AN IMPROVED TEXT-TO-TEXT TRANSFER TRANSFORMER (T5) FOR TEXT SUMMARIZATION OF NEWS ARTICLES

Charles Munyao Muia

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science of Murang'a University of Technology

July, 2024

DECLARATION

Student Declaration

I hereby declare that this Thesis is my original work and to the best of my knowledge has not been presented for a degree award in this or any other university. I understand what plagiarism is and I am aware of the University's policy in this regard. I understand that any false claim in respect of this work shall result in disciplinary action in accordance with the University Anti-Plagiarism Policy.

_____ _____

Charles Munyao Muia Date

SC400/5708/2021

Supervisors' Approval

The undersigned certify that they have read and hereby recommended for acceptance of Murang'a University of Technology a thesis entitled **"An Improved Text-to-Text Transfer Transformer (T5) for Text Summarization of News Articles."**

_____ _____

Dr. Aaron Mogeni Oirere, PhD Date

Department of Computer Science,

Murang'a University of Technology.

_____ _____

Dr. Rachael Njeri Ndung'u, PhD Date

Department of Information Technology,

Murang'a University of Technology.

DEDICATION

This thesis is dedicated to my beloved parents, Mr. and Mrs. J.M. Nzioka. You have sacrificed so much to provide me with the best opportunities in life, and your belief in me has been my greatest source of motivation. You have always believed in me, even when I doubted myself, which has carried me through the most challenging moments. Your dreams for my future have fueled my determination and perseverance. I am deeply grateful for the countless ways you have nurtured and guided me and the immense love you have always shown. This achievement is as much yours as it is mine. This thesis is also dedicated to my wonderful sisters, Charity Kaluki and Annastacia Mueni, who have always been there to lift me during difficult times and to celebrate my achievements. Our bond has been a pillar of support throughout this research, and I am deeply thankful for the love and encouragement that you have always provided. This thesis is a testament to the love, support, and encouragement I have received from the entire family. Therefore, I dedicate it to you with immense gratitude and heartfelt appreciation.

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to God for His abundant blessings, guidance, and strength throughout this academic journey. Without His grace, this achievement would not have been possible. I am profoundly grateful to my supervisors, Dr. Aaron Oirere and Dr. Rachael Ndung'u, whose expertise, guidance, and encouragement, along with their invaluable feedback and unwavering support, have been instrumental in completing this thesis.

I sincerely thank the Ministry of Education through the Murang'a East Sub County Education Office. The understanding and flexibility of the director of education, Mr. S.K. Ruitha, allowed me to balance my responsibilities and academics effectively. I am deeply thankful for the opportunities and support the entire office provided. Special thanks go to Prof. Geoffrey Muchiri, Dr. Stephen Kahara, Kevin Agina, and Jackson Kamiri, for whom I am deeply thankful for the support and insights you shared. To my MSc. Computer Science and IT classmates, thank you for the camaraderie, support, and shared experiences, which made this journey enjoyable and fulfilling.

Lastly, to anyone who has contributed to my academic journey, whether directly or indirectly, please know that your support and encouragement have been greatly appreciated. This achievement is a collective success, and I am deeply grateful to each of you.

ABSTRACT

Text summarization is a Natural Language Processing (NLP) task that condenses textual content while preserving essential information. With the exponential growth in digital news content, efficient summarization techniques have become imperative. Transformers have revolutionized tasks in natural language processing, including text summarization, due to their self-attention mechanism and adeptness at capturing long-range dependencies. However, these transformers are not task-specific optimized, are sensitive to hyperparameter configuration, and are associated with significant training and inference costs, which pose a challenge. Therefore, the main objective of this study was to develop an improved transformer-based model optimized for text summarization of news articles to solve the stated limitations. A true experiment was adopted for the research design, specifically, the pretest-posttest control group design. Four transformers, Generative Pretrained Transformers (GPT), Text-to-Text Transfer Transformer (T5), Bidirectional and Auto-Regressive Transformer (BART), and Pre-training with Extracted Gap-sentences for Abstractive Summarization (PEGASUS), were subjected to a comparative stud on the CNN/DailyMail dataset. The T5 was identified as the baseline model due to its text-to-text framework and the encoder-decoder architecture. Subsequently, the T5 was improved through adaptive model quantization to reduce training and inference costs, hyperparameter optimization using manual tuning and gradient descent optimization to optimize its performance, and output layer modification by incorporating temperature and top-k sampling parameters to improve the quality of the generated summary. The improved T5 model was trained on the CNN/DailyMail dataset and evaluated on the XSUM and CNN/DailyMail datasets using ROUGE and BLEU metrics. The improved T5 model outperformed the baseline T5 across the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) and Bilingual Evaluation Understudy (BLEU) evaluation metrics on both benchmark datasets. The improved T5 model cumulatively outperformed the baseline T5 on CNN/DailyMail by 3.05% on ROUGE metrics and 4.2% on the BLEU score, and on the XSUM dataset by 7.27% in ROUGE metrics and 6.53% in BLEU score. The findings demonstrated that the improved T5 model outperformed the baseline across two evaluation metrics on both datasets, indicating improved summarization and generalization capabilities. This study has underscored the training and inference efficiency gains from quantization, the pivotal role of hyperparameter optimization in model convergence and generalization, and the effectiveness of output layer modification in improving text summarization using transformers. The improved T5 model can be deployed in real-world settings or applied in other domains beyond text summarization and utilized for multi-document summarization.

TABLE OF CONTENTS

DECLARATION ..... **ii**
DEDICATION ..... **iii**
ACKNOWLEDGEMENT ..... **iv**
ABSTRACT ..... **v**
LIST OF TABLES ..... **ix**
LIST OF FIGURES ..... **xi**
LIST OF EQUATIONS ..... **x**
ACRONYMS AND ABBREVIATIONS ..... **xii**
CHAPTER ONE: INTRODUCTION **1**
1.1 Background of the Study 1
1.2 Problem Statement 4
1.3 Research Objectives 5
1.3.1 General Objective 5
1.3.2 Specific Objectives 5
1.4 Research Questions 5
1.5 Significance of the Study 5
1.6 Scope of the Study 6
1.7 Limitations of the Study 6
1.8 Contributions of the Thesis 7
1.9 Organization of the Thesis 7
CHAPTER TWO: LITERATURE REVIEW **9**
2.1 Introduction 9
2.2 The Transformer Architecture 9
2.3 Existing Transformers for Text Summarization 14
2.3.1 BART (Bidirectional and Auto-Regressive Transformers) 14
2.3.2 T5 (Text-to-Text Transfer Transformer) 15
2.3.3 PEGASUS 17
2.3.4 GPT (Generative Pre-Trained Transformer) 18
2.4 Existing Techniques for Developing Transformer-based Models 20
2.4.1 Pre-Trained Transformers for Text Summarization 20
2.4.2 Transfer Learning 22
2.4.3 RAG (Retrieval Augmented Generation) 25
2.4.4 Hyperparameter Optimization 27

2.4.5 Model Quantization. 29
2.5 Evaluation Metrics in Text Summarization 31
2.5.1 BLEU Score 32
2.5.2 ROUGE Metrics 33
2.5.3 METEOR 34
2.5.4 Human Evaluation 35
2.6 Summary 35
CHAPTER THREE: METHODOLOGY 37
3.1 Introduction 37
3.2 Research Design 37
3.3 Research Process 38
3.4 Data Acquisition 39
3.4.1 Dataset Description 39
3.4.2 Data Pre-processing 40
3.5 A Comparative Study of Text Summarization Transformers 41
3.6 Developing the Improved Model 43
3.6.1 Model Quantization 44
3.6.2 Hyperparameter Optimization 46
3.6.3 Modifying the Model's Output Layer 49
3.6.4 Model Training and Validation 51
3.7 Evaluating the Performance of the Improved Model. 54
3.8 Experimental Materials 55
3.9 Ethical Considerations 56
3.10 Summary 57
CHAPTER FOUR: RESULTS AND DISCUSSION 58
4.1 Introduction 58
4.2 The Comparative Study of Transformers for Text Summarization 58
4.2.1 Model's Performance on Text Summarization 59
4.2.2 Training and Validation Loss of the Models 60
4.3 The Improved Transformer-based Model. 65
4.3.1 Model Quantization 65
4.3.2 Hyperparameter Optimization 67
4.3.3 Modifying the Model's Output Layer 68
4.3.4 The Architecture of the Improved Model 70

4.4 Evaluation Results of the Improved Model 71
4.4.1 Model Performance on ROUGE Metrics 72
4.4.2 Model Performance on BLEU Score 73
4.4.3 Training and Validation Loss of the Model. 74
4.4.4 Comparing the Improved Model with the Existing Models. 75
4.5 Discussion 77
4.5.1 The Comparative Study of Transformers. 77
4.5.2 The Improved Transformer-based Model. 78
4.5.2.1 Model Quantization 78
4.5.2.2 Impact of Hyperparameter Optimization on Model Performance. 79
4.5.2.3 Modifying the Model's Output Layer. 81
4.5.3 The Evaluation Results of the Improved Model 82
4.6 Summary 83
CHAPTER FIVE: CONCLUSION, RECOMMENDATIONS AND FUTURE WORK 84
5.1 Conclusion 84
5.2 Recommendations 85
5.2.1 Real-World Deployment 86
5.2.2 Generalizing the Improved Model to other Domains 86
5.2.3 Multi-Document Summarization Utilization 86
5.3 Future Work 86
5.3.1 Inclusion of Additional Model Optimization Techniques 87
5.3.2 Fine-Tuning for Multilingual Summarization 87
5.3.3 Introduction of Interactive Summarization Approaches 87
REFERENCES 89
APPENDICES 101

LIST OF TABLES

Table 3.1: Dataset Distribution in the CNN/DailyMail and the XSUM Datasets. 40
Table 3.2: Experimental Materials....56
Table 4.1: Model Variations Across Parameters, Layers, and Architecture....58
Table 4.2: ROUGE Metrics in the Comparative Study 60
Table 4.3: Performance of Different Model Quantization Schemes....65
Table 4.4: A Suite of Optimal Hyperparameters. 67
Table 4.5: Comparing the Default and the Modified Output Layer....69
Table 4.6: Evaluation Results on ROUGE Metrics. 72
Table 4.7: Evaluation Results on BLEU Score....73
Table 4.8: Comparing the Improved Model with Existing Models 76

LIST OF FIGURES

Figure 2.1: The Transformer Architecture 10
Figure 2.2: MLM and NSP Pre-training Objectives 21
Figure 2.3: Transfer Learning in NLP 23
Figure 2.4: RAG Architecture 26
Figure 3.1: The Research Process 39
Figure 3.2: Hyperparameter Optimization 48
Figure 4.1: Training and Validation Loss – PEGASUS 61
Figure 4.2: Training and Validation Loss - GPT 62
Figure 4.3: Training and Validation Loss - BART 63
Figure 4.4: Training and Validation Loss – T5 64
Figure 4.5: The Architecture of the Improved T5 Model 71
Figure 4.6: Training and Validation Loss – Improved T5 Model 74

LIST OF EQUATIONS

Equation 2.1: Attention Mechanism 11
Equation 2.2: Multihead Attention 11
Equation 2.3: Feed-forward Neural Network 12
Equation 2.4: Positional Encoding 12
Equation 2.5: Next Sentence Prediction Loss 21
Equation 2.6: Learning Function 24
Equation 2.7: BLEU Score 32
Equation 2.8: Rouge Metrics 33
Equation 2.9: METEOR 34
Equation 3.1: Gradient Descent 47
Equation 3.2: Hyperparameter Update Rule 47
Equation 3.3: The Temperature Parameter 49
Equation 3.4: Softmax Activation Function with Temperature 50
Equation 3.5: Cross-entropy Loss Function 52

ACRONYMS AND ABBREVIATIONS

ADAM	Adaptive Moment Estimation
BBC	British Broadcasting Corporation
BLEU	Bilingual Evaluation Understudy
BPS	Board of Postgraduate Studies
CNN	Convolutional Neural Network
CLT	Computational Learning Theory
CUDA	Compute Unified Device Architecture
FCL	Fully Connected Layers
FFNN	Feed Forward Neural Network
GPU	Graphics Processing Unit
GSG	Gap Sentence Generation
LLM	Large Language Model
METEOR	Metric for Translation with Explicit Ordering
MLM	Masked Language Modelling
NeurIPS	Neural Information Processing Systems
NLP	Natural Language Processing
NSP	Next Sentence Prediction
PAC	Probably Approximately Correct
RAG	Retrieval Augmented Generation
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROUGE	Recall Oriented Understudy for Gisting Evaluation
SGD	Stochastic Gradient Descent
T5	Text-to-Text Transfer Transformer
UCI	University of California, Irvine
VC	Vapnik-Chervonenkis
XSUM	Extreme Summarization

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

Text summarization is one of the subdomains of Natural Language Processing (NLP) that involves condensing the content of a given text while retaining its essential information [1]. In this era of information overload, text summarization plays a pivotal role in helping users comprehend the key points of extensive documents, saving time and improving information retrieval efficiency [2]. The exponential growth of digital content, especially news articles, calls for efficient methods to sift through and extract valuable insights. Text summarization addresses this need for quick information retrieval, aiding readers, researchers, and professionals often overwhelmed with large volumes of textual data [2], [3]. Therefore, through the condensation of information, summarization facilitates easier comprehension, decision-making, and information extraction from diverse sources.

The ever-increasing volume of daily news content poses a significant big-data challenge for individual readers and news information consumers [4]. As the pace of news reporting accelerates and the number of sources multiplies, it has become increasingly difficult for people to stay informed and digest the key details from lengthy news articles[5]. This information overload problem is particularly acute for news consumers, who may lack the time or attention span to read entire reports yet still require a comprehensive understanding of current events and trends[6], [7]. Reading and synthesizing important information from a large corpus of news articles is demanding and time-consuming. Therefore, developing robust text summarization systems for news content has become a pressing necessity, enabling the extraction of the most salient points concisely and informally.

There are two primary types of text summarization: extractive and abstractive summarization[8]. Extractive summarization involves identifying and selecting crucial sentences or phrases directly from the source text[9], [10]. This method relies on algorithms to pinpoint the most significant segments and assemble them effectively to create a summary. Extractive summarization is advantageous in preserving the original wording and coherence of the source but may face challenges when dealing with redundancy or the need to rephrase information for conciseness.

Abstractive summarization alternatively takes a more dynamic approach by generating entirely new sentences to convey the key information in a condensed form. This method involves a higher level of natural language understanding, as the system must interpret and rephrase content to capture the essence of the original content while maintaining conciseness[11]. Abstractive summarization offers greater flexibility and creativity but poses challenges in maintaining coherence and fidelity to the source material[12]. Both approaches have their merits and drawbacks, and the choice between them often hinges on the specific requirements of the task and the desired balance between preserving original language and generating new content.

In addition to extractive and abstractive methods, the field of text summarization has also explored hybrid techniques that combine the two. Hybrid models aim to leverage the strengths of both approaches, first identifying salient sentences or phrases through extractive means and then feeding this content into an abstractive summarization component to generate a final summary[13]. This hybrid design allows the model to retain the factual correctness of extractive summarization while also introducing abstractive summarization's fluency and abstraction capabilities [14]. Hybrid summarization has shown promising results, often outperforming pure extractive or abstractive approaches, particularly when a balance between conciseness and semantic coherence is required.

The field of text summarization has undergone remarkable milestones and transformation over the past decades, transitioning from rule-based, heuristic-driven approaches to increasingly sophisticated machine learning and deep learning-based techniques[15], [16]. Early methods relied heavily on manually crafted linguistic rules and pattern-matching algorithms to extract salient sentences or phrases from input documents[17]. The introduction of statistical models, such as Latent Semantic Analysis (LSA) and Bayesian models, represented a significant advancement, allowing summarization systems to learn from data and capture the relationships within the text[18], [19].

The breakthrough in deep learning, exemplified by the emergence of neural networks, marked a pivotal moment, enabling the development of end-to-end summarization models to learn contextual representations and generate coherent, abstractive

summaries[20]. More recently, the rise of transformer architectures, with their powerful self-attention mechanisms, has further propelled the field, leading to state-of-the-art summarization models that can capture long-range dependencies and produce highly informative, fluent summaries[21]. This study, therefore, builds upon these milestones to contribute further to the field by leveraging the capabilities of the Transformer architecture.

Transformers, first presented by Vaswani et al. in their influential paper "Attention is All You Need," have revolutionized sequence-to-sequence tasks [22]. This neural network architecture is centered around an attention mechanism, which allows the model to assign dynamic importance to different parts of the input sequence during its processing [23]. Key elements of a Transformer include its encoder-decoder structure, multi-head and self-attention mechanisms, and position-wise feed-forward networks. Transformers have demonstrated exceptional effectiveness in a variety of natural language processing (NLP) tasks, achieving leading performance in machine translation, sentiment analysis, and text generation [24]. As a class of neural networks, Transformers have transformed the field of NLP by consistently surpassing previous performance benchmarks across numerous language tasks since their introduction.

The core of the transformer architecture is the self-attention mechanism [22], [25]. This mechanism ensures the transformer assigns different attention weights to different sections of the input sequences according to their relevance to the current position. Three vectors, Key (K), Query (Q), and Value (V), which are the linear projections of the input embeddings, are used to represent each token [26]. Next, a weighted sum of the value vectors is calculated using the attention weights acquired from the softmax operation. As observed in [27], the attention score between the vectors is computed using a dot product to calculate the relevance between corresponding tokens and then transformed into probabilistic distributions by a softmax activation function.

The transformer architecture also includes an encoder-decoder architecture that processes both input and output sequences in parallel [28]. The encoder part processes the input sequence by capturing its contextual information. On the other hand, the decoder part generates the output sequence by focusing on the encoded representation

from the encoder as well as the previously generated tokens. The encoder-decoder components enable the model to effectively handle long-range dependencies by projecting the input into various subspaces and calculating the attention weights for each subspace [29]. The multi-head attention mechanism, another crucial element of transformers, allows the model to focus on different parts of the input sequence simultaneously. Therefore, transformer architecture has made significant advancements in numerous natural language processing tasks and has paved the way for text summarization techniques that can produce more contextually aware and human-like summaries.

1.2 Problem Statement

Transformers have become a game-changer in Natural Language Processing, particularly in tasks such as text summarization [28]. The strength of these transformers is their ability to capture long-range dependencies and to model context. Despite their capabilities, transformers still face challenges that limit their applicability in text summarization and the quality of the generated summary.

One limitation is the significant computational cost during training and inference[30]. According to [31], this has remained to be a challenge due to the numerous matrix multiplications and attention operations, which are computationally intensive. Secondly, transformers are not task-specific optimized. Generalizing the pre-trained transformers to adapt to new tasks is at the expense of reducing the quality and coherence of the generated summaries [32]. The generated summaries may misrepresent the content of the original text, including incorrect information or missing key details. Thirdly, transformers are highly sensitive to hyperparameter configurations, which leads to suboptimal model performance on the default pre-training hyperparameters [33],[34]. Achieving optimal model performance is challenged by this sensitivity, which consequentially results in the generation of suboptimal summaries. Therefore, there is a need for an improved transformer-based model for text summarization of news articles, which is devoid of the above limitations.

1.3 Research Objectives

1.3.1 General Objective

The general objective of this study was to develop an improved transformer-based model for text summarization of news articles.

1.3.2 Specific Objectives

i. To determine features that can improve the performance of a suitable transformer-based model for text summarization of news articles.
ii. To develop an improved variant of the identified model for text summarization of news articles.
iii. To evaluate the improved model on benchmark text summarization datasets.

1.4 Research Questions

i. What features can improve the performance of a suitable transformer-based model for text summarization of news articles?
ii. How to develop an improved model for text summarization of news articles?
iii. How to evaluate the improved model on benchmark text summarization datasets?

1.5 Significance of the Study

The significance of this study lies in its ability to address the critical need for advanced text summarization techniques in the context of news articles. The successful development of an improved transformer-based model for summarizing news articles benefits a wide range of stakeholders, beneficiaries, and consumers of this work. Foremost, news consumers, whether individual readers, content aggregators, or media monitoring services, will significantly appreciate the capacity to quickly digest the key details from lengthy news reports through concise, informative summaries. This will enable more efficient and adequate consumption of the ever-growing daily news content. Beyond the media industry, businesses, policymakers, and researchers who rely on staying informed about current events and trends will also find immense value in access to high-quality, transformer-based text summarization capabilities.

This study is also significant in the context of transformers in natural language processing, given that the application of transformers in news article summarization

remains an active area of research, with existing models still exhibiting room for improvement. Therefore, the development of an improved transformer-based model contributes valuable insights to the broader transformer literature through the various improvements that the researcher explored. The improvements could broadly apply beyond the news domain and inform the development of more powerful and versatile transformer-based models for various natural language processing applications.

1.6 Scope of the Study

The domain of this study lies within Natural Language Processing, one of the main application areas of Artificial Intelligence. In natural language processing, various subtasks exist; however, this study has only focused on text summarization through transformer models. In the context of text summarization, this study only concentrated on news article summarization, specifically in the English language, of all areas and languages in which text summarization can be applied.

The specific scope of this study was confined to systematically investigating the impact of quantization, modifying the output layer, and hyperparameter optimization techniques on the T5 transformer for the specific task of text summarization in the context of news articles. The research, therefore, only focused on assessing how these techniques improve the model performance while reducing the training and inference cost of the T5 model on a text summarization task.

1.7 Limitations of the Study

The study solely relied on secondary data sources, specifically the CNN/DailyMail and XSUM datasets, instead of collecting primary data. While these datasets are widely adopted for summarization tasks, they may not fully represent the specific characteristics or context of the target application domain. The use of secondary data restricts the ability to tailor the data collection process to address specific research questions or to explore emerging trends and unique cases that primary data collection might have revealed. This reliance on pre-existing datasets may have limited the study's ability to address certain subtleties and variations inherent in more customized or contemporary datasets, potentially affecting the applicability of the study's findings to more specific contexts.

1.8 Contributions of the Thesis

The contributions made by this thesis are highlighted below:

1. An improved transformer-based model for text summarization achieved through the following:
    a. A novel hyperparameter optimization technique that leveraged both manual tuning and gradient descent optimization.
    b. An adaptive quantization approach that introduced dynamism into the model quantization process.
    c. Modification of the output layer by incorporating token selection and sequence generation parameters.

2. The study's findings conform to the Computational Learning Theory (CLT), which further extends this theory along the principles and concepts of the VC dimension of model complexity, bias-variance tradeoff, and Probably Approximately Correct (PAC) learning.

1.9 Organization of the Thesis

There are five chapters in this thesis, which are organized in the following order:

Chapter one describes the background of the study, the problem statement, the research objectives and research questions, the study's significance, the scope of the study, the limitations of the study, and the contributions made by this thesis.

Chapter two presents the scholarly literature that was reviewed in this study. The literature synthesizes the existing transformer models for text summarization, techniques for developing these models, and the evaluation metrics used in text summarization.

Chapter three outlines a detailed research methodology followed to achieve the objectives of this study. The Research design, research process, data acquisition and preprocessing, ethical considerations, and experimental materials are covered in this chapter.

Chapter four describes the results obtained from this study and the implication of the findings to transformers for text summarization in the discussion section.

Chapter five presents the conclusion of the findings of this study, the proposed recommendations from the findings, and future research directions that could be adopted for future work.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

This chapter presents the scholarly literature reviewed to lay a foundation for this research. The chapter begins with a detailed description of the general transformer architecture and then presents four state-of-the-art models based on the transformer architecture. A scholarly critique in the form of research gaps is included for each model covered in this section. This is then followed by a section covering the literature on various approaches and techniques used in the design and development of transformers. The researcher proceeds to review the literature on various evaluation metrics for text summarization. A summary of the chapter is then provided at the end.

2.2 The Transformer Architecture

Transformers have emerged as a robust neural network architecture in natural language processing, which adopts a self-attention mechanism to process all the input text concurrently. The attention mechanism makes these transformers more efficient and capable of various Natural Language Processing tasks[29]. The development of Transformers was backed by the need to address the issue of sequence transduction by replacing the sequential nature of recurrent neural networks (RNNs) and the convolutional operations in convolutional neural networks (CNNs), offering improved parallelization and capturing long-range dependencies more effectively[36], [37]. The transformer was introduced in the "Attention is All You Need" seminal paper authored by researchers at Google Brain[22]. This paper was presented at the Neural Information Processing Systems (NeurIPS) conference in 2017. The general architecture of the transformer is illustrated in figure 2.1.

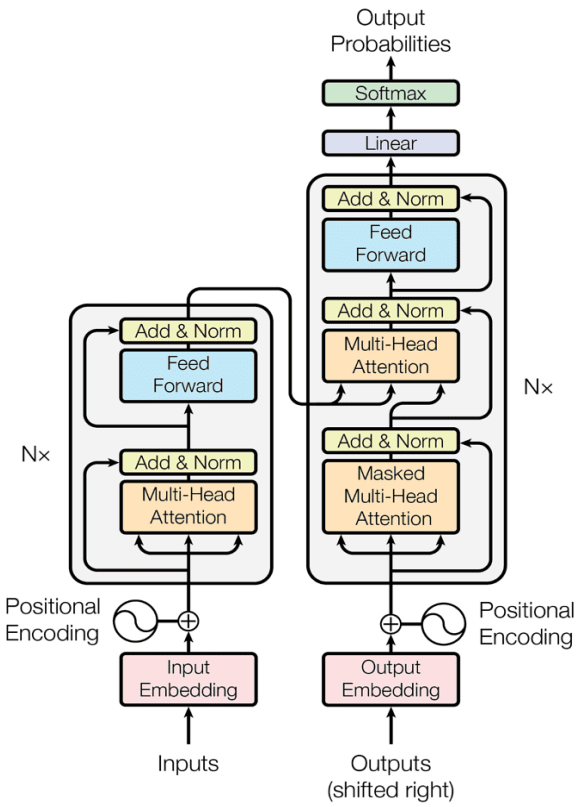


Figure 2.1: The Transformer Architecture (Source: [22])

The Transformer's architecture, illustrated in Figure 2.1, is based on a sequence-to-sequence framework comprising six encoders and six decoders. Each encoder iteratively processes input sequences while the decoder generates output sequences. The encoder on the left side of the architecture maps input sequences to the decoder in a sequence-to-sequence manner[38]. Encoding layers process input iteratively, each building on the contextual information generated by the preceding layer. The decoder, positioned on the right side, utilizes the combined contextual data from the encoders to generate output sequences[39]. The input into the transformer receives a sequence of tokens, which could be words or sub-word units. Each token is converted into a vector through learned embeddings to capture the semantic information needed for subsequent processing.

The self-attention mechanism in the transformer is tasked with attending to the pertinent information throughout the encoding and decoding phases. This self-attention mechanism in a transformer operates on a series of input vectors, typically word embeddings that represent different words in a particular sentence or the tokens

in a sequence[26]. Each word or token can attend to all other words or tokens in the sequence, including itself, using the self-attention mechanism to assess its significance or relevance to the different words or tokens. The three weight matrices learned by transformers for each attention unit are the Query, Key, and Value weights: $W_{Q,}$ $W_{K,}$ *and Wv*, respectively[40]. The softmax activation function is used to express how attention is calculated for the tokens using a matrix computation that represents attention as:

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

Equation 2.1: Attention Mechanism (Source:[40])

Here,

Q, K, and V are the matrices

Softmax normalizes the value of *a* to a 0 to 1 scale

d_k is the square root of the dimension of the key vectors

The transformer architecture also contains a multi-head attention mechanism to allow for the parallel processing of attention heads in different representation subspaces. Each attention head in the multi-head attention mechanism learns a different input data representation. Each attention head can, therefore, concentrate on capturing distinct data components, such as grammatical structure, positional information, or semantic meaning, by projecting the various parts of the input sequence into distinct subspaces[41]. This representation diversity allows the model to extract a more comprehensive understanding of the input data. The Multi-head attention mechanism can be formally defined as follows:

$$\text{MhA} = \text{Concat}_{i\in[\ \#\text{heads}\ ]}\left(\text{Attention}\left(XW_i^Q, XW_i^K, XW_i^V\right)\right)W^O$$

Equation 2.2: Multihead Attention (Source: [41])

Here;

Matrix X is the summation of the token embeddings

Matrices W_i^Q, W_i^K, *and* W_i^V are projection matrices of each head i

W^o *is* a summated projection matrix of the multi-head attention unit

Following the multi-head attention layer, each token's representation is passed through a position-wise feed-forward neural network. The FNN consists of two linear transformations separated by a non-linear activation function, such as the Rectified Linear Unit (ReLU)[42]. The FNN aims to introduce non-linearity and increase the model's capacity to capture the complex patterns in the data. This network enables the model to learn complex representations of the input tokens by applying ReLU activations between linear transformations, which facilitates higher-level abstraction and better generalization to unseen data[42]. The equation of the FNN is as follows:

$$\text{FNN}\ (x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Equation 2.3: Feed-forward Neural Network (Source: [42])

Whereby,

X represents the input token representation

W_1 and W_2 denote the learnable weight matrices

b_1 and b_2 are the bias parameters.

Positional encoding is yet another crucial component of the Transformer architecture, which is designed to provide the model with information about the positions of tokens in the input sequence. Since the transformer architecture does not inherently account for the sequential order of tokens, positional encoding is added to the input embeddings to provide positional information into the model's representations[43]. Positional encoding is typically achieved using sinusoidal functions of different frequencies for each position in the sequence[44]. The positional encodings can be represented as;

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

Equation 2.4: Positional Encoding (Source: [43])

Here,

K is the position of an object in the input sequence

d is the dimension of the output space

p is the position function for mapping k in the input sequence

n is a user-defined scalar

i is for mapping column indices to both sine and cosine functions

After computing the positional encodings for each position and dimension, they are added element-wise to the corresponding token embeddings. This addition ensures that the model's representations for each token in the sequence include the token's semantic content and positional information, enabling it to distinguish between tokens based on their positions in the sequence[45]. The Transformer architecture can capture sequential relationships between tokens by incorporating positional encoding, which is essential for language modelling and sequence generation tasks.

The transformer architecture also contains residual connections, also known as skip connections, which are a fundamental building block of transformer models. They work by adding the input of a sublayer directly to its output, creating a connection that allows the information to flow through the network more effectively[46]. This residual connection helps mitigate the vanishing gradient problem in deep neural networks, allowing the model to learn higher-level representations without degrading performance[47]. Layer normalization is another component of the transformer architecture. It normalizes the inputs to a sublayer by their mean and standard deviation computed across the feature dimension[48]. The layer normalization helps maintain a stable distribution of activations throughout the network, preventing internal covariate shifts and facilitating faster and more stable convergence during training[48].

The final layer in the transformer architecture is the output layer, which generates the model's predicted output sequence from the hidden representations produced by the decoder stack[49]. This output layer is a simple linear transformation that projects the decoder's hidden states into the entire vocabulary space. The resulting logit values are then passed through a softmax function, which converts them into a valid probability

distribution over the entire vocabulary[49]. During training, this output layer is optimized to maximize the likelihood of the ground-truth output sequences.

2.3 Existing Transformers for Text Summarization

This section reviews commonly used transformers in the literature, all based on the transformer architecture discussed above. The transformer architecture, with its attention mechanisms and parallel processing capabilities, has become the foundation for various models in text summarization. These models leverage the transformer's ability to capture long-range dependencies and contextual information, leading to significant improvements in text summarization. In this section, the researcher synthesizes the most prevalent transformer-based models to understand their functionality, strengths, and applications in the domain of text summarization and their critique.

2.3.1 Bidirectional and Auto-Regressive Transformers (BART)

BART has emerged as a groundbreaking technique for natural language processing (NLP). BART, which Google researchers first presented, uses unsupervised pre-training and the transformer architecture to learn contextualized word representations[50]. BART adopts a bidirectional approach that considers both the left and right direction to capture each word's context. This bidirectional approach allows this model to capture deeper semantic and syntactic information from the input text[50]. This model was pre-trained using unsupervised learning on a large unlabeled corpus. During pre-training, BART predicts hidden words within sentences and comprehends how sentences relate[50]. The process's ability to capture detailed contextual information enhances its performance on subsequent NLP tasks.

This model can be fine-tuned for a smaller task after initial pre-training by adding task-specific layers that are now trained on labelled data. Fine-tuning adapts BART to the specific task by optimizing the model's parameters using task-specific objectives[51]. The strengths of this model lie in the ability to capture contextualized word representations effectively, leading to a great understanding of sentence-level semantics. Despite its noticeable strengths, BART is computationally expensive in pre-training due to its large-scale architecture, and this architecture's size hinders its deployment in resource-constrained devices.

Various studies have been conducted using this architecture to generate text summaries. Studies such as [52]utilized this model in a Google Collaboratory platform powered by powerful GPUs and has access to TensorFlow and PyTorch. The dataset used in this study was from the German Wikipedia, which was readily accessible and had 100,000 Wikipedia articles and the corresponding reference summary. The transformers library was used to train the model using the Pytorch framework. A batch size of 64 was used, and the most extensive token length for the input and output was set at 1024 and AdamW optimizer, along with a specified learning rate of 10-4 and a weight decay of 10-2. Fine-tuning was conducted for 10 epochs using the AdamW optimizer while maintaining the same learning rate and weight decay value. The model's performance was evaluated, and commendable results were achieved on ROUGE metrics.

BART, while a promising approach to text summarization, has, however, faced criticism within the literature. Studies such as [53] have pointed out its limitations in generating coherent and contextually accurate summaries, especially when dealing with longer texts or complex language structures. Additionally, research by [54]claims that BART struggles with preserving the original context and tone of the input text in its summaries, potentially resulting in misleading or inaccurate representations. While BART has demonstrated promise in text summarization, these critiques underscore the importance of continuous enhancement and evaluation to improve its effectiveness and reliability in text summarization tasks.

2.3.2 Text-to-Text Transfer Transformer (T5)

In recent years, the Text-to-Text Transfer Transformer (T5) has emerged as a trademark model for several natural language processing tasks, including text summarization[55]. T5 is based on the transformer architecture, which Vaswani *et al*. [22] first suggested. T5 has a stack of self-attention mechanisms and feed-forward layers, allowing it to gather contextual information effectively. T5 introduces a "text-to-text" framework whereby tasks, including text summarization, are framed as text generation problems. The strength of this model lies in leveraging its large-scale pre-training to capture contextual information effectively, leading to coherent and informative summaries.

The T5 model comes in several variants that differ primarily in size and capacity to suit various computational resources and performance requirements. These variants include T5-Small, T5-Base, T5-Large, T5-3B, and T5-11B, which respectively contain approximately 60 million, 220 million, 770 million, 3 billion, and 11 billion parameters. The T5-Small and T5-Base models are designed for tasks that require faster inference and less memory, making them suitable for real-time applications or environments with limited hardware resources. The larger models, such as T5-3B and T5-11B, are designed to deliver superior performance on complex language tasks due to their higher capacity to capture complex patterns in data. However, they demand significant computational power and memory, making them ideal for research and high-stakes production scenarios where performance outweighs computational cost. Each variant follows the same underlying architecture but scales up in size and depth, allowing users to choose the appropriate model based on their specific needs and resource availability.

T5 employs a pre-training phase to learn general language representations from a vast amount of unlabeled data. It utilizes a masked language-modelling approach analogous to BART, where a random subset of tokens is masked, and the model is tasked with predicting the masked tokens[55]. This pre-training enables T5 to capture rich contextual information and efficiently generalize tasks such as text summarization. After pre-training, T5 is fine-tuned on specific summarization datasets to adapt its knowledge to the summarization task[56]. Fine-tuning involves training the model on labelled data, where input-output pairs of document summaries are provided. T5 uses a sequence-to-sequence framework with encoder-decoder layers, where the input text is tokenized for the decoder to create the desired summary[57].

The T5 transformer has been used in various studies. For instance, this study [58]summarized text using the T5 transformer on a drugs review dataset annotated by the UCI (University of California, Irvine). Human summaries were manually created for 10 of the most relevant reviews of 500 specific drugs from the entire dataset. Following that, the T5 model was improved to provide an abstractive summary, with the dataset being divided into 80% for training, 10% for testing, and 10% for validation. ROUGE-1, ROUGE-2, and ROUGE-L received scores of 31.62, 22.58, and 26.53 to assess the model's efficiency.

However, the researcher contests that a different set of hyperparameter configurations would have yielded better results since the authors used a conservative learning rate of 0.0005 and a weight decay value of 0.001 while excluding the layer normalization weights. The author of the work [58] also recommended that future researchers use a task-specific dataset that is more robust to enhance the model's performance. Despite recording competitive scores, the model was still challenged in its computational cost, specifically during training and inference. The huge parameter counts in this model translated into a quadratically growing training and inference cost, which resulted in scalability issues and high computational requirements in its training and deployment.

2.3.3 PEGASUS

PEGASUS (Pre-training with Extracted gap sentences for Abstractive Summarization) is a transformer-based model that has emerged as a powerful tool for text summarization in natural language processing[59]. The core PEGASUS architecture follows the standard encoder-decoder structure common to many transformer models, leveraging the power of self-attention to capture long-range dependencies in the input text[60]. Several variants of PEGASUS have been developed, scaling the model size and depth to achieve increased performance. The base PEGASUS model consists of 12 transformer layers and approximately 223 million trainable parameters. Larger variants, such as PEGASUS-Large, expand the model depth to 16 layers and increase the parameter count to 568 million[59].

In PEGASUS, pre-training is conducted through two self-supervised objective functions: Masked Language Modeling (MLM) and Gap Sentence Generation (GSG)[56]. As a pre-training objective in transformer-based models, MLM involves randomly masking tokens within input sequences and training the model to predict the masked tokens based on the surrounding context[61]. Conversely, as a pre-training objective, GSG replaces the entire encoder input sentences with a second mask token and inputs them into the decoder. Like a standard auto-regressive transformer decoder, the decoder employs a causal mask to conceal future words [62]. This objective enables PEGASUS to learn how to generate abstractive summaries by inferring missing information and encouraging a deeper understanding of text coherence and relevance within the summarization context.

Numerous studies have utilized this model for text summarization; for instance, this study [63] used a 12-layer baseline Pegasus model with approximately 223M trainable and non-trainable parameters. The study used varying batch sizes for 100 iterations, with a gradual learning rate of 0.0002 and a model checkpoint decided by the optimal validation performance. However, the researcher critiques this hyperparameter combination, and as evident in their published results, the model converged in the local minima instead of the global minima. This is undesirable since the model ends up stuck in suboptimal solutions, as seen in its unusually high model loss of 3.24%.

Even though Pegasus represents a significant advancement in abstractive summarization techniques, studies such as [30] have pointed out scalability and computational cost challenges. Pegasus relies on large-scale pre-training and fine-tuning procedures that call for substantial computational resources and infrastructure, which may pose barriers to widespread adoption, especially for researchers and practitioners with limited computational resources. In addition, Pegasus struggles with handling out-of-domain or ambiguous inputs, as noted in this study [64], potentially leading to suboptimal summarization performance, particularly in longer documents or complex texts.

2.3.4 Generative Pre-Trained Transformer (GPT)

Generative Pre-Trained Transformer (GPT) represents a pioneering approach in natural language processing (NLP), characterized by its transformer-based architecture and sequence generation capabilities. Developed by OpenAI, GPT models are renowned for generating coherent and contextually relevant text based on input prompts. The architecture of GPT models consists of multiple layers of transformer blocks, each comprising self-attention mechanisms and feed-forward neural networks[65]. Notably, GPT employs a decoder-only architecture, meaning it generates text autoregressively, token by token, based on previously generated tokens without access to future context.

GPT models are pre-trained on large-scale corpora to learn contextualized representations of language. During pre-training, GPT models typically employ unsupervised learning objectives such as language modelling and masked language modelling (MLM). These objectives involve predicting the next token in a sequence

given previous context or reconstructing masked tokens within a sentence, encouraging the model to learn meaningful language representations[61]. In MLM, a portion of tokens within the input sequence are masked randomly, and the model's goal is to predict these masked tokens using the surrounding context. This pretraining objective encourages the model to capture bidirectional dependencies and contextual information from both preceding and subsequent tokens by training the model to fill in the masked tokens[56]. This enables GPT models to robustly understand language semantics and syntactic structures, enhancing their performance in various natural language processing tasks, including text generation and summarization.

Numerous studies have leveraged GPT models for text summarization tasks and recorded competitive results across various domains and datasets. For instance, a researcher in [66] employed a GPT-based approach to summarize news articles in the Chinese language. The researcher fine-tuned the pre-trained GPT model on a task-specific dataset using supervised and semi-supervised learning techniques. Additionally, a study by [67] explored reinforcement learning and multi-task learning techniques to further enhance the performance of GPT-based summarization systems. The findings from these studies suggest that GPT models exhibit promising performance in generating concise and informative summaries, albeit with certain limitations regarding factual accuracy and coherence, particularly in longer texts.

Although studies that have used this model show that it can perform well in summarization, it is still limited because of its pretraining objective, which is not text summarization. The unidirectional nature of GPT models restricts their ability to fully grasp bidirectional context and dependencies within a text. This limitation becomes particularly pronounced in tasks that require language understanding or modelling long-range dependencies, such as text summarization or document comprehension[65]. Due to their autoregressive nature, decoder-only GPT models generate text token by token, leading to a tendency for repetition and a lack of global coherence in the generated sequences. Additionally, the reliance on token-level predictions during training hinders the model's comprehension of higher-level structures and semantic relationships within the text[24]. These challenges highlight the need for further exploration and refinement of decoder-only architectures to

overcome their intrinsic limitations and enhance their effectiveness across a broader spectrum of NLP tasks.

2.4 Existing Techniques for Developing Transformer-based Models

In the natural language processing (NLP) domain, transformer-based models' emergence has revolutionized the language understanding and generation landscape. Therefore, this section synthesizes various techniques and approaches for designing, developing, and refining these transformer-based models. The researcher reviewed techniques such as pre-training strategies, model fine-tuning, and hyperparameter optimization, among others, to provide a comprehensive overview of the methodologies and best practices employed in developing transformer models.

2.4.1 Pre-Trained Transformers for Text Summarization

Pre-training is important in developing transformer models, generally for large-scale language models. It involves training the model on a massive corpus of unlabeled text data, allowing it to learn general language representations and patterns. The pre-training phase aims to capture the underlying statistical properties of the language, such as syntax, semantics, and contextual information[68]. These pre-trained models can be fine-tuned on specific downstream tasks, significantly improving performance and reducing the need for large amounts of labelled data.

Two popular pre-training objectives for transformers are masked language modelling (MLM) and next sentence prediction (NSP)[69]. In the Masked Language Modelling pre-training objective, a certain percentage of tokens in the input sequence are randomly masked. The model is then tasked with predicting the masked tokens based on the surrounding context[61]. MLM objective encourages the transformer to learn meaningful representations of words and phrases within the context of the entire input sequence by training the model to predict missing tokens. Studies have demonstrated the effectiveness of the Masked Language Model (MLM) pre-training objective in the context of the BERT (Bidirectional Encoder Representations from Transformers) model. This objective helps the model capture the syntactic and semantic relationships between words and generates coherent and contextually appropriate outputs during fine-tuning for a downstream task[61]. Figure 2.2 shows how the two pretraining objectives, MLM and NSP, are achieved in transformer-based models.

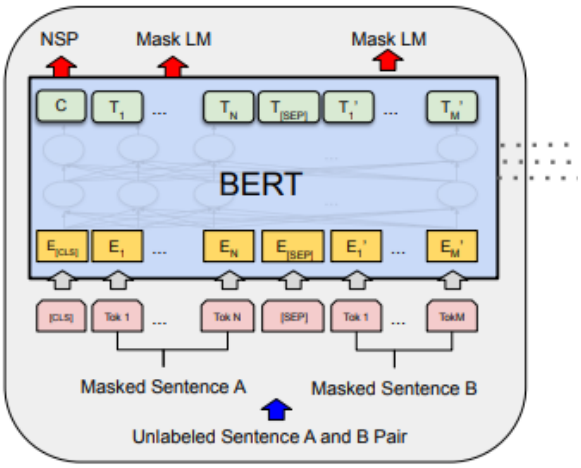


Figure 2.2: MLM and NSP Pre-training Objectives (Source: [69])

The Next Sentence Prediction pre-training objective focuses on the relationship between pairs of sentences. During pre-training, as observed in this study [69], the model is presented with pairs of sentences and is tasked with predicting whether the second sentence follows the first one in the original text. This objective encourages the model to understand the discourse structure and coherence between sentences, facilitating its ability to generate cohesive and contextually relevant text[70]. NSP is particularly useful in tasks that involve understanding and generating multi-sentence texts, such as text summarization and question answering. Mathematically, it can be represented using the following loss function equation for NSP:

$$\text{NSPloss} = -\sum_{i=1}^{N} \left(y_i \log\ P(y_i) + (1 - y_i) \log\left(1 - P(y_i)\right) \right)$$

Equation 2.5: Next Sentence Prediction Loss (Source:[70])

Here,

N is the total number of sentence pairs.

y_i Represents the true label (1 for true, 0 for false), indicating whether the second sentence follows the first one.

$P(y_i)$ Is the predicted probability of the next sentence being true.

Several technical considerations are crucial for successful pre-training of Transformer models. For instance, the choice of pre-training objectives significantly impacts the model's learned representations. A study by [71] demonstrated the effectiveness of the above pretraining objectives in capturing contextual word relationships and understanding sentence-level coherence. However, recent research by [72] submits that these objectives might not be sufficient for tasks requiring reasoning or factual grounding. This highlights the limitations of relying solely on context-based prediction for pre-training and emphasizes the need for incorporating objectives that encourage logical reasoning, factual grounding, and diverse text generation capabilities into the pre-training process. The quality and size of the pre-training data also play a vital role, and as noted in [72], large, high-quality datasets lead to more robust and generalizable representations.

While pre-training has its benefits in NLP, it is not without limitations. One significant challenge arises from potential biases and factual inaccuracies inherent in large unlabeled text corpora used for pre-training. The study in [71] highlighted how pre-trained models can inherit and amplify societal biases in the training data, leading to factually incorrect outputs. Therefore, careful selection of pre-training data and potentially incorporating debiasing techniques to mitigate the transfer of biases to downstream tasks is necessary.

Another critique associated with pretraining lies in ensuring the meaningfulness and coherence of the learned representations. This challenge stems from the inherent limitations of unsupervised learning objectives commonly used in pre-training[72]. For instance, Masked Language Modeling (MLM), a prevalent pre-training objective, predicts masked words based on context. However, this objective can lead the model to prioritize statistically frequent co-occurrences over capturing deeper semantic relationships.

2.4.2 Transfer Learning

Transfer learning is a technique that leverages knowledge gained from one task or domain to improve performance on a related but different task or domain. In the context of transformers, transfer learning involves using the weights of a pre-trained model as a starting point for training on a new task or domain[55]. Transfer learning

is instrumental when dealing with limited labelled data for a specific task or domain, as it allows the model to leverage the knowledge acquired from large-scale pre-training on diverse data sources. Figure 2.3 is a diagrammatic representation of the transfer learning process in transformers.

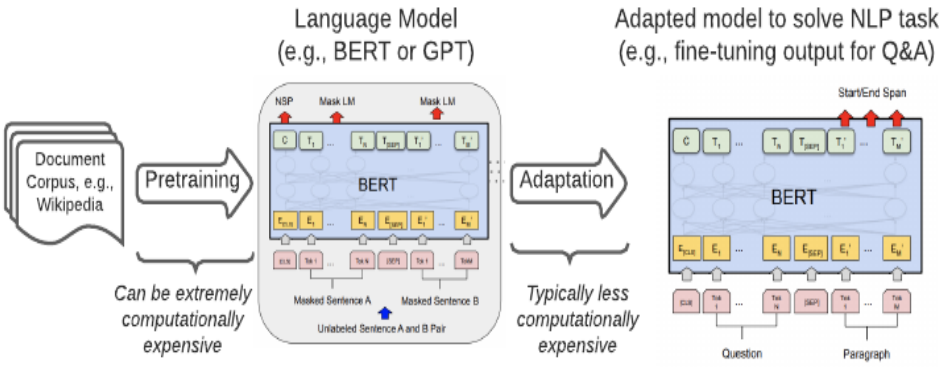


Figure 2.3: Transfer Learning in NLP (Source: [73])

The pre-trained model's weights can be used differently for transfer learning. Feature extraction is one of the techniques in which the pre-trained model is used as a feature extractor, and only the final layers are trained on the new task or domain. Feature extraction is a widely explored approach in the context of NLP and pre-trained models. This study has [73] revealed that the intermediate representations learned by these models during pretraining can effectively be used as a starting point for downstream tasks. The study has also highlighted that the lower layers of pre-trained transformer models capture more general, task-agnostic features, while the higher layers learn more task-specific representations.

For instance, this study [74] demonstrated that freezing the lower layers of the pre-trained model and treating them as a fixed feature extractor while fine-tuning only the final layers on the target task can be particularly effective when the target task is closely related to the pretraining objective or when the available labelled data for the target task is limited. This study also demonstrated that feature extraction can significantly reduce the computational requirements and training time compared to fine-tuning the entire model, making it a practical solution for resource-constrained environments.

Fine-tuning is the second approach, whereby the entire pre-trained model is fine-tuned on the new task or domain, with all layers updated during training. This is best described as adapting a pre-trained transformer model to a specific downstream task or domain[52]. It involves initializing the model with pre-trained weights and training it on a labelled dataset relevant to the target task. During fine-tuning, the pre-trained model's weights are updated using a smaller learning rate compared to the pre-training phase.

This approach allows the model to retain the general language knowledge acquired during pre-training while adapting to the new task-specific data. Fine-tuning can significantly improve performance on various NLP tasks, such as text classification, named entity recognition, question answering, and summarization. The learning function in the process of fine-tuning can be represented as;

$$\theta_{\text{fine-tuned}} = \theta \text{argmin}(\mathcal{L}_{\text{task}}(\theta))$$

Equation 2.6: Learning Function (Source: [73])

Here:

$\theta_{\text{fine-tuned}}$ represents the fine-tuned parameters of the transformer model.

$\mathcal{L}_{\text{task}}(\theta)$ denotes the task-specific loss function.

θ represents the parameters of the pre-trained transformer model.

In this equation, the objective of fine-tuning is to minimize the task-specific loss function with respect to the model parameters. The model learns to adjust well to the characteristics of the target task by iteratively adjusting the parameters through gradient descent or other optimization techniques, thereby improving its performance on that task[75]. Fine-tuning enables the model to leverage the knowledge acquired during pre-training while tailoring its representations to the characteristics of the downstream task, ultimately enhancing its effectiveness and adaptability.

The third approach is Multi-Task Learning, whereby the pre-trained model is trained simultaneously on multiple tasks. In multitask learning, a pre-trained language model

is the foundation for learning representations across various tasks[76]. This is achieved by jointly optimizing the model's parameters on diverse tasks so that the model can capture common patterns beneficial for all tasks while also adapting to the specific requirements of each task[76]. One key advantage of multitask learning is its ability to facilitate transfer learning across tasks with varying degrees of data availability. This aspect has been highlighted in studies such as [77], which demonstrated that multitask learning can mitigate data sparsity's impact and improve low-resource tasks' performance by transferring knowledge from tasks with sufficient data.

Numerous studies in the literature have demonstrated the effectiveness of multitask learning in NLP by applying this approach to various NLP tasks, including sentiment analysis, text classification, and named entity recognition, showing that jointly training a single model on multiple tasks improved performance compared to training separate models for each task. Multitask learning has also been used for various sequence tagging tasks such as part-of-speech tagging, chunking, and named entity recognition, establishing that multitask learning can perform well in low-resource tasks and enhance model generalization.

While transfer learning shows promise in leveraging shared representations across tasks, its practical implementation introduces complexities in optimization and may lead to slower convergence or suboptimal solutions[30]. Optimizing multiple objectives simultaneously requires careful tuning of hyperparameters to ensure effective learning across all tasks. Task similarity and pretraining objective compatibility are crucial factors influencing the effectiveness of multitask learning and could potentially result in imbalanced learning dynamics and suboptimal performance[53]. Therefore, while transfer learning offers advantages in knowledge transfer, its implementation requires consideration of task compatibility, optimization strategies, and model architecture design.

2.4.3 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) has become highly regarded in the field of natural language processing, especially for tasks that require extensive knowledge,

such as question answering and text summarization. RAG enhances the accuracy and informativeness of responses by integrating the language comprehension strengths of transformer models with the capability to retrieve and utilize pertinent factual information.

In the RAG approach, a retriever component retrieves relevant documents or passages from a large knowledge base (e.g., Wikipedia) based on the input query or context. The retrieved information is then passed to a generator component, typically a pre-trained transformer-based language model[78]. The generator processes the input and the retrieved data to generate the final output. Figure 2.4 illustrates the RAG architecture.

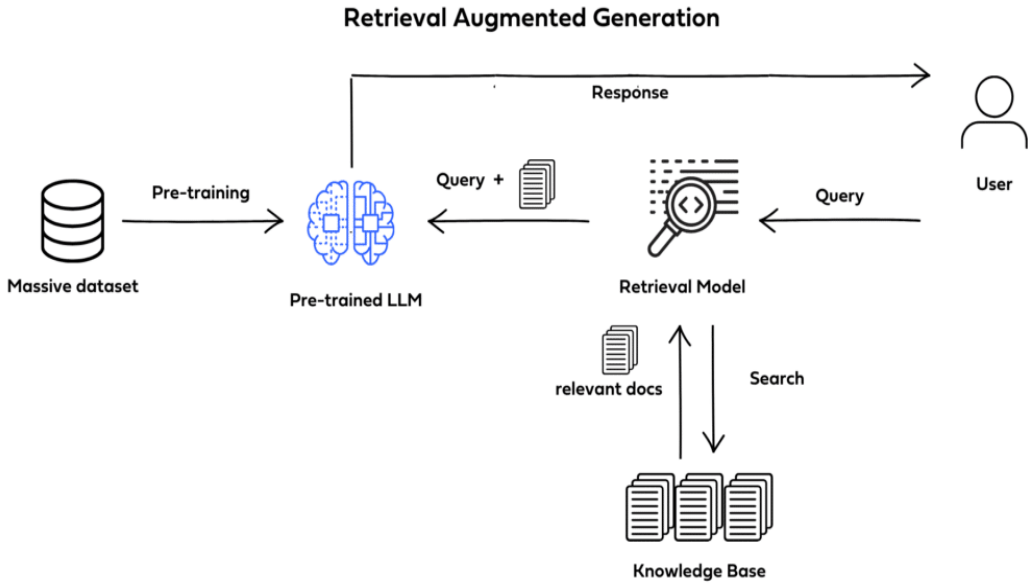


Figure 2.4: RAG Architecture (Source: [78])

The field of Natural Language Processing has demonstrated the effectiveness of RAG models in various knowledge-intensive tasks. For instance, this study [79] applied the RAG architecture and showed significant improvements in performance on open-domain question-answering tasks compared to models that rely solely on language modelling. In the above study, the authors explored the application of RAG models in the context of open-domain dialogue systems, showcasing their ability to provide more informative and relevant responses by leveraging external knowledge sources.

The authors also investigated the impact of different retrieval strategies and knowledge base representations on the performance of RAG models[79]. Their findings highlighted the importance of efficient and effective retrieval techniques and the quality and diversity of the knowledge base used for retrieval.

RAG models have demonstrated promising results; however, they also face several challenges and limitations. One critique is the potential for hallucinations or factual inconsistencies, as the generated output may occasionally deviate from the retrieved information or introduce erroneous details[80]. This issue has been noted by researchers who emphasize the importance of addressing such discrepancies to enhance the reliability and trustworthiness of RAG-generated content. Additionally, the performance of RAG models relies heavily on the quality and coverage of the external knowledge base. Limited or biased knowledge bases may hinder the model's ability to retrieve relevant information accurately, particularly in specialized domains or languages[80]. Researchers have proposed various strategies to mitigate these challenges, including incorporating fact-checking mechanisms, refining the retrieval process with improved ranking and filtering techniques, and enhancing the knowledge base through data augmentation.

2.4.4 Hyperparameter Optimization

Hyperparameter optimization is an important process in developing and refining transformer models, as it significantly impacts their performance, convergence, and generalization capabilities[81]. Hyperparameters are model parameters that are predetermined rather than learned during the training process, and can significantly affect the performance and behaviour of the model. Examples of hyperparameters include learning rate, batch size, and the model's architecture. The goal of hyperparameter optimization is to find the optimal values for these hyperparameters that result in the highest performance of the model on a validation set. This process often involves systematically searching through the hyperparameter space, either exhaustively or using optimization techniques[33]. Selecting appropriate hyperparameter values is a non-trivial task, and various optimization techniques have been proposed and employed in the literature.

One common approach to hyperparameter optimization is manual tuning, where experienced researchers or practitioners adjust hyperparameters based on domain knowledge, intuition, and empirical observations[82]. While this method can be effective, it is time-consuming and relies heavily on human expertise, which may not always be available or optimal. Automated hyperparameter optimization techniques have been widely explored in the literature to address this issue. Grid and random search are popular methods that evaluate the model's performance across a predefined grid of hyperparameter combinations or randomly sampled combinations, respectively[82]. These techniques are straightforward to implement and offer a high-dimensional hyperparameter search space, especially for large transformer models. Grid search has been successfully employed in various transformer-based applications, such as natural language processing (NLP), computer vision, and speech recognition. In the NLP domain, grid search has optimized hyperparameters for transformer models in tasks like machine translation, language modelling, and text summarization.

Other optimization techniques, such as Bayesian and gradient-based optimization, have also gained traction in the literature[83], [84]. Bayesian optimization uses probabilistic models to guide the search toward promising regions of the hyperparameter space, reducing the number of evaluations required. Gradient-based optimization treats differentiable hyperparameters as additional model parameters and optimizes them using gradient descent techniques. Gradient-based optimization for hyperparameter tuning in transformer models has showcased a high potential for efficient and scalable optimization[83]. These optimization methods have been successfully applied to transformer models, demonstrating their effectiveness in finding optimal or near-optimal hyperparameter configurations. For instance, [84]employed Bayesian optimization to tune the hyperparameters of a transformer-based machine translation model, achieving competitive performance while reducing computational costs compared to random search.

While these hyperparameter optimization techniques have shown promising results, they still have challenges and limitations. One critique is some optimization methods' computational complexity and resource requirements, particularly for the high-dimensional, multi-modal, and non-convex hyperparameter spaces encountered in

large-scale transformer models[33]. Additionally, the performance of these approaches may vary based on the specific task, dataset, and model architecture, necessitating careful consideration and domain-specific tuning. Therefore, researchers need to explore combining multiple approaches to hyperparameter optimization or consider developing new techniques for approximations of the objective function to guide the search space for optimization.

2.4.5 Model Quantization

Transformers have transformed the Natural Language Processing domain, achieving remarkable performance in text summarization, machine translation, and question-answering. However, their complex architecture often leads to high computational costs and memory demands, which hinders their deployment in resource-constrained environments[31]. Therefore, quantization techniques offer a promising solution to address the above limitation by reducing transformer models' memory footprint and computational requirements. Quantization reduces the precision of a transformer model's weights and activations, typically from 32-bit floating-point representation to lower-precision formats such as 16-bit or 8-bit integers[85].

Several quantization techniques can be applied to transformers. For instance, the study in [85] demonstrates that quantizing model weights to lower precision formats, such as 8-bit integers (INT8), can significantly reduce memory footprint and potentially accelerate computations. This study showed minimal accuracy degradation for certain NLP tasks when using INT8 quantization. This approach led to a 4x reduction in memory footprint and inference speedup on optimized hardware, demonstrating its effectiveness in compressing the model while maintaining acceptable accuracy. Post-training quantization (PTQ) is another approach that quantizes a pre-trained model after training. A study conducted in [86]showcased the effectiveness of PTQ for Transformers, whereby their results achieved significant memory reduction and speedups with minimal impact on accuracy.

Another common quantization type is weight quantization, where only the model's weights are quantized while the activations remain in full precision. Weight quantization reduces the storage requirements of the model by compressing the parameters, thereby enabling faster inference and a lower memory footprint[87].

However, this approach may lead to accuracy degradation, especially in tasks where fine-grained parameter values are crucial for performance. The study in [87] has established that weight quantization can often achieve significant model size and inference speed-ups with relatively modest accuracy degradation. This is because the transformer's self-attention mechanism is relatively robust to quantization errors in the weights as long as the general patterns of the attention weights are preserved.

Activation quantization is another technique to reduce the numerical precision of activation values within a transformer model. These activation values represent the intermediate outputs calculated during the forward pass of the model[85]. Activation values result from applying linear transformations to the previous layer's outputs before feeding them into the next layer for further processing. According to this study[86], activation quantization has a more direct impact on the intermediate feature representations computed by the transformer layers. Thus, more careful calibration is required to avoid excessive information loss.

Another technique is Quantization-Aware Training (QAT), which incorporates quantization during the training process. Studies have demonstrated that QAT can lead to better performance compared to PTQ, as the model learns from the outset with lower precision representations[88]. The study in [88] directly integrated quantization into the training process, exposing the model to the initial lower-precision representations of weights and activations. This allowed the model to adapt and optimize its internal calculations for the lower precision format. Their findings show that this approach led to superior performance compared to PTQ, making it a promising technique for achieving significant efficiency gains for Transformers without compromising accuracy.

Although quantization techniques have remarkably reduced transformer models' computational and memory footprint, the literature has highlighted several significant limitations and challenges. A key concern is the inherent trade-off between model efficiency and accuracy - aggressive quantization can lead to substantial degradation in the transformer's performance, particularly on demanding tasks that require rich feature representations[89]. Moreover, the effectiveness of quantization is highly

sensitive to the specific model architecture and the target task, making it difficult to develop universal techniques that generalize well[86].

Existing quantization approaches also suffer instability and optimization challenges, often requiring careful hyperparameter tuning and model optimization techniques[89]. Therefore, cautious quantization strategies and methods need to be introduced to help mitigate these trade-offs and maintain high performance while achieving substantial efficiency gains. Additionally, given that the optimal quantization strategy may depend on the specific model, task, and deployment constraints, there is a need for techniques that can dynamically adjust the quantization parameters based on these factors.

2.5 Evaluation Metrics in Text Summarization

Evaluation metrics play a crucial role in assessing the performance and effectiveness of text summarization techniques, both in research and industry applications. These metrics provide quantitative measures to evaluate the quality of generated summaries and the effectiveness of different summarization approaches. Evaluation in text summarization can be broadly categorized into intrinsic and extrinsic evaluation[90]. Intrinsic evaluation focuses on assessing the quality of the summaries directly by examining their coherence, informativeness, and overall quality against the source text. This type of evaluation emphasizes the internal characteristics of the generated summaries and their ability to convey the key information in the original text[90]. In contrast, extrinsic evaluation evaluates how the summarization technique affects the performance of other measures, such as the readability and conciseness measure in the summary.

Various evaluation metrics have been developed and utilized in the field to measure the effectiveness of text summarization techniques. Commonly used evaluation metrics include ROUGE (Recall-Oriented Understudy for Gisting Evaluation), BLEU (Bilingual Evaluation Understudy), METEOR (Metric for Evaluation of Translation with Explicit Ordering), and Human Evaluation. Each metric has strengths and limitations, and researchers often employ multiple metrics to evaluate text summarization comprehensively. The metrics discussed below enable researchers to

identify effective approaches, compare different methods, and advance the text summarization domain by providing an objective measure of summarization quality.

2.5.1 BLEU Score

The BLEU (Bilingual Evaluation Understudy) score is a widely used metric for evaluating the quality of machine-generated text, particularly in machine translation and text summarization[91]. Developed by Papineni *et al.*, BLEU aims to measure the similarity between a generated text and one or more reference texts by comparing their n-gram overlap. BLEU effectively captures the lexical and semantic similarity of the generated text to the original content by focusing on the n-gram overlap between generated summaries and reference summaries[91].

The mathematical definition of this metric involves calculating the geometric mean of the precision scores for individual n-gram orders (usually up to 4-grams). Each precision score represents the ratio of the number of matched n-grams in the generated text to the total number of n-grams in the reference texts. BLEU also incorporates a brevity penalty to penalize shorter generated texts that may artificially inflate precision scores. The BLEU score can be mathematically expressed as follows:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \cdot \log\, p_n\right)$$

Equation 2.7: BLEU Score (Source: [91])

Here,

BP is the brevity penalty, which penalizes shorter generated texts.

N is the maximum order of n-grams considered.

w_n Is the weight assigned to the precision score for n-grams of order n.

p_n Is the precision score for n-grams of order n.

Generally, the widespread use of the BLEU score in text summarization literature reflects its effectiveness as a reliable and widely accepted evaluation metric[35]. Its ability to provide quantitative measures of summarization quality, ease of use, and

adaptability makes it one of the go-to evaluation metrics for researchers and practitioners aiming to develop and evaluate various text summarization techniques.

2.5.2 ROUGE Metrics

In text summarization, the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics have become the de facto standard for assessing the performance of summarization systems. These metrics were introduced by Chin-Yew Lin *et al.* and are based on measuring the overlap between the generated summary and one or more references (human-written) summaries[92].

This metric encompasses several variants, each designed to evaluate different aspects of summary quality. ROUGE-1 focuses on unigram overlap, measuring the similarity between the generated summary and the reference summary based on the occurrence of individual words. ROUGE-2 extends this evaluation to bigrams, considering pairs of consecutive words in the summaries. ROUGE-L evaluates the longest common subsequences between the summaries, which allows for a more flexible assessment of summary similarity, accounting for differences in word order[92]. The ROUGE metric can be defined as:

$$R_{\text{ROUGE-N}} = \frac{\sum_{S \in \{\text{ Reference }\}} \sum_{N_{n-\text{ gram}} \in S} \text{Count}_{\text{match}} \left(N_{n-\text{ gram}}\right)}{\sum_{S \in \{\text{ Reference }\}} \sum_{N_{n-\text{ gram}} \in S} \text{Count}\left(N_{n-\text{ gram}}\right)}$$

Equation 2.8: Rouge Metrics (Source: [92])

Whereby

{reference} denotes the reference summaries.

Count (N_n-gram) represents the count of n-grams in the reference summary.

$\text{Count}_{\text{match}}$ (N_n-gram) is the number of n-grams in the reference and generated summary.

The ROUGE metrics have gained widespread acceptance and are predominantly used in text summarization research, backed by the consulted literature. One of the primary reasons for the widespread adoption of ROUGE is its strong correlation with human judgments of summary quality. Numerous studies have demonstrated that ROUGE scores align well with human evaluations, making it a reliable automatic metric for

assessing the performance of summarization systems[90]. This correlation with human judgments lends credibility to using ROUGE as a substitute for manual evaluation, which is often time-consuming and expensive. Therefore, this metric's widespread adoption and endorsement in the text summarization research community attest to its reliability, practicality, and effectiveness as an automatic evaluation metric.

2.5.3 METEOR

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is an evaluation metric for text summarization and machine translation. It combines precision, recall, and alignment-based measures to assess the overall fluency and quality of the generated summary. METEOR considers several linguistic and semantic elements, such as sentence stemming, synonymy matching, and exact word matching, to evaluate the summary quality [93]. Recall is given a higher weight than precision, and the measure is based on the harmonic mean of unigram precision and recall.

The METEOR metric begins by aligning the candidate and reference translation strings. This alignment consists of mappings between unigrams, ensuring that each unigram in the candidate translation maps to either zero or one unigram in the reference [93]. The selected mappings aim to create an alignment with minimal crosses, meaning fewer intersections between the mappings. Once the alignment is done, the unigram precision and recall are calculated as follows;

$$P/R = \frac{m}{w_t}$$

Equation 2.9: METEOR (Source: [93])

Whereby,

m is the no. of unigrams in the generated and the reference sequence.

W_t is the no. of unigrams in the generated sequence.

In literature, METEOR has been shown to produce a good correlation with human judgment at the sentence or segment level; however, it has some limitations; for example, METEOR does not take into account the semantic similarity of words, and it does not capture clusters of words in the candidate that matches closely with clusters

of words in the reference[90]. Another limitation of METEOR is that this metric is based on the assumption that the reference summary is the gold standard against which the generated summary should be compared[93]. However, in some cases, the reference summary may not be the best, and the generated summary may be better. In such cases, METEOR may not accurately evaluate the quality of the generated summary.

2.5.4 Human Evaluation

Natural language processing research has employed human evaluation in addition to the automatic measures mentioned above to gauge the quality of text produced by machines. A typical human evaluation process involves assigning a quality rating to the created text by human judges. This evaluation type is an important complement to automated metrics, as it provides insights into some aspects of text quality that automated metrics may not capture[94]. However, researchers should be cautious when interpreting human evaluation results and consider the limitations and biases inherent in the human evaluation process.

There are several limitations and challenges associated with human evaluation, which have been identified from the literature. One of the main criticisms of human evaluation is its subjectivity and inconsistency. Different human judges may have varying opinions and interpretations of text quality, leading to discrepancies in the ratings assigned to the same text[94]. Human judges may be influenced by factors such as personal preferences, cultural background, and linguistic fluency, which can introduce bias into the evaluation results. Additionally, the lack of standardized evaluation protocols and guidelines can further intensify these biases and inconsistencies, undermining the reliability and validity of human evaluation in text summarization[94]. As a result, human evaluation was not feasible or practical for this study; hence, the researcher settled for automatic metrics for evaluation.

2.6 Summary

The second chapter of this thesis has covered a comprehensive literature review on various aspects of text summarization using transformer-based models. Four models, GPT, T5, PEGASUS, and BART, which leverage the transformer architecture, were reviewed in line with the first objective. The aim of this objective covering the

transformer-based models for text summarization was to analyze, appreciate, and critique what has been presented in the literature. For the second specific research objective, the chapter presented approaches and techniques for designing, developing, and improving models based on the transformer architecture. Evaluation metrics such as the ROUGE Metrics, Human evaluation, and the BLEU and METEOR score, used in text summarization evaluation, were also covered with respect to the third specific objective of the research, which was evaluating the improved model.

CHAPTER THREE
METHODOLOGY

3.1 Introduction

This chapter describes the research methodology that guided this research. The chapter begins with defining the research design adopted for this study and then follows with the research process that was followed to accomplish the three specific objectives. The data acquisition process is also described, along with detailed data pre-processing steps performed on the dataset. The experimental materials utilized in conducting this study are also described in this chapter. Finally, the ethical considerations that guided this study are included at the end of this chapter.

3.2 Research Design

The researcher adopted a True Experimental research design. This research design is characterized by the ability to manipulate an independent variable, control extraneous variables, and randomly assign participants to different groups [95]. True Experimental research designs are considered the gold standard for establishing cause-and-effect relationships. There exist various types of true experimental research designs, however, the researcher settled for the Pretest-Posttest Control Group Design for this study.

The research design was justified by the fact that the researcher usually has complete control over the research process, allowing for the manipulation of one or more independent variables while observing the effects on dependent variables [96]. This level of control is achieved by carefully controlling for extraneous variables that could potentially influence the outcome, thereby reducing errors and increasing the confidence levels and the reliability of the findings.

In this research design, the researcher randomly assigns participants to an experimental or control group [95], [96]. Therefore, the experimental group was subjected to the intervention or treatment being investigated, while the control group did not receive the treatment. Consequently, a pretest and posttest were carried out on both the improved model that had been developed (experimental group) and the baseline model (control group), with measurements taken before and after the intervention to assess the effectiveness of the improvements. In comparing the pretest

and the posttest performance of the two groups, this research design allowed for a rigorous assessment of the effectiveness of the improved model for text summarization of news articles. This research design, therefore, helped the researcher establish a causal relationship between the independent variables and the dependent variables in this study.

3.3 Research Process

A research process describes the steps the researcher undertook to achieve the research objectives and answer the research questions [95]. The general objective of this study was broken down into three specific objectives, which needed to be procedurally solved in chronological order to achieve the general objective.

The study's first specific objective was to answer a research question on determining a suitable transformer-based model and performance-improving features in text summarization of news articles. A comparative study was conducted as described in section 3.5 to answer this research question, whereby the four transformers for text summarization were subjected to the same set-up experiment. The output from the first specific objective was the best-performing model in text summarization alongside features that can improve its performance.

The second specific objective was to improve on the best-performing model from the first objective. The improvements were fueled by the need to address the overarching transformer limitations, which were pointed out in the literature and included in the problem statement. Section 3.6 describes how the researcher developed the improved model, which was the deliverable of this specific objective.

This study's third specific objective was to evaluate the performance of the improved model on benchmark text summarization datasets. Model evaluation was conducted as described in section 3.7, whereby two datasets and two evaluation metrics were considered. The illustration in Figure 3.1 shows the research process flow across the three specific research objectives.

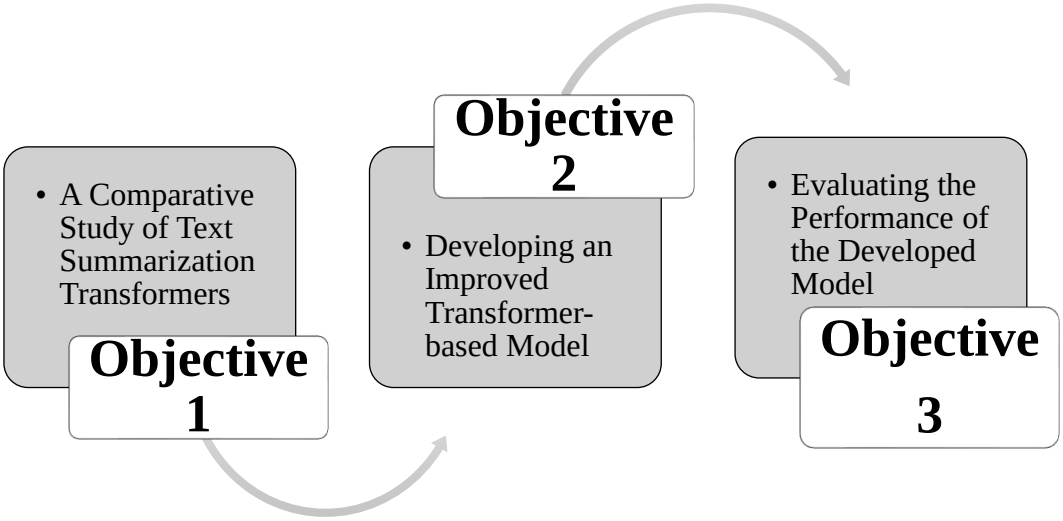


Figure 3.1: The Research Process

3.4 Data Acquisition

Secondary data was used in this study, and it was obtained from the publicly available text summarization benchmark datasets. Previous studies such as [60], [97], [98], which are aligned with this research, have also used secondary data; hence, there was a need to stick to secondary data for comparability and replicability. The choice of secondary data was also justified by its accessibility and comprehensiveness, providing a broader scope of news article coverage than could be obtained through primary data collection. Additionally, the time and resource constraints made secondary data a more practical and efficient choice for this research.

3.4.1 Dataset Description

Two datasets were selected from the publicly available text summarization benchmark datasets in the Hugging Face Hub, which is a comprehensive platform for natural language processing research, as it offers a vast repository of pre-trained models and datasets that are readily accessible for various NLP tasks. The CNN/DailyMail dataset was selected as the training and validation set, while the XSUM dataset was used to evaluate how the proposed model would generalize on data it had not been trained on.

The CNN/DailyMail dataset is a widely used benchmark dataset for text summarization tasks in natural language processing (NLP). The researcher considered

the 3.0.0 non-anonymized version of the CNN/DailyMail dataset, which is justified by the fact that it is specifically curated for abstractive summarization. This dataset is made up of a collection of news articles paired with human-generated summaries from two major news outlets: CNN and DailyMail. The dataset spans diverse topics such as politics, sports, entertainment, and technology, reflecting the breadth of news coverage typically encountered in journalistic contexts [3]. Each article-summary pair in the dataset is carefully curated to ensure high quality and relevance, making it suitable for adoption in text summarization research.

The XSUM dataset is a prominent and widely utilized benchmark dataset for evaluating abstractive summarization models because of its large scale, topical diversity, and human-written summaries. Created by harvesting online articles from the British Broadcasting Corporation (BBC), this dataset comprises over 227,000 article-summary pairs in news articles[8]. Notably, the summaries in XSUM are professional multi-sentence summaries generated by skilled writers, ensuring high-quality and accurate target texts for the summarization task. The articles cover various topics, including news, politics, weather, business, and more, providing a comprehensive evaluation framework for summarization models. The two datasets contain training, validation, and testing instances of news article pairs, as shown in the illustration of the dataset distribution in Table 3.1.

Table 3.1: Dataset Distribution in the CNN/DailyMail and the XSUM Datasets.

Dataset Split	Distribution of Articles	
	CNN/DailyMail	XSUM
Train Set	287,226	204,045
Validation Set	13,368	11,354
Test Set	11,490	11,332

3.4.2 Data Pre-processing

Data pre-processing was conducted on both datasets, CNN/DailyMail and XSUM, to transform the raw data into a suitable format for the models to process the sequences effectively, given that this is a necessary procedure in deep learning and natural language processing tasks, as observed in [99]. This study's first step in data pre-processing involved tokenizing the input text sequences. Tokenization was conducted

to segment the text into smaller units referred to as tokens, which can be words, sub-words, or characters[100]. The researcher used the T5Auto-tokenizer from the Hugging Face Transformers library, specifically designed for the T5 model. The tokenizer converted the input text into a sequence of token ids and attention masks. The token ids represented the numerical encoding of the tokens, while the attention mask indicated the tokens that should be attended to by the model's encoder.

The second step in the pre-processing stage was performing sequence truncation and padding to accommodate the transformer's context length for both datasets. For input sequences longer than the maximum sequence of 512, the truncation step truncated them to fit the specified length. This process involved retaining only the initial segment of the sequence up to the maximum length and then pushing the remaining tokens to the next sequence. On the other hand, for input sequences shorter than the maximum length, the researcher padded them with special tokens of 0s to ensure a consistent sequence length. The padding step involved appending the appropriate number of padding tokens to the end of the sequence, allowing the encoder to handle all sequences simultaneously in a batch.

Following the pre-processing steps above, the third and final step was to split the dataset into three subsets: training, validation, and testing, on a ratio of 70%, 20%, and 10%, respectively, which is a standard practice in machine learning and is essential for model training, validation, and testing, as per [34]. The training set was used to train the model, allowing it to learn the patterns and relationships necessary for text summarization. The validation set served as a checkpoint during training to enable the monitoring of the model's performance, while the test set evaluated the model's generalization performance on unseen data.

3.5 A Comparative Study of Text Summarization Transformers

For the first specific objective in this study, a comparative study was conducted to identify a suitable transformer-based model for text summarization of news articles, along with features that can improve performance. The suitability was defined by the model's performance in the same set-up experiment on a text summarization task. The researcher was guided by the existing literature conducted on transformers for text summarization of news articles and selected four commonly used transformer-based

models for text summarization of news articles [4], [23], [98]. The four selected models were: Generative Pretrained Transformer (GPT), Text-to-Text Transfer Transformer (T5), Bi-directional Auto Regressive Transformer (BART), and Pre-training with Extracted Gap sentence for Abstractive Summarization. (PEGASUS). The selection of these four models—GPT, T5, BART, and PEGASUS—was justified by their widespread recognition, their varied approaches to handling summarization, and their demonstrated effectiveness in the literature on text summarization tasks.

The comparative study was conducted under the same experimental setup, and a detailed description of the experiment is provided below. The experiment was simulated on a Python notebook in the Google Collaboratory environment that provided additional hardware accelerating GPUs and afforded access to CUDA version 12.2, the Transformers library version 4.35.2, and PyTorch version 2.2.1. Only the CNN/DailyMail dataset obtained from the Hugging Face Hub, whose composition in the train, validation, and test set distribution is as follows: the train split has 287,226 instances, the validation split contains 13,368 instances, and the test split has 11,490 instances, was used for the comparative study. The dataset was pre-processed according to the procedure described in section 3.4.2 to format the dataset appropriately for transformer models.

The four models considered for this comparative study were then subjected to similar hyperparameters to ensure the fairness and comparability of the evaluation results. Therefore, a fixed number of 8 attention heads was used, and 2048 was the feedforward neural network dimension in each transformer block. A low learning rate of 1e-5 and a weight decay of 10^{-2} were adopted since this was a fine-tuning task for the pre-trained models. The AdamW optimizer maximized model performance while preventing the learned representations from being catastrophically forgotten. Each transformer model was trained for 10 epochs with a batch size of 12 to reduce the memory used during each training and validation iteration. The number of steps needed to accumulate gradients before performing an update pass was set at 16 to speed up the training process. The evaluation and model save strategy, which controls the number of saved model checkpoints, was determined by the predefined number of epochs.

In the comparative study, the pre-trained models, with their pre-trained weights, were sourced directly from the Hugging Face hub, and the training process was initiated by invoking the Trainer class, which incorporated the specified training arguments to establish the training loop. During this phase, the parameters of the pre-trained models were adjusted with a lower learning rate to prevent disrupting the learned representations acquired during the pretraining stage. This careful tuning ensured the models retained their pretraining knowledge while adapting to the new tasks. The training process entailed calculating the loss, backpropagating to compute the gradient, and then making parameter updates guided by the predefined gradient accumulation steps. This procedure was justified by the fact that, as this study [97] showed, the target task was comparable to the model's pretraining task. The training process was optimized using the AdamW optimizer in the training phase, with the cross-entropy loss serving as the loss function.

The researcher used ROUGE metrics to evaluate the model's performance by quantifying the alignment between the generated text and the reference summary. ROUGE-1 measured the unigram overlaps, ROUGE-2 measured bigram overlap, and ROUGE-L determined the longest common subsequence, while ROUGE-Lsum focused on the longest common subsequence of words between the generated and reference summaries, according to [92]. The preference for ROUGE metrics was justified by their widespread acceptance and effectiveness in evaluating text summarization quality, as they provide a standardized and comprehensive measure of the overlap between generated summaries and reference texts. The evaluation results were compared to determine the model with the highest ROUGE scores, which the researcher would then proceed to improve on in the second objective of this study. The detailed results obtained from the objective one's comparative study are presented and discussed in Chapter 4 of this thesis.

3.6 Developing the Improved Model

The second specific objective of this study was to improve on the model identified from the first specific objective. Guided by the evaluation results from the comparative study, the researcher settled for the T5 Model, given that it recorded the highest evaluation scores in comparison with other models. However, with the complaint statements raised in the problem statement of this study [30], [31], [32], [33], [34],

the T5 transformer still has limitations that need to be addressed. Therefore, it was imperative to improve on this model, aiming at addressing the reported limitations and consequentially improving its performance in text summarization of news articles. The improvements that the researcher integrated are described in detail in the following sections.

3.6.1 Model Quantization

The first reported limitation of the T5 model is that it involves a lot of attention operations and matrix multiplications, which translates into high computational demands during training and inferencing. Addressing this challenge prompted the researcher to perform model quantization, as illustrated in the following model quantization algorithm.

Model Quantization Algorithm

Require: Model with ***L*** layers, desired precision levels
Ensure: Quantized model

```
// Define model layers and desired precision levels
model_layers = get_model_structure ()
high_precision = 32 // For activations in fully-connected layers
low_precision = 16 // For weights in linear layers

// Perform selective quantization on the model
for layer_index from 1 to L, do:
   layer = model_layers[layer_index]
   if layer. type == "Fully-Connected":
      layer. activations = quantize_to_precision (layer. activations,
high_precision)
   else if layer. type == "Linear":
      layer. weights = quantize_to_precision (layer. weights, low_precision)
   end if
end for

// If performance loss is significant, apply PQ fine-tuning
performance_loss = evaluate_model_loss(model_layers)
if performance_loss >= 2%:
    model_layers = fine_tune_quantized_model(model_layers)
end if
return quantized_model
```

The model quantization process was geared towards reducing the training and inference cost of the T5 model. This was achieved by selectively quantizing the model's layers to lower precision levels, hence reducing the computational cost associated with full-precision operations. The quantization process was as follows: the first step was defining the model's layer structure and specifying the desired precision levels for different layer types. Specifically, 32-bit full precision was applied for activation matrices in the fully connected layers, which are highly sensitive to quantization, while a lower 16-bit precision format was applied to the weight matrices in linear layers, which are tolerant to quantization.

The quantization approach described above forms a part of the contributions made from this research, whereby the researcher has introduced an adaptive technique in its operation to strike a balance between preserving model performance and the representative capacity of its weight and activations. The basic operation of the algorithm was to quantize the activations and weights based on their respective layer types and their response to precision reduction. For fully-connected layers, the activations were quantized to the high-precision level of 32 bits, ensuring minimal loss of information during the computation of non-linear transformations. The weights were then quantized to the lower 16-bit precision levels in the linear layers to capitalize on the inherent flexibility in weight representations in these layers in neural networks.

Quantization is often associated with a slight model performance drop [89]. Therefore, the researcher incorporated a post-quantization fine-tuning (PQFT) step to mitigate potential performance drops from the quantization process. After quantizing the model, the algorithm evaluated the performance loss by comparing the quantized model's outputs to the baseline model's outputs. If the performance loss exceeded a predefined threshold of 2%, the algorithm self-initiated a fine-tuning step, where the quantized model's weights were re-adjusted on a fixed low learning rate. The fine-tuning step readjusted any performance drop introduced by the quantization process, ensuring that the quantized model's performance remained within an acceptable range compared to the baseline model.

3.6.2 Hyperparameter Optimization

The second limitation of the T5 model, as highlighted in the problem statement, was hyperparameter sensitivity, which is evidenced in overfitting, unstable training, and convergence while using the default training hyperparameters [84]. Therefore, the researcher utilized hyperparameter optimization to optimize the performance of the T5 model in text summarization of news articles.

Hyperparameter optimization was conducted using two approaches. The first approach, manual optimization, entailed careful testing of numerous combinations of hyperparameter settings and analyzing the model's performance for each combination to determine the tuple that produced the best results. Static hyperparameters such as the batch size, number of epochs, and gradient accumulation steps were tuned manually, guided by the available computational resources regarding GPU memory. Manual tuning was also done on some dynamic hyperparameters, such as the learning rate, weight decay, and dropout rate during the training process, where the model's performance on the validation set was closely monitored for signs of overfitting and underfitting. The researcher also manually incrementally increased the weight decay and dropout rate values during training to introduce more regularization to curb overfitting.

The second hyperparameter optimization approach used was automated tuning. The researcher used the RayTune and Optuna libraries to automatically find the best tuple of hyperparameters during the optimization process. These libraries provide automatic hyperparameter tuning techniques using various optimization techniques in neural networks and deep learning. The researcher opted for gradient descent, one of the gradient-based hyperparameter optimization techniques in neural networks [83], [84]. Gradient-based hyperparameter optimization involves tuning hyperparameters by leveraging gradient descent or a related optimization algorithm. The main advantage of gradient-based optimization is that it can handle continuous and high-dimensional hyperparameter spaces more effectively, which is often the case in neural network architectures such as the transformer [83]. Gradient descent can also adapt to the hyperparameter optimization landscape's complexity, allowing for more efficient exploration of non-convex optimization surfaces. In gradient descent optimization, the aim was to iteratively update the parameters in the direction that minimizes the loss

function, ultimately reaching a local minimum and convergence point where the loss function is minimized, and was formulated as follows;

$$\theta_{j+1} = \theta_j - \alpha \nabla J(\theta_j)$$

Equation 3.1: Gradient Descent (Source: [83])

In this equation:

θ_j Represents the current estimate of the parameters.

α (alpha) is the learning rate to control the parameter updates.

$\nabla J(\theta_j)$ denotes the gradient of the loss function J wrt the parameters θ_j.

θ_{j+1} represents the updated estimate of the parameters after taking a step in the direction of the negative gradient.

In this approach, the researcher began by initializing the hyperparameter search space $\Theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ with initial values $\Theta^{(0)}$. The initialization for each hyperparameter was random within a predefined range to make room for the exploration of a wide range of the hyperparameter search space. The goal was to minimize the model loss $\mathcal{L}(\mathrm{w}, \Theta)$, where W represents the model parameters and Θ denotes the hyperparameters. At each iteration t, the model loss $\mathcal{L}^{(t)}$ was calculated and optimized by adjusting the hyperparameters iteratively based on the gradient of the objective function with respect to each hyperparameter, as observed in this study [82]. The update rule for each hyperparameter θ_i was defined as:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \cdot \nabla_{\theta_i} \mathcal{L}$$

Equation 3.2: Hyperparameter Update Rule

Where:

η is a fixed scalar controlling the step size of the update.

θ_i represents the hyperparameters

t represents each iteration

$\mathcal{L}$ is the model loss

This iterative process continued until convergence, whereby the changes in the loss function $\mathcal{L}(w, \Theta)$ and the updates to w and Θ were the lowest, ensuring that the model converged to a global minimum of the loss function. Hyperparameters that formed the search space for gradient-based optimization included the learning rate, weight decay, optimizer, dropout rate, activation function, and warm-up steps. The ranges were selected based on prior domain knowledge, literature review, and the default values to cover a reasonable scope without being overly broad or narrow. This approach efficiently explored the hyperparameter space while avoiding exploring unpromising regions, leading to faster convergence to the optimal solution. The flowchart in Figure 3.2 illustrates the process of hyperparameter optimization.

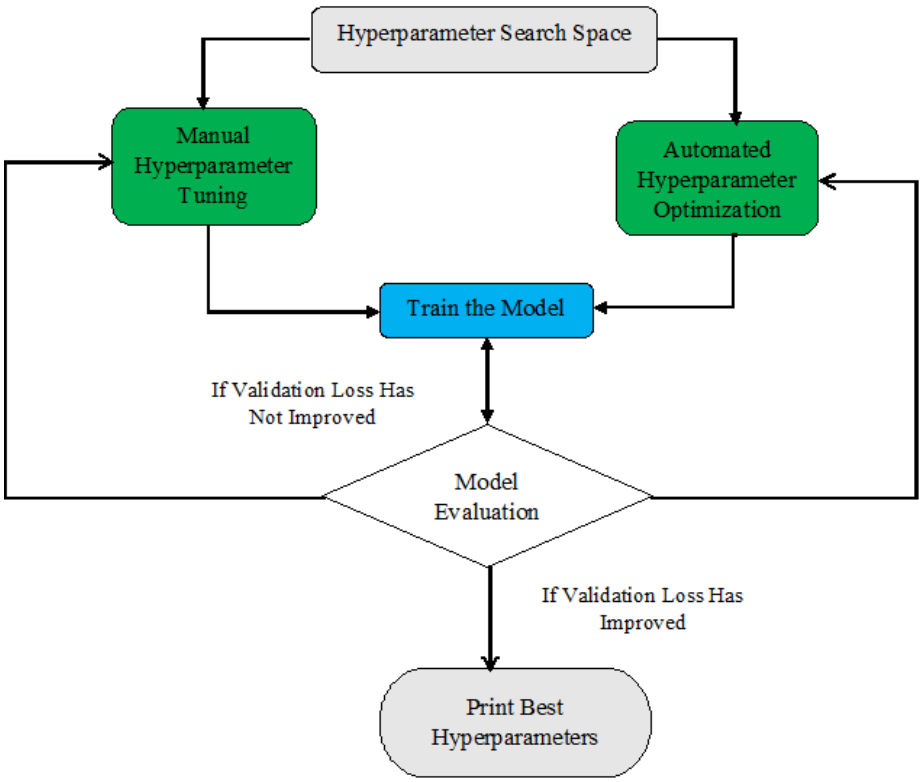


Figure 3.2: Hyperparameter Optimization

The combination of manual tuning for certain hyperparameters and automated tuning via gradient descent for other hyperparameters helped the model explore the hyperparameter space exhaustively. Therefore, this hybrid-like optimization approach resulted in a more effective hyperparameter optimization process tailored to the specific task and the model architecture. A suite of the optimal hyperparameter settings that yielded optimal model performance is presented in Chapter 4 of the thesis.

3.6.3 Modifying the Model's Output Layer

The default output layer of the T5 transformer is pre-trained for several tasks; however, this output head is not adapted specifically for a task-specific requirement such as text summarization. This raised a concern about the model's task-specific optimization and generalizability on a downstream task. Therefore, the researcher modified the output head of the model to adapt it for a text summarization task as described below.

The modification involved the integration of two parameters for text generation, namely, sequence generation and token selection. These parameters are crucial for generating summary tokens from the encoded input representation, which the default output head struggled to handle explicitly. The researcher integrated the Temperature parameter, which is a sequence generation parameter that modulates the conservativeness or randomness of the generated output sequences by controlling the softmax distribution over the model's output vocabulary[101]. The temperature parameter T was applied to the logits $\mathrm{z} = (z_1, z_2, \dots, z_n)$, which were the raw scores output from the model's final layer before the application the activation function. The adjusted logits z' were computed as follows;

$$z_i' = \frac{z_i}{T}$$

Equation 3.3: The Temperature Parameter

Whereby T was the temperature value between 0 and 1, z_i represented the logits and z_i' the adjusted logits. The probability distribution flattened for a temperature value closer to 1, resulting in more random outputs, while the distribution became sharper for values closer to 0, leading to a more uniform distribution due to the decrease in entropy.

The Softmax activation function was then applied to these adjusted logits to obtain the resultant probability distribution. This activation function is usually applied to the logits (raw scores) generated by the final fully connected layer, representing the model's predictions for the next token in the summary sequence[40]. The Softmax activation function with the temperature parameter *T* was defined as:

$$\sigma_T(\mathbf{z})_i = \frac{e^{z_i/T}}{\sum_{j=1}^{K} e^{z_j/T}}$$

Equation 3.4: Softmax Activation Function with Temperature

Here,

$\sigma_T(\mathbf{z})_i$ is the probability for the i-th element of the input vector **z**,
T is the temperature parameter,
K is the number of elements in the input vector **z**.
e is the exponential function.
e^{z_i} Is the exponentiation of the i-th element of the input vector.
$\sum_{j=1}^{K} e^{z_j}$ Is the sum of the exponentials of all elements in the input vector.

The top-k sampling parameter was also considered, which is particularly important for ensuring that the generated text is coherent and contextually relevant. This parameter works by limiting the selection process to a subset of the most probable tokens, thereby enhancing the quality of the generated text. Instead of considering the entire vocabulary, this parameter selects the top k tokens with the highest probabilities, effectively narrowing down the candidate tokens, which increases the likelihood of producing high-quality text. In this parameter, the first step was computing the logits ($\mathrm{z} = z_1, z_2, \ldots, z_n$) from the model's output layer and then applying the softmax activation function to convert these logits into probabilities ($\mathrm{p} = p_1, p_2, \ldots, p_n$), just as in the temperature parameter. The probabilities were then sorted in descending order to identify the top k tokens, denoted as $\{p_{(1)}, p_{(2)}, \ldots, p_{(k)}\}$ where $p_{(1)} \geq p_{(2)} \geq \cdots \geq p_{(k)}$. The selection was then restricted to these top k tokens, and their probabilities were renormalized to ensure that they formed a valid probability distribution: $p'(z_i \mid z_i \in \{z_{(1)}, z_{(2)}, \ldots, z_{(k)}\})$. The renormalized distribution was then used to sample the next token, either deterministically by selecting the token with the highest probability or stochastically by sampling according to the renormalized probabilities.

The Softmax activation function, therefore, played two vital roles in the output layer. The first role was to normalize the attention weights and logits to ensure proper distribution of attention over the input sequences. The resulting values, the output probabilities, represented the model's confidence in predicting each token as the next token in the summary sequence. This normalization ensures that the sum of the probabilities across all tokens in the vocabulary equals 1, satisfying the properties of a valid probability distribution[48]. The second role was to help in token selection during the summary generation process. The modified output head of the model could convert the logits into interpretable probabilities by applying the Softmax activation function to the logits, which improved the selection of the next token in the output sequence of the generated summary.

3.6.4 Model Training and Validation

The improved model was trained as described in this section. The training and validation set of the CNN/DailyMail dataset was used during model training and validation. The training process was as follows: firstly, the input text sequences and their corresponding target summaries were tokenized using a tokenizer specifically designed for T5. These tokenized inputs were then fed into the model and passed through the model’s architecture, which is comprised of self-attention mechanisms and feed-forward neural networks.

The model was trained on the CNN/DailyMail dataset in two phases. In the first phase, only the fully connected layers (FCLs) were updated as the base model remained frozen. Freezing the base model and updating the FCLs first ensured that the updated model parameters were task-specific to adapt the model to the downstream task easily. In the second phase, a comprehensive training approach was used, whereby all the layers in the model were updated, rendering the model to function as a randomly initialized encoder. Training the entire model helped it extensively learn the dataset and find a balance between task-specific optimization and model generalization.

The model iteratively adjusted its internal parameters through backpropagation to minimize a predefined loss function, the cross-entropy loss, between the predicted and ground truth summaries. Mathematically, cross-entropy loss is defined as;

$$\text{Cross-entropy Loss } = -\sum_{t=1}^{T} \log P(y_t \mid x, y_{<t})$$

Equation 3.5: Cross-entropy Loss Function (Source: [102])

Here,

T is the length of the target summary.

y_t Is the token at time step t in the target summary.

x is the input sequence.

$y_{<t}$ are the tokens preceding time step t in the target summary.

$P(y_t \mid x, y_{<t})$ is the conditional probability assigned by the model to the token y_t given the input sequence x and the preceding tokens $y_{<t}$.

Model optimization during training was guided by the AdamW optimizer, which efficiently updated the model parameters based on gradients computed from the loss function. The training process was iterated over multiple epochs until the model converged to a satisfactory level of performance, as determined by predefined evaluation metrics on a validation dataset. The following model training and validation algorithm describes the process of training and validating the model on the CNN/DailyMail dataset.

Model Training and Validation Algorithm

Require:

Pre-processed dataset of text-summary pairs ***(D_train, D_eval)***
Pre-trained ***M*** parameters ***θ_pretrained***
Hyperparameters: Passed to ***Seq2Seq TrainingArgs***

Load pre-trained ***M*** with parameters *θ_pretrained*
Fine-tune ***M*** on text summarization dataset *D_train*

for epoch = *1* to *e* **do**
Shuffle D_train at the start of each epoch
for each batch (X, Y) in D_train, **do**
Generate predictions Y' by inputting **X** into ***M***
Calculate loss using Y and Y' (cross-entropy)
Update model parameters *θ* by backpropagation using AdamW
end for
Calculate and print average loss for epoch
Save model checkpoint if validation loss improves
end for

return final model parameters *θ_finetuned*

Additionally, model validation was carried out to evaluate the trained model's performance using an independent validation dataset. This is necessary to prevent the model from overfitting the training set and to ensure that it generalizes well to new data. The validation dataset, different from the training dataset used in model training, contained news articles and their conforming summaries. During model validation, checkpoints were implemented to save the model's weights at regular intervals during the training, allowing for the restoration of the best-performing model configuration in case of early stopping or unforeseen interruptions. The researcher also applied early stopping to prevent overfitting and ensure optimal model convergence. Early stopping mechanisms monitored validation loss, stopping the training when the loss ceased to decrease or displayed signs of divergence, thereby preventing the model from overfitting to the training data and enhancing better generalization.

The training and validation loss was also monitored as indicators of the learning dynamics of the model during its training phase. The training loss, calculated as the average loss over each training batch, provided insights into the model's capacity to lessen errors and appropriately fit the training data. Concurrently, validation loss, computed on the validation dataset, offered a measure of the model's performance on unseen data. Tracking the validation loss helped the researcher detect overfitting and make informed decisions in the hyperparameter optimization process.

3.7 Evaluating the Performance of the Improved Model

The third specific objective of this study was to evaluate the performance of the developed model on benchmark text summarization datasets. Two datasets were used to perform model evaluation while enhancing the validity and reliability of the results. A test split from the CNN/DailyMail dataset, constituting 10% of the total dataset, was used as one of the testing datasets for the model. Additionally, the XSUM dataset was also used as a separate dataset of unseen data to test how generalizable the improved model was.

The model evaluation step provided an objective and a quantitative measure of the model's effectiveness in generating summaries on unseen data. Therefore, two evaluation metrics, ROUGE and BLEU scores, were considered to provide a more comprehensive evaluation of the generated summaries, with ROUGE capturing recall-oriented aspects and BLEU focusing on precision, thereby providing a balanced assessment of summary quality [91], [92]. The performance of the improved model was then compared against the baseline model without any modifications to validate the effectiveness of the improvements. The following model evaluation algorithm illustrates the process of model evaluation.

Model Evaluation Algorithm

Require:
Pre-processed evaluation dataset of text-summary pairs ***(D_eval)***
Trained Model's parameters ***θ_finetuned***
Evaluation metrics: ***ROUGE Metrics, BLEU Score***

load fine-tuned ***M*** with parameters θ_finetuned
initialize evaluation metrics accumulators

for each sample (X_eval, Y_eval) in D_eval **do**
generate summary Y'_eval using the model with input X_eval
compute (**ROUGE, BLEU**) comparing Y_eval and Y'_eval.
accumulate metric scores
end for

calculate average evaluation metrics for the entire D_eval.
return evaluation metrics (ROUGE Metrics, BLEU Score)

End Algorithm

In this algorithm, each text-summary pair (X_eval, Y_eval) from the pre-processed evaluation dataset (D_eval) was passed into the evaluation loop. For each sample, the algorithm generated a summary (Y'_eval) by feeding the corresponding input text (X_eval) into the model. This essentially simulated real-world usage where the model would encounter unseen text data and produce a summary. To gauge the quality of the generated summaries (Y'_eval), the pre-defined evaluation metrics, ROUGE and BLEU scores, were automatically computed. These metrics compared the generated summaries against the reference summaries (Y_eval) present within the dataset. Through this evaluation process, the researcher effectively compared how closely the generated summaries aligned with the reference summaries objectively and quantitively.

3.8 Experimental Materials

The experimental materials utilized in this research included a comprehensive array of software tools and frameworks essential for developing and evaluating the proposed model. Table 3.2 outlines the experimental materials used by the researcher in this study.

Table 3.2: Experimental Materials

Experimental Material	Description
Desktop	HP Intel® Core i7-6500U CPU @ 2.60 GHz
CUDA Device	NVIDIA® RTX A6000 GPU
Programming Language	Python Version 3.9.19
Deep Learning Framework	PyTorch Version 2.2.1
NLP Libraries	NLTK, spaCy
Hugging Face Libraries	Transformers Version 4.35.2
Visualization Libraries	Seaborn, Matplotlib
Simulation Environment	Google Colab

The primary computing platform was an HP desktop with Intel® CoreTM i7-6500U CPU @ 2.60 GHz. It was supplemented by a CUDA (Compute Unified Device Architecture) device, specifically an NVIDIA® RTX A6000 GPU card. The programming language used in this study was Python version 3.9.19, whereby PyTorch version 2.2.1 was utilized as the deep learning framework to provide the requisite environment for neural network implementations. The Natural Language Processing (NLP) libraries for text processing used in this study were the NLTK (Natural Language Toolkit) and spaCy libraries. The Hugging Face Transformers library, version 4.35.2, was used to provide access to checkpoints of pre-trained transformers, their corresponding tokenizers, and the datasets. The Seaborn and Matplotlib libraries enabled the visualization of the results via graphs and plots. The experiment was simulated in Google Colab, which provided a convenient and scalable platform for model development, training, and evaluation.

3.9 Ethical Considerations

This study upheld ethical considerations throughout the research process, underscoring the researcher's commitment to ethical research practices. Prior to commencing the research, the researcher obtained necessary permits and approvals to ensure compliance with regulatory requirements. A permit was acquired from NACOSTI (National Commission for Science, Technology, and Innovation) shortly after receiving a research approval letter from the Board of Postgraduate Studies (BPS)

at the Murang'a University of Technology. Copies of the research approval letter from the BPS and the NACOSTI research permit are attached in Appendices 1 and 2, respectively.

3.10 Summary

This chapter describes the methodology that was followed by the researcher in this study. A true experiment was identified as the research design for this study. The chapter describes the entire research process in detail, starting with choosing the best-performing transformer, data acquisition and its pre-processing, model training, validation, and evaluation. This chapter also described the experimental materials used in this study. The chapter concluded by mentioning the ethical considerations that were considered while conducting this study.

CHAPTER FOUR
RESULTS AND DISCUSSION

4.1 Introduction

The fourth chapter of this thesis presents the results of the study that the researcher conducted. The chapter chronologically describes the results, starting from the comparative study conducted in this study's first objective. This is followed by the evaluation results recorded after validating and testing the improved model developed in the study's second objective. Afterwards, a comparison is also presented to compare the improved T5 model's performance against the baseline model to ascertain the effectiveness of the improvements. The results are then followed by a discussion section that synthesizes their implications of the performance of the model in text summarization of news articles.

4.2 The Comparative Study of Transformers for Text Summarization

In the first objective of this study, a comparative study of four state-of-the-art transformers for text summarization was conducted. This objective aimed to determine the best-performing transformer when these models are subjected to the same dataset, same hyperparameters, and same evaluation metrics. Table 4.1 illustrates the number of parameters and layers across variants of the covered transformer-based models, namely BART, T5, PEGASUS, and GPT, and their specific types of architecture.

Table 4.1: Model Variations Across Parameters, Layers, and Architecture

Transformer	Parameters		Layers		Type of Architecture
	Base	Large	Base	Large	
GPT	124M	762M	12	36	decoder-only
T5	220M	770M	12	24	encoder-decoder
PEGASUS	175M	568M	16	32	encoder-decoder
BART	140M	406M	12	24	encoder-decoder

Table 4.1 outlines the architectural specifications and parameter counts of four state-of-the-art transformer models utilized in natural language processing tasks, specifically focusing on text summarization. The comparison is split between base and

large configurations for each model, which differ in the number of parameters and layers. Parameters refer to the model's complexity in terms of the weights and biases and the capacity to learn from data, with a higher number indicating a more complex model that can potentially capture complex patterns in the data [56]. On the other hand, the number of layers shows the depth of the model's neural network, with more layers typically allowing for more levels of abstraction in data processing.

GPT is distinguished by its decoder-only architecture, primarily because of its design goal and usage focus, which operates in an autoregressive manner, making it unique among the listed models, which are all encoder-decoder architecture-based [65]. The base and large configuration for GPT show a substantial increase in parameters from 124 million to 762 million and layers from 12 to 36, respectively. T5, PEGASUS, and BART all employ encoder-decoder architectures, which are particularly effective for tasks requiring an understanding of input data and generating corresponding outputs, such as text summarization. PEGASUS and BART, while varying in their parameter sizes and layer counts, emphasize the diversity in design and application focus within this domain, with PEGASUS aimed at summarization through pre-training on gap sentences and BART being a versatile model capable of both sequence generation and language comprehension tasks[50], [59]. The T5 model, with 220M and 770M parameters for the base and large variant, utilizes 12 and 24 layers, respectively, and employs an encoder-decoder architecture.

4.2.1 Model's Performance on Text Summarization

The study compared the performance of four transformer models (*gpt2-xl*, *T5-large*, *Bart-large-cnn*, *and Pegasus-cnn_dm*) on text summarization using the CNN/DailyMail dataset from the Hugging Face hub. Model performance was evaluated quantitatively using ROUGE Metrics: ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-Lsum. Table 4.2 presents the reported ROUGE metric scores for each transformer model evaluated in the study.

Table 4.2: ROUGE Metrics for the Comparative Study

Transformer	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum
GPT	24.83	16.92	22.14	21.07
BART	27.61	18.37	28.52	25.84
T5	**35.12**	**22.75**	**32.82**	**28.59**
PEGASUS	33.69	21.58	28.43	23.76

The comparative study, as reflected in table 4.2, noticeably positioned the T5 model as the highest performing across all ROUGE metrics that are established benchmarks for evaluating the quality of text summarization. T5's leading ROUGE-1 score of 35.12 indicated its superior ability to capture the source text's most important unigrams (single words). Its ROUGE-2 score of 22.75 demonstrated that it also excelled at identifying bigrams (word pairs), which are crucial for understanding the text's coherence and structure. Moreover, T5's ROUGE-L and ROUGE-Lsum scores, 32.82 and 28.59, respectively, highlight its strength in capturing longer sequences and maintaining the structure of the original text, which is essential for producing summaries that are not only accurate but also fluent and readable.

The features that ensured that T5 outperformed the other models are the encoder-decoder architecture and pre-training objective effectively leverage contextual understanding and text interpretation better than the other models in generating coherent and concise summaries[55]. The T5 model outperforms the other models because it formulates all NLP problems as text-to-text tasks. This allows for a unified framework that can be effectively leveraged through transfer learning after pre-training on a labelled dataset using supervised or unsupervised approaches [58].

4.2.2 Training and Validation Loss of the Models

In this section, the researcher compared the training and validation loss of the four state-of-the-art transformer models, GPT, BART, T5, and PEGASUS, fine-tuned for 10 epochs on the CNN/Daily Mail dataset. Uniform training hyperparameters were selected to enhance the comparability, with a learning rate set at 1e-5 and a weight

decay of 0.01. The training process was optimized for all models using the AdamW optimizer, renowned for its ability to handle sparse gradients and adaptive learning rates [81].

This setup allowed the researcher to closely monitor and analyze the convergence patterns of each model based on their training and validation loss curves. The loss curves provided insights into the model's learning efficiency and stability during training over the number of specified epochs. The training and validation loss curves for each model are presented along with an explanation of the model convergence based on the training and validation loss.

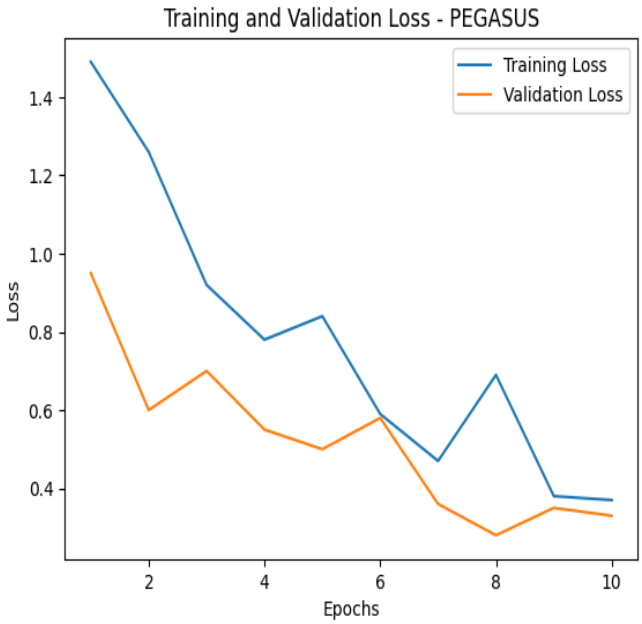


Figure 4.1: Training and Validation Loss – PEGASUS

The graph in Figure 4.1 illustrates the training and validation loss for the PEGASUS model across 10 epochs. Initially, the training loss started at a high value and demonstrated a steep decline, indicating that the model quickly learned from the training data. As the number of epochs progressed, the training loss continued to trend downwards, albeit at a slower rate, suggesting that the model had started to converge and was extracting the relevant patterns from the training data.

While starting slightly lower than the training loss, the validation loss also decreased rapidly, which indicates the model's ability to generalize to unseen data. Notably,

there are points where the validation loss plateaued and increased slightly, which could point to the model beginning to overfit due to the variability in the validation data, which is undesired during model training.

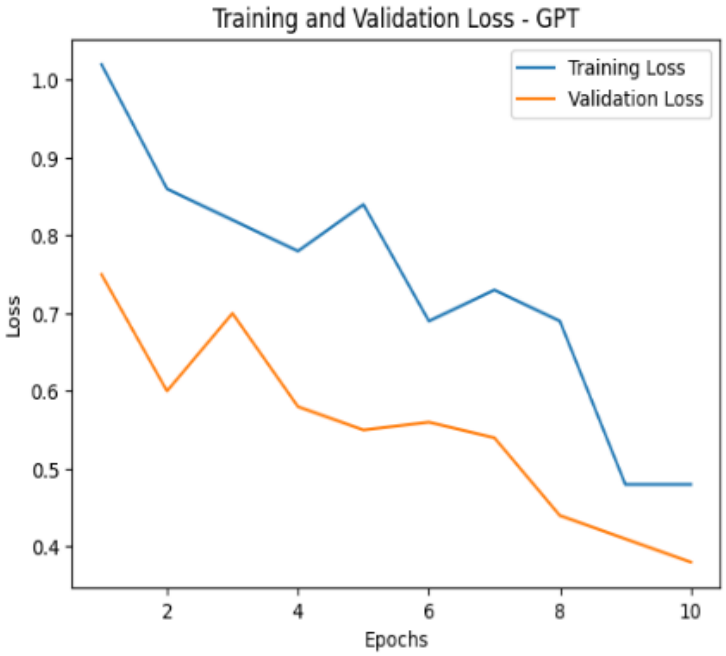


Figure 4.2: Training and Validation Loss - GPT

The graph in Figure 4.2 depicts the training and validation loss for the GPT model over 10 epochs, demonstrating a less-than-ideal convergence pattern. While the training loss began at a high value and dropped sharply, indicative of the model rapidly learning from the training data, the subsequent fluctuations disclose a lack of stability in the learning process.

Although decreasing across the epochs, the validation loss does not mirror the expected smooth, gradual decline. Instead, it exhibits variability with certain points where the loss increases, a telltale sign of the model's struggle to generalize the validation data consistently. These irregularities in the loss curves raise concerns about the model's ability to improve reliably with additional data or epochs and point towards further tuning the model's parameters or the training process to achieve optimal performance.

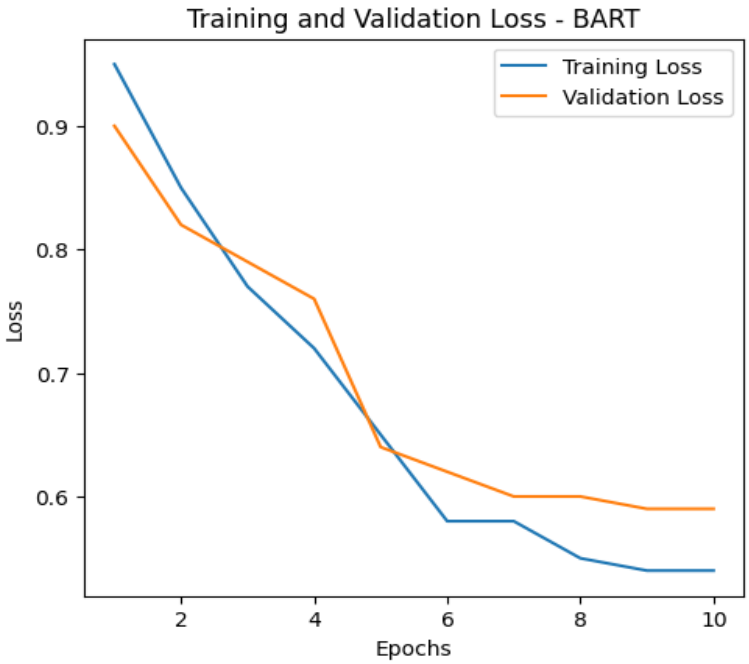


Figure 4.3: Training and Validation Loss - BART

The loss curves for the BART model exhibit a characteristic decline across the 10 epochs, as depicted in Figure 4.3. The training loss started notably high but decreased sharply, indicating a rapid initial learning phase. It is commendable that the validation loss not only mirrored this sharp decrease but also sustained a consistent reduction, similarly close to that of the training loss, which emphasized the model's capacity in generalization.

However, the convergence of the training and validation loss suggested that while the model is learning and adapting, there might be a ceiling to the performance gain achievable with the current setup. The almost parallel nature of the curves toward the final epochs pointed to a potential underfitting scenario where the model may not have fully captured the patterns of the training dataset.

Despite this, the fact that the validation loss remained consistently below the training loss is encouraging. It implied that additional fine-tuning, perhaps with a more aggressive learning rate or a longer training duration, could benefit the model's predictive performance.

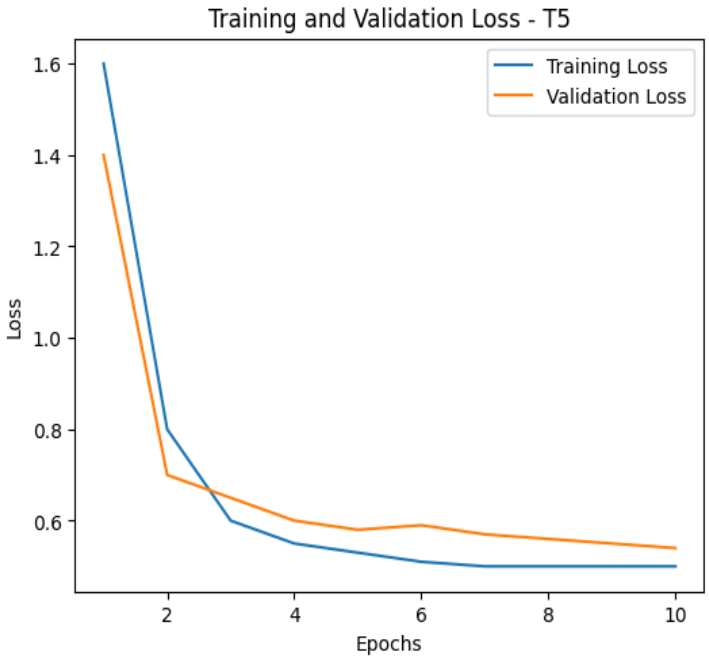


Figure 4.4: Training and Validation Loss – T5

The training and validation loss presented in Figure 4.4 for the T5 model in the provided figure portrayed a near-desirable learning and generalization trend across 10 epochs. The training loss began at a high value and swiftly descended, suggesting that the model could rapidly adapt to the underlying patterns in the training dataset.

As the epochs increased, both the training and validation loss continued to drop, with the validation loss closely following the training loss, a strong indicator of the model's ability to generalize well without overfitting. The curves proceeded to plateau towards the later epochs, which is typical behavior as the model starts to reach its convergence on the given data with the current hyperparameters.

While the T5 model's loss curves are the most preferable among the four models considered, indicating effective learning and optimization, the curves still left room for improvement. Further refinement to ensure that the training and validation losses were minimized effectively could be achieved by optimizing the model's hyperparameters, extending the training process with more epochs of data, and employing techniques such as early stopping to prevent potential overfitting.

4.3 The Improved Transformer-based Model

Guided by the results of the comparative study presented above, the researcher selected the T5 transformer to make improvements on since it displayed a desirable convergence pattern out of the four considered models in terms of model loss and ROUGE Metrics scores. Therefore, the following section present the results of the improvements applied to the selected T5 model in chronological order, starting with the model's training and inference cost after quantization, a suite of hyperparameters that optimized models' performance, evaluation scores after modifying the output head of the model, and lastly, the architectural pipeline of the improved model.

4.3.1 Model Quantization

The existing quantization schemes were evaluated against the quantization technique devised by the researcher based on the Inference Cost and Training Cost in Floating-Point Operations per Second (FLOPs). Inference cost represents the computational requirements needed by a model to perform inference (make predictions on the test set, while the Training Cost denotes the total computational needs required during the model's training phase[103]. The table also contains a model configuration column, which shows the model's architecture, specifically the attention mechanism and activation function used for each training run. In Table 4.3, experimental results of different quantization schemes applied to the baseline T5 transformer model are presented, showing their implications on the training and inference cost of the model.

Table 4.3: Performance of Different Model Quantization Schemes

Quantization Scheme	Inference Cost	Model Configuration		Training Cost
		Attention Mechanism	Activation Function	
T5 (Zero-Q)	$29.45^{\pm 0.42}$	S.dot-product	ReLU	2.1×10^{8}
T5 (Weight-Q)	$26.74^{\pm 1.6}$	Self-attention	Sigmoid	1.8×10^{8}
T5 (Activation-Q)	$25.92^{\pm 0.9}$	Self-attention	ReLU	1.7×10^{8}
Proposed Approach	$\mathbf{22.42^{\pm 0.06}}$	**Multi-Head**	**Softmax**	$\mathbf{1.5\times 10^{8}}$

Table 4.3 presents the experimental results of different quantization schemes applied to the baseline T5 model. The T5 model, with 32-bit full precision and without any

quantization labelled as 'T5 (Zero-Q)', served as the baseline without any form of quantization. The model configuration was as follows: scaled-dot product attention as the attention mechanism and ReLU as the activation function. The training and inference costs, standing at 2.1×10^{8} and $29.45^{\pm 0.42}$ in floating point operations per second (FLOPs), respectively, were the highest for the baseline model without quantization.

The researcher also experimented with a lower bit floating-point precision level for weight quantization, denoted by 'T5 (Weight-Q)', which recorded a slight decrease in the inference and training cost compared to the baseline model. Additionally, the researcher also experimented with activation-quantization, labelled 'T5 (Activation-Q)', and a notable reduction was observed in the inference and training cost, with scores of $25.92^{\pm 0.9}$ and 1.7×10^{8}, respectively, in comparison with the baseline and the weight quantization approach.

Remarkably, the proposed quantization technique, a part of the contributions made by the researcher, employs adaptive quantization whereby quantization is dynamically applied to specific layers of the model as follows: Fully Connected Layers for activation quantization and Linear Layers for weight quantization achieved the lowest inference and training cost. The model configuration for the experiment with the proposed approach was softmax as the activation function and multi-head attention as the attention mechanism. The multi-head attention enabled the model to simultaneously attend to different parts of the input sequence. At the same time, the softmax activation function normalized the attention scores, ensuring that the model assigned appropriate weights to each token for effective information processing and generation. The proposed technique recorded an inference cost of $22.42^{\pm 0.06}$ and a training cost of 1.5×10^{8} in FLOPs, which is significantly and desirably lower in comparison with the baseline and the weight and activation quantization techniques. In the proposed approach, there was a FLOP decrease of $7.03^{\pm 0.42}$ and 0.6×10^{8} in the inference and training costs, respectively. This decrease demonstrated that the proposed approach managed to reduce the training and inference costs of the T5 model to lower levels desirable in neural network models.

4.3.2 Hyperparameter Optimization

Hyperparameter optimization is crucial in developing neural networks where the parameters that govern the training process are adjusted to enhance the model's performance. Unlike model parameters learned from the data, hyperparameters are set before the commencement of training and can significantly influence the model's performance [82].

As described in Chapter 3, the hyperparameter optimization process involved a combination of both manual and automatic tuning to establish the most effective hyperparameter set that minimized the predefined loss function while maximizing model performance. Table 4.4 presents a suite of hyperparameters that delivered optimal model performance in the improved T5 model.

Table 4.4: A Suite of Optimal Hyperparameters

Hyperparameter Configurations during Model Training	
Hyperparameter	**Value**
Optimizer	AdamW
Loss function	Cross entropy
Learning rate	1e-5
Weight decay	10^{-2}
Learning rate scheduler	CosineAnnealingLR
Dropout rate	0.1
gradient_accumulation_steps	4
num_train_epochs	20
warmup_steps	500
per_device_train_batch_size	8
per_device_eval_batch_size	16
evaluation_strategy	epochs

The table above outlines the optimized hyperparameters selected for model training, each chosen with a specific rationale and justification. The optimizer, Adaptive Moment Estimation with Weight Decay) AdamW was selected for its efficiency in updating model weights and handling weight decay, a technique that reduces overfitting in neural networks. As recommended in these studies [81] and [102], the

cross-entropy loss function works well for sequence generation tasks in guiding the model by quantifying the difference between predicted probabilities and the actual labels.

A low initial learning rate of 1e-5 was set, which was conservative enough to prevent getting stuck in the local minima phenomena. In contrast, a weight decay value of $10^{\wedge-2}$ was set as a regularization technique during training. The Cosine Annealing learning rate scheduler was selected to adjust the learning rate dynamically, emulating a cosine cycle, potentially leading to better model convergence over the training epochs.

Furthermore, a dropout rate of 0.1 was used to prevent co-adaptation of neurons by randomly omitting a subset of neurons during the training iterations. Gradient accumulation steps were also set to enable larger effective batch sizes and smoother gradients with reduced memory requirements. The model was trained over 20 epochs, providing ample opportunity to refine the weights without underfitting.

The warmup steps were set at 500 to provide a gradual ramp-up period for the learning rate, and it helped the model avoid abrupt changes in the parameter updates, which can cause the optimization process to diverge or get stuck in an unfavorable region of the loss trajectory. Batch sizes were set to 8 for training and 16 for evaluation, striking a balance between computational resource constraints and the effectiveness of batch normalization. The evaluation strategy of epochs ensured the model's performance was periodically assessed by facilitating a checkpoint-based approach to model validation.

4.3.3 Modifying the Model’s Output Layer

In the context of neural networks for Natural Language Processing, the Output Layer is crucial in transforming the model’s hidden representations into meaningful predictions[49]. However, the default architecture of the Text-to-Text Transfer Transformer includes an output layer that is pre-trained for various tasks such as machine translation, classification, summarization, and question-answering [55].

Therefore, to optimize or adapt the output head for a text summarization task, the researcher modified the output layer of the model as described in Chapter 3. The

modified output layer incorporated sequence generation and token selection parameters, contributing to the improved model's summary generation process. The temperature and top-k sampling parameters, used as the sequence generation and token selection parameters, allowed the model to generate the most probable sequences of tokens while focusing only on the relevant tokens during prediction.

The researcher compared the evaluation scores generated by the default output head with those generated by the modified output layer to demonstrate the improved model's performance on a text summarization task. Therefore, Table 4.5 presents the ROUGE Metric scores before and after the modifications, showing a numerical increase demonstrating improved performance.

Table 4.5: Comparing the Default and the Modified Output Layer

Model	CNN/DailyMail			XSUM		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
T5 (Default)	22.83	15.92	21.14	20.81	15.25	24.68
T5 (Modified)	**25.37**	**17.62**	**23.86**	**23.30**	**17.43**	**26.45**

The table presents the ROUGE scores obtained by the T5-Default output layer as the baseline and the T5-Modified output layer as the improved model on the CNN/DailyMail and XSUM datasets. On the CNN/DailyMail dataset, the T5-modified model outperforms the T5-Default model across all ROUGE metrics. Specifically, the T5-Modified model achieved ROUGE-1, ROUGE-2, and ROUGE-L scores of 25.37, 17.62, and 23.86, respectively, compared to 22.83, 15.92, and 21.14 for the default output layer.

Similarly, on the XSUM dataset, the T5 model with the modified output layer demonstrates improved performance with ROUGE-1, ROUGE-2, and ROUGE-L scores of 23.30, 17.43, and 26.45, respectively, while the default output layer records 20.81, 15.25, and 24.68 on these evaluation metrics. The higher ROUGE scores across both datasets indicate that the modified output layer of the T5 model generates summaries that better align with the reference summaries, showcasing its improved text summarization capabilities compared to the default T5’s output layer.

The improved performance is attributed to modifying the output layer, whereby the researcher applied a bi-phase fine-tuning approach to adapt the model to the task at hand. In the first phase of this approach, only the fully connected layers (FCL) were updated while the base model remained frozen. This phase ensured that the initial learning was task-specific so that the model could adapt to the specific requirements of the downstream task, which was text summarization[34]. In the second phase, a comprehensive training approach was adopted, whereby all layers of the base model were updated, treating the base model as a randomly initialized encoder. The bi-phase approach balanced task-specific optimization and generalization, ultimately improving model performance on text summarization.

4.3.4 The Architecture of the Improved Model

In this section, the researcher describes the architecture of the improved model, which clearly illustrates the improvements this study has proposed and integrated them into the baseline model. The algorithmic pipeline of the model starts with defining the requisite inputs and the desired output. Step 1 conducts data preprocessing: tokenizing, truncating, and padding the input sequences. Step 2 shows the proposed model quantization approach, one of the researcher's contributions, where the model layers are quantized to the desired precision levels.

In Step 3, hyperparameters are optimized by leveraging both manual and automatic tuning, which was the second improvement proposed in this study. Following this, Step 4 initiates the training and validation process on the preprocessed data using the optimized hyperparameters. Then, Step 5 modifies the output layer of the model to incorporate sequence generation and token selection parameters, which forms another contribution made by the researcher. Finally, Step 6 evaluates the model on the test set. The final outputs from the models are finetuned model parameters, metrics scores, a quantized model, and the generated summaries. The algorithmic pipeline of the improved model is attached in the appendices.

To complement the model’s algorithmic pipeline, the researcher additionally presents the diagrammatic architecture of the improved model, showing the entire process starting from the loading of the dataset and the data preprocessing steps, namely, tokenization, padding, and truncation. The architecture also shows the splitting of the

preprocessed dataset into three distinct splits for training, validation, and testing and where exactly they were fed into the proposed model. Three improvements, namely, model quantization, hyperparameter optimization, and modifying the output layer, are also illustrated in the architecture. The final step in the architecture is the model evaluation module, which computes the evaluation scores of the generated summaries. Figure 4.5 illustrates the model's architecture.

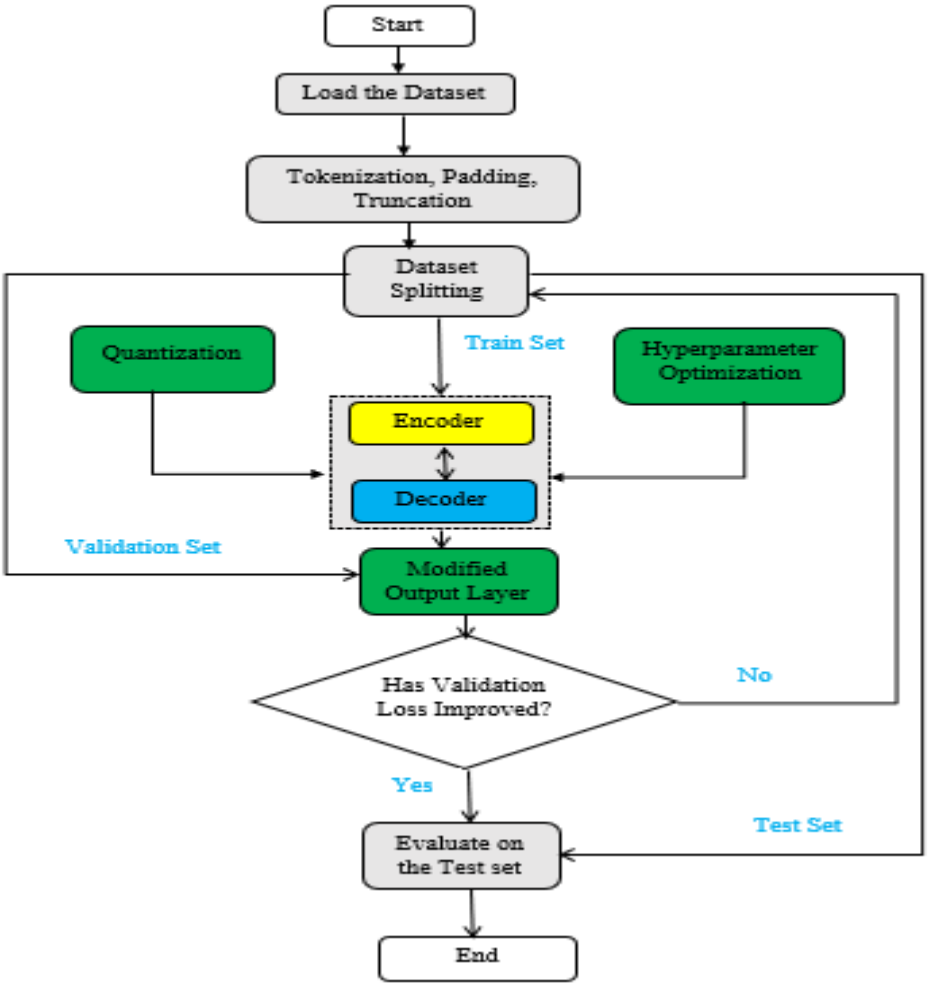


Figure 4.5: The Architecture of the Improved T5 Model

4.4 Evaluation Results of the Improved Model

Following the improvements implemented in the section above, the improved model was subjected to evaluation to ascertain the performance improvement. Therefore, this section presents detailed evaluation results, the datasets used in the evaluation, the comparison benchmarks, and the specific evaluation metrics used. The evaluation of

the proposed model was conducted as illustrated in the algorithm included in Chapter 3 under model evaluation in Section 3.7

Two datasets were used to perform the evaluation: the CNN/DailyMail, also used for model training, and the XSUM dataset, a completely unseen dataset, to evaluate how the model generalizes on unseen data. Two evaluation metrics were utilized: the ROUGE Metrics and the BLEU score; with the ROUGE having three different metrics in ROUGE-1, ROUGE-2, and ROUGE-L, while the BLEU score evaluated the model across zero and few-shot evaluation.

4.4.1 Model Performance on ROUGE Metrics

Table 4.6 shows the evaluation results on two text summarization benchmark datasets using the ROUGE evaluation metrics.

Table 4.6: Evaluation Results on ROUGE Metrics.

Model	CNN/DailyMail			XSUM		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
T5 (Baseline)	33.62	21.45	31.76	33.17	21.45	29.32
T5 (Improved)	**36.44**	**23.24**	**34.06**	**35.61**	**24.12**	**31.48**

Table 4.6 presents the evaluation results of the baseline model and improved model on the CNN/DailyMail and XSUM benchmark datasets using ROUGE Metrics. On the CNN/DailyMail dataset, the improved model outperforms the T5 (Baseline) across all ROUGE metrics by 3.05% cumulatively. The improved model achieves ROUGE-1, ROUGE-2, and ROUGE-L scores of 36.44, 23.24, and 34.06, respectively, while the baseline model records a score of 33.44, 21.45, and 31.76, respectively.

A similar trend is observed on the XSUM dataset, where the improved model outperforms the baseline model by a cumulative 7.27% in ROUGE metrics. The improved model achieved a ROUGE-1, ROUGE-2, and ROUGE-L scores of 35.61, 24.12, and 31.48, respectively, whereas the baseline trails behind with scores of 33.17, 21.45, and 29.32 for the same metrics. The improved model's consistently higher

ROUGE scores across benchmark datasets indicate its ability to generate summaries that match the reference summaries more accurately than the baseline model.

4.4.2 Model Performance on BLEU Score

The second evaluation metric used was the BLEU score, and the table in 4.7 shows the recorded results of the final model across two datasets, the XSUM and CNN/DailyMail, for Zero-shot and Few-Shot evaluation. The zero-shot approach facilitated model evaluation in generating summaries without training samples and relying on prior knowledge. It provided a suitable way to test the model's generalization ability on unseen data, which it was not explicitly trained on[104].

On the other hand, the few-shot approach involved training the model with a small sample that is typically smaller than the requisite size for supervised learning[104]. The researcher used this approach to test the model's performance on new tasks with a few training examples to determine the model's generalization ability to tasks with minimal training data. Table 4.7 presents the BLEU scores of the baseline model against the improved model on two different datasets across zero-shot and few-shot learning evaluations.

Table 4.7: Evaluation Results on BLEU Score

Model	BLEU SCORE			
	CNN/DailyMail		XSUM	
	Zero-Shot	Few-Shot	Zero-Shot	Few-Shot
T5 (Baseline)	23.64	29.86	15.30	17.43
T5 (Improved)	**27.14**	**32.56**	**18.14**	**22.12**

The table presents a comparison of the BLEU scores achieved by the baseline model and the improved model on the Zero-Shot and Few-Shot Text Summarization tasks across the CNN/DailyMail and XSUM datasets. For the CNN/DailyMail dataset, the proposed model outperforms the baseline model in both Zero-Shot and Few-Shot evaluation cumulatively by 4.2%, with BLEU scores of 27.14 and 32.56, respectively, compared to 23.64 and 29.86 for the baseline. On the XSUM dataset, the improved

model also demonstrates superior performance with a cumulative difference of 6.53%, attaining BLEU scores of 18.14 for Zero-Shot and 22.12 for Few-Shot summarization, while the baseline model scored 15.30 and 17.43, respectively. The higher BLEU scores across both datasets indicated that the improved model performed better in zero-shot and few-shot settings, in comparison with the baseline model.

4.4.3 Training and Validation Loss of the Model

The improved model was also evaluated based on its training and validation loss after fine-tuning it on the train set of the CNN/DailyMail dataset for 20 epochs. The loss curves for training and validation runs were recorded and presented in figure 4.6. The training and validation loss curves in Figure 4.6 demonstrate a highly desirable convergence pattern, indicating successful hyperparameter optimization and effective model training and learning.

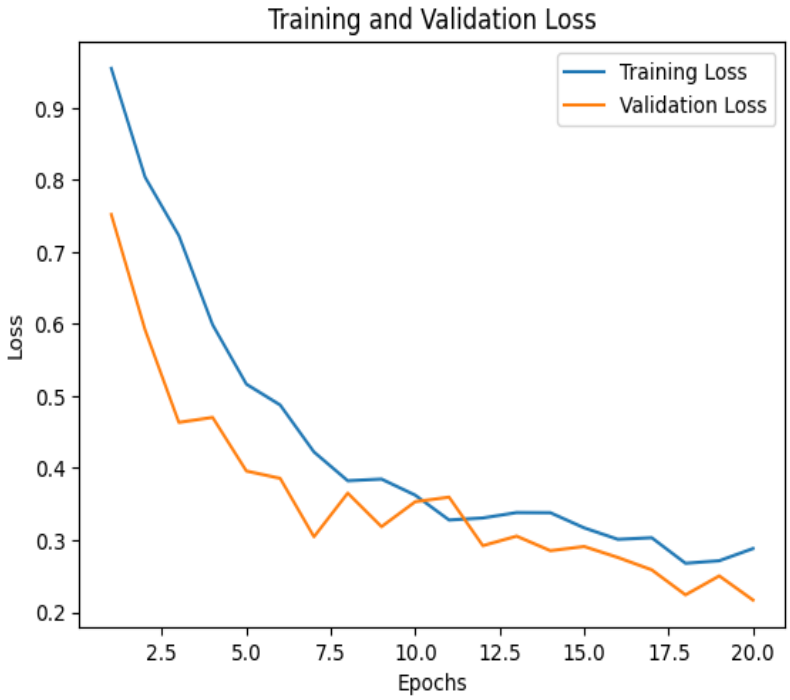


Figure 4.6: Training and Validation Loss – Improved T5 Model

The loss curves in figure 4.6 show that optimizing the model's hyperparameters has resulted in minimal training and validation losses, ultimately leading to a well-performing model. The training loss curve exhibits a remarkably rapid and consistent decrease from the initial epochs, signifying that the model effectively learns from the training data and minimizes the training loss. This steep decline can be attributed to

the choice of the AdamW optimizer, a learning rate with a scheduler, and weight decay as regularization techniques, which facilitated efficient convergence. The optimization also prevented the model from getting stuck in suboptimal solutions (Local Minima) during the trajectory of the loss curve.

The validation loss curve closely follows the training loss pattern, maintaining a relatively small gap between the two. This highly desirable characteristic suggests that the model generalizes well to unseen data and does not suffer from significant overfitting. The researcher effectively employed dropout, weight decay, and early stopping techniques to mitigate overfitting and ensure that the model's performance on the validation set remained consistent with its training performance.

While both loss curves exhibited minor fluctuations, particularly around epochs 7.5 and 12.5, these can be viewed as natural occurrences in the training process and did not significantly hinder the overall convergence of the improved model. The researcher attributes these fluctuations to outliers and potential noise in the data. However, the model's ability to recover and continue its descent toward lower loss values demonstrates its robustness and the effectiveness of the utilized hyperparameter optimization strategies.

4.4.4 Comparing the Improved Model with the Existing Models

In this section, the researcher compares the evaluation results of the improved model and the existing models in the literature. The key performance metrics, including the ROUGE and zero-shot BLEU scores, guided the comparison. The scope of the comparison was confined to text summarization models evaluated on evaluation metrics similar to those of the improved model. Benchmarking the improved model against existing models assisted this study in demonstrating the superior performance of the improved model across several evaluation metrics. Table 4.8 compares the evaluation results on two benchmark datasets.

Table 4.8: Comparing the Improved Model with Existing Models

Model	CNN/DailyMail				XSUM			
	R-1	**R-2**	**R-L**	**BLEU Score**	**R-1**	**R-2**	**R-L**	**BLEU Score**
Lead-3 Sentences	28.37	19.62	23.56	23.86	26.30	17.43	26.45	16.30
Radf et al.,2022	24.83	16.92	22.14	22.56	29.81	15.25	24.68	15.81
Senich et al.,2023	34.12	20.15	34.41	18.13	33.17	22.45	28.75	16.47
Kitaev et al.,2024	27.61	18.37	26.52	21.82	25.68	17.97	23.55	15.68
Zhang et al.,2023	33.69	21.58	28.43	19.86	35.79	22.82	27.48	16.19
Lapata et al.,2022	31.62	20.75	32.82	24.13	31.47	21.45	25.12	13.47
Improved Model	**36.44**	**23.24**	**34.06**	**27.14**	**35.61**	**24.12**	**31.48**	**18.14**

Based on the table comparing the improved model with existing models, it is evident that it has outperformed the other models across various evaluation metrics. For the CNN/DailyMail dataset, the proposed model achieves the highest R-1, R-2, and zero-shot BLEU Scores compared to the existing models in the literature. Similarly, on the XSUM dataset, the improved model outperforms the existing models in the literature evaluated on the same dataset and evaluation metrics in R-2, R-L, and zero-shot BLEU Score. Notably, the improved model's slightly lower performance in R-L for CNN/DailyMail and R-1 for XSUM is likely due to its optimization for overall performance across multiple datasets and metrics rather than specializing in these specific areas. However, the difference is relatively small and it is worth noting that despite these specific instances, the improved model showed strong performance across several metrics on both datasets, suggesting that the model has achieved a good balance in overall summarization performance.

4.5 Discussion

This section synthesizes the results obtained from this study. The discussion flows sequentially, guided by the results of the objectives of this study, starting with a discussion about the results from the comparative analysis conducted in objective one. This is followed by a discussion of the improvements that the researcher integrated into the model and then, finally, an interpretation of the final evaluation results of the improved model.

4.5.1 The Comparative Study of Transformers

The results of the first objective of this study, which was conducting a comparative analysis of four state-of-the-art transformers for text summarization, have the following interpretations that can be drawn from them. First of all, the findings presented in the figures showing the training and validation loss curves and the convergence patterns for the four models offer significant implications to researchers and practitioners. For instance, the fluctuation observed in BART's, PEGASUS's, and GPT's loss curves demonstrated the challenges transformer-based models might face when adapting pre-trained checkpoints to a downstream task. This finding has underscored the need to consider regularization techniques to prevent overfitting when adapting a pre-trained model to a smaller task-specific dataset[32].

From the findings, the researcher has also established that the T5 model distinguishes itself in text summarization primarily because of the unique text-to-text framework. This framework formulates every NLP task as a text-to-text problem, simplifying this model's pre-training and fine-tuning procedures. Additionally, this framework converts input text into a target text, creating a more versatile pre-training objective and allowing the model to generalize on various text tasks, including summarization[55]. The researcher has also confirmed that T5 is pre-trained on a large corpus with a Masked Language Modelling (MLM) objective. This enables it to easily understand the context and semantics of languages in natural language understanding and generation.

Moreover, the results have also shown that the T5 transformer's technical advantages are derived from its encoder-decoder architecture, which has a unique bidirectional encoder and a sequential decoder. The encoder in this architecture converts the input

sequences into a fixed-length vector representation, which the decoder subsequently decodes to produce output sequences [39]. Its encoder's bidirectional nature and the decoder's sequential focus allow T5 to focus on relevant text parts when creating summaries. This led to the observed high ROUGE Metric scores, which specify the superior capacity of the model in generating high-quality summaries. The researcher, therefore, attributes the high performance of the T5 in comparison with the other three transformers to its text-to-text formulation framework and the encoder-decoder architecture, making it the most suitable choice for the text summarization task.

4.5.2 The Improved Transformer-based Model

This section presents an interpretation of the conclusions drawn from the model improvements and how they contributed to improved performance in the proposed model. The results presented by the researcher show that the improved model has achieved significant performance improvement compared to the baseline model. Therefore, the section starts with a discussion on the results of model quantization, then proceeds to discuss the significance of hyperparameter optimization on the performance of the proposed model, and then ends with a discussion on the implications of modifying the output layer of the model on its text summarization capabilities.

4.5.2.1 Model Quantization

Based on the experimental results presented on model quantization, the researcher made several key observations and conclusions regarding the proposed quantization approach's impact on the proposed model's training and inference costs. This study's findings demonstrate that quantization is an effective compression technique in large language models, given that there was a FLOPs decrease of $7.03^{\pm 0.42}$ and 0.6×10^{8} in the inference and training costs, respectively.

The baseline T5 model, without quantization, served as a reference point with a relatively high training and inference cost, indicating the computational demands of the uncompressed model during training and inference[31]. However, with a reduction in the precision levels, as seen in the lower bit weight and activation quantization schemes, there was a decrease in the training and inference cost. This reduction in computational requirements during inference can be attributed to the compressed

model parameters, which enabled more efficient and quicker matrix multiplications and attention operations in the model encoder-decoder architecture.

The proposed approach recorded the lowest inference cost of $22.42^{\pm 0.06}$ in FLOPs among the evaluated approaches in this study. This decrease shows that the proposed quantization approach, which selectively applies quantization to performance-critical layers while preserving the precision levels in quantization-sensitive layers, can effectively reduce computational requirements, as demonstrated in the recorded results. The researcher also noted that the proposed model's training cost of 1.5×10^{8} in FLOPs was the lowest among the quantization approaches. This finding specifies that the proposed model excelled in reducing the inference cost efficiency and presented an optimized training process by reducing the computational cost during the model's training phase.

The findings have emphasized the importance of the quantization technique in optimizing Large Language Models to reduce their training and inference computational costs. However, the researcher points out that it is also important to note that while quantization can yield significant efficiency gains in terms of reducing computational costs, it may also introduce trade-offs in terms of model precision and performance, depending on the specific task and architecture[87], [88]. Therefore, the researcher contends that evaluating the quantization scheme and its impact on the target application is crucial when establishing the optimal balance between the effectiveness of the quantization scheme and the model's predictive performance.

4.5.2.2 Impact of Hyperparameter Optimization on Model Performance

The established suite of hyperparameters demonstrated that hyperparameter optimization contributed to the improved performance, as observed in the findings. This attests that this is an important step towards improved performance, convergence, and generalization in neural networks, particularly transformers. The final training and validation loss curves illustrated that the model portrayed the desired training and validation loss, which indicates smooth convergence and effective learning.

The researcher drew the following conclusions from the findings presented in the table showing the optimal hyperparameter suite. The choice of the AdamW optimizer was

driven by its effectiveness in handling sparse gradients and its ability to independently adapt the learning rates for each parameter [33]. Through the model's loss curves, the researcher ascertained that this optimizer proved advantageous for a huge-parameter model with a high-dimensional parameter space, as it helped mitigate the potential issue of radically different gradients across the parameter space.

The weight decay factor acted as a regularization technique that played a crucial role, as it enhanced generalization by applying L2 regularization directly to the weight tensors, avoiding the prospect of the prevention of the desired gradient descent[105]. The dropout rate of 0.1 also served as a regularization technique by randomly dropping neurons as the training progressed. This prevented overfitting by introducing sufficient noise to prevent the model from overlearning the training data. The introduction of this sufficient noise prevented the specific neurons from participating in forward and backward passes, which reduced the co-adaptation of neurons and helped the model generalize better to unseen data. The researcher appreciates these regularization techniques as essential for avoiding overfitting, particularly in transformers with many parameters, as they encouraged the model to learn more robust and generalizable representations[105]. Evaluating the model on an unseen test set demonstrated that this regularization was crucial in achieving a commendable model generalization performance.

The researcher has also demonstrated that the cross-entropy loss function, although a de facto standard for classification tasks, as it directly measures the performance of the model by quantifying the dissimilarity between the predicted probability distributions and the true label distributions, worked well for a sequence-to-sequence task[102]. This confirmed that this loss function can provide a smooth and convex optimization surface for several Natural Language Processing tasks, including text summarization. The loss function allowed for efficient convergence and gradient-based optimization, which the researcher believes led to the model's smooth convergence and better performance.

The findings have also demonstrated that the Cosine Annealing learning rate scheduler proved beneficial due to its ability to gradually reduce the learning rate throughout the training process based on a cosine curve. This approach has been empirically shown

to improve convergence and generalization performance compared to the traditional step-based or exponential decay schedulers[106]. This study has also attested that a warmup phase, such as the 500 warmup steps used in this study, facilitated a gradual increase in the learning rate during training. The loss curves have shown that this warmup phase reduced the chances of the model getting stuck in sub-optimal solutions while enabling the optimizer to adapt to the loss steps more effectively[106]. The researcher contests that this hyperparameter combination contributed to the desired model convergence, as observed in the training and validation loss.

The batch sizes, 8 for the per-device train batch size and 16 for the per-device validation batch size across 20 training epochs, were selected to optimize GPU memory utilization while enabling efficient parallelization provided by the PyTorch framework. The researcher used this specific batch size and number of epochs; therefore, this study has shed light on the importance of the requisite batch size and number of epochs when working with neural networks. The findings have established that there is a need to find the right balance in the number of epochs and batch sizes for training and evaluation of your model, given that batch sizes that are too small or too large can impact the quality of the gradient estimates, and consequently, the convergence pattern of a given model.

4.5.2.3 Modifying the Model's Output Layer

The T5 model's notable performance gains with the modified output layer emphasize how crucial it is to modify the model architecture to fit the characteristic requirements of the particular downstream task. While pre-trained language models such as the T5 have achieved remarkable success in various natural language processing tasks, their default architectures may not be optimally suited for every downstream task [105]. Therefore, the findings of this study have underscored the importance of fine-tuning and adapting the model to enhance its performance on specialized tasks such as text summarization.

The results presented in Table 4.5 demonstrated the effectiveness of modifying the output layer of the T5 model in improving its performance on the text summarization task. The researcher has attested that incorporating temperature and top-k sampling parameters in the modified output layer was crucial in guiding the summary generation

process, as evidenced by the higher ROUGE scores across the CNN/DailyMail and XSUM datasets. These parameters enabled the model to focus on the most relevant tokens during prediction, generating more coherent and informative summaries. The before-and-after samples of the generated summaries further illustrate the qualitative improvements, showcasing the proposed model's ability to capture the essence of the input text while maintaining conciseness and coherence in the generated summary.

The researcher also attributes the improved performance to the two-step fine-tuning approach employed during the training of the modified output layer. In first adapting the Fully Connected Layers to the text summarization task and then fine-tuning the entire model, the model balanced task-specific optimization and generalization[32]. This approach allowed the model to effectively learn the specific requirements of the text summarization task while leveraging the knowledge and representations learned from the pre-training phase.

4.5.3 The Evaluation Results of the Improved Model

The evaluation results have demonstrated the effectiveness of the model's improvements in improving its text summarization performance across different datasets and evaluation metrics. The results show that the improved T5 model consistently outperformed the baseline T5 model on the ROUGE metrics across the CNN/DailyMail and XSUM datasets. The improved model achieved higher ROUGE-1, ROUGE-2, and ROUGE-L scores, indicating its ability to generate summaries that accurately capture the key information from the input text and better align with the reference summaries. The researcher credits the improved performance to the improvements integrated into the baseline model, which allowed the model to better adapt to the specific requirements of the text summarization task.

This study has additionally demonstrated the improved model's superior performance in both zero-shot and few-shot evaluation, as evidenced by the evaluation results on the BLEU score metric. The researcher has demonstrated that the improved model can effectively generalize to unseen data and tasks with limited training examples, achieving higher BLEU scores compared to the baseline model on the CNN/DailyMail and XSUM datasets. This finding is particularly significant as it showcases the model's ability to leverage its pre-trained knowledge and rapidly adapt to new domains or

tasks, even with minimal training data. The improved generalization capabilities of the improved model can be attributed to the fine-tuning approach and the incorporation of token selection and sequence generation parameters, which enabled the model to generate more relevant and coherent summaries.

The inferences from the training loss and validation loss curves further validate the proposed model's effectiveness and the researcher's optimization strategies. The findings displayed a rapid and consistent decline in training loss and a close alignment between the training and validation loss curves, indicating successful hyperparameter optimization and effective model training. The insignificant difference between the training and validation losses demonstrates that the model generalizes well to new data without severe overfitting. The researcher's thoughtful use of hyperparameters such as dropout rate, weight decay, and early stopping played a crucial role in mitigating overfitting and ensuring the model's effectiveness. Even though minor fluctuations in the loss curves were observed, this study has demonstrated that the model's ability to recover and continue its descent towards lower loss values underscored the success of the model optimization strategies, which the researcher used in this study.

4.6 Summary

This chapter has covered the findings inferred from this study and discusses the interpretations made from those findings. The chapter began with a presentation of the comparative study that highlighted the architectural features and the performance of four transformer models, GPT, BART, T5, and PEGASUS, on text summarization tasks. The comparative study identified T5 as the most suitable transformer for text summarization. Therefore, the researcher selected this model as the baseline and implemented the described improvements onto its' baseline architecture. The improved T5 model was evaluated on the specified evaluation metrics, and the evaluation results were presented. Since the researcher employed a Pretest-Posttest Control Group research design, the evaluation results of the improved T5 model were compared to those of the baseline T5 model and presented in this chapter. A detailed discussion section was provided for each of the results presented in this chapter, guided by the study's specific objectives.

CHAPTER FIVE
CONCLUSION, RECOMMENDATIONS AND FUTURE WORK

5.1 Conclusion

This study's main objective was to develop an improved transformer-based model for text summarization of news articles. Therefore, the entire research was grounded on an in-depth exploration of text summarization literature, its foundational concepts and principles, and the practical implementation of the improvement strategies.

Three specific research objectives guided this study. The first specific objective was to determine a suitable transformer-based model among the four selected models from the literature, namely BART, GPT, T5, and PEGASUS, and how to improve its performance. These four transformers have recorded state-of-the-art performances in numerous NLP tasks and have been backed by literature to be high-performance models. For that reason, the researcher was justified to conduct a comparative study of the performance of these four models on a text summarization task. The results of the comparative study identified T5 as a suitable model in comparison with others because of its text-to-text formulation framework.

The second specific objective of this study was to develop an improved version of the model that had been identified in the first objective. Therefore, the researcher improved the baseline T5 model through three approaches. Firstly, the training and inference costs of the model were reduced through an adaptive quantization technique, whereby quantization was only applied to specific layers of the model; fully connected layers for activation quantization and linear layers for weight quantization, based on their tolerance and sensitivity to quantization. Hyperparameter optimization was also conducted using a combination of manual and automatic tuning techniques to optimize the model while preventing overfitting and stabilizing the training process. The researcher also modified the output head of the baseline model to adapt it to text summarization, given that the default output layer was pre-trained for several tasks but not yet optimized for a specific task, text summarization in this case.

This study’s third specific objective was to evaluate the performance of the improved model on benchmark text summarization datasets. The CNN/DailyMail and the XSUM datasets were used for model evaluation. Two metrics, the BLEU score and

ROUGE metrics, were utilized to assess the model's performance using the test splits of the two datasets. The XSUM dataset, which the model was not trained on, was considered primarily to ascertain the generalizability of the improved model on unseen data and to ensure model validity and reliability. The evaluation results demonstrated that implementing these improvements improved the T5 model's performance on a text summarization task in comparison with the performance of the baseline model.

The findings of this study have demonstrated the effectiveness of the proposed improvements to the transformer-based model for text summarization. The T5 model outperformed the state-of-the-art transformers in a comparative study, owing to its encoder-decoder architecture and text-to-text framework. The quantization technique reduced computational costs while maintaining performance, as evidenced by the lowest recorded training and inference costs. Hyperparameter optimization for hyperparameters, such as the dropout rate, weight decay, loss function, and learning rate scheduling, contributed to improved convergence and generalization. Modifying the output layer with temperature and top-k sampling parameters improved the model's ability to generate coherent and concise summaries.

The evaluation results showed consistent outperformance of the improved T5 model over the baseline T5 across ROUGE and BLEU metrics on two different datasets, indicating its effectiveness in capturing key information and aligning with reference summaries. The model's superior performance in zero-shot and few-shot settings additionally demonstrated its generalization capabilities, leveraging pre-trained knowledge to rapidly adapt to new domains or tasks with limited training data. The analysis of training and validation loss curves further validated the effectiveness of the optimization strategies in mitigating overfitting and ensuring the model's optimal performance.

5.2 Recommendations

Based on the research findings, the researcher proposes several recommendations for policy and practice. These recommendations offer valuable guidance for stakeholders, and policymakers and practitioners should consider them to progress text summarization research in particular and the Natural Language Processing domain.

5.2.1 Real-World Deployment

Conducting real-world deployment is imperative to practically utilize the effectiveness of the improved text summarization model. The researcher recommends the implementation of the model in real-world settings such as newsrooms or content aggregation platforms and soliciting feedback from end-users and domain experts. Through this deployment, valuable insights can be gained into various aspects, including the model's performance in a real-world setting and its practical usability.

5.2.2 Generalizing the Improved Model to Other Domains

The researcher recommends the application of the improved model to other domains beyond text summarization. Collaborating with domain experts and stakeholders is recommended to facilitate fine-tuning the model parameters for other closely related tasks. The generalizability of the model will be utilized by applying the model across different domains and datasets, hence underscoring its applicability to a broader range of natural language processing tasks.

5.2.3 Multi-Document Summarization Utilization

The researcher also recommends the utilization of the capabilities of the model to handle multi-document summarization. This recommendation can apply the improved model to process multiple input documents simultaneously, enabling the model to generate comprehensive summaries that encapsulate information from various sources. This extended utilization would significantly broaden the model's applicability, making it more useful in real-world settings where users often need to consume and synthesize information from numerous news reports on a given topic.

5.3 Future Work

This study has laid the groundwork for further research in text summarization. While the researcher has achieved the objectives within the study's scope, exciting opportunities exist for future exploration. In this section, the researcher highlights potential avenues for extending this work, which were not within the delimitations of this study. These future works can, therefore, be adopted to progress text summarization research while building upon the findings of this study.

5.3.1 Inclusion of Additional Model Optimization Techniques

In addition to quantization and hyperparameter optimization, further research into alternative model optimization techniques presents an opportunity to identify synergies that can further refine the performance of transformers in text summarization. This exploration could include various methods for model compression, such as knowledge distillation and pruning, aiming at reducing the model size and computational overhead while preserving model performance. Additional approaches could include exploring ensemble methods, which can involve the combination of different transformer models with varied hyperparameters or architectures. This approach could yield more robust and accurate summarization outputs by leveraging the complementary strengths of individual transformer-based models.

5.3.2 Fine-Tuning for Multilingual Summarization

To broaden the applicability of the developed text summarization model across diverse linguistic contexts, the researcher endorses fine-tuning it for multilingual summarization tasks as a future research direction. This could involve training the model on multilingual datasets and adapting its architecture to handle different languages effectively. The model can simultaneously learn to summarize in multiple languages by incorporating cross-lingual transfer learning techniques while leveraging shared representations across linguistic contexts. Furthermore, exploring zero-shot and few-shot learning approaches would also enable the model to generate summaries in languages for which it has not been explicitly fine-tuned, thus enhancing its versatility in multiple languages.

5.3.3 Introduction of Interactive Summarization Approaches

An exciting research direction in advancing text summarization could involve introducing interactive approaches that allow users to engage with the summarization process. The model can interact with users in real-time through reinforcement learning techniques, receiving feedback on generated summaries and adjusting its behaviour to produce more informative and contextually relevant outputs. Moreover, exploring active learning strategies would enable the model to strategically query users for input, iteratively refining the summarization process and optimizing summary quality. Additionally, designing explainable summarization interfaces that provide users with

insights into the summarization process, such as highlighting meaningful sentences or phrases in the source text and explaining the model's reasoning behind summary generation decisions, would enhance transparency and user trust in the summarization system.

REFERENCES

[1] R. Chettri, "Automatic Text Summarization," *International Journal of Computer Applications*, vol. 161, no. 1, pp. 5–7, Mar. 2019, doi: 10.5120/ijca2017912326.

[2] H. S., A. S., A. V., and R. K. Grace, "Summarization of News Articles Using Transformers," in *2022 5th International Conference on Advances in Science and Technology (ICAST)*, Mumbai, India: IEEE, Dec. 2022, pp. 159–163. doi: 10.1109/ICAST55766.2022.10039608.

[3] A. P. Widyassari, A. Affandy, E. Noersasongko, A. Z. Fanani, A. Syukur, and R. S. Basuki, "Literature Review of Automatic Text Summarization: Research Trend, Dataset and Method," in *2019 International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, Indonesia: IEEE, Jul. 2019, pp. 491–496. doi: 10.1109/ICOIACT46704.2019.8938454.

[4] A. Aries, D. Eddine Zegour, and W. K. Hidouci, "Automatic text summarization: What has been and has to be done." arXiv, Apr. 01, 2019. doi: 10.48550/arXiv.1904.00688.

[5] D. Bawden and L. Robinson, "Information Overload: An Overview," Oxford: Oxford University Press, 2020. doi: 10.1093/acrefore/9780190228637.013.1360.

[6] J. B. Schmitt, C. A. Debbelt, and F. M. Schneider, "Too much information? Predictors of information overload in the context of online news exposure," *Information, Communication & Society*, vol. 21, no. 8, pp. 1151–1167, Aug. 2018, doi: 10.1080/1369118X.2017.1305427.

[7] S. T. Ahmed, "Managing News Overload (MNO): The COVID-19 Infodemic," *Information*, vol. 11, no. 8, p. 375, Jul. 2020, doi: 10.3390/info11080375.

[8] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications*, vol. 165, p. 113679, Mar. 2021, doi: 10.1016/j.eswa.2020.113679.

[9] H. Siddiqui, S. Siddiqui, M. Rawat, A. Maan, S. Dhiman, and M. Asad, "Text Summarization using Extractive Techniques," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India: IEEE, Dec. 2021, pp. 28–31. doi: 10.1109/ICAC3N53548.2021.9725501.

[10] S. JUGRAN, A. KUMAR, B. S. TYAGI, and V. ANAND, "Extractive Automatic Text Summarization using SpaCy in Python & NLP," in *2021 International Conference on Advance Computing and Innovative Technologies in*

Engineering (ICACITE), Mar. 2021, pp. 582–585. doi: 10.1109/ICACITE51222.2021.9404712.

[11] J. Li, C. Zhang, X. Chen, Y. Cao, P. Liao, and P. Zhang, "Abstractive Text Summarization with Multi-Head Attention," in *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary: IEEE, Jul. 2019, pp. 1–8. doi: 10.1109/IJCNN.2019.8851885.

[12] M. A. I. Talukder, S. Abujar, A. K. M. Masum, S. Akter, and S. A. Hossain, "Comparative Study on Abstractive Text Summarization," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India: IEEE, Jul. 2020, pp. 1–4. doi: 10.1109/ICCCNT49239.2020.9225657.

[13] M. Kirmani, N. Manzoor Hakak, M. Mohd, and M. Mohd, "Hybrid Text Summarization: A Survey," in *Soft Computing: Theories and Applications*, K. Ray, T. K. Sharma, S. Rawat, R. K. Saini, and A. Bandyopadhyay, Eds., Singapore: Springer, 2019, pp. 63–73. doi: 10.1007/978-981-13-0589-4_7.

[14] J. Mugi Karanja and A. Matheka, "A Hybrid Model for Text Summarization Using Natural Language Processing," *OJIT*, vol. 5, no. 2, p. 65, Nov. 2022.

[15] I. Awasthi, K. Gupta, P. S. Bhogal, S. S. Anand, and P. K. Soni, "Natural Language Processing (NLP) based Text Summarization - A Survey," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India: IEEE, Jan. 2021, pp. 1310–1317. doi: 10.1109/ICICT50816.2021.9358703.

[16] D. Huang *et al.*, "What Have We Achieved on Text Summarization?" arXiv, Oct. 09, 2020. doi: 10.48550/arXiv.2010.04529.

[17] M. F. Mridha, A. A. Lima, K. Nur, S. C. Das, M. Hasan, and M. M. Kabir, "A Survey of Automatic Text Summarization: Progress, Process and Challenges," *IEEE Access*, vol. 9, pp. 156043–156070, 2021, doi: 10.1109/ACCESS.2021.3129786.

[18] K. Merchant and Y. Pande, "NLP Based Latent Semantic Analysis for Legal Text Summarization," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2018, pp. 1803–1807. doi: 10.1109/ICACCI.2018.8554831.

[19] F. Wang, J. L. Zhang, Y. Li, K. Deng, and J. S. Liu, “Bayesian Text Classification and Summarization via A Class-Specified Topic Model,” *Journal of Machine Learning Research*, vol. 22, no. 89, pp. 1–48, 2021.

[20] P. G. Magdum and S. Rathi, “A Survey on Deep Learning-Based Automatic Text Summarization Models,” in *Advances in Artificial Intelligence and Data Engineering*, N. N. Chiplunkar, and T. Fukao, Eds., Singapore: Springer Nature, 2021, pp. 377–392. doi: 10.1007/978-981-15-3514-7_30.

[21] F. Zhang, G. An, and Q. Ruan, “Transformer-based Natural Language Understanding and Generation,” in *2022 16th IEEE International Conference on Signal Processing (ICSP)*, Beijing, China: IEEE, Oct. 2022, pp. 281–284. doi: 10.1109/ICSP56322.2022.9965301.

[22] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[23] L. Tunstall, L. von Werra, and T. Wolf, *Natural language processing with transformers: building language applications with Hugging Face*, [First edition]. Sebastopol, CA: O’Reilly Media, Inc., 2022.

[24] A. Gillioz, J. Casas, E. Mugellini, and O. A. Khaled, “Overview of the Transformer-based Models for NLP Tasks,” presented at the 2020 Federated Conference on Computer Science and Information Systems, Sep. 2020, pp. 179–183. doi: 10.15439/2020F20.

[25] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate.” arXiv, May 19, 2016. doi: 10.48550/arXiv.1409.0473.

[26] X. Zhu, D. Cheng, Z. Zhang, S. Lin, and J. Dai, “An Empirical Study of Spatial Attention Mechanisms in Deep Networks.” arXiv, Apr. 11, 2019. doi: 10.48550/arXiv.1904.05873.

[27] S. Leelaluk, T. Minematsu, Y. Taniguchi, F. Okubo, T. Yamashita, and A. Shimada, “Scaled-Dot Product Attention for Early Detection of At-risk Students,” in *2022 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Dec. 2022, pp. 316–322. doi: 10.1109/TALE54877.2022.00059.

[28] A. Chernyavskiy, D. Ilvovsky, and P. Nakov, “Transformers: ‘The End of History’ for Natural Language Processing?,” in *Machine Learning and Knowledge Discovery in Databases. Research Track*, N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, and J. A. Lozano, Eds., Cham: Springer International Publishing, 2021, pp. 677–693. doi: 10.1007/978-3-030-86523-8_41.

[29] A. M. P. Braşoveanu and R. Andonie, “Visualizing Transformers for NLP: A Brief Survey,” in *2020 24th International Conference Information Visualisation (IV)*, Sep. 2020, pp. 270–279. doi: 10.1109/IV51561.2020.00051.

[30] W. Merrill and A. Sabharwal, “The Parallelism Tradeoff: Limitations of Log-Precision Transformers,” *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 531–545, Jun. 2023, doi: 10.1162/tacl_a_00562.

[31] Z. Li *et al.*, “Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers,” in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 5958–5968. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.mlr.press/v119/li20m.html

[32] J. Petty and R. Frank, “Transformers Generalize Linearly.” arXiv, Sep. 24, 2021. doi: 10.48550/arXiv.2109.12036.

[33] Y. Chen *et al.*, “Towards Learning Universal Hyperparameter Optimizers with Transformers.”

[34] P. Xu *et al.*, “Optimizing Deeper Transformers on Small Datasets.” arXiv, May 31, 2021. doi 10.48550/arXiv.2012.15355.

[35] M. Bhandari, P. Gour, A. Ashfaq, P. Liu, and G. Neubig, “Re-evaluating Evaluation in Text Summarization,” 2020, doi: 10.48550/ARXIV.2010.07100.

[36] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks.” arXiv, Feb. 22, 2018. doi: 10.48550/arXiv.1801.01078.

[37] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A review of convolutional neural networks in computer vision,” *Artif Intell Rev*, vol. 57, no. 4, p. 99, Mar. 2024, doi: 10.1007/s10462-024-10721-6.

[38] X.-C. Tai, H. Liu, and R. Chan, “PottsMGNet: A Mathematical Explanation of Encoder-Decoder Based Neural Networks,” *SIAM J. Imaging Sci.*, vol. 17, no. 1, pp. 540–594, Mar. 2024, doi: 10.1137/23M1586355.

[39] K. Aitken, V. Ramasesh, Y. Cao, and N. Maheswaranathan, “Understanding How Encoder-Decoder Architectures Attend,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 22184–22195. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/hash/ba3c736667394d5082f86f28aef38107-Abstract.html

[40] S. Sharma, S. Sharma, and A. Athaiya, “ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” *IJEAST*, vol. 04, no. 12, pp. 310–316, May 2020, doi: 10.33564/IJEAST.2020.v04i12.054.

[41] J. Baan, M. ter Hoeve, M. van der Wees, A. Schuth, and M. de Rijke, “Understanding Multi-Head Attention in Abstractive Summarization.” arXiv, Nov. 10, 2019. doi: 10.48550/arXiv.1911.03898.

[42] C. Wei, J. Lee, Q. Liu, and T. Ma, “On the Margin Theory of Feedforward Neural Networks,” Sep. 2018, Accessed: Apr. 25, 2024. [Online]. Available: https://openreview.net/forum?id=HJGtFoC5Fm

[43] P.-C. Chen, H. Tsai, S. Bhojanapalli, H. W. Chung, Y.-W. Chang, and C.-S. Ferng, “A Simple and Effective Positional Encoding for Transformers.” arXiv, Nov. 03, 2021. doi: 10.48550/arXiv.2104.08698.

[44] Y.-A. Wang and Y.-N. Chen, “What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding.” arXiv, Oct. 10, 2020. doi: 10.48550/arXiv.2010.04903.

[45] Z.-H. Jiang *et al.*, “All Tokens Matter: Token Labeling for Training Better Vision Transformers,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 18590–18602. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/hash/9a49a25d845a483fae4be7e341368e36-Abstract.html

[46] F. Liu, X. Ren, Z. Zhang, X. Sun, and Y. Zou, “Rethinking Skip Connection with Layer Normalization in Transformers and ResNets.” arXiv, May 15, 2021. doi: 10.48550/arXiv.2105.07205.

[47] H. H. Tan and K. H. Lim, “Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization,” in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, Jun. 2019, pp. 1–4. doi: 10.1109/ICSCC.2019.8843652.

[48] R. Xiong *et al.*, “On Layer Normalization in the Transformer Architecture,” in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 10524–10533. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.mlr.press/v119/xiong20b.html

[49] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, “Structured Output Layer neural network language model,” in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 5524–5527. doi: 10.1109/ICASSP.2011.5947610.

[50] M. Lewis *et al.*, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.” arXiv, Oct. 29, 2019. Accessed: Dec. 01, 2023. [Online]. Available: http://arxiv.org/abs/1910.13461

[51] A. Venkataramana, K. Srividya, and R. Cristin, “Abstractive Text Summarization Using BART,” in *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, Oct. 2022, pp. 1–6. doi: 10.1109/MysuruCon55714.2022.9972639.

[52] H. Yadav, N. Patel, and D. Jani, “Fine-Tuning BART for Abstractive Reviews Summarization,” in *Computational Intelligence*, A. Shukla, B. K. Murthy, N. Hasteer, and J.-P. Van Belle, Eds., Singapore: Springer Nature, 2023, pp. 375–385. doi: 10.1007/978-981-19-7346-8_32.

[53] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: state of the art, current trends and challenges,” *Multimed Tools Appl*, vol. 82, no. 3, pp. 3713–3744, Jan. 2023, doi: 10.1007/s11042-022-13428-4.

[54] Y. Chen and Q. Song, “News Text Summarization Method based on BART-TextRank Model,” in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Mar. 2021, pp. 2005–2010. doi: 10.1109/IAEAC50856.2021.9390683.

[55] C. Raffel *et al.*, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” arXiv, Sep. 19, 2023. Accessed: Dec. 01, 2023. [Online]. Available: http://arxiv.org/abs/1910.10683

[56] Y. Tay *et al.*, “Scale Efficiently: Insights from Pre-training and Fine-tuning Transformers.” arXiv, Jan. 30, 2022. doi: 10.48550/arXiv.2109.10686.

[57] A. Garg *et al.*, "NEWS Article Summarization with Pretrained Transformer," in *Advanced Computing*, D. Garg, K. Wong, J. Sarangapani, and S. K. Gupta, Eds., Singapore: Springer, 2021, pp. 203–211. doi: 10.1007/978-981-16-0401-0_15.

[58] A. G. Etemad, A. I. Abidi, and M. Chhabra, "Fine-Tuned T5 for Abstractive Summarization," *International Journal of Performability Engineering*, vol. 17, no. 10, p. 900, Oct. 2021, doi: 10.23940/ijpe.21.10.p8.900906.

[59] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization." arXiv, Jul. 10, 2020. Accessed: Dec. 01, 2023. [Online]. Available: http://arxiv.org/abs/1912.08777

[60] T. Goodwin, M. Savery, and D. Demner-Fushman, "Flight of the PEGASUS? Comparing Transformers on Few-shot and Zero-shot Multi-document Abstractive Summarization," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 5640–5646. doi: 10.18653/v1/2020.coling-main.494.

[61] Y. Bitton, G. Stanovsky, M. Elhadad, and R. Schwartz, "Data Efficient Masked Language Modeling for Vision and Language." arXiv, Sep. 05, 2021. doi: 10.48550/arXiv.2109.02040.

[62] S. Rothe, J. Maynez, and S. Narayan, "A Thorough Evaluation of Task-Specific Pretraining for Summarization," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 140–145. doi: 10.18653/v1/2021.emnlp-main.12.

[63] T.-H. Yang, C.-C. Lu, and W.-L. Hsu, "More than Extracting 'Important' Sentences: the Application of PEGASUS," in *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Taichung, Taiwan: IEEE, Nov. 2021, pp. 131–134. doi: 10.1109/TAAI54685.2021.00032.

[64] D. Jin, S. Gao, S. Kim, Y. Liu, and D. Hakkani-Tür, "Towards Textual Out-of-Domain Detection Without In-Domain Labels," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1386–1395, 2022, doi: 10.1109/TASLP.2022.3162081.

[65] G. Yenduri *et al.*, "GPT (Generative Pre-trained Transformer) – A Comprehensive Review on Enabling Technologies, Potential Applications,

Emerging Challenges, and Future Directions," *IEEE Access*, pp. 1–1, 2024, doi: 10.1109/ACCESS.2024.3389497.

[66] Q. Zhu, L. Li, L. Bai, and F. Hu, "Chinese text summarization based on fine-tuned GPT2," in *Third International Conference on Electronics and Communication; Network and Computer Technology (ECNCT 2021)*, M. K. Mohiddin, S. Chen, and S. F. EL-Zoghdy, Eds., Harbin, China: SPIE, Mar. 2022, p. 109. doi: 10.1117/12.2629132.

[67] Z. Yang, Y. Dong, J. Deng, B. Sha, and T. Xu, "Research on Automatic News Text Summarization Technology Based on GPT2 Model," in *2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture*, in AIAM2021. New York, NY, USA: Association for Computing Machinery, Mar. 2022, pp. 418–423. doi: 10.1145/3495018.3495091.

[68] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, "Pre-Trained Language Models and Their Applications," *Engineering*, vol. 25, pp. 51–65, Jun. 2023, doi: 10.1016/j.eng.2022.04.024.

[69] W. Shi and V. Demberg, "Next Sentence Prediction Helps Implicit Discourse Relation Classification within and across Domains," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5790–5796. doi: 10.18653/v1/D19-1586.

[70] Y. Sun, Y. Zheng, C. Hao, and H. Qiu, "NSP-BERT: A Prompt-based Few-Shot Learner through an Original Pre-training Task —— Next Sentence Prediction," in *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 3233–3250. Accessed: Apr. 25, 2024. [Online]. Available: https://aclanthology.org/2022.coling-1.286

[71] L. Li, Q. Wang, B. Zhao, X. Li, A. Zhou, and H. Wu, "Pre-Training and Fine-Tuning with Next Sentence Prediction for Multimodal Entity Linking," *Electronics*, vol. 11, no. 14, Art. no. 14, Jan. 2022, doi: 10.3390/electronics11142134.

[72] A. Alajrami and N. Aletras, “How does the pre-training objective affect what large language models learn about linguistic properties?” arXiv, Mar. 19, 2022. doi: 10.48550/arXiv.2203.10415.

[73] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, “Transfer Learning in Natural Language Processing,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, A. Sarkar and M. Strube, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 15–18. doi: 10.18653/v1/N19-5004.

[74] I. Sarhan and M. Spruit, “Can We Survive without Labelled Data in NLP? Transfer Learning for Open Information Extraction,” *Applied Sciences*, vol. 10, no. 17, Art. no. 17, Jan. 2020, doi: 10.3390/app10175758.

[75] E. Radiya-Dixit and X. Wang, “How fine can fine-tuning be? Learning efficient language models,” in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, PMLR, Jun. 2020, pp. 2435–2443. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.mlr.press/v108/radiya-dixit20a.html

[76] S. Chen, Y. Zhang, and Q. Yang, “Multi-Task Learning in Natural Language Processing: An Overview.” arXiv, Sep. 19, 2021. doi: 10.48550/arXiv.2109.09138.

[77] Z. Huang, M. Rao, A. Raju, Z. Zhang, B. Bui, and C. Lee, “MTL-SLT: Multi-Task Learning for Spoken Language Tasks,” in *Proceedings of the 4th Workshop on NLP for Conversational AI*, B. Liu, A. Papangelis, S. Ultes, A. Rastogi, Y.-N. Chen, G. Spithourakis, E. Nouri, and W. Shi, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 120–130. doi: 10.18653/v1/2022.nlp4convai-1.11.

[78] J. Chen, H. Lin, X. Han, and L. Sun, “Benchmarking Large Language Models in Retrieval-Augmented Generation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, Art. no. 16, Mar. 2024, doi: 10.1609/aaai.v38i16.29728.

[79] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 9459–9474. Accessed: Apr. 25, 2024. [Online]. Available:

https://proceedings.neurips.cc/paper_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[80] C. Wang *et al.*, “Survey on Factuality in Large Language Models: Knowledge, Retrieval, and Domain-Specificity.” arXiv, Dec. 16, 2023. doi: 10.48550/arXiv.2310.07521.

[81] S. Wanjau, G. Wambugu, and A. Oirere, “Empirical Evaluation of Adaptive Optimization on the Generalization Performance of Convolutional Neural Networks,” *Information Systems*, 2021.

[82] E. Bartz, T. Bartz-Beielstein, M. Zaefferer, and O. Mersmann, Eds., *Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide*. Singapore: Springer Nature Singapore, 2023. doi: 10.1007/978-981-19-5170-1.

[83] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, “Gradient-based optimizer: A new metaheuristic optimization algorithm,” *Information Sciences*, vol. 540, pp. 131–159, Nov. 2020, doi: 10.1016/j.ins.2020.06.037.

[84] D. Maclaurin, D. Duvenaud, and R. P. Adams, “Gradient-based Hyperparameter Optimization through Reversible Learning.” arXiv, Apr. 02, 2015. Accessed: Apr. 25, 2024. [Online]. Available: http://arxiv.org/abs/1502.03492

[85] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A Survey of Quantization Methods for Efficient Neural Network Inference,” in *Low-Power Computer Vision*, Chapman and Hall/CRC, 2022.

[86] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, “Post-Training Quantization for Vision Transformer,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 28092–28103. Accessed: Apr. 25, 2024. [Online]. Available: https://proceedings.neurips.cc

[87] Y. Boo, S. Shin, and W. Sung, “Quantized Neural Networks: Characterization and Holistic Optimization,” in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2020, pp. 1–6. doi: 10.1109/SiPS50750.2020.9195245.

[88] M. Shen, F. Liang, R. Gong, Y. Li, C. Li, and C. Lin, “Once Quantization-Aware Training: High Performance Extremely Low-Bit Architecture Search,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 5340–5349.

[89] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, “A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification.” Oct. 23, 2023. doi: 10.1145/3623402.

[90] A. R. Fabbri, W. Kryściński, B. McCann, C. Xiong, R. Socher, and D. Radev, “SummEval: Re-evaluating Summarization Evaluation,” *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 391–409, Apr. 2021, doi: 10.1162/tacl_a_00373.

[91] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation”.

[92] M. Barbella and G. Tortora, “Rouge Metric Evaluation for Text Summarization Techniques.” Rochester, NY, May 26, 2022. doi: 10.2139/ssrn.4120317.

[93] S. Banerjee and A. Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments,” in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Association for Computational Linguistics, Jun. 2017, pp. 65–72. Accessed: Apr. 25, 2024. [Online]. Available: https://aclanthology.org/W05-0909

[94] N. Iskender, T. Polzehl, and S. Möller, “Reliability of Human Evaluation for Text Summarization: Lessons Learned and Challenges Ahead,” in *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, Online: Association for Computational Linguistics, Apr. 2021, pp. 86–96. Accessed: Apr. 25, 2024. [Online]. Available: https://aclanthology.org/2021.humeval-1.10

[95] E. Brent and P. D. Leedy, “Practical Research: Planning and Design,” *Teaching Sociology*, vol. 18, no. 2, p. 248, Apr. 2016, doi: 10.2307/1318509.

[96] D. Wyse, L. E. Suter, E. Smith, and N. Selwyn, “The BERA/SAGE Handbook of Educational Research,” pp. 1–1170, 2016.

[97] J. Kamiri, G. M. Wambugu, and A. M. Oirere, “A Comparative Study of Deep Learning and Transfer Learning in Detection of Diabetic Retinopathy,” *IJCATR*, vol. 11, no. 07, pp. 247–254, Jul. 2022, doi: 10.7753/IJCATR1107.1001.

[98] A. Choudhary, M. Alugubelly, and R. Bhargava, “A Comparative Study on Transformer-based News Summarization,” in *2023 15th International Conference on Developments in eSystems Engineering (DeSE)*, Baghdad & Anbar, Iraq: IEEE, Jan. 2023, pp. 256–261. doi: 10.1109/DeSE58274.2023.10099798.

[99] R. N. Ndung'u, "Data Preparation for Machine Learning Modelling," 2022, Accessed: Apr. 26, 2024. [Online]. Available: http://repository.mut.ac.ke:8080/xmlui/handle/123456789/6104

[100] C. P. Chai, "Comparison of text preprocessing methods," *Natural Language Engineering*, vol. 29, no. 3, pp. 509–553, May 2023, doi: 10.1017/S1351324922000213.

[101] Y. Zhu *et al.*, "Hot or Cold? Adaptive Temperature Sampling for Code Generation with Large Language Models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, Art. no. 1, Mar. 2024, doi: 10.1609/aaai.v38i1.27798.

[102] A. Mao, M. Mohri, and Y. Zhong, "Cross-Entropy Loss Functions: Theoretical Analysis and Applications," in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 2023, pp. 23803–23828. Accessed: Apr. 26, 2024. [Online]. Available: https://proceedings.mlr.press/v202/mao23b.html

[103] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the Computational Cost of Deep Learning Models," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018, pp. 3873–3882. doi: 10.1109/BigData.2018.8622396.

[104] S. Rahman, S. Khan, and F. Porikli, "A Unified Approach for Conventional Zero-Shot, Generalized Zero-Shot, and Few-Shot Learning," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, Nov. 2018, doi: 10.1109/TIP.2018.2861573.

[105] T. van Laarhoven, "L2 Regularization versus Batch and Weight Normalization." arXiv, Jun. 16, 2017. doi: 10.48550/arXiv.1706.05350.

[106] Z. Liu, "Super Convergence Cosine Annealing with Warm-Up Learning Rate," in *CAIBDA 2022; 2nd International Conference on Artificial Intelligence, Big Data and Algorithms*, Jun. 2022, pp. 1–7. Accessed: Apr. 26, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10104453

APPENDICES

Appendix 1: Data Pre-processing Algorithm

Data Pre-processing Algorithm

Require: Raw dataset
Ensure: Pre-processed dataset

Step 1. Tokenization:
  for all input sequences in dataset, **do**
    token_ids, attention_mask = AutoTokenizer.tokenize(input_sequence)
    input_tensors. append ((token_ids, attention_mask))
  end for

Step 2. Truncation and Padding:
  max_input_length = 512
  for all (token_ids, attention_mask) in input_tensors **do**
    if len(token_ids) > max_input_length then
      token_ids = token_ids[:max_input_length]
      attention_mask = attention_mask[:max_input_length]
    Else
      padding_length = max_input_length - len(token_ids)
      token_ids = token_ids + [PAD_TOKEN] * padding_length
      attention_mask = attention_mask + [0] * padding_length
    end if
  end for

Step 3. Dataset Splitting:
  train_data, eval_data, test_data = split_data (input_tensors, ratios= [0.7, 0.2, 0.1])

return train_data, eval_data, test_data

Appendix 2: The Comparative Study's Algorithm

Algorithm for the Comparative Study of Transformers

Input: $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ // A corpus of text-summary pairs

Output: $R = \{R_1, R_2, ..., R_k\}$ // ROUGE Metric scores

Step 1: Load the Dataset

$(X) \leftarrow$ Access(D) // Access the text corpus from the dataset

$(X) \leftarrow$ Split $(X_{tr}, X_{val}, X_{te})$ // Split the dataset into train, eval, and test sets

Step 2: Pre-process the Dataset

for $(x, y) \in \mathcal{D}$ do:

$(\tilde{x}, \tilde{y}) \leftarrow$ Tokenize $(x, y, \mathcal{T})$ //Tokenize using auto-tokenizer T

$(\hat{x}, \hat{y}) \leftarrow$ Encode $(\tilde{x}, \tilde{y})$ // Convert tokens into input IDs

$(\hat{x}, \hat{y}) \leftarrow (\hat{x}, \hat{y}, l_{max})$ //Truncate/Pad sequences

$(\hat{x}, \hat{y}) \leftarrow a_x, a_y)$// Create attention masks

end for

Step 3: Load the Pre-trained Model from Hugging Face

$m \leftarrow$modelvariant $(\{gpt2\text{-}xl, T5\text{-}large, bart\text{-}large, Pegasus\ cnn_dm\})$

$m \leftarrow$ LoadModel (m, θ_0) // Load model with pre-trained weights θ

Step 4: Define the Training Arguments

Set:

Optimizer to AdamW

Initialize learning rate α_0 and scheduler

Gradient accumulation steps

Batch size for training and evaluation

Warmup steps, weight decay, and num_train_epochs

Checkpointing and logging configurations

Evaluation strategy to epochs

Step 5: Train and validation step

for $(x, y) \in [1, n_e]$ in num_train_epochs **do:**

$\alpha \leftarrow$lr. Step () // Apply learning rate scheduler step

Perform a forward pass of the model

Compute cross-entropy loss

Perform a backward pass

if gradient accumulation $\neq 0$ and steps = = 0:

n. step () // Update model weights

n. zero_grad () // Clear gradients

end if

if evaluation strategy = steps, EvaluateEverySteps () // Validate on (X_{val})

if improved validation loss, save checkpoint

end if

end for

Step 6: Evaluate the Models using ROUGE Metrics

$\hat{m} \leftarrow$ LoadCheckpoints

for $x \in \mathcal{X}_{te}$ **do**: // Evaluate on the test set

$\hat{y} \leftarrow \hat{m}$. generate($x$) // Generate the summary

$R \leftarrow$ ROUGE ($\hat{y}$, y) // Compare the generated vs the reference summary

end for

Calculate and return the average ROUGE scores across the test set

End Algorithm

Appendix 3: The Improved Model's Algorithm

Improved Model's Algorithm

Input: Dataset ***D***, Pre-trained model ***M*** with parameters (***θ_pretrained***), Hyperparameters ***H***, Quantization Scheme ***Q***, Evaluation metrics ***S.***

Output: ***θ_*** finetuned parameters, quantized_model, Evaluation Results, Generated Summary.

Step 1: Data Preprocessing

```
for input_sequence in D:
        token_ids, attention_mask = AutoTokenizer.tokenize(input_sequence)
input_tensors.
        append ((token_ids, attention_mask))
end for
        max_length = 512
for (token_ids, attention_mask) in input_tensors:
        token_ids = token_ids[:max_length]
        attention_mask = attention_mask[:max_length]
        padding_length = max_length - len(token_ids)
        token_ids += [PAD_TOKEN] * padding_length
        attention_mask += [0] * padding_length
end for
       train_data, eval_data, test_data = split_data (input_tensors, ratios= [0.7, 0.2,
0.1])
return train_data, eval_data, test_data
```

Truncation and Padding

Step 2: Model Quantization

```
model_layers = get_model_structure ()
high_precision, low_precision = 32, 16

for layer in model_layers:
   if layer. type == "Fully-Connected":
      layer. activations = quantize_to_precision (layer. activations, high_precision)
   elseif layer. type == "Linear":
      layer. weights = quantize_to_precision (layer. weights, low_precision)

end for
```

Quantization is done selectively, based on the type of layer

Step 3: Hyperparameter Optimization

```
# Define hyperparameter search space
search_space = {“learning_rate", "weight_decay" "batch_size" "dropout_rate"
"optimizer" "gradient_accumulation_steps" "warmup_steps"}

# Perform manual tuning
manual_tune (model, train_data, eval_data)

# Perform automatic hyperparameter optimization
best_config = tune. Run (
   tune. with_parameters (train_model, data=train_data, eval_data=eval_data),
   metric="val_loss", search_alg = tune. gradient_search,
gradient_descentOpt(space=search_space), num_samples=10, mode= “min”)
print (f “Best hyperparameters: {best_config.get_flat_config ()}")
```

Finding the best fit via both Manual and Automatic Tuning

Step 4: Training and Validation

```
Train M on D_train using H
for (x, y) ∈ [1, n_e] in num_train_epochs do:
   Shuffle D_train
     for batch (X, Y) in D_train Y’ on _M_(train_data)
        Update θ using loss (Y, Y')
     end for
      Validate M on (X_eval) using Epochs ()
       Save best checkpoint
 end for
```

Step 5: Modifying the Output Layer

```
Define ModifiedOutputLayer:

   Initialize model, vocab_size, temp, top_k
   self. model = model, self. lm_head [Linear (model. config. d_model,
vocab_size)]
   function generate_summary (encoder_outputs, attention_mask):
   summary_ids = []
   outputs = self. model. decoder (inputs, encoder_outputs, attention_mask)
   logits = self. lm_head (outputs [logits = Softmax(logits)])
```

while not done **do**:
token = sample (logits, temp, top_k)
summary_ids. append(token)
end while
return summary_ids

Step 6: Model Evaluation

for $x \in \mathcal{X}_{\text{test}}$ **do**: // Evaluate on the test set
X'_test = test ***M*** on (X'_test)
Update ***S*** using ***_E_*** (X_test, X'_test)
end for

return θ_finetuned, metrics_scores, quantized_model, generated_summary

End Algorithm

Appendix 4: Approval of Research Proposal and Supervisors

MURANG'A UNIVERSITY OF TECHNOLOGY

Office of the Director, Postgraduate Studies

E-mail: bps@mut.ac.ke
Website: www.mut.ac.ke

P.O. Box 75-10200
Murang'a, Kenya

Ref: MUT/ARP/PGS/19/2020 /VOL.1 **Date:** 7th November, 2023

Mr. Charles Munyao Muia (SC400/5708/2021)
School of Computing and Information Technology,
Murang'a University of Technology,
P.O. Box 75-10,200
Murang'a.

Dear Mr. Muia,

RE: APPROVAL OF RESEARCH PROPOSAL AND SUPERVISORS.

I am pleased to inform you that the Directorate of Postgraduate Studies on 13th October, 2023 considered and approved your Masters research proposal entitled ***"An Improved Transformer-Based Model for Text Summarization of News Articles"*** and appointed the following as supervisors:

1. Dr. Aaron Mogere Oirere - Murang'a University of Technology
2. Dr. Rachael Njeri Ndung'u - Murang'a University of Technology

You may now proceed with your data collection subject to obtaining research permit from NACOSTI, if required. You should also begin consulting your supervisors and submit through them quarterly progress reports to the Director Postgraduate Studies through your CoD and School Dean. Progress Reports forms can be accessed in the Student Portal.

It is the policy and regulations of the University that you observe deadlines. The guidelines on Postgraduate supervision can be accessed in the post graduate Handbook.

Your responsibilities as a student will include, among others;

i. Maintain regular consultation with your supervisor(s), at least once a month
ii. Submit quarterly reports on time, through your supervisors, CoD, Dean and to the Director of Postgraduate Studies;
iii. Ensure quality work all through;

iv. Present your research findings at 2-3 seminars/conferences prior to thesis examination.
v. Publish one article from your research findings in a refereed journal prior to thesis examination.

For any further clarification please contact the undersigned.

Yours Sincerely,



Prof. Geoffrey Muchiri, PhD
Director, Postgraduate Studies
GM/PIM



Copy to:
1. Registrar – Academics and Students Affairs
2. Dean, School of Computing and Information Technology

Appendix 5: NACOSTI Research License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **258983**

Date of Issue: **22/November/2023**

RESEARCH LICENSE



This is to Certify that Mr.. CHARLES MUNYAO MUIA of Murang'a University of Technology, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Muranga on the topic: AN IMPROVED TRANSFORMER-BASED MODEL FOR TEXT SUMMARIZATION OF NEWS ARTICLES for the period ending : 22/November/2024.

License No: **NACOSTI/P/23/31530**

258983

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions

Appendix 6: Publication

The following paper has been published from this thesis:

[1] Muia, C. M., Oirere, A. M., & Ndung'u, R. N. (2024). A Comparative Study of Transformer-based Models for Text Summarization of News Articles. *International Journal of Advanced Trends in Computer Science and Engineering*, *13*(2), 37–43. https://doi.org/10.30534/ijatcse/2024/011322024