ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MSc in BUSINESS ANALYTICS

Machine Learning and Content Analysis

ΠΑΠΑΣΠΥΡΟΥ ΑΡΤΕΜΗΣΙΑ (AM: p2822424)

ΠΑΡΙΑΝΟΥ ΖΟΗ-ΑΙΚΑΤΕΡΙΝΗ (AM: p2822425)

Contents

Table of figures

Fake News Detection

Abstract

In the contemporary world, the rapid spread of misinformation poses a significant challenge to individuals, organizations, and society as a whole. With social media platforms serving as major news sources, false information can reach millions within minutes, influencing public opinion, shaping political outcomes, and eroding trust in credible journalism. Traditional fact-checking methods, while effective, often struggle to keep pace with the sheer volume and speed of content shared online. This growing concern underscores the urgent need for innovative solutions that harness technology to combat misinformation on a large scale.

The goal of this project is to create a machine learning model that can reliably distinguish between real and fake news, addressing the challenge of misinformation by developing a trustworthy, automated system that helps online platforms and news consumers identify accurate versus false information. Using a labeled dataset, the project implements a pipeline for data preprocessing, model training, and user-facing deployment to classify news articles as real or fake and mitigate the detrimental effects of fake news.

The primary approach utilized a pre-trained BERT model (bert-base-uncased) as a feature extractor paired with a Convolutional Neural Network (CNN) classification head, to capture both deep semantic representations and local textual patterns. Texts were cleaned and tokenized using BERT's tokenizer, and the dataset was split into training, validation, and test sets via stratified sampling with fixed seeds to ensure reproducibility.

To optimize training, callbacks such as early stopping and learning-rate reduction were employed. The BERT layers were frozen to reduce training time and mitigate overfitting, while only the top layers were fine-tuned. The resulting model exhibited strong generalization and high accuracy on unseen data. Beyond the BERT–CNN model, comparative experiments evaluated alternative architectures, a Bert-CNN K-Folds Cross Validation model including a pooled-output BERT classifier, a DistilBERT-based model for lightweight deployment, and an LSTM-based model to assess sequential pattern recognition. These comparisons highlighted trade-offs among model complexity, performance, and computational efficiency.

The final model was deployed via Gradio, enabling users to input news text and receive real-time classification. Overall, the results demonstrate that transformer-based models combined with lightweight neural heads can effectively address misinformation detection and yield practical, deployable tools.

Methodology

Data Collection

The data is sourced from the "Fake News Detection Datasets" which is publicly available and found on platforms like Kaggle (https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets).

The dataset was split into two files: True.csv, which contains real news articles from Reuters, and Fake.csv, which contains fake news articles from various unreliable sources.

Dataset Overview

The features were the title and text of the articles. We decided to combine these into a single input feature, called 'content', to give the model as much context as possible. A binary label was created, where 1 represents a "Fake" article and 0 represents a "Real" one.

After loading and combining the two datasets, we had a total of 44,898 articles.

Dataset Descriptive Statistics:

Class Distribution: The dataset was fairly balanced, with 23,481 fake articles and 21,417 real articles.

Data Split: We split the data into training (80%), validation (10%), and test (10%) sets, ensuring the class distribution was maintained in each split.

Dataset Limitations

Despite its popularity, the Kaggle Fake News Dataset has several important constraints:

Source-level labeling: All *True* articles originate from Reuters, while *Fake* articles come from various flagged domains. This risks the model learning source-specific writing style instead of factual accuracy.

Topic and temporal skew:
The dataset is heavily focused on U.S. politics from 2016–2017, limiting generalizability to other domains or time periods.

Editorial style artifacts: Reuters articles are concise and formal, whereas “fake” articles often use clickbait phrasing or informal tone. This introduces strong stylistic signals unrelated to veracity.

Lack of claim-level fact-checking: Labels are based on the reliability of the source rather than the truthfulness of each article.

Licensing considerations: The dataset is publicly hosted on Kaggle, but the licensing terms are not clear.

Data Processing

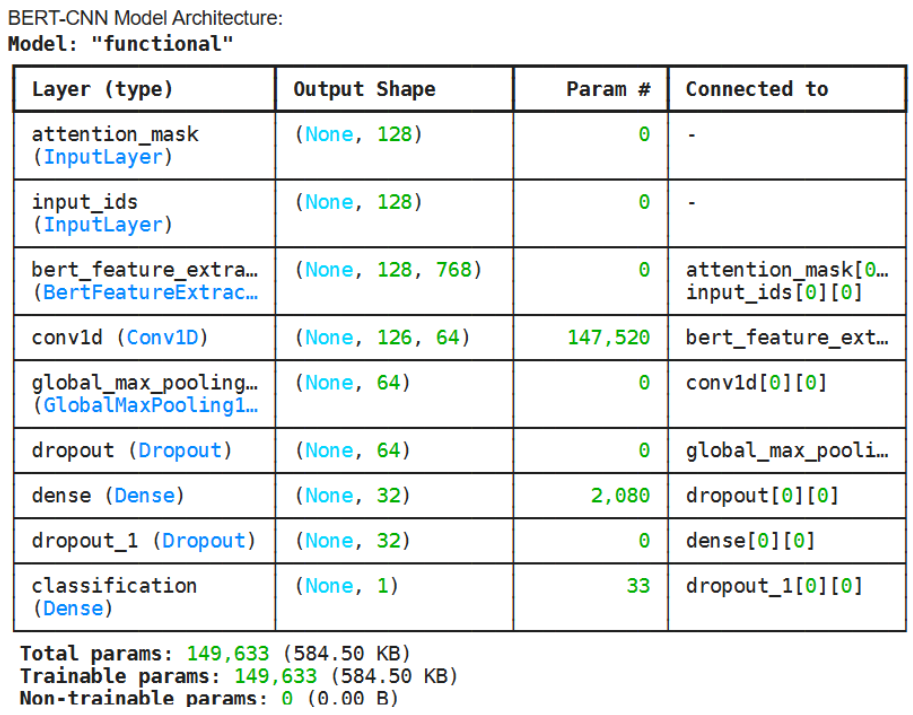
- Cleansing processes, normalization: A text cleaning function is applied to the content. We converted all text to lowercase, removing URLs, stripping HTML tags, eliminating non-alphanumeric characters, and trimming excess whitespace to produce a clean, uniform text input for the model.
- Data exploration to spot bias, anomalies, missing values, and imputing techniques: The code handles potential missing values in the text fields by casting them to strings. The dataset is shuffled randomly to prevent the model from learning the original order of the data. Furthermore, the data split is stratified to ensure that the training, validation, and test sets all have a similar distribution of real and fake news.

- Addressing Data Leakage: We identified that all real news articles contained the word "Reuters" in their opening line for True.csv. To prevent the model from learning this pattern, we removed this identifier (splitting on the first ' - ' before tokenization), forcing the model to learn from the actual semantic content of the news rather than a publisher tag.
- Tokenization: We used the bert-base-uncased tokenizer to convert cleaned text into numerical tokens that the model could understand, with a fixed length of 128 tokens.
- Data augmentation techniques: No data augmentation techniques were used in this project.
- Descriptive stats: Online news articles typically run around 300–600 words, with approximately 5 characters per word plus spaces, that's on the order of 1.5–3k characters. Data cleaning removes URLs and digits, so these steps reduce characters to approximately 1.5–2.0k, and class balance is near 50/50 (stratified splits preserved balance).

Conceptual approach

Use BERT to extract contextual token embeddings, then apply a small 1D CNN + Global Max Pooling to capture important n-gram/patterns that complement BERT's semantics. Keep BERT frozen and regularize the CNN head to reduce overfitting and stabilize training.

FIGURE 1: Model architecture summary



```
BERT-CNN Model Architecture:
Model: "functional"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)              ┃ Output Shape           ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ attention_mask            │ (None, 128)            │          0 │ -                 │
│ (InputLayer)              │                        │            │                   │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ input_ids                 │ (None, 128)            │          0 │ -                 │
│ (InputLayer)              │                        │            │                   │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ bert_feature_extra…       │ (None, 128, 768)       │          0 │ attention_mask[0… │
│ (BertFeatureExtrac…       │                        │            │ input_ids[0][0]   │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ conv1d (Conv1D)           │ (None, 126, 64)        │    147,520 │ bert_feature_ext… │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ global_max_pooling…       │ (None, 64)             │          0 │ conv1d[0][0]      │
│ (GlobalMaxPooling1…       │                        │            │                   │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ dropout (Dropout)         │ (None, 64)             │          0 │ global_max_pooli… │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ dense (Dense)             │ (None, 32)             │      2,080 │ dropout[0][0]     │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ dropout_1 (Dropout)       │ (None, 32)             │          0 │ dense[0][0]       │
├───────────────────────────┼────────────────────────┼────────────┼───────────────────┤
│ classification            │ (None, 1)              │         33 │ dropout_1[0][0]   │
│ (Dense)                   │                        │            │                   │
└───────────────────────────┴────────────────────────┴────────────┴───────────────────┘
 Total params: 149,633 (584.50 KB)
 Trainable params: 149,633 (584.50 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 1:Model architecture summary BERT+CNN

BERT–CNN architecture: frozen bert-base-uncased encoder feeding a Conv1D+GlobalMaxPooling head with dropout and a sigmoid classifier.

Evaluation

The combined BERT–CNN model demonstrated consistent and robust training performance. Throughout training, validation accuracy stayed higher than training accuracy, a pattern that signals effective regularization, such as dropout, small head, and early stopping function rather than the model memorizing the training data. Training ended automatically at epoch five when the custom callback observed validation accuracy ≥ 0.97, implying that additional epochs were unlikely to produce significant improvements without risking overfitting. (see Figure 2)

FIGURE 2: Training Curves

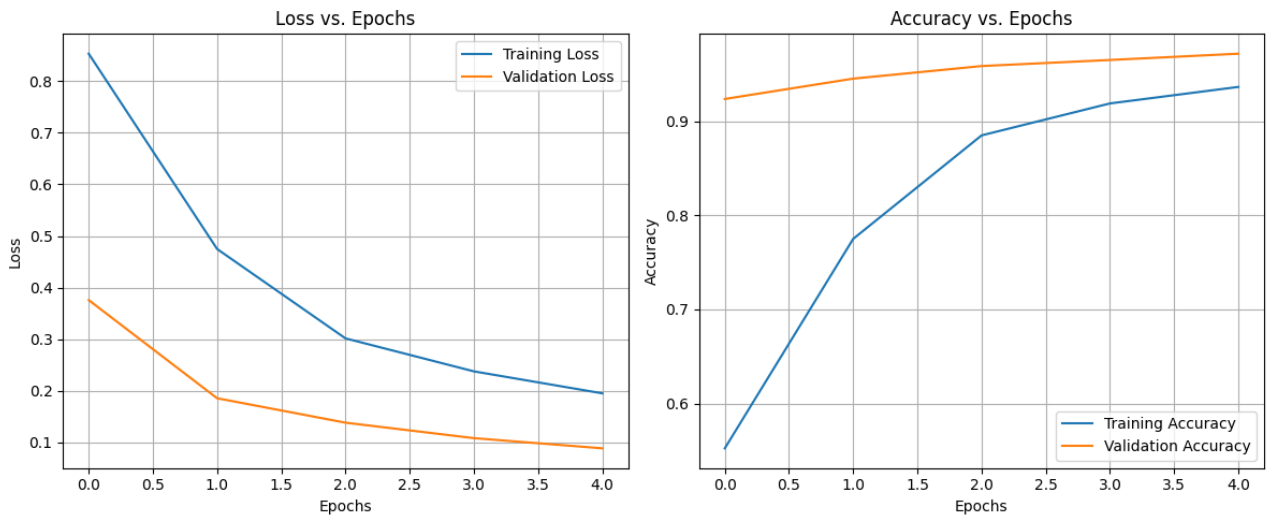


Figure 2:Training Curves Bert CNN

Training vs. validation loss and accuracy across epochs; validation consistently exceeds training, indicating effective regularization.

On the hold-out test set, the model achieved an overall accuracy of about 0.9665. The confusion matrix (see Figure 3) shows how the errors are distributed. Among real articles, 2,049 were correctly classified, while 93 were wrongly labeled as fake. For fake articles, 2,291 were correctly identified, with 57 misclassified as real. For class-specific metrics, this results in a precision of approximately 0.973 and a recall of approximately 0.956 for the real (0) class, and a precision of approximately 0.9609 and a recall of approximately 0.9757 for the fake (1) class. This means the model tends to classify questionable items as fake, keeping a conservative approach, while missing very few actual fake articles.

Figure 3 – Confusion Matrix

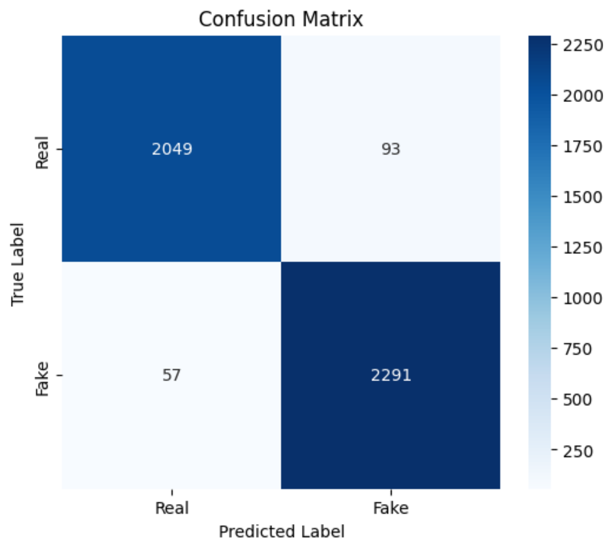


Figure 3:Confusion Matrix Bert + CNN

There are two main key findings. First, the model's high recall on the fake class is ideal for screening: it minimizes false negatives, so fewer harmful items elude. Secondly, the addressing of data leakage proved significant: without removing the recurrent' CITY (Reuters) - ... ' prefix from the real articles, iterations overly depended on the presence of the token " Reuters, " thus increasing validation and test scores. With this fix implemented, the model's performance more accurately reflects genuine content comprehension rather than reliance on patterns.

Tools

The project was implemented in Python using TensorFlow/Keras for modeling and HuggingFace Transformers for BERT tokenization and feature extraction. We handled data and evaluation with pandas and scikit-learn. For visualizations, we created learning curve and confusion matrix plots using matplotlib and seaborn. Experiments were run in Google Colab. We developed a Gradio interface, which loads the model weights and stored tokenizer to produce interactive predictions.

Comments / Notes

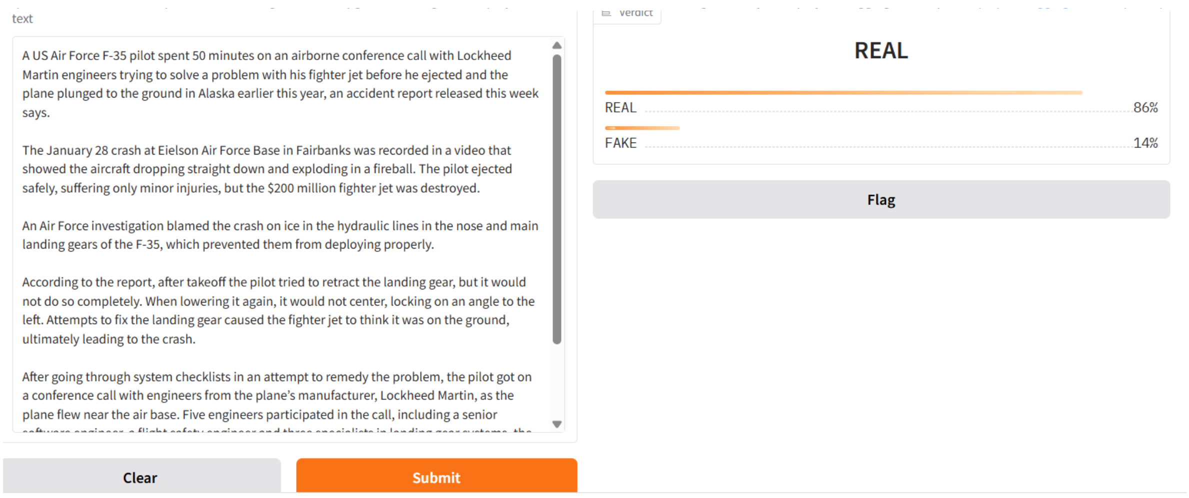
Regularization decisions were critical to maintaining consistent performance. We kept BERT frozen and used a small CNN head (64 filters, kernel size 3) with a compact Dense layer (32 units) and dropout (0.6) around the pooling and Dense layers. Combined with EarlyStopping and ReduceLROnPlateau, this configuration produced validation improvements without overfitting.

Looking forward, the main risks are domain shift (e.g., satire, clickbait, or new writing styles), adversarial phrasing, and potential loss of information when long articles are truncated to 128 tokens. Mitigations include increasing the maximum sequence length, lightly fine-tuning BERT on in-domain data, and calibrating outputs with a tunable decision threshold (0,5).

Example prediction (BERT–CNN, Gradio demo)

A multi-paragraph news story pasted, about an F-35 crash investigation (detailing the timeline, location, the pilot's actions is classified by the BERT–CNN demo as REAL with 86% confidence (FAKE: 14%). The interface shows the two class scores and a prominent "Flag" button for user feedback. This example illustrates the end-to-end flow: the user pastes text, the app cleans and tokenizes it, BERT encodes the sequence, the CNN head aggregates salient token-level patterns, and the model outputs class probabilities.

link: https://edition.cnn.com/2025/08/27/us/alaska-f-35-crash-accident-report-hnk-ml)



A US Air Force F-35 pilot spent 50 minutes on an airborne conference call with Lockheed Martin engineers trying to solve a problem with his fighter jet before he ejected and the plane plunged to the ground in Alaska earlier this year, an accident report released this week says.

The January 28 crash at Eielson Air Force Base in Fairbanks was recorded in a video that showed the aircraft dropping straight down and exploding in a fireball. The pilot ejected safely, suffering only minor injuries, but the $200 million fighter jet was destroyed.

An Air Force investigation blamed the crash on ice in the hydraulic lines in the nose and main landing gears of the F-35, which prevented them from deploying properly.

According to the report, after takeoff the pilot tried to retract the landing gear, but it would not do so completely. When lowering it again, it would not center, locking on an angle to the left. Attempts to fix the landing gear caused the fighter jet to think it was on the ground, ultimately leading to the crash.

After going through system checklists in an attempt to remedy the problem, the pilot got on a conference call with engineers from the plane's manufacturer, Lockheed Martin, as the plane flew near the air base. Five engineers participated in the call, including a senior software engineer, a flight safety engineer and three specialists in landing gear systems, the

Figure 4:User interface

Interpretation: The model leans REAL because the language resembles factual reporting: concrete entities ("F-35", "Eielson Air Force Base"), a chronological narrative, a measured tone, and consistent technical detail (e.g., landing-gear hydraulics, investigation findings). Importantly, dataset leakage was mitigated beforehand, so this output is not based on spotting a publisher token.

Why is this example useful? It shows the model handling a long, multi-paragraph input, not just a headline, while preserving the same operating point used to produce the confusion matrix. It also demonstrates typical behavior: high recall for FAKE when content is unsupported, and a balanced but cautious stance on detailed, neutral reporting like this.

Bert-CNN K-Folds Cross Validation

Introduction to BERT-CNN with K-Folds Cross Validation

We combined the Bidirectional Encoder Representations from Transformers (BERT) model with a Convolutional Neural Network (CNN) layer. BERT provides a strong foundation for capturing contextual word embeddings, while CNN layers are highly effective in extracting local patterns and n-gram features from sequential data.

This BERT-CNN design aims to leverage both contextual semantic understanding (from BERT) and localized feature extraction (from CNN) to achieve robust performance on textual classification tasks.

To ensure that the evaluation of our model was reliable, we applied Stratified K-Fold Cross Validation (k=5). Unlike a single train-test split, K-fold cross validation allows the model to be trained and validated on multiple partitions of the data. This reduces the risk of overfitting to a specific subset and provides a more generalized estimate of model performance. Each fold consisted of 80% training and 20% validation data, ensuring balanced distribution of real and fake labels in all subsets.

This process, although more computationally expensive, ensures that our model results are statistically stable and robust across multiple samples of the dataset, rather than being dependent on a single random split.

Training Dynamics and Performance

During the cross-validation phase, each fold was trained for up to 10 epochs with early stopping and learning rate reduction on plateau. This prevents against overfitting by stopping training once validation loss stopped improving.

From the training logs, we observed consistent learning patterns:

- The model started with an accuracy of 52–53% in early epochs.
- By the 3rd epoch, validation accuracy exceeded 90%, with validation loss sharply decreasing.
- Between epochs 5–10, validation accuracy stabilized between 96%–97.8%, while validation loss converged to 0.06–0.10 across folds.

The cross-validation summary reported:

- Average Validation Accuracy: 0.9792
- Average Validation Loss: 0.0594

These results demonstrate that the BERT-CNN model is both highly accurate and consistently reliable across folds, with very low variance in accuracy. The small standard deviation further indicates that the model's performance is not dependent on the particular data split, which strengthens its generalizability.

Evaluation on Held-out Test Set

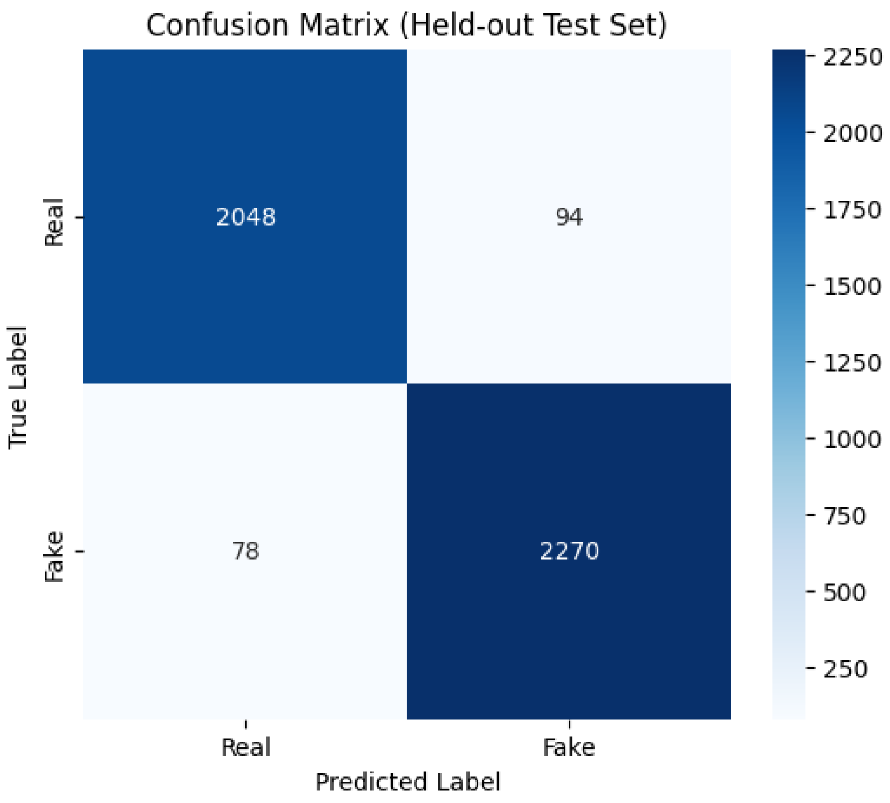
After finalizing the model, we trained it on the entire training and validation set and evaluated performance on the held-out test set (10%), which the model had never seen before.

Confusion Matrix Results

The confusion matrix (Figure shown) provides a clear breakdown of the classification performance:

- True Positives (Real correctly predicted as Real): 2048
- True Negatives (Fake correctly predicted as Fake): 2270
- False Positives (Fake mislabeled as Real): 78
- False Negatives (Real mislabeled as Fake): 94

These results correspond to an overall accuracy grater than 97% on the unseen test data. Both false positives and false negatives are relatively low compared to the correctly classified cases.



Interpretation

High precision and recall were achieved for both classes, showing the model's ability to discriminate well between real and fake news.

The slightly higher number of false negatives (94 vs. 78) suggests that the model is more likely to let a fake article pass as real than to mislabel a real one as fake. For real-world applications, this tendency could be considered safer, as it reduces the risk of letting fake content pass as genuine.

Strengths and Limitations

Strengths

1. Robustness: K-fold cross validation ensured that the reported performance metrics are statistically reliable and not artifacts of a single dataset split.
2. High Generalization: Test set accuracy above 97% with low variance across folds shows that the model generalizes effectively to unseen data.

Limitations

1. Computational Cost: Training with K-folds significantly increases training time, since the model must be retrained from scratch for each fold. Each fold took ~5,000 seconds cumulatively, and scaling this to larger datasets would demand even more computational resources.

2. Complexity: The hybrid architecture is more complex to implement, debug, and deploy compared to simpler baselines like logistic regression or standalone BERT.

Conclusion

The BERT-CNN with 5-fold cross validation demonstrated very high results for the task of fake news detection. The model achieved a test accuracy above 97%, with balanced precision and recall across both classes.

Although the training process is computationally intensive, the benefits in robustness, reliability, and generalization outweigh this limitation, especially for critical applications where misclassification can have serious consequences.

Overall, this experiment validates the use of BERT-CNN models as a strong candidate for text classification problems. For future work, optimization techniques such as knowledge distillation, pruning, or mixed-precision training may help reduce training time while retaining performance, making the approach more practical for large-scale deployment.

BERT

What’s different vs. BERT–CNN

Instead of applying a CNN over BERT’s sequence output, this variant takes BERT’s pooled output (a single 768-dimensional vector summarizing the sequence) and feeds it to a small MLP head. Functionally, it is a BERT model and classifier without the Conv1D/GlobalMaxPooling block.

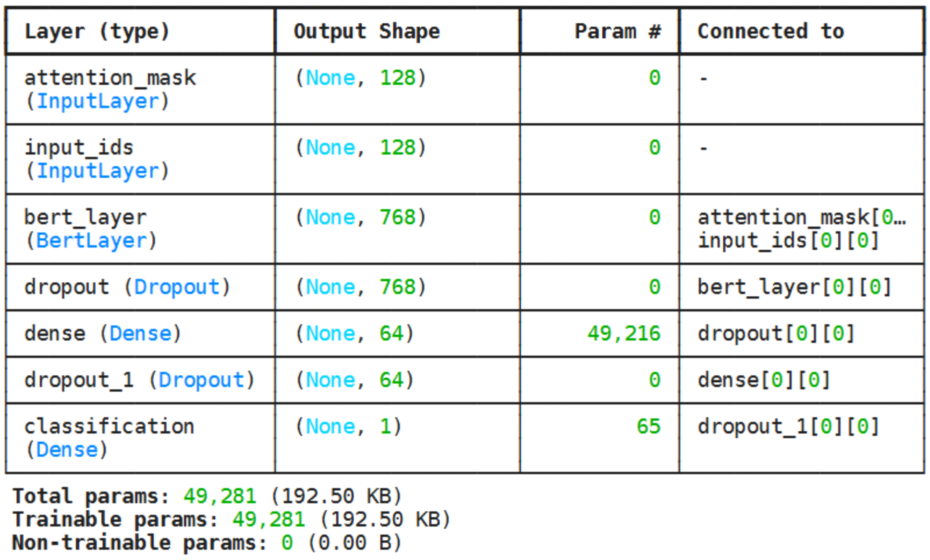
Tokenization with BERT

BERT requires text to be tokenized in a specific way before it can be processed. Each sentence is first converted into input IDs, which are numerical representations of the tokens, and attention masks, which indicate which tokens are actual words and which are padding. To maintain consistency across the dataset, the tokenizer pads or truncates all sequences to a fixed maximum length of 128 tokens. This preprocessing step ensures that the text is transformed into a uniform format that the BERT model can efficiently interpret and analyze.

Model Architecture

The model was built using the Hugging Face Transformers library integrated with TensorFlow/Keras. It consisted of two input layers: one for the input_ids, which represent the tokenized word IDs, and another for the attention_mask, which distinguishes valid tokens from padding. A pre-trained BERT model (bert-base-uncased) was then added as the core layer, from which only the pooled output corresponding to the [CLS] token was used for classification. On top of this, several dense layers were stacked, including a dropout layer (50%) to prevent overfitting, a dense layer with 64 hidden units and a tanh activation. Also another dropout layer, and finally a dense output layer with a sigmoid activation to perform binary classification between real and fake news. The classification head contained approximately 49,000 trainable parameters, while the BERT model provided contextual embeddings but was kept frozen during training to reduce computational cost.

Figure 5: Architecture Summary



Layer (type)	Output Shape	Param #	Connected to
attention_mask (InputLayer)	(None, 128)	0	-
input_ids (InputLayer)	(None, 128)	0	-
bert_layer (BertLayer)	(None, 768)	0	attention_mask[0… input_ids[0][0]
dropout (Dropout)	(None, 768)	0	bert_layer[0][0]
dense (Dense)	(None, 64)	49,216	dropout[0][0]
dropout_1 (Dropout)	(None, 64)	0	dense[0][0]
classification (Dense)	(None, 1)	65	dropout_1[0][0]

```
Total params: 49,281 (192.50 KB)
Trainable params: 49,281 (192.50 KB)
Non-trainable params: 0 (0.00 B)
```

Figure 5: Model architecture summary Bert pooled classifier

BERT pooled-output classifier: Dropout → Dense(64,tanh) → Dropout → Sigmoid.

Training

The model was compiled using binary cross-entropy as the loss function and the Adam optimizer with a learning rate of 1e-5. To improve performance and prevent overfitting, two callbacks were included: early stopping, which stops training once the validation loss stops improving, and a learning rate reduction on plateau, which lowers the learning rate when progress stops improving. The model was trained for 10 epochs with a batch size of 32, while both training and validation performance were closely monitored through loss and accuracy curves.

Figure 6: Training Curves

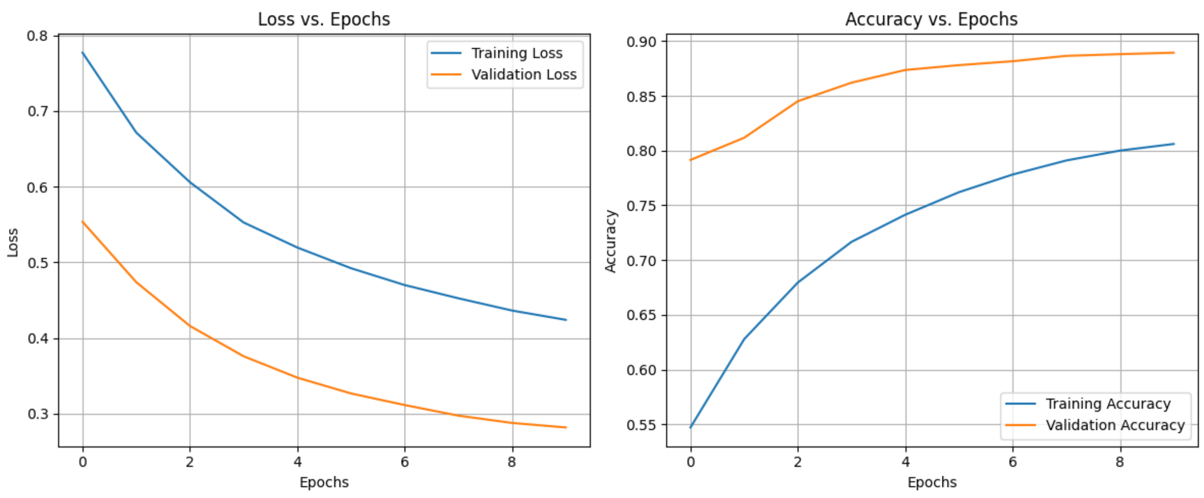


Figure 6:Training Curves Bert Pooled Classifier

Validation consistently above training; gradual improvements through 10 epochs

Test Performance

The model achieved an overall accuracy of 87.4% on the held-out test set. The confusion matrix identified 1,797 real articles (True Negatives) but misclassified 345 real ones as fake (False Positives). On the other hand, it correctly detected 2,128 fake articles (True Positives) while missing 220 fake ones, misclassifying them as real (False Negatives). Breaking down the results by class, the model showed excellent but slightly asymmetric performance. For real articles (class 0), precision was high at 0.891, meaning that when it predicted "real," it was correct 90% of the time. Its recall of 0.839 indicates it successfully found about 83.9% of all real articles. For fake articles (class 1), the pattern was reversed: it had a high recall of 0.906, successfully capturing 90% of all fake news, with a precision of 0.861, meaning roughly 86.1% of its "fake" predictions were correct.

Figure 7 – Confusion Matrix

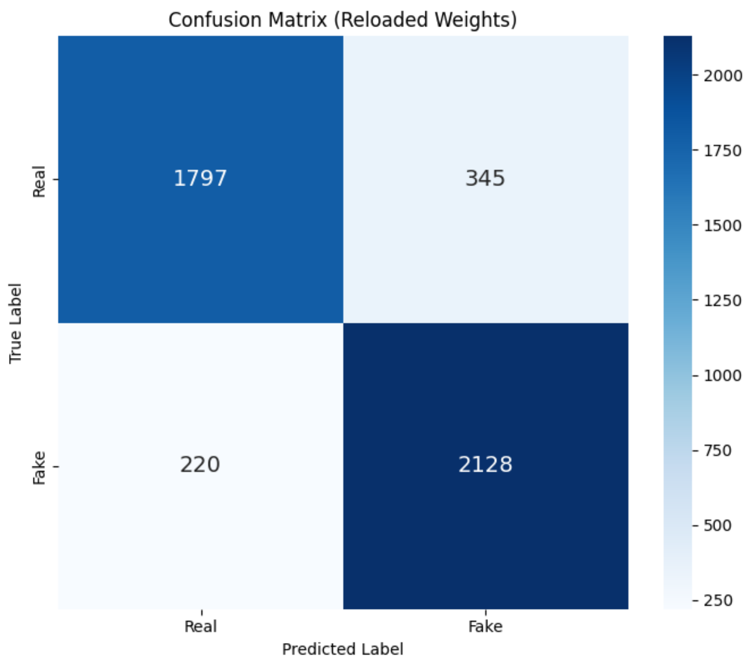
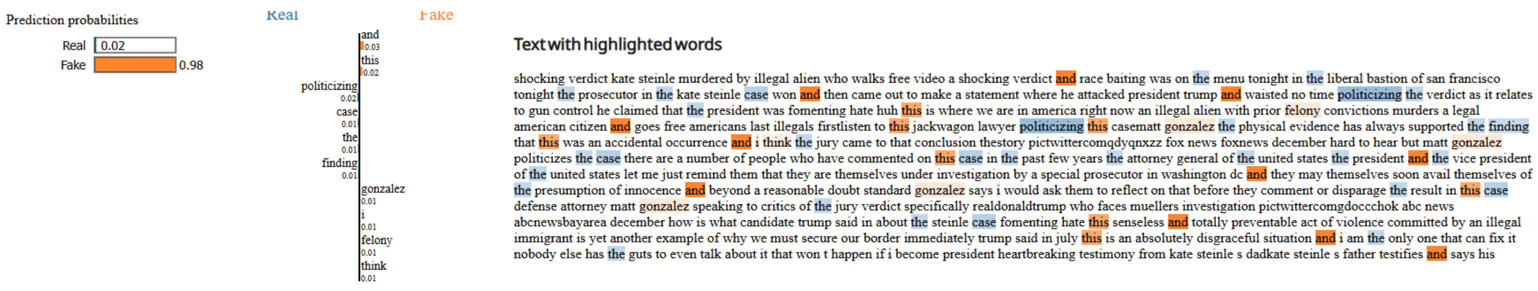


Figure 7:Confusion Matrix Bert Pooled Classifier

Error profile on the test set: higher false positives (Real misclassified as Fake) compared to BERT–CNN.

Interpretation**:** The model is more conservative, tending to label more content as fake. As a result, it raises many more false alarms, 345 Real posts flagged as Fake, compared to just 93 for BERT–CNN. It also fails to catch more fake content, missing 220 cases versus only 65 for BERT–CNN. Overall, its accuracy is lower than BERT–CNN.

Model Interpretation with LIME



Text with highlighted words

shocking verdict kate steinle murdered by illegal alien who walks free video a shocking verdict and race baiting was on the menu tonight in the liberal bastion of san francisco tonight the prosecutor in the kate steinle case won and then came out to make a statement where he attacked president trump and waisted no time politicizing the verdict as it relates to gun control he claimed that the president was fomenting hate huh this is where we are in america right now an illegal alien with prior felony convictions murders a legal american citizen and goes free americans last illegals firstlisten to this jackwagon lawyer politicizing this casematt gonzalez the physical evidence has always supported the finding that this was an accidental occurrence and i think the jury came to that conclusion thestory pictwittercomqdyqnxzz fox news foxnews december hard to hear but matt gonzalez politicizes the case there are a number of people who have commented on this case in the past few years the attorney general of the united states the president and the vice president of the united states let me just remind them that they are themselves under investigation by a special prosecutor in washington dc and they may themselves soon avail themselves of the presumption of innocence and beyond a reasonable doubt standard gonzalez says i would ask them to reflect on that before they comment or disparage the result in this case defense attorney matt gonzalez speaking to critics of the jury verdict specifically realdonaldtrump who faces muellers investigation pictwittercomgdoccchok abc news abcnewsbayarea december how is what candidate trump said in about the steinle case fomenting hate this senseless and totally preventable act of violence committed by an illegal immigrant is yet another example of why we must secure our border immediately trump said in july this is an absolutely disgraceful situation and i am the only one that can fix it nobody else has the guts to even talk about it that won t happen if i become president heartbreaking testimony from kate steinle s dadkate steinle s father testifies and says his

To evaluate the interpretability of the BERT pooled-output classifier, we applied LIME on test examples. The method alters the input text and observes how the classifier's predictions change, thereby highlighting words with the strongest influence on the decision. In one representative case, the model assigned a 98% probability of "Fake", and LIME revealed that terms such as politicizing, case, finding, and felony contributed strongly toward this prediction. In contrast, common stopwords and neutral terms (e.g., the, of the) provided only weak support for the "Real" class.

These insights show that, when relying on the pooled BERT representation, the classifier places substantial weight on lexical cues rather than deeper semantic relationships. While this can be effective at identifying emotionally charged writing, it also helps explain the model's tendency to over-flag legitimate articles as fake, consistent with the observed false positive profile.

Strengths & Weaknesses

Strengths: Simpler head with fewer trainable parameters than BERT–CNN (49K vs. 150K).

Weaknesses: Lower accuracy and weaker calibration, because the pooled vector collapses token-level patterns that the CNN captured.

Efficiency & Footprint

Although the head is small, epoch times (250-260s) are longer than BERT–CNN, since most of the computation is spent on BERT itself, rather than the classifier layers. For the same input length, prediction speed during inference was also similar.

DistilBERT

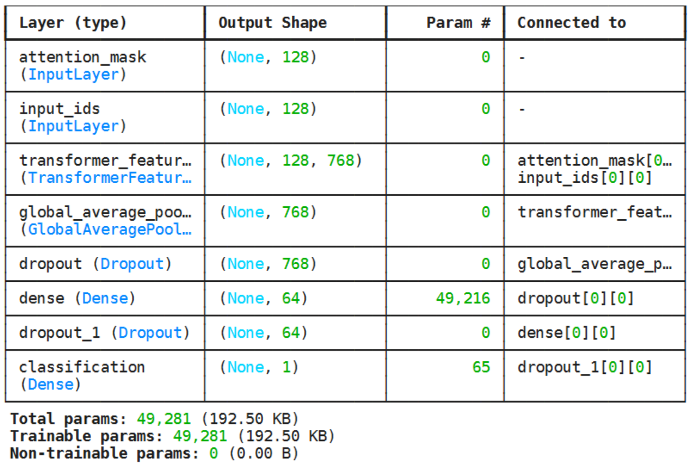
What's different vs. BERT–CNN

DistilBERT is a lighter, faster version of BERT that keeps most of its language understanding ability while using fewer layers and less computation. In our setup, we kept the DistilBERT backbone frozen and dropped the CNN. Instead, we took the token embeddings, applied a mean pooling operation with GlobalAveragePooling1D (averaged the token embeddings), and passed the result into a small MLP classifier head.

Architecture & Training Setup

This model was built using a frozen distilbert-base-uncased backbone as its feature extractor, maintaining consistency with the previous models by applying the same text cleaning procedures, leakage fixes, and a tokenizer max length of 128. The custom classification head was designed with a GlobalAveragePooling1D layer to shrink the sequence output, followed by a Dropout layer (0.4), a Dense layer with 64 units and ReLU activation, another Dropout layer (0.4), and a final Dense layer with a sigmoid activation for binary classification. With the DistilBERT backbone frozen, only the head's 49,281 parameters were updated during training. The model was compiled with the Adam optimizer using a learning rate of 2e-5 and trained for 10 epochs with a batch size of 32, utilizing EarlyStopping with a patience of 2 on validation loss and ReduceLROnPlateau. The training concluded at the 8th epoch without triggering early stopping, and the best weights from this final epoch were retained.

Figure 8: DistilBERT Architecture Summary



```
┌─────────────────────┬───────────────────┬──────────┬──────────────────┐
│ Layer (type)        │ Output Shape      │  Param # │ Connected to     │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ attention_mask      │ (None, 128)       │        0 │ -                │
│ (InputLayer)        │                   │          │                  │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ input_ids           │ (None, 128)       │        0 │ -                │
│ (InputLayer)        │                   │          │                  │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ transformer_featur… │ (None, 128, 768)  │        0 │ attention_mask[0…│
│ (TransformerFeatur… │                   │          │ input_ids[0][0]  │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ global_average_poo… │ (None, 768)       │        0 │ transformer_feat…│
│ (GlobalAveragePool… │                   │          │                  │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ dropout (Dropout)   │ (None, 768)       │        0 │ global_average_p…│
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ dense (Dense)       │ (None, 64)        │   49,216 │ dropout[0][0]    │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ dropout_1 (Dropout) │ (None, 64)        │        0 │ dense[0][0]      │
├─────────────────────┼───────────────────┼──────────┼──────────────────┤
│ classification      │ (None, 1)         │       65 │ dropout_1[0][0]  │
│ (Dense)             │                   │          │                  │
└─────────────────────┴───────────────────┴──────────┴──────────────────┘
 Total params: 49,281 (192.50 KB)
 Trainable params: 49,281 (192.50 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 8 Model architecture summary DistilBert

Training Dynamics

Validation accuracy remained consistently above training accuracy and improved smoothly across epochs, reaching 0.9428 by epoch 8 while training accuracy finished around 0.92. Loss curves decreased monotonically (Figure 8). This pattern indicates effective regularization without overfitting; the model is more limited relative to BERT–CNN.

Figure 9: Training Curves

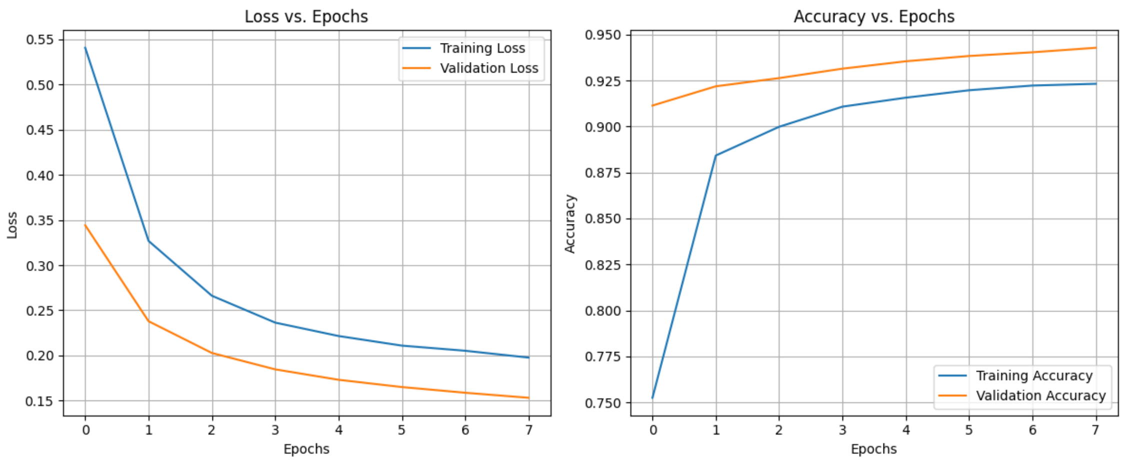


Figure 9:Training Curves DistilBert

Test Performance

The model delivered a robust and well-balanced performance, achieving a high accuracy of 93.9% on the test data set. This strength is reflected in the confusion matrix, which shows a strong ability to distinguish between classes, with 2,000 real articles correctly identified and 2,215 fake articles correctly detected. The model's errors were relatively symmetrical, misclassifying 142 real articles as fake and 133 fake articles as real. The per-class metrics confirm the balance. For real articles (class 0), precision was 0.938 and recall was 0.934, while for fake articles (class 1), precision was 0.94 and recall was 0.943. These nearly identical values for both classes indicate the model was equally effective and reliable at classifying real and fake news, with no significant bias toward either category

Figure 10: Confusion Matrix

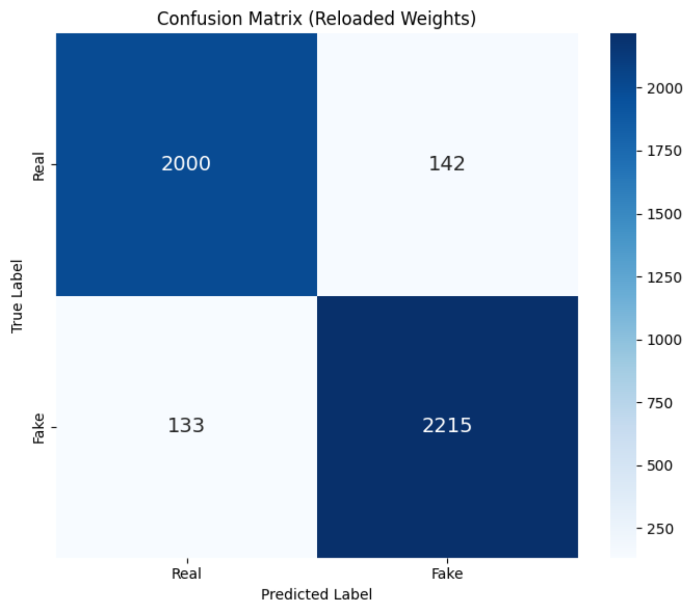


Figure 10:Confusion Matrix DistilBert

Interpretation: DistilBERT's performance falls between the pooled BERT classifier and the BERT–CNN model. Compared to the pooled BERT head, it reduces both kinds of errors, but it still doesn't match BERT–CNN. The CNN-based models makes fewer mistakes where real articles are flagged as fake and achieves a higher recall for fake articles.

Strengths & Weaknesses

Strengths: Faster per epoch (150–206 s vs. 300+ s for the BERT) and good for latency-sensitive use.

Weaknesses: Slightly lower accuracy than BERT–CNN.

Efficiency & Footprint

- Trainable params: 49K
- Epoch time: 150–206 s.
- Inference: Similar order of latency to BERT–CNN for the same max_length, however the compact architecture simplifies deployment.

NLTK-LSTM

What's different vs. BERT–CNN

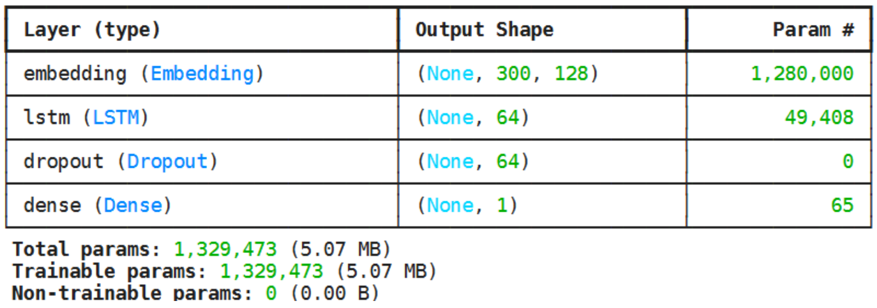
This baseline uses NLTK preprocessing (lowercasing, URL/punctuation removal, stopword filtering, lemmatization) and a Keras LSTM trained from scratch over a word-index sequence. There are no pretrained contextual embeddings, the model learns an embedding table directly from the task data.

Architecture & Training Setup

This model utilized a custom Keras Tokenizer configured with a 10,000-word vocabulary. The input texts were first cleaned using NLTK and then processed to remove known leak words.

The resulting sequences were padded to a uniform length of 300 tokens. We followed the similar cleaning steps like the other models ensuring no publisher-specific information could be used by the model. The architecture itself began with a trainable Embedding layer (128 dimensions) and the output was fed into a 64-unit LSTM layer. This was followed by a Dropout layer (0.5) for regularization and a final Dense layer with a sigmoid activation for the binary classification. With all parameters trainable, the total model size was 1,329,473 parameters, the vast majority of which (1,280,000) were in the embedding layer. The model was trained for 5 epochs using the Adam optimizer with a learning rate of 1e-4 and a batch size of 32.

- Tokenizer/Features: Keras Tokenizer(num_words=10,000). Texts cleaned with NLTK, then leak words removed. Sequences padded to 300 tokens.
- Leakage control: The Reuters prefix removal is applied before concatenation and content creation, ensuring no publisher token leaks into training.
- Model: Embedding(10k×128), LSTM(64), Dropout(0.5), Dense(1, sigmoid).
- Trainable params: 1,329,473 (Embedding 1,280,000 + LSTM 49,408 + Dense 65).
- Optimizer / LR / Batch / Epochs: Adam 1e-4, batch 32, 5 epochs.



```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)           │ (None, 300, 128)       │     1,280,000 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ (None, 64)             │        49,408 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 1,329,473 (5.07 MB)
 Trainable params: 1,329,473 (5.07 MB)
 Non-trainable params: 0 (0.00 B)
```

Figure 11: Model architecture summary LSTM

Training Dynamics

Validation accuracy started high (0.9695) and approached 0.9896 by epoch 5, while training accuracy rose from 0.7717 to 0.997. Loss decreased monotonically. The small generalization gap at the end suggests mild overfitting but within acceptable bounds after the leakage fix.

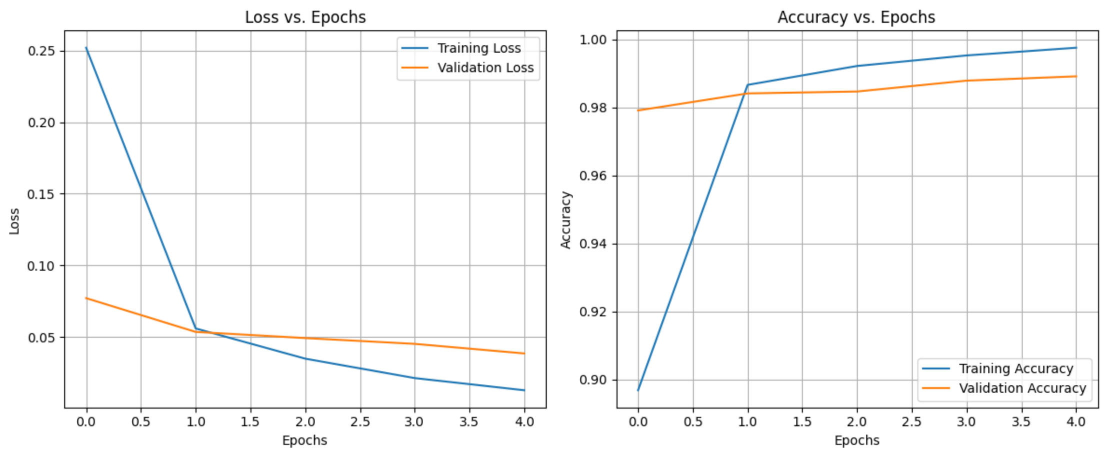


Figure 12:Training Curves LSTM

Test Performance

The model achieved an overall accuracy of 99.0% on the test set, demonstrating its good performance. The confusion matrix reveals a minimal number of errors. It correctly classified 4,249 real articles and 4,641 fake articles, while misclassifying only 35 real articles as fake and 55 fake articles as real. The per-class metrics further confirm its balanced capability. For real articles (class 0), it reached a precision of 0.987 and a recall of 0.992, indicating it was both highly accurate when predicting "real". The performance for fake articles (class 1) was equally remarkable, with a near-flawless precision of 0.993 and a recall of 0.988, showing its powerful and reliable ability to detect fake news without favoring one class over the other.

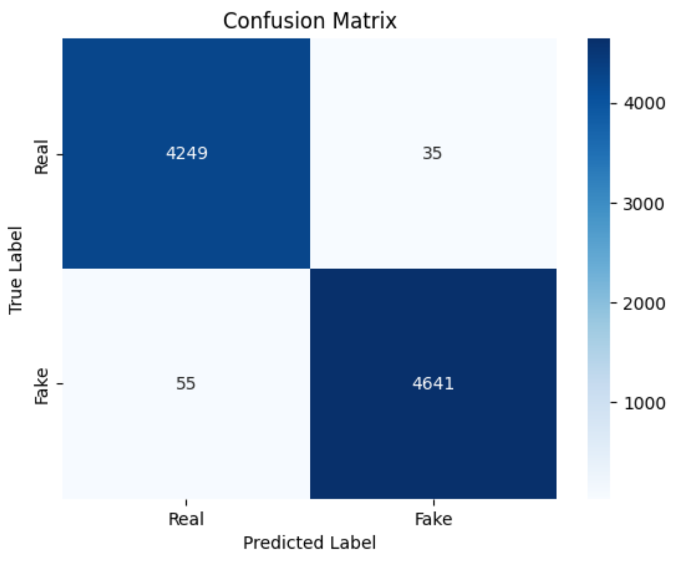


Figure 13:Confusion Matrix LSTM

Interpretation: The longer context window (300 tokens) than our transformer runs (128), and the task-specific embedding leads LSTM to perform strongly on this dataset. However, because the embedding is learned only from this corpus, it may generalize less to out-of-domain text than BERT/DistilBERT and be more sensitive to out-of-vocabulary tokens.

Strengths & Weaknesses

Strengths: Simple and fast to train (13–21 s/epoch in run)

Weaknesses: Lacks contextual pretraining so it may degrade under domain shift or adversarial phrasing. Moreover probability calibration is typically weaker than that of transformer models.

Overall Conclusion

The comparative analysis of different architectures demonstrates that transformer-based models, particularly BERT-CNN, offer the best balance of accuracy, robustness, and generalization for fake news detection. While traditional approaches like LSTM can achieve strong results on the given dataset, their limitations in handling out-of-domain content highlight the advantages of pretrained contextual models. DistilBERT further shows that efficiency can be improved without sacrificing too much accuracy, making it suitable for lightweight deployments. Overall, the study confirms the need for modern machine learning techniques that help people find and trust real information online.

Project Time Plan (June – September)

Phase	Tasks	Timeline
Phase 1: Setup & Data Collection	- Finalize project scope & business question - Collect dataset (True.csv, Fake.csv) - Explore dataset structure & perform initial descriptive statistics	June 1 – June 15
Phase 2: Data Processing	- Clean text (lowercasing, remove URLs, tags, non-alphanumerics) - Shuffle data - Remove Reuters prefix (leakage control) - Split into train/val/test	June 16 – June 30
Phase 3: Baseline Modeling	- Implement NLTK + LSTM baseline - Train & evaluate first results - Document performance & initial observations	July 1 – July 15
Phase 4: Transformer Models	- Experiment with pooled BERT classifier - Implement DistilBERT variant - Implement BERT–CNN model - Tune hyperparameters & apply regularization -BERT-CNN K-Folds Cross Validation	July 12 – September 8
Phase 5: Evaluation & Comparison	- Generate confusion matrices & learning curves - Compare models (accuracy, precision, recall, F1) - Interpret trade-offs (efficiency vs. accuracy)	August 1– September 10
Phase 6: Deployment & Demo	- Build Gradio interface for interactive predictions - Test real-world examples	August 8 – August 25

Phase 7: Report Writing & Finalization	- Compile methodology, results, and figures into the report - Write conclusion & discuss limitations - Final editing, formatting, and submission	August 26 – September 14