CSE 256 PA2 Report

Lizhe Zhang
University of California, San Diego

November 5, 2024

1 Part 1: Encoder Trained With Classifier

1.1

Briefly talk about my implementation
Encoder Implementation: The TransformerEncoder class processes input sequences using token and positional embeddings. It has multiple TransformerBlock layers, each containing multi-head self-attention and feedforward neural network layers, along with normalization. The encoder applies an attention mask to ignore padding tokens during self-attention.

1.2

Feedforward Classifier Implementation: The FeedForwardClassifier class has a simple neural network classifier with one hidden layer. It consists of two linear layers with a ReLU applied after the first layer. This classifier takes the output from the encoder and return logits.

1.3

Joint Encoder and Classifier Training: The TransformerClassifier class combines the encoder and the feedforward classifier into a single model. It passes input through the encoder, applies a mask to zero out padding positions then averages the embeddings then fed into the classifier to predict the output classes.

1.4

Sanity Checks: Below are the plots of attention matrices for sentence "This is a sample sentence for attention visualization. " of head '0':

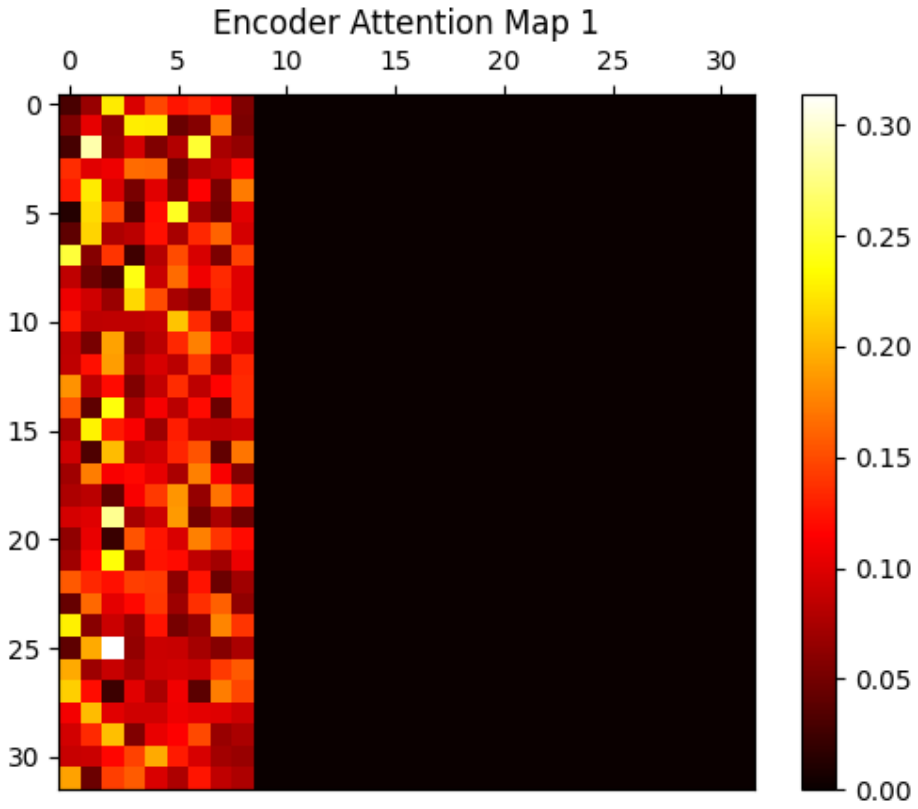

Figure 1: Encoder Attention Map 1

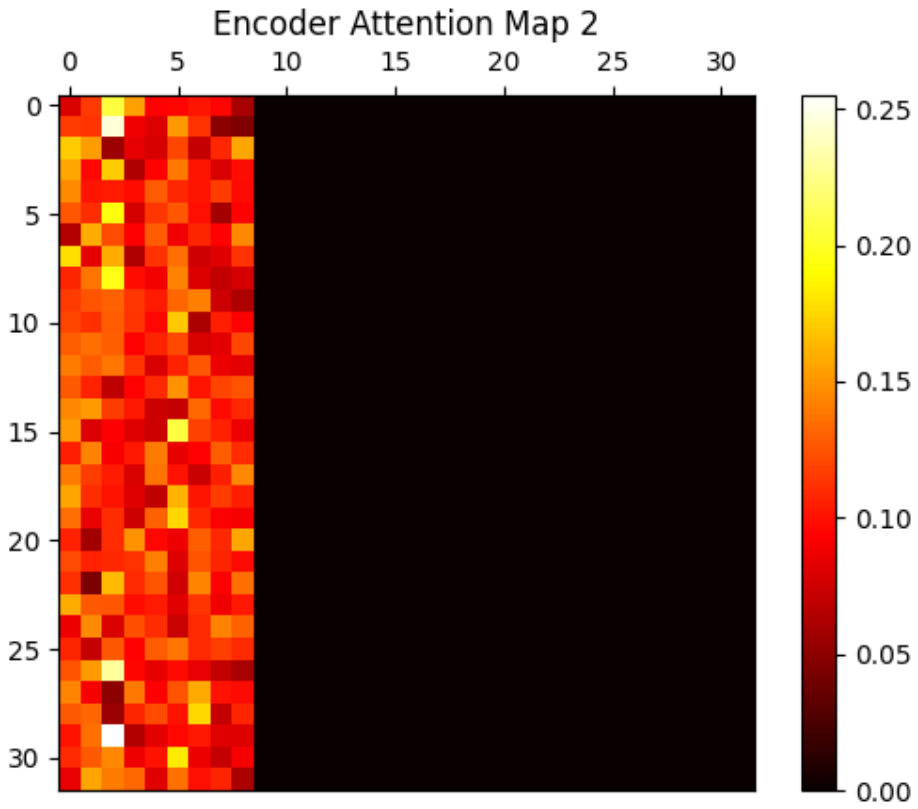

Figure 2: Encoder Attention Map 2

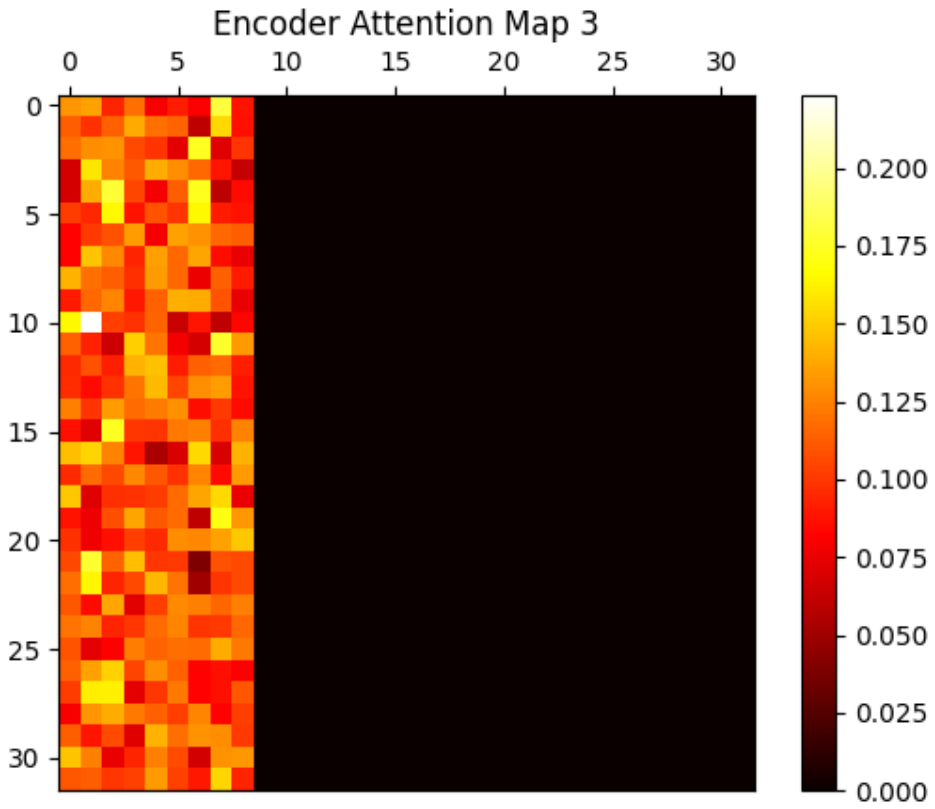

Figure 3: Encoder Attention Map 3

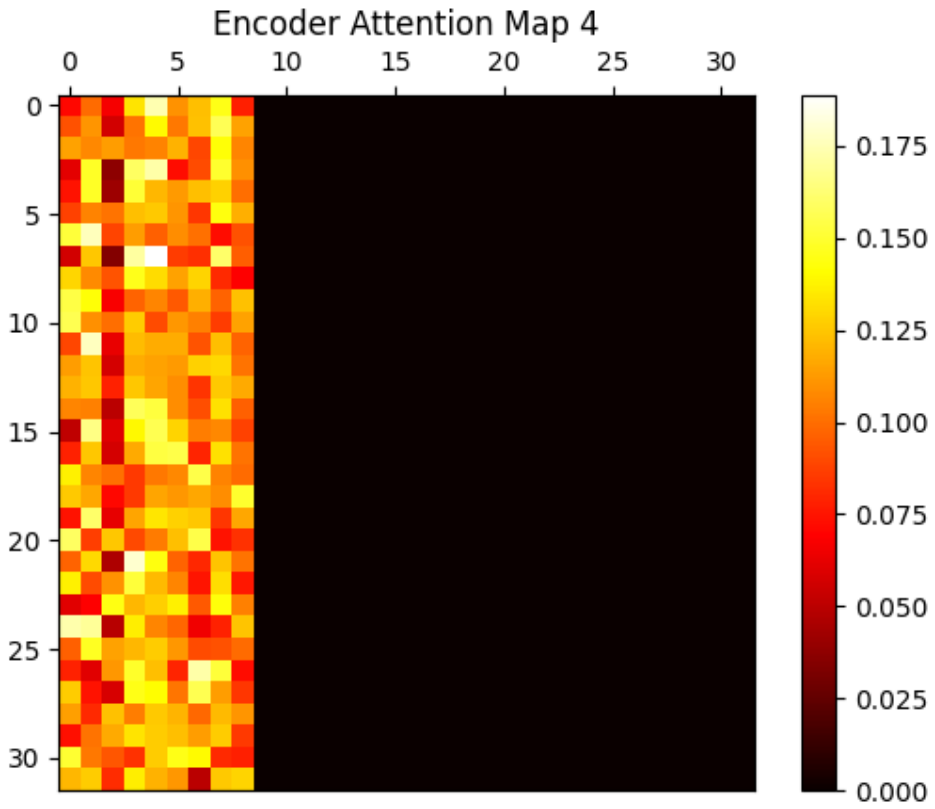

Figure 4: Encoder Attention Map 4

For these heatmaps, each square refers to attention weights between two positions in the input sequence. Map 1 shows a high level of variance in attention over different token pairs, which means that in the first layer, the model is possibly focusing on different types of relationships between tokens. Map 2 shows a diversity in attention with slightly less intensity compare to the first one. This suggests that the model has begun refining its focus in the second layer. Map 3 attention appears more focused compared to the first two layers, but still relatively dispersed. Map 4 shows further refined attention, with certain areas receiving much more focus than others. This suggests that by the final layer, the model has identified the most crucial token relationships for the task.
The black columns in the maps represent padding tokens which are masked out to prevent the model from attending to irrelevant positions.

1.5

Evaluation Number of parameters in the model: 583907
Epoch 1/15, Loss: 140.5901, Train Acc: 44.65%, Test Acc: 33.33%
Epoch 2/15, Loss: 133.9419, Train Acc: 51.91%, Test Acc: 46.13%
Epoch 3/15, Loss: 123.3875, Train Acc: 59.51%, Test Acc: 49.87%
Epoch 4/15, Loss: 113.0313, Train Acc: 62.81%, Test Acc: 56.00%
Epoch 5/15, Loss: 97.9308, Train Acc: 70.36%, Test Acc: 58.27%

Epoch 6/15, Loss: 88.0955, Train Acc: 72.94%, Test Acc: 60.00%
Epoch 7/15, Loss: 76.9519, Train Acc: 81.36%, Test Acc: 67.87%
Epoch 8/15, Loss: 63.1404, Train Acc: 85.09%, Test Acc: 72.27%
Epoch 9/15, Loss: 54.4837, Train Acc: 85.90%, Test Acc: 73.07%
Epoch 10/15, Loss: 46.8788, Train Acc: 90.58%, Test Acc: 75.47%
Epoch 11/15, Loss: 37.9585, Train Acc: 88.67%, Test Acc: 74.27%
Epoch 12/15, Loss: 25.7742, Train Acc: 87.52%, Test Acc: 71.87%
Epoch 13/15, Loss: 20.9980, Train Acc: 94.31%, Test Acc: 80.00%
Epoch 14/15, Loss: 16.3862, Train Acc: 97.56%, Test Acc: 82.53%
Epoch 15/15, Loss: 12.7030, Train Acc: 98.18%, Test Acc: 82.80%

2 Part 2: Pretraining Decoder Language Model

2.1

Decoder Implementation: The TransformerDecoder class generates output sequences token by token. It uses token embeddings and positional embeddings to represent input sequences. The decoder has multiple TransformerDecoderBlock layers, each containing multi-head self-attention with a causal mask to prevent attending to future tokens, a feedforward network, and layer normalization. The final layer normalization and a linear layer map the decoder outputs to logits over the vocabulary for next-token prediction.

2.2

Decoder Pretraining: During pretraining, the model minimizes the cross-entropy loss between the predicted token probabilities and the actual next tokens in the target sequences. The causal mask ensures that the model only attends to past tokens.

2.3

Sanity Checks: Below are the plots of attention matrices for sentence "This is a sample sentence for attention visualization. " of head '0':

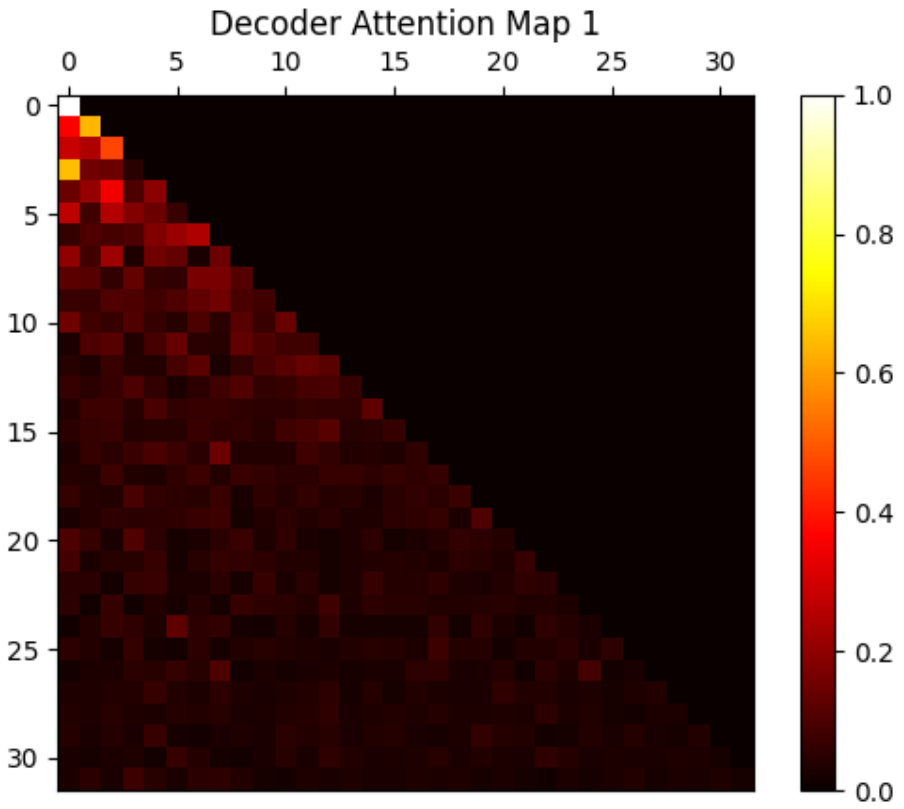

Figure 5: Decoder Attention Map 1

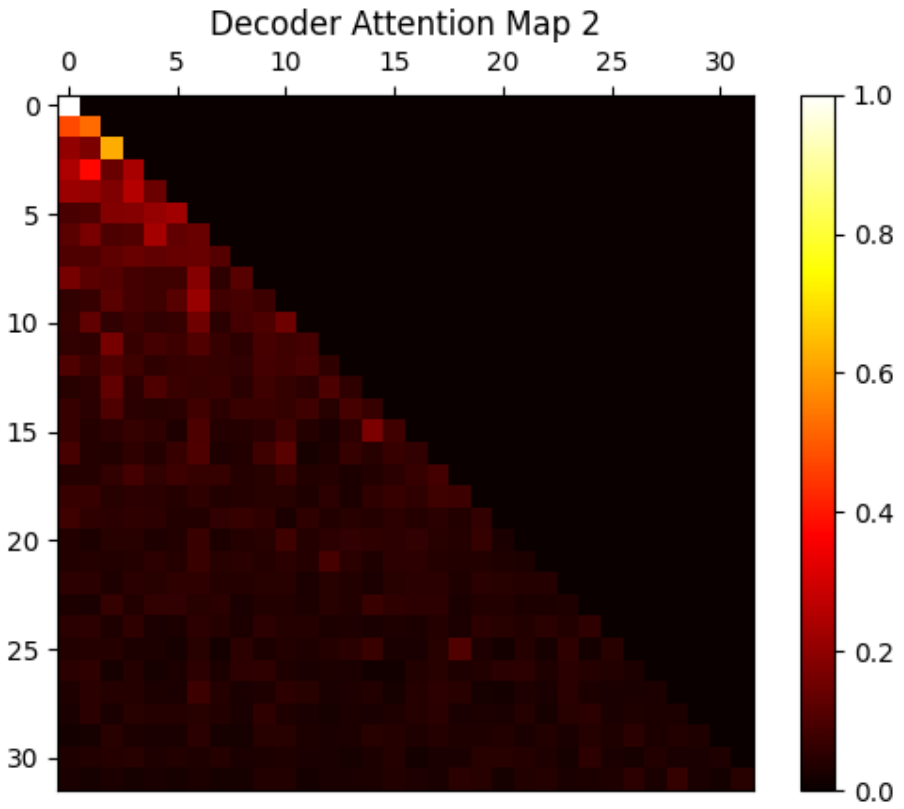

Figure 6: Decoder Attention Map 2

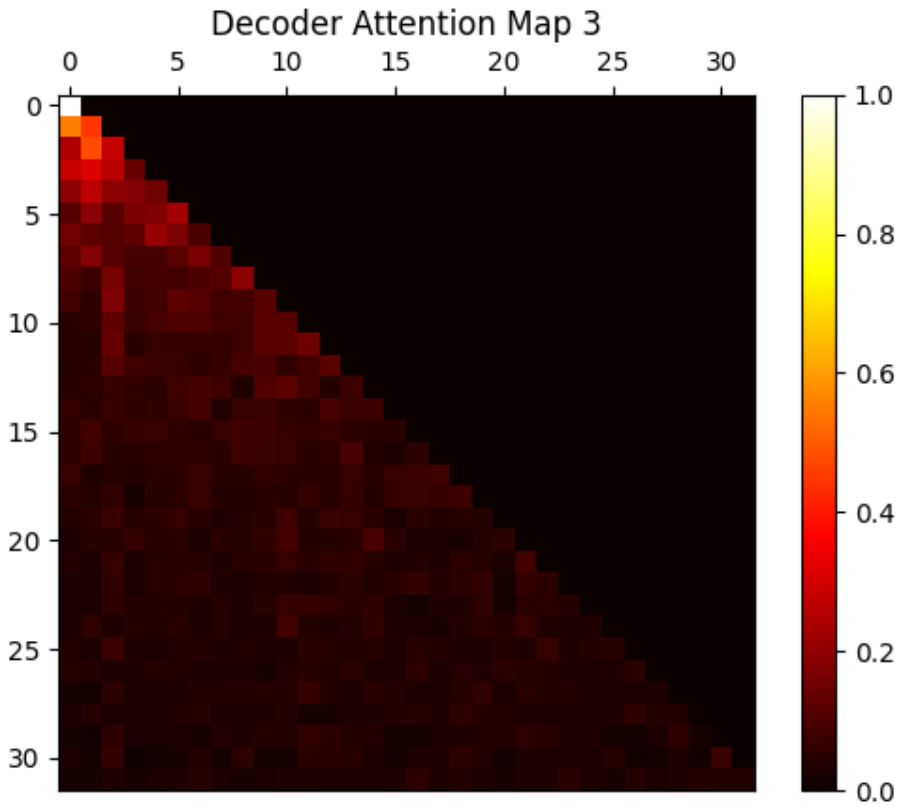

Figure 7: Decoder Attention Map 3

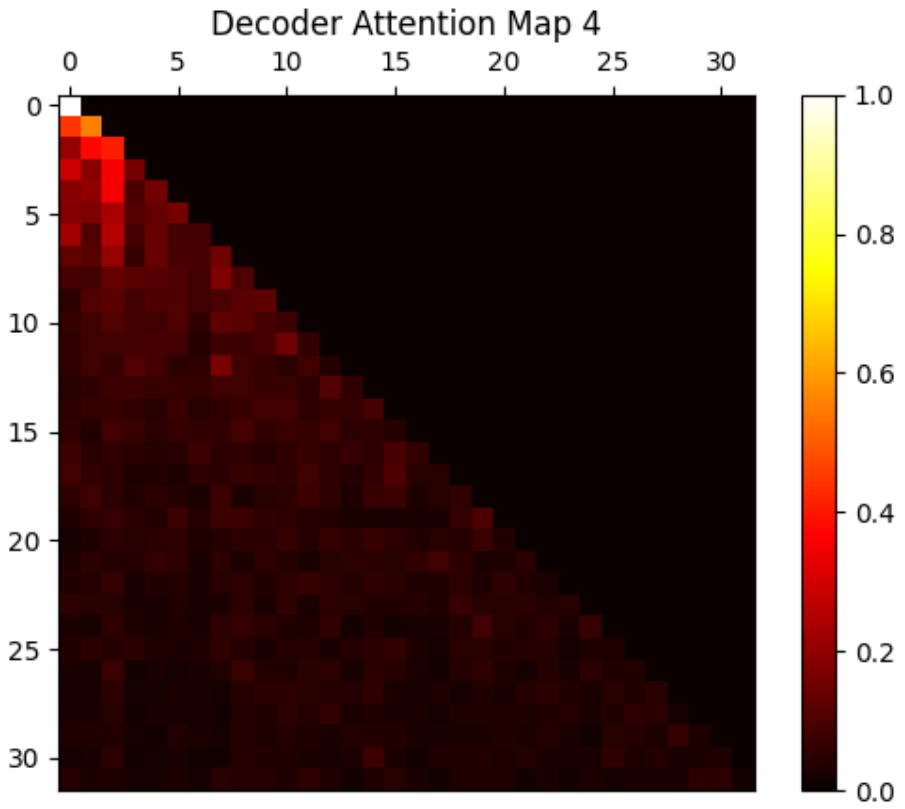

Figure 8: Decoder Attention Map 4

Map 1 shows high attention values at the beginning of the sequence. This indicates the model's focus on the start of the sentence, which is important in setting up the rest of the sequence. In Map 2 the attention is slightly more spread out but still focuses near the diagonal. This suggests that as the sequence progresses through the layers, the model begins to consider more context within a closer range. Map 3 shows further dispersion of attention along the diagonal. This pattern can indicate integration of more context as the decoder layers attempt to refine the output by considering both previous and following tokens. Map 4 presents a similar pattern to the third, with attention distributed almost uniformly along the diagonal and immediate off-diagonal positions. This distribution could reflect the decoder's continued focus on a balanced window of preceding context.

2.4

Evaluation
Number of parameters in the decoder: 858256
iter 0/500, loss: 8.80586051940918
Perplexity - train: 6323.0078125, Obama: 6323.58642578125, WBush: 6267.9169921875, HBush: 6321.8681640625
iter 100/500, loss: 6.373094081878662
Perplexity - train: 584.3173828125, Obama: 711.505615234375, WBush: 800.0965576171875, HBush: 738.1101684570312

iter 200/500, loss: 5.892664432525635
Perplexity - train: 416.93878173828125, Obama: 547.1564331054688, WBush: 652.3721313476562, HBush: 575.7341918945312
iter 300/500, loss: 5.752065181732178
Perplexity - train: 293.51092529296875, Obama: 463.2708435058594, WBush: 579.63720703125, HBush: 507.2281799316406
iter 400/500, loss: 5.359716415405273
Perplexity - train: 227.60296630859375, Obama: 414.0304260253906, WBush: 511.5280456542969, HBush: 453.8175964355469
iter 500/500, loss: 5.06597375869751
Perplexity - train: 171.37835693359375, Obama: 381.2608642578125, WBush: 495.1454772949219, HBush: 436.2565002441406

Discussion: The perplexity on the test sets is higher than the training set is reasonable as the model not having seen the test data during training. Obama has lower perplexity compared to the Bushes might be due to the vocabulary usage from Obama is more consistent with the training data.

3 Architectural Exploration Alibi

AliBi modifies the attention scores by adding a bias that depends on the distance between query and key positions instead of adding positional embeddings to the input tokens. Below are the sanity check plots:

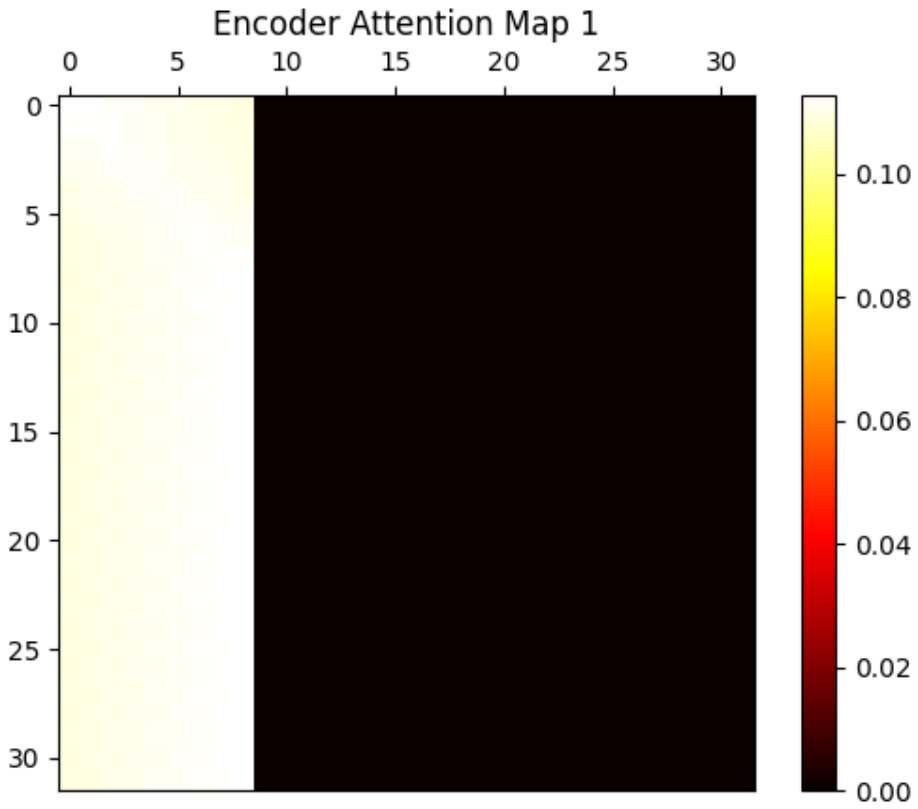

Figure 9: Alibi Encoder Attention Map 1

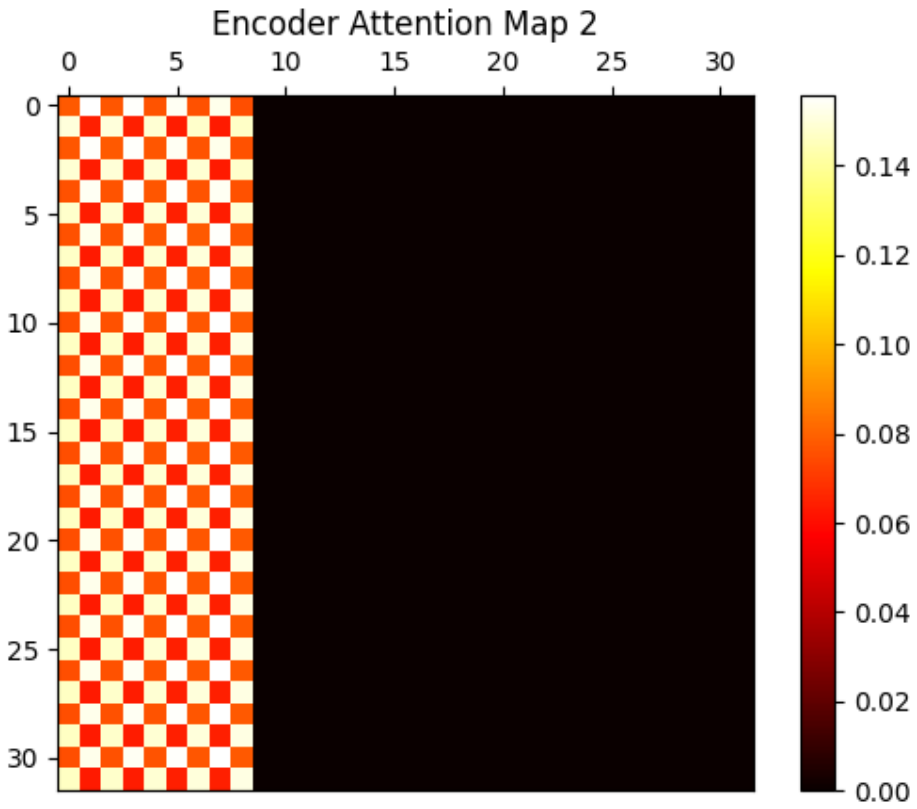

Figure 10: Alibi Encoder Attention Map 2

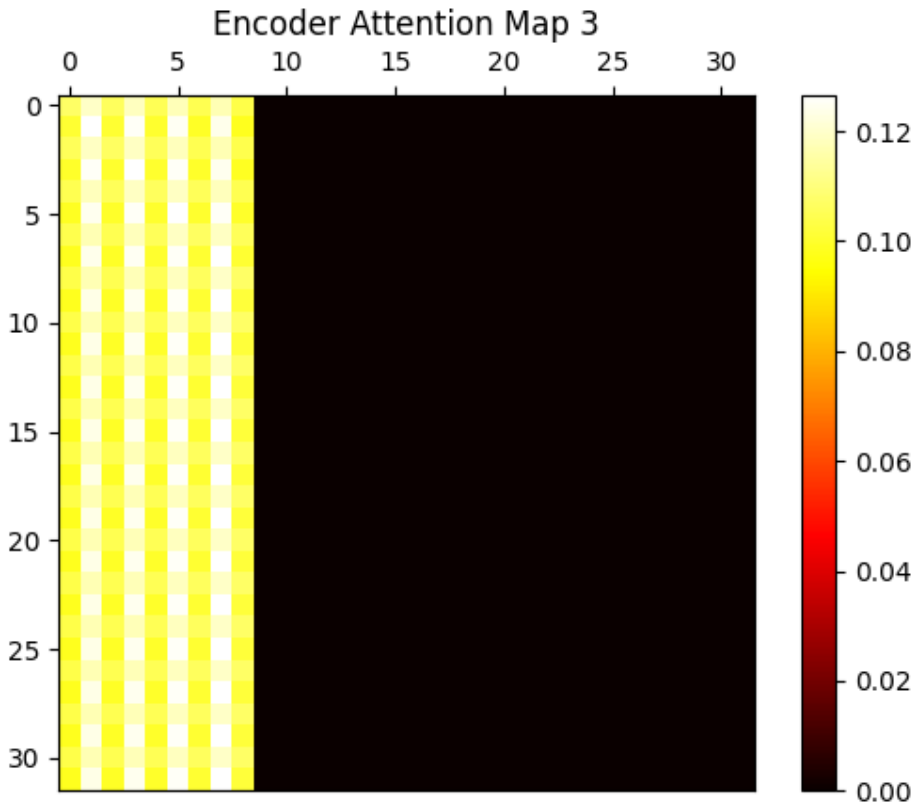

Figure 11: Alibi Encoder Attention Map 3

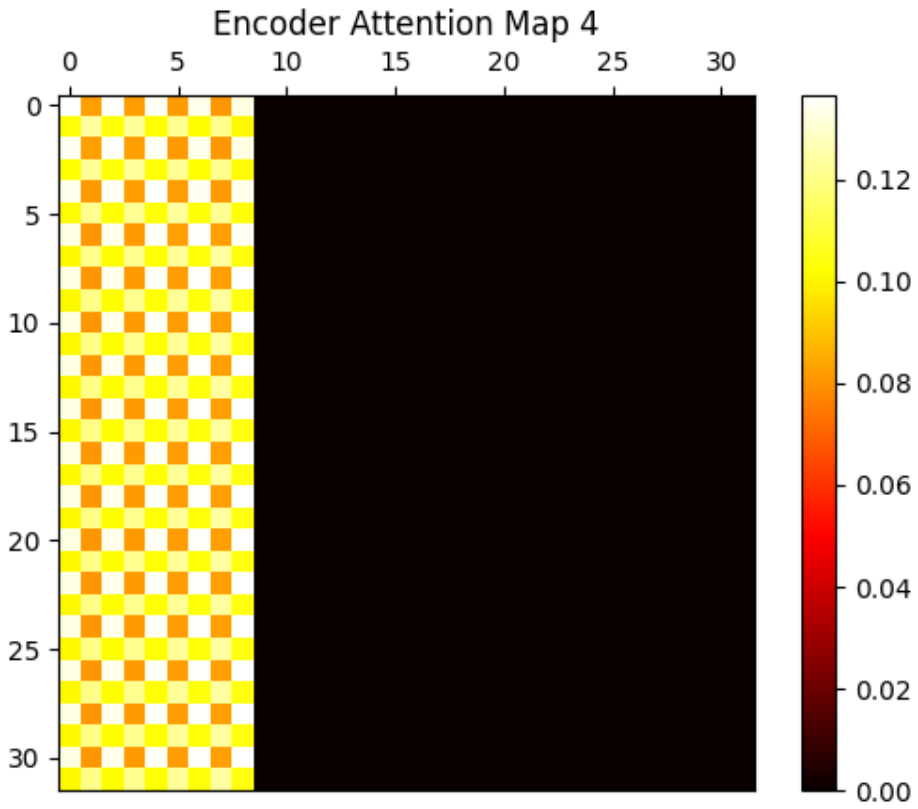

Figure 12: Alibi Encoder Attention Map 4

The attention patterns in these AliBi encoder maps are like checkerboard, especially for Maps 2, 3 and 4. This pattern suggests that the attention mechanism alternates focus between different token positions across the sequence, there fore potentially provides a more dynamic context integration that traditional self-attention mechanisms might miss.

Below are the results:

Number of parameters in the model: 581859 Epoch 1/15, Loss: 140.1086, Train Acc: 35.99%, Test Acc: 36.13%
Epoch 2/15, Loss: 134.2894, Train Acc: 55.21%, Test Acc: 49.60%
Epoch 3/15, Loss: 125.4874, Train Acc: 61.04%, Test Acc: 50.93%
Epoch 4/15, Loss: 111.3115, Train Acc: 66.06%, Test Acc: 56.40%
Epoch 5/15, Loss: 93.7771, Train Acc: 75.81%, Test Acc: 64.93%
Epoch 6/15, Loss: 78.4701, Train Acc: 78.59%, Test Acc: 68.27%
Epoch 7/15, Loss: 62.5075, Train Acc: 85.42%, Test Acc: 74.27%
Epoch 8/15, Loss: 45.5738, Train Acc: 88.91%, Test Acc: 76.53%
Epoch 9/15, Loss: 36.6173, Train Acc: 90.15%, Test Acc: 80.80%
Epoch 10/15, Loss: 28.5401, Train Acc: 95.51%, Test Acc: 83.87%
Epoch 11/15, Loss: 23.7119, Train Acc: 94.07%, Test Acc: 83.47%
Epoch 12/15, Loss: 17.6140, Train Acc: 96.03%, Test Acc: 81.60%
Epoch 13/15, Loss: 25.1847, Train Acc: 95.70%, Test Acc: 84.40%
Epoch 14/15, Loss: 19.3052, Train Acc: 96.70%, Test Acc: 85.20%
Epoch 15/15, Loss: 11.1621, Train Acc: 98.14%, Test Acc: 84.93%

We can see that the test acc is slightly higher than then one from traditional encoder, while the loss is slightly lower. It is likely because AliBi allows the model to capture relative positional relationships more effectively without adding extra parameters from positional embeddings. So, the model can generalize better to unseen data, leading to the observed increase in test accuracy compared to traditional absolute positional embeddings.