Emg2qwerty Spatially Conscious Architectures and Optimizations

Zach Berman
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024
`zachberman@ucla.edu`

Abstract

The models explored in this project aim to classify QWERTY characters from wrist sEMG signals. The report conducts a comparative analysis of model performance given different levels of training data, types of preprocessing methods, and varying model architectures. Explored high-level architectures include Convolutional Neural Networks (CNNs), CNN+LSTM (Long Short-Term Memory) Hybrids, and notably, special CNN modules that improve generalization by allowing the model to consider the spatial layout of the EMG data collection apparatus.

1 Introduction

The specific goal of this project was to minimize the Character Error Rate (CER) of a personalized typing prediction model, i.e., one trained and tested exclusively on a single individual's QWERTY typing data. Indeed, such a personalized approach is practically motivated by applications like assistive typing prediction for upper-limb amputees, who would have strong incentives to actively contribute data toward training their own accurate, customized models.

It is important to make some baseline assumptions before diving into the experiments. Compute was limited to about 100 Google Colab compute units for this project, meaning the amount of training data and epochs used had to be measured. These parameters were not chosen arbitrarily, though, as the report will go on to describe the steps taken to find appropriate amounts for both.

2 Methods

2.1 Training baselines

The baseline CNN model architecture and dataset used in this project were derived from the work by Sivakumar et al. (2024). Due to computational constraints discussed in Section 1, it was necessary to determine the minimal amount of training data and epochs required to achieve acceptable decoding performance. The dataset provided included 18 sessions of training data from the selected individual. One session was saved for validation and training, respectively. Out of the 16 remaining, three different amounts of sessions were tested on the baseline model, and their CERs were compared to find a minimal amount of sessions that still achieved good CER. For all subset tests, the CER was monitored across epochs to identify an optimal baseline epoch count.

2.2 Model architectures

Once these baseline hyperparameters were established, three different model architectures were developed and their CERs compared. For purposed of concision, I will only focus on each model's *trainable* layers;

2.2.1 Baseline Temporal CNN (5.3M)

After preprocessing, a rotation invariant multilayer perceptron (MultiBandRotationInvariantMLP) layer processes EMG signals while rotating the input by zero or one channels (electrodes), employing fully connected layers designed explicitly to extract meaningful features while maintaining invariance to electrode rotations around the wrist. Subsequently, the extracted features pass through a Time-Depth Separable Convolutional (TDSConvEncoder) that performs a convolvutional layer over the time dimension of the input, capturing temporal relationships within the EMG data, which is followed by a fully connected layer. Finally, a linear classification layer maps these temporal features directly to character logits, translating EMG-derived representations into corresponding text or blank characters. This model had 5.3 million trainable parameters.

2.2.2 Temporal CNN + Spatial CNN Hybrid (5.3M)

The only change in this architecture from that described in the previous section 2.2.1 is the addition of a "Circular-Spatial" Convolutional module. The motivation here comes from the biological fact that neighboring muscle fibers or groups often work together to product motion. This means that close-together electrodes should measure EMG activity more similarly to those of far apart ones. The previous architecture does not encourage these relationships to be learned, as the convolutions in the *TDSConvEncoder* are performed over the temporal dimension and not the channel (space) dimension. So logically, this new module does essentially the opposite, convolving over the channel dimension for each timestep. Additionally, instead of the input to this module being a flat array of electrodes 1-16, circular padding is employed, where the last electrodes are appended to the beginning and the first electrodes are appended to the end of the array. This creates a structure like [..., 14, 15, 16, 1, 2, 3, ..., 14, 15, 16, 1, 2, 3, ...] that preserves the circular arrangement of electrodes around the wrist. When the convolution kernel slides over this padded array, electrodes at the edges (like 1 and 16) can properly interact with their true physical neighbors, encouraging learning of spatial relationships that exist in the circular electrode array during training. These relationships are intuitively very high level and likely don't help predict the exact characters, just a general range, so this module is placed as the first trainable layer. This model had 5.3 million trainable parameters.

2.2.3 Spatial CNN + LSTM Hybrid (2.9M)

The only change in this architecture from that described in the previous section 2.2.2 is the substitution of the CNN encoder for an LSTM encoder. The LSTM encoder (*TDSLSTMEncoder*) processes temporal dependencies more effectively than the CNN one by maintaining memory of previous inputs through its recurrent structure. Unlike the CNN encoder which uses TDS convolutions with fixed receptive fields, the LSTM encoder employs bidirectional LSTM layers (with 128 hidden units across 4 layers in my implementation) that can capture long-range dependencies in both forward and backward directions. This (similarly to the CNN encoder) followed by a fully connected layer. This bidirectional nature allows the model to incorporate both past and future context when making predictions at each timestep, which is particularly valuable for sequential typing EMG data where temporal patterns may span variable lengths of time, and key-press motions may depend on both the previously pressed, and next-up keys. This model had 2.9 million trainable parameters, a significant reduction from the previous two models.

3 Results

3.1 Training baselines

The optimal epoch count chosen, where a lot of learning took place but started to plateau, was 30. Table 1 presents the CER after 30 epochs and the corresponding training time for different amounts of sessions used out of the 16 available. Notably, the model trained with 13 sessions achieved a substantial improvement in CER (33.888) compared to 11 sessions (58.266), demonstrating the effectiveness of additional training sessions. However, while increasing the session count from 13 to 16 led to a further reduction in CER (29.976), the improvement was relatively modest compared to the significant rise in training time from 30 to 50 minutes. Given this diminishing return in performance improvement versus the increased computational cost, 13 sessions was selected as the baseline.

# of Sessions	**11**	**13**	**16**
CER after 30 epochs	58.266	33.888	29.976
Training Time (mins)	21	30	52

Table 1: CER and Training Time for Different Session Counts

3.2 Model performances

Table 2 presents the CER after 30 epochs for the three different model architectures described in Section 2.2. The Baseline CNN model (2.2.1) achieved a CER of 33.888. Incorporating both temporal and spatial features in the Temporal CNN + Spatial CNN model (2.2.2) significantly improved performance, reducing the CER to 26.852, which represents a 20.76% improvement over the Baseline CNN. Further enhancement was observed with the Spatial CNN + LSTM model (2.2.3), which achieved the lowest CER of 24.388, yielding a 28.03% improvement over the Baseline CNN.

Model	**Base CNN**	**CNN (Temp. + Spat.)**	**Spatial CNN+LSTM**
CER after 30 epochs	33.888	26.852	24.388
Improvement over Baseline	-	20.76%	28.03%

Table 2: Model Performance Comparison: CER after 30 Epochs and Improvement over Baseline

4 Discussions

The experiments have clearly highlighted the benefits of integrating spatial and temporal modeling for sEMG typing prediction. While the inital, baseline temporal CNN provided reasonable performance, explicitly adding spatial awareness significantly boosted performance. The addition of the spatial convolutional module improved CER by approximately 21%. This improvement makes sense as the module aligns the model with physiological relationships between neighboring electrodes.

Replacing the temporal CNN encoder with an LSTM encoder improved performance even further, achieving the lowest CER (24.388) while also notably reducing parameter count from 5.3M to 2.9M. The bidirectional LSTM encoder was likely able to capture temporal dependencies more effectively and efficiently than the CNN, and given that typing inherently involves sequential patterns where preceding keystrokes strongly influence upcoming ones, the LSTM's capability to model these temporal relationships likely explains its performance gains despite having almost half as many parameters.

Additionally, identifying an optimal number of training sessions (13) proved important in balancing performance and computational efficiency, allowing me to conduct this research in an efficient manner, demonstrating the ever-important principle of evaluating the trade-offs between data volume and compute resources.

References

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. *arXiv preprint arXiv:2401.10988*, 2024.