Anatomy and Structured Pruning of Large Language Models

Ryan Chen, Qichen Liu
W266: Natural Language Processing
UC Berkeley School of Information
{bread12035, qichen.liu}@berkeley.edu

Abstract

Large language models have recently achieved state-of-the-art performance across a wide variety of natural language tasks. Meanwhile, as the task of language models gets more complicated, the size of the models also grows drastically so that they become a black box and hard to deploy on edge devices. The mechanism of the Transformer has been fully addressed recently, but as the model gets complicated, the information delivery throughout layers of transformers remains unknown. In this paper, we will parse through both encoder-based BERT and decoder-based GPT2 having twelve layers of the Transformer stacked as the main body of the model, and demonstrate the information delivery of each layer of transformation and neurons in the feed-forward network (FFN) by comparing with the token fed to the classification layer to make the final decision. In the second part of this paper, we will develop a structured pruning strategy based on the information delivery of the models we got and further prune off the neurons in the feed-forward network (FFN) to solve the issue of latency and still maintain the capability of the models. Current compressing method like network pruning, knowledge distillation, low-rank approximation, and quantization inevitably requires post-training after actions to regain the performance of the model and this takes tons of computational power and time. Here, we will introduce a structured pruning strategy without any costly post-training step while it's also compatible with every method mentioned

1. Introduction

Oftentimes, scientists use the human brain as an analogy to a language model, yet there is no clear definition of the functionality of each part of the model like a brain has. We only define the basic structure of the Transformer but as for each layer of the Transformer stacked inside the model like GPT or BERT, their functionality and the information delivery remain unclear. Also, for the neurons in the feed-forward network (FFN), they would take the information with the size of embedding dimension from attention layers, expand it, and affine it back to the original size to digest and filter out the information that can help the model making the final decision. As a result, we can further dive into each neuron to get to know how the model works from the bottom. The recently published paper[1] proves that we can know the specific task of some neurons in the MLP layers of GPT2. Also, according to paper[2], it pointed out that it is challenging to train a pruned model by initializing the parameters randomly but if one can inherit the parameters from the original model, the training process seems viable which indicates only a certain portion of the model plays a major role of doing the task and others somehow idle at the time.

Pre-trained models have been widely used across different kinds of natural language tasks and have achieved splendid results in recent years. Along with that, the studies of compressing the model start with network pruning[3], knowledge distillation[4], and quantization[5]. These methods can successfully reduce the size of the model and speed up the inferring time but either way needs a post-training process and it takes lots of time and resources to do so. As a result, recent studies are focusing on trying to minimize the cost of post-training. For instance, structurally pruning Transformer networks[7], removing entire layers[8], pruning heads[8], intermediate dimensions[9,] and blocks in weight matrices[10]. Some may combine pruning and distillation by layer[11] to achieve

better performance and reduce the process of post-training. The concept of low-rank adaptation and approximation was published recently[12] and used for accelerating the process of fine-tuning the model and there is also research building up the strategy of pruning based on the concept of low-rank adaptation[13]. In this study, we will demonstrate a post-training free structured pruning strategy based on the importance of the neurons. Furthermore, it is also compatible with any method mentioned above and can be treated as the final polish of the work.

2. Background

2.1 Anatomy of the model

The major task of a decoder-based model like GPT2 is to take the last token of the output with the shape of embedding dimension and feed it to the softmax layer to find out the next token. Iteratively, the next token would combine the original input and generate new input. Between layers of the transformer, the input remains with the shape of batch size, sequence length, and embedding dimension as the original input. Intuitively, after each layer of transformation, the original input would gradually get similar to the final output. However, according to paper[1], the final decision tends to be made in the last layer of the transformer. As for the encoder-based model, the CLS tokens were used as a representation of the output sequence and have the shape of an embedding dimension. As a result, we can extract cls tokens from each layer of the Transformer and neurons inside of the feed-forward network (FFN) and compare them with the cls token of the last layer to identify their tasks. In this paper, we would like to not only delve into decoder-based model GPT but also encoder-based BERT to figure out the information delivery behind the hood.

2.2 Pruning methods

The studies of compressing the model start with network pruning[3], and knowledge distillation[4], quantization[5]. In the process of network pruning, the model would first be pruned randomly and it requires post-training to regain the capability of doing the task because, throughout the pruning process, the neurons responsible for making decisions may be removed as well. Next, the way knowledge distillation reduces the model is by letting a teacher model pass down its experience to a smaller student model. To be specific, take the supervised multiple-classification-task training as an example, The larger teacher model was trained by the labeled data but as for the student model, labeled data may not be enough for the smaller model to learn the pattern. As a result, the teacher model can feed not only the labeled data but also its processed result to the student model like the output of the softmax layer. 0.3 on class zero and 0.7 on class one would be much more insightful than simply providing 1 on class one as labeled data. As for quantization, it focuses on solving the computational bottleneck and speeding up the inference process. For example, conventionally in TensorFlow, we would save our weights with data type float32 which requires four bytes of memory and if we can transfer the data type from float32 to float8, ignoring some digits, we can accelerate the calculation of the matrix multiplication throughout the train and inferring process. As for the low-rank adaptation-based method, FLOP (Factorized Low-rank Pruning)[13], firstly, it would apply l_0 regularization when training to encourage zero in the matrix and create a sparse matrix. Next, apply low-rank approximation to simplify matrix W into the product of two smaller matrices. The way FLOP (Factorized Low-rank Pruning) skips post-training is by combining fine-tuning process and pruning which still requires time and resources. In this paper, we will demonstrate the structured pruning strategy developed around the functionalities of each part of the language model without any post-training step and still keep the capability of the models. Furthermore, the pruning strategy is compatible with any other methods mentioned and can be treated as the last polish of pruning.

3. Methods

3.1 Anatomy of each transformer layer

In this paper, we will adapt base BERT and GPT2 and corresponding tokenizers. Both of them have twelve layers of the Transformer as their main body. On top of the main body is a binary classification layer which incorporates a dropout layer, a dense layer, and a sigmoid layer for the sentiment analysis of movie reviews from IMDB. The linear layer and softmax layer after the GPT model were tossed away and there is no autoregressive operation since we only need a token for the classification layer to do their job. Next, we will fine-tune the model using IMDB review from the package of lfds with training data of twenty thousand and validation data of five thousand. The testing data has one thousand and will be used to evaluate the inference time and accuracy after pruning.

Before doing the analysis, we expected that there would be various tasks for each layer and to distinguish them from each other, we needed customized inputs to help us. As a result, we asked GPT4 to split the task of sentiment analysis into five sub-tasks and generate input sequences corresponding to them such as identifying emotional tone, recognizing extremes in sentiment, and detecting sarcasm and irony. Detailed inputs are shown in Table 1.

For the BERT model, we will take the cls token from the last layer of the Transformer as our target token with the shape of embedding dimension and extract every cls token from each layer of the Transformer and put them into comparison to figure out the information delivery. As for the GPT model, we will take the last token, logit, of the output sequence from the last layer of the Transformer and set it as the target token like what we did in BERT and put the logit token from each layer of the Transformer into comparison. We decided to apply cosine similarity as our metric because it's not only unified but also can both be positive and negative which is helpful in the following analysis.

3.2 Anatomy of neurons of the feed-forward network (FFN)

Next, We delved into the feed-forward network (FFN) and the second weights with the shape of neuron number and embedding dimensions which are responsible for selecting, and boiling down the patterns captured by attention layers. Each neuron has the shape of embedding dimensions so we can compare them with the cls token from BERT and the logit token from GPT. Here, we noticed that the information delivery inside the model is not only forwarding but also can be cut off at a certain layer like dangling nodes, all information is gathered to this node but somehow it doesn't pass down the result. Or there may be some situations in which one neuron is offset by another neuron in the following layer. As a result, we treated the information delivery as a graph problem. Lastly, We made a basic presumption that neurons having high cosine similarity (>0.35) in one layer are fully connected to the next layer and plotted the graph to further analyze the information delivery.

3.3 Randomness of the model

Originally, we expected that each neuron would have its job so the neuron presented would be different with each input, at least we can identify the neuron handling positive or negative by comparing the set of neurons in input 1 and 2. However, we noticed that for each input, the model almost uses the same set of neurons to handle the information and task Figure 2a, Figure 3a. For example, the neurons in the first layer for inputs 1 and 2 are the same, starting with numbers 75, 239, 283, and so on. This phenomenon is observed across all layers of the Transformer and for each input. The same phenomenon is found in the GPT model. It may be decided on the initialization stage of pre-training. On the other hand, the randomness may come from batching the fine-tuning data but it doesn't affect the pre-trained model, and each time we check out the neurons after fine-tuning, the result remains the same. So the high-level pattern observed can be the basis of our pruning strategy and it can be

applied to the same type of model. Also, we noticed that for each layer, there are less than five percent of neurons are being activated, so there is much room for further pruning.

3.4 Strategy for structured pruning

After an understanding of our language models in depth, we then plotted a strategy of pruning off the neurons around the result we got. Firstly, we preserved the important neurons and randomly pruned off neurons with certain portions and set it as the baseline. However, we noticed that even though some neurons have low cosine similarity, it still contributes to decision-making.

The BERT model has the habit of thinking and rethinking like the human nature of second thought. Furthermore, in every iteration, the number of neurons having high cosine similarity gets less and less and we also need to take the cosine similarity of the Transformer layer into account so we come up with a formula for the threshold of pruning.

$$Neuron\_similarity < abs\,(Layer\_similarity * alpha + Layer * beta)$$

Where alpha is the ratio of cosine similarity of the Transformer layer we take into account and beta is to describe the nature of model getting clear in each iteration. For the layer of thinking and making the decision, since the information is concentrated in certain neurons, we would make the beta higher. In these layers and for the last layer, all the information is gathered in one neuron so we make alpha three times higher. The number of neurons being pruned off is listed in Figure 4a.

As for the GPT model, it has the habit of capturing patterns in the first couple of layers and making the final decision in the last layer. As a result, we need to be careful in the first couple of layers and can ruthlessly prune off the neurons in the final layer of the Transformer since it already has its answer in a few neurons. Here we have the formula for the threshold of pruning for the GPT-based model.

$$Neuron\_similarity < abs\,(Layer\_similarity * alpha)$$

The way we pruned off the neurons is by zeroing out them, most GPUs can take benefits from the sparse data by saving only the non-zero items of the matrics and can accelerate the computation by applying the optimized algorithm. Lastly, we will evaluate the result by accuracy, inferring time for a thousand testing examples, and the number of neurons pruned off.

5. Results and Discussion

5.1 Anatomy of each transformer layer

The information flow for the BERT model and the GPT2 model of input 1 is shown in Figure 1. For every input, the plots are shown in Figure 1a

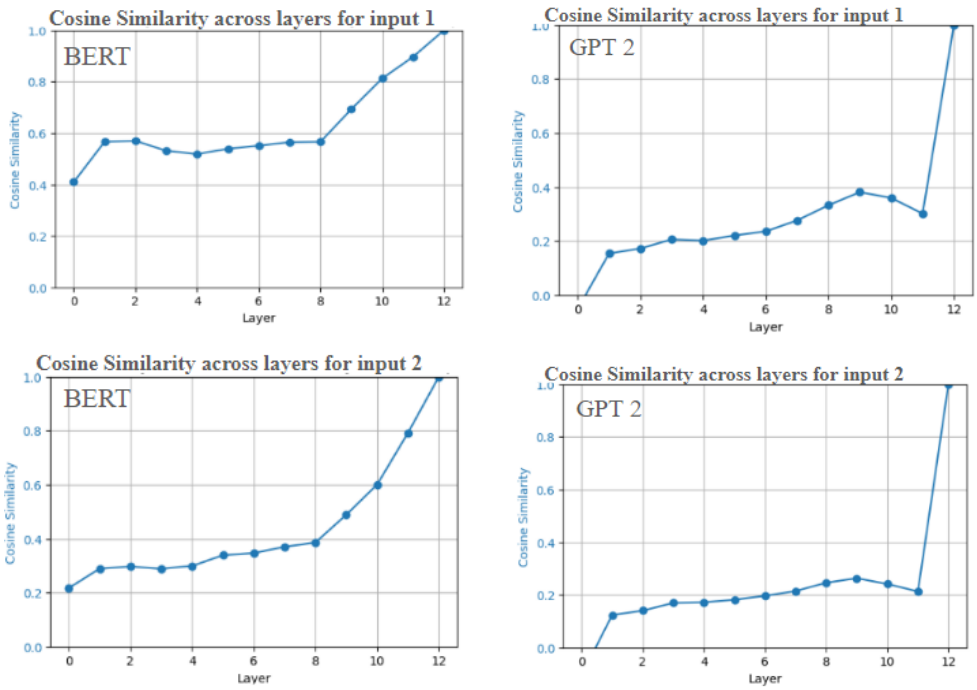


Figure 1. The plots on the left are cosine similarity of the target token and cls token from each layer of the Transformer for the BERT model. The plots on the right are for the GPT model. Layer0 represents the embedding layer and layer12 represents the final layer of the Transformer.

The sub-tasks of Input 1 and 2 identify positive and negative tone respectively. For input 1 of the BERT model, we can see cosine similarity increase a bit right after the first layer but in the following couples of layers, it keeps steady but not for input 2, the similarity somehow starts to increase slowly so we can say the BERT model is more sensitive to negative input over positive in the beginning. The cosine similarity starts to surge up after layer 8 which implies the BERT model is getting close to the final decision layer by layer.

As for the GPT model, The cosine similarity starts with a negative value which is reasonable since it's the last token of the input sequence without any process. Along with that, the cosine similarity increase slowly and gets a miner drop in layer 9 to 11 and all of a sudden shoot up to one. It can be attributed to the fact that the nature of the GPT model is to collect information and make the decision of the next token based on the information collected in the final layer. Likewise, for other inputs, the trend remains the same for both BERT and GPT models.

5.2 Anatomy of neurons of the feed-forward network (FFN)

Next, We delved into the feed-forward network (FFN) and the second weights. The following Figure 2 is the graph consisting of neurons of BERT with high cosine similarity.

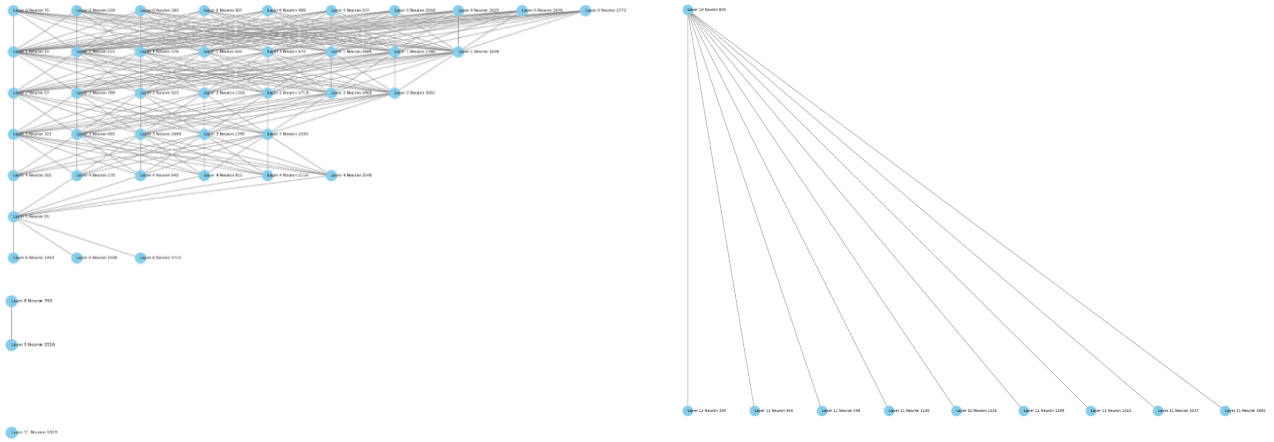


Figure 2 (Left). The BERT-model graph of neurons having cosine similarity larger than 0.35 with cls token. Neurons are lining up by layer and listed from layer 0 (top) to layer 11 (bottom) Figure 3(Right). The GPT 2-model graph of neurons having cosine similarity larger than 0.35 with logit token. Neurons are lined up by layer and listed from layer 0 (top) to layer 11 (bottom)

In Figure 2., we can see the number of neurons having cosine similarity larger than 0.35 decrease along with layers but something worth mentioning is that there are funnel shapes spotted between layers 2,3,4 and layer 4,5,6. In these layers, the model seems to approach the task by thinking and rethinking iteratively and each time it rethinks the problem, the cosine similarity of neurons gets higher. This phenomenon is insightful since we never taught the BERT model to do so and We believe the way it approaches the task stems from the way it was trained. By filling the space and doing the bidirectional training, the model seems to act like how humans handle information on second thoughts.

On the other hand, the GPT model handles the information differently. In Figure 3., there are no neurons having cosine similarity larger than 0.35 in the first couple of layers and the model seems to be collecting patterns from the information. Until layer 10 and layer 11, all of a sudden, neurons having high cosine similarity pop up which implies the mechanism of the GPT model. It collects information and based on the information it got, makes the final decision in the last layer of the Transformer.

5.3 Strategy for structured pruning

The way FLOP (Factorized Low-rank Pruning)[13] reduces the model size is by encouraging zero in wights and concentrating the information into low-rank approximations. As for our strategy of structure pruning, we targeted and pruned the neurons like Targeted therapy used to cure cancers. Hence, It's compatible with any methods mentioned and can be treated as the final polish of the pruning process. Figure 4. Shows the result of the BERT model after structured pruning. The arbitrary pruning would be set as the baseline and structured pruning as the experiment.

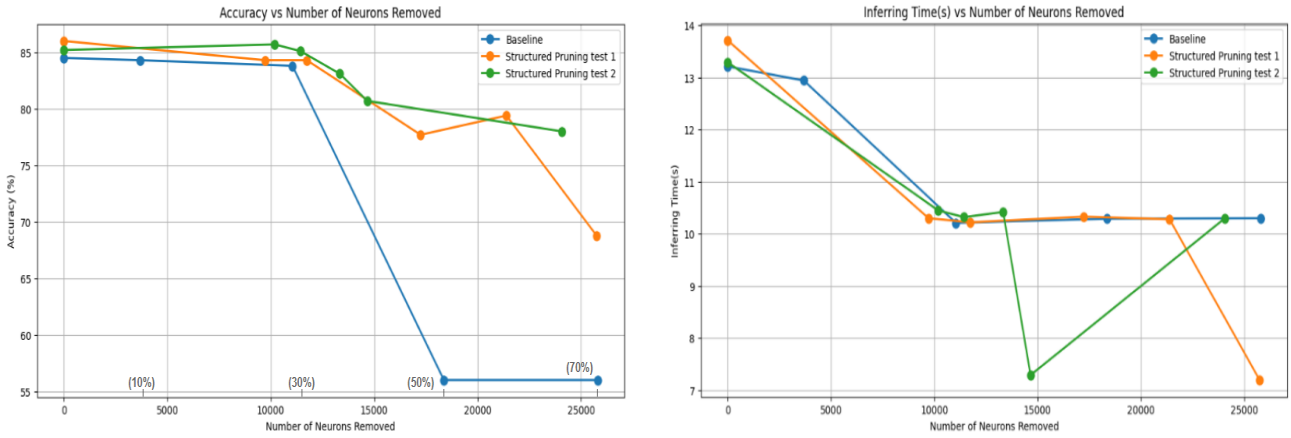


Figure 4 (Left). The accuracy of the BERT model after structured pruning in this work. Figure 4-1(Right). The inferring time for a thousand testing examples of the BERT model after structured pruning in this work.

From Figure 4-1, the accuracy of the BERT model after pruning arbitrarily except for the preserved neurons keeps steady even after pruning off 30% of the neurons. However, it drops drastically after pruning off 50% of the neurons which can stem from the fact that information delivery is interrupted. Some neurons with low cosine similarity may still play a major role in decision-making or information delivery. On the other hand, if we prune off the neurons by the well-designed strategy, we can mitigate the loss of performance even after pruning off 30% of the model, the accuracy maintained at the same level from 85.2% to 85.1% and drops to 78% after removing 70% of neurons. Meanwhile, the accuracy of the baseline model drops to 58.2% after removing 50% of neurons. The structure pruning strategy works very well on the BERT model. As for the inferring time for processing a thousand testing examples, it drops from 13.71 seconds to 10.3 seconds after pruning off 10% of the neurons and has stopped improving ever since. The reason could be the GPU we use (T4 GPU) can't make use of a sparse matrix though we have zeroed out vast amounts of neurons. Here, if we can physically prune off the matrix and reduce the size of the matrix or apply quantization in this case, we believe we can have a faster inferring time.

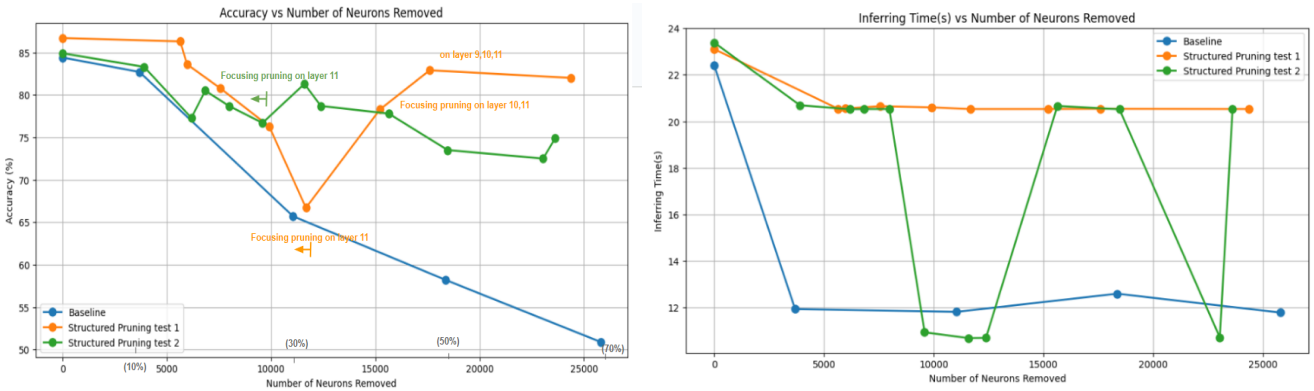


Figure 5 (Left). The accuracy of the GPT 2 model after structured pruning in this work.
Figure 5-1 (Right). The inferring time for a thousand testing examples of the GPT 2 model after structured pruning in this work.

As for the GPT model in Figure 5., the baseline model drops when the number of neurons removed increases. On the other hand, experiments of structural pruning also drop along with the increment of the number of neurons removed in the first couple of layers. Originally, we focused on pruning the last layer of the Transformer because GPT 2 has the nature of deciding in the last layer and the information at the layer is highly focused on a few neurons, but when we finished the pruning of the last layer, We started to advance to the lower layers and surprisingly noticed that the accuracy went up rather than down as expected. We think it may be because there are negative cosine similarities spotted suggesting the offset effect between layers. Information in one layer may be canceled out by the previous layer. As for inferring time, it seems to be stuck at the level of 20 seconds from 23 seconds which can be improved by applying other techniques instead of zeroing out. The detailed experiment setup is listed in Figure 5a.

6. Conclusion

In the first section, we demonstrated the information delivery of the BERT model and the GPT 2 model both having 12 layers of the Transformer. Surprisingly, we found that the BERT model approaches the task by thinking and rethinking which we've never taught the model to do so. On the other hand, the GPT 2 model applies a different way of handling information, it captures the pattern from the data and makes the decision in the last layer of the Transformer. We then adapted the graph to describe the information delivery and point out there are many types of information flow. We have 768 embedding dimensions, so cosine similarity is just a comprehensive comparison across embedding dimensions, and it can be further improved in future work. Next, we built up the pruning strategy based on the findings we got and significantly beat the baseline. We speed up 21.3% of the inferring time for 0.1% accuracy loss. The method we designed can be adapted after any structured pruning methods are up to date without any further post-training to improve the latency issue.

7. References

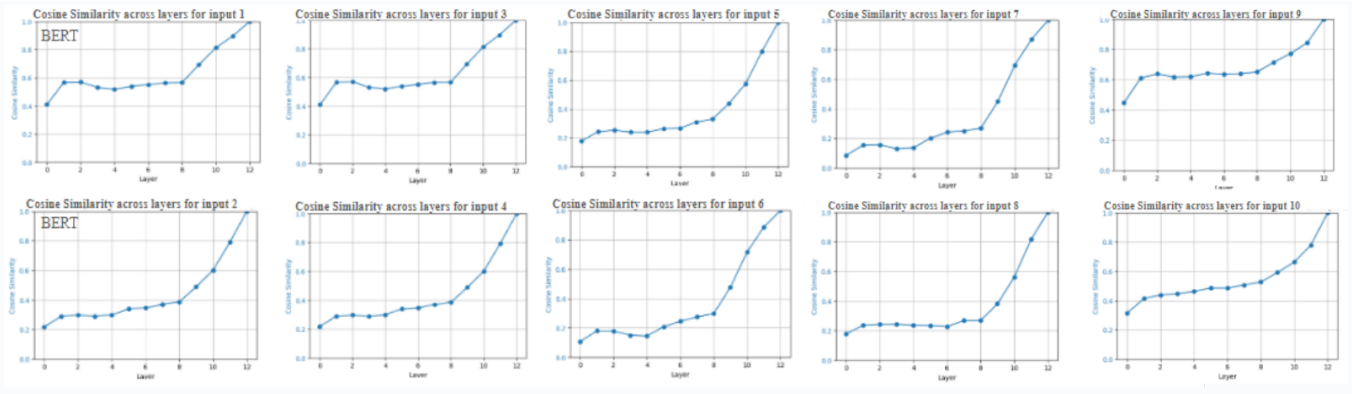
1. Joseph Miller, Clement Neo. 2023. We Found An Neuron in GPT-2. Blog post of Clement Neo
2. Jonathan Frankle, Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. arXiv:1803.03635
3. Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. arXiv:1608.03665v4
4. Geoffrey Hinton, Oriol Vinyals, Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531
5. Yuri Nahshan YairHanani Ron Banner DanielSoudryImproving 2020. Post Training Neural Quantization: Layer-wise Calibration and Integer Programming. arXiv:2006.10518v2
6. Mengzhou Xia ZexuanZhong DanqiChen 2022. Structured Pruning Learns Compact and Accurate Models. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics Volume 1: Long Papers, pages 1513- 1528
7. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in Neural Information Process ing Systems (NIPS), 30:5998–6008.
8. Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing Transformer depth on demand with struc tured dropout. In International Conference on Learn ing Representations (ICLR).
9. Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020b. Early bert: Efficient bert training via early-bird lottery tick ets. arXiv preprint arXiv:2101.00063.

10. François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. 2021. Block pruning for faster transformers. arXiv preprint arXiv:2109.04838.
11. Ziheng Wang, Jeremy Wohlwend, Tao Lei. 2021. Structured Pruning Learns Compact and Accurate Models. arXiv:1910.04732v2
12. Edward Hu Yuanzhi Li Yelong Shen Shean Wang Microsoft Corporation Phillip Wallis Lu Wang Zeyuan Allen-Zhu Weizhu Chen. 2021. LORA: LOW-RANK ADAPTATION OF LARGE LAN GUAGE MODELS. arXiv:2106.09685v2
13. Ziheng Wang, Jeremy Wohlwend, Tao Lei. 2021. Structured Pruning of Large Language Models arXiv:1910.04732v2

8.Appendix

Input number	Sub-task	Inputs
0	Identifying Emotional Tone-Positive	The movie's breathtaking scenery and exceptional soundtrack added depth to its rich storytelling
1	Identifying Emotional Tone-Negative	The film was a letdown with its lackluster plot and uninspired performances.
2	Analyzing Subjective Statements-Subjective	In my opinion, the film's portrayal of historical events was highly inaccurate.
3	Analyzing Subjective Statements-Objective	The movie won three Academy Awards, including Best Picture.
4	Evaluating Specific Aspects-Positive acting	The acting was superb, with each character bringing depth and emotion to the screen.
5	Evaluating Specific Aspects-Negative acting	The plot was predictable and lacked originality, making the movie quite boring.
6	Recognizing Extremes in Sentiment-Extremely Negative	This is possibly the worst movie ever made, with no redeeming qualities whatsoever.
7	Recognizing Extremes in Sentiment-Extremely Positive	An absolute masterpiece, every moment was captivating and a joy to watch.
8	Detecting Sarcasm or Irony-Sarcasm	Oh great, another predictable rom-com, just what the world needs.
9	Detecting Sarcasm or Irony-Irony	I loved how the movie ended abruptly without resolving any plot points

Table 1. Inputs of each sub-tasks of sentiment analysis of movie review generated by commercial GPT 4 plus.



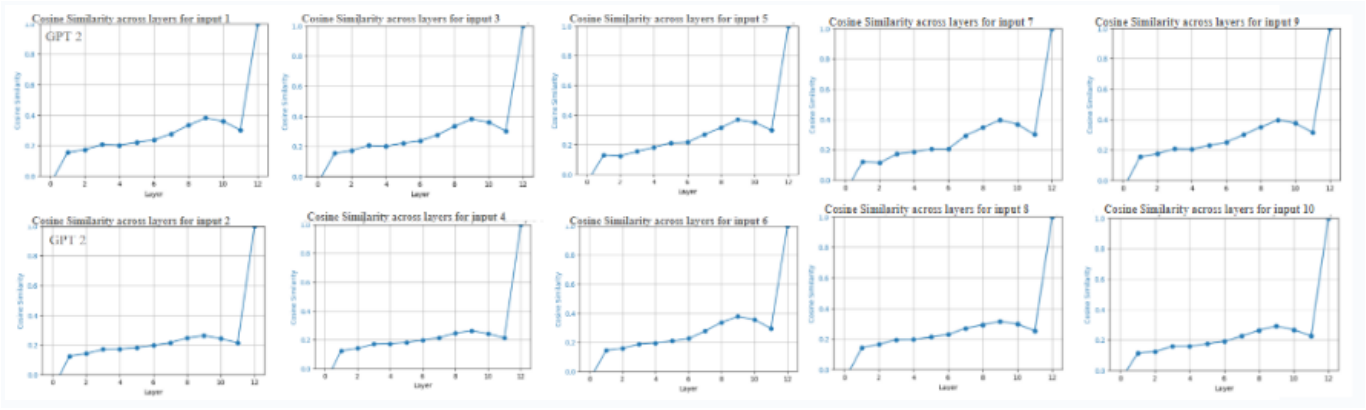


Figure 1a. The plots on the top are cosine similarity of the target token for every input and cls token from each layer of the Transformer for the BERT model. The plots on the bottom are for the GPT model. Layer0 represents the embedding layer and layer12 represents the final layer of the Transformer.

==========Input 0==========
Layer 0, Neurons: ['Layer 0 Neuron 75', 'Layer 0 Neuron 239', 'Layer 0 Neuron 283', 'Layer 0 Neuron 307', 'Layer 0 Neuron 489', 'Layer 0 Neuron 537', 'Layer 0 Neuron 2018', 'Layer 0 Neuron 2625', 'Layer 0 Neuron 2670', 'Layer 0 Neuron 2773']
Layer 1, Neurons: ['Layer 1 Neuron 11', 'Layer 1 Neuron 113', 'Layer 1 Neuron 376', 'Layer 1 Neuron 524', 'Layer 1 Neuron 672', 'Layer 1 Neuron 1685', 'Layer 1 Neuron 1766', 'Layer 1 Neuron 1838']
Layer 2, Neurons: ['Layer 2 Neuron 57', 'Layer 2 Neuron 789', 'Layer 2 Neuron 923', 'Layer 2 Neuron 1316', 'Layer 2 Neuron 2719', 'Layer 2 Neuron 2801', 'Layer 2 Neuron 3062']
Layer 3, Neurons: ['Layer 3 Neuron 323', 'Layer 3 Neuron 693', 'Layer 3 Neuron 1489', 'Layer 3 Neuron 1795', 'Layer 3 Neuron 2233']
Layer 4, Neurons: ['Layer 4 Neuron 102', 'Layer 4 Neuron 170', 'Layer 4 Neuron 642', 'Layer 4 Neuron 915', 'Layer 4 Neuron 2116', 'Layer 4 Neuron 2548']
Layer 5, Neurons: ['Layer 5 Neuron 55']
Layer 6, Neurons: ['Layer 6 Neuron 1443', 'Layer 6 Neuron 1506', 'Layer 6 Neuron 1712']
Layer 7, Neurons: []
Layer 8, Neurons: ['Layer 8 Neuron 763']
Layer 9, Neurons: ['Layer 9 Neuron 2528']
Layer 10, Neurons: []
Layer 11, Neurons: ['Layer 11 Neuron 1023']

==========Input 1==========
Layer 0, Neurons: ['Layer 0 Neuron 75', 'Layer 0 Neuron 283', 'Layer 0 Neuron 489', 'Layer 0 Neuron 2625', 'Layer 0 Neuron 2773']
Layer 1, Neurons: ['Layer 1 Neuron 1685', 'Layer 1 Neuron 1766']
Layer 2, Neurons: ['Layer 2 Neuron 789']
Layer 3, Neurons: ['Layer 3 Neuron 693']
Layer 4, Neurons: ['Layer 4 Neuron 170']
Layer 5, Neurons: []
Layer 6, Neurons: ['Layer 6 Neuron 1443']
Layer 7, Neurons: []
Layer 8, Neurons: ['Layer 8 Neuron 763']
Layer 9, Neurons: ['Layer 9 Neuron 2528']
Layer 10, Neurons: []
Layer 11, Neurons: []

==========Input 2==========
Layer 0, Neurons: ['Layer 0 Neuron 75', 'Layer 0 Neuron 239', 'Layer 0 Neuron 283', 'Layer 0 Neuron 307', 'Layer 0 Neuron 489', 'Layer 0 Neuron 537', 'Layer 0 Neuron 2018', 'Layer 0 Neuron 2625', 'Layer 0 Neuron 2670', 'Layer 0 Neuron 2773']
Layer 1, Neurons: ['Layer 1 Neuron 11', 'Layer 1 Neuron 113', 'Layer 1 Neuron 376', 'Layer 1 Neuron 524', 'Layer 1 Neuron 672', 'Layer 1 Neuron 1685', 'Layer 1 Neuron 1766', 'Layer 1 Neuron 1838']
Layer 2, Neurons: ['Layer 2 Neuron 57', 'Layer 2 Neuron 789', 'Layer 2 Neuron 923', 'Layer 2 Neuron 1316', 'Layer 2 Neuron 2719', 'Layer 2 Neuron 2801', 'Layer 2 Neuron 3062']
Layer 3, Neurons: ['Layer 3 Neuron 323', 'Layer 3 Neuron 693', 'Layer 3 Neuron 1489', 'Layer 3 Neuron 1795', 'Layer 3 Neuron 2233']
Layer 4, Neurons: ['Layer 4 Neuron 102', 'Layer 4 Neuron 170', 'Layer 4 Neuron 642', 'Layer 4 Neuron 915', 'Layer 4 Neuron 2116', 'Layer 4 Neuron 2548']
Layer 5, Neurons: ['Layer 5 Neuron 55']
Layer 6, Neurons: ['Layer 6 Neuron 1443', 'Layer 6 Neuron 1506', 'Layer 6 Neuron 1712']
Layer 7, Neurons: []
Layer 8, Neurons: ['Layer 8 Neuron 763']
Layer 9, Neurons: ['Layer 9 Neuron 2528']
Layer 10, Neurons: []
Layer 11, Neurons: ['Layer 11 Neuron 1023']

Figure 2a. Detailed number of BERT neurons with high cosine similarity for different inputs. List only input 0 to 2 for space limits

==========Input 0==========
Layer 0, Neurons: []
Layer 1, Neurons: []
Layer 2, Neurons: []
Layer 3, Neurons: []
Layer 4, Neurons: []
Layer 5, Neurons: []
Layer 6, Neurons: []
Layer 7, Neurons: []
Layer 8, Neurons: []
Layer 9, Neurons: []
Layer 10, Neurons: ['Layer 10 Neuron 804']
Layer 11, Neurons: ['Layer 11 Neuron 309', 'Layer 11 Neuron 458', 'Layer 11 Neuron 598', 'Layer 11 Neuron 1120', 'Layer 11 Neuron 1126', 'Layer 11 Neuron 1289', 'Layer 11 Neuron 1322', 'Layer 11 Neuron 1637', 'Layer 11 Neuron 1860']

==========Input 1==========
Layer 0, Neurons: []
Layer 1, Neurons: []
Layer 2, Neurons: []
Layer 3, Neurons: []
Layer 4, Neurons: []
Layer 5, Neurons: []
Layer 6, Neurons: []
Layer 7, Neurons: []
Layer 8, Neurons: []
Layer 9, Neurons: []
Layer 10, Neurons: ['Layer 10 Neuron 804', 'Layer 10 Neuron 1410', 'Layer 10 Neuron 1521', 'Layer 10 Neuron 1783', 'Layer 10 Neuron 2931']
Layer 11, Neurons: ['Layer 11 Neuron 309', 'Layer 11 Neuron 458', 'Layer 11 Neuron 480', 'Layer 11 Neuron 598', 'Layer 11 Neuron 796', 'Layer 11 Neuron 847', 'Layer 11 Neuron 869', 'Layer 11 Neuron 975', 'Layer 11 Neuron 1120', 'Layer 11 Neuron 1126', 'Layer 11 Neuron 1231', 'Layer 11 Neuron 1289', 'Layer 11 Neuron 1322', 'Layer 11 Neuron 1428', 'Layer 11 Neuron 1575', 'Layer 11 Neuron 1637', 'Layer 11 Neuron 1860', 'Layer 11 Neuron 1998', 'Layer 11 Neuron 2375', 'Layer 11 Neuron 2378', 'Layer 11 Neuron 2600', 'Layer 11 Neuron 2822']

==========Input 2==========
Layer 0, Neurons: []
Layer 1, Neurons: []
Layer 2, Neurons: []
Layer 3, Neurons: []
Layer 4, Neurons: []
Layer 5, Neurons: []
Layer 6, Neurons: []
Layer 7, Neurons: []
Layer 8, Neurons: []
Layer 9, Neurons: []
Layer 10, Neurons: ['Layer 10 Neuron 804']
Layer 11, Neurons: ['Layer 11 Neuron 309', 'Layer 11 Neuron 458', 'Layer 11 Neuron 598', 'Layer 11 Neuron 1120', 'Layer 11 Neuron 1126', 'Layer 11 Neuron 1289', 'Layer 11 Neuron 1322', 'Layer 11 Neuron 1637', 'Layer 11 Neuron 1860']

Figure 3a. Detailed number of GPT neurons with high cosine similarity for different inputs. List only input 0 to 2 for space limits

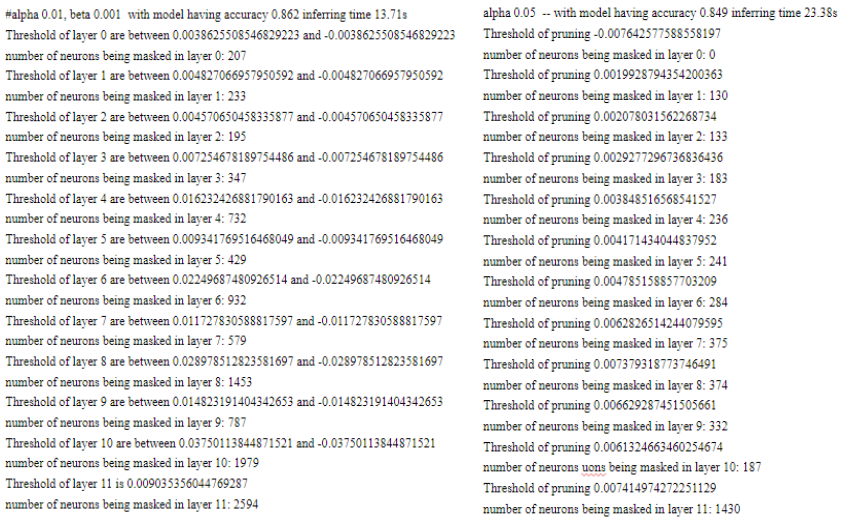


Figure 4a. Neurons of BERT being pruned by layers (Left), Neurons of GPT 2 being pruned by layers (Right)

Model	Alpha	Beta	Focusing layers
BERT	0.01	0.001	4,6,8,10,11
BERT	0.025	0.001	4,6,8,10,11
BERT	0.05	0.001	4,6,8,10,11
BERT	0.05	0.005	4,6,8,10,11
BERT	0.05	0.01	4,6,8,10,11
GPT 2	0.05	N/A	11
GPT 2	0.075	N/A	11
GPT 2	0.1	N/A	11
GPT 2	0.125	N/A	11
GPT 2	0.15	N/A	11
GPT 2	0.15	N/A	10,11
GPT 2	0.15	N/A	8,9,10,11

Figure 5a. Experiment setup.