Breanna Burd
ChatpGPT from Scratch - EN.705.743.8VL.FA25
Final Project

ABSTRACT

Position embeddings are used within large language models (LLMs) to provide context about the order or words within a sentence and which words usually appear near one another. In this paper, we focus on how changing the number of transformer layers within a transformer-based model affects the learning rate of the LLM by comparing models trained using absolute position embeddings to models trained using Rotary Position Embeddings (RoPE). RoPE applies the absolute position within a rotation matrix. This adaptation allows the model to recognize sentences with similar word sequences appearing at varying parts of the sentence to be identified as similar. My experiments show that despite the number of layers within each transformer block of the model, the absolute position embedding trains faster at the beginning.

1 Introduction

Languages must speak words in specific orders to retain the meaning of the sentence. If two models are trained with “The dog chased the cat” and “The cat chased the dog” without position embeddings, the result would be equivalent. Absolute position embeddings were the first embedding technique implemented and they perform relatively well for the general case. Absolute position embeddings have some limitations such as having a maximum sequence length defined during training. This limitation makes it nearly impossible for a model to perform well on sequences larger than the specified sequence length because the model has not been trained on positional information. Another limitation is that sentences with similar sequences may not be identified as similar when using absolute position embeddings. An example would be “The lion slept deeply” compared to “At the end of the day, the lion slept deeply”. It is clear to English-speaking humans that these sentences have the same general meaning, but absolute position embeddings would disagree with that statement.

The RoPE implementation is meant to solve both of the absolute position embedding limitations listed above. Rotary position embeddings can be used within a rotation matrix, allowing the model to learn information about how words relate to the other words around it. Thus, learning the meaning of the word. RoPE is also able to adapt to larger sequences after training due to its nature of applying the sequence within a rotation matrix. These experiments compare a base model, which is a GPT model that uses absolute position embeddings, to a modified version that uses RoPE. I implemented the changes needed to modify the existing base model by updating each layer within the transformer block to calculate the rotary position embedding.

Based on the current research and existing LLMs that implement RoPE into the training process, I expected the RoPE trained models to have a significantly lower model loss throughout training and that the more layers within the transformer block would significantly decrease loss as well. The results and findings from my experiments showed that the base model actually performed better in every iteration, but that the number of layers within the transformer block did show a slight improvement in the model loss.

2 Background

When using absolute position embeddings, the position is applied only once in each iteration, but when using RoPE, the rotational position matrix is applied in each layer of the transformer block. The multi-head attention (MHA) layer needs to be modified to support RoPE. The MHA layer is where the model is able to learn multiple relationships for words to other words. After splitting the input Tensor into queries, keys, and values, RoPE is applied. Additionally, the absolute position embeddings were removed from the model so that the model only includes a single position embedding.

Some related works that could provide additional information regarding the importance and mathematical explanations for position embeddings include the RoFormer paper and the Hugging Face article. Both of these resources cited below have been used to implement the changes to the model, create the experiments, and expand to future research options.

3 Methods

The goal of this experiment was to show how the RoPE embeddings affect a GPT model's loss as compared to a GPT model that uses absolute position embeddings. The '`wikitext-103-v1`' dataset provided by Hugging Face was the only dataset used for these experiments. This dataset was used to create a custom training dataset that applies a tokenization, packs the sequences to contain the maximum sequence length (in this case 256), and shuffles the sequences. This dataset was created prior to training in an earlier module of the course. This tokenized dataset was then fed into the model in batches and model loss is calculated for each step.

Each model is trained using the following various number of transformer block layers: 2, 4, 8, and 12. For each case, the loss curve was produced frequently using the total number of tokens trained as the independent variable. Another metric that was looked at is the amount of time needed to train 1,000 steps of the model. This metric is important to monitor because it will greatly affect the overall training time needed to fully train each model. The compute time information can be used to determine if the complete model loss is improved enough to justify the additional training time.

4 Results

The RoPE models (denoted by 'FinalProject' within Figures 1 and 2) are much more stable and have less noise than the base model. All experiments show a sharp learning curve at the beginning of the training with a significant decrease in loss after only a few million tokens are trained. **Figure 1** shows the raw loss curves (without smoothing) so that the noise is visible. Each point in the plots defines 1,000 steps trained within the model.

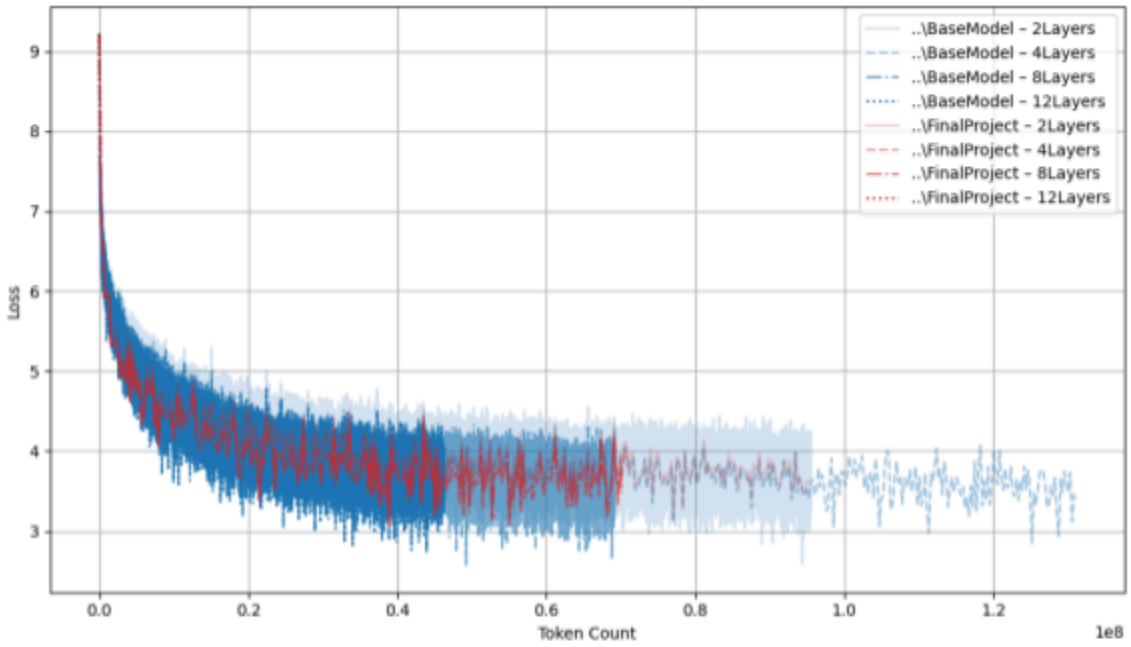

Figure 1
Plot of Token Count vs Loss for each iteration of the experiment.

Figure 2 shows the smoothed loss curves after applying exponential moving average (EMA). Since the RoPE results are much more stable, an alpha of 0.2 was used to smooth the loss curves as compared to the base models that I applied an alpha of 0.01 to apply more smoothing. After applying smoothing, it is displayed that the 8-layer and 12-layer models have slightly lower model loss as the loss curve moves towards convergence. It is also shown that the RoPE 12-layer model consistently has the smallest model loss (performs the best).

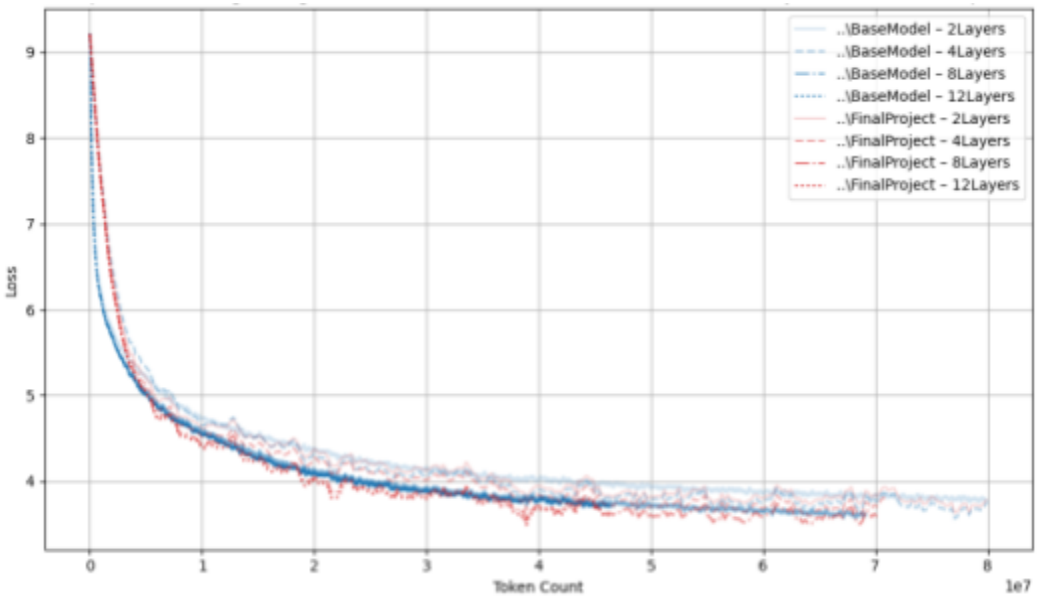

Figure 2
Smoothed plot of Token Count vs Loss using Exponential Moving Average (EMA).

The training time for both models is consistent for all experiments. The amount of time needed to train 1,000 steps is roughly the same for absolute position embeddings and rotary position embeddings.

Number of Layers	Base GPT Model (absolute position embeddings)	RoPE GPT Model
2	0.7443	0.7303
4	0.9449	0.9759
8	1.6686	1.6881
12	2.4168	2.6053

Table 1
Time needed to train each model 1,000 steps in seconds.

5 Conclusions

I did not get the results that I had expected from the experiments. I learned that absolute position embeddings models have noisier loss curves than the models that implemented RoPE. This was shown in **Figure 1**. The smoothed curves in **Figure 2** show that all experiments produced very similar model loss despite using various numbers of transformer block layers. Despite the 12-layer RoPE model having a consistently smaller model loss as compared to all of the other experiments, I was unable to justify the training time needed because of my lack of compute resources available to me.

I faced a major obstacle along the way because it took a very long time to train each experiment without any GPUs. I ended up not being able to train each model to completion and ended the model training early for most of the experiments. This means that I was not able to acquire information about the ending model loss for a fully trained model using RoPE.

6 Impact and Future Work

Due to compute resources, it was not achievable to perform any additional experiments. In the future, it would be important to perform similar experiments with larger sequences. Based on existing research for the RoPE embeddings, a larger sequence would greatly improve the model's loss curve. This is exceedingly important to note that absolute position embeddings require that the sequences provided are never longer than the maximum sequence allowed. The absolute position embeddings will trim the excess of the sequence, meaning it is never passed into the model. On the other hand, RoPE can support any sequence size. Within the lecture notes provided throughout the course, it was stated that "RoPE models consistently outperform previous models by a few % better loss", but I was unable to train each of the models to completion and was unable to verify that statement. In the future, I would hope to have the available compute power and time to train each of the experiments to determine if the final training loss is substantially smaller than the loss for the absolute position embedding base model. These experiments did not produce the results that I had expected, but I believe most of the fallbacks of these experiments were because each of the models were trained using a maximum sequence length of 256. Based on existing research on this topic, RoPE performs much better on longer sequences due to its ability to use context within the sequences.

It is not clear how these results would scale for me to say with certainty. I believe that if these maximum sequence length was increased significantly, to 4096 or above, there would be more interesting results within the model loss of these experiments. Since the results did not show a significant slow-down for the training time when using RoPE, it would make sense for large-scale models to default to use RoPE due to the other research that has been presented elsewhere that shows that RoPE performs better that absolute position embeddings and does not have any limitations. If I had additional resources of thousands of GPUs, I would also want to test varying maximum sequence lengths to study how the sequence length affects model performance. Additionally, I would like to formulate a better training dataset. The dataset used for training the models in this experiment are very small and limited, thus the model is unable to produce many meaningful results. Microsoft's Phi LLMs are trained on only high quality data, thus the model is able to be trained on fewer parameters and produce better results than many of the other popular LLMs that have billions of more parameters. Building a high quality training dataset may be the most critical piece for decreasing model loss.

References

Aju. (n.d.). *RoPE-PyTorch/RoPE.ipynb at main · aju22/RoPE-PyTorch*. GitHub. https://github.com/aju22/RoPE-PyTorch/blob/main/RoPE.ipynb

Doshi, K. (2025, February 26). *Transformers Explained Visually (Part 3): Multi-head Attention, deep dive*. Towards Data Science. https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853/

Fleetwood, Christopher. *You could have designed state of the art positional encoding*. (2001, February 4). https://huggingface.co/blog/designing-positional-encoding

Mehdi Hosseini Moghadam. (2025, June 17). 🔥 *Master RoPE (Rotary Positional Encoding) - The SECRET Behind GPT & LLaMA's Success! Code and math* [Video]. YouTube. https://www.youtube.com/watch?v=Mwyj1DmduCM

Salesforce/wikitext · Datasets at Hugging Face. (2022, March 30). https://huggingface.co/datasets/Salesforce/wikitext

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., & Liu, Y. (2023b). RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, *568*, 127063. https://arxiv.org/pdf/2104.09864

Tam, Adrian. *Positional Encodings in Transformer Models*. (2025, September 12).Machine Learning Mastery. https://machinelearningmastery.com/positional-encodings-in-transformer-models/

Thor, W. (n.d.). *Limitations of absolute positional encodings*. https://apxml.com/courses/how-to-build-a-large-language-model/chapter-13-positional-encoding-variations/limitations-absolute-positional-encodings