Keystroke Prediction from Electromyography Signals

Asish Lowrance
UCLA
asishll@g.ucla.edu

Fatemeh Arkannezhad
UCLA
fatemeharkan@ucla.edu

Melis Fidansoy
UCLA
melis.fidansoy.ucla@gmail.com

Varun Agarwal
UCLA
varunagarwal1@g.ucla.edu

Abstract

This work is an analysis of keystroke prediction from surface electromyography (sEMG) signals for enabling keyboardless human-computer interaction by choosing models that minimize the Character Error Rate (CER). Our experimental setup is divided into three distinct test stages: single-user, multi-user, and unseen user testing. In the single-user scenario, we compare baseline CNN models with recurrent models (RNN, LSTM, GRU) and CNN-RNN hybrid models, quantifying the impacts of adaptations such as resampling, short-time Fourier transform (STFT) parameters, and augmentation methods like band rotation and temporal jitter. GRU + CNN show cased the best CER performance of 14.15 for a single user. Though optimized architectures achieve promising CER performance on single-subject cases, tests also reveal challenges in cross-domain generalizability to multi-user and novel-user data.

[1]

1 Introduction

Keystroke prediction using surface electromyography (sEMG) signals is useful to enable human-computer interaction where a keyboardless system can be envisioned. sEMG signals can be collected from wrist sensors and can be decoded to perform character prediction. This project tries to build a model to minimize the Character Error Rate, or CER [4]. This metric compares a target sequence with a predicted sequence and calculates the difference between the two. In effect, this acts as a loss metric in the form of a percentage difference from the expected result.

This work explores the performance benchmarks for three separate use-cases: single-user, multi-user, and unseen users. We first investigated and maximized the CER performance for a single user by exploring various architectures, preprocessing (including sampling rate) and data augmentation techniques, variable data size and channel counts. Minimizing CER for a single test subject will help us find the best model that is precise for a given user, where retraining after deployment is acceptable. The best performing single-user model was then evaluated for multiple users. Finally, we evaluated the model on unseen user. A generalized model which works for multiple-users or unseen users generates the best model for deployment without further tuning.

[1]The final architecture's code is submitted to the Bruinlearn. To see the codes for whole process/Trial and error please refer to the github link: "https://github.com/ananymel/Project.git" or https://github.com/ananymel/Project

2 Methods

2.1 Single User Testing

We investigate various architectures to minimize CER for a single user (89335547). However, there are a plethora of combinations for different architectures and it's corresponding hyper parameters. Investigating all combinations is practically impossible and provides no intuition. Therefore, we adopt the following structured methodology. We first use the baseline CNN architecture and change only one aspect/variable – for example, use RNN instead of CNN keeping everything else the same, or use minimal band rotation keeping everything else the same, etc. This enables us to understand and keep track of the effect of each architectural and hyperparameter change for accurate comparison.

As part of preprocessing techniques, we explore resampling as well as STFT with and without Uniform Sampling. We investigate the effects of data augmentation such as band rotation, temporal jitter, temporal, and frequency masking. For model building, we explore architectures such as RNN, GRU, and LSTM, Transformers, CNN, FC Linear Layer, etc. Many of these architectures are aimed at extracting the temporal features from the EMG data, while others are focused more towards the spatial features. In addition, we dive into the implementations of the CTC loss function and language model and their effect on the CER. We also investigate the effect of data size and number of channels on CER.

After we get a quantitative idea of the effect of different data preprocessing techniques, augmentation, architectures, etc., we choose the best combinations of these to maximize CER performance.

2.2 Multi User Testing

After obtaining the best model for single user testing, we train the model on multiple users (89335547, 9456349, 75228805) both with and without data augmentation. We report the best model by comparing the CERs.

2.3 Unseen User Testing

The best single user model is used to test the performance of unseen user data. The users 89335547, 9456349 are used to train and validate the model, and the user 75228805 is used to test the model. Like with multi-user testing, we investigate CER performance with and without data augmentation and fine-tuning. We report the best model by comparing the CERs.

3 Results

3.1 Single User Testing

Tables 1, 2, 3, 4 and 5 represent various tests performed to ascertain the best-architecture, preprocessing and augmentation techniques and other hyper parameters for the obtaining good CER. The rows marked in tan represent the baseline parameter values.The rows marked in red represent the worst CER performance of that category. The rows marked in green represent the best CER performance for that category. Most of the comparison was done for 30 Epochs due to limited compute resources. Some of the comparisons, especially for the architectures, was done for 150 Epochs for better final accuracy. Table 1 represents different preprocessing and data-augmentation techniques and the resultant CER. Table 2 shows the effect of CER by changing number of EMG channels and training with less data. Table 3 represents the best CNN Kernel/Filter size. Table 4 represents the CTC Decoder hyper parameters for lowest CER. The table 5 representing various architectures showcase the best observed hyper parameter for lowest CER. Using this table as reference, the final best architectures are shown in Table 6.

Table 1: Pre-Processing and Data Augmentation Techniques (30 Epochs)

Pre-Processing and Data Augmentation Technique	Hyper Parameter Change	CER Validation	CER Testing
Resampling	1 KHz	83.96	78.4
	1.2 KHz	52.59	48.28
	1.5 KHz	36.68	30.06
	1.7 KHz	32.03	29.06
	2 KHz	29.35	29.19
	2.5 KHz	42.84	41.92
Band Rotation	0	27.11	27.68
	1	29.35	29.19
	2	33.49	33.69
	3	40.82	38.18
Temporal Jitter	0	29.08	28.85
	60	27.97	28.74
	120	29.35	29.19
	240	31.76	31.18
STFT	32	31.9	31.3
	64	29.35	29.19
	128	26.18	27.85
	256	29.17	29.37
STFT w/ Non-Uniform Sampling	64 bins (0-500 Hz 80%, 500-1000 Hz 20%)	37.97	37.85
	128 bins (0-500 Hz 70%, 500-1000 Hz 30%)	25.09	26.92
	128 bins (0-500 Hz 60%, 500-1000 Hz 40%)	25.29	26.19
	128 bins (0-200 Hz 80%, 200-1000 Hz 20%)	27.62	27.51
	128 bins (0-300 Hz 80%, 300-1000 Hz 20%)	25.12	25.91
	128 bins (0-400 Hz 80%, 400-1000 Hz 20%)	25.91	26.79
	128 bins (0-300 Hz 70%, 300-1000 Hz 30%)	24.81	26.32
Log Spectrogram Hop Length	8	91.8	92.5
	16	29.35	29.19
	32	87.21	82.14
Spectral Augmentation time_mask	0	26.29	26.79
	6	26.31	27.08
	12	28.11	28.98
	25	29.35	29.19
Spectral Augmentation freq_mask	0	26.47	28.24
	4	29.35	29.19
	8	31.85	31.23

Table 2: Number of Channels, Change Train-Validation-Test Split, Less Data

		30 epochs		150 epochs	
Name of Architecture	**Hyper Parameter**	**CER (Val)**	**CER (Test)**	**CER (Val)**	**CER (Test)**
Baseline	16-1-1	28.444	29.047	19.650	22.670
Decrease Number of Channels	Channel=8, In_features=264	45.746	44.931	25.831	28.917
Change Train-Validation-Test Split	14-2-2	39.133	35.851	22.364	21.510
Train with Less Data	14-1-1	33.429	33.542	20.071	23.233

Table 3: CNN Kernel Size (30 Epochs)

Name of Architecture Change	Hyper Parameter Chosen	CER Validation	CER Testing
Kernel	16	55.893	55.522
	20	47.563	43.764
	32	28.35	29.19
	40	34.537	30.602
	64	100	100

Table 4: Decoder Parameter (30 Epochs)

Decoder Parameter	Hyper Parameter Change	CER Validation	CER Testing
Beam Size	50	29.35	29.19
	100	28.62	29.09
Language Model Weight	2	29.35	29.19
	4	28.82	29.73
Maximum Label per Time Stamp	10	29.35	29.19
	20	27.95	28.83

Table 5: Architecture Comparison (30 Epochs and 150 Epochs)

		30 epochs		150 epochs	
Name of Architecture Change	**Hyper Parameter Chosen**	**CER (Val)**	**CER (Test)**	**CER (Val)**	**CER (Test)**
Baseline		28.444	29.047	19.65	22.67
Unidirectional LSTM	Number of Layers = 4 Hidden_states = 128	41.692	39.658	20.22	21.914
Bidirectional GRU	Number of Layers = 4 Hidden_states = 128	27.647	30.344	13.248	15.842
Bidirectional RNN	Number of Layers = 4 Hidden_states = 128	74.23	72.207	49.18	42.295
Bidirectional LSTM	Number of Layers = 4 Hidden_states = 128	27.625	28.766	14.067	15.842
Linear Layer+ReLU	Additional linear layer size = 128	25.21	25.719	18.631	21.158
GRU-bi+Linear Layer+relu	Number of Layers = 4 Hidden_states = 128	30.018	31.294	14.821	15.993
LSTM-bi+Linear Layer+ReLU	Number of Layers = 4 Hidden_states = 128	33.54	33.564	14.843	15.798
GRU+CNN	Number of Layers = 4 Hidden_states = 128	23.416	23.416	14.532	14.415
LSTM+CNN	Number of Layers = 2 Hidden_states = 256	24.280	23.838	14.577	16.144
GRU+CNN+Linear Layer+ReLU	Number of Layers = 4 Hidden_states = 256 (with dropout and normalization)	22.508	22.887	16.238	15.734
GRU+CNN+Linear Layer+ReLU	Number of Layers = 4 Hidden_states = 256 (without dropout and normalization)	100	100	100	100
LSTM+CNN+Linear Layer+ReLU	Number of Layers = 4 Hidden_states = 256 (with dropout and normalization)	21.777	21.893	15.108	15.128
LSTM+CNN+Linear Layer+ReLU	Number of Layers = 4 Hidden_states = 256 (without dropout and normalization)	100	100	100	100
Transformer Encoder	Number of Layers = 3 Hidden_states = 512	99.975	98.92	–	–
Transformer Encoder with Positional Encoding	Number of Layers = 3 Hidden_states = 512	100	99.92	–	–
Transformer + CNN	Number of Layers = 3 Hidden_states = 256	100	99.79	–	–

Table 6: Final Architecture (150 Epochs)

Architecture	Hyper Parameters	CER Validation	CER Testing
GRU + CNN + Linear Layer + ReLU + dropout + normalization	Number of layers = 4 Hidden size = 256 With preprocessing and no augmentaiton	13.469	15.582
GRU + CNN + Linear Layer + ReLU + normalization	Number of layers = 4 Hidden size = 256 With preprocessing and no augmentation	13.491	15.841
GRU + CNN	Number of layers = 4 Hidden size = 256 With preprocessing and no augmentation	12.959	15.885
GRU + CNN	Number of layers = 4 Hidden size = 256	14.532	14.415
GRU + CNN	Number of layers = 4 Hidden size = 256 No augmentation	12.073	16.771

3.2 Multi User Testing

After obtaining the best model for single-user, the model was trained using multiple-users and results are shown in Table 7.

Table 7: Multiple users (150 epochs): 9456349, 75228805, 89335547

Name of Architecture	Train-Validation-Test	CER (Val)	CER (Test)
GRU + CNN	16-1-1	14.532	14.415
3 users	16-3-3	17.59	19.718

3.3 Unseen User Testing

The single-user best model was also tested in unseen data and shown in Table 8.

Table 8: Unseen users (150 epochs): 9456349, 75228805, 89335547

Name of Architecture	Train-Validation-Test	CER (Val)	CER (Test)
GRU + CNN	16-1-1	14.532	14.415
2 train + 1 unseen	16-1-1	17.855	71.055

4 Discussion

4.1 Single User Testing

4.1.1 Preprocessing and Data Augmentation

Table 1 shows the various pre-processing and data-augmentation techniques that were performed.

The EMG data was initially sampled at 2 KHz. EMG data is band-limited to 500 Hz and most of the power is concentrated under 300 Hz [5]. Therefore, it made sense to resample the data to lower frequencies to capture most of the EMG data and getting rid of the high frequency noise. From the table, it was observed that CER is minimal for native sampling rate of 2 KHz, which is contradictory to our intial assumption. This is likely because the data has minimal high frequency noise and valuable information about the EMG is also present between 500-1000 Hz although lower in power.

The STFT is used to obtain the spectrogram of the EMG sequence. Increasing the number of frequency bins, in theory, should provide better resolution at the cost of more compute power. This prediction was true for 128 frequency bins. For 256 bins, CER performance dropped. This indicates that we picked up too many frequency details which prevents generalization.

The STFT's frequency bins are sampled uniformly. Since most of the data is concentrated under 300 Hz, it would make sense to use more number of bins between 0-300 Hz and less number of bins from 300 - 1000 Hz. From the table it was observed that allocating 80% of 128 bins to frequencies below 300 Hz and 20% of 128 bins to frequencies above 300 Hz provided the best performance.

The log spectrogram's frame rate or hop-length should be chosen such that it doesn't omit too much information, where the resolution will be poor, or has too much information, leading to poor generalization. Empirically, it was observed that a hop length of 16 provided the best frame rate for CER as shown in the Table.

Data augmentation techniques like Band Rotation, Temporal Jitter, Spectrogram Time Masking and Frequency Masking can improve regularization performance for multiple-users. However, we first aim to improve CER performance for a single user. Therefore these additional augmentations negatively impact performance. Therefore removing these augmentations improve performance for a single user as is clear in the Table.

4.1.2 Decoder Parameter

A decoder was used along with a language model to effectively decode characters from the EMG data. A larger beam size would make the decoding more accurate by parsing through more sequence candidates before choosing the right characters. Tests in Table 4 show that Beam size does increase performance very marginally. This indicates diminishing returns for increasing Beam Size above 50. A larger language model weight would weigh the decoded data more with a dictionary rather than depending solely on the raw data. It is observed that, even with a larger weight of 4, CER performance is comparable to baseline. This could indicate that Raw CER decoding performance is the bottle-neck for higher performance. The parameter Maximum label per time-steps searches around the decoding language space for all the most likely character combinations. A value of 20 would increase this search space and unsurprisingly enough, this improves CER performance because we have more possibilities to work with.

4.1.3 Number of Channels and Data Size

Table 2 show cases the effect of reducing the number of channels for the EMG data. As expected[1], less number of EMG channels, 16 as opposed to 32, worsens CER performance. Training on less data, say 14 files instead of 16, also worsens CER. Hence for our project, we take the full 32 channels and use the full 16 files for training.

4.1.4 Architecture

A CNN architecture was provided in the baseline architecture. We initially try to optimize the kernel/filter size. Smaller kernel size would carry more spatial information. Large Kernel size would have less spatial information and in the limit case would be similar to a full connected layer. Empirically, the baseline kernel size of 32 shows the best CER performance as shown in table 3.

We now explore other deep-learning architectures. Since the EMG data provided represents a time-series, RNN based architectures, including LSTM and GRU, are expected to provide good performance. A combination of CNN and RNN is expected to have good CER because it would capture good spatial and temporal features of EMG spectrogram. The attention layer of transformers may also zero-in better on the right spectral features and hence expected to provide good CER.

Table 5 showcases the results. Here is the summary of the findings.

1. Vanilla RNN architecture reported poor CER performance. This is because of poor-learning due to exploding/vanishing gradients.

2. Other RNN architectures such as LSTM and GRU performed very well on its own and especially when combined with CNN, confirming it's efficacy with time-series signals. It also gets rid of the vanishing gradient problem similar to ResNet [2], [3].

3. Bi-directional LSTM and GRU performed really well compared to uni-dimensional variants. This is because previous and future timesteps data are used, as opposed to only previous time-step for the uni-dimensional model. Bidirectional models provides more context for decoding, allowing increased accuracy when predicting keystrokes.

4. A simple but effective architectural change is by adding linear layer and ReLU activation after the baseline TDSCONV Encoder Block. It showed an increase in the performance due to a larger structural complexity. By adding ReLU activation, nonlinearity is also fused into the architecture (instead of directly mapping from the features to classes). This improved the ability to generalize better as is shown in the Table.

5. For larger networks, such as GRU+CNN+Linear Layer+ReLU, the networks on it's own showed poor learning due to vanishing gradients. By normalizing the activations of each batch(batch-norm), we saw the CER performance improved dramatically and the convergence time decreased.

6. Transformers were expected to show good performance. Different transformer architectures were tried. The CER performance is unexpectedly poor. Good optimization is required to get good CER performance. Due to time-constraints, this avenue was not explored thoroughly.

7. GRU + CNN shows the best performance of **14.415** Test CER

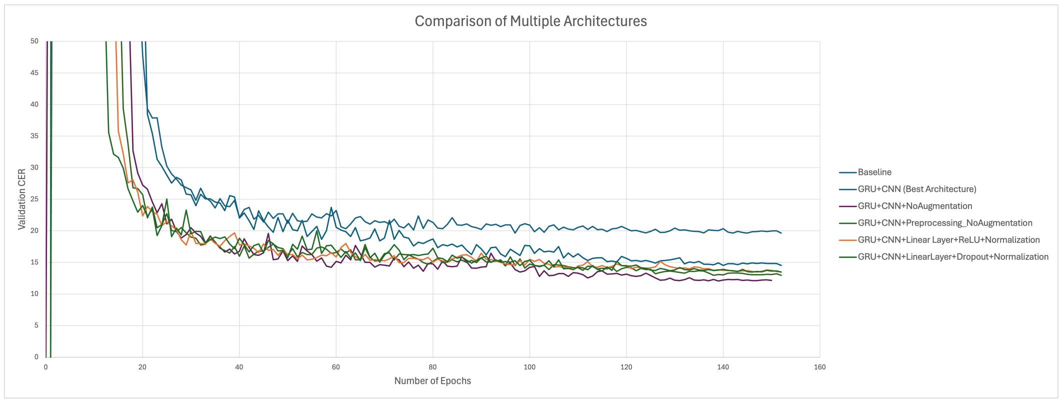

Figure 1: Validation CER for Final Architecture

4.1.5 Final Architecture Run

The best pre-processing and data augmentation techniques was identified in the previous discussion. The best architectures were also identified. Now we combine both of these to arrive at a final architecture as shown in Table 6. The Validation CER for Final Architecture is shown in Figure 1

1. Suprisingly, pre-processing techniques and no augmentation implemented along with the architecture degraded test CER performance. This seems to indicate that with a more-sophisticated model, the right features are learnt well even though pre-processing is absent.
2. Lack of data augmentation specifically also hurt the CER performance because of less regularization effect, since CER decreased from 14.415 to 16.771 for GRU+CNN
3. The best model for single user testing is GRU + CNN with a Test CER of **14.415**

4.2 Multi User Testing

The best model was trained with 3 users and the results are shown in Table 7. The Test CER got worse from 14.415 to 19.718. Interestingly. this number is still better than the baseline architecture.

4.3 Unseen User Testing

The best model was trained with 2 users and tested on a unseen user and results are shown in Table 8. The test CER is very large. This suggests more data-augmentation techniques are required for regularization, if the model is to be extended for more users. Training on larger data set with diverse users will also help improve the model generalizability.

4.4 Summary

In the overall task of building a model to precisely forecast keystrokes from the EMG signals, there are several variables to be managed and adjusted wherein accuracy must be preserved, generalizability of the model is maintained, and computationally it is sensible. We show that by using data pre-processing and augmentation strategically, there can be a significant impact on the degree of generalizability of a model through the introduction of variance into the dataset. We found that selecting good-quality hybrid architectures are more likely to outperform simpler models. And we highlighted the model for a general case for one user versus multiple users, and using these models on new users. Based on our large-scale testing, we were able to significantly improve the baseline EMG-based keystroke prediction CER to 14.415 for a single-user. These results guide the larger goal of developing long-term neural interfaces for assistive technology, where detailed decoding of muscle activity would restore communication and control for motor-disabled individuals.

References

[1] EITCA. What is the meaning of number of input channels, the 1st parameter of nn.conv2d? `https://eitca.org/artificial-intelligence/eitc-ai-dlpp-deep-learning-with-python-and-pytorch/convolution-neural-network-cnn/training-convnet/what-is-the-meaning-of-number-of-input-channels-the-1st-parameter-of-nn-conv2d/`, 2023. Accessed: March 13, 2025.

[2] Jonathan C. Kao. Lecture 15: Rnns + object detection and segmentation. Lecture notes, ECE 247, University of California, Los Angeles, 2025. pp. 1–39.

[3] Jonathan C. Kao. Lecture 15: Rnns + object detection and segmentation. Lecture notes, ECE 247, University of California, Los Angeles, 2025. pp. 12–23.

[4] 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.

[5] D. Xiong, D. Zhang, X. Zhao, and Y. Zhao. Deep learning for emg-based human-machine interaction: A review. *IEEE/CAA Journal of Automatica Sinica*, 8(3):512–533, March 2021.