Predicting Keystrokes from Electromyography Signals

Abby Seseri
Department of Computer Science
University of California, Los Angeles
abbyseseri@ucla.edu

Abstract

In this project, I attempted to improve a model that predicts keystrokes from surface electromyography (sEMG) signals using deep learning architectures. The baseline model, which utilized a Time-Depth Separable Convolutional (TDSConv) encoder, was altered to include recurrent architectures such as LSTMs, GRUs, and hybrid CNN-RNN models. I evaluated the performance of these architectures on the emg2qwerty dataset by reducing the Character Error Rate (CER) for a single subject. My experiment included changing model architectures, hyperparameters, and the number of electrode channels. The results suggested that the TDSLSTMEncoderModified architecture, with a hidden size of 512 and dropout of 0.2, achieved the best performance, significantly outperforming the baseline and other architectures. This project offers insights into the effectiveness of different architectures for sequential data and potential directions for further work.

1 Introduction

The emg2qwerty dataset allows the decoding of sEMG signals into keystrokes to be studied. The applications of this can greatly benefit a disabled person, allowing the participant to communicate or move using a neural interface. While the baseline TDSConv model has a decent performance, I attempted to use recurrent architectures that are better suited for sequential data to improve the accuracy further. This project tests the effectiveness of LSTM, GRU, and hybrid CNN-RNN architectures in decreasing the CER. Additionally, I experimented with the impact of different architectural changes, such as dropout and layer normalization. My goal is to design deep learning models for sEMG signal decoding that balance performance and computational efficiency. I focused on minimizing CER in a single subject because a personalized model can help reach a higher accuracy for that subject.

1.1 Motivation

The ability to accurately decode sEMG signals into keystrokes has crucial implications for assistive technologies, particularly for disabled people with motor impairments. Although the baseline TDSConv model has reasonable performance, its reliance on convolutional layers may prevent it form capturing long-term temporal dependencies that is inherent in sEMG data. Recurrent architectures like LSTMs and GRUs are suited for sequential data and should improve performance. Furthermore, hybrid architectures combining convolutional and recurrent layers could potentially benefit from both spatial and temporal features. By comparing these architectures, this project aims to find the most effective approach for sEMG-based keystroke prediction.

2 Methods

2.1 Dataset and Preprocessing

The emg2qwerty dataset includes sEMG signals recorded from 16 electrode channels on each wrist, sampled at 2 kHz, and respective keystroke labels. For this project, I focused on a single subject (#89335547). The dataset was split into training, validation, and test sets as provided in the baseline code. Inputs were preprocessed using spectrogram normalization and multi-band rotation-invariant MLP transformations.

2.2 Model Architectures

I replaced the baseline TDSConv encoder with the following architectures:

2.2.1 TDSLSTMEncoder

The TDSLSTMEncoder is a bidirectional LSTM-based encoder with a fully connected block. LSTMs (Long Short-Term Memory networks) are appropriate for sequential data because they can find long-term dependencies in time series. The bidirectional nature of the LSTM allows the model to process the input sequence in both forward and backward directions. This architecture was chosen to determine if a purely recurrent model could outperform the baseline convolutional model.

2.2.2 TDSLSTMEncoderModified

The TDSLSTMEncoderModified is an enhanced version of the TDSLSTMEncoder because it includes dropout and layer normalization. Dropout randomly deactivates neurons during training to prevent overfitting, and layer normalization stabilizes the training process by normalizing the inputs. These alterations can help the model by improving its generalization ability and robustness. This architecture was chosen to check if the enhancements could further improve performance.

2.2.3 TDSGRUEncoder

The TDSGRUEncoder is a bidirectional GRU-based encoder with a fully connected block. GRUs (Gated Recurrent Units) are a variant of LSTMs that use fewer parameters and are computationally more efficient. They also still are capable of capturing temporal dependencies. This architecture was chosen to compare the performance of GRUs against LSTMs.

2.2.4 CNNRNNEncoder

The CNNRNNEncoder is a hybrid architecture combining 1D convolutional layers for feature extraction and an LSTM for temporal modeling. The convolutional layers are effective at extracting local patterns and spatial features from the sEMG signals, and the LSTM finds temporal dependencies. This architecture was chosen to explore whether combining the strengths of CNNs and RNNs could lead to better performance by using the benefits of both models.

Otherwise, each architecture used the same components of the baseline. For instance, a CTC loss function for training and a decoder for character sequence prediction was present for all architectures.

2.3 Training and Evaluation

Models were trained with a learning rate scheduler over 30 epochs, and performance was evaluated using the CER. I experimented with different hyperparameters, including hidden sizes, number of layers, and dropout rates to improve performance.

3 Results

3.1 Performance Comparison

Table 1 summarizes the CER achieved by each architecture in the test and validation sets. Note that the TDSLSTMEncoderModified architecture with the best hyperparameters is reported here:

Table 1: Architecture Performance Comparison

Architecture	Test CER	Validation CER	Minutes Running
Baseline TDSConv	28.7876	28.4448	54.31
TDSLSTMEncoder	27.1882	27.8245	52.43
TDSLSTMEncoderModified	21.6771	20.4697	88.92
TDSGRUEncoder	28.1608	27.1156	52.39
CNNRNNEncoder	74.8649	71.1342	52.42

Table 2 describes the test and validation CER of the TDSLSTMEncoderModified architecture with different parameters:

Table 2: TDSLSTMEncoderModified Hyperparameter Performances

Hidden Size	LSTM Layers	Dropout	Test CER	Validation CER	Minutes Running
64	4	0.2	37.3892	41.0058	60.72
128	4	0.2	28.4201	28.4448	63.96
256	4	0.2	24.3354	22.973	67.5
512	4	0.2	21.6771	20.4697	88.92
512	6	0.3	54.96	57.4657	124.5

When attempting to train the TDSLSTMEncoderModified architecture with 8 channels instead of 16, Google colab ran out of GPU resources. However, the validation CER in epoch 20 was 36.5086, which was worse than the same model with 16 channels at this point. However, the model could have been promising with more computational resources.

3.2 Loss Per Epoch

Figures 1-5 illustrate the loss over time in seconds during training. All models saw a significant decrease in loss immediately, and the loss continued to decrease with time, yet more slowly.

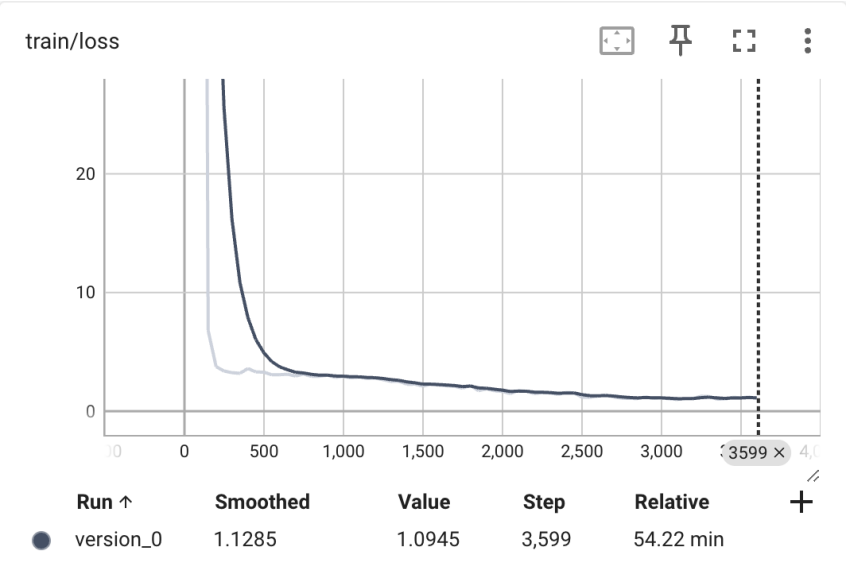

Figure 1: Baseline TDSConv Loss

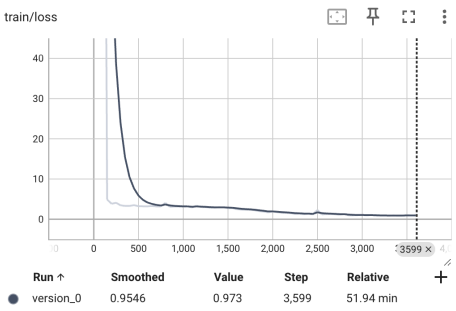

Figure 2: TDSLSTMEncoder Loss

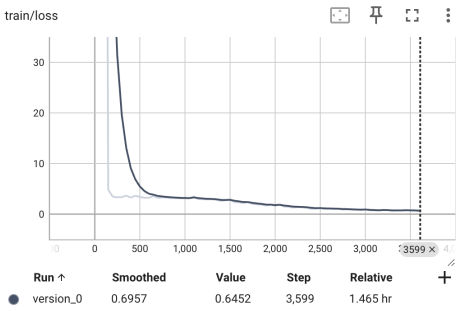

Figure 3: TDSLSTMEncoderModified Loss

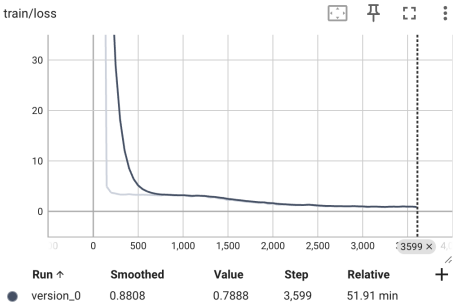

Figure 4: TDSGRUEncoder Loss

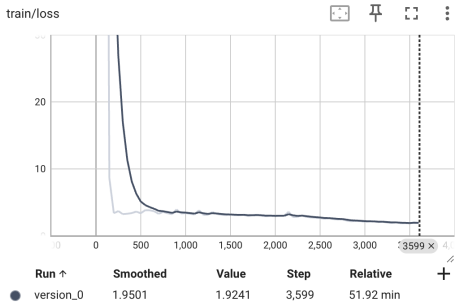

Figure 5: CNNRNNEncoder Loss

3.3 Key Observations

The TDSLSTMEncoderModified architecture achieved the lowest CER on both the test (21.6771) and validation (20.4697) sets, outperforming the baseline and other architectures. However, the tradeoff is that it required nearly double the runtime compared to other models.

The recurrent architectures like LSTMs and GRUs generally outperformed the baseline TDSConv model, with the TDSLSTMEncoderModified showing the most promise due to its hyperparameter tuning.

The CNNRNNEncoder performed poorly, with a test CER of 74.8649, suggesting that the hybrid approach may not appropriate for this task without further tuning. Alternatively, this architecture may learn slowly and needs more time to reach a reasonable performance.

Increasing the hidden size from 64 to 512 consistently improved performance, with the best results reached at 512. This suggests that larger hidden sizes better capture the complexity of sEMG signals, causing better decoding accuracy. At the same time, increasing the hidden size also increased the minutes to train, which may not be scalable.

A dropout rate of 0.2 was effective in preventing overfitting, as evidenced by the improved validation CER.

Increasing the number of LSTM layers from 4 to 6 with a higher dropout rate (0.3) degraded performance, likely due to over-regularization or increased model complexity. Moreover, the increased complexity may have needed more epochs or time to train to reach a performance similar to the best model.

All of the architectures demonstrate a loss that continues to decrease over time. Therefore, with more time and computational resources, all of the above model may be able to reach a better CER.

3.4 Impact of Electrode Channels

Decreasing the number of channels in the TDSLSTMEncoderModified architecture, such as reducing from 16 to 8 channels, can result in a noticeable loss of information captured during feature extraction and processing. The channels play a critical role in representing different aspects of the input data, allowing the model to learn diverse features. By reducing the number of channels, the model's capacity to encode these rich representations diminishes. This can lead to the model struggling to fully capture the complexity of the data, which in turn may hinder its ability to generalize effectively during validation or testing.

3.5 Computational Limitations

Throughout this project, there have been limitations due to the finite access of GPUs in Google Colab. With more time and computation, several of the models that are performing poorly in 30 epochs may be able to reach better CERs. For instance, Figure 6 shows the validation CER of the CNNRNN architecture during training. Only after the 2500 seconds does the model start making significant progress towards reducing the CER. With more time, the figure demonstrates that the model will be able to reach a much lower CER because the CER is not near plateauing.

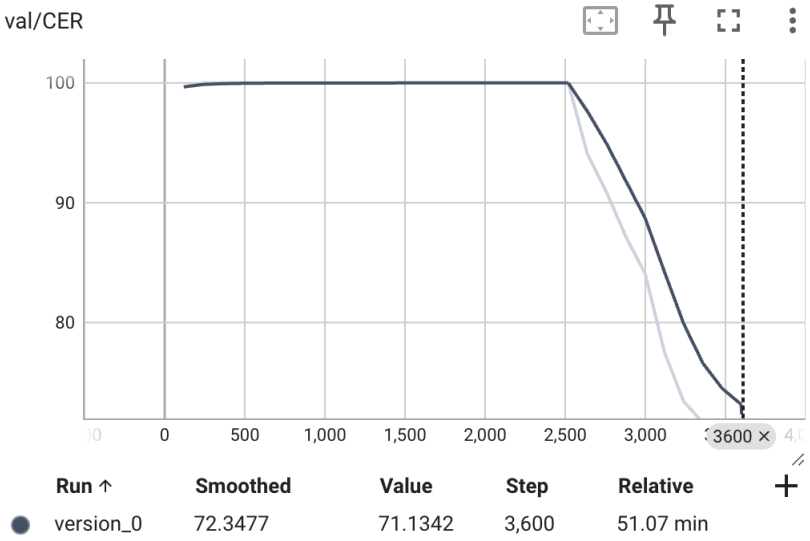

Figure 6: CNNRNNEncoder Validation CER

4 Discussion

4.1 Insights

The success of the TDSLSTMEncoderModified architecture highlights the importance of temporal modeling for sEMG signal decoding. The bidirectional LSTM, combined with dropout and layer normalization, effectively captures long-term dependencies and prevents overfitting at the same time. It is logical that models that are more appropriate for sequential data perform better than their counterparts because they are designed to find temporal patterns and predict future values based on past data.

The performance of the TDSLSTMEncoderModified architecture was highly sensitive to hyperparameters, particularly hidden size and dropout. Careful tuning is essential for achieving the best results. Specifically, increasing the hidden size continued to improve the performance of the model, despite increasing the time to train. This occurs because a larger hidden size makes the model more complex and able to learn more information from the data. However, this increased complexity takes longer

to train. With more computational resources, even larger hidden sizes can be tested until the model begins to overfit.

The poor performance of the CNNRNNEncoder suggests that hybrid architectures may require additional tuning or alternative designs to effectively combine spatial and temporal feature extraction. On the other hand, this model showed promise because the validation CER continued to decrease rapidly in the last few epochs. Therefore, this architecture may simply need more time to train to reach better performance.

4.2 Limitations

The TDSLSTMEncoderModified architecture, while effective, required significantly more runtime, which could be a limitation for real-time applications or large-scale training. Similarly, I was unable to train the TDSLSTMEncodeModified architecture with 8 channels to completion because of the computational limitations. With better GPU resources, these modifications can be evaluated more accurately.

4.3 Future Work

Exploring transformer-based models could further improve performance, given their success in sequential tasks.

Attempting data augmentation techniques such as time warping or noise injection could improve a model's robustness.

While the single-subject dataset was sufficient for this project, scaling to multiple subjects or the full dataset could provide more robust insights in the future. The purpose of this project was to improve the accuracy in a single patient. However, improving generalization can make the model for useful in applications.

References

Cha, J., et al. (2021). SWAD: Domain Generalization by Seeking Flat Minima. NeurIPS.

Graves, A., Fernandez, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. ICML.

Sivakumar, V., et al. (2024). emg2qwerty: A Large Dataset with Baselines for Touch Typing Using Surface Electromyography. arXiv.

*AI-generated code was used as a supplement for my starting point.