UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Exploring Causal Attention Models in Transformers for Large Language Models

João Filipe Gonçalves Vieira Terroa

Mestrado em Informática

Dissertação orientada por:
Prof. Doutor André Osório E Cruz De Azerêdo Falcão

2024

Acknowledgments

I would like to begin by expressing my sincerest gratitude to my supervisor, Professor André Falcão, for his unwavering guidance, support, and encouragement throughout this work. His genuine interest in my work has been both contagious and inspiring. I am truly thankful for the opportunity to work under his supervision and for the freedom he granted me to explore my ideas and develop my research path. I greatly appreciate the time he dedicated to reading and discussing my work, and for the valuable feedback he provided.

I dedicate this thesis to my family and friends, whose love, support, and encouragement have been my greatest source of strength. To my family, I offer my deepest thanks for always being there for me, for believing in me, and for the sacrifices they made to help me achieve my goals, both in my studies and in life.

To my friends, I am deeply grateful for the good times we shared, the laughter, conversations, and the relentless support they gave me. Thank you for listening patiently to the ramblings of an excited but tired student, and for helping me relax and take my mind off work when needed. The time we spent together and the memories we made will always be special to me.

Resumo Alargado

Os transformadores têm desempenhado um papel preponderante no campo do Processamento de Linguagem Natural (PLN) desde a sua introdução em 2017. Esta arquitetura inovadora baseia-se no mecanismo de auto-atenção, que permite capturar dependências a longa distância em sequências de dados, superando as limitações inerentes aos modelos anteriores, como as redes neurais recorrentes (RNNs) e as redes de memória de curto e longo prazo (LSTMs). A arquitetura do transformador distingue-se, principalmente, por permitir processamento paralelo eficiente, facilitando o treino em grandes volumes de dados. A capacidade de processar estas dependências de maneira eficaz permitiu avanços significativos em diversas aplicações, desde a tradução automática até à geração de texto, moldando a forma como interagimos com a tecnologia diariamente. Modelos baseados em transformadores encontram-se amplamente integrados em assistentes virtuais, que processam comandos de voz para fornecer respostas precisas. Em sistemas de recomendação, frequentemente utilizados em plataformas de e-commerce e entretenimento, estes modelos identificam padrões comportamentais, oferecendo sugestões personalizadas de produtos, filmes ou músicas. Ferramentas de correção gramatical e funções de predição de texto em *smartphones*, também beneficiam desta tecnologia, tornando mais eficiente a comunicação digital e melhorando a experiência do utilizador. Além disso, os transformadores também são amplamente utilizados em sistemas de diagnóstico médico, onde auxiliam na análise de imagens e relatórios clínicos para identificar doenças e apoiar decisões médicas automatizadas. Porém, apesar dos benefícios evidentes, os modelos baseados em transformadores enfrentam desafios significativos, entre os quais o elevado custo computacional, resultante do grande número de parâmetros e da complexidade dos cálculos. Este fator, aliado ao consumo substancial de energia, levanta preocupações ambientais, uma vez que o treino e a execução de modelos de larga escala exigem centros de dados de alta capacidade, cuja operação envolve um elevado consumo energético.

Com a crescente preocupação acerca da sustentabilidade e acessibilidade tecnológica, a pesquisa tem-se focado no desenvolvimento de modelos mais compactos e eficientes. O objetivo é possibilitar a execução de modelos comparáveis aos de estado da arte, em dispositivos com recursos limitados, como *smartphones*, eliminando a dependência de infraestruturas extensivas em *cloud*. Isto não só democratiza o acesso a tecnologias avançadas, como também reduz a latência e melhora a privacidade dos utilizadores, uma vez que os dados não precisam de ser transmitidos para servidores remotos. O desenvolvimento de modelos compactos e eficientes garante que a inovação tecnológica pode continuar a evoluir sem comprometer o acesso ou a escalabilidade. Neste

contexto, a otimização e eficiência dos transformadores tornaram-se áreas de destaque, impulsionando investigações que buscam equilibrar desempenho e recursos computacionais.

Apesar do interesse crescente, o bloco fundamental dos transformadores manteve-se relativamente inalterado desde a sua conceção inicial. A compreensão do funcionamento interno desta arquitetura baseia-se, em grande medida, em observações empíricas, deixando lacunas no entendimento teórico das suas funções exatas. Além disso, estudos recentes sugerem que é possível remover componentes tradicionalmente considerados essenciais—como conexões residuais, parâmetros de projeção ou valor, sub-blocos sequenciais e camadas de normalização—sem comprometer a velocidade de treino ou a eficácia do modelo. Estes resultados desafiam as noções estabelecidas sobre a indispensabilidade de certos componentes e sugerem que existe espaço para reavaliar os componentes fundamentais do transformador. Explorar configurações alternativas pode oferecer vantagens em termos de eficiência e desempenho, permitindo a criação de modelos mais leves e rápidos.

Em particular, o mecanismo de auto-atenção, elemento central dos transformadores, permaneceu fundamentalmente inalterado desde 2017. Embora várias otimizações tenham sido propostas para melhorar a sua eficiência, a implementação básica mantém-se. Por exemplo, abordagens como a atenção esparsa e a atenção linear têm sido exploradas para reduzir a complexidade computacional, mas geralmente envolvem compromissos que afetam a capacidade do modelo de capturar dependências globais. Esta persistência levanta questões sobre se a abordagem atual representa, de facto, a forma mais eficaz de realizar a atenção em transformadores para modelos de linguagem em larga escala (em inglês, *large language models* ou LLMs). Com base nestas considerações, propõe-se uma investigação que visa analisar teórica e empiricamente o mecanismo de auto-atenção, examinando as bases teóricas subjacentes aos esforços de otimização existentes e avaliando que alternativas podem conduzir a melhorias.

A tarefa consiste em implementar e avaliar configurações alternativas do mecanismo de auto-atenção em diferentes conjuntos de dados. O objetivo é identificar modificações que melhorem métricas de desempenho, como precisão, eficiência e estabilidade, sem recorrer ao desenvolvimento de novas arquiteturas que exigiriam uma investigação extensa desde o início. Ao focar-se em ajustes dentro do paradigma estabelecido, pretende-se contribuir para a otimização prática dos modelos existentes, tornando-os mais adequados para aplicações em ambientes com recursos limitados.

O objetivo central deste trabalho é fornecer uma análise teórica e prática inovadora do mecanismo de auto-atenção nos transformadores, questionando a suposição de que a implementação atual representa a forma ótima de atenção. Ao desafiar paradigmas estabelecidos e explorar alternativas fundamentadas, espera-se abrir caminhos para o desenvolvimento de modelos mais eficientes e acessíveis, contribuindo para o avanço do estado da arte no processamento de linguagem natural.

Para este estudo, foram desenhadas e implementadas cinco modificações distintas no mecanismo de auto-atenção: Simple Self-Attention (SSA), Layered Self-Attention (LSA), Variable

Self-Attention (VSA), Simple Layered Self-Attention (SLSA) e Variable Layered Self-Attention (VLSA). A implementação consistiu em duas fases principais: um trabalho exploratório e uma análise confirmatória. A fase exploratória iniciou-se com alterações no mecanismo de auto-atenção, seguida de testes mais extensivos das alternativas promissoras (LSA, SLSA e VSA) utilizando o repositório nanoGPT. Diversas configurações foram testadas em diferentes tamanhos de conjuntos de dados, com *scripts* automatizados a facilitar o processo de treino. A fase de análise confirmatória centrou-se no ajuste fino dos mecanismos de atenção com melhor desempenho, avaliando cinco versões modificadas juntamente com a auto-atenção original, que serviu como baseline, utilizando parâmetros de treino consistentes em múltiplos ensaios.

Os resultados das análises exploratória e confirmatória indicaram que os modelos Variable Layered Self-Attention (VLSA), especialmente aqueles com valores mais elevados de k, superaram o mecanismo de auto-atenção padrão. Em ambos os cenários de 40.000 e 80.000 iterações, os modelos VLSA demonstraram desempenho superior, alcançando menores perdas de validação e evidenciando capacidades de generalização aprimoradas. Notavelmente, os modelos VLSA com $k = 2$ e $k = 3$ atingiram resultados superiores ao modelo baseline, mesmo quando treinados com menos iterações, destacando uma utilização mais eficiente dos recursos computacionais.

Estes achados sugerem que mecanismos alternativos de atenção, especificamente os modelos VLSA, podem melhorar o desempenho de modelos de linguagem baseados em transformadores sem a necessidade de alterações arquiteturais extensivas. Ao integrarem-se harmoniosamente em estruturas existentes, estas modificações oferecem uma abordagem prática para aprimorar a eficiência e a precisão dos modelos.

Em conclusão, o estudo demonstra que a implementação atual de auto-atenção nos transformadores não representa necessariamente a abordagem ótima, e que a exploração de mecanismos alternativos pode resultar em melhorias significativas. Os resultados sublinham o potencial para avanços adicionais na modelagem de linguagem, mesmo dentro das restrições de arquiteturas estabelecidas.

Como perspetiva futura, propõe-se a aplicação dos mecanismos desenvolvidos a modelos e conjuntos de dados de maior escala, bem como a expansão das métricas de avaliação para incluir um espectro mais amplo de benchmarks. Ao escalar os modelos e incorporar métricas de avaliação padronizadas, como perplexidade e perda de entropia cruzada, espera-se que as melhorias observadas se generalizem, proporcionando uma compreensão mais abrangente dos benefícios dos mecanismos de atenção propostos.

Palavras-chave: transformadores, mecanismos de auto-atenção, processamento de linguagem natural, modelos de linguagem, otimização de desempenho

Abstract

Transformers have played a significant role in Natural Language Processing (NLP) since 2017, enabling significant advancements in applications like machine translation and text generation. Despite their success, they face challenges such as high computational costs and environmental impacts due to energy consumption.

Recent research has focused on the development of more compact and efficient models operable on resource-limited devices without reliance on cloud infrastructures. Optimizing transformers seeks to balance performance with computational resources.

The fundamental structure of transformers remains largely unchanged, with understanding based on empirical observations, leaving theoretical gaps. Studies suggest that components considered essential can be removed without compromising performance, indicating potential for reevaluating transformer components.

This study aims to analyze the self-attention mechanism theoretically and empirically, examining existing optimizations and evaluating alternatives for improvements. Five modifications were designed: Simple Self-Attention (SSA), Layered Self-Attention (LSA), Variable Self-Attention (VSA), Simple Layered Self-Attention (SLSA), and Variable Layered Self-Attention (VLSA).

Implementation involved exploratory and confirmatory phases. The exploratory phase altered the self-attention mechanism and extensively tested promising alternatives using the nanoGPT repository. The confirmatory phase fine-tuned the best-performing mechanisms, evaluating modified versions alongside the original self-attention as a baseline.

Results indicated that Variable Layered Self-Attention (VLSA) models, especially with higher k values, outperformed the standard self-attention mechanism, achieving lower validation losses and improved generalization, even with fewer training iterations.

These findings suggest that alternative attention mechanisms can enhance transformer-based language models without extensive architectural changes, offering a practical approach to improving efficiency and accuracy.

In conclusion, this study demonstrates that the current self-attention implementation may not be optimal, and exploring alternative mechanisms can lead to significant improvements. Future work proposes applying these mechanisms to larger models and datasets and expanding evaluation metrics to include broader benchmarks, aiming to generalize the improvements and better understand the benefits of the proposed attention mechanisms.

Keywords: Transformer architecture, self-attention mechanism, natural language processing, language models, model efficiency

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The Transformer architecture, introduced by Vaswani et al. [2] in 2017, has marked a significant milestone in the field of NLP and artificial intelligence. Since its introduction, the Transformer has been widely adopted for various tasks, such as machine translation, text summarization, and language modeling [5]. Central to the Transformer is the self-attention mechanism [6], which enables the model to dynamically assign importance to different parts of the input, thus enhancing its ability to capture subtle linguistic variations and contextual nuances. This capability contrasts with traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which struggle to capture long-range dependencies due to their sequential or localized operations.

Despite the widespread adoption of Transformers in NLP research [7, 8], and the emergence of state-of-the-art models like OpenAI's GPT-4o[1], Anthropic's Claude 3.5 Sonnet[2], and Google's Gemini 1.0 Ultra [9], the theoretical foundations of the self-attention mechanism remain largely unexplored. Huang et al. [10] point out that,

> "current explanations of this mechanism are mainly based on intuitions and experiences, while there still lacks direct modeling for how the self-attention mechanism helps performance".

While many components of the Transformer, such as positional encoding schemes, layer normalization techniques, and training strategies, have undergone a significant evolution, the core scaled dot-product self-attention mechanism has remained largely unchanged since its introduction in 2017, albeit with optimizations for improved efficiency and performance.

Another important issue is the current dominance of Transformer-based architectures in NLP. This dominance is not solely due to performance but is also influenced by practical considerations, such as cost and ease of integration. Transformers have established themselves as successful across a variety of tasks and benefit from pre-trained models, which makes them more attractive than developing entirely new architectures (see Figure 1.1). Training large-scale models from scratch is

[1] OpenAI, "Hello GPT-4o". Available at: `https://openai.com/index/hello-gpt-4o/`.

[2] Introducing Claude 3.5 Sonnet. Available at: `https://www.anthropic.com/news/claude-3-5-sonnet` [Accessed 24 June 2024].

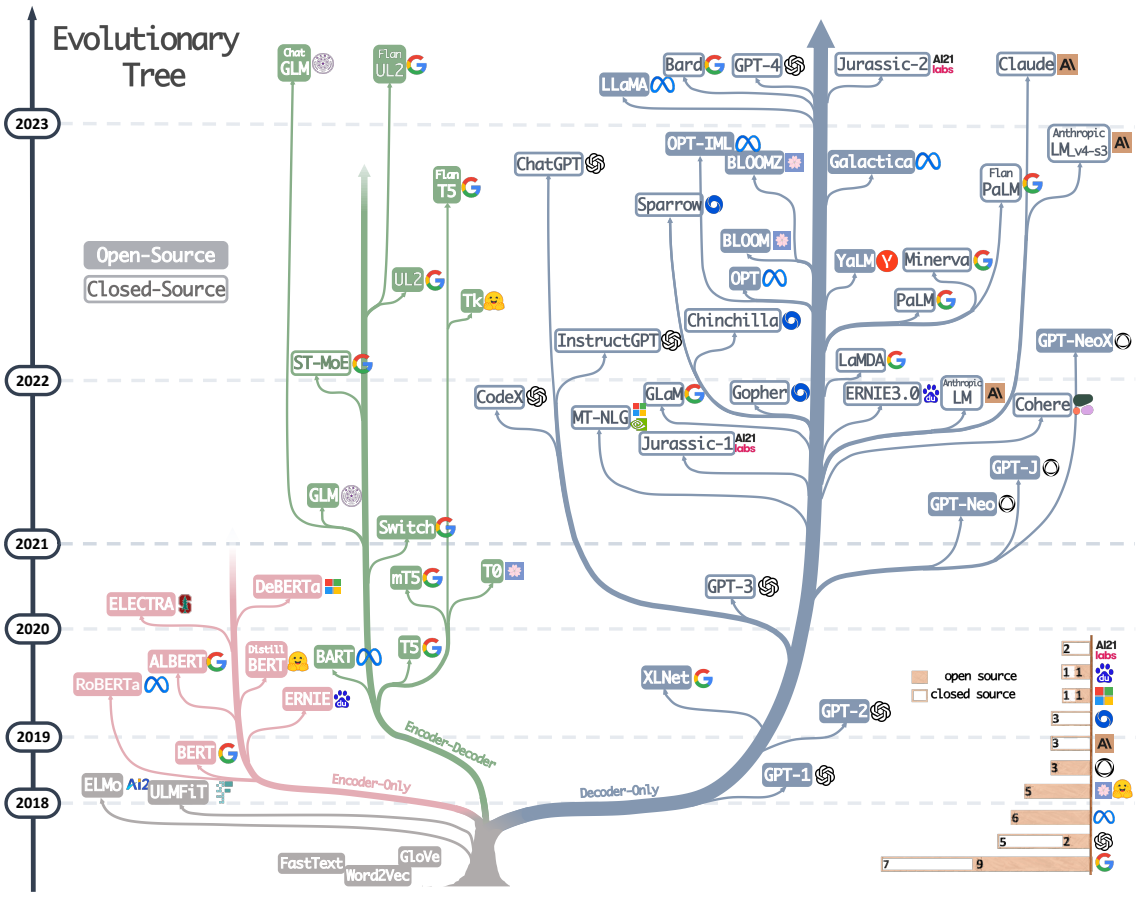

Figure 1.1: The evolutionary tree of modern LLMs traces the development of language models in recent years and highlights some of the most well-known models. Models on the same branch have closer relationships. Transformer-based models are shown in non-grey colors: decoder-only models in the blue branch, encoder-only models in the pink branch, and encoder-decoder models in the green branch. The vertical position of the models on the timeline represents their release dates. Open-source models are represented by solid squares, while closed-source models are represented by hollow ones. The stacked bar plot in the bottom right corner shows the number of models from various companies and institutions. (Figure from J. Yang et al. [1])

resource-intensive, both economically and environmentally[3]. It requires substantial computational power, time, and energy, making it impractical to retrain or explore fundamentally different architectures. Companies such as NVIDIA, which provide the hardware supporting these models, along with other AI developers, have clear incentives to enhance and fine-tune existing architectures[4], rather than pursue novel designs. Open-source initiatives, like Meta's Llama2 and Llama3, further reinforce reliance on Transformer models by offering accessible, pre-trained architectures[5]. This

[3]K. Hao, "AI model carbon emissions", MIT Technology Review. `https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/` [Accessed June 2024]

[4]A. Patel, "Nvidia accelerates Meta Llama 3", NVIDIA Blog. `https://blogs.nvidia.com/blog/meta-llama3-inference-acceleration/` [Accessed June 2024]

[5]K. McCann, "Meta Llama 3.1: Largest open-source AI model", AI Magazine. `https://aimagazine.com/articles/meta-llama-3-1-the-worlds-largest-open-source-ai-model` [Accessed July 2024]

commitment to Transformers presents challenges for the adoption of new models, which would require addressing issues such as compatibility with existing infrastructure, retraining from scratch, and the need for novel optimizations—all at significant cost and with uncertain benefits.

A further challenge arises from the mathematical ambiguity of self-attention. While the term "attention" provides an intuitive description of the model's operation, the output of the self-attention mechanism, prior to softmax normalization, is essentially a cubic polynomial of the input data [11]. Recent studies suggest that the conventional design of causal self-attention may be suboptimal for LLMs [12]. Additionally, research shows that removing components traditionally considered essential, such as skip connections and sequences of sub-blocks, has little impact on performance [13]. These findings indicate that the understanding of Transformers, particularly the self-attention mechanism, remains incomplete.

This study seeks to offer a new perspective on attention mechanisms in Transformer-based language models. The scaled dot-product self-attention is not interpreted as an algorithm that directly models attention, but rather as a polynomial function that approximates attention when iteratively applied across multiple layers with sufficient data. In this context, potential modifications to the standard self-attention mechanism are explored, with the aim of identifying alternatives that do not directly compete with existing architectures. This approach is intended to facilitate a smoother integration of the potential modifications into current frameworks, while also providing a foundation for future research and development.

1.2 Goals

The main objective of this thesis is to investigate how various changes to the self-attention mechanism affect the performance of Transformer-based Large Language Models (LLMs). The goal is to explore which modifications improve model accuracy, efficiency, and stability when applied to different datasets, in a small-scale environment. To achieve this, this work aimed to accomplish the following objectives:

- **Understanding foundational concepts in NLP and Transformer models:** A comprehensive literature review was carried out to understand the original self-attention mechanism presented in "Attention is All You Need" and its later adaptations. In addition, this review also covered recent advancements related to modifications of the self-attention mechanism.
- **Data collection and preparation:** A range of datasets was chosen to enable a well-rounded evaluation of the proposed modifications. These datasets were carefully gathered and processed to ensure high quality and diversity, which are essential for accurate model assessments.
- **Implementation, testing, and performance evaluation:** The proposed modifications to the self-attention mechanism were implemented and tested on the selected datasets. The modified models were then evaluated to identify configurations that improve model accuracy, efficiency, and stability across different datasets.

1.3 Document Structure

This document is organized as follows:

- **Chapter 1: Introduction**
  - This chapter provides the motivation behind the study and outlines the main goals of the thesis.
- **Chapter 2: Background**
  - This chapter provides an overview of the foundational elements of Transformer-based large language models, covering key concepts such as attention mechanisms, tokenization, and embeddings.
- **Chapter 3: Related Work**
  - This chapter reviews existing research on Transformer-based architectures, focusing on model efficiency, generalization, and application-specific modifications, as well as architecture-level and attention mechanism improvements. The chapter concludes with potential future research directions in the field.
- **Chapter 4: Methods**
  - This chapter outlines the proposed modifications to the self-attention mechanism explored in this study. It also provides an overview of the evaluation metrics employed to assess the models' performance.
- **Chapter 5: Data Acquisition and Preparation**
  - This chapter provides an overview of the datasets utilized, detailing their properties and characteristics. It also describes the preprocessing techniques applied to prepare the data for model training.
- **Chapter 6: Implementation**
  - This chapter focuses on the implementation of the proposed models and methods. It outlines the exploratory work conducted to develop and test initial hypotheses regarding causal attention models. The confirmatory analysis section describes the more formalized tests and experiments used to validate findings from the exploratory phase. Furthermore, model benchmarking is presented, including detailed performance evaluations of models from both the exploratory and confirmatory phases. This section highlights the computational setup, software tools, and model training processes employed in the study.
- **Chapter 7: Results and Discussion**

  - The results from this work are presented in this chapter. It provides a comparative performance analysis and discusses the statistical test results for both phases of modifications. The discussions critically analyze the findings in relation to this work goals.

- **Chapter 8: Conclusion**

  - The results are presented in this chapter, which includes a comparative performance analysis and a discussion of statistical test outcomes for both phases of the modifications. The findings are critically examined in light of the study's objectives.

- **Appendices**

  - The appendices include detailed results from the Exploratory Work and Confirmatory Analysis, as well as individual dataset outcomes and statistical test results.

Chapter 2

Background

This chapter presents an overview of Transformer-based large language models, highlighting the primary concepts and components derived from the original Transformer architecture. These include attention mechanisms, multi-head attention, residual connections, normalization, and position-wise feed-forward networks. In addition, the discussion extends to tokenization and embeddings, essential elements for comprehending the process by which these models interpret and represent textual data.

2.1 Introduction

The Transformer architecture, introduced in the 2017 paper *Attention Is All You Need* [2], was originally proposed as a sequence-to-sequence model [14] for machine translation, and served as an alternative to using recurrent or convolutional layers. The vanilla Transformer consists of multiple identical Transformer blocks in sequence, with the overall Transformer model organized into an encoder-decoder structure. Each encoder and decoder block comprises two main sub-blocks [15]: a multi-head self-attention mechanism module [16] and a position-wise feed-forward network (FFN), alongside residual connections [17] and layer normalization [18]. In contrast to the encoder blocks, decoder blocks additionally insert cross-attention modules between the multi-head self-attention modules and the position-wise FFNs. Furthermore, the self-attention modules in the decoder are adapted to prevent each position from attending to subsequent positions. The overall architecture of the vanilla Transformer is shown in Figure 2.1.

The following subsections introduce the key modules of the vanilla Transformer.

2.1.1 Self-Attention

The concept of attention in neural networks is rooted in the principles of *selective attention* from cognitive science. Selective attention in psychology refers to the conscious prioritization and processing of certain stimuli over others [19], a process mirrored in LLMs, where attention mechanisms and positional encodings determine which input elements are emphasized and the order of their processing [6, 2], respectively.

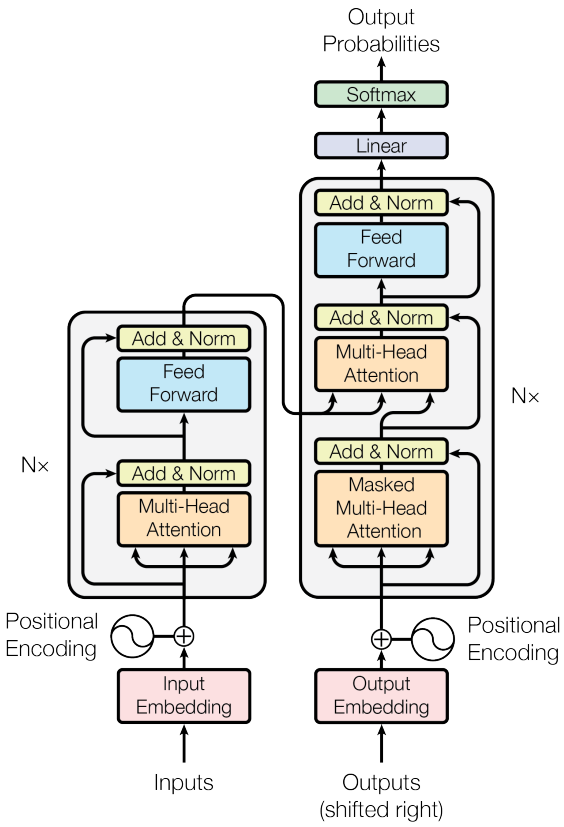

Figure 2.1: The original Transformer architecture, with the encoder and decoder components. (Figure from A. Vaswani et al. [2])

The self-attention mechanism allows the Transformer to attend to each token in an input sequence and determine its relative importance to every other token in that sequence. This enables the model to capture complex relationships between words and phrases, even if they are not sequential. In a Transformer, the self-attention mechanism can be formalized as follows:

Given an input sequence of tokens $x_1, x_2, ..., x_n$, each token x_i is first mapped to a triplet of vectors: a query vector q_i and a key vector k_i of dimension d_k, and a value vector v_i of dimension d_v. These vectors are computed using learned linear transformations of the input embeddings. The matrices Q, K, and V are then formed by concatenating the query, key, and value vectors for all tokens in the sequence, respectively. The self-attention weights between tokens x_i and x_j are then computed using a scaled dot-product function [2]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V = \mathbf{A}V \tag{2.1}$$

with d_k being the dimensionality of the key vectors used for scaling; $\mathbf{A} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$ is often called *attention matrix*; softmax is applied in a row-wise manner. The dot-products of queries and keys are divided by $\sqrt{d_k}$ to alleviate the gradient vanishing problem of the softmax function. The self-attention layer directly connects all positions in the sequence, allowing the model to capture long-range dependencies with a computational complexity of $O(N^2 \cdot d)$ [20,

21, 22, 23], where N is the sequence length and d is the dimensionality of the vectors. The quadratic complexity comes from the pairwise computation of attention weights for all tokens in the sequence.

2.1.2 Causal Attention

Causal attention, also known as autoregressive attention, is a specific type of self-attention that ensures that the prediction for a token can only be influenced by preceding tokens in the sequence. This is especially important for tasks such as text generation, where the future tokens should not influence the current token's output. Causal attention is implemented by masking out future tokens in the self-attention weight matrix:

$$\text{CausalAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V \tag{2.2}$$

The mask M ensures that the attention mechanism only focuses on previous positions and not future positions. It has an upper triangular form, where the lower part (including the diagonal) contains zeros (allowing attention) and the upper part contains negative infinity[1], preventing tokens from peeping into the future. Mathematically, for a matrix M of size $n \times n$ (where n is the sequence length), the element $M_{i,j}$ (where i is the row index and j is the column index) is set as:

$$M_{i,j} = \begin{cases} -\infty & \text{if } i < j \\ 0 & \text{if } i \geq j \end{cases} \tag{2.3}$$

This mask ensures that during the softmax step of the attention mechanism, positions after the current one have a negligible impact due to the effect of the large negative value after applying softmax.

2.1.3 Cross-Attention

Cross-attention allows the Transformer to integrate information from different sources. While self-attention processes the input sequence by relating each token to every other token in the same sequence, cross-attention lets the model attend to tokens in another sequence. This is particularly useful in tasks like machine translation, where one sequence represents the source language and the other represents the target language.

In cross-attention, the query vectors Q come from the target sequence, while the key K and value V vectors come from the source sequence. The attention weights are then computed similarly to self-attention. This mechanism allows the model to focus on relevant parts of the source sequence when generating each token in the target sequence.

2.1.4 Multi-Head Attention

Multi-head attention extends the idea of self-attention by using multiple sets of query, key, and value projections. Instead of performing a single attention function, the multi-head mechanism

[1] In practice, this infinity value can be viewed as a very large negative number.

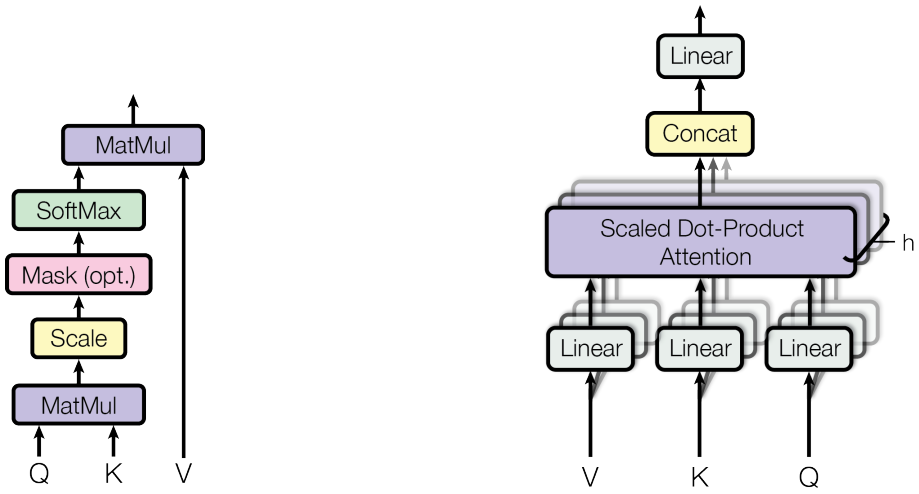

Figure 2.2: (left) Scaled dot-product attention. (right) Multi-head self-attention. Self-attention operates in parallel across multiples "heads". Each head has its own set of queries, keys, and values. The outputs are concatenated, and the linear transformation $\mathbf{W}^O$ is used to recombined them. (Figure from A. Vaswani et al. [2])

runs h parallel attention operations, or *heads*, each with its own set of learned projection parameters. The outputs of these attention heads are then concatenated and linearly transformed to form the final output. Multi-head attention enables the model to focus on different parts of the input sequence in multiple ways simultaneously, allowing it to capture a variety of relationships and patterns in the data more effectively, when compared to using a single attention function. Formally, the multi-head attention mechanism can be defined as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{2.4}$$

$$\text{where head}_i = \text{Attention}(Q\mathbf{W}_i^Q, K\mathbf{W}_i^K, V\mathbf{W}_i^V) \tag{2.5}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $\mathbf{W}^O \in \mathbb{R}^{H \cdot d_v \times d_{\text{model}}}$ are project parameter matrices, H is the number of heads, and d_k and d_v are the dimensionalities of the key and value matrices. Because each head operates on a reduced dimension, multi-head attention enhances the model's representational power without significantly increasing computational cost, keeping the total computation similar to single-head attention with full dimensionality.

2.1.5 Residual Connection and Normalization

In a transformer model, the inputs and output of the multi-headed self-attention module are connected using residual connections [17] and a layer normalization layer [18]. After the multi-headed self-attention process, the data goes through a position-wise feed-forward network, which also uses residual connections and layer normalization. This can be described by the formula:

$$\mathbf{X}' = \text{LayerNorm}(\text{MHA}(\mathbf{X})) + \mathbf{X} \tag{2.6}$$

$$\mathbf{X}'' = \text{LayerNorm}(\text{FFN}(\mathbf{X}')) + \mathbf{X}' \tag{2.7}$$

2.1.6 Position-wise Feed-Forward Networks (FFN)

After the attention mechanism, the output goes through a Position-wise Feed-Forward Network. Each token in the sequence is processed independently using the same network. The FFN consists of two linear transformations with a ReLU activation in between [24]:

$$\text{FFN}(x) = \max(0, x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2 \tag{2.8}$$

where x is the outputs of the previous layer, $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices, and b_1 and b_2 are biases. While the linear transformations are the same across different positions, they use different parameters from layer to layer. This adds non-linearity and allows the model to learn more complex functions.

2.1.7 Model Usage

Generally, Transformers can be used in three different ways. These variants are defined by the application of the attention and the connection of transformer blocks. Figure 2.6 provides an illustration of the attention masking patterns employed in these architectures.

1. **Encoder-Decoder:** This configuration uses the full Transformer architecture as introduced by Vaswani et al. [2] (see Figure 2.3). The encoder processes the input sequence, converting it into a series of vectors that carry the contextual information of the input tokens. Then, the decoder generates the target sequence one token at a time, basing each new prediction on the output from the encoder. Cross-attention layers, within the decoder, allow it to focus on relevant parts of the encoder's output, while causal masking ensures that predictions for a given token can only utilize information from previous tokens, not future ones. This is typically used in sequence-to-sequence modeling (e.g., neural machine translation).

2. **Encoder-Only:** As shown in Figure 2.4, in this configuration, only the encoder part of the Transformer architecture is utilized. The model processes the entire input sequence simultaneously, creating contextual embeddings for each token in the sequence. These embeddings encapsulate information about the token itself as well as its surrounding tokens, effectively capturing the meaning of the input in a non-sequential manner. This is normally used for tasks such as text classification or named entity recognition.

3. **Decoder-Only:** This configuration, depicted in Figure 2.5, uses only the decoder, and the encoder-decoder cross-attention module is also removed. The model predicts subsequent tokens, t_k, based on preceding context up to t_{k-1}, using a causal mask to ensure future

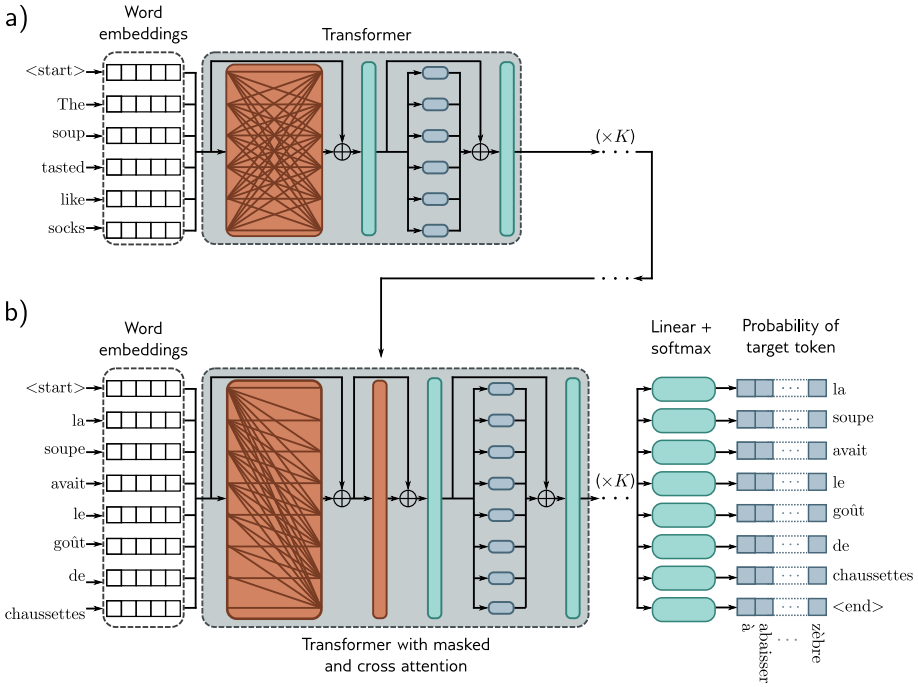

Figure 2.3: Encoder-decoder architecture for translation. An encoder processes the input sentence, while a decoder generates the output, using masked self-attention and cross-attention to encoder outputs. The system maximizes the probability of each subsequent output word. (Figure from Simon J. D. Prince [3])

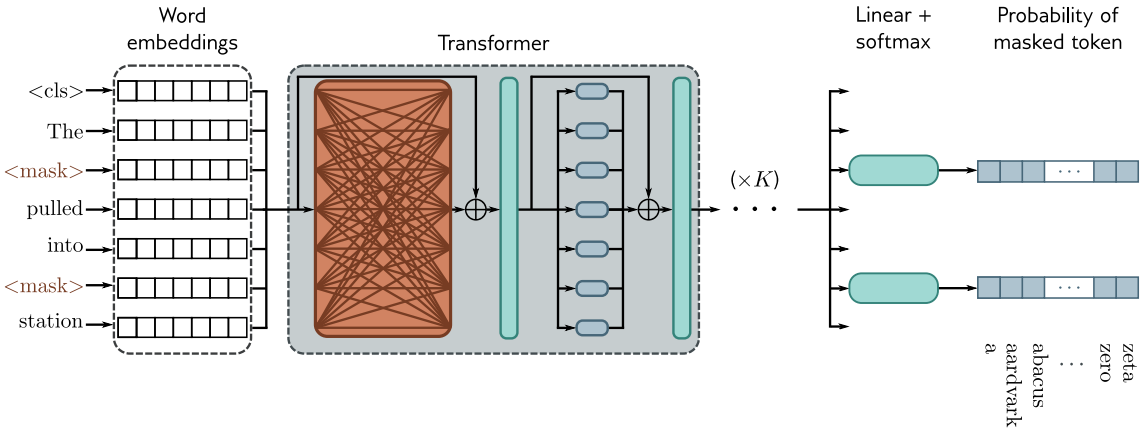

Figure 2.4: BERT-like encoder pre-training. Input tokens are embedded and processed through transformer layers with full attention. Some tokens are masked, and the model predicts these from surrounding context. This approach uses bidirectional context, but is less data-efficient. (Figure from Simon J. D. Prince [3])

tokens do not influence the predictions. Despite losing some input richness without an encoder, this simplified architecture aligns with traditional language modeling, with practical evidence showing its adequacy in LLMs [25]. This approach is the most commonly adopted in state-of-the-art LLMs, and is employed by widely acknowledged models like the GPT series [26, 5, 27] and other leading LLMs [9].

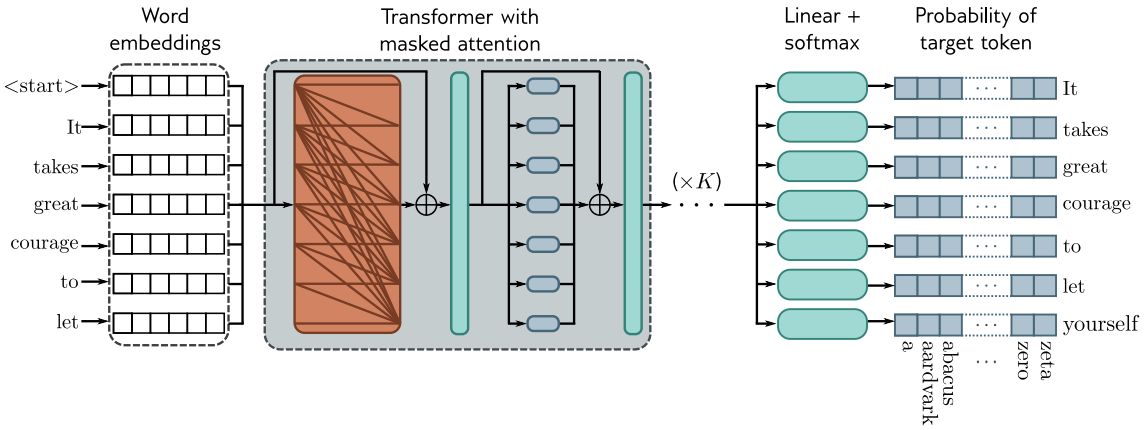

Figure 2.5: GPT3-style decoder network training. Tokens are embedded and processed through transformer layers with masked self-attention. The model aims to predict each next token in the sequence, using only previous context. This method efficiently uses data but limits context to preceding tokens. (Figure from Simon J. D. Prince [3])

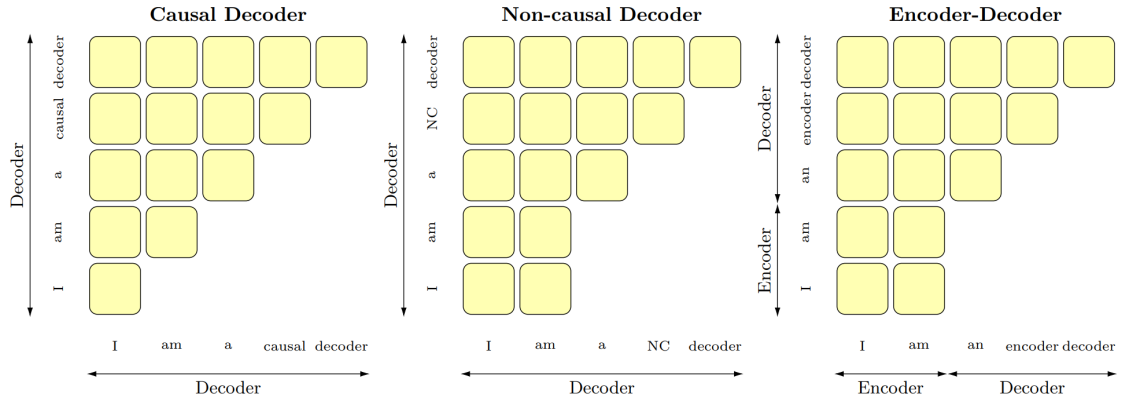

Figure 2.6: An example of attention patterns in language models. Causal decoders attend only to prior tokens, while non-causal and encoder-decoder architectures allow bidirectional attention to any conditioning information, with the encoder handling this in encoder-decoder models. (Figure from T. Wang et al. [4])

2.2 Tokenization and Embeddings

This section covers the process of tokenization and embeddings in Transformer-based language models. It explains how text is broken down into smaller units through character and subword modeling. It also describes how tokens are represented and positioned, in order for the model to make use of the order of the sequence.

2.2.1 Tokenization

LLMs are trained on text to predict text, and similar to other natural language processing systems, they use tokenization [28] as the essential preprocessing step. At this stage, the raw text is split into smaller basic units, called tokens. These tokens can be individual characters, subword units [29], symbols [30], or whole words, depending on the size and type of the model. The main goal of tokenization is to convert the text into a format that can be easily processed by computational models. Tokenization strategies have a major impact on model performance [31, 32, 33], since they affect both the model's ability to understand language nuances and the balance between vocabulary size and representation granularity.

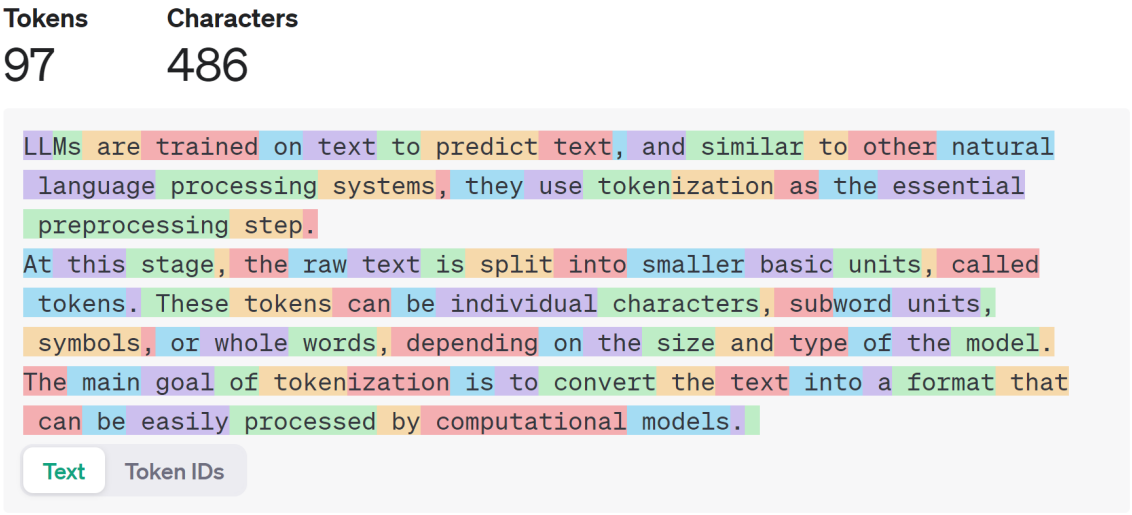

Figure 2.7: An example of text tokenization, illustrating how raw text is split into smaller basic units called tokens. This process is essential for preparing the text for computational models. The tokenizer shown is used for GPT-3.5 and GPT-4 models.

Character Modeling

Character-level models, represent inputs as character sequences. These methods have generally not been as widely employed for large language models, as the token sequences are much longer. Since each token in a transformer encoder model interacts with every other token, the computational complexity scales quadratically with the length of the sequence. For a decoder model, each token only interacts with previous tokens, so there are roughly half the number of interactions, but the complexity still scales quadratically, which introduces significantly higher costs for both training and inference [34, 35, 36].

Subword Modeling

Subword-based models tokenize their inputs into words and word pieces, most of which are longer than individual characters. Below are the most prominent subword tokenization methods:

[1] OpenAI, "OpenAI Tokenizer". `https://platform.openai.com/tokenizer` [Accessed July 2024]

1. **Byte Pair Encoding (BPE) [30]:** BPE is an incremental method where frequently occurring pairs of characters or bytes in the text are merged to form new tokens, efficiently representing common subword units in the corpus (see Figure 2.7).

2. **WordPiece [37]:** WordPiece optimizes the vocabulary by maximizing the probability of n-gram sequences, balancing vocabulary size and context coverage.

3. **Unigram Language Model [29, 38]:** This method progressively refines the vocabulary by discarding the least likely tokens, guided by token probabilities derived from an unigram language model.

In addition to these methods, other tokenization strategies exist, such as SentencePiece [39], which is a framework that allows the implementation of both BPE and Unigram Language Model tokenization without requiring language-specific preprocessing. These methods have become standard for large pre-trained language models [40, 27, 41].

2.2.2 Token Embedding

After being encoded, numerical tokens are transformed into unique embeddings, allowing the model to work in a continuous vector space instead of discrete token indices. This enables the model to capture semantic information and relationships between tokens.

To create the embedding matrix E, we initialize a matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $|\mathcal{V}|$ is the vocabulary size and d is the embedding dimension. Each row of this matrix corresponds to a unique token in the vocabulary, and each entry in a row represents a feature of that token in the embedding space. The embedding matrix E is learned like any other network parameter. Initially, the embedding matrix E is typically filled with small random values. During the training process, the embeddings are adjusted via backpropagation along with other model parameters. The objective is to learn embeddings such that tokens with similar contexts in the training data end up with similar representations in the embedding space. This is achieved through the loss function of the model, which encourages the embeddings of contextually similar tokens to be closer together.

For state-of-the-art models, a typical embedding size d is 8192, and a typical vocabulary size $|\mathcal{V}|$ is around 128.000, leading to a large number of parameters in E even before the main network.

2.2.3 Positional Encoding

Traditional Feed-Forward Networks (FFNs) and attention mechanisms in the Transformer architecture ignore the order of tokens in a sequence, meaning they treat each token independently of its position. This can be an issue when dealing with problems where input order matters, such as text modelling. To solve this, there needs to be a way to add positional information to the input embeddings. In the original Transformer [2], this is achieved by using absolute sinusoidal position encodings. These positional encodings are represented by a matrix $PE \in \mathbb{R}^{N \times d}$, where N is the maximum sequence length and d is the embedding dimension. Each row of PE represents

a unique position in the sequence and is added element-wise to the corresponding input token embedding. The PE matrix is constructed as follows:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{2.9}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{2.10}$$

where pos is the position and i is the dimension. Each dimension of the positional encoding corresponds to a sinusoid with wavelengths that form a geometric progression from 2π to $10000 \cdot 2\pi$. For $d = 128$ dimensions and for a maximum sequence length of $N = 50$, the positional encoding visualization is shown in Fig. 2.8. This allows the model to learn and generalize relative positions because any positional offset i can be linearly related to the original position's encoding.

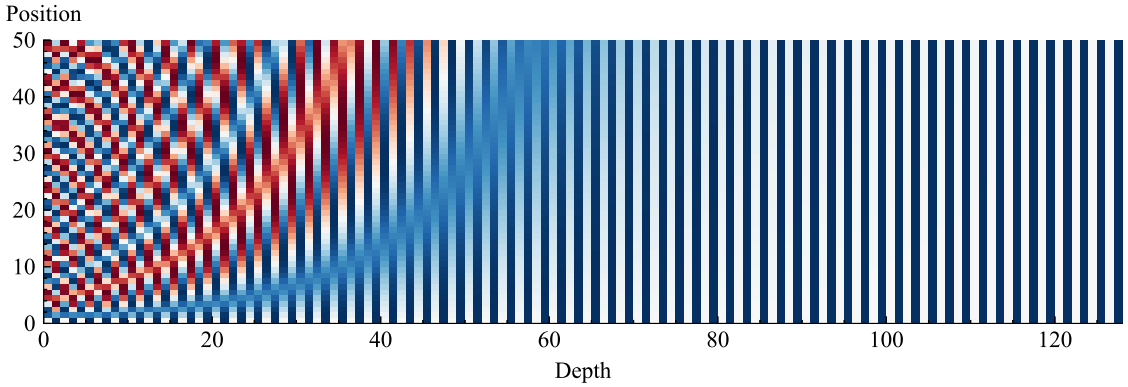

Figure 2.8: The 128-dimensional positional encoding for a sentence with the maximum length of 50. Each row represents a positional encoding vector at a specific position in the sequence, while each column corresponds to a dimension of the encoding.

As shown in Fig. 2.1, the two matrices, i.e., the token embeddings E and the positional encoding PE are added to generate the input representation:

$$\mathbf{X} = E + PE \in \mathbb{R}^{N \times d} \tag{2.11}$$

Another method for injecting positional information into the transformer is by learning a set of positional embeddings for each position. Both sinusoidal and learned positional encodings perform similarly in experiments, but the sinusoidal method is often preferred because it helps the model generalize to sequence lengths beyond those seen during training [2].

However, the number of embeddings is limited to a maximum sequence length determined before training, which means this approach cannot handle sequences longer than those seen during training.

Chapter 3

Related Work

The Transformer has made significant contributions to a wide range of artificial intelligence domains, including language understanding [42, 27, 40], computer vision [43, 44, 45], and audio processing [46, 47, 48]. Beyond these areas, Transformers have also been applied in fields like chemistry [49] and life sciences [50]. Given their extensive success, a substantial body of research has focused on improving the Transformer model [51, 52, 53]. This ongoing interest has led to the development of various Transformer-based architectures, commonly referred to as X-formers, over the past few years [54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]. Research into these X-formers primarily addresses three key areas [24]:

1. **Model Efficiency**. Applying Transformers to long sequences presents challenges due to the high computational and memory requirements of the self-attention mechanism. Efficiency improvements have been pursued through lightweight attention mechanisms, such as sparse attention, and strategies like recurrent and hierarchical architectures that break down tasks.

2. **Model Generalization**. Transformers can struggle to generalize when trained on smaller datasets, as they make fewer assumptions about the input data structure. Solutions to this issue involve incorporating structural biases, applying regularization techniques, and leveraging large-scale pre-training on unlabeled datasets.

3. **Model Applications**. This research path focuses on adapting the Transformer architecture for specific downstream tasks and applications.

This chapter provides a comprehensive review of the literature on architecture modifications. Given the central role of the attention mechanism in Transformers, Section 3.1 details the various attention-related variants. Section 3.2 introduces modifications at the module level, while Section 3.3 covers other architecture-level variants.

3.1 Attention

Despite its central role in the success of the Transformer, the self-attention mechanism introduces two primary challenges in practical applications:

1. **Complexity**. As outlined in Section 2.1, self-attention exhibits a complexity of $\mathcal{O}(N^2)$, scaling quadratically with the input sequence length N. This significant computational cost leads to inefficiencies, particularly when handling long sequences.

2. **Lack of Structural Bias**. Self-attention inherently lacks assumptions about the input data structure, requiring the model to learn the order of the data from scratch. This absence of built-in bias can contribute to overfitting, especially on smaller datasets.

Addressing these challenges, most improvements to the attention mechanism focus on two main directions [66]: (i) reducing the computational burden of the attention process, and (ii) enhancing the mechanism's learning efficiency. Recent approaches often combine or generalize these methods by compressing or selectively retaining context, though this can result in some loss of information [67, 68, 69, 70]. Techniques include context compression using summary or landmark tokens [71, 72] and semantic compression guided by importance [73, 74, 75, 76]. Other notable methods involve dynamically adjusting the number of tokens used in attention [77, 78, 79, 80, 56, 81].

3.1.1 Complexity Reduction

This line of work focuses on reducing the computational complexity by introducing sparsity bias into the attention mechanism. Sparse attention differs from standard self-attention by limiting the number of connections each element can attend to, rather than considering all possible pairs. In standard self-attention, each element in the sequence attends to every other element, leading to a quadratic complexity in relation to the sequence length. Sparse attention, on the other hand, reduces this complexity by allowing each output position to only compute weightings from a subset of input positions, such as a fixed number of neighbors or elements at regular intervals. This significantly lowers the computational burden, making the process more efficient while still capturing essential dependencies in the data.

The Sparse Transformer [82] addresses the quadratic complexity by employing sparse factorizations of the attention matrix. This approach enables the model to focus only on a subset of tokens, reducing the number of attended positions and thus the overall computation.

Longformer [56] introduces a combination of global and local attention patterns. This architecture maintains full attention on a few important tokens while using sparse attention for the remaining tokens, reducing computational requirements while effectively modeling long-range dependencies.

BigBird [83], in addition to sparse attention, also applies global attention as well as random attention to the input sequence. This hybrid approach approximates full attention, allowing the model to handle very long sequences with linear complexity.

Star-Transformer [84], arranges tokens in a star-shaped topology, where each token connects to a central relay node. This structure reduces the complexity from quadratic to linear, improving efficiency while preserving the ability to capture global context.

For vision tasks, Image Transformer [85] utilizes local attention mechanisms tailored for image data. By restricting attention to local patches within an image, the model achieves a significant

reduction in computational complexity while maintaining performance. Similarly, Axial Transformer [86] decomposes the attention mechanism along different axes of the input tensor. This method reduces the complexity by focusing attention along one axis at a time, allowing efficient processing of high-dimensional data such as images.

Reformer [54] introduces locality-sensitive hashing to approximate the attention mechanism. This technique reduces the quadratic complexity to logarithmic, making it feasible to handle very long sequences without significant computational overhead.

Lastly, Routing Transformer [55] employs k-means clustering to dynamically route tokens to different clusters. This strategy ensures that each token only attends to a subset of tokens within the same cluster, reducing the complexity and enhancing the efficiency of the attention mechanism.

3.1.2 Linearized Attention

The goal of linearized attention methods is to untangle the attention matrix with kernel feature maps. This approach allows the attention computation to be performed in reverse order, resulting in linear complexity with respect to the sequence length.

Linear Transformer [57] achieves this by expressing the self-attention as a linear dot-product of kernel feature maps. This approach reduces the complexity from quadratic to linear with respect to the sequence length, enabling more efficient processing of long sequences.

Similarly, Performer [87, 62] uses a kernel-based approximation to the attention mechanism. This method is implemented by a FAVOR+ algorithm, which provides scalable low-variance and unbiased estimation of attention mechanisms that can be expressed by random feature map decompositions that approximate the scoring function of Transformers.

3.1.3 Low-Rank Attention

Low-rank attention methods reduce the complexity of the attention mechanism by approximating the attention matrix with a low-rank decomposition. There is empirical as well as theoretical evidence [88, 59] that the self-attention matrix $\mathbf{A} \in \mathbb{R}^{T \times T}$ is often low-rank, i.e., the rank of $\mathbf{A}$ is far lower than input length T. This characteristic leads to two significant consequences. Firstly, explicit parameterization could be employed to model the low-rank attribute. Secondly, a low-rank approximation might serve as a substitute for the self-attention matrix.

Guo et al. [88] leverage the low-rank structure of the attention matrix to perform efficient matrix multiplications, thereby reducing the overall computational load. Nyströmformer [89] extends this idea by using the Nyström method to approximate the standard self-attention, allowing the model to achieve linear complexity.

3.1.4 Query Prototyping

Query prototyping methods focus on reducing the number of queries that need to be processed during the attention computation. Rather than employing sparse attention or kernel-based linearized

attention methods, another approach for reducing the complexity of the attention mechanism is to reduce the number of queries, or key-value pairs, that need to be processed.

Clustered Attention [65] groups queries into several clusters and then computes attention distributions for cluster centroids. All queries in a cluster share the attention distribution calculated with the corresponding centroid.

Informer [90] selects prototypes from the most informative queries based on their sparsity measurements.

Set Transformer [91] uses a pooling operation to create prototypes from sets of queries, allowing for efficient attention computation on sets of varying sizes. This reduces the quadratic complexity of self-attention to linear complexity with respect to sequence length.

Linformer [59] approximates the self-attention mechanism by projecting the input sequence into a lower-dimensional space, thus reducing the complexity from quadratic to linear.

3.1.5 Attention with Prior

Attention with prior methods incorporate prior knowledge or structural bias into the attention mechanism. In the attention mechanism, the expected attended value, represented by the attention matrix (before softmax), is normally derived from the input sequence. However, this attention distribution can also be supplemented or even replaced by other sources, which are often referred to as *prior*. Usually, the fusion of the two attention distributions is achieved by calculating a weighted sum of the scores from the prior and the generated attention before applying softmax.

Yang et al. [92] proposes Local Transformer, which restricts attention to local neighborhoods, reducing complexity and improving the model's ability to capture local dependencies.

Gaussian Transformer [93] introduces a Gaussian bias to the attention mechanism, allowing it to focus more on nearby tokens.

He et al. proposes RealFormer [94], a technique to create Residual Attention Layer Transformer networks. It works by reusing attention scores, which allows it to maintain high performance while reducing computational overhead.

LazyFormer [95] uses a lazy update mechanism to reduce the frequency of attention computations, thereby improving efficiency. It consists of multiple lazy blocks, each containing several Transformer layers. Within each block, the self-attention distribution is computed only once in the first layer and reused in the subsequent layers. This approach significantly lowers computational cost while maintaining performance, making it effective for long-sequence tasks.

Synthesizer [96] proposes to replace generated attention scores with synthetic attention, which is generated through learnable weights or random projections. Experiments on machine translation and language modeling show that these variants can match the performance of the vanilla Transformer. Notably, the reason why these variants work is not explained, but the empirical results are intriguing.

3.1.6 Multi-head Attention

This area of study focuses on exploring different alternatives to the standard multi-head attention mechanism. Multi-head attention is valued for its ability to simultaneously focus on information from different parts of the input. However, there is no guarantee that each attention head captures different features.

Sukhbaatar et al. [64] proposes using a learnable attention span. This model dynamically adjusts the attention span of each head based on the input sequence. Experiments on character-level language modeling show that the adaptive-span models outperform baseline models, while having considerably fewer FLOPS.

On the other hand, Multi-scale Transformer [97] proposes to use a fixed attention span. It introduces attention heads that operate at different scales, allowing the model to capture both local and global dependencies more effectively. Experiments on several tasks show that the model can outperform baseline models while accelerating inference on long sequences.

3.2 Other Module-Level Modifications

3.2.1 Positional Representations

Innovations in positional encoding have progressed significantly beyond the original absolute positional encodings introduced by Vaswani et al. [2]. Approaches such as relative positional encodings [98], which provide explicit information about token distances, and Rotary Positional Encoding (RoPE) [99], which represents absolute positions through rotations, offer alternative methods for capturing token relationships. Additionally, techniques like ALiBi [100], which introduce adaptive biases for relative positions in attention mechanisms, further improve the model's ability to comprehend contextual relationships.

3.3 Architecture-level Modifications

To date, efforts to modify the vanilla Transformer architecture have largely fallen into five categories [24]: reducing memory footprint and computation, adding connections between transformer blocks, enabling adaptive computation time (such as early stopping during training), introducing recurrence or hierarchical structures, and exploring more drastic architectural changes (e.g., neural architecture search). These modifications have been designed to address both efficiency and interpretability challenges.

He et al. [15] simplified the standard transformer block by eliminating components like residual connections and layer normalization, and restructuring the MLP sub-blocks. Zhao et al. [101, 102] introduced a parallel block architecture that allows simultaneous computation of MLP and attention sub-blocks, significantly improving efficiency with minimal performance loss. In another approach, Trockman et al. [103] observed that trained transformers maintain a strong identity component in the value and projection matrices, leading to an initialization method that

reduces architectural complexity without sacrificing performance. Similarly, there have been attempts to reduce the frequency of MLP sub-blocks for greater efficiency [104, 105], as well as efforts to explore more efficient alternatives to softmax attention [57, 106, 62]. Furthermore, Sukhbaatar et al. [107] proposed integrating the MLP directly into the attention mechanism using persistent memory, removing the need for a separate MLP block.

Numerous lightweight transformer models have been developed, including the Lite Transformer [108], Funnel Transformer [109], and DeLight [110]. Modifications aimed at enhancing cross-block connectivity include the Realformer [94] and Transparent Attention [111].

Adaptive computation time is another area of interest, with notable models such as the Universal Transformer [51], Conditional Computation Transformer [112], and DeeBERT [113] exploring dynamic computation approaches.

Recurrence has also been integrated into transformer architectures, with models like Transformer-XL [114], Compressive Transformer [61], and Memformer [115], while hierarchical structures are represented by HIBERT [116] and Hi-Transformer [117].

Finally, more radical architectural changes have been seen in models like the Macaron Transformer [118], Sandwich Transformer [119], and Differentiable Architecture Search [120], which have extended transformer efficiency and performance.

3.4 Interpretability of Attention

To date, several studies have investigated the role of self-attention within Transformer architectures. Clark et al. [121] and Kovaleva et al. [122] have offered insights into patterns of attention. Additionally, Voita et al. [123] and Brunner et al. [124] investigated redundancy in attention heads and the role they play in capturing linguistic features. In contrast, Jain and Wallace [125] questioned the explanatory value of attention weights, arguing that they may not consistently reflect feature importance. Wiegreffe et al. [126] furthered this debate by proposing specific tests to assess the utility of attention mechanisms as explanations. Despite varying perspectives, attention mechanisms remain central to efforts in enhancing model interpretability and performance. For instance, Vig et al. [127] developed tools for visualizing the behavior of individual attention heads, and Rogers et al. [128] conducted a comprehensive survey on BERT models, including attention analysis discussions.

3.5 Conclusion and Future Directions

Current advancements in Transformer research target multiple aspects, including efficiency, generalization, and domain-specific applications. Significant progress has been made by incorporating structural priors, developing lightweight architectures, and enhancing pre-training techniques.

However, despite their widespread effectiveness, X-formers continue to encounter certain limitations. In addition to ongoing challenges in efficiency and generalization, future areas for development include:

3.5.1 Theoretical Understanding of Transformers

Transformers have demonstrated effectiveness in processing large-scale training datasets, frequently surpassing CNNs and RNNs in performance. This advantage is attributed to their minimal assumptions regarding data structure, rendering them more adaptable than CNNs and RNNs. Despite these empirical successes, the theoretical underpinnings of this flexibility are not yet fully understood, highlighting the need for further theoretical investigation into Transformer capabilities.

3.5.2 Global Interaction Mechanisms

Transformers exhibit a notable capability in modeling global dependencies through attention mechanisms. However, recent studies suggest that full attention may not be required for all nodes, as its indiscriminate application can lead to inefficiencies. Opportunities for improving the modeling of global interactions exist, particularly by reinterpreting the self-attention module as a fully connected neural network with adaptable connection weights, enabling dynamic aggregation of non-local information. Dynamic routing strategies and memory-augmented models also represent viable alternatives for optimizing these processes.

3.5.3 Multimodal Data

Integrating multimodal data often leads to improved performance across various tasks. To achieve robust AI systems, it is essential to capture the semantic relationships between different modalities. Transformers have shown considerable success in handling text, images, video, and audio, presenting an opportunity to design a unified framework that more effectively models the interconnections across multimodal data.

Chapter 4

Methods

Building on the motivation to explore alternatives to the standard self-attention mechanism, and grounded in the research landscape discussed in Chapter 3, this chapter outlines the proposed modifications to the self-attention mechanism.

Excluding optimizations of the original code, the primary strategies in the literature for improving self-attention focus on two main objectives: (i) reducing the computational complexity, ideally leading to faster training times without significant degradation in performance, or (ii) enhancing generalization capabilities with minimal or no increase in training time. In this study, both strategies were explored, leading to three distinct core modifications, along with two additional variations derived from combinations of the core implementations.

4.1 Reducing Complexity

4.1.1 Simple Self-Attention (SSA)

As discussed in Section 2.1.1, the self-attention formula is expressed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{4.1}$$

Focusing on the numerator of the softmax function, $QK^\top$, the matrices Q and K are the queries and keys, respectively. For each attention head, they result from the linear transformation of the input matrix $\mathbf{X}$ by the weight matrices $\mathbf{W}_Q$ and $\mathbf{W}_K$, as follows:

$$Q = \mathbf{X} \cdot \mathbf{W}_Q^\top \quad , \quad K = \mathbf{X} \cdot \mathbf{W}_K^\top \tag{4.2}$$

where $\mathbf{W}_Q^\top \in \mathbb{R}^{d_{model} \times hs}$ and $\mathbf{W}_K^\top \in \mathbb{R}^{hs \times d_{model}}$, with d_{model} representing the embedding size and hs the head size. The numerator of the softmax function can be rewritten as:

$$Q \cdot K^\top = \mathbf{X} \cdot \mathbf{W}_Q^\top \cdot (\mathbf{X} \cdot \mathbf{W}_K^\top)^\top, \tag{4.3}$$

which simplifies, using matrix transpose properties, to:

$$Q \cdot K^\top = \mathbf{X} \cdot \left(\mathbf{W}_Q^\top \cdot \mathbf{W}_K\right) \cdot \mathbf{X}^\top . \tag{4.4}$$

Taking advantage of this procedure, it is possible to define a *Simple Self Attention* (SSA) mechanism, which despite being simpler (with fewer matrix operations), it in fact should have at least the same explanation power than Self-Attention, as it is a generalization of the basic procedure. The complete SSA operation can thus be expressed as follows:

$$U = \mathbf{X} \cdot \mathbf{W}_U \cdot \mathbf{X}^\top \tag{4.5}$$

$$V = \mathbf{X} \cdot \mathbf{W}_V \tag{4.6}$$

$$\mathrm{SSA}(U, V) = \mathrm{softmax}\left(\frac{U}{\sqrt{d_k}}\right) V \tag{4.7}$$

This maintains, in all other respects, the original self-attention formulation. For each attention head, the input matrix $\mathbf{X}$ undergoes two parallel transformations: one through the unified weight matrix $\mathbf{W}_U$ to compute attention scores, and another through the value weights $\mathbf{W}_V$. Attention scores are scaled by $\sqrt{d_k}$ and processed using a softmax function before being applied to the transformed values. The final output is obtained through a projection layer that combines the outputs from all attention heads, maintaining compatibility with subsequent network layers.

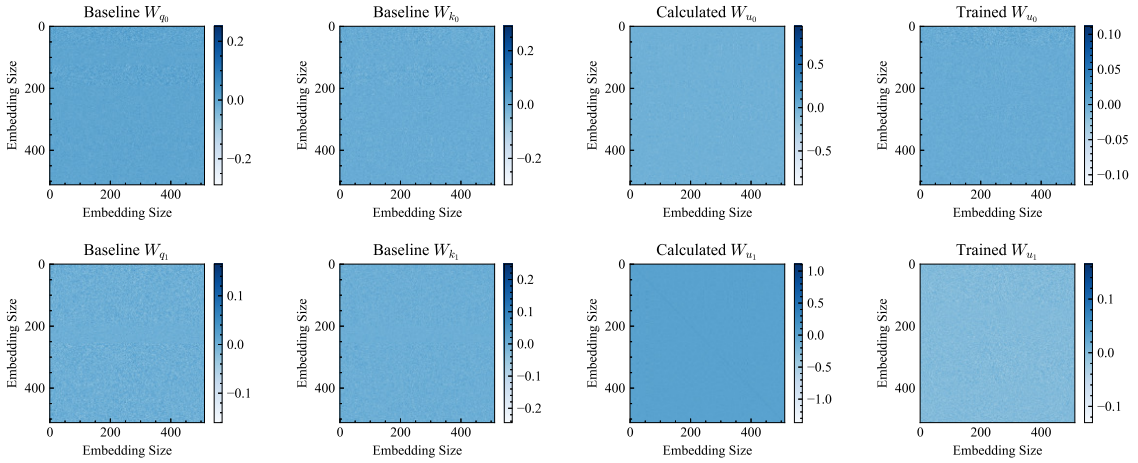

Figure 4.1: Comparison of weight matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_U$ from the Baseline and SSA models for heads 0 and 1. The figures show matrices for the query weights $\mathbf{W}_{Q_0}$ and $\mathbf{W}_{Q_1}$, key weights $\mathbf{W}_{K_0}$ and $\mathbf{W}_{K_1}$, calculated attention weights $\mathbf{W}_{U_0}$ and $\mathbf{W}_{U_1}$ derived from the Baseline model, and directly trained attention weights from the SSA model.

4.2 Improving Generalization

4.2.1 Layered Self-Attention (LSA)

As shown in Figure 4.3, the `LSA` mechanism computes the attention matrix in the same way as the original scaled dot-product attention but introduces two additional linear layers, `l1` and `l2`, which

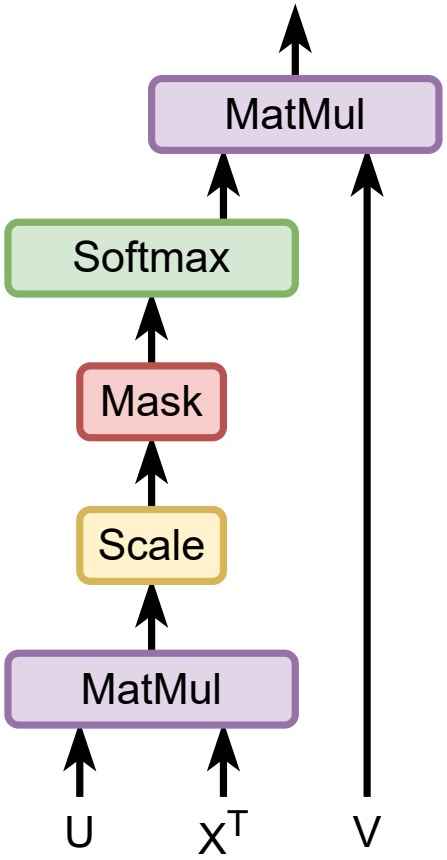

Figure 4.2: Diagram representing the Simple Self-Attention model.

operate on the value matrix V. The `l1` layer performs a linear transformation on V and adds the result back to V, functioning similarly to a residual connection.

The `l2` layer also applies a linear transformation, but its output is passed through a sigmoid function before being multiplied by V. This serves as a gating mechanism, regulating the propagation of elements within V. Values close to 0 are effectively "turned off", while those nearing 1 are more freely allowed to pass through. This modification results in a 0.20% increase in the number of parameters compared to the baseline scaled dot-product attention.

4.2.2 Variable Self-Attention (VSA)

This attention mechanism closely resembles the original but incorporates a variable k, which adjusts the dimensions of the queries and keys. In the computation of the attention matrix, the inner dimensions of $\mathbf{W}_Q$ and $\mathbf{W}_K$ must align, though this dimension can be varied. The original mechanism employs square matrices for $\mathbf{W}_Q$ and $\mathbf{W}_K$, effectively setting k to 1. By setting k to 2, the width of the $\mathbf{W}_Q$ and $\mathbf{W}_K$ matrices is doubled, while setting k to 3 triples the size, and so forth.

Aside from this modification, the rest of the mechanism remains unchanged, with the only adjustment being the reshaping of the queries and keys to accommodate the new matrix dimensions.

When $k = 2$, this results in a 50% increase in the number of parameters compared to the

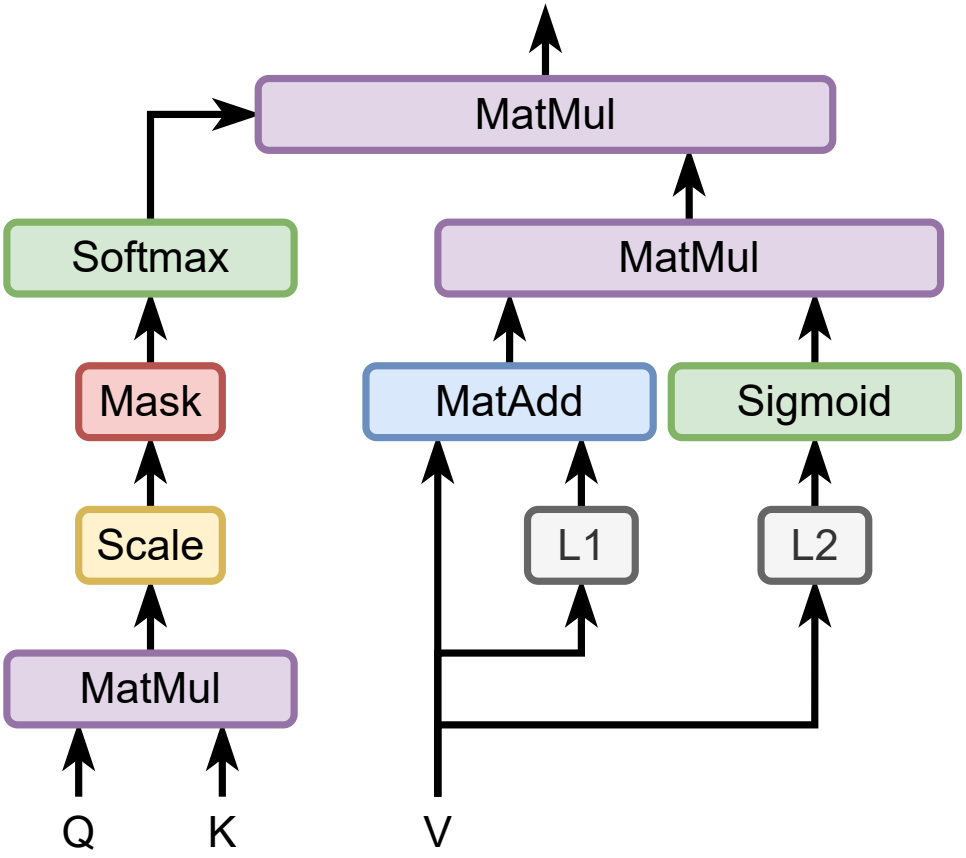

Figure 4.3: Diagram representing the Layered Self-Attention model.

baseline.

4.3 Extra Variants

4.3.1 Simple Layered Self-Attention (SLSA)

The Simple Layered Self-Attention (SLSA) mechanism is a combination of the Simple Self-Attention (SSA) mechanism from 4.1.1 and the Layered Self-Attention (LSA) from 4.2.1 (see Figure 4.4).

4.3.2 Variable Layered Self-Attention (VLSA)

The Variable Layered Self-Attention (VLSA) mechanism integrates features from both 4.2.1 and 4.2.2. Shown in 4.5, it incorporates the layers `l1` and `l2` from 4.2.1 and the variable k from 4.2.2, with k treated as a tunable parameter rather than a fixed value, with three configurations tested: $k = \{1, 2, 3\}$.

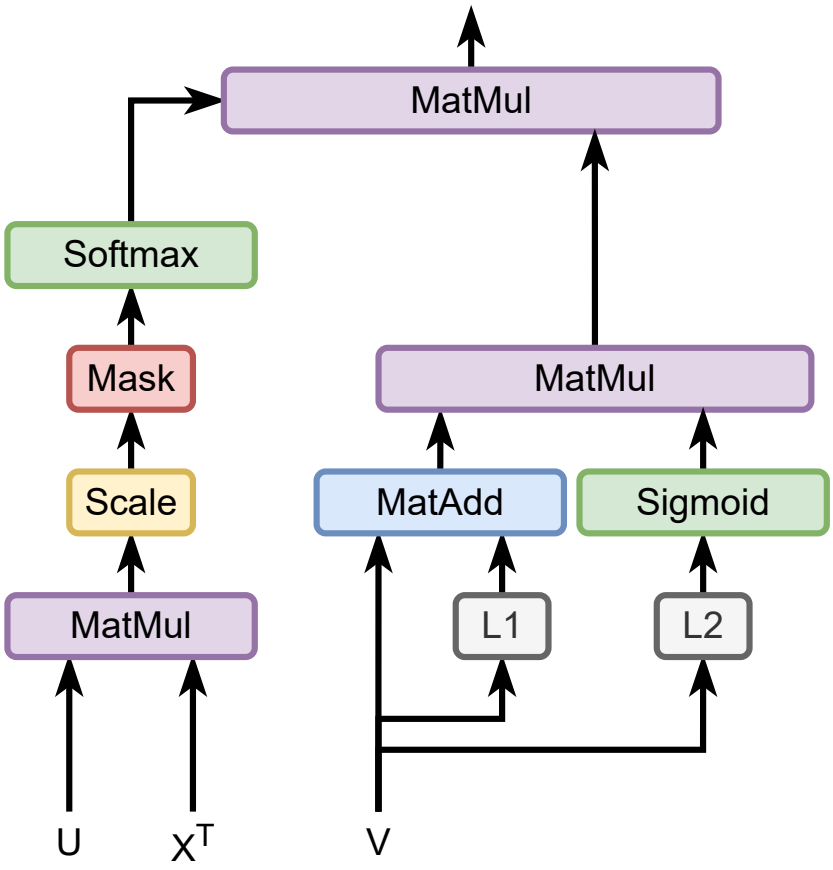

Figure 4.4: Diagram representing the Simple Layered Self-Attention model.

4.4 Evaluation Metrics

This section describes the performance metrics and statistical tests used to evaluate the models.

4.4.1 Performance Metrics

1. **Change From Baseline (CFB)**: The change from the Baseline is defined as the percentage difference between the model's performance metric, M_{model}, and the Baseline performance metric, M_{baseline}, expressed as:

$$\text{CFB } (\%) = \left(1 - \frac{M_{\text{model}}}{M_{\text{baseline}}}\right) \times 100 \tag{4.8}$$

4.4.2 Statistical Tests

To assess the statistical significance of the differences observed among the models, several statistical tests were conducted:

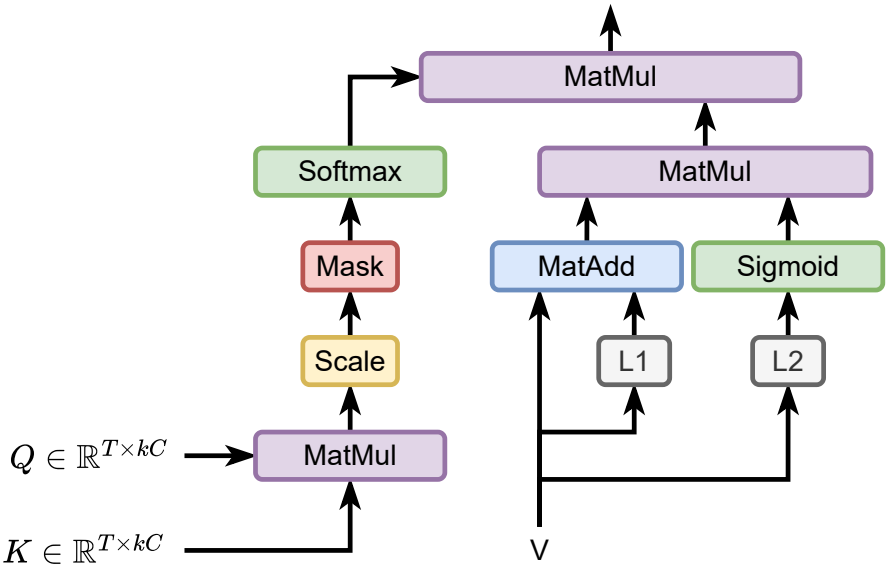

Figure 4.5: Diagram representing the Variable Simple Self-Attention model.

Analysis of Variance (ANOVA)

ANOVA tests whether there are statistically significant differences in the mean validation loss across different models by comparing the variation between group means to the variation within groups. If the between-group variation is significantly larger than the within-group variation, it suggests that the groups are different. The test calculates a ratio called the F-statistic, which is the between-group variability divided by the within-group variability. A large F-statistic suggests significant differences between groups. The p-value associated with this F-statistic represents the probability of obtaining such results if the null hypothesis (all group means are equal) is true. A small p-value (typically < 0.05) indicates statistical significance.

Friedman Test

Given that the assumptions of ANOVA may not hold due to potential non-normality and heteroscedasticity, a non-parametric Friedman test was also conducted. The Friedman test is suitable for comparing more than two related groups and does not assume normality of the data. It works by ranking the performance of each model within each dataset and then comparing the average ranks across models. The test statistic follows a chi-square distribution, with a low p-value indicating significant differences among the groups.

Post-hoc Analysis with Tukey's HSD

To identify which specific models differ from each other, Tukey's Honestly Significant Difference (HSD) test compares all possible pairs of means while controlling for the family-wise error rate. This test calculates a single threshold value for the difference between means that must be

exceeded for any pair to be considered significantly different. It identifies which specific groups differ after a significant ANOVA.

Wilcoxon Signed-Rank Test

The Wilcoxon signed-rank test compares two related samples. It's a non-parametric test that does not assume normality of the data and works by ranking the absolute differences between paired observations and comparing the sum of ranks for positive and negative differences. The test statistic W is the smaller of the sums of positive and negative ranks. A small W value leads to rejection of the null hypothesis that the two samples come from the same distribution.

Chapter 5

Data Acquisition and Preparation

This section outlines the datasets and preprocessing techniques employed for training the models.

5.1 Datasets

For this work, various datasets with unique linguistic features were used to test the models' performance. Each dataset is described below, and their properties are summarized in Table 5.1.

Dataset	Vocab Size	Tokens (M)	Size	Type	Ref.
TinyShakespeare	65	1.12	Small	Poetic	[129]
Cultural Corpus	68	3.65	Medium	Prose	[130]
Spelling Data	93	6.49	Medium	Prose	[131]
Gutenberg	340	165.02	Large	Prose	[132]
Chess Games	1973	790.90	Large	Game Moves	[133]
Random Chess Games	1973	792.00	Large	Game Moves	N/A

Table 5.1: Summary of Dataset Properties

5.1.1 TinyShakespeare

The TinyShakespeare corpus [129] is a dataset that contains 40,000 lines of Shakespeare from a variety of his plays. The Tiny Shakespeare corpus is a popular choice for training language models due to its manageable size and the complexity of Shakespeare's language. It provides a good balance between computational efficiency and the ability to generate interesting text. Featured in Andrej Karpathy's blog post '*The Unreasonable Effectiveness of Recurrent Neural Networks*'. This dataset contains 1,115,394 tokens and 65 unique tokens.

5.1.2 Cultural Corpus

Referred to as "*pbooks*", this dataset is a collection of anthropological literature from Project Gutenberg. It includes academic-style writings, contrasting with the poetic and dramatic texts of Shakespeare. The collection comprises the following four books:

- *The Golden Bough: A Study of Magic and Religion* by James George Frazer [130]
- *Myth, Ritual and Religion, Vol. 1 (of 2)* by Andrew Lang [134]
- *The Religion of the Ancient Celts* by J. A. MacCulloch [135]
- *Religion and Art in Ancient Greece* by Ernest Arthur Gardner [136]

It has 3,652,868 tokens and 68 unique tokens, offering a rich and diverse source for analyzing the model's performance on informational and analytical text.

5.1.3 Spelling Data

This dataset, titled "big"[1], comes from the UMBC CMSC 331-2 Spring 2010, Principles of Programming Languages course [131]. It was initially designed for Peter Norvig's Spelling Corrector, a Python program intended to suggest corrections for misspelled words. The data consists of a small collection of books from the Project Gutenberg Corpus. This dataset contains 6,488,666 tokens and 93 unique tokens, providing another source for evaluating the model's performance on diverse textual data.

5.1.4 Gutenberg

To create this dataset, the first 100 largest books from a collection were used. This collection is derived from a small subset of the Project Gutenberg corpus, consisting of 3,036 English books by 142 authors [132]. The data has been cleaned to remove metadata, license information, and transcribers' notes. This dataset comprises 165,022,139 tokens and 340 unique tokens.

5.1.5 Chess Games and Random Chess Games

The *Chess Games* dataset consisted of approximately 2.2 million chess games played since 1990, with player ELO ratings above 2000 [133]. Each game included up to 360 moves, resulting in a total of 790,899,120 tokens and 1,973 unique tokens. The *Random Chess Games* dataset was generated using a Python script and consisted of exactly 2.2 million randomly generated, valid chess games. This dataset contained 792,000,000 tokens and 1,973 unique tokens.

In both datasets, each game began with the symbol `BEG` (beginning of the game), followed by the moves in algebraic notation, and concluded with the game's result (e.g., `1-0`, `0-1`, `1/2-1/2`). If a game was shorter than 360 moves, the remaining tokens were filled with the symbol `GMO` (game over). The games were tokenized such that each move constituted a token. The vocabulary size (unique tokens) for both datasets was 1,973, derived from a pickle file, which mapped chess moves in algebraic notation to integers. This dictionary was used to create the encoder and decoder functions necessary for tokenizing the games.

[1] The name comes from the original source's filename and doesn't reflect the dataset's actual size.

5.2 Preprocessing

Prior to training, each dataset underwent several preprocessing steps:

1. The vocabulary size was determined by counting the unique characters in the dataset.
2. Each character was mapped to a unique integer using two dictionaries: one for character-to-integer mapping and another for integer-to-character mapping.
3. This mapping was utilized to encode the text data into sequences of integers.
4. Each dataset was divided into training and validation sets, with 90% allocated for training and 10% for validation.
5. Both the training and validation sets were encoded into integer sequences.

Chapter 6

Implementation

6.1 Exploratory Work

The research process began with the utilization of Andrej Karpathy's ng-video-lecture[1] as a foundation for designing multiple alternatives to the self-attention mechanism. A trial-and-error approach was employed on a small scale to evaluate the performance of these alternatives across various datasets, with the objective of identifying the most promising modifications. This initial phase culminated in the selection of three top-performing alternatives: the LSA, SLSA, and VSA mechanisms.

Subsequently, the research transitioned to the nanoGPT repository[2]. While more complex in its handling of the training process and data loading, this repository offered significantly enhanced efficiency compared to the previous implementation. The LSA, SLSA, and VSA mechanisms were then implemented within this codebase and subjected to comprehensive testing alongside the original self-attention mechanism across all collected datasets.

A Python script was developed to automate the creation of configuration files necessary for training the models. As outlined in Chapter 5, the datasets were categorized into small, medium, and large, with specific datasets assigned to each category (see Table 5.1). For each dataset size, distinct block sizes and configuration tuples were defined. The script generated configuration files by populating a template with specific values. These values included a consistent batch size of 64, 40000 iterations, and a dropout value of 0.2 across all configurations. The block size varied based on the dataset, while the number of layers, number of heads, embedding size, and learning rate were adjusted according to each specific configuration. Table 6.1 provides a summary of the different configurations. In total, 18 configuration files were generated, encompassing various datasets and hyperparameter combinations, totaling 72 training runs. This approach facilitated the training of models with diverse configurations.

To streamline the training process, a shell script was created to execute all configuration files sequentially, eliminating the need for manual intervention during the training of multiple models.

[1] Andrej Karpathy, "ng-video-lecture," GitHub repository, 2023, available at: `https://github.com/karpathy/ng-video-lecture` [Accessed September 2023]

[2] Andrej Karpathy, "NanoGPT: GPT-like training code in PyTorch that's small enough to read," 2023, available at: `https://github.com/karpathy/nanogpt`[Accessed October 2023]

Table 6.1: Hyperparameter configurations for initial experimentation.

For a given dataset, to get a hyperparameter configuration, start with the "Block Size" value, then choose the three values either above or below the shorter horizontal line for "Embedding Size", "Layers", and "Heads", and finally select a "Learning Rate". The horizontal lines separate values that were not combined.

Dataset Size	Dataset	Block Size	Embedding Size	Layers	Heads	Learning Rate
Small	shakespeare	128	128	2	2	1e-3
			256	4	4	3e-4
Medium	big	256	256	4	4	1e-3
	pbooks		384	6	6	3e-4
Big	gutenberg	256	384	6	6	3e-4
	chess		512	8	8	

6.2 Confirmatory Analysis

The confirmatory analysis phase built upon the exploratory work, focusing on refining and validating the most promising attention mechanisms identified previously. This process involved returning to Andrej Karpathy's ng-video-lecture codebase, where the focus was on fine-tuning the highest-performing variants and conducting more thorough evaluations.

Five modified attention mechanisms were evaluated during this phase: SSA and VLSA with k values ranging from 1 to 3, alongside the baseline scaled dot-product attention. To ensure consistency and comparability, all models were trained exclusively on the Gutenberg dataset, the largest available corpus. A batch size of 64 was maintained, ensuring the parallel processing of 64 independent sequences, and the block size was set to 254, reflecting the maximum context length for predictions.

Each model underwent two distinct iterations of training: three trials for 40,000 iterations and another three trials for 80,000 iterations, with evaluations performed every 500 iterations. The learning rate was fixed at 1×10^{-3}, and performance evaluations were conducted over 50 iterations. The models adhered to a consistent architecture featuring 4 attention heads and 4 layers, with an embedding size of 256, computed as the product of the number of heads (4) and the head size (64)[3]. Additionally, a dropout rate of 0.1 was applied to mitigate overfitting during training.

[3]The head size is calculated as d divided by the number of heads.

Chapter 7

Results and Discussion

7.1 Exploratory Analysis

This section presents the results of the exploratory work, where the models were evaluated across different datasets and hyperparameter configurations.

7.1.1 Performance Metrics

Table 7.1 summarizes the performance metrics for each model, and Figures 7.1 and 7.2 illustrate the mean validation loss and the distribution of the minimum validation loss for each model, respectively. The Change From Baseline (CFB) metric (see Section 4.4) is used to quantify improvements over the Baseline model.

Table 7.1: Combined results and percentage differences for models trained for 40,000 iterations.

Head Type	Avg Min Validation Loss		Avg Train Time (hh:mm:ss)	
	Value	CFB (%)	Value	CFB (%)
Baseline	1.4426	(– %)	00:08:55	(– %)
LSA	1.4364	(+0.43 %)	00:09:42	(8.79 %)
SLSA	1.4285	(+0.98 %)	00:14:03	(57.57 %)
VSA	1.4424	(+0.01 %)	00:09:43	(8.97 %)

7.1.2 Statistical Analysis

To assess the statistical significance of the observed differences among the models, several statistical tests were conducted (see Section 4.4 for details on the statistical methods).

Analysis of Variance (ANOVA)

An ANOVA test was performed to determine if there are any statistically significant differences in the mean validation loss across the different models. ANOVA works by comparing the variation

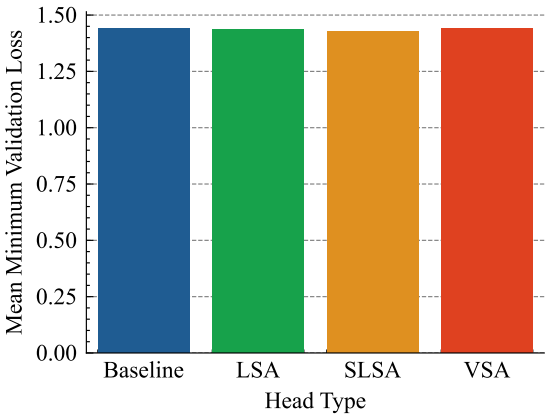

Figure 7.1: Mean validation loss across models.

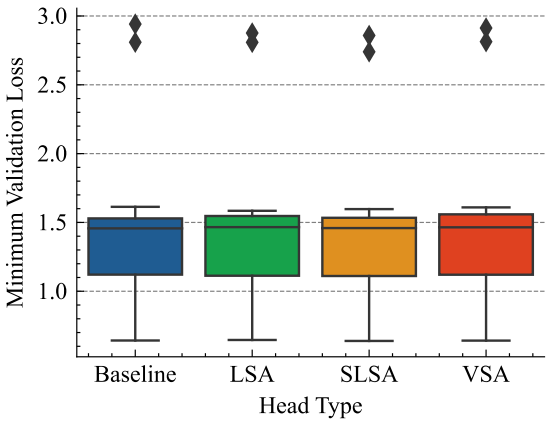

Figure 7.2: Box plot of minimum validation loss by model.

between group means to the variation within groups. If the between-group variation is significantly larger than the within-group variation, it suggests that the groups are different.

The test calculates a ratio called the F-statistic, which is the between-group variability divided by the within-group variability. A large F-statistic suggests significant differences between groups. The p-value associated with this F-statistic represents the probability of obtaining such results if the null hypothesis (all group means are equal) is true. A small p-value (typically < 0.05) indicates statistical significance.

The results of the ANOVA test are presented in Table 7.2. The ANOVA results indicate that there is no statistically significant difference in the mean validation loss across the models ($F(3, 68) = 0.0023$, $p = 0.9998$). The very small F-statistic and large p-value suggest that the observed variations in validation loss across models are likely due to random chance rather than true differences in model effectiveness.

Table 7.2: ANOVA Test Results

Source	Sum of Squares	Degrees of Freedom	F-Statistic	p-value
Model	0.002386	3	0.002304	0.999846
Residual	23.479393	68	–	–

Friedman Test

The Friedman test indicates a statistically significant difference among the models ($\chi^2 = 11.2667$, $p = 0.0104$). This suggests that at least one model performs differently from the others, contradicting the ANOVA results and highlighting the importance of using multiple statistical approaches.

Post-hoc Analysis with Tukey's HSD

The Tukey HSD post-hoc test was performed to identify which specific models differ from each other. The results are summarized in Table 7.3. None of the pairwise comparisons between models

are statistically significant ($p > 0.05$).

Table 7.3: Tukey HSD Post-hoc Test Results

Group 1	Group 2	Mean Difference	p-value	Lower	Upper
Baseline	LSA	-0.0062	1.0000	-0.5221	0.5097
Baseline	SLSA	-0.0141	0.9999	-0.5300	0.5018
Baseline	VSA	-0.0002	1.0000	-0.5160	0.5157
LSA	SLSA	-0.0079	1.0000	-0.5238	0.5080
LSA	VSA	0.0060	1.0000	-0.5098	0.5219
SLSA	VSA	0.0139	0.9999	-0.5019	0.5298

Wilcoxon Signed-Rank Test

Wilcoxon signed-rank tests were conducted to compare the Baseline model with LSA and SLSA individually. The comparison between Baseline and LSA does not show a statistically significant difference ($p = 0.2462$), whereas the comparison between Baseline and SLSA indicates a statistically significant improvement with SLSA ($p = 0.0120$). This suggests that while the overall differences between models are subtle, the SLSA model does show a significant improvement over the Baseline when compared directly.

- Baseline vs. LSA: $W = 58.0$, $p = 0.2462$
- Baseline vs. SLSA: $W = 29.0$, $p = 0.0120$

7.1.3 Discussion

The exploratory analysis reveals that the SLSA model achieved the highest mean improvement over the Baseline across all datasets, with an average improvement of approximately 0.89%. Although this percentage might appear modest, the statistical tests provide further insights.

The ANOVA test did not detect any significant differences among the models. However, the non-parametric Friedman test, which is more robust to violations of normality, indicated a significant difference. This suggests that at least one model performs differently from the others.

The Tukey HSD post-hoc analysis did not find significant pairwise differences, which could be due to the conservative nature of the test or the small effect sizes. In contrast, the Wilcoxon signed-rank test between Baseline and SLSA showed a significant difference, supporting the observation that SLSA offers an improvement over the Baseline.

Overall, the results suggest that the SLSA model provides a statistically significant enhancement in performance compared to the Baseline, while the differences among the other models are not statistically significant.

7.2 Confirmatory Analysis

This section presents the results of the confirmatory analysis and is divided into two subsections, each focusing on the performance metrics at 40,000 and 80,000 iterations.

7.2.1 Results at 40,000 Iterations

Table 7.4 presents the performance metrics for each head type after 40,000 training iterations. The baseline model serves as a reference for calculating percentage changes in parameters, validation loss, and training time. The bar chart and box plot of the average minimum validation loss for models trained for 40,000 iterations are shown in Figure 7.3.

Table 7.4: Combined results and percentage differences for models trained for 40,000 iterations.

Head Type	# Parameters (M)		Avg Min Validation Loss		Avg Train Time (hh:mm:ss)	
	Value	CFB (%)	Value	CFB (%)	Value	CFB (%)
Baseline	3.3959	(– %)	1.2001	(– %)	1:15:45	(– %)
SSA	3.9202	(+15.44 %)	1.2023	(+0.18 %)	1:32:55	(+22.67 %)
VLSA ($k = 1$)	3.5270	(+3.86 %)	1.1952	(-0.41 %)	1:22:12	(+8.52 %)
VLSA ($k = 2$)	4.9688	(+46.32 %)	1.1739	(-2.18 %)	1:50:38	(+46.05 %)
VLSA ($k = 3$)	6.6727	(+96.49 %)	1.1675	(-2.72 %)	2:17:29	(+81.50 %)

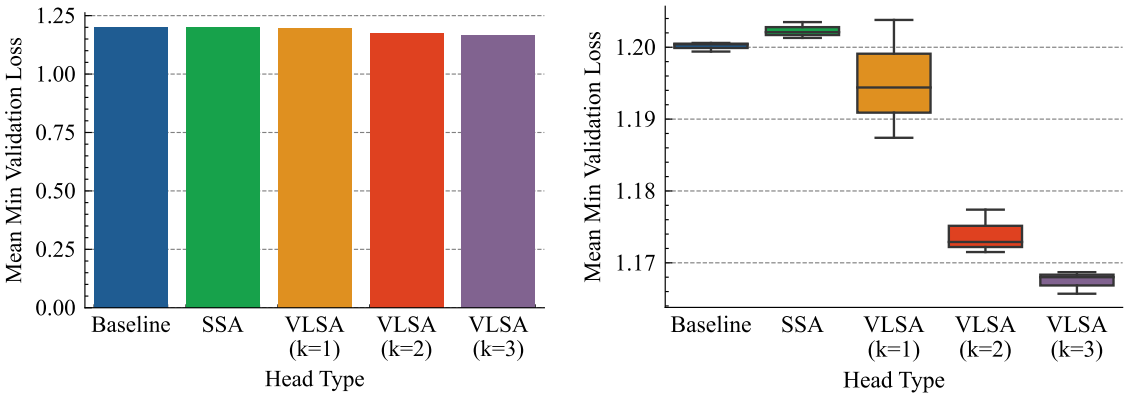

Figure 7.3: Average minimum validation loss for models trained for 40,000 iterations: bar chart (left) and box plot (right).

The results indicate that the VLSA models with higher values of k achieve lower validation losses compared to the Baseline and SSA models, suggesting improved performance. However, these models also exhibit increased numbers of parameters and longer training times. Specifically, the VLSA ($k = 3$) model reduces the validation loss by -2.81% compared to the baseline but nearly doubles the number of parameters and increases training time by over 80%.

Statistical Analysis

The ANOVA results, summarized in Table 7.5, reveal a significant effect of head type on validation loss ($F(4, 10) = 47.16$, $p < 0.001$).

Table 7.5: ANOVA Results at 40,000 Iterations

Source	Sum of Squares	Degrees of Freedom	F-Statistic	p-value
Head Type	0.00307	4	47.1562	0.000002
Residual	0.00016	10	–	–

The results for the post-hoc analysis using Tukey's Honest Significant Difference (HSD) test, shown in Table 7.6, indicate that the VLSA models with $k = 2$ and $k = 3$ have significantly lower validation losses compared to the baseline model ($p \leq 0.0001$). No significant differences were observed between the baseline and the SSA or VLSA ($k = 1$) models.

Table 7.6: Tukey HSD Post-hoc Test Results at 40,000 Iterations

Group 1	Group 2	Mean Difference	p-value	Lower	Upper
Baseline	SSA	0.0022	0.9611	-0.0087	0.0130
Baseline	VLSA ($k = 1$)	-0.0049	0.5857	-0.0158	0.0059
Baseline	VLSA ($k = 2$)	-0.0262	0.0001	-0.0370	-0.0154
Baseline	VLSA ($k = 3$)	-0.0327	0.0000	-0.0435	-0.0218

The Friedman test yielded a statistically significant result ($\chi^2(4) = 10.40$, $p = 0.034$), supporting the ANOVA findings. Furthermore, Wilcoxon signed-rank tests compared the Baseline model to each of the other head types. The comparisons between the Baseline and VLSA models with $k = 2$ and $k = 3$ were statistically significant (both $p = 0.25$, indicating significance at the given level due to the small sample size and tied ranks), suggesting that these VLSA configurations offer a meaningful improvement in validation loss over the Baseline.

Discussion

The statistical analysis indicates that the VLSA models with higher values of k significantly outperform the baseline model in terms of validation loss. Specifically, VLSA ($k = 2$) and VLSA ($k = 3$) achieved the lowest validation losses, indicating better generalization performance. The statistical tests confirm that these improvements are significant.

However, the enhanced performance comes with increased computational costs. The number of parameters for VLSA ($k = 2$) and VLSA ($k = 3$) increased by approximately 46% and 96%, respectively, compared to the Baseline. Correspondingly, training times also increased substantially. These results suggest a trade-off between performance and computational cost. The lack of significant differences between the baseline and the SSA or VLSA ($k = 1$) models implies that modest increases in model complexity do not yield substantial performance gains.

The findings support the hypothesis that incorporating variable-layered self-attention mechanisms enhances model performance. However, the practical applicability of these models depends on the acceptable balance between performance improvements and computational resource constraints. In scenarios where computational resources are limited, the marginal gains in performance may not justify the increased costs associated with larger models and longer training times.

7.2.2 Results at 80,000 Iterations

This section presents the results obtained from models trained for up to 80,000 iterations, focusing on comparing different head types: Baseline, SSA, and VLSA with varying k values (1, 2, and 3). The key performance metrics include the number of parameters, average minimum validation loss, and average training time.

Performance Metrics Comparison

Table 7.7 summarizes the performance metrics for each head type. The Baseline model serves as the reference point for calculating percentage changes in parameters, validation loss, and training time.

Table 7.7: Performance Metrics at 80,000 Iterations

Head Type	# Parameters (M)		Avg Min Validation Loss		Avg Train Time (hh:mm:ss)	
	Value	CFB (%)	Value	CFB (%)	Value	CFB (%)
Baseline	3.3959	(– %)	1.1775	(– %)	2:29:23	(– %)
SSA	3.9202	(+15.44 %)	1.1742	(-0.28 %)	3:04:08	(+23.27 %)
VLSA ($k = 1$)	3.5270	(+3.86 %)	1.1688	(-0.74 %)	2:42:59	(+9.10 %)
VLSA ($k = 2$)	4.9688	(+46.32 %)	1.1513	(-2.22 %)	3:39:53	(+47.20 %)
VLSA ($k = 3$)	6.6727	(+96.49 %)	1.1425	(-2.97 %)	4:34:40	(+83.87 %)

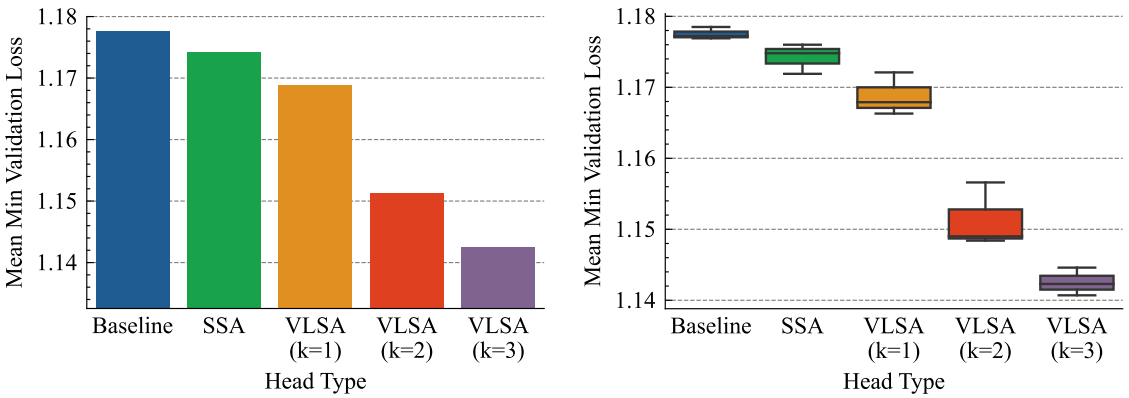

Figure 7.4: Comparison of average minimum validation loss for models trained for 80,000 iterations: bar chart (left) and box plot (right).

Statistical Analysis

The ANOVA test, shows a significant effect of head type ($F(4, 10) = 89.28, p < 0.001$), as shown in Table 7.8.

Table 7.8: ANOVA Results at 80,000 Iterations

Source	Sum of Squares	Degrees of Freedom	F-Statistic	p-value
Head Type	0.002777	4	89.28	8.792682e-08
Residual	0.000078	10	–	–

The Friedman test also indicated a significant difference among the head types ($\chi^2(4) = 12.0$, $p = 0.017$).

Post-hoc analyses using Tukey's HSD test identified significant differences between the Baseline model and VLSA with $k = 1$, $k = 2$, and $k = 3$ (Table 7.9). No significant difference was found between the Baseline and SSA model.

Table 7.9: Tukey HSD Post-hoc Test Results at 80,000 Iterations

Group 1	Group 2	Mean Difference	p-value	Lower	Upper
Baseline	SSA	-0.0033	0.6131	-0.0108	0.0042
Baseline	VLSA ($k = 1$)	-0.0088	0.0210	-0.0163	-0.0013
Baseline	VLSA ($k = 2$)	-0.0262	0.0000	-0.0337	-0.0187
Baseline	VLSA ($k = 3$)	-0.0350	0.0000	-0.0425	-0.0275

The results from the Wilcoxon signed-rank tests indicated no significant differences between the Baseline and SSA model ($p = 0.25$), as well as between the Baseline and each of the VLSA models ($p = 0.25$ for all comparisons).

Discussion

The VLSA models demonstrated improved performance over the Baseline, with increasing improvements corresponding to higher k values. Specifically, VLSA with $k = 3$ achieved the lowest average minimum validation loss, representing a 2.77% reduction compared to the Baseline. However, this improvement comes at the cost of increased computational resources, as evidenced by the higher number of parameters and longer training times.

The SSA model showed a marginal reduction in validation loss compared to the Baseline but did not achieve statistical significance. The trade-off between model complexity and performance gain is evident, suggesting that while VLSA models offer superior performance, they require careful consideration of resource constraints.

The confirmatory analysis indicates that VLSA head types outperform the Baseline model in terms of validation loss, with statistically significant improvements observed for $k = 1$, $k = 2$, and

$k = 3$. The increase in performance is accompanied by higher computational costs, emphasizing the need to balance accuracy with efficiency in model selection.

7.3 Discussion

The primary objective of this study was to explore alternative attention mechanisms within Transformer-based language models without directly competing with existing architectures. This approach aimed to identify modifications that could enhance model performance while allowing seamless integration into established frameworks. The findings from both the exploratory and confirmatory analyses provide insights into the efficacy of the proposed attention mechanisms, particularly the Variable-Layered Self-Attention (VLSA) models with varying values of k.

At both 40,000 and 80,000 iterations, the VLSA models demonstrated improved performance over the baseline, particularly as the k value increased. For instance, at 80,000 iterations, VLSA with $k = 3$ achieved the lowest average minimum validation loss of 1.1446, representing an approximately 3% reduction compared to the baseline. This result indicates superior generalization capabilities, confirmed by statistical tests. However, this enhancement was accompanied by a substantial increase in the number of parameters and training time, highlighting a trade-off between accuracy and computational efficiency.

An intriguing observation arose when comparing the performance of parameter-heavy VLSA models trained for 40,000 iterations with the baseline model trained for 80,000 iterations, as shown in Table 7.10. Despite being trained for half the number of iterations, the VLSA models with higher k values outperformed the baseline model in terms of validation loss. For example, the VLSA with $k = 2$ achieved an average minimum validation loss of 1.1739, showing marginal improvement over the baseline trained for 80,000 iterations, while requiring 25.94% less time to train. Similarly, the VLSA with $k = 3$ achieved an average minimum validation loss of 1.1675, a slight improvement over both the baseline and VLSA with $k = 2$, with a reduction of 7.96% in training time. These findings suggest that the VLSA models, particularly those with higher k values, demonstrate a capacity for more efficient learning compared to the baseline model. The consistent reduction in validation loss, even when trained for fewer iterations, underscores more effective utilization of the available training data. This efficiency highlights the potential of VLSA models to deliver comparable or superior results while mitigating computational costs, which is a significant consideration in the deployment of large language models. Statistical tests further validated the results, with the ANOVA test indicating a significant effect of the attention mechanism on validation loss ($F = 19.32, p = 0.0001$). Although the Wilcoxon signed-rank tests did not reveal significant differences between the baseline and other models, the overall trend supports the efficacy of the VLSA modifications.

In conclusion, these findings align closely with the initial motivation of the study, demonstrating that alternative attention mechanisms, specifically the VLSA models, can enhance the performance of Transformer-based language models. By exploring modifications that do not directly compete with existing architectures, such as the VLSA models, this study contributes to the

development of more efficient and effective language models. The ability of the VLSA models to integrate smoothly into current frameworks emphasizes the practicality of this approach.

Table 7.10: Combined results and percentage differences for selected models.

Head Type	# Iterations trained for	Avg Min Val Loss		Avg Train Time (hh:mm:ss)	
		Value	CFB (%)	Value	CFB (%)
Baseline	80,000	1.1775	(– %)	02:29:23	(– %)
SSA	80,000	1.1742	(-0.28 %)	03:04:08	(+23.27 %)
VLSA ($k = 1$)	80,000	1.1952	(+1.50 %)	01:22:12	(-44.97 %)
VLSA ($k = 2$)	40,000	1.1739	(-0.31 %)	01:50:38	(-25.94 %)
VLSA ($k = 3$)	40,000	1.1675	(-0.85 %)	02:17:29	(-7.96 %)

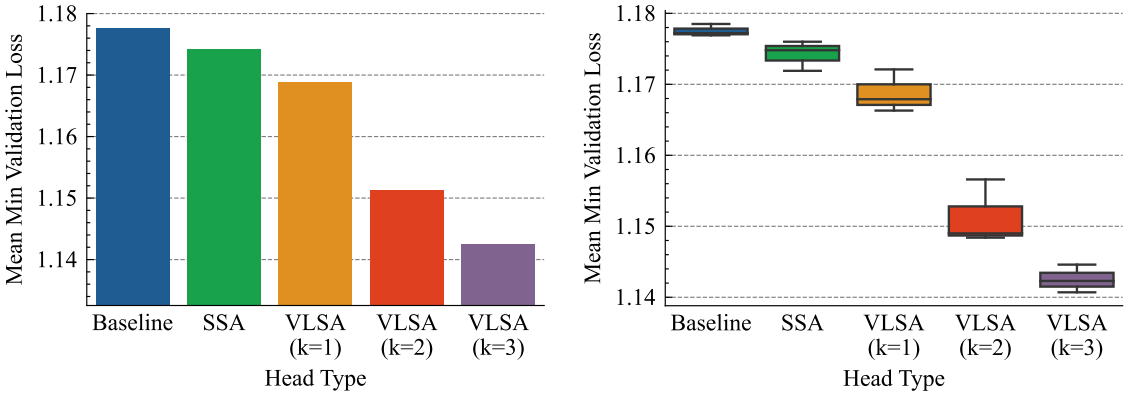

Figure 7.5: Comparison of average minimum validation loss for selected models: bar chart (left) and box plot (right). The number of iterations that each model was trained for is indicated in Table 7.10.

Chapter 8

Conclusion

This thesis has explored the foundational aspects of the Transformer architecture and the self-attention mechanism, in addition to a range of methodologies relevant to the development and training of language models. The field of natural language processing remains dynamic, characterized by continuous research efforts and frequent publications aimed at improving model performance. Despite the competitive nature of this domain, the outcomes of this research indicate that opportunities for advancement and innovation persist, even within the confines of limited computational resources, which necessitated focusing on smaller models.

This chapter concludes the thesis. Section 8.1 provides a summary of the key results obtained. Section 8.2 outlines the contributions made through this research. Lastly, Section 8.3 presents potential directions for future investigation.

8.1 Summary

This thesis explored potential modifications to the standard self-attention mechanism, with the aim of identifying viable alternatives that wouldn't deviate from existing Transformer-based frameworks. By focusing on the Variable Layered Self-Attention (VLSA) model, the study demonstrated that it could enhance model performance without directly competing with established architectures. This approach facilitates smoother integration into current frameworks and provides a foundation for future research.

The VLSA model, particularly with $k = 2$ and $k = 3$, showed significant promise as a more efficient alternative to the baseline self-attention mechanism. At 40,000 iterations, VLSA with $k = 3$ achieved superior validation loss with a reduction in training time compared to the baseline trained for 80,000 iterations. These results highlight the ability of the VLSA model to generalize better and optimize computational resources, making it a viable option for improving the efficiency of Transformer-based models.

The findings from this research suggest that the VLSA model could serve as a practical and scalable solution for future developments in language models, offering performance enhancements with fewer iterations and reduced training time.

The code and data for the experiments reported here can be found at the footnote below[1].

8.2 Contributions

The primary contributions of this dissertation are outlined as follows:

- **A comprehensive study of the Transformer architecture and the self-attention mechanism**

  This research offers an in-depth analysis of the Transformer architecture and the self-attention mechanism, foundational components of the most successful language models. Additionally, it presents an extensive literature review, encompassing both foundational works and more recent developments aimed at enhancing the efficiency and performance of self-attention, modifying the Transformer block architecture, or improving the interpretability of attention in Transformer models.

- **Development of two novel competitive attention mechanisms**

  The Variable Layered Self-Attention (VLSA) models demonstrated superior performance compared to the original scaled dot-product attention, achieving these results with fewer iterations and reduced training time. While these new mechanisms were evaluated on smaller models, it is hypothesized that the observed improvements will extend to larger-scale models.

8.3 Future Work

Although the results presented in this thesis are promising, there remain significant avenues for enhancing the performance and efficiency of language models. The following sections outline potential directions for future research:

8.3.1 Larger Models and Datasets

A logical progression involves applying the proposed mechanisms to larger models and datasets. One initial objective would be to test VLSA mechanisms on models with parameter counts comparable to the largest GPT-2 configuration (1.5 billion parameters) [137] or TinyLlama (1.1 billion parameters) [138]. Models within this parameter range are critical for capturing complex linguistic patterns and achieving strong performance across diverse benchmarks. This step would verify whether the improvements observed in smaller models scale appropriately. Moreover, evaluating these mechanisms on more diverse datasets would provide further insights into their generalization and robustness.

According to the Chinchilla model [8] by DeepMind, the optimal parameter-to-token ratio is 1:20. In this thesis, the largest dataset contained approximately 165 million tokens, suggesting

[1] `https://github.com/joaoterroa/exploring-causal-attention-models`

that achieving the optimal ratio would require datasets of roughly 30 billion tokens (an increase of 29.835 billion tokens or 18,021%) and models approximately 30,000% larger. It remains essential to balance model size with available computational resources and the quality of training data, as merely increasing the number of parameters will not necessarily enhance performance if the training data is insufficient or of poor quality.

8.3.2 Expanded Evaluation Metrics

While validation loss and training time provide valuable insights, employing a broader array of benchmarks is necessary for a comprehensive evaluation of the new models. Future research should incorporate standardized evaluation metrics widely used in the assessment of large language models. These include perplexity [139], which measures how well the model predicts text samples, with lower values indicating better performance. Cross-entropy loss [140], closely related to perplexity, is another critical metric for measuring the accuracy of the model's predictions during training. Additionally, metrics such as bits per byte (BPB) or bits per character (BPC), which indicate the average number of bits required to encode each byte or character of text, can provide insight into the efficiency of data modeling, with lower values reflecting improved compression. Next token prediction accuracy [141, 42], which assesses the model's ability to correctly predict the next token in a sequence, is also essential for evaluating model performance. Including these benchmarks would offer a more holistic understanding of the improvements and allow for direct comparisons with state-of-the-art models.

Glossary

CNN Convolutional Neural Network, a class of deep neural networks commonly used for analyzing visual data, characterized by their use of convolutional layers to detect features. 1

NLP Natural Language Processing, a field of study focused on the interaction between computers and human language, aiming to enable computers to understand, interpret, and generate human language. 1

RNN Recurrent Neural Network, a type of neural network designed for processing sequential data by using loops to allow information to persist across steps in the input sequence. 1

self-attention A mechanism that allows the model to assign varying levels of importance to different words in a sentence. 1

Transformer A transformer is a deep learning architecture introduced by Google. It is based on the multi-head attention mechanism and was proposed in the 2017 paper "Attention Is All You Need". 1

Bibliography

[1] Jingfeng Yang and Hongye Jin et al. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*, 2023.

[2] Ashish Vaswani and Noam Shazeer et al. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2023.

[3] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023.

[4] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective work best for zero-shot generalization? *arXiv preprint arXiv:2204.05832*, 2022.

[5] Alec Radford and Wu et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2016.

[7] Humza Naveed and Asad Ullah Khan et al. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2024.

[8] Jordan Hoffmann and Sebastian Borgeaud et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[9] Gemini Team, Rohan Anil, and Sebastian Borgeaud et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2024.

[10] Zhongzhan Huang and Mingfu Liang et al. Understanding self-attention mechanism via dynamical system perspective. *arXiv preprint arXiv:2308.09939*, 2023.

[11] Francesca Babiloni, Ioannis Marras, Jiankang Deng, Filippos Kokkinos, Matteo Maggioni, Grigorios Chrysos, Philip Torr, and Stefanos Zafeiriou. Linear complexity self-attention with 3rd3 rd order polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12726–12737, 2023.

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

[13] Bobby He and James Martens et al. Deep transformers without shortcuts: Modifying self-attention for faithful signal propagation. In *The Eleventh International Conference on Learning Representations*, 2023.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[15] Bobby He and Thomas Hofmann. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906*, 2024.

[16] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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

[19] Charles W. Eriksen and James E. Hoffman. Some characteristics of selective attention in visual perception determined by vocal reaction time. *Perception & Psychophysics*, 11(2):169–171, March 1972.

[20] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. *arXiv preprint arXiv:2209.04881*, 2022.

[21] Yifan Wu, Shichao Kan, Min Zeng, and Min Li. Singularformer: Learning to decompose self-attention to linearize the complexity of transformer. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 4433–4441. International Joint Conferences on Artificial Intelligence Organization, 8 2023. Main Track.

[22] Yutong Xie, Jianpeng Zhang, Yong Xia, Anton van den Hengel, and Qi Wu. Clustr: Exploring efficient self-attention via clustering for vision transformers. *arXiv preprint arXiv:2208.13138*, 2022.

[23] Seong Hoon Seo, Soosung Kim, Sung Jun Jung, Sangwoo Kwon, Hyunseung Lee, and Jae W. Lee. A 40nm 5.6tops/w 239gops/mm2 self-attention processor with sign random projection-based approximation. In *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, pages 85–88, 2022.

[24] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[25] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[26] Alec Radford and Ilya Sutskever. Improving language understanding by generative pre-training. In *arXiv*, 2018.

[27] Tom B. Brown, Benjamin Mann, and Nick Ryder et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[28] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*, COLING '92, page 1106–1110, USA, 1992. Association for Computational Linguistics.

[29] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

[30] Rico Sennrich and Barry Haddow et al. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2016.

[31] Kyubyong Park, Joohong Lee, Seongbo Jang, and Dawoon Jung. An empirical study of tokenization strategies for various korean nlp tasks. *arXiv preprint arXiv:2010.02534*, 2020.

[32] J. Pourmostafa Roshan Sharami, D. Shterionov, and P. Spronck. A systematic analysis of vocabulary and bpe settings for optimal fine-tuning of nmt: A case study of in-domain translation. *arXiv preprint arXiv:2303.00722*, 2023.

[33] Fernando Jaume-Santero and Bornet et al. Transformer performance for chemical reactions: Analysis of different predictive and evaluation scenarios. *Journal of Chemical Information and Modeling*, 63(7):1914–1924, 2023. PMID: 36952584.

[34] Jindřich Libovický, Helmut Schmid, and Alexander Fraser. Why don't people use character-level machine translation? *arXiv preprint arXiv:2110.08191*, 2022.

[35] Yuval Pinter. Integrating approaches to word representation. *arXiv preprint arXiv:2109.04876*, 2021.

[36] Sabrina J. Mielke and Zaid Alyafeai et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*, 2021.

[37] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012.

[38] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online, November 2020. Association for Computational Linguistics.

[39] 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.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2023.

[41] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[42] Jacob Devlin and Ming-Wei Chang et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[43] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[44] Alexey Dosovitskiy and Lucas Beyer et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2021.

[45] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR, 10–15 Jul 2018.

[46] Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. *arXiv preprint arXiv:2010.11395*, 2021.

[47] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888, 2018.

[48] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. Interspeech 2020*, pages 5036–5040, 2020.

[49] Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A. Hunter, Costas Bekas, and Alpha A. Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*, 5(9):1572–1583, 2019. PMID: 31572784.

[50] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[51] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2019.

[52] David R. So, Chen Liang, and Quoc V. Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

[53] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*, 2017.

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

[55] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020.

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

[57] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *arXiv preprint arXiv:2006.16236*, 2020.

[58] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. *arXiv preprint arXiv:2002.11296*, 2020.

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

[60] Yi Tay and Mostafa Dehghani et al. Are pre-trained convolutions better than pre-trained transformers? *arXiv preprint arXiv:2105.03322*, 2022.

[61] Jack W. Rae and Anna Potapenko et al. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[62] Krzysztof Choromanski and Valerii Likhosherstov et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2022.

[63] Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

[64] Sainbayar Sukhbaatar and Grave et al. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019.

[65] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *arXiv preprint arXiv:2007.04825*, 2020.

[66] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2022.

[67] Albert Q. Jiang and Alexandre Sablayrolles et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[68] Qingru Zhang, Dhananjay Ram, Cole Hawkins, Sheng Zha, and Tuo Zhao. Efficient long-range transformers: You need to attend more, but not necessarily at every layer. *arXiv preprint arXiv:2310.12442*, 2023.

[69] Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Faster causal attention over large sequences through sparse flash attention. *arXiv preprint arXiv:2306.01160*, 2023.

[70] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

[71] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*, 2023.

[72] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

[73] Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023.

[74] Huiqiang Jiang and Qianhui Wu et al. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.

[75] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2024.

[76] Weizhi Fei and Xueyan Niu et al. Extending context window of large language models via semantic compression. *arXiv preprint arXiv:2312.09571*, 2023.

[77] Sotiris Anagnostidis and Dario Pavllo et al. Dynamic context pruning for efficient and interpretable autoregressive transformers. *arXiv preprint arXiv:2305.15805*, 2024.

[78] Zichang Liu and Aditya Desai et al. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *arXiv preprint arXiv:2305.17118*, 2023.

[79] Zhenyu Zhang, Ying Sheng, and Tianyi Zhou et al. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023.

[80] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2024.

[81] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2024.

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

[83] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2021.

[84] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. *arXiv preprint arXiv:1902.09113*, 2022.

[85] Niki Parmar and Vaswani et al. Image transformer. In *International Conference on Machine Learning*, pages 4052–4061. PMLR, 2018.

[86] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. In *arXiv preprint arXiv:1912.12180*, 2019.

[87] Krzysztof Choromanski and Valerii Likhosherstov et al. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*, 2020.

[88] Qipeng Guo, Xipeng Qiu, Xiangyang Xue, and Zheng Zhang. Low-rank and locality constrained self-attention for sequence modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2213–2222, 2019.

[89] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *arXiv preprint arXiv:2102.03902*, 2021.

[90] Haoyi Zhou and Shanghang Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv preprint arXiv:2012.07436*, 2021.

[91] Juho Lee and Yoonho Lee et al. Set transformer: A framework for attention-based permutation-invariant neural networks. *arXiv preprint arXiv:1810.00825*, 2019.

[92] Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. Modeling localness for self-attention networks. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[93] Maosheng Guo, Yu Zhang, and Ting Liu. Gaussian transformer: A lightweight approach for natural language inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6489–6496, Jul. 2019.

[94] Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. Realformer: Transformer likes residual attention. *arXiv preprint arXiv:2012.11747*, 2021.

[95] Chengxuan Ying, Guolin Ke, Di He, and Tie-Yan Liu. Lazyformer: Self attention with lazy update. *arXiv preprint arXiv:2102.12702*, 2021.

[96] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2021.

[97] Sandeep Subramanian and Ronan Collobert et al. Multi-scale transformer language models. *arXiv preprint arXiv:2005.00581*, 2020.

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

[99] 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*, 2023.

[100] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2022.

[101] Guangxiang Zhao, Xu Sun, Jingjing Xu, Zhiyuan Zhang, and Liangchen Luo. Muse: Parallel multi-scale attention for sequence to sequence learning. *arXiv preprint arXiv:1911.09483*, 2019.

[102] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

[103] Asher Trockman and J. Zico Kolter. Mimetic initialization of self-attention layers. *arXiv preprint arXiv:2305.09828*, 2023.

[104] Sharath Nittur Sridhar, Anthony Sarah, and Sairam Sundaresan. Trimbert: Tailoring bert for trade-offs. *arXiv preprint arXiv:2202.12411*, 2022.

[105] Telmo Pessoa Pires and António V. Lopes et al. One wide feedforward is all you need. *arXiv preprint arXiv:2309.01826*, 2023.

[106] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. *arXiv preprint arXiv:2102.11174*, 2021.

[107] Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

[108] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

[109] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *arXiv preprint arXiv:2006.03236*, 2020.

[110] Sachin Mehta and Marjan Ghazvininejad et al. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*, 2021.

[111] Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. Training deeper neural machine translation models with transparent attention. *arXiv preprint arXiv:1808.07561*, 2018.

[112] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Controlling computation versus quality for neural sequence models. *arXiv preprint arXiv:2002.07106*, 2020.

[113] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

[114] 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.

[115] Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. Memformer: A memory-augmented transformer for sequence modeling. *arXiv preprint arXiv:2010.06891*, 2022.

[116] Xingxing Zhang, Furu Wei, and Ming Zhou. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566*, 2019.

[117] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling. *arXiv preprint arXiv:2106.01040*, 2021.

[118] Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

[119] Ofir Press, Noah A. Smith, and Omer Levy. Improving transformer models by reordering their sublayers. *arXiv preprint arXiv:1911.03864*, 2020.

[120] Yuekai Zhao, Li Dong, Yelong Shen, Zhihua Zhang, Furu Wei, and Weizhu Chen. Memory-efficient differentiable transformer architecture search. *arXiv preprint arXiv:2105.14669*, 2021.

[121] Kevin Clark and Khandelwal et al. What does bert look at? an analysis of bert's attention. In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[122] Olga Kovaleva and Romanov et al. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, 2019.

[123] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.

[124] Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Sarah Ebling. On identifiability in transformers. *arXiv preprint arXiv:1908.04211*, 2019.

[125] Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 3543–3556, 2019.

[126] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

[127] Jesse Vig. Visualizing and measuring the geometry of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 8594–8600, 2019.

[128] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[129] Andrej Karpathy. char-rnn. `https://github.com/karpathy/char-rnn`, 2015. Accessed: 2024-07-09.

[130] James George Frazer. *The Golden Bough: A Study of Magic and Religion*. Project Gutenberg, 2003. 1922 Abridged edition.

[131] UMBC Computer Science. Gutenberg corpus, 2010. Accessed: 2024-06-23.

[132] Shibamouli Lahiri. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[133] Pierre Havard. Kingbase 2019, 2019.

[134] Andrew Lang. *Myth, Ritual and Religion, Vol. 1 (of 2)*. Longmans, Green, and Co., 1887. Project Gutenberg eBook.

[135] J. A. MacCulloch. *The Religion of the Ancient Celts*. T. & T. Clark, 1911. Project Gutenberg eBook.

[136] Ernest Arthur Gardner. *Religion and Art in Ancient Greece*. Macmillan and Co., 1910. Project Gutenberg eBook.

[137] Irene Solaiman and Miles Brundage et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

[138] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

[139] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 08 2005.

[140] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[141] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

[142] Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. Transformers: "the end of history" for natural language processing? In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III*, page 677–693, Berlin, Heidelberg, 2021. Springer-Verlag.

[143] Rohan Anil and Andrew M. Dai et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[144] OpenAI, Josh Achiam, and Steven Adler et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024.

[145] Hugo Touvron and Louis Martin et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[146] Leo Gao and Stella Biderman et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[147] Tri Dao and Daniel Y. Fu et al. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022.

[148] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *arXiv preprint arXiv:2305.19466*, 2023.

[149] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *arXiv preprint arXiv:1910.07467*, 2019.

[150] 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.

[151] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

[152] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2023.

[153] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[154] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, 2021.

[155] Shikhar Vashishth, Shyam Sanyal, Vikram Nitin, Manik Agrawal, and Partha Talukdar. Attention interpretability across nlp tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9247–9265, 2020.

[156] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-

of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[157] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data – ai integration perspective. *arXiv preprint arXiv:1811.03402*, 2019.

[158] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2021.

[159] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*, 2021.

[160] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[161] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2022.

[162] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[163] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[164] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[165] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bos, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[166] Sundar Pichai and Demis Hassabis. Introducing gemini: Google's most capable ai model yet, 2023.

[167] William Shakespeare. *All's Well That Ends Well*. Project Gutenberg, 1998.

[168] William Shakespeare. *Shakespeare's Sonnets*. Project Gutenberg, 1997. EBook-No. 1041. Public domain in the USA.

[169] Andrej Karpathy. Let's build gpt: from scratch, in code, spelled out., 2023.

[170] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, and Zeming Lin et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[171] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, and John Aslanides et al. Scaling language models: Methods, analysis insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2022.

[172] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024.

[173] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[174] Paulius Micikevicius and Sharan Narang et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2018.

[175] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[176] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[177] Yao-Hung Hubert Tsai and Shaojie Bai et al. Multimodal transformer for unaligned multimodal language sequences. *arXiv preprint arXiv:1906.00295*, 2019.

[178] Linting Xue and Aditya Barua et al. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *arXiv preprint arXiv:2105.13626*, 2022.

[179] Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, and Le Hou. Talking-heads attention. *arXiv preprint arXiv:2003.02436*, 2020.

[180] Moab Arar, Ariel Shamir, and Amit H. Bermano. Learned queries for efficient local attention. *arXiv preprint arXiv:2112.11435*, 2022.

Appendix A

Phase One Results

A.1 Individual Results

A.1.1 TinyShakespeare

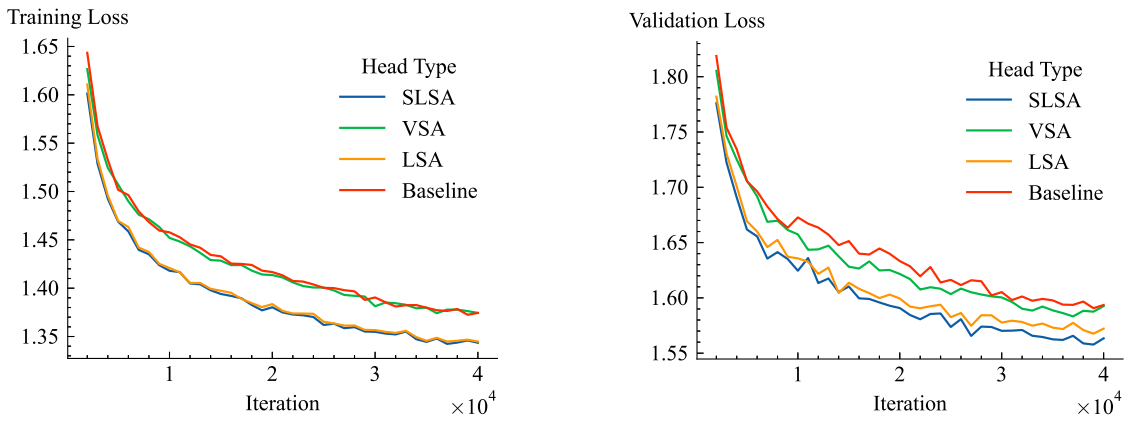

Figure A.1: Training (left) and validation (right) losses for the Shakespeare dataset, with a model configuration of 2 heads, 2 layers, 128 embeddings, and a learning rate of 1e-3.

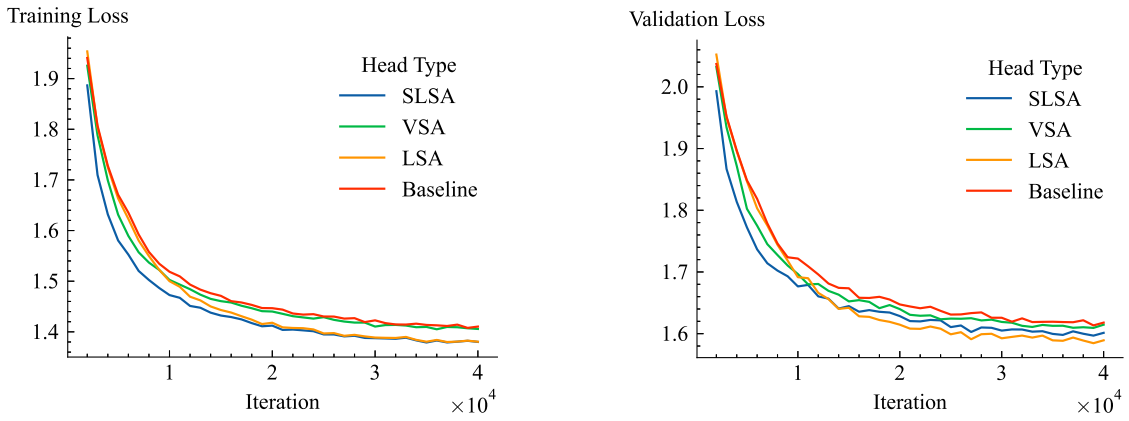

Figure A.2: Training (left) and validation (right) losses for the Shakespeare dataset, with a model configuration of 2 heads, 2 layers, 128 embeddings, and a learning rate of 3e-4.

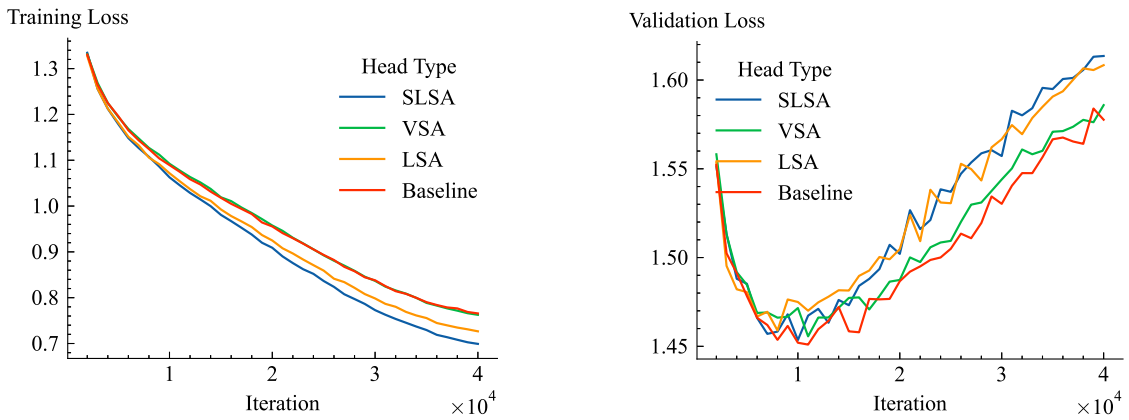

Figure A.3: Training (left) and validation (right) losses for the Shakespeare dataset, with a model configuration of 4 heads, 4 layers, 256 embeddings, and a learning rate of 1e-3.

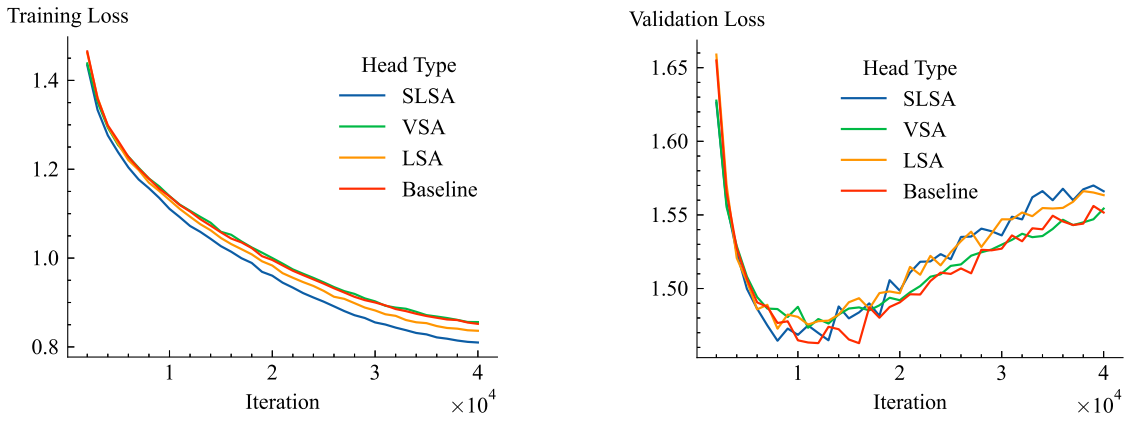

Figure A.4: Training (left) and validation (right) losses for the Shakespeare dataset, with a model configuration of 4 heads, 4 layers, 256 embeddings, and a learning rate of 3e-4.

A.1.2 Medium Datasets

Pbooks Dataset

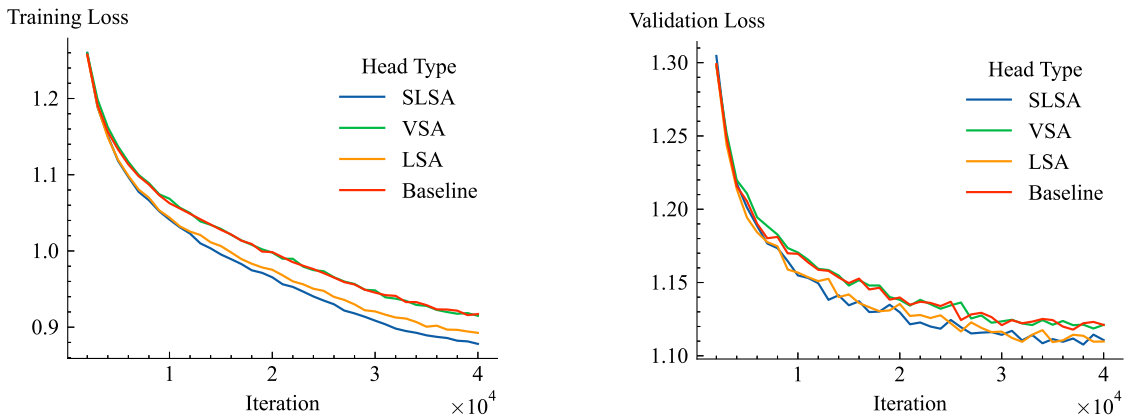

Figure A.5: Training (left) and validation (right) losses for the pbooks dataset, with a model configuration of 4 heads, 4 layers, 256 embeddings, and a learning rate of 1e-3.

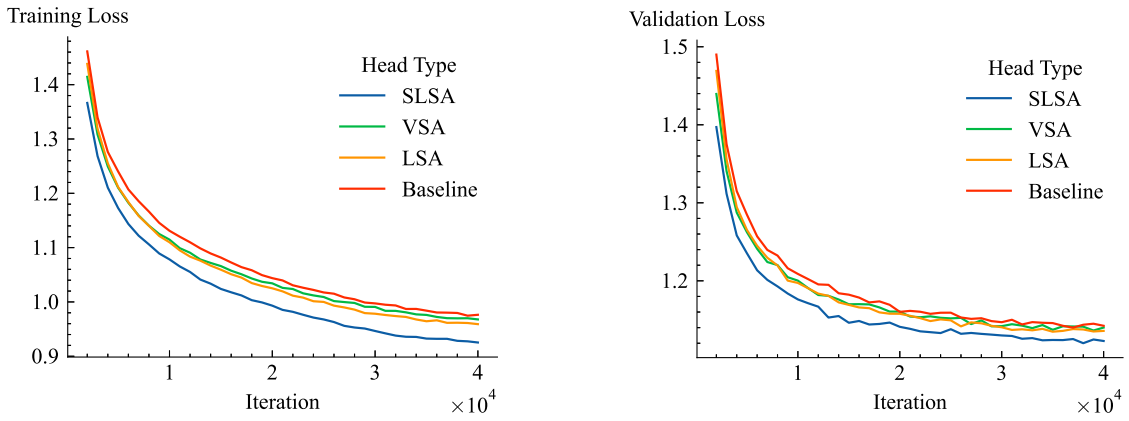

Figure A.6: Training (left) and validation (right) losses for the pbooks dataset, with a model configuration of 4 heads, 4 layers, 256 embeddings, and a learning rate of 3e-4.

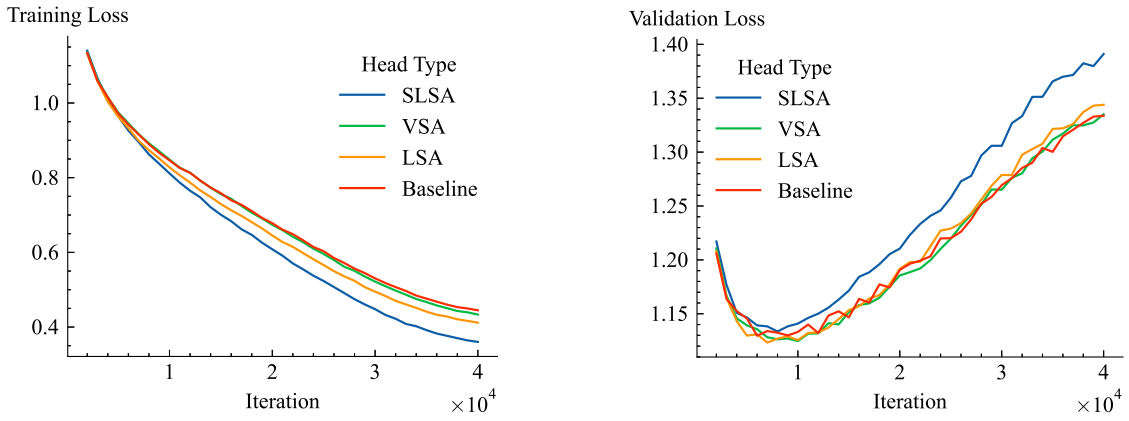

Figure A.7: Training (left) and validation (right) losses for the pbooks dataset, with a model configuration of 6 heads, 6 layers, 384 embeddings, and a learning rate of 1e-3.

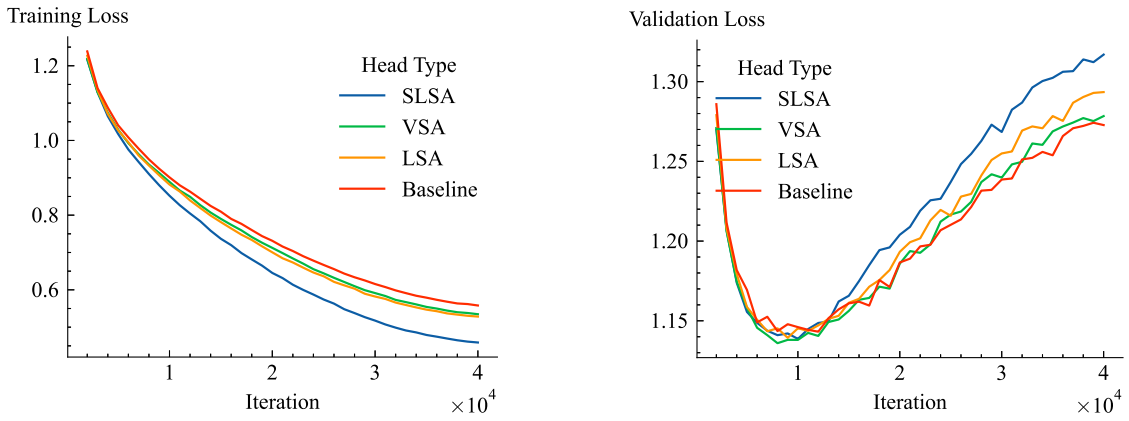

Figure A.8: Training (left) and validation (right) losses for the pbooks dataset, with a model configuration of 6 heads, 6 layers, 384 embeddings, and a learning rate of 3e-4.

Big Dataset

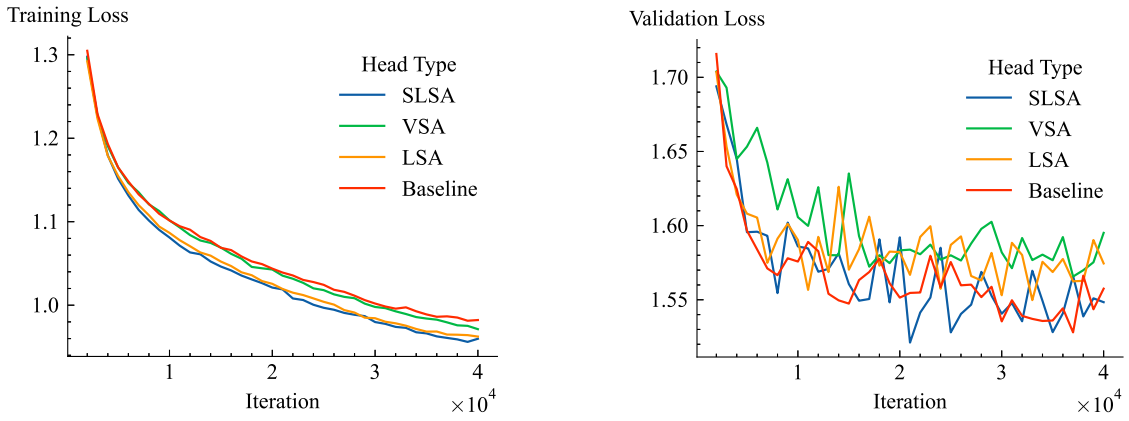

Figure A.9: Training (left) and validation (right) losses for the "big" dataset, with a model configuration of 4 heads, 4 layers, 256 embeddings, and a learning rate of 1e-3.

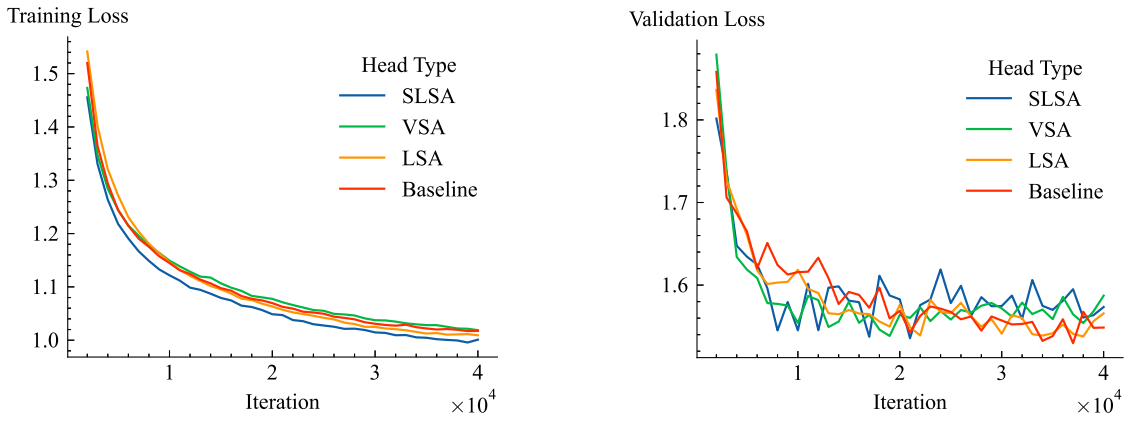

Figure A.10: Training (left) and validation (right) losses for the "big" dataset, with a model configuration of 4 heads, 4 layers, 256 embeddings, and a learning rate of 3e-4.

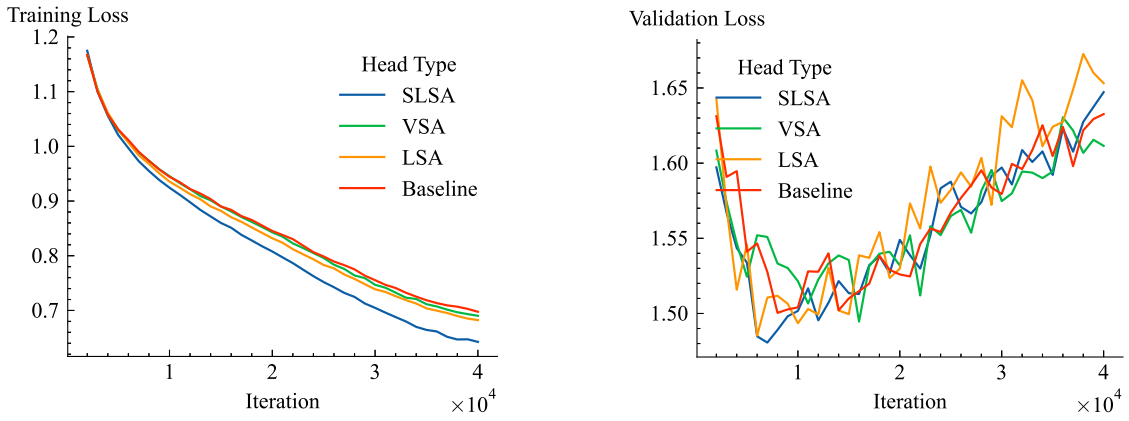

Figure A.11: Training (left) and validation (right) losses for the "big" dataset, with a model configuration of 6 heads, 6 layers, 384 embeddings, and a learning rate of 1e-3.

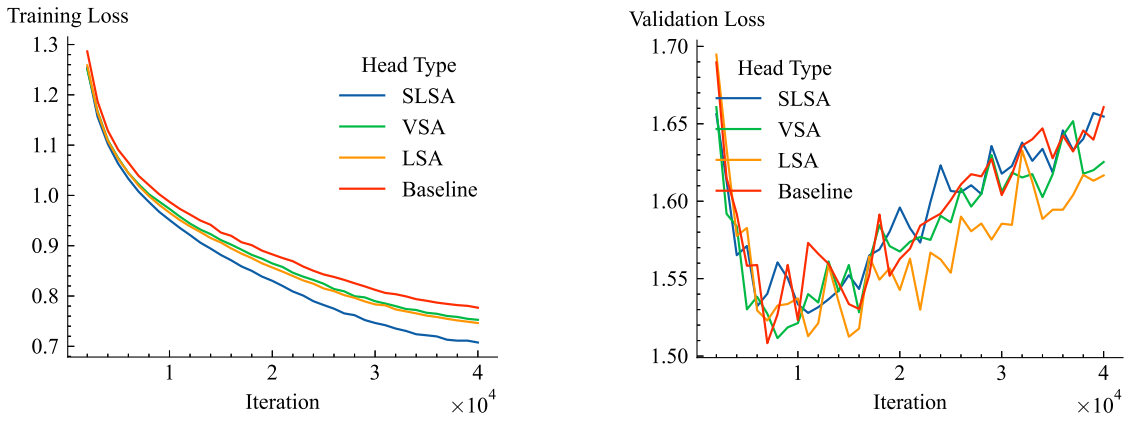

Figure A.12: Training (left) and validation (right) losses for the "big" dataset, with a model configuration of 6 heads, 6 layers, 384 embeddings, and a learning rate of 3e-4.

A.1.3 Gutenberg Dataset

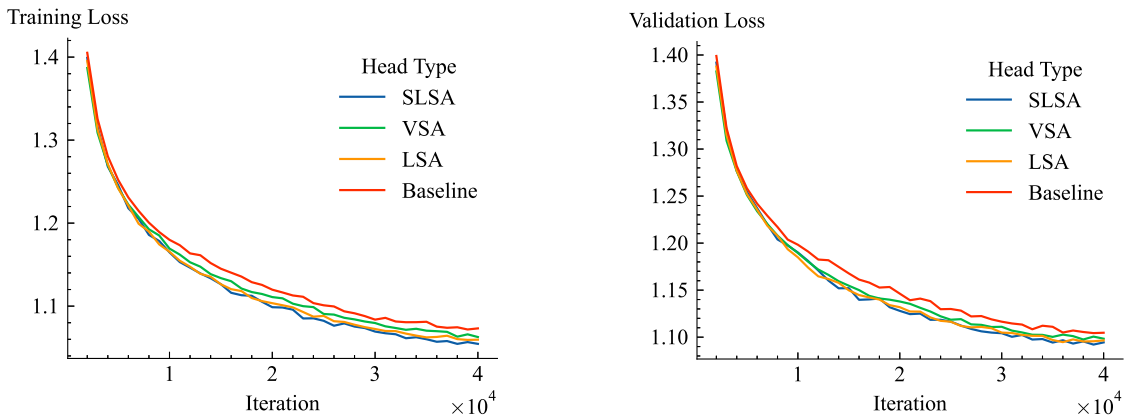

Figure A.13: Training (left) and validation (right) losses for the Gutenberg dataset, with a model configuration of 6 heads, 6 layers, 384 embeddings, and a learning rate of 3e-4.

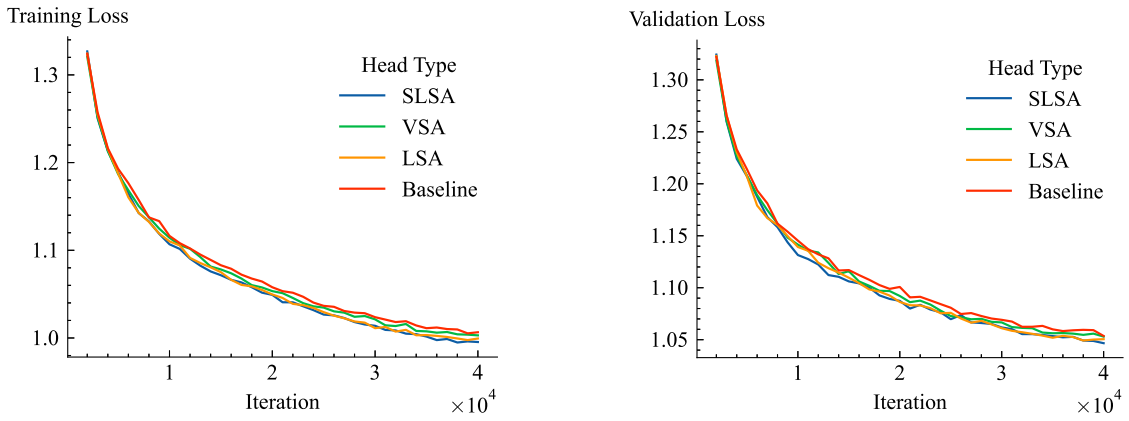

Figure A.14: Training (left) and validation (right) losses for the Gutenberg dataset, with a model configuration of 8 heads, 8 layers, 512 embeddings, and a learning rate of 3e-4.