Beyond Fixed Receptive Fields: Decoding Touch Typing from sEMG via Recurrent and Transformer Architectures

Leon Liu*
Department of Computer Science
University of California, Los Angeles
liuleon360@gmail.com

Rathul Anand*
Department of Computer Science
University of California, Los Angeles
rathul@g.ucla.edu

Abstract

This project investigates the effectiveness of various sequence processing architectures, specifically Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs) [1], Long Short-Term Memory networks (LSTMs) [6], and Transformers [13, 12], in decoding surface electromyography (sEMG) signals into corresponding key press sequences. These signals are collected from wearable wrist sensors during touch typing on a QWERTY keyboard. By leveraging the emg2qwerty dataset [11], we aim to determine which sequence processing architecture offers superior performance in decoding sEMG signals for accurate text input recognition from a single user. Our findings show that transformer and recurrent architectures with more flexible receptive fields significantly outperform the baseline Time-Depth Separable ConvNet trained by Sivakumar et al. [11], addressing the hypothesis that fixed receptive fields are necessary for this task. Additionally, we demonstrate that log spectrograms can be downsampled more aggressively (down to 50Hz from the baseline 125Hz) without significant performance degradation, which helps offset the computational costs associated with these more powerful sequence processing architectures and offers efficiency improvements upon the previous baseline. We additionally explore the impacts of training sequence length on generalization, revealing limitations on generalization of transformer models to longer sequences for limited computational training and single user data.

1 Introduction

In prior work by the creators of this dataset, Sivakumar et al. used Time-Depth Separable ConvNets (TDS), originally developed by [5] for ASR, as their primary sequence processing module to perform convolution across a set temporal receptive field pointing to the the parameter efficiency of convolutional modules and the fairly short duration of sEMG activity corresponding to a single key press[11] . This approach leverages the parameter efficiency of TDS convolutional modules and the relatively short duration of sEMG activity associated with individual key presses. They noted the prevalence of co-articulation within the signal data, where the sEMG of a keystroke is influenced by preceding and succeeding keystrokes due to preparatory behaviors [11] . The rationale for a TDS module thus is that the fixed temporal window in TDS models introduces an inductive bias, focusing the model on the signal corresponding to a few key presses, thereby mitigating overfitting and reducing reliance on language modeling [11] .

This previous work also mentioned the potential usage of alternative architectures such as ResNets, RNNs, and transformers which have seen usage in Automatic Speech Recognition (such as in OpenAI's Whisper ASR model [10]), a similar sequence-to-sequence signal processing task. Thus, this project aims to explore these alternative sequence processing models and test the hypothesis of

whether these models with the capability of incorporating information from across the entire sequence will overfit or exhibit worse generalization.

We hypothesize that models capable of incorporating information from the entire sequence may exhibit different generalization patterns compared to those with constrained temporal windows. "Additionally, considering variations in typing speeds among users, architectures like Transformers [13, 12], which can attend to the entire sequence rather than a human-selected fixed temporal window, might better adapt to such variability However, due to computational limitations, this project only experiments with a single user's data in which the observed variability between typing speeds may be limited. Future work could extend our approach to multiple users to better asses zero-shot generalization.

We explore different variations of recurrent neural networks and transformers to address pitfalls observed in our iterative experimenting. Notably, we experiment with the effect of different position embedding paradigms (sinusoidal and RoPE) in attempts to improve transformer generalization to unseen sequence lengths. We also examine bidirectional encoder transformers in contrast to causal transformers, which employ causal attention masking. This masking mechanism restricts each input element to attend only to previous elements, mirroring the logical receptive field of recurrent architectures to make a comparison between the transformer's attention mechanism with recurrent formulations.

With our limited computational budget, we additionally explore data augmentation in the effect of increasing the sequence/window length on model performance and the performance deterioration of increasing the hop length of log spectrograms, which effectively lowers the sample frequency, whilst speeding up training and inference.

1.1 Background

Recurrent Neural Networks (RNNs) are architectures that adopt a recurrent approach to processing sequences by processing the sequence timestep by timestep whilst processing and condensing information from each timestep into a hidden state or cell state in the case of a GRU (Gated Recurrent Unit) or LSTM (Long Short Term Memory) model. RNNs process sequences iteratively, updating a hidden state at each timestep to capture temporal dependencies. However, they are susceptible to the vanishing gradient problem, hindering their ability to learn long-term dependencies. LSTMs mitigate this issue through gating mechanisms that gate information flow into a long-term (across timesteps) cell state, improving the modeling of dependencies across long time intervals [6]. Similarly, Gated Recurrent Units (GRUs) incorporate gating mechanisms to improve information retention but use a more simplified structure compared to LSTMs, making them computationally more efficient [1].

Alternatively, transformers utilize the self-attention mechanism to perform sequence processing, allowing each input element to weigh the importance of all other elements in the sequence and move information from them [13]. In particular this mechanism is implemented through a Query, Key, Value (QKV) system in which every element produces a query, key, and value vector via learned linear layers [13]. The cosine similarity between an element's query and all other keys (in bidirectional attention / without causal attention masking) and then moves information in the form of values from them using the weighted average of the softmax of these similarity scores. The attention mechanism can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

as introduced in [13]. In particular, because this task is a sequence-to-sequence domain transfer task, we opt to use a bidirectional encoder transformer architecture that allows each timesteps' feature vector to attend to past and future time steps [2]. This architecture allows the model to move information from any timestep to any other timestep based on learnable query and key similarity. However, to perform an ablation study and provide a more direct comparison to the inherent causal formulation of RNNs and LSTMs, we also investigate the effectiveness of a causal transformer, where attention is restricted to past and present timesteps. The use of causal architectures is also motivated by the potential future real-time decoding applications where sEMG signals are practically used to type.

A key consideration in transformer-based sequence modeling is the lack of inherent positional information, as the self-attention operation is invariant to permutation (without causal attention masking). To address this, we experiment with two commonly used positional embedding schemes in transformers: sinusoidal positional embeddings and rotary positional embeddings (RoPE) [13, 12]. Sinusoidal embeddings encode absolute positions using predefined trigonometric functions, ensuring that the model can infer positional relationships from fixed encodings. In contrast, RoPE introduces relative positional encoding by applying rotational transformations to query and key vectors. This formulation ensures that the inner product (which determines the similarity scores in self-attention) between any two timesteps are invariant to absolute position shifts, enabling the RoFormer (transformer using RoPE) model to generalize more effectively to variable sequence lengths and unseen temporal structures [12].

2 Methods

2.1 Data Pre-processing

When testing the architectures, to focus on comparing the architectures themselves, the data pre-processing steps were kept the same for all architectures and were the original baseline configurations described in Sivakumar et al. [11].

The data pre-processing transformations are as follows:

1. **Random Band Rotation** (shift of electrode channels) with offset sampled uniformly from $\{-1, 0, 1\}$.
2. **Temporal Alignment Jitter** of the alignment between left and right hand sEMG timeseries by an offset between 0 to 120 samples which translates to 60 ms jitter for 2kHz EMG
3. **Log Spectrogram** for each electrode, extract a log spectrogram with 33 linearly spaced frequency bands, the sampling frequency is downsampled from 2kHz to 125 Hz as in the original paper
4. **SpecAugment** [9] applies time and frequency aligned masks which masks out time intervals and frequency intervals respectively
   (a) 3 time masks with max 200 ms masking
   (b) 2 frequency masks with max 4 frequency bins

2.2 Learning Rate Scheduler

We use the same learning rate schedule that linearly warms up to the target learning rate and decays with a cosine annealing schedule. Sivakumar et al. [11] use a learning rate of 1e-3 which we also use for the TDS Convolution module that we train, but we then use a lower learning rate of 1e-4, 4e-4, and 5e-4 when training our other modules with lower learning rates generally corresponding to the larger models, namely the transformers for more stability.

2.3 Optimizer

We also instead opt to use AdamW [8] over Adam [7] because prior work has demonstrated that AdamW empirically improves generalization when using weight decay, which we use for our larger models [8].

2.4 Loss Function and Metrics

We continue to use the CTCLoss function for computing model loss during training [4]. We also use Character Error Rate (CER) as a metric for evaluating model performance. CER is defined as:

$$\text{CER} = \frac{S + D + I}{N} \tag{2}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of characters in the text.

2.5 Time Depth Separable ConvNets

As a baseline, we maintained the same hyperparameters for the TDS convolution module [5] using 24 channels uniformly over the conv blocks, 384 MLP features, and a kernel width of 32 (a receptive field of 1 second) [11].

2.6 Recurrent Neural Networks

We evaluated two different RNN architectures of varying sizes, with both Gated Recurrent Units (GRUs) [1] and Long Short-Term Memory Units (LSTMs) [6] as the recurrent unit to better capture longer-term dependencies throughout the sequences. As for architectures, we trained both a vanilla RNN and a residual RNN that included residual connections between layers and introduced gradient clipping.

2.7 Bidirectional Encoder Transformer with Sinusoidal Position Embeddings

We experiment with the use of a bidirectional transformer for this task, that is, the model performs sEMG to qwerty sequence-to-sequence domain transfer so a bidirectional encoder is applicable as this is not necessarily an autoregressive task. In particular, in contrast to decoder transformers which use an autoregressive formulation, our bidirectional encoder allows input elements to attend to elements in both directions [2] rather than only at previous elements in a causal or autoregressive formulation. We explored two variations of transformer model sizes. The smaller model ("Transformer Small") has 384 MLP features, a model dimension of 384, 4 encoder layers, and 8 attention heads per layer. The larger model ("Transformer Med") has 768 MLP features, and a model dimension of 768 to match, 12 encoder layers, and 8 attention heads per layer.

2.8 RoFormer (Bidirectional Transformer with Rotary Position Embedding)

We trained a smaller transformer with rotary embeddings [12] ("RoFormer Small") similar to the small transformer with sinusoidal position embeddings after observing the overfitting of the larger model. We tried rotary embeddings in an attempt to achieve better generalization on longer sequences, since rotary embeddings allow for invariance to absolute positional shifts [12]. This RoFormer had 384 MLP features, a model dim of 384, 4 encoder layers, and 6 attention heads per layer for more computational efficiency due to the compute limits of this project.

Regarding the rationale behind the model sizes of these transformers architectures, we decided to use similar model sizes to OpenAI's Whisper architecture which uses an encoder-decoder (we only use an encoder) transformer structure on the similar task of Automatic Speech Recognition to convert log-spectrograms of speech audio to text [10]. Namely, our "Small" corresponds to Whisper's parameters for their "Tiny" model, and our "Medium" corresponds to their "Small" as we do not train up to their larger model sizes due to computational budget limitations.

Discussed further below, we also explore the effectiveness of increasing the training sequence length as the baseline test performance is gathered from unsplit/untruncated test sessions whilst the models are trained on smaller sequence lengths.

2.9 Causal Transformer with Rotary Position Embedding

We also tested a causal transformer model based on the small RoFormer architecture by adding a causal mask and pre-layer norm. Although this also slightly harms performance, such an architecture could be scaled up for real-time decoding performance.

2.10 Data Augmentation Experiments on Sampling Rate of Log Spectrograms

We explore the deterioration of performance of increasing the hop length / downsampling of the input 2kHz sequence data with larger hops when generating the log spectrograms that are input into the model. We trained our best performing model parameters, the RoFormer Small, from random initialization on datasets with varying hop lengths for 500 epochs: the baseline hop length of 16 which downsamples to 125Hz spectrograms, and hop lengths of 20, 40, and 80 which downsample to 100, 50, and 25 Hz respectively.

3 Results and Discussion

Table 1: Model Performance Comparison

Model	Val Loss	Val CER (%)	Test Loss	Test CER (%)
TDS Conv	1.12	18.96	1.48	22.95
Transformer Small	0.66	12.69	3.28	33.04*
Transformer Med	0.83	25.76	2.26	43.40*
RoFormer Small	0.83	**12.23**	1.32	**17.52***

*Test metrics from sliced windows evaluation

Table 2: Casual Model Performance Comparison

Model	Loss	CER (%)
Causal Transformer Small (Rotary Embeddings, Pre-norm)	0.74	14.69
Causal Transformer Med (Rotary Embeddings, Pre-norm)	0.67	**13.91**
RNN (GRU)	0.65	16.88
Residual RNN (GRU)	0.62	**16.79**
RNN (LSTM)	0.64	19.30
Residual RNN (LSTM)	0.66	19.52

Table 3: Window Length / Hop Length for RNN Experiments

Unit	Hop Length	Window Length	Val CER (%)	Test CER (%)
GRU	16	4 sec	17.61	18.44*
GRU	40	4 sec	18.45	20.10*
GRU	80	4 sec	22.00	22.82*
GRU	40	10 sec	**16.71**	**16.81***
GRU	80	10 sec	17.61	18.44*

*Test metrics from sliced windows evaluation

3.1 RNN and Causal Transformer Results

We conducted experiments across hyperparameters for RNN-based architectures using both GRU and LSTM cells, performing two groups of experiments: *regular RNN models* and *residual RNN models*, with results summarized in tables 6 and 7. A summarized version of the validation loss and CER from each model in this group, along with our causal transformer experiments, is shown in Table 2.

Notably, GRU-based models generally require fewer parameters and offer faster training times due to their simplified gating mechanisms, making them particularly attractive for on-device or resource-constrained deployment scenarios. The GRU configurations reached a lowest validation CER of 16.88% with 120K steps, while the best LSTM model in this group offered a competitive validation CER of 19.30% with fewer (30K) training steps. However, the LSTM model was much larger (twice as many layers, and an MLP dimension of 512 compared to 384) and thus overall less efficient. The high performance of GRUs suggests that the GRU's simpler architecture is sufficient to capture the short-term dependencies necessary for competitive decoding in the emg2qwerty task, and the LSTM's higher capacity to capture and store long-term dependencies may have only help marginally by the nature of this data.

Residual connections were introduced in a second set of RNN experiments to facilitate gradient flow in deeper recurrent models. The performance of these models is mixed: overall, we see minor gains from adding residual connections for both GRU and LSTM models of varying depth and width. Notably, a residual GRU variant trained for 30K steps achieved slightly better performance than an equivalent non-residual vanilla GRU model trained for 60K steps, suggesting that residual connections can improve performance by improving gradient quality and propagation, but the benefits

Table 4: Log Spectrogram Hop Length / sEMG Sample Frequency for RoFormer Experiments

Model	Sample Freq (Hz)	Val Loss	Val CER (%)	Test Loss	Test CER (%)
RoFormer Small	125	0.83	**12.23**	1.32*	**17.52***
RoFormer Small	100	0.78	12.69	2.17*	31.79*
RoFormer Small	50	0.82	13.27	1.33*	20.20*
RoFormer Small	25	0.98	16.37	1.91*	28.77*

*Test metrics from sliced windows evaluation

seem to be sensitive to choice of hyperparameters and training duration. It is plausible that the benefits of residual connections would become more significant in deeper recurrent models beyond our maximum tested depth of 4 layers.

However, we see that the initial experiments with a causal transformer architecture outperforms all of our recurrent models and we thus continue to iterate on our transformer architecture. This aligns with the improvements we see from transformers over recurrent models in other domains, such as language modeling. Comparing models with varying depths (up to 6) and hidden dimensions (up to 512), transformers achieved impressive training performance with CERs as low as 3.58%, but validation CERs remained at 13.91%, suggesting that although transformers are highly effective at capturing complex and long-range dependencies, further regularization or data augmentation may be necessary to improve generalization. While GRU-based models sometimes show slightly higher CERs than transformers, they do provide a competitive alternative when balancing against training speed and hardware limitations for on-device deployment, due to their fewer parameters and simpler gating mechanisms resulting in reduced memory usage and faster convergence.

3.2 Bidirectional Transformer Encoder Results vs Baseline TDS Conv Encoder

As shown in Figure 1a and Figure 1b in the Appendix, the larger transformer, Transformer Med, overfit during training and fails to generalize despite using weight decay and the data augmentation strategies mentioned above. In fields like computer vision, transformer-based architectures have been empirically shown to require much more data to scale in comparison to CNN-based architectures because of their lack of inductive biases that the CNN has, namely the fixed receptive field and kernels of CNN based models, but ultimately scale much better on greater amounts of data [3]. Given that we are training on only a single user's data from a random initialization, it is not entirely surprising to see that a relatively large transformer model ($\sim$ 68.2 M parameters) would overfit. It is possible that a larger model would be more appropriate for larger scale data regimes such as the entire dataset of multiple users, but our smaller transformers have demonstrated good performance and stable training, suggesting that at least for the task of training on a single user's data, a smaller architecture could be sufficient.

Notably, our smaller transformers (Transformer Small and RoFormer Small) both beat the baseline TDS Conv module's validation loss and CER significantly (Table 1), and demonstrate a significant improvement upon the original paper's observed val CER of 15.65 ± 5.95 for training their TDS Conv model from random initialization with no Language Model [11]. For our particular experiments, despite using their exact hyperparameters, we do observe that our Val CER achieved by the TDS Conv module is higher than their observed average but it is still within their observed interval, and can possibly be chalked up to the particular user selected to train on.

3.3 Bidirectional Transformers on longer testing sequences

For our transformer models, our observed test metrics marked with "*" are collected from testing data that was split into windows that matched the window sizes of the training data. This is because both transformer models exhibited poor generalization to the much longer testing data (achieving 100 test CER on the entire unsplit test session) as the models are trained on the sessions that were split into 4 second windows by default. This is fairly reasonable since the models technically have receptive fields of the full sequence due to the attention mechanism but have not seen positional embeddings beyond those in the much shorter training sequence lengths and perhaps do not have queries and keys suitable to these unseen positional embeddings. We observed this first with the

absolute sinusoidal position embeddings and were thus motivated to experiment with rotary positional embeddings, due to their theoretical capability to formulate positional embeddings such that matching relative distances will have an equivalent relative rotational difference in query and key vectors to improve generalization of positional "understanding". However, we hypothesize that the model is still not trained to handle longer sequences that may unintentionally add more information to the residual stream than expected for redistribution sequence lengths or to make use of relative positional embeddings beyond the trained-on sequence lengths. That is, perhaps information from significantly further in the past or the future that may overwhelm the residual stream as again the unsliced testing data has many more input elements and is very out of distribution. Thus, both of these models have 100 testing CER on the unsplit testing sequences.

We experimented on further training on a doubled window (8 second sequences) to test the impact of this on the RoFormer model's performance on the unsplit testing sequence, but observed similar val and split test sequence performance and no improvement on the unsplit testing sequences (Table 5 in the Appendix).

We hypothesize that further experiments training on larger window sizes could potentially mitigate this, or even fine-tuning our current models on longer sequence lengths could help improve this performance but due to the quadratic scaling of the attention mechanism in compute and memory requirements, this is not within the scope of our work. Another concern would be that for a single user, there may also be insufficient unique sequences in training data with unsplit / larger training session windows.

Despite this, we still can test our hypothesis of whether having a larger receptive field is detrimental to generalization and performance, which was one potential justification for using a TDS Conv architecture that intrinsically has a limited receptive field. Since our receptive fields of 4 or 8 seconds (matching input sequence size) are large enough to carry context that are not immediately relevant to the current key press and not part of "co-articulation" of the previous and next key presses; rather, they potentially include additional information about the user's sEMG history and which could inform the model of their typing speed, style, tendencies, and so on. The superior performance of transformers and RNNs over TDS ConvNet suggests that flexible receptive fields enhance decoding by capturing long-range dependencies, though generalization to longer sequences remains limited by training sequence length. The superior performance of transformers and RNNs over the TDS ConvNet suggest that flexible receptive fields enhance decoding by capturing long-range dependencies and effectively filtering out irrelevant information. The model also does not qualitatively appear to simply be modelling language as the transformer exhibits the ability to correctly identify typos and backspaces in unseen data, though generalization to longer sequences remains limited by training sequence length.

3.4 Data Augmentation on Hop Length / Sampling Frequency

In our experiments with increasing the hop lengths (decreasing the sampling rate) from the baseline 16 hops (2kHz downsampled to 125 Hz Log Spectrograms), we find that performance of our RoFormer Small is roughly maintained up to a sample frequency of 50 Hz (Table 4) from which performance degrades at the next sample frequency of 25 Hz.

These results align with what we see in an extension of these experiments to recurrent architectures. When the sampling frequency is maintained around 50 Hz, the validation CERs remain comparable to those obtained at our baseline 125 Hz configuration, and such robustness was demonstrated in both vanilla and residual variants of GRU and LSTM models. However, when decreased to 25 Hz, a notable performance degradation is achieved across all RNN configurations.

This suggests that we can sample at a frequency lower than in the baseline, and capture longer time intervals within a shorter sequence length, allowing for faster inference and scaling up the training of our models at a smaller computational and memory cost without significantly impacting performance, up to a certain critical threshold. Additionally, this could significantly improve the inference time of sequence processing models as we can effectively half the sequence length, which results in a rough estimate of a 4x speedup for operations that scale quadratically with sequence length like attention. This is especially impactful in the context of practically using this technology for real-time typing applications and on-device deployment.

References

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[4] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376, 2006.

[5] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert. Sequence-to-sequence speech recognition with time-depth separable convolutions, 2019. URL https://arxiv.org/abs/1904.02619.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[9] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition, September 2019. URL http://dx.doi.org/10.21437/Interspeech.2019-2680. In Interspeech 2019.

[10] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.

[11] 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. URL https://arxiv.org/abs/2410.20081.

[12] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

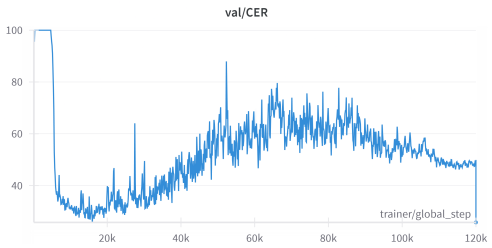
[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

4 Appendix

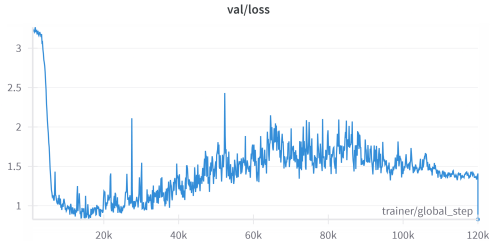
Table 5: Training Data Sequence Length / Window Length for Transformer Experiments

Model	Window Size	Val Loss	Val CER (%)	Test Loss	Test CER (%)
RoFormer Small	4 sec	0.83	12.23	1.32*	**17.52***
RoFormer Small	8 sec	0.81	**12.05**	1.55*	20.64*

*Test metrics from sliced windows evaluation



(a) Validation CER



(b) Validation Loss (early steps excluded due to high CTC loss)

Figure 1: Training metrics for Transformer Med model exhibiting overfitting. The validation loss plot excludes early training steps due to high CTC loss with random initialization.

Table 6: Regular RNN Models: Hyperparameters and Performance

Unit	MLP	HS	Layers	Dropout	GSteps	Train CER	Val CER
GRU	384	256	2	0.20	120K	9.10	**16.88**
GRU	512	512	4	0.30	60K	8.32	19.07
LSTM	512	256	4	0.40	30K	17.89	19.30
GRU	384	256	2	0.20	60K	21.21	20.80
GRU	512	384	3	0.30	20K	22.63	23.08
GRU	384	256	2	0.20	18K	28.65	25.72
GRU	512	512	4	0.30	20K	26.44	26.65

Table 7: Residual RNN Models: Hyperparameters and Performance

Unit	MLP	HS	Layers	Dropout	Steps	Train CER	Val CER
GRU	512	512	4	0.30	30K	10.82	**16.79**
LSTM	512	256	4	0.40	30K	19.96	19.52
GRU	384	256	2	0.20	30K	21.91	21.47
GRU	512	512	4	0.30	9K	24.44	22.46
GRU	384	256	2	0.20	9.1K	34.02	30.66
GRU	384	256	2	0.20	9.7K	32.85	31.30