Gated ResNet Architecture for Transformer Multi-head Attention Block

by

Ahnaf Hassan
21341009
Mubtasim Fuad Mahde
24341116

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Science
Brac University
January 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ahnaf Hassan
21341009

Mubtasim Fuad Mahde
24341116

Approval

The thesis/project titled "Gated ResNet Architecture for Transformer Multi-head Attention Block" submitted by

1. Ahnaf Hassan(21341009)
2. Mubtasim Fuad Mahde(24341116)

Examining Committee:

Supervisor:
(Member)

Dr. Farig Yousuf Sadeque
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Jawaril Munshad Abedin
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Member)

Sadia Hamid Kazi
Associate Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Transformer model architectures show excellent performance over machine learning-related tasks, primarily focusing on natural language processing by removing the need for recurrence and convolutional techniques. Residual networks (ResNet) are a core component for transformer models to retain long-term dependencies for sequence-to-sequence-related tasks. This paper introduces a new ResNet for the attention layer of a transformer model. Through a ResNet connection between layers of multi-head blocks, we flow information from one layer to the next while keeping the number of parameters the same. We also explore two-layer deep connections that retain long-term dependencies even more than one layer. Furthermore, we implement a gating mechanism on the ResNet that will selectively allow less redundant information to flow through and ensure that the gradient convergence of the model can be accelerated. In this study, we intend to prove that the performance of a model can increase even if the parameter increases due to introducing the gated unit which is insignificant in comparison to the overall number of parameters. Therefore, enhancing the model's performance on seq2seq tasks without demanding additional training time and memory by adjusting a few trainable parameters introduced for the gating mechanism is the primary goal of the paper. Our model showed promising results, especially on long-term dependent seq2seq tasks, by achieving a better performance score as well as maintaining similar efficiency.

Keywords: Transformer model, Residual networks (ResNet), Attention layer, Multi-head blocks, Long-term dependencies, Sequence-to-sequence tasks, Gating mechanism, Gradient convergence, Parameter increase, Memory reduction, Seq2seq tasks, Training stability, Sparse attention scores, Model performance, Information flow, BLEU score.

Acknowledgment

We acknowledge the accomplishment of this research paper and owe all to Almighty Allah, who provided us the opportunity to study and explore. We owe a great deal of gratitude to our supervisor, Dr. Farig Yousuf Sadeque, and to our co-supervisor, Jawaril Munshad Abedin, for their valuable assistance and support throughout the study. We extend this gratitude further to all the lecturers and professors of Brac University who helped us improve the quality of our research. Finally we would like to end by thanking our beloved parents for their support and positive encouragement.

Table of Contents

List of Figures

List of Tables

Index of Acronyms and Shorthands

The next list describes several symbols and abbreviation that will be later used within the body of the document.

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Networks

API Application Programming Interface

ASR Automatic Speech Recognition

BERT Bidirectional Encoder Representations from Transformers

BLEU Bilingual Evaluation Understudy

BPE Byte Pair Encoding

CEC Content-Encoding Component

CNN Convolutional Neural Networks

CPU Computer Processing Unit

F1 F1 Score

FFNN Feed Forward Neural Network

GAN Generative Adversarial Network

GRU Gated Recurrent Unit

GPU Graphics Processing Unit

GPT Generative Pretrained Transformer

IWSLT International Conference on Spoken Language Translation

LDA Latent Dirichlet Allocation

LSTM Long Short-Term Memory

LLM Large Language Model

Llama Large Language Model Meta AI

MT Machine Translation

ML Machine Learning

NLP Natural Language Processing

NER Named Entity Recognition

NN Neural Network

PaLM Pathways Language Model

POS Part of Speech

QA Question Answering

REAL Residual Attention Layer Transformer

ResNet Residual Network

RNN Recurrent Neural Networks

RTE Recognizing Textual Entailment

SOTA State of the Art

SDU Self Dependency Unit

TTS Text to Speech

VAE Variational Autoencoder

WMT Conference of Machine Translation

Shorthands

- **acc**: Accuracy
- **avg**: Average
- **dims**: Dimensions
- **epoch**: A complete pass through the entire dataset
- **G1_E4_H6**: Gated ResNet Transformer 4 Encoder with 6 Heads
- **G1_E4_H8**: Gated ResNet Transformer 4 Encoder with 8 Heads
- **G1_E6_H6**: Gated ResNet Transformer 6 Encoder with 6 Heads
- **G2_E4_H6**: Gated ResNet Transformer 4 Encoder with 6 Heads
- **G3_E4_H6**: Gated ResNet Transformer 4 Encoder with 6 Heads
- **loss**: Loss function (used to measure the performance of a model)
- **lr**: Learning rate
- **M2_E4_H6**: Multi-Layered RealFormer 4 Encoder with 6 Heads
- **M2_E4_H8**: Multi-Layered RealFormer 4 Encoder with 8 Heads
- **M2_E6_H6**: Multi-Layered RealFormer 6 Encoder with 6 Heads
- **M3_E4_H6**: Multi-Layered RealFormer 4 Encoder with 6 Heads
- **R_E4_H6**: RealFormer 4 Encoder with 4 Heads
- **R_E4_H8**: RealFormer 4 Encoder with 8 Heads
- **R_E6_H6**: RealFormer 6 Encoder with 6 Heads
- **T_E4_H6**: Transformer 4 Encoder with 4 Heads
- **T_E4_H8**: Transformer 4 Encoder with 8 Heads
- **T_E6_H6**: Transformer 6 Encoder with 6 Heads

Chapter 1

Introduction

The Transformers [13] model is the peak state-of-the-art (SOTA) technology when talking about language understanding and processing using neural network and machine learning. The versatility of the model in application are immense. Similarly, the model also excels in performance as well. Consequently, the model finds widespread application in various tasks such as NLP,computer vision, etc. Transformers became the building blocks for many models, including BERT and GPT. The overall architecture of transformer models is complex and difficult to interpret, so it is often termed a "black box."
The base-model transformer introduced in the study [13] contains primarily two blocks. The first block contains a stack of six identical encoders, followed by another block containing a stack of six identical decoders. For simplicity in understanding the model, let's consider input as series of word that are in a list or sequence. These sequences of words are converted to tokens and represented as continuous vectors. Position information about the sequence of words is added to the continuous vector representation and fed into the model. Inside each of the encoder, there are mainly two sub-layers. self-attention sub-layer is followed by a simple feed-forward neural network. The input sequence is replicated into multiple heads, which pass through a linear projection to create the query, key, and value matrices. Since each token of the input sequence attends to every other token in the sequence, it is termed self-attention. These matrices are computed together to get the attention score. Furthermore, each of the Q, K, and V matrices is unique to each other as the trainable parameter matrices (W_Q, W_K, and W_V) used in the linear projection are randomized. Consequently, this allows the model to capture similarities between the tokens based on different contexts by using operating a dot product multiplication between the query and key matrices. Later, the weighted sum of the scores is pushed through a simple neural network to generate an output of the attention scores. Additionally, there are ResNet connections between each sub-layer to avoid vanishing or exploding gradient problems as well as to conserve the original information of the input sequence. Unlike the encoders, the decoders contain three sub-layers. Firstly, the masked multi-head self-attention sub-layer shifts the input sequence by one position and then lets the model predict the masked words, generating masked attention scores. Followed by the first sub-layer, the second sub-layer also computes attention scores, taking the query and key input from the encoder output and the value from the masked attention scores calculated beforehand. Finally, similar to the encoder layers, the third sub-layer of the decoder is also a

feed-forward neural network that converts the attention scores into an output. The generated output is passed through a SoftMax function that calculates the probability of the next sequence of tokens. The representation of large sequences of data is computationally costly when it comes to deep neural networks (DNN). Additionally, many problems require the importance of tracking sequences. For instance, in "machine translation," the conversion of sentences from one language to another requires the DNNs to retain information about sequences of words and, conversely, generate sequences of translated sentences [7]. Furthermore, "speech recognition" also requires understanding the sequence of words and interpreting those sequences to generate an output [7]. To extend, sequence-to-sequence (seq2seq) learning is a crucial aspect of many problem-solving tasks. Transformer models are particularly more efficient than recurrent neural networks (RNN) and long-short-term memory (LSTM). According to the paper, "Transformers can be trained significantly faster than architectures based on recurrent or convolutional layers."[13]. This is because transformers do not process data sequentially but rather allow parallel processing of data. This allows transformers to be more efficient in training compared to other models. Moreover, long sequences of input mean longer dependencies, which result in loss of information for models like RNN and LSTM. But transformer models can capture global dependencies of words, which makes this model highly effective for seq2seq-related tasks. Scalability and flexibility of transformers inherently give the model the ability to be tweaked for solving specific problems. To summarize, transformer models excel at seq2seq-related tasks through their ability to compute parallelly and their potential to capture global decencies with large-scale data. Although transformer models show excellent success in multiple fields of computer science, they also have several limitations. For example, there are high computational resource requirements as well as memory requirements. The self-attention mechanism itself scales quadratically as the sequence length increases. Furthermore, retaining information about learnable parameters is extremely difficult, resulting in the necessity of high memory. Even with fast Graphical or Tensor Processing Unit, transformer models tends to take a long time for training over large-scale data.

1.1 Research Problem

It is established that transformer models revolutionized various seq2seq tasks, yet limitations such as computation cost and training efficiency are still a major challenge to overcome. This paper will primarily emphasize increasing the training efficiency of transformer models as well as improving performance. Moreover, the increase in learnable parameters leads to higher memory requirements. Because an increase in parameters and memory requirements will decrease the performance and efficiency of the model, On the other hand, the interpretability of the model is also important. Creating complexity in the model often leads to lack of interpretability making the models behaviour unpredictable. To address these challenges, we took inspiration from Highway Transformer [25] and RealFormer [30]. Firstly, highway transformers utilize skip connections—highway networks, to be precise. By replacing the normal ResNet connections in the base transformer model, they utilize gated skip connections. As a result, the gating mechanism can selectively leave out redundant information, resulting in faster gradient convergence. According to a paper, "subsidiary content-based SDU gates allow for the information flow of modulated

latent embeddings through skipped connections, leading to a clear margin of convergence speed with gradient descent algorithms" [25]. This process will accelerate the model's push towards suboptimal points during optimization. Consequently, there are drawbacks to this architecture. The gating mechanism known as SDU increases additional parameters and subsequently increases memory requirements. Accordingly increases the computational cost and training time and decreases the overall performance of the model, especially for deeper layered transformers. Secondly, RealFormer also utilizes skip connections, i.e., ResNets, in the multi-head sub-layer. The residual connections carry information about the scaled dot product attention score ($QK^T/root(D_k)$) from the previous encoder/decoder layer to the next layer. Furthermore, the analysis of the results empirically proves that "RealFormer stabilizes training and leads to models with sparser attention" [30], particularly tasks related to "masked language modeling, neural machine translation, and long document modeling." Similar to highway transformers, RealFormer also possesses potential drawbacks in its architecture. Since the model carries information within heads and across heads of adjacent layers, it requires additional memory requirements. Unlike the SDU gate mechanism, RealFormer does not use any gating mechanism. Thus reducing the necessity of extra parameters and computational costs.

1.2 Research Objective

In simple terms, our research objective is to increase model performance as well as the training efficiency of transformer models. We aim to achieve these goals by implementing three approaches that involve highway networks and ResNet connections on the scale dot-product score of the self-attention sub-layer. This paper will empirically evaluate the performance as well as training efficiency of proposed model compared to other transformer models. We will also highlight any anomalies in our results. Furthermore, implementation of the highway network containing gating mechanisms will increase learnable parameters for the model, and so optimization for memory usage is also considered one of our priorities. We evaluate the results of three approaches by comparing them with other base-level transformer architectures. We will also try to understand which behavior or nature of the model yielded such results. We will also provide suggestions on how to improve the model.

Chapter 2

Related Work

The influence of artificial intelligence (AI) and artificial neural networks (ANN) on day-to-day human life is immense. These technologies and neural network models are rapidly changing every moment because, according to a study [14], "ANNs are new computational models with rapid and large uses for handling various complex real-world issues." To extend this, neural networks tend to observe and identify data patterns and predict outcomes based on the problem statement. Furthermore, one of the core components of human civilization is language, and "linguistics is the science of language, which includes phonology, which refers to sound, morphology, which refers to understanding word formation, semantics syntax and pragmatics." [34]. As a result, understanding and generating nature language or human spoken language are important technologies to explore and develop for computers to understand human language. From the two survey papers [14] and [34], we identify the current trends involving natural language processing (NLP) technologies. Such as the "transformers". Moreover, our research strategy involves understanding the architecture of transformer models such as "Transformer-XL," "BERT," and "gated recurrent unit (GRU)" and further improving them [34]

The transformer model was introduced initially in the paper "Attention is All You Need," established by Vaswani et al. [13]. The translation of languages from one to another was the key purpose behind the new model. Moreover, the model utilizes the concept of "attention mechanisms to draw global dependencies between input and output" [13]. Consequently, the model was able to show "significantly more parallelization" compared to older NLP models like RNN and LSTM [13]. Consequently, it allowed the model to be "trained significantly faster than architectures based on recurrent or convolutional layers" [13]. Each encoder layer of the model contains two sub-layers, one of which is a "multi-head self-attention mechanism" and the other a "position-wise fully connected feed-forward network"[13]. The input sequences are encoded position-wise and fed to the model, and the output originated from the final encoder layer is further passed into the decoder. In addition to this, the first sub-layer of the decoder is a masked attention layer where the position of the word sequence is shifted by one. The paper [13] evaluated the model based on the BiLingual Evaluation Understudy (BLUE) score and compared it with other models. It was evident that the transformer outperformed all other models and became the new SOTA technology not just in the field of NLP but in many other computer science fields as well. Furthermore, the base-transformer model has recently evolved

into a more complex and efficient model.

The concept of additive attention was first introduced by Bahdanau et al. [8], which later became a building block for the transformer [13] model. For instance, in a transformer model, each word attends to every other word, allowing the model to understand the meaning of words effectively by attending to other words. This is termed self-attention, where each position generates an attention score that helps the model capture dependencies regardless of having long sequences or distances. As a result, this helps language models avoid having to "encode a whole source sentence into a fixed-length vector"[8] like RNN or LSTM. Furthermore, this attention mechanism increased the ability of a model to understand context from limited length to global. Achieving better performance and parallelization. For deeper understanding, the input sequence is replicated into n heads, and each head is passed through a learned linear projection to generate the Q,K, V matrices [29]. Since the input sequence is the same for all, positional context of a word is also captured by the model. Hence the term self-attention. The "scaled dot-product attention" is then multiplied by the value (V) to get the attention scores and later fed to a FFNN [13]. Finally, the heads are concatenated, and the encoder generates an output for the decoder in the transformer model.

An encoder-only type model is the "BERT" where the self-attention mechanism is bidirectional. There are two methods for training this model: masked LM and next-sentence prediction. These two methods enable the utilization of bidirectional self-attention mechanisms. BERTLARGE and BERTBASE surpass all other systems on all tasks by a significant margin, with average accuracy improvements of 4.5% and 7.0%, respectively, compared to the previous SOTA technology. The results of the BERT model show that increasing the size of a model improves large-scale tasks, but it also largely improves small-scale tasks. The two models introduced have around 110M parameters and 340M parameters, respectively. According to the paper [17], "scaling to extreme model sizes also leads to large improvements on very small-scale tasks." This paper introduces methods to train and represent a transformer using bidirectional encoders, which is revolutionary.

Similarly, in another paper, "larger models make increasingly efficient use of in-context information" [24]. This paper introduces three different models with parameters in the range of 125M and 175B. The model GPT-3 has the same architecture as its predecessor GPT-2 [19], which is a decoder-only transformer model. In contrast to the BERT model, GPT-3 uses three different approaches to pre-train the model. The approaches are: (i) Fine-tuning: repeated gradient updates but requires a large database; (ii) Few-shot: given few examples of the task and task description, the model attempts to predict without any gradient updates; (iii) One-shot: like few-shot, except only one example of the task is given; and (iv) Zero-shot: only the task description is given. The paper [24] explains that these approaches have various advantages and trade offs depending on the problem setting. One notable limitation of GPT-3 is its pre-training efficiency. "While GPT-3 takes a step towards test-time sample efficiency closer to that of humans, it still sees much more text during pre-training than a human sees in their lifetime"[24].

Another well-known transformer model, known as the T5 [28], or text-to-text transfer transformer, showed promising results and became another SOTA technology in the fields of NLP. The key concepts and design behind the T5 model are primarily focused on the conversion of any input-output problem into a text-based format. The unification of input and output lets the model be more versatile. Moreover, the architecture of the T5 model is similar to that of a that of a normal base-model transformer, which contains both an encoder and a decoder. This architectural structure is in contrast to models like BERT. "Modifications to the BERT-style denoising objective" [28] was also one of the objectives of the paper. BERT-based models are only composed of encoder layers trained specifically for "span predictions as well as classification" [28]. Furthermore, the model was pre-trained over the "Colossal Clean Crawled Corpus (C4)" [28], which is considered a large and diverse dataset. This allowed the model to capture more context and information from the inputs. The model excels at multiple tasks that include translation, summarization, classification, and more. The paper also solves the issues related to parameters and reports, "encoder-decoder model which uses twice as many parameters as "encoder-only" or "decoder-only" (language model) architectures, it has a similar computational cost" [28].

One of the key components of a deep neural network architecture is the activation function, which determines whether a neuron will be fired or not. Choosing an activation function is crucial for both training and optimization. For recurrent neural networks, the tanh function is mostly used. Another variant of this function is the Hardtanh function, which is reportedly both faster and more accurate than tanh [31]. According to the paper [8], due to the vanishing gradient problem in both sigmoid and tanh functions, ReLU is the preferred choice. Another mechanism that assists backpropagation is the gated mechanism. In such an architecture, like the paper published by Hochreiter and Schmidhuber [1], a gated mechanism is used to input some of the features of a unit and forget some of the features from all the previous units through an input and forget gate, respectively. As explained in the paper, "The multiplicative input gate affords protection of the CEC from perturbation by irrelevant inputs. Likewise, the multiplicative output gate protects other units from perturbation by currently irrelevant memory contents" [1]. This allows the architecture to bridge long-term gaps. Over the years, several other gating mechanisms have been introduced, such as GRU, peephole connections, etc.

The advantages that gated mechanisms bring to transformers have been explored in the paper [25]. In their work, they use residual connections along with a gated mechanism in a block of transformers. The SDU gates run parallel with the multi-headed pairwise attention block and the feedforward network. This allows bridging between long time gaps with minimal time loss and an increase in a few trainable parameters. According to study [25], it is evident that shallow layers tend to focus on local regions, and each layer attends to different semantic aspects. Furthermore, these gating mechanisms are essentially activation functions that are utilized traditionally in NN. According to a study, "divergences between biological and machine learning models concern non-linear activation functions" [3] and gave a comparative study over activation functions. Furthermore, the paper explained that "logistic sigmoid neurons are more biologically plausible than hyperbolic tangent neurons, the latter

work better for training multi-layer neural networks" [3]. Moreover, the paper also somewhat explains the importance of having sparser data. Sparse data is easier to understand and evaluate while maintaining memory efficiency. "if a representation is both sparse and robust to small input changes, the set of non-zero features is almost always roughly conserved by small changes of the input" [3]. Sparser data are easier to differentiate by "non-linear machinery" [3]. Skip connections that are modified with a gating mechanism are usually referred to as highway networks [10]. The paper proposed a new gating mechanism known as "Rectified Linear Unit (RELU)" [3] which yielded better results compared to sigmoid functions or hyperbolic tangent functions. In 2013, further study in activation functions gave us the Leaky RELU [4] which is an improved version of the normal RELU.

The Residual Attention Layer Transformer, also known as the RealFormer, was initially established by Google researchers [30]. The proposed model "significantly outperforms the canonical Transformer and its variants (BERT, ETC, etc.)" [30]. In this variant of Transformer, the model utilizes ResNet connections between multi-head attention layers. To extend, the model adds $(QK^T/root(d_k))$ of the previous layer to the next multi-head sublayer of an encoder before calculating the attention scores. As a result, "it does not add expensive multiplication operations; performance is expected to be comparable" with other variants of transformer models [30]. On top of that, RealFormer has sparser attention in deeper layers and a lower discrepancy across all layers, indicating that attention density is less input-dependent. The model adds no extra parameters or hyper-parameters. The paper further reports that the proposed model excels at "masked language modeling, neural machine translation, and long document modeling" [30]. Moreover, the proposed model was experimented with over different dropout rates to understand the impact of the ResNet connection over the multi-head attention layer. The analysis showed promising results and outperformed on recent state-of-the-art transformer architectures like BERT-base, ADMIN, etc.

A neural network tends to work well with deeper connections, which generates better performance overall. It is empirically proven that the depth of neural networks will result in better gradient convergence and lead to more accuracy overtime [9]. But a common problem that persists in deep neural networks is the vanishing gradient. To address this vanishing gradient problem, a common solution is a "temporal shortcut path" to avoid exploding or vanishing gradients in the "temporal domain" [12]. These connections, in general, are termed skip connections or shortcut connections in neural networks. For transformer architecture models, ResNets [10] connections are important to maintain the original input without extreme distortion of data. Furthermore, the mechanism behind residual connection is a simple function where original input is annexed with the output of a layer. By addition of the original input with the output, we are able to preserve information. On the other hand, highway networks [9] are another type of skip connection. The implementation of this shortcut path involves a gating mechanism. Just like ResNets, highway networks also add original input to the output from a layer or node. But before adding, the output is multiplied by a transformed matrix of the input. According to study, "depending on the output of the transform gates, a highway layer can smoothly vary its behavior between that of a plain layer and that of a layer that simply passes

its inputs through" [9]. The transformation function T is simply a linear projection passed into an activation function. This transformation function T acts as the gating mechanism. Furthermore, "various activation functions that may be more suitable for particular problems can be used in deep highway networks" [9]. Compared to ResNet, [12] report that "the highway path of a highway network is not always turned on." For instance, if T(X) = 1, then the model can simply ignore the highway input, or if T(X) = 0, then the model can bypass an output layer altogether.

Parameters and memory requirements for training complex and large language models like transformers are high and expensive. Utilization of virtual memory is necessary for transformer models due to the architectural design and use of attention mechanisms. A survey paper reported, "A well-known concern with self-attention is the quadratic time and memory complexity" [37]. Time and memory are valuable computational resources when it comes to generating attention scores. Furthermore, "The efficiency of a model can be interpreted in a variety of ways" [37] like computation, memory, learnable parameters, training and inferencing, etc. Firstly, the input is represented in chunks of batches with a dimensionality of "$R^B X R^N$" [37]. Later, "the input word sequences are converted to one-hot token, increasing the dimensionality of the matrices to $R^B x R^N x R^D$" [37]. Longer sequences of words will result in larger-sized matrices. Although longer sequences prove to show better results, they are often limited by computational resources. Moreover, "The inputs and outputs of the multi-headed self-attention module are connected by residual connectors and a layer normalization layer" [37]. Information flow from previous layers to consecutive layers will also cost us memory. "Transformers can primarily be used in three ways" [37], primarily the most resource-intensive way of usage is the utilization of both encoder-decoder cross-attention architecture. The survey paper provides an in-depth history of how the transformer models have improved over time. Among a few of the improvements, there is the "Memory Compressed Transformer" [37], which used two main approaches: "localizing the attention span and using memory-compressed attention" [37]. Other methods involved the implementation of custom computational units, such as "custom GPU kernels" [37]. Unfortunately, these custom hardware solutions are not very versatile and can be a problem when it comes to the scalability of the model. A well-known optimized model of the transformer is the transformer-XL, which utilizes the concept of recurrence like RNNs. As reported in the paper, "Segment-based recurrence can be considered an orthogonal approach to the other techniques discussed since it does not explicitly sparsify the dense self-attention matrix" [37].

Chapter 3

Description of the model

3.1 Introduction

Realformer model focuses on long term dependency performance for seq2seq models and it showed promising results compared to other transformer models. Another closely related model, Highway Transformer focuses on minimizing the cost of improving long term dependency performance by improving the models efficiency. These two architectures can be combined together to potentially create a model that excels at performing in long term dependent tasks while also efficiently utilizing the potential increase in parameters and memory requirements. Residual attention from RealFormer allows it to influence the attention scores of the current attention layer by retaining the attention score from the previous layer. In order to enable attention scores of previous layers to bypass certain layers, we incorporated Highway mechanisms that allow efficient information flow across the layers.

Our models are built on the idea that each attention layer focuses on different dependencies. The immediate previous layer may focus on short-term dependencies while the attention score from two layers back may focus on long-term dependencies. Using information from both attention scores would give a more nuanced value of attention over time. However, it is also important to consider the importance of the previous layer scores and the impact of those scores in the current layer attention scores. Thus, in order to achieve a dynamic control over this impact, a highway mechanism is incorporated. A highway mechanism introduces a gate through which we pass the attention score information from the previous layers based on their importance. It allows us to be able to regulate the information flow from the previous layers.

In regions of the input sequence where the current attention scores are more important than those of the previous layers. The current layer may not require additional refinement from earlier attention scores. In this case, the gate will **close**, allowing only the most recent attention scores to be processed.

In regions of the input sequence that are dependent on distant tokens, the gate will **open wider**, allowing attention scores from the previous layers to contribute to the refinement of the current attention scores.

3.2 Dimension Analysis for Base Transformer Model and RealFormer Model

In this section, the mathematical model of RealFormer is explained. Since the RealFormer model doesn't use any gated mechanisms the number of parameters remains the same. To prove it, we must first understand the dimensionality of the multihead block of a base transformer. Let X be the input after positional embedding, with dimensions [sequence_length $\times D_{\text{model}}$] and each weight matrix for Q, K, and V has dimensions $[D_{\text{model}} \times d_k]$, where sequence_length is the input sequence length, D_{model} is the number of hidden representation size, and d_k is the .

$$X \cdot W_i^Q = Q_i \quad \text{for one head of Query,}$$

This is similar for K and V matrices as well. So the dimensions for Q, K, V can be said: [sequence_length $\times D_{\text{model}}] \cdot [D_{\text{model}} \times d_k]$ = [sequence_length $\times d_k$]. Thus, we can calculate the attention scores for the base transformer model as:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{3.1}$$

For the RealFormer model as:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{prev}_{(n-1)}\right) \cdot V \tag{3.2}$$

where:

$$\text{prev}_{(n-1)} \text{ is the } \frac{QK^T}{\sqrt{d_k}} \text{ of the previous layer.}$$

Thus, the dimensionality of $\frac{QK^T}{\sqrt{d_k}}$ is: [sequence_length $\times d_k] \cdot [d_k \times$ sequence_length] = [sequence_length $\times$ sequence_length].

3.3 Our Contribution

With the help of RealFormer model and existing gated mechanism works in Transformer models, we came up with three approaches.

3.3.1 Approach 1: 2 Layer Deep Extended RealFormer

Our first approach focuses on extending the existing RealFormer model. It relies on the concept of using previous attention layer scores to influence the long-term dependencies captured by the model. Theoretically, extending the RealFormer model would allow the model to handle complex sentence structures or tasks that require long term dependencies such as machine translation or summarization better than the existing model. Therefore, the attention score for a specific layer will be calculated as follows:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{prev}_{(n-1)} + \text{prev}_{(n-2)}\right) \cdot V \tag{3.3}$$

Here, $n-1$ and $n-2$ refer to the previous two layers of the transformer.

3.3.2 Approach 2: 3 Layer Deep Extended RealFormer

In order to compare the feasibility and the tradeoffs that come with the extension of the model, our second approach uses the previous 3 attention layer scores to calculate current attention scores. Theoretically, approach 2 should perform better than approach 1. However, it should also be noted that any extension would result in increased operations thus leading to increased computational complexity. This added complexity may also increase the training time for the model. Thus, the primary objective of these two approaches is to understand if the model is feasible despite these drawbacks. The attention score for this approach can be calculated by the following:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \text{prev}_{(n-1)} + \text{prev}_{(n-2)} + \text{prev}_{(n-3)}\right) \cdot V \tag{3.4}$$

Here, $n-1$, $n-2$, and $n-3$ refer to the previous three layers of the transformer, respectively.

3.3.3 Approach 3: Extended RealFormer With Gated Mechanism Like SDU from Highway Transformer

Our third approach introduces a gating mechanism inspired by "information highways" where it allows the neural networks to have paths along which information can flow across layers [9]. We use a tanh gating function that will selectively cancel out any information that has less importance for the current input region from the previous layer and add it to the current layer. Introducing a gating mechanism also increases the complexity due to the increase in parameters. The parameter increase is due to the self-gating mechanism which uses its own weights that also requires training. Training requires operations and memory, increasing the complexity of the model, therefore, increasing training time. The gating function to be used in our model is as follows:

$$T(\text{prev}_{(n-1)}) = \tanh(\text{prev}_{(n-1)} \cdot W + B) \tag{3.5}$$

Thus, the output that will be passed to the next layer is,

$$\text{Gate} = T'(\text{prev}_{(n-1)}) = \text{prev}_{(n-1)} \odot T(\text{prev}_{(n-1)}) \tag{3.6}$$

where $\odot$ is the symbol for element-wise multiplication. Therefore, the attention score for the current layer can be calculated as follows:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + T'(\text{prev}_{(n-1)})\right) \cdot V \tag{3.7}$$

where T is the transformed matrix after passing through the tanh activation gate. We can further extend this idea and implement it for approach 1 and 2 as well with the following equations:
For approach 1:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + T'(\text{prev}_{(n-1)} + \text{prev}_{(n-2)})\right) \cdot V \tag{3.8}$$

Or, for approach 2:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + T'(\text{prev}_{(n-1)} + \text{prev}_{(n-2)} + \text{prev}_{(n-3)})\right) \cdot V \quad (3.9)$$

3.3.4 Visualizing the models

We can visualize the model as follows:

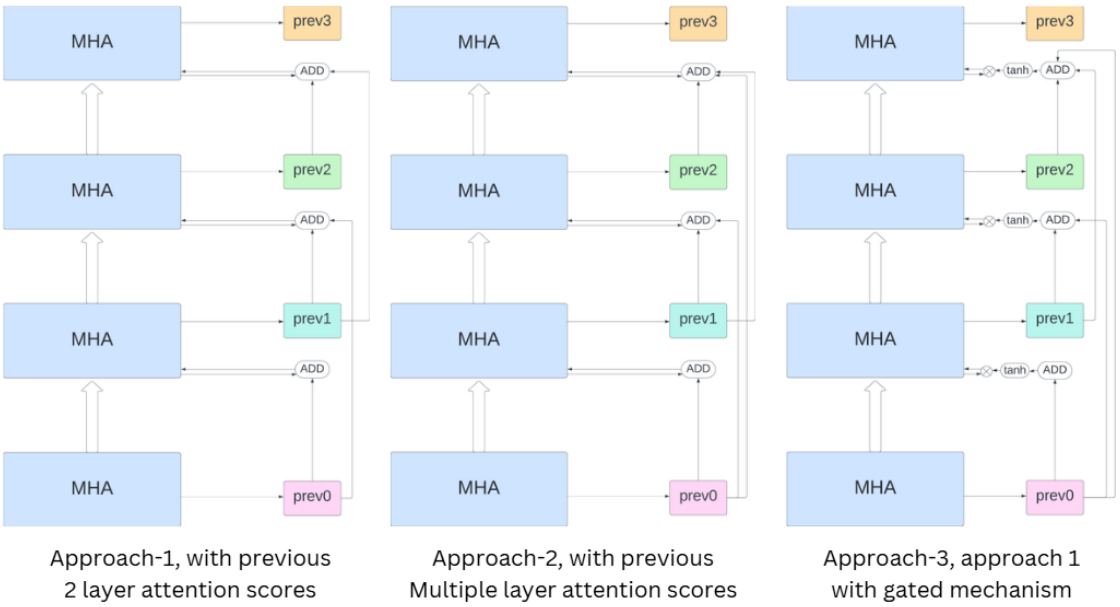

Figure 3.1: Visualization of Novel Transformer Models

Chapter 4

Description of the data

Our primary objective is to develop a new architecture for the transformer model to increase its efficiency as well as performance without sacrificing computational resources. Therefore, training and evaluation of the model are key to understanding the performance and efficiency of said model. As a result, this paper selected machine translation dataset English-to-Portuguese for benchmark purposes. For instance, organized by the Conference on Machine Translation (WMT), the WMT14 dataset for the English-to-French (fr-en) dataset as well as the English-to-German (de-en) dataset were selected by the original transformer paper [5]. These datasets provide a standard format for evaluating, training, and testing the performance of a neural network machine focusing primarily on translation and generation from one language to another. From observation of the original transformer paper, "the model achieved 28.4 BLEU on the WMT 2014 English-to-German translation task and on the WMT 2014 English-to-French translation task, which achieved a single-model state-of-the-art BLEU score of 41.8" [13]. Furthermore, just as the canonical transformer, the RealFormer paper [30] also evaluated their model using the WMT14 machine translation dataset [5]. Providing this paper the opportunity to compare the new model verses the canonical transformer and the RealFormer using a machine translation dataset English-to-Portuguese. Moreover, consistency among the three models must be maintained, and so all data pre-processing will be done following the same techniques. As for the evaluation matrices, both papers evaluated their results based on the bilingual evaluation understudy (BLEU) standards, specifically by compressing n-grams (usually up to 4 grams) in the translated output [2].

The language-pair dataset WMT14 English-to-German contains approximately 4.5 million sentence pairs, providing vast information regarding the two languages. On the other hand, the WMT14 English-to-French dataset contains sentence pairs around 40 million [5]. The source of information comes primarily from news articles, parliamentary proceedings, and web crawls. A sort of bias is involved when using these sets due to the dataset focusing only on news articles and web crawls because spoken German and English is not always similar to the formal style of news articles. None the less, it still provides an excellent magnitude of information regarding the languages, their structure, and alignments, along with a vast size of vocabulary ranging around 1 million unique German words from the WMT14 English-to-German dataset and around 2 million unique French words from the WMT14 English-to-French dataset. Although it should be kept into consideration that these are the

raw vocabulary size and can differ from the training data after data pre-processing. Typically, for large datasets, training data contains a vocabulary size limit of 32,000 or more [32]. Due to lack of computational resources we are unable to use large dataset for our models evaluation and so we resort to a smaller dataset which is the English-to-Portuguese (IWSLT) dataset [6].

For data analysis, we are incorporating the smaller dataset, that is, the International Conference on Spoken Language Translation (IWSLT) English to Portuguese dataset [6], due to a lack of computation resources. Compared to the WMT14 English to German dataset and the WMT14 English to French dataset, the IWSLT TED-HRLr English to Portuguese (pt-en) dataset is much smaller. Which, of course, means that training our tokenizer as well as the model will take much less time. As a result, the IWSLT TED-HRLr English to Portuguese dataset is selected for preliminary analysis of the models. As for the dataset, there are a total of 52978 sentence pairs. The source of the dataset comes from the TED Talk translation dataset, used in IWSLT evaluation campaigns, which is from the transcriptions and translations of TED Talks. These talks are public speeches on various topics, given by experts and thought leaders, which have been transcribed and translated into multiple languages by volunteers and professional translators.

4.1 Data analysis

The Conference on Spoken Language Translation (IWSLT) English to Portuguese dataset contains a total of 52978 sentence pairs. Out of which sentences longer than 50 words were discarded. This was done to keep the training data manageable and to avoid performance issues with very long sentences. After discarding, the new dataset contains 51965 sentence pairs. Upon further analysis, English average sentence length is 14.74, with a median sentence length of 12. Portuguese average sentence length is 13.77, with a median sentence length of 11. English language vocabulary size is equal to 27640, while Portuguese vocabulary size is equal to 39468.

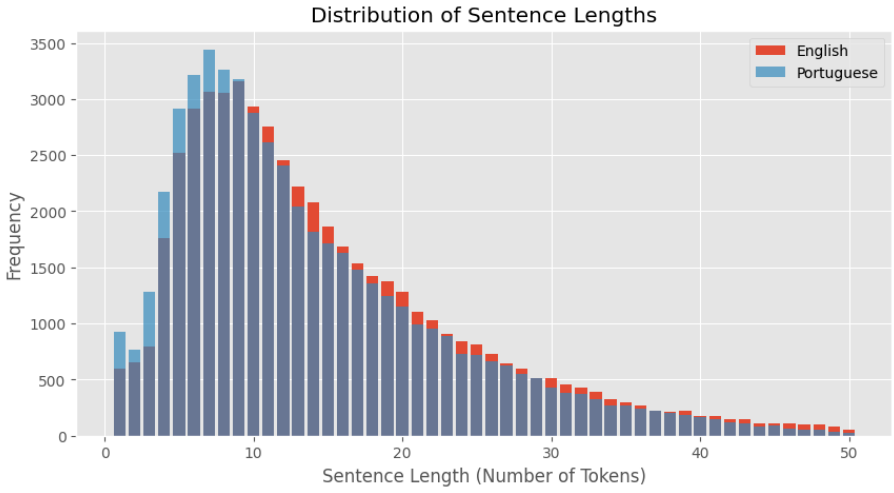

Figure 4.1: Sentence Length Distribution

Comparing the most common words in the Portuguese and English word frequency distribution. We can observe that high-frequency words correspond to each other. This is in fact expected since the translation of frequent words will also be similar. For example, in Portuguese, ”a” is the feminine singular definite article, which is the most frequent word, whereas in English, ”the” is the most common. Both are articles, but in the English language there is only one definite article (”the”), whereas Portuguese has gender-specific articles, “o” for masculine and “a” for feminine, resulting in a number of high-frequency forms. When considering conjunctions, the English language “and” also corresponds to “e” in Portuguese, which means and in Portuguese. Prepositions such as ”de” (of) and ”para” (for) are used frequently in both languages. For that, they correspond to each other as expected. This demonstrates a shared linguistic pattern in the use of prepositions. The Portuguese list places less emphasis on pronouns because the language is pro-drop, with subject pronouns frequently omitted. As a result, we see the use of pronouns in English to be higher than the Portuguese translation.

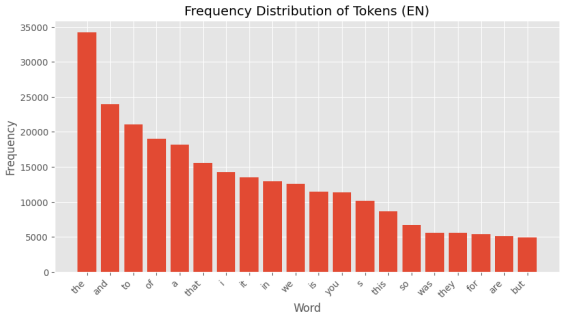

Figure 4.2: Frequency Distribution (EN)

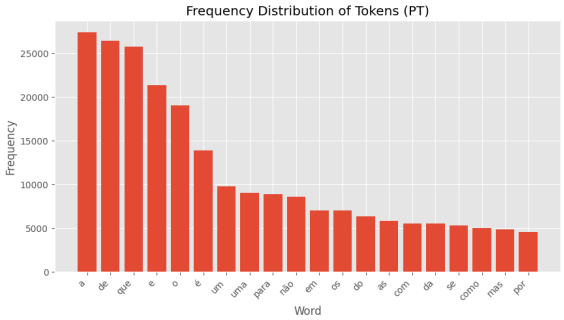

Figure 4.3: Frequency Distribution (PT)

Chapter 5

Methodology

This section of the paper will describe the methods and setup configuration for training. The Conference on Spoken Language Translation (IWSLT) English to Portuguese datasets was used for data pre-processing, as well as model training and evaluation. And later extend this analysis over other models along with our novel architecture transformer variant that we have constructed for our machine translation task.

5.1 Data Prepossessing

5.1.1 Tokenization

Point to be noted: the vocabulary size of the dataset is very big and can cause performance issues when training. As a result, Byte Pair Encoding (BPE) [11] was implemented by the original transformer paper for tokenization purposes so that the vocabulary size can be maintained in a lower range, such as 50,000 or less. For our data tokenization, this paper will implement subword tokenization introduced by Google, which is also known as SentencePiece [15]. The Portuguese dataset contains a very small number of vocabulary words; hence, only 8500 top frequent words were selected as the vocabulary size of the tokenizer. This will provide us the opportunity to fully explore the dataset without having to bear high memory constaints and train the model with lower parameters.

5.1.2 Train-Validation Split

Subword tokenization was done only on the training dataset. The training dataset of English to Portuguese contained approximately 97.5% of sentence pairs from the main dataset, which is 51785 sentence pairs. And the remaining 1193 sentence pairs were set as the validation set. Note that the split was done after data was shuffled. The maximum length of tokenized English sentences came about 72 units whereas the maximum length of tokenized Portuguese sentences was 75 units. But the mean value was much lower around 22.7 units. Transformer models tend to increase the training time complexity quadratically as the sentence length increases. The typical attention mechanism in transformers has a time complexity that scales quadratically with token length [47]. And so, to reduce the computation cost we

dropped sentences with word length longer than 35 words and set the token length of the input or sequence length to be 64.

5.2 Model Implementation

The model that we have implemented is the Transformer-based sequence-to-sequence translation model designed for machine translation. The dataset used for our model is the IWSLT TED-HRLr Portuguese-to-English (pt-en) dataset [6] which is widely used for machine translation. We have used a pre-tokenized dataset to speed up the training process as it avoids real-time tokenization and it also allows us to ensure consistency and efficient data pipelines during training. In order to handle the diverse vocabulary in these languages, we used a sub-word tokenizer, SentencePiece [15]. Using sub-word tokenization allows us to handle unknown words by breaking them down into smaller and more meaningful units. It also allows us to compress the vocabulary size, as uncommon words have little impact on increasing the vocabulary size.

5.2.1 Positional Encoding

Positional encoding is used to allow transformer models to gain the information about the position of each token in a sequence. Our model uses a custom positional encoding which is calculated by applying sine and cosine functions to the position indices [13], thus, allowing the model to gain positional context for each token. We have incorporated positional encoding directly into the token embedding layer as it enables the model to learn the embeddings and positional encoding jointly. The following is the function used for the positional encoding:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

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

Where:

- pos is the position of the word in the sequence.
- i is the dimension index.
- d is the dimensionality of the model (embedding size).

5.2.2 Encoder and Decoder

All the models utilized for this study almost have similar structure except for the attention block. The encoder and decoder of the transformer model and its variants follow the same principle as shows in the original paper [13]. Simply the encoders are stacked upon each other and then the output is passed to the decoder layer where the decoder is also stacked upon each other. Now each of the encoder and decoder layers are looped through when training. Extracting results after each loop and passing the information to the next encoder/decoder became easy. Therefore,

this gave the perfect opportunity to add the residual connection between the encoder/decoder layers. Note, creating residual connection with gating mechanism mean that the computational unit must retain the information in memory like in the virtual memory of a GPU. This will mean that the model parameters compared to a base transformer is likely to increase. Further in the study we will discuss broadly regarding the change in parameter count.

5.3 Training

The model created tensors of batch size equal to 64. Set the maximum length of a sentence in terms of words to be 64 as well. The model was trained using the following hyper-parameters:

- Batch Size (`BATCH`) = 128
- Input Sequence Length (`MAX_LENGTH`) = 64
- Number of Layers (`num_layers`) = 4 | 6
- Model Dimension (`d_model`) = 512
- Feed-Forward Dimension (`dff`) = 2048
- Number of Attention Heads (`num_heads`) = 6 | 8
- Dropout Rate (`dropout_rate`) = 0.1
- Number of Epochs (`epochs`) = 16

Batch size indicates that the training data and validation data were sliced into chunks, each of size 128 sentence pairs. Input sequence length indicates that each sentence was tokenized to have a max token length of 64. Sentences less than that size of 64 were padded as necessary. Since this is a sequence-to-sequence transformer model, there exist both encoders and decoders. The encoder block consists of 4 encoder layers, and the decoder block consists of 4 decoder layers. For further experimentation the number of layer was changed to 6 to observe the changes in the models behavior when the number of encoder/decoder increased. The embedding size of the model sequences is 512. The feed-forward dimension is 2048. The number of attention heads calculated in each multi-headed attention block inside each encoder and decoder layer is 6. Similar to the number of layers, the number of heads were also increased to 8 to observe any changes in the models behavior. The dropout rate was set to 10% which indicates that 10% of the nodes will be turned off, forcing the other nodes to learn more context. Finally, the number of epochs was set to 16 which means that the model will run max up-to 16 times on the same training data. Furthermore, for each of the model we setup patience level to be 2 focusing on the validation loss of the model.
Due to a lack of computation resources, we were unable to run bigger transformer models with higher hyper-parameters. For the implemented base transformer models, the selected platform for training was the Google Colab Pro using the L4 GPU. The GPU hardware, allowed us to utilize 23 Gb of virtual memeory to fit our model.

5.4 Evaluation Metrics

For our experiment, this study trained a total of 15 models. The models are compared based on their validation loss and validation accuracy as well as the BLEU score.

5.4.1 Validation loss and Masked Accuracy

Two functions were implemented in our training code, one for calculating the validation loss after each epoch and another for calculating the masked accuracy after each epoch. These functions are suitable for sequential data, for instance, our machine translation task. The function ignores padded tokens and computes the loss and accuracy based on the true labels. Finally average to find the proper loss and accuracy generated. The loss is being calculated using the sparse categorical cross-entropy loss function implemented by the TensorFlow Keras library [52]. Similarly, in masked accuracy, the predicted values are compared to the true labels. Here also only the non-padded tokens are considered.

5.4.2 BLEU Score

When it comes to the sequence-to-sequence machine translation task, Bilingual Evaluation Understudy scores are often used to evaluate the translation from one language to another. A BLEU implementor's main programming task is to compare the candidate's n-grams to the reference translation and monitor the number of matches. These matches are position-independent. The more matches, the more accurate the candidate translation. To keep things simple, we'll start by computing unigram matches [2]. The original transformer paper [13] as well as the Realformer paper [30] used the BLEU score evaluation metric to compare the results of the generated output. Following their footsteps, this paper also utilized the BLEU score to compare how well the transformer model translates Portuguese text to English text.

5.4.3 Efficiency Test

Throughout the experiment, each of the models was trained on the exact hardware as well as the exact hyperparameters on the same data. The only difference would be the actual architecture of the models. This study empirically tests the training time as well as the inference time. If a model trains faster than a baseline model without losing too much performance, then it would mean that the new variant is more efficient when it comes to training. But training efficiency is not the same as the inference time. Inference time represents the time required for a model to translate a given sequence of inputs. For our experiment, we calculated the inference time on 500 sentences, each tokenized the same manner as the training dataset. By comparing the inference time, we can conclude if the model is fit for real-world application and deployment. Whereas training efficiency is equally important as the performance of a model and can prove the variant to be better. This will effeminately help contribute to the future endeavor of NLP in computer science.

5.4.4 Sparsity Comparison

Sparsity is a key concept used when training deep networks, as it increases the efficiency of a model and utilizes less memory. It is demonstrated that the sparsity of residual blocks serves as an implicit gating mechanism for deep residual networks, avoiding the exploding or vanishing gradient problem, which is known to contribute to training difficulty [21]. This finding underscores the role of sparsity in facilitating the training of deep residual networks by mitigating issues related to gradient propagation. For our experiment we calculated how many parameters in our model are equal to or close to zero value.

Chapter 6

Experimental Analysis and Evaluation

This section of the paper highlights all the observations from the experiment. To fully understand the process and results of our observation, we need to clarify certain aspects of the training first. Below are multiple subsections on different observations filled with graphs and tables for better visualization. We primarily trained our model using the same hyper-parameters explained in our methodology section. Here we observed models' behavior across variation in number of layers as well as variation in number of heads. This provides us an insight into how the model will behave as it scales. Furthermore, the BLEU score generated by each model and the sparsity of each model are monitored. Efficiency was also calculated based on the training time, i.e., how long it took for the model to converge and saturate. Moreover, the inference time was also monitored to understand how the model would perform in real-time applications. Furthermore, to clarify the naming convention we used to report on our observation is presented here. T stands for $Transformer$ which is the vanilla baseline transformer model. R stands for the $RealFormer$ model. M stands for $Multi-Layered\ RealFormer$ where instead of working with a single previous attention score the model implements multiple previous attention scores. And so, $M2$ and $M3$ represents $Multi-Layered\ RealFormer$ with 2 previous attention scores and $Multi-Layered\ RealFormer$ with 3 previous layered attention scores respectively. Finally, we have our novel transformer variant $G1$, $G2$ and $G3$ representing the same as $Multi-Layered\ RealFormer$ with the gating mechanism. We call this model $Gated-ResNet\ Transformer$ model. For instance, $G1_E4_H8$ would mean a $Gated-ResNet$ with single ResNet connection trained with 4 layers and 8 heads.

6.1 Loss and Accuracy

In this section of the paper we will observe the validation masked accuracy as well as the validation loss incurred by the model through the entire training of the model. For clarity of the graphs, we have divided the models into 4 sections. Each section representing a specific model and how it performed with variable hyper-parameters. We can see the accuracy and loss trends for both training and validation dataset with different numbers of encoder/decoder layers and attention heads. The variation only involved increasing the number of layers while keeping the head the same and vice versa. For each of the graphs on the x-axis, we show the number of epochs the model trained for, and on the y-axis, we show the accuracy or loss generated from the 1st epoch to the last epoch. Different color combinations were used by the graphs for better identification of the graphs. Furthermore, the graph also includes not just the validation loss/accuracy but the training loss and accuracy as well.

6.1.1 Transformer with variable Hyper-parameters

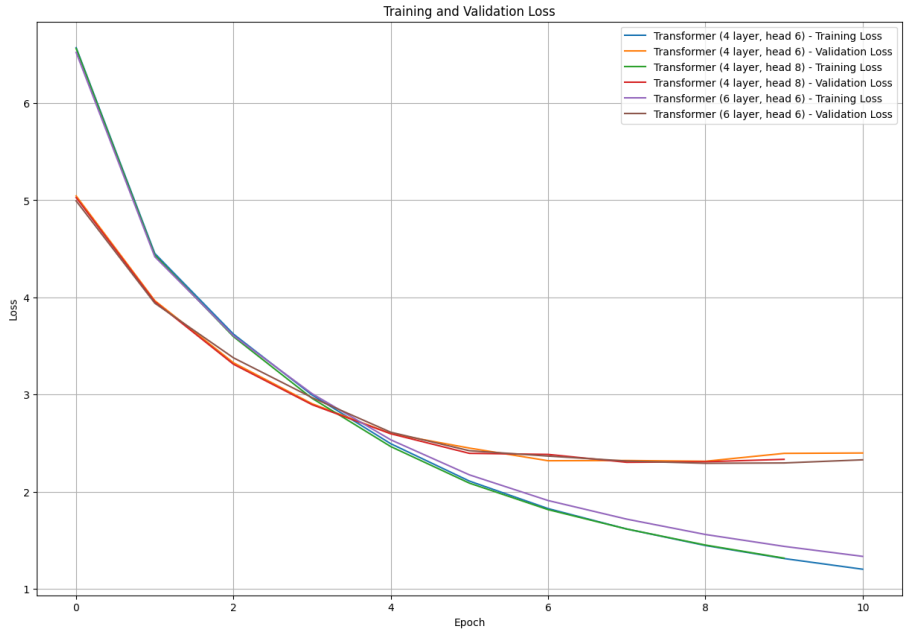

Figure 6.1: Transformer Loss Curve

From the above graph we can conclude that the transformer model is showing a smooth transition from epoch to epoch. The transformer model with 4 encoders and 6 heads, as well as the 4 encoders and 8 heads combination, shows a lower loss generated compared to the transformer model with the 6 encoders and 6 heads combination. All ran a full length of 10 epochs.

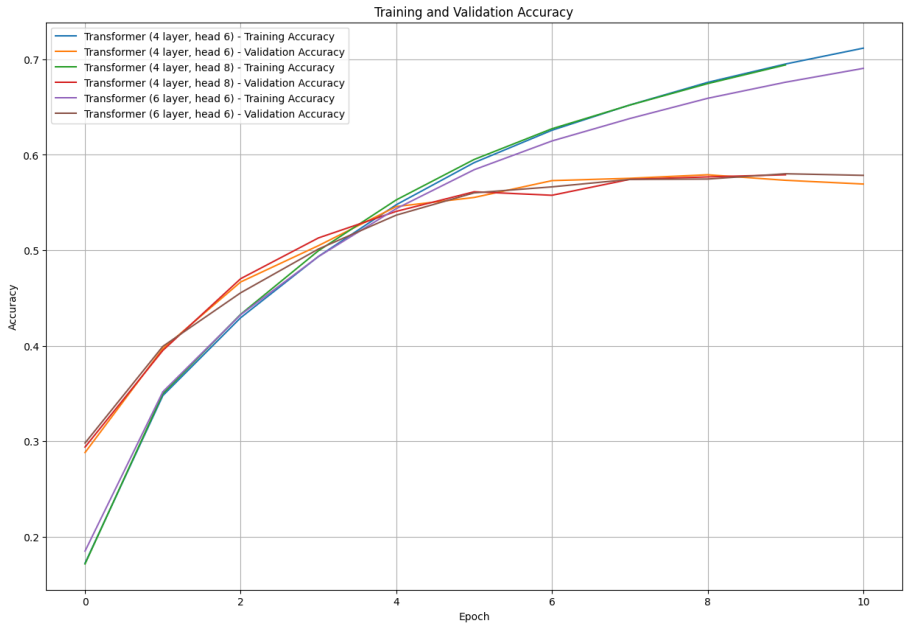

Figure 6.2: Transformer Accuracy Curve

The graph indicates that the model with 4 layers and 8 heads reaches its peak at the 9th epoch compared to the other two models, 4 layers, 6 heads and 6 layers, 6 heads which reaches their peak at the 10th epoch. Among all these models, 4 layers and 6 heads have the highest peak accuracy. Notably, all validation accuracies are closely aligned with each other and exhibit significantly lower peaks compared to their respective training accuracies.

6.1.2 RealFormer with variable Hyper-parameters

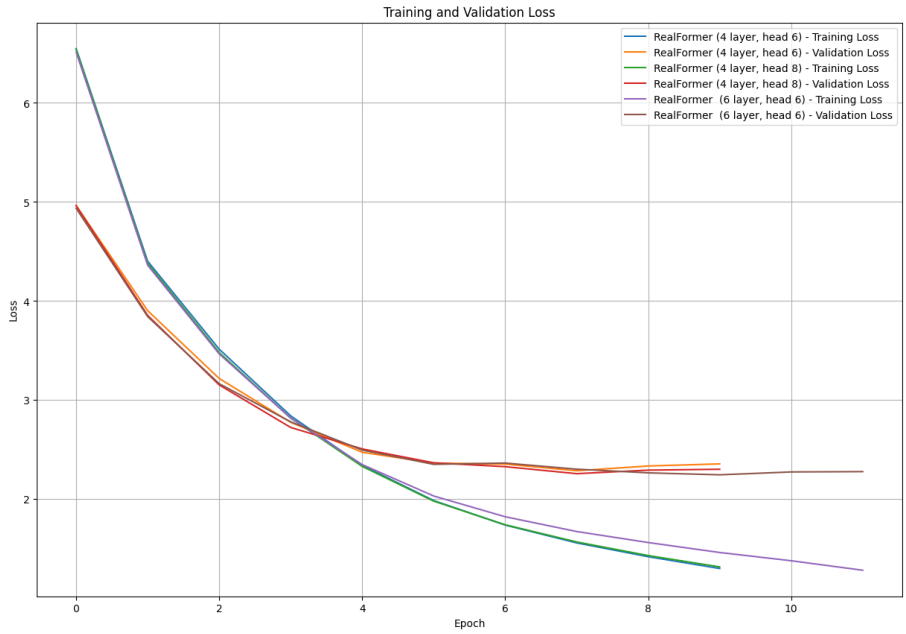

Figure 6.3: RealFormer Loss Curve

The RealFormer models with 4 layers and 6 heads reach their minimum loss at the 9th epoch, whereas, the model with 6 layers and 6 heads achieve their minimum at the 11th epoch. The same trend is also observed for validation losses. The difference in minimum training loss for the configurations are minimal.

The training accuracy for the configurations with 4 layers, 6 heads and 4 layers,

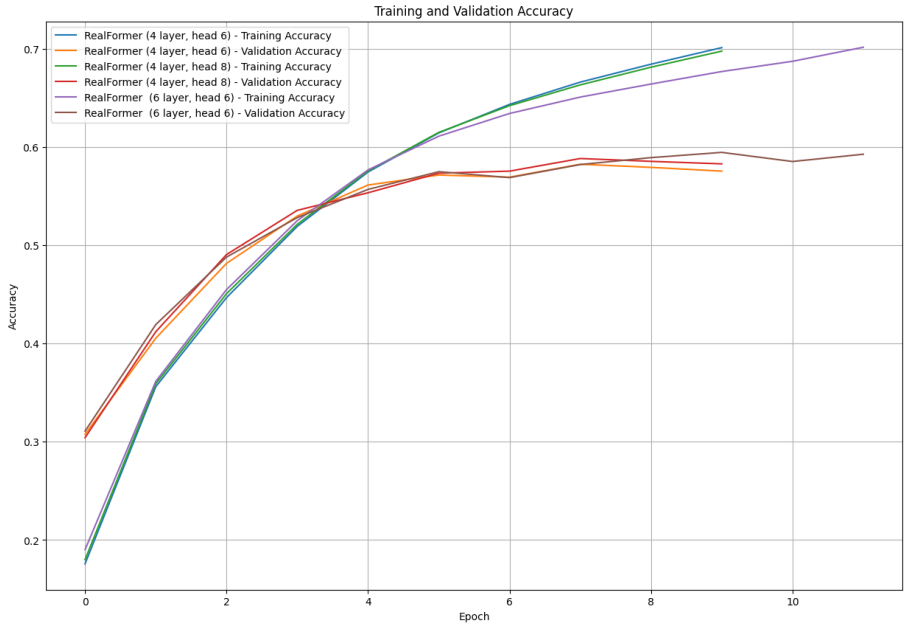

Figure 6.4: RealFormer Accuracy Curve

8 heads reaches their peak at the 9th epoch whereas the model with 6 layers and 6 heads reaches peak training accuracy at the 11th epoch. The same pattern is observed for validation accuracy. The peak training accuracy for all configurations is almost equal.

6.1.3 Multi Layered Realformer with variable Hyper-parameters

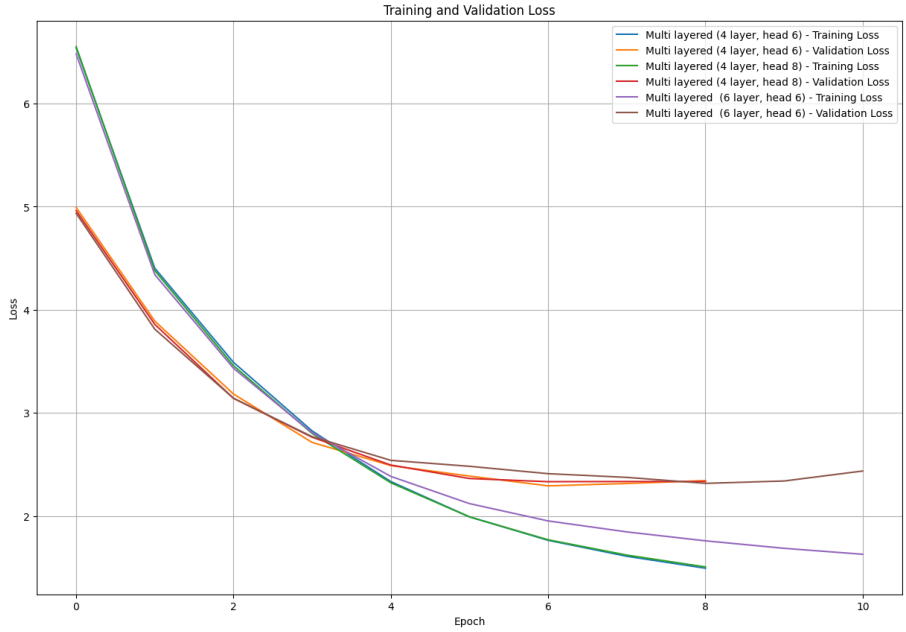

Figure 6.5: Multi Layered RealFormer Loss Curve

In this diagram, we compare our multi layered model training loss with different configurations for the number of layers and attention heads. For training accuracy, models with configurations of 4 layers, 6 heads and 4 layers, 8 heads both show almost exactly the same trend reaching almost the same minimum loss at the 8th epoch. The model with 6 layers and 6 heads takes 10 epochs to reach its minimum even though its minimum is greater than the other two models. This model started deviating from the trend at around the 4th epoch as its magnitude of the gradient decreased compared to other models.

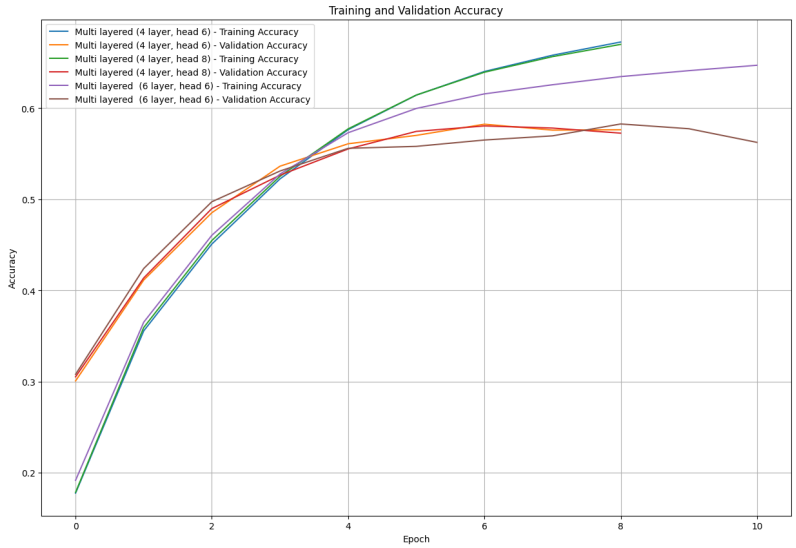

Figure 6.6: Multi Layered RealFormer Accuracy Curve

In the above graph, it is evident that the training accuracy for the multi layered model with 4 layers and 6 heads has the highest peak accuracy, closely followed by

the model with 4 layers and 8 heads. Both of these models took 8 epochs to attain best weights. In contrast, the 6 layers and 6 heads configuration has noticeable lower accuracy while also taking 10 epochs to reach its peak. While the training accuracy in the first few epochs for each model followed a similar pattern, the model with 6 layers and 6 heads had less increment to its accuracy after the 4th epoch compared to the other two models. The validation accuracy for all of these models seem to follow a similar trend with a noticeable dip in accuracy between 4th and 8th epoch for the model with 6 layers and 6 heads.

6.1.4 Gated Multi Layered with variable Hyper-parameters

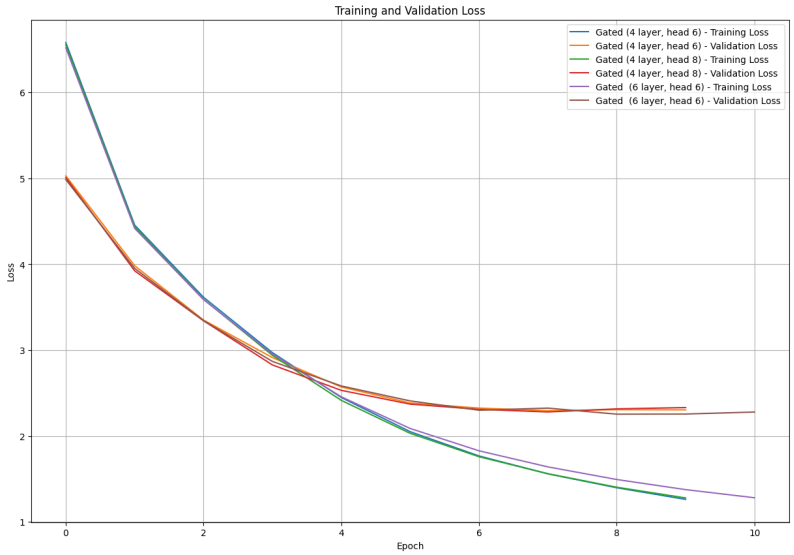

Figure 6.7: Gated ResNet Loss Curve

Here, we can compare the trend in training and validation loss for our gated model. The models with the configurations 4 layers, 6 heads and 4 layers, 8 heads show almost exactly the same trend reaching their minimum at the 9th epoch whereas the model with 6 layers and 6 heads have slower decrease in loss after the 5th epoch than the other two models and reaches its minimum at the 10th epoch.

For the graph below, we can see a similar trend where the models with 4 layers, 8 heads and 4 layers 6 heads follow almost the same pattern reaching their peak accuracy at the 9th epoch. In contrast, the model with 6 layers, 6 heads reaches its peak accuracy at the 10th epoch. Even though it took one more epoch to reach its peak, the peak accuracy for the latter model is lower than the former two models.

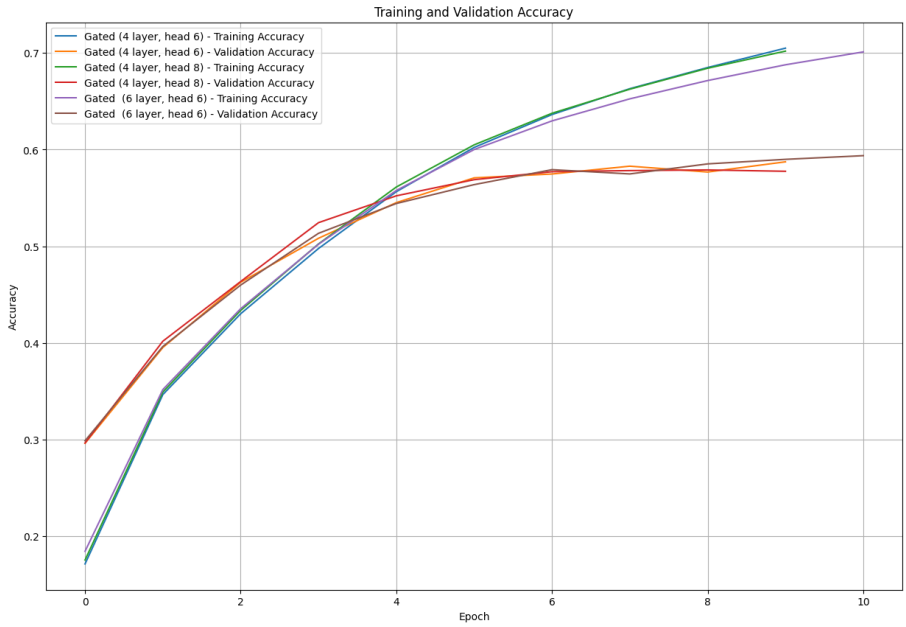

Figure 6.8: Gated ResNet Accuracy Curve

6.1.5 Loss and Accuracy Across 4 Encoder 6 Heads Combination

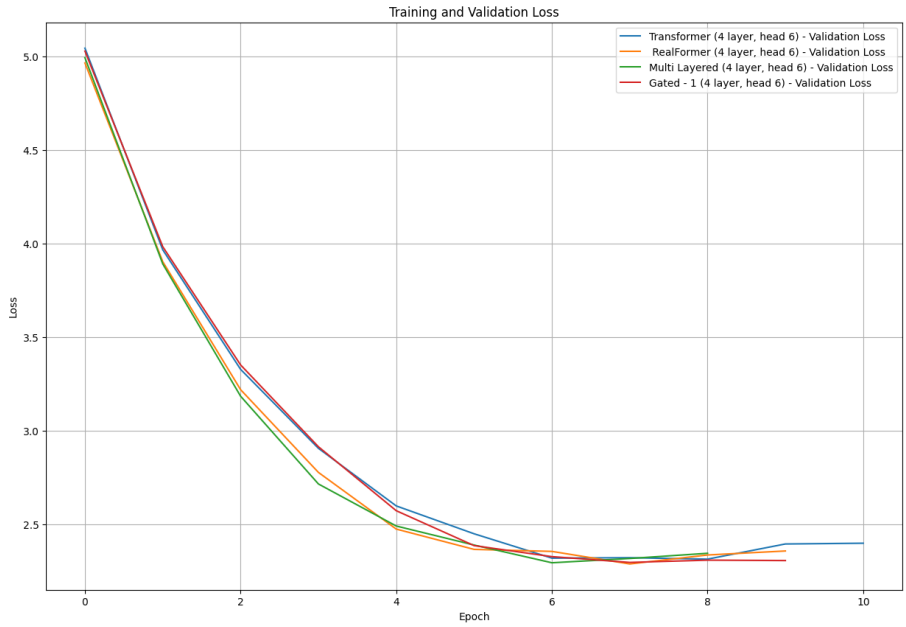

Figure 6.9: Loss Curve 4 Encoder 6 Heads Combination

In the graph, we compare the loss of the models with 4 layers and 6 heads. The RealFormer and Gated models both take 9 epochs to reach their minimum loss while the Multi layered model took 8 epochs and Transformer model took 10 epochs which is the most number of epochs. It can also be observed that the Gated model achieves the lowest loss among all the models, indicating its superior performance in this regard. Meanwhile, the Multilayered and RealFormer models exhibit nearly

identical loss values, showing comparable performance. In contrast, the Transformer model records the highest loss, suggesting it is the least effective of the models in minimizing loss during training and validation.

In the next graph, we evaluate the accuracy of all four models for 4 layers and 6 heads configuration. Over 9 epochs the RealFormer and Gated model reached best weights achieving peak accuracy, however, the peak accuracy for the Gated model is significantly higher than the other models. Even though the Multi Layered model reaches its peak accuracy at the 8th epoch, it's more than the regular Transformer model. We can also observe that the initial increase in accuracy for the RealFormer and Multilayered model has a higher gradient than the other two.

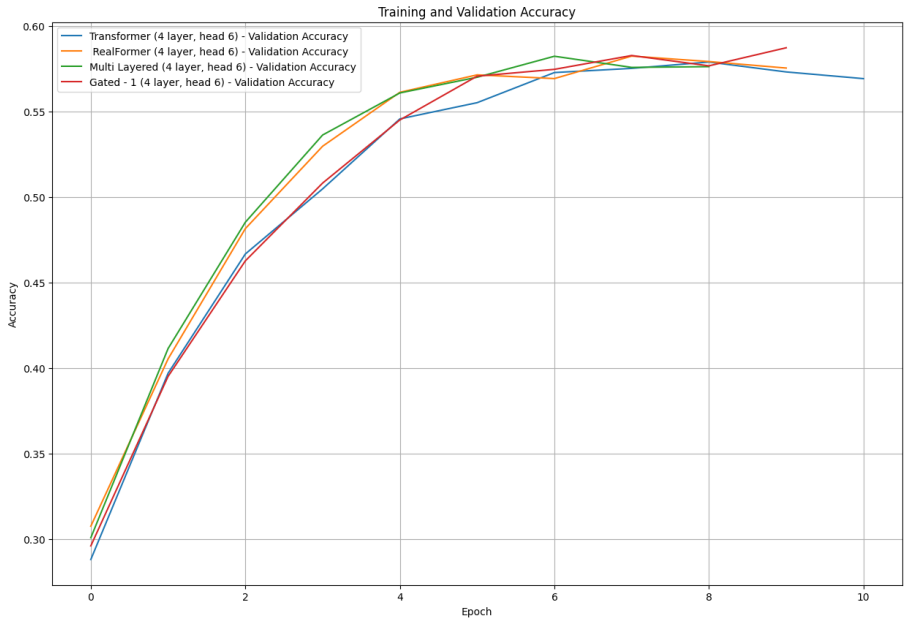

Figure 6.10: Accuracy Curve 4 Encoder 6 Heads Combination

6.1.6 Loss and Accuracy Across 4 Encoder 8 Heads Combination

As we can see in this graph 6.11, RealFormer has the lowest loss among all four models. The RealFormer and Multi Layered model decreases with almost the same gradient and the Transformer and Gated model also decreases with the same gradient but with a lower magnitude. Overall, all the models reach a similar loss, however, the Multi Layered model reaches its best weight at the 8th epoch while all the other models reach theirs at the 9th epoch.

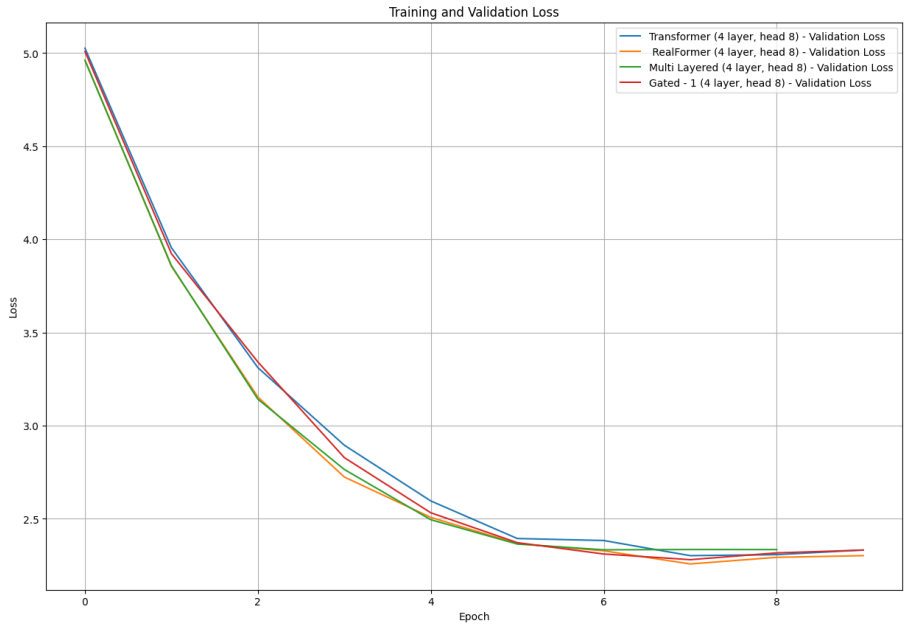

Figure 6.11: Loss curve 4 Encoder 8 Heads Combination

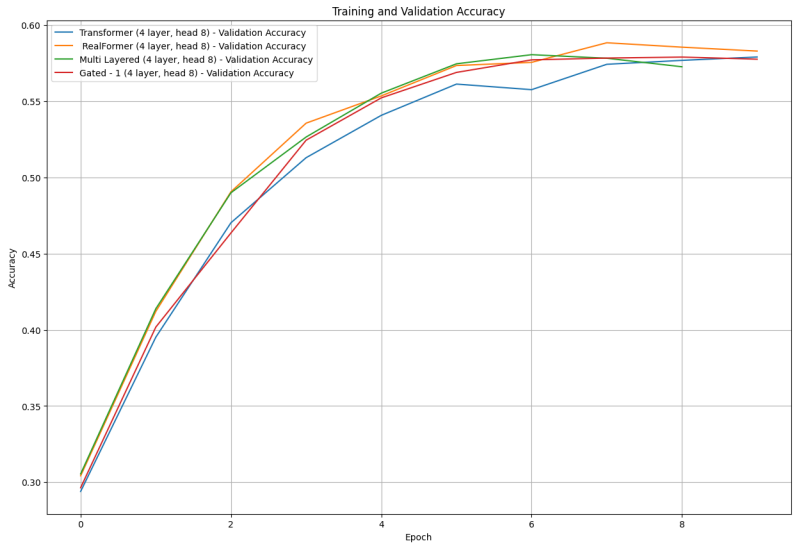

Figure 6.12: Accuracy curve 4 Encoder 8 Heads Combination

In this graph, we compare the models with 4 layers and 8 heads. We can see RealFormer and Multi Layered models had faster increase in accuracy initially until the 4th epoch where all models except regular Transformer model reached a similar accuracy. After the 4th epoch, those 3 models follow a similar pattern until the 7th epoch where the RealFormer model has a noticeably increased accuracy than the other models. At the 8th epoch, the Multi Layered, Transformer and Gated model have a similar accuracy, where the Multi Layered model reached its peak accuracy at the 6th epoch. Notably, the RealFormer had the highest accuracy out of all the models.

6.1.7 Loss and Accuracy Across 6 Encoder 6 Heads Combination

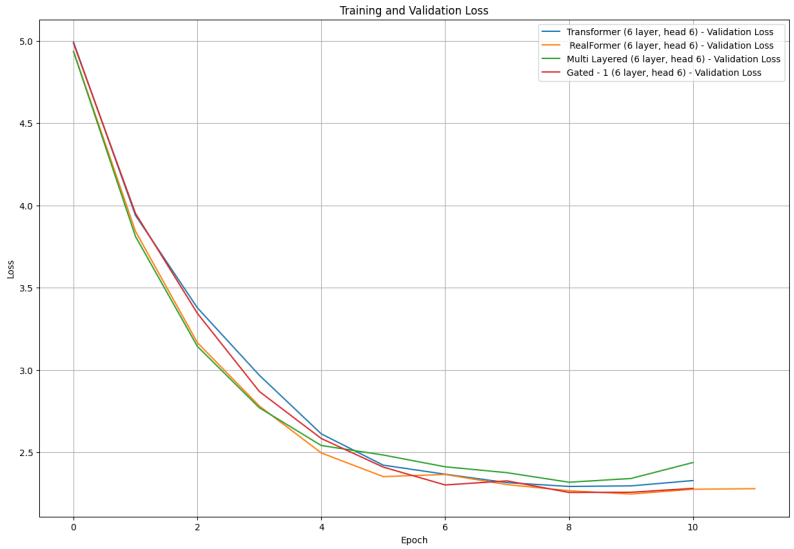

Figure 6.13: Loss curve 6 Encoder 6 Heads Combination

Moving on to the models with 2 more encoder/decoder layers, we have the configuration of 6 layers and 6 heads. Here, we can observe that the RealFormer and Multi Layered models have a higher magnitude of gradient compared to the other two. At the 8th epoch, the Multi Layered model reached its best weight even though its loss is the maximum out of all the models. Over the next two epochs, the Gated and RealFormer model had almost the same loss with the Transformer model having a little more than the other models. Overall, the Gated model reached the minimum loss at the 10th epoch whereas the RealFormer model reached almost the same minimum at the 11th epoch. In this graph, the accuracy for 6 layers and 6 heads configuration

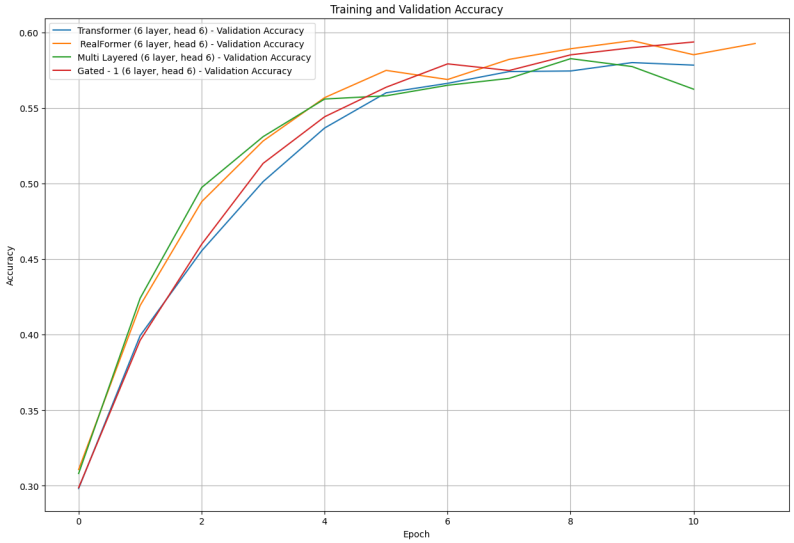

Figure 6.14: Accuracy curve 6 Encoder 6 Heads Combination

is compared. While the RealFormer model had a greater gradient than the Gated

model initially, both models reached a similar peak in accuracy. Like the previous configurations, the RealFormer and Multi Layered model still has an increased gradient than the other two. The Multi Layered model reached its peak accuracy at the 8th epoch whereas the Transformer model, despite having lower accuracy, reached its peak at the 9th epoch. Overall, the Gated model has the highest accuracy.

6.2 Bilingual Evaluation Understudy Score

6.2.1 BLEU Score vs Accuracy

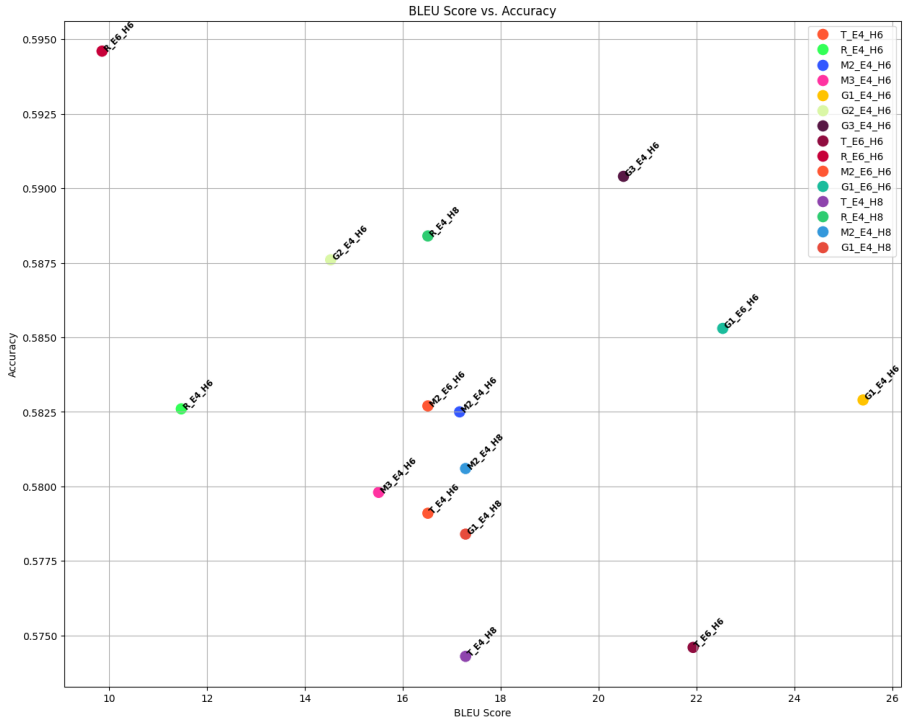

Figure 6.15: BLEU Score Vs Accuracy

Here we compare the accuracy and BLEU scores of all our models. We can see that the graph is spread throughout indicating no relationship between them. A relationship between these two metrics would cause the graph to follow a linear or non-linear pattern. It should be noted that there is one value that stands out, which is the Gated model with 1 ResNet connection, 4 layers and 6 heads. It has the highest BLEU score out of all the models even though it has an average accuracy.

6.2.2 BLEU Score vs Loss

When we compare the loss against the BLEU score, we can see that around 7 of our models have a similar BLEU score of around 17 despite having significant differences in their loss values. While the Gated model with 1 ResNet connection, 4 layers and 6 heads has the highest BLEU score, its loss value at the end of training is average compared to the other models. The graph 6.16 in the page will give proper insight.

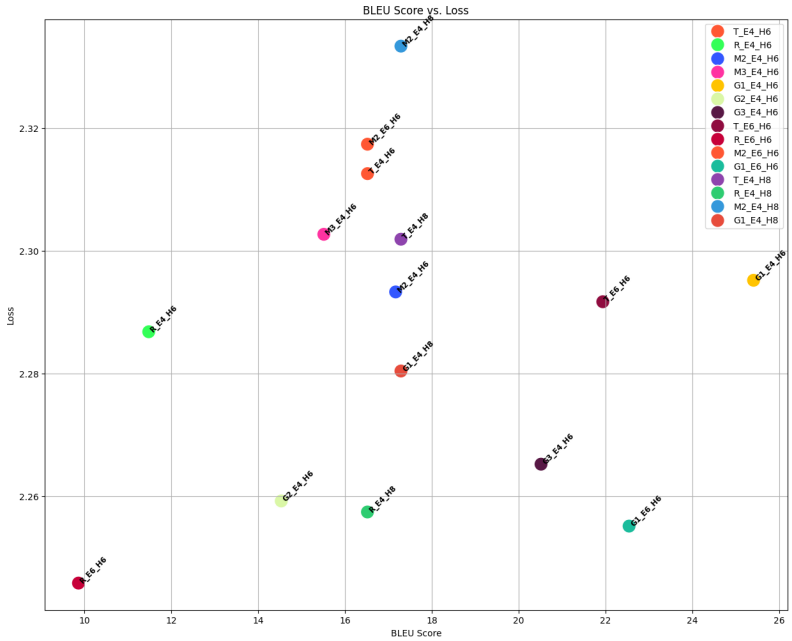

Figure 6.16: BLEU Score Vs Loss

6.2.3 BLEU Score Across Different Head parameter

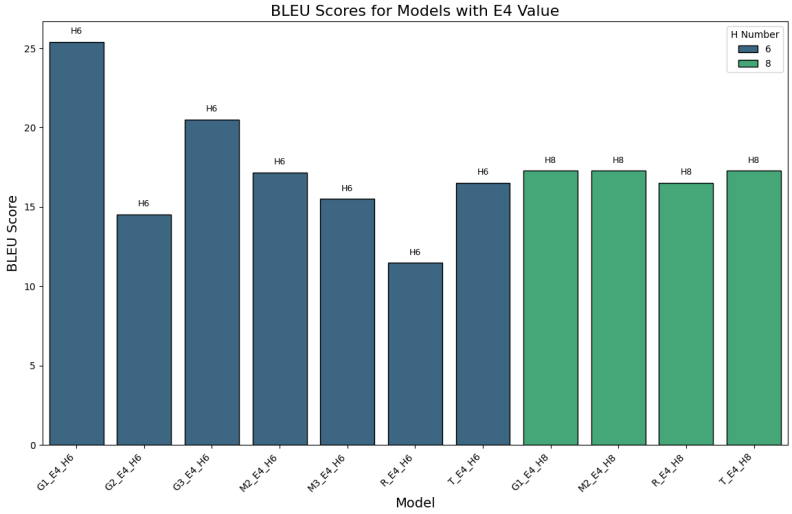

Figure 6.17: BLEU Score with variable number of heads

For all the models with 4 encoder/decoder layers, this bar chart gives a clear view of the relationship between BLEU score and the increase in the number of attention heads. The light green indicates the models with 8 heads and blue represents the models with 6 heads. All 4 models with 8 heads resulted in a similar BLEU score indicating less impact of the architectural changes present within these models. In contrast, the models with 6 attention heads show significant differences in their BLEU score. The most significant difference can be seen in the Gated model where the model with 1 ResNet connection has almost double the BLEU score of the Gated model with 2 ResNet connections. For both Gated and Multi Layered models, it showed a decrease in the BLEU score with the increase in ResNet connections. The

RealFormer model with 6 attention heads showed the lowest BLEU score out of all the models.

6.2.4 BLEU Score Across Different Encoder Layer

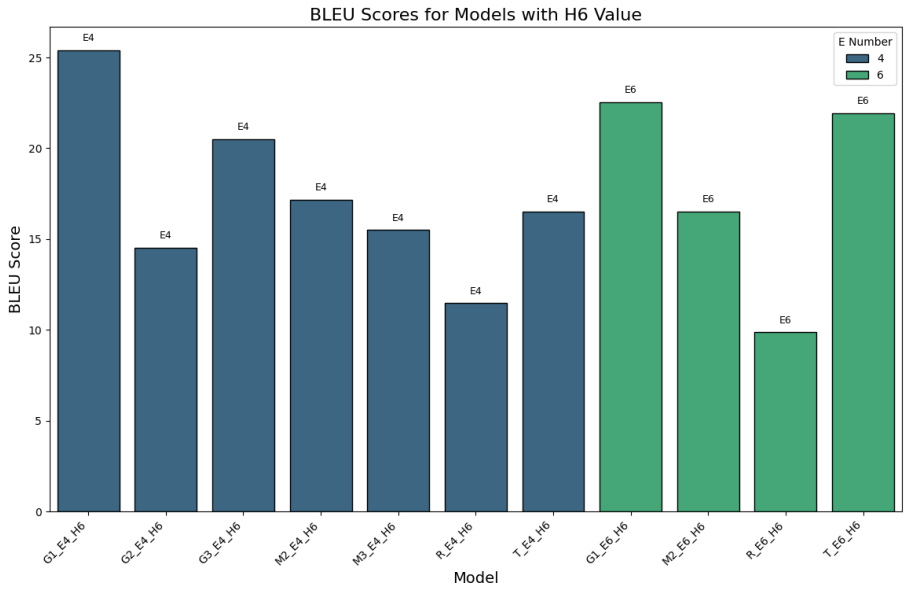

Figure 6.18: BLEU score with variable number of layers

In this graph, we compare the models with 6 attention heads. The models with 6 encoder/decoder layers seemed to have a lower BLEU score compared to their respective variants with the Transformer model being an exception.

6.2.5 BLEU Score based on Model

Transformer BLEU Score

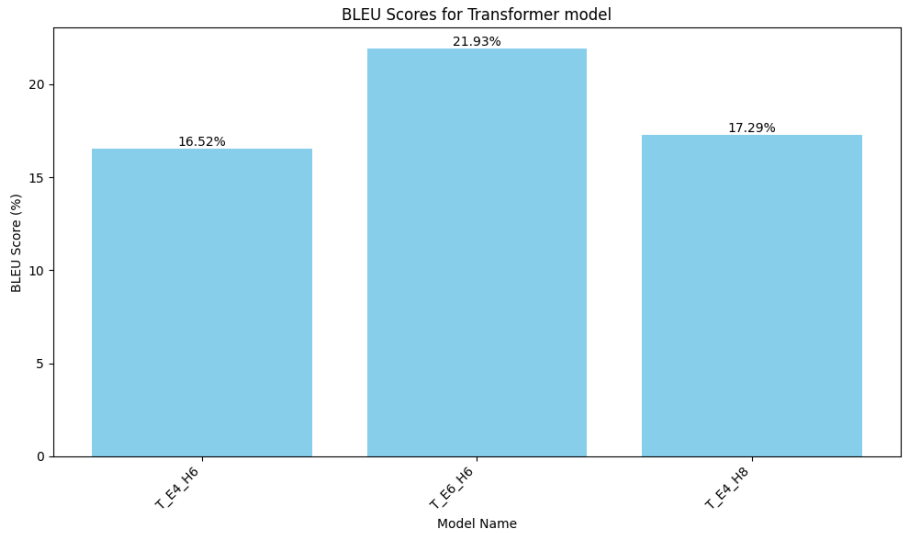

Figure 6.19: Transformer BLEU score

For the regular Transformer model, we compare our 3 variants of configurations. The model with 6 encoder/decoder layers and 6 attention heads showed the highest BLEU score out of all 3. The 6 layers, 6 heads model showed significant increase in BLEU score compared to the model with 4 layers, 6 heads as expected. The increase in attention heads from 6 to 8 showed little increase in BLEU score.

RealFormer BLEU Score

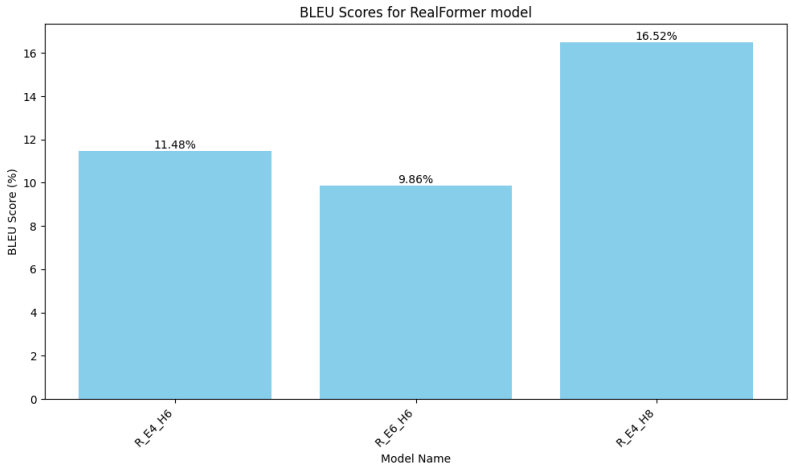

Figure 6.20: RealFormer BLEU Score

In contrast to the previous model, the RealFormer model showed significant increase in BLEU Score with the increase in the number of attention heads compared to the increase in the number of encoder/decoder layers. It should also be noted that the increase in the number of layers decreased the BLEU score of the RealFormer model.

Multi Layered RealFormer BLEU Score

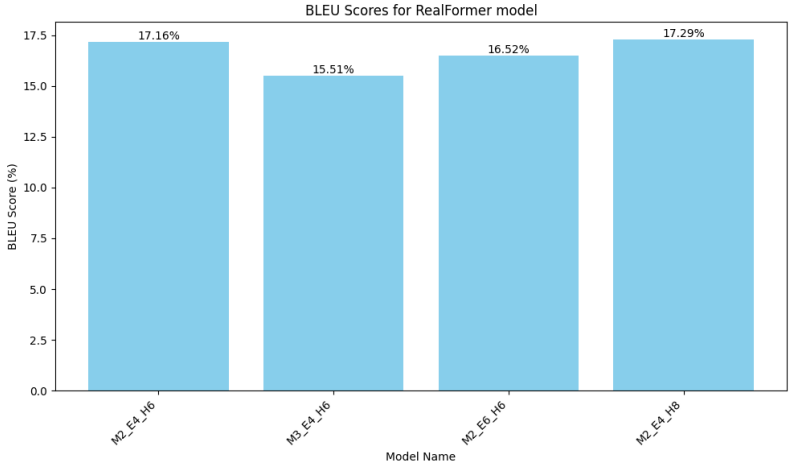

Figure 6.21: Multi Layered RealFormer BLEU Score

In this graph we compare our Multi Layered models with different configurations of encoder/decoder layers, attention heads and ResNet connections. All Multi Layered models resulted in a similar BLEU score. The model with 2 ResNet connections, 4 layers and 8 heads showed the highest BLEU score while the model with 3 ResNet connections, 4 layers and 6 heads showed the lowest. While the differences are insignificant, it should be noted that the increase in the number of layers resulted in a decrease in BLEU score while the increase in attention heads showed an increase.

Gated ResNet BLEU Score

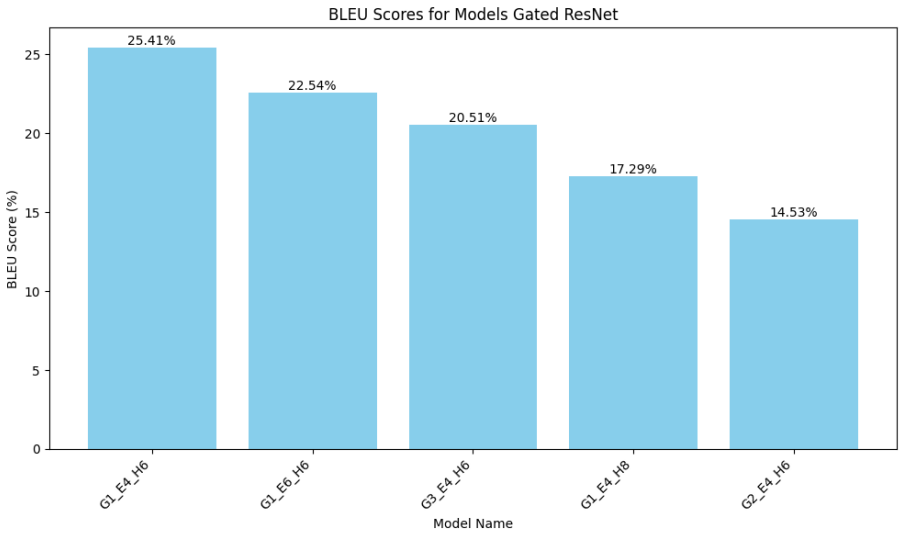

Figure 6.22: Gated ResNet BLEU Score

In this graph, the different configurations for the Gated model are compared. The models with 1 ResNet connection have higher BLEU score than the others with the model with 8 heads being an exception. It can be seen that increasing the number of encoder/decoder layers or increasing the number of attention heads in this model decreased the BLEU score. It can also be observed that the model with 2 ResNet connections has the lowest BLEU score while the model with 3 ResNet connections had a better BLEU score but still lower than the model with 1 ResNet connection.

6.2.6 Overall BLEU score

The following represents a table containing the Model names with their hyper-parameter combination of number of layers and number of heads as well as the BLEU score of the corresponding models.

Table 6.1: Model Configuration and BLEU Scores

Model Type	Encoder	Heads	BLEU Score
Transformer (T)	4	6	16.5158
RealFormer (R)	4	6	11.4780
Multi-Layered (M2)	4	6	17.1640
Multi-Layered (M3)	4	6	15.5100
Gated ResNet (G1)	4	6	25.4060
Gated ResNet (G2)	4	6	14.5300
Gated ResNet (G3)	4	6	20.5120
Transformer (T)	6	6	21.9340
RealFormer (R)	6	6	9.8600
Multi-Layered (M2)	6	6	16.5150
Gated ResNet (G1)	6	6	22.5400
Transformer (T)	4	8	17.2860
RealFormer (R)	4	8	16.5150
Multi-Layered (M2)	4	8	17.2860
Gated ResNet (G1)	4	8	17.2860

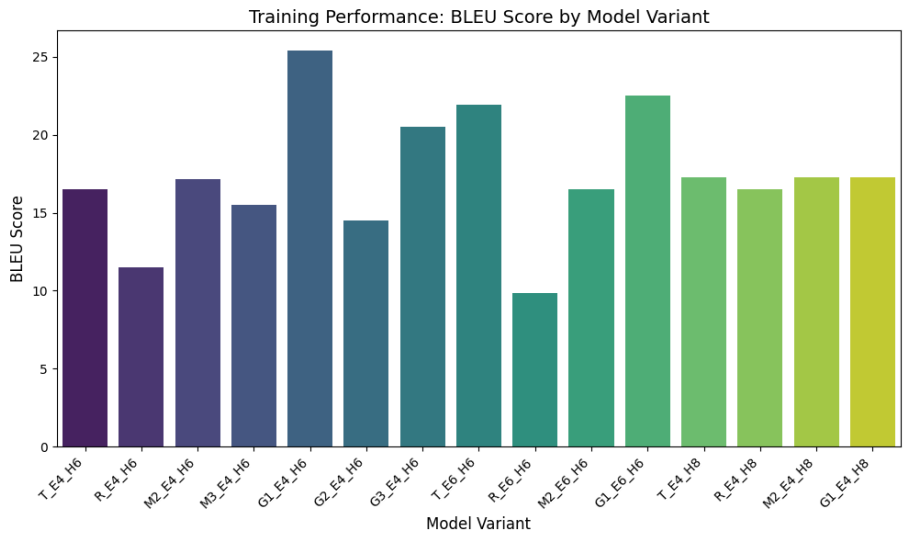

Figure 6.23: All Model BLEU Score

In this graph, we compare the BLEU scores for all our model variants. It can be seen that the RealFormer models had the minimum BLEU score despite having different configurations. The Gated models for both 4 and 6 layer configurations have the best BLEU score out of all the models. The Multi Layered models all have similar BLEU scores and do not show any significant difference from regular Transformer models.

6.3 Sparsity

The following table contains the calculated sparsity of each model in percentage. The calculation was done by counting all the paramters of the model which contained values equal to zero or near zero. The counted value was divided by the total number of parameters in the models and finally represented in percentage format.

Table 6.2: Model Configuration and BLEU Scores with Sparsity

Model Type	Encoder	Heads	BLEU Score	Sparsity
Transformer (T)	4	6	16.5158	5.011
RealFormer (R)	4	6	11.4780	5.345
Multi-Layered (M2)	4	6	17.1640	5.95
Multi-Layered (M3)	4	6	15.5100	5.279
Gated ResNet (G1)	4	6	25.4060	5.458
Gated ResNet (G2)	4	6	14.5300	5.467
Gated ResNet (G3)	4	6	20.5120	5.404
Transformer (T)	6	6	21.9340	5.103
RealFormer (R)	6	6	9.8600	4.388
Multi-Layered (M2)	6	6	16.5150	4.660
Gated ResNet (G1)	6	6	22.5400	5.001
Transformer (T)	4	8	17.2860	5.948
RealFormer (R)	4	8	16.5150	5.643
Multi-Layered (M2)	4	8	17.2860	6.328
Gated ResNet (G1)	4	8	17.2860	5.843

6.3.1 Sparsity Vs Training Time

As we can see in the above graph, the models show a certain trend when training time is compared to the sparsity of the model weights. The models with higher sparsity took lower time to train than the models with higher sparsity. Our Gated model that had the best BLEU score has one of the highest sparsity percentages. While the Multi Layered model with 2 ResNet connections, 4 layers and 8 heads had an above average BLEU score, its weights resulted in having the highest sparsity percentage. It can be concluded that while sparsity may have impacted training time, its relationship with BLEU score is uncertain.

6.3.2 Sparsity Vs BLUE Score

This graph represents the relationship between BLEU score and sparsity. As previously stated, the relationship between sparsity and BLEU score remains uncertain and the graph shows our models scattered across the domain and range proving the uncertainty. Our Gated models seemed to have a similar sparsity percentage while having significant differences in their BLEU score ranging from about 14 all the way to 25. For the Multi Layered models, the sparsity values tend to have a greater range while insignificant changes in their BLEU score. Each model exhibits a different relationship between the BLEU score and sparsity, thus, these two metrics can be assumed to have little correlation.

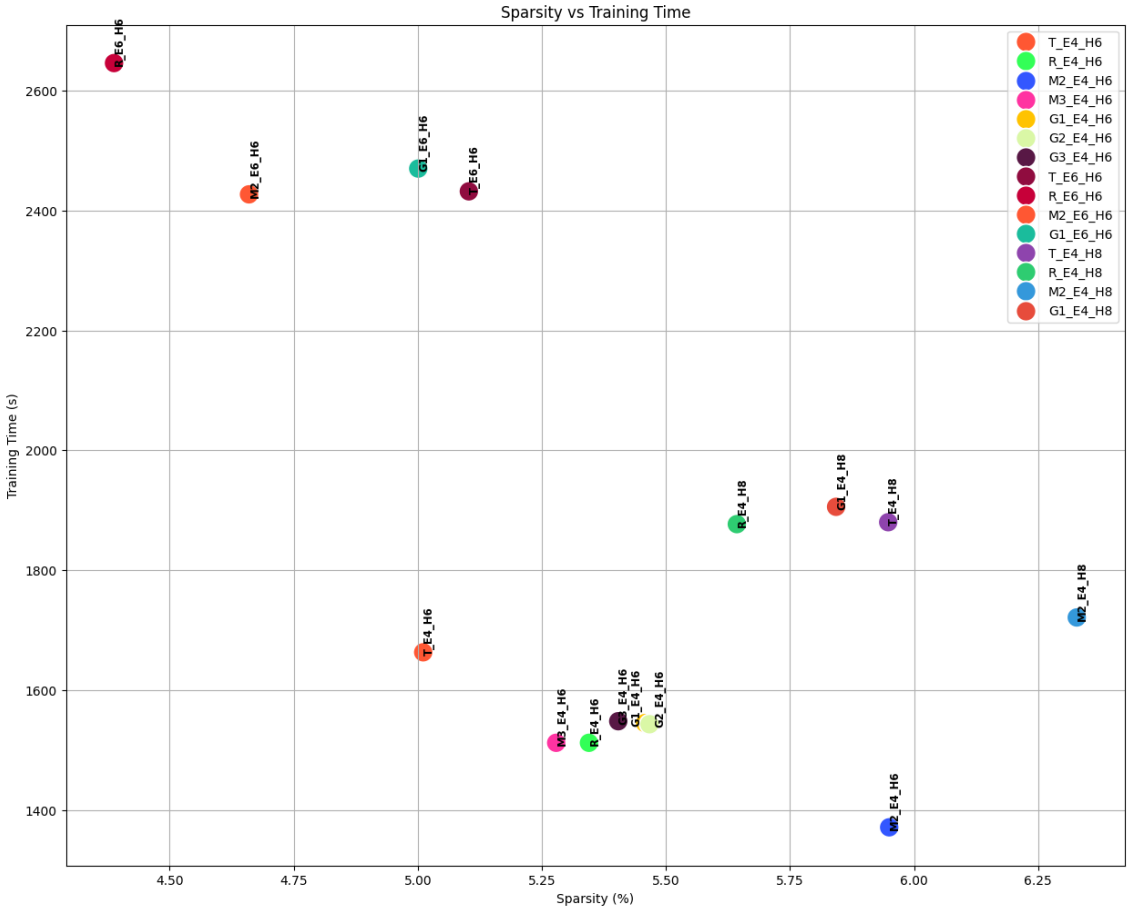

Figure 6.24: Sparsity Effect on Training Time

6.4 Efficiency

Table 6.3: Model Configuration, Training and Inference Times, and BLEU Scores

Model Type	Encoder	Heads	Train Time	Inference Time	BLEU Score
Transformer (T)	4	6	1663.00	1866.64	16.5158
RealFormer (R)	4	6	1512.00	1865.60	11.4780
Multi-Layered (M2)	4	6	1371.00	2002.60	17.1640
Multi-Layered (M3)	4	6	1512.00	1953.50	15.5100
Gated ResNet (G1)	4	6	1545.00	2095.20	25.4060
Gated ResNet (G2)	4	6	1543.00	1732.10	14.5300
Gated ResNet (G3)	4	6	1548.00	2111.40	20.5120
Transformer (T)	6	6	2432.00	2851.20	21.9340
RealFormer (R)	6	6	2646.00	2938.30	9.8600
Multi-Layered (M2)	6	6	2427.00	2915.96	16.5150
Gated ResNet (G1)	6	6	2470.00	3067.00	22.5400
Transformer (T)	4	8	1880.00	1801.80	17.2860
RealFormer (R)	4	8	1876.80	1970.20	16.5150
Multi-Layered (M2)	4	8	1721.10	1834.30	17.2860
Gated ResNet (G1)	4	8	1905.70	1949.30	17.2860

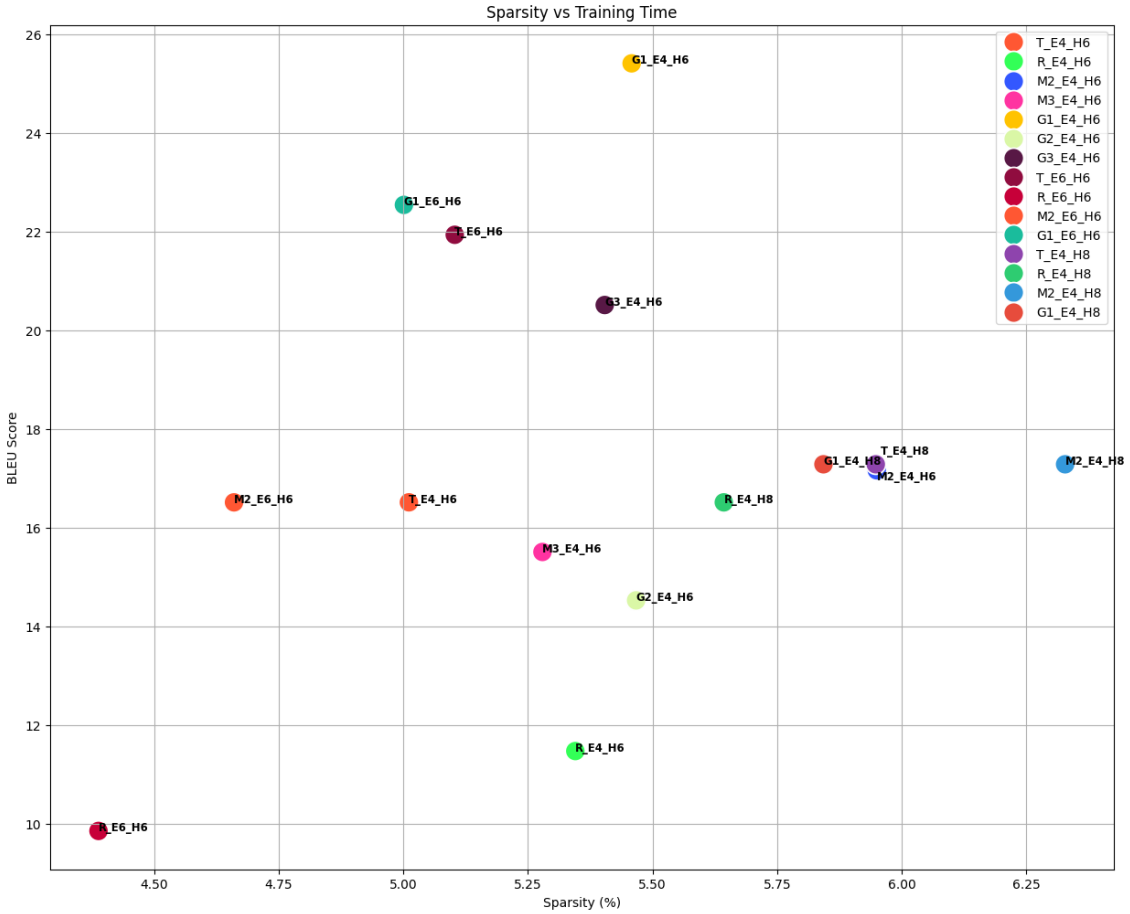

Figure 6.25: Sparsity Effect on BLEU

Based on the above table, we can understand the different training times taken by a model in seconds. Furthermore, we also logged the inference time. By observation of the training time, we understand that $Multi - Layered\ RealFormer$ was the fastest in training time with any combination. Unfortunately, the BLEU score is not up to the mark, and potentially the model lags significantly behind the other variants. The same goes for the $RealFormer$ model. Moreover, the inference time was calculated for translating 500 sentences from Portuguese to English. From the table we understand that our gated model with combination of 4 Encoder 6 Heads took longer time for inference compared to other varients with the same combination. We see a similar observation when it comes to the combination of 6 encoders and 6 heads.

6.5 Attention Score Heat Map

This figure shows the attention distribution for the source tokens ”este é o primeiro livro que eu fiz” shown along the X-axis. These source tokens are translated to the target tokens “this is the first book i did” shown along the Y-axis. The [START] and [END] tokens are special tokens that allow the model to know the beginning and the end of the sentence. In this heatmap we can see a strong diagonal pattern which indicates how the model focuses on the corresponding source and target tokens for each word. An example of this would be the word “livro” which translates to “book”

and we can see that the bright yellow cell indicates how the attention mechanism is targeting the corresponding word. We can also see that the words before "book" also have high attention weights when translating the word "livro" which can be explained by the singularity of the word as the words "is" and "this" correspond to singular nouns. Following are the images of the last decoder's attention heat map from each model when generating the translation of the explained sentence. We

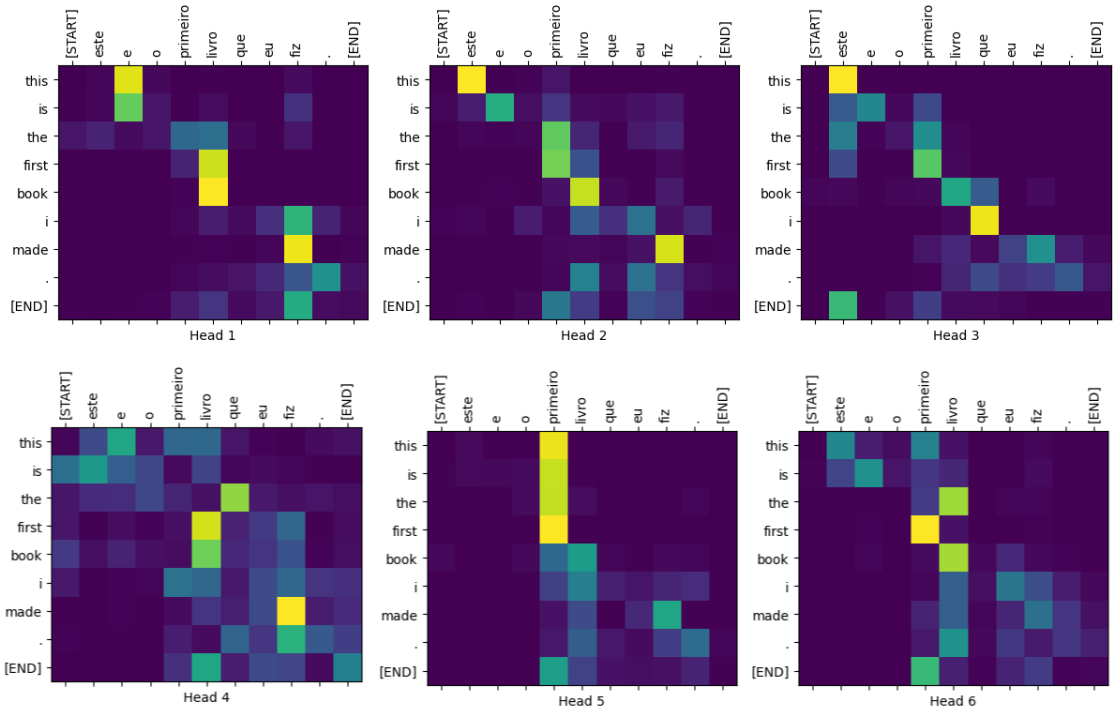

Figure 6.26: Transformer Attention Heat Map 4 Encoder 6 Heads

used 6 attention heads for our models, thus allowing us to capture more dependencies that is not possible with only one attention head. The advantage of using a multi-headed attention mechanism can be visualized using the attention heatmaps for each attention head. We generated these heatmaps for a single example thus allowing us to visualize the distribution of focus across the input and output sentences during translation for all the models.

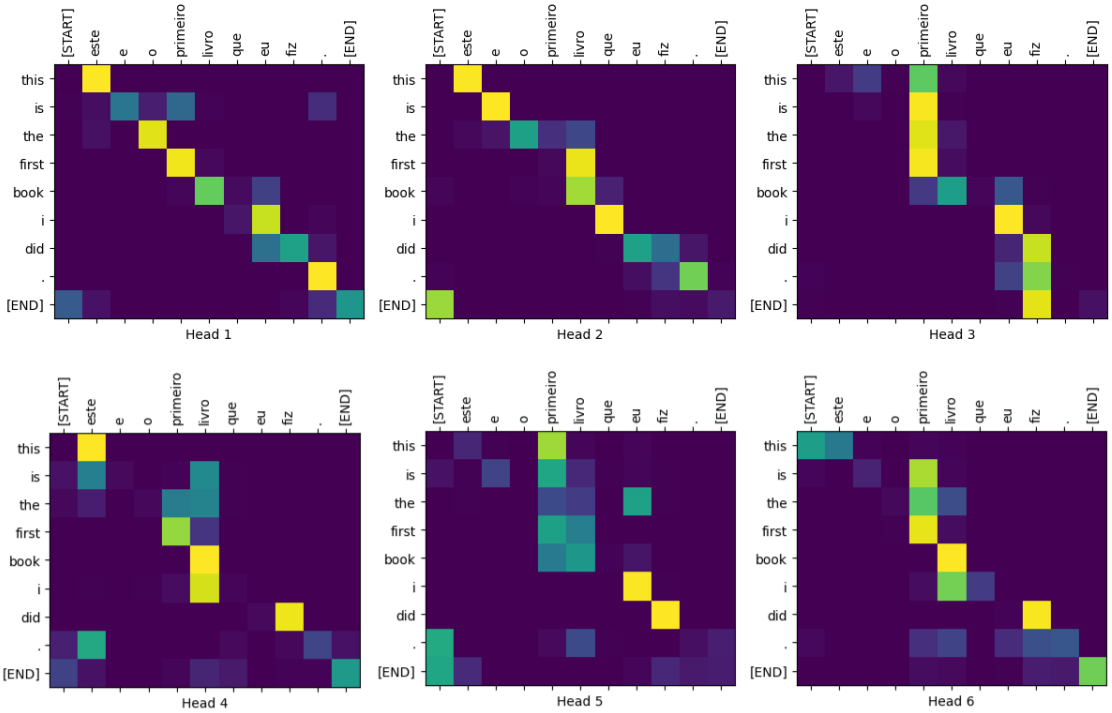

Figure 6.27: RealFormer Attention Heat Map 4 Encoder 6 Heads

The given figures 6.26, 6.27, 6.28, 6.29 shows how each head is diverse and how some heads are specialized to focus on certain parts of the translation. For example, for heads $Multi - Layered\ RealFormer$ model, we can see a strong diagonal patterns indicating the importance of the relationship between each corresponding token. These attention heads attend heavily to word-level correspondences between the source and target tokens suggesting a one-to-one alignment between the two sentences. Another example, from the $Transformer$ model head number 3 shows that the weights are more sparsely distributed across all the words which refers to how each word in the sentence is connected to one another indicating the importance of disambiguation in translation tasks. In machine translation, it is important for the attention heads to capture context from the entire input sequence. From these graphs, we can see how a multi-headed attention mechanism allows the model to gain contextual information and positional focus. From these complex patterns, the transformer model as well as all the varients including out *Gated ResNet* model is able to understand both literal word-for-word translations and translations that are more abstract, meaning it can understand linguistic structures that are different for each language.

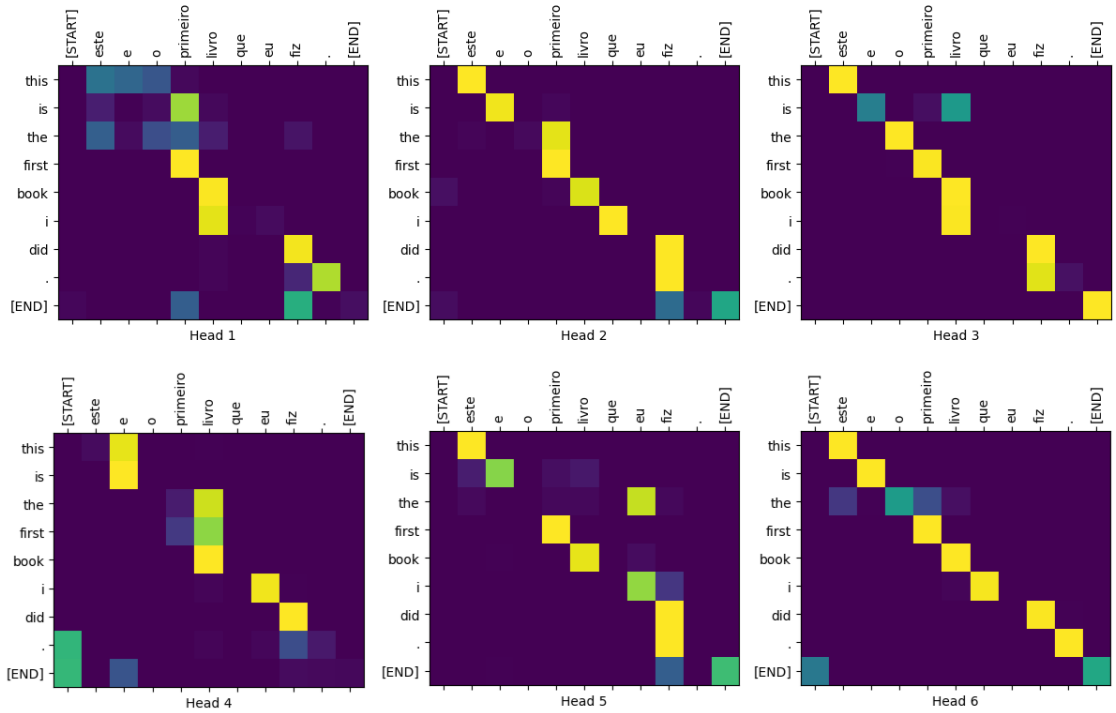

Figure 6.28: Multi-Layered RealFormer Attention Heat Map 4 Encoder 6 Heads

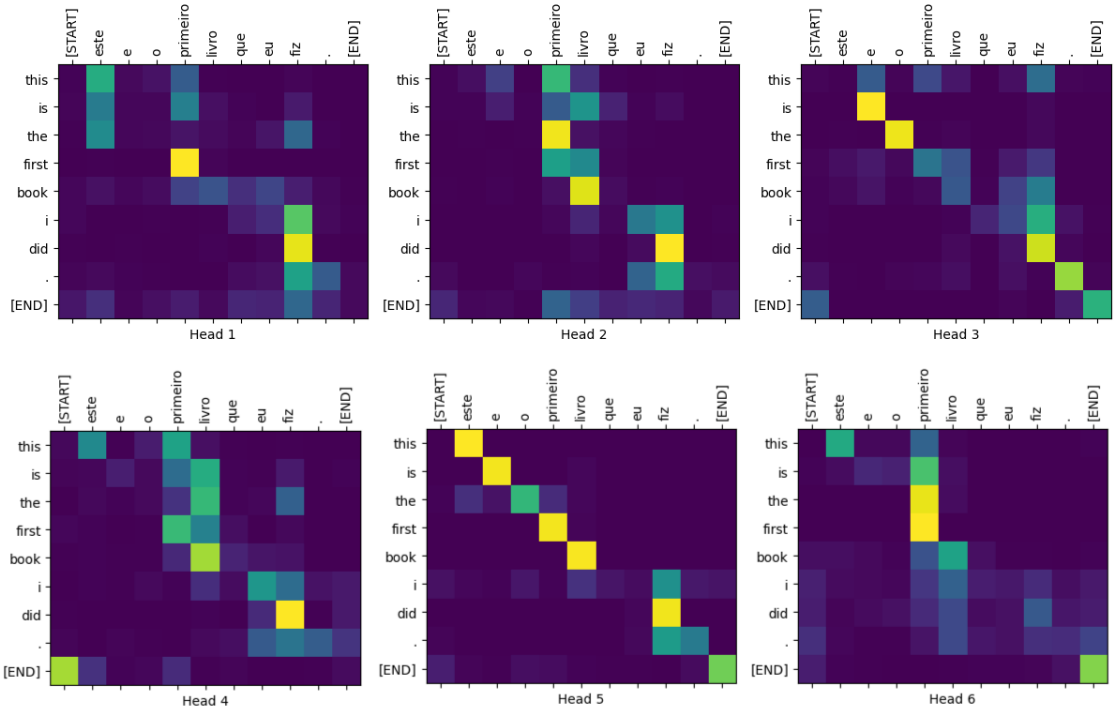

Figure 6.29: Gated ResNet Attention Heat Map 4 Encoder 6 Heads

Chapter 7

Key Findings and Discussion

7.1 Model Performance

Experimental observation suggests that our novel transformer variant, i.e., the *Gated ResNet Transformer* model with a configuration of 4 encoders and 6 heads, achieved the best BLEU score, 25.406, out of all the other models. Furthermore, models with a combination of 6 Encoders and 6 Heads were also observed, and even among these combinations, the *Gated ResNet Transformer* model achieved the peak score of 22.54. Finally, among the combinations with 4 Encoders and 8 Heads, we observe that $Transformer$, $Multi-Layered\ RealFormer$, as well as the *Gated ResNet* model achieve the same BLEU score of 17.286. We did multiple tests several times to ensure that this was not a random coincidence. The score remained the same as before even after multiple tests, and we can hypothesize that this phenomenon occurred because the model is now saturated and cannot learn any more. Without increasing the encoder numbers and making the model bigger, it is difficult for it to achieve a better score. Moreover, the BLEU score that we have generated using the IWSLT English-to-Portuguese dataset is not state-of-the-art performance. A conference paper established in 2019 reported that the best score achieved in the English-to-Portuguese IWSLT dataset was approximately 26.53 [18]. Our novel variant achieved a score of 25.406, which is 1.12 less than the state-of-the-art performance score. Therefore, our novel architecture is significantly close to achieving the best score possible. By conducting future research and optimizing the model, we feel like we can achieve state-of-the-art performance. And be able to compete against state-of-the-art LLMs like the *Llama* [45], $PaLM$ [33], $GPT-4$ [49], etc.

7.2 Model Efficiency

Our primary research goal was to achieve a model that is better in terms of both performance as well as efficiency. Through our experimental observation, we understand that the novel architecture model takes slightly more time to train than that of the $Transformer$ model. Using the combination of 4 Encoders and 6 Heads, the $Transformer$ model achieved a training time total of 1663 seconds, whereas our *Gated ResNet Transformer* model achieved a training time of 1545 seconds. Although this reflects that the *Gated* model was approximately 7.09% more efficient than the $Transformer$ model, it is still lacking behind when it comes to other combinations of encoder and head hyper-parameters. This is due to the $Transformer$

model finishing its training in the 9th epoch, whereas the *Gated* model finished training by the 8th epoch. But for other combinations, such as 4 Encoders and 8 Heads or 6 Encoders and 6 Heads, the $Transformer$ model as well as the *Gated* model finished training at the same epoch. This suggests that per epoch, the *Gated* model takes more time, as the gating mechanism introduces more computation cost, but convergence of *Gated* is slightly faster than the vanilla $Transformer$ model. In the case of $RealFormer$ and $Multi-Layered\ RealFormer$, although the training time is much less, meaning the models are more efficient but lack in terms of performance. For instance, the $Multi-Layered\ RealFormer$ model achieved an average of 5.97% efficiency compared to the baseline $Transformer$ model in all combinations. It may converge faster but significantly lack in performance, and so this efficiency achievement cannot be considered. We were hoping for our *Gated ResNet* $Transformer$ model to achieve more efficiency, but it seems that the gating mechanism is not perfect and is left for future research to make it more efficient and may be able to compete against efficient transformers like Reformer [27].

7.3 Inference Time

When it comes to real-world application, software architectures and models often depend on the inference time. The inference time refers to the total amount of time required for a model to generate an output, also known as the throughput of a model. Our *Gated ResNet* $Transformer$ model takes slightly higher inference time compared to the baseline vanilla $Transformer$ model. According to statistical analysis, the $Transformer$ model for combination 4 Encoders and 6 Heads takes around 1866.64 seconds to infer 500 sentences that are tokenized and sent as an input. Whereas the *Gated ResNet* model takes around 2095.5 seconds. Another example, for a combination of 4 Encoders and 8 Heads, the $Transformer$ model takes 1801.8 seconds for inference, whereas the *Gated* model takes around 1949.3 seconds. As for the combination of 6 Encoders and 6 Heads, the *Gated* model still lags behind. Overall estimation states that the *Gated ResNet* model is approximately 6.8% less efficient than the $Transformer$ model. The need to maintain high-quality outputs while adhering to tight inference time constraints, especially in safety-critical applications, is extremely important [51]. As a result, we conclude that the inference time of the model must be improved. This is due to the gating mechanism in our *Gated* model. While inferencing, the gate weights are being sliced first, and then the attention scores are being computed. Which adds to the computation overheads. Moving forward, as for the other models, the performance of $RealFormer$ as well as the $Multi-Layered\ RealFormer$ is not up to the mark. Furthermore, the inference time is also higher than that of the $Transformer$ model. All combinations of $RealFormer$ and $Multi-Layered\ RealFormer$ performed less than the standard baseline in terms of performance, efficiency, and inference time.

7.4 BLEU score Behavior

For our machine translation task, we used BLEU score as the evaluation metric of our translation. This section will analyze and show if the BLEU score has any correlation with other metrics we used to evaluate the models.

7.4.1 BLEU Score Vs Accuracy

From experimental observations 6.15, we understand that BLEU score does not reflect any relationship with the validation masked accuracy of a model. Which implies that if the validation accuracy of a model is high, that does not necessarily translate to a better BLEU score. Furthermore, a transformer model cannot be judged and evaluated only based on the validation accuracy. Transformer models are complicated, and each model is specific to its task. As a result, each transformer model must be evaluated based on the task it has been opted for. For instance, a model's performance on a benchmark dataset is often evaluated using only a few metrics [23]. Furthermore, the single metric that is being used to evaluate our model, which is the BLEU score, may not be enough to understand the full extent and potential of our novel transformer variant. For example, when our *Gated* mode translates the sentence "este é o primeiro livro que eu fiz" from Portuguese to English, the translation often comes as "this is the first book I did, whereas the ground truth is "this is the first book I've ever done." From this, we understand that the meaning is being portrayed to some extent, but the words and the grammar are not yet perfect. BLEU only prioritizes the words of the sentence as well as their position but lacks the ability to identify the meaning and grammar of a sentence. This shows that only a single metric is not enough to understand the behavior of the model. To conclude, BLEU scores have no significant correlation with the accuracy of the models.

7.4.2 BLEU Score Vs Loss

Similar to the relationship between BLEU score and accuracy, BLEU score also shows no correlation with the loss of a model. From the graph 6.16 we can easily determine that the BLEU score of the model shows no correlation with the validation masked loss of a model. For example, with the combination of 4 Encoders and 6 Heads, the least amount of validation loss was incurred by the *Gated ResNet* $Trasnformer$ model with 2 residual connections, which was 2.2592. Unfortunately, the BLEU score generated was only 15.51. Which actually proves that the validation loss of a model does not reflect the behavior of the BLEU score generated by the model.

7.5 Sparsity of Models

One of our primary objectives of the study was to create a model that is efficient in terms of training time. As suggested by the paper [25], the gating mechanism helps the model to create a sparser distribution of weights. Sparsity can lead to a model being more efficient, effectively reducing the number of parameters being used. This will in fact help the model use lower computational resources as well as memory consumption. This section of the paper will try to understand the behavior of our model based on sparsity and to figure out if there is any correlation between sparsity and that of the BLEU score and efficiency of the model.

7.5.1 Sparsity Vs Efficiency

Sparsity of a model contributes mostly to the training time required by the model. As the sparsity of a model increases, the number of zero values also increases in the model. This evidently helps to reduce the number of computations required by the model to converge faster. Because anything multiplied by zero is essentially set to zero automatically by the operating system. Furthermore, zero values are not stored by the hardware; as a result, less memory is required to fit the model into GPU virtual memory. The study "Low-Memory Neural Network Training: A Technical Report" looks into a variety of techniques, including sparsity, to reduce the memory footprint during training [35]. The authors show that using sparsity can result in significant memory savings with little impact on model accuracy. From the excremental graph 6.24 we understand that as the sparsity of the model increases, the training time required by the models also decreases. For better understanding, a best-fit line is used in the figure to explain that the sparsity does contribute little to none to the efficiency of a model. Unfortunately, the experimental results show that this is true for all the models and not specifically the $Gated\ ResNet\ Transformer$ model. And so, the gating mechanism is not necessarily creating sparser weights or parameters in the model. For more concrete evidence, further investigation of the gating mechanism and sparsity metric is necessary. From this we can conclude that sparsity does somewhat reflect a lower training time, providing a more efficient model, but the gating mechanism is not necessarily creating the sparsity of the models.

7.5.2 Sparsity Vs BLEU Score

Observations from the experiment 6.25 show that the sparsity of each of the models is similar to each other. We see no noticeable difference between the non-gated transformer models and the gated transformer models. The average sparsity of each of the models is around 5.5% for the type of model and for all combinations as well. As a result, we can conclude that sparsity does not reflect any relationship with the BLEU score generated by the models. Furthermore, when specifically focusing on the gated models, we observe that the gating mechanism doesn't contribute much to the overall sparsity of the model. And so, we conclude that sparsity does not affect the BLEU score generated by the models.

7.6 Multiple ResNet Connections

We extended the $RealFormer$ model by implementing not just a single ResNet connection to the previous encoder multi-head attention scores but also using multiple previous layers to incorporate approach-1 3.1 as well as approach-2 **??**. We further pushed this concept to the $Gated\ ResNet\ Transformer$ model and implemented multiple previous layers as well. From experimental observation of the $Multi-Layered\ RealFormer$ model's BLEU score, we understand that multiple ResNet connections do not necessarily help the model to achieve a better BLEU score. This is hypothetically easy to understand why. The attention score in neural networks, specifically transformer models, quantifies the importance of one element in a sequence to another. This attention calculation mechanism allows the model

to focus on specific parts of the input when producing outputs, effectively capturing dependencies regardless of their position in the sequence [36]. The Transformer architecture, which eliminates recurrence and convolution in favor of attention mechanisms, is introduced in the seminal paper "Attention Is All You Need" [13]. For simplicity, let's consider the attention matrix to create a vector with magnitude and direction. Each head tries to capture the same sentence from multiple perspectives. And so, each head produces a different vector. As a result, the model learns the sentence translation from multiple contexts. When creating a ResNet connection to the previous layer, the attention scores slightly align together in the same direction. As a result, the vectors also start to align in the same direction. Adding more and more residual connections would mean that the vectors align together and lose the idea of capturing context from multiple perspectives or contexts. The generated BLEU score from the model $Multi-Layered\ RealFormer$ 6.21 and *Gated ResNet* $Transformer$ 6.22 reflects that as the residual connection increases by layers, the BLEU score also reduces proportionally. From this analysis we can state that multiple residual connections do not necessarily help the model in terms of the machine translation task. But in other cases, like image classification and ViTs, the model might excel. Further investigation and testing are necessary in different domains as well to understand the potential of the model fully.

7.7 Key Insights

After careful evaluation and analysis of our model, we can concur that the *Gated ResNet* $Transformer$ model's performance is excellent and does have the potential to be even better. With a 25.406 BLEU score on the IWSLT English-to-Portuguese dataset, this model outperforms every other model compared to our testing candidates. As for efficiency, we figure that the model does not show much of a significant difference compared to other models. Furthermore, the gating mechanism does not create more sparse weights in the model rather it slightly increases the inference time. Moreover, we discover that sparsity as well as BLEU score does not reflect the loss and accuracy incurred by the models. Also, the sparsity measure in each model is roughly equivalent to other models. Hence, the gate mechanism does not contribute directly to the sparsity of the model. We also notice that there is little to no correlation between the efficiency of the model and the sparsity. Lastly, multiple residual network connections, like the $Multi-Layered\ RealFormer$ model and *Gated ResNet* $Transformer$ model with multiple ResNet, do not show better results. Even if the models are better in terms of training time, the BLEU score is not up to par compared to other models like the $Transformer$ and *Gated ResNet* $Transformer$ with a single residual connection.

Chapter 8

Limitations

Throughout the entire study, oftentimes we faced multiple problems and tried to figure out a solution to those problems. In this section of the paper, we would like to address some of the limitations of our experimental study as well as a few limitations of our novel transformer variant. These limitations stem from different aspects of our study, such as methodology, resources, and generalization.

8.1 Experimental Limitations

8.1.1 Computational Resources

Transformer models are termed as large language models. Typically, these models have billions of parameters, and when training the models, they utilize a huge amount of GPU virtual memory as well as computational units. Furthermore, the computational cost in terms of hardware resources scales quadratically with the sequence length/token length. And so, training transformer models became a challenging task where large input sequences cannot be used [42]. Unfortunately, due to a lack of computational resources, we were unable to conduct a more robust experiment with large transformer models. As a result, we had to shrink our models into sizable parameters just to fit the model into our GPU virtual memory. Our base model vanilla transformer had approximately 17 million parameters, whereas our own Gated ResNet model had 0.1 million more than the base transformer. Among our various models, the peak number of parameters reached was 23 million approximately. Each of the models utilized more than 20 GB of virtual memory while training. Our study had to maintain all these constraints due to a lack of computational resources.

8.1.2 Dataset Size

Each of the experimented models takes a tedious amount of training time, and with low computation units to spare, we addressed the problem by using smaller datasets. According to multiple researches conducted prior to ours, we understand that transformer models benefit substantially when trained on larger datasets compared to smaller datasets and achieve state-of-the-art performance [39]. Since the dataset is small, it cannot be considered best practice for training and experimentation on transformer models. For our machine translation task, the state-of-the-art datasets

that are typically used are WMT14 dataset English-to-French (fr-en) as well as the WMT14 English-to-German (de-en) [5]. We have seen the use of these exact datasets in the paper "Attention is all you need" [13] as well as the Realformer paper [30].

8.1.3 Evaluation Matrics

A model's performance on a benchmark dataset is often evaluated using only a few metrics. While this allows for quick comparisons, it runs the risk of not accurately reflecting the model's effectiveness if the metric does not eventually cover every aspect of performance characteristics [23]. Evaluating NN architectures like Transformer models involves considering various factors which includes the limitations of traditional metrics. That is, the subjectivity in interpreting performance gains, and the trade-offs between model complexity and performance. For instance, the usage of BLEU to evaluate machine translated text from one language to another is only plausible when conducting validation studies based on direct assessments [16]. This highlights the necessity of supplementary validation methods to ensure comprehensive evaluation. Furthemore, BLEU score does not capture the essence of meaning behind the words or any sort of grammaticality [22]. BLEU usually tends to represent how much the translated sentence is as to the ground truth. This underscores the subjective nature of certain metrics and the potential disconnect between statistical significance and practical relevance.

8.1.4 Baseline model comparison

When comparing novel transformer architecture with a baseline model, it is important to ensure fair and meaningful comparisons. For our study we compared our novel architecture with a baseline model such as the vanilla transformer from the paper "Attention is all you need" [13]. We also utilized the Realformer architecture [30] since our core architectural changes involved Realformer. Unfortunately, it is important to remember that when comparing new models against older transformer variants, it may not provide a fair evaluation. As seen previously, baseline models lack optimization, which is present in recent versions. As a result, the rapid evolution of transformer models emphasizes the importance of selecting up-to-date baselines for comparison [50].

8.1.5 Generalization Across Domains

Our novel architecture model proves to be better in terms of performance and sometimes proves to be more efficient in some cases. But the effectiveness of our gated model was only observed through a specific sequence-to-sequence task, i.e., machine translation. Ensuring that machine learning models generalize effectively across diverse domains and datasets is a critical challenge. Some of the key concerns include things like task-specific performance, limitations, and risk of overfitting specific benchmarks. Overfitting is frequently used as a generic term to describe any unintended performance drop in machine learning or transformer-based architecture models [20]. This highlights the necessity for models to be evaluated on diverse datasets as well as different tasks to ensure robust performance beyond specific benchmarks.

8.1.6 Scalability and Interpretability

Real-time application and deployment of real-world applications demand that neural network architectures be scalable and interpretable. A theoretical framework for transformer models is important to understand how they perform and scale based on the model size as well as the data dimensions. Identifying distinct scaling regimes related to the model can provide valuable insights on the limitations of the novel architecture[43]. For instance, our model shows variable results when scaled by encoder or heads. Although it outperforms the traditional vanilla transformer model, it is still a black box. Our gating mechanism scales as the model hyperparameters scale. For instance,

$$\text{Let Sequence length} = \text{Seq}$$

$$\text{Let number of heads} = \text{H}$$

$$\text{Let number of Encoders} = \text{N}$$

$$\text{Gate params for single head} = \text{seq} \times \text{seq}$$

$$\text{Gate params for single encoder} = \text{seq} \times \text{seq} \times \text{H}$$

$$\text{Gate params for all encoder} = \text{seq} \times \text{seq} \times \text{H} \times \text{N}$$

$$\text{Gate params considering decoder} = \text{seq} \times \text{seq} \times \text{H} \times \text{N} \times 2$$

Consider the baseline model hyperparameters. Let the sequence length be equal to 64, the number of heads per encoder be 6, and the total number of encoders be 4. Input these values into the gate param equations. Consider the baseline model hyperparameters. Let the sequence length be equal to 64, the number of heads per encoder be 6, and the total number of encoders be 4. Input these values into the gate param equations.

$$\text{Gate params considering decoder} = 64 \times 64 \times 6 \times 4 \times 2$$

$$\text{Total Gate Params} = 196608$$

As a result, we can understand from the equation that the gating mechanism will scale based on the sequence length as well as the number of heads and encoders in a specific model. Furthermore, it is also difficult to interpret what the gate is doing and how it is optimizing the attention scores for better performance and efficiency. More extensive testing must be applied to the gating mechanism to understand its function and implement it in a better way.

8.2 Model Limitations

8.2.1 Inference Time

Upon monitoring the experimental results of inference time, we understand that our novel architecture model takes more time. Inferencing is important when considering real-world application. Even when considering machine translation tasks, real-time translation is more acceptable to clients for application-based implementation. Deep neural networks, as well as transformers, are cutting-edge methods for many tasks related to learning because of their ability to obtain increasingly better features

with each layer. Unfortunately, this level of complexity can lead to longer inference times [41]. Furthermore optimization of GPUs is also important when it comes to transformer models. For instance, batch size, or the amount of input sentences processed by the model at a time will greatly depend on the hardware it is operating on. A high-throughput is open hindered because of the memory constraints of the hardware [44]. As a result a GPU-based hardware will be much more efficient when inferencing compared to a CPU or low-memory hardware resource. Many studies collectively highlight the importance of optimizing neural network architectures and transformer architectures to address increased latency, batch processing limitations, and real-time constraints, ensuring efficient and practical deployment in various applications.

8.2.2 Gating Mechanism

The gating mechanism introduced in our novel transformer architecture model becomes complex to handle when the input sequence becomes dynamic. For instance, when inferring, the input sequence is not always the same. We applied solutions to the problem. Firstly, we considered dropping the gate mechanism altogether when inferencing, but theoretically not using the gate would mean we are bypassing some weights of the model and not utilizing the trained aspect of the model itself. Which led the model to perform poorly compared to other models. Secondly, we considered using the gating mechanism by slicing the gate weights based on the sequence length of the input. As a result, we were able to partially use the gate weights and maintain a better performance. This itself is a complex task for the GPU/CPU to execute, and a better solution must be considered.

Chapter 9

Future Works

9.1 Efficient Mechanism

Optimizing the performance of a transformer model involves implementing an efficient and optimized attention mechanism so that the model improves in efficiency without losing performance. Our novel transformer architecture utilizes the gating mechanism to increase the sparsity of the model, but through observation from the experimental data, we can conclude that sparsity was not a major aspect for the model. As a result, we need to incorporate a better sparse mechanism for our novel architecture. A paper introduced SPARSEK Attention, a novel sparse attention mechanism designed to overcome computational and memory challenges in handling long sequences [46]. Furthermore, another paper introduced the SEA, a linear attention method that performs sparse attention in linear time using a generated attention mask based on compressed attention matrix estimation [40]. The implementation of this new architecture can help improve the sparsity of transformer models and achieve a faster training time. Moreover, hyperparameter tuning is important to achieve state-of-the-art performance by any model. In this experimental study, the paper did not focus on hyperparameter tuning. And so, this can be a potential aspect for future studies.

9.2 Generalization Across Domains

This study compared the novel transformer variant with other baseline transformer models, specifically in a sequence-to-sequence machine translation task. In order to understand the full potential of a novel architecture model, we need to consider experimenting with the model across multiple tasks. Deep learning applications generally assume that the training and testing distributions are identical. To accomplish this, it is critical to create models that can generalize to previously unknown distributions [38]. Not just machine translation but other tasks such as text classification, image classification, question-answer applications, chatbots, etc. Generalization across domains is crucial in machine learning and transformer models because it enables the model to be applied in different aspects of real-life problems. Furthermore, a single dataset is not optimal for comparison as the model might overfit to that certain dataset and show biased results. Therefore, machine learning models should be trained across multiple sets of datasets as well as across multiple tasks.

9.3 Improve Gating Mechanism

The novel transformer variant introduced in this paper utilizes a basic *tanh* gating mechanism. Furthermore, the gating mechanism increases the complexity as well as the computation. For the gate to function, first the attention score of the previous encoder layer is multiplied with the gate weights, secondly the transformed scores are passed through a *tanh* gate and multiplied with the previous attention score again, finally the transformed attention score is added to the current attention score. We can already observe that there are multiple arithmetic operations that are taking place. Furthermore, when considering not just adding the previous attention score from one layer but also multiple layers can also increase the computational cost. The model can be explored more by placing each individual previous score through separate gates assigned specifically for that nth previous score. This will mean that the gate weights will not be shared among all the previous layers but will increase the number of gates. Which will result in more computational cost but do have the potential to provide new observations. Last but not least, this paper used hyperbolic tangent also known as *tanh* for the activation function used in the gating mechanism. Other types of gating functions such as the sigmoid gate, were not experimented with. For the *sigmoid* activation function, the gradient is relatively small, particularly for inputs that are far from zero. Which can exacerbate the vanishing gradient problem in deep networks. As for *tanh*, the gradient is steeper than that of the sigmoid especially around zero. This may result in stronger gradients and eventually lead to more significant weight updates during training, which may lead to improved performance [48].

9.4 Dynamic Gating Mechanism

While proceeding with the theoretical analysis of the novel transformer variant in this paper, we had a potential interest in experimenting with the model using a dynamic gating mechanism. The dynamic gating mechanism of the model will selectively drop out certain gates just like how a regular neural network dropout system works. By doing so we are forcing the model to use the gate less. In this way we might potentially have a balance between performance as well as efficiency. This would also enable the model to run a bit faster since the gates are being dropped and not computed. Another theoretical idea that can be explored is dynamic layered previous attention. From our model's architecture, we observe the gating mechanism's equation as:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + T'(\text{prev}_{(n-1)})\right) \cdot V \tag{9.1}$$

We can change the architecture and observe not the previous layer's attention score but potentially the previous-previous layer's attention score. The equation would go as:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + T'(\text{prev}_{(n-2)})\right) \cdot V \tag{9.2}$$

We can further explore this dynamic gate theory for any nth layer as well:

$$\text{ATT\_Score} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + T'(\text{prev}_{(N-n)})\right) \cdot V \tag{9.3}$$

9.5 Scalability

Due to a lack of computational resources and hardware, this paper was unable to explore a bigger transformer model. For our circumstance, the experiment was done with very low-level hyper-parameters. From previous research studies conducted on transformers, we understand that scaling a transformer model leads to better performance. A transformer model's performance is power-law dependent on model size, dataset size, and computation time [26]. This indicates that increasing the size of transformer models, along with the dataset and computational resources, leads to consistent improvements in performance. This does not primarily focus only on the model size but also the dataset size. As discussed before, dataset size also matters when it comes to model performance. With bigger computational hardware, we can explore and experiment further on this model.

Chapter 10

Conclusion

In this paper, we establish three simple and generic novel approaches for the attention layer of the transformer model aimed at improving the performance of the model as well as increasing the training efficiency. By incorporating various techniques of skip connections like ResNets and Highway Network, we facilitate the information flow of attention scores of heads from one layer to adjacent layers. Furthermore, we attain an overall performance gain with an insignificant increase in parameters and memory requirements. This paper empirically proves that the model achieves better results compared to other generic transformer models. Specifically, in sequence-to-sequence tasks, by adjusting a few trainable parameters, long-term dependencies can be maintained without requiring additional training time or memory. The novel approaches established in the paper can be incorporated with other SOTA transformer models and researched further to study the impacts of skip connections over attention heads. The model can be further tuned for specific tasks in future applications.

Bibliography

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu," *Association for Computational Linguistics*, pp. 311–318, Jul. 2001. DOI: 10.3115/1073083.1073135. [Online]. Available: https://doi.org/10.3115/1073083.1073135.

[3] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 315–323. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html.

[4] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, Atlanta, GA, vol. 30, 2013, p. 3.

[5] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," *Association for Computational Linguistics*, pp. 12–58, Jun. 2014. DOI: 10.3115/v1/w14-3302. [Online]. Available: https://doi.org/10.3115/v1/w14-3302.

[6] M. Cettolo, C. Girardi, and M. Federico, "The IWSLT 2014 evaluation campaign: Evaluation of the spoken language translation track," in *Proceedings of the 11th International Workshop on Spoken Language Translation*, Lake Tahoe, California, USA, 2014.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Google Research*, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1409.3215.

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2015. [Online]. Available: https://arxiv.org/pdf/1409.0473.

[9] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015. [Online]. Available: https://arxiv.org/abs/1505.00387.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. [Online]. Available: https://doi.org/10.1109/cvpr.2016.90.

[11] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *Association for Computational Linguistics*, Jan. 2016. DOI: 10.18653/v1/p16-1162. [Online]. Available: https://doi.org/10.18653/v1/p16-1162.

[12] J. Kim, M. El-Khamy, and J. Lee, "Residual lstm: Design of a deep recurrent architecture for distant speech recognition," in *Interspeech*, Samsung Semiconductor, Inc., 2017. [Online]. Available: https://doi.org/10.21437/interspeech.2017-477.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, vol. 30, pp. 5998–6008, 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762v5.

[14] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. M. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, e00938, 2018. DOI: 10.1016/j.heliyon.2018.e00938.

[15] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *Association for Computational Linguistics*, Jan. 2018. DOI: 10.18653/v1/p18-1007. [Online]. Available: https://doi.org/10.18653/v1/p18-1007.

[16] E. Reiter, "A structured review of the validity of BLEU," *Computational Linguistics*, vol. 44, no. 3, pp. 393–401, Sep. 2018. DOI: 10.1162/coli_a_00322. [Online]. Available: https://aclanthology.org/J18-3002/.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2019. [Online]. Available: https://doi.org/10.48550/arxiv.1810.04805.

[18] J. Niehues, R. Cattoni, S. Stüker, M. Negri, M. Turchi, E. Salesky, R. Sanabria, L. Barrault, L. Specia, and M. Federico, "The iwslt 2019 evaluation campaign," *International Conference on Spoken Language Translation*, Nov. 2019. DOI: 10.5281/zenodo.3525578. [Online]. Available: https://cris.fbk.eu/handle/11582/321912.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019. [Online]. Available: https://insightcivic.s3.us-east-1.amazonaws.com/language-models.pdf.

[20] R. Roelofs, V. Shankar, B. Recht, S. Fridovich-Keil, M. Hardt, J. Miller, and L. Schmidt, "A meta-analysis of overfitting in machine learning," *Neural Information Processing Systems*, vol. 32, pp. 9175–9185, Jan. 2019. [Online]. Available: https://papers.nips.cc/paper/9117-a-meta-analysis-of-overfitting-in-machine-learning.pdf.

[21] S. Salman and X. Liu, "Sparsity as the implicit gating mechanism for residual blocks," *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, Jul. 2019. DOI: 10.1109/ijcnn.2019.8851903. [Online]. Available: https://doi.org/10.1109/ijcnn.2019.8851903.

[22] R. Tatman, "Evaluating text output in nlp: Bleu at your own risk," *Towards Data Science*, 2019.

[23] K. Blagec, G. Dorffner, M. Moradi, and M. Samwald, "A critical analysis of metrics used for measuring progress in artificial intelligence," *arXiv preprint arXiv:2008.02577*, 2020.

[24] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," in *Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[25] Y. Chai, S. Jin, and X. Hou, "Highway transformer: Self-gating enhanced self-attentive networks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.616.

[26] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[27] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html.

[29] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "Multi-head attention: Collaborate instead of concatenate," *arXiv preprint arXiv:2006.16362*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2006.16362.

[30] R. He, A. Ravula, B. Kanagal, and J. Ainslie, "Realformer: Transformer likes residual attention," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-acl.81.

[31] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2021. [Online]. Available: https://arxiv.org/pdf/1811.03378.pdf.

[32] A. Ramesh, *Finding the optimal vocabulary size for neural machine translation*, non-en, Mar. 2021. [Online]. Available: https://www.rws.com/language-weaver/blog/issue-121-finding-the-optimal-vocabulary-size-for-neural-machine-translation/#:~:text=A%20vocabulary%20of%2032K%20or,seem%20to%20be%20sub%2Doptimal..

[33] A. Chowdhery, S. Narang, *et al.*, *Palm: Scaling language modeling with pathways*, 2022. arXiv: 2204.02311 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2204.02311.

[34] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia Tools and Applications*, vol. 82, no. 3, pp. 3713–3744, 2022. [Online]. Available: https://doi.org/10.1007/s11042-022-13428-4.

[35] N. S. Sohoni, C. R. Aberger, M. Leszczynski, J. Zhang, and C. Ré, *Low-memory neural network training: A technical report*, 2022. arXiv: 1904.10631 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1904.10631.

[36] D. Soydaner, "Attention mechanism in neural networks: Where it comes and where it goes," *Neural Computing and Applications*, vol. 34, no. 16, pp. 13 371–13 385, May 2022, ISSN: 1433-3058. DOI: 10.1007/s00521-022-07366-3. [Online]. Available: http://dx.doi.org/10.1007/s00521-022-07366-3.

[37] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, Dec. 2022, ISSN: 0360-0300. DOI: 10.1145/3530811. [Online]. Available: https://doi.org/10.1145/3530811.

[38] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu, "Generalizing to unseen domains: A survey on domain generalization," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, Jan. 2022. DOI: 10.1109/tkde.2022.3178128. [Online]. Available: https://doi.org/10.1109/tkde.2022.3178128.

[39] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1204–1213, Jun. 2022. DOI: 10.1109/cvpr52688.2022.01179. [Online]. Available: https://doi.org/10.1109/cvpr52688.2022.01179.

[40] H. Lee, J. Kim, J. Willette, and S. J. Hwang, "Sea: Sparse linear attention with estimated attention mask," *arXiv preprint arXiv:2310.01777*, 2023.

[41] P. H. Rahmath, V. Srivastava, and K. Chaurasia, "A strategy to accelerate the inference of a complex deep neural network," in *Lecture notes in networks and systems*. Jan. 2023, pp. 57–68. DOI: 10.1007/978-981-19-7615-5_5. [Online]. Available: https://doi.org/10.1007/978-981-19-7615-5_5.

[42] B. Zhuang, J. Liu, Z. Pan, H. He, Y. Weng, and C. Shen, "A survey on efficient training of transformers," *International Joint Conference on Artificial Intelligence*, pp. 6823–6831, Aug. 2023. DOI: 10.24963/ijcai.2023/764. [Online]. Available: https://doi.org/10.24963/ijcai.2023/764.

[43] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma, "Explaining neural scaling laws," *Proceedings of the National Academy of Sciences*, vol. 121, no. 27, Jun. 2024. DOI: 10.1073/pnas.2311878121. [Online]. Available: https://doi.org/10.1073/pnas.2311878121.

[44] S. Cao, S. Liu, T. Griggs, P. Schafhalter, X. Liu, Y. Sheng, J. E. Gonzalez, M. Zaharia, and I. Stoica, "Moe-lightning: High-throughput moe inference on memory-constrained gpus," *arXiv preprint arXiv:2411.11217*, 2024.

[45] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[46] C. Lou, Z. Jia, Z. Zheng, and K. Tu, "Sparser is faster and less is more: Efficient sparse attention for long-range transformers," *arXiv preprint arXiv:2406.16747*, 2024.

[47] X. Ma, Y. Wang, G. Jia, X. Chen, Z. Liu, Y.-F. Li, C. Chen, and Y. Qiao, "Latte: Latent diffusion transformer for video generation," *arXiv preprint arXiv:2401.03048*, 2024.

[48] H. Nguyen, N. Ho, and A. Rinaldo, "Sigmoid gating is more sample efficient than softmax gating in mixture of experts," *arXiv preprint arXiv:2405.13997*, 2024.

[49] OpenAI, J. Achiam, S. Adler, *et al.*, *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.08774.

[50] A. R. Sajun, I. Zualkernan, and D. Sankalpa, "A historical survey of advances in transformer architectures," *Applied Sciences*, vol. 14, no. 10, p. 4316, May 2024. DOI: 10.3390/app14104316. [Online]. Available: https://doi.org/10.3390/app14104316.

[51] Z. Yang, W. Gao, C. Luo, L. Wang, F. Tang, X. Wen, and J. Zhan, *Quality at the tail of machine learning inference*, 2024. arXiv: 2212.13925 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2212.13925.

[52] TensorFlow, *Sparsecategoricalcrossentropy | tensorflow core v2.9.1*, https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy, Accessed: 2025-01-20, 2025.

Appendix - Work Plan