ECE C147/C247 Report: Predicting Keystrokes from Electromyography Signals

Aparna Hariharan
UID: 705715914
Department of Computer Science
University of California, Los Angeles
aparnahariharan@ucla.edu

Joseph Janssen
UID: 905797640
Department of Computer Science
University of California, Los Angeles
josephjanssen11@gmail.com

Arathi Nair
UID: 805749584
Department of Computer Science
University of California, Los Angeles
arathinair@g.ucla.edu

Sruthi Rangarajan
UID: 405720436
Department of Computer Science
University of California, Los Angeles
sruthi.ranga3@gmail.com

Abstract

Surface electromyography (sEMG) signals have the potential to enable intuitive and efficient human-computer interaction, particularly for text entry applications. In this paper, we preprocess the emg2qwerty dataset using min-max normalization, time stretching, and downsampling to enhance signal quality. We also evaluate multiple deep learning architectures, including the baseline CNN model, after incorporating different regularization techniques like dropout, batch normalization, L2 regularization, and elastic net. We contrasted the baseline convolutional model with an alternative recurrent neural network (RNN) model and a hybrid model of CNN and RNN. We found that the best performing model was achieved through adding L2 regularization and a dropout layer to the baseline CNN architecture. Additionally, decreasing the sampling rate increased the resulting CER. These findings highlight the role that techniques such as data preprocessing and architectural choices can have in model performance while maintaining that not all of these techniques always result in performance improvements.

1 Introduction

Accurately decoding surface electromyography (sEMG) signals for text entry remains a challenging problem in human-computer interaction. The emg2qwerty dataset provides a rich source of high-resolution sEMG recordings, capturing fine motor control dynamics. Given the complexity and variability of sEMG signals, improving model generalization is crucial for robust performance across different contexts. Extensive data preprocessing techniques, including downsampling, were explored, but they worsened performance. This suggests that the dataset's inherent diversity—even within a single subject—already introduces sufficient variability for effective learning. Consequently, the focus shifted toward model-based generalization techniques, including dropout, batch normalization, L2 regularization, and elastic net regularization.

In addition to these generalization techniques, a recurrent neural network (RNN) was investigated as an alternative to the given convolutional neural network (CNN). Unlike feedforward architectures, RNNs are well-suited for capturing temporal dependencies in sequential data. Since sEMG signals exhibit strong temporal correlations due to the continuous nature of muscle activations, an RNN has the potential to better model these dependencies and improve decoding accuracy. A hybrid

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

model with both RNN and CNN components was also explored. This report details the approach to preprocessing, model selection, and evaluation, with the goal of optimizing character error rate (CER) while ensuring strong generalization performance.

2 Methods

We chose to run a majority of our tests using 30 epochs, including the baseline model provided to us. During experimentation for techniques that approached the baseline validation CER, we chose to run those tests for a higher number of epochs, which are specified in the results sections.

2.1 Data Preprocessing

To improve the generalization of the model and enhance the quality of input features, several data preprocessing techniques were applied to the sEMG signals. These methods aimed to reduce noise, normalize signal values, and introduce variations that could improve model robustness. Two different preprocessing trials were conducted: one using only downsampling and another incorporating all three augmentation techniques. Both of these trials included the baseline preprocessing pipeline.

2.1.1 Downsampling

To reduce the computational load and eliminate high-frequency noise, the raw sEMG signals were downsampled by a specified factor, *n*. This process involved selecting every *n*-th sample from the original signal while preserving temporal dependencies. Downsampling helps mitigate redundancy in high-resolution data while maintaining critical signal information.

2.1.2 Min-Max Normalization

Min-max normalization was applied to scale the sEMG signals within a defined range, in this case [0,1]. This technique ensures that all input features have comparable magnitudes, improving numerical stability during training. Given a signal x, the transformation follows:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

where $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values of the signal, respectively.

2.1.3 Time Stretching

Time stretching was employed to introduce variations in the temporal dynamics of sEMG signals. This augmentation technique expands or compresses the x-axis (time) of the signal without altering its frequency content. It aids in improving model robustness by exposing it to diverse time-scaled versions of the data while preserving the frequency patterns.

2.2 Regularization Techniques

To enhance the generalization capability of our model and mitigate overfitting, several regularization techniques were incorporated into the training pipeline. These methods aim to improve model robustness by restricting complexity and preventing the network from overfitting to the variability of the emg2qwerty training dataset. By integrating these regularization techniques, our implementation aimed to develop a model that generalizes well across different datasets.

2.2.1 Dropout

Dropout is a stochastic regularization technique that randomly deactivates a subset of neurons during each training iteration. This prevents the network from relying too heavily on specific neurons and encourages redundancy in feature learning, which improves generalization to unseen data. In our implementation, dropout was applied, with probability values chosen through empirical validation.

2.2.2 Batch Normalization

Batch normalization (BN) standardizes the activations within each mini-batch by normalizing them to have unit statistics, followed by a learnable affine transformation. Learning thus becomes simpler because parameters in the lower layers do not change the statistics of the input to a given layer. BN was applied to ensure consistent feature scaling, allowing the network to adapt effectively to variations in the input signal.

2.2.3 L2 Regularization

L2 regularization, or weight decay, penalizes large weight magnitudes by adding a term proportional to the squared L2 norm of the weights to the loss function. This encourages the model to maintain parameters closer to 0, thereby reducing overfitting. The loss function is as follows:

$$\mathcal{L}_{\text{L2}} = \mathcal{L} + \lambda \sum_i w_i^2 \tag{2}$$

where:

- $\mathcal{L}_{\text{L2}}$ is the total loss with L2 regularization.
- $\mathcal{L}$ is the original loss function (e.g., cross-entropy or mean squared error).
- w_i represents the weight parameters of the model.
- λ is the L2 regularization strength (hyperparameter).

2.2.4 Elastic Net Regularization

Elastic net combines L1 and L2 regularization, promoting both sparsity and weight decay.This hybrid approach is especially useful in high-dimensional feature spaces, as it leverages L1 regularization to promote sparse feature selection while using L2 to prevent excessive sparsity and improve stability. Given the complex structure of sEMG data, Elastic Net was explored as an alternative to pure L2 regularization to assess its impact on feature selection and model generalization. The loss function is as follows:

$$\mathcal{L}_{\text{ElasticNet}} = \mathcal{L} + \lambda_1 \sum_i |w_i| + \lambda_2 \sum_i w_i^2 \tag{3}$$

where:

- $\mathcal{L}_{\text{ElasticNet}}$ is the total loss with Elastic Net regularization.
- $\mathcal{L}$ is the original loss function.
- w_i represents the weight parameters of the model.
- λ_1 controls the strength of L1 regularization, encouraging sparsity.
- λ_2 controls the strength of L2 regularization, promoting weight decay.

2.3 Model Architectures

To effectively decode sEMG signals for text entry, multiple neural network architectures were explored, each designed to capture different aspects of the input data. This section describes the convolutional neural network (CNN), the recurrent neural network (RNN), and a hybrid approach combining both architectures. By evaluating these architectures, we assessed their effectiveness in improving character error rate (CER).

2.3.1 Convolutional Neural Network (CNN)

A convolutional neural network (CNN) was employed as the baseline architecture due to its ability to extract spatial patterns from high-dimensional sEMG data. The CNN follows a time-depth separable convolutional encoder design. The convolutional layers process the spectrogram-normalized input features, applying temporal convolutions. Despite CNNs' strong ability to extract spatial features, their inability to model long-term temporal dependencies prompted further exploration of alternative architectures.

2.3.2 Recurrent Neural Network (RNN)

Given the sequential nature of sEMG signals, an RNN-based architecture was investigated as an alternative to the CNN. Unlike CNNs, RNNs leverage recurrent connections to capture long-range dependencies, making them well-suited for modeling time-series data. The RNN model employs a stacked LSTM or GRU architecture, with configurable parameters such as hidden size, number of layers, and bidirectionality. These recurrent layers process the extracted features, maintaining temporal dependencies across time steps.

2.3.3 Hybrid (CNN + RNN)

To combine the strengths of both architectures, a hybrid CNN-RNN model was developed. In this approach, convolutional layers first extract spatial features from sEMG signals, which are then passed to recurrent layers to capture temporal dependencies. It uses both a convolutional encoder to perform local temporal modeling and a recurrent encoder (LSTM or GRU) to account for long-range dependencies.

3 Results

Table 1: Comparison of different models and techniques with their Validation Character Error Rate (CER) and Validation Loss. The model with the lowest validation CER is bolded.

Model + Technique	Validation CER	Validation Loss
Baseline Model and Data	26.1631	0.8681
Baseline Model + Downsampling	93.9965	3.0466
Baseline Model + Downsampling, Time Stretching, & Min-Max Norm.	100.0000	3.6808
Baseline Model + Dropout=0.2	29.3531	0.9317
Baseline Model + Dropout=0.3	29.6854	0.9425
Baseline Model + Dropout=0.4	30.8595	0.9559
Baseline Model + Batch Norm	82.2330	12.1359
Baseline Model + L2 Regularization	21.5773	1.5503
Baseline Model + L2 Regularization + Dropout=0.2	**20.7798**	**1.5496**
Baseline Model + Elastic Net + Dropout=0.2	22.3748	2.2133
RNN Encoder Model with 2 Layers & 256 Nodes	49.2025	1.3598
RNN Encoder Model with 3 Layers & 256 Nodes	73.4603	3.0090
RNN Encoder Model with 3 Layers & 512 Nodes	73.8148	2.6043
RNN + CNN Hybrid Model with 3 Layers & 256 Nodes + Downsampling and Min-Max Norm.	100.0000	3.2801

Some key plots are shown in the current section, but more can be found in the Appendix 6.

3.1 Baseline

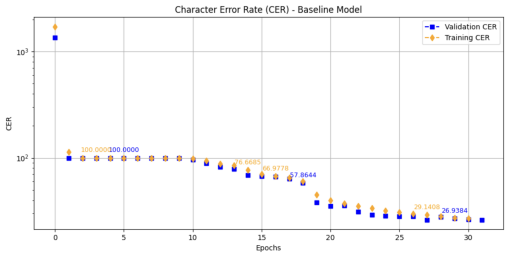

Figure 1: Training and validation CERs for baseline CNN model across 30 epochs.

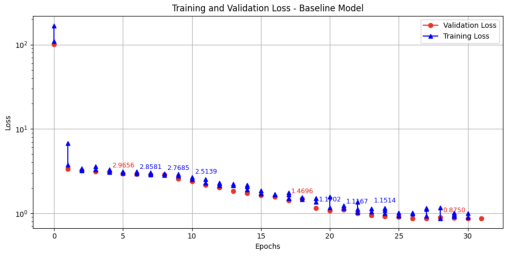

Figure 2: Training and validation losses for baseline CNN model across 30 epochs.

As shown in in Table 1, running the baseline model resulted in a validation CER of 26.1631.

3.2 Preprocessing the Data by Downsampling

After downsampling the data, the resulting validation CER was 93.9965. This is drastically higher than the validation CER attained by simply just running the baseline model on the data. The intention of downsampling is to remove high-frequency noise through reducing the amount of data, but the results in Table 1 show that this reduced data was insufficient for the model to make accurate predictions.

3.3 Preprocessing the Data by Downsampling, Min-Max Normalization, and Time Stretching

As shown in the results in Table 1, this combination of data augmentation techniques performed extremely poorly with the validation CER being 100.0000. Min-max normalization and time stretching were meant to scale the data into a consistent range and modify the speed or duration of the signal without changing its overall shape. However, this combination of data augmentation techniques likely stripped essential information needed for accurate predictions, causing the increase in the validation CER.

3.4 Adding a Dropout Layer

As shown in Table 1, we experimented with incorporating a dropout layer at three different probabilities: 0.2, 0.3, and 0.4. The results indicate that the validation CER values for all three configurations were consistently slightly higher than the validation CER of the baseline model. This suggests that while Dropout introduces regularization by preventing overfitting, it may also lead to a slight degradation in performance when applied independently.

3.5 Adding L2 Regularization

We added L2 Regularization to the baseline CNN model with $\lambda = 0.0001$. After 52 epochs, this technique resulted in a validation CER of 21.5773, the second-lowest validation CER recorded amongst all of the techniques we tried, as seen in Table 1. This technique proved to be the first that improved Validation CER below that of the baseline model.

3.6 Combining L2 Regularization and a Dropout Layer

Our results, as shown in Table 1, indicate that the combination of a dropout layer (probability = 0.2) and L2 regularization ($\lambda = 0.0001$) yielded the best performance among all tested configurations, with a validation CER of 20.7798 after 50 epochs.

3.7 Combining Elastic Net Regularization and a Dropout Layer

Our results indicate that the combination of dropout and elastic net performed better than the baseline validation CER, yet not as well as the combination of dropout and L2 regularization. We ran this experiment with $\lambda_1 = 0.00001$ and $\lambda_2 = 0.0001$ values. The validation CER for elastic net and a dropout layer (probability = 0.2) is 22.3748 after 50 epochs, as shown in Table 1.

3.8 Adding Batch Normalization

When adding 1D batch normalization and running for 43 epochs, the results in Table 1 show that the validation CER is higher than the baseline model at 82.2330. Further iterations of 1D batch normalization cause the validation CER to increase back up to 100.00 after hitting the minimum value of 82.2330.

3.9 RNN

After trying multiple modification methods on the baseline CNN model, we implemented a RNN Encoder Model and tested it with a variation in the number of layers and input nodes. As per the results shown in Table 1, running the RNN Encoder Model with 2 Layers and 256 Nodes results in a validation CER of 49.2025. When the number of layers is increased, running the RNN Encoder Model with 3 Layers and 256 Nodes results in a validation CER of 73.4603 after 40 epochs. Furthermore,

when the number of nodes is increased, running the RNN Encoder Model with 3 Layers and 512 Nodes results in a validation CER of 73.8148, a CER very similar to the previous configuration of 3 Layers and 256 Nodes. This leads us to hypothesize that the increase in layers is more detrimental to the robustness of the model than the increase in nodes. Overall, a Recurrent Neural Network Encoder Model proved to perform more poorly than the CNN model with our architectural modifications.

3.10 RNN+CNN

To further our experiments with the RNN Encoder Model, we created a hybrid RNN+CNN model that we ran with 3 layers and 256 nodes alongside augmentation of the data using downsampling and min-max normalization. The results in Table 1 show that this combination technique resulted in a validation CER of 100.0000.

4 Discussion

4.1 Effectiveness of Dropout and L2 Regularization

We hypothesized that the combination of dropout and L2 regularization would outperform other configurations because they address two different aspects of overfitting. Dropout reduces dependencies between neurons, while L2 regularization constrains model complexity by discouraging extreme weight values. Our results show that this combination outperforms using L2 regularization alone with the baseline model. This is because dropout forces the model to utilize multiple neurons for the same tasks, resulting in greater robustness and generalization compared to relying solely on L2 regularization.

This balance between dropout and L2 regularization likely enhances the model's ability to generalize to unseen data, reducing overfitting without sacrificing performance. Consequently, the validation CER is lower in models that incorporate both techniques rather than using them in isolation.

4.2 Challenges with Batch Normalization

In contrast, our results indicate that incorporating batch normalization in the baseline model leads to poor performance. During early epochs, batch normalization causes a slight decrease in the CER, but as training continues, the CER begins to rise again and plateaus at 100. This pattern may be due to discrepancies between mini-batch statistics and the running averages accumulated during training, which can negatively impact model stability. This could also be attributed to persistent issues faced in the evaluation pipeline, where both training and validation CERs plateaued at 100.

Despite this issue, when training and validation loss were examined (Figure 9), a general downward trend was observed, indicating that the model continued optimizing its objective. Also, running the batch normalization variant of the model caused the notebook to time out at around 40 epochs multiple times, preventing a complete evaluation of its overall performance.

4.3 Impact of Preprocessing Techniques

Our results suggest that not all preprocessing techniques are beneficial, and in some cases, they can significantly reduce performance. For instance, while downsampling is often used to reduce computation time, it resulted in a validation CER of 93.99, indicating that the model struggled to differentiate between keystrokes.

Moreover, combining downsampling with time stretching and min-max normalization caused the model's performance to collapse entirely. We initially hypothesized that this combination would help the model learn features robustly; however, our findings suggest that while normalization can be useful in certain contexts, it must be carefully tuned to avoid distorting the original signal. This could also be attributed to persistent issues faced in the evaluation pipeline, where both training and validation CERs plateaued at 100.

4.4 Limitations of RNN-Based Architectures

We also observed that RNN-based models did not perform as well as expected. The RNN encoder with three layers and 256 nodes achieved a validation CER of 73.46, which was significantly worse than the best-performing CNN model. This poor performance may be due to vanishing gradients or difficulty in capturing long-term dependencies in the sEMG signals.

While RNNs are typically effective for sequential data, our results indicate that additional modifications, such as layer reordering or alternative optimization techniques, may be necessary to improve performance. Additionally, increasing the number of layers from 2 to 3 and the number of nodes from 256 to 512 led to even worse performance than our original RNN, likely due to RNNs being prone to overfitting. The hybrid CNN-RNN architecture performed poorly, with a plateaued validation CER of 100, possibly due to issues in the evaluation pipeline.

4.5 Key Takeaways

Ultimately, our findings highlight several key insights:

- Regularization techniques such as dropout and L2 regularization play a crucial role in preventing overfitting, leading to better generalization.
- Preprocessing techniques must be carefully selected, as excessive transformations and noise can strip away critical information and degrade performance.
- RNN-based architectures have potential but require additional refinements from the configurations we tested. The poor performance in our implementation was likely due to external factors, such as the evaluation pipeline, rather than inherent flaws in the model itself.

5 References

Hannun, A. (2017). Sequence modeling with CTC. *Distill*, *2*(11), e8. `https://doi.org/10.23915/distill.00008`.

Sivakumar, V., Seely, J., Du, A., Bittner, S. R., Berenzweig, A., Bolarinwa, A., Gramfort, A., & Mandel, M. I. (2024). *emg2qwerty: A Large Dataset with Baselines for Touch Typing using Surface Electromyography*. arXiv preprint arXiv:2410.20081. Retrieved from `https://arxiv.org/abs/2410.20081`.

6 Appendix

6.1 Additional Figures

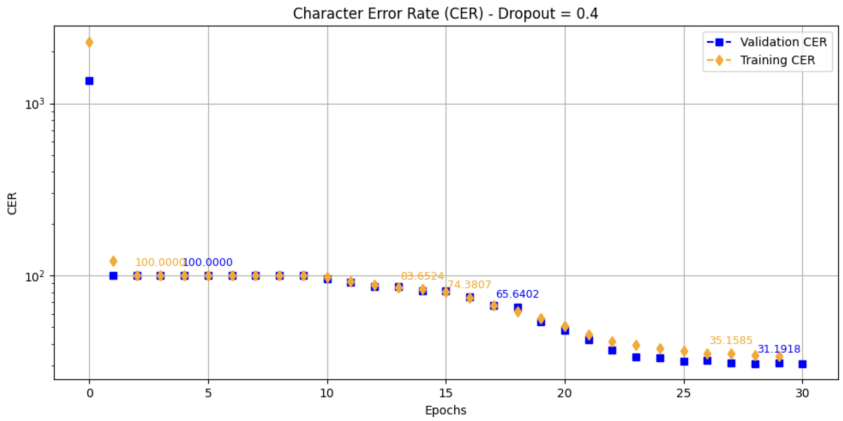

Figure 3: Baseline CNN Model CER with Dropout=0.4.

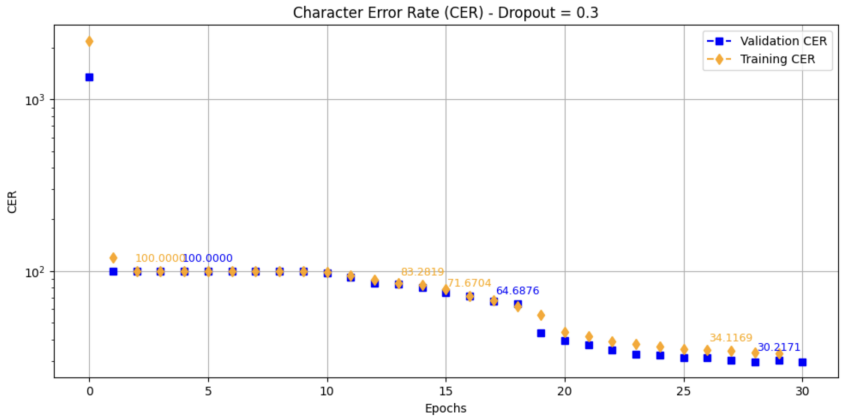

Figure 4: Baseline CNN Model CER with Dropout=0.3.

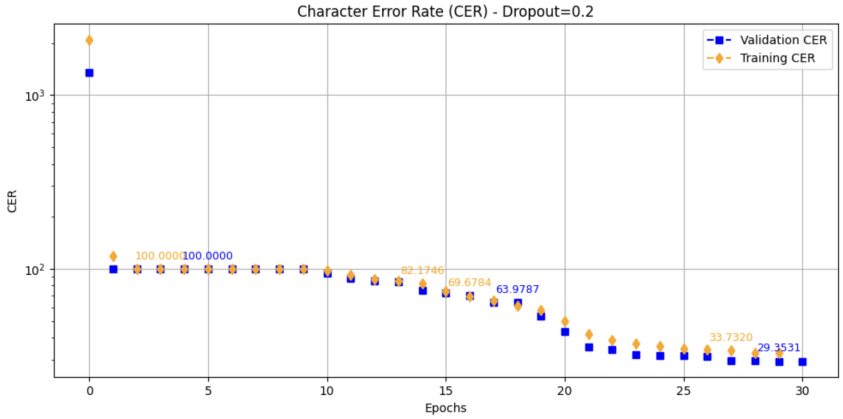

Figure 5: Baseline CNN Model CER with Dropout=0.2.

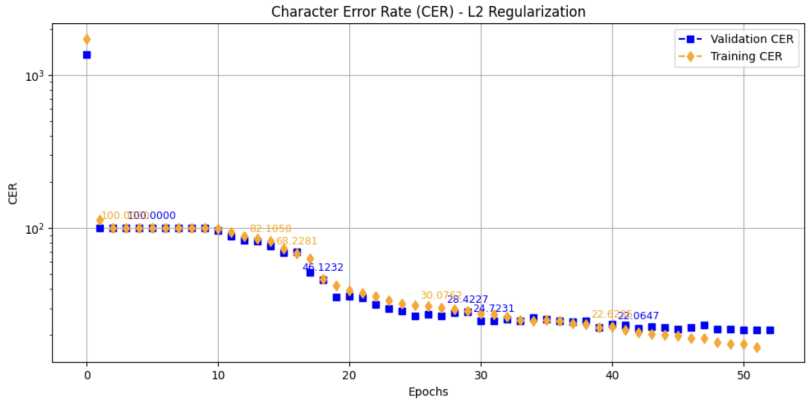

Figure 6: Baseline CNN Model CER with L2 Regularization.

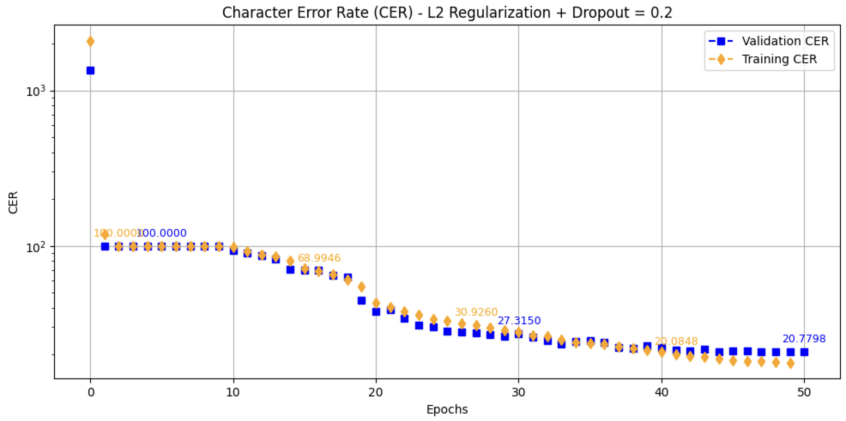

Figure 7: Baseline CNN Model CER with Dropout=0.2 and L2 Regularization.

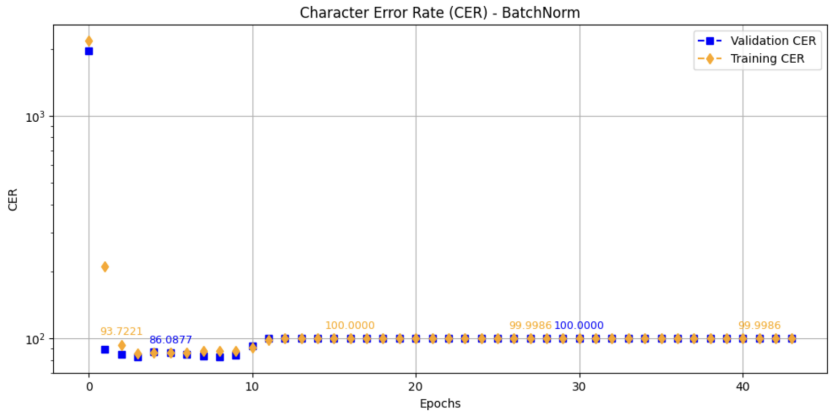

Figure 8: Baseline CNN Model CER with Batch Normalization.

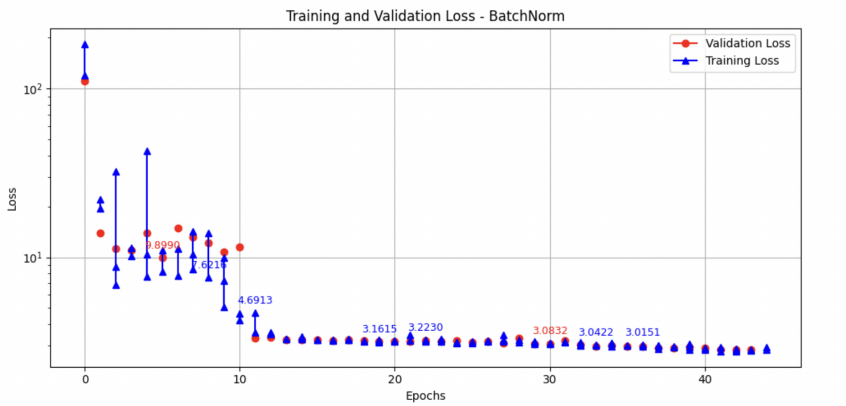

Figure 9: Baseline CNN Model Training/Validation Loss with Batch Normalization.

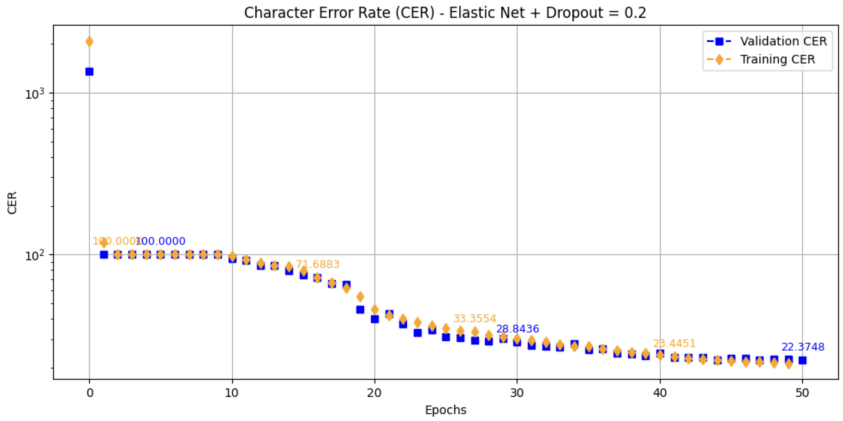

Figure 10: Baseline CNN Model CER with Dropout=0.2 and Elastic Net.

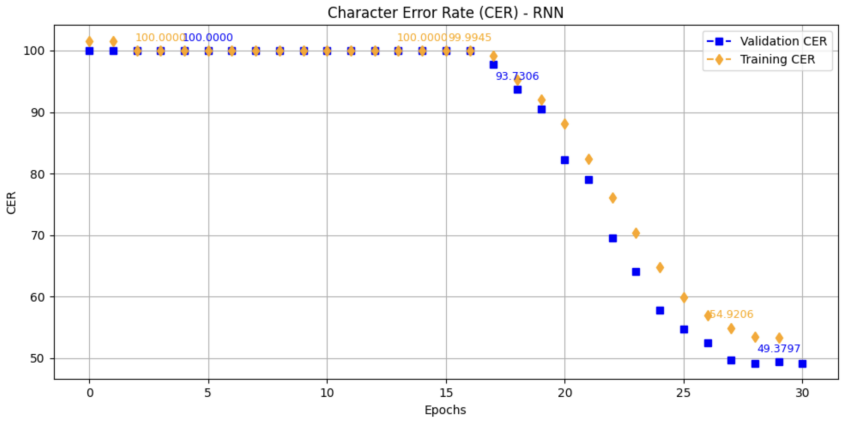

Figure 11: RNN Encoder Model with 2 Layers & 256 Nodes CER.

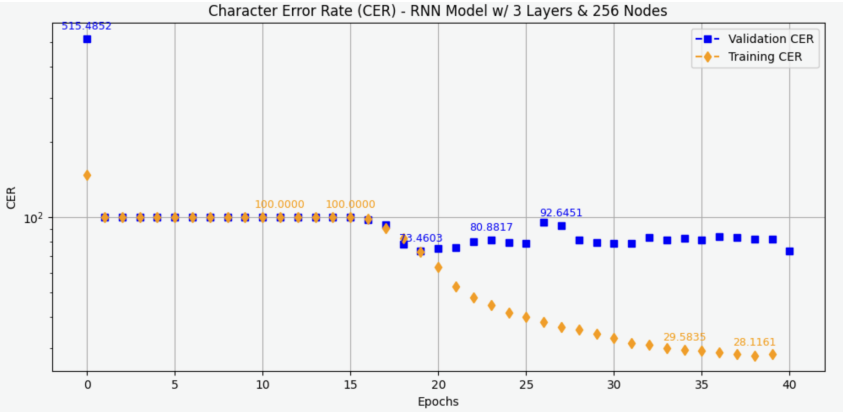

Figure 12: RNN Encoder Model with 3 Layers & 256 Nodes CER.

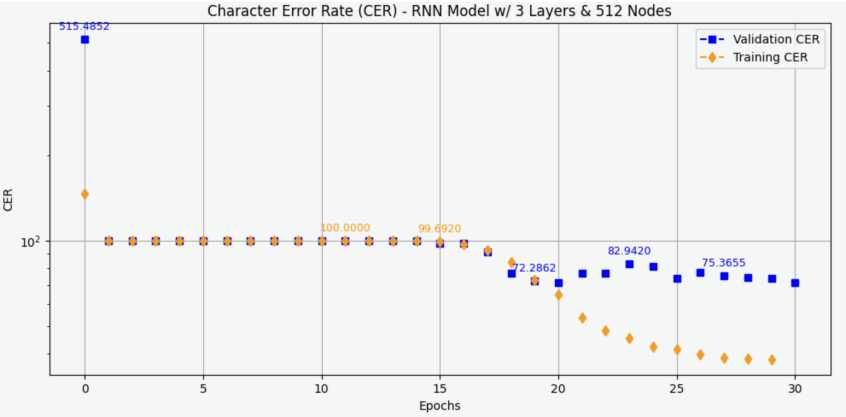

Figure 13: RNN Encoder Model with 3 Layers & 512 Nodes CER.

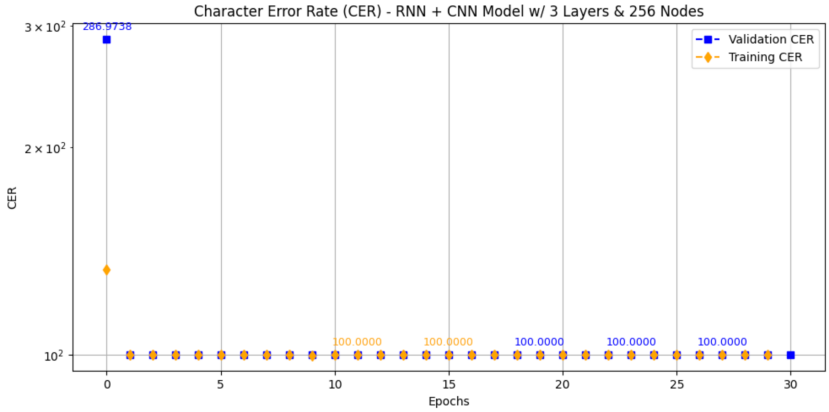

Figure 14: RNN + CNN Hybrid Model with 3 Layers & 256 Nodes CER.

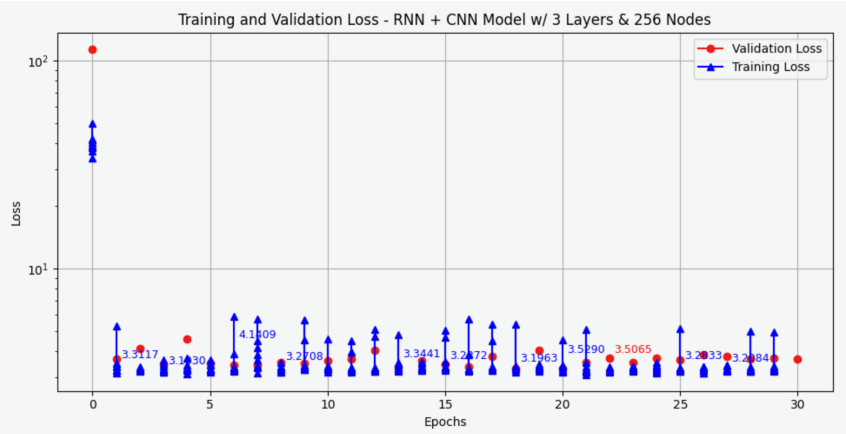

Figure 15: RNN + CNN Hybrid Model with 3 Layers & 256 Nodes loss.