CSE256: PA2

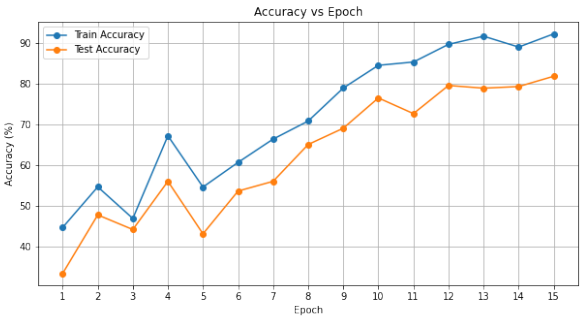
Samyak Mehta

May 17, 2024

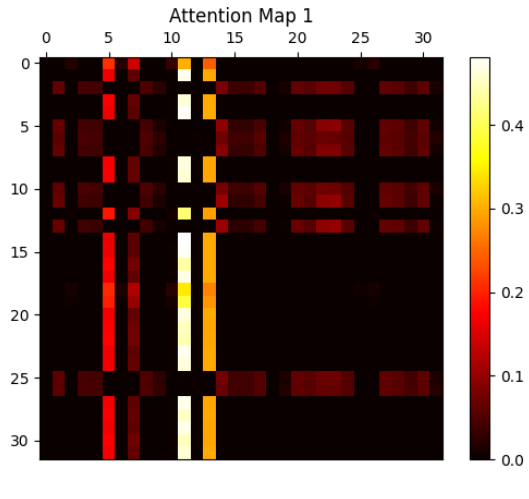
1 Part1: Encoder

Table 1: Training and Test Results for Encoder Model

Epoch	Training Loss	Train Accuracy (%)	Test Accuracy (%)
1	1.0710	44.6463	33.33
2	0.9777	54.6367	47.73
3	0.9514	46.8451	44.13
4	0.9245	67.2084	56.00
5	0.9821	54.5889	43.07
6	0.9745	60.6597	53.60
7	0.9136	66.3958	56.00
8	0.8651	70.8413	65.07
9	0.8171	78.9675	69.07
10	0.7665	84.5602	76.53
11	0.7157	85.3728	72.67
12	0.6835	89.7228	79.60
13	0.6618	91.7304	78.93
14	0.6466	89.0535	79.33
15	0.6408	92.3040	81.87



(a) Accuracy vs Epoch



(b) Attention Map

Analysis: For this model we have Vocabulary size of 5755 and 0.576467M parameters. On running this model for 15 epochs we achieve an accuracy of 81.87 %.
The heat map was created for the last layer's last head. In the heat map we can see that the diagonal is almost zero showing that self-attention is minimal. We also see that tokens between the range 10 to 15 get maximum attention from all other tokens. But also the heat map does not make a lot of semantic sense. This is probably because the dataset is not too large.

2 Part2: Decoder

Table 2: Perplexity Results

Itr	Perp (Training)	Perp (Test - hbush)	Perp (Test - obama)	Perp (Test - wbush)
1	5766.5879	5779.4487	5780.8818	5792.0073
100	455.5193	578.6859	554.8329	645.2823
200	225.1104	402.8569	379.3434	460.7332
300	150.4885	385.8007	350.7429	457.4491
400	107.1659	385.3359	336.8222	454.4980
500	84.6778	409.0681	342.8661	492.9612

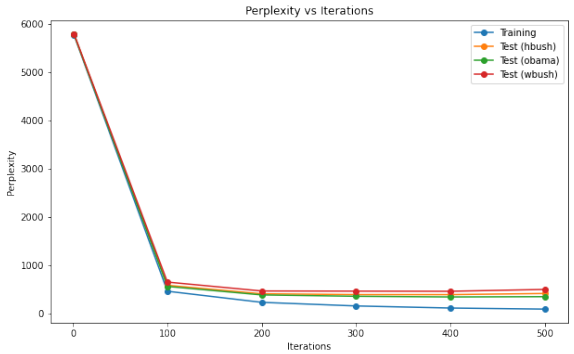
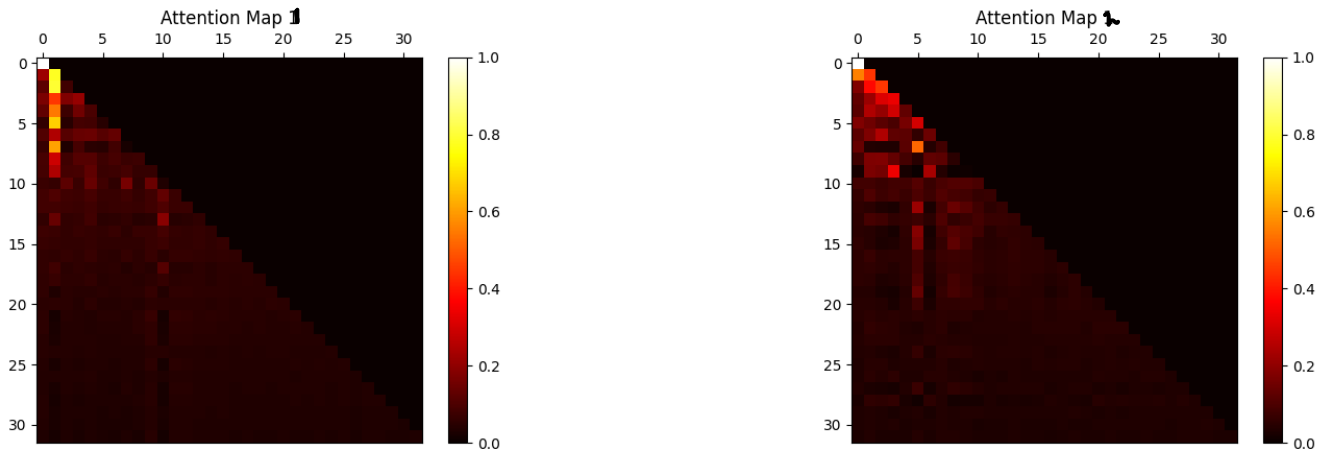


Figure 2: Accuracy vs Iterations



(a) Attention Map 1

(b) Attention Map 2

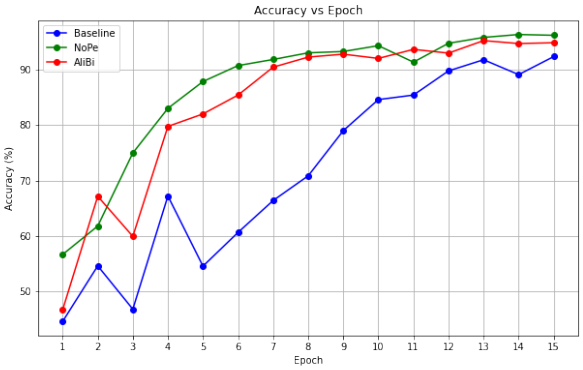
Figure 3: Attention Map for last two heads of last layer

Analysis: For this model I have Vocabulary size of 5755 and 0.943739M parameters.
For the language modelling task, the training perplexity came down to 84.6778 whereas test perplexity are in the 300-400 range for all three datasets. This is probably because when we are training it we are training it on all three presidents speech combined but when we are testing it we are testing on specific presidents. If training was done for a particular president and tested on that presidents speech then results could have been better.
The heatmap created for the last layer's last two heads is shown in 3. The upper triangle is black since we are masking it. The lower triangle seems equally distributed and is difficult to say if any token is paying attention to specific tokens. The diagonal is also dark red meaning self-attention is low too. The difference between the two attention maps is that the top right part of the first map has tokens 0-10 paying more attention to token 1-2 whereas in map 2 it becomes more evenly distributed.

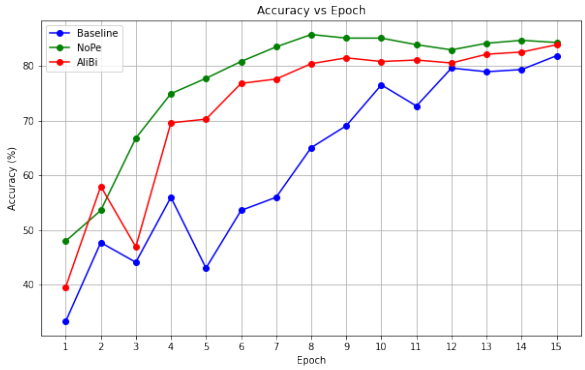
3 Part3A: Architecture Exploration

Table 3: Perplexity Final Results (Decoder Task)

Position Encoding	Perp (Training)	Perp (Test - hbush)	Perp (Test - obama)	Perp (Test - wbush)
Absolute	84.6778	409.0681	342.8661	492.9612
NoPe	80.2034	402.2962	330.2813	489.7674
AliBi	80.7541	396.9869	328.5650	476.2233



(a) Train Accuracy vs Epoch for Classification Task



(b) Test Accuracy vs Epoch for Classification Task

Model	**Final Train Accuracy (%)**	**Final Test Accuracy (%)**
Baseline	92.3040	81.87
NoPe	96.1281	84.27
AliBi	94.7897	83.87

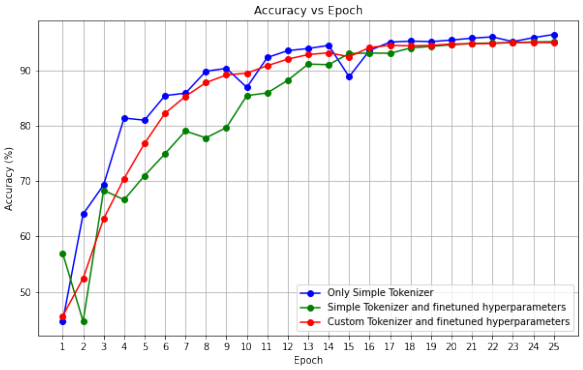
Table 4: Train and test accuracies on classification Task of different postion encoding models

I explored different types of positional embedding in the transformer architecture.

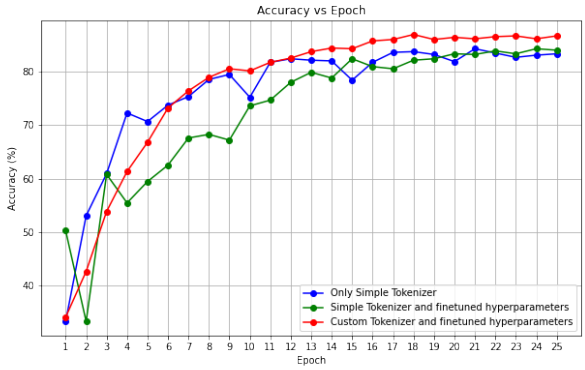
1. **Absolute Positional Embedding**: Traditional method in transformers, assigning unique positions to tokens for sequence understanding. Effective but struggles with longer sequences due to fixed encoding.
2. **NOPE (Non-Positional Embedding)** [1]: Removes positional information, focusing solely on token semantics. Offers robustness for tasks where sequence order is less relevant, like sentiment analysis, but may lack precision for tasks requiring sequential understanding.
3. **AliBi Embedding** [3]: Utilizes alias sampling to dynamically encode token positions, improving adaptability for varying sequence lengths. Enhances model's long-range dependency understanding.

Analysis: AliBi embeddings dynamically adapt to sequence lengths and effectively capture long-range dependencies, making them ideal for tasks like language modeling. Hence AliBi performs better than abosolute and NOPE in language modelling task as seen in the Table 3. Conversely, NOPE embeddings prioritize token semantics over positional information, benefiting tasks where sequence order matters less, such as classification. NOPE simplifies model architecture and avoids positional biases. Hence we see that NOPE performs slightly better than AliBi in Classification task as shown in Figure4a and Table 4.
We also see here that with AliBi and NOPE, the training and test accuracy curves show less deviation as epoch progresses, and converges to a value smoothly as compared to Absolute positional encoding. AliBi dynamically adjusts to varying sequence lengths, while NOPE disregards positional information. Both methods lead to more stable training and testing performance compared to absolute positional encoding.

4 Part3B: Performance Evaluation



(a) Train Accuracy vs Epoch



(b) Test Accuracy vs Epoch

Model	Final Train Accuracy (%)	Final Test Accuracy (%)
Baseline	96.5105	83.33
SimpleTokenizer and fine-tuned Hyper-parameters	95.2199	84.00
CustomTokenizer and fine-tuned Hyper-parameters	95.0765	86.67

Table 5: Final values of train and test accuracies for each tokenizer.

The three models have the following difference:

1. **Baseline**: This is the same model as the one in Part1, only difference is that it is run for 25 epochs.
2. **Simple Tokenizer with fine-tuned hyper-parameters**: This model uses the same tokenizer provided in the boiler plate code but involves changes to some hype-parameters obtained after fine-tuning. The following hyper parameters were changed: The training epochs were increased to 25 epochs as we also introduced a learning rate scheduler (STEP) which reduced as epochs progress to facilitate better learning. The transformer architecture was changed to have 8 layers, with each layer featuring 2 attention heads, promoting in-depth context comprehension. Additionally, I opted for a block size of 48 tokens and the hidden layer of the feed-forward network for classification was made to have 2 times the input neurons.
3. **CustomTokenizer with fine-tuned hyper-parameters**: To optimize it further, I also changed the tokenizer. Both tokenizers, "SimpleTokenizer" (the one provided by default) and "CustomTokenizer" (the one I made), utilize NLTK for text tokenization and provide encoding/decoding functionality. However, "CustomTokenizer" offers advanced customization options, including maximum vocabulary size control and lowercase tokenization.

Analysis: The three graphs shown in 5b show that with introduction of these hyper-parameter changes, we can achieve better performance as compared to the performance achieved by using the default values given.
Changing the tokenizer and using a maximum Vocabular size of 3002 along with using the hyper-parameters as stated above helped achieve smoother curves as shown in 5a by the red curve. We can see that both the training and testing curves are smoother and achieve better accuracy of 86.67 %.
For the model that achieves the best accuracy we have Vocabulary size of 3002 and 0.602371M parameters.

5 Acknowledgement

I gratefully acknowledge the invaluable guidance and inspiration provided by [2] and Andrew Kaparthy's insightful work, available at `https://github.com/karpathy/ng-video-lecture`. Their pioneering contributions served as a beacon throughout the implementation process, shaping the development of this project and enriching its outcomes.

References

[1] Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers, 2023.

[2] Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.