Aalto University
School of Science
Bachelor's Programme in Science and Technology

Variants of Transformer Architectures

Bachelor's Thesis

18. toukokuuta 2022

Joona Rantanen

Aalto-yliopisto
Perustieteiden korkeakoulu
Teknistieteellinen kandidaattiohjelma

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä:	Joona Rantanen
Työn nimi:	Variants of Transformer Architectures
Päiväys:	18. toukokuuta 2022
Sivumäärä:	23
Pääaine:	Tietotekniikka
Koodi:	SCI3027
Vastuuopettaja:	Professori Eero Hyvönen
Työn ohjaaja(t):	DI Sam Spilsbury (Tietotekniikan laitos)

Transformeri (engl. Transformer) on vuonna 2017 julkaistu syväoppimismalli, joka on saavuttanut uusia ennätystuloksia useissa luonnollisen kielen prosessoinnin tehtävissä. Alkuperäisessä arkkitehtuurissa on kuitenkin vakava pullonkaula, joka estää sitä skaalautumasta pidemmille datajonoille, kuten esimerkiksi kokonaisille tekstidokumenteille. Tämän johdosta alkuperäistä transformeria on kehitetty ja siitä on julkaistu useita muunnelmia. Tämä työ käsittelee kirjallisuudessa julkaistuja transformerin muunnelmia ja sitä, miten nämä muunnelmat yrittävät ratkaista skaalautuvuusongelman. Tavoitteena on antaa lukijalle käsitys transformerin ongelmasta ja siitä, miten sitä on yritetty ratkaista.

Tämä työ on toteutettu kirjallisuuskatsauksena ja lähteinä toimivat pääasiassa tieteelliset julkaisut. Työ käsittelee useaa tapaa ratkaista transformerin skaalautuvuusongelma ja jakaa eri lähestymistavat eri kategorioihin. Näitä kategorioita tutkitaan tarkemmin, ja jokaiseen on valittu joukko tieteellisiä julkaisuja tarkempaan tarkasteluun.

Tutkimuksessa havaittiin, että transformerin skaalautuvuusongelma on teoriassa onnistuttu ratkaisemaan. Jokaiseen eri muunnelmaan kuluu kuitenkin omat haittansa, ja niitä yritetään jatkuvasti parantaa. Skaalautuvuusongelman ratkaiseminen saattaa kuitenkin heikentää muunnelman suoritusta eri tehtävissä, mikä omalta osaltaan lisää tutkimuksen tarvetta tranformerien parissa.

Uusia transformerin muunnelmia julkaistaan jatkuvasti, ja tutkimus alalla etenee nopeasti. Skaalautuvuusongelmaan löydetään uusia ratkaisuja, ja uudet muunnelmat saavuttavat uusia ennätystuloksia useissa tehtävissä.

Avainsanat:	Transformer, Deep learning, Self-attention
Kieli:	Suomi

Contents

1 Introduction

Transformer is a popular deep learning model introduced in 2017 (Vaswani et al., 2017). It is increasingly the model chosen for various natural language processing tasks such as machine translation and question answering replacing recurrent neural networks. It has also gained popularity in computer vision tasks replacing standard convolutional neural networks (Dosovitskiy et al., 2021). Since its introduction, the Transformer has gained much popularity and many variants have been proposed to make improvements to the original model. These variants mostly try to improve the computational and memory efficiency of the original Transformer. In addition other variants have been proposed for problems such as image generation and reinforcement learning.

The goal of this thesis is to make the reader aware of the scalability issues of the original Transformer and how this can be overcome with different mechanisms. The main focus will be on how the different variants tries to solve this problem in detail. This thesis does not cover all the different mechanism and variants proposed in the literature; instead the purpose is to review the most relevant and interesting ones.

After the introduction this thesis contains three sections. The first section covers the original Transformer and gives the reader sufficient knowledge about the recent research directions. The second section covers different mechanisms to overcome the main scalability issue of the original Transformer. This section is further divided into three parts, which all cover different mechanisms to solve this problem. The last section is the summary which summarizes the main points of this thesis.

The main section of this thesis is the Analysis of Transformer variants. It is divided into three distinct parts. Architectures which try to sparsify the attention matrix will be covered first. This part covers architectures like Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020). After that an overview is given on the non-sparsity based architecture improvements. This part covers architectures such as Linformer (Wang et al., 2020) and Performer (Choromanski et al., 2021). The third part covers non-attention improvements which covers architectures like Universal Transformer (Dehghani et al., 2018) and Transformer-XL (Dai et al., 2019).

2 Background

2.1 Original Transformer Architecture

This section gives an overview of the basic Transformer architecture. The idea is to give the reader sufficient knowledge on Transformers so that the reader is able to understand

the discussion on different architecture variants.

Transformer is a deep learning model that uses a multi-head self-attention mechanism. It is a sequence-to-sequence model: it processes input sequence and returns output sequence. Unlike recurrent neural networks, Transformer does not process input sequentially making it easily parallelizable.

Transformer consists of two main parts: encoder and decoder. Encoder contains self-attention and decoder contains optionally masked self-attention and cross-attention with the encoder outputs.

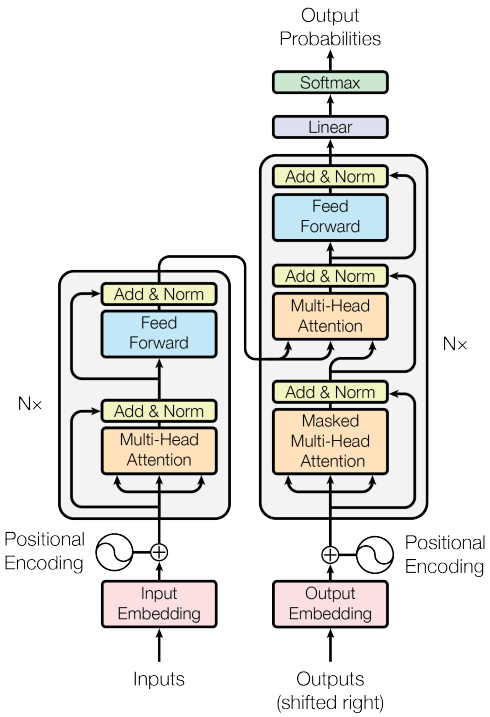

Figure 1: Original Transformer architecture (Vaswani et al., 2017)

2.1.1 Input Embedding

The Transformer usually uses learned embeddings to convert the input tokens into vectors. The purpose is that similar words or words which usually appear in the same context are close to each other in the vector space.

The input embedding layer takes the word indices in the vocabulary and converts them into word embeddings.

2.1.2 Positional Embedding

After the input embedding layer is a positional embedding module. Self-attention operation is equivariant to a permutation, but usually the order of the tokens matters. That is why positional embedding is needed. The original Transformer uses sin and cos functions to calculate the position embeddings which are then added to the input embedding vectors. Sin and cos functions are used as a function of the dimension of the positional embedding, meaning that sin function is used for when i is even and cos is used when i is odd.

$$PE_{(pos,2i)} = \sin(\text{pos}/10000^{2i/d_{\text{model}}}) \tag{1}$$

$$PE_{(pos,2i+1)} = \cos(\text{pos}/10000^{2i/d_{\text{model}}}) \tag{2}$$

2.1.3 Scaled Dot-Product Attention

The scaled dot-product attention block takes queries, keys and values as input. The query and key matrices are multiplied and scaled by dividing by the dimension of the key vector. After that the softmax function is applied to the matrix. Lastly, the resulting matrix is multiplied with the value matrix and that is the output of the scaled dot-product attention.

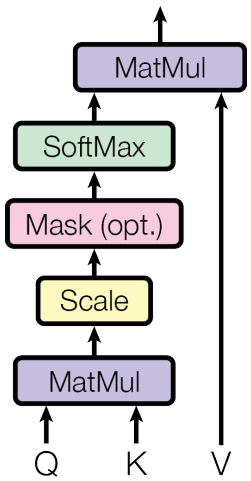

Figure 2: Scaled Dot-Product Attention (Vaswani et al., 2017)

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

2.1.4 Multi-Head Attention

The next component of the Transformer architecture is the multi-head attention. The multi-head attention utilizes self-attention mechanism. The multi-head attention gets queries, keys and values as input and projects those with the linear layers to a lower dimensional representations. This is done to utilize the attention function multiple times while keeping the total computational cost the same. Because the use of parallel attention heads the distinct heads need to be concatenated and finally they are fed to the linear layer and the output of this layer becomes the final output of the multi-head attention.

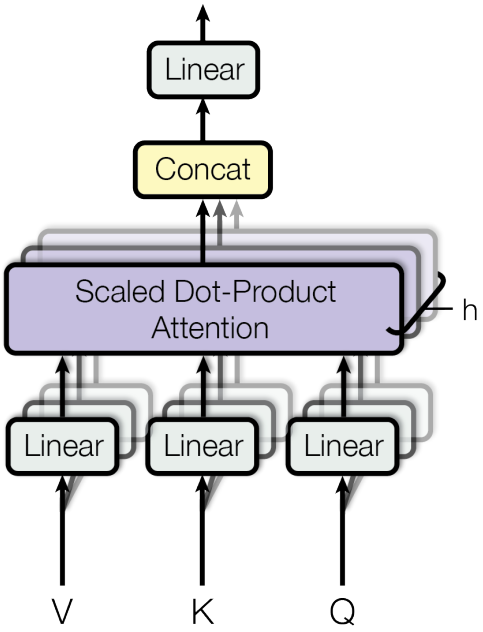

Figure 3: Multi-Head Attention (Vaswani et al., 2017)

2.1.5 Encoder and Decoder

The encoder consists of a stack of identical layers. Each layer has a multi-head self-attention mechanism layer and a position-wise fully connected feed forward network. Both the multi-head attention and feed forward layers utilize residual connection after which layer normalization is applied.

The decoder is very similar to the encoder. It contains optionally masked self-attention and cross-attention with the encoder outputs.

2.1.6 Position-wise Feed-forward Layers

Each layer in the encoder and decoder contains a fully connected feed-forward network. The linear layer consists of two linear transformations with a ReLU activation function in between and is applied to each position separately. The linear transformation is the same for each position but it uses different parameters from layer to layer.

2.1.7 Masked Multi-Head Attention

The use of masked multi-head attention comes in when training the Transformer. In the training phase, the target sequence tokens are masked so that the model can not see the true answer until it has made a prediction and needs the true value for the loss function. In practice, this is achieved by giving negative infinity values for all the future tokens in the mask filter matrix which is then added to the attention matrix.

2.2 Problem with self-attention

The main problem with the original Transformer is the self-attention matrix. Memory and computational complexity required to compute the self-attention matrix is quadratic with respect to the sequence length N. This complexity comes from the N^2 dot products needed to calculate the attention matrix. This is a highly restrictive thing in applications which operate on long sequences. Most of the Transformer variants discussed in this thesis try to tackle this problem with different kinds of mechanisms to reduce the memory and computational complexity.

2.3 Summary of Recent Research Directions in Transformer Architectures

Tay et al. (2020) propose a taxonomy of efficient transformers. It categorizes different transformer variants into multiple categories. These categories are characterized by the way the original Transformer architecture is improved and some of the variants can belong to multiple categories. In the following a brief introduction to these categories is given.

2.3.1 Fixed/Factorized/Random Patterns

Fixed patterns are the earliest modification to self-attention. Its idea is to sparsify the attention matrix by limiting the field of view to predefined patterns. By doing this only a subset of the calculations are needed, which reduces the complexity. This class has three different patterns. The first and simplest one is the blockwise patterns. It simply chunks the input sequence into fixed blocks and by doing this it reduces the complexity from

N^2 to B^2. The second one is strided patterns which works by only attending at fixed intervals. The third one is compressed patterns which uses pooling operator to reduce the sequence length.

2.3.2 Learnable Patterns

Learnable patterns is an extension to fixed patterns. It tries to learn the access pattern from the data. The idea is to determine a notion of token relevance and after that assign tokens to clusters or buckets. Reformer (Kitaev et al., 2020) belongs to this class and it uses hash based similarity measures to cluster tokens. In these type of models, the similarity function is trained end-to-end jointly with the network

2.3.3 Memory

Memory category covers architectures which use a side memory module that can access multiple tokens at once. Global memory is a commonly used form since it can access the entire sequence. One variant belonging to this category also discussed later in this thesis is Longformer (Beltagy et al., 2020).

2.3.4 Low Rank/Kernels

Low rank/kernels category covers architectures which leverage low rank methods and the kernel trick to the attention matrix. Low rank methods improve the efficiency by either approximating the attention matrix or approximating multiplication with it. Linformer (Wang et al., 2020) is one example that falls into this category. Its main idea is to assume a lower rank structure in the $N \times N$ matrix. The kernel part of this category concerns methods which improve the efficiency of the Transformer by using the kernel on the attention matrix and therefore avoiding explicitly computing the $N \times N$ matrix.

2.3.5 Recurrence

Recurrence category covers architectures which use recurrence to connect the blocks in the Transformer. Transformer-XL (Dai et al., 2019) falls into this category. It proposed a recurrence mechanism to connect multiple segments and blocks.

3 Analysis of Transformer variants

This section is divided into three different parts. The first part discusses sparse attention. The second part discusses the non-sparsity based improvements. Finally the third one

discusses the non-attention improvements.

3.1 Sparse Attention

In the first section covering the different variants sparse attention mechanisms are discussed. This section covers many variants which reduce the memory and computational complexity of the attention matrix by making the self-attention mechanism sparse by only attending to some of the input tokens instead of the full attention. The sparse attention can be achieved by many ways such as only attending to local neighbors in predefined distance or making the attention partly random.

3.1.1 Sparse Transformer

The first architecture discussed is the Sparse Transformer (Child et al., 2019) which belongs to the class of fixed patterns. The Sparse Transformer introduces an attempt to reduce the quadratic complexity of the self-attention mechanism. It proposes two types of factorized attention. The main idea is to reduce the dense attention matrix to a sparse version by only computing attention on a subset of query and key pairs. Sparse Transformer uses fixed attention patterns which are defined by strides and local neighborhoods. These are illustrated in the figure 4.

In Sparse Transformer half of the attention heads are dedicated to the local attention and the other half is dedicated to the fixed strided patterns.

The local attention heads are obtained via:

$$\hat{A_{ij}} = \begin{cases} Q_i(K)_j^T, & \text{if } [j/N] = [i/N] \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

And the strided attention heads via:

$$\hat{A_{ij}} = \begin{cases} Q_i(K)_j^T, & \text{if } (i-j) \bmod N = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

In addition to the sparse attention matrix, Sparse Transformer paper introduces several other changes. These changes include restructured residual block and weight inflation, a set of sparse attention kernels and recomputation of attention weights during backward pass. These are not discussed in more detail but are still mentioned to inform that Sparse Transformer includes also other changes.

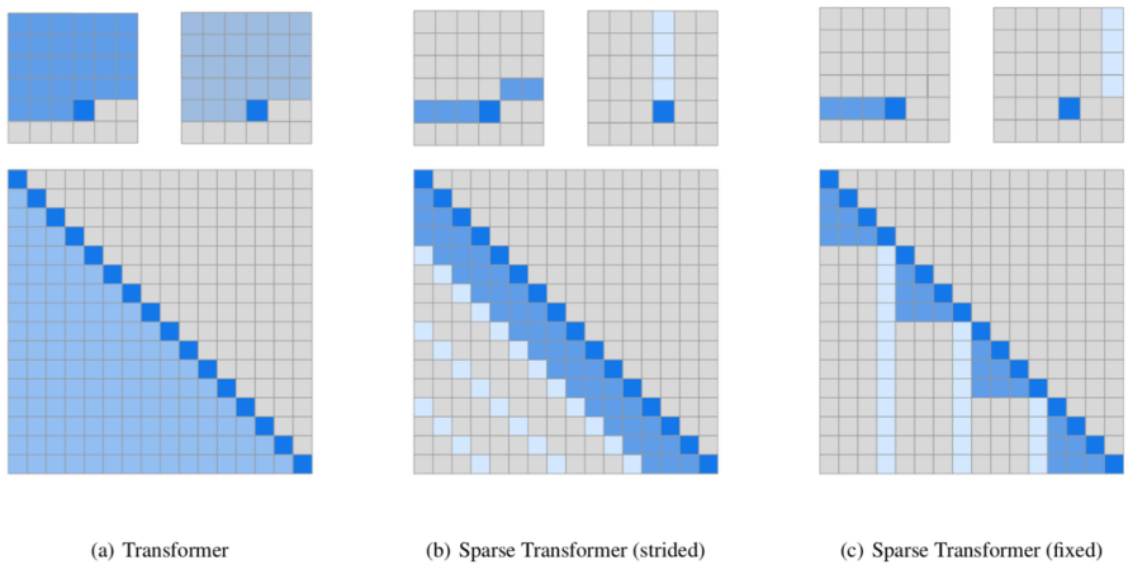

Figure 4: Sparse Transformer attention patterns (Child et al., 2019)

Sparse Transformer manages to reduce the complexity to $\mathcal{O}(N\sqrt{N})$ where N is the sequence length.

3.1.2 Longformer

Longformer (Beltagy et al., 2020) is another architecture which uses strided attention patterns to compute the attention matrix. It can be seen as a variant to Sparse Transformer. It differs from the Sparse Transformer by using a dilated sliding window, which enables better long-range coverage without sacrificing sparsity. Dilated sliding window means that the attention pattern has gaps in the receptive field and therefore it makes the attention have wider coverage.

Longformer can also contain tokens which have global attention meaning that they can attend to every other token. This can be applied for example in classification tasks where the special CLS token can attend to every other token.

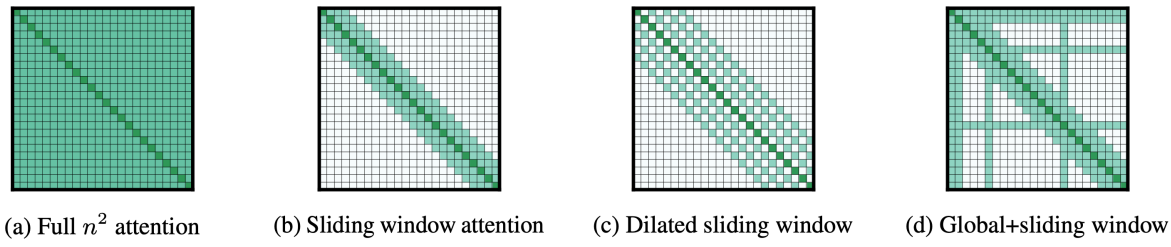

Figure 5: Longformer attention pattern (Beltagy et al., 2020)

Longformer manages to reduce the complexity to $\mathcal{O}(Nk)$ where k is the size of the window. When k is constant the complexity reduces to $\mathcal{O}(N)$. The tradeoff is, however that Longformer loses some information during the attention calculation.

3.1.3 Big Bird

The BigBird (Zaheer et al., 2020) is an architecture which also uses sparse attention to reduce the attention matrix complexity. It combines global attention, sliding window attention and random attention. The idea of using global attention is similar to that used in Longformer but instead of using it to only special tokens like the CLS token it is extended to contain tokens within the sequence. This is called internal transformer construction meaning that some existing tokens are made global. The sliding window in BigBird works by attending to $w/2$ tokens to left and to $w/2$ tokens to right and the random attention works by attending to fixed random tokens.

BigBird achieves a complexity of $\mathcal{O}(N)$, but it comes with a similar tradeoff to that of Longformer.

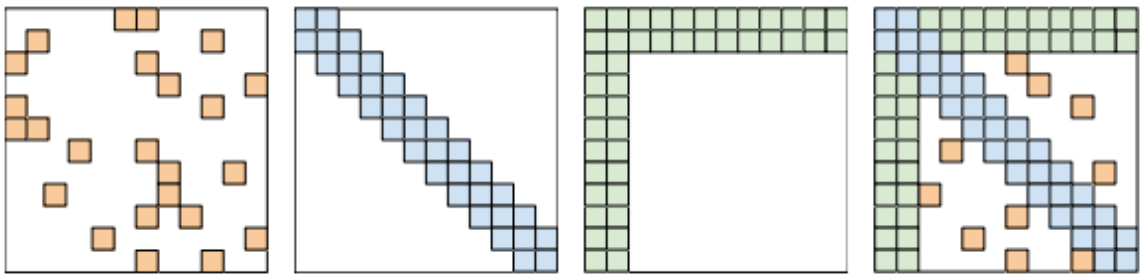

Figure 6: Big Bird attention pattern (Zaheer et al., 2020)

3.1.4 Adaptive Attention Span

Adaptive attention span (Sukhbaatar et al., 2019) is a proposed self-attention mechanism which tries to learn an optimal attention span. With this technique the maximum context size used in Transformer can be extended significantly. The authors hypothesized that within the same context window different attention heads might assign scores differently and thus the optimal span would be trained separately per head.

For each head, a masking function is added to control the span of the attention. The masking function is a non-increasing function that maps a distance to a value in $[0, 1]$. The authors chose the following soft masking function m_z parametrized by a real value z in $[0, S]$:

$$m_z(x) = \min\left[\max\left[\frac{1}{R}(R + z - x), 0\right], 1\right] \tag{6}$$

where R is a hyper-parameter which controls the softness. The attention weights are then computed with the following formula:

$$a_{tr} = \frac{m_z(t-r)\exp(s_{tr})}{\sum_{q=t-S}^{t-1} m_z(t-q)\exp(s_{tq})} \tag{7}$$

where s_{tr} is defined as follows:

$$s_{tr} = x_t^T W_q^T (W_k x_r + p_{t-r}) \tag{8}$$

where p_{t-r} is a relative positional embedding.

In addition to that, the l_1 penalization term is added to the parameters z_i for each attention head i in the loss function to penalize longer spans. The formulation is differentiable with respect to z_i and they are trained jointly with the model.

3.2 Non-Sparsity Based Improvements

In the second section covering the different variants non-sparsity based improvements are discussed. Although sparse attention is also improving the computational and memory efficiency there are several other mechanisms to achieve that. These include low rank approximations of the attention matrix and learned access pattern to reduce the complexity of the attention matrix.

3.2.1 Reformer

Reformer (Kitaev et al., 2020) is a model belonging to the class of learnable patterns. It uses hash-based similarity measures to cluster tokens into chunks. The idea of using clusters is to learn the sparsity pattern of the attention matrix.

Reformer is a model based on locality sensitive hashing. It hashes the queries and keys into buckets by using random projection. The goal is to cluster nearby vectors into the same cluster. This is achieved when nearby vectors obtain similar hash values. During the attention calculation tokens can only attend to the same bucket in its own chunk and previous chunk. To perform the hashing, Reformer uses random matrix $R \in \mathbb{R}^{k \times b/2}$. The hashing function is then defined as follows:

$$h(x) = argmax([xR; -xR]) \tag{9}$$

where [;] is concatenation of two vectors. The attention is then computed only for when the query and key are in the same bucket or an adjacent bucket, so when $h(q_i) = h(k_j)$

Reformer achieves the complexity of $\mathcal{O}(n \log(n))$

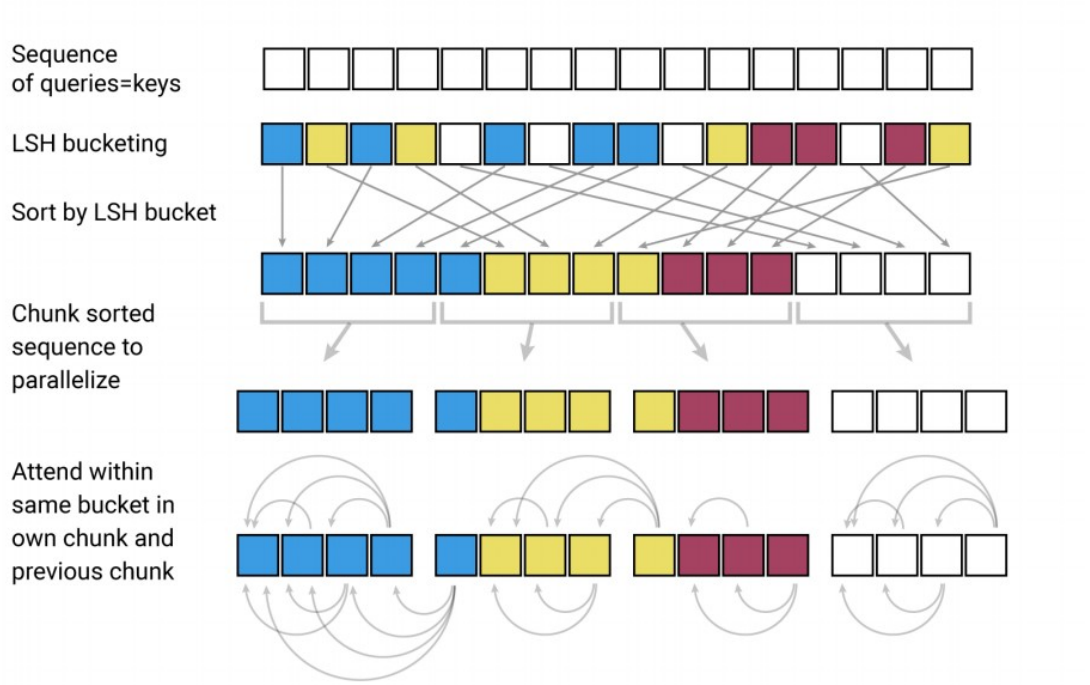

Figure 7: Simplified LSH Attention (Kitaev et al., 2020)

3.2.2 Performer

Performer (Choromanski et al., 2021) is an architecture belonging to the kernel class. Its main idea is that it tries to estimate the softmax full rank attention matrix by using only linear complexity without using any priors. It uses generalized attention and random kernels.

Performer uses approximation tricks to avoid the quadratic complexity. The approximation of softmax attention-kernel is based on Fast Attention Via positive Orthogonal Random features approach (FAVOR+). By using orthogonal random features (ORF) the attention is obtained via:

$$\hat{Att}(Q, K, V) = \hat{D}^{-1}(Q'((K')^T V)) \tag{10}$$

where $\hat{D}$, Q' and K' are defined as follows:

$$\begin{aligned} \hat{D} &= \mathrm{diag}(Q'((K')^T 1_N)) \\ Q' &= \phi(Q^T)^T \\ K' &= \phi(K^T)^T \end{aligned} \tag{11}$$

The definition of ϕ is very complex and it is out of the scope of this thesis.

Performer is compatible with the original Transformer and it has theoretical guarantees of unbiased estimation of the attention matrix, uniform convergence and low estimation variance.

Performer achieves linear complexity $\mathcal{O}(N)$

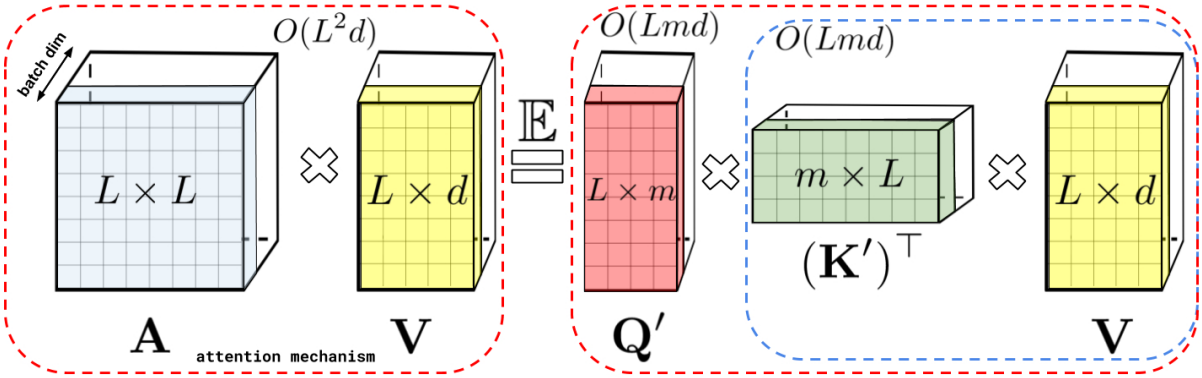

Figure 8: Approximation of the regular attention mechanism via feature maps. The dashed blocks indicate the order of computation (Choromanski et al., 2021)

3.2.3 Linformer

The Linformer (Wang et al., 2020) is a low-rank based method since it projects the keys and values to lower dimensional representation. Linformer uses additional projection layers to project the $N \times d$ dimensional keys and values to $k \times d$ dimension. By doing this the $N \times N$ attention matrix is decomposed to $N \times k$. It is based on the idea that the attention matrix is low rank and can therefore be approximated with a matrix of lower dimension.

Attention heads are calculated with the following formula:

$$Softmax(\frac{1}{\sqrt{d_k}} XW_i^Q (E_i XW_i^K)) * F_i XW_i^V \tag{12}$$

Where W^Q, W^K and W^V are the original linear transformations and E and F matrices are the additional projection matrices. Linformer reduces the complexity to $\mathcal{O}(N)$.

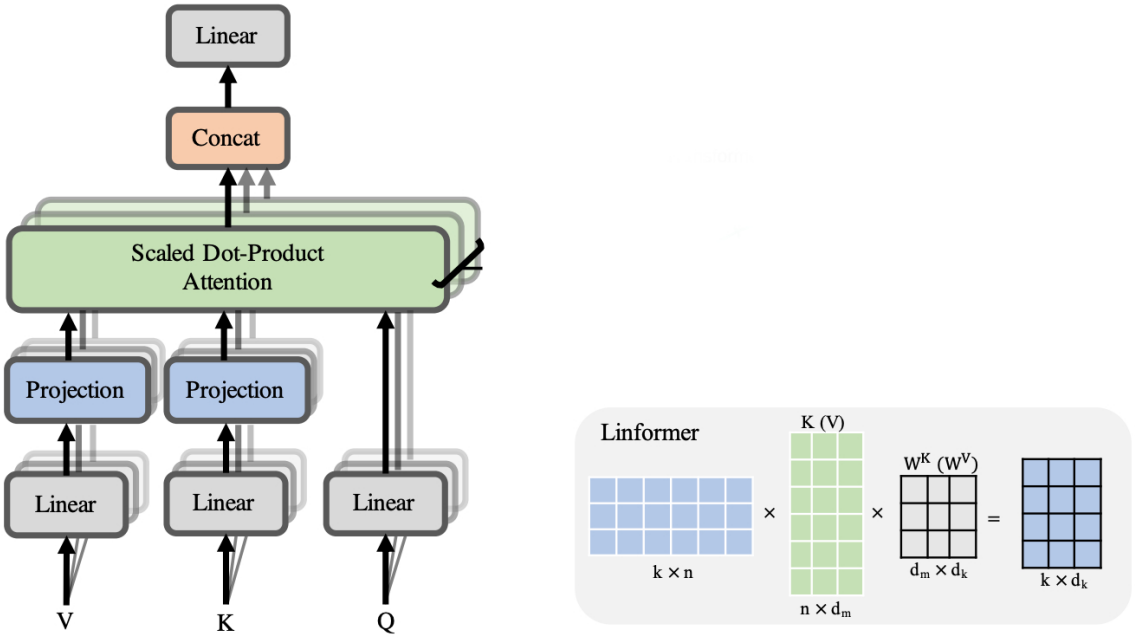

Figure 9: Linformer architecture (Wang et al., 2020)

3.2.4 Linear Transformer

Linear Transformer (Katharopoulos et al., 2020) is another Transformer model in the kernel class. It improves the complexity of self-attention of the original Transformer from quadratic to linear by using the kernel function. The generalized attention equation with arbitrary similarity function can be written as follows:

$$V_i^{'} = \frac{\sum_{j=1}^{N} \text{sim}(Q_i, K_j) V_j}{\sum_{j=1}^{N} \text{sim}(Q_i, K_j)} \tag{13}$$

The similarity function in Linear Transformer is expressed as kernel function. It is defined with the following formula:

$$\text{sim}(q, k) = \phi(q)^T \phi(k) \tag{14}$$

Given this formulation and using the associative property of matrix multiplication the equation for $V_i^{'}$ can be rewritten as follows:

$$V_i^{'} = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)} \tag{15}$$

3.3 Non-Attention Improvements

This section discusses some other variants that are introduced in the literature like Transformer-XL (Dai et al., 2019) and Universal Transformer (Dehghani et al., 2018). These architectures do not try to attempt the problem with the self-attention, instead they use other mechanisms to improve the vanilla Transformer architecture. These mechanisms include tricks such as adding recurrence to the Transformer or improving the model's input embedding by using relative positional embedding instead of absolute positional embedding.

3.3.1 Self-Attention with Relative Position Representations

The self-attention with relative position representations (Shaw et al., 2018) paper proposes an alternative approach to the absolute position encodings used in the original Transformer. It introduces a way of using pairwise distances as a way of creating positional encodings. The relative positional information is added to the model both in values and keys. In the equation below relative positional information is added to the model as an additional component to the keys when calculating the dot product between queries and keys.

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}} \tag{16}$$

The softmax operation remains the same from the original Transformer. After that the relative positional information is also added to the value matrix.

$$z_i = \sum_{j=1}^{n} \alpha_{ij} (x_j W^V + a_{ij}^V) \tag{17}$$

In the equations above a_{ij}^K and a_{ij}^V are the edge representations. These are defined via the following equations:

$$\begin{aligned} a_{ij}^K &= w_{clip(j-i,k)}^K \\ a_{ij}^V &= w_{clip(j-i,k)}^V \\ \text{clip}(x,k) &= \max(-k, \min(k,x)) \end{aligned}$$

3.3.2 Transformer-XL

One problem with the original Transformer architecture is that it has a fixed and limited attention span meaning that the model can only attend to other tokens within the

same segment and information can not flow across separate fixed-length segments. This approach has several issues. The first one is that the model can not capture very long term dependencies. The second one is that it is hard to predict the first few tokens in each segment given no context. The third one is that the evaluation is expensive which comes from the need to re-process each new segment when the segment is shifted to the right by one even though it contains a lot of overlapped tokens.

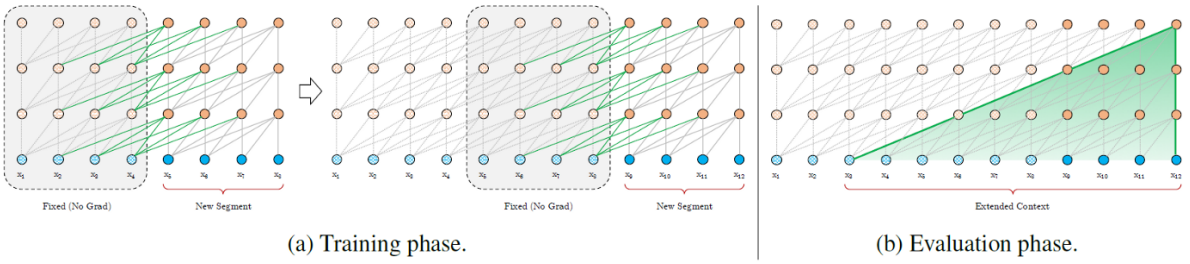

Figure 10: Transformer-XL with segment length 4 (Dai et al., 2019)

The Transformer-XL (Dai et al., 2019) solves the context segmentation problem by reusing hidden states between segments and adopting new positional encoding that is suitable for reused states. It introduces recurrent connection between segments by continuously using the hidden states from the previous segments.

The Transformer-XL uses another type of positional encoding. Instead of using the absolute position encoding which is used in the original Transformer, Transformer-XL uses relative positional encoding. This prevents the previous and current segment being assigned with the same encoding.

3.3.3 Universal Transformer

Universal Transformer (Dehghani et al., 2018) is an architecture which aims to address the issues the original Transformer model fails to generalize to that recurrent based models handle well. Universal Transformer can be thought of as a generalization of the Transformer model. It combines self-attention with recurrent mechanism and tries to benefit from both a long-term global receptive field of Transformer and learned inductive biases of RNN.

Instead using fixed number of layers, Universal Transformer dynamically adjust the number of steps both in the encoder and decoder layers. It also uses different input representation, namely timestep embedding which is added to the positional encoding.

At each time step t the Universal Transformer computes $H^t \in \mathbb{R}^{m \times d}$ for all input positions m with the following formula:

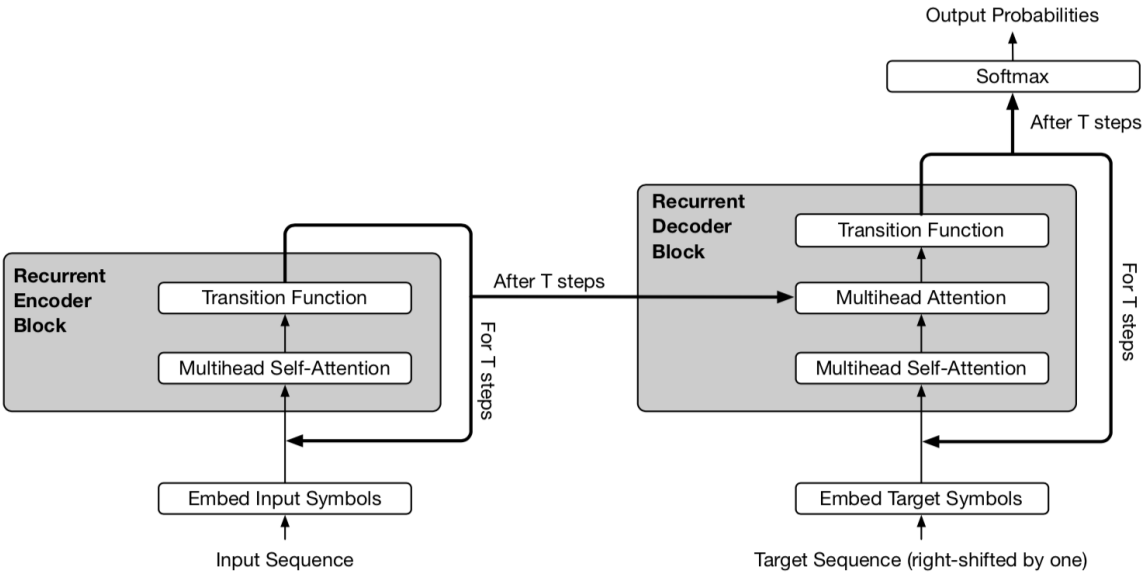

Figure 11: The recurrent blocks of he Universal Transformer encoder and decoder (Dehghani et al., 2018)

$$H^t = \text{LayerNorm}(A^t + \text{Transition}(A^t)) \tag{18}$$

$$A^t = \text{LayerNorm}((H^{t-1} + P^t) + \text{MultiHeadSelfAttention}(H^{t-1} + P^t)) \tag{19}$$

where Transition() can be either a separable convolution or a fully-connected neural network that consists of two position-wise affine transformation and ReLU function.

4 Key Takeaways

As seen, scaling the original Transformer model for longer sequences becomes computationally expensive. This problem comes mainly from the quadratic complexity of the self-attention matrix. This has resulted in the many Transformer variants published in the literature. These variants have proposed many different ways of solving this scalability issue. This thesis covered some of these techniques. Perhaps most importantly sparsifying the attention matrix and the kernel/low rank based methods. In addition to improving the computational complexity of the attention matrix, this thesis also covered some non-attention based improvements.

5 Summary

This thesis has covered many variants of Transformer architecture. The first part discussed the sparse attention mechanism. This part covered architectures such as Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020). The second part discussed the non-sparsity based improvements made on the original Transformer architecture. This part covered architectures such as Reformer (Kitaev et al., 2020) and Performer (Choromanski et al., 2021). Finally the third part covered non-attention improvements which included architectures like Universal Transformer (Dehghani et al., 2018) and Transformer-XL (Dai et al., 2019).

The goal of this thesis was to introduce different mechanisms to overcome the problem of the self-attention matrix in the original Transformer. As seen there are many variants proposed in the literature to solve this problem. Although this thesis covers only some of them, there are many more proposed architectures and the research is progressing rapidly.

References

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

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser et al. Rethinking attention with performers. *International Conference on Learning Representations*, 2021.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

Nikita Kitaev, Łukasz Kaiser and Anselm Levskaya. Reformer: The efficient transformer. *International Conference on Learning Representations*, 2020.

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

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski and Armand Joulin. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019.

Yi Tay, Mostafa Dehghani, Dara Bahri and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

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.

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

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.