of this paper due to our limited computing resources.

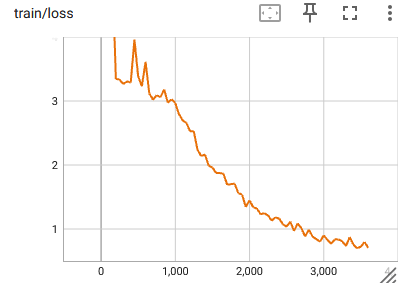
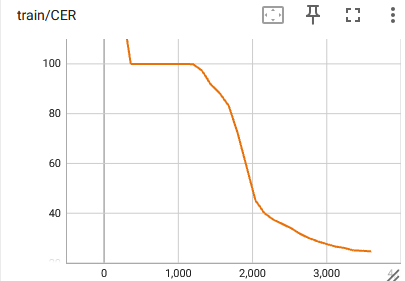
## 

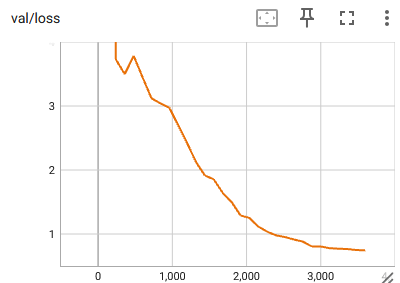
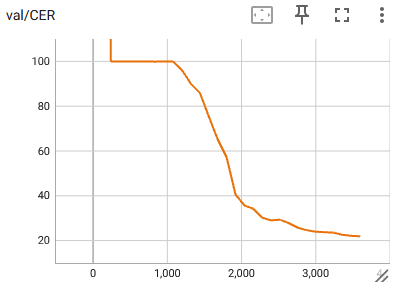
## 4.5. CNN + Transformer



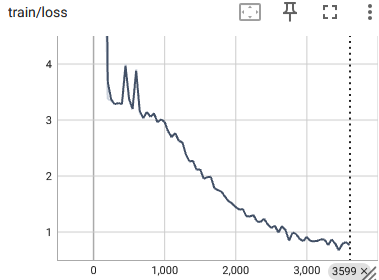
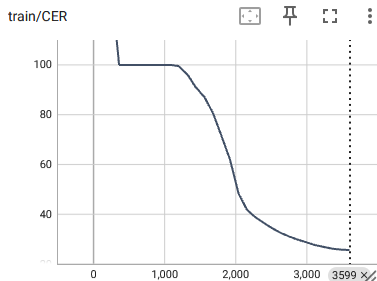


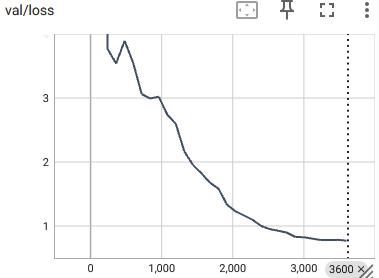
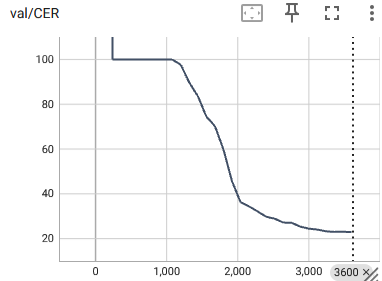
[caption] Figure X. Training and validation loss of hybrid 4 layer CNN + transformer with 0.1 dropout for 50 epochs.





[caption] Figure X. Training and validation loss of hybrid 2 layer CNN + transformer with 0.3 dropout for 30 epochs.

****

****

[caption] Figure X. Training and validation loss of hybrid 1 layer CNN + transformer with 0.3 dropout for 30 epochs.

The effects of adding dropout to the models were quite significant. The CNN + Transformer model was selected to test the effect of dropout on the generalization of the models. A small dropout of 0.1 was tested with a 4 layer model while a larger dropout of 0.3 was tested on both a 1 layer and 2 layer model. From the graphs above, it is shown that all the models have reached relatively stable plateaus when training was stopped. When comparing all models with 30 epochs of training, the CER validation error drops from the 1 layer to 2 layer model, which is to be expected since the 2 layer model is more complex. However, the validation error of the 4 layer model at 30 epochs is larger than the validation error of the 2 layer model, which could be a sign of overfitting. But, since the 4 layer model is much larger, there could be some other factors due to the size of the model resulting in slower loss optimization. To directly compare the CER validation loss for 30 epochs: 1 layer model loss was 22.884, 2 layer model loss was 21.931, 4 layer model loss was 24.435. Since larger models should perform at least as well as their smaller models, the 4 layer model was trained to 50 epochs when the validation CER reached 17.013.

The test CER for the 4 layer model at 50 epochs was 39.658 while the 2 layer model at 30 epochs was 28.225 and the 1 layer model also at 30 epochs was 26.777. The increase in test error with respect to model size is a sign of overfitting of the models. However, the validation error of the 0.3 dropout 1 layer and 2 layer models were very similar to the 4 layer model validation error despite being trained for fewer epochs. In addition, the test errors of the 1 and 2 layer models were very similar, only differing around 2 CER. But the test error of the 4 layer model was much higher than both the 1 and 2 layer models, demonstrating a worse generalization ability despite running for more training epochs. From this data, it can be concluded that the higher generalizability of the 1 and 2 layer models was due to the higher dropout rate during training, allowing the smaller models to outperform the larger model while receiving less training compute time.

# 5. Discussion

Overall, after trying various methods across the most telling parameters, we found it useful to observe how the baseline TDS-CNN consistently improved with increased epoch. Notice both training and testing time increased with increasing epoch (with minimum runtimes executed at 15 epoch for proof of concept). Therefore, we were able to use our resources effectively by carrying out substantially improved data trends by modifying parameters at a realistic epoch of 30. Note that once we found the favorable algorithm performance, additional GPU time was purchased for a more substantive discussion due to Google Colab limitations/constraints.

The RNN model was tested due to the assumption that previous keystrokes read by the EMG would affect the predictions of the following characters. Since NLP applications work by predicting the next tokens based on the current tokens, EMG prediction for typing output should be similar since it is only an adding an extra step in interpreting the signals to keystrokes. However, empirically testing the models has shown a very poor performance. Similar to some other models tested, the RNN model results in a plateau at validation CER 100.0, which was found to be resolved by increasing the number of epochs. Even at 50 epochs of training, the validation CER is 53.035 and the testing CER is 44.845. This is significantly worse performance than the baseline CNN model and the CNN + LSTM models which reached a CER below 20 for both the test and validation loss at 50 epochs. A possible cause of the lacking performance of the RNN model was the limited context window of the autoregressive connections. In addition, backpropagation through time is well known to have issues with vanishing gradients, which can cause learning to plateau. Dropout and adding more diverse architecture layers to the RNN could create a better model, but since LSTMs are known to be superior to vanilla RNNs in most applications, pursuing replacing the RNNs with LSTMs would likely yield much better results.

Next, we employed vanilla LSTM to test our theory, which similarly performed better with increasing epoch. Therefore, we were able to solidify that this was the most advantageous configuration at 50 epoch, resulting in the overall lowest test/CER of 19.213. Notice we clearly have developed an understanding that performance generally improves with increased epoch per the validation and testing parameters. Especially with vanilla LSTM, the graphs of validation CER and loss show no sign of plateauing at 50 epochs and rather they continue to decrease, which means that the performance has potential to improve with further training. Though it is essential to caution that with high epoch values (accumulating unrealistic runtimes) the risk of overfitting may be introduced, since improved validation parameters may not come with the appropriately improved testing performance.

In addition, when employing our initial CNN + LSTM hybrid model, we had more potent observations when decomposing the hidden layer sizes (consisting of 4 total layers) of 64, 128, 256. Overall best performance was at 30 epoch, utilizing a hidden layer size of 128. This reasonably fitted configuration optimizes the best of both worlds between complexity (i.e. after 256 overfits) and simplicity (i.e. before 64 underfits) at these hidden layers. Note that with a further increase to 50 epoch, the test/CER increased from 21.742 to 27.426 despite the training Val/CER decreasing from 20.359 to 17.988, which could point to overfitting of data. As trended in the plots complementing our findings, as iterations increased, both CER and loss parameters clearly decreased to stable training/validation performance.

The Transformer model was also tested because of its generalizability and robustness in multimodal applications. Since the problem the model needs to learn is to transform EMG signals into text, the thought was that transformers would perform well since they were originally designed to improve machine translation. Only the transformer encoder was used in the tested model because the task is more similar to classification than generation. A 4 layer transformer encoder model was tested originally but memory bottlenecks prevented the model from running properly. Instead, a 1 layer transformer encoder model was tested which required 50 epochs to break the 100.0 CER plateau. The final validation CER at 50 epochs was only 98.493, which is very poor compared to all other models. A possible problem with the encoder only approach is the lack of local feature extraction, which is a strength of CNNs. Since transformers use its attention mechanism to gather global context, local features could be drowned out. Transformers also require a relatively large amount of data to generalize well, which would result in poorer performance compared to CNNs for smaller datasets.

To combine the strengths of the two architectures, a CNN + Transformers model was tested, which also showed that dropout allowed smaller models to outperform larger models on test error when the dropout parameter was set high enough. The dropout of the smaller models was set to 0.3 while the larger model was set to 0.1. The smaller models received only 30 epochs of training while the larger model received 50 epochs due to the validation error being higher than the smaller models when the larger model was at 30 epochs. From the results of this data, it is apparent that dropout has a significant effect on the generalization error of our models and make the models less prone to overfitting. The models with increased dropout had a lower difference between the CER loss of the validation and test runs, while the lower dropout model had a larger difference between the validation and test CER loss.

Due to lack of compute resources, multiple variables (epochs, model size, dropout) were changed between runs in order to obtain the best model possible rather than attempting to narrow down the effect of each change individually. While it can be assumed that dropout had an effect on the generalization error of the models, since the model size and number of epochs were also changed, it is can not be definitively determined whether the improved generalization was purely due to dropout or whether the performance of the larger 4 layer model was degraded due to overfitting. For future test runs, more A/B testing with identical models and only differing dropout rates are required to determine definitive causation between dropout and generalization for this particular application.

the pure transformer encoder model was lacking, requiring 50 epochs to break past the 100.0 CER plateau and ending the training run with 99.351 CER. However, hybrid models containing transformer layers in addition to other non-transformer layers achieved much higher performance than a pure transformer model. The CNN+Transformer models achieved sub 30 CER on test data after only 30 epochs for both the 1 layer and 2 layer models. Since dropout was implemented during the training of the hybrid model and not the pure transformer model, it is difficult to attribute the increased performance to only a single one of the changed architecture or the addition of dropout layers. However, the hybrid architecture is most definitely the largest cause of the performance improvement since the validation CER was 24.435 for a 4 layer CNN+Transformer model at 30 epochs even when dropout was reduced from 0.3 to 0.1.

However, empirical testing showed that the RNN models performed poorly and learned very slowly, drawing doubt to the initial assumption. However, subsequent testing of models with LSTM layers performed very well, with the 4 layer 128 hidden parameter LSTM model performing the best out of all models tested, achieving a test CER of 19.213 in 50 epochs. In comparison, the 4 layer 128 hidden parameter RNN model only achieved a test CER of 49.427 after 50 epochs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Key Take-Away (last concluding sentence may be group effort)

Deep learning classification has been successfully applied to many EEG tasks, including sequence recognition. In particular, our results verifies that the proposed hybrid methods are quite promising choices for keystroke prediction, because of its powerful ability to learn features from raw data directly. We recommend more in-depth research of these CNN+Transformer and CNN+LSTM combinations, particularly the number and arrangement of different layers especially pertaining to. EEG classification tasks explored within deep learning applications…We can decipher that hybrid designs incorporating transformer layers in addition to other non-transformer layers achieved much higher performance than a pure transformer model. Notable considerations for the lack of performance for the pure transformer was the global attention mechanism resulting in the failure to learn local features. In addition, transformers require a large amount of data to generalize well, which could not be achieved with our limited data set. Thus, by adding a CNN, which is strong at classifying local features and feeding them into a transformer encoder, the CNN output acts as an input embedding, which had no analogous component in the pure transformer model. As a result, the CNN + Transformer model performed much better than other hybrid models. Although the hybrid architecture performed worse than the CNN alone, the scalability of transformers is well known, which leads to the hypothesis that performance of the transformer should improve beyond the performance plateau of the CNN. However, this requires further testing to verify.

**6. References**

1. Craik A, He Y, Contreras-Vidal JL. Deep learning for electroencephalogram (EEG) classification tasks: a review. J Neural Eng. 2019 Jun;16(3):031001. doi: 10.1088/1741-2552/ab0ab5. Epub 2019 Feb 26. PMID: 30808014.
2. Salma Alhagry, Aly Aly Fahmy and Reda A. El-Khoribi, “Emotion Recognition based on EEG using LSTM Recurrent Neural Network” International Journal of Advanced Computer Science and Applications(IJACSA), 8(10), 2017.
3. R. Schirrmeister et al. Deep learning with convolutional neural networks for eeg decoding and visualization. Hum Brain Mapp., 14:5391–5420, Nov 2017.
4. Xu G, Ren T, Chen Y, Che W. A One-Dimensional CNN-LSTM Model for Epileptic Seizure Recognition Using EEG Signal Analysis. Front Neurosci. 2020 Dec 10;14:578126.
5. Alex Graves, Santiago Ferna ́ndez, Faustino Gomez, and Ju ̈rgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural ’networks. volume 2006, pages 369–376, 01 2006.
6. ​​Sutskever I, Martens J and Hinton G E 2011 Generating text with recurrent neural networks Proc. of the 28th Int. Conf. on Machine Learning pp 1017–24
7. Moradi, Milad & Samwald, Matthias. (2022). Deep Learning, Natural Language Processing, and Explainable Artificial Intelligence in the Biomedical Domain. 10.48550/arXiv.2202.12678.
8. [Transformer Figure Link] <https://d2l.ai/chapter_attention-mechanisms-and-transformers/transformer.html>

Elmekki, H., Bentahar, J., Rjoub, G., & Pedrycz, W. (2023). A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. *arXiv.Org*, *abs/2306.07303*. https://doi.org/10.48550/arXiv.2306.0730