Building Trust: FakeStack Replication with RoBERTa

Arash Zarebidmeshki
arashz@usf.edu

Dakota Smith
drsmith@usf.edu

Abstract

This study investigates several crucial aspects of neural network architecture replication and optimization, focusing on performance comparisons and enhancements. Specifically, it addresses the following research questions: How does adjusting hyperparameters such as learning rate, kernel size, and number of epochs influence the performance of a replicated architecture compared to the original findings? Does replacing BERT with RoBERTa in the replicated architecture yield measurable improvements or deviations from the baseline established in the original research??

1 Introduction

Our re-creation of the FakeStack project explores fake news detection, a challenge in today's world where misinformation can have large consequences. Building upon prior research, this project seeks to replicate and optimize a neural network architecture to perform binary classification of news articles as real or fake. By integrating RoBERTa embeddings with CNN and LSTM layers, our re-creation of FakeStack aims to provide accurate classification.

Using publicly accessible Fake News from Kaggle, this project combines multiple NLP technologies to offer insights into the architecture, potential biases, and future growth of fake news detection. Initially, we worked with the LIAR dataset, which provides manually labeled short statements but lacked sufficient context and depth for accurate fake news detection. Due to these limitations, we transitioned to the Fake News dataset from Kaggle, which offered more comprehensive articles better suited for our model's requirements.

The original paper, provided an understanding of the architecture they used however it did not provide code to implement this.

2 Methods

To address the issue of fake news detection, we replicated the model architecture proposed in FakeStack [5], which employs a deep convolutional neural network (CNN) with skip connections and an LSTM model. However, instead of using BERT, we leverage pre-trained RoBERTa embeddings for enhanced feature representation.

2.1 Data Collection

2.1.1 Datasets

The initial step in developing our fake news detection model was to collect and pre-process the dataset. For this research, we utilized the Fake News dataset available on Kaggle (https://www.kaggle.com/c/fake-news/data), which is publicly accessible. Specifically, we used the train.csv file, which consists of 20,800 rows and 5 columns, representing news articles. Each row contains a unique identifier for the article, the article title, the author, the body text, and a binary label indicating whether the article is categorized as 'fake' or 'real' news. Unlike the original methodology in FakeStack, we did not merge the test.csv file with the training data, as it lacks labels and is therefore unsuitable for supervised learning. This decision allowed us to focus exclusively on labeled data for training and evaluation, ensuring the integrity of our classification process.

We initially considered incorporating the LIAR dataset [10], a publicly accessible repository designed for fake news detection. The LIAR dataset comprises 12.8K manually labeled short statements across diverse contexts, offering detailed analytical reports and source document links for each instance. However, we identified certain limitations within the LIAR dataset that could impact the effectiveness of our model. Specifically, the dataset's brevity and limited context in its statements may not provide sufficient information for accurate fake news detection. This concern is echoed in the study An Enhanced Fake News Detection System With Fuzzy Deep Learning, which highlights the limitations of the LIAR dataset and introduces LIAR2 as a new benchmark to address these issues [11].

Consequently, we opted to utilize the Kaggle Fake News dataset exclusively, as it offers more comprehensive articles that align better with our model's requirements.

2.1.2 Data preprocessing

In the initial stage of the study, the train.csv file of the Fake News dataset was pre-processed to prepare the data for analysis and modeling. We began by removing any rows containing empty or NaN values to ensure data quality. The label column was renamed to labels, and its values were converted to float format for compatibility with our model.

To assess the balance of the dataset, we counted the number of occurrences for each label in both the training and testing datasets after splitting the data. Using the train_test_split function with a test size of 0.2, we divided the dataset into training and testing sets. The resulting datasets exhibited a balanced distribution, with the labels approximately split 51-49 between "fake news" and "real news." This ensures that the model has a representative sample for learning and evaluation.

By verifying label counts in both datasets, we confirmed the balanced distribution, which contributes to the robustness of our model training and testing process.

2.2 Architecture

2.2.1 Embedding with pretrained RoBERTa

In recent years, natural language processing (NLP) has made significant strides, transforming fields like sentiment analysis, machine translation, and question answering. A key driver of this progress is the development of transformer-based architectures, which have redefined how models process and understand text. Among these, BERT (Bidirectional Encoder Representations from Transformers) [2] stands out as a foundational model due to its innovative ability to capture contextual relationships within text. By leveraging a large-scale unsupervised pre-training approach, BERT learns language representations that account for the broader context of words and phrases [6]. This contextual understanding enables BERT to excel across a variety of tasks, from extracting sentiments to classifying text.

At the heart of BERT's success is its unique transformer-based framework, which uses layers of encoders combined with self-attention mechanisms to grasp complex patterns. For instance, BERT_BASE includes 12 encoder layers, whereas BERT_LARGE extends this to 24, enhancing its ability to model intricate dependencies. Unlike traditional word embeddings that treat words in isolation, BERT's pre-trained embeddings incorporate the meaning of words within their sentence-level context, making them more robust and semantically rich. In our initial experiments with fake news detection, BERT's embeddings provided a strong foundation for understanding textual subtleties and uncovering nuanced patterns in news articles.

However, upon further experimentation, we switched to RoBERTa (Robustly Optimized BERT Pretraining Approach), which demonstrated improved performance for our binary classification task. RoBERTa, introduced by Liu et al. (2019) [8], builds on BERT's architecture but incorporates several enhancements:

Using a larger training dataset for pre-training. Removing the next sentence prediction (NSP) objective, which can sometimes lead to suboptimal performance for classification tasks. Dynamically changing the masking pattern during training, improving the robustness of the learned representations. Increasing the batch size and training epochs, leading to better generalization. These optimizations allow RoBERTa to extract richer contextual embeddings, making it particularly well-suited for text classification tasks like ours. Fake news detection often requires understanding subtle linguistic cues and contextual relationships within text, and RoBERTa's ability to model such nuances proved advantageous. For our binary classification task, RoBERTa's embeddings provided more discriminative features, resulting in better classification accuracy compared to BERT.

By switching to RoBERTa, we observed improved performance in detecting fake news, as its embeddings better captured the underlying patterns in our dataset. This improvement highlights the importance of model selection in NLP tasks, particularly for binary classification problems where subtle distinctions in the data are critical.

2.2.2 Convolutional Neural Network (CNN)

The CNN architecture is a powerful tool for recognizing localized patterns and extracting features from sequential data, making it an excellent choice for text classification. Originally created for processing images, CNNs have also shown strong performance in natural language processing (NLP) by identifying structural patterns in text. These models generate multi-level representations of input data, enabling the detection of key features necessary for understanding text. This capability is especially valuable for tasks like identifying fake news, where detecting subtle textual cues can play a critical role in classification [9].

A CNN typically includes layers designed for specific tasks, such as convolutional layers to identify features, non-linear activation functions, pooling layers for reducing dimensionality, and dense layers to handle final classification. Filters within the convolutional layers analyze embeddings to identify relevant features, while pooling techniques like max pooling simplify data and highlight the most crucial patterns. Dense layers then use these patterns to make predictions, and dropout is often employed between layers to prevent overfitting.

Deeper CNNs, as opposed to shallow architectures, are better suited for large datasets, as they can learn representations directly from raw text without relying on manual feature design [3]. For instance, Kaliyar et al. [4] proposed FNDNet, a deep CNN with multiple hidden layers, which effectively learns distinct patterns for identifying false information. However, these deeper models, while offering greater generalization, also demand more computational resources and longer training durations.

2.2.3 Skip Convolution Block

Incorporating skip connections into learning models has shown significant advantages for handling complex tasks. Specifically, in fake news detection, these connections enable CNNs to manage textual data more effectively by bridging earlier and deeper layers, which helps analyze both fine-grained details and overarching patterns [1]. By maintaining a direct flow of information across layers, skip connections help integrate diverse feature sets, ensuring that detailed and contextual information work together. Although this method introduces additional computational costs and increases the model's structural intricacy, it supports stable gradient flow, safeguards critical features, and enhances the model's interpretability during training.

2.2.4 Our CNN Implementation

Our CNN model is designed to harness these principles effectively. Instead of using Keras, we implemented the architecture using the torch.nn module, enabling precise customization. The architecture begins with a 1D convolutional layer (torch.nn.Conv1d), configured with 64 filters, a kernel size of 5, and a ReLU activation function. This layer extracts local features from the input sequence using a sliding window mechanism. Following the convolutional layers, we incorporated max-pooling layers (torch.nn.MaxPool1d) with a kernel size of 2, reducing the dimensionality of feature maps while retaining critical features.

To prevent overfitting, we applied dropout regularization with a rate of 0.2 between each layer. The architecture includes multiple convolutional layers with progressively smaller filter sizes (32, 16, and 8) and kernel sizes, interleaved with pooling layers for dimensionality reduction. These layers enable the model to extract increasingly abstract and discriminative features as the data flows through the network.

The inclusion of skip connections further enhances our architecture by allowing the direct flow of information between layers. For instance, the output from an early convolutional layer is combined with the features of a deeper convolutional layer, enabling the model to retain both fine-grained and broader structural information. Similarly, outputs from pooling layers are augmented with preceding convolutional layer features, creating a balance between local and global dependencies.

This combination of convolutional layers, max-pooling, dropout, and skip connections allows our CNN model to effectively capture intricate patterns within news articles, providing a robust framework for binary classification of fake news.

2.2.5 LSTM

We next explore the Long Short-Term Memory (LSTM) model, a specialized type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data [1]. Unlike standard RNNs, LSTMs address the vanishing gradient problem through a gating mechanism that regulates the flow of information, allowing them to identify patterns and relationships spanning long sequences effectively.

The core of the LSTM is its cell state, which functions as a memory unit capable of retaining essential information across extensive sequences.

This functionality is managed by three primary gates—input, forget, and output gates—each controlled by learnable parameters. These gates dynamically adjust to determine how information is updated, retained, or discarded:

- **Input gate**: Manages the addition of new information to the cell state.
- **Forget gate**: Decides which information to remove from the cell state.
- **Output gate**: Regulates the amount of information passed to subsequent layers or used for predictions.

By tuning these gates based on the input sequence, LSTMs achieve an adaptive mechanism for learning and retaining meaningful long-term dependencies.

LSTMs are widely regarded as a robust solution for processing sequential data in NLP tasks. Their ability to understand and model text sequences makes them effective for a variety of applications, such as language modeling, machine translation, and named entity recognition. In the context of this study, LSTMs excel at capturing temporal and contextual relationships in news articles, enabling them to interpret word order and the sequential structure of text. This capability is crucial for learning semantic representations that support accurate classification.

In our implementation, we utilize the `torch.nn.LSTM` module to integrate LSTM layers into the model, leveraging their strength in capturing long-term dependencies. These layers process the sequential embeddings derived from pre-trained RoBERTa models, enabling the extraction of temporal features that enrich classification. By incorporating LSTMs, the model gains an enhanced ability to learn contextual and sequential patterns, boosting its predictive accuracy.

In the next subsection, we introduce a novel hybrid approach that combines skip connections and a deep CNN-LSTM architecture. This design integrates the complementary strengths of both methods, improving the model's ability to classify fake news with greater resilience and precision.

2.3 FakeStack: Final Model Architecture

The proposed model, **FakeStack**, builds on the architecture presented by Keya et al. and introduces key modifications to enhance its performance in detecting fake news. By integrating **pre-trained RoBERTa embeddings**, **convolutional layers with skip connections**, and **LSTM layers**, FakeStack captures both local patterns and long-term dependencies, addressing the challenges of fake news detection.

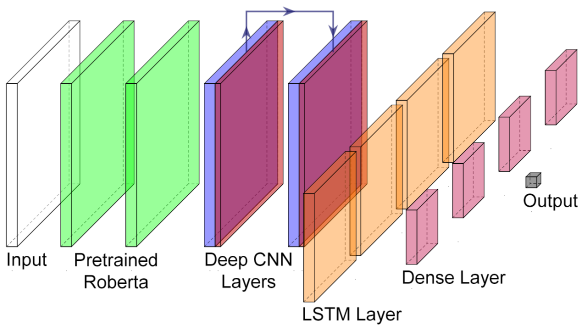

Figure 1: FaceStack architecture Overview

The architecture of FakeStack consists of several stages, starting with embedding generation, followed by feature extraction, temporal modeling, and final classification. The following sections provide a comprehensive breakdown of each component.

2.3.1 Embedding Generation with RoBERTa

The first stage of the FakeStack model involves embedding generation using the pretrained `roberta-base` model from Hugging Face. RoBERTa, an optimized variant of BERT, offers robust contextual embeddings by leveraging a large corpus and dynamic masking strategies.

Input text is tokenized and passed through the RoBERTa encoder to generate high-dimensional embeddings:

$$X \in \mathbb{R}^{m \times n \times d},$$

where m is the batch size, n is the sequence length, and d is the embedding dimension (768 for RoBERTa). These embeddings encapsulate the semantic relationships between words in the text. To mitigate overfitting, a dropout layer with a rate of 0.3 is applied:

$$X = \text{Dropout}(R(\text{tokens})).$$

The RoBERTa embeddings serve as the foundation for feature extraction, providing a rich representation of the input text.

2.3.2 Feature Extraction with CNN Layers and Skip Connections

To capture localized patterns within the text, the RoBERTa embeddings are passed through a stack

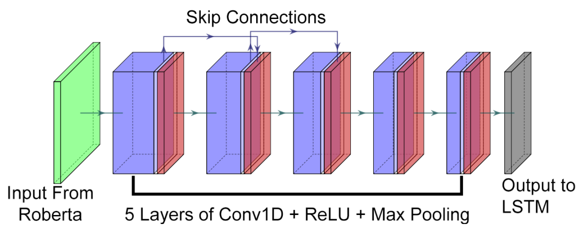

Figure 2: CNN Layers Overview

of one-dimensional convolutional layers. Each convolutional layer learns hierarchical features using kernels of varying sizes, enabling the model to detect linguistic cues at different granularities.

Each CNN layer consists of:

- **Convolution Operation:** Applies learnable filters to the input tensor to extract local features.
- **ReLU Activation:** Adds non-linearity, enabling the model to learn complex patterns.
- **Max-Pooling:** Reduces the spatial dimensions by retaining the most salient features.
- **Dropout:** Regularization with a rate of 0.2 prevents overfitting.

To improve gradient flow and feature reuse, **skip connections** are introduced. These connections propagate features from earlier layers directly to deeper layers via 1×1 convolutions, ensuring that the model captures both low-level and high-level features. The filter sizes progressively reduce across layers: $64 \rightarrow 32 \rightarrow 16 \rightarrow 8$. This hierarchical architecture enables the model to extract increasingly abstract patterns.

Algorithm 1 Feature Extraction with CNNs

```
1: Input: Contextual embeddings $X$ from RoBERTa
2: for each CNN layer do
3:     Apply convolution
4:     Apply ReLU activation
5:     Apply max-pooling and dropout
6:     Add skip connections with $1 \times 1$ convolution
7: end for
8: Output: Processed features $F$
```

2.3.3 Temporal Modeling with LSTM Layers

While CNNs excel at capturing local patterns, they lack the ability to model sequential dependencies over long distances. To address this, the output from the CNN stack is fed into a Long Short-Term Memory (LSTM) layer. LSTMs are specifically designed to retain important information across long sequences using memory cells and gating mechanisms.

The LSTM layer in FakeStack has 128 hidden units and processes the sequential features from the CNN stack:

$$H = \text{LSTM}(\text{CNN output}),$$

where $H[-1]$ represents the final hidden state, summarizing the sequence.

The LSTM layer enables the model to understand contextual relationships and dependencies within the text, making it particularly effective for tasks like fake news detection, where such nuances are critical.

2.3.4 Classification Layer and Loss Function

The final hidden state from the LSTM is passed through a fully connected layer to generate logits:

$$y_{\text{logit}} = WH[-1] + b,$$

where W and b are trainable weights and biases. These logits are converted into probabilities using the sigmoid function:

$$\hat{y} = \sigma(y_{\text{logit}}).$$

To address class imbalance, the model employs the `Binary Cross-Entropy with Logits Loss` (`BCEWithLogitsLoss`), which combines the sigmoid activation and binary cross-entropy loss into a single operation. The loss function incorporates a positive class weight of 1.5, ensuring the model pays greater attention to underrepresented classes:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\Big[w \cdot y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\Big],$$

where $w = 1.5$ is the positive class weight.

2.3.5 Advantages of FakeStack

The architecture of the FakeStack re-creation combines the unique strengths of its components:

- **RoBERTa Embeddings:** Provide robust contextual representations, enhancing the model's understanding of the semantic relationships in text.

- **Convolutional Layers with Skip Connections:** Extract local patterns and enable efficient feature reuse, improving the model's capacity to learn both low-level and high-level features.

- **LSTM Integration:** Captures long-term dependencies and temporal relationships, allowing the model to understand sequential dynamics within the text.

- **Custom Loss Function:** Addresses class imbalance with a weighted binary cross-entropy.

- **Dynamic Thresholding:** Maximizes evaluation metrics by dynamically adjusting the classification threshold.

By integrating these components, our FakeStack achieves robust and interpretable fake news detection, effectively addressing the challenges posed by this task.

2.4 Evaluation

The evaluation process for our proposed **FakeStack** model focuses on assessing its ability to detect fake news accurately and efficiently. The model was evaluated using a binary classification task, where the objective was to classify news articles as either *fake* or *real*. This subsection details the metrics used, the dynamic threshold optimization, and the steps involved in evaluating the model's performance.

2.4.1 Evaluation Metrics

To comprehensively evaluate the model, we utilized the following standard metrics:

- **Accuracy:** The proportion of correctly classified instances among the total instances.

- **Precision:** The proportion of true positive predictions among all positive predictions, defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}.$$

- **Recall:** The proportion of true positives correctly identified, defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}.$$

- **F1-Score:** The harmonic mean of precision and recall, providing a balanced metric for imbalanced datasets:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision + Recall}}.$$

Among these metrics, F1-score is emphasized as the primary evaluation criterion, given the imbalanced nature of the dataset.

2.4.2 Dynamic Threshold Optimization

Instead of relying on the default threshold of 0.5 for converting probabilities into binary predictions, we implemented a dynamic threshold optimization approach to maximize the F1-score. The steps involved are:

1. Compute probabilities from the model's logits using the sigmoid function.
2. Evaluate multiple thresholds $t \in [0.1, 0.9]$ to find the one that maximizes the F1-score.
3. Use the optimal threshold t^* for final predictions.

This approach ensures the model is tuned to achieve optimal performance on precision and recall.

Algorithm 2 Dynamic Threshold Optimization and Metrics Computation

```
1:  **Input:** Model logits $L$, true labels $Y$, thresholds $T = [0.1, 0.9]$
2:  **Output:** Optimal threshold $t^*$, metrics (accuracy, precision, recall, F1-score)
3:  Apply sigmoid to $L$ to compute probabilities:
        $P = \sigma(L)$
4:  Initialize $t^* = 0.5$, best_f1 = 0
5:  for $t$ in $T$ do
6:      Convert probabilities to binary predictions:
            $\hat{Y} = \begin{cases} 1 & \text{if } P \geq t \\ 0 & \text{otherwise} \end{cases}$
7:      Compute F1-score for predictions $\hat{Y}$
8:      if F1-score($\hat{Y}$) $>$ best_f1 then
9:          Update $t^* = t$, best_f1 = F1-score($\hat{Y}$)
10:     end if
11: end for
12: Compute metrics (accuracy, precision, recall, F1-score) for $t^*$
13: **Return:** $t^*$, metrics
```

2.4.3 Evaluation Procedure

The evaluation procedure for the FakeStack model is outlined as follows:

1. Train the model using the `BCEWithLogitsLoss` loss function and the Adam optimizer.

2. Evaluate the model on the validation set using dynamic threshold optimization to maximize the F1-score.

3. Compute and record all evaluation metrics (accuracy, precision, recall, F1-score) using the optimal threshold.

4. Analyze the model's performance across different datasets and ensure its generalizability.

This comprehensive procedure ensures that the model's performance is robustly assessed, with particular emphasis on handling class imbalance and achieving optimal predictions.

3 Results

3.0.1 Performance Highlights

The evaluation results demonstrate the effectiveness of the FakeStack model in detecting fake news, achieving significant improvements in F1-score compared to baseline models. By combining pre-trained embeddings, CNNs, LSTMs, and dynamic threshold optimization, FakeStack successfully addresses the challenges posed by imbalanced datasets and nuanced text classification tasks.

3.1 Roberta vs Bert

The comparison between BERT and RoBERTa shows that RoBERTa marginally outperforms BERT, but the differences in accuracy, precision, recall, and F1 score are relatively small 1. This suggests that while RoBERTa's training optimizations, such as the removal of the Next Sentence Prediction (NSP) task and the use of larger batch sizes, may offer slight improvements, BERT remains competitive in tasks where these optimizations have a limited impact. The small performance gap could indicate that for this specific dataset or task, BERT's pretraining and architecture are already well-suited, and the additional modifications in RoBERTa do not drastically alter the overall outcomes. Although with expansion to larger datasets, or with exploration to different areas of topics, it is likely that the performance gap would continue to grow in RoBERTa's favor.

3.2 Kaggle Test Dataset Results Performance

The FakeNews dataset used in this research stemmed from a Kaggle competition where users were tasked with building a system that could classify various articles into two categories: reliable (0) or potentially fake (1).

The two implementations of the system were submitted to the contest for scoring purposes (as the contest window has closed). The BERT based system achieved an evaluation score of (Private: 0.98928, Public: 0.98974). The RoBEERTa based model achieved slightly better scores (Private: 0.99175, Public: 0.99230).

Kaggle provided the test dataset without labels, and the scoring was performed on the Kaggle platform, ensuring fairness and consistency across all submissions. Submissions were required to include two columns: id (identifying the article) and label (indicating the predicted class). This competition emphasized the challenge of distinguishing fake news in a structured format while maintaining high predictive accuracy.

3.3 Research Questions and Findings

RQ1: How does adjusting hyperparameters influence the performance of a replicated architecture? Hyperparameter tuning, such as adjusting the number of epochs, kernel, and learning rate, showed incremental improvements in performance. For example, BERT's accuracy increased from 0.973 at epoch 1 to 0.991 at epoch 6, while RoBERTa improved to 0.999. However, the gains compared to earlier epochs are marginal and align closely with the results of the original study.
RQ2: Does replacing BERT with RoBERTa yield measurable improvements? Replacing BERT with RoBERTa led to slightly higher accuracy, with RoBERTa achieving a maximum of 0.999 compared to BERT's 0.991. The difference is not substantial but does indicate a slight advantage of RoBERTa in this context.

4 Conclusion

4.1 Limitations and Project Progression

The LIAR dataset presented several challenges, including limited size and class imbalance, which negatively affected the performance of our model. As shown in Table 2, the evaluation metrics after training with the LIAR dataset were relatively low, with accuracy, precision, recall, and F1 score all hovering around 0.27. These metrics highlighted

Epoch	Training Loss		Validation Loss		Accuracy		Precision		Recall		F1	
	BERT	Roberta	BERT	Roberta	BERT	Roberta	BERT	Roberta	BERT	Roberta	BERT	Roberta
1	0.11	0.03	0.16	0.03	0.973	0.997	0.997	0.997	0.950	0.996	0.973	0.997
2	0.08	0.02	0.09	0.05	0.987	0.993	0.998	0.998	0.976	0.988	0.987	0.993
3	0.02	0.00	0.08	0.02	0.988	0.998	0.988	0.999	0.988	0.998	0.988	0.998
4	0.05	0.00	0.07	0.02	0.991	0.998	0.993	0.999	0.989	0.998	0.991	0.998
5	0.02	0.03	0.09	0.01	0.990	0.998	0.996	0.999	0.984	0.998	0.990	0.998
6	0.02	0.00	0.09	0.01	0.991	0.999	0.995	0.999	0.987	0.999	0.991	0.999

Table 1: Comparison of Training and Validation Metrics Across Epochs for BERT and Roberta Models

the difficulty of training a model on a dataset with such challenges, leading us to shift our focus to the Fake News dataset, which offered a more balanced and robust set of examples for better model performance.

4.2 Future Work

In this study, we implemented an AI architecture that utilizes a binary classification model to distinguish between truthful and fabricated news articles, leveraging the Fake News Dataset. While the current work has demonstrated significant potential in addressing the problem of fake news detection, there are several areas for future exploration that could enhance the model's capabilities and broaden its scope.

4.2.1 Dataset

One promising direction for future work is to expand the dataset to include a wider range of topics beyond the political and world news focus. By incorporating articles from diverse fields such as health, technology, and entertainment, the model's ability to gauge various real-world scenarios could be significantly improved. This expansion could involve using existing datasets or creating a unique dataset set up to cover other topics. The use of other areas of literature would allow further evaluation based on the robustness and adaptability of the classification model.

4.2.2 Model Architecture

In our current model, we have incorporated residual connections only at the layer level. As part of future work, we plan to explore incorporating residual connections at the block level, as well as investigate variations of the model that combine both block-level and layer-level residual connections, or rely solely on block-level residual connections.

In our work, we attempted to replicate the main study, implementing a 5-layer CNN. This approach leaves room for further exploration, such as reducing the number of layers to simplify the model or increasing them to handle larger datasets. Additionally, we could consider training the model for more epochs, although this would be computationally more expensive.

Another key enhancement involves integrating a retrieval-augmented generation (RAG) system into the architecture. A RAG system combines retrieval-based and generative approaches, allowing the model to incorporate factual evidence during the decision-making process. By implementing a self-fact-checking mechanism, the system could cross-reference the information with verified sources before classifying an article as truthful or fabricated.[7] This feature could significantly improve the reliability of the model's predictions and provide users with a transparent explanation of the reasoning process.

4.2.3 Bias

Finally, a thorough evaluation of the model's fairness and bias should be conducted. Given the dataset's origin and focus, biases may exist, particularly in topics related to political leanings or regional coverage. Future iterations could incorporate techniques such as adversarial debiasing to ensure the model performs equitably across diverse contexts.[12]

4.2.4 Final thought

By pursuing these directions, the proposed system can evolve into a more versatile and reliable tool for combating misinformation in an ever-expanding digital landscape.

References

[1] S. Deepak and B. Chitturi. Deep neural approach to fake-news identification. *Procedia Computer Science*, 167:2236–2243, 2020.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Epoch	Validation Loss	Accuracy	Precision	Recall	F1 Score
4.0	1.6751	0.2757	0.2784	0.2757	0.2594

Table 2: Evaluation Metrics for DistilBERT on the LIAR Dataset.

[3] R. K. Kaliyar. Fake news detection using a deep neural network. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pages 1–7. IEEE, 2018.

[4] R. K. Kaliyar, A. Goswami, P. Narang, and S. Sinha. Fndnet: A deep convolutional neural network for fake news detection. *Cognitive Systems Research*, 61:32–44, 2020.

[5] Ashfia Jannat Keya, Hasibul Hossain Shajeeb, Md. Saifur Rahman, and M. F. Mridha. FakeStack: Hierarchical tri-BERT-CNN-LSTM stacked model for effective fake news detection. *PLOS ONE*, 18(12):1–31, 12 2023.

[6] S. Kula, M. Choraś, and R. Kozik. Application of the bert-based architecture in fake news detection. In *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, pages 239–249. Springer, 2021.

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[9] J. C. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto. Explainable machine learning for fake news detection. In *Proceedings of the 10th ACM Conference on Web Science*, pages 17–26, 2019.

[10] W. Y. Wang. Liar, liar pants on fire: A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[11] Cheng Xu and M-Tahar Kechadi. An enhanced fake news detection system with fuzzy deep learning. *IEEE Access*, 12:88006–88021, 2024.

[12] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, page 335–340, New York, NY, USA, 2018. Association for Computing Machinery.