Testing LSTM and Transformers on emg2qwerty

Ava Gonick
Department of Computational and Systems Biology
University of California - Los Angeles
Los Angeles, CA 90095
gonicka656@g.ucla.edu

Wren Xue
Department of Computer Science
University of California - Los Angeles
Los Angeles, CA 90095
yxu404@g.ucla.edu

Mark Epstein
Department of Computer Science
University of California - Los Angeles
Los Angeles, CA 90095
mepstein0003@g.ucla.edu

John Connell
Department of Computer Science
University of California - Los Angeles
Los Angeles, CA 90095
johnconnell@g.ucla.edu

Abstract

Our goal was to improve upon the test CER of the baseline CNN model in classification of the emg2qwerty set [1]. Due to limited computing resources, we only trained with the data from a single subject. We tested modifying the baseline model with batch normalization and additional rotations. Additionally, we implemented a Long Short-Term Memory (LSTM) model, LSTM/CNN hybrids, a transformer model, and transformer/CNN hybrids. Only one of the LSTM/CNN hybrids managed to achieve marginally better performance than the baseline CNN model.

1 Introduction

Significant research has been conducted in the field of brain-computer interfaces with the aim of restoring motor and communication abilities to individuals with paralysis. Applying a proper classification of wrist sEMG data could assist people with carpal tunnel syndrome, paralysis, or nerve damage in the hands, enabling them to type efficiently, a skill essential in today's digital age.

Our objective was to minimize the CER of the model on a single subject. Since typing cadence varies across individuals, differences in temporal information could impact model performance and personalization. Given that the data was both sequential and temporal, we believed that maintaining some idea of the past would improve our model. To incorporate this, we explored transformers, which can capture sequential and temporal information through their self-attention mechanism and positional encoding. Transformers generally perform well on sequential text, and we anticipated similar effectiveness in our application. Additionally, we explored LSTMs, another architecture suited for sequential/temporal information, in order to compare performance against transformers. We also explored hybrid approaches, combining both transformers with CNNs and LSTMs with CNNs to combine the benefits of feature extraction from CNNs with the sequential/temporal modeling strengths of transformers and LSTMs. Finally, we explored various optimization techniques to improve the performance of the baseline CNN model, ensuring that the performance we were getting was not trivially achievable.

2 Methods

2.1 Batch Normalization

The baseline CNN model did not contain any batch normalization between the TDSTransformerTotal module and the TDSConvEncoder module. We added batch normalization to normalize the activations before passing them into the convolutional encoder. The model was trained for thirty epochs.

2.2 Additional Rotations

Checking if more rotation invariance would account for band movement on the subject, we added two new layers for two rotations in the counter-clockwise direction and two rotations in the clockwise direction. This gives five rotations (-2, -1, 0, 1, 2). The model was trained for thirty epochs.

2.3 Transformer

2.3.1 First Implementation

We implemented a transformer to replace the TDS ConvEncoder. This involved implementing a positional encoder after the MultiBandRotationInvariantMLP module so our transformer could learn information about the sequential order of the data which the CNN could extract intrinsically. The created TransformerEncoderLayer block has the same number of features as taken into the CNN, with the number of attention heads equal to the number of block channels in the baseline CNN (twenty-four). We added an attention mask that allows the model to only look at one second in the past, paralleling the window given by the baseline CNN. Each TransformerEncoderLayer block includes three Transformer Layers. To mimic the four CNN layers in the baseline model, we used four TransformerEncoderlayers.

Due to the large number of parameters in this model, training took much longer than the baseline CNN implementation and could only be run for 15 epochs on Google Colab.

2.3.2 Reducing Complexity

Due to compute limitations, we had to make the model smaller. We went from three Transformer Layers to two and had only one TransformerEncoderLayer block instead of four. We also halved the number of attention heads to twelve. This allows us to train the model for thirty epochs.

2.3.3 Tuning

We also tested increasing the attention mask window from one second to two seconds as well as removing positional encodings. All tests were done for thirty epochs.

2.4 Transformer/CNN Hybrid

2.4.1 First Implementation

We took the architecture of the best transformer (smaller architecture with one second window), and made it less complex by having only one transformer layer instead of two. We then took the baseline CNN and made it use two blocks instead of four. The model was trained for thirty epochs.

2.4.2 Tuning

Based on testing, our model was overfitting. We tested with dropout values of 0.2 and 0.3 in the transformer to try and limit the complexity of the transformer. We also separately tested adding decay rate values of e^{-4} and e^{-3} to add L2 regularization. The model was trained for thirty epochs.

2.4.3 Increased Complexity

Based on our tuning testing, the model was underfitting with a plateauing CER. We added an additional block to the CNN part of the model to have three blocks and decreased the dropout rate to

0.15.

Due to issues with Google Colab, this could only be run for fifteen epochs.

2.5 LSTM

2.5.1 Minimal Two Layer LSTM

We implemented a minimal two-layer LSTM with a hidden size of 128. The model was originally trained for thirty epochs, but we further tested with fifty-five epochs.

2.5.2 Four Layer LSTM

We increased the number of layers to four and the hidden size to 200. The model was trained for thirty epochs.

2.5.3 CNN + LSTM

We fed the output of the CNN to the previous four layer LSTM network. The model was trained for thirty epochs.

2.5.4 LSTM Layer

We modified the Minimal Two Layer LSTM with a 0.1 dropout rate between the layers, as well as normalization applied to the short-term output of each layer. The model was trained for thirty epochs.

2.5.5 CNN + LSTM Layer

We fed the output of the CNN to the previous LSTM Layer. The model was trained for thirty epochs.

3 Results

3.1 Baseline

We first discuss the baseline results to set a control in our comparison. The baseline implementation has a character error rate (CER) of 28.9172 on test. This matches with the expected CER of 30 that the spec claims the baseline achieves.

3.2 Batch Normalization

The implementation of batch normalization on the baseline model made the model perform significantly worse with a CER of 34.7742.

3.3 Rotation

Adding extra rotations to the model insignificantly improved the model with a CER of 28.8308. Adding additional rotations beyond the 5 tested made the model perform significantly worse, indicating that the average rotation in the band generally fell between 0 and 2 rotations in either direction.

3.4 Transformer

3.4.1 First Implementation

Our large transformer model made the model substantially worse with a CER of 99.5. It is possible that we could have improved with more epochs, but since Google Colab crashed it is not possible to recover the graphs to verify this.

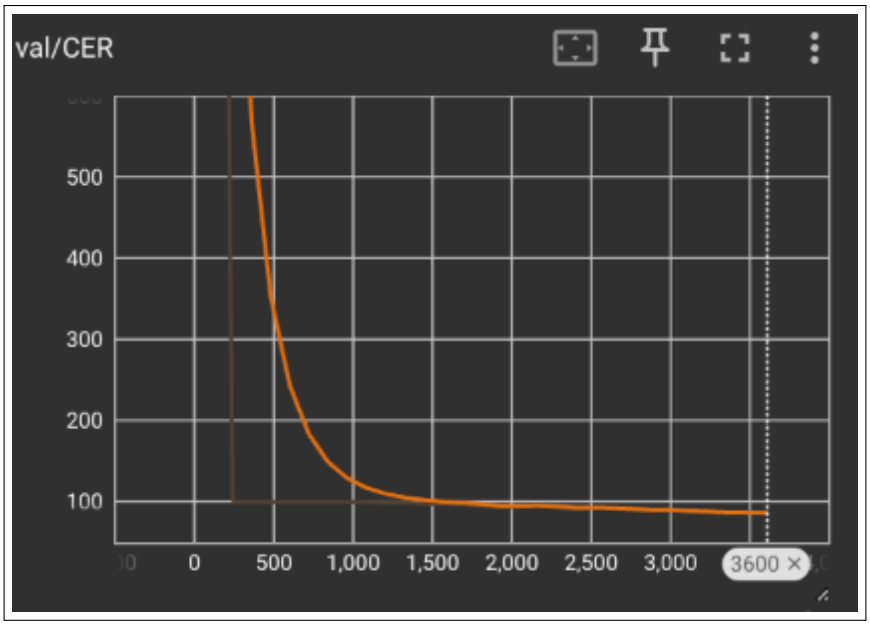

Figure 1: Validation CER of Reduced Complexity Transformer model.

3.4.2 Reducing Complexity

Making the model smaller led to a final test CER of 89.5 likely due to the longer training time. Looking at Figure 1, the validation CER is still decreasing albeit slowly so it could be the model could perform better with more epochs.

3.4.3 Tuning

We tested increasing the attention mask window to two seconds from one, yielding a CER of 100. This seems to show that only the last second of information is relevant. Similarly, removing positional encodings, irrespective of attention mask window size, while mostly a sanity check on the importance of sequential information, expectedly also yields a CER of 100.

3.5 Transformer/CNN hybrid

3.5.1 First Implementation

When training the model, we achieved a validation CER of 16.99, but during testing we achieved a CER of 38.7.

3.5.2 Tuning

Adding dropout to the model increased the test CER to 100. As per Figure 2, we observed that validation CER stayed at 100 throughout training.

Implementing a decay rate instead also failed to improve performance, with a decay rate of e^{-4} yielding a testing CER of 43.8285 and e^{-3} yielding 46.8385.

3.5.3 Increased Complexity

Increasing the complexity of the model did not improve on the original Transformer/CNN hybrid.

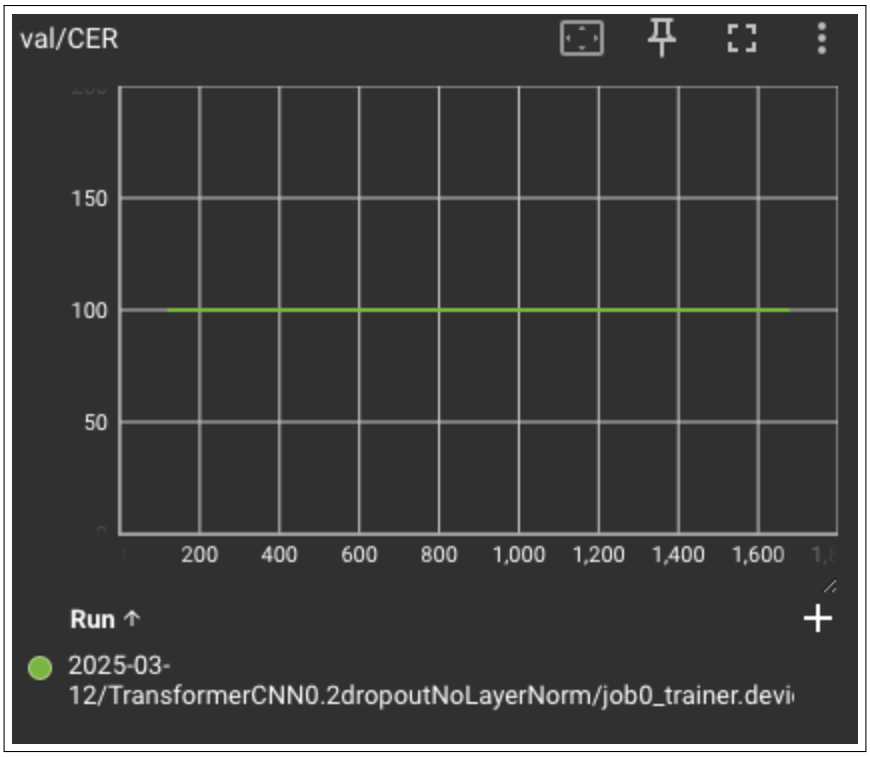

Figure 2: Validation CER of Transformer/CNN hybrid with 0.2 dropout

3.6 LSTM

3.6.1 Minimal Two Layer LSTM

Ultimately, we didn't manage to achieve anything with a test CER of 100 when testing for thirty epochs. However, with fifty-five epochs, we can see in Figure 3 that the validation error breaks out of the plateau and we manage to achieve a test CER of 36.5.

3.6.2 Four Layer LSTM

The model plateaued, achieving a test CER of 100.

3.6.3 CNN + LSTM

The CNN did not manage to change the input to the four layer LSTM enough to stop the test CER from being 99.978.

3.6.4 LSTM Layer

As seen in Figure 4, the LSTM Layer is able to pass the plateau faster, and is thus able to get better accuracy within thirty epochs. We expect that with more epochs, the LSTM Layer would perform better than the minimal LSTM. With thirty epochs, the LSTM Layer model gets a test CER of 89.89.

3.6.5 CNN + LSTM Layer

As seen in Figure 5, we manage to pass the original plateau very quickly and start to plateau again near the end at a much lower CER. Our final test CER with this is 27.64, an 8.6% improvement over the validation CER of 30 measured by the spec and a 4.7% improvement over the test CER measured in our baseline model.

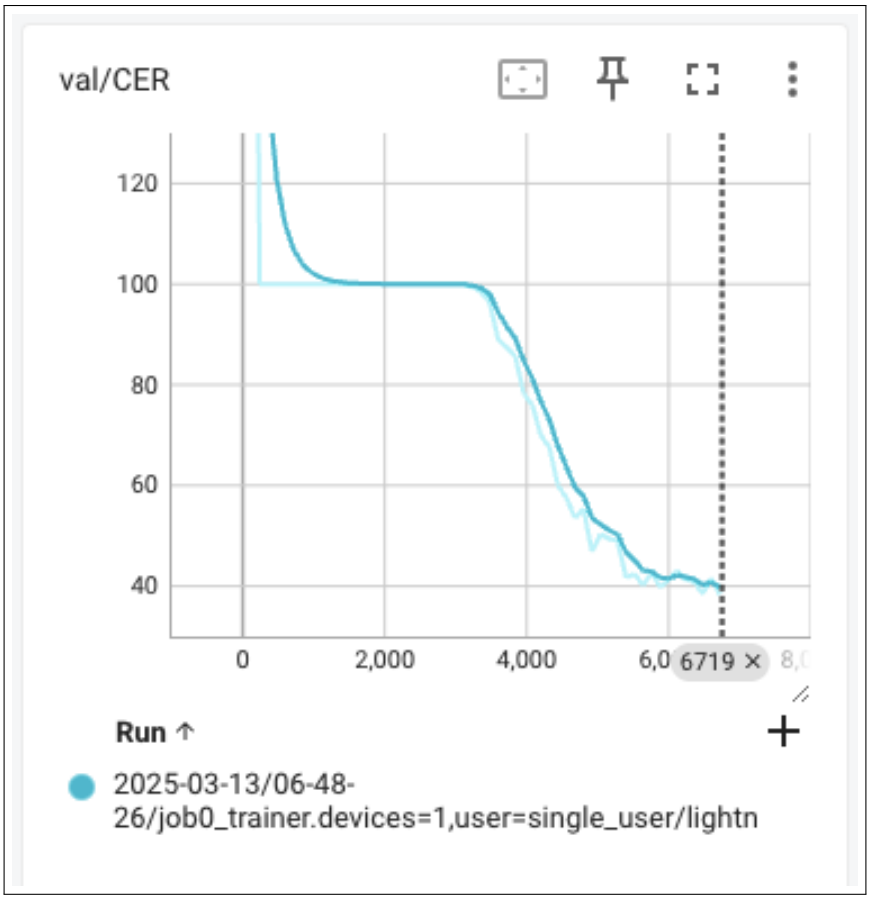

Figure 3: Validation CER of Minimal LSTM with 55 epochs.

4 Discussion

Ultimately, we believe we didn't have the compute resources necessary to truly explore the possibility of using more complex models on the problem. Thirty epochs is generally not optimal when training larger models, and we saw in the LSTM that increasing the epoch count could increase performance substantially. It is clear that some sequential information is very helpful, as transformer models without positional encodings did terribly, while those that had them in addition to correct window information did decently well, and LSTM also performed quite well. We suspect transformers could have performed better had we used the entirety of emg2qwerty, as transformers generally require a lot of data to perform well. Modifications to the baseline CNN model had minimal positive impact, suggesting that the original model was already well-optimized for several parameters. We hoped that the transformer combined with the CNN would be able to perform better as transformers are better at getting long range information and CNNs are better at short range, but ultimately we found that we were overfitting. Attempts to stop the overfitting were unsuccessful, either still overfitting or badly underfitting. We believe that the CNN + LSTM Layer performed well because the CNN helps extract local features such as spikes or frequency-specific details, leading to a reduced and more informative feature representation. This improved data can be fed to the LSTM Layer to track information across the entire sequence, helping to keep the correct important information relevant.

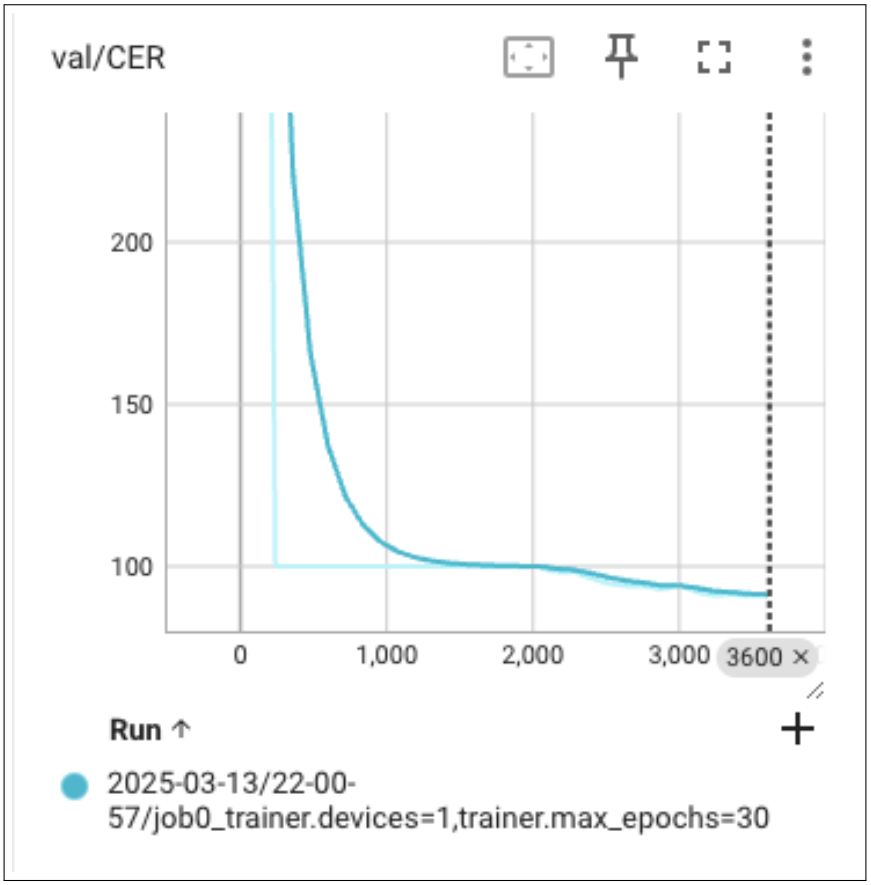

Figure 4: Validation CER of LSTM Layer.

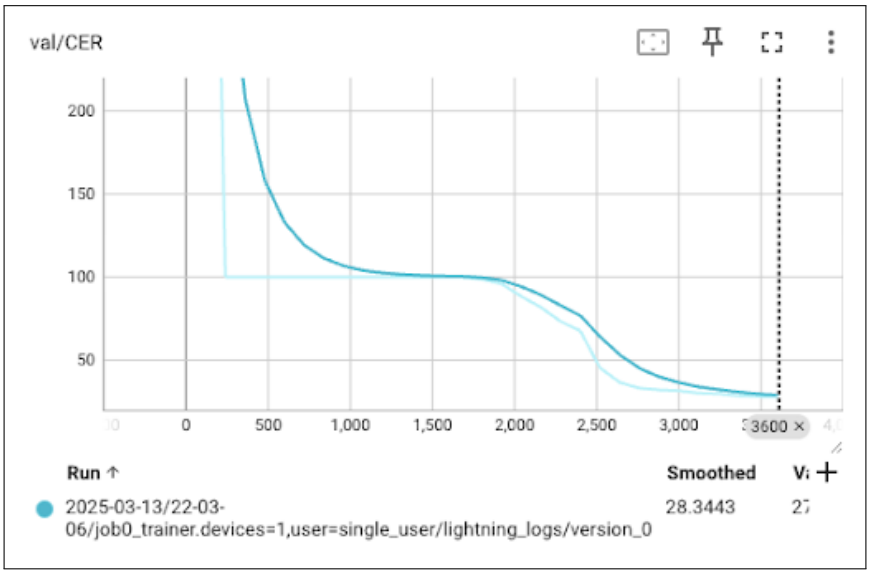

Figure 5: Validation CER of CNN + LSTM layer.

References

[1] Viswanath Sivakumar, Jeffrey Seely, Alan Du, Sean R Bittner, Adam Berenzweig, Anuoluwapo Bolarinwa, Alexandre Gramfort, and Michael I Mandel. emg2qwerty: A large dataset with baselines for touch typing using surface electromyography, 2024.