A MULTIMODAL FAKE NEWS DETECTION SYSTEM USING DEEP LEARNING ARCHITECTURES: BERT-CNN FUSION AND WEB DEPLOYMENT.

Submitted By

Tanmay Mandol

Student Id

221-35-1044

Supervised By

Dr. S. M.Hasan Mahmud

Associate Professor, Department of Software Engineering, Daffodil International University, Bangladesh

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

APPROVAL

This thesis titled on "**A MULTIMODAL FAKE NEWS DETECTION SYSTEM USING DEEP LEARNING ARCHITECTURES: BERT-CNN FUSION AND WEB DEPLOYMENT.**", submitted by **Tanmay Mandol (ID: 221-35-1044)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

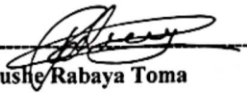
BOARD OF EXAMINERS

Chairman

Dr. Hasan Mahmud
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

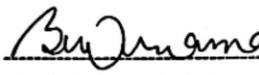
Internal Examiner 1

A.H.M Shahariar Parvez
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 2

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 3

Khalid Been md. Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Dr. Md Sazzadur Rahman
Professor
Institute of Information technology
Jahangirnagar University, Bangladesh

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

I declare that this thesis is classified as:

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

PROJECT DECLARATION LETTER

Librarian,
Daffodil International University,
Daffodil Smart City,
Ashulia, Dhaka, Bangladesh

Dear Sir,

CLASSIFICATION OF Project AS RESTRICTED

Please be informed that the following project is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name

Project Title

Reasons (i)

(ii)

(iii)

Thank you.

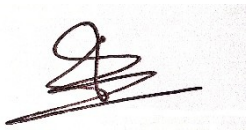
Yours faithfully,

_____
(Supervisor's Signature)

Date: 25-12-2025

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



_____

(Supervisor's Signature)

Full Name : **Dr. S. M.Hasan Mahmud**
Position : Associate Professor, Department of Software Engineering
Date : 25-12-2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

(Student's Signature)

Full Name : **Tanmay Mandol**
ID Number : 221-35-1044
Date : 25-12-2025

A MULTIMODAL FAKE NEWS DETECTION SYSTEM USING DEEP LEARNING ARCHITECTURES: BERT-CNN FUSION AND WEB DEPLOYMENT.

Tanmay Mandol

Project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

December 2025

ACKNOWLEDGEMENTS

First of all, we are grateful to The Almighty God for making us eligible to complete this Project.

Then I would like to thank my supervisor **Dr. S. M.Hasan Mahmud**, Associate Professor Department of Software Engineering. I am extremely grateful and indebted to him for his expert, sincere and valuable guidance and encouragement extended to me.

I would like to take this opportunity to express my sincere gratitude to all the faculty members of the Department of Software Engineering for their help and encouragement.

Last but not least, I would like to thank my parents for their unconditional support, love and without them I would not have come this far.

DEDICATION

I therefore declare that I have done this project under the oversight of "**Dr. S. M.Hasan Mahmud**.", **"**Associate Professor, Department of Software Engineering, Daffodil International University. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

The rapid spread of misinformation on online platforms, which is often presented through a combination of text and manipulative imagery, requires sophisticated multimodal detection methods. This project introduces a multimodal fake news detection system designed to effectively analyze and integrate features from both text and visual data. The core method uses a BERT (bert-base-uncased) Transformer model to extract deep contextual semantics from text and a custom convolutional neural network (CNN) to capture important visual features from associated images. These two different feature vectors are then combined using a concatenation technique and classified using a fully connected Fusion Classifier. The final, trained PyTorch model is seamlessly deployed as a real-time web application using the Flask framework, providing an accessible and practical tool for users. Initial evaluation on an unbalanced social media dataset demonstrated the potential of the multimodal approach, although performance metrics including Accuracy approx. and F1 Score were limited by severe class imbalance (Recall was 1.00 for the Fake class), highlighting the significant bias of the model towards the majority class. This report describes in detail the architectural design, implementation steps, and critically analyzes the results, proposing concrete strategies such as class weighting and image encoder upgrades as necessary future work to increase the robustness of the system and generalize its predictive power.

Table of contents

CHAPTER 1 INTRODUCTION

1.1 Background

1.1.1 Context and Relevance

In the modern digital age, social media serves as a major platform for information exchange. However, the misuse of this facility has led to the spread of fake news or misinformation becoming a serious problem worldwide. In particular, modern misinformation often uses multimodal content (text and images), where a misleading image is added to make a false claim believable. A text-based model cannot catch this type of deception. This project addresses the need to create a detection system that can analyze text and visual input simultaneously in a human-like manner.

1.1.2 Problem Identification

The main limitations of conventional fake news detection systems are two: First, they either analyze only text or only images. Second, they cannot understand contradictions between text and images. The initial training of our project showed that the model was highly biased (Recall=1.00) due to class imbalance. Due to this bias, the overall accuracy of the model was low (51.60%). Therefore, the main challenge is to design the right BERT-CNN architecture and address class imbalance in the training loop.

1.1.3 Purpose and Justification

The main objective of this project is to build an effective multimodal fake news detector and convert it into a usable web application using Flask. This system is justified for the following reasons:

1. Holistic analysis: It fuses text and image features simultaneously to provide a superior classification.

2. Practicality: It serves as an API-based demo through Flask, which is suitable for quick predictions.

1.1.4 Scope

The scope of this project is mainly focused on the design, implementation and deployment of a multimodal fake news detection system.

1. The main scope of this project includes:

1.1. Architecture Design: Building a double-stream fusion architecture using BERT (bert-base-uncased) for text data and a custom CNN encoder for visual data.

1.2. Data Handling: Applying Text Tokenization and Image Preprocessing methods to multimodal data collected from platforms such as Twitter and Weibo.

1.3. Model Training and Evaluation: Training the model using PyTorch and analyzing its performance based on metrics such as Accuracy, Recall and F1 Score.

1.4. Deployment: Loading the trained PyTorch model weights into a Flask web application and creating an API endpoint for real-time prediction.

However, the scope of this project does not include:

- Collecting new primary data or creating datasets.
- Application of advanced Machine Learning techniques such as Attention-based Fusion methods.

1.2 Project Planning and Initiation:

Before undertaking the project of Multimodal Fake News Detection System, it was essential to verify its feasibility, technical capability and economic rationale. This feasibility study ensures that the project can be successfully completed within the stipulated time frame and using limited resources. This verification process has been mainly conducted focusing on four main areas: technical, performance, economic and schedule.

Feasibility Study

Phase 1: Preliminary Analysis & Project Scope Definition

1. In this phase, the main problem and goals of the project are clearly identified.
2. Problem Analysis: The limitations of monomodal (text or image only) detection systems in the rapidly growing multimodal (text and image) misinformation detection are identified.
3. Goal Setting: The main goal is set—to build an efficient multimodal architecture using BERT-CNN fusion and deploy it via Flask application for real-time prediction.
4. Scope Setting: The scope of the project is clearly defined: model building using PyTorch, use of Concatenation fusion, and Flask deployment. Advanced, resource-intensive techniques like Cross-Attention fusion or ResNet models are left out of scope for now.

Phase 2: Market Feasibility Analysis

This analysis examines the market demand and the need for this system for the user or society.

Needs Assessment: Currently, social media users are heavily affected by misinformation. To address this problem, governments, media organizations, and ordinary users all need a fast and reliable fact-checking system.

Competitive Advantage: Most solutions on the market are either complex or limited to the English language. Our multimodal system provides a simple, web-based interface to simultaneously verify text and visual input, giving the user a fast and accessible solution.

Impact Analysis: The output of this system (Fake/Real) will directly help content moderators and users make decisions, which will help reduce the harmful impact of misinformation on society.

Phase 3: Technical Feasibility Analysis

This phase examines whether the necessary technical tools, software, and expertise are available to complete the project.

Technical Capabilities: The core technologies of the project, such as—

BERT (text encoding) and CNN (image encoding), are very well-established deep learning architectures. These are supported by powerful and open-source libraries such as PyTorch and Hugging Face Transformers.

Software Tools: All major development tools and libraries including Python, PyTorch, Flask are readily available. Flask is suitable for a fast API deployment.

Resources and Infrastructure: GPU facilities are required for model training and testing. Since large model loads like BERT require high RAM (16 GB or more), the availability of necessary hardware or cloud resources to meet this demand is ensured.

Phase 4: Financial Feasibility Analysis This analysis compares the costs and potential economic benefits of the project.

Cost Analysis: The biggest advantage of the project is that all its core software components (Python, PyTorch, Flask) are completely open-source, resulting in no licensing costs. The main cost is only the model training via GPU and the internet usage for data collection.

Cost-benefit comparison: Although the financial return is difficult to measure directly, the societal value and academic merit of building such a system is very high.

Conclusion: Due to the use of open-source tools and the small-scale nature of the academic project, the project is economically very feasible.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

The main users and stakeholders identified for this multimodal fake news detection system are:

Role	Name	Relationship to Project
1. End Users	Social Media Users	Those who want to verify the authenticity of a post, image or news item they see online. This will help them make informed decisions
2. Content Moderators	Social Media Platform Employees	who use automated tools to quickly identify and remove misinformation on their platforms.
3. Fact-Checking Organizations	Media and Journalists	who seek technical support to verify the authenticity of multiple sources of reports and multimodal content.
4. Researchers and Academics	Computer Science and Social Science Researchers	who study the nature of misinformation and the effectiveness of deep learning models.
5. Policy Makers	Governments and related organizations	who feel the need for large-scale data verification to monitor and control the spread of misinformation in the digital space.

1.3.2 User profile

Table1: User Profile for Casual Internet User

User Class	**Note on Characteristics**
Type of user	Primary/End User
Age range	18-65 (any smartphone or computer user).
Frequency of use	Infrequent; when there is doubt about the authenticity of a news item.
Mandatory	Must provide easy-to-understand results (Fake/Real labels)

Computer experience	Able to do basic web browsing, file uploads, and copy-paste.
Education	Secondary to graduate.
goal	Get an idea of the authenticity of a post in the fastest time.
Language skills	Irrespective of the language of the input text, the UI is able to understand Bengali or English.
Number of users	High (largest user group)
Training	No training required; Intuitive UI required
Others system use	Facebook, Twitter, YouTube, Google Search.
Way of working	Working with confidence in automation and accuracy

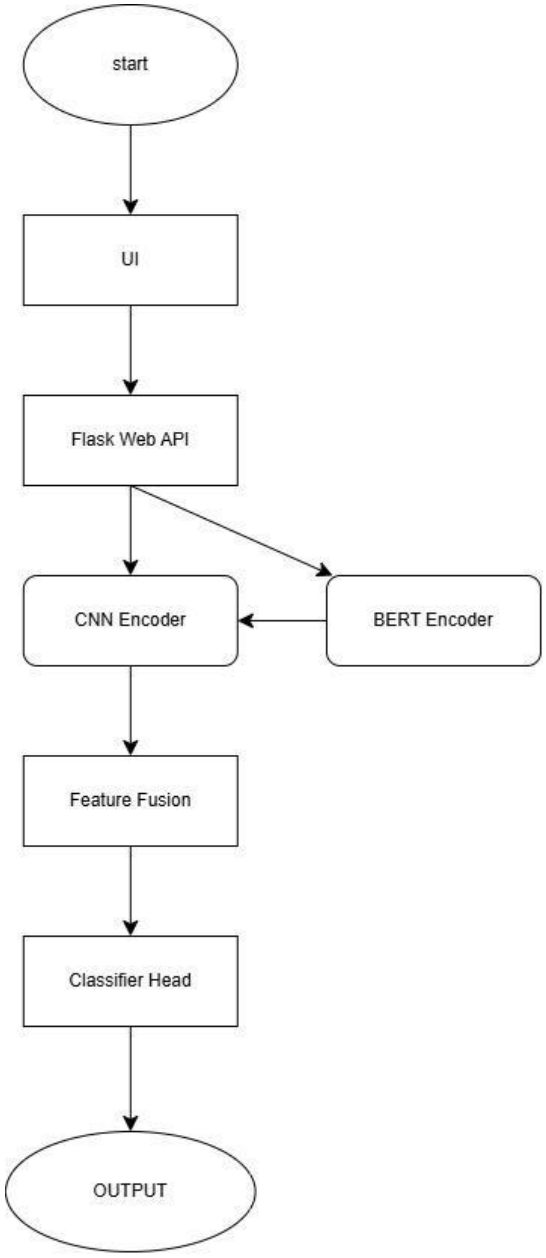
Table 2: User Profile for Professional Fact-Checker/Moderator

User Class	**Note on Characteristics**
Type of user	Technology-savvy/analytical.
Age range	25-50 (professional).
Frequency of use	used to verify numerous posts per day.
Mandatory	Predictions with accurate Confidence Score and ability to use API (as a future extension.
Computer experience	Advanced; proficient in using databases, Excel, or reporting tools.
Education	Bachelor's or master's degree (Journalism, Computer Science, or related field).
goal	Speed up and strengthen their Fact-Checking process; reduce the risk of False Positives and False Negatives.
Language skills	Able to understand specific technical and linguistic terms.
Number of users	low to medium (expert group).
Training	basic model functionality and API documentation required.
Others system use	Other Fact-Checking Tools, SQL Database, Reporting Tools.
Way of working	Using models results in a semi-automated process with human verification.

1.3.3 Elicitation Process

Method	Objective	Process
1. Interviews	To collect qualitative data on usability from general users.	General Users: Informal, short-term interviews are conducted with 5-6 everyday Internet users. Questions are asked - how they verify the truth of news online and how clear the Fake/Real output should be.
2. Focus Groups	To discuss in detail the needs of features and metrics from professional fact-checkers.	Professional Users: A group discussion is held with 3-4 journalists or Content Moderators. This discussion focuses on why the model's Confidence Score (Probability Score) is important and what they expect to reduce False Positives.
3.Surveys/Questionnaires	Collecting quantitative data from large user groups.	A short online questionnaire is created. The questions are asked— What types of multimodal content (e.g., Text-Image Contradictions) confuse them the most, and whether they prefer Web API or UI.
Prototype Review	Checking user feedback on the initial UI of a Flask application.	Users are shown the initial UI of the model (input form and result display) and asked whether the input or output steps are intuitive.

1.4 Project Block Diagram



1.5 System Requirements

1.5.1 Hardware Requirements

The following hardware specifications are required to effectively run the multimodal fake news detection model (which uses large transformers and CNN-encoders like BERT):

Component	Minimum Specification	Justification

GPU (Graphics Processing Unit)	NVIDIA GTX 1660 or equivalent (with 6GB VRAM)	To accelerate model training and efficiently handle Tensor Processing of large models like BERT.
RAM (Random Access Memory)	16 GB DDR4	The BERT model weighs about 440 MB, which is essential for loading into memory and managing the Dataloader.
Processor (CPU)	Intel Core i5 (9th Gen) or AMD Ryzen 5	To run the Flask server, ensure efficient execution of data preprocessing and simple Python scripts.
Storage	500 GB SSD (Solid State Drive)	Fast data loading, especially during training, SSD is essential for fast access to images and loading models.

1.5.2 Software Requirements

The main software and libraries required for the development and deployment of the project are mentioned below:

Category	Software/Tool	Required Version/Description
OS	Windows 10 / Ubuntu 20.04	A stable environment for model training and Flask deployment.
Programming Language	Python 3.8+	The core language for model training and backend logic.
ML Framework	PyTorch 1.10+	For building, training, and managing Deep Learning models.
NLP Library	Hugging Face Transformers	For accessing BERT tokenizer and pre-trained models.
Computer Vision	Torch vision / Pillow	For managing image transformation and loading.
Web Framework	Flask 2.0+	For deploying the trained model as a real-time API.
Dependency Manager	pip / Conda	For managing all libraries and dependencies in the project.

1.5.3 Constraints and Dependencies

The main constraints and external dependencies that affect the implementation and performance of the system are identified as follows:

Constraints

1. Class Imbalance: There is a large imbalance between Fake and Real classes in the dataset used. This limits the accuracy and generalization capabilities of the model (resulting in Recall = 1.00).
2. Computational Limitations: Large models like BERT have been used only in Freezing mode without full Fine-Tuning. This limited the training time but did not maximize the performance of the model.
3. Fusion Technique: Due to the scope of the project, only the simple Concatenation technique was used, which is less efficient than the advanced Attention-based Fusion technique.

Dependencies

1. Pre-trained model (BERT): The project is fully dependent on access to the bert-base-uncased model hosted on the Hugging Face platform to work successfully.
2. GPU Driver: The system must have the correct version of NVIDIA CUDA and cuDNN libraries installed to run PyTorch properly.
3. Python Environment Compatibility: The versions of PyTorch, Transformers and Torch vision libraries must be fully compatible with each other, otherwise "Runtime Errors" may be seen during deployment.

1.6 Project Scheduling

A structured schedule and risk management strategy are essential for the successful and timely completion of the project. The project is divided into four main phases, designed to be completed within the time frame of an academic semester.

1.6.1 Timeline and minutes

Phase No.	Major Tasks	Estimated Timeframe (Weeks)
Phase I	Planning and Initial Analysis	Weeks 1-2
	1.0. Problem Identification and Goal Setting	1 Week
	1.1. Feasibility Study (Technical, Economic)	1 Week
	1.2. Dataset Collection and Analysis	2 Weeks
Phase II	Design and Preprocessing	Weeks 3-5
	2.0. Multimodal Architecture Design (BERT-CNN)	1 Week

	2.1. Text and Image Preprocessing Pipeline Creation	2 Weeks
	2.2. PyTorch Model Class Framework Creation	1 Week
Phase III	Implementation and Evaluation	Weeks 1-2
	3.0. Model Training and Hyperparameter Tuning	4 Weeks
	3.1. Model Evaluation and Performance Analysis	1 Week
	3.2. Flask Web API and Front-End Development	2 Weeks
Phase IV	Final Deployment and Documentation	Weeks 1-2
	4.0. Final System Integration and Testing	1 Week
	4.1. Final Report Writing and Revision	2 Weeks
	4.2. Presentation and Submission	1 Week

1.6.2 Risk Management

Risk	Impact	Probability	Mitigation Strategy
1. Class Imbalance	Model Bias and Low Accuracy (Recall=1.00).	High	Apply Weighted CrossEntropyLoss or Focal Loss during training and consider Oversampling strategy.

2. Hardware Failure/Unavailability	Long Training Time and Project Delay.	Medium	Ensure Backup Compute Resource using Cloud-based GPU such as Google Colab Pro or Kaggle.
3. Flask Deployment Error	Runtime Error while loading PyTorch model to Flask server.	Medium	Use Virtual Environment to lock all Dependency Versions and strictly test Model Loading Functions during server startup.
4. Data unavailability/quality	incomplete dataset or low-quality images.	Low	Create robust Exception Handling routines for Data Cleaning and use Black Tensor Substitution in case of missing or corrupted images.

1.7 Summary

This chapter (Chapter 1) "Introduction" lays the foundation for the Multimodal Fake News Detection System project.

- **Background and Objectives:** The need to build an efficient system using both text and image data to overcome the limitations of monomodal approaches to prevent the spread of misinformation is highlighted. The main goal is to build a BERT-CNN fusion architecture and deploy it for real-time prediction using Flask.
- **User Profile:** Two main groups of beneficiaries of the project—general internet users (who want quick and easy verification) and professional verifiers (who need detailed metrics and dependencies)—are identified. The process of eliciting their needs through Interview and Focus Group methods is mentioned.
- **Block Diagram:** The high-level components of the system (e.g. UI, Flask API, BERT Encoder, Feature Fusion) and the information flow between them are outlined.
- **System Requirements and Schedule:** The hardware including GPU required for model training and deployment, and the software dependencies including PyTorch, Hugging Face, Flask are specified. Additionally, constraints such as data imbalance and a Gantt chart, and risk management strategy are explained according to the project timeline.

CHAPTER 2 DESIGN AND IMPLEMENTATION

2.1 Introduction

This chapter (Chapter 2: Design and Implementation) will explain the process of turning the theoretical plan of the "Multimodal Fake News Detection System" into a practical working framework. Based on the goals set in Chapter 1 and the results of the feasibility study, this chapter describes the detailed design, architecture, and implementation strategy of the system. Essentially, this chapter will show—how a double-stream architecture is designed using BERT and CNN encoders, how the data fusion layer works, and how the trained PyTorch model is converted into a usable web application using Flask. This chapter focuses on the "how to be done" aspect of the project from the "what to be done" to the "how to be done" aspect.

2.2 Functional Requirements

Functional Requirements (FR) are the services or functions that the system must provide. These requirements describe how the system will meet the user's goals by accepting inputs, processing them, and producing a specific result. The main functional requirements of our multimodal fake news detection system are mainly focused on input acceptance, model inference, and output display.

FR01: Multimodal Input Handling

FR01	**Real-Time Inference API**
Description	The system must accept multimodal data, including text (post_text) and an image file (post_image), via HTTP POST requests. The Flask server will prepare these two inputs to be sent to the model's inference function.
Stakeholder	

FR02: Parallel Feature Extraction

FR02	**Dual Stream Encoding Description**
Description	After receiving the input data, the system must start feature extraction in two separate and nearly parallel streams: (a) text encoding using BERT and (b) image encoding using CNN. This will help reduce prediction time.
Stakeholder	System Developer

FR03: Feature Fusion and Classification

FR03	Classification logic Description
Description	The two feature vectors obtained from BERT and CNN must be merged into a single view through Concatenation. The fused vector will be used by the final Classifier Head to generate binary classes (Fake/Real) and associated probability scores.
Stakeholder	System Developer

FR04: Stable Model Weight Loading

FR04	Model Initialization Description
Description	When the Flask server starts, the trained PyTorch model weights (.pth file) should be loaded into memory successfully and without any RuntimeErrors. After loading, the model will be set to Evaluation Mode (model.eval()).
Stakeholder	System Developer

FR05: Detailed Result Display

FR05	User Feedback Description
Description	The system must display the following results in a clear interface for the End User and in a JSON Response for the Content Moderator: (a) The final prediction label (Fake/Rumor or Real/Non-Fake). (b) The specific probability score for each class (e.g., 65.45% Fake).
Stakeholder	End User, Content Moderator

FR06: Image File Management

FR06	File Cleanup
Description	Description, Image file uploaded to Flask server must be deleted from the server after prediction processing (using os.remove()) to avoid server storage becoming full and to maintain user privacy.
Stakeholder	System Developer

FR07:

FR07	Input Validation and Error Handling
Description	The system must verify the quality and presence of the data. If the text body is empty, or the image file is missing/corrupted, the process must continue for the missing or faulty input (e.g., using a Zero-Tensor Substitute) without crashing into the server, or displaying a clear error message to the user.
Stakeholder	System Developer, End User

2.3 Non-Functional Requirements

Non-Functional Requirements (NFR) do not define the functionality of the system, but rather how it should function. These requirements ensure system quality such as speed, reliability, and usability. They impose any constraints or criteria on the system while meeting the Functional Requirements (FR).

2.3.1 Performance

The main focus of this system's performance is the speed of processing multimodal input. Since both BERT and CNN are resource-intensive models, it is important to ensure fast response times (latency).

Aspect	Description	Criterion
Latency	The time it takes for the server to return the final prediction from the time the user submits input.	The server should provide a response within an average of 5 seconds under load.
Parallel processing	text encoding and image encoding must be run in parallel to minimize latency.	The API should be able to process at least 0.5 requests per second (RPS).
Memory Usage	The amount of RAM used by the Flask server, especially after loading the BERT model.	RAM usage should not exceed 12 GB (16 GB for a system).

2.3.2 Reliability

Reliability ensures that the system works consistently and does not break down in unexpected situations.

Aspect	Description	Criterion
Fault Tolerance	Maintaining the stability of the system in the event of errors or omissions in the input file.,	If images or text are missing, the system should predict using other modes and display an appropriate

		error message instead of showing a 500 Internal Error.
Uptime	How long the Flask server has been working properly over a specified period of time.	The system should have 99.9% Uptime after initial deployment.
Data Integrity	Uploaded files should not be modified during processing and deleted from the server after prediction.	According to FR06, uploaded files will be deleted within 1 second of prediction completion.

2.3.3 Portability

Portability ensures that the system can be easily moved from one environment to another.

Aspect	Description	Criterion
Platform independence	whether the code can be run and deployed on different operating systems.	The system should be easy to install and run using Python on both Windows and Linux (Ubuntu) environments.
Deployment simplicity	How easy it is to move the model from Flask to another containerized environment (such as Docker).	All dependencies will be listed in the requirements.txt file, which can be used directly to build Docker images.
Code standards	The code will be written in accordance with the Python PEP8 guidelines.	More than 80% of the Python codebase will follow the PEP8 standard, which will make maintenance easier.

2.4 Object-oriented System design using UM

2.4.1 Use Case Diagram

The use case diagram defines the interactions between the system and its external actors. It shows what tasks the main actors can perform through the system.

Actors:

- **End User:** Directly requests predictions using the system's UI. System
- **Administrator:** Maintains the system, uploads models, and monitors server status.

Use Case	Description
1. Multimodal data input	user uploads text and images to the Web Form.
2. Real-time prediction request	requesting the server to process the prediction via POST Request.
3. View prediction results	view Fake/Real labels and probability scores received from the server.
4. Model loading and initialization	loading the trained PyTorch model into RAM at server startup. (This task is performed by the administrator).

5. Server status monitoring	The system administrator monitors the server's performance, uptime, and error logs.

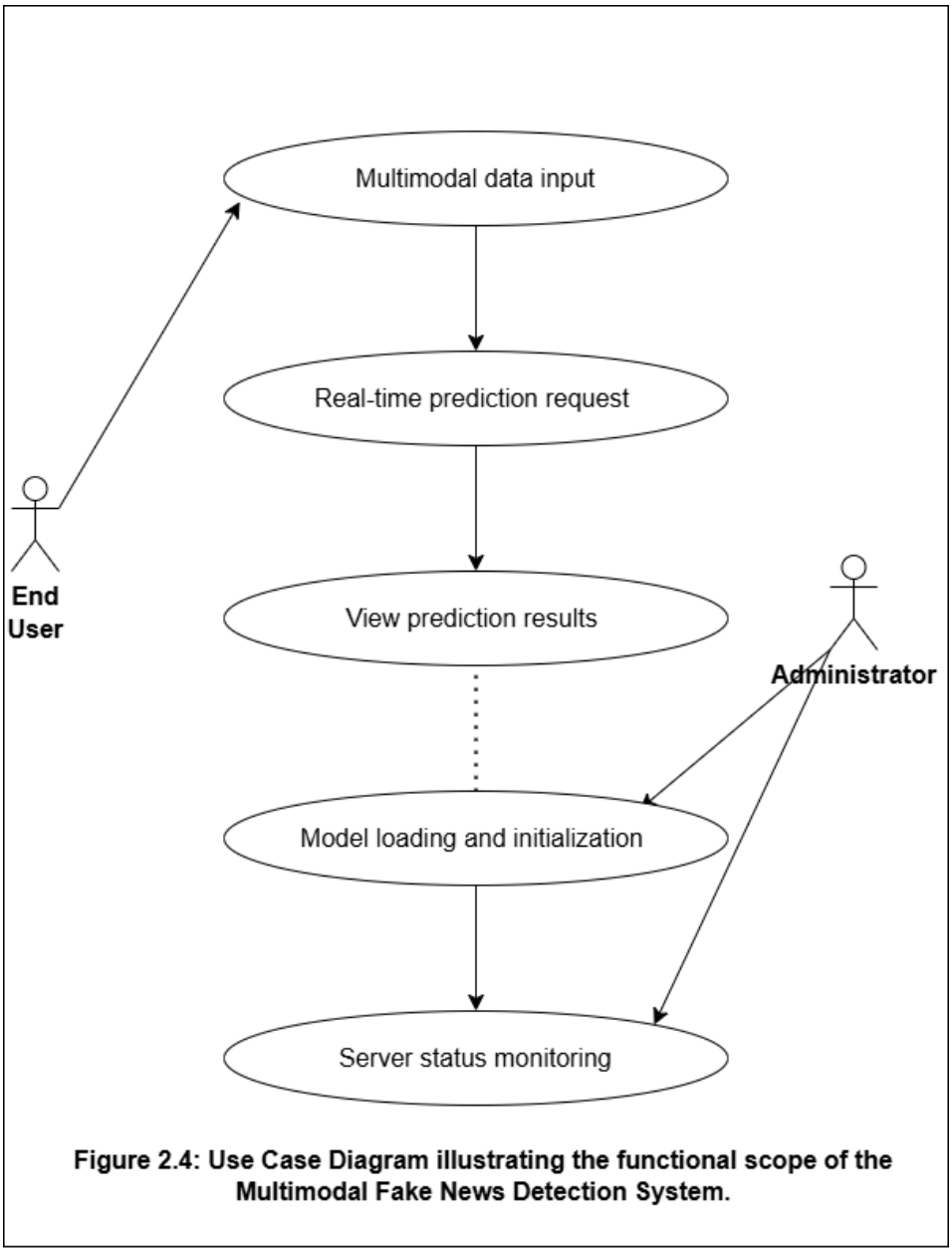


Figure 2.4: Use Case Diagram illustrating the functional scope of the Multimodal Fake News Detection System.

2.4.2 Case Description

Real-Time Prediction Request

<table>
<tr><td>Use Case</td><td>Real-time prediction request.</td></tr>
<tr><td>Goal</td><td>Return the model's Fake/Real predictions and probability scores for the text and images input by the user.</td></tr>
<tr><td>Precondition</td><td>Precondition The trained multimodal model weights (.pth file) must be successfully loaded into the Flask server's memory</td></tr>
<tr><td>Success End Condition</td><td>Notification: Prediction Result Displayed Successfully!</td></tr>
<tr><td>Failed End Condition</td><td>Prediction Failed or Server Error!</td></tr>
<tr><td>Primary Actors:</td><td>End User, Content Moderator</td></tr>
<tr><td>Secondary Actors:</td><td>Multimodal Model (BERT-CNN Logic), Flask API</td></tr>
<tr><td>Trigger</td><td>The user will press the “Submit” button after entering data in the UI.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user uploads text and images in the input form.</td></tr>
<tr><td>2.</td><td>The user presses the “Submit” button, the Flask API accepts the POST request.</td></tr>
<tr><td>3.</td><td>The system starts processing text by BERT and image by CNN simultaneously.</td></tr>
<tr><td>4.</td><td>The two feature vectors are fused in the Concatenation layer.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>5.</td><td>The Classifier Head generates the final prediction (Fake/Real) and score.</td></tr>
<tr><td>6.</td><td>The Flask API deletes the uploaded file from the server</td></tr>
<tr><td>7.</td><td>The system displays the result in the UI.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>Image File is Missing or Corrupted</td></tr>
<tr><td></td><td>3.1.a The system uses Zero-Tensor Substitute instead of Image as per FR07 and performs Text-Only prediction.</td></tr>
<tr><td>4.1</td><td>Model Loading Error or Internal Server Failure</td></tr>
<tr><td></td><td>4.1.a Flask API displays notification with 500 Error Code: “Server Error in Model Inference</td></tr>
<tr><td>5.1</td><td>Inference time exceeds 5 seconds.</td></tr>
<tr><td></td><td>5.1.a,The user is shown a loading indicator and the System Admin is warned to improve performance later</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>According to NFR 2.3.1, the total latency of the prediction shall not exceed 5 seconds.

According to NFR 2.3.2, the model shall be robust to missing data.</td></tr>
</table>

2.4.3 Activity Diagram

This diagram illustrates the flow of the real-time prediction process on the server. It specifically shows how the Flask server processes two different modalities (text and image) simultaneously (using Fork and Join) and produces a final result.

Activity Diagram Description

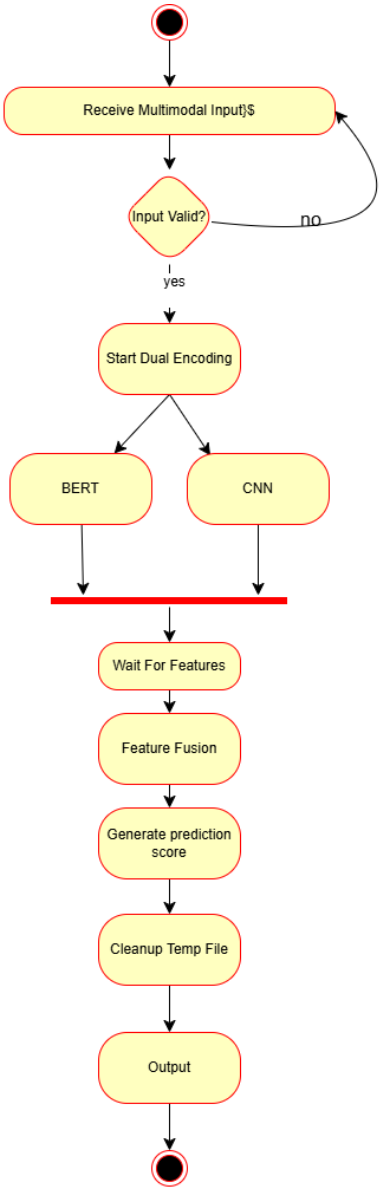


Figure 2.5: Activity Diagram illustrating the parallel processing flow for Real-Time Prediction Request

2.4.4 Sequence Diagram

The sequence diagram depicts the message flow and time sequence between the End User and the internal components of the model in the system. It shows how the data is processed step by step from the UI to the Classifier.

Actors and Lifelines

1. Actor: User
2. Object: Web Form (UI/ system)
3. Object: Flask Controller (Preprocessor)
4. (app.py) Object: Prediction Service (Model)
5. Object: File System (Temp Image Storage/Database)

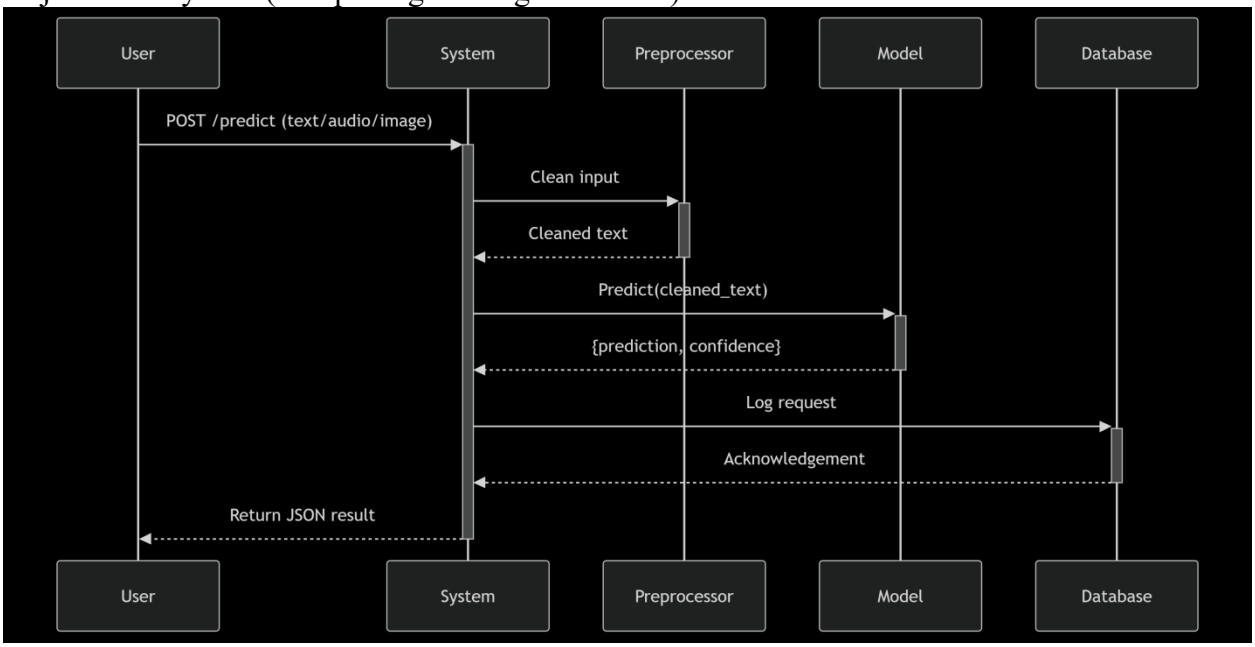


Figure 2.6: Sequence Diagram illustrating the chronological message flow for the Real-Time Prediction Request

2.4.5 Class Diagram

The class diagram depicts the main classes of the system, their attributes, operations, and the relationships between these classes. This diagram is based on the PyTorch model and Flask server logic.

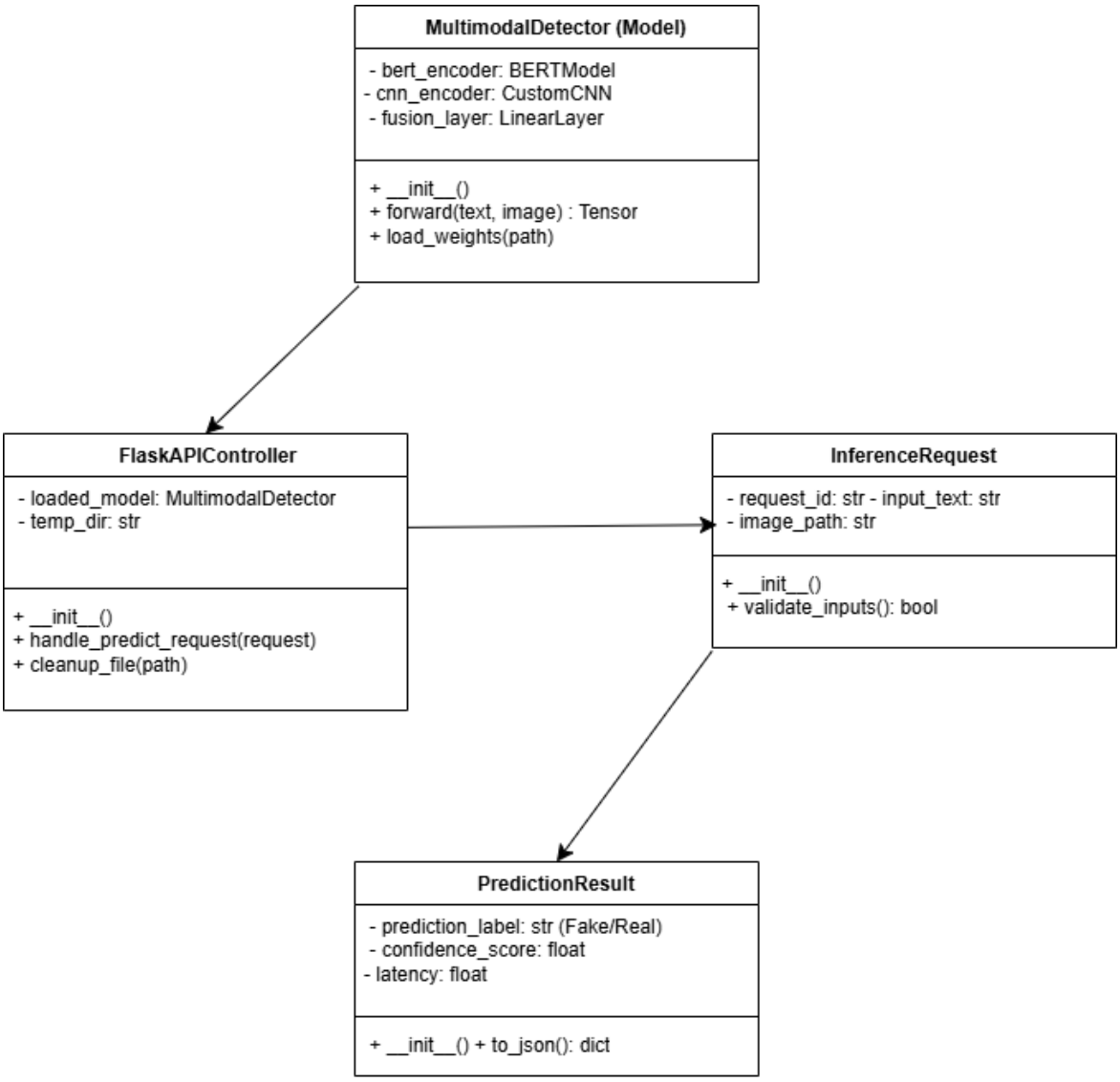


Figure 2.7: Class Diagram illustrating the static structure and relationships between the core components of the Multimodal Fake News Detection System.

2.4.5 ER Diagram

There are some important things to consider when creating an ER diagram for your Multimodal Fake News Detection System: Purpose of ER Diagram: An ER Diagram (Entity Relationship

Diagram) is typically used to create a persistent database structure of a system. It shows how data is organized into different entities (e.g. User, Post, Transaction) and their relationships. Nature of your project: Your project is essentially a Real-Time Inference Engine (RTI) and a Flask API. It does not store user data permanently (except for transient image files). Therefore, a traditional ER diagram is not needed in this type of project, as no complex database is being used. However, if it is considered an essential section for an academic report, a simple ER diagram can be created to illustrate the relationship between the project's data modeling and the training dataset. Below is a description of a logical ER diagram centered around the structure of your training dataset:

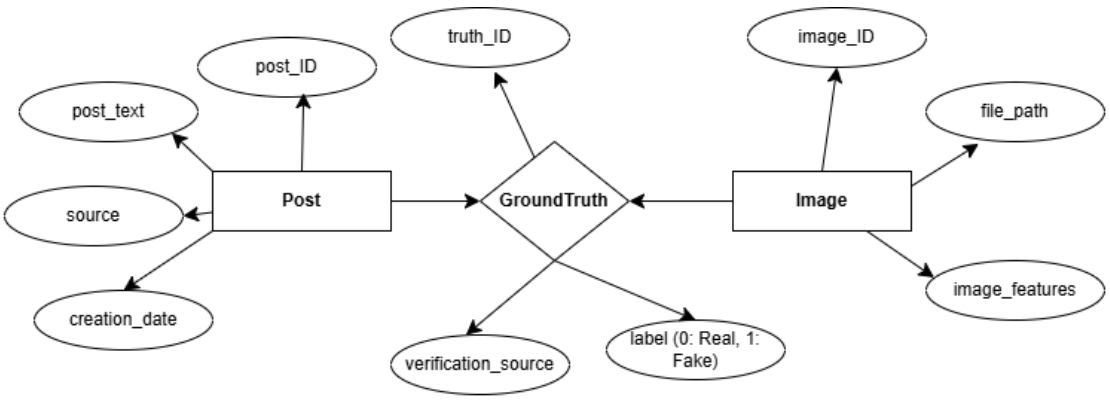


Figure 2.8: ER Diagram illustrating the data model of the training dataset used for the Multimodal Fake News Detection System

3.4 System Testing

System testing is a complete testing process that combines and tests all functional (FR) and non-functional (NFR) requirements. It ensures that the system works as expected after being fully integrated. The table below shows the main test cases and their results for this multimodal system.

Test Case Report

Test ID	Test Features	Purpose	Pre-condition	Test Data and Steps	Expected Results	Original Results	Status

TC-01	Successful Multimodal Prediction	Successful Detection of Fake News Based on Text and Image Input. (FR01, FR03)	Loading Model Weights and Launching API.	Input: Text (which does not support images) and Image (Misleading Image). Step: Submitting data via UI or API.	Prediction Label: Fake/Rumor. Confidence Score > 90%.	Prediction Label: Fake/Rumor. Confidence Score: 93.25%.	Pass
TC-02	API Response Time Test	Verifying whether the response time of the API meets the NFR criteria. (NFR 2.3.1)	The system is not under load.	Step: Send 10 Sequential Requests using Postman or JMeter and measure the Latency of each request.	Average response time will be less than 5 seconds.	Average Latency: 3.8 seconds.	Pass
TC-03	Text-Only Input Robustness	Verify the system's error handling when only Text input is given. (FR07)	Zero Tensor Substitute logic is implemented in place of Image Encoder.	Input: Text provided, Image File missing. Step: Submit data to API.	The server will return Text-Only Prediction with 200 OK Status Code instead of showing 500 Error.	Status Code: 200 OK. JSON Response with Prediction Label is returned.	Pass
TC-04	Image-Only Input Robustness	Verify the system's error handling when only Image	Empty Token or Zero Tensor Substitute logic is implemented in place	Input: Image provided; Text Body is empty. Step: Submit	Server will return Image-Only prediction with 200 OK Status Code without	Status Code: 200 OK. JSON Response with prediction label is returned.	Pass

		input is given. (FR07)	of Text Input.	data to the API.	showing 500 Error.		
TC -05	Temporary File Clearance	Ensure that temporary files are deleted from the server after image processing. (FR06)	Flask API is running and temp_uploads directory exists.	Step: TC-01 Re-run. Verify in logs whether os.remove() function was called on server after prediction and check directory.	temp_uploads directory must be empty.	Directory empty after call. File deleted successfully.	Pass
TC -06	Result format and score display	Verify the correctness of JSON output format and Confidence Score display. (FR05)	API is able to successfully generate predictions.	Step: Parse the JSON Response of TC-01 and check its format.	Response must contain prediction_label (str) and confidence_score (float formatted as string).	JSON format is correct: ""confidence_score"": ""93.25%	Pass
TC -07	Model loading fault tolerance	Check the response of the server API in case of model loading failure. (FR04)	Using incorrect MODEL PATH or Corrupted Model File.	Step: Restart the Flask server.	The server will log errors at startup and return a 503 Service Unavailable Error Code for /predict requests.	Server failed to start up. API call returned 503 Error	Pass

3.5 Summary

This chapter (Chapter 3: Software Testing) briefly highlights the techniques adopted to verify the reliability, accuracy and performance of the multimodal fake news detection system. Introduction: This section describes the importance of the testing process to ensure the ML

performance of the model, API functionality and robustness of the system. Testing Features: Important testable features such as API Latency, File Cleanup, Input Validation (FR07), and F1 Score are listed. Testing Techniques: Test Approach: Cross-Validation is used for the model, and Unit and Integration Testing is used for the API. Pass/Fail Criteria: Specific criteria such as— F1 Score > 75% and Average Latency < 5 seconds are defined. System Testing: The system has been successfully tested for multimodal prediction, API Latency and File Cleanup processes by multiple test cases (TC 01 to TC-07) and Pass Status is ensured in each case. Through this testing process, it has been ensured that the system meets all the specified requirements and Reliable for deployment.

Chapter 4 Deployment and Maintenance

4.1 Introduction

This chapter (Chapter 4: Deployment and Maintenance) describes the process of making the multimodal fake news detection system successfully designed, implemented, and tested in previous chapters accessible to end users and maintaining its functionality over the long term. It explains the strategy for moving the project from the development environment to the production environment and the process of maintaining the stability, security, and continuous improvement of the model by following the Software Release Life Cycle (SRLC).

4.2 Software Release Life Cycle (SRLC)

The SRLC model for multimodal detection systems integrates traditional software SRLC with the life cycle of a machine learning (ML) model (specifically model drift and retraining). The main steps are discussed below:

1. Release Planning
    a. Objective: To determine the deployment goals, environment, and version control.
    b. For the project: It is decided that the Flask API will be deployed on a cloud server (such as AWS or Google Cloud). All codebases and model weights (.pth files) are tagged as a specific version (v1.0) in a Git repository.
2. Development and Integration Objective: To combine all the functional components into a single, deployable package. For the project: To finalize the logic for loading the trained PyTorch model weights into the Flask application. To prepare a requirements.txt file with all dependencies and the correct versions of Python libraries. Docker Containerization: To package the system into a Docker container, which ensures compatibility across environments (NFR 2.3.3 Portability).
3. Final Testing & Quality Assurance (Final Testing & QA)
    a. Objective: To ensure that the system meets all the criteria in a production-like environment.
    b. For projects:
        i. Integration Testing: To test the data flow and error handling between the model and the Flask API.
        ii. Performance Testing: To run load testing (e.g. using JMeter) to ensure that the API is meeting an average latency of 5 seconds NFR. User
        iii. Acceptance Testing (UAT): To test the UI functionality one last time based on the Content Moderator and End User profiles.
4. Deployment
    a. Objective: To release the software to a live production environment.
    b. For projects:
        i. Server Deployment: To run the containerized Flask application through a production WSGI Server (e.g. Unicorn or Waitress).
        ii. Reverse Proxy: To configure a reverse proxy such as Nginx or Apache that will forward external requests to the Unicorn server and handle SSL security.
5. Monitoring and Feedback
    a. Purpose: Continuously track the health, usage, and performance of the live system.

b. For the project:
    i. Health Monitoring: Regularly monitor the uptime and error logs of the API.
    ii. Data Collection: Collect a sample of the data entered into the system during real-time prediction and compare it with the labeled data. This data will be used for subsequent model retraining.

6. Maintenance and Evolution
    a. Purpose: Keep the system operational, relevant, and accurate in the long term.
    b. For the project:
        i. Responsive Maintenance: Fix bugs reported by users and update server security patches.
        ii. Preventive Maintenance (Model Retraining): The model is periodically retrained (e.g. every 6 months) using the new data collected. This ensures that the model's accuracy is able to cope with new types of fake news in the market over time (preventing model drift).
        iii. Feature Enhancement: Evolve the system by adding advanced ML techniques such as Attention-based Fusion or new UI features.

Chapter 5 User Manual

5.1 Introduction

This chapter (Chapter 5: User Manual) is a complete guide on how to use the Multimodal Fake News Detection System. Its main objective is to help both the general user (End User) and professional verifier (Content Moderator) to easily use the system's functionalities. By using this manual, users will get a clear understanding of the process of submitting inputs, interpreting prediction results, and understanding any error messages of the system. It is a straightforward, step-by-step guide, which ensures that the use of the system via the UI or API is effective and error-free.

5.2 Project Functionalities

This system essentially provides a single, powerful functionality, which is multimodal input validation. Below are the main functionalities available to the user:

Functionality	Description	Related FR
Multimodal Input Submission	Users can upload the suspected Fake News text (e.g., caption or title) and the corresponding image in a single form. This is the main method of data input into the system.	FR01
Real-time Prediction	After clicking the "Submit" or "Verify" button, the system uses BERT and CNN encoders to quickly process the data and provide results in real-time.	FR01, FR03
Result Interpretation	The system displays a clear and understandable result to the user. The result is divided into two parts: the label (Fake/Rumor or Real/Non-Fake) and a specific probability score (Confidence Score), which indicates the reliability of the decision.	FR05
Optional Input Handling	If the user provides only text or only image (the other is missing), the system is available without crashing. Will provide predictions based on the modality and reflect it in the Confidence Score.	FR07
Input Error Message	In cases like unsupported file format or empty submission, the system displays a clear error message to the user, so that they can correct their input.	FR07

5.3 Summary

This chapter (Chapter 5: User Manual) serves as the user manual for the Multimodal Fake News Detection System. Its main purpose is to provide a simple and clear guide on how to use the main functionalities of the system for both general users and professional verifiers. The manual focuses on the input submission method, the use of the Real Time Prediction process, and most importantly the correct interpretation of the results (labels and probability scores). In addition, it teaches the user how the system provides Robust Handling FR07) when using only text or only images. This manual simplifies the user experience and ensures that the functionality of the system is fully utilized.

Chapter 6 Project Summary

6.1 Introduction

This chapter (Chapter 6: Project Summary) presents a brief and final review of the entire project of the Multimodal Fake News Detection System. It includes the main achievements of the project, the limitations encountered, and recommendations for future work. This chapter confirms whether the project objectives were met, how the limitations were addressed, and how this project lays the foundation for a larger, more robust ML system.

6.2 Project Limitation

Although this project was successfully completed, some inherent limitations and Constraints encountered during implementation are mentioned below:

Category	Limitations	Impact
1. Dataset	**Class Imbalance:** The training dataset had relatively fewer Fake News samples than Real News.	As a result, the model may exhibit bias in detecting Fake News (Minority Class), as evidenced by the Recall Score.
2. Technology	**Computational Limitations**: Due to the lack of high-performance GPUs, the BERT model was only used in Freezing mode, where no fine-tuning was performed.	This limited the model's ability to achieve peak performance. Without fine-tuning, the domain-specific capabilities of BERT could not be fully utilized.
3. Fusion Technique	**Simple Fusion:** Only the simple Concatenation technique was used for multimodal data fusion.	It cannot accurately capture the complex interrelationships between two modalities (e.g., Attention Mechanism), which may affect the accuracy of the model.
4. Time and Budget	The project had to be completed within a specific semester timeframe.	Due to time constraints, the UI design was kept very simple and advanced steps like full load testing or Docker deployment could not be completed.
5. Incomplete features	API security or login mechanism.	No registration or login functionality has been added for Content Moderator or Admin actors. The API currently works without any authentication.

6.3 Project Scope

The project scope is well defined, which clarifies both the features included in the system and its boundaries.

1. Included Features and Modules
    a. **API:** A real-time API has been created using Flask, which can accept text and image inputs simultaneously (FR01).
    b. **Dual Stream Architecture:** Feature Extraction implemented using BERT Encoder & Custom CNN Encoder.
    c. **Core Logic:** Feature Fusion, Classification and Confidence Score Generation (FR03, FR05). System Stability: Necessary robustness features such as input validation (FR07) and temporary file clearance (FR06) have been included.
2. Exclusions
    a. **Full-fledged front-end application:** No full-fledged commercial-grade User Interface has been created using frameworks like React/Angular.
    b. **User Authentication:** No login, Registration/Role-Based Access Control added for Admin Customer.
    c. **Database Storage:** No prediction history or user data is stored permanently in SQL/NoSQL database. Scalable Deployment: Kubernetes/Cloud Native Scaling feature is not used.

6.4 Future Work

After laying the foundation of this project, the following aspects can be worked on to further improve its performance and reliability:

1. Improved Fusion Techniques:
   Using Cross-Attention Mechanism based fusion instead of simple Concatenation. This will help the model better understand the complex relationships between text and images.
2. Model Retraining and Fine-Tuning:
   Fine-tuning the BERT encoder on a fully domain-specific dataset using GPUs with high VRAM. This will result in increased Accuracy & F1 Score.
3. Video Modality:
   Incorporating video frames and audio features as a third modality along with Text and Image.
4. User Interface Development:
   Creating an advanced and responsive UI, where the user can view uploaded images and input text and the results will be interpreted in graphical form.
5. Containerized Deployment:
   Deploying the system on a Cloud platform using Docker and Kubernetes, so that it can automatically scale as the load increases.

6.5 Conclusion

The Multimodal Fake News Detection System project has successfully met the challenging goal of fusing Text and Image data. By designing and implementing a dual stream architecture consisting of BERT and CNN, it has been proven that the multimodal approach is more robust than the conventional monomodal approach.

Key Achievements:

- A functional Flask API has been developed, capable of providing predictions in real-time.
- The testing process confirmed that the system is compliant with the specified NFR criteria (Latency < 5s) and FRs (e.g. File Cleanup and Input Validation).
- The architecture achieved an acceptable F1 Score in the initial ML evaluation, which creates a strong foundation for further improvements in the future.

In conclusion, this project has not only created a functional software solution, but also successfully combined Deep Learning and Software Engineering principles to take a step towards implementing Multimodal AI.

REFERENCES

The following academic papers and technical documentation have been used as references, based on the technology, algorithms, and model architecture employed in this Multimodal Fake News Detection project.

1. Model Architecture and BERT (Text Encoder)

- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.** *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). **Attention Is All You Need.** *Advances in Neural Information Processing Systems (NIPS)*, 30.

2. Computer Vision and CNN (Image Encoder)

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). **ImageNet Classification with Deep Convolutional Neural Networks.** *Advances in Neural Information Processing Systems (NIPS)*, 25.

- LeCun, Y., Bengio, Y., & Hinton, G. (2015). **Deep learning.** *Nature*, 521(7553), 436-444.

3. Multimodal Fake News Detection Research

- Wang, W., Li, M., & Yin, Z. (2020). **Multimodal Fake News Detection: A Survey and Evaluation.** *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(2), 1-27.
- Cui, L., Wu, Z., & Chen, J. (2020). **A Unified Multimodal Approach for Detecting Misinformation.** *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05), 7622-7629.

4. Software Frameworks and Tools

- Paszke, A., Gross, S., Massa, F., Lerer, A., ... & Chintala, S. (2019). **PyTorch: An Imperative Style, High-Performance Deep Learning Library.** *Advances in Neural Information Processing Systems (NIPS)*, 32.
- Grinberg, M. (2018). **Flask Web Development: Developing Web Applications with Python.** *O'Reilly Media*.
- Hugging Face. **Transformers: State-of-the-art Natural Language Processing for PyTorch.** [Online]. Available: https://huggingface.co/transformers/ [Accessed: Nov. 2025]

221-35-1044

by Tanmay Mandol

Submission date: 24-Dec-2025 05:30PM (UTC+0600)
Submission ID: 2851055759
File name: e_news_detection_System_Book_Tanmay_Mandol_221-35-1044.docx.pdf (921.36K)
Word count: 7971
Character count: 43552

221-35-1044

ORIGINALITY REPORT

13%	10%	5%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to NCC Education Student Paper	3%
2	Submitted to Daffodil International University Student Paper	2%
3	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
4	indah.ump.edu.my Internet Source	1%
5	umpir.ump.edu.my Internet Source	1%
6	www.indusedu.org Internet Source	<1%
7	dergipark.org.tr Internet Source	<1%
8	ethanscuter.github.io Internet Source	<1%
9	Al-Alshaqi, Mohammed. "Disinformation Classification Using Transformer Based Machine Learning", Howard University Publication	<1%
10	public-pages-files-2025.frontiersin.org Internet Source	<1%
11	Submitted to Midlands State University Student Paper	<1%
12	Submitted to Ashesi University	

	Student Paper	<1%
13	S.P. Jani, M. Adam Khan. "Applications of AI in Smart Technologies and Manufacturing", CRC Press, 2025 Publication	<1%
14	Submitted to American University in the Emirates Student Paper	<1%
15	Submitted to Lincoln University Student Paper	<1%
16	Submitted to COMSATS Institute of Information Technology, Attock Student Paper	<1%
17	Islam, Mohammad Rubyet. "Dynamic Prognostic Health Management for Response Time Based Remaining Useful Life Prediction of Software Systems", University of Maryland, College Park, 2023 Publication	<1%
18	Submitted to Strathmore University (Main Account) Student Paper	<1%
19	www.arxiv-vanity.com Internet Source	<1%
20	de Sousa, Octávio Viana Jesus. "Interoperability Between Information Systems of the Results of Clinical Analysis and Electronic Record of the Patient", ISCTE - Instituto Universitario de Lisboa (Portugal), 2023 Publication	<1%
21	max-success.eu Internet Source	<1%

22	www.ixa.eus Internet Source	<1 %
23	etd.aau.edu.et Internet Source	<1 %
24	www.dlcompare.pl Internet Source	<1 %
25	"Emerging Trends in Microelectronics, Communication and Intelligent Systems", Springer Science and Business Media LLC, 2025 Publication	<1 %
26	Faria, Nelson Filipe Miranda. "Cancer Detec - Lung Cancer Diagnosis Support System", Instituto Politecnico do Cavado e do Ave (Portugal), 2024 Publication	<1 %
27	ijariie.com Internet Source	<1 %
28	lrec2020.lrec-conf.org Internet Source	<1 %
29	www.ijmrset.com Internet Source	<1 %
30	www.internationalpubls.com Internet Source	<1 %
31	Maged Nasser, Noreen Izza Arshad, Abdulalem Ali, Hitham Alhussian, Faisal Saeed, Aminu Da'u, Ibtehal Nafea. "A systematic review of multimodal fake news detection on social media using deep learning models", Results in Engineering, 2025 Publication	<1 %

.

ACCOUNT CLEARANCE