ECE C147/C247 Baseline Project Writeup

Members: Zihan Jiang, Bleu Xu, Anping Tao, Loc Yee Monique Wong

1 Abstract

Abstract

In this *emg2qwerty* project, our team explored different model modifications to improve the Character Error Rate (CER) for a single subject. By integrating a Long Short-Term Memory (LSTM) layer into the baseline Connectionist Temporal Classification (CTC) model, we achieved significant performance improvements. Our best results reduced the CER to **15.5%** on the validation set and **16%** on the test set. This demonstrates the effectiveness of incorporating temporal modeling into sEMG-based keystroke prediction.

1.1 Monique Wong- Spatial Warping Augmentation

Motivation We investigated the effects of spatial warping augmentation on the performance of the emg2qwerty model. Spatial warping was introduced to simulate electrode misalignment and non-uniform muscle movements, aiming to improve model robustness. However, the impact of this augmentation on recognition accuracy and generalization remained uncertain.

Implementation A SpatialWarping class was created to apply nonlinear distortions by shifting electrode channels based on sinusoidal perturbations. This transformation was applied before spectrogram computation in the preprocessing pipeline. Initially, spatial warping was introduced with parameters shift_range $= 0.2$ and warp_strength $= 0.1$. The model was trained for 30 epochs, but it showed a significant increase in error rates. To better understand its effects, training was extended to 150 epochs while keeping the augmentation parameters unchanged. Although the loss decreased, some error metrics remained higher than the baseline. Subsequently, an attempt was made to refine the augmentation with shift_range $= 0.1$ and warp_strength $= 0.05$, but computational constraints prevented training for 150 epochs.

Results and Analysis The table below (Table 1) summarizes the results of the baseline model trained for 150 epochs and the augmented model with spatial warping, also trained for 150 epochs.

From the results, we observe that CER increased from 19.14 to 23.81 in validation and from 22.04 to 24.59 in testing. DER also showed a sharp increase from 1.22 to 5.65 in validation and from 1.56 to 4.52 in testing. While IER improved, SER worsened. Interestingly, the loss slightly decreased in the augmented model, suggesting that the model converged but did not generalize well.

Conclusion Spatial warping augmentation did not necessarily improve performance under the tested parameters. Despite a reduction in loss, error rates increased. The model struggled with the added transformations. Future work should explore adjusting the augmentation parameters further and training for more epochs to assess if the model can better adapt. Additionally, combining spatial warping with other augmentations or regularization techniques might yield improved generalization.

Metric	Baseline (150 epochs)	Augmentation (150 epochs)
val/CER	19.14	23.81
val/DER	1.22	5.65
val/IER	6.31	4.03
val/SER	11.61	14.13
val/loss	1.03	0.99
test/CER	22.04	24.59
test/DER	1.56	4.52
test/IER	6.27	2.68
test/SER	14.22	17.40
test/loss	1.10	0.82

Table 1: Comparison of Baseline and Augmented Model Performance (150 epochs)

1.2 Bleu Xu- Replace the batch normalization with a running time normalization

Motivation Batch normalization (BatchNorm2d) is widely used to stabilize training and improve convergence speed by normalizing feature distributions across a batch. However, it has limitations when dealing with smaller batch sizes or domain adaptation scenarios, as it relies on batch statistics. Instance Normalization (InstanceNorm2d) is an alternative that normalizes features across spatial dimensions for each individual sample, making it potentially more robust to input variations.

Implementation In this experiment, we replaced all occurrences of `BatchNorm2d` in the baseline model with `InstanceNorm2d`. The primary motivation behind this change was to improve generalization and robustness to domain shifts in electromyography (sEMG) signals. The modification was implemented as follows:

```
self.norm = nn.InstanceNorm2d(num_features)
```

instead of:

```
self.norm = nn.BatchNorm2d(num_features)
```

This ensures that normalization is performed independently for each sample, rather than relying on batch statistics.

Results and Comparison The results of this modification are summarized in Table 2.

Table 2: Effect of Replacing BatchNorm2d with InstanceNorm2d on CER

Normalization Type	Validation CER (%)	Test CER (%)
BatchNorm2d (Baseline)	19.14	22.04
InstanceNorm2d (Modified)	18.83	21.20

Analysis The results indicate that replacing BatchNorm2d with InstanceNorm2d led to a reduction in CER, achieving 18.83% on the validation set and 21.20% on the test set, compared to 19.14% and 22.04%, respectively, with BatchNorm2d. This suggests that InstanceNorm2d is more effective for this task.

1.3 Zihan Jiang - Use 6 log-spaced frequency bins instead of the original 33 linear-spaced frequency bins

Motivation The original model used 33 linearly spaced frequency bins for feature extraction. We experimented with reducing the number of bins to 6 log-spaced frequency bins to condense spectral information and potentially improve efficiency. Log-spacing emphasizes lower frequencies, which often contain more meaningful signals in sEMG-based keystroke prediction.

Implementation The spectrogram transformation was modified to use 6 log-spaced bins instead of 33 linear-spaced bins. In addition, the modified feature representation was tested with CTC decoding to evaluate its impact on Character Error Rate (CER).

Result and Analysis With CTC decoding, the model was able to train successfully using the 6-bin spectrogram, achieving a CER comparable to the baseline. However, when integrating the LSTM layer, we observed that the reduced number of frequency bins limited the available spectral features, making it difficult for the LSTM to learn meaningful temporal dependencies.

This resulted in a higher CER compared to using 33 bins, as the model lacked sufficient frequency resolution to capture variations in the sEMG signals. To optimize final performance, we reverted to 33 frequency bins, which provided a more comprehensive feature set and led to better accuracy.

Conclusion Although 6 log-spaced frequency bins were sufficient for CTC decoding, they proved to be suboptimal for LSTM-based models, as the reduced feature representation hampered temporal learning. As a result, we retained the 33-bin configuration in the final model to achieve the best performance.

1.4 Anping Tao- Tried out different architecture of the model.

Motivation The baseline TDS model does not capture chronic dependencies, so we added RNN blocks and tested the performance.

Implementation We leveraged the Pytorch API of LSTM and attention block, as well as added them after the TDS block.

Result and Analysis After adding the LSTM block and with dropout enabled, the model improves about 4%.

Conclusion We tried applying batch norm and layer norm to the model. However, the experimental result was not as good as expected.

2 Introduction

The prediction of keystroke using surface electromyography (sEMG) signals is an important area in research, particularly for assistive technology applications. The *emg2qwerty* dataset provides a benchmark for this task, where the goal is to decode sEMG signals into QWERTY keystrokes while minimizing the Character Error Rate (CER).

In this project, our objective is to improve the baseline Connectionist Temporal Classification (CTC) model by exploring different modifications. Among the models tested, the addition of a Long Short-Term Memory (LSTM) layer yielded the best performance, reducing CER in both validation and test sets. This suggests that integrating recurrent architectures enhances temporal feature extraction in sEMG-based keystroke decoding.

3 Methods

3.1 Baseline Model

We adopted the original baseline model architecture and made some modifications.

3.2 Consideration According to The Baseline Model

In the original paper on the TDS model, it is indicated that the model treats every data point in the data set as a combination of different steps. TDS uses a convolutional-like strategy to transform the combination in some other way with the benefit of easy handling in the downstream work.

The input and output of this model are sequences. It is natural to develop an RNN model. Although the TDS paper mentioned that it can save computations and give performances similar to those of

RNN models, we believe that it is possible to improve the model by attaching an RNN model after the TDS block, especially with an RNN with memory. Therefore, we tried LSTM and attention.

The attention model was successfully trained. However, it blew up the memory footprint during testing. Due to unfamiliarity, unfortunately we did not solve this problem. Therefore, we just collected data with LSTM models.

For the LSTM model, we also tried to attach batch norm and layer norm layer to see whether the generalization CER can be improved. The details are listed in the next section.

3.3 Model Modifications

Each team member introduced a different modification to the baseline:

Table 3: Model Modifications and Their Impact

Modification	**Description**
	Data augmentation to improve robustness
6 log-spaced frequency bins	Using log-spaced frequency bins to reduce input dimension
Add an LSTM block after the TDS block	Using a deeper RNN to catch the time-wise dependency for sequence
Add batch norm after TDS	Using batch norm, trying to get a better generalization for the model
Add layer norm after TDS	Using layer norm, trying to get a better generalization for the model
Add batch norm after LSTM block	Using batch norm, trying to get a better generalization for the model
Add layer norm after LSTM block	Using layer norm, trying to get a better generalization for the model
Add dropout layer between blocks	Using dropout strategy to get a better generalization for the model
Add an FC layer to reduce dimension for RNN	Using fewer parameters to avoid overfitting
Running Time Normalization	Replaced BatchNorm for better stability

4 Results

To assess the impact of our model modifications, we evaluated the Character Error Rate (CER) on both the validation and test sets. Due to the variability in the collected data, we present a subset of the results, as the others were less representative. The results are summarized in Table 4.

Table 4: CER Results for Different Model Configurations

Model	**Validation CER**	**Test CER**
Baseline Model	19.14%	22.04%
Baseline Model + 6 log-spaced frequency bins	17.52%	21.16%
Baseline + LSTM layer	17.30%	20.66%
Baseline + dim reduction(128) + LSTM layer + dropout	16.68%	17.64%
Baseline + LSTM layer + dropout	14.18%	14.98%

As shown in the results, adding an LSTM layer significantly reduced CER compared to the baseline model.

5 Discussion

Our results indicate that replacing BatchNorm2d with InstanceNorm2d led to a reduction in the Character Error Rate (CER), suggesting improved generalization. This improvement may stem from the fact that instance-wise normalization eliminates batch dependency, allowing the model to generalize better across different recording conditions. Furthermore, since sEMG signals exhibit high variability between subjects, InstanceNorm2d can help mitigate this by ensuring more consistent feature normalization between samples. In contrast, BatchNorm2d relies on running statistics, which can introduce inconsistencies when batch sizes fluctuate, potentially making it less stable for certain types of data.

By experiment, we found that adding batch norm after the original TDS block and RNN block does not help the model get a better generalization. This is probably due to the small batch size (or training set data), which does not help with generalization.

In a separate experiment, we also evaluated the effect of reducing the input feature dimension by using 6 log-spaced frequency bins instead of the original 33 linear-spaced bins. While this modification produced a modest improvement in CER compared to the baseline when using CTC decoding, the reduced spectral resolution proved to be a bottleneck when combined with the LSTM layer. The LSTM could not effectively capture temporal dependencies with the limited frequency information, resulting in a higher CER. Thus, we decided not to include this change in the final model.

The training loss and CER are still much lower than the validation and testing set, by the Occam Razor principle, we tried to reduce the parameters used in LSTM. However, we cannot improve the validation and testing CER. We believe the primary limitation lies not in the model itself but in the data preprocessing and inherent randomness. This presents a valuable area for future research.

Future work could explore Group Normalization (GroupNorm) as a middle ground between batch and instance normalization, or Layer Normalization, which operates across feature dimensions rather than spatial dimensions. Further hyperparameter tuning, such as adjusting learning rate and weight decay, may reveal additional performance gains. Additionally, testing on larger and more diverse datasets would help determine whether the observed improvements generalize across different subjects and recording conditions.

6 References

Hochreiter, S., and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. https://deeplearning.cs.cmu.edu/S23/document/readings/LSTM.pdf

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks*. In Proceedings of the 23rd International Conference on Machine Learning (ICML) (pp. 369–376). https://www.cs.toronto.edu/ graves/icml_2006.pdf

A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions," in Advances in Neural Information Processing Systems (NeurIPS), 2024. [Online]. Available: https://arxiv.org/pdf/1904.02619.