***Abstract*—Transformers have led innovative developments in language and vision modeling ever since the seminal paper "Attention is all you need" was published. However, transformers remain a mystery to their users, acting as a black-box solution to every problem. Many people are working with transformers and to apply the power of transformers to their applications they are experimenting with the hyperparameters and hoping to achieve better results. This is a major hurdle in the development of more superior and powerful models since no one clearly understands how transformers work so well and the internal functioning is a mystery. Due to this, the improvement of the models is left to luck. Several attempts have been made to analyze transformers but have not been able to produce results that would expose their inner workings. We follow the trail of Mechanistic Interpretability left by Nelson et. al and try to provide meaningful interpretations of attention-only transformer models to identify a portion of the workings of a transformer. We believe this would be a starting point for a much deeper analysis of the transformer circuit laying the ground for what is to come. In this project, we built a transformer from scratch and attempted to train it in order to understand what the model has learned by analyzing the attention scores of the model.**

Github:https://github.com/avisinghal6/Transformer-Training

I. Introduction

The rise of Transformers when the paper Attention is all you need was introduced brought forth an immense change in Deep Learning's ability to utilize language data. Transformers are used by almost every industry, and sequence and vision-related tasks are also making use of transformers and achieving great results. RNNs are considered obsolete after the introduction of transformers, this was a shift of paradigm. While GPT2 and GPT3 have established the power of transformer models, their inner workings remain unknown to their users and owners. There have been several attempts at interpreting transformer models, but no solid proof of concept has been established for how and why transformers work.

Very recently, OpenAi released ChatGpt which is a very smart chatbot. It has been trained on huge data and is able to answer any question remarkably well, it does not make any grammatical errors and pays attention to the details in the question. People are shocked by the results of ChatGpt and are wondering how it is able to achieve this behavior. This project tries to bridge the gap with the help of Mechanistic Interpretability. Being a new field within interpretability, we believe that this approach of attempting to unveil transformers would provide a better understanding of what the attention layers are up to, provide reasoning for the results produced by the transformer and help to improve models. There has been some success in understanding the convolutional neural networks, Matthew D. Zeiler proposed the idea of visualizing and understanding CNNs, this enabled them to improve the state-of-the-art models as they were able to understand what was wrong with the models. We feel that our project is a step towards this and can help in improving the current state-of-the-art models in a more deterministic way.

II. Motivation

The paper "Visualizing and Understanding Convolutional Networks" by Matthew D. Zeiler [1]focused on understanding what the hidden layers learned and the features captured by the activations. This helped them improve the existing models with a proper understanding rather than just trying out different variations of hyperparameters. A project by distill.pub on distill circuits [3] has shown the interpretation of vision models by reverse engineering these models and understanding the hidden neurons using this as motivation, we believe that understanding the attention layers is the key to improving the existing state-of-the-art NLP models.

III. Problem Statement

Understanding of how the inner circuits of transformers work by analysis of attention layers of these models, leading to interpretation of attention layers, leading to improved interpretability of transformer models.

IV. Literature Review

C. Olah et Al. attempted to reverse-engineer transformers. They started by analyzing autoregressive decoder-only transformers with 0,1,2 attention layers. They found that 1 layer attention-only transformers are an ensemble of bigrams and skip trigrams. They were able to show that a few heads of the 1-layer attention-only transformer were performing a copying behavior. If the target token was paying the most attention to a specific source token, then the output was the same source token. The authors trained their models on a variety of datasets, they even tried python code generation, and the model was able to predict the next code snippet quite decently. The 2-layer attention-only transformer becomes more capable of in-context learning, the authors claim that it is due to the formation of induction heads, which search over the context for previous examples of the present token. If they don't find it, they attend to the first token. The authors claimed that this behavior was due to the composition of heads i.e the outputs of the heads from the first attention layer served as the queries for the head in the second attention layer, this is called Q composition. The authors trained their model on various datasets to establish their findings and provided mathematical reasoning for the induction heads.

Elena et Al. also evaluated the contribution made by individual heads in the encoder to the overall performance of the model and analyzed the roles played by them. They also pruned the heads which were insignificant and found that a few heads were contributing the most and were the last to be pruned. Kevin et Al. analyzed that BERT's attention heads exhibit patterns such as attending to delimiter tokens, specific positional offsets, or broadly attending over the whole sentence, with heads in the same layer often exhibiting similar behaviors. They showed that a few attention heads correspond well to linguistic notions of syntax and coreference.

V. Objective

To train a transformer model and to establish a proper analysis of each attention layer of a transformer model with the help of mechanistic interpretability of transformer circuits. To adapt this framework towards real-life natural language processing domains such as text generation, named entity recognition, etc.

VI. Experimental Settings

1) Datasets used: IMDB(50,000), Wikitext(1.8M)
2) Tools used: Kaggle, PyTorch, colab, wandb
3) Hyperparameters: no. of attention heads, embedding size, learning rate, optimizer, sequence length, tokenizer, epochs, size of the dataset.
4) A decoder-only transformer with attention layers as well as a few variations explained in the next sections was built from scratch and trained.

The key components of the decoder-only transformer include:

1) Padding mask: Since all sequences were not of the same length, they were padded with 0s to ensure the same max length, so the 0s were not to be included in the calculation of attention scores and training.
2) Causal mask: for decoder only transformers the words in the future are not known, so when calculating the attention of a certain token, only the words before it was taken into consideration, to ensure this a causal mask was created.
3) Tokenizer: The words had to be converted to tokens so that then they can be converted to embeddings, we tried different tokenizers that suited our model.
4) Embedding layer and positional encoding: The tokens had to be assigned an embedding, and we experimented with different embedding sizes. The embedding layers were also trainable. Since transformers by default do not capture any positional information of the sequence, this information is passed by encoding the position of each word, positional embedding was also trainable in our model.
5) Multihead attention block: this block was used for finding the attention scores for the queries, this block provided the scores which showed how much a certain query attends to a specific key. This block gives out the context vector for each query.
6) Unembedding layer: the output from the multi-head attention block needs to be converted to a token, the token represents a word. So this is a fully connected layer mapping the context vector to the dictionary elements, each element is assigned a logit, and the higher the logit, the more probable that element is.
7) Sampling the output: Since the output is of size dictionary size, each element has an associated logit, to predict the output we need to sample an element according to the logits. We selected the top K elements and then performed sampling to get the output. This approach is similar to how gpt2 selects the output.

VII. Experiments

We started by training 1-layer attention-only decoder-only transformer. We started with the IMDB dataset. We ran multiple experiments by varying the tokenizer, epochs, number of heads and max length. We experimented with BERT and GPT2 tokenizers. BERT tokenizer has vocab size of 30,000, however, when we used it, we found that it was generating a lot of words with hashes i.e "#". Since the hashes would not allow us to interpret anything meaningful, we decided to change the tokenizer. Below is the kind of output we obtained using BERT. We think that the vocabulary size was not enough

```
##sad ##quisition ##hala ##spense ##dian ##spense ##ept ##ducing ##
```

Fig. 1: Losses in a batch size of 60

for our dataset and hence we chose a tokenizer that had a larger vocabulary size. GPT2 has a vocabulary size of 50,000.

We also considered training a 2-layer attention-only decoder-only transformer. We did this because we thought that the second layer might build up on what layer 1 has learned, and it might be something more interpretable.

We also added MLP layers to the model and tried training, the reason for this was that we thought the MLP layers would allow complex features to be captured that attention layers by themselves might not be able to capture. We also added residual connections to the MLP layers in order to try to get similar experiment settings as Chris Olah et Al.

In this project, we focussed mainly on two datasets i.e IMDB, and Wikitext. IMDB was a comparatively small dataset and we were able to experiment with different hyperparameters and settings. The simulation time for IMDB was much faster compared to wikitext. IMDB dataset took close to 30-45 minutes to complete, whereas the wikitext dataset took 12 hours to complete 15 epochs (same as IMDB). So it was not feasible to run multiple variations on wikitext. We kept our focus to train a small model with huge data so that it learns better, this would help us train comparatively faster. For wikitext, we only considered 1 attention layer. In the next section, we provide a detailed analysis of the results obtained for each of our experiments.

VIII. Analysis

A. IMDB Dataset

For the IMDB dataset, we experimented with various hyperparameters. We tried changing the maximum sequence length (20,50,80), the learning rate (1e-4, 1e-3), the number of heads(4,8), and the embedding size(128,256). We used wandb to monitor the results of the sweeps that we performed. Figure 2 shows the plots for the test and training statistics.

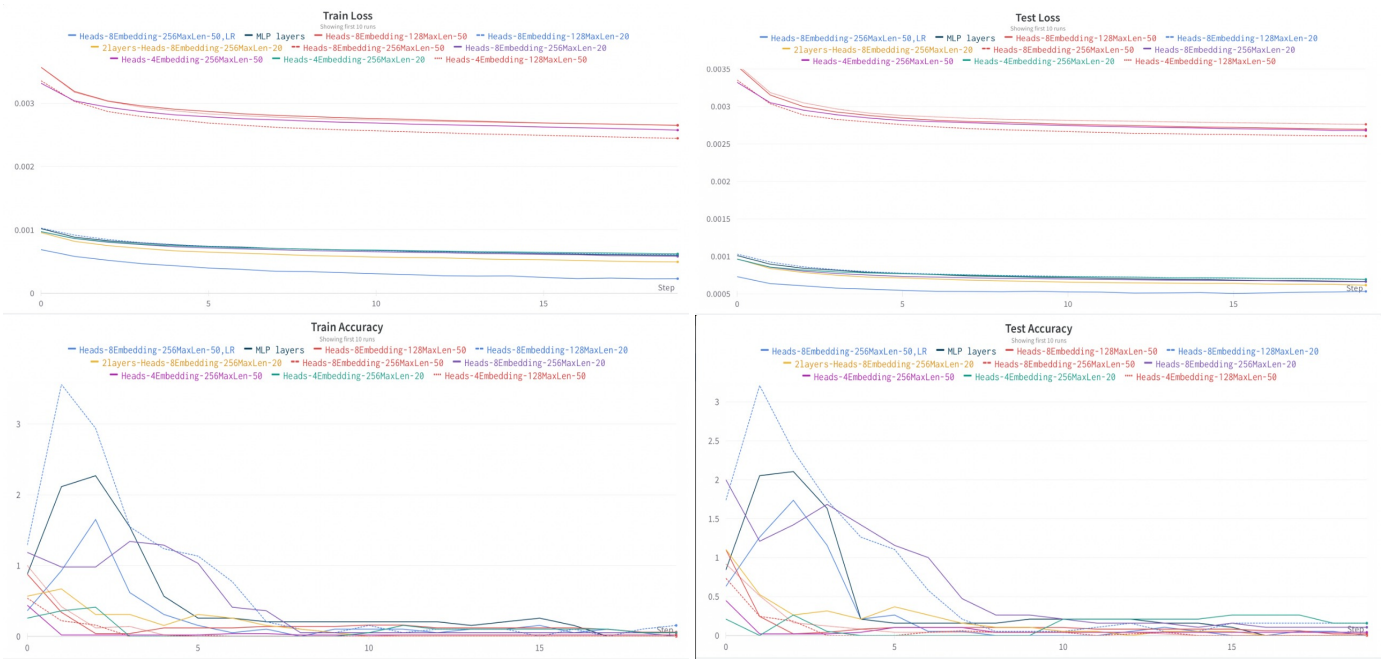

Fig. 2: Sweep Results for IMDB dataset

There are a number of interesting things in Figure 2. The train and test loss clearly indicate that the model is training as the losses are decreasing with epochs. The decrease in values is very slow with epochs, this is a key limitation in training. For all the different variations that we experimented with, the trend of losses is the same, and the rate at which the loss falls is also very similar. The difference in losses for the experiments is due to different initializations. We do not see any major improvement in training with MLP layers, using 2 layers of attention. This is also one of the reasons we decided to train the wikitext dataset using only 1 layer of attention. It can be seen that the value of the loss is very low, however, it is not low enough to ensure good training. The model is still very poor right now which is also evident from the train/test accuracy plots. The variations seen in the accuracy for train/test loss are insignificant because the model is training but training poorly. In order to train the model better, we need more number of epochs, more computation power, more time, and more hyperparameter tuning. Currently, the model is trying to learn, but there are too many parameters that need to be tuned and the data supplied is comparatively low and the computation power is also low which makes the training slow. The number of parameters is high due to the following reasons: we are training the embeddings (positional embeddings as well), weight matrices inside the multi-head attention blocks, and output fully connected layer. In our experiments, we added 2 layers, MLP layers, which led to a further increase in the number of parameters. The more the number of parameters more is the time spent in gradient calculation. This is another motivation to use only 1 layer of attention for training the wikitext dataset, it will also make the training faster.

We think the reason that IMDB dataset did not train well even though it is a small dataset can be due to a large number of outliers which would require rigorous training to fit. Hence we try to train another dataset which is larger and would be less prone to such behavior.

In order for the model to train well, the loss value needs to be at least in the range of 1e-6, for an approximate accuracy of 75%. This small value is due to the fact that our vocabulary size is 50,000. So, when the model predicts the outputs, it actually logits over 50,000 tokens. Taking the softmax for such a large number of elements yields a very small value.

Below we analyze the heatmaps of attention scores for the different experiments we conducted. The heatmaps represent the amount of attention that a particular query pays to the different keys, this is then further used for the prediction of the next token. In our model, we feed it with a few starting words and our model is an autoregressive decoder model, so whatever output it generates will be added to the input sequence in order to predict the next token. This process is repeated until the max number of tokens has been reached. We have given the model the same input for almost all the results for maintaining consistency and fair comparison. The input that we give is ”This movie is” and then we let the model generate about 7 words. In the heatmaps below, the x-axis is the keys, which represent the words that are available in the input at a particular time. The y-axis represents the query, these are the tokens for which

we are trying to predict the next word.

Figure 3 shows the attention heat maps of the best results that we obtained from the sweep for 1 layer attention model. The configuration was embedding size:256, a number of heads:8, and a max sequence length of 20.

The attention plots have zeros in the right-upper triangle of the matrix. This is due to the causal attention mask, the model only pays attention to words before them and not after them. For head 1, when the model predicts the word ”average”, it pays the most attention to the word ”movie”. In this case, the output that the model produces makes sense logically and it is paying attention to the right context. This is true even when the model predicts ”average” again as the last word. Since the model has not trained well enough, we cannot expect it to learn a lot of things at this stage. For head 6, we see that this particular head pays the most attention to the word ”movie” for all queries. It seems that the model is able to detect the important part of the context but not able to use it well for predictions. This is primarily due to poor training. There were a total of 8 heads out of which we selected only 2 because some heads had information that was not interpretable at all, and some heads did not learn anything and were completely focusing on one element all the time.

Figure 4 illustrates the attention maps for 2 layers attention-only model. These are the outputs of the second attention layer. In head 1, we can see that this head places the most attention on the previous word and copies(when it starts to predict the next token after ”is”) the same word in the output. It is performing the copying mechanism. The authors Chris Olah et Al also showed that one of their heads was performing a similar coping mechanism. The heatmap for layer 2 head 2 shows that the particular head focuses on the first and third words mostly all the time. Figure 5 shows the attention score heatmap for the layer 1 head of 2 layer attention model shows that the particular head always focuses on the first two words for all its predictions.

Figure 6 represents the attention score heatmaps for the model trained with MLP layers+ 2Layer attention. With the addition of MLP layers, the analysis becomes even more difficult, this is because the features captured by the attention layers undergo a complex transformation due to the MLP layers. Head 4 is focusing the most on the word ”movie” but it does not lead to anything which can be interpreted. Similarly, head 7 also focuses on the first word but is not interpretable. In figure 7, for the head in attention layer 1 of this model, the heat map shows a pattern where the focus is always on the second last word for predicting the next token. So when ”necessarily” is predicted, the focus is always on ”achieved”.

B. WikiText Dataset

The dataset consists of a total of 1801350 train texts and 4358 test texts extracted from handpicked articles in Wikipedia. This dataset was chosen after the IMDB dataset due to its text richness as it varies over different genres of articles written across Wikipedia, as well as the quantity of data since a major phenomenon we observed during our experiments was the hunger for transformers for computation and data. This allows for better learning within the parameters in the model as the large dataset allows for better expressiveness of the scattered data.

Experiments were conducted with the hyperparameters, mainly over embedding size(128, 256) after what was observed within the IMDB run. A total of 35 epochs was run for the model with the model embedding size 256 whereas the model with an embedding size of 128 was run and stopped at epoch 15 due to computational constraints.

For both the model of embedding sizes 128 and 256, as seen in Fig 8 and 9, the training and test loss curves move downward indicating the stability of the training. The decrease is as slow as the IMDB run, while the amount of computation per epoch rose by approximately 5 times. This posed a great limitation in obtaining a fully trained transformer model. The test loss is much higher than the training loss due to the size of the dataset and the covariance between the train and test dataset, thus making it more difficult to learn generalized features in the initial epochs. This again informs us that the model is training, but not enough to learn anything meaningful. Due to the sparse amount of training it was subjected to, the model could not capture the key information from the dataset, which was limited by the very high computational factor, a large number of parameters, and the need for time.

The attention plots in Fig 8 represent the attention plots of the model of embedding size 128. It is evident that the right image acts similarly to the experiment seen previously: attention to the word ”movie”. The image on the left, on the other hand, interestingly shows attention towards ”movie” and the immediate word indicating that a big amount of focus is placed on the movie, the topic of the conversation, as well as what to go for next word based on the immediate word, a combination of the results we saw previously performed by 2 different attention heads.

When looking at the attention plots in Fig 9 for an embedding size of 256, there are several interpretations one can observe within the plots coinciding with the results from the IMDB dataset. The left plot shows that this particular attention head 1 focused on the 2 immediate words that most sequential models perform with. Whereas, the right plot show clearly the focus of the attention map over the

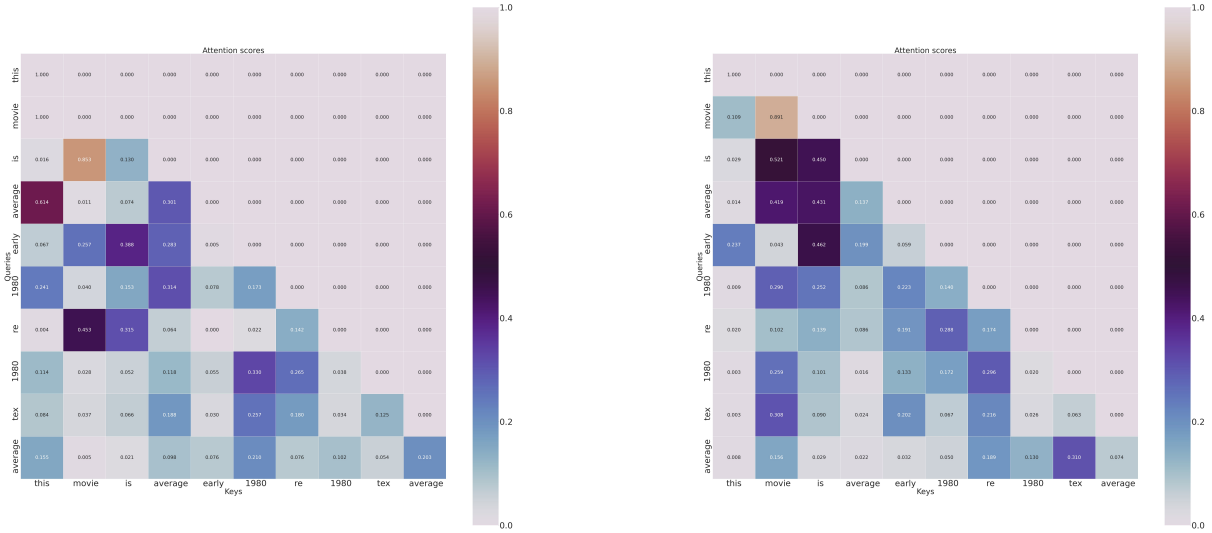

Fig. 3: Attention score heatmaps for 1 Layer Attention only. Left image: Head 1, right image: Head 6

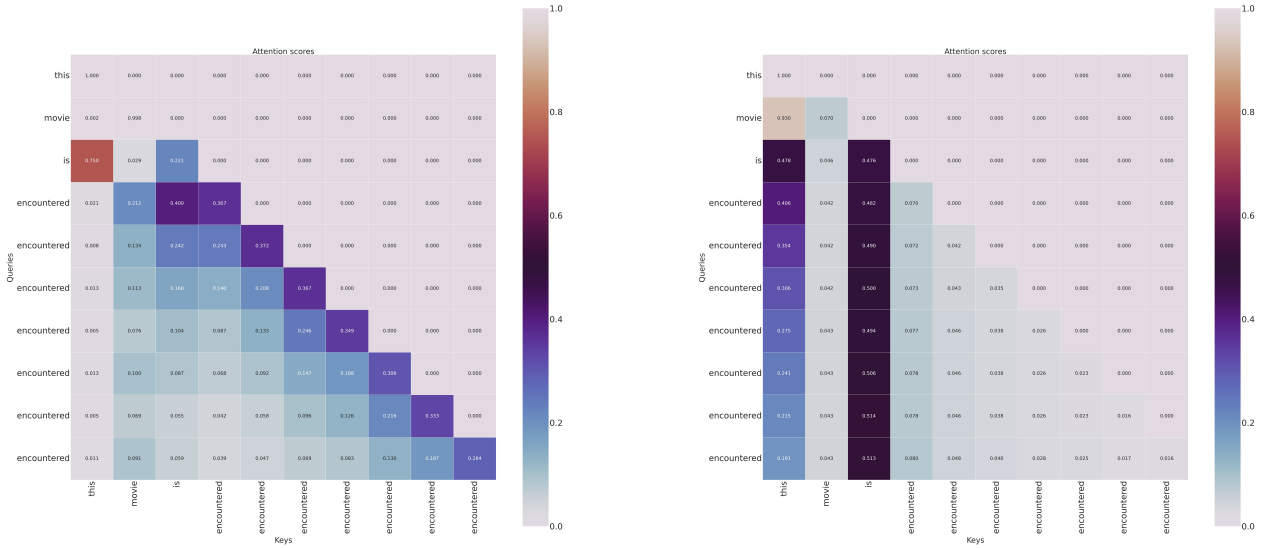

Fig. 4: Attention score heatmaps for 2^{nd} Layer Attention only model. Left image: Head 1, right image: Head 2

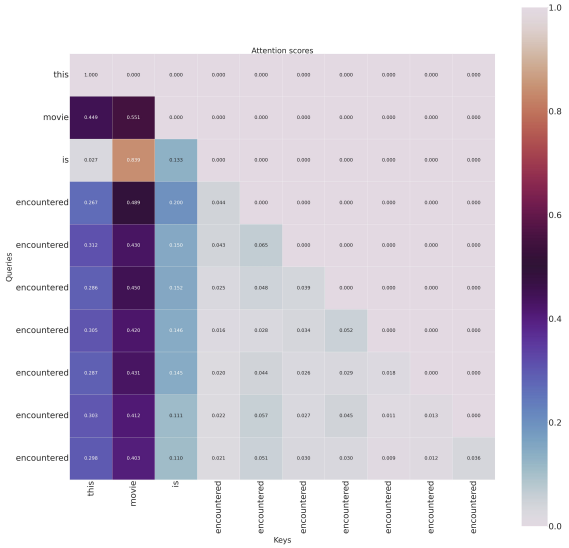

Fig. 5: Attention score heatmap for Layer 1 of 2^{nd} Layer Attention only model. Head 3

word "movie" throughout the predictions indicating that the layer has assigned a high value of interest towards the topic that is being talked about in the sentence, showing a high grammatical sense.

IX. Difference in Experiment Settings from Reference Work

The experiment settings were not clearly outlined in the reference paper, which made it difficult to replicate the experiments. Below is the key difference that we could identify. Other things like the use of pre-trained embeddings, training time, list of datasets, etc. were not specified. We think that the training time of the models for the authors would also have been very long as transformers are difficult to train and require huge training.

Reference Paper Settings	**Our Settings**
Always used residual connections	Used residual connections only for 1 experiment of IMDB dataset training
Used 12,32 attention heads	Used 4,8 attention heads
Used large embeddings (768)	Used smaller embeddings (128,256)
Used gpt2neo tokenizer	Used gpt2, Bert tokenizer
Used huge datasets for training	Our datasets were comparatively smaller
Use of max length of 2048	We used max length up to 80

TABLE I: Differences in experiment settings

X. Challenges Encountered

- Obtaining a clear understanding of transformers and their mathematics was extremely challenging as transformers are relatively new. The material on the internet is describing mathematics very vaguely and does not provide any intuition for it. There exist several different interpretations in order to understand transformers (for example, what Dr. Allen, guest lecturer, mentioned in class compared to what Mr. Chris Nolah defined in the blog). Also, there are many components in a transformer and it is critical to understand each of them in order to implement it properly. We could not find many concrete examples/ templates of transformers available which would enable us to understand them easily. So to understand the working of transformers, we built each component of the transformer from scratch without using any library function. This was for the sole purpose of understanding, we found a blog

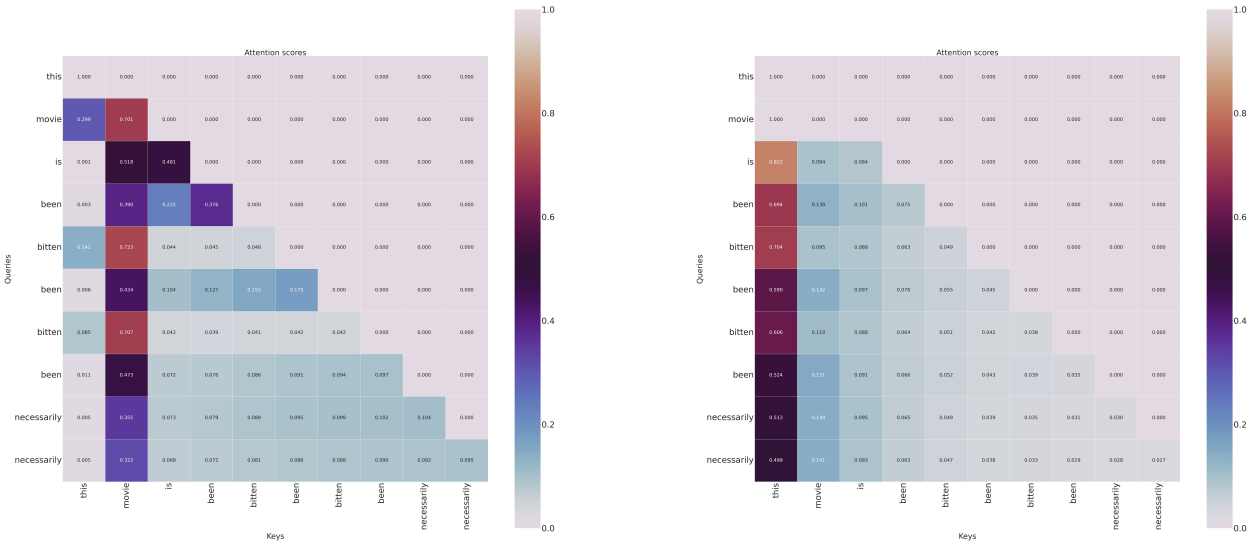

Fig. 6: Attention score heatmaps for 2^{nd} Layer Attention +MLP layers model. Left image: Head 4, right image: Head 7

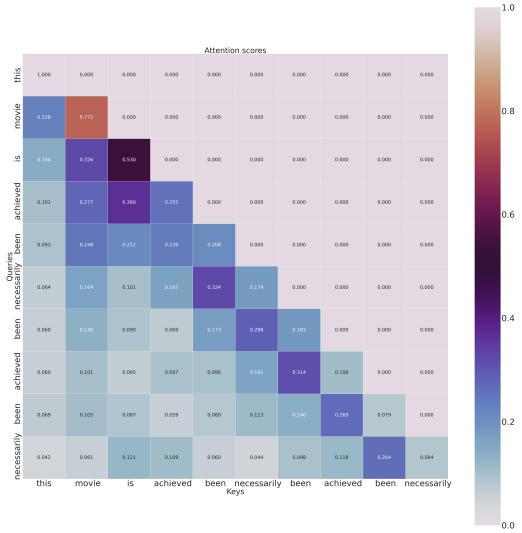

Fig. 7: Attention score heatmaps for 1^{st} Layer Attention +MLP layers model. Head: 5

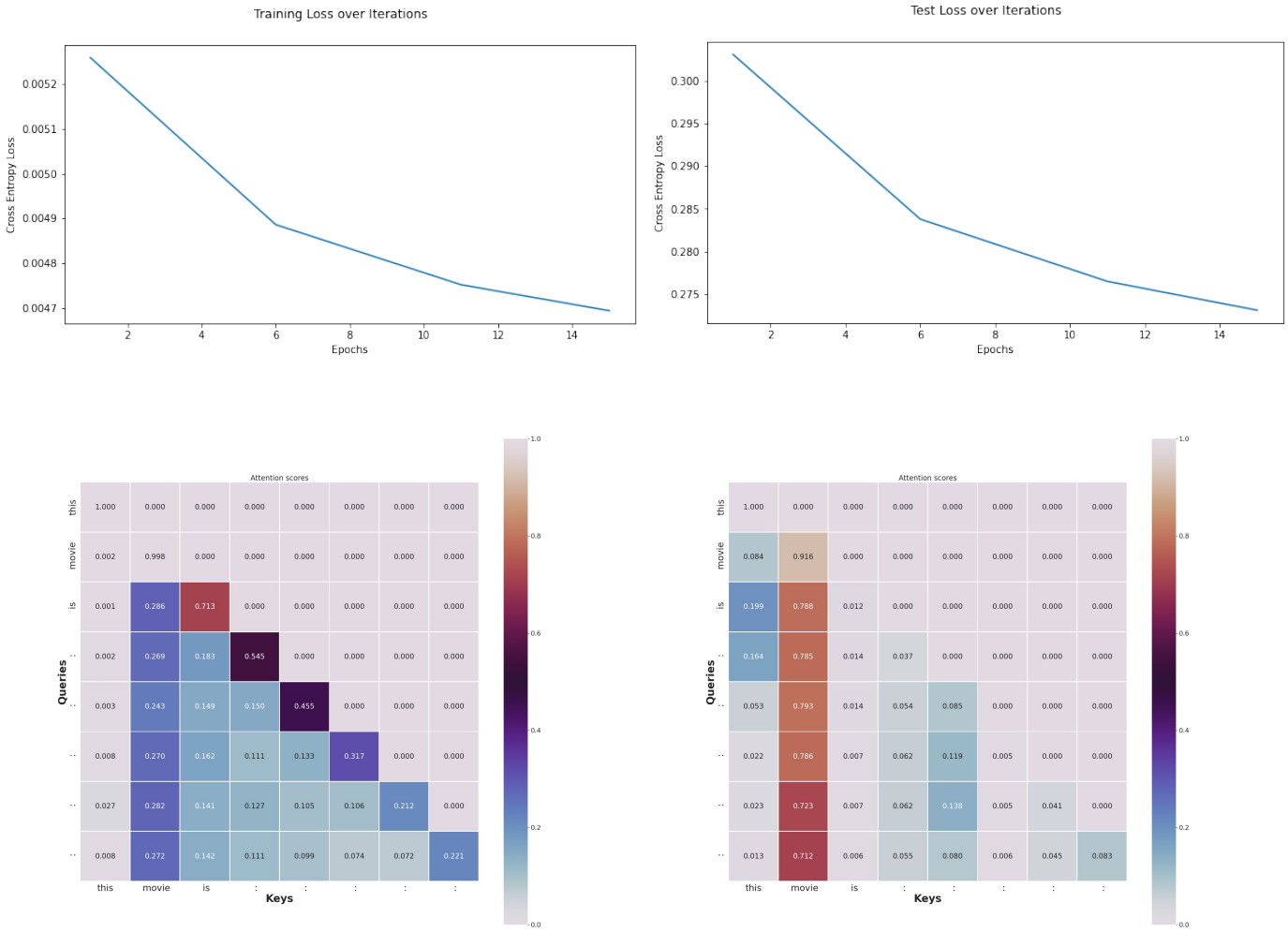

Fig. 8: Training Results on WikiText Dataset with Embed dims 256:- 1(Top-Left): Training Loss, 2(Top-right): Test Loss, 3, 4(Bottom) Attention score maps:

that explained each of these components and had some code snippets. After performing this exercise, we got a clear understanding of each of the different components in the transformer and then we were able to use this knowledge to build the model that we eventually trained.

- Creating the working pipeline for a decoder-only transformer was a difficult task. Since the transformer has so many components, it becomes difficult to identify which part could be creating the issue. After we built the model, we were continuously getting ”Nan”, we tried to debug the issue thinking that the gradient flow could be causing such problems. After extensive debugging, we figured out that the issue was in the causal padding mask, the padding mask was masking the tokens which it should consider during the calculation of attention. Similarly during the text generation, implementing the autoregressive nature of the decoder required significant efforts.
- Training transformers require a lot of data and computation power. We tried training it on a small dataset (IMDB) and the model did not train well.Hence we trained the model on a large dataset which took a very long time (45 hours for 35 epochs.
- There are many hyperparameters that can be tuned to optimize performance, however running experiments on all of them is not feasible. So we tried a subset of the hyperparameters and used those results to decide the future experiments. Hyperparameter tuning is a challenge in all fields, in specific, deep neural networks, when it comes to transformers the problem further escalates.
- We searched for a pretrained model which had only attention layers but we could not find any such models. All the models that were available had MLP layers embedded, so the analysis of what the attention layers truly learned would not be fruitful. We tried to get the weights

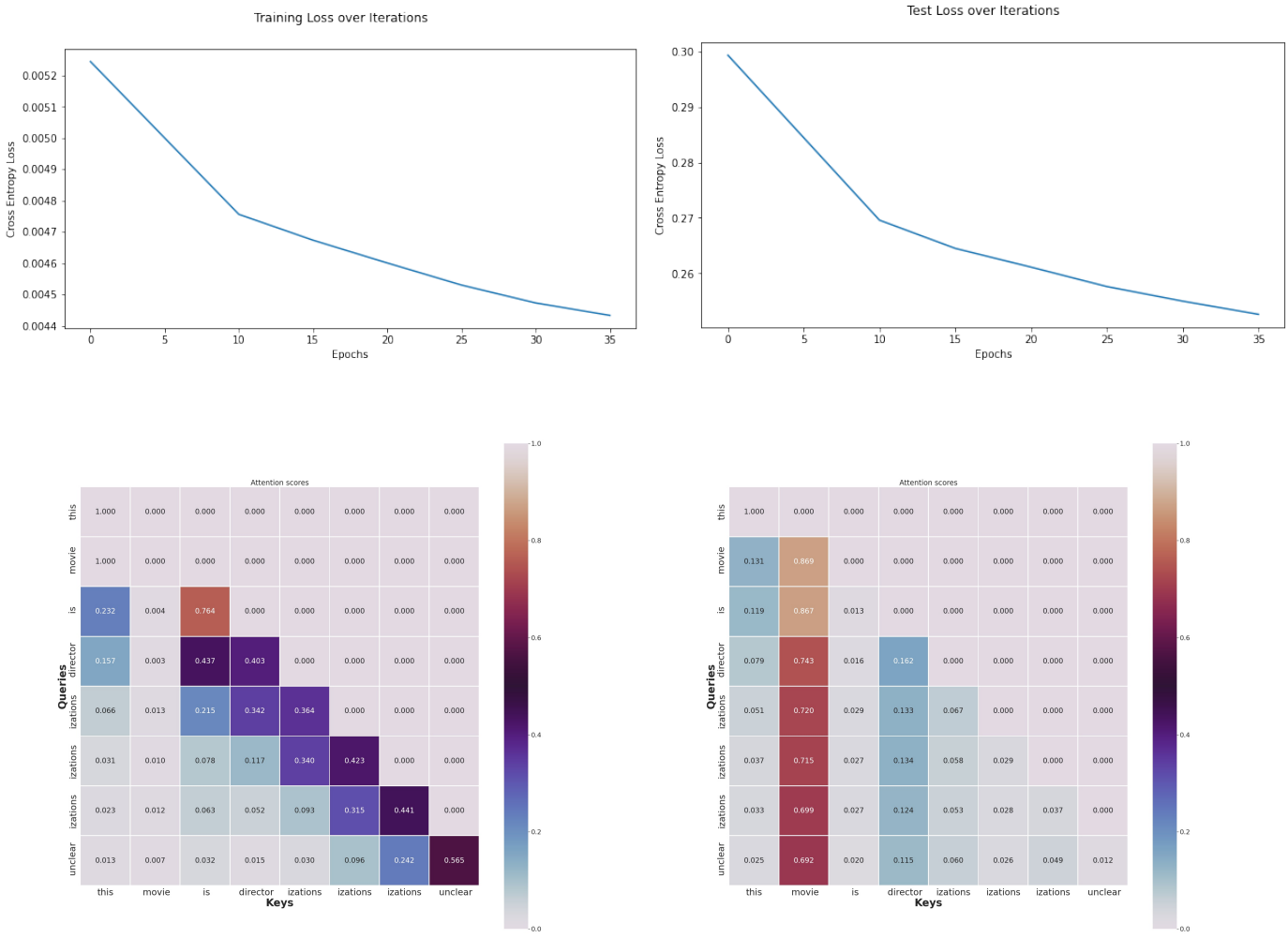

Fig. 9: Training Results on WikiText Dataset with Embed dims 256:- 1(Top-Left): Training Loss, 2(Top-right): Test Loss, 3, 4(Bottom) Attention score maps:

for some of these models, but even after extracting we could not directly use these weights in our model as the configuration of the model had to be exactly the same. Short experiments were conducted on GPT2 attention-only single-layer model pretrained and imported from huggingface, resulting in outputs non-meaningful outputs similar to our experimental outputs.

- We used Kaggle for all our experiments, Kaggle has an issue when we run experiments locally. It does not properly save the outputs in the directory, due to this we had to rerun the same experiment more than once. We checked on forums and stack exchange for solutions, the solution was to save the notebook and run on the server. However, at any time we could only run 2 experiments on the server.
- Since we wanted to perform a sweep over some of the hyperparameters, we used wandb. In order to use wandb, we had to change the flow of our code quite a bit. But when we ran the sweep, multiple sweeps crashed due to memory errors. So we could not run all the sweeps at once and were forced to run them one at a time.

XI. Conclusion and Future Work

We were able to successfully build a decoder-only transformer, the model is also correctly trained. But the training is not good enough for the attention layers to learn a lot of something which is interpretable. We realized that transformer training is data and computation hungry and it is not feasible to train a transformer using a single GPU. We tried using small datasets to train the model and also tried with large like wikitext. To save on the training time, we tried to train a single-layer attention model with the wikitext dataset.

A single layer would have lower parameters compared to the other experiment settings that we considered. After

analyzing the attention score heatmaps for the different experiments, we found that the attention layers were able to focus on the right context in a few heads, and in some cases, we were able to detect some copying mechanisms. However, this behavior cannot be generalized at the moment as the model has not trained well yet, but it does show what the attention layers are trying to do. The attention layers are understanding the context and then trying to predict the next token on that basis. This could stand as a proof of concept towards what could be done by performing analysis on the attention heads of transformer networks.

In this project, we were able to gain a very good understanding of transformers and also tried to analyze what the attention layers are learning. Due to challenges in training, we were not able to completely analyze the attention scores but we provided analysis based on the attention heatmaps we obtained. For future work, we need to work on training models which have only attention layers. This will provide all the researchers in the transformers domain with a very good starting point toward interpretability. We will also try to replicate the configurations of the models that we found so that we can use those weights to try and analyze the attention layers.

REFERENCES

[1] Zeiler, M. D., Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. arXiv, 1311.2901. Retrieved from https://arxiv.org/abs/1311.2901v3

[2] Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases. (2022, July 14). Retrieved from https://transformer-circuits.pub/2022/mech-interp-essay/index.html

[3] Cammarata, N., Carter, S., Goh, G., Olah, C., Petrov, M., Schubert, L., ...Lim, S. K. (2020). Thread: Circuits. Distill, 5(3), e24. doi: 10.23915/distill.00024

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ...Polosukhin, I. (2017). Attention Is All You Need. arXiv, 1706.03762. Retrieved from https://arxiv.org/abs/1706.03762v5

[5] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D.. Language Models are Unsupervised Multitask Learners. Retrieved from https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf . Retrieved from https://arxiv.org/abs/1406.2661v1