Attention and LSTM with CNNs to minimize CER

Tommy Thach
University of California, Los Angeles
ttthach100@g.ucla.edu

Abstract

Using electromyography signals from the emg2qwerty dataset, I focused on real-world use of the model and tested on windows of 300ms for a single user. Additionally, due to limited time to test different models, I aimed to implement models that were relatively efficient and had lower numbers of parameters. Using a slightly edited version of the baseline model, I tested architectures using different combinations of the baseline model with additions of long short-term memory and multi-headed attention modules. In addition to these models, I also implemented different transforms and edited the baseline transforms. Though these new architectures and transforms did not get amazing performances with these new architectures and transforms, it was still largely improved upon the baseline model with about a 25 percent decrease in character error rate. The best model I found came from two attention modules surrounding around the original the baseline convolutional model.

1 Introduction

The dataset for emg2qwerty gathered surface electromyography (sEMG) data from participants typing (6). As such, accurate decoding of these sEMG signals would allow for subjects to type through use of only movements of their hands. To allow for this accurate decoding, the character error rate (CER) must be minimized so that each movement and signal correctly corresponds to the character that the subject is trying to type. These signals can vary greatly due to a variety of conditions, especially from person to person, so focusing on a single subject and training deep learning models based off of only that subject's sEMG data would allow for more specialized and accurate decoding for that subject.

In this project, to support this accurate decoding for a single subject, I focused on minimizing the CER using different architectures with some changes to the transforms. Additionally, to support real-time usage of the decoding, I used test-time windows of 300ms and focused on keeping the number of parameters lower. Being able to train more quickly due to the lower number of parameters also helped with training and testing more models during the limited amount of time given for the project.

This project mainly focuses on the implementation of long short-term memory (LSTM) recurrent neural networks and multi-headed self-attention (MHSA) with relative positional encoding (1) in addition to the baseline model of time-depth separable (TDS) convolutional neural networks (3) that was originally used with the emg2qwerty dataset and found improvements to CER through the additions of LSTM and attention. While the largest increase in improvement was a minor change to the padding of the original TDS model, adding attention modules to TDS increased performance further. LSTM and TDS improved upon the baseline, but was greatly outperformed by attention and TDS models.

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

2 Methods

2.1 Pre-Processing

While Sivakumar et al. in the emg2qwerty paper originally train on windows of 4 seconds and feeds the whole sequence during test-time (6), I believed that to be unrealistic for real-time decoding. It was found that windows will need to be 300ms or less (4) in order to be used in real-time with EMG data. Thus, feeding the entire sequence at test-time would allow for the model to possibly use data from a larger window than this or data from the future to predict the current key. In keeping with this, I trained with windows of 300ms, 600ms, or 1200ms and used 300ms windows for validation and testing.

To keep the ratios similar to the original transforms, I changed the temporal jitter to be 10ms instead of 60ms and the time mask in SpecAugment to be 8ms instead of 200ms. I also trained without any transforms.

Additionally, as I noticed that insertion error rates were a significant portion of CER while deletion error rates were lower, I also implemented a Butterworth filter to act as a low-pass filter and tested values of 400Hz, 450Hz and 500Hz. However, this filter did not improve performance and only increased training time due to additional pre-processing so I did not include results. I also attempted to use the raw waveforms but found that this did not achieve good performance and took much longer to run than just using spectral features so those results will also not be included.

2.2 Model Architecture

The baseline TDS Convolution model worked fairly well and was efficient in its number of parameters so I chose to keep the baseline model and add other modules alongside it. The kernel size was reduced from 32 due to being too large for the smaller windows of data.

One change made from the original TDS Convolution model was the addition of padding so the sequence lengths remained the same after every convolution layer. This padded TDS model was what I used alongside other models.

Taking inspiration from the Gulati et al. and their Conformer model for automatic speech recognition (ASR) (2), I wanted to try to capture global dependencies within each window while the TDS convolution captured more local dependencies. To capture these global dependencies, I attempted to tested both multi-headed attention with relative positional encoding, as was implemented in the Conformer, and LSTM and compared performances.

For the LSTM module, I used a bidirectional LSTM with four layers with a hidden dimension size of 128. This was followed by the feedforward module from the baseline TDS implementation and then a linear layer to expand from the hidden dimension back to the original number of features.

For the attention module, I used an implementation from labml.ai (5) of the relative positional encoding from the Transformer-XL model by Dai et al (1). This choice to keep relative positional encoding was because I wanted to encode positions for attention while keeping the memory usage low.
I used the MHSA module from the Conformer model which was a layer normalization followed by multi-headed self-attention with positional encoding and dropout.

To further lower the number of parameters and memory needed for the MHSA module, I added a 1D convolution layer to reduce the number of features, ran it through the MHSA module, and used 1D convolution to expand back to the original number along with a residual connection.

For both the LSTM module and subsampled MHSA module, I tested different orders with the padded TDS encoder. These orders included having a module before TDS, after TDS, and modules "sandwiched" around TDS.

Everything else was kept the same as the original model implementation with a Spectrogram Norm and Multilayer Perceptron at the beginning and Linear Layer and Softmax at the end.

3 Results

For consistency, I used 30 epochs for all models with testing and validation windows of 300ms, though some seemed as though they were continuing to learn when the limit was reached. All models unless specified were trained on 300ms windows as well.

As a baseline for my results, the original TDS encoder using a kernel size of 7 with windows of 300ms achieved a validation CER of 56.1 and test CER of 58.7. After preliminary testing, larger kernel sizes worked better due to the larger receptive field and 7 was the largest that worked.

With the minor change of keeping sequence lengths the same with padding, the TDS encoder was able to use a kernel size of 9 and then achieved validation CER of 42.8 and test CER of 47.9, resulting in the biggest jump in performance I found through my experiments.

Using 4 layers and a hidden dimension of 128, LSTM alone achieved 50.4 and 53.8.

LSTM after TDS encoder achieved 43.8 and 50.1 for validation and testing, respectively. LSTM before and LSTM sandwiching TDS both had losses greatly increasing halfway through training so I ended early.

As we can see, LSTM did not seem to perform well alongside TDS or alone. Thus, I chose to focus on attention, after its success in the Conformer model, in order to lower performance.

My first test using an Attention module, without subsampling, "sandwiching" TDS achieved the best performances up to that point with 41.1 and 46.6 CER. I continued to pursue this architecture as a result. However, it had 13.1 million parameters in comparison to the baseline model's 5.3 million so I chose to use subsampling in my Attention module to reduce that number and to increase efficiency.

I tested subsampling to many different values but most performed around the same level as the original attention module without subsampling. Thus, I chose to stick with the value of 128 as it was the lowest number with the same performance without subsampling.

Another decrease in CER came at the removal of all transforms and data augmentation. With subsampled attention sandwiching TDS and transforms removed, it was able to achieve my overall best performance of 37.0 and 45.0 CER.

I also tested out changing the training window sizes to 600ms and 1200ms. While I saw a large decrease in training loss going to about 0.7 whereas other models achieved about 1.1 loss, it achieved about the same performance as training with 300ms windows, suggesting possible overfitting.

While I was unable to do a complete check of hyperparameters, my model was able to perform much better than the baseline with what I had found, improving by over 25 percent from the baseline model.

4 Discussion

The best jumps in performance came due to the padding, attention sandwiching the TDS, and removal of all transforms. I believe the padding helped increase performance significantly because of the new window sizes that I used. While I did not test, I would guess that this change in padding might not be as impactful with larger window sizes since convolution would not significantly decrease sequence lengths so most of the sequential information would be retained. However, because window sizes were much smaller here, much more information would be lost and fewer features would be gathered as it goes through the layers. Thus, padding was a very significant change here in order to retain more information through the layers.

As explained in the Conformer paper (2), attention could capture global dependencies while convolution captures local information. Because of this, using attention before convolution could highlight some important aspects for convolution to focus on locally. Then, attention after might again highlight some important information and connect global and local dependencies.

Finally, while transforms might be useful for longer scale training, the issues with performances in my experiments seemed to be due to underfitting rather than overfitting. Thus, data transforms and augmentation might help generalize for deeper models that have longer time to train but might make it more difficult for shallow models like mine to find patterns. Thus, removing those transforms and allowing it to train on the data without augmentation helps it capture those patterns better.

References

[1] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., Salakhutdinov, R. (2019). *Transformer-xl: Attentive language models beyond a fixed-length context* (arXiv:1901.02860). arXiv. https://doi.org/10.48550/arXiv.1901.02860

[2] Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., Pang, R. (2020). *Conformer: Convolution-augmented transformer for speech recognition* (arXiv:2005.08100). arXiv. https://doi.org/10.48550/arXiv.2005.08100

[3] Hannun, A., Lee, A., Xu, Q., Collobert, R. (2019). *Sequence-to-sequence speech recognition with time-depth separable convolutions* (arXiv:1904.02619). arXiv. https://doi.org/10.48550/arXiv.1904.02619

[4] Jaramillo-Yánez, A., Benalcázar, M. E., Mena-Maldonado, E. (2020). *Real-time hand gesture recognition using surface electromyography and machine learning: A systematic literature review.* Sensors, 20(9), 2467. https://doi.org/10.3390/s20092467

[5] labml.ai. (2021). *Relative Multi-Headed Attention* [Computer software]. Github Repository. https://github.com/labmlai/annotated_deep_learning_paper_implementations/blob/master/labml_nn/transformers/xl/relative_mha.py

[6] 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:2410.20081). arXiv. https://doi.org/10.48550/arXiv.2410.20081