Deep Head: Self-Attention for Rich Hierarchical Feature Representations

Michael Cruz [*1] **Marcus Fernandez Decastro** [*1] **Samuel Duval** [*1]
https://github.com/mbcruz96/Deep-Head

Abstract

Natural Language Processing (NLP) and Computer Vision (CV) are rapidly evolving machine learning sectors. New techniques and technologies are being innovated in these fields which have revolutionized the technology industry. This is due, in large part, to both the availability of vast amounts of data from the internet as well as essential learning model backbones that have facilitated the technology boom. In particular, the creation of the Transformer and attention mechanism have been cornerstones to the increased popularity in this research sector. The transformer benefits both NLP and CV; numerous up-and-coming models use this architecture as a backbone to create technology that would be impossible 10 years ago. Through experimentation, we aim to create an innovative transformer architecture with the capability to generate enhanced feature representations of both language and image modalities. Unlike conventional Transformers that utilize a constant number of attention heads per transformer in the encoder and decoder modules, the model we propose adjusts the number of attention heads across different transformer layers in the encoder, with the objective to create a rich feature hierarchy that improves performance on all downstream tasks. Through a comprehensive analysis of experimental results, we endeavor to shows the effectiveness of our approach compared to the traditional Transformer architecture.

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of Central Florida, Orlando, USA. Correspondence to: Michael Cruz <mi484725@ucf.edu>, Marcus Fernandez Decastro <mdecastro@ucf.edu>, Samuel Duval <sa565904@ucf.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

The Transformer architecture was a novel concept introduced to combat the inefficiencies introduced by Recurrent Neural Networks (RNNs) and Long-Term Short Memory (LSTM) models [2] [3]. In particular, the architecture of these models is not conducive to parallelized training techniques. Constant efforts have been made in the fields of machine learning particularly natural language processing and computer vision to rectify this issue and the answer was self-attention. This mechanism allows for sequence data to be processed in parallel. Instead of computing the hidden layers of all tokens collectively at each step like former models, self-attention computes the attention scores between each token and all other tokens separately, allowing the process to be computed in parallel. This scheme computes the weighting of the significance of different parts of input data, which directs the model to capture contextual relationships between tokens, regardless of the distance between tokens within the sequence. While transformers have helped advance various fields, its architecture, particularly this self-attention mechanism, has the potential for improvement.

This paper explores the potential of variability within multi-head self-attention in the transformer encoder framework. We hypothesize that varying the number of attention heads per layer, allows the model to analyze both local and global input features in efforts to generate hierarchical representation of encoded features. In this scheme, initial layers capture a better understanding of local features, and later layers analyze the entire input allowing a better comprehension of global features while also integrating the local features into a hierarchical understanding of features. We suggest an architecture where initial layers have more attention heads, and later layers uniformly decrease the number heads. This architecture could lead to an increase in performance in complex tasks due to its closer alignment with how information is processed cognitively. This would also improve the performance of all other models using Transformers as an encoder backbone.

1.1. Background

Since transformers have been introduced, many different adaptations have been made, but most focus on scaling the model or making its components more optimized for a specific task. Little work has been done to more closely analyze the self attention mechanism itself. Most current transformer models apply a uniform, fixed number of attention heads at all layers, which was likely done to make the model more computationally efficient, rather than a more optimal model.

Our novel approach more closely represents neural processing seen in biological systems, where sensory information is processed at different resolutions and integrated and different stages. We suggest that transformers can also benefit from this type of processing. This approach could potentially allow for the model to first learn detailed information, and then progressively apply these details into higher level features.

2. Related Work

2.1. Recurrent Neural Networks

Many models have attempted to solve problems relating to processing sequential data, or time-step data. One of the original models that implemented a method of incorporating sequential data into a model was the Recurrent Neural Network (RNN) architecture [2]. RNNs sequentially process inputs in order. Each input is processed and produces a hidden layer as well as an output from that time step. The computation for each subsequent input is dependent on that input as well as the hidden layer of the previous input. This method allowed for sequential data to maintain its ordering during training. However, there were many inherent problems in this method. For instance, when training text data, there is no way to determine long range dependencies between words in a sequence. In other words, because the token being processed only depends on the previous token, there is no way to determine if there is a relationship between tokens at the beginning of the sequence and tokens at the end. Additionally, RNNs require data to be processed one step at a time, meaning a token can only be processed after the token directly preceding it. This causes training of the model to be much longer.

2.2. Long-Term Short Memory

The Long-Term Short Memory (LSTM) model [3] built upon the objective of RNNs to create a model with the ability to process sequential data. However, unlike RNNs, LSTMs employ a mechanism that observes long-term dependencies between tokens. Each time-step of an LSTM processes input data and produces an output and a cell state representing the information relevant to the current token. These cell states are passed to the next time-step compute the next cell state through three information gates gates; A forget gate that determines what information to keep from the previous gate, an input gate which weighs the importance of the current input data, and an output gate which determines the output of the current time step. LSTMs not only solve the issue of long-term dependencies within input data, but also prevent vanishing and exploding gradients. In traditional RNNs, the backpropagation method requires layers to propagate information in the reverse order of forward propagation. Consequently, as the gradients of each layer are computed, the gradient can either get extremely small or extremely large, resulting in vanishing or exploding gradients respectively. The implementation of gates resolved this issue, making LSTMs perform better than RNNS. However, LSTMs are still slow to train because, like RNNs, the data must be processed in sequential order, preventing the parallelization of computation.

2.3. Transformers

Transformers have become the building block of most NLP and CV models [4] due to their impressive performance and computational efficiency. For each token in a sequence, the model computes the dot product between that token and every other token in the sequence in parallel. This technique is called self-attention [6] and is the basis for the Transformer architecture. It allows for parallel data processing which drastically improves the computation efficiency of the model. In addition, because scores are calculated between each token and every other token, long-term dependencies are explicitly defined and observed as opposed to LSTMs. Using the attention technique, Transformers utilize multiple heads, each receiving a certain segment of the input sequence [1]. Different heads analyze different sequence features and, at the end, concatenate all features together to represent all the features in the sequence. Current models use a constant number of attention heads at each encoder and decoder in the model, which may hinder the model in its performance on various tasks.

2.4. Transformer Variants

Some previous works have tried to tweak the attention mechanism for better performance, similar to the task detailed here. For example, [6] introduced an external memory unit to help assist self-attention with historical attention information. Similarly, [5] showed the importance in varying different heads, making the suggestion that some heads could be ”pruned” without hurting performance.

Building on these similar works, this work strays away from conventional attention head architecture and presents a dynamic and hierarchical structure for the attention heads.

3. Proposed Method

Building on the existing transformer architecture, we propose a modified structure that allows for the implementation of variable numbers of attention heads at each attention block. Traditional transformer models use a fixed number of attention heads at each layer, constraining the model to capture features at the same level of granularity. Our model starts feature extraction with 64 attention heads and then divides the the number of heads in half at each subsequent layer until the final layer which has 2 attention heads.

The idea behind the proposed architecture stems from the hypothesis that allowing more heads at initial layers allows the model to handle a more diverse set of features within the data, similar to a high-resolution scan. This could help the model identify more intricate patterns and dependencies. By reducing the number of heads over time, the model is forced to consolidate these features, helping the model to generalize from more specific details, allowing a bigger picture to form. At the same time, the later layers use the information learned from earlier heads to see the entirety of the data.

To help facilitate this hierarchical processing, we introduce an adaptive scaling mechanism for the attention heads. Each transformer layer's dimensional is divided by the dynamic number of heads, represented by

$$d_k = \frac{d_{model}}{number of heads} \tag{1}$$

This scaling factor helps ensure that each head can focus on a segment of the input features, and as the number of heads is reduced, the remaining heads can broaden the scope and extract more general features. Skip connections were also implemented between the final cross-attention scores of successive transforms. This allows for the feature hierarchies to aggregate at different levels and ensure a more robust feature representation.

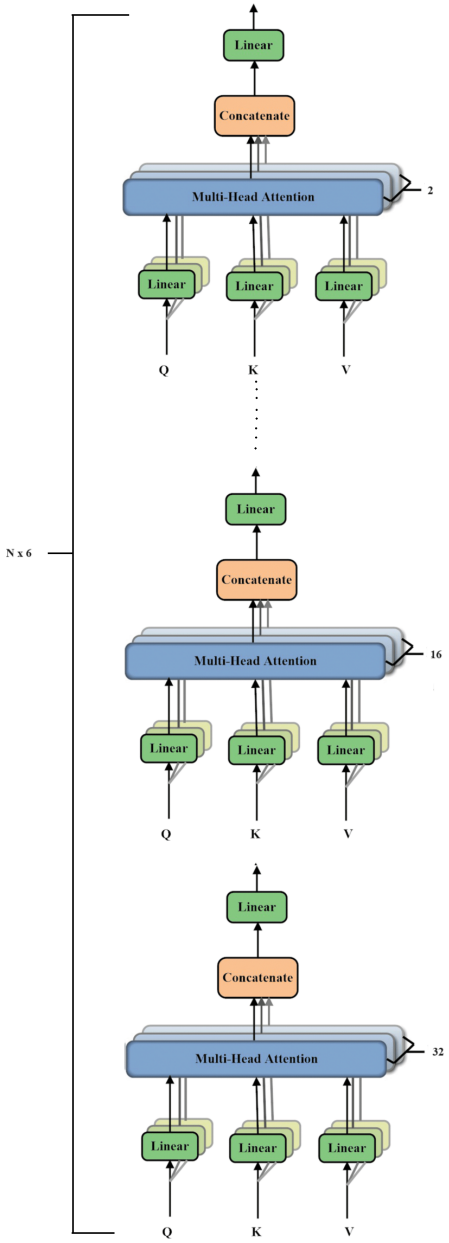

Figure 1. Deep Head Variable Heads Architecture

3.1. Methodology

In order to evaluate the effectiveness of the newly proposed transformer model, a clear and comprehensive experimental setup was established to ensure there were no issues in testing.

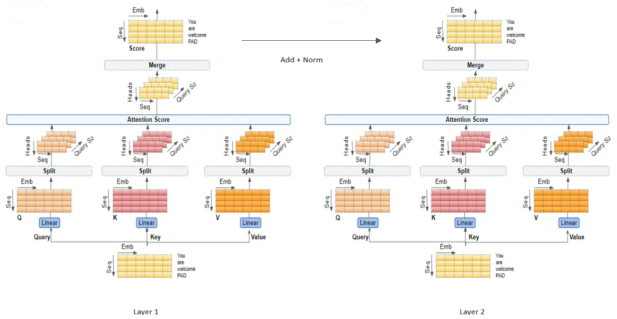

Figure 2. Deep Head Skip Connection

The Deep Head model was configured with varying numbers of attention heads across its layers. The model started with an initial 64 heads, then reduced the following layers by half the number of heads for each subsequent layer, until the last layer which has 2 attention heads as seen in Figure 1. The dimensionality of the model was kept constant to maintain uniformity in the size of the feature representations, as well as to ensure the attention scores of each layer would be able to be added to the next layer and normalized. To facilitate the adding and normalization of attention score between Deep Head levels, we engineered a skip connection between a layer and the layer directly after it. The skip connection sends the attention scores from the previous layer to the next to be added and normalized in order to preserve localized features from previous layers in later layers as demonstrated in Figure 2. The mode was then trained on the same dataset used by the original authors in [4]. This was done to maintain consistency in order to ensure our model, would be performing on the same data used by the original transformer model. The training process was conducted using the UCF ARCC cluster to leverage the computational power needed to train deep learning models with more heads than the original transformer model. Each model took around 3 days to train. Hyperparameters were tuned to find the optimal settings for the learning rate, batch size, and attention head scaling. We used a grid search for this purpose, and the best performing hyperparameters were selected based on the validation set performance. Many parameters were kept the same as the original transformer model, in order to maintain consistency, and to ensure the model is not specialized to any specific parameter or dataset.

The model's performance was evaluated against the SOTA Transformer that use a fixed number of attention heads. A variety of metrics suited to the specific translation task were computed on each models' performance, such as BLEU score for language tasks, character error rate, and word error rate. The software stack included PyTorch for model implementation and training as well as various other python packages used for data handling and visualization.

4. Experiment

The experimentation for this model was intended to directly compare the performance of the Deep Head model against the SOTA Transformer [4]. To accurately demonstrate the performance of each model, we recreated the architecture of the original Transformer model instead of using the Transformer library from HuggingFace. This allowed us to add the components of the Deep Head model to the original architecture, ensuring the models had similar functionality. Both models were trained on the same data set and then subsequently were tested on a validation set for the metrics which will be reported in this paper. Specifically, BLEU scores, character error rates, and word error rates were computed for each model to compare their performance on text-to-text translation. For this experiment, the models were trained on the Opus books English to Spanish dataset from HuggingFace instead of the English to German dataset from the original Transformer experiment. The English to Spanish dataset was used because we have individuals fluent in Spanish on the development team which allowed us to qualitatively analyze the both models inferencing abilities better.

4.1. Results

When evaluating our proposed transformer model with dynamically varying attention heads, we conducted a series of experiments aimed at comparing our model against the baseline transformer model with a static number of attention heads. The baseline model was configured as closely as possible to the original transformer model with the fixed attention heads. Our evaluation included several benchmarks and translation tasks in order to assess the performance variations that came from the variation of attention heads. Quantitative analysis showed that our model performs similarly to the baseline. Specifically in language translation tasks, BLEU scores of our dynamic head model were very similar to those of the baseline. In our accuracy metrics, there also showed only small deviations between the two models. The results collected were consistent across multiple runs with varying data samples, although we were not able to complete as many runs as desired due to time constraints. Although our model did not exceed the performance of the baseline when comparing the primary metrics, we were able to successfully implement a different type of attention allocation, which could suggest a more nuanced feature processing capability. Qualitative observations of the attention maps confirmed these conclusions, as the ini-

tial layers with more heads attended to a wider array of features, while the next layers with fewer heads focused on integrating these features.

5. Conclusion

Our initial hypothesis of the research was that a transformer model with a dynamic distribution of attention heads across layers would outperform a traditional model with static heads. The empirical results from the experiments refute this hypothesis, and show equivalent results to the baseline model. We are able to draw a few conclusions from this outcome. First, this indicates how robust the original transformer model is, which continues to be a competitive and well performing model even as different architectures emerge in attempts to increase performance. Second, our model's ability to achieve similar performance to the baseline shows that this method could be potentially improved to eventually reach a better performance than the baseline. The qualitative differences in attention maps could have some benefits in specific applications that require a more granular understanding of input data for tasks like more complex reasoning, or hierarchical feature extraction. Our research helps contribute to the growing body of knowledge around the transformer architecture, and helps provide some insight into the potential of varying attention head counts. While our modified architecture did not yield any significant improvements in the general benchmarks, the results could help lead further investigation into whether there exists specific tasks where dynamic attention heads could offer benefits.

5.1. Limitations

There were several unexpected obstacles that prevented the optimal performance of the Deep Head model. The most glaring being the unavailability of resources for training. We trained the model on 1 GPU for several days, but multiple GPUs would have increased the performance of the model significantly. In addition, the decoding methods used for experimentation may have decreased the accuracy of generated text. For both models, we utilized a greedy decoding strategy where the token with the highest logit score was chosen as the next decoded token. We did smooth the output from the softmax activation probabilities by .1 to add some randomness to the text generation.

5.2. Future Work

There are many future applications for Deep Head. Many of the models currently being engineered in the NLP and CV sectors use transformers as a backbone to encode text and image input data into their models and build upon the backbone to fine-tune the models for specified tasks. If Deep Head was to be used in these models instead of Transformers, performance of models across the board would increase for whatever downstream task. Deep Head could also be a starting point to analyzing different possibilities of attention mechanisms within Transformers to attain better performance.

References

[1] Xiao Luwei et al. "Multi-head self-attention based gated graph convolutional networks for aspect-based sentiment classification". In: *Multimedia Tools and Applications* 81 (June 2022), pp. 1–20. DOI: 10.1007/s11042-020-10107-0.

[2] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. arXiv: 1912.05911 [cs.LG].

[3] Ralf C. Staudemeyer and Eric Rothstein Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019. arXiv: 1909.09586 [cs.NE].

[4] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[5] Elena Voita et al. *Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned*. 2019. arXiv: 1905.09418 [cs.CL].

[6] Qiang Wang et al. *Learning deep transformer models for machine translation*. 2019. arXiv: 1906.01787 [cs.CL].