INVOLUTION AND BEYOND: RETHINKING AND OPTIMIZING TRANSFORMER ARCHITECTURES

A PREPRINT

Akshay Nuthanapati
National Institute of Technology
Warangal
akshay.nuthanapati1@gmail.com

Raj Dandekar
Massachusetts Institute of Technology
rajd@mit.edu

ABSTRACT

Transformers have significantly impacted various domains, particularly due to their powerful self-attention mechanisms. However, their computational cost remains a challenge, especially for large context sizes. This paper delves into various architectural improvements to mitigate this issue, introduces the concept of viewing self-attention through the lens of graph neural networks, and proposes involution as a potential pathway. We further explore the potential future of Transformer architectures by examining efficient and innovative models, aiming to provide a comprehensive guide to recent advancements and potential future directions in this domain.

0.1 Introduction

Self-Attention mechanisms have been crucial in galvanizing the wave of Generative AI currently enrapturing the world. From Language Modelling to Computer Vision Applications, there have been few areas untouched by the powerful Transformer Architectures. Driving the success of Transformers is the underlying architecture which allows for direct interaction with every pair of elements allowing for capture of long term dependencies - a feature missing in the previously ubiquitous RNNs [32].

Alas, the incredibly potent Transformer Architecture comes with a price - the price of high computational costs arising from attention [6] operations with a quadratic time complexity. This in turn limits Transformers from scaling to inputs with large context sizes. It is thus imperative to solve the scalability issue in Transformers. A number of powerful techniques like sparsity [18], low rank decomposition [21] and kernel approximation [11] among others have sprung up over the years to address this limitation.

0.2 Architectural Improvements to Transformers

There have been numerous architectural improvements to the Transformer [44] over the past few years. Some of them optimize for computational efficiency while others have tried incorporating old ideas in NLP in conjunction with the self-attention mechanism. Large Language Models which have become so ubiquitous in toady's world, all incorporate some variants of these architectural changes. It would be prudent to analyse these various techniques to obtain a better understanding of the underlying mechanism of the Transformers and dispel the notion of Transformers as "Black Boxes" [9].

- Multi-Query Attention

One of the most popular variant of multi-head attention, Multi-Query Attention (MQA), was introduced by Noam Shazeer [39], to improve the efficiency of incremental decoding by reducing the memory bandwidth requirements. This is achieved by sharing keys and values across different attention heads, resulting in smaller tensors and lower memory costs.

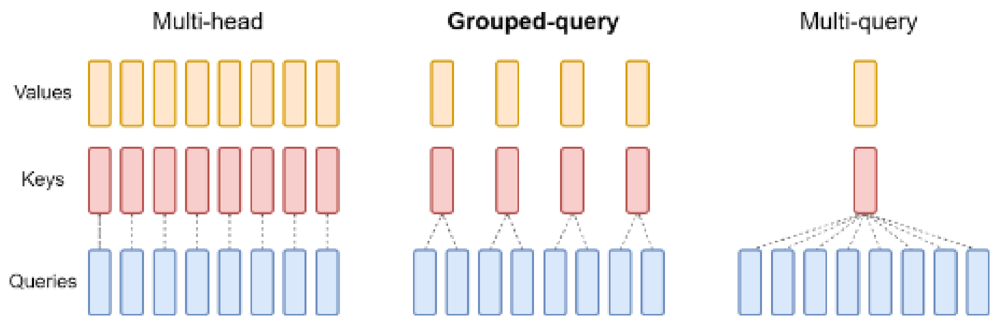


Figure 1: Visualization of Multi-Query Attention

The traditional approach of using separate keys and values for each attention head can be memory-intensive, particularly during incremental inference. By sharing keys and values, the size of these tensors is reduced, leading to reduced memory bandwidth requirements.

In certain scenarios, such as self-attention layers in auto regressive language models, parallel computation is not possible due to data dependencies. In such cases, incremental computation becomes necessary, and the proposed multi-query attention can be particularly beneficial.

- Attention Free Transformer

The Attention Free Transformer (AFT) [49] does not approximate or use the standard dot product attention. Rather it uses a model called the "Attention Free Transformer". AFT is composed of interactions between the usual Q, K, V values with the difference being that the key and value (context) are first combined with a set of learned position biases and the results are multiplied with the query element-wise.

AFT has linear memory complexity with respect to context size and the dimension of features. At the very heart of every Transformer architecture is the Multi-Head Attention (MHA) Operation. AFT is a replacement of MHA without the need for changing other architectural aspects of Transformers.

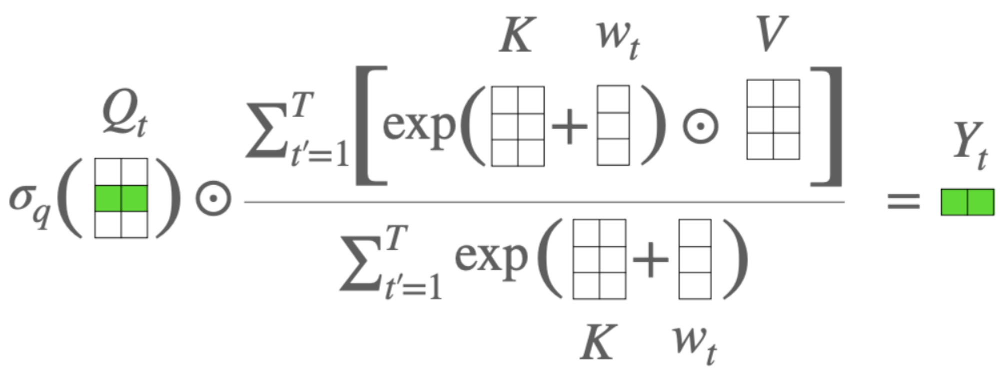


Figure 2: An Illustration of AFT

For each target position, AFT performs a weighted average of values, the result of which is combined with the query with element-wise multiplication. In particular, the weighting is simply composed of the keys and a set of learned pair-wise position biases. This provides the immediate advantage of not needing to compute and store the expensive attention matrix, while maintaining the global interactions between query and values as MHA does.

There are variants of the AFT:

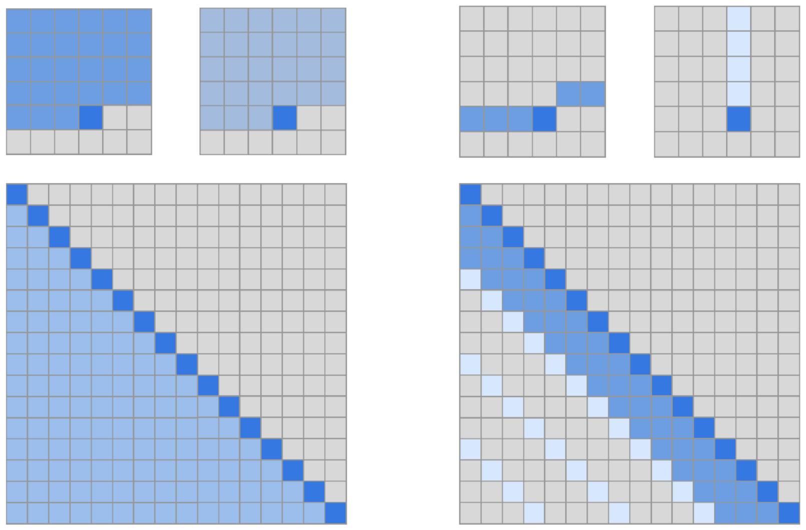
1.) AFT-Local: The learned position biases are constrained to a local region while maintaining global connectivity. It provides better parameter and computational efficiency, improving the model's performance in all tasks.

2.) AFT-Conv: Extends the design of AFT-local by imposing spatial weight sharing, effectively making it a variant of CNN with a global receptive field. It combines the benefits of locality and spatial weight sharing, further enhancing parameter and computational efficiency while maintaining global connectivity.

- Sparse Transformers

To address the quadratic time and memory requirements with sequence length “sparse factorization” of the attention matrix are introduced [10] Sparse factorization scales as $O(n\sqrt[p]{n})$ with the sequence length without sacrificing performance.

Sparse factorization is employed to reduce the computational complexity of the attention mechanism. The attention matrix, which represents the relationships between input tokens, is factorized into smaller, more manageable sub-matrices. This factorization is achieved by dividing the input sequence into smaller segments or blocks.



(a) Transformer (b) Sparse Transformer (strided)

Figure 3: Sparse Transformer Visualized

By using sparse factorization, the computational complexity of the attention mechanism scales as $O(n\sqrt[p]{n})$,where n is the sequence length and p is the sparsity factor. This means that the computational requirements grow linearly with the sequence length, resulting in significant memory and time savings compared to the traditional quadratic scaling.

Despite the reduction in computational complexity, Sparse Transformers maintain comparable [28] or even improved performance compared to traditional Transformers. This is achieved by carefully designing the factorization scheme and employing techniques to ensure that the model can still effectively capture long-range dependencies between tokens.

Additionally several changes are introduced in a sparse transformer.

- A restructured residual block [24] and weight initialization [23] to improve training of very deep networks
- A set of sparse attention kernels which efficiently compute subsets of the attention matrix
- Recomputation of attention weights during the backwards pass to reduce memory usage

- Flash Attention

FlashAttention [12] is a new attention algorithm that computes exact attention with far fewer memory access. This is achieved by avoiding reading and writing the attention matrix to and from High Bandwidth Memory (HBM). For this

1.) Softmax reduction needs to be computed without access to the whole input
2.) We need to avoid storing the large intermediate attention for the backward pass.

The attention computation is restructured to split the input into blocks and we make several passes over the input blocks, incrementally performing softmax reduction. This is known as Tiling.

The softmax normalization factor is stored from the forward pass to quickly recompute attention on-chip in the backward pass. This is faster than the standard approach of reading the intermediate attention matrix from HBM.

FlashAttention is implemented in CUDA to achieve fine-grained control over memory access and fuse all the memory access into one GPU Kernel.

Even with the increased FLOPs due to recomputation, our algorithm both runs faster (up to 7.6x on GPT-2 , and uses less memory—linear in sequence length—than standard attention, thanks to the massively reduced amount of HBM access.

1.) FlashAttention trains Transformer models faster in wall-clock time.

2.) FlashAttention scales Transformers to longer sequences, which improves their quality and enables new capabilities

3.) FlashAttention is up to 3 faster than the standard attention implementation across common sequence lengths from 128 to 2K. Up to sequence length of 512, Flash Attention is both faster and more memory-efficient than any existing attention method.

FlashAttention can be extended to Block-Sparse [20] Attention resulting in an approximate attention algorithm that is smaller than FlashAttention by a factor proportional to sparsity. Block-sparse Flash Attention uses a predefined block sparsity mask to compute only the nonzero blocks of the attention matrix, skipping the zero blocks. Block-sparse FlashAttention enables faster attention computations by reducing the amount of computation and memory access required, resulting in improved speed and memory efficiency.

- Activation Techniques: SoLU and SwiGLU/GeLU

Replacing the activation function [37] with a softmax linear unit called SoLU [16] which significantly increases the fraction of neurons in the MLP layers which seem to correspond to readily human-understandable concepts, phrases, or categories on quick investigation, as measured by randomized and blinded experiments.

SoLU increases the fraction of MLP neurons which appear to have clear interpretations, while preserving performance. Specifically, SoLU increases the fraction of MLP neurons for which a human can quickly find a clear hypothesis explaining its activations from 35 percent to 60 percent, as measured by blinded experiments.

SoLU's benefits may come at the cost of "hiding" other features. Despite the benefits mentioned above, SoLU is potentially a double-edged sword. We find theoretical and empirical evidence that it may "hide" some non-neuron-aligned features by decreasing their magnitude and then later recovering it with LayerNorm.

$$SOLU(x) = x * softmax(x)$$

SwiGLU (Switched Gated Linear Unit) [40] and GeLU (Gaussian Error Linear Unit) [25] are activation functions commonly used in deep learning models to introduce non-linearity and enhance model performance.

SwiGLU is a variant of the Gated Linear Unit (GLU) activation function [13]. It consists of two components: a linear transformation and a gating mechanism. The linear transformation computes the weighted sum of the input features. The gating mechanism [7] controls the flow of information by selectively activating or deactivating parts of the linear transformation output. Mathematically, SwiGLU can be defined as follows:

$$SwiGLU(x) = x * sigmoid(Wx + b)$$

Here, x represents the input, W and b are the weight matrix and bias term respectively, and sigmoid is the sigmoid activation function. The sigmoid function squashes the linear transformation output to a value between 0 and 1, allowing the gating mechanism to regulate the flow of information.

GeLU is another activation function widely used in neural networks. It approximates the Rectified Linear Unit (ReLU) function with a smooth transition. GeLU applies the Gaussian cumulative distribution function (CDF) to the input. Mathematically, GeLU can be defined as follows:

$$GeLU(x) = 0.5x(1 + erf(x/sqrt(2)))$$

Here, ‘erf‘ represents the error function. The application of the CDF introduces non-linearity and allows the model to capture more nuanced and continuous relationships between the input and the output.

Both SwiGLU and GeLU have demonstrated promising results in various natural language processing (NLP) tasks and image classification tasks [35]. They have contributed to the advancement of deep learning models by improving their performance and accuracy in capturing complex patterns and relationships.

- Alternatives to Layer Normalization

DeepNorm [45] is a novel normalization function that is used to modify residual connections in Transformers. It is an alternative to LayerNorm [4], which is commonly used in Transformer architectures.

DeepNorm introduces additional non-linearity and depth to the normalization process by incorporating multiple linear transformations in its computation. It operates on the residual connections in Transformers, enhancing the flow of information through the network.

In detail, DeepNorm computes the normalized activations by performing the following steps:

1. First, it applies a linear transformation to the input activations, followed by an element-wise activation function such as ReLU [1].
2. Next, it applies another linear transformation to the resulting activations.
3. Then, it adds the original input activations to the output of the second linear transformation, creating the normalized activations.

By incorporating multiple linear transformations and non-linear activation functions, DeepNorm allows for a more expressive normalization function compared to LayerNorm. This can potentially improve the performance and learning capabilities of Transformer models.

Further research and experimentation are required to fully understand the benefits and trade-offs of DeepNorm compared to traditional normalization techniques like LayerNorm. However, its introduction represents an exciting development in the field of Transformers and offers new possibilities for improving the performance of these models.

Root Mean Square Layer Normalization (RMSLN) [50] is another alternative to LayerNorm in Transformer architectures. It aims to address some limitations of traditional normalization techniques by introducing a different computation approach.

RMSLN computes the normalized activations by following these steps:

1. First, it calculates the root mean square (RMS) value of the input activations across the hidden dimensions.
2. Next, it applies a linear transformation to the input activations, followed by an element-wise activation function.
3. Then, it divides the transformed activations by the RMS value to normalize them.
4. Finally, it adds a learnable bias term to the normalized activations.

By using the RMS value as a normalization factor, RMSLN takes into account the overall magnitude of the activations in addition to their distribution. This can help in handling situations where the activations have different scales or exhibit varying levels of variability.

RMSLN offers a different perspective on normalization in Transformers and provides a potential avenue for further exploration and experimentation. Its advantages and performance compared to LayerNorm and other normalization techniques are still an active area of research.

- SentencePiece Tokenizers

SentencePiece Tokenizers [30] provide several advantages in transformers:

1. *Language Flexibility*: SentencePiece Tokenizers support a wide range of tokenization algorithms, including BPE [41], unigram, and word-based models. This flexibility allows them to handle various languages more efficiently. Transformers often deal with multilingual tasks, and SentencePiece Tokenizers enable better language coverage.

2. *Custom Tokenization Rules*: SentencePiece Tokenizers allow users to define custom tokenization rules. This is particularly useful for domain-specific or specialized tasks where specific patterns or rules need to be followed during tokenization. Custom rules can be defined to handle specific cases and improve the overall performance of the transformer.

3. *Out-of-Vocabulary (OOV) Handling*: OOV [5] words are a common challenge in NLP tasks. SentencePiece Tokenizers excel in handling OOV words by using subword tokenization. By dividing words into smaller subword units, SentencePiece Tokenizers can effectively handle OOV words and reduce the number of unknown tokens in the transformer input.

4. *Efficient Resource Utilization*: SentencePiece Tokenizers allow for efficient resource utilization by dynamically adjusting the vocabulary size. This is particularly beneficial when working with limited computational resources or memory constraints. By adjusting the vocabulary, SentencePiece Tokenizers can optimize the tokenization process and improve overall efficiency.

These advantages make SentencePiece Tokenizers a valuable tool in transformers, enabling better language coverage, improved tokenization rules, and efficient handling of OOV words.

- Rotary Position Embeddings

RoPE(Rotary Position Embeddings) [42] is a technique which allows us to encode positional information [38] in self-attention layers of Transformers without increasing the model size. With RoPE, instead of using separate positional embeddings, the input embeddings [2] are rotated by a fixed matrix before self-attention. The rotation is different for each position, thus encoding position information.
The rotated embeddings align with other tokens based on their positional similarity allowing the model to learn positional relationships. RoPE has the distinct advantage of being able to handle sequences of variable lengths without the need for a fixed-sized positional embedding matrix.

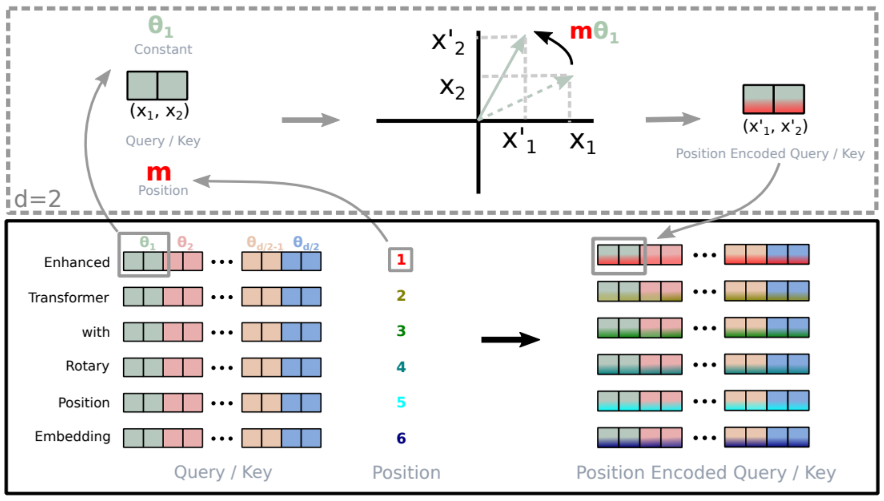


Figure 4: Implementation of Rotary Position Embedding

0.3 Self-Attention as a Graph Neural Networks

Self-attention can be thought of as a graph neural network [36], where each token in the input sequence is a node, and edges denote the relationship between each tokens. This makes sense, as different context gives different meaning to how different tokens are connected.

In dot-product attention, we multiply the query (Q) with the key (K) using dot product. This effectively tells us the scores of similarity between Q and K.

If we think of tokens as signals, and self-attention as a measure of correlation between those signals, then attention can be used to capture both long-range dependencies and delay. Correlation is great when there is delay in signals, and cross-correlation is a more general way to find similarity between signals (tokens) effectively. Dot product is just cross-correlation with zero lag. If we introduce lags explicitly, then we can capture more in-context relationships. This is because we now have more ways in which tokens can be connected, which gives us more context. Cross-correlation is a measure of two series as a function of the displacement of one relative to the other. It is similar to convolution of two functions, and can be used to measure the degree between data.

Convolutions have not been used in Transformer-type architectures, even though they capture positional information, because Transformers provide flexibility. We can obtain learnable embeddings [26] in Transformers, but not in CNNs [3], because the kernel becomes static once learned. Additionally, CNNs can be computationally expensive. Attention is different. The query (Q), key (K), and value (V) matrices allow context to be taken into account.

Involution [17] is a method that is somewhere between CNNs and Attention [31]. It is more effective and efficient than CNNs, and it is much simpler than self-attention.

0.4 Efficient Transformer Variants: Future Directions

- ***Transformers for Large Contexts***

Longformer [8] is a transformer model that is specifically designed to process long sequences of text. It is a modified version of the standard transformer architecture, with a key difference being the way that it handles self-attention.

Self-attention is a key component of the transformer architecture, and it allows the model to learn long-range dependencies in the input text. However, the standard self-attention mechanism scales quadratically with the length of the input sequence, which makes it impractical to use for processing very long sequences.

Longformer addresses this issue by using a different attention mechanism that scales linearly with the sequence length. This is done by combining two different attention mechanisms: a local windowed attention and a global attention. The local windowed attention mechanism allows the model to attend to tokens that are close together in the sequence. This is important for capturing local dependencies in the text. The global attention mechanism allows the model to attend to tokens that are anywhere in the sequence. This is important for capturing long-range dependencies in the text.

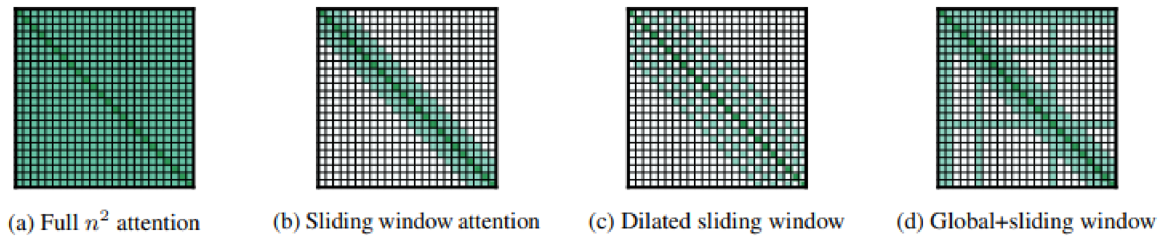

(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

Figure 5: Comparison of the full self-attention pattern and the configuration of attention patterns in Longformer.

The global attention mechanism in Longformer is also task-motivated, which means that it is configured differently depending on the task that the model is being used for. For example, if the model is being used for question answering, the global attention mechanism will be configured to give more attention to the question and the answer passage.

$$Attention(X) = alpha * Localattention(X) + beta * Globalattention(X)$$

Big Bird [48] is a sparse-attention transformer, which means that it uses a sparse attention mechanism to reduce the quadratic computational complexity of the Transformer model to linear. This allows BigBird to handle much longer sequences than previous Transformer models, such as BERT [22].

BigBird uses a sparse attention mechanism that is composed of three components:

- Random attention: Each token attends to a set of random tokens.
- Local attention: Each token attends to a set of local neighboring tokens.
- Global attention: A set of global tokens attend to all parts of the sequence.

The random attention component allows BigBird to learn long-range dependencies in the input sequence without having to compute attention weights for all pairs of tokens. The local attention component allows BigBird to model local dependencies in the input sequence. The global attention component allows BigBird to model global dependencies in the input sequence.

BigBird can handle sequences that are up to 8 times longer than previous Transformer models, such as BERT [14]. This makes BigBird suitable for a wider range of NLP tasks, such as long document classification and summarization. BigBird has also achieved state-of-the-art results on a variety of NLP tasks, including question answering, summarization, and long document classification. BigBird's sparse attention mechanism reduces the computational cost of training and deploying Transformer models. This makes BigBird more accessible to a wider range of users.

- Focus on Efficiency: Linformer, Performer, Reformer.

The **Linformer** [46] is a modification of the standard Transformer model that reduces the computational complexity of self-attention from $O(n^2)$ to $O(n)$, where n is the sequence length. This makes the Linformer much more efficient to train and deploy, especially for long sequences.

The Linformer achieves this by using a low-rank approximation of the self-attention matrix. The key idea is that the self-attention matrix is often low-rank, meaning that it can be approximated by a much smaller matrix with minimal loss of information. To perform low-rank approximation, the Linformer first projects the input sequence into a lower-dimensional space using two linear projection matrices. The resulting low-dimensional representations are then used to compute the self-attention matrix. To compensate for the loss of information due to low-rank approximation, the Linformer uses a technique called multi-head attention. Multi-head attention computes multiple self-attention matrices using different linear projection matrices. The outputs of the different heads are then concatenated and projected to the original dimension using a final linear projection matrix.

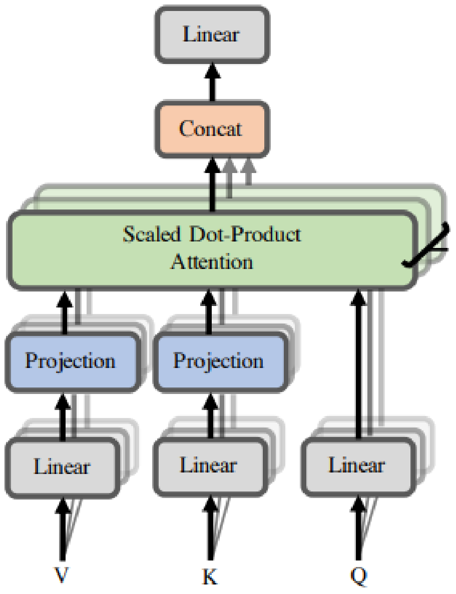


Figure 6: Multihead Linear Self Attention

The Linformer has several benefits over standard Transformer models, including:

- Linear computational complexity: The Linformer has linear computational complexity in the sequence length, while standard Transformer models have quadratic computational complexity. This makes the Linformer much more efficient to train and deploy, especially for long sequences.
- Lower memory usage: The Linformer also has lower memory usage than standard Transformer models. This is because the Linformer does not need to store the entire self-attention matrix. Instead, it can store the low-dimensional representations of the input sequence and the projection matrices used to compute the self-attention matrix.

- On par performance: The Linformer has been shown to perform on par with standard Transformer models on a variety of natural language processing tasks.

The **Reformer** [29] addresses two of the main sources of inefficiency in the Transformer:

- Quadratic attention complexity: The standard self-attention mechanism in the Transformer has quadratic complexity, meaning that the amount of computation required to compute attention between all pairs of tokens in a sequence grows quadratically with the length of the sequence. This makes the Transformer impractical for processing very long sequences. - Memory usage: The Transformer also requires a significant amount of memory to store the attention weights and other intermediate representations. This can make it difficult to train Transformer models on large datasets or on long sequences.

The Reformer addresses these problems using two key techniques:

- Locality-sensitive hashing (LSH) attention: The Reformer replaces the standard self-attention mechanism with an LSH-based attention mechanism [34]. LSH is a technique for finding similar items in a large dataset efficiently. The Reformer uses LSH to find a small set of tokens that are likely to be relevant to each token in the sequence. This reduces the quadratic complexity of attention to $O(nlogn)$, where n is the length of the sequence.
- Reversible residual layers: The Reformer also uses reversible residual layers [19] instead of the standard residual layers used in the Transformer. Reversible residual layers allow the Reformer to store activations only once in the training process, instead of multiple times, which saves memory.

Reformer has been shown to achieve comparable performance to the Transformer on a variety of tasks, such as machine translation, text summarization, and question answering. However, the Reformer is much more memory-efficient and faster than the Transformer on long sequences. The Reformer is also significantly faster than the Transformer on long sequences. For example, the Reformer can translate a 1 million token sequence in 17 seconds, while the Transformer takes over 1 hour to translate the same sequence.

- Newer architectures and their promises: Retentive Network, RWKV.

Retentive Network [43], often shortened to RetNet, is a new neural network architecture, which is designed to address the limitations of the Transformer architecture, particularly in terms of training parallelism and inference efficiency.

Transformers have become the dominant architecture for large language models due to their ability to learn long-range dependencies in sequential data. However, Transformers have two key limitations:

1. Training parallelism: Transformers are difficult to train in parallel due to their sequential nature. This can limit the scalability of Transformer-based models to very large datasets.
2. Inference efficiency: Transformers are also relatively inefficient to run at inference time, as they require attention to be computed over the entire sequence length. This can limit the applicability of Transformer-based models to real-time applications.

RetNet addresses these limitations by introducing a new retention mechanism that is both parallel and recurrent. This allows RetNet to be trained in parallel and to run at inference time with linear complexity in the sequence length. The retention mechanism in RetNet works by maintaining a hidden state that summarizes the context of the sequence up to the current position. This hidden state is updated recurrently, but the updates can be computed in parallel for different positions in the sequence.

RetNet also introduces a new chunkwise recurrent computation paradigm that allows for efficient modeling of long sequences. In chunkwise recurrent computation, the sequence is divided into chunks, and each chunk is encoded parallelly while the chunks are summarized recurrently. Experimental results show that RetNet achieves comparable performance to Transformer-based models on language modeling tasks, while providing more efficient training and inference. RetNet also outperforms other models in terms of memory consumption, throughput, and latency during inference.

Receptence Weighted Key Value (RWKV) [33] is a novel model architecture that combines the efficient parallelizable training of Transformers with the efficient inference of RNNs. RWKV leverages a linear attention mechanism and allowing the model to be formulated as either a Transformer or an RNN. This enables RWKV to parallelize computations during training and maintain constant computational and memory complexity during inference, leading to the first non-transformer architecture to be scaled to tens of billions of parameters. RWKV's layer structure is similar to that of a Transformer, with alternating self-attention and feed-forward layers. However, RWKV's self-attention layers use the linear attention mechanism described above. Additionally, RWKV's feed-forward layers are replaced with residual connections.

RWKV was evaluated on a variety of NLP tasks, including machine translation, text summarization, and question answering. RWKV achieved on-par or better performance than similarly sized Transformers on all tasks. One limitation of RWKV is that it is more difficult to train than Transformers. This is because RWKV's linear attention mechanism is less expressive than the Transformer's scaled dot-product attention. Additionally, RWKV is more sensitive to the choice of hyper parameters.

0.5 Discussion and Conclusions

In this work, a comprehensive overview of the various architectural improvements to the Transformer since its inception in 2017 have been analysed. However, the quadratic nature of self attention mechanisms pose a significant challenge for scaling Transformers to long contexts. Numerous techniques and insights have been developed over the past few years to address this fundamental limitation. From architectural improvements like multi-query attention and sparse factorization to reviving old ideas from RNNs, all the techniques presented have been adopted to varying degrees for specific tasks.

Mutli-query attention improves efficiency of incremental decoding and reduces the memory bandwidth requirements. This is achieved by sharing keys and values across different attention heads, resulting in smaller tensors and lower memory costs. FlashAttention also reduces memory usage of the Transformer by avoiding reading and writing the attention matrix from High Bandwidth Memory (HBM). Popular techniques like AFT and sparse factorization introuduce variants of dot product attention to reduce time complexity. Activation functions like SoLU, SwiGLU, and GeLU improve model performance by introducing non-linearities. SoLU in particular improves interpretabilty to a great extent. Alternatives, to layer norm like DeepNorm and RMSLN (Root Mean Square Layer Normalization) incorporate non-linearity and depth into normalization to enhance information flow. SentencePiece tokenizers support multiple languages and custom rules. They efficiently handle Out-Of-Vocabulary (OOV) words using subword units. Rotary positional embeddings encode position without increasing model size, by rotating input embeddings.

The Longformer is a Transformer model designed specifically to process long sequences of text. It combines two different attention mechanisms, a local windowed attention and a global attention. Big Bird is a sparse-attention based transformer which reduces the quadratic computational complexity of the Transformer models. Some of the newer models introduced like Linformer and Reformer use low-rank approximations and locality sensitive hashing respectively to tackle the quadratic source of inefficiency.Retentive Network addresses two limitations of the Transformer architecture, particularly in terms of training parallelism and inference efficiency.

By viewing self-attention as a graph neural networks, new possibilities open up. Thinking of tokens as signals and self-attention as correlation enables techniques like convolution and involution to capture dependencies more effectively. Involution strikes a balance between CNNs and attention, providing efficiency without sacrificing too much in terms of expressiveness.

The success and popularity of almost every single Large Language Model is in part due to the robust Transformer architecture. It can very clearly be concluded that the field of Artificial Intelligence would have stagnated without the revolutionary paper "Attention Is All You Need". The practical applications range from conventional NLP [47] tasks to esoteric implementations in fields such as Cheminformatics [27] and Biology [15], demonstrating once again the potentness of the mighty Transformer. It is thus of utmost importance that the scientific community strives towards complete interoperability of Transformers.

Bibliography

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers. *arXiv preprint arXiv:2004.08483*, 2020.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzębski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*, 2017.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Jorge A Balazs and Yutaka Matsuo. Gating mechanisms for combining character and word-level word representations: an empirical study. *arXiv preprint arXiv:1904.05584*, 2019.

[8] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[9] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164, 1997.

[10] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[11] Sankalan Pal Chowdhury, Adamos Solomou, Avinava Dubey, and Mrinmaya Sachan. On learning the transformer kernel. *arXiv preprint arXiv:2110.08323*, 2021.

[12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[13] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Yannick Djoumbou-Feunang, Jarlei Fiamoncini, Alberto Gil-de-la Fuente, Russell Greiner, Claudine Manach, and David S Wishart. Biotransformer: a comprehensive computational tool for small molecule metabolism prediction and metabolite identification. *Journal of cheminformatics*, 11(1):1–25, 2019.

[16] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/solu/index.html.

[17] Arash Ghaani Farashahi. Convolution and involution on function spaces of homogeneous spaces. *arXiv preprint arXiv:1201.0297*, 2011.

[18] Kazi Ahmed Asif Fuad and Lizhong Chen. A survey on sparsity exploration in transformer-based accelerators. *Electronics*, 12(10):2299, 2023.

[19] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

[20] Scott Gray, Alec Radford, and Diederik P Kingma. Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 3(2):2, 2017.

[21] Habib Hajimolahoseini, Walid Ahmed, Mehdi Rezagholizadeh, Vahid Partovinia, and Yang Liu. Strategies for applying low rank decomposition to transformer-based models. In *36th Conference on Neural Information Processing Systems (NeurIPS2022)*, 2022.

[22] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Visualizing and understanding the effectiveness of bert. *arXiv preprint arXiv:1908.05620*, 2019.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[26] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve transformer models with better relative position embeddings. *arXiv preprint arXiv:2009.13658*, 2020.

[27] Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

[28] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

[29] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[30] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[31] Firas Laakom, Kateryna Chumachenko, Jenni Raitoharju, Alexandros Iosifidis, and Moncef Gabbouj. Learning to ignore: rethinking attention in cnns. *arXiv preprint arXiv:2111.05684*, 2021.

[32] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[33] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[34] Frithjof Petrick, Jan Rosendahl, Christian Herold, and Hermann Ney. Locality-sensitive hashing for long context neural machine translation. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 32–42, 2022.

[35] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[36] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[37] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[38] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[39] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[40] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[41] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. *Technical Report DOI-TR-161, Department of Informatics, Kyushu University*, 1999.

[42] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[43] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[45] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

[46] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[47] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[48] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[49] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021.

[50] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.