

# An Integrated Framework for Misinformation Detection Using Streamlit Application

<sup>1</sup>Arati M Chabukswar, <sup>2</sup>Vishwa B, <sup>3</sup>Deepa Shenoy P and <sup>4</sup>Venugopal K R

<sup>1,3,4</sup>Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bengaluru, Karnataka, India.

<sup>2</sup>School of Computing and Information Technology, REVA University, Bengaluru, Karnataka, India.

<sup>1</sup>arati.chabukswar0106@gmail.com, <sup>2</sup>b.vishwa0786@gmail.com, <sup>3</sup>shenoydp1@gmail.com,

<sup>4</sup>venugopalkr@uvce.ac.in

Correspondence should be addressed to Arati M Chabukswar : arati.chabukswar0106@gmail.com

## Article Info

Journal of Machine and Computing (<https://anapub.co.ke/journals/jmc/jmc.html>)

Doi: <https://doi.org/10.53759/7669/jmc202606020>

Received 24 June 2025; Revised from 02 August 2025; Accepted 02 October 2025.

Available online 22 October 2025.

©2026 The Authors. Published by AnaPub Publications.

This is an open access article under the CC BY-NC-ND license. (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

**Abstract** – Misinformation detection is a crucial task in today’s digital age, aiming to identify whether a news article is true or fabricated. An AI-powered application is developed that utilizes advanced Large Language Models (LLMs), Transformer-Based Pretrained Language Models namely BERT and LLaMA named as LAMBERT model, to classify news content as true or fabricated. The system is trained using a dataset obtained from Kaggle, which comprises approximately 21,417 true news articles and 23,502 fake ones. The primary focus is on detection of fabricated news on political news dataset, as this domain is particularly vulnerable to the spread of misinformation. By fine-tuning the models on this specific dataset, the model performance is improved because of the usage of BERT and LLaMA which enhances the capability of the system to record nuanced contextual and semantic features inherent in natural language. BERT’s bidirectional transformer architecture is adept at understanding the context from both preceding and succeeding words, which is vital in discerning subtle linguistic cues often present in fabricated news. Meanwhile, LLaMA contributes by efficiently processing huge amount of text data and learning complex patterns that are characteristic of political misinformation. Together, these models provide a robust framework for distinguishing between true and false news, thereby mitigating the spread of false news effectively.

**Keywords** – BERT, LLM, (LLaMA), NLP, Political News, Streamlit App.

## I. INTRODUCTION

In this digital world, people rely highly on social platforms for communication and exchange of information which is accessible by public easily and here comes the problem where this information can be fake and mislead the original news. According to statistics, as of January 2023, there were roughly 5.6 billion active Internet users globally and it is even more now [1]. In this work, discussion on detecting the false news based on world politics is done. As the false information of politics not only causes economic damage but also increase social threats.

In this work, a website is created that is trained by a pre-trained model BERT, which can read the given information in both the directions and increase the prediction of false news, and the model here used is LAMA that integrates the information to BERT model. As a result, it is leading the way in creating automated methods for detecting fraudulent information. But this model uses labelled dataset which have separation of True and Misinformation data in it which makes easy to detect the news that is given from the dataset.

## Motivation

Misinformation can manipulate public opinion, sway political outcomes, and trigger social unrest particularly in politically sensitive contexts. Traditional fact-checking methods are slow and labour intensive, this underscores the need for an automated and intelligent detection system that can detect false news headlines accurately, in real-time, and help users distinguish between true and fake information.

One example is the 2013 AP tweet relating to the eruption that hurt Barack Obama in [2], which in only a few minutes lost \$130 billion in market worth, such statements are given to BERT model from the dataset which reads and predicts the news is false or true on addition to that LLM (LAMA) will add the explanation for the given statement [3].

Key contributions are summarized as follows:

- Usage of pre-trained model BERT which helps in binary classification that is 0 as false news and 1 as true news and LLM i.e., LAMA which is used as an API to integrate the data into model and helps in explaining the statement with appropriate proofs. Together an integrated model named LAMBERT (BERT+LLaMA) is developed for detection of false news. "LAMBERT" is meant to serve as a clear and understandable shorthand for this hybrid model while adhering to the name standards widely employed in the NLP field.
- To detect false news, the proposed model is implemented using Streamlit which is an interactive web interface allowing users to enter text and check predictions.
- Usage of benchmark datasets from existing work in the proposed model's implementation, to compare the model's performance with the existing work.
- Furthermore, the SheepDog [4] with BERT model and its variant from the base study is re-implemented, with both the benchmark and custom datasets and its performance is assessed. The accuracy and efficacy of the baseline and proposed approaches are fairly compared.

The format of the document is as follows. Previous studies on the topic are detailed in Section 2. Section 3 discuss the methodology and outcomes. Section 4 provides the discussion and insights. In Section 5, the paper is concluded.

## II. LITERATURE SURVEY

The proliferation of misinformation online has spurred research into advanced misinformation detection systems. Recent studies employ diverse techniques to counter evolving threats.

### *Traditional Machine Learning Models*

In [5], authors have done the evaluation using LIAR Dataset wherein Linear SVM outperformed Naive Bayes and Logistic Regression in F1-score, though specific accuracy metrics were not reported; the dataset's bias toward U.S. politics may limit generalizability. In [6], comparative Analysis on ISOT and Kaggle Datasets is performed wherein Support Vector Machines achieved high accuracies ranging from 98.6% to 99.8% on the ISOT dataset and 94% to 96.5% on the Kaggle dataset, outperforming Logistic Regression, with cross-validation accuracies between 92% and 100%, highlighting SVM's robustness. Authors in [7] have done the comparison on Kaggle Dataset by comparing SVM, Naive Bayes, and LSTM models over 15,000 articles, among which LSTM obtained the best F1-score, recall, and precision (all at 0.94), followed by SVM as well as Naive Bayes (precision: 0.68, recall: 0.86, F1-score: 0.76, and precision: 0.85, recall: 0.93, F1-score: 0.89) indicating LSTM's superior performance in this context.

### *Ensemble Learning Models*

The BharatAuthenticNet (BAN) model in [8], uses a voting classifier integrating five algorithms: Naive Bayes, Decision Tree, K-Nearest Neighbours (KNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN). Trained on the Bharat Fake News Kosh (BFNK) dataset, encompassing 26,232 entries across 60 domains in nine Indian languages, BAN's accuracy, precision, recall, and F1-score are 93.41%, 96.19%, 98.89% and 94.21%, respectively outperforming baseline models like SVM and MLP.

### *Convolutional and Recurrent Neural Network-Based Models*

In [9], the study employs a CNN architecture with preprocessing steps like lowercase conversion, stop word removal, stemming, and tokenization, achieving an accuracy of 93.64% on the Reuter dataset, surpassing baseline methods. The authors in [10], integrated CNN layers for feature extraction and an LSTM layer for capturing long-term dependencies, this model, trained on multiple datasets including ISOT, achieves F1- scores up to 99%, indicating high accuracy in misinformation detection. To increase the accuracy of fake news identification, the T-LSTM approach in [11] makes use of neighbourhood information and transductive learning. Sequence modelling and PPCA for feature reduction are combined to improve test data adaptability. This method uses test-specific information to improve classification performance over current approaches. The approach in [12] evaluates three LSTM-based models on a Kaggle dataset of 44,919 articles. To accurately identify false information and stop its spread, authors in [13] employed ML and DL techniques to create an efficient method for detecting fake news related to COVID-19. The method focused on the use of various word embedding techniques in conjunction with models such as LSMT and Bi-LSTM.

### *Large Language Models*

LLaMA 7B with Self-Instruct Fine-Tuning is done by authors in [3]. This study employs a 7B-parameter LLaMA model fine-tuned using over 70,000 self-instruction prompts and 30 alignment tasks, utilizing mixed-precision training and 8-bit quantization to reduce GPU memory requirements from 112 GB to 28 GB. While specific accuracy metrics are not reported, the model demonstrates alignment with human judgment in identifying misinformation examples. In [14], LLAMA-3 8B for Spanish News Reliability Detection is performed utilizing the RUN-AS dataset with 5W1H annotations, this approach fine-tunes an 8B-parameter LLAMA-3 model using Low-Rank Adaptation (LoRA) and 4-bit quantization, achieving a Macro F1-score of 0.59658 and ranking second in the IberLEF 2024 FLARES Subtask 2. In [4], a model SheepDog (Style-Agnostic Fake news Detector), addressed vulnerabilities to style-based attacks, SheepDog combines a fine-tuned RoBERTa-base model with GPT-3.5 for style-diverse reframing, employing multi-task learning to

focus on content veracity. It achieves F1-scores ranging from 80.99% to 88.32% across datasets like PolitiFact, Gossip Cop, and LUN, outperforming baselines such as RoBERTa optimized model incorporating L2 regularization, batch normalization, and the Adam optimizer, attaining a 98% accuracy rate and a 0.98 F1-score. The study in [15] contrasts BERT-like models with LLMs and presents a validated dataset for fake news identification. It concludes that while LLMs are more adaptive, BERT models are more accurate. AI-human annotation also enhances detection performance and label quality.

### Transformer-Based Models

In [16], Fine-tuned BERT and RoBERTa on COVID-19 Tweets are used. BERT, fine-tuned on the ConstraintAI 2021 COVID-19 Fake news Dataset, achieved 98.31% accuracy, outperforming RoBERTa (97.52%) and prior BERT models, highlighting the effectiveness of tweet-specific pre-training. In [17], BERTurk and CNN/Bi-LSTM Extensions on Turkish Data is used and named it as BERTurk+ CNN model with frozen parameters which attained 95% accuracy and 94% F1-score on the TR\_FaRe\_News dataset, demonstrating strong performance in Turkish fake news detection. In [18], RoBERTa with Modified DNN on WELFake Dataset is used by authors. A RoBERTa-based model combined with a deep neural network achieved 99.76% accuracy and up to 99.72% F1-score on the WELFake dataset, surpassing previous models. Authors contributed towards detection of misinformation on political news data using hybrid deep learning model BiGRU and CNN with ALBERT pretrained model achieving an accuracy of 98.25 on ALBERT+BiGRU in [19]. Authors in [20], used CT-BERT with BiGRU for identifying COVID-19 fake news. The model of CT-BERT+BiGRU achieved 97.57% accuracy and 98.50% F1-score on a dataset of 10,700 social media posts, outperforming other transformer-based models and baseline machine learning approaches.

### III. METHODOLOGY

This section describes the methodology employed for false news detection utilizing a BERT-based deep learning model, as implemented in the provided Python scripts ('main.py' and 'app.py'). The approach is structured into six principal phases: Data Collection, Data Visualisations, Model Architecture, Training Procedure, Evaluation, and Prediction.

#### Data Collection

The political dataset contains about 44,919 political news articles in total. It has 21,417 true and 23,502 fake articles as shown in **Table 1**. This almost equal split makes sure the data is balanced between both groups. A balanced dataset helps to stop bias when training models letting the model learn patterns from both types well. This leads to reliable testing and makes the model better at spotting misinformation news and the dataset is considered from the online repository Kaggle (<https://www.kaggle.com/datasets/clmentbisailon/fake-and-real-news-dataset/data>).

**Table 1.** Distribution of True and False News Articles in the Dataset

| Article Type | Number of Articles |
|--------------|--------------------|
| True         | 21,417             |
| Fake         | 23,502             |

#### Data Visualisations

It helps people understand the structure of datasets. As seen in **Figs 1 and 2**, respectively, Word-Cloud descriptions are utilized to visualize data for both fake and real news. The most significant and frequently used terms in both categories of material are displayed prominently in word clouds for fake and real news. The increased font size of a word in a WordCloud indicates the frequency of that word in the articles.



**Fig 1.** Fake News Word-Cloud

**Fig 2.** True News Word-Cloud.

#### Model Architecture

The work begins with the collection of labelled datasets containing both false and true news headlines sourced from two CSV files, which are merged and assigned binary labels 0 for true news and 1 for misinformation news forming the core of supervised learning. After data collection, preprocessing is carried out to clean the text and to guarantee balance, divide the dataset into training, validation, test sets and effective learning. The headlines are then passed through the

BERT tokenizer, which is a pre-trained model called "BERT-base-uncased," which is well-known for producing contextual embeddings that capture text's subtle semantics. It converts them into token IDs and attention masks, enabling the BERT model to process textual input efficiently. These tokenized inputs are used to train a BERT-based neural network that includes additional layers for classification, and its performance is evaluated using standard metrics to ensure reliability. To adapt BERT for the binary classification task of misinformation detection, the following layers are append to the BERT output (specifically, the 'pooler\_output' representing the  $\backslash[\text{CLS}]$  token embedding):

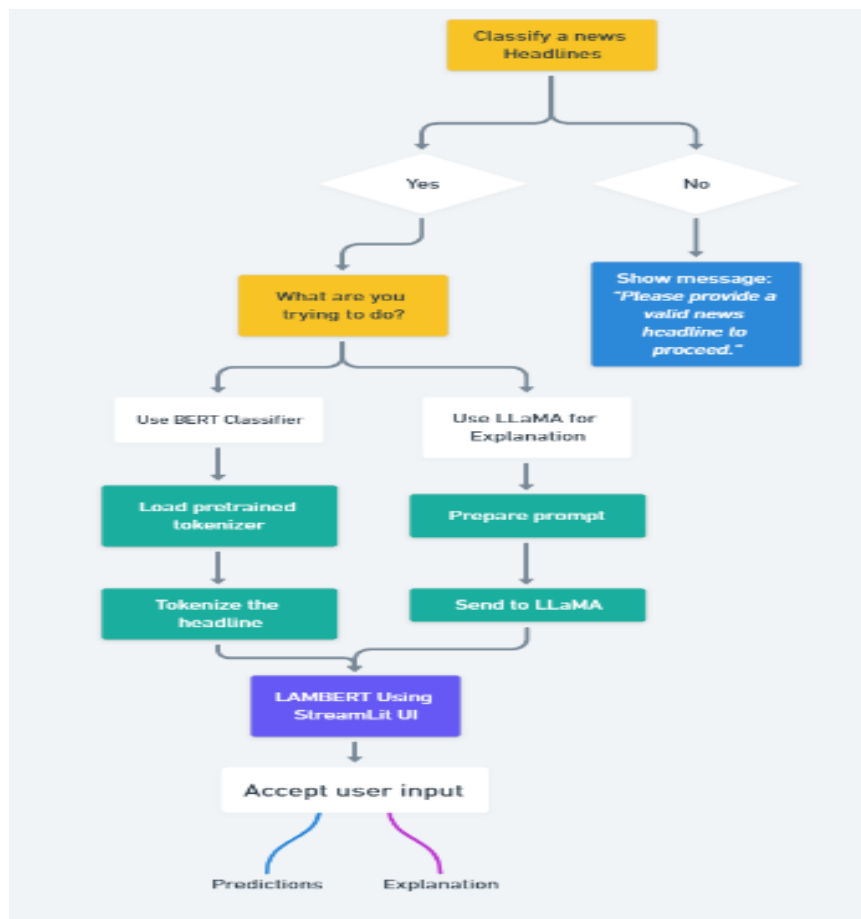
- **Dropout Layer:** A dropout layer, which has a 0.1 chance of randomly deactivating some of the neurons during training, is utilized to prevent overfitting.
- **First Fully Connected Layer:** A linear layer reduces the dimensionality from 768 (BERT's hidden size) to 512, then a ReLU activation function to add non-linearity and improve the model's capacity to recognise intricate patterns.
- **Second Fully Connected Layer:** Another linear layer maps the 512-dimensional output to 2 classes, corresponding to true (0) and fake (1) news.
- **LogSoftmax Layer:** A LogSoftmax function is used to produce log probabilities for each class, facilitating the computation of the loss function and prediction probabilities.
- The architecture is shown in **Fig 1** leverages BERT's contextual understanding while tailoring it for the specific task of misinformation detection, ensuring robust classification performance. As indicated in **Table 2**, the data is split into train-validate-test sets using a 70-15-15 split ratio.

### Training Procedure

With a learning rate of  $1 \times 10^{-5}$ , the AdamW optimiser is used to train the model to minimize overfitting through weight decay. Negative Log Likelihood Loss (NLLLoss) is employed to optimize classification performance using log-softmax outputs. For training, a batch size of 32 is utilized, which is spread throughout 30 epochs. The model is trained on a GPU when available, with weights saved post-training for reuse in prediction tasks. The loss for a single sample is defined as equation (1):

$$L = -\log p_y \quad (1)$$

where,  $p_y$  is the predicted chances of the actual class "y". For a batch, the loss is averaged over all samples, guiding the optimization process to maximize the likelihood of correct classifications.



**Fig 3.** Architecture of Proposed Model LAMBERT.

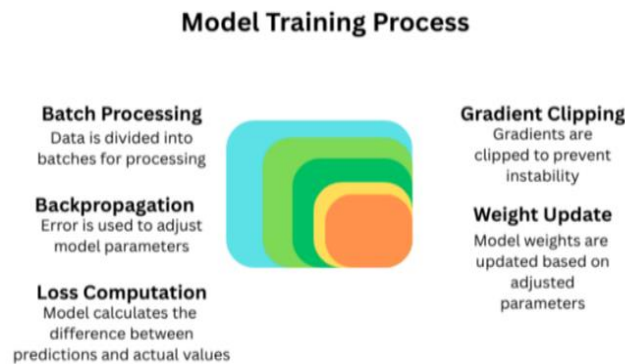
Training is conducted over 10 epoch with a batch size of 1123, utilizing PyTorch’s ‘Data Loader’ for efficient batching and shuffling of the training data. The model processes a batch of inputs, calculates the loss, backpropagates, and changes the weights at each training iteration. To stabilize training and avoid bursting gradients, a common problem in deep learning, gradient clipping is used with a maximum norm of 1.0. The training process is executed on a GPU if available, otherwise on a CPU, optimizing computational efficiency. The model’s weights are saved to a file (‘model.pth’) upon completion, enabling reuse in prediction tasks. **Fig 3** shows the model training process covering batch processing, backpropagation, loss computation, gradient clipping, and weights update.

**Table 2.** Train-Validate-Test Split Samples of Dataset

| Train | Validate | Test |
|-------|----------|------|
| 31449 | 6735     | 6735 |

### Evaluation

The accuracy of the model is used to evaluate its performance, described as the proportion of headlines that were correctly classified to all predictions. This metric offers a direct measure of classification effectiveness. Evaluation is conducted on the test set by comparing predicted and true labels, with accuracy computed using standard functions from the ‘sklearn’ library.



**Fig 4.** Model Training Process.

- **Accuracy**: Accuracy is defined as the proportion of accurately anticipated outcomes to all input predictions. It is calculated as shown in equation (2):

$$\text{Accuracy} = \frac{Tp + Tn}{Tp + Tn + Fn + Fp} \quad (2)$$

where,  $Tp$  stands for true positive in which model correctly predicts positive class,  $Tn$  stands for true negative in which model correctly predicts negative class,  $Fp$  stands for false positive in which model incorrectly predicts positive when its actually negative and  $Fn$  stands for false negative in which model incorrectly predicts negative when its actually positive.

- **Precision**: The metric that displays the Real/True class's accuracy is called precision. For the Real class, it determines the likelihood that the forecast is accurate.

$$\text{Precision} = \frac{Tp}{Tp + Fp} \quad (3)$$

- **Recall**: The percentage of accurately identified positive predictions among all actual positive forecasts is known as recall.

$$\text{recall} = \frac{Tp}{Tp + Fn} \quad (4)$$

- **F1-score**: It is the precision and recall harmonic mean.

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})} \quad (5)$$

These metrics provide a straightforward assessment of the model’s classification efficacy. The evaluation is performed by passing the test set through the trained model, comparing the predicted labels to the true labels, and computing the accuracy score using functions from the ‘sklearn’ library.

### Prediction

To classify new headlines, inputs are tokenized using the BERT tokenizer and converted to tensors consistent with the training format. The model outputs log probabilities, which are exponentiated to obtain actual probabilities. The prediction is made for the class with the highest probability. A confidence threshold of 0.6 is applied to indicate uncertainty when predictions are not confident. This prediction logic is integrated into a Streamlit web interface for deployment. The predicted class is determined as equation (6):

$$p = \exp(\log_p) \quad (6)$$

The predicted class is determined as:

$$y^{\wedge} = \arg \max (p) \quad (7)$$

Additionally, a confidence threshold of 0.6 is applied; if the maximum probability is below this threshold, the model indicates uncertainty, suggesting that the headline may not align closely with the training data. This prediction mechanism is implemented in both ‘main.py’ for testing and ‘app.py’ for deployment in a Streamlit web interface, enabling real-time classification of user-provided headlines.

### Key Formulas and Their Elaboration

The following formulas are integral to the methodology, as derived from the implementation in the Python scripts:

#### Negative Log Likelihood Loss (NLLLoss)

This loss function measures the negative log probability of the true class, encouraging the model to assign high probabilities to correct predictions as shown in equation (7)

$$L = -\log p_y \quad (8)$$

For a batch of (N) samples, the total loss is the average as in equation (8):

$$L_{batch} = -\frac{1}{N} \sum_{i=1}^N \log p_{yi} \quad (9)$$

This formulation ensures that the model optimizes for accurate classification across all samples.

### Accuracy

Accuracy quantifies the model’s performance by calculating the fraction of predictions that match the true labels, providing a clear metric for evaluating classification success.

### Prediction Probabilities

As the model outputs log probabilities through the LogSoftmax layer, exponentiation is used to convert these to actual probabilities, which are then used to determine the predicted class and assess prediction confidence as shown in equation (9). The use of a confidence threshold (0.6) enhances the reliability of predictions by flagging uncertain cases.

$$p = \exp(\log_p) \quad (10)$$

These formulas underpin the training, evaluation, and prediction phases, ensuring the robustness and quantifiability of the model’s performance.

### Details of Implementation

The methodology is implemented in two Python scripts:

- ‘main.py’: Handles data preparation, model training, evaluation, and initial prediction testing. It includes functions for loading and preprocessing data, defining the BERT-based model architecture, training the model, evaluating accuracy, and saving the trained weights.
- ‘app.py’: Deploys the trained model in a Streamlit web application, allowing users to input news headlines and receive predictions. It loads the saved model weights and implements the prediction functionality, ensuring seamless integration with the training pipeline.

The use of PyTorch for model training and the ‘transformers’ library for BERT ensures scalability and efficiency. The methodology excludes external AI services for explanations, focusing exclusively on the BERT-based classification process, as per the implementation in the provided scripts.

*Parameter Summary*

The following **Table 3** summarizes the key parameters used in the methodology, providing a concise reference for implementation details

**Table 3.** Hyperparameters Tuning of BERT Model

| Parameter                       | Value                                 |
|---------------------------------|---------------------------------------|
| Model                           | BERT (BERT-base-uncased)              |
| Dropout Probability             | 0.1                                   |
| Learning Rate                   | $(1 \times 10^{-5})$                  |
| Batch Size                      | 32                                    |
| Epochs                          | 30                                    |
| Train-Validate-Test Split Ratio | 70%-15%-15%                           |
| Maximum Sequence Length         | 15 tokens                             |
| Evaluation Metric               | Accuracy, Precision, Recall, F1-Score |
| Confidence Threshold            | 0.6                                   |
| Optimizer                       | ADAM                                  |

This table encapsulates the critical settings that define the model's configuration and training process, facilitating replication and further development.

LAMBERT proposed model workflow is shown in Algorithm 1 below.

**Algorithm 1** LAMBERT model for classification of Fake/True news on political news dataset

**Input:** a1\_True.csv and a2\_Fake.csv Political news (PN) datasets containing news headlines and labels. Political news: Training sample  $t_{train}$ , Validation sample  $v_{valid}$ , and Testing sample  $t_{test}$ .

**Output:** Truthfulness class TC= {Misinformation, Genuine} where, Probability P: TC -> [0,1]

1: Read the input text from the specified location.

3: for TC  $\in$  PN do

    Load BERT Tokenizer objects, format the input text using input\_id and attention mask, obtain the contextual embeddings.

    end for

4. for TC  $\in t_{train}$  do

    Datasets are labeled, combined, shuffled, and split into training, validation and testing subsets.

    Tokenize news titles using the *bert-base-uncased* tokenizer with a fixed max length, padding, and truncation.

    Define a custom classifier using pretrained BERT with two linear layers, trained using NLLoss and AdamW, save the final model as model.pth.

    end for

5. for TC  $\in t_{train}, v_{valid}$  do

    Store loss/accuracy values, obtain training/validation loss/accuracy curves

    end for

6. for TC  $\in t_{test}$  do

    Loading saved model, evaluating tokenized test data,

    Deploy StreamLit app to accept user input (news headline) and display predicted label and optional explanation

    Predict the labels using **argmax** from the model's output and set Flag=false

    if data("headline") == user\_headline

        output corresponding label

        set found to "**true**"

    exit the loop

    else continue to the next record

    if found is still "**false**" after the loop ends

    return "Data not found in the dataset"

end for

7. Prompt LLAMA API with Headline and its predicted Label.

Generate natural language explanation justifying the classification as Misinformation or Genuine. Return Explanation using Streamlit app.

## IV. RESULTS AND DISCUSSION

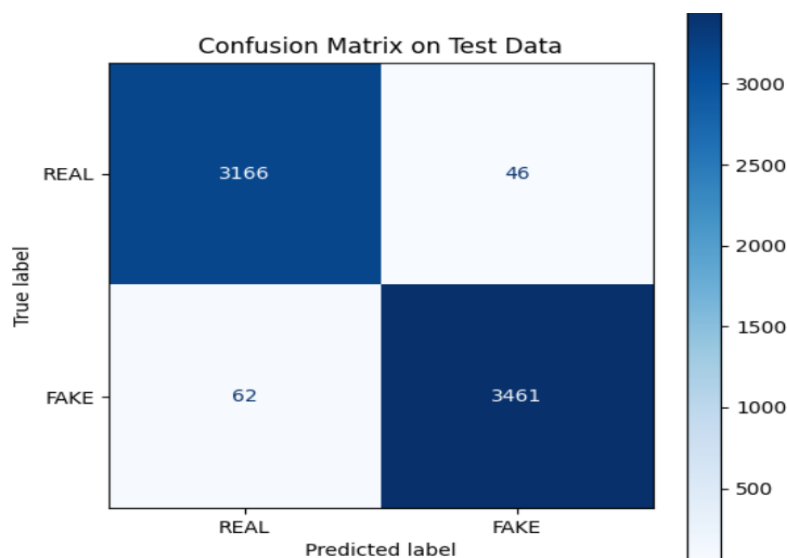
In this section, the experimental findings from the false news detection system developed using LAMBERT on a political news dataset.

### Experiment Setup

The experimental setup is powered by a 13th Generation Intel Core i9 processor paired with 16 GB of RAM, providing robust computational performance. An NVIDIA RTX 4070 GPU is integrated to accelerate deep learning and parallel processing tasks. This combination ensures efficient handling of large datasets and complex model training. The programming language used is Python, and the BERT implementation is used from Hugging Face which is built primarily in PyTorch which makes the loading of pretrained model easier, TensorFlow for scalable training and production deployment, scikit-learn to evaluate BERT outputs. Overall, the system is well-suited for high-performance AI research and real-time inference applications.

### Confusion Matrix

A tabular representation that assesses a confusion matrix measures a categorization model's efficacy in terms of "accuracy" by contrasting the dataset's actual values with the model's predicted values [21]. **Fig 4** displays the confusion matrix using test data for the proposed misinformation detection study, where True Positive = 3166, True Negative = 3461, False Positive = 62 and False Negative = 46. As shown in the confusion matrix for a two-class situation, the model works effectively on political news dataset. Performance evaluation using confusion matrices showed excellent classification results. The Test Accuracy, Test Precision, Test Recall and Test F1-score is 0.9840, 0.9808, 0.9857 and 0.9827 respectively.



**Fig 5.** Confusion Matrix on Test Data Using BERT Model.

### Ablation Study on BERT Model

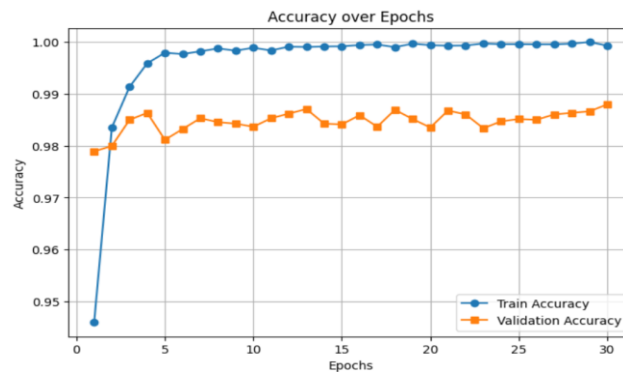
**Table 4** presents an ablation study demonstrating the results of BERT with Version 1, 2, 3, 4 by varying Hyperparameter values and recording the accuracy results achieving an optimal performance with a small learning rate of 1e-5 where optimizer takes small steps lowering the jeopardy of overshooting, batch size of 32, threshold value of 0.6 with 30 epochs. This configuration yields an accuracy of 98.40%. To comprehend the effect of hyperparameters, this examination is centred on the encoder model, which is known for its efficacy in tasks of classification.

**Table 4.** Results Of BERT Model by Varying the Hyperparameter Values

| Model | Version | Hyperparameter                            | Values tested      | Accuracy (%) | Precision (%) | Recall (%)   | F1-score (%) |
|-------|---------|-------------------------------------------|--------------------|--------------|---------------|--------------|--------------|
| BERT  | 1       | Learning rate, Dropout, Threshold, Epochs | 2e-5, 0.2, 0.4, 30 | 98.26        | 98.27         | 98.26        | 98.26        |
|       | 2       | Learning rate, Dropout, Threshold, Epochs | 5e-6, 0.3, 0.6, 30 | 98.16        | 98.19         | 98.16        | 98.16        |
|       | 3       | Learning rate, Dropout, Threshold, Epochs | 1e-4, 0.5, 0.8, 30 | 47.69        | 22.74         | 47.69        | 30.80        |
|       | 4       | Learning rate, Dropout, Threshold, Epochs | 1e-5, 0.1, 0.6, 30 | <b>98.40</b> | <b>98.08</b>  | <b>98.57</b> | <b>98.27</b> |

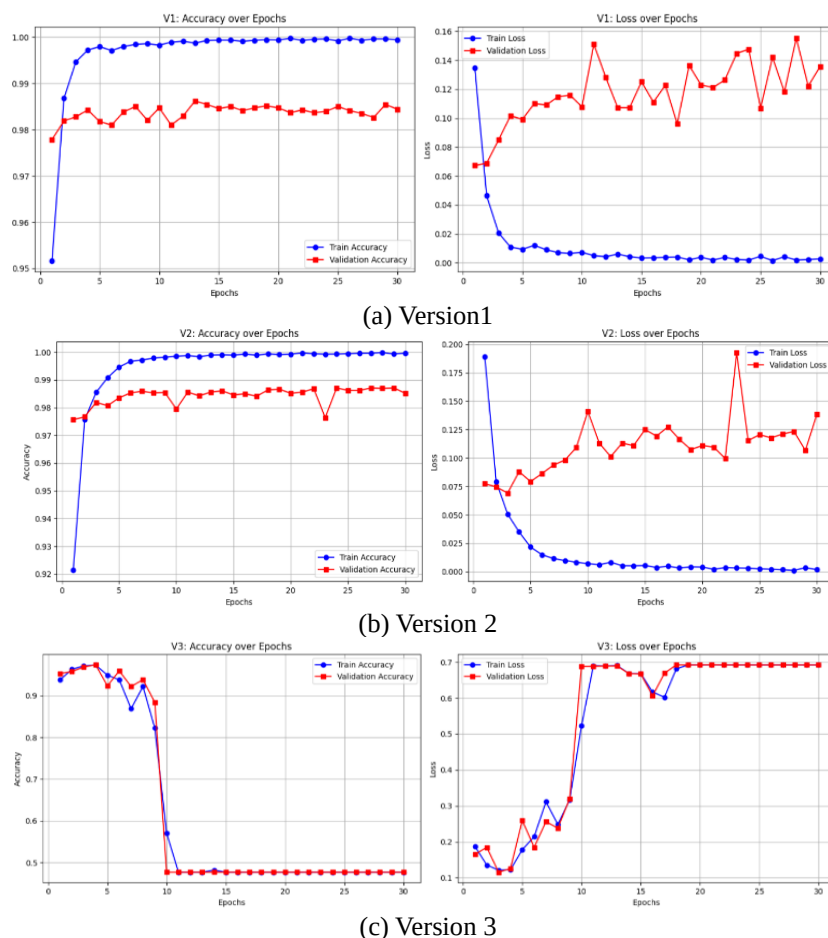
### Accuracy and Loss Curve

The variance among expected and actual values from the training data is measured by training loss. Gradient descent is used to decrease it during training by modifying model parameters. Calculating validation loss on a separate dataset allows one to assess the model's functionality on unknown data. Early underfitting is evident when training loss is large and validation loss is low. Training accuracy measures effectiveness of the model's learning from training data. Validation accuracy demonstrates how well the model generalizes to novel, untested data. **Fig 5** shows the train/validate accuracy curve of version 4 BERT model. The training accuracy approached 99.93%, while validation accuracy stabilized around 98.2–98.4%, indicating slight overfitting due to small, stable updates to the weights with  $1e-5$  learning rate, dropout value of 0.1 used to prevent overfitting and 0.6 of threshold value used for balanced performance and the model was trained for 30 epochs.



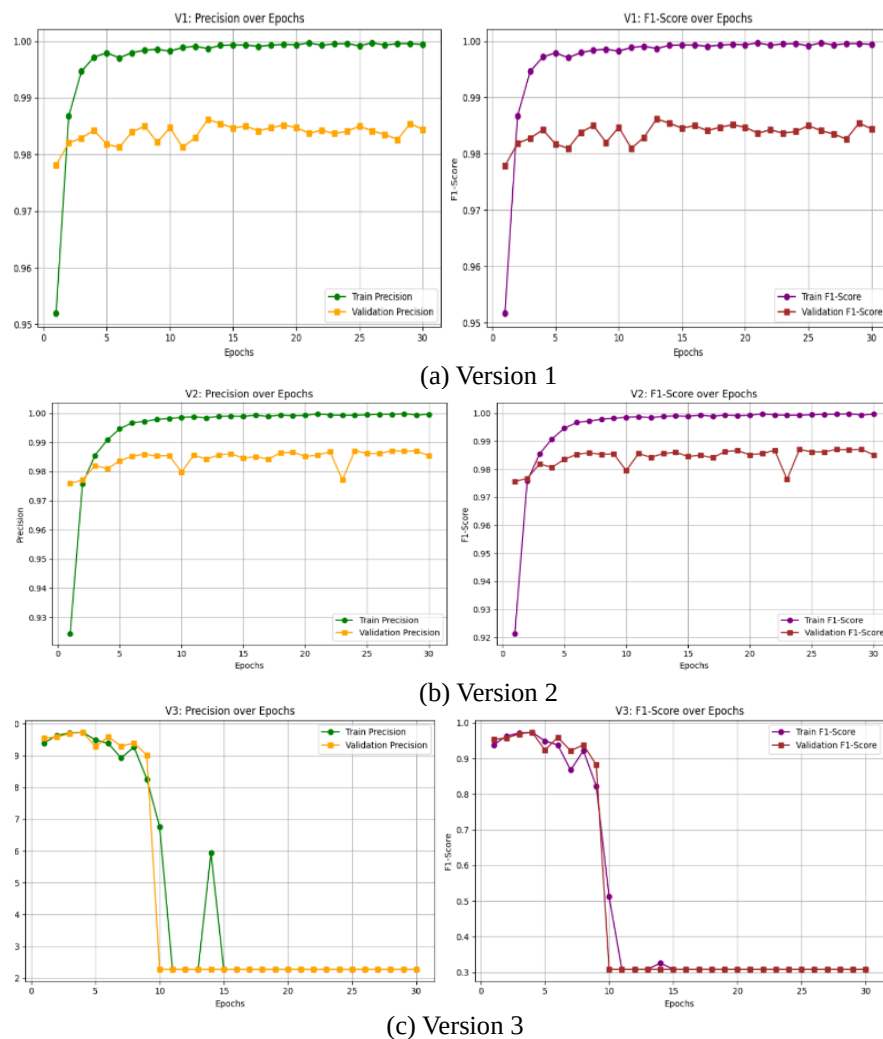
**Fig 6.** Train/Validation Accuracy of Version4 BERT Model.

The graphs in **Fig 6** shows various versions of proposed model by varying the Hyperparameter values, where validation accuracy stays constant but training accuracy increases gradually and validation loss is moderately high in comparison with training loss. The training process spanned for 30 epochs, during which accuracy and loss were tracked.



**Fig 7.** Various Train/Validation Accuracy/Loss of BERT Model.

Similarly, the Train and Validation Precision and F1-score is also plotted on various versions of BERT model by changing the Hyperparameter values as shown in **Fig 7** below.



**Fig 8.** Various Train/Validate Precision/F1-Score of BERT Model.

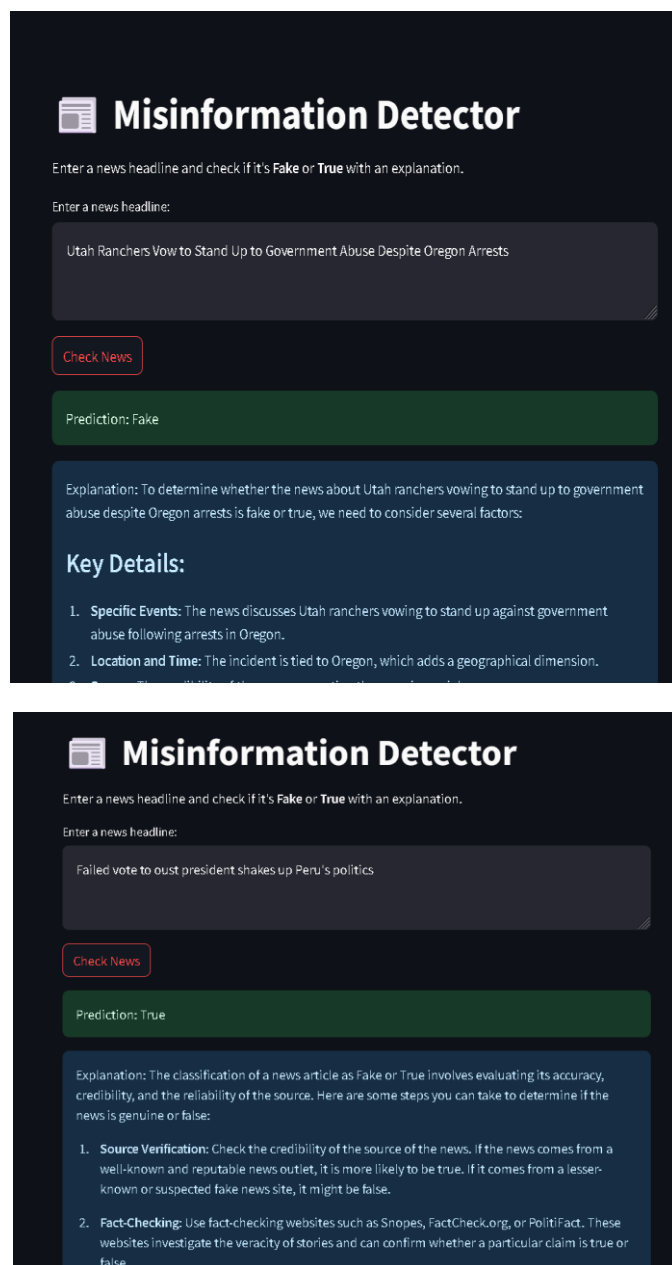
### Large Language Model Meta AI (LLaMA)

A potent transformer-based language model called LLaMA [22] was created by Meta to produce and comprehend text that is human-like. It is essential for detecting fake news because it provides thorough, contextually relevant justifications for the predictions produced by classifiers such as BERT. Although BERT is accurate in classifying news as either “real” or “fake,” it functions as a black-box and provides no logic. To fill this gap, LLaMA generates natural-language explanations of why a piece of content might be incorrect, pointing out logical errors, inconsistent information, or a lack of reliable sources. This increases openness, aids people in comprehending AI choices, and fosters trust particularly in fields that are politically or socially sensitive. LLaMA is a potent tool for explainable AI in disinformation detection since it can be readily modified for various audiences and material genres by prompting.

### Streamlit Application

To create an interactive online application that allows users to enter news text and receive immediate results, Streamlit is utilized in misinformation identification. The backend model, which is usually a trained machine learning or NLP model, examines the incoming text and makes predictions on its authenticity. Streamlit makes it simple to integrate this approach using a simple Python interface. By pasting news stories, users can view predictions along with explanations or confidence scores with just a single click through structured prompts, integrating LLaMA for explanation and BERT for prediction to improve prompting for false news identification. A classification prompt like “Determine whether the following news is fake or real: [news text]”, is given to BERT when the news sentence “Utahn ranchers vow to stand up to government abuse despite Oregon arrests” is entered. If BERT identifies emotionally charged language and patterns associated with false information, it may predict the news sentence as Fake. An explanation prompt such as “Explain why the following news might be fake: [news text], Prediction: Fake” is then used to deliver this prediction to LLaMA. Then, LLaMA says that the statement is ambiguous, utilizes claims of “government abuse” that are frequently seen in politically motivated disinformation, and lacks specific details. This prompt-driven method creates a more transparent

and dependable system for detecting fake news by fusing the interpretability of LLaMA with the detection BERT accuracy. BERT is serving as backend for news classification, generates predicted label, it is then passed to LLAMA API which captures broader contextual information from large scale text with the Streamlit frontend to provide news explanations, like how HTML and CSS structure and display web content. To improve comprehension, as demonstrated in **Fig 8** some of the visuals that are seen in the Streamlit app interface is shown representing a news article as either Fake/True by the detection model. This makes methods for detecting misinformation quick, easy to use, and accessible.



**Fig 9.** Samples of Streamlit App on Fake/True News.

### Comparative Analysis

As demonstrated, the BERT+LLAMA (LAMBERT) model outperformed or matched state-of-the-art techniques with a train accuracy of 99.93% as shown in **Table 4**. This high performance is attributed to the corresponding strengths of BERT's contextual embeddings and LLAMA's generative reasoning. The balanced dataset also contributed to robust generalization, making this model effective for real-world political misinformation detection. This table below presents a comparative analysis of different FND models based on their classification accuracy. The BharatAuthenticNet (BAN) model [8] from the Bharat Fake news study achieves the highest accuracy of 98.49%, indicating strong performance in distinguishing false from true news. The Ensemble Model [23], used as a baseline, records a train accuracy of 98.18%, demonstrating competitive but slightly lower performance. The SheepDog model [4] achieves 98.40% accuracy, positioning it between the BAN and Ensemble models. This comparison is demonstrated as shown in **Table 5**.

**Table 5.** Comparison of Proposed Work with Existing Work by Running Datasets of Existing Work in Working Environment of Proposed Work

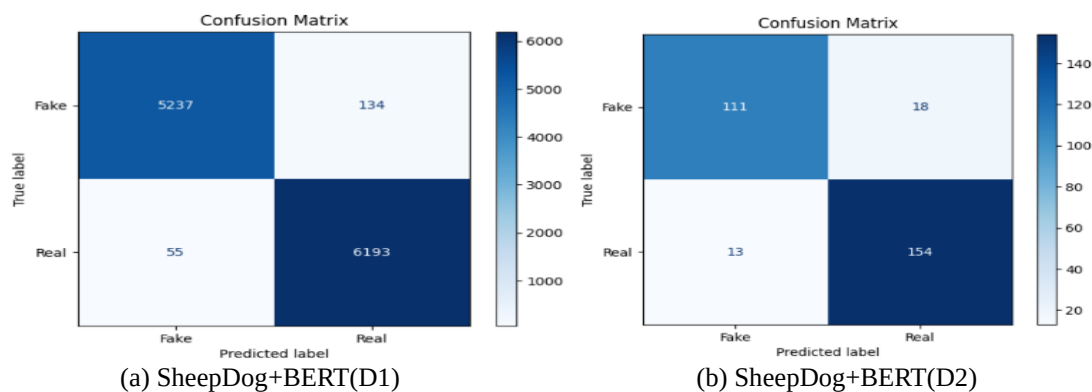
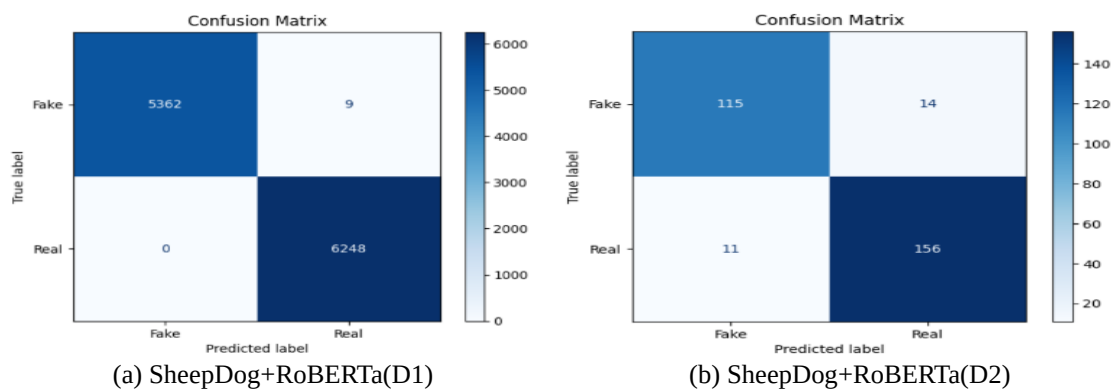
| Reference             | Dataset                           | Model          | Train Accuracy (%) | Test Accuracy (%) |
|-----------------------|-----------------------------------|----------------|--------------------|-------------------|
| [4]                   | PolitiFact                        | BERT           | 98.40              | 85.80             |
| [8]                   | BharatFakeNewsKosh (BFNK)         | BERT           | 98.50              | 58.43             |
| [23]                  | LIAR                              | BERT           | 98.18              | 97.89             |
| <b>Proposed model</b> | <b>Fake and Real News dataset</b> | <b>LAMBERT</b> | <b>99.93</b>       | <b>98.40</b>      |

Additionally, both custom datasets named Fake and Real News dataset denoted as D1 and the benchmark dataset named PolitiFact [4] denoted as D2 are used to reproduce the implementation of the current work. Few model configurations were used in the trials. The tokenizers used are “bert\_base\_uncased” and “roberta\_base” for producing contextualized embeddings. The models are trained for 20 epochs, with a batch size of 16, learning rate of  $2e-5$ , an ADAM optimizer as hyperparameter tuning of the model. When the models were trained and tested using custom dataset D1 with 44919 records, the findings consistently demonstrate that the models obtained higher accuracy related to the benchmark dataset D2 with 450 records as represented in **Table 6**. This could be attributed by D1 dataset's enhanced class balance, domain relevance, and data quality. To guarantee the best possible model performance on both datasets, hyperparameter tuning is used.

**Table 6.** Comparison of Model Performance Using Existing Models by Running Both Custom (D1) and Benchmark (D2) Datasets

| Ref. | Model Name       | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) | Accuracy (%) | Precision (%) | Recall (%) | F1-Score |
|------|------------------|--------------|---------------|------------|--------------|--------------|---------------|------------|----------|
|      |                  | D1 dataset   |               |            |              | D2 dataset   |               |            |          |
| [4]  | SheepDog+BERT    | <b>98.37</b> | 97.88         | 99.12      | 98.50        | 89.53        | 89.53         | 92.22      | 90.86    |
|      | SheepDog+RoBERTa | <b>99.92</b> | 99.86         | 100.0      | 99.93        | 91.55        | 91.76         | 93.41      | 92.58    |

The confusion matrix of all the combinations of SheepDog+BERT and SheepDog+RoBERTa models on D1 and D2 datasets is as shown below in **Fig 9** and **Fig 10**.

**Fig 10.** Confusion Matrix.**Fig 11.** Confusion Matrix.

## V. CONCLUSION

The system can comprehend false content in an effective manner by combining LLaMA's generative competence with BERT's deep language comprehension. Thorough hyperparameter optimization increases the efficiency and model flexibility to many dissimilar inputs **Fig 11** shows Confusion Matrix. This combination gives the hybrid model the ability to recognize and comprehend text's intricate patterns. LAMBERT, the proposed model, has the maximum training accuracy of 99.93% and testing accuracy of 98.40% as compared to existing work when ran the datasets of previous work in the planned work's workspace. Additionally, the evaluation of the current techniques is made by comparing the performance metrics obtained from running the datasets of the proposed and existing studies where the results obtained are better on the custom (D1) datasets than the benchmark (D2) datasets. Streamlit connects end-user accessibility with AI insights, by converting the intelligence into an interactive, real-time platform. The solution, taken as a whole, is a clever, scalable, and user-focused strategy for preventing digital false stories.

### Future Scope

The LAMBERT model can be adjusted to various domains such as COVID-19, sports, or healthcare news etc., by fine-tuning on each unique dataset, enabling it to capture domain-specific language and maintain good performance since the model is based on pre-trained language architectures and has a high level of generic language comprehension. Also, it can be on cross-lingual misinformation detection or in real-time deployment could.

### CRedit Author Statement

The authors confirm contribution to the paper as follows:

**Conceptualization:** Arati M Chabukswar, Vishwa B, Deepa Shenoy P and Venugopal K R; **Methodology:** Arati M Chabukswar and Vishwa B; **Software:** Deepa Shenoy P and Venugopal K R; **Data Curation:** Arati M Chabukswar and Vishwa B; **Writing- Original Draft Preparation:** Arati M Chabukswar, Vishwa B, Deepa Shenoy P and Venugopal K R; **Visualization:** Arati M Chabukswar and Vishwa B; **Investigation:** Deepa Shenoy P and Venugopal K R; **Supervision:** Arati M Chabukswar and Vishwa B; **Validation:** Deepa Shenoy P and Venugopal K R; **Writing- Reviewing and Editing:** Arati M Chabukswar, Vishwa B, Deepa Shenoy P and Venugopal K R; All authors reviewed the results and approved the final version of the manuscript.

### Conflicts of Interest

The authors declare no conflict of interest

### Data Availability Statement

The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

### Funding

No funding agency is associated with this research.

### Consent to publish

All the authors gave permission to Consent to publish.

## References

- [1]. 25 İCİN 2023+ Sosyal Medya İstatistikleri, Gerçekler Ve Trendler. Accessed: 2023. [Online]. Available: <https://www.websiterating.com/tr/research/internet-statistics-facts/#chapter-1>.
- [2]. Rapoza, K., 2017. Can “fake news” impact the stock market? Forbes.
- [3]. M. Aman, “Large Language Model Based Fake News Detection,” *Procedia Computer Science*, vol. 231, pp. 740–745, 2024, doi: 10.1016/j.procs.2023.12.144.
- [4]. J. Wu, J. Guo, and B. Hooi, “Fake News in Sheep’s Clothing: Robust Fake News Detection Against LLM-Empowered Style Attacks,” *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3367–3378, Aug. 2024, doi: 10.1145/3637528.3671977.
- [5]. V. Agarwal, H. P. Sultana, S. Malhotra, and A. Sarkar, “Analysis of Classifiers for Fake News Detection,” *Procedia Computer Science*, vol. 165, pp. 377–383, 2019, doi: 10.1016/j.procs.2020.01.035.
- [6]. M. Jadhav, M. Patil, P. Giri, and Y. Hande, “Advancing Fake News Detection: A Comparative Analysis of SVM and LR Models and Prospects for Dynamic Model Updating,” *2024 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI)*, pp. 1–5, Nov. 2024, doi: 10.1109/idicaiei61867.2024.10842732.
- [7]. P. Jain, S. Sharma, Monica, and P. K. Aggarwal, “Classifying Fake News Detection Using SVM, Naive Bayes and LSTM,” *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 460–464, Jan. 2022, doi: 10.1109/confluence52989.2022.9734129.
- [8]. M. K. Singh, J. Ahmed, K. K. Raghuvanshi, and M. A. Alam, “A Novel Ensemble Model BharatAuthenticNet for Detecting Fake News: Experimentation and Performance Analysis,” *Arabian Journal for Science and Engineering*, vol. 50, no. 19, pp. 15857–15883, Feb. 2025, doi: 10.1007/s13369-025-09975-1.
- [9]. S. Garg and D. K. Sharma, “Fake News Classification via CNN,” *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, pp. 61–63, Dec. 2022, doi: 10.1109/smart55829.2022.10047560.
- [10]. K. L. Tan, C. Poo Lee, and K. M. Lim, “Fake News Detection with Hybrid CNN-LSTM,” *2021 9th International Conference on Information and Communication Technology (ICoICT)*, pp. 606–610, Aug. 2021, doi: 10.1109/icoict52021.2021.9527469.

- [11]. S. E. V. S. Pillai, R. Avacharmal, R. A. Reddy, P. Kumar Pareek, and P. Zanke, “Transductive – Long Short-Term Memory Network for the Fake News Detection,” 2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), pp. 1–4, Apr. 2024, doi: 10.1109/icdcece60827.2024.10548223.
- [12]. T. S. Camelia, F. R. Fahim, and Md. M. Anwar, “A Regularized LSTM Method for Detecting Fake News Articles,” 2024 IEEE International Conference on Signal Processing, Information, Communication and Systems (SPICSCON), pp. 01–06, Nov. 2024, doi: 10.1109/spicscon64195.2024.10941441.
- [13]. A. Chabukswar, P. D. Shenoy, and K. R. Venugopal, “Detecting Misinformation in COVID-19 Content: A Machine Learning and Deep Learning Approach with Word Embeddings,” SN Computer Science, vol. 6, no. 1, Jan. 2025, doi: 10.1007/s42979-024-03534-5.
- [14]. Ibrahim Michael. “Fine-Grained Languagebased Reliability Detection in Spanish New with Fine-Tuned Llama-3 Model.” In In Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2024), co-located with the 40th Conference of the Spanish Society for Natural Language Processing (SEPLN 2024), CEURWS. org. 2024.
- [15]. S. Raza, D. Paulen-Patterson, and C. Ding, “Fake news detection: comparative evaluation of BERT-like models and large language models with generative AI-annotated data,” Knowledge and Information Systems, vol. 67, no. 4, pp. 3267–3292, Jan. 2025, doi: 10.1007/s10115-024-02321-1.
- [16]. T. Pavlov and G. Mirceva, “COVID-19 Fake News Detection by Using BERT and RoBERTa models,” 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 312–316, May 2022, doi: 10.23919/mipro55190.2022.9803414.
- [17]. G. K. Koru and Ç. Uluyol, “Detection of Turkish Fake News From Tweets With BERT Models,” IEEE Access, vol. 12, pp. 14918–14931, 2024, doi: 10.1109/access.2024.3354165.
- [18]. Md. A. Rakib Mollah, M. Md. J. Kabir, M. Kabir, and Md. S. Reza, “Detection of Fake News with RoBERTa Based Embedding and Modified Deep Neural Network Architecture,” 2023 26th International Conference on Computer and Information Technology (ICCIT), pp. 1–6, Dec. 2023, doi: 10.1109/iccit60459.2023.10441206.
- [19]. A. Chabukswar et al., “Misinformation Detection Using a Hybrid Deep Learning Model with Transformer Based Pre-Trained ALBERT Model,” 2024 IEEE 3rd International Conference on Data, Decision and Systems (ICDDS), pp. 1–6, Dec. 2024, doi: 10.1109/icdds62937.2024.10910538.
- [20]. J. Alghamdi, Y. Lin, and S. Luo, “Towards COVID-19 fake news detection using transformer-based models,” Knowledge-Based Systems, vol. 274, p. 110642, Aug. 2023, doi: 10.1016/j.knosys.2023.110642.
- [21]. A. Chabukswar, P. D. Shenoy, and V. K. R., “A Hybrid DistilBERT-BiGRU Model for Enhanced Misinformation Detection: Leveraging Transformer-Based Pretraining Language Model,” 2024 IEEE Region 10 Symposium (TENSYP), pp. 1–6, Sep. 2024, doi: 10.1109/tensymp61132.2024.10752290.
- [22]. X. Ma, Y. Zhang, K. Ding, J. Yang, J. Wu, and H. Fan, “On Fake News Detection with LLM Enhanced Semantics Mining,” Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 508–521, 2024, doi: 10.18653/v1/2024.emnlp-main.31.
- [23]. S. Hakak, M. Alazab, S. Khan, T. R. Gadekallu, P. K. R. Maddikunta, and W. Z. Khan, “An ensemble machine learning approach through effective feature extraction to classify fake news,” Future Generation Computer Systems, vol. 117, pp. 47–58, Apr. 2021, doi: 10.1016/j.future.2020.11.022.

**Publisher’s note:** The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. The content is solely the responsibility of the authors and does not necessarily reflect the views of the publisher.