Predicting Keystrokes from Electromyography Signals

Ayush Agrawal
Dept. of Mechanical Engineering, UCLA
Los Angeles, CA
ayushagrawal26@ucla.edu

Lakshman Sundaram
Dept. of Computer Science, UCLA
Los Angeles, CA
lakshman.sun27@gmail.com

Premkumar Sivakumar
Dept. of Mechanical Engineering, UCLA
Los Angeles, CA
premkumar22@ucla.edu

Zhirong Lu
Dept. of Computer Science, UCLA
Los Angeles, CA
zhironglu23@g.ucla.edu

Abstract

In this technical report, we outline the choice of deep learning models with various architectures to predict keystrokes from electromyography signals from subject 89335547 in emg2qwerty dataset. In order to beat the performance of the baseline model (MLP+CNN), we define individual BiLSTM[1] introduced by Sak et al. [2014], GRU [2] introduced by Cho et al. [2014], and Transformer layers introduced byVaswani et al. [2017] using PyTorch to build more capable deep learning architectures. We stack these layers in a particular arrangement based on empirical and logical reasoning. Later, we pick the best performing model and run it for different number of electrode channels to find if the model performs well if each band has electrode channels ≤ 16.

1 Introduction

Building efficient deep learning architecture requires understanding the unique purpose each layer serves. This motivates us to first look into how CNN, RNN, and Transformer layer manipulate the data. Based on our reading from different sources online, we summarize the comparison of CNN, RNN, and Transformer based on **data processing**, **weakness** and **strengths**.

Model	**Data Processing**	**Strengths**	**Weaknesses**
BiLSTM	Sequential processing (remembers past inputs)	Good for long-term dependencies in sequential data	Slow, sequential processing limits scalability
GRU	Similar to BiLSTM, but has fewer parameters	Faster than BiLSTM, retains long-term dependencies well	Less expressive than BiLSTM
Transformer	Processes entire sequence in parallel using self-attention	Handles long-range dependencies efficiently, highly parallelizable	High computational cost, data-hungry
CNN	Processes data in local patches (spatial features)	Good for spatial feature extraction, and is fast	Does not inherently model sequential dependencies

Table 1: Comparison Summary of BiLSTM, GRU, Transformer, and CNN

[1] BiLSTM is a type of RNN, which stands for Bidirectional Long Short Term Memory
[2] GRU is another type of RNN, which stands for Gated Recurrent Unit

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

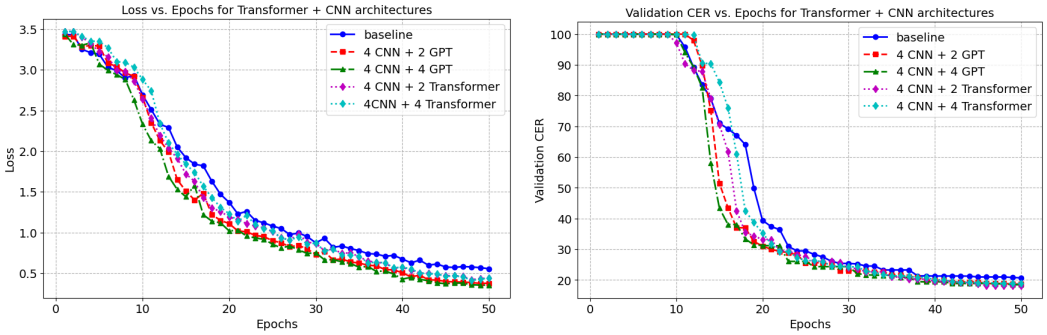


Figure 1: Loss and CER VS Epoch for various CNN + Transformer architectures

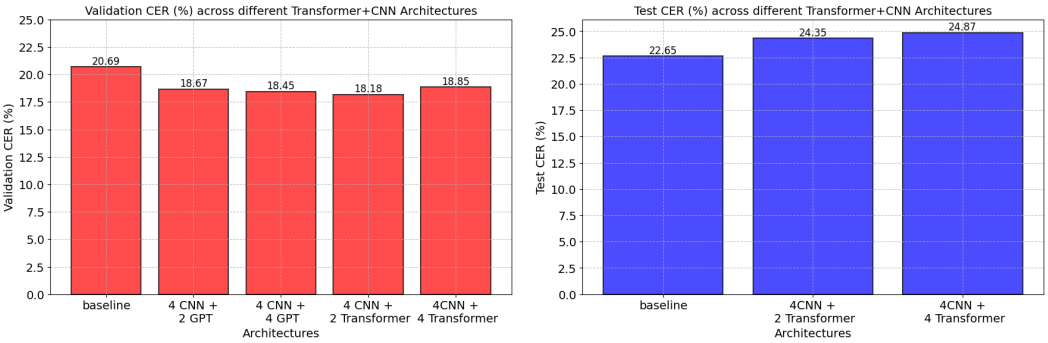


Figure 2: Comparison of final validation and test CER of various CNN + Transformer architectures

2 Methods

For this project, we experimented with several architectures to analyze their effectiveness in processing the *emg2qwerty* data of one subject. When designing each model, we decided to play around with combinations of different architectures that we had learned about in class and elsewhere. The goal was to design them to capture different aspects of the data while balancing computational efficiency and model expressiveness. All models were implemented in Pytorch. We trained the models for **50** epochs and compared their performance at the end of the training periods.

2.1 CNN + Transformer

We know that transformers can capture long-range dependencies in global context. Since the keystrokes are in natural language, we believe borrowing an architecture good at language processing would be very helpful. Thus, in the first architecture we tested, we added a transformer block in the very end of the baseline model, after the Spectrogram Normalization (S-Norm), Rotation-Invariant MLP (RI-MLP), and convolution (CNN) layers.

S-Norm → RI-MLP → 4 conv layers → N Transformer layers → 1 FC layer → Softmax

We also tried other hyperparameters, such as 2, 6, or 8 convolution layers, and none of them would perform as well as 4 layers, except that 2-layer transformer has similar performance as 4-layer ones.

2.2 GP2 Integration

Since we cannot get more powerful transformer networks through training, loading a pretrained transformer became a great alternative. We can view the signals as a special language, so a LLM might be good at processing it. We loaded the hidden layers of the pre-trained GPT2 model introduced by Radford et al. [2019] through Huggingface to replace the transformer layers mentioned in the previous section, and "fine-tuned" it. Because of the limitation in computing power, we chose the smallest 12-hidden-layer version of GPT2. Here is a graph of the architecture:

S-Norm → RI-MLP → 4 conv layers → GPT2 small → 1 FC layer → Softmax

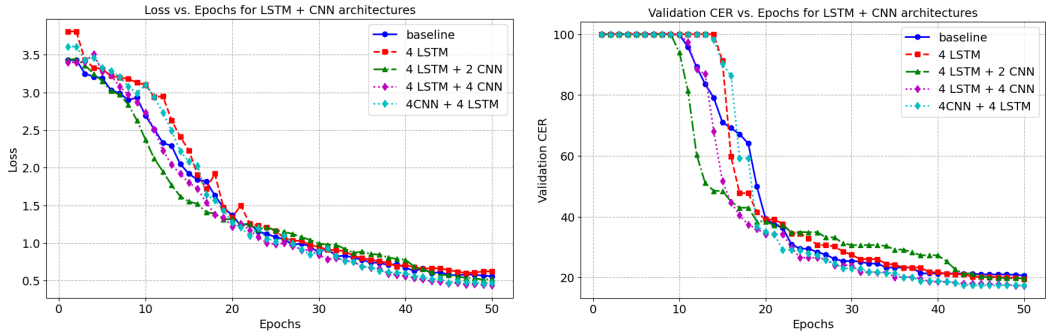


Figure 3: Loss and CER VS Epoch for various LSTM + CNN architectures

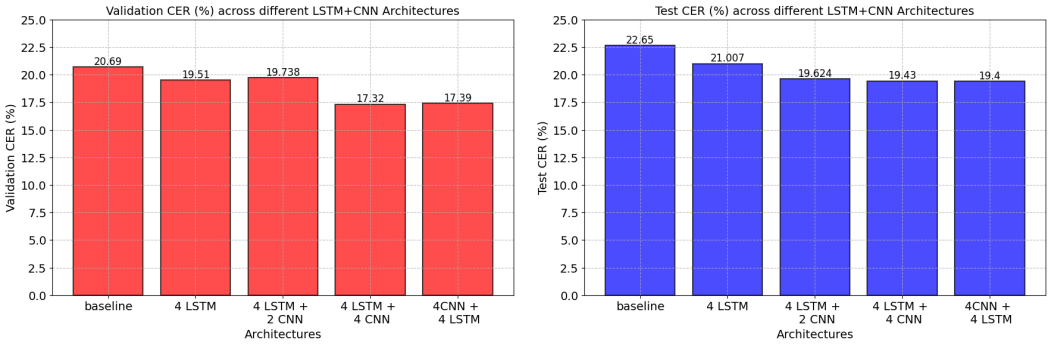


Figure 4: Comparison of final validation and test CER of various LSTM + CNN architectures

At first, we set the first 2 and last 2 of the hidden layers trainable. However, the new model performs at a similar level as naive CNN+transformers. So we tried to only let the first 1 and last 1 layer be trainable, however, the model still does not perform much better(see figure **??**).

Considering the huge cost of an LLM and the small boost in performance, we decided to abandon the plan of integrating a pretrained LLM. Instead, we tried to combine multiple architectures. Also, when we combine transformer and other models, we will limit the number of trainable layer of transformer to 2.

2.3 Long short term memory (BiLSTM) layer

In the baseline model, we observed that none of the layers (S-Norm, RI-MLP, and CNN) are capable of processing sequential data. The sequential nature in *emg2qwerty* data come from its temporal component. To process this component of data effectively, we utilized BiLSTM in conjunction with CNN layers to build the following architectures.

1. S-Norm → RI-MLP → 4 layers of BiLSTM → 1 FC Layer → Softmax
2. S-Norm → RI-MLP → 4 layers of BiLSTM → 2 layers of CNN → 1 FC Layer → Softmax
3. S-Norm → RI-MLP → 4 layers of BiLSTM → 4 layers of CNN → 1 FC Layer → Softmax
4. S-Norm → RI-MLP → 4 layers of CNN → 4 layers of BiLSTM → 1 FC Layer → Softmax

2.4 GRU

2.4.1 Pure GRU

This architecture replaces the baseline convolution encoder with a Gated Recurrent Unit (GRU) network. The components before the GRU (Spectrogram Normalization and Rotation-Invariant MLP) and after the GRU (Linear Layer and Softmax Loss) remained the same.

GRUs are expected to perform better than BiLSTMs in certain scenarios due to their simpler structure and fewer parameters. GRUs use only two gates (reset and update) compared to BiLSTM's three

gates (input, output, and forget), making them computationally more efficient. This simplicity allows GRUs to train faster and potentially perform better on smaller datasets.

The reduced number of parameters in GRUs also makes them less prone to overfitting, especially when working with limited data. This is particularly relevant for our project, where we worked with a constrained dataset from a single subject. We anticipate that the GRU model will show a smaller gap between training and testing performance compared to more complex architectures like BiLSTMs.

In general, GRUs can be thought of as a simpler, faster variant of BiLSTM models. They offer a good balance between computational efficiency and the ability to capture long-term dependencies in sequential data. This makes them an attractive option for our EMG-based keystroke prediction task, as we need to process temporal information efficiently while avoiding overfitting.

2.4.2 GRU + CNN

After observing the improved performance of the BiLSTM + CNN model and noting the advantages of the pure GRU architecture, we decided to explore a GRU + CNN hybrid model. The combination of BiLSTM and CNN layers showed better results than either architecture alone, suggesting that a similar hybrid approach with GRU might yield positive outcomes.

2.5 Deeper architectures

2.5.1 Transformer + GRU + Conv

Since both the pure BiLSTM/GRU model and the BiLSTM/GRU + CNN model showed promising results, we decided to take it a step further and add a third layer to see if we could reduce the CER even more. To do this, we structured our model by stacking four GRU layers between a two-layer transformer and 4-layer convolutional encoders, i.e., S-Norm $\rightarrow$ RI-MLP $\rightarrow$ 2-layers Transformer $\rightarrow$ 4-layers GRU/BiLSTM $\rightarrow$ 4-layer CNN $\rightarrow$ 1 FC Layer $\rightarrow$ Softmax.

This specific arrangement takes advantage of the strengths of each architecture. The transformer comes first to capture long-range dependencies through self-attention before the sequential processing

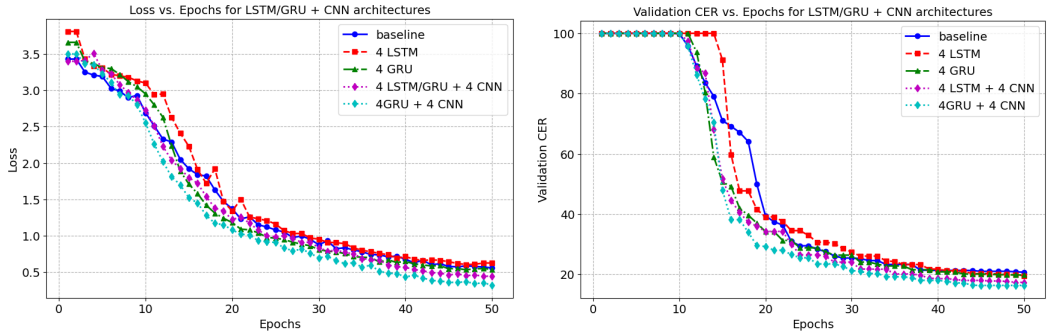


Figure 5: Training loss & Validation CER for LSTM/GRU + CNN architectures

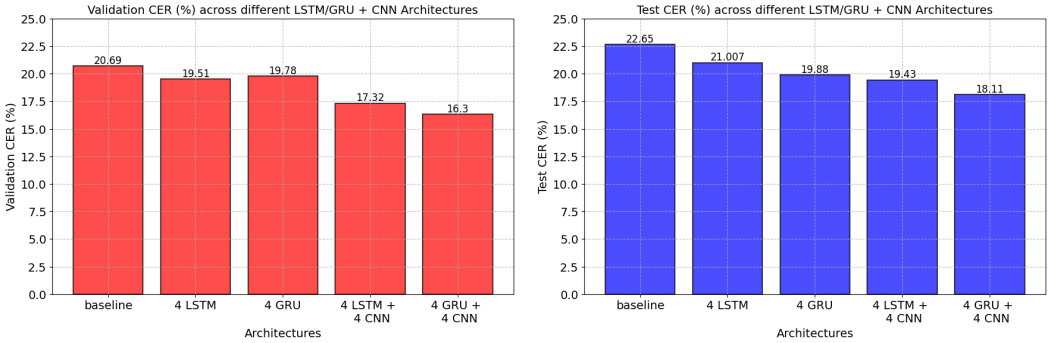


Figure 6: Validation & test CER for LSTM/GRU + CNN architectures

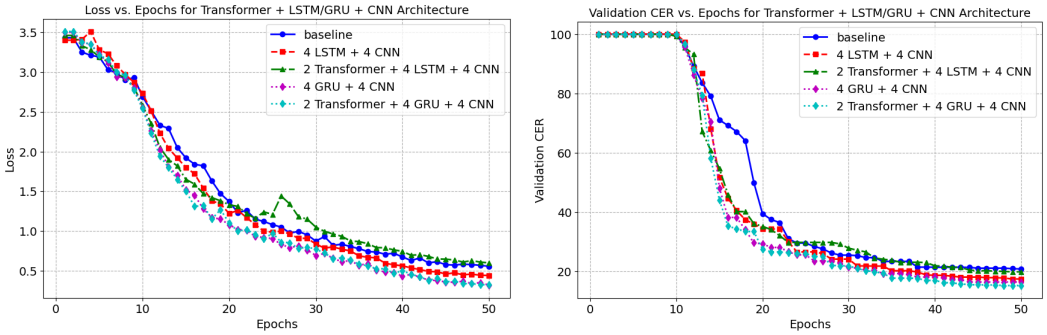


Figure 7: Training loss & Validation CER for Transformer + LSTM/GRU + CNN architectures

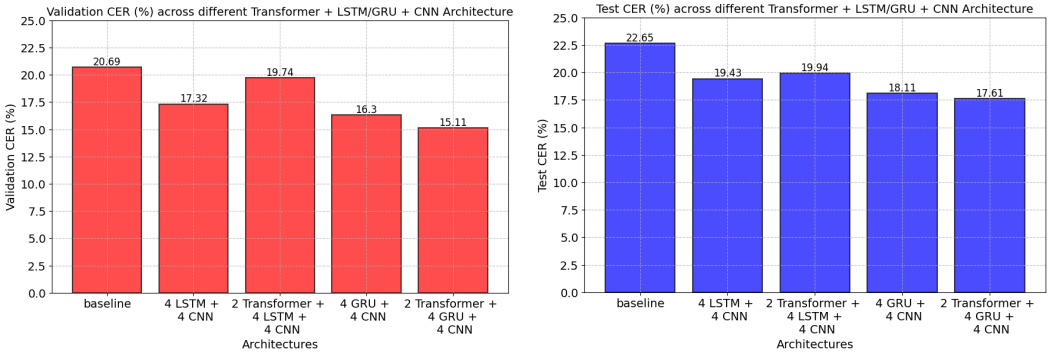


Figure 8: Validation & test CER for Transformer + LSTM/GRU + CNN architectures

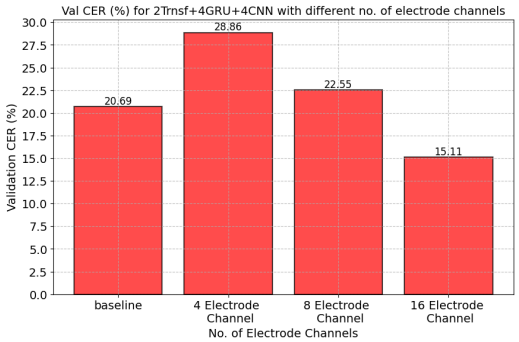
stage. Then, the BiLSTM/GRU refines these high-level temporal features while maintaining long-term memory. Notably, GRU offers a more efficient approach to handling temporal features compared to BiLSTM due to its simpler structure.

Finally, the convolutional layers extract spatial features, ensuring that the model integrates information across all electrode channels before classification. By applying concepts learned in class and experimenting with different layer orderings, we found that this structure delivered impressive results.

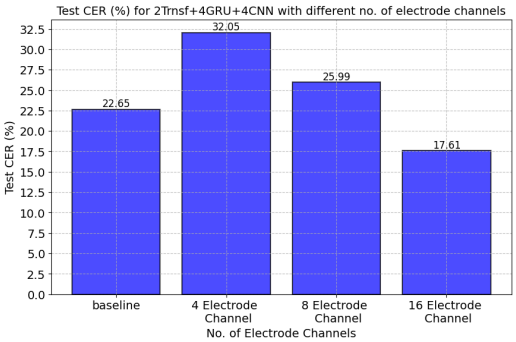
2.6 Relationship between the number of electrode channels and CER

We also wanted to explore how the number of electrode channels affects model performance. To do this, we trained our best-performing model— $2\times$Transformer + $4\times$GRU + $4\times$Convolution — on single patient dataset, but each with a different number of electrode channels, i.e., 4, 8, and 16. Since electrode channels are independent of each other, reducing their number effectively limits the available information. Our results showed a clear drop in performance (CER) as the number of channels decreased. This suggests that each channel provides unique information, and having more channels increases data diversity, ultimately improving model performance.

The following plots support the above observation:



(a) Val CER (%) for electrode channels = 4, 8, 16



(b) Test CER (%) for electrode channels = 4, 8, 16

Figure 9: Comparison of validation & test CER for Transformer + GRU + CNN architectures with number of electrodes = 4, 8, 16

3 Results

As a benchmark, the baseline model achieved a validation CER of 20.69% after trained of 50 epoch.

For transformers,

- the best architecture is 4-ConvEncoder(24 channels) + 2-transformer(8 heads, 512 feedforward dimensions), with a validation CER of **18.18%**
- best architecture using conv+GPT2(small) has similar performance with naive conv+transformer, but costs a lot more to train.

For LSTM+CNN architectures, we can make the following observations from figures 3, 4

- Out of the four architectures defined in subsection 2.3, 4-layer LSTM + 4-layer ConvEncoder performed the best with validation CER of **17.39%** and training CER of **19.4%**.
- On increasing the number of ConvEncoder layers from 2 $\rightarrow$ 4, the validation CER decreased from **19.74%** to **17.31%**
- 4-layer LSTM + 4-layer ConvEncoder achieved the least training loss of value **0.441**.

In subsection 2.4, we compared two well know RNN architectures namely - LSTM, GRU. We can draw following inferences from figures 5, 9.

- 4-layer GRU + 4-layer ConvEncoder beats all other models including the 4-layer LSTM + 4-layer ConvEncoder in figure 9. It achieves a validation CER of **16.3%**.
- 4-layer GRU + 4-layer ConvEncoder achieves a training loss of **0.32**.

For deeper architectures(see figure 7),

- adding transformers at the very start improves performance for GRU+CNN (CER drop from 16.30% to 15.11%)
- adding transformers at the very beginning harms performance for LSTM+CNN (CER rise from 17.32% to 19.74%)
- overall, 2-layer transformer + 4-layer GRU + 4-layer CNN performs the best with validation CER of **15.11%** and test CER of **17.63%** - a significant leap over the baseline model.

Note: **(2-layer transformer + 4-layer GRU + 4-layer CNN)** is our best performing model.

4 Discussion

4.1 Transformers

For transformer, adding more layers above 2 layers would not improve performance because transformers are very data-hungry (requires lots of training data to perform well). When we have too many layers, we do not have enough data to train the model properly. Or maybe we reached the upper performance limit of the architecture, such that no matter how many layers we add, the performance cannot be better.

GPT2 does not perform much better than naive transformers probably because it was trained on text data, which is totally different from electromyography signals we are processing. Also, these layers might be too shallow to fully "encode" the complex patterns in the input signal. However, if we increase the number of trainable parameters, the model would become too large to train. GPT2 has too many parameters for our data, about 5 times as much as any other model we have.

4.2 GRU & LSTM

The pure GRU model performed as we hypothesized. There was only a 0.10% difference between validation and test CER in this model. In contrast, other models, including the baseline and BiLSTM variants, showed more substantial increases in CER from validation to test data, typically around 2%. The GRU's ability to maintain performance across validation and test sets can be attributed to its simpler structure and fewer parameters compared to BiLSTM, making it less prone to overfitting, especially with our limited dataset. Additionally, we observed that GRU-based models tended to train slightly faster than their BiLSTM counterparts, likely due to their computational efficiency. While this speed difference might be negligible for smaller datasets, it became noticeable with our large training set, further highlighting the practical advantages of GRU in this context.

4.3 Why Transformer $\rightarrow$ GRU $\rightarrow$ CNN turned out best for sEMG Keystroke Prediction?

The Transformer $\rightarrow$ GRU $\rightarrow$ CNN architecture is ideal for sEMG-based keystroke prediction because it effectively captures both temporal and spatial dependencies. The Transformer, placed first, models long-range dependencies in sEMG signals using self-attention, ensuring a global understanding of muscle activation patterns. Next, the GRU refines these temporal features, efficiently filtering noise and retaining key sequential patterns without excessive computational overhead. Finally, the CNN extracts spatial features from electrode data, leveraging the structured temporal representations provided by the previous layers. This order ensures optimal temporal modeling, noise reduction, and spatial feature extraction, leading to highly accurate keystroke predictions.

In this architectire, GRU turned out better than LSTM because GRU prioritizes efficiency and simplicity. It has fewer parameters (no separate cell state) and is computationally faster, making it ideal for real-time sEMG processing. GRUs also retain long-term dependencies well while avoiding vanishing gradients, performing comparably to LSTMs but with lower memory and faster convergence.

4.4 Other Details in Implementations

Our exploration of various model architectures provided valuable insights into the performance of different neural network structures for EMG-based keystroke prediction. While we initially considered exploring various data augmentation techniques, we ran a few simulations with random cropping (guided by the helper video and TA office hours) that led to a significant decrease in the baseline model's performance (exact data isn't available because we ran out of GPU credits to re-run and save).

This unexpected result led us to reconsider the necessity of data augmentation for this particular task. Given the large size of the *emg2qwerty* dataset, we think that the abundance of training data already provides sufficient variability and examples for the models to effectively learn. In this context, we hypothesize that artificial augmentation may introduce unnecessary noise or distortions that hinder rather than help the learning process. It might also be hard to do data augmentation since we have to carefully model the way human type. Simple tricks like reversing the signal would not be helpful since nobody would type backwards.

References

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Hasim Sak, Oriol Vinyals, and Georg Heigold. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

link to code: `https://drive.google.com/file/d/1BYVRTGhAtJwE6jk_D2lx31MQVTQadbr0/view?usp=sharing`