# Comparing Encoder Architectures for Electromyography Classification

**Andrew Chen, Travis Ha, Edi Zhang, Sophie Zhu**
Department of Electrical and Computer Engineering
University of California–Los Angeles
Los Angeles, CA 90024

## Abstract

Electromyography (EMG) signals can be used to predict keystrokes and can serve as a non-invasive interface for people with paralysis or amputations. Various neural network architectures, namely Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM)[2], and Gated Recurrent Units (GRU)[3] were used to assess accuracy, which was measured by Character Error Loss (CER). The best performing model was then fine-tuned with data augmentation/preprocessing methods such as adding Gaussian noise and lower sampling rates. The CNN + GRU hybrid with Gaussian noise was found to have the best test/CER.

## Introduction

Multiple combinations of model architecture and data preprocessing/augmentation were tested to find a model that best minimized test/CER in a single subject. A lower test/CER indicates better generalizability of the model on unseen data. A CNN encoder served as the baseline model from which we attempted to optimize. We wanted to test architectures that would best capture the fine-grain features in the high-frequency data as well as make the necessary long-term correlations that are present in time-series data such as EMG signals. Data augmentation/preprocessing were then performed on the best architecture to fine-tune the model to be more generalizable and robust.

## Methods

The baseline model involves EMG signals collected through 16-channel wristbands at 2 kHz per hand, with a random rotation for each hand. These signals were also collected relative to key-logging times. The signal features were temporally downsampled using a log spectrogram with 33 bins from 0 to 1000 Hz to avoid generating 2 kHz outputs. A baseline stride of 16 was used to further reduce resolution to 125 Hz, and SpecAugment was applied to mask frequencies and times. Since log spectrograms can have a wide value range of signal strength and noise level, batch normalization was then applied on the log spectrograms to normalize each electrode's input distribution.Next, a "Rotation Invariant MLP" was applied by averaging the linear transformation followed by a ReLU activation of the input with two of its rotated copies (one left and one right), which outputs 384 features. While this method does not technically achieve rotation invariance, it is shown to be empirically helpful.

Connectionist Temporal Classification (CTC) loss was used instead of Cross Entropy (CE) loss to prioritize correct sequence decoding rather than accurate timing. It is designed to handle unknown timings of discrete events and functions by maximizing the probability of all "valid" alignments. Sequences augmented with a blank token that can be "collapsed" to correct sequences are considered "valid". In prior testing, CTC loss empirically performed better than CE loss, so it was thus selected for our model.[1]

Following the "Rotation Invariant" MLP, the 384 features from left and right signals were added together for a total of 768 features, and Time-Depth Separable (TDS) Convolution Blocks were applied. Here, we experimented with different architectures before finally applying the linear character decoding layer.

Five encoder architectures were trained on 40 epochs:

- CNN (baseline)
- LSTM
- GRU
- CNN + LSTM
- CNN + GRU

LSTM networks retain information over extended sequences, using input, forget, and output gates to control information flow. This memory allows for better processing of sequential data. GRU, a simplified LSTM, combines the forget and input gates into an "update" gate and is often more computationally efficient than LSTM despite achieving similar empirical results.

Our final implementation of both LSTM and GRU encoders received 128 input features, 128 hidden layers, and four recurrent layers. These LSTM/GRU layers were then passed through a fully connected block and a linear output layer.

Each of the four recurrent or hybrid models, as with the baseline, was trained and tested through 40 epochs due to computing restraints. The Test CER and Validation CER were recorded for all models. The best-performing model on Test CER was selected, and the following methods of augmentation/preprocessing were used to fine-tune the model:

- 1kHz preprocessing
- 0.01-Scale Gaussian Noise
- 0.01-Scale Gaussian Noise + 1kHz preprocessing
- 0.01-Scale Gaussian Noise + 512 RandomCrop
- 0.01-Scale Gaussian Noise + 30Hz FrequencySmoothing

Random noise following a Gaussian distribution with 0.01-scale was added to the training data at the transform stage to improve the robustness of the model.

By increasing the hop_length parameter in log_spectrogram.yaml and the LogSpectrogram class in transforms.py from 16 to 32, fewer samples were taken to decrease model complexity and focus on more relevant frequency ranges. The window_length in tds_conc_ctc.yaml was also halved from 8000 to 4000 to maintain the 4-second context window.

Random Crop, as tested in our implementation, randomly selects a continuous segment of size 512 from the input data (tensors smaller than this size were ignored). As a method of data augmentation, it reduces effects of position on the learning of features.

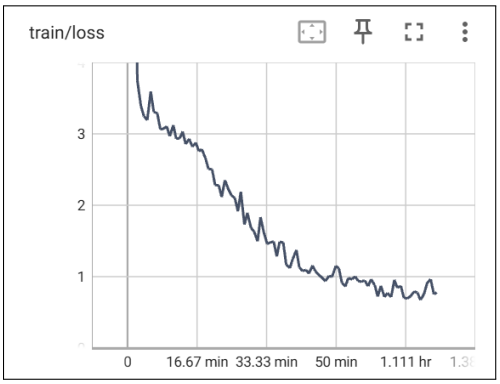
Frequency Smoothing was implemented as low-pass filtering to reduce noise. As the baseline data was 125 Hz, the frequency domain would represent frequencies up to 62.5 Hz (the Nyquist Frequency). Using a parameter of 30 smoothes variations in lower-frequency components. Information up to 30Hz is preserved, and higher frequencies are "smoothed" and altered.

## Results

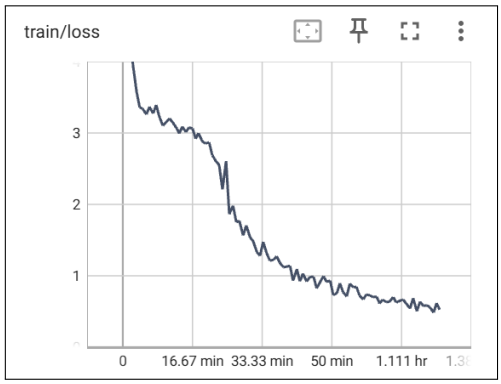
The aforementioned architectures that were used in the TDS Convolution Block yielded the following results:

| Model | Parameters | Test/CER | Val/CER |
|---|---|---|---|
| CNN (baseline) | 5.3M | 24.59 | 21.67 |
| LSTM | 2.9M | 22.48 | 21.93 |
| GRU | 2.4M | 22.95 | 22.73 |
| CNN + LSTM | 7.7M | 20.53 | 18.74 |
| CNN + GRU | 7.2M | 19.84 | 19.89 |

While the LSTM and GRU encoders were significantly more compact and had a lower test/CER compared to the Baseline CNN model, the combination of the CNN + GRU model ultimately had the lowest test/CER and the second-lowest val/CER, which best balances generalizability and reduces overfitting.




CNN Baseline training loss



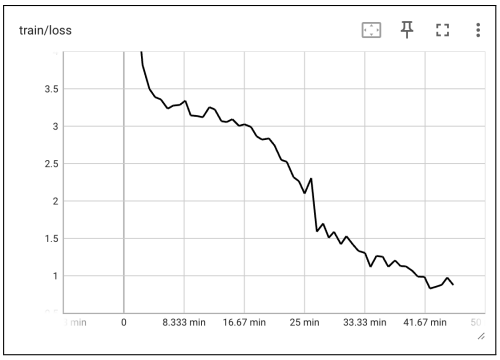

CNN + GRU training loss

While both models achieved similar results up until around 40 minutes of training, past 40 minutes, the CNN + GRU model experienced a more significant and consistent decrease in training loss.

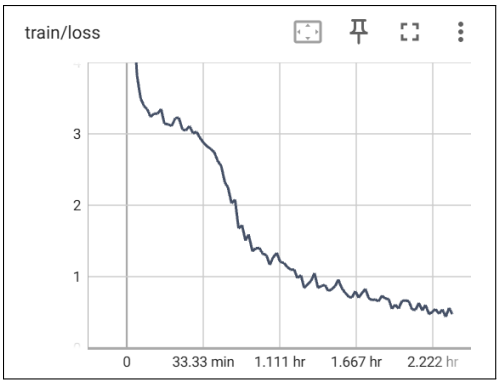
The CNN + GRU model, which performed the best, was thus selected for fine-tuning, which then yielded the following results:

| Model | Parameters | Test/CER | Val/CER |
|---|---|---|---|
| CNN + GRU | 7.2M | 19.84 | 19.89 |
| CNN + GRU + 5 Frequency Masks | 7.2M | 20.48 | 20.71 |
| CNN + GRU + 1kHz | 7.2M | 23.34 | 30.15 |
| CNN + GRU + Gaussian noise | 7.2M | 19.45 | 19.54 |
| CNN + GRU + Gaussian noise + 1kHz | 7.2M | 23.75 | 29.82 |
| CNN + GRU + Gaussian noise + RandCrop | 7.2M | 23.04 | 25.45 |
| CNN + GRU + Gaussian noise + RandCrop + FreqSmoothing | 7.2M | 33.20 | 33.91 |

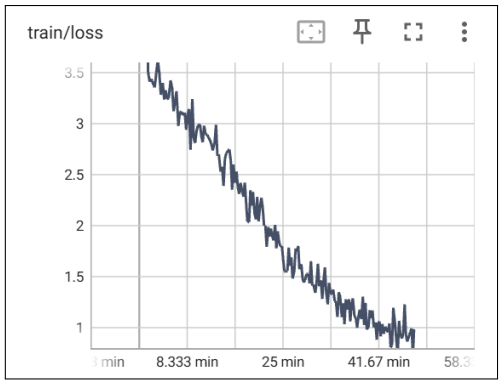
Adding Frequency Smoothing resulted in the worst performance, with a higher Val/CER and Test/CER. Downsampling the data into 1kHz led to worse Test/CER performance and a high Val/CER, which indicates significant overfitting. Conversely, Random Cropping saw a better Test/CER than Val/CER, indicating that more epochs may be needed to further reduce error. Finally, adding only Gaussian Noise to the CNN + GRU model saw a successful $20.9\%$ decrease in Test/CER over the baseline CNN model.




CNN + GRU + 5 Frequency Masks

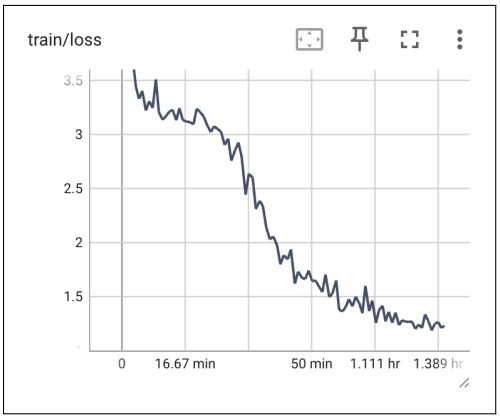



CNN + GRU + Gaussian noise training loss

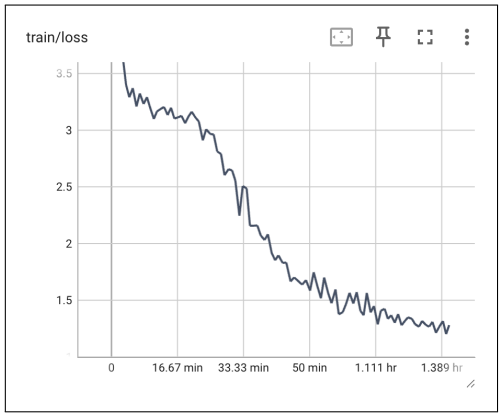



CNN + GRU + 1kHz training loss

Preprocessing with 1kHz is faster in training but has worse overall accuracy compared to only using the baseline preprocessing techniques. Additionally, it shows more oscillations in training loss. Using five frequency masks reduced both training time and training loss.




CNN + GRU + Gaussian noise + RandCrop training loss




CNN + GRU + Gaussian noise + RandCrop + FreqSmoothing training loss

Both Random Cropping and Frequency Smoothing resulted in similar training loss patterns, with the addition of Frequency Smoothing displaying a slightly smoother graph overall.

| Runningstage.validating metric | DataLoader 0 |
|---|---|
| val/CER | 19.89366340637207 |
| val/DER | 1.7058041095733643 |
| val/IER | 4.962339401245117 |
| val/SER | 13.225520133972168 |
| val/loss | 0.6252787709236145 |

LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing DataLoader 0: 100% 1/1 [00:04<00:00, 4.11s/it]

| Runningstage.testing metric | DataLoader 0 |
|---|---|
| test/CER | 19.840068817138672 |
| test/DER | 2.009941577911377 |
| test/IER | 3.090555429458618 |
| test/SER | 14.739572525024414 |
| test/loss | 0.5988753437995911 |

The CNN + GRU was the best base model with test/CER at 19.84

| Runningstage.validating metric | DataLoader 0 |
|---|---|
| val/CER | 19.53921127319336 |
| val/DER | 1.7279574871063232 |
| val/IER | 4.585733413696289 |
| val/SER | 13.225520133972168 |
| val/loss | 0.6435275077819824 |

LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing DataLoader 0: 100% 1/1 [00:03<00:00, 3.96s/it]

| Runningstage.testing metric | DataLoader 0 |
|---|---|
| test/CER | 19.451047897338867 |
| test/DER | 2.312513589859009 |
| test/IER | 2.7879836559295654 |
| test/SER | 14.350550651550293 |
| test/loss | 0.6028895974159241 |

Fine-tuning the CNN + GRU model with Gaussian had the lowest test/CER at 19.45

Overall, the best fine-tuning method slightly improved generalizability and reduced overfitting.

## Discussion

It can be observed that CNN + recurrent model hybrids perform better than the standalone CNN or LSTM/GRU models, as they complement each other's weaknesses. In particular, CNNs are great at capturing local spatial features and patterns, such as muscle activations, which might be harder for RNNs such as GRUs and LSTMs to capture directly.

On the other hand, GRUs and LSTMs excel at handling sequential patterns and capturing long-term correlations, especially in time-series data, such as EMG signals. This allows these architectures to form long-term dependencies crucial to accurate character predictions.

By combining both the CNN and LSTM/GRU architectures, the hybrid models provided both feature richness (from the CNN) as well as sequence modeling (from the LSTM/GRU), leading to more accurate and generalizable models.

After finding a model that performed considerably better than the baseline model, we wanted to look at possible data augmentation/preprocessing methods that could lower the test/CER error rate.

We decided to test several methods of data augmentation, including adding random Gaussian noise with scale 0.01 to the training data and using random cropping in the time dimension with a size of 512. For data preprocessing, we attempted to lower the sampling rate from 2kHz to 1kHz, as well as using frequency smoothing as a low-pass filter with a 30Hz cutoff.

Increasing the number of frequency masks from 3 to 5 was intended to improve the model's robustness by encouraging it to generalize across a wider range of spectral variations. In theory, this should prevent overfitting by forcing the model to learn features that are not reliant on specific frequency bands. However, our results showed a slight increase in test CER from 19.8 to 20, suggesting that additional masking may have inadvertently removed too much useful frequency information. While frequency masking helps simulate missing or noisy spectral components, excessive masking can degrade the quality of learned representations, leading to a minor drop in performance. This trade-off highlights the importance of carefully tuning augmentation parameters to balance regularization and information retention.

Adding Gaussian noise, in theory, acts as regularization, preventing the model from overfitting due to high-frequency variations or irrelevant details within the training data. Similarly, using random cropping encourages the model to learn more generalizable features, as it encourages learning from more portions of the image. Given the current Test/CER and Val/CER with 40 epochs, it seems like the model with random cropping is performing well, though it may result in a more robust model if given more time to train.

Lowering the sampling rate to 1kHz, in theory, would also reduce overfitting, since the lower rate would reduce the amount of data points, forcing the model to generalize with coarser data.

However, our experiment showed opposite results. While implementing frequency smoothing aims to reduce noise, there is a possibility of removing too many high-frequency features and losing critical information, which may have been the case in our testing.

Our results showed that only the CNN + GRU + Gaussian noise model performed better than the CNN + GRU hybrid model, with the CNN + GRU + 1kHz and the CNN + GRU + Gaussian noise + random cropping + frequency smoothing model performing substantially worse than the baseline.

We believe that adding the Gaussian did have a regularizing effect, forcing the model to focus more on the high-level features rather than overfitting on minor variations in the data. Given that the CNN acted as a feature extractor, training on data injected with small noise allowed it to learn invariant features by ignoring small perturbations. Due to GRUs nature of handling sequential data, they are able to learn smoother representations over time and are more resistant to small noise-induced changes.

However, we believe that the lowering of the sampling rate to 1kHz and frequency smoothing resulted in the loss of high-frequency features. For example, fast muscle activation changes could have been affected due to the sample rate lowering, which could have had severe impacts on the rest of the model. Furthermore, lowering the sample rate reduces the resolution of the features, making it harder for the CNN to detect fine-grained patterns. As a result, the GRU encoder receives lower-quality temporal features, which would then adversely affect its ability to capture motion sequences. While these preprocessing techniques may work, our training may not have been long enough to observe these changes.

Due to limited computational power and computation time, each model was run on 40 epochs. However, models with increased training time may have improved performance as the training loss graphs indicated that the loss was still slightly decreasing by the time training stopped.

## References


[1] Kao, J. (2025) Winter 2025 ECE C147 Lecture 12 Video, UCLA, Los Angeles, CA.

[2] Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory. Neural Computation*, 9(8), 1735–1780.

[1] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* arXiv preprint arXiv:1406.1078.