Unveiling Fake News with an Attention-Based CNN Model Using Hybrid Features

Submitted By

Francis Rudra D Cruze

ID: 211-16-563

Supervised By

Mohammad Sarwar Hossain Mollah

Associate Professor and Head

Semester: Fall - 2024

Department of Computing and Information System

Faculty of Science and Information Technology

Daffodil International University

Daffodil Smart City (DSC), Birulia, Savar, Dhaka-1216

A thesis in fulfilment of the requirements for the degree of Bachelor of Science (B.Sc.)

Submission Date: 12th January 2025

Approval

This thesis titled "**Unveiling Fake News with an Attention-Based CNN Model Using Hybrid Features**," submitted by **Francis Rudra D Cruze**, ID: **211-16-563** to the Department of Computing & Information System (CIS), Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computing & Information System (CIS) and approved as to its style and contents. The presentation has been held on **12-01-2025**.

BOARD OF EXAMINERS

12.9.25

Mohammad Sarwar Hossain Mollah — **Chairman**
Associate Professor and Head
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

12.01.25

Md. Nasimul Kader — **Internal Examiner**
Assistant Professor
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

12.01.25

Md. Mehedi Hassan — **Internal Examiner**
Lecturer (Senior Scale)
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

12.01.2025

Dr. Muhammad Shahin Uddin — **External Examiner**
Professor
Department of ICT
Mawlana Bhashani Science and Technology University

Declaration

This thesis entitled: **"Unveiling Fake News with an Attention-Based CNN Model Using Hybrid Features"** is submitted to Daffodil International University, Dhaka, Bangladesh in partial fulfillment of the requirements for the degree of Bachelor of Science (B. Sc.) (**Major AI in IoT**) in Computing and Information System (CIS) is a recent novelty work, supervised by **Mohammad Sarwar Hossain Mollah (Associate Professor and Head)**. I hereby declare that, the contents of this thesis and to the best of my knowledge have not been submitted by me for the award of any other degree in any other universities or institution. In addition, a special form of this work has been accepted and has been presented at the **"2024 2nd International Conference on Information and Communication Technology (ICICT 2024)"**; nonetheless, the material and work given here is original and has not been submitted to any other institution.

Supervised By

12.07.25

Mohammad Sarwar Hossain Mollah
Associate Professor and Head
Department of CIS
Daffodil International University

Submitted By

18.01.2025

Name: Francis Rudra D Cruze
ID: 211-16-563
Department of CIS
Daffodil International University

Certificate of Originality

I **Francis Rudra D Cruze**, confirm that I am the exclusive author of this thesis and that no portion has been published or submitted for publication.

To the best of my knowledge, my thesis does not infringe upon any copyright or violate any proprietary rights. All ideas, techniques, quotations, or other materials from external sources included in my thesis, whether published or not, are fully acknowledged in accordance with standard referencing practices.

I affirm that this is an authentic copy of my thesis, encompassing all final modifications sanctioned by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution. A special form of this work has been accepted and has been presented at the "**2024 2nd International Conference on Information and Communication Technology (ICICT 2024)**."

Signed:.....

Date:.....12.01.2025.....

Certificate of Authenticity

I herewith state that this thesis is solely an independent work and research of my own and has made use of no source, facility, or other aids except for those expressly acknowledged. It contains no material that was published or written previously by anyone else, except where due acknowledgement and proper citation are made within the text. This relates to any ideas, concepts, or information taken directly or indirectly from printed books, articles, and online sources, in addition to any translations of sources in languages other than English that I have made.

I also assert that this research will not be submitted for any academic institution. Further, I understand and agree that this thesis is subject to plagiarism detection software checks to verify its originality and conformity with academic integrity standards. I am fully aware that any deviation from the norms of sound scientific practice and academic honesty will have severe consequences, including expulsion from the program.

Lastly, I certify that the digital copy of this thesis submitted to the university library is an exact and unaltered copy of the final officially approved manuscript.

Signed:.....

Date:.....12.01.2025.....

Dedication

To my beloved Father and Mother,

Every arduous undertaking needs not only personal work but also the counsel of people who occupy a significant position in our affections. I dedicate this modest effort to you, my dear and affectionate parents, whose love, support, encouragement, and unwavering prayers have facilitated my success and honour.

To my esteemed Teachers,

In addition to your industrious efforts, you have consistently served as a profound source of inspiration. Your advice, counsel, and blessings have been vital to my achievements. I am profoundly appreciative of your significant impact on my academic career.

With deepest gratitude,
Francis Rudra D Cruze

Acknowledgements

First of all, I am deeply appreciating the Almighty Creator for His continuous blessings and for helping me to finish my thesis. Over my academic career, his constant counsel and grace have been the inspiration.

I would want to sincerely thank you and great respect to my esteemed supervisor, **Mohammad Sarwar Hossain Mollah**, Associate Professor and Head, Department of Computing and Information System, Daffodil International University, for his great direction, ongoing support, and mentoring over my research activities. This thesis has been much shaped by his rigorous review, encouragement, and patience. My particular gratitude goes to **Md. Faruk Hosen** sir for his great insights and assistance during the course of this project. His knowledge and direction have been much appreciated in helping to produce this job successfully.

Over my study years, the academic members of the Department of Computing and Information System of Daffodil International University have been much appreciated for their insightful comments, inspiration, and encouragement. In addition, I would especially want to thank Daffodil International University for giving me the tools I need to properly conduct my studies and interesting intellectual surroundings. My future professional activities will benefit much from the skills and information I have gained throughout my time in the institution.

Finally, but not least, I am always appreciative of my family and friends for their constant love, support, and moral encouragement during this difficult but worthwhile road. Their belief of me has been a continual source of strength and inspiration.

Abstract

Recent explosion of false news on social media websites has created a monument challenge to the integrity of information in the digital age and therefore sophisticated detection mechanisms are needed. The pragmatic problem of the dissemination of misleading information calls for effective systems that can identify and counter the propagation of fake news. In this paper, we propose a new approach for the fake news detection based on hybrid embedddind techniques and deep learning architectures. We combine CountVectorizer and Global Vectors for Word Representation (GloVe) embeddings and take advantage of both statistical patterns and semantic correlations appeared in the texts. We enhance the comprehension and classification processes of complex textual content using Attention-Based Convolutional Neural Network (AttCNN) models with well-optimized hyper-parameters. We develop our methodology in a manner to be scalable and robust, which is verified via k-fold Cross Validation (CV) and grid search hyper-parameter optimization. It has been extensively delightful and turns out to be strong and versatile across assorted false news situations. Using an AttCNN integrated with our hybrid CV+GloVe embedding technique, we achieve state-of-the-art precision with respective final statistics of accuracy: 99.50%, F1: 99.50%, precision: 99.38% and recall: 99.62%. The proposed model is further tested using various validation settings to enhance its reliability for application in real-world scenarios for fake news detection.

Preface

This bachelor's thesis examines false news detection via an attention-based convolutional neural network with hybrid feature integration, with the objective of improving detection accuracy through sophisticated deep learning methodologies.

This thesis has five chapters, summarized as follows:

Chapter 1: Introduction
Examines issues in detecting misinformation, the impetus for study, aims, and importance within the contemporary digital environment. The chapter delineates the research scope and anticipated contributions.

Chapter 2: Literature Review
Analyzes current studies on false news identification, encompassing conventional methods, deep learning strategies, and cutting-edge methodologies. Evaluates existing constraints and identifies research deficiencies.

Chapter 3: Methodology
Elucidates the implementation strategy, encompassing dataset characterization, preprocessing procedures, hybrid feature extraction methods (TF-IDF, CountVectorizer, Word2Vec, GloVe), and attention-driven CNN architecture.

Chapter 4: Results and Analysis
Provides experimental findings, performance indicators, and a comparative assessment with current methodologies. Comprises cross-validation outcomes and model optimization results.

Chapter 5: Conclusion
Summarizes major findings and research contributions, emphasizing the efficacy of the suggested methodology.

List of Publications

Conference Paper

[1] **D Cruze, F. R.**, Hosen, M. F., Ali, M. S., Hossain Mollah, M. S., Uddin, M. S., & Morshed, M. (2024). Unveiling Fake News with an Attention-Based CNN Model Using Hybrid Features. 2nd International Conference on Information and Com- Munication Technology (ICICT 2024).

[2] Wasima, J., Hosen, M. F., **D Cruze, F. R.**, Kader, M. N., Hossain Mollah, M. S., & Uddin, M. S. (2024). Integrated Bioinformatics and Machine Learning Analysis Uncovers Key Pathways and Therapeutic Targets for Hypertension and Chronic Kidney Disease. 27nd International Conference on Computer and Information Technology (ICCIT 2024).

Contents

List of Tables

List of Figures

Acronyms

CNN Convolutional Neural Network

RNN Recurrent Neural Network

CV Cross Validation

TF-IDF Term Frequency - Inverse Document Frequency

GloVe Global Vectors for Word Representation

NLP Natural Language Processing

RF Random Forest

XGBoost eXtreme Gradient Boosting

KNN K-Nearest Neighbor

AUC Area Under the ROC Curve

SVM Support Vector Machines

LSTM Long Short-Term Memory

RNN Recurrent Neural Networks

NN Neural Network

BiLSTM Bidirectional Long Short-Term Memory (LSTM)

CBOW Continuous Bag of Words

AttCNN Attention-Based Convolutional Neural Network

HA Hierarchical Attention

MHA Multi-Head Attention

LR Logistic Regression

NB Naive Bayes

DT Decision Tree

W2V Word2Vec

CBOW Continuous Bag of Words

IDF Inverse Document Frequency

TF Term Frequency

Chapter 1

Introduction

1.1 General Background

The digital revolution has changed the way knowledge is produced, disseminated, and consumed in the modern world. The digital world has become the primary means of news consumption and information distribution, not surprising since for 2024, over 5.52 billion internet users world-wide to the equivalent of 65.7% of the global population [3]. Perhaps, most clearly seen in social media, an estimated 72% of adults receive most of their news from social media platforms including Facebook, Twitter, and Instagram [4].

Over the course of time, how we disseminate information has changed almost entirely, at speeds and depths never before possible, powered by a highly developed technical infrastructure and algorithmically based content delivery systems. But this transformation has also posed new and formidable challenges to info integrity. Misinformation in the digital ecosystem is spreading six times faster than the real news [5], according to recent studies that show how fake information is rampant in social media.

The emergence of social media platforms like Facebook and Twitter has di-

minished the authority and control that traditional media outlets had over news delivery, enabling users to tailor their news consumption experience. While this democratization of information access has enabled the rapid spread of misinformation, it has also created challenges when it comes to distinguishing reliable sources from less credible ones. 66% of U.S. consumers believe that 76% and more of news social media post is biased, indicating a serious trust gap in traditional media outlets [1].

(a) Jews were dancing while fire rages on Temple Mount.

(b) Russian Air Forces fighting against the Wagner troop.

(c) Kumar expressing support for Prime Minister Narendra Modi ahead of the Elections.

(d) In Chennai, a person's photo gets displayed on a signboard if they violate traffic rules.

Figure 1.1: Instances of misinformation in multimodal contexts. The source for subgraph (a) is: Source a. The sources for subgraphs (b–d) are: Source b-d, accessed on 5 July 2023.

Fake news has become a hot button issue because it misleads the public and effects political discourse and social behavior. Fake news refers to intentionally misleading or false information that appears to be real news and includes different kinds of misinformation, such as hoaxes and propaganda. For instance, according to recent statistics, 60% of the considered respondents around the world claim

that news organizations regularly publish false stories, consequently leading to a drop in the public's trust in the media [6]. Additionally, 38.2% of U.S. news consumers admitted to unintentionally sharing fake news as they could not identify it, moreover, during the same period, 47% of U.S. adults encountered significant levels of false information regarding COVID-19 treatments [7]. Figure 1.1 presents samples of fabricated news that we gathered from the network, incorporating both textual and visual elements [8].

The growing sophistication of misinformation highlights the need for clear definitions and best practices for identifying and responding to these harmful messages. However, it is apparent that what worked in one time period does not work against the plethora of fake news that we currently deal with, and technology needs to be utilized to counter the effects of fake news on people.

1.2 Specific Background

1.2.1 Understanding Fake News

The surge of false news has emerged as a major worldwide issue in recent years, characterised by the fast dissemination of misinformation and disinformation via internet channels [9]. Fake news, referred to as "alternative facts" or "hoaxes," denotes falsified or modified material presented as authentic news content [10]. It may manifest in several ways, such as deceptive headlines, false narratives, altered photos or videos, and propaganda [11]. The motivations for producing and distributing false news are many, including financial profit via clickbait, political manipulation, and societal upheaval [12]. The ramifications of misinformation can be extensive, affecting public perception, moulding political dialogue, and perhaps

provoking violence or societal upheaval [13].

The digital era has intensified the proliferation of misinformation owing to the simplicity of information creation and dissemination online [14]. Social media platforms have emerged as prolific venues for the spread of misinformation, enabling individuals to rapidly share anything with their networks without requisite verification of its veracity [15]. This has resulted in the creation of "echo chambers" where users predominantly encounter material that reinforces their own opinions, rendering them more vulnerable to misinformation [16].

1.2.2 Current Detection Landscape

The identification of misinformation has emerged as a crucial domain of inquiry and advancement [17]. Initial methodologies depended on manual fact-checking and source validation, which are labour-intensive and challenging to scale. Recently, automated approaches employing machine learning and natural language processing techniques have become prominent [18]. These approaches can be classified into the following categories:

- **Content-based methods**: These approaches examine the content of news stories to discern language patterns, stylistic characteristics, and semantic discrepancies that may signify false news [19].
- **Contextual methods**: These approaches analyse the wider context of news dissemination, including the information source, the social network distributing the material, and the historical reliability of the news organisation [17].
- **Hybrid techniques**: These technique integrate content-based and context-based elements to enhance detection accuracy [20].

Deep learning models, especially CNN and Recurrent Neural Networks (RNN), have demonstrated encouraging outcomes in the identification of bogus news. CNNs excel in identifying local patterns and characteristics in text, but RNNs are adept at modelling long-range relationships and contextual information [21]. Attention mechanisms have significantly improved the efficacy of these models by enabling them to concentrate on the most pertinent segments of the input text.

Nevertheless these developments, obstacles persist in the domain of false news identification. This encompasses the necessity for real-time detection systems, the amalgamation of social media and network analysis, and the advancement of explainable AI models. Moreover, the accessibility of extensive and annotated datasets remains a critical impediment to the advancement of efficient fake news detection algorithms [22].

1.3 Research Motivation

The imperative for effective fake news identification systems is propelled by several intriguing aspects that highlight the necessity of this study. The swift advancement of disinformation strategies, together with the inadequacies of current detection systems, necessitates the development of more advanced solutions to this complex issue. Contemporary detection systems have substantial difficulties in processing and analysing the growing volume of digital input. Conventional machine learning methods frequently fail to discern the many subtleties and contextual connections inside bogus news material. This constraint becomes especially significant when the intricacy of misinformation persists in advancing. The advent of AI-generated material has radically transformed the domain of false news identification. Ex-

isting detection methods, mostly intended to recognise human-produced misinformation, frequently struggle to accommodate the language patterns and structural attributes of AI-generated material. This technical disparity is a considerable weakness in our information environment.

The computing efficiency of current detecting systems is a significant issue. Although many contemporary models attain elevated accuracy rates, they frequently necessitate considerable computational resources and processing duration, rendering real-time identification problematic in high-volume contexts. This constraint is especially troublesome in social media contexts when knowledge disseminates swiftly. The amalgamation of several feature extraction methodologies has demonstrated potential however remains insufficiently investigated. Hybrid methodologies that include several text representation techniques may enhance detection precision. Nevertheless, the majority of existing algorithms depend on singular feature extraction techniques, constraining their capacity to encompass several dimensions of false news material. Moreover, the attention mechanism in neural networks has shown considerable promise in enhancing text categorisation tasks. Attention-based models can improve feature selection relative to conventional methods. Nonetheless, the complete potential of attention processes in the detection of fake news, especially when integrated with hybrid feature extraction, has yet to be thoroughly investigated.

These considerations jointly highlight the pressing necessity for creative methodologies in false news detection, exemplified by the proposed attention-based CNN model using hybrid characteristics.

1.4 Research Objectives and Questions

1.4.1 Primary Objectives

The objective of this study is to create an improved system for detecting fake news through the use of attention-based CNN and a combination of feature extraction methods. The primary objective is to enhance detection accuracy, computational efficiency, and model robustness by integrating various text representation techniques and sophisticated deep learning architectures. Recent studies indicate that hybrid approaches can greatly improve detection capabilities, establishing a solid basis for this area of inquiry.

This study aims to design and implement a CNN architecture that utilizes attention mechanisms, specifically tailored for the detection of fake news. This architecture will be improved by creating a hybrid feature extraction framework that integrates various complementary techniques: Term Frequency - Inverse Document Frequency (TF-IDF), CountVectorizer, Word2Vec, and GloVe embeddings. The combination of these varied feature extraction techniques seeks to encompass multiple dimensions of textual content, ranging from statistical trends to semantic connections, thereby offering a more thorough depiction of the text.

Additionally, this study aims to apply and assess sophisticated optimization methods, particularly Grid Search CV and K-fold CV, to improve model performance and guarantee thorough validation. The optimization methods will be methodically implemented to refine the model parameters and assess its performance across various data scenarios. This study seeks to showcase the effectiveness of the proposed hybrid approach by conducting a thorough evaluation against current leading methods, utilizing standard performance metrics.

1.4.2 Research Questions

These research questions will systematically help us investigate whether the proposed approach works. First question is how does the combination of various feature extraction techniques affect the accuracy and robustness of fake news detection? This particular question deals with the orthogonal property of TF-IDF, CountVectorizer, Word2Vec, and GloVe embeddings having orthogonal properties with respect to each other.

The second research question explores how attention mechanisms are impacting the CNN architectures, mainly how it helps CNN architectures to find minor pattern from the fake news content. This study is important because attention mechanisms have performed well in several natural language processing tasks. The third question thus examines which is the best Opt Techniques that we can use in optimizing a model and how the model reliability is improved through the use of Grid Search CV and K-fold CV.

Finally, the last research question assesses the degree of improvement that the proposed hybrid approach achieves over some of the existing state-of-the-art methods. The model is not only evaluated against standard metrics of accuracy, precision, and computational efficiency; it also receives a qualitative usage assessment in order to measure the practical utility of the model in real-world applications.

1.5 Expected Contributions

This research aims to make several significant contributions to the field of fake news detection, addressing critical gaps in current methodologies and advancing the state-of-the-art in this crucial area. Firstly, We expect to create a new attention-

based CNN model that uses hybrid features in fake news detection. The goal is by evaluating this type of misinformation across different contexts and formats, this model should be able to perform relatively well. This works in leveraging the strengths of CNNs with those of attention mechanisms to achieve a more balanced focus on the most relevant (and least relevant) aspects of news content, resulting in a more accurate and nuanced capability for news detection.

Second, we seek to enhance hybrid feature extraction for fake news detection. And utilization of various feature types, such as textual, semantic, and contextual features, should give us insight into which feature combinations have the potential to improve detection performance. In future studies this contribution may open avenues for more powerful and complete feature engineering approaches.

Thirdly, we hope that through the training of our work, we will target the functionality of attention mechanisms with respect to CNN fake news detection. Through systematic analysis of the effect of attention layers on model performance, we hope to gain insights into the best way to utilize these mechanisms in this space. This may guide architectural choices for future fake news detection systems.

In addition, we anticipate that our study will provide a comparative evaluation of the new model with current state-of-the-art techniques. Hopefully, this benchmarking exercise will help open up our understanding of the strengths and potential weaknesses of this approach, and also contribute to the wider dialogue on what works best in the fight against disinformation.

Finally, we seek to discuss practical implications for deploying state-of-the-art fake news detection systems in the real world. Through analysis of scalability and performance implications, our work should provide insights about the practical drawbacks of complex deep learning models in high-throughput, low-latency sce-

narios, such as social media channels. We hope to contribute to the field of fake news detection so that we can not only provide theoretical insights into the detection of false information but also practical tools in the fight against misinformation in the digital age.

1.6 Applications

As the spread of disinformation is increasingly becoming a crucial challenge in today's digital-driven era, the need for developing fake news detection systems is rising across all industries. Such systems are crucial for fighting misinformation and preserving the integrity of information in our interconnected world.

1.6.1 Integration with social media platforms

Social media platforms are one of the most important application areas for fake news detection systems. When developed, these platforms can put in place algorithms to help detect and flag misleading content before it spreads widely. Intervention in real time aids in preventing misinformation that can travel to millions of users in minutes from going viral. It allows online platforms to keep content quality while shielding users from harmful misinformation.

1.6.2 Implementation at News Organizations

Integrating fake news detection systems in editorial workflows offers great value to organizations and media outlets. In fact, these tools help journalists and editors in the process of fact-checking, as they can verify the sources of the information and look for inconsistencies quickly. The integration helps ensure that news organiza-

tions maintain credibility and mitigate the risk of order news may make it to print. It also helps to uphold high standards of journalism in a time of near-insatiable news speed.

1.6.3 Public Health Communication

Fake news detection systems are critical for the public health sector to ensure the integrity of health information. These systems work during health crises to stem the spread of harmful medical misinformation that can pose a public health and safety threat. These tools are used by health care providers and organizations to track and challenge incorrect health claims, helping to ensure that accurate medical information is shared with the general public. This use has been especially useful during worldwide health crises.

1.6.4 Educational Integration

Fake news detection systems are used by educational institutions to supplement media literacy programmes and to improve school children's critical thinking skills. If teachers can show students how this can be done and integrate this into their courses, they will serve an important function where students become more critical users of what they see on the Internet. From the classroom to the workplace and the concert hall, the library, the museum, the meeting, the humanitarian mission and the public square, this application encourages a more knowledgeable and critically aware society.

1.6.5 Corporate Brand Protection

Fake news detection systems find another important application area in the corporate sector. Businesses use these tools to safeguard their brand reputation by detecting and reacting to misleading claims about their goods or services, or their corporate business activities. This application supports companies to keep their customers trust and market integrity intact while potentiating to reduce or significantly diminish the impact of potential derogatory misinformation campaign orchestrated by malicious actor. It also enhances proactive reputation management in a rapidly changing digital marketplace.

1.6.6 Political Speech and Defense of Electoral Process

Fake news detection systems are used by political organizations and electoral bodies to maintain the integrity of the democratic process. These tools assist in monitoring political discourse, identifying potential disinformation campaigns, and ensuring voters have access to accurate information when making electoral choices. The implementation ensuring that political content is sourced transparently and performed impartially reduces the scope of intentional proliferations of political content and helps in preventing foreign intervention attempts.

1.6.7 Consumer Applications

Simplified applications of fake news detection technology for individual consumers are available as browser extensions and mobile applications. These tools enable users to check facts on their own and decide for themselves whether the content they see online is accurate or not. Such applications act as intermediaries between

complex anti-fake news technologies and average internet users by offering intuitive interfaces for fake news detection, thus encouraging digital literacy and informed decision-making.

1.7 Scope and Limitations

This research is bound by certain technical and methodological parameters that dictate its merits and constraints. In order to limit the scope of experiments and model tuning, the proposed attention-based CNN model is implemented and evaluated with hybrid features in a pre-existing computational environment.

Technical Boundary: Since the research is specifically devoted toward creating and analyzing one deep learning architecture (attention-based CNN) for fake news detection. Although this method looks promising, it might not be all potential deep learning solutions to the issue. The study is limited by the ability of the selected model architecture and by the hybrid feature extraction approaches used.

For both training and evaluation, the study uses a dataset from Kaggle (123.81 MB). Though relatively large, this dataset possibly does not encompass the wide variety of fake news types and formats found on all digital territories. This selection of this dataset could introduce characteristics and potential biases that may limit the model performance and generalizability. Hardware constraints: All experimentation is carried out on a Mac Mini M2 (16GB RAM and 256GB storage) This is capable as is, but it may limit the size of experiments, models, and datasets you can efficiently process. Resizing or larger demonstrations may not be possible in local hardware, so look for solutions in your cloud.

This could affect its applicability to cases of fake news disseminated in lan-

guages or cultural contexts other than the predominant focus in the study. Misinformation works differently in different linguocultural spaces, which may impact model performance across global scenarios. Limited environment setup: The development is based on Jupyter Notebook and PyCharm, which is great for exploratory work but could be less suitable for production, as the code may have to be refactored. This research is not focused on production level implementation or large-scale, real-time processing.

The training data and feature extraction methods are also mono-lingual, which constrains the scope of the research. By restricting the model to English-lingual text, its capacity to detect fake news in other languages or cross-lingual setting is limited. This is especially important as misinformation knows no linguistic boundaries, and detection systems that operate across languages are in high demand.

1.8 Reader Background

This thesis is primarily aimed at academics, practitioners, and graduate students in computer science, artificial intelligence, and natural language processing. Readers should have a fundamental comprehension of deep learning principles, especially CNNs and attention processes, as these are integral to the proposed model.

A fundamental understanding of Natural Language Processing (NLP) approaches and proficiency in Python programming will aid in grasping the strategies utilised in this work. The multidisciplinary nature of false news identification suggests that ideas from communication studies, psychology, and linguistics might augment comprehension.

This project aims to build sophisticated false news detection algorithms for

social media platforms, news aggregators, and content verification services. The results may operate as a reference for academics investigating hybrid feature extraction techniques and attention-based models in diverse fields outside false news detection. This study seeks to enhance the efficacy of disinformation mitigation measures in digital contexts.

Recent research indicate that the capacity to identify false news differs markedly among demographic groups, shaped by variables such as age, political affiliation, and educational attainment [23]. This research addresses both technical audiences and seeks to educate policymakers and educators on the significance of media literacy in addressing disinformation. By enhancing comprehension of fake news dynamics and detection techniques, we want to contribute to a better informed society adept at navigating the intricacies of digital information.

To effectively leverage this research, readers need possess a fundamental understanding of:

- Fundamentals of machine learning and designs of neural networks
- Python programming and popular deep learning libraries
- Methods for text preparation and feature extraction
- Fundamental statistics and performance metrics in machine learning

1.9 Summary

In this chapter, we have established the fundamental structure for exploring fake news detection using attention-based CNN and hybrid feature extraction tech-

niques. This chapter began with an analysis of the landscape of digital information, highlighting the major issues brought forth by the emergence of false news in modern culture. The discussion progressed to identifying existing detection methods through their features and limitations followed by exploration of some potential opportunities to study in hybrid extraction and deep learning approaches.

They found there is a lack of existing systems and solutions to perform the detection tasks and provided references to fill in the detected gaps, especially addressing the integration of different feature extraction techniques and attention mechanisms within CNN-based architectures. The limits of the study have been clearly established, including both technological restrictions and aspects related to the dataset that would define the terms of the research.

1.10 Thesis Organization

The following chapters of the thesis are organized such that Chapter 2 presents a literature review in the context of the area Chapter 3 presents the proposed method Chapter 4 presents details on implementation comparative analysis against current state-of-the-art methods. We show this this evolution illustrates how our integrated methodology improves capabilities to identify false information.

Chapter 2

Background and Related Work

2.1 Evolution of Misinformation in Digital Media

The spread and impact of misinformation over digital media has prompted discussion in the popular press— but it is an issue that is not new. It discusses the historical context of misinformation, the impact of social media on news dissemination, algorithmic content distribution challenges, and the current landscape in which fake news proliferates.

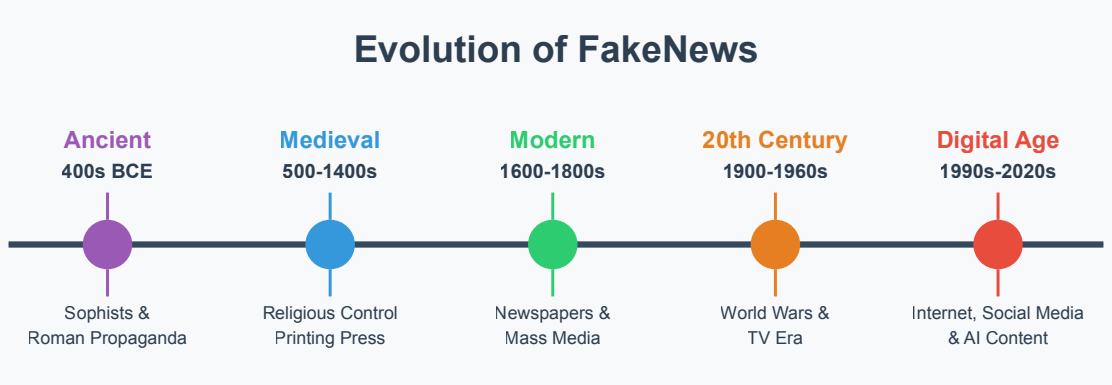

Figure 2.1: Timeline depicting the growth of misinformation from antiquity to the digital era

2.1.1 Historical Context of Misinformation Spread

Misinformation is not a new event. Its roots go back to antiquity with notable cases being reported in the history. In Ancient Rome, Octavian (later known as Emperor Augustus) used a particularly cheeky strategy of propaganda to fight his enemy Mark Antony spreading rumors and untruths about him in short phrases stamped on coins — similar to today's tweets. Through this example, it is emphasize that deception has been used well before the advent of modern computing both as a tool for political gain and to manipulate society [24]. The Figure 2.1 visualizing a historical timeline tracing the spread of false information from antiquity to the modern era.

The year was 1439 when the Gutenberg printing machine entered into the scene, any master-strokes to run down the operations of disseminating knowledge and information, including that of misinformation. This technology made it possible to spread accurate and false information in seconds, on a scale never seen before. This technique enabled one of the earliest major false news scandals, however, known as the Great Moon Hoax of 1835. On the website for the New York Sun, an array of stories claimed to show the discovery of life on the moon with forged photos.

2.1.2 Impact of Social Media on News Dissemination

The introduction of social media in the dissemination of news has revolutionized the process, improving reach and those speed in which information is distributed, including disinformation. News dissemination through social media platforms such as Facebook, Twitter, and Instagram has become a powerful tool for both indi-

viduals and organizations to share news articles instantly with a global audience [25]. The major impacts of social media on the news distribution include:

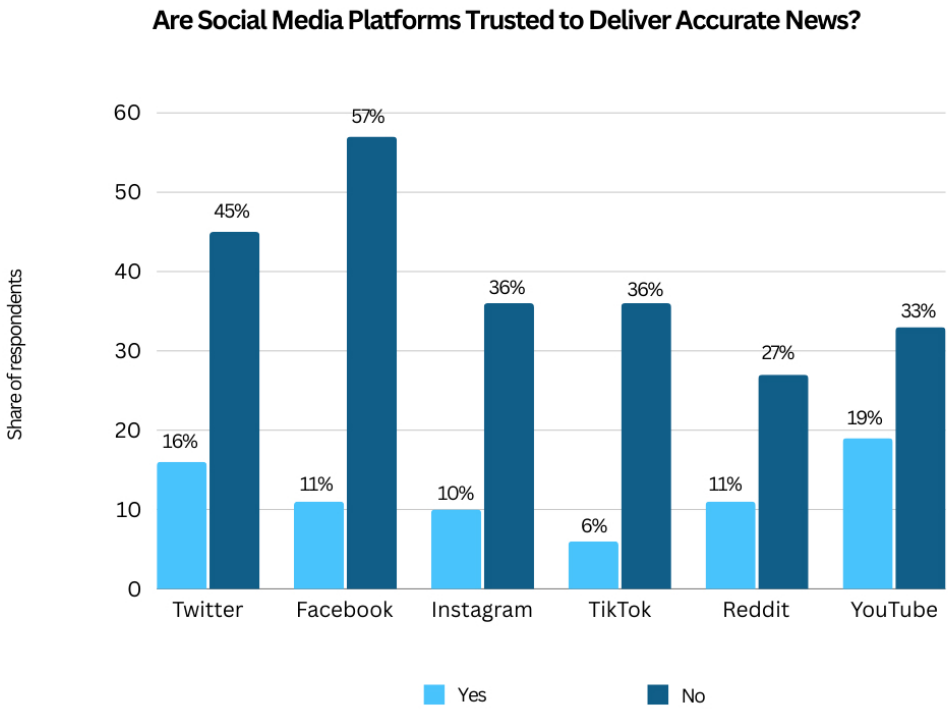

Figure 2.2: Survey findings on the trustworthiness of social media platforms, indicating the percentage of respondents that affirm ("Yes") or deny ("No") their confidence in these platforms to provide reliable news. Facebook exhibits the highest level of distrust at 57%, whilst TikTok demonstrates the lowest trust rating at 6%. Source of data: Security.org & Redline [1]

- **Improved Speed and Range**: News can now spread around the world in seconds and reach people beyond geographic boundaries.
- **Democratization of Information**: Social media has enabled citizen journalism by allowing anyone to report and share news in real-time.
- **Real-time Updates**: Social media platforms have made it possible to share news instantly, allowing for real-time updates and breaking news stories.

- **Customized News Aggregation**: Algorithmic tailoring of content on social media outlets leads to the personalization of news content aligned with people' interests, possibly creating echo chambers.

2.1.3 Algorithmic Content Distribution Challenges

The algorithmic dissemination of material on social media platforms has created new challenges of combating disinformation. These algorithms, which have been designed to maximize user engagement, often promote content that is either sensationalist or aligned with users' existing beliefs, increasing the spread of false information [26]. Main challenges include:

- **Filter Bubbles**: Algorithms can create echo chambers that serve users almost nothing but content reaffirming their original thoughts.
- **Faster Spread of Misinformation**: The rate of algorithmic spread may outpace fact-checking efforts [27].
- **Exploitative Use by Malicious Actors**: Bad actors can manipulate algorithms to prioritize false or misleading information.

2.1.4 Current Landscape of Fake News Propagation

The present environment of misinformation dissemination is marked by its intricacy and the numerous elements facilitating its proliferation [1]. Recent figures underscore the prevalence of this issue:

- 66% of consumers in the United States say that 76% or more of news shared via social media is biased.

- 60% of people said news media often provide false information around the world.
- 82% of Argentinians reported frequent exposure to intentional disinformation narratives, compared to lower numbers in Germany, Japan and South Korea.
- Some involvement or perception of past influence (60% of U.S. journalists show great concern about limiting press freedom).
- 94% of journalists consider fake news a serious problem.
- 38.2% of U.S. news readers on social media unwittingly spread false information.
- Widely varying proportions based on respondent political leanings did find all of them proving void of any news information on the Twitter app as accurate, but across all respondents combined, only 16% would agree that would be the case.
- 66% of the bots that discussed COVID-19 spread falsehoods.
- 47% true-level of U.S. adults gave considerable volume of COVID-19 fake news.
- Video deepfakes increased by 300% and audio deepfakes by 800% from 2022 to 2023.
- In 2023, there were estimated to be around 500,000 deepfakes shared on social media.

2.2 Theoretical Framework of Fake News Detection

2.2.1 Conceptual Understanding and Characteristics

Fake news is a multifaceted problem necessitating a comprehensive strategy for comprehension and detection. Fake news is fundamentally characterized as inaccurate or deceptive material conveyed in a journalistic context with the purpose to mislead [28]. There are three fundamental pillars for this definition:

- **Low factual accuracy**: Includes false or misleading content.
- **Intent to deceive**: The creator has a purpose to influence or misguide the audience.
- **Journalistic format**: The content is presented as real news.

2.2.2 Types and Taxonomies of Misinformation

A range of taxonomies has been proposed to categorize the different kinds of disinformation [29]:

- **Disinformation**: False information spread deliberately to mislead or cause harm.
- **Misinformation**: Wrong information spread without intent to deceive.
- **Mal-information**: Accurate information used out of context to cause harm.

Wardle's taxonomy describes seven classes of misinformation and disinformation [30], illustrated in Figure 2.3:

1. Satire or parody
2. Erroneous association
3. Deceptive content
4. Erroneous context
5. Counterfeit material
6. Altered content
7. Falsified content

A different approach is to differentiate misinformation according to two axes [31]:

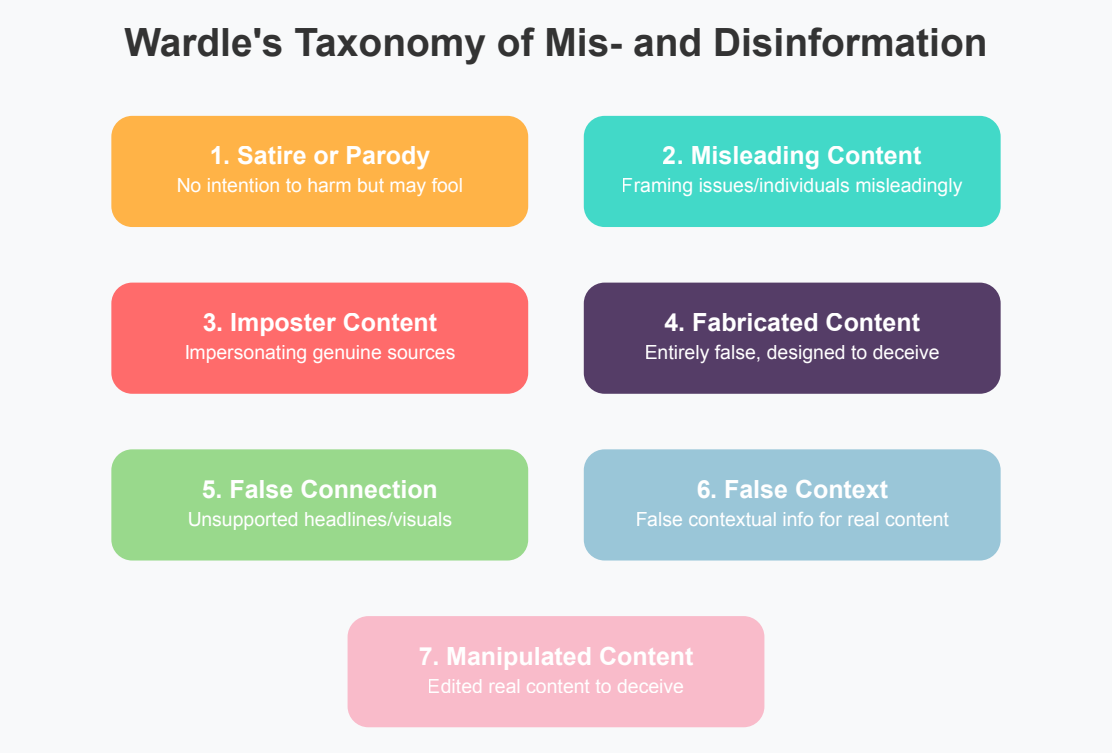

Figure 2.3: Wardle's taxonomy classifies mis- and disinformation into seven categories, ranging from satire/parody to manipulated content which is represend in color-coded. The diagram provides concise definitions for each type, distinguishing between content created to deceive and content that may unintentionally mislead.

- **Falsity level**: High (fabricated content) to low (misused content).
- **Evidence type**: Statistical and/or narrative.

2.2.3 Impact Across Various Domains

The spread of misinformation has wide-reaching consequences in every sector:

1. **Political**: Influencing the attitudes of the masses electorally.
2. **Social**: Erodes trust in institutions and media.
3. **Economic**: Driving market and consumer action.

4. **Healthcare**: Leaking false information on medical interventions and public health.

2.2.4 Detection Challenges and Considerations

There are several distinct challenges in identifying false information:

1. **Timeliness:** News content is time-critical and requires quick detection algorithms [32].
2. **Changing strategies**: Misinformation providers are constantly changing their tactics.
3. **Multimodal content**: Disinformation may include text, graphics, and videos.
4. **Battling misinformation**: Spotting satire and parody requires nuanced knowledge.

In order to address the mentioned problems, scientists have proposed a framework for the detection of fake news, consisting of four key areas [33]:

- **Knowledge-based:** Checking news content against pools of factual knowledge.
- **Style-oriented:** Looks at writing style and trends in language.
- **Propagation-based:** Studying dispersion patterns within social networks.
- **Credibility-oriented:** Assessing the trustworthiness of sources, headlines, and user reviews.

2.3 Machine Learning Approaches (Conventional)

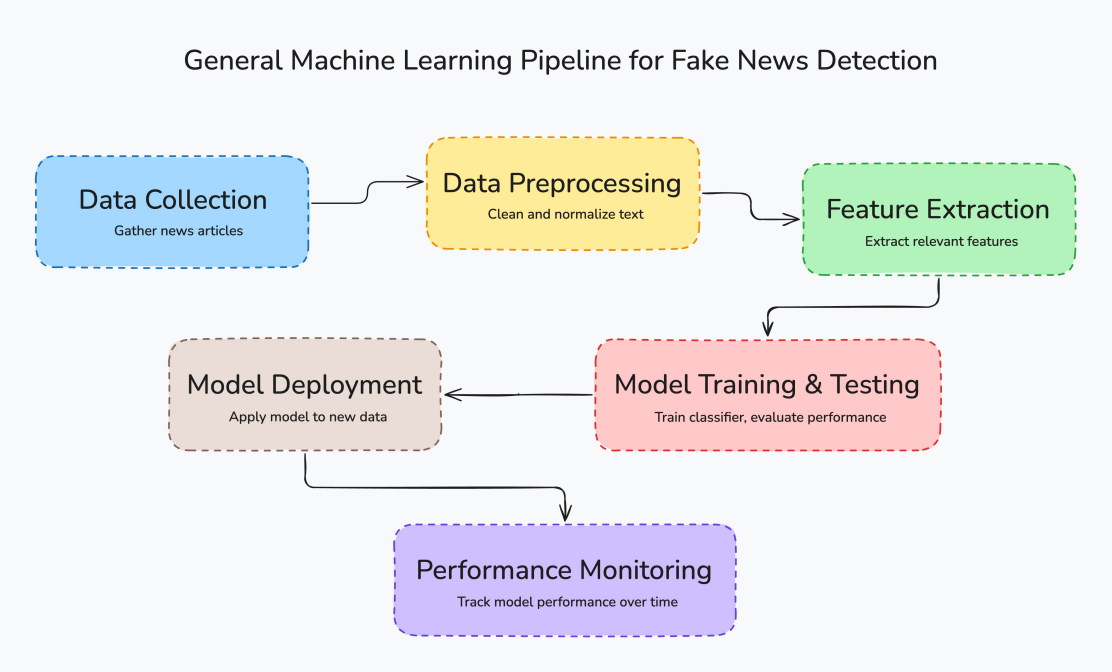

Figure 2.4: An overview of the machine learning process for detecting false news, highlighting the essential phases from data collection to model deployment and performance evaluation. The figure illustrates the data flow throughout the pipeline, with each stage distinctly color-coded for clarity and accompanied by a concise description of its function.

Feature engineering is an essential phase in creating efficient machine learning models for the identification of false news. It entails collecting pertinent features from the raw text data to represent news stories in a manner appropriate for machine learning algorithms. Prevalent methodologies including tokenization, elimination of stop words, and lemmatization [34, 35]. These procedures facilitate noise reduction and the extraction of fundamental semantic information from the text. The overall flow of the machine learning pipeline for the false news detection task is captured in Figure 2.4, which highlights the import steps that ranges from data

collection phase to model deployment and performance evaluation. This figure illustrates the flow of data through the pipeline, from the collection of news stories, to preprocessing, feature extraction, model training and testing, deployment, and ongoing performance monitoring. The figure 2.4 explains the relationships between the steps, it provides a simple and clear representation of a complex process behind machine learning based false news detection systems.

2.3.1 Traditional Classification Methods

Numerous conventional machine learning techniques have demonstrated encouraging outcomes in the detection of false news:

2.3.1.1 Random Forest (RF)

RF is an ensemble learning technique that generates several decision trees and amalgamates their outputs to provide predictions. It has shown efficient in detecting false information owing to its capacity to manage high-dimensional data and elucidate intricate correlations among attributes [36].

2.3.1.2 eXtreme Gradient Boosting (XGBoost)

XGBoost is an enhanced version of gradient boosting that has become widely used in the domain of false news identification. It provides superior performance and can manage skewed datasets efficiently [37].

2.3.1.3 K-Nearest Neighbor (KNN)

KNN is a straightforward yet efficient method for the classification of false news. It operates by contrasting the attributes of a specific news story with those of

annotated instances in the training dataset [38].

2.3.2 Model optimization techniques

To guarantee the reliability and generalizability of false news detection algorithms, CV approaches are frequently utilized:

2.3.2.1 K-fold CV

K-fold CV entails partitioning the dataset into K subgroups, training the model on K-1 subsets, and verifying it on the remaining subset. This procedure is executed K times, with each subset utilized as the validation set once [39].

2.3.2.2 Grid Search Optimization

Grid search CV is employed to identify the ideal hyperparameters for machine learning models. It methodically evaluates several combinations of parameter adjustments, employing cross-validation to ascertain which adjustment yields optimal performance [40].

2.3.3 Evaluation frameworks and metrics

The efficacy of algorithms for detecting false news is generally evaluated using a range of metrics:

1. **Accuracy**: The ratio of correct predictions (including true positives and true negatives) to the total number of instances analyzed.

2. **Precision**: The proportion of accurately predicted positive observations to the total expected positives, reflecting the model's capacity to prevent the

misclassification of authentic news as fraudulent.

3. **Recall**: The proportion of accurately predicted positive instances to the total actual positives, assessing the model's capacity to identify all fraudulent news stories.

4. **F1 Score**: The harmonic mean of accuracy and recall, yielding a singular metric that equilibrates both measures.

5. **Specificity**: The ratio of true negatives (authentic news) accurately recognized, enhancing memory in evaluating the model's discriminatory capability.

6. **Area Under the ROC Curve (AUC)**: A metric assessing the model's proficiency in differentiating between classes at different classification criteria.

These metrics assist us in comparing various methodologies and evaluating the overall efficacy of false news detection systems.

2.3.4 Limitations of conventional approaches

Traditional machine learning techniques for identifying fake news, no matter how effective, have significant limitations in the digital age. Applying methods based on classic classifiers including Support Vector Machines (SVM) and Random Forests, with challenges in scalability and flexibility, is common [41]. They may not sufficiently capture the complex and evolving nature of misinformation largely or at least when interacting with multimodal information that blends text, images, and video [42]. An important step in these above methodologies is feature engineering, which can be an arduous process that lacks generalizability to new types of

misinformation [33]. In addition, these models often rely on hand-crafted features that may become irrelevant as disinformation tactics evolve [43]. Cross-platform analysis is a challenge, as existing approaches do not track the propagation of misinformation across diverse social media platforms with unique characteristics [44]. These restrictions illustrate the need for more innovative, flexible approaches to effectively tackle the changing landscape of digital disinformation.

2.4 Deep Learning in Fake News Detection

2.4.0.1 Neural Network (NN) Fundamentals and Evolution

Deep learning approaches have revolutionized fake news detection by offering powerful models capable of capturing complex patterns in textual data. Neural networks, inspired by the human brain, consist of interconnected nodes organized in layers. These networks learn by adjusting the weights of connections between neurons through backpropagation, optimizing their performance on the given task. The application of deep learning to fake news detection has evolved from early shallow networks to deeper architectures, allowing for more complex feature extraction. Specialized architectures like CNNs and RNNs were adapted for text processing, followed by advanced models incorporating attention mechanisms and transfer learning. [45]

2.4.1 Key Architectures

The hierarchical framework and relationships across neural network designs illustrate the advancement of deep learning methodologies in the identification of false news. Figure 2.5 illustrates the mapping of several architectures, including CNNs

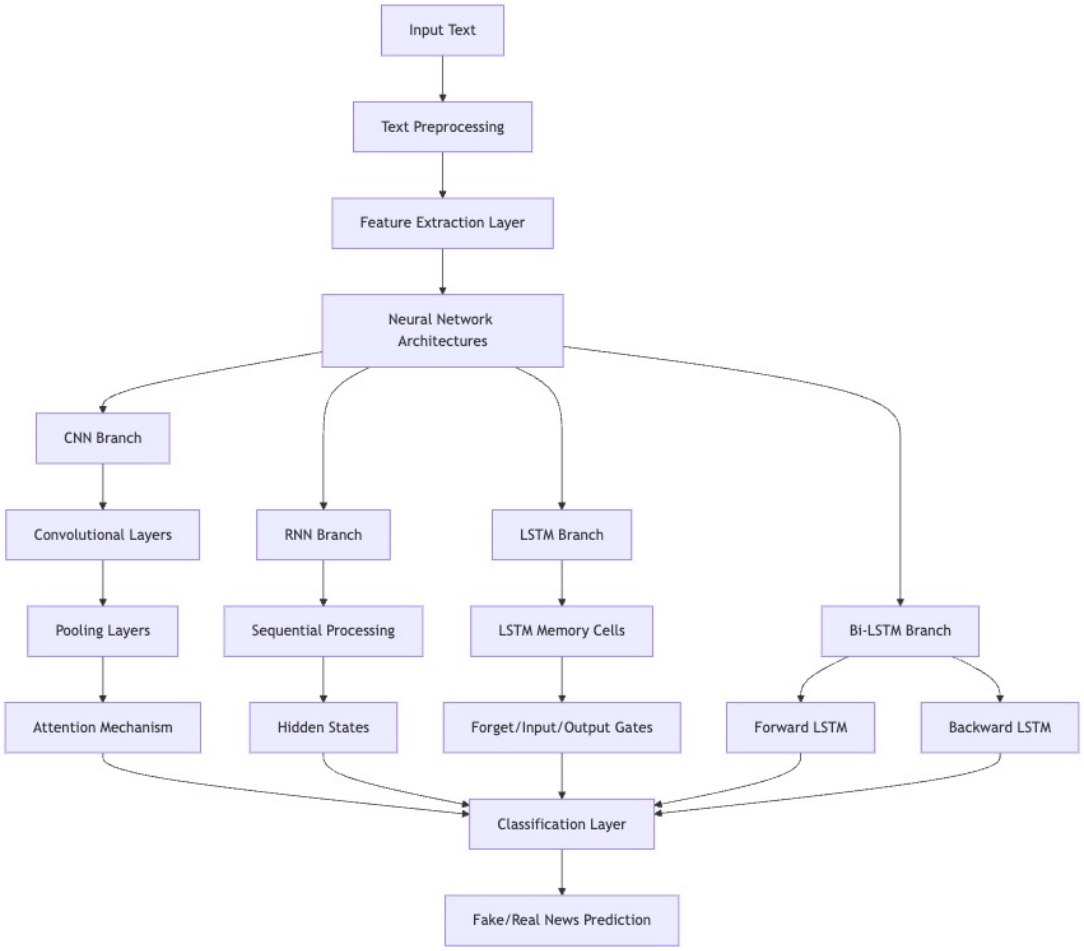

Figure 2.5: Deep learning architectures for the identification of fake news, demonstrating the hierarchical structure and interrelations across several neural networks (CNN, RNN, LSTM, BiLSTM). The graphic illustrates the transition from text input to feature extraction, numerous processing paths, and attention mechanisms to final classification, highlighting the complementary roles of different architectures in identifying misleading information.

and BiLSTMs, in their processing and analysis of news information from initial input to final categorization. Each pathway utilizes distinct strengths: CNNs identify local patterns, RNNs and LSTMs process sequential information, whilst BiLSTMs examine text in both directions. The attention mechanism unifies various architectures, improving feature concentration capacities. This architectural

variety illustrates the complex nature of false news detection, wherein several techniques enhance one another to address different facets of misleading information.

2.4.1.1 CNN

CNN, originally designed for image processing, have been successfully adapted for text classification tasks, including fake news detection. In text applications, words are represented as vectors (embeddings), and convolutional filters slide over the text, capturing local patterns. Pooling layers then reduce dimensionality and extract key features. CNNs excel at capturing local contextual information and are computationally efficient, making them a popular choice for initial deep learning approaches to fake news detection [46, 47].

2.4.1.2 RNN and LSTM

RNNs are designed to process sequential data, making them well-suited for text analysis. They maintain internal state, process words in sequence, and handle variable-length input. However, standard RNNs struggle with long-term dependencies due to the vanishing gradient problem. LSTMs, an advanced form of RNN, address this limitation through gating mechanisms (input, forget, and output gates). This allows LSTMs to capture long-term dependencies and better handle the vanishing gradient problem, leading to superior performance in many fake news detection tasks [47, 48].

2.4.1.3 Bidirectional LSTM (BiLSTM) Implementations

Bidirectional LSTM! (**Bidirectional LSTM!**)s process input sequences in both forward and backward directions, allowing the network to capture context from

both past and future states. This bidirectional processing often leads to improved performance in fake news detection tasks, as it provides a more comprehensive understanding of the textual context [48, 49].

2.4.1.4 Comparative Analysis of Architectures

A comparative analysis of these architectures reveals their strengths and limitations in the context of fake news detection. Table 1 provides an academic comparison of the key deep learning architectures used in this field.

Table 1: Comparison of Neural Network Architectures for Information Processing

Architecture	Key Features	Strengths	Limitations
CNN	Local pattern recognition, pooling layers	Captures local context, computationally efficient	Limited in capturing long-range dependencies
RNN	Sequential processing, internal memory	Processes sequential data, maintains context	Struggles with long sequences, vanishing gradients
LSTM	Gating mechanisms, long-term memory	Captures long-term dependencies, handles vanishing gradients	More complex, potentially slower training
BiLSTM	Bidirectional processing	Captures bidirectional context, often highest accuracy	Most computationally expensive

2.4.1.5 Future Directions

The field continues to evolve with:

- Integration of multiple architectures for enhanced performance

- Development of more sophisticated attention mechanisms
- Exploration of multi-modal approaches
- Focus on computational efficiency and scalability
- Investigation of cross-lingual capabilities

These advancements suggest a promising future for deep learning in combating misinformation and maintaining information integrity in the digital age.

2.5 Text Representation Methods

Text representation is an essential phase in natural language processing and machine learning projects that use textual input. This section examines many methodologies for representing text as numerical vectors, encompassing conventional procedures, contemporary embedding methods, and hybrid strategies.

2.5.1 Conventional Methods

2.5.1.1 TF-IDF

TF-IDF is a quantitative metric employed to assess the significance of a term in a document relative to a collection or corpus [50]. It integrates the term frequency inside a document with its scarcity throughout the whole corpus, attributing greater significance to phrases that are prevalent in a specific document yet seldom in the overall corpus [51].

2.5.1.2 CountVectorizer

Enumeration Vectorizer is an effective method that transforms a set of text documents into a matrix of token frequencies. Each document is represented as a vector, with each dimension corresponding to a phrase from the lexicon, and the value indicating the frequency of that term inside the text [52].

2.5.2 Contemporary Embedding Methods

2.5.2.1 Word2Vec Framework

Word2Vec, created by Google researchers, is a neural network-based method for acquiring word embeddings [53]. It employs two primary architectures: Continuous Bag of Words (CBOW) and Skip-gram. Word2Vec encapsulates semantic links among words, facilitating significant vector arithmetic [54].

2.5.2.2 GloVe Implementation

GloVe is an unsupervised learning approach that integrates the benefits of global matrix factorization with local context window techniques. GloVe generates word vectors by examining word co-occurrence data over the whole corpus [55].

2.5.3 Hybrid Feature Extraction

Hybrid methodologies integrate several text representation techniques to utilize their synergistic advantages. A research introduced a hybrid feature extraction technique that integrates TF-IDF with word embeddings to encompass both syntactic and semantic information. Hybrid approaches frequently surpass singular strategies by offering a more thorough representation of the text [56].

2.5.4 Comparative Performance Analysis

To illustrate the relative performance of different text representation methods, consider the following table summarizing their key features, strengths, and limitations:

Table 2: Comparative Analysis of Text Representation Methods in Natural Language Processing

Method	Key Features	Strengths	Limitations
Count Vectorizer	Tokenization, counting, and vectorization of words	Simplicity, interpretability, speed, and efficiency	Ignores semantic information, bias towards frequent words, lack of normalization
TF-IDF	Combines term frequency with inverse document frequency	Captures term importance, reduces impact of common words	Still lacks semantic understanding, high dimensionality
Word2Vec	Neural network-based word embeddings	Captures semantic relationships, allows vector arithmetic	Requires large training corpus, fixed-length vectors
GloVe	Global word-word co-occurrence statistics	Efficient training, performs well on analogy tasks	May struggle with rare words or domain-specific vocabulary

The choice of text representations depends on many factors, including the characteristics of the task, the size and properties of the dataset, and your computational resources. While modern methods of embedding usually outperform older approaches in terms of semantic relational encoding, a fusion of multiple methods will often lead to a better overall performance due to the additive improvement of multiple methodologies.

Overall, text representation is a constantly evolving area, and researchers are looking for better ways to represent the nuances of language in vectors. The grow-

ing sophistication of NLP applications highlights the importance of well-designed text representation methods for a range of tasks (e.g. sentiment analysis, text classification, information retrieval).

2.6 Attention Mechanisms in Neural Networks

Attention mechanisms have become a formidable instrument in neural networks, especially for problems involving bogus news identification. These strategies enable models to concentrate on the most pertinent aspects of the input data, enhancing performance and interpretability.

2.6.1 Theoretical foundations

Attention mechanisms draw inspiration from human cognitive processes, wherein we selectively concentrate on significant information while disregarding extraneous data. In neural networks, attention allocates significance weights to various components of the input, enabling the model to emphasize specific aspects [57].

2.6.2 Architectural variations

Numerous attention architectures have been suggested for the detection of false information.

- **Self-Attention**
  - Relates different positions within the same sequence
  - Particularly effective for capturing long-range dependencies
  - Enables parallel processing of sequence elements

- **MHA**
  - Projects inputs into multiple representation subspaces
  - Allows attention to focus on different aspects simultaneously
  - Enhances model's ability to capture complex relationships
- **Hierarchical Attention (HA)**
  - Processes text at multiple levels (word, sentence, document)
  - Enables capture of document structure
  - Improves interpretation of context

2.6.3 Integration with CNN frameworks

Attention processes may be efficiently incorporated with CNNs for the identification of bogus news. A research introduced an attention-based CNN model that integrates local pattern recognition with global context awareness, enhancing the network's capacity to detect nuanced indicators of disinformation [58]. The Figure 2.6 and 2.7 showed the clear architecture diagram of AttCNN.

2.6.4 Implications for performance

The use of attention processes has resulted in substantial enhancements in the accuracy of false news identification. A research utilizing attention-based deep learning models (CNN, LSTM, BiLSTM) on the LIAR dataset shown superior performance relative to conventional methods.

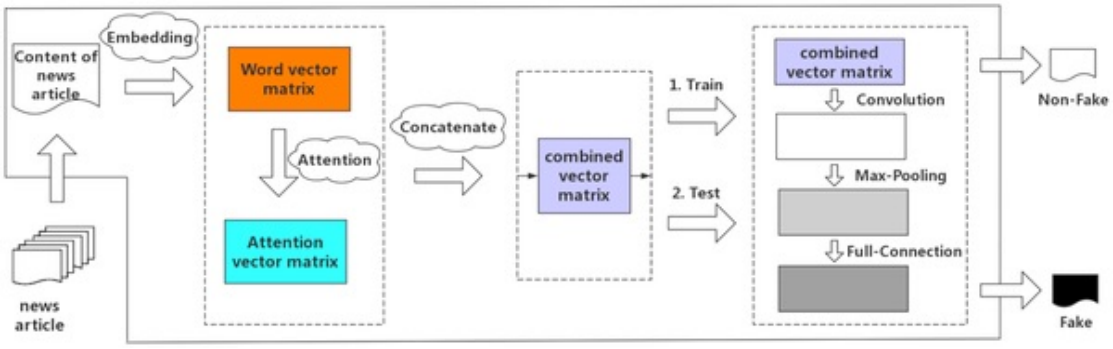

Figure 2.6: Pipeline design for the detection of misinformation, illustrating the conversion of news items via embedding, attention mechanisms, and neural network layers. The method combines word vectors with attention vectors to develop a thorough feature representation prior to the final categorization into authentic or fabricated news categories. [2]

2.6.5 Existing constraints and obstacles

Although attention mechanisms have significant potential, they encounter hurdles like heightened computational complexity and the necessity for bigger datasets for successful training. Furthermore, the interpretation of attention weights in relation to false news detection continues to be a prominent study focus.

In summary, attention mechanisms provide an effective method to augment false news detection models by enabling them to concentrate on the most important characteristics of news stories, hence enhancing overall accuracy.

2.7 Research Metrics

2.8 Research Gap Analysis

The analysis of recent literature in fake news detection reveals several critical research gaps requiring attention. While studies demonstrate progress with hybrid models and advanced embeddings, significant limitations persist in current

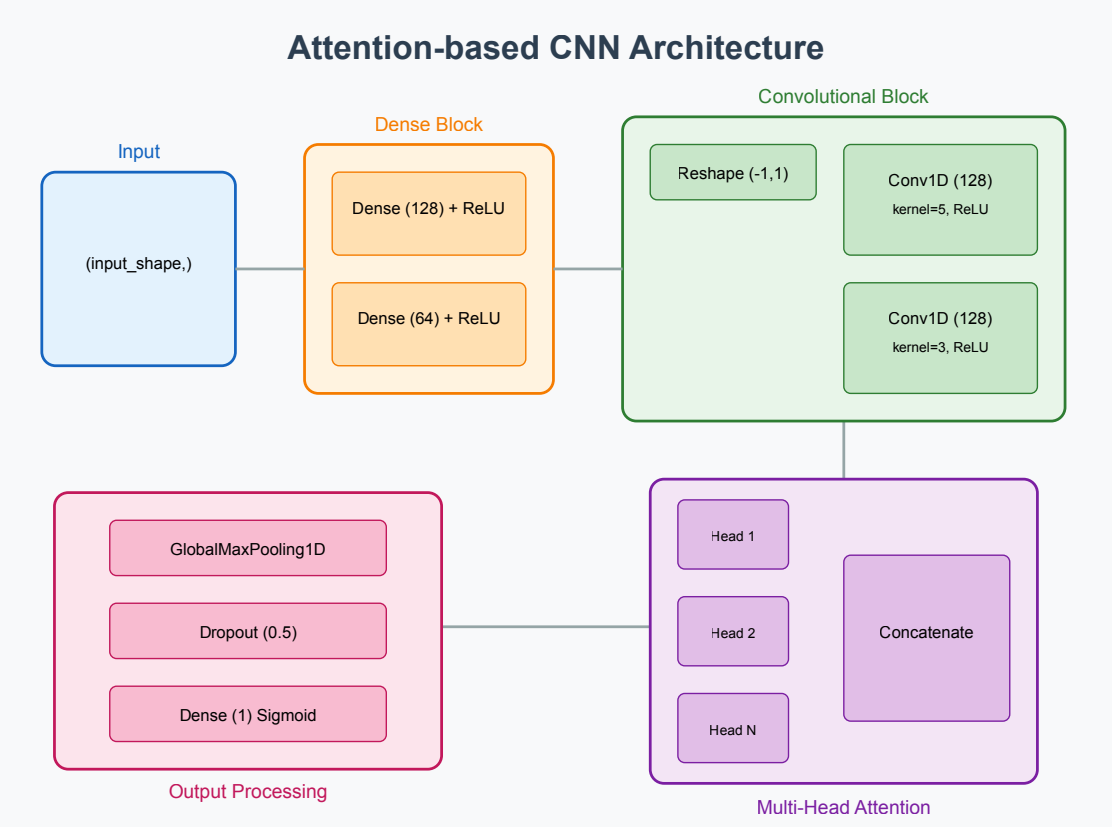

Figure 2.7: Architectural schematic of the AttCNN model illustrating the incorporation of dense layers, parallel convolutional branches, MHA mechanism, and output processing elements. The model integrates conventional CNN components with attention methods to facilitate efficient feature learning and classification. Various colors denote specific functional blocks: input (blue), dense (orange), convolution (green), attention (purple), and output processing (pink).

approaches [59, 60]. Key gaps include the lack of adaptive architectures capable of handling diverse fake news patterns, insufficient exploration of real-time detection capabilities, and limited cross-lingual validation. The emergence of AI-generated content, highlighted in [63], presents new challenges that current models are not fully equipped to address. Traditional embedding techniques, though effective, show limitations in capturing the full complexity of misinformation patterns. Dataset challenges, including imbalanced data and limited diversity, continue to

2.8. Research Gap Analysis

Table 3: Comparison of Studies on Fake News Detection Models

Author	Year	Key Findings	Model Used	Embedding Technique	Limitations
Prachi et al. [34]	2022	Developed a fake news detection system using ML/DL models like BERT and LSTM, achieving high accuracy.	Logistic Regression (LR), Decision Tree (DT), Naive Bayes (NB), SVM, LSTM, BERT	TF-IDF, BERT	Data dependency, overfitting, and language/context specificity.
Ouassil et al. [59]	2022	Proposed a hybrid CNN-BiLSTM model, achieving 97.74% accuracy using Word2Vec and GloVe embeddings.	CNN, BiLSTM	Word2Vec (CBOW, Skip-Gram), GloVe	Increased computational complexity; requires feature selection and dimensionality reduction.
Ahmed et al. [36]	2021	Highlighted effectiveness of supervised classifiers (e.g., SVM, NB) in fake news detection.	SVM, NB, LR, DT, KNN	TF-IDF, N-Gram, Bag of Words	Challenges with imbalanced datasets and need for labeled training data.
Altamimi [60]	2024	Combined FastText, FastText-Subword, and GloVe embeddings with CNN, achieving high metrics (e.g., 94.58% accuracy).	CNN	FastText, FastText-Subword, GloVe	Limited validation for generalization across languages and news genres.
Farha et al. [61]	2023	Highlighted need for models analyzing historical/current datasets to improve detection accuracy.	DT, NB, SVM	TF-IDF, N-Gram	Imbalanced datasets and ineffective feature representation leading to overfitting.
Ogunsuyi et al. [62]	2022	Proposed a KNN-Bayesian model with TF-IDF and Word2Vec, achieving better performance than standalone KNN.	KNN, NB	TF-IDF, Word2Vec	KNN unsuitable for real-time predictions; further training/testing required.
Aslan et al. [63]	2024	Demonstrated effectiveness of linear/ensemble models and introduced a ChatGPT-generated dataset.	Passive-aggressive classifiers, SVM, RF	null	Performance varies across datasets; complex models needed to handle AI-generated content.

affect model performance across studies. Furthermore, there's a notable absence of comprehensive validation frameworks that assess models' generalization ability,

robustness against adversarial attacks, and adaptation to evolving misinformation tactics. These gaps suggest the need for developing hybrid architectures with adaptive attention mechanisms, creating lightweight models for real-time detection, and establishing standardized validation methodologies that can effectively evaluate performance across multiple dimensions.

Chapter 3

Methodology

3.1 Research Design

3.1.1 Framework Overview

The proposed research offers an extensive framework for detecting false news via advanced natural language processing and deep learning methodologies. The framework is designed to tackle the issues of misinformation detection using a systematic method that integrates hybrid feature extraction with attention-based neural networks. Our methodology prioritizes the amalgamation of many text representation techniques to include both statistical and semantic attributes of news material, facilitating strong classification efficacy across a range of news items. The framework has four essential components: data preparation, feature extraction, classifier selection, and prediction validation, as seen in Figure 3.1.

3.1.2 System Architecture

During the preliminary stage, the data preprocessing module manages the cleaning, normalization, and standardization of raw text. The feature extraction module in-

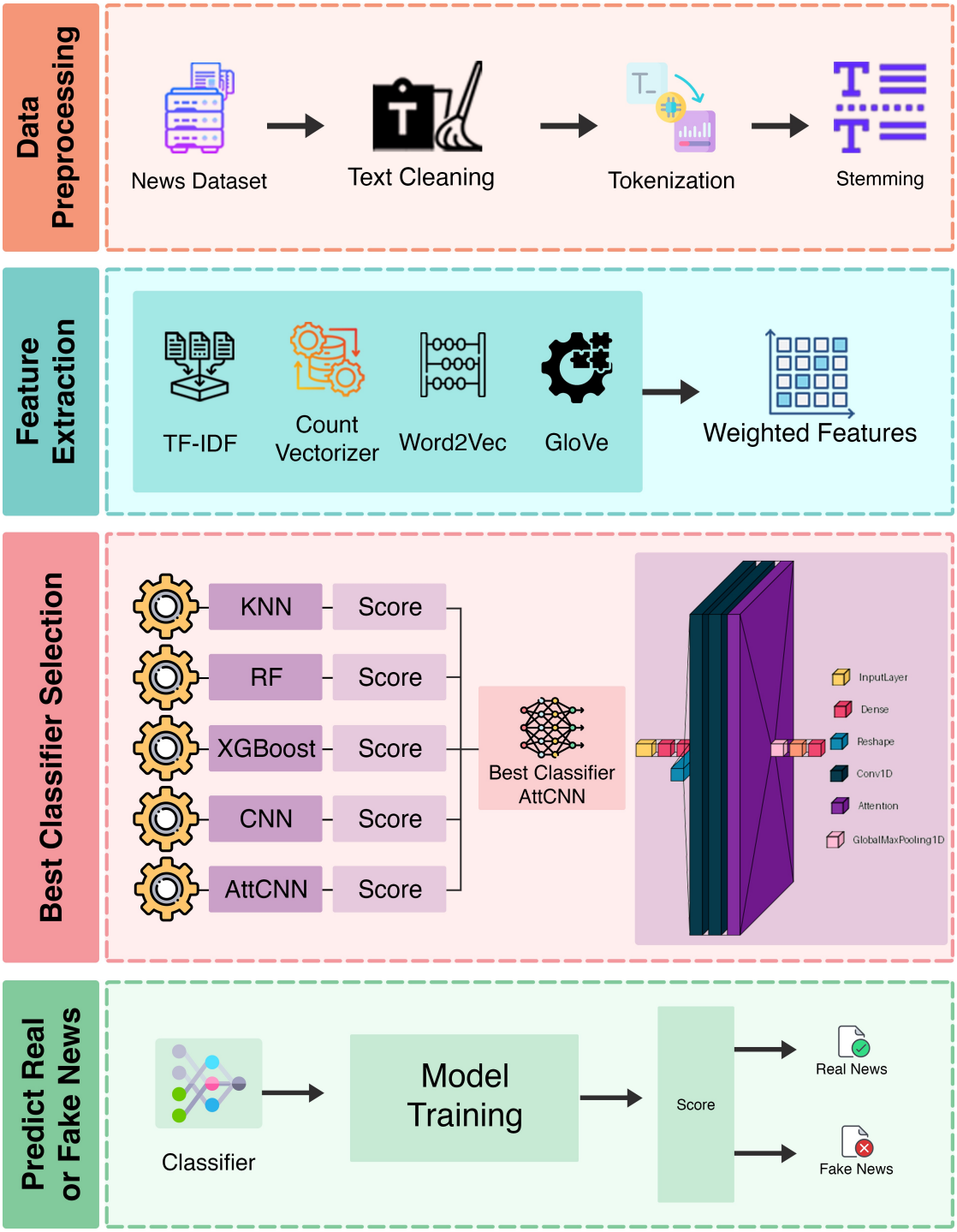

Figure 3.1: Modular system architecture illustrating the interconnections among four principal components: (a) data preprocessing for text cleansing and normalization, (b) hybrid feature extraction utilizing GloVe and CountVectorizer, (c) AttCNN model architecture featuring an attention mechanism, and (d) validation module for performance evaluation.

corporates many text representation methods, such as TF-IDF, CountVectorizer, Word2Vec (W2V), and GloVe. The model architecture module uses our AttCNN design, which integrates an attention strategy to improve feature learning. The validation module guarantees model dependability via thorough performance assessment and cross-validation methods. The modules are linked via an optimized pipeline that facilitates the efficient processing of news items from input to final categorization.

3.1.3 Method of Implementation

This framework's implementation utilizes contemporary deep learning libraries and adheres to software engineering best practices to guarantee repeatability and extensibility. The system architecture is structured to provide future improvements and adjustments to changing fake news trends while ensuring computational performance.

3.2 Dataset

3.2.1 Overview and Preprocessing

The research utilizes an extensive dataset initially obtained from Kaggle and previously employed by Padalko, H et al. [48]. This dataset constitutes a meticulously assembled compilation of news stories, particularly intended to furnish fair and impartial training material for the identification of false information. The corpus consists of 20,800 news pieces, with a balanced mix of 10,413 genuine news articles and 10,387 fraudulent news items. The feaures detailes present in the Table 1, and the details of data preprocessing is visualized in the Figure 3.2(A). The dataset's

balanced composition guarantees effective model training and assessment, reducing bias toward any class.

Table 1: Attributes of a News Article Dataset

Attribute	Description
id	Unique identifier for a news article.
title	The title of a news article.
author	The author of the news article.
text	The content of the article; may be incomplete.
label	A label marking the article as potentially unreliable.

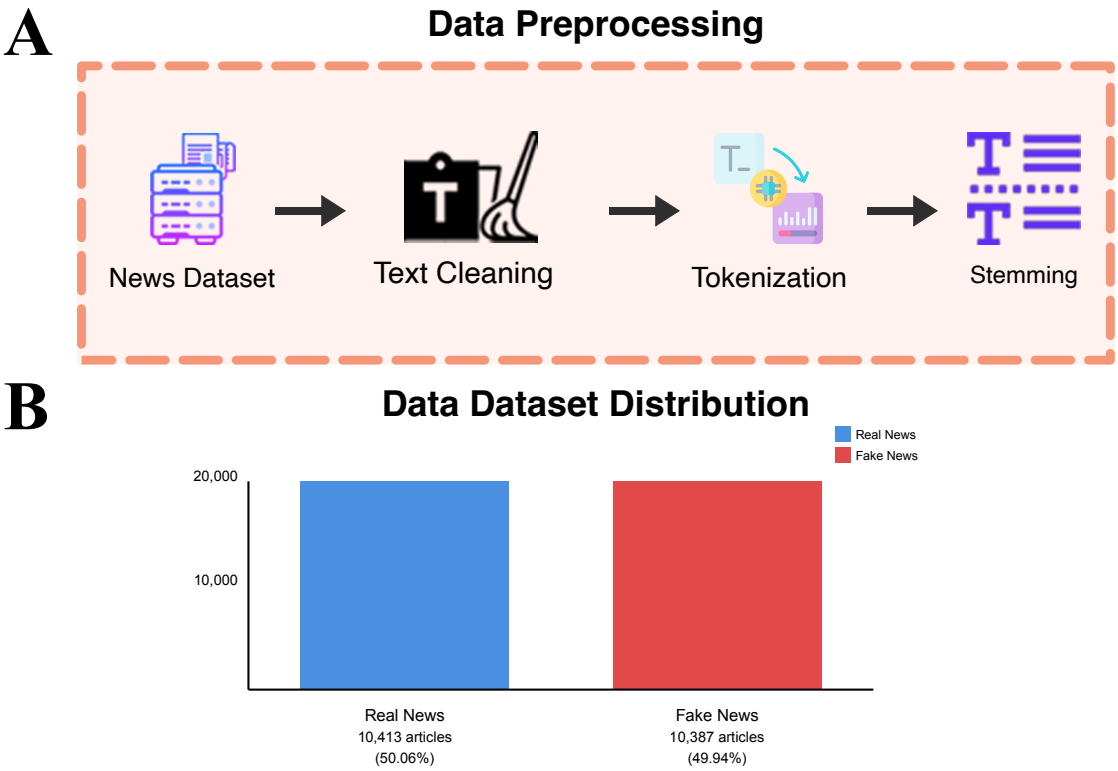

Figure 3.2: (A) Illustrates the data preparation procedure, encompassing collecting, cleaning, tokenization, and streaming. (B) Dataset distribution illustrating the equitable proportion of authentic and fabricated news pieces. The image depicts a roughly equal distribution of actual news (10,413 articles, 50.06%) and false news (10,387 articles, 49.94%), as well as the 80:20 train-test split ratio employed for model evaluation.

3.2.2 Data Cleansing and Preparation

The data preparation procedure entails a thorough cleaning pipeline to provide optimal quality for model training. Initially, all content is normalized by converting to lowercase, removing special characters and punctuation, eliminating HTML elements and URLs, and standardizing whitespace and newlines. This standardization guarantees uniformity across all articles and minimizes noise in the dataset. The content processing phase entails the elimination of stop words and the lemmatization of terms to their root forms, hence improving the efficacy of future feature extraction procedures. The pipeline also addresses absent values in the author and title fields and amalgamates the title and text fields for thorough analysis.

3.2.3 Dataset Partitioning Strategy

The dataset is divided using an 80:20 ratio for independent testing, a method used to provide thorough model evaluation while preserving adequate training data. As illustrated in Figure 3.2 (B) The training set consists of 16,640 articles, representing 80% of the dataset, utilized for model training and internal validation. The remaining 4,160 articles (20%) are designated only for the final model evaluation, functioning as an independent test set. This partitioning method preserves the original class distribution in both sets, guaranteeing that the model is trained and assessed on representative samples of both authentic and fabricated news items.

3.3 Feature Engineering

3.3.1 Word Embeddings

Word embeddings are essential for capturing semantic links among words in our false news detection algorithm. We employ two principal word embedding methods: Word2Vec and GloVe, each providing unique benefits in the representation of textual characteristics.

3.3.1.1 Word2Vec Implementation and Architecture

Word2Vec is a neural network-based embedding method that acquires vector representations of words by training on an extensive text corpus [64]. It produces compact word vectors that encapsulate semantic links among words. Word2Vec utilizes two fundamental models: CBOW and Skip-gram. The CBOW model forecasts a target word utilizing its contextual words, whereas the Skip-gram approach anticipates context words based on a target word. The objective functions for these models are delineated as follows:

$$\mathcal{L}_{\text{CBOW}} = -\log p(w_1 | w_{i-n}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+n}) \tag{3.1}$$

$$\mathcal{L}_{\text{skip-gram}} = -\log p(w_{i-m}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+n} | w_i) \tag{3.2}$$

where w_t denotes the target word, and $w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m}$ signify the context words. The resultant word vectors are dense and include semantic commonalities, thus analogous words possess comparable vectors.

3.3.1.2 GloVe embedding integration

GloVe is an unsupervised learning method for word vector representation. This method utilizes a log-bilinear regression model to identify statistical correlations among words in a corpus [65]. The model is trained with a global word-to-word co-occurrence matrix, comprising non-zero values that quantify the probability of simultaneous word appearances inside a corpus. The ratio of non-zero entries in the matrix is significantly lower than the total number of words in the corpus. A comprehensive traversal of the whole corpus is necessary to get statistics for the subsequent matrix. These traversals for extensive corpora may incur substantial costs. Furthermore, the resulting representations illustrate the significant linear elements inside the word vector spaces.

$$\sum_{i,j}^{N} \left[f(X_{i,j}) \left(v_i^T v_j + b_i + b_j - \log(X_{i,j}) \right) \right]^2 \tag{3.3}$$

In this context, v_i and v_j represent the word embeddings of i and j, respectively, whereas X signifies a word co-occurrence matrix. The co-occurrence frequency of word j with word i is denoted by $X_{i,j}$. Furthermore, the likelihood of word j occurring in the context of word i is denoted as follows:

$$X_{i,j} = P(i) = \frac{X_{i,j}}{X_i} \tag{3.4}$$

The fundamental design of the GloVe model is illustrated in 3.3.

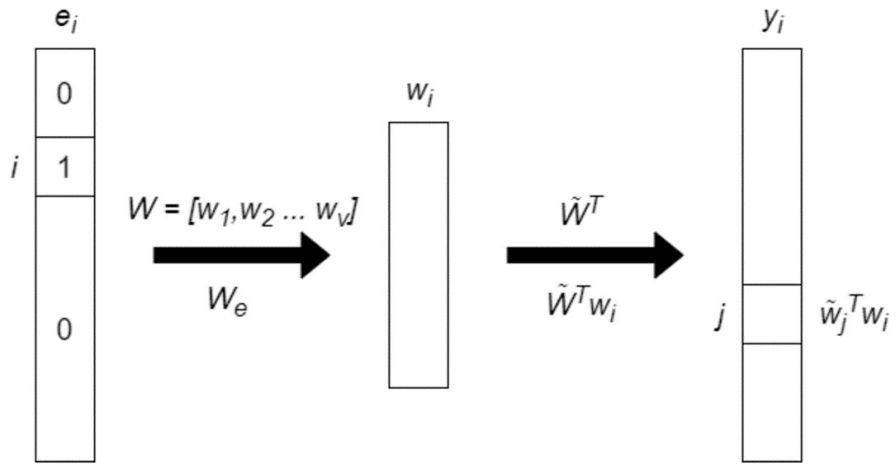

Figure 3.3: The basic architecture of GloVe model.

3.3.2 Statistical Features

3.3.2.1 TF-IDF

The TF-IDF is a statistical metric employed to assess the significance of a term inside a document in relation to a corpus of documents. It integrates two metrics: Term Frequency (TF), which quantifies the occurrence of a term inside a document, and Inverse Document Frequency (IDF), which assesses the uniqueness or rarity of a word throughout the corpus. TF-IDF is mathematically defined as follows

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{3.5}$$

where

$$\text{TF}(t, d) = \frac{f(t, d)}{\sum_{t' \in d} f(t', d)} \tag{3.6}$$

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3.7}$$

Here, $f(t, d)$ represents the frequency of term t in document d, $|D|$ is the total number of documents, and $|d \in D : t \in d|$ is the number of documents containing term t. TF-IDF assigns higher scores to terms that are frequent in a document but rare across the corpus, thus highlighting terms that are more relevant to the specific document.

3.3.2.2 CountVectorizer Processing

CountVectorizer is crucial for identifying bogus news by transforming text into numerical vectors according to word frequencies. The expressions "Fake news spreads misinformation" and "Misinformation is harmful" are depicted as word count vectors: 'Fake': 1, 'news': 1, 'spreads': 1, 'misinformation': 2, 'is': 1, 'harmful': 1. This approach facilitates the identification of lexical use patterns that signify misinformation, hence allowing for effective analysis and categorization by machine learning algorithms.

3.4 Hybrid Feature

The feature fusion strategy (hybrid feature) utilizes a systematic approach for combining multiple feature extraction techniques, taking advantage of the complementary strengths of different text representation methods. The FIC framework, as shown in Figure 3, establishes a variety of combinations of the W2V, GloVe embeddings, TF-IDF, and CountVectorizer features to further facilitate the model in revealing fake news. By trying various merges on past data (W2V+GloVe, TF-IDF+W2V, CountVectorizer+W2V, CountVectorizer+TF-IDF, CountVectorizer+

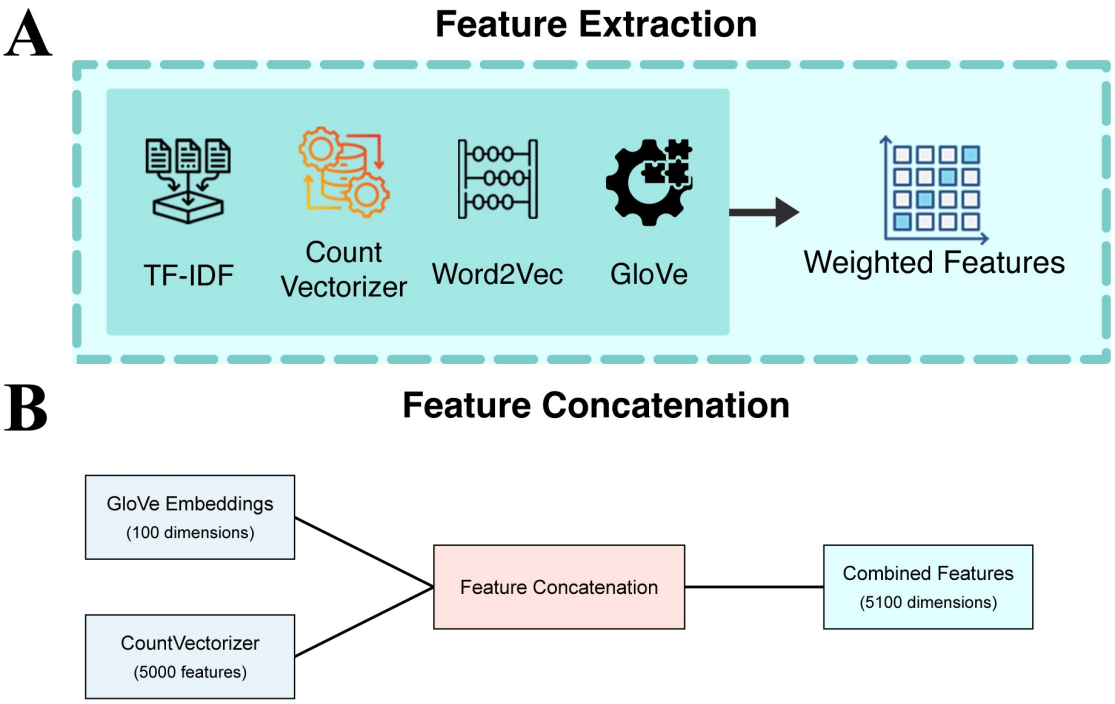

Figure 3.4: An optimal feature fusion architecture illustrating the combination of GloVe embeddings with CountVectorizer features, including the concatenation process and the resultant feature dimensions.

GloVe, TF-IDF+GloVe), we came across there best results with GloVe embeddings along with CountVectorizer.

Constructing the features: The concatanation of features is done as GloVe embeddings (100D) + countvectorizer Features (5000D) as shown in Figure 3.4. The Outcome of this integration is a character of 5100 dimensions which contains both the semantic relations(through GloVe embeddings) as well as the statistical patterns (through CountVectorizer). The combination of GloVe+CountVectorizer outperformed others across various fine-grained metrics and retained statistical effectiveness on fake news detection, striking an appropriate semantic significance balance through an extensive analysis and ablation studies.

3.5 Machine Learning Techniques

Various classification techniques in machine learning have proven useful in identifying bogus news. This research applies and assesses three notable algorithms: RF, KNN, and XGBoost. Each algorithm possesses distinct advantages and methodological frameworks for distinguishing between genuine and counterfeit news information.

3.5.1 Random Forest

RF is an ensemble learning method that integrates numerous decision trees to improve classification accuracy and mitigate overfitting. In the realm of false news identification, RF functions by generating several decision trees, each trained on distinct subsets of the data using bootstrap aggregation (bagging). Each tree in the forest autonomously analyzes the input elements of news stories and formulates its own forecast. The ultimate categorization choice is established via majority voting among all trees, ensuring a strong and dependable prediction method. This method is very adept at managing the high-dimensional characteristics of text data and demonstrates resilience to noise within the feature space. The algorithm's capacity to deliver feature significance rankings provides significant insights into which elements of news items are most suggestive of legitimacy [66, 67].

3.5.2 K-Nearest Neighbors

The KNN algorithm classifies false news using a distance-based method, determining the validity of a news article by the majority class of its K nearest neighbors in the feature space. In our implementation, KNN evaluates the feature vectors

extracted from news articles, taking into account both semantic and statistical attributes. The algorithm's efficacy is fundamentally contingent upon the selection of K and the distance metric employed to assess similarity across articles. KNN is non-parametric and adaptive to emerging data patterns; nevertheless, its efficacy may be compromised by the high dimensionality of text characteristics and the processing demands associated with huge datasets [68].

3.5.3 eXtreme Gradient Boosting

XGBoost is a sophisticated ensemble learning method that has exhibited remarkable efficacy in the identification of false news. The approach constructs an ensemble of weak learners in a sequential manner, with each subsequent model aimed at rectifying the flaws of its predecessors using gradient boosting. XGBoost processes hybrid feature representations of news items by employing both semantic embeddings and statistical features in our approach. The algorithm's efficacy arises from its capacity to address intricate feature interactions, mitigate overfitting via regularization, and effectively process missing information. XGBoost has inherent capabilities for early halting and feature significance assessment, rendering it especially appropriate for the evolving landscape of false news identification [69].

3.6 Proposed model (AttCNN)

The AttCNN for fake news detection integrates advanced deep learning techniques to effectively analyze textual data. Our model work is illustrated in Figure 3.5. The Attention-based CNN consists of five key components [48], this model is tailored to discern between genuine and fake news articles:

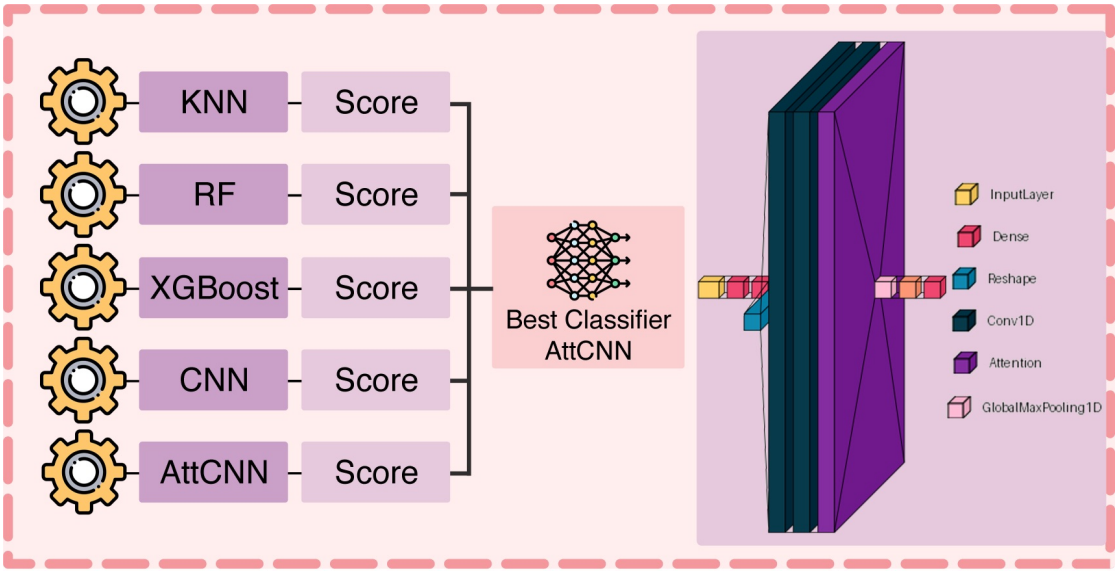

Figure 3.5: Comparative evaluation framework of machine learning classifiers (KNN, SVM, XGBoost, CNN, AttCNN) for fake news detection, with corresponding performance scores used for model selection. The architecture of the winning AttCNN model shows its layer composition including Input, Dense, Conv1D, Attention, and GlobalMaxPooling1D layers

The CNN structure consists of multiple convolutional layers followed by pooling operations. Each convolutional layer applies a set of filters to the input text, represented as a matrix of word embeddings. The convolution operation can be expressed as:

$$h_i = f(\sum_{j=1}^{k} w_j \cdot x_{i+j-1} + b) \tag{3.8}$$

Where $x_{i:i+j-1}$ is the input slice, w_j are the filter weights, b is the bias term, and f is an activation function (typically ReLU).

The attention mechanism is integrated after the convolutional layers to weigh the importance of different parts of the text. The attention weights are computed as:

$$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)} \tag{3.9}$$

Where e_i is the alignment score for the $i-th$ element, and n is the sequence length.

The hybrid feature integration combines the CNN outputs with the attention-weighted features, providing a rich representation for classification. This integration can be represented as:

$$F_{final} = [F_{CNN}; F_{attention}] \tag{3.10}$$

3.6.1 Convolutional Layers

These layers are crucial in the CNN design for detecting bogus news. The system utilizes filters to extract crucial characteristics from the textual input. Every filter examines the sequence of words, detecting patterns and certain characteristics that are essential for categorization.

3.6.2 Attention Mechanism

The attention mechanism in a CNN improves its capacity to prioritize significant portions of textual input. Attention, via the assignment of weights to words or phrases according to their significance, enables CNN to concentrate on critical aspects that differentiate genuine news from fabricated news. This adaptive methodology enhances precision by emphasizing noteworthy textual indicators such as deceptive information or prejudiced language, hence enhancing the model's resilience in managing varied and intricate datasets.

3.6.3 Pooling Layers

Pooling layers, such as max-pooling or average-pooling, are employed to decrease the dimensions of the feature maps generated by the convolutional layers. This process decreases the number of dimensions in the extracted features while still keeping the most important information, making it easier to classify efficiently.

3.6.4 Fully Connected Layers

The layers serve as the ultimate classifier in the CNN model. The convolutional and pooling layers provide the processed characteristics, which are then aggregated to determine the authenticity of the input news item. The outputs generated by these layers indicate the likelihood distribution among the different classes.

3.6.5 Output Layer

The last layer of the model computes the weighted total of the CNN outputs generated by the attention mechanism and generates the ultimate classification, either false or genuine.

The AttCNN technique not only integrates information from the entire sequence but also possesses the capacity to selectively highlight its key elements. The detailed architecture of our proposed model is shown in Figure 3.6. This feature has the potential to improve the efficiency of the model. The training procedure is conducted using TensorFlow, a popular deep-learning framework, inside a Python setup. The simulations are executed on a Mac Mini M2 (16 GB RAM, 256 GB storage) in the Anaconda environment using PyCharm and Jupyter Notebook. The datasets and programs used in this research will be available on GitHub at

the following link: https://github.com/rudradcruze/Fake-News-Thesis-BSc

3.7 Model Optimization and Validation

3.7.1 Cross-Validation Strategy

K-fold CV is employed to guarantee thorough model assessment. The dataset is partitioned into K subsets, utilizing K-1 subsets for training and one for verification at each cycle. This procedure is executed K times, with each subset utilized as the validation set once. This study examined K values of 10, 20, and 30 to evaluate their effect on performance stability. The cross-validation process can be represented as:

The cross-validation process can be represented as:

$$CV_{score} = \frac{1}{K} \sum_{i=1}^{K} score_i \tag{3.11}$$

Where $score_i$ is the performance metric for the i-th fold.

Performance stability is analyzed by examining the variance in scores across folds:

$$\sigma^2 = \frac{1}{K} \sum_{i=1}^{K} (score_i - \overline{score})^2 \tag{3.12}$$

Statistical significance testing, such as paired t-tests, is conducted to compare different model configurations and ensure that performance improvements are not due to chance.

3.7.2 Hyperparameter Optimization

The Grid Search CV technique is utilized for hyperparameter tuning. A parameter grid is constructed to investigate diverse combinations of hyperparameters, such as learning rate, batch size, and network architectural characteristics.

$$\theta^* = \arg\max_{\theta \in \Theta} CV_{score}(\theta) \tag{3.13}$$

Where Θ is the set of all hyperparameter combinations, and $CV_{score}(\theta)$ is the cross-validation score for a given set of hyperparameters.

Techniques for optimizing search space, including random search and Bayesian optimization, are seen as enhancements to computing efficiency. Optimal parameter selection criteria generally entail maximizing the mean cross-validation score while accounting for performance stability over folds.

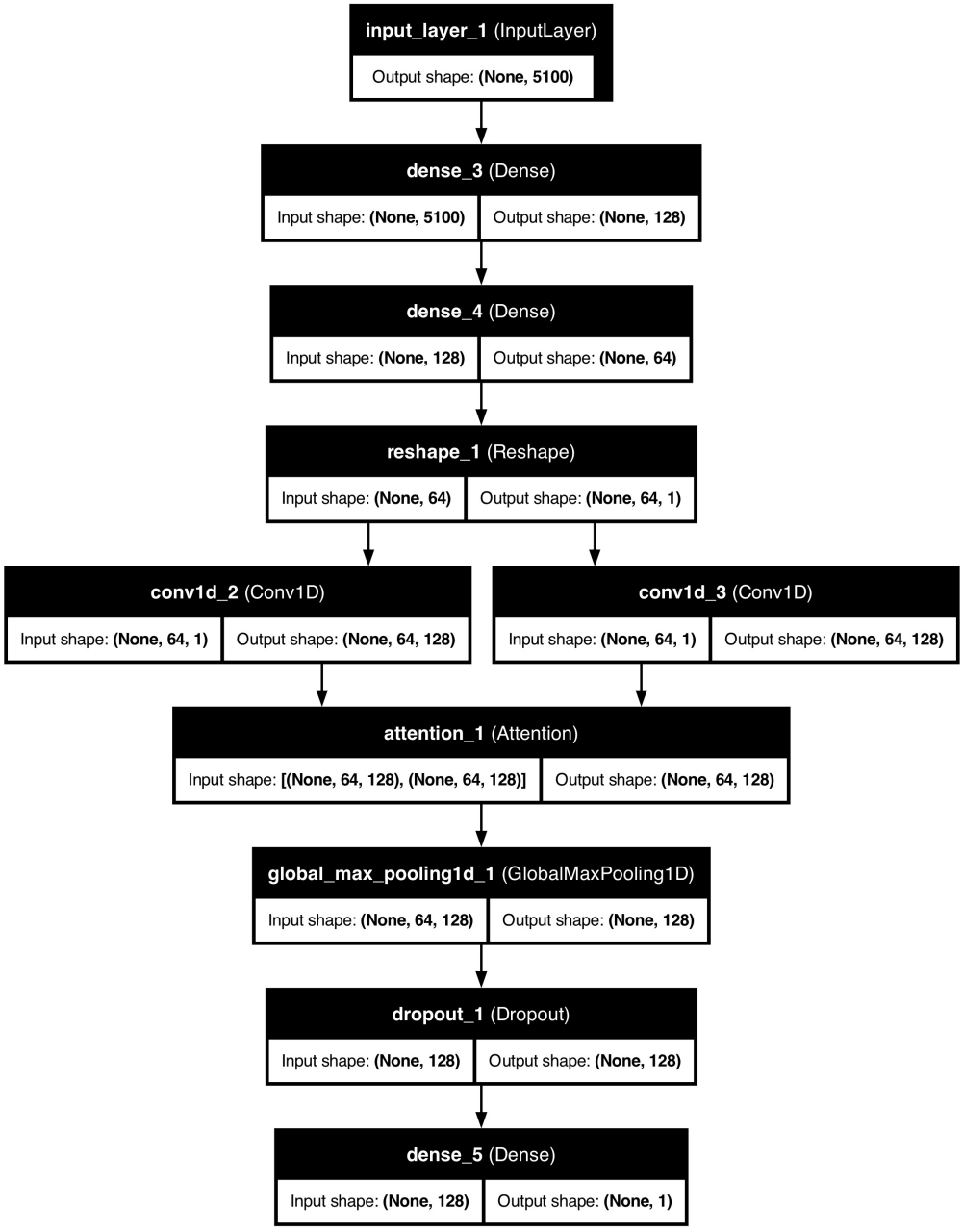

Figure 3.6: Comprehensive architecture schematic of the AttCNN model for false news detection, illustrating the sequential flow and dimensional changes across many levels. The network analyzes input via dense layers (5100→128→64), reshapes the data, executes concurrent Conv1D operations, incorporates an attention mechanism, and culminates with global max pooling, dropout, and a final dense classification layer, detailing the input/output forms at each level.

Chapter 4

Results and Analysis

4.1 Experimental Setup

4.1.1 System Configuration and Development Environment

The experiments of this work are executed on a Mac Mini M2 (16GB RAM and 256GB storage) based on the Apple Silicon architecture. PyCharm IDEs and Jupyter Notebook was used within both IDEs and within a Conda environment to form a strong but flexible model development and testing environment. During the software implementation, we used the modules and libraries for machine learning for the latest versions, such as (optimized for Apple Silicon) TensorFlow, Scikit-learn, NLTK, NumPy, and Pandas, which worked most proficiently and efficiently with our hardware setup.

4.1.2 Performance Measures

Assessing the effectiveness of machine and deep learning models for false news identification requires a thorough evaluation of many performance measures. These metrics offer essential insights into the models' proficiency in effectively distin-

guishing between fraudulent and authentic news items, guaranteeing that the detection method is both efficient and dependable.

1. Accuracy measures a model's overall performance by calculating the ratio of properly categorized examples, encompassing both TPs[1] and TNs[2], to the total number of occurrences. The model's overall performance is indicated, although it may not adequately reflect its efficacy in identifying bogus news if class distributions are skewed.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

2. Precision evaluates the model's capacity to accurately identify counterfeit news items, hence reducing the incidence of FPs[3]. It is especially crucial in situations when false alarms incur significant costs, since it quantifies the ratio of true positives to the overall occurrences identified as fake news.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

3. Recall assesses the model's ability to identify all genuine instances of bogus news. It computes the ratio of true positives to the overall count of real fake news stories, encompassing those that were overlooked (FNs[4]). High recall guarantees the identification of most bogus news stories, which is essential for reducing the danger of neglected misleading information.

[1]TP - True Positive
[2]TN - True Negative
[3]FP - False Positive
[4]FN - False Negative

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

4. The F1 Score offers a fair assessment of a model's accuracy and recall by calculating their harmonic mean. This metric is particularly valuable in detecting misinformation, as it balances the trade-off between precision and memory, preventing the predominance of false positives or false negatives in the assessment.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

4.2 Baseline Models and Feature Extraction

In our experimental framework, we employed many baseline models and feature extraction methods to thoroughly assess the efficacy of our suggested strategy. The specifics of the implementation are as follows:

4.2.1 Feature Extraction Methods

Table 1: Feature Extraction Methods

Method	Parameters	Description
TF-IDF	`max_features=5000`	Converts text to numerical vectors using TF-
CountVectorizer	`max_features=5000`	Creates a vocabulary of words and represents documents through word counts.
Word2Vec	`vector_size=200, window=5, min_count=1`	Generates dense vector representations of words based on their context.
GloVe	`dimension=100`	Pre-trained word embeddings capturing global word-word co-occurrence statistics.

4.2.2 Traditional Machine Learning Models

Table 2: Parameters of Traditional Machine Learning Models

Parameter	XGBoost	Random Forest	KNN
n_estimators	300	100	-
max_depth	6	-	-
learning_rate	0.005	-	-
subsample	0.8	-	-
colsample_bytree	0.8	-	-
gamma	1	-	-
min_child_weight	3	-	-
reg_alpha	0.1	-	-
reg_lambda	1	-	-
random_state	-	101	-
n_neighbors	-	-	5

4.2.3 Deep Learning Models

This section breaks down the specifications for two deep learning models: the Standard CNN and AttCNN, as seen in Table 3. The models vary in design; the Standard CNN employs fundamental dense and dropout layers, whereas AttCNN integrates more sophisticated elements, including batch normalization, multi-head convolutions, and attention algorithms. The table further contrasts the learning rates, batch sizes, and epochs employed for training each model.

4.2.4 Training Configuration

This section delineates the training configuration settings, as specified in Table 4. The table encompasses critical parameters like the train-test split ratio, model-

Table 3: Parameters of Deep Learning Models

Parameter	Standard CNN	AttCNN
Architecture	- Input - Dense(128) - Dropout(0.5) - Dense(64) - Dropout(0.5) - Output	- Input - Dense(256) - BatchNorm - Dense(128) - MultiHead Conv - Attention - GlobalPooling - Dense(256) - Dense(128) - Output
learning_rate	0.005	0.01
batch_size	32	32
epochs	20	25

dependent batch size, the Adam optimizer, and the binary cross-entropy loss function. Furthermore, early halting with a patience of 5 and a learning rate scheduler are established to modify the learning rate throughout training for maximum efficacy.

4.3 Performance Analysis

4.3.1 Overview of Embedding Evaluation

For false news detection, our extensive study of text embedding techniques involves examining four distinct embedding methods, specifically CountVectorizer, TF-IDF, Word2Vec (W2Vec), and GloVe embeddings. To validate the study, different train-test split ratios were used, and the main results for the 60:40 split are

Table 4: Training Configuration Parameters

Parameter	Value
Train-Test Split	80-20
Batch Size	32/64 (model dependent)
Optimizer	Adam
Loss Function	Binary Cross-entropy
Early Stopping	`patience=5, monitor='val_loss'`
Learning Rate Scheduler	`Initial LR=0.0001, drop=0.5, epochs_drop=5`

presented in Table 5, while the complete results for the 70:30 and 80:20 splits are supplemented in the supplementary materials A. The ROC curve for the 60:40 train-test split is shown in Figure 4.1, while the remaining curves for different embedding techniques are provided in the supplementary materials A.

4.3.2 Analysis of CountVectorizer and TF-IDF Performance

The experimental results show that CountVectorizer achieved better performance on most of the evaluation metrics. For instance, Table 5 shows that an accuracy of 99.09%, a precision of 99.23%, is in turn achieved by the CountVectorizer embedding and AttCNN while having a superior AUC score of 99.90%. CountVectorizer is scalable, and it retains the structural representation of the text data, thereby contributing to the efficiency of This performance. Figure 4.2 plots the performance metrics for all embedding approaches vs. various models showing the comparing ROC curves for the 60:40 split ratio.

TF-IDF embeddings were not much more effective than CountVectorizer type

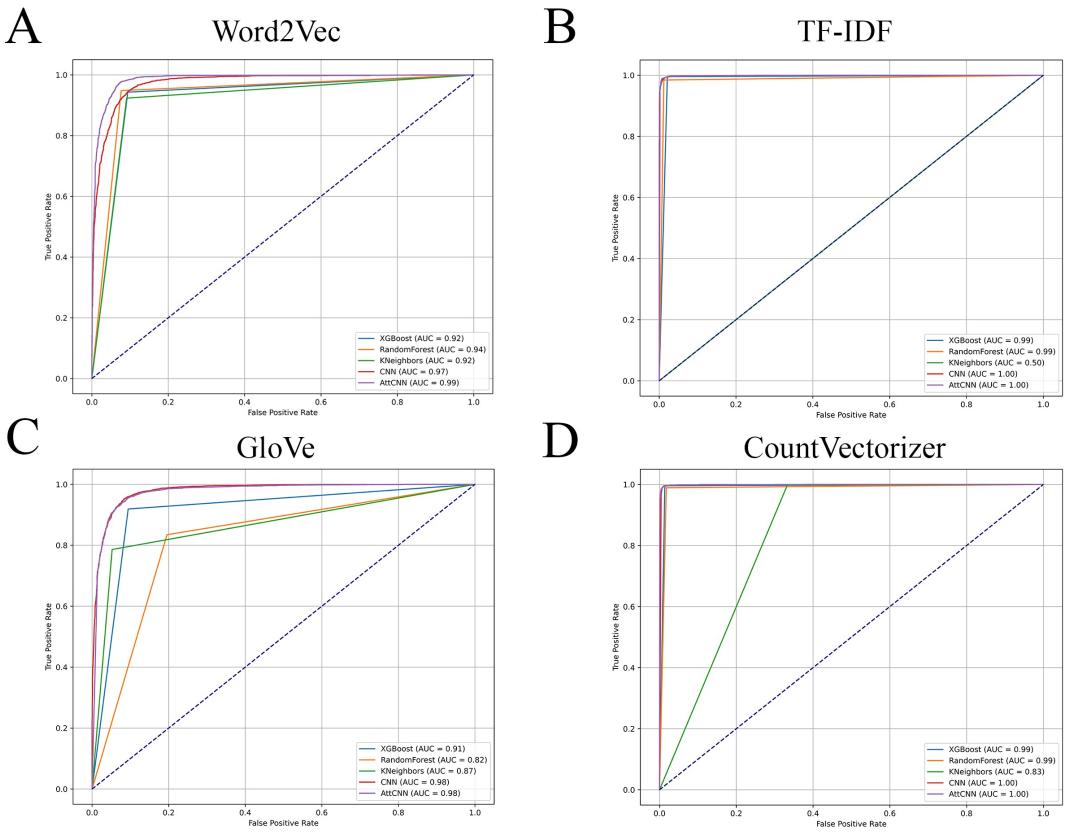

Figure 4.1: ROC curve for the 60:40 train-test split. Comparing various machine learning classifiers (XGBoost, RF, KNN, CNN, and AttCNN) using different text encoding methods: (A) Word2Vec, (B) TF-IDF, (C) GloVe, and (D) CountVectorizer for fake news detection.

embeddings, particularly when incorporated into deep learning pipelines. Out of all machine learning algorithms, the AttCNN model with TF-IDF embeddings achieved the highest accuracy of 99.06% and the best AUC score of 99.92% among all feature extraction methods used, suggesting that the inverse document frequency weighting enhanced its discriminative power for detecting false news. This finding supports the idea that among documents, the relative importance of specific terms helps to distinguish between real and fake news articles.

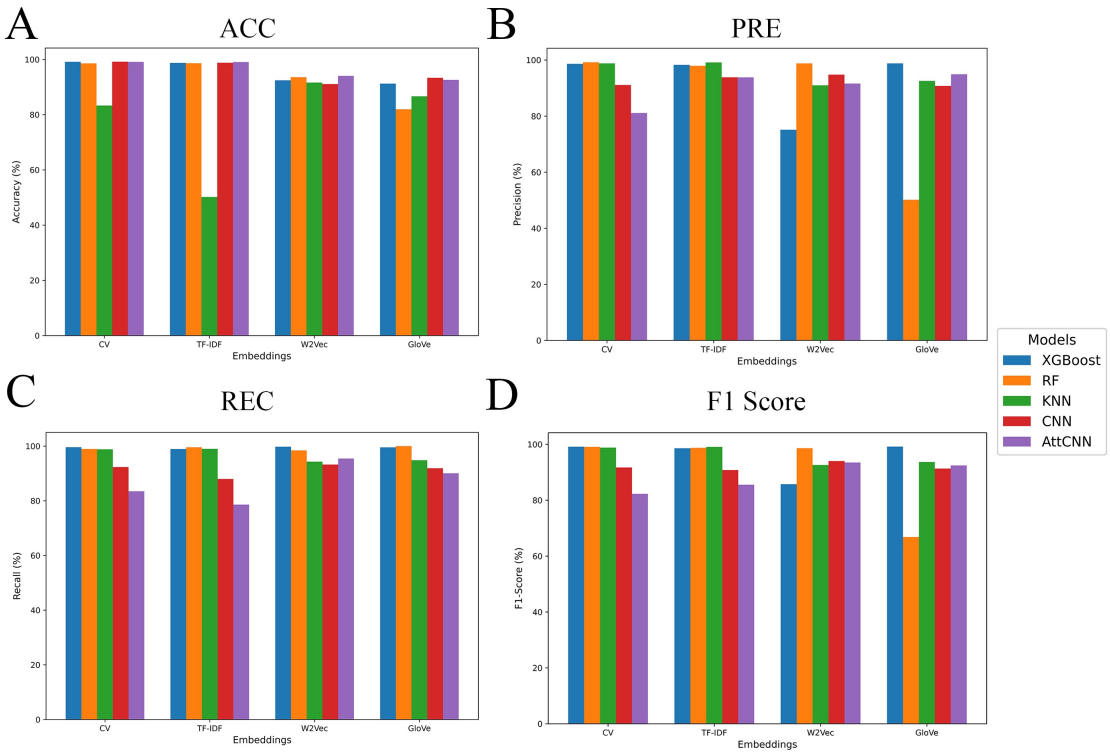

Figure 4.2: Comparison of the performance of machine learning classifiers (XGBoost, RF, KNN, CNN, and AttCNN) trained on a 60:40 train-test split with various embedding approaches (CV, TF-IDF, W2Vec, and GloVe). The assessment metrics shown are (A) Accuracy, (B) Precision, (C) Recall, and (D) F1 Score, all expressed as percentages.

4.3.3 Word Embedding Methods Performance

While they had lower absolute accuracy scores, Word2Vec embeddings performed better in terms of capturing semantic linkages in the text. Table 5 presents the results of Accuracy (94.04%) and AUC score (98.59%) achieved by the AttCNN model equipped with Word2Vec word embeddings. The comparatively lower effectivity compared to frequency-based methods suggests that while semantic embeddings are able to summarise important word relations, they can occasionally miss out on superficial statistical characteristics that may help classify false news.

GloVe embeddings gave comparable accuracy (92.60%) and AUC score (97.59%) to the Word2Vec embeddings when used with the AttCNN mode The fine-tuned features of GloVe embeddings gave a very good performance across a number of models, with no gradual decrease in AUC observed across any deep learning architectures, and AUC values recorded of at least 96% across all models. The robustness reflects that the GloVe embeddings, in a global sense, supply substantial features for the classification of false news by their co-occurrence statistics.

4.3.4 Model-Specific Performance Comparison

A higher performance of the AttCNN architecture can be observed for all embedding approaches in Figure 4.2. The success of the attention mechanism approach is attributable to its ability to focus on important components of a text while simultaneously maintaining awareness of context. Traditional machine learning algorithms (e.g. XGBoost and RF) performed well (compared to the deep learning methods) when tested with frequency-based embeddings, but less so with word embeddings, suggesting these models would be more suited for interpreting discrete frequency data than dense semantic vectors.

Table 5: Performance Metrics for Various Models and Embeddings at Different Data Ratios

Ratio	Embedding	Model	Acc	Prec	Rec	F1	AUC	Spec
60 : 40	CV	XGBoost	99.11	98.65	99.59	99.12	99.11	98.62
		RF	98.58	98.26	98.92	98.59	98.58	98.24
		KNN	83.31	75.12	99.78	85.71	83.25	66.71
		CNN	99.17	98.81	99.54	99.18	99.75	98.79

Ratio	Embedding	Model	Acc	Prec	Rec	F1	AUC	Spec
		AttCNN	99.09	99.23	98.95	99.09	99.90	99.23
	TF-IDF	XGBoost	98.73	97.93	99.57	98.74	98.72	97.88
		RF	98.62	98.80	98.44	98.62	98.62	98.79
		KNN	50.18	50.18	100.00	66.83	50.00	0.00
		CNN	98.81	98.80	98.83	98.81	99.84	98.79
		AttCNN	99.06	99.14	98.99	99.07	99.92	99.13
	W2Vec	XGBoost	92.45	90.99	94.30	92.61	92.45	90.59
		RF	93.58	92.56	94.83	93.68	93.58	92.33
		KNN	91.62	91.09	92.34	91.71	91.62	90.90
		CNN	91.06	93.82	87.98	90.80	97.41	94.16
		AttCNN	94.04	94.79	93.25	94.01	98.59	94.84
	GloVe	XGBoost	91.24	90.75	91.90	91.32	91.24	90.57
		RF	81.96	81.12	83.47	82.28	81.95	80.43
		KNN	86.66	93.80	78.61	85.54	86.69	94.76
		CNN	93.33	91.63	95.43	93.49	98.01	91.22
		AttCNN	92.60	94.93	90.06	92.43	97.59	95.15
70 : 30	CV	XGBoost	99.15	98.77	99.55	99.16	99.15	98.74
		RF	98.70	98.11	99.33	98.72	98.70	98.07
		KNN	83.85	75.77	99.78	86.13	83.76	67.74
		CNN	99.18	99.27	99.11	99.19	99.88	99.26
		AttCNN	99.36	99.33	99.39	99.36	99.89	99.32
	TF-IDF	XGBoost	98.88	98.12	99.68	98.89	98.87	98.07
		RF	99.10	99.14	99.08	99.11	99.10	99.13

Ratio	Embedding	Model	Acc	Prec	Rec	F1	AUC	Spec
		KNN	54.63	52.57	99.94	68.89	54.38	8.83
		CNN	99.02	99.49	98.57	99.02	99.92	99.48
		AttCNN	98.99	99.14	98.85	98.99	99.92	99.13
	W2Vec	XGBoost	92.23	90.50	94.45	92.43	92.22	89.98
		RF	93.53	92.24	95.12	93.66	93.52	91.91
		KNN	91.81	91.16	92.70	91.92	91.81	90.91
		CNN	91.99	92.44	91.55	91.99	97.38	92.43
		AttCNN	94.92	93.12	97.07	95.05	98.58	92.75
	GloVe	XGBoost	91.55	91.09	92.22	91.65	91.55	90.88
		RF	82.36	81.09	84.64	82.83	82.34	80.05
		KNN	87.50	93.96	80.30	86.59	87.54	94.78
		CNN	93.77	93.56	94.07	93.82	98.48	93.46
		AttCNN	93.48	92.87	94.26	93.56	98.07	92.68
80 : 20	CV	XGBoost	99.42	99.15	99.71	99.43	99.42	99.13
		RF	99.71	99.48	99.95	99.71	99.71	99.47
		KNN	83.77	75.70	99.86	86.11	83.65	67.44
		CNN	99.52	99.43	99.62	99.52	99.98	99.42
		AttCNN	99.47	99.15	99.81	99.48	99.95	99.13
	TF-IDF	XGBoost	99.06	98.49	99.67	99.08	99.06	98.45
		RF	99.21	99.01	99.39	99.21	99.21	99.02
		KNN	55.11	52.91	99.96	69.03	54.93	8.68
		CNN	99.06	99.72	98.42	99.05	99.92	99.70
		AttCNN	98.91	99.17	98.66	98.91	99.90	99.15

Ratio	Embedding	Model	Acc	Prec	Rec	F1	AUC	Spec
	W2Vec	XGBoost	93.50	92.27	94.57	93.63	93.50	91.87
		RF	94.21	93.12	95.35	94.29	94.21	92.52
		KNN	91.51	91.14	92.09	91.58	91.51	90.57
		CNN	92.44	92.12	92.67	92.44	97.63	92.09
		AttCNN	94.21	93.57	94.99	94.31	98.58	93.91
	GloVe	XGBoost	91.99	91.64	92.46	91.99	91.99	91.44
		RF	83.32	82.31	84.89	83.32	83.32	81.38
		KNN	87.99	94.62	80.53	86.63	87.92	94.68
		CNN	94.44	94.10	94.65	94.42	98.56	94.06
		AttCNN	94.28	93.98	94.59	94.29	98.34	93.91

The performance trends noted with the 60:40 split were largely consistent with other split ratios, as described in our supplemental materials A. However, the up-mentioned absolute performance metrics exhibited slight improvements with larger training data, specifically for word embedding methods. This trend suggests that semantic embeddings could gain more from augmented training data than frequency-based approaches. We conduct an exhaustive study of multiple alternative embedding methods and find a clear separate tradeoff between statistical and semantic approaches to text representation. While frequency-based approaches (CountVectorizer and TF-IDF) usually reached better classification metrics, the semantic richness from Word2Vec and GloVe embeddings provide valuable additional perspectives, particularly when used in more sophisticated deep learning architectures with attention. Try something like this: The results suggest future research would benefit from hybrid approaches, capturing the best of both

frequency- and semantic embedding-based approaches.

4.4 Evaluation of Computational Efficiency

Comprehensive time analysis was performed to provide a comprehensive evaluation of the computing necessity of different embedding methods and model architectures. The training and testing times for the various configurations are detailed in Table **??** for the 60:40, 70:30 and 80:20 split ratios, providing an overview of the practical implications of each approach.

4.4.1 Efficiency of Training Duration

An analysis of training times reveals different patterns across different combinations of embedding and models. At least XGboost required 3.16 seconds of training time at 60:40 split, whereas CountVectorizer was quick work in frequency-based embeddings. The Random Forest classifier had inconsistent computing demands with very high training times which increased sharply from 1.86 seconds on the 60:40 split to 27.89 seconds on the 80:20 split, indicating its high sensitive to the amount of training data. TF-IDF has relatively more computational power than CountVectorizer, as we see that the training time of XGBoost varied from 3.77 to 8.52 seconds depending on the split ratio.

Word embeddings showed different computational properties. In this study, GloVe embeddings sequentially demonstrated lower computational overhead, with training times ranging from 0.22 to 2.77 seconds depending on the model and split ratio employed. This is efficient as the characteristics of GloVe embeddings are pre-trained which reduces the complication in the training process. In turn,

Table 6: Training and Testing Times (sec) for Models and Embeddings

Embedding	Model	60:70		70:20		80:20	
		Train(s)	Test(s)	Train(s)	Test(s)	Train(s)	Test(s)
CV	XGBoost	3.1612	0.0782	2.4766	0.0203	3.1877	0.0562
	RF	1.8612	0.0618	1.2377	0.0409	27.8894	0.2441
	KNN	0.0020	11.2041	0.0015	5.8093	0.0015	5.2785
	CNN	1.9874	0.4706	1.5828	0.2370	1.8124	0.1535
	AttCNN	4.4394	1.0999	3.8940	0.5934	4.2727	0.3834
TF-IDF	XGBoost	3.7654	0.0768	6.8061	0.0444	8.5215	0.0414
	RF	1.9875	0.0734	2.4031	0.0525	28.4725	0.2698
	KNN	0.0126	6.9206	0.0147	5.2381	0.0443	4.0475
	CNN	1.0224	0.2541	1.2600	0.2032	1.5675	0.1473
	AttCNN	3.1326	0.7385	4.1583	0.5835	3.7369	0.3665
W2Vec	XGBoost	2.8503	0.0087	2.8875	0.0080	3.4530	0.0044
	RF	9.1833	0.0481	11.6149	0.0400	13.9340	0.0297
	KNN	0.0014	0.2409	0.0013	0.3158	0.0016	0.2435
	CNN	0.4206	0.1197	0.2602	0.1026	0.4761	0.0802
	AttCNN	10.0534	2.6062	12.3503	1.9711	16.7668	2.1036
GloVe	XGBoost	0.6459	0.0038	0.8598	0.0035	0.7392	0.0028
	RF	0.4333	0.0041	0.4911	0.0032	11.3370	0.0357
	KNN	0.0018	0.1885	0.0025	0.1419	0.0018	0.1911
	CNN	0.2223	0.1263	0.2336	0.0992	0.4836	0.0768
	AttCNN	1.9007	0.5760	2.2879	0.4034	2.7705	0.2989

Word2Vec embeddings used in combination with the AttCNN model required the highest computing resources, taking 10.05–16.77 s for the training. The increased computational cost indicates the complexity of utilizing dense vector representations within the attention mechanism framework.

4.4.2 Inference Time Consideration

A study of testing times identified very impressive and operationally important trends. As depicted in Table 6, the KNN classifier exhibited a unique computational pattern, with very short durations for training (0.001-0.002 seconds) and much longer durations for testing, in particular with the CountVectorizer embeddings (11.20 seconds for the 60:40 split). This characteristic is due to KNN's instance-based learning approach, where the computational load is primarily during the inference stage. To affirm the suitability of XGBoost for real time applications, the testing time remained generally below 0.08 seconds for all embedding techniques which showed its constantly efficient testing performance throughout.

The inference times of the deep learning models were plausible, requiring 0.12-0.47 seconds for the usual CNN model over different embeddings. The AttCNN model provided inference times 0.58–2.61 s, which were reasonable considering its increased training costs, being that GloVe embeddings yielded the best inference time with much lower costs in all set-up compared to other models. This balance of computing demand and model performance makes AttCNN with GloVe embeddings an appealing option for real-life applications where both prediction quality and response time are decisive aspects.

These comprehensive timing benchmarks show large trade-offs between computing efficiency and model performance. While GloVe embeddings are undoubtedly the most computationally efficient way to represent text, particularly in the setting of conventional machine learning models — this comes at a small cost in terms of classification performance compared to frequency-based embeddings. Despite being computationally intensive, the AttCNN approach proves its worth by

significantly improving classification performance on all evaluation metrics. By providing an understanding of the trade-offs involved in computational efficiency, this thesis presents important guidance for practitioners at one point when deciding on the right combinations of model-embedding architectures depending on their scenario-specific requirements and resources.

4.5 Hybrid Embedding Analysis with AttCNN

As a result of the extraordinary performance of the Attention-based CNN model observed in our initial tests, we conducted a thorough investigation of hybrid embedding techniques based on the AttCNN architecture. Table 7 shows the performance details of each embedding combination, single and combined as well as the ROC curves for these combinations in Figure 4.3 and performance are shown in Figure 4.4.

It is demonstrated through experiment that hybrid embedding methods significantly enhance the performance of the model compared to only using one embedding. The hybrid approach combining CountVectorizer and GloVe embeddings (CV+GloVe) achieved the best results with 99.50% accuracy, 99.38% precision, 99.62% recall, and excellent AUC score (99.98%). These embeddings bring complementary features as CountVectorizer preserves local statistical information and GloVe embeddings provide extensive semantic knowledge from global word co-occurrences, which leads to even better performance.

A combination of different hybrids reveals interesting trends in model behavior. TF-IDF combined with GloVe embeddings performed strongly (accuracy: 99.33%, AUC: 99.96%) and combing CountVectorizer with TF-IDF and GloVe embeddings

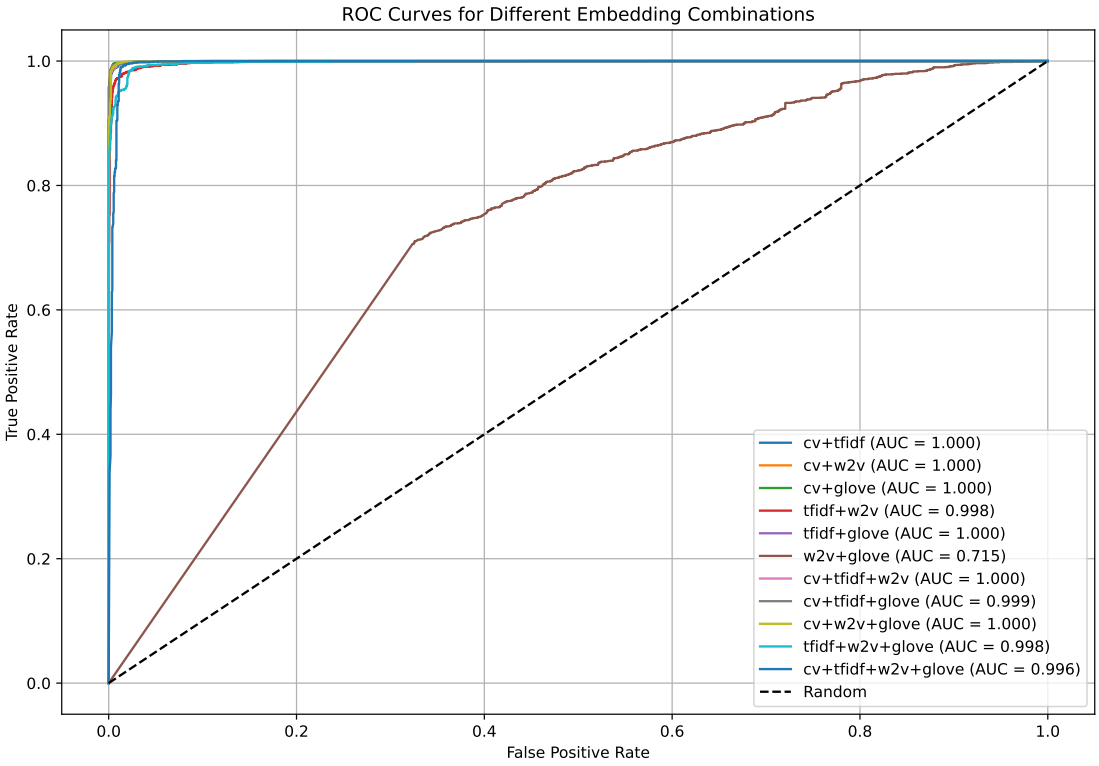

Figure 4.3: ROC curves comparing performance of different hybrid embedding combinations for fake news detection, showing superior performance of hybrid approaches with most achieving AUC scores of 1.000 except w2v+glove (AUC = 0.715)

provided accuracy: 99.45%. However, not every combination led to performance improvement, The combined of Word2Vec with GloVe (W2V+GloVe) performed poorly (accuracy: 51.18%, AUC: 71.52%), suggesting that similar word embedding can interact.

These observations are further confirmed by the ROC curvespresented in Figure 3, where the optimal curves were foundfor the combination of CV+GloVe, showing optimal features atdifferent classification thresholds in terms of the trade-offbetween true positive and false positive rates. The image clearly depicts the superior discriminative power of our hybrid approach compared to other embed-

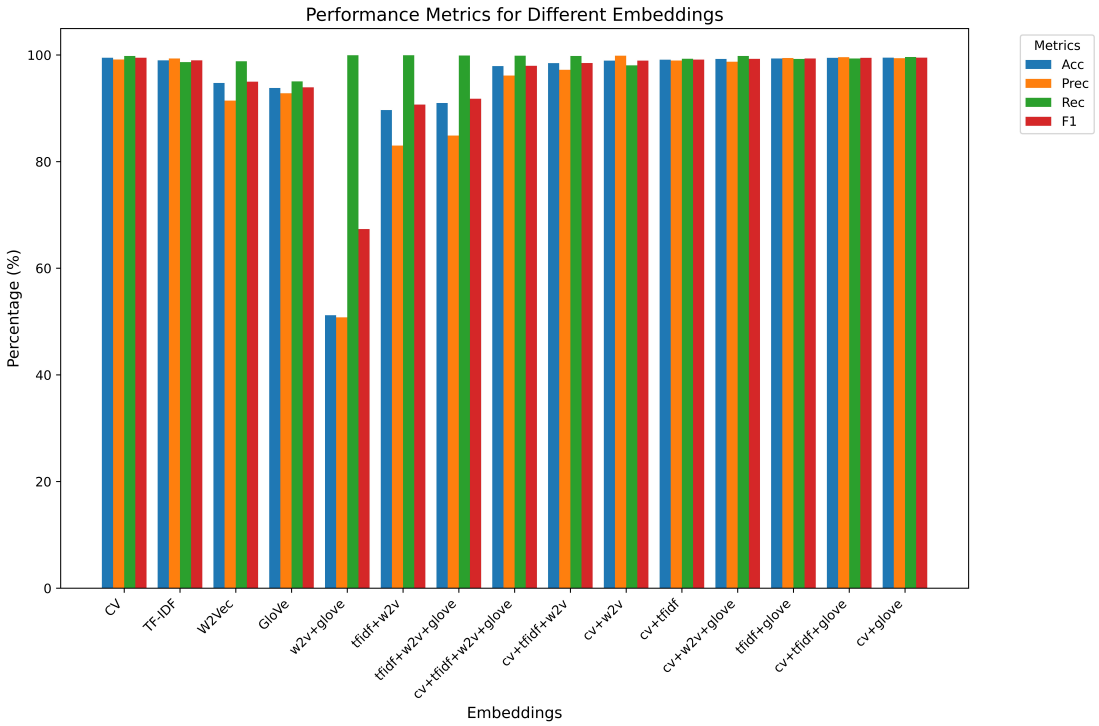

Figure 4.4: Performance comparison of AttCNN model trained with different optimal feature sets.

ding combinations.

We observe that the hybrid methods are computationally heavy and require longer training (076.76 seconds) and testing times (0.401 seconds) for the CV+GloVe combination, as indicated by the evaluation of computational efficiency. This signifies a substantial improvement over single embedding methods and the significant performance increases demonstrate the additional computational cost is worth it in instances precision is paramount. More complex hybrids like: CV+TF-IDF+W2V+GloVe took longer to train(103.42 seconds) without providing a proportional increase in performance, thus, more straightforward hybrid strategies could be practical.

These findings show that appropriately selected embedding combinations from

Table 7: Performance Comparison of Different Embedding Combinations

Embedding	Acc	Prec	Rec	F1	AUC	Spec	Train(s)	Test(s)
CV	99.47	99.15	99.81	99.48	99.95	99.13	4.2727	0.3834
TF-IDF	98.99	99.33	98.66	98.99	99.94	99.32	3.7369	0.3665
W2Vec	94.74	91.43	98.81	94.98	98.74	90.60	0.4761	0.0802
GloVe	93.80	92.82	95.04	93.92	98.21	92.54	2.7705	0.2989
w2v+glove	51.18	50.79	99.95	67.35	71.52	1.65	57.7200	0.3100
tfidf+w2v	89.66	83.00	99.95	90.69	99.82	79.22	80.5200	0.3900
tfidf+w2v+glove	90.99	84.88	99.90	91.78	99.80	81.93	80.1200	0.3900
cv+tfidf+w2v+glove	97.91	96.14	99.86	97.96	99.64	95.93	103.4200	0.5000
cv+tfidf+w2v	98.46	97.21	99.81	98.49	99.97	97.09	103.2900	0.4800
cv+w2v	98.94	99.85	98.04	98.94	99.96	99.85	81.8100	0.3900
cv+tfidf	99.11	98.95	99.28	99.12	99.96	98.93	96.8700	0.4500
cv+w2v+glove	99.25	98.73	99.81	99.26	99.96	98.69	73.5300	0.3600
tfidf+glove	99.33	99.43	99.24	99.33	99.96	99.42	85.1900	0.4400
cv+tfidf+glove	99.45	99.57	99.33	99.45	99.93	99.56	93.2700	0.4300
cv+glove	**99.50**	**99.38**	**99.62**	**99.50**	**99.98**	**99.37**	**76.7600**	**0.4000**

different types can leverage the merits and mitigate the demerits of different embedding methods. Overall, the CV+GloVe merging approach demonstrates a perfect balance between statistical and semantic text representations and enhances false news detection performance as input to the AttCNN architecture. This finding suggests potential pathways for future research in embedding approaches and their subsequent improvement on text categorization tasks.

4.6 Cross-Validation Analysis

Based on the hybrid embedding experiments (see performance in Table 7), we could see that the CV + GloVe vocabulary structure produced the best relative results in accuracy (99.50%) and ROC-AUC (99.98%) since the k-fold CV tests do not change much (20-folds and 30-folds produced practically the same results),

we decided to proceed these tests using 10-fold 20 folds and 30 folds, also to check for model robustness. As in Table 8, the model demonstrated remarkable stability in evaluating all fold settings, in which the 30-fold layout exhibited the highest accuracy (the accuracy of 99.09%), precision (the precision of 99.11%) and ROC-AUC (the ROC-AUC of 99.89%) scores. The components of the confusion matrix performed consistently, showing 30-fold validation with 343.13 true negatives and 343.87 true positives across folds with negligible false positive (3.1) and false negative (3.23) rates.

Table 8: Performance Metrics for k-Fold Cross-Validation

Metric	**10-Fold**	**20-Fold**	**30-Fold**
TN (avg)	1028.9	514.15	343.13
FP (avg)	9.8	5.2	3.1
FN (avg)	11.1	4.85	3.23
TP (avg)	1030.2	515.8	343.87
Accuracy (%)	98.99	99.03	99.09
Precision (%)	99.06	99.00	99.11
Recall (%)	98.93	99.07	99.07
F1-Score (%)	98.99	99.04	99.09
ROC-AUC (%)	99.88	99.85	99.89
Specificity (%)	99.06	99.00	99.11
Train (s)	46.2395	61.117	49.515
Test (s)	0.238	0.2095	0.1489

The CV results from Table 8 also provided intuitive information bearing on computational efficiency for the different fold arrangements. The 20 fold configuration was the slowest, requiring 61.117 seconds to train while the 30 fold

design was the most efficient, needing only 49.515 seconds. The testing durations exhibit a consistent pattern, progressively reducing with the growth of fold numbers—diminishing from 0.238 seconds in 10-fold validation to 0.1489 seconds in 30-fold validation. These results, along with consistently high performance metrics across all fold configurations (ROC-AUC scores in the range of 99.85%99.89%), further substantiate the strong and generalizability of our hybrid embedding approach to the detection of false news.

4.7 Hyperparameter Analysis and Model Optimization

Extensive hyperparameter optimization was performed via grid search cross validation and correlation studies to achieve AttCNN optimality for CV+GloVe embedding. Figure 4 illustrates the results from this investigation into two groups: (A) the distribution of performance measures by class and (B) the sensitivity of key hyperparameters on model accuracy. The results of the best hyperparameter combinations selected through grid search cross-validation for each algorithm are shown in Table 9.

Table 9: Grid Search CV Results for Hyperparameter Tuning

Filters	Kernel Sizes	Dropout	Learning Rate	Batch Size	Accuracy (%)
64	[2, 3, 4]	0.3	0.001	32	98.58
128	[2, 3, 4]	0.3	0.001	32	98.53
64	[2, 3, 4]	0.5	0.001	32	98.51
128	[2, 3, 4]	0.3	0.001	64	98.34
128	[2, 3, 4]	0.5	0.001	32	98.34

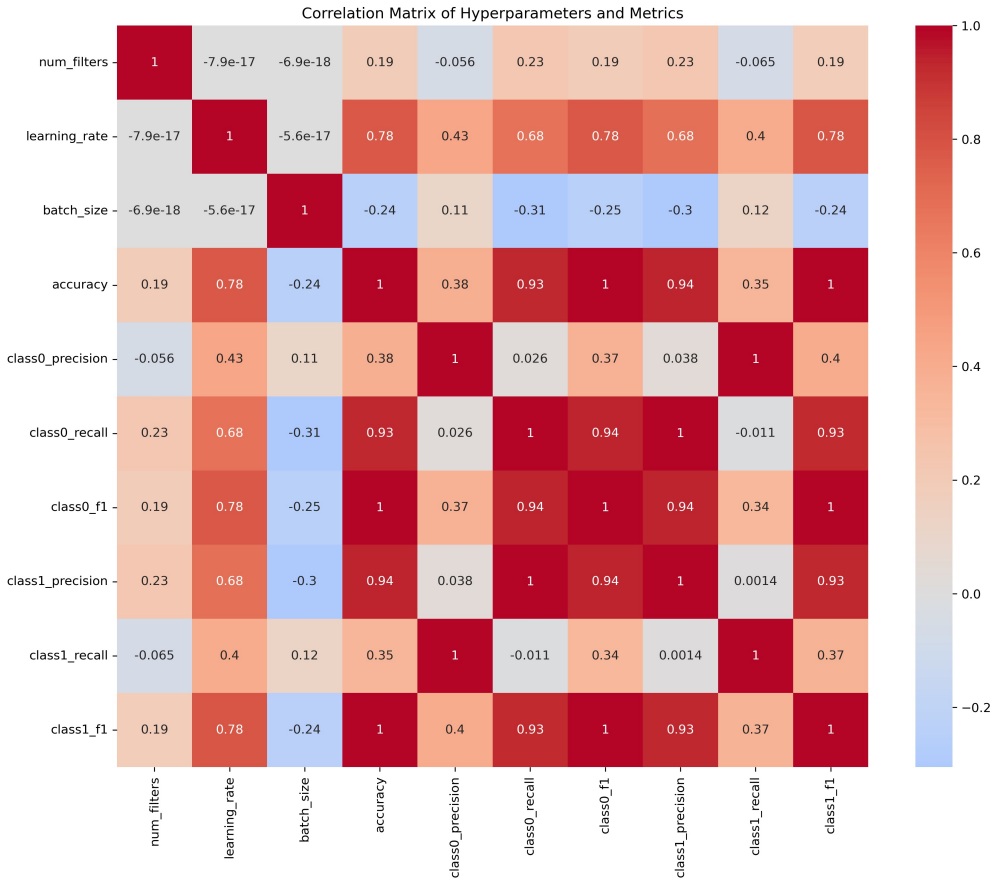

Figure 4.5: A correlation matrix illustrating the correlations between hyperparameters and performance indicators, emphasizing the substantial effect of learning rate on model performance and the modest effect of batch size.

4.7.1 Analysis of Model Performance Distribution

Class 0 (genuine news) and Class 1 (false news) show more homogeneous performance with Class 1 being slightly more stable regarding recall values (median = 0.987) (refer to Fig. 4.6(A)). Both groups present similar F1-scores which demonstrate low interquartile ranges indicating relatively consistent model behavior. This distribution shows that the model is producing strong performance, given the base class imbalance of the data set.

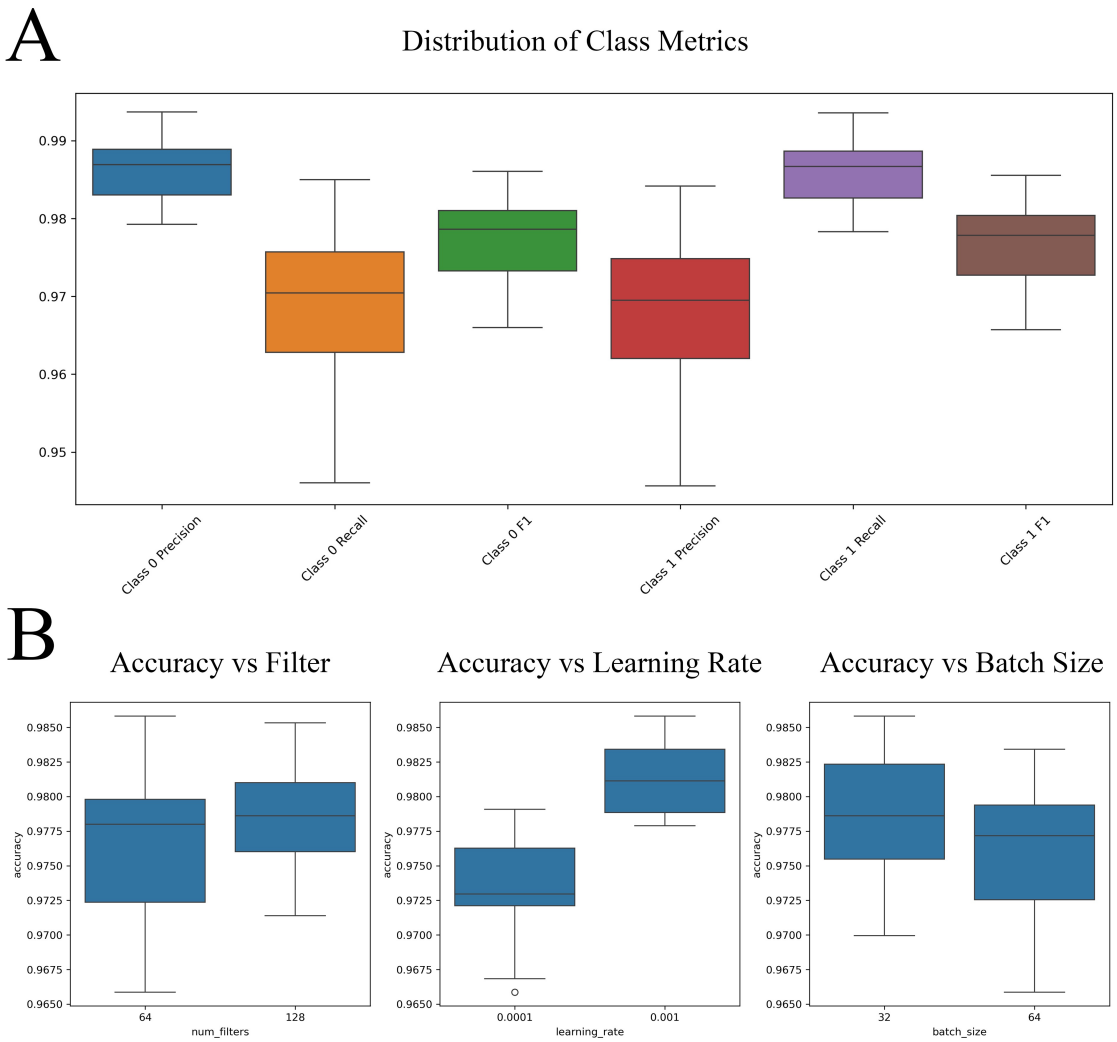

Figure 4.6: (A) Box plot illustrating the distribution of performance metrics for both classes, demonstrating balanced model performance in the categorization of authentic and fraudulent news. (B) The influence of critical hyperparameters (filter quantity, learning rate, and batch size) on model correctness.

4.7.2 Hyperparameter Sensitivity Analysis

The critical hyperparameters' impacts are clearly demonstrated in Figure 4.6(B) using three independent box plots. The explanation suggests that increasing the filter from 64 to 128 brings slight improvement in accuracy (median increase from

0.977 to 0.980). For this model, we can see the learning rate has a much more pronounced effect where 0.001 is not only better but also more stable than 0.0001. Batch Size: Smaller sizes (32) performed marginally better than larger ones (64); more updates result in better optimization dynamics.

4.7.3 Parameter Correlation Analysis

Correlation matrix of hyperparameters and performance indicators is shown in Figure 4.5. The research indicates robust positive correlations between learning rate and model performance metrics (correlation coefficient = 0.78 for accuracy), although batch size exhibits a marginal negative association with most performance measures (coefficient ≈ -0.24 for accuracy). These results informed our hyperparameter optimization approach.

4.7.4 Grid Search Optimization

Grid search results showing the optimal configuration of hyperparameters, with 64 filters, kernel sizes of [2, 3, 4], a dropout of 0.3, a learning rate of 0.001, and a batch size of 32 obtained an accuracy of 98.58% (Table 9). This design achieves the best trade-off between model complexity and computation efficiency as indicated by the small performance drop (0.05%) while increasing the filters to a maximum of 128 while keeping other parameters.

4.8 Comparing AttCNN with existing state-of-the-art methods

To assess the efficacy of our AttCNN technique for detecting false news, we juxtaposed its predictive performance with several leading methodologies, as outlined in

Table 10: Evaluating AttCNN in comparison to the state-of-the-art techniques

Model	F1 Score	Accuracy	Precision	Recall
Naïve Bayes [70]	0.7283	0.7589	0.8293	0.6492
Random Forest [70]	0.8329	0.8437	0.8362	0.8296
Non-memory-based ensemble model [70]	0.8531	0.8639	0.8837	0.8246
NN with Keras [70]	0.8579	0.8674	0.8637	0.8522
Memory-based ensemble model [70]	0.8645	0.8730	0.8617	0.8672
Decision Tree [71]	0.8924	0.8992	0.8610	0.9262
LSTM [72]	0.8930	0.9015	0.9278	0.8607
SVM [70]	0.8974	0.9005	0.9216	0.8744
KNN [72]	0.8978	0.9016	0.8902	0.9055
Naïve Bayes [72]	0.9185	0.9212	0.9145	0.9225
BiLSTM [70]	0.9160	0.9200	0.9189	0.9131
Bagging [72]	0.9500	0.9531	0.9178	0.9846
AdaBoost [72]	0.9502	0.9532	0.9181	0.9846
N-Gram with TF-IDF and LSTM [71]	0.9580	0.9600	0.9550	0.9620
SVM [72]	0.9656	0.9673	0.9460	0.9861
N-Gram with TF-IDF and BERT [71]	0.9630	0.9680	0.9650	0.9700
BiLSTM [48]	0.9748	0.9749	0.9674	0.9841
Att-BiLSTM [48]	0.9762	0.9766	0.9770	0.9767
AttCNN	**0.9950**	**0.9950**	**0.9938**	**0.9962**

Table III. This comparison encompassed many algorithms, including Naïve Bayes, Random Forest, non-memory-based ensemble models, Neural Networks utilizing Keras, memory-based ensemble models, Decision Trees, LSTM, SVM, KNN, BiLSTM, Bagging, AdaBoost, and hybrid models employing N-Gram with TF-IDF and LSTM or BERT. AttCNN had exceptional performance, attaining an F1 score of 0.9950, an accuracy (ACC) of 0.9950, a precision (Pre) of 0.9938, and a sensitivity (Sen) of 0.9962. These measures underscore its efficacy in accurately identifying incidents while reducing false positives and negatives, rendering it a potent instrument for detecting fake news. Conversely, the subsequent highest-performing approach, Att-BiLSTM, attained an F1 score of 0.9762, an accuracy of 0.9766, a precision of 0.9770, and a sensitivity of 0.9767, signifying a significant disparity

in performance. AttCNN surpassed Att-BiLSTM by 0.0147 in F1 score, 0.0145 in accuracy, 0.0122 in precision, and 0.0159 in sensitivity. Alternative models, including BiLSTM and SVM, had markedly lower performance relative to AttCNN, with BiLSTM attaining an F1 score of 0.9748 and SVM reaching an F1 score of 0.9656. AttCNN not only exceeds previous methods in predicting accuracy but also attains an optimal balance between precision and recall, positioning it as a premier solution for false news identification and demonstrating its applicability in addressing disinformation in real-world scenarios.

Chapter 5

Conclusion

This problem of distinguishing real news from fake news was the focus of this study. For this, the researchers developed and evaluated a new architecture, AttCNN that combined multiple feature extraction methods. The hybrid embeddings-based (CV+GloVe) AttCNN model showed outstanding performance with an F1 score of 0.9950, accuracy of 0.9950, precision of 0.9938 and recall of 0.9962, outperforming the both traditional machine learning and advanced deep learning models. We also performed k-fold cross validation on our model to ensure its robustness (setting 10, 20, and 30 folds) Confirming optimal hyperparameters (64 filters, [2 3 4] filters, 0.3 drop out and 0.001 learning rate) through grid search CV. By leveraging the potential of deep learning to enhance the accuracy and reliability of classification, this work signifies a significant advancement towards the detection of fake news. The AttCNN practicality could provide media houses and social media organizations with stronger tools to help safeguard the integrity of information dissemination. Further research might explore fusion of these structures with other advanced models or hybrid multimodal false news detection as a browser plugin to stimulate a continuous pace of evolution in preserving the truthfulness of the information in the digital age.

Bibliography

[1] Redline Digital. (2024) Fake news statistics & facts (2024). Redline Digital. Accessed: January 7, 2025. [Online]. Available: https://redline.digital/fake-news-statistics/

[2] Y. Fang, J. Gao, C. Huang, H. Peng, and R. Wu, "Self multi-head attention-based convolutional neural networks for fake news detection," *PloS one*, vol. 14, no. 9, p. e0222713, 2019.

[3] DataReportal. (2024) Digital around the world. DataReportal – Global Digital Insights. Accessed: 2024. [Online]. Available: https://datareportal.com/global-digital-overview

[4] R. K. Nielsen, N. Newman, R. Fletcher, and A. Kalogeropoulos, "Reuters institute digital news report 2020," 2023.

[5] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[6] Statista. (2024, January) Topic: False news in the U.S. Statista. Accessed: January 7, 2025. [Online]. Available: https://www.statista.com/topics/3251/fake-news/

[7] ——. (2024, July) Topic: False news worldwide. Statista. Accessed: January 7, 2025. [Online]. Available: https://www.statista.com/topics/6341/fake-news-worldwide

[8] L. Yuan, H. Jiang, H. Shen, L. Shi, and N. Cheng, "Sustainable development of information dissemination: A review of current fake news detection research and practice," *Systems*, vol. 11, no. 9, p. 458, 2023.

[9] M. Del Vicario, A. Bessi, F. Zollo, F. Petroni, A. Scala, G. Caldarelli, H. E. Stanley, and W. Quattrociocchi, "The spreading of misinformation online," *Proceedings of the national academy of Sciences*, vol. 113, no. 3, pp. 554–559, 2016.

[10] NSPCC. (2023) Fake news, hoaxes and misinformation. NSPCC. Retrieved January 7, 2025. [Online]. Available: https://www.nspcc.org.uk/keeping-children-safe/online-safety/inappropriate-explicit-content/fake-news/

[11] Internet Matters. (2024) Learn about fake news to support children. Internet Matters. Retrieved January 7, 2025. [Online]. Available: https://www.internetmatters.org/issues/fake-news-and-misinformation-advice-hub/learn-about-fake-news-to-support-children/

[12] J. P. Baptista and A. Gradim, "Understanding fake news consumption: A review," *Social Sciences*, vol. 9, no. 10, p. 185, 2020.

[13] Democratic Schools for All, "Dealing with propaganda, misinformation and fake news," October 2023. [Online]. Avail-

able: https://www.coe.int/en/web/campaign-free-to-speak-safe-to-learn/dealing-with-propaganda-misinformation-and-fake-news

[14] Adminsmaha, "Combating fake news in the digital age," December 2024. [Online]. Available: https://smaislamhidayatullah.sch.id/blog/2024/12/26/combating-fake-news-in-the-digital-age/

[15] D. Medzerian, "Usc study reveals the key reason why fake news spreads on social media," December 2023. [Online]. Available: https://today.usc.edu/usc-study-reveals-the-key-reason-why-fake-news-spreads-on-social-media/

[16] P. Törnberg, "Echo chambers and viral misinformation: Modeling fake news as complex contagion," *PLoS one*, vol. 13, no. 9, p. e0203958, 2018.

[17] S. K. Hamed, M. J. Ab Aziz, and M. R. Yaakub, "A review of fake news detection approaches: A critical analysis of relevant studies and highlighting key challenges associated with the dataset, feature representation, and data fusion," *Heliyon*, 2023.

[18] M. D. Ibrishimova and K. F. Li, "A machine learning approach to fake news detection using knowledge verification and natural language processing," in *Advances in Intelligent Networking and Collaborative Systems: The 11th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2019)*. Springer, 2020, pp. 223–234.

[19] S. Raza and C. Ding, "Fake news detection based on news content and social contexts: a transformer-based approach," *International Journal of Data Science and Analytics*, vol. 13, no. 4, pp. 335–362, 2022.

[20] F. Mohsen, B. Chaushi, H. Abdelhaq, D. Karastoyanova, and K. Wang, "Automated detection of misinformation: A hybrid approach for fake news detection," *Future Internet*, vol. 16, no. 10, p. 352, 2024.

[21] J. Alghamdi, Y. Lin, and S. Luo, "A comparative study of machine learning and deep learning techniques for fake news detection," *Information*, vol. 13, no. 12, p. 576, 2022.

[22] B. S. Abu Nasser and S. S. Abu-Naser, "Leveraging ai for effective fake news detection and verification," *Arab Media & Society*, December 2024. [Online]. Available: https://www.arabmediasociety.com/leveraging-ai-for-effective-fake-news-detection-and-verification/

[23] K. P. Arin, D. Mazrekaj, and M. Thum, "Ability of detecting and willingness to share fake news," *Scientific Reports*, vol. 13, no. 1, p. 7298, 2023.

[24] J. Posetti, "A short guide to the history of 'fake news' and disinformation," *International Center for Journalist*, 2018.

[25] Yellowbrick, "The power of social media in news spreading," March 2024. [Online]. Available: https://www.yellowbrick.co/blog/journalism/the-power-of-social-media-in-news-spreading

[26] Admin_Eticas, "Why and how media curation by algorithm contributes," May 2024. [Online]. Available: https://eticas.ai/why-and-how-media-curation-by-algorithm-contributes/

[27] E. Aïmeur, S. Amri, and G. Brassard, "Fake news, disinformation and mis-

information in social media: a review," *Social Network Analysis and Mining*, vol. 13, no. 1, p. 30, 2023.

[28] J. L. Egelhofer and S. Lecheler, "Fake news as a two-dimensional phenomenon: A framework and research agenda," *Annals of the international communication association*, vol. 43, no. 2, pp. 97–116, 2019.

[29] Hive Mind, "Disinformation and 7 common forms of information disorder," 2025, retrieved January 8, 2025. [Online]. Available: https://en.hive-mind.community/blog/169, disinformation-and-7-common-forms-of-information-disorder

[30] M. Haigh and T. Haigh, *Fighting and Framing Fake News*. The SAGE Handbook of Propaganda, 05 2020, pp. 303–323.

[31] X. Zhao and S. Tsang, "How people process different types of misinformation on social media: A taxonomy based on falsity level and evidence type," *Available at SSRN 4259593*, 2022.

[32] X. Zhou, R. Zafarani, K. Shu, and H. Liu, "Fake news: Fundamental theories, detection strategies and challenges," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 836–837.

[33] X. Zhou and R. Zafarani, "A survey of fake news: Fundamental theories, detection methods, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–40, 2020.

[34] N. N. Prachi, M. Habibullah, M. E. H. Rafi, E. Alam, and R. Khan, "Detection of fake news using machine learning and natural language processing

algorithms [j]," *Journal of Advances in Information Technology*, vol. 13, no. 6, 2022.

[35] S. Maham, A. Tariq, M. U. G. Khan, F. S. Alamri, A. Rehman, and T. Saba, "Ann: adversarial news net for robust fake news classification," *Scientific Reports*, vol. 14, no. 1, p. 7897, 2024.

[36] A. A. A. Ahmed, A. Aljabouh, P. K. Donepudi, and M. S. Choi, "Detecting fake news using machine learning: A systematic literature review," *arXiv preprint arXiv:2102.04458*, 2021.

[37] J. Subramanian, G. J. Sendur, P. Adithiyan, S. P. Kumar, S. M. Kumar, and M. R. Kumar, "Enhancing fake news detection: A comprehensive framework leveraging xgboost and cnns," in *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)*. IEEE, 2024, pp. 1–6.

[38] A. Kesarwani, S. S. Chauhan, and A. R. Nair, "Fake news detection on social media using k-nearest neighbor classifier," in *2020 international conference on advances in computing and communication engineering (ICACCE)*. IEEE, 2020, pp. 1–4.

[39] A. Mahabub, "A robust technique of fake news detection using ensemble voting classifier and comparison with other classifiers," *SN Applied Sciences*, vol. 2, no. 4, p. 525, 2020.

[40] A. Dheyaa Radhi, H. Ali Hussein Al Naffakh, A.-I. Fuqdan, B. A Hakim, and B. Al-Attar, "A comprehensive review of machine learning-based models for fake news detection," in *BIO Web of Conferences*, vol. 97. EDP Sciences, 2024, p. 00123.

[41] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD explorations newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[42] K. Sharma, F. Qian, H. Jiang, N. Ruchansky, M. Zhang, and Y. Liu, "Combating fake news: A survey on identification and mitigation techniques," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 3, pp. 1–42, 2019.

[43] N. K. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the association for information science and technology*, vol. 52, no. 1, pp. 1–4, 2015.

[44] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of economic perspectives*, vol. 31, no. 2, pp. 211–236, 2017.

[45] N. Shakya and P. Poudyal, "Detection of fake news using deep neural networks," *Kathmandu University Journal of Science Engineering and Technology*, vol. 16, no. 2, 2022.

[46] Y. Yang, L. Zheng, J. Zhang, Q. Cui, Z. Li, and P. S. Yu, "Ti-cnn: Convolutional neural networks for fake news detection," *arXiv preprint arXiv:1806.00749*, 2018.

[47] K. L. Tan, C. P. Lee, and K. M. Lim, "Fake news detection with hybrid cnn-lstm," in *2021 9th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2021, pp. 606–610.

[48] H. Padalko, V. Chomko, and D. Chumachenko, "A novel approach to fake

news classification using lstm-based deep learning models," *Frontiers in big Data*, vol. 6, p. 1320800, 2024.

[49] T. Jiang, J. P. Li, A. U. Haq, and A. Saboor, "Fake news detection using deep recurrent neural networks," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2020, pp. 205–208.

[50] H. Zulkifli, "Word vectors and lexical semantics — introduction to count-based vectors," *Medium*, December 2021. [Online]. Available: https://towardsdatascience.com/word-vectors-and-lexical-semantics-part-1-c9dbc8932c1d

[51] GeeksforGeeks, "Understanding TFIDF (Term Frequency-Inverse Document Frequency)," January 2023. [Online]. Available: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

[52] A. Patel and K. Meehan, "Fake news detection on reddit utilising countvectorizer and term frequency-inverse document frequency with logistic regression, multinominalnb and support vector machine," in *2021 32nd Irish signals and systems conference (ISSC)*. IEEE, 2021, pp. 1–6.

[53] Wikipedia contributors, "Word2Vec," December 2024, retrieved January 9, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Word2vec

[54] I. L. O. Bolgurtseva, "Word2VEC: Why do we need word representations?" May 2023. [Online]. Available: https://serokell.io/blog/word2vec

[55] L. Abualigah, Y. Y. Al-Ajlouni, M. S. Daoud, M. Altalhi, and H. Migdady,

"Fake news detection using recurrent neural network based on bidirectional lstm and glove," *Social Network Analysis and Mining*, vol. 14, no. 1, p. 40, 2024.

[56] T. Shahi, C. Sitaula, and N. Paudel, "A hybrid feature extraction method for nepali covid-19-related tweets classification," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, p. 5681574, 2022.

[57] H. Jeong, *Hierarchical Attention Networks for Fake News Detection*. The Florida State University, 2021.

[58] S. Ramya and R. Eswari, "Attention-based deep learning models for detection of fake news in social networks," *International Journal of cognitive informatics and natural intelligence (IJCINI)*, vol. 15, no. 4, pp. 1–25, 2021.

[59] M.-A. Ouassil, B. Cherradi, S. Hamida, M. Errami, O. EL GANNOUR, and A. Raihani, "A fake news detection system based on combination of word embedded techniques and hybrid deep learning model," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 10, 2022.

[60] A. Altamimi, "Novel approach for predicting fake news stance detection using large word embedding blending and customized cnn model," *PloS one*, vol. 19, no. 12, p. e0314174, 2024.

[61] A. Farha *et al.*, "Fake news detection using machine learning: An exhaustive review," *Fake News Detection Using Machine Learning: An Exhaustive Review (April 26, 2023)*, 2023.

[62] J. Ogunsuyi Opeyemi and K. Adebola, "K-nearest neighbors bayesian approach to false news detection from text on social media," *Int. J. Educ. Manag. Eng*, vol. 12, pp. 22–32, 2022.

[63] L. Aslan, M. Ptaszynski, and J. Jauhiainen, "Are strong baselines enough? false news detection with machine learning," *Future Internet*, vol. 16, no. 9, p. 322, 2024.

[64] S. J. Johnson, M. R. Murty, and I. Navakanth, "A detailed review on word embedding techniques with emphasis on word2vec," *Multimedia Tools and Applications*, vol. 83, no. 13, pp. 37 979–38 007, 2024.

[65] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[66] M. Kanahuati-Ceballos and L. J. Valdivia, "Detection of depressive comments on social media using rnn, lstm, and random forest: comparison and optimization," *Social Network Analysis and Mining*, vol. 14, no. 1, p. 44, 2024.

[67] T. Tong-Ern, H. Su-Cheng, N. Kok-Why, M. Al-Tarawneh, and T. Gee-Kok, "Performance evaluation on resolution time prediction using machine learning techniques," *JOIV: International Journal on Informatics Visualization*, vol. 8, no. 2, pp. 583–591, 2024.

[68] J. Zhang, Y. Li, F. Shen, Y. He, H. Tan, and Y. He, "Hierarchical text classification with multi-label contrastive learning and knn," *Neurocomputing*, vol. 577, p. 127323, 2024.

[69] P. Carmona, F. Climent, and A. Momparler, "Predicting failure in the us banking sector: An extreme gradient boosting approach," *International Review of Economics & Finance*, vol. 61, pp. 304–323, 2019.

[70] V. S. Nirban, T. Shukla, P. S. Purkayastha, N. Kotalwar, and L. Ahsan, "The role of ai in combating fake news and misinformation," in *International Conference on Innovations in Bio-Inspired Computing and Applications*. Springer, 2022, pp. 690–701.

[71] N. Kausar, A. AliKhan, and M. Sattar, "Towards better representation learning using hybrid deep learning model for fake news detection," *Social Network Analysis and Mining*, vol. 12, no. 1, p. 165, 2022.

[72] P. K. Verma, P. Agrawal, I. Amorim, and R. Prodan, "Welfake: word embedding over linguistic features for fake news detection," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 4, pp. 881–893, 2021.

Appendix A

Supplementary Materials

A.1 Necessary Links and Files

- GitHub Repository For Codes:

  https://github.com/rudradcruze/Fake-News-Thesis-BSc

- ALL Material Figure, Plots, Tables, and Dataset:

  https://cutt.ly/fake-news-thesis-bsc-rudra

A.2 All Plots & Figures

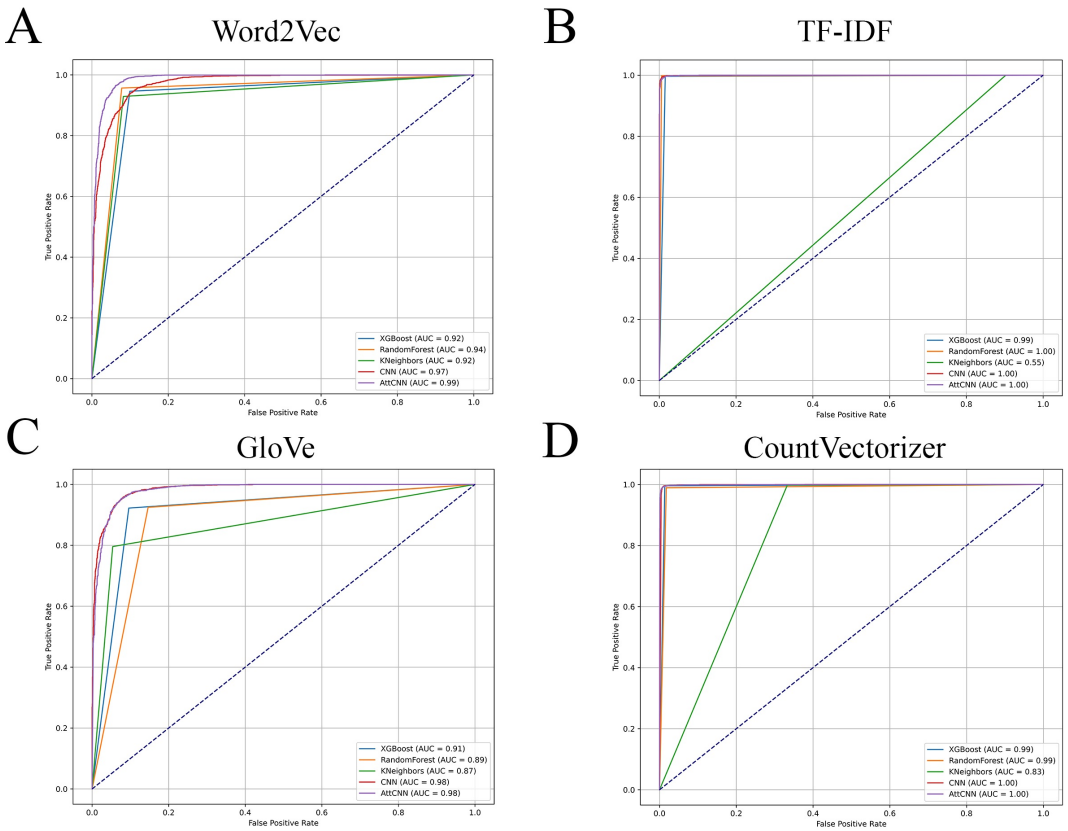

Figure A.1: For 80 : 20 splitting ratio single view feature ROC Curve for all model

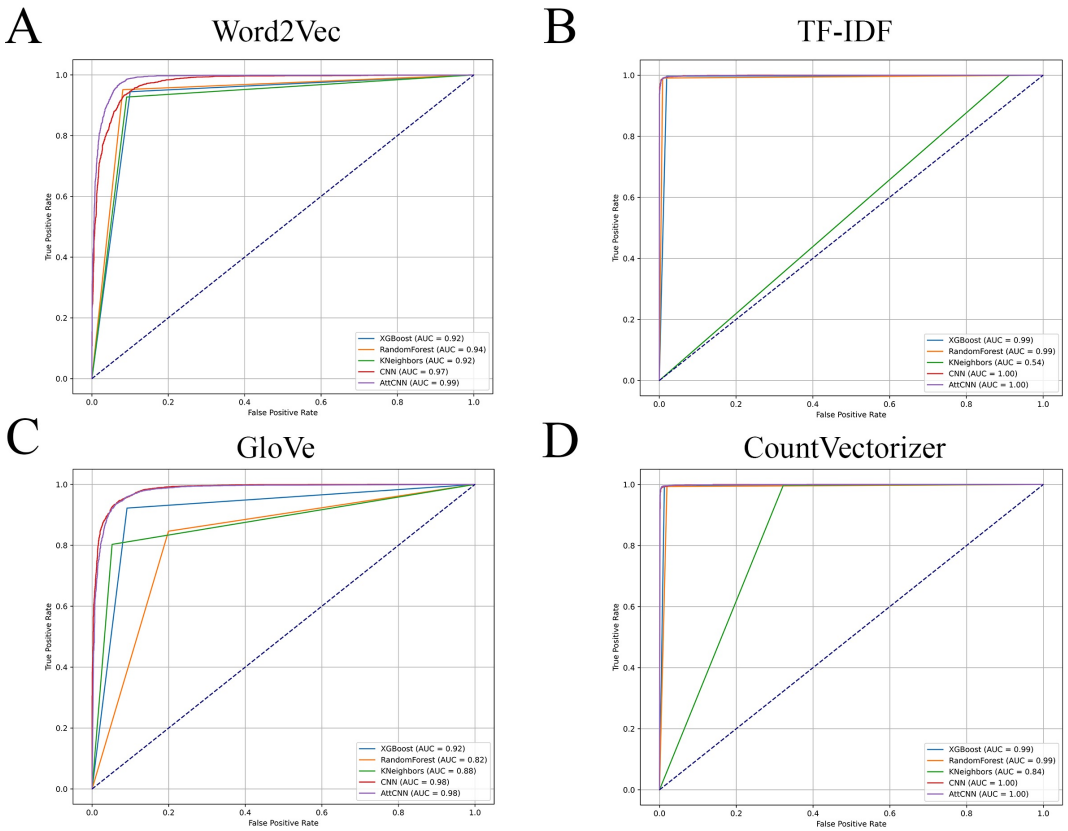

Figure A.2: For 70 : 30 splitting ratio single view feature ROC Curve for all model

Appendix B

Code Materials

B.1 Python Code for ALL Models with Single View Features

B.1.1 Python Code for ALL Models with CountVectorizer

```
# Importing required libraries

import re
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import (confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score, roc_auc_score,
    matthews_corrcoef, classification_report, mean_absolute_error,
    mean_squared_error, log_loss, cohen_kappa_score, roc_curve)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,
    AdaBoostClassifier
from xgboost import XGBClassifier
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.layers import (Dense, Dropout, Conv1D,
    GlobalMaxPooling1D, Attention, Input, BatchNormalization,
    Reshape)
```

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
import time
import matplotlib.pyplot as plt

# Load and Preprocess Data

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Define a function for text preprocessing (stemming and cleaning)
def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = text.lower()
    text = text.split()
    ps = PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text if not word in
        set(stopwords.words('english'))])
    return text

# Load dataset (adjust path as per your file location)
news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')
news_dataset = news_dataset.fillna('')

# Combine author and title into content
news_dataset['content'] = news_dataset['author'] + ' ' +
    news_dataset['title']
news_dataset['content'] = news_dataset['content'].apply(
    preprocess_text)

# Split dataset into train and test sets
X = news_dataset['content']
y = news_dataset['label']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.4, andom_state=101)

# Vectorization using Count Vectorizer
count_vectorizer = CountVectorizer(max_features=5000)
X_train_cv = count_vectorizer.fit_transform(X_train).toarray()
```

```
X_test_cv = count_vectorizer.transform(X_test).toarray()

def evaluate_model(model, X_train, X_test, y_train, y_test,
    threshold=0.5, model_name=""):
    metrics = {}

    # Measure training time
    start_time = time.time()
    model.fit(X_train, y_train)
    metrics['training_time'] = time.time() - start_time

    # Measure testing time
    start_time = time.time()
    y_pred = model.predict(X_test)
    metrics['testing_time'] = time.time() - start_time

    # For CNN: y_pred already represents probabilities, so we
       apply threshold to get binary predictions
    y_pred_binary = (y_pred > threshold).astype(int)  # Convert to
        binary (0 or 1)
    y_pred_proba = y_pred  # Probabilities (continuous values)

    # Classification report
    print("Classification report:")
    classification_report_str = classification_report(y_test,
       y_pred_binary)
    print(classification_report_str)

    # Metrics
    cm = confusion_matrix(y_test, y_pred_binary)
    TN, FP, FN, TP = cm.ravel()
    metrics.update({
        "accuracy": accuracy_score(y_test, y_pred_binary),
        "precision": precision_score(y_test, y_pred_binary),
        "recall": recall_score(y_test, y_pred_binary),
        "f1_score": f1_score(y_test, y_pred_binary),
        "roc_auc": roc_auc_score(y_test, y_pred_proba),
        "specificity": TN / (TN + FP),
        "false_positive_rate": FP / (FP + TN),
        "false_negative_rate": FN / (FN + TP),
        "negative_predictive_value": TN / (TN + FN),
        "false_discovery_rate": FP / (FP + TP),
```

```
94          "mean_absolute_error": mean_absolute_error(y_test,
               y_pred_binary),
95          "mean_squared_error": mean_squared_error(y_test,
               y_pred_binary),
96          "root_mean_squared_error": np.sqrt(mean_squared_error(
               y_test, y_pred_binary)),
97          "log_loss": log_loss(y_test, y_pred_proba),
98          "cohen_kappa": cohen_kappa_score(y_test, y_pred_binary)
99      })
100
101     file_name = f"{model_name}_evaluation_cv-2.txt"
102
103     with open(file_name, "w") as file:
104         file.write(f"{model_name} Evaluation\n")
105         file.write("Classification Report:\n")
106         file.write(classification_report_str + "\n\n")
107         file.write("Metrics:\n")
108         for key, value in metrics.items():
109             file.write(f"{key}: {value}\n")
110
111     # Print metrics
112     for key, value in metrics.items():
113         print(f"{key}: {value}")
114
115     # Store ROC curve data
116     fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
117     return metrics, fpr, tpr
118
119 # Model Evaluation
120
121 # XGBoost
122 xgb_model = XGBClassifier(random_state=101)
123 xgb_metrics, xgb_fpr, xgb_tpr = evaluate_model(xgb_model,
       X_train_cv, X_test_cv, y_train, y_test, model_name="XGBoost")
124
125 # RandomForest
126 rf_model = RandomForestClassifier(n_estimators=5, random_state
       =101)
127 rf_metrics, rf_fpr, rf_tpr = evaluate_model(rf_model, X_train_cv,
       X_test_cv, y_train, y_test, model_name="RandomForest")
128
129 # KNN
```

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_metrics, knn_fpr, knn_tpr = evaluate_model(knn_model,
    X_train_cv, X_test_cv, y_train, y_test, model_name="KNeighbors"
    )

# CNN
# Define Dense model (removing Conv1D)
input_layer = Input(shape=(X_train_cv.shape[1],))  # Shape for
    CountVectorizer input
dense_layer_1 = Dense(128, activation='relu')(input_layer)
dropout_layer_1 = Dropout(0.5)(dense_layer_1)
dense_layer_2 = Dense(64, activation='relu')(dropout_layer_1)
dropout_layer_2 = Dropout(0.5)(dense_layer_2)
output_layer = Dense(1, activation='sigmoid')(dropout_layer_2)

model_cnn = Model(inputs=input_layer, outputs=output_layer)

model_cnn.compile(optimizer=Adam(learning_rate=0.005), loss='
    binary_crossentropy', metrics=['accuracy'])

print("\nTraining Dense Network with CountVectorizer Features...")
history = model_cnn.fit(X_train_cv, y_train, epochs=20, batch_size
    =32, validation_data=(X_test_cv, y_test), verbose=1)
cnn_metrics, cnn_fpr, cnn_tpr = evaluate_model(model_cnn,
    X_train_cv, X_test_cv, y_train, y_test, model_name="CNN")

# AttCNN
# Define CNN with Attention
input_layer = Input(shape=(X_train_cv.shape[1],))
dense_layer = Dense(128, activation='relu')(input_layer)
dense_layer = Dense(64, activation='relu')(dense_layer)

# Expand dimensions for Conv1D
expanded_dense_layer = layers.Reshape((-1, 1))(dense_layer)

conv_layer = Conv1D(filters=128, kernel_size=5, activation='relu')
    (expanded_dense_layer)
attention_data = Conv1D(filters=128, kernel_size=3, activation='
    relu')(expanded_dense_layer)
attention_layer = Attention()([conv_layer, attention_data])

flatten_layer = GlobalMaxPooling1D()(attention_layer)
```

```
dropout_layer = Dropout(0.5)(flatten_layer)
output_layer = Dense(1, activation='sigmoid')(dropout_layer)

model_attention = Model(inputs=input_layer, outputs=output_layer)

model_attention.compile(optimizer=Adam(learning_rate=0.001), loss=
    'binary_crossentropy', metrics=['accuracy'])

# Train the model
print("\nTraining CNN with Attention and CountVectorizer Features
    ...")
history_attention = model_attention.fit(X_train_cv, y_train,
    epochs=20, batch_size=64, validation_data=(X_test_cv, y_test),
    verbose=1)
att_cnn_metrics, att_cnn_fpr, att_cnn_tpr = evaluate_model(
    model_attention, X_train_cv, X_test_cv, y_train, y_test,
    model_name="AttCNN")

# RCO Plot
def plot_roc_curves(models_roc_data):
    plt.figure(figsize=(10, 8))
    for model_name, roc_data in models_roc_data.items():
        fpr, tpr = roc_data['fpr'], roc_data['tpr']
        auc_score = roc_data['auc']
        plt.plot(fpr, tpr, label=f"{model_name} (AUC = {auc_score
            :.2f})")

    plt.plot([0, 1], [0, 1], color="navy", linestyle="--")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC Curve")
    plt.legend(loc="lower right")
    plt.grid(True)
    plt.savefig('ROC_Curves_Comparison_CV_60_40.png', dpi=1000)
    plt.show()

# Storing metrics for future ROC plotting
roc_data = {
    "XGBoost": {"fpr": xgb_fpr, "tpr": xgb_tpr, "auc": xgb_metrics
        ["roc_auc"]},
    "RandomForest": {"fpr": rf_fpr, "tpr": rf_tpr, "auc":
        rf_metrics["roc_auc"]},
```

```
    "KNeighbors": {"fpr": knn_fpr, "tpr": knn_tpr, "auc":
        knn_metrics["roc_auc"]},
    "CNN": {"fpr": cnn_fpr, "tpr": cnn_tpr, "auc": cnn_metrics["
        roc_auc"]},
    "AttCNN": {"fpr": att_cnn_fpr, "tpr": att_cnn_tpr, "auc":
        att_cnn_metrics["roc_auc"]}
}

# Plot ROC curves for all models
plot_roc_curves(roc_data)
```

B.1.2 Python Code for ALL Models with TF-IDF

```
# Importing Libraries

import re
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import (confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score, roc_auc_score,
    classification_report, mean_absolute_error, mean_squared_error,
     log_loss, cohen_kappa_score, roc_curve)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from tensorflow.keras import layers
from tensorflow.keras.layers import (Dense, Dropout, Conv1D,
    GlobalMaxPooling1D, Attention, Input, BatchNormalization,
    Reshape)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
import time
import matplotlib.pyplot as plt

# Download NLTK resources
nltk.download('punkt')
```

```
nltk.download('stopwords')

# Define a function for text preprocessing (stemming and cleaning)
def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = text.lower()
    text = text.split()
    ps = PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text if not word in
        set(stopwords.words('english'))])
    return text

# Load dataset (adjust path as per your file location)
news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')
news_dataset = news_dataset.fillna('')

# Combine author and title into content
news_dataset['content'] = news_dataset['author'] + ' ' +
    news_dataset['title']
news_dataset['content'] = news_dataset['content'].apply(
    preprocess_text)

# Split dataset into train and test sets
X = news_dataset['content']
y = news_dataset['label']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.4, random_state=101)
```

B.1.3 Python Code for ALL Models with GloVe

```
import re
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.metrics import (confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score, roc_auc_score,
```

```
    classification_report, mean_absolute_error, mean_squared_error,
     log_loss, cohen_kappa_score, roc_curve)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from tensorflow.keras import layers
from tensorflow.keras.layers import (Dense, Dropout, Conv1D,
    GlobalMaxPooling1D, Attention, Input, BatchNormalization,
    Reshape)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
import time
import matplotlib.pyplot as plt
from numpy.linalg import norm
from tqdm import tqdm

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Define a function for text preprocessing (stemming and cleaning)
def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = text.lower()
    text = text.split()
    ps = PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text if not word in
        set(stopwords.words('english'))])
    return text

# Load dataset (adjust path as per your file location)
news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')
news_dataset = news_dataset.fillna('')

# Combine author and title into content
news_dataset['content'] = news_dataset['author'] + ' ' +
    news_dataset['title']
news_dataset['content'] = news_dataset['content'].apply(
    preprocess_text)

def load_glove_embeddings(glove_file, dimension=100):
    """
```

```
    Loads GloVe embeddings from a local file
    """
    print(f"Loading {dimension}-dimensional GloVe embeddings from
       {glove_file}...")
    embeddings = {}

    with open(glove_file, 'r', encoding='utf-8') as f:
        for line in tqdm(f):
            values = line.split()
            word = values[0]
            vector = np.asarray(values[1:], dtype='float32')
            embeddings[word] = vector

    print(f"Loaded {len(embeddings)} word vectors.")
    return embeddings

def text_to_glove_vector(text, embeddings, dimension=100):
    """
    Convert text to average GloVe vector
    """
    words = text.split()
    word_vectors = []

    for word in words:
        if word in embeddings:
            word_vectors.append(embeddings[word])

    if word_vectors:
        # Calculate average vector
        vector = np.mean(word_vectors, axis=0)
        # Normalize the vector
        if norm(vector) > 0:
            vector = vector / norm(vector)
        return vector
    else:
        return np.zeros(dimension)

glove_file = '../Dataset/glove.6B.100d.txt'
glove_embeddings = load_glove_embeddings(glove_file)

X = np.array([text_to_glove_vector(text, glove_embeddings)
              for text in tqdm(news_dataset['content'])])
```

```
y = news_dataset['label']

X_train_g2vec, X_test_g2vec, y_train, y_test = train_test_split(X,
    y, test_size=0.4, random_state=101)
```

B.1.4 Python Code for ALL Models with Word2vec

```
import re
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from sklearn.metrics import (confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score, roc_auc_score,
    classification_report, mean_absolute_error, mean_squared_error,
     log_loss, cohen_kappa_score, roc_curve)
from sklearn.utils.class_weight import compute_class_weight
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from tensorflow.keras import layers
from tensorflow.keras.layers import (Dense, Dropout, Conv1D,
    GlobalMaxPooling1D, Attention, Input, BatchNormalization,
    Reshape, Input, GlobalAveragePooling1D, Add, Concatenate,
    Attention, Concatenate)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping,
    ModelCheckpoint, LearningRateScheduler
from tensorflow.keras.metrics import AUC
import time
import matplotlib.pyplot as plt
import math

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

```

```
# Define a function for text preprocessing (stemming and cleaning)
def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = text.lower()
    text = text.split()
    ps = PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text if not word in
        set(stopwords.words('english'))])
    return text

# Load dataset (adjust path as per your file location)
news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')
news_dataset = news_dataset.fillna('')

# Combine author and title into content
news_dataset['content'] = news_dataset['author'] + ' ' +
    news_dataset['title']
news_dataset['content'] = news_dataset['content'].apply(
    preprocess_text)

# Tokenize text for Word2Vec training
tokenized_text = [text.split() for text in news_dataset['content'
    ]]

# Train Word2Vec model
wv_model = Word2Vec(sentences=tokenized_text, vector_size=200,
    window=5, min_count=1, workers=4)

# Function to average Word2Vec vectors for a text
def word_averaging(wv_model, words):
    mean = np.zeros((wv_model.vector_size,))
    count = 0.
    for word in words:
        if word in wv_model.wv:
            mean += wv_model.wv[word]
            count += 1.
    if count != 0:
        mean /= count
    return mean

# Convert each document into average Word2Vec vectors
X = np.array([word_averaging(wv_model, words) for words in
```

```
    tokenized_text])
y = news_dataset['label']

X_train_word2vec, X_test_word2vec, y_train, y_test =
    train_test_split(X, y, test_size=0.4, random_state=101)
```

B.2 Python Code for AttCNN Model with Hybrid Features

```
import re
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from sklearn.feature_extraction.text import CountVectorizer,
    TfidfVectorizer
from sklearn.metrics import (confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score, roc_auc_score,
    classification_report, log_loss, cohen_kappa_score, roc_curve)
from tensorflow.keras.layers import (Dense, Dropout, Conv1D,
    GlobalMaxPooling1D, Attention, Input, BatchNormalization,
    Reshape)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
import time
import matplotlib.pyplot as plt
import os

# Text Preprocessor
class TextPreprocessor:
    def __init__(self):
        nltk.download('punkt')
        nltk.download('stopwords')
        self.ps = PorterStemmer()
        self.stop_words = set(stopwords.words('english'))

    def preprocess_text(self, text):
        text = re.sub('[^a-zA-Z]', ' ', text)
```

```
        text = text.lower()
        text = text.split()
        text = [self.ps.stem(word) for word in text if word not in
            self.stop_words]
        return ' '.join(text)

# Embedding Generator
class EmbeddingGenerator:
    def __init__(self, max_features=5000):
        self.cv = CountVectorizer(max_features=max_features)
        self.tfidf = TfidfVectorizer(max_features=max_features)
        self.w2v_model = None
        self.glove_model = None
        self.max_features = max_features

    def fit_transform_cv(self, texts):
        return self.cv.fit_transform(texts).toarray()

    def transform_cv(self, texts):
        return self.cv.transform(texts).toarray()

    def fit_transform_tfidf(self, texts):
        return self.tfidf.fit_transform(texts).toarray()

    def transform_tfidf(self, texts):
        return self.tfidf.transform(texts).toarray()

    def train_word2vec(self, texts, vector_size=100):
        tokenized_texts = [text.split() for text in texts]
        self.w2v_model = Word2Vec(sentences=tokenized_texts,
            vector_size=vector_size,
                                  window=5, min_count=1, workers=4)
        return self._get_document_vectors(texts, self.w2v_model)

    def load_glove(self):
        self.glove_model = {}
        with open('../Dataset/glove.6B.100d.txt', 'r', encoding='
            utf-8') as f:
            for line in f:
                values = line.split()
                word = values[0]
                vector = np.asarray(values[1:], dtype='float32')
```

```
                self.glove_model[word] = vector

    def get_glove_vectors(self, texts):
        if self.glove_model is None:
            self.load_glove()
        return self._get_document_vectors(texts, self.glove_model)

    def _get_document_vectors(self, texts, model):
        doc_vectors = []
        for text in texts:
            words = text.split()
            word_vectors = []
            for word in words:
                try:
                    vector = model.wv[word] if hasattr(model, 'wv'
                        ) else model[word]
                    word_vectors.append(vector)
                except KeyError:
                    continue
            if word_vectors:
                doc_vectors.append(np.mean(word_vectors, axis=0))
            else:
                doc_vectors.append(np.zeros(model.vector_size if
                    hasattr(model, 'vector_size')
                                           else len(model['the'])))
        return np.array(doc_vectors)

# Attention CNN Model
class AttCNNModel:
    def __init__(self, input_shape):
        self.model = self._build_model(input_shape)

    def _build_model(self, input_shape):
        input_layer = Input(shape=(input_shape,))

        # Dense layers
        dense_layer = Dense(128, activation='relu')(input_layer)
        dense_layer = Dense(64, activation='relu')(dense_layer)

        # Reshape layer for Conv1D
        reshape_layer = Reshape((-1, 1))(dense_layer)

```

```
107         # Convolutional layers
108         conv_layer = Conv1D(filters=128, kernel_size=5, padding='
                same', activation='relu')(reshape_layer)
109         attention_data = Conv1D(filters=128, kernel_size=3,
                padding='same', activation='relu')(reshape_layer)
110 
111         # Attention mechanism
112         attention_layer = Attention()([conv_layer, attention_data
                ])
113 
114         # Output layers
115         flatten_layer = GlobalMaxPooling1D()(attention_layer)
116         dropout_layer = Dropout(0.5)(flatten_layer)
117         output_layer = Dense(1, activation='sigmoid')(
                dropout_layer)
118 
119         # Create and compile model
120         model = Model(inputs=input_layer, outputs=output_layer)
121         model.compile(optimizer=Adam(learning_rate=0.0005),
122                       loss='binary_crossentropy',
123                       metrics=['accuracy'])
124         return model
125 
126 # Model Evaluator
127 class ModelEvaluator:
128     @staticmethod
129     def evaluate_model(model, X_train, X_test, y_train, y_test,
            threshold=0.5, model_name=""):
130         metrics = {}
131 
132         # Training time
133         start_time = time.time()
134         history = model.fit(X_train, y_train, epochs=20,
                batch_size=32,
135                             validation_data=(X_test, y_test),
                                verbose=1)
136         metrics['training_time'] = time.time() - start_time
137 
138         # Testing time
139         start_time = time.time()
140         y_pred = model.predict(X_test)
141         metrics['testing_time'] = time.time() - start_time
```

```
142
143         y_pred_binary = (y_pred > threshold).astype(int)
144
145         # Calculate metrics
146         cm = confusion_matrix(y_test, y_pred_binary)
147         TN, FP, FN, TP = cm.ravel()
148
149         metrics.update({
150             "Accuracy": accuracy_score(y_test, y_pred_binary),
151             "Precision": precision_score(y_test, y_pred_binary),
152             "Recall": recall_score(y_test, y_pred_binary),
153             "F1-Score": f1_score(y_test, y_pred_binary),
154             "ROC-AUC": roc_auc_score(y_test, y_pred),
155             "Specificity": TN / (TN + FP),
156             "False Positive Rate": FP / (FP + TN),
157             "False Negative Rate": FN / (FN + TP),
158             "Log Loss": log_loss(y_test, y_pred),
159             "Cohen's Kappa": cohen_kappa_score(y_test,
                    y_pred_binary)
160         })
161
162         # Save metrics to file
163         file_name = f"results/{model_name}_metrics.txt"
164         os.makedirs('results', exist_ok=True)
165
166         with open(file_name, 'w') as f:
167             f.write(f"Performance Metrics for {model_name}
                    Embedding Technique\n")
168             f.write("=" * 50 + "\n\n")
169
170             # Model Architecture
171             f.write("Model Architecture:\n")
172             f.write("-" * 20 + "\n")
173             model.summary(print_fn=lambda x: f.write(x + '\n'))
174             f.write("\n")
175
176             # Training Information
177             f.write("Training Information:\n")
178             f.write("-" * 20 + "\n")
179             f.write(f"Training Time: {metrics['training_time']:.2f
                    } seconds\n")
180             f.write(f"Testing Time: {metrics['testing_time']:.2f}
```

```
                seconds\n")
181             f.write(f"Input Shape: {X_train.shape}\n\n")
182
183             # Performance Metrics
184             f.write("Performance Metrics:\n")
185             f.write("-" * 20 + "\n")
186             for metric, value in metrics.items():
187                 if metric not in ['training_time', 'testing_time'
                        ]:
188                     f.write(f"{metric}: {value:.4f}\n")
189             f.write("\n")
190
191             # Confusion Matrix
192             f.write("Confusion Matrix:\n")
193             f.write("-" * 20 + "\n")
194             f.write("[[TN  FP]\n")
195             f.write(f" [FN  TP]]\n")
196             f.write(f"{cm}\n\n")
197
198             # Classification Report
199             f.write("Classification Report:\n")
200             f.write("-" * 20 + "\n")
201             f.write(classification_report(y_test, y_pred_binary))
202
203         # Get ROC curve data
204         fpr, tpr, _ = roc_curve(y_test, y_pred)
205
206         return metrics, fpr, tpr, history
207
208 # Combine Embeddings
209 def combine_embeddings(*embeddings):
210     """Combines multiple embedding matrices horizontally."""
211     return np.concatenate(embeddings, axis=1)
212
213 # Plot ROC Curves
214 def plot_roc_curves(results_dict):
215     plt.figure(figsize=(12, 8))
216     for name, data in results_dict.items():
217         plt.plot(data['fpr'], data['tpr'],
218                  label=f"{name} (AUC = {data['metrics']['ROC-AUC
                         ']:.3f})")
219
```

```
    plt.plot([0, 1], [0, 1], 'k--', label='Random')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curves for Different Embedding Combinations')
    plt.legend(loc='lower right')
    plt.grid(True)

    # Save with different formats
    os.makedirs('results/plots', exist_ok=True)
    plt.savefig('results/plots/roc_curves_comparison.png', dpi
        =1000, bbox_inches='tight')
    plt.savefig('results/plots/roc_curves_comparison.pdf', dpi
        =1000, bbox_inches='tight')
    plt.show()
    plt.close()

def main():
    # Load and preprocess data
    preprocessor = TextPreprocessor()
    news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')
    news_dataset = news_dataset.fillna('')
    news_dataset['content'] = news_dataset['author'] + ' ' +
        news_dataset['title']
    news_dataset['content'] = news_dataset['content'].apply(
        preprocessor.preprocess_text)

    # Split dataset
    X = news_dataset['content']
    y = news_dataset['label']
    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2, random_state=101)

    # Initialize embedding generator
    embedding_gen = EmbeddingGenerator()

    # Generate all embeddings
    embeddings = {
        'cv': (embedding_gen.fit_transform_cv(X_train),
               embedding_gen.transform_cv(X_test)),
        'tfidf': (embedding_gen.fit_transform_tfidf(X_train),
                  embedding_gen.transform_tfidf(X_test)),
        'w2v': (embedding_gen.train_word2vec(X_train),
```

```
                embedding_gen.train_word2vec(X_test)),
        'glove': (embedding_gen.get_glove_vectors(X_train),
                  embedding_gen.get_glove_vectors(X_test))
    }

    # Define embedding combinations to test
    combinations = [
        # Pairwise
        [('cv', 'tfidf'), ('cv', 'w2v'), ('cv', 'glove'),
         ('tfidf', 'w2v'), ('tfidf', 'glove'), ('w2v', 'glove')],
        # Triple
        [('cv', 'tfidf', 'w2v'), ('cv', 'tfidf', 'glove'),
         ('cv', 'w2v', 'glove'), ('tfidf', 'w2v', 'glove')],
        # All four
        [('cv', 'tfidf', 'w2v', 'glove')]
    ]

    results = {}

    # Test each combination
    for combo_group in combinations:
        for combo in combo_group:
            name = '+'.join(combo)
            print(f"\nTesting combination: {name}")

            # Combine embeddings
            X_train_combined = combine_embeddings(*[embeddings[e
                ][0] for e in combo])
            X_test_combined = combine_embeddings(*[embeddings[e
                ][1] for e in combo])

            # Create and evaluate model
            model = AttCNNModel(X_train_combined.shape[1])
            metrics, fpr, tpr, history = ModelEvaluator.
                evaluate_model(
                 model.model, X_train_combined, X_test_combined,
                 y_train, y_test, model_name=name
            )

            results[name] = {
                'metrics': metrics,
                'fpr': fpr,
```

```
                'tpr': tpr,
                'history': history
            }

            print(f"\nMetrics for {name}:")
            for metric, value in metrics.items():
                print(f"{metric}: {value:.4f}")

    # Plot ROC curves
    plot_roc_curves(results)

    return results

if __name__ == "__main__":
    results = main()
    plot_roc_curves(results)
```

B.3 Python code for the AttCNN Model using CountVectorizer and GloVe embeddings with k-fold cross-validation

```
import numpy as np
import pandas as pd
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import KFold
from sklearn.metrics import *
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from numpy.linalg import norm
from tqdm import tqdm
import time

# Download NLTK resources
nltk.download('punkt')
```

B.3. Python code for the AttCNN Model using CountVectorizer and GloVe embeddings with k-fold cross-validation

```
nltk.download('stopwords')

def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = text.lower()
    text = text.split()
    ps = PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text if not word in
        set(stopwords.words('english'))])
    return text

def load_glove_embeddings(glove_file, dimension=100):
    print(f"Loading {dimension}-dimensional GloVe embeddings...")
    embeddings = {}
    with open(glove_file, 'r', encoding='utf-8') as f:
        for line in tqdm(f):
            values = line.split()
            word = values[0]
            vector = np.asarray(values[1:], dtype='float32')
            embeddings[word] = vector
    print(f"Loaded {len(embeddings)} word vectors.")
    return embeddings

def text_to_glove_vector(text, embeddings, dimension=100):
    words = text.split()
    word_vectors = []
    for word in words:
        if word in embeddings:
            word_vectors.append(embeddings[word])

    if word_vectors:
        vector = np.mean(word_vectors, axis=0)
        if norm(vector) > 0:
            vector = vector / norm(vector)
        return vector
    return np.zeros(dimension)

def create_model(input_shape):
    input_layer = Input(shape=(input_shape,))
    dense_layer = Dense(128, activation='relu')(input_layer)
    dense_layer = Dense(64, activation='relu')(dense_layer)

```

```
    expanded_dense_layer = Reshape((64, 1))(dense_layer)

    conv_layer = Conv1D(filters=128, kernel_size=5, activation='relu', padding='same')(expanded_dense_layer)
    attention_data = Conv1D(filters=128, kernel_size=3, activation='relu', padding='same')(expanded_dense_layer)
    attention_layer = Attention()([conv_layer, attention_data])

    flatten_layer = GlobalMaxPooling1D()(attention_layer)
    dropout_layer = Dropout(0.5)(flatten_layer)
    output_layer = Dense(1, activation='sigmoid')(dropout_layer)

    model = Model(inputs=input_layer, outputs=output_layer)
    model.compile(optimizer=Adam(learning_rate=0.001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

def calculate_metrics(y_true, y_pred, y_pred_proba):
    conf_matrix = confusion_matrix(y_true, y_pred)
    tn, fp, fn, tp = conf_matrix.ravel()

    metrics = {
        'TN': tn,
        'FP': fp,
        'FN': fn,
        'TP': tp,
        'accuracy': accuracy_score(y_true, y_pred),
        'precision': precision_score(y_true, y_pred),
        'recall': recall_score(y_true, y_pred),
        'f1_score': f1_score(y_true, y_pred),
        'roc_auc': roc_auc_score(y_true, y_pred_proba),
        'specificity': tn / (tn + fp),
        'true_negative_rate': tn / (tn + fp),
        'true_positive_rate': tp / (tp + fn),
        'false_positive_rate': fp / (fp + tn),
        'false_negative_rate': fn / (fn + tp),
        'negative_predictive_value': tn / (tn + fn),
        'false_discovery_rate': fp / (fp + tp),
        'mean_absolute_error': mean_absolute_error(y_true, y_pred),
        'mean_squared_error': mean_squared_error(y_true, y_pred),
```

```
        'root_mean_squared_error': np.sqrt(mean_squared_error(
            y_true, y_pred)),
        'log_loss': log_loss(y_true, y_pred_proba),
        'cohen_kappa': cohen_kappa_score(y_true, y_pred)
    }

    return metrics, conf_matrix

def plot_metrics(history, k):
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title(f'Model Accuracy (Fold {k})')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title(f'Model Loss (Fold {k})')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')

    plt.tight_layout()
    plt.savefig(f'training_metrics_fold_{k}.png')
    plt.close()

def plot_confusion_matrix(conf_matrix, k):
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix (Fold {k})')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.savefig(f'confusion_matrix_fold_{k}.png')
    plt.close()

# Main execution
def main():
```

B.3. Python code for the AttCNN Model using CountVectorizer and GloVe embeddings with k-fold cross-validation

```
    # Load and preprocess data
    news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')
    news_dataset = news_dataset.fillna('')

    news_dataset['content'] = news_dataset['author'] + ' ' +
        news_dataset['title']
    news_dataset['content'] = news_dataset['content'].apply(
        preprocess_text)

    # Load GloVe embeddings
    glove_embeddings = load_glove_embeddings('../Dataset/glove.6B
        .100d.txt')

    # Process text data
    texts = news_dataset['content'].values
    y = news_dataset['label'].values

    # Initialize CountVectorizer
    count_vectorizer = CountVectorizer(max_features=5000)

    # Convert texts to GloVe vectors
    glove_vectors = np.array([text_to_glove_vector(text,
        glove_embeddings)
                              for text in tqdm(texts)])

    # Perform K-fold cross-validation
    k_values = [10, 20, 30]
    results = []

    for k in k_values:
        print(f"\nPerforming {k}-fold cross-validation...")
        kf = KFold(n_splits=k, shuffle=True, random_state=42)
        fold_metrics = []

        for fold, (train_idx, val_idx) in enumerate(kf.split(texts
            )):
            print(f"\nFold {fold + 1}/{k}")

            # Split data
            X_train_text = texts[train_idx]
            X_val_text = texts[val_idx]
            X_train_glove = glove_vectors[train_idx]
```

```
            X_val_glove = glove_vectors[val_idx]
            y_train, y_val = y[train_idx], y[val_idx]

            # Get CV features
            X_train_cv = count_vectorizer.fit_transform(
                X_train_text).toarray()
            X_val_cv = count_vectorizer.transform(X_val_text).
                toarray()

            # Combine CV and GloVe features
            X_train_combined = np.hstack([X_train_cv,
                X_train_glove])
            X_val_combined = np.hstack([X_val_cv, X_val_glove])

            # Create and train model
            model = create_model(X_train_combined.shape[1])

            start_time = time.time()
            history = model.fit(X_train_combined, y_train,
                                validation_data=(X_val_combined,
                                    y_val),
                                epochs=10, batch_size=32, verbose=1)
            training_time = time.time() - start_time

            # Evaluate model
            start_time = time.time()
            y_pred_proba = model.predict(X_val_combined)
            testing_time = time.time() - start_time

            y_pred = (y_pred_proba > 0.5).astype(int)

            # Calculate metrics
            metrics, conf_matrix = calculate_metrics(y_val, y_pred
                , y_pred_proba)
            metrics['training_time'] = training_time
            metrics['testing_time'] = testing_time

            # Plot metrics and confusion matrix
            plot_metrics(history, fold + 1)
            plot_confusion_matrix(conf_matrix, fold + 1)

            fold_metrics.append(metrics)
```

```
216
217         # Calculate average metrics across folds
218         avg_metrics = {key: np.mean([m[key] for m in fold_metrics
                ])
219                        for key in fold_metrics[0].keys()}
220
221         print(fold_metrics)
222
223         results.append({
224             'k_fold': k,
225             'metrics': avg_metrics
226         })
227
228     # Export results to text file
229     with open('results.txt', 'w') as f:
230         for result in results:
231             f.write(f"\nResults for {result['k_fold']}-fold CV:\n"
                    )
232             for metric, value in result['metrics'].items():
233                 f.write(f"{metric}: {value}\n")
234             f.write("\n" + "="*50 + "\n")
235
236 if __name__ == "__main__":
237     main()
```

B.4 Python Code of Attention-Based CNN Model with CountVectorizer and GloVe Embeddings Using Grid Search CV

```
1  import torch
2  import torch.nn as nn
3  import torch.optim as optim
4  from torch.utils.data import Dataset, DataLoader
5  import numpy as np
6  import pandas as pd
7  from sklearn.model_selection import train_test_split
8  from sklearn.feature_extraction.text import CountVectorizer
9  from sklearn.metrics import accuracy_score, classification_report,
       confusion_matrix
10 from nltk.tokenize import word_tokenize
11 import matplotlib.pyplot as plt
```

```
import seaborn as sns
from datetime import datetime
import json
import nltk
import re
from nltk.corpus import stopwords
from tqdm import tqdm
import itertools
import torch.nn.functional as F
from nltk.stem.porter import PorterStemmer

# Set device
device = torch.device("mps" if torch.backends.mps.is_available()
    else "cpu")
print(f"Using device: {device}")

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')


class HybridTextDataset(Dataset):
    def __init__(self, count_vectors, glove_vectors, labels):
        self.count_vectors = torch.FloatTensor(count_vectors)
        self.glove_vectors = torch.FloatTensor(glove_vectors)
        self.labels = torch.FloatTensor(labels)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        return {
            'count_vectors': self.count_vectors[idx],
            'glove_vectors': self.glove_vectors[idx],
            'labels': self.labels[idx]
        }


class AttentionLayer(nn.Module):
    def __init__(self, input_dim):
        super(AttentionLayer, self).__init__()
        self.attention = nn.Sequential(
```

```
            nn.Linear(input_dim, input_dim),
            nn.Tanh(),
            nn.Linear(input_dim, 1),
            nn.Softmax(dim=1)
        )

    def forward(self, x):
        attention_weights = self.attention(x)
        attended = torch.sum(x * attention_weights, dim=1)
        return attended


class HybridCNNAttention(nn.Module):
    def __init__(self, count_vocab_size, glove_dim, num_filters,
        kernel_sizes, dropout):
        super(HybridCNNAttention, self).__init__()

        # CNN for count vectors
        self.conv_count = nn.ModuleList([
            nn.Conv1d(1, num_filters, k) for k in kernel_sizes
        ])

        # CNN for GloVe vectors
        self.conv_glove = nn.ModuleList([
            nn.Conv1d(glove_dim, num_filters, k) for k in
                kernel_sizes
        ])

        # Attention layers
        self.attention_count = AttentionLayer(num_filters * len(
            kernel_sizes))
        self.attention_glove = AttentionLayer(num_filters * len(
            kernel_sizes))

        # Fully connected layers
        total_features = num_filters * len(kernel_sizes) * 2
        self.fc = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(total_features, total_features // 2),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(total_features // 2, 1),
```

```
            nn.Sigmoid()
        )

    def forward(self, count_vectors, glove_vectors):
        # Process count vectors
        count_x = count_vectors.unsqueeze(1)
        count_features = [torch.relu(conv(count_x)) for conv in
            self.conv_count]
        count_features = [F.max_pool1d(f, f.size(2)).squeeze(2)
            for f in count_features]
        count_features = torch.cat(count_features, dim=1)
        count_attended = self.attention_count(count_features.
            unsqueeze(1))

        # Process GloVe vectors
        glove_x = glove_vectors.permute(0, 2, 1)
        glove_features = [torch.relu(conv(glove_x)) for conv in
            self.conv_glove]
        glove_features = [F.max_pool1d(f, f.size(2)).squeeze(2)
            for f in glove_features]
        glove_features = torch.cat(glove_features, dim=1)
        glove_attended = self.attention_glove(glove_features.
            unsqueeze(1))

        # Combine features
        combined = torch.cat([count_attended, glove_attended], dim
            =1)
        output = self.fc(combined)
        return output


def train_model(model, train_loader, val_loader, criterion,
    optimizer, num_epochs, device):
    best_val_loss = float('inf')
    train_losses = []
    val_losses = []

    for epoch in range(num_epochs):
        model.train()
        total_train_loss = 0

        for batch in tqdm(train_loader, desc=f'Epoch {epoch + 1}/{
```

```
            num_epochs}'):
125             count_vectors = batch['count_vectors'].to(device)
126             glove_vectors = batch['glove_vectors'].to(device)
127             labels = batch['labels'].to(device)
128 
129             optimizer.zero_grad()
130             outputs = model(count_vectors, glove_vectors)
131             loss = criterion(outputs.squeeze(), labels)
132             loss.backward()
133             optimizer.step()
134 
135             total_train_loss += loss.item()
136 
137         # Validation
138         model.eval()
139         total_val_loss = 0
140         with torch.no_grad():
141             for batch in val_loader:
142                 count_vectors = batch['count_vectors'].to(device)
143                 glove_vectors = batch['glove_vectors'].to(device)
144                 labels = batch['labels'].to(device)
145 
146                 outputs = model(count_vectors, glove_vectors)
147                 loss = criterion(outputs.squeeze(), labels)
148                 total_val_loss += loss.item()
149 
150         avg_train_loss = total_train_loss / len(train_loader)
151         avg_val_loss = total_val_loss / len(val_loader)
152         train_losses.append(avg_train_loss)
153         val_losses.append(avg_val_loss)
154 
155         print(f'Epoch {epoch + 1}: Train Loss = {avg_train_loss:.4
            f}, Val Loss = {avg_val_loss:.4f}')
156 
157         if avg_val_loss < best_val_loss:
158             best_val_loss = avg_val_loss
159             torch.save(model.state_dict(), 'best_model.pth')
160 
161     return train_losses, val_losses
162 
163 
164 def plot_training_history(train_losses, val_losses, timestamp):
```

```
    plt.figure(figsize=(10, 6))
    plt.plot(train_losses, label='Training Loss')
    plt.plot(val_losses, label='Validation Loss')
    plt.title('Training History')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.savefig(f'training_history_{timestamp}.png')
    plt.close()


def save_confusion_matrix(y_true, y_pred, timestamp):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.savefig(f'confusion_matrix_{timestamp}.png')
    plt.close()


def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)  # Remove non-alphabet
        characters
    text = text.lower()  # Convert to lowercase
    text = text.split()  # Tokenize
    ps = PorterStemmer()  # Initialize PorterStemmer
    text = [ps.stem(word) for word in text if not word in set(
        stopwords.words('english'))]  # Remove stopwords and stem
    text = ' '.join(text)  # Join back into a single string
    return text


def main():
    device = torch.device("mps" if torch.backends.mps.is_available
        () else "cuda" if torch.cuda.is_available() else "cpu")

    # Load your dataset
    print("Loading and preprocessing dataset...")
    news_dataset = pd.read_csv('../Dataset/fakeNewsDataset.csv')

```

```
    # Combine author and title, then preprocess
    news_dataset['content'] = (
        news_dataset['author'].fillna('') + ' ' + news_dataset['
            title'].fillna('')
    )
    news_dataset['content'] = news_dataset['content'].apply(
        preprocess_text)

    # Extract features and labels
    X = news_dataset['content']  # Features
    y = news_dataset['label']  # Labels

    # Count Vectorizer (no need to pass stop_words as they're
        handled in preprocessing)
    print("Applying CountVectorizer...")
    cv = CountVectorizer(max_features=5000)
    count_vectors = cv.fit_transform(X).toarray()

    # Load GloVe embeddings
    print("Loading GloVe embeddings...")
    embedding_dim = 100
    glove_path = '../Dataset/glove.6B.100d.txt'
    embedding_index = {}
    with open(glove_path, 'r', encoding='utf-8') as f:
        for line in f:
            values = line.split()
            word = values[0]
            coefs = np.asarray(values[1:], dtype='float32')
            embedding_index[word] = coefs

    # Create GloVe vectors for each document
    max_length = 100  # Set maximum sequence length
    glove_vectors = np.zeros((len(X), max_length, embedding_dim))

    for i, text in enumerate(X):
        words = word_tokenize(text.lower())
        for j, word in enumerate(words[:max_length]):
            if word in embedding_index:
                glove_vectors[i, j] = embedding_index[word]

    # Split data
    print("Splitting data into training and testing sets...")
```

```
    X_count_train, X_count_test, X_glove_train, X_glove_test,
        y_train, y_test = train_test_split(
        count_vectors, glove_vectors, y.values, test_size=0.2,
            random_state=42
    )

    # Create datasets and dataloaders
    train_dataset = HybridTextDataset(X_count_train, X_glove_train
        , y_train)
    test_dataset = HybridTextDataset(X_count_test, X_glove_test,
        y_test)

    # Grid search parameters
    param_grid = {
        'num_filters': [64, 128],
        'kernel_sizes': [[3, 4, 5], [2, 3, 4]],
        'dropout': [0.3, 0.5],
        'learning_rate': [0.001, 0.0001],
        'batch_size': [32, 64]
    }

    # Generate timestamp for saving files
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

    # Store results
    results = []

    print("Starting grid search...")
    for params in tqdm(list(itertools.product(*param_grid.values()
        )), desc='Grid Search'):
        param_dict = dict(zip(param_grid.keys(), params))

        # Create model with current parameters
        model = HybridCNNAttention(
            count_vocab_size=count_vectors.shape[1],
            glove_dim=embedding_dim,
            num_filters=param_dict['num_filters'],
            kernel_sizes=param_dict['kernel_sizes'],
            dropout=param_dict['dropout']
        ).to(device)

        # Create dataloaders with current batch size
```

```
        train_loader = DataLoader(train_dataset, batch_size=
            param_dict['batch_size'], shuffle=True)
        test_loader = DataLoader(test_dataset, batch_size=
            param_dict['batch_size'])

        # Training
        criterion = nn.BCELoss()
        optimizer = optim.Adam(model.parameters(), lr=param_dict['
            learning_rate'])

        train_losses, val_losses = train_model(
            model, train_loader, test_loader, criterion, optimizer
                , num_epochs=10, device=device
        )

        # Evaluation
        model.eval()
        y_pred = []
        y_true = []

        with torch.no_grad():
            for batch in test_loader:
                count_vectors = batch['count_vectors'].to(device)
                glove_vectors = batch['glove_vectors'].to(device)
                outputs = model(count_vectors, glove_vectors)
                predictions = (outputs.squeeze() > 0.5).cpu().
                    numpy()
                y_pred.extend(predictions)
                y_true.extend(batch['labels'].cpu().numpy())

        # Calculate metrics
        accuracy = accuracy_score(y_true, y_pred)
        report = classification_report(y_true, y_pred, output_dict
            =True)

        # Store results
        results.append({
            **param_dict,
            'accuracy': accuracy,
            'report': report,
            'train_losses': train_losses,
            'val_losses': val_losses
```

```
        })

    # Save results
    print("Saving grid search results...")
    results_df = pd.DataFrame(results)
    results_df.to_csv(f'grid_search_results_{timestamp}.csv',
        index=False)

    # Find best model
    best_result = max(results, key=lambda x: x['accuracy'])

    # Plot training history for best model
    plot_training_history(
        best_result['train_losses'],
        best_result['val_losses'],
        timestamp
    )

    # Save confusion matrix for best model
    save_confusion_matrix(y_true, y_pred, timestamp)

    # Save summary report
    summary = {
        'timestamp': timestamp,
        'best_parameters': {k: best_result[k] for k in param_grid.
            keys()},
        'best_accuracy': best_result['accuracy'],
        'classification_report': best_result['report']
    }

    with open(f'summary_report_{timestamp}.json', 'w') as f:
        json.dump(summary, f, indent=4)

    print("Grid search and evaluation completed successfully!")


if __name__ == "__main__":
    main()
```

Fake_News_Rudra_v4

ORIGINALITY REPORT

12%	8%	10%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	2%
2	Mutaz A. B. Al-Tarawneh, Omar Al-irr, Khaled S. Al-Maaitah, Hassan Kanj, Wael Hosny Fouad Aly. "Enhancing Fake News Detection with Word Embedding: A Machine Learning and Deep Learning Approach", Computers, 2024 Publication	1%
3	www.nature.com Internet Source	<1%
4	www.ncbi.nlm.nih.gov Internet Source	<1%
5	core.ac.uk Internet Source	<1%
6	Ghazi Zadeh Hashemi, Mahya Sadat. "Synthetic Aperture Radar in Agriculture With AI-Enhanced Techniques for Crop Classification, Crop Monitoring, and Yield Prediction.", Michigan State University, 2024 Publication	<1%