A lightweight cross-attentive Tinybert with LoRA and Bi GRU for fake news detection

Ivan Peleshchak[*,†], Vasyl Lytvyn[†], Victoria Vysotska[†], Oleksii Khobor[1] and Mykhailo Luchkevych[†]

Lviv Polytechnic National University, Lviv, S. Bandery str. 12, 79000, Ukraine

Abstract

This study introduces a lightweight neural architecture built on TinyBERT, incorporating a Cross-Attention mechanism, Low-Rank Adaptation (LoRA), and a bidirectional Bi-GRU layer for the task of automatic fake news detection. The explosive growth of the digital information space has intensified the spread of disinformation, while traditional fact-checking techniques struggle to keep pace. Current neural solutions based on large transformer models are often impractical, as they demand extensive computational resources and lack the ability to explicitly capture semantic mismatches between a news headline and its body. The proposed approach addresses these issues by utilizing a distilled TinyBERT backbone that reduces computational overhead, together with a Cross-Attention block designed to highlight headline–body incongruence. LoRA adapters further minimize the number of trainable weights, enabling efficient fine-tuning, while the Bi-GRU layer preserves sequential context and refines information aggregation. Experiments on the Fake News Classification dataset show that the model surpasses classical machine learning methods, reaching 99% accuracy, an F1-score of 0.985, and a ROC AUC of 0.998. With its efficiency, rapid adaptability to new domains, and strong interpretability through attention visualization, the model is well-suited for deployment in real-time fact-checking systems.

Keywords

transformer models, fake news detection, AI fact-checking, TinyBERT, Cross-Attention, Bi-GRU, LoRA

1. Introduction

The rapid growth of digital content has made "AI-driven disinformation" one of the most serious global threats. Disinformation can undermine elections, polarize societies, and cost the world economy tens of billions of dollars annually. Traditional, manual fact-checking cannot keep up with the speed of these "digital wildfires": news flows spread at a rate of hundreds of thousands of publications per hour, while the full verification of a single article may take journalists from half an hour to several days.

With the emergence of large language models (LLMs), automated verification has become technically feasible; however, the vast majority of existing neural solutions still face significant limitations.

Modern "heavyweight" transformers (such as Longformer-Large and multimodal GNN stacks) contain hundreds of millions of parameters, making them unsuitable for deployment on editorial CMS servers, in browser extensions, or on mobile devices. Single-branch models with CLS pooling perform significantly worse at capturing the main rhetorical trick of fake news – the incongruence between a "sensational" headline and a relatively innocuous news body. Studies show that this mismatch is one of the most informative indicators of media manipulation, yet conventional encoders fail to capture it – requiring a dedicated mechanism for the interaction of two text streams [1]. Moreover, fully fine-tuning large transformer architectures demands substantial

*CIAW-2025: Computational Intelligence Application Workshop, September 26-27, 2025, Lviv, Ukraine

[1*] Corresponding author.

[†] These authors contributed equally.

ivan.r.peleshchak@lpnu.ua (I. Peleshchak); vasyl.v.lytvyn@lpnu.ua (V. Lytvyn); victoria.a.vysotska@lpnu.ua (V. Vysotska); oleksii.r.khobor@lpnu.ua (O. Khobor); mykhailo.m.luchkevych@lpnu.ua (M. Luchkevych)

0000–0002–7481–8628 (I. Peleshchak); 0000–0002–9676–0180 (V. Lytvyn); 0000-0001-6417-3689 (V. Vysotska); 0000–0002–3210–036X (O. Khobor); 0000-0002-2196-252X (M. Luchkevych)

computational resources and time, limiting their applicability in dynamic information environments.

Thus, there is a need for a compact, highly efficient, and interpretable neural network model capable of detecting fake news in real time, particularly through precise analysis of the relationship between headline and body content, while remaining suitable for rapid adaptation to new topics and linguistic contexts without significant resource costs.

To address these issues, this study proposes a new compact neural network architecture. Its backbone is TinyBERT-4L-312D – a distilled transformer that preserves the quality and functionality of BERT-Base while being 7.5 times smaller [2]. On top of its outputs, a bidirectional cross-attention mechanism is constructed between the headline and the news body, making the model sensitive to semantic conflicts. To maintain the number of parameters within the range of several hundred thousand, we employ Low-Rank Adaptation (LoRA): instead of full fine-tuning, only low-rank adapters are updated, reducing the number of trainable weights by a factor of 32 without loss of quality [3]. Additionally, sequential context is aggregated through a lightweight bidirectional Bi-GRU (64 hidden units).

Thus, the proposed network is:

- Accessible – suitable for real-time deployment even on resource-constrained devices (e.g., mobile platforms);
- Specialized – designed to detect headline–body incongruence, which is often overlooked by more general LLMs;
- Interpretable – cross-attention heatmaps and Grad-CAM visualizations over the GRU reveal the reasoning behind classifications;
- Flexible in fine-tuning – LoRA adapters are lightweight enough to adapt the model to a local media environment within minutes.

2. Analysis of recent research and publications

The first substantial attempts at automatic fake-news detection relied on recurrent and convolutional networks. In [4], a two-layer Bi-LSTM augmented with lexical and stylistic features outperformed traditional SVMs and became a springboard for deep approaches. Subsequently, [5] applied hierarchical attention by decomposing text into words and sentences; this architecture added roughly three percentage points to the F1 score, demonstrating the value of multi-level text modeling. The authors of [6] showed that a news item's propagation path on social media is no less informative than the content itself: their hierarchical propagation network markedly improved detection accuracy by integrating temporal and authorial relations.

The next phase of countering disinformation was marked by rapid advances in graph-based and transformer approaches. In [7], a heterogeneous graph network with attention over node types – authors, topics, and the text itself – brought models closer to the real structure of the media ecosystem. Around the same time, [8] showed that deep CNNs combined with LSTM stacks deliver steady gains, though their model approached the limits of available GPU memory.

The advent of distilled transformers [9] enabled injecting external knowledge into hierarchical attention. Meanwhile, [10] demonstrated that combining CNN filters with HAN and simple stylistic cues can boost accuracy without substantially inflating the model.

Researchers then turned to robustness and multimodality. The authors of [11] were the first to leverage cross-attention between headline and body, showing that incongruence between these components significantly improves classification accuracy. In [12], an adversarially trained network was proposed to build resilience to lexical substitutions, but the model became heavier and more resource-hungry. Work [13] fused text, images, and user graphs, surpassing state-of-the-art results, yet at the cost of hundreds of millions of parameters and reliance on social-media APIs.

The survey in [14] concluded that fully fine-tuning large transformers is increasingly impractical; instead, Low-Rank Adaptation (LoRA) allows one to update only a tiny fraction of

weights. This idea was further developed in [15], which showed that small LoRA layers can be ensembled with LLMs to reach results competitive with flagship systems – albeit at a high hardware cost.

Despite these advances, significant gaps remain. Most models still impose excessive hardware demands, making deployment on newsroom laptops or in browser plugins infeasible [16-20]. Many approaches cannot explicitly assess the mismatch between headline and body, even though this imbalance is a defining characteristic of propagandistic material.

The primary objective of this study is to address the aforementioned limitations and to design and experimentally validate a compact neural network architecture based on the distilled transformer TinyBERT, enhanced with a Cross-Attention mechanism, Low-Rank Adaptation (LoRA), and a bidirectional recurrent Bi-GRU layer for automatic fake news detection. The proposed solution is intended to deliver high classification accuracy with significantly reduced computational costs, while enabling explicit analysis of semantic consistency between headline and body text, rapid adaptation to new topics, and seamless integration into real-time fact-checking systems on devices with limited computational resources.

3. Materials and Methods

3.1. Data flow scheme in the model

The data stream in the model begins at the level of raw CSV records, where each news item is represented as a pair of rows – the headline and the body text. These two fragments pass through a preprocessing block that removes HTML tags, normalizes Unicode encoding, and converts the text to lowercase. The cleaned strings are then fed into the TinyBERT WordPiece tokenizer: the headline is transformed into a sequence of up to 64 tokens, while the body is represented as a sequence of up to 448 tokens; if necessary, excess words are truncated, and shorter texts are padded with [PAD] tokens. Both sequences are supplemented with special markers [CLS] and [SEP], and a binary *attention_mask* is generated in parallel to distinguish actual tokens from padding (Fig. 1).

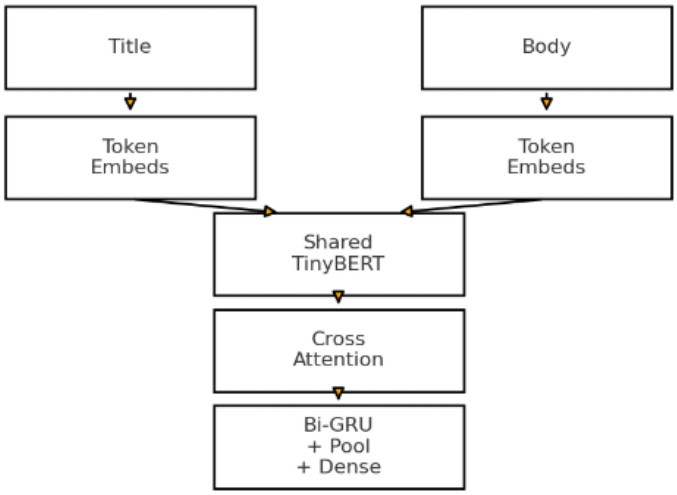

Figure 1: Data flow scheme in the model.

Next, both sections are independently fed into a shared TinyBERT encoder, which keeps its weights frozen during the initial training phase and functions as a contextual transformer: each token receives a 312-dimensional vector enriched with both local and global semantics. The outputs of the headline and body are then passed into the cross-attention module: headline tokens act as queries, while body tokens serve as keys and values, and vice versa. This layer aligns the two text streams, highlighting those word and sentence pairs where semantic conflicts or confirmations

occur. All large projection matrices in this block remain frozen; only their low-rank LoRA adapters are updated, keeping the size of the new trainable space minimal.

The result of the cross-attention is a unified sequence containing the "stitched" information of the headline and body. This sequence is passed into a compact bidirectional GRU with 64 hidden units. The recurrent layer processes it forward and backward, accumulating sequential context, and at each position produces a combined importance vector. To reduce the temporal dimension into a fixed size, global max-pooling is applied: the maximum activations across all tokens are selected, representing the strongest "arguments" for or against the article's credibility. The resulting 128-dimensional vector undergoes dropout and is then passed through a minimal linear layer that outputs a single logit – the fake-news score. A sigmoid function converts this logit into a probability, after which a threshold decision determines whether the news is labeled as fake.

Thus, data flows from raw text to final classification through five logical stages – normalization, tokenization, contextualization via the transformer, cross-attention fusion, and compact recurrent aggregation – each adding informational value while staying within the computational limits of even resource-constrained platforms.

3.2. Data Preprocessing

The Fake News Classification dataset [16], used for training and testing this model, contains 37,106 Fake entries and 35,028 True entries. Each record consists of four fields: Serial Number (starting from 0), Headline (the news headline), Text (the news body), and Label (1 = fake, 0 = real).

Before feeding the news content into the tokenizer, each headline and body undergo a preprocessing pipeline designed to remove inconsistencies that could distort frequency statistics and corrupt the feature space. First, the text is cleaned of residual HTML markup: tags, service attributes, and entity codes are converted into plain characters. Next, Unicode normalization is applied so that all "composite" letters are reduced to a single code – this is particularly important for words with diacritics, which would otherwise be treated as separate tokens.

After encoding unification, the text is converted to lowercase, since TinyBERT is trained in uncased mode and does not differentiate between uppercase and lowercase characters. The next step involves harmonizing typography: quotation marks («…», „…“) and different types of dashes are replaced with standard ASCII equivalents. This operation reduces the number of unique tokens and ensures that the WordPiece tokenizer correctly merges word segments without introducing extraneous tokens.

Subsequently, multiple consecutive spaces are compressed into a single space, while tabs and line breaks are replaced with spaces to prevent the tokenizer from creating empty or partial tokens. Finally, non-printable control characters are removed; these often appear in copied text fragments and can cause rendering errors or an explosion in the number of unknown tokens.

As a result of these steps, a clean, stylistically uniform string is obtained, which can be safely passed to the TinyBERT WordPiece tokenizer. This guarantees a stable vocabulary and overall improves the quality of subsequent text encoding.

3.3. Tokenization and Sequence Construction

After normalization, the text proceeds to the tokenization stage, where raw characters are converted into numerical indices interpretable by the neural network. For this purpose, the WordPiece tokenizer trained alongside TinyBERT is employed. Its vocabulary contains 30,522 units, ranging from whole words to characteristic subwords. The operation of the TinyBERT WordPiece tokenizer (with vocabulary $|V|=30522$) is described by formula (1). For a string s, it returns a sequence of indices $T(s)=(x_1,\dots,x_L)$.

$$T : str \to \{0,\dots,|V|-1\} \tag{1}$$

This segmentation makes it possible to preserve rare terms and proper names by splitting them into shorter, already known lexemes, while keeping the vocabulary compact. The tokenizer reads the string from left to right, selecting the longest substring present in the vocabulary; if a word is not found in full, the algorithm recursively splits it until each part is recognized. Unknown fragments are replaced with the special token [UNK], although after the earlier normalization steps such cases are rare.

During batch formation, the headline and article body are processed separately. The maximum length is set to 64 tokens $L_t = 64$ (including special markers) for the headline and 448 tokens $L_b = 448$ for the body. If a sequence is shorter, it is padded with [PAD] tokens until it reaches the fixed length. If it is longer, the headline is truncated to the required size, while the body is processed using a sliding window with a stride of 128 tokens, ensuring the model can gradually see the entire text without catastrophic loss of the ending.

Next, special markers are inserted into both sequences. At the beginning, [CLS] is placed – a universal flag whose vector is often used for classification in classical BERT-like architectures, though in our case it remains within the sequence as a regular token. At the end of each section, [SEP] is added to signal a logical boundary to the neural network. As a result, two arrays of input_ids are obtained: one for the headline tokens and one for the body tokens.

In parallel, binary attention_masks are generated: elements corresponding to actual tokens are marked with ones, while positions filled with [PAD] are marked with zeros. During further processing in the transformer, these zeros are masked, preventing the network from spending computation on empty positions and ensuring no gradient "leakage" into padding tokens.

Finally, for each token, a positional index is assigned – its order number within the sequence. This index is mapped to a vector from a dedicated positional embedding table and added to the token embedding vector, together forming a complete 312-dimensional token representation. After this step, the headline and body matrices share a unified format and are ready for input into the shared TinyBERT block.

3.4. TinyBERT-4L-312D Block

The core of the entire architecture is the distilled transformer TinyBERT, consisting of only four layers of the "classical" Attention + Feed-Forward block. At the input, it receives the preprocessed embedding matrices of the headline and body, where each token is represented by a 312-dimensional vector.

The first operation is the addition of positional and token embeddings: the lexical vector extracted from the embedding table is summed with the positional vector encoding the token's order in the sequence.

For each input token $x_i \in \{0, \ldots, V-1\}$, the vector representation is formed as: $\bar{e} = E_{x_i} + P_i$, where $E \in R^{V \times d}$ is the token embedding and $P \in R^{L_{max} \times d}$ is the positional embedding.

Table 1
Number of Hyperparameters

Hyperparameter	Value
Number of transformer layers (M)	4
Hidden dimension (d)	312
Number of heads in Self-Attention (h)	12

The resulting matrices are sequentially passed through four transformer layers. Each layer consists of twelve self-attention heads that simultaneously construct query, key, and value

matrices. After the attention weights are normalized with the softmax function, tokens receive contextual vectors enriched with information about lexical surroundings, syntax, and general linguistic patterns. Each such block concludes with a two-layer feed-forward network employing the GELU nonlinearity, along with two residual connections followed by LayerNorm. This internal "rhythm" – attention → normalization → FFN → normalization – is repeated four times, providing sufficient depth to capture complex political and journalistic vocabulary without overloading memory.

A key aspect is that the same TinyBERT copy processes both the headline and the body (2). As a result, weights are effectively shared between the two streams, reducing memory consumption by half compared to approaches that maintain separate encoders for headline and body.

$$H_t = TinyBERT\left(X^{title}, m^t\right), H_b = TinyBERT\left(X^{body}, m^b\right), \tag{2}$$

where $H_t \in R^{L_t \times d}$ and $H_b \in R^{L_b \times d}$ denote the hidden states of the news title and body text, respectively.

In the first training stage, all transformer parameters are frozen: it functions as a ready-made extractor of linguistic features distilled from the larger BERT-Base model. This allows for rapid convergence, as the network optimizes only lightweight LoRA adapters within the cross-attention module and the weights of the Bi-GRU. Only after the classifier stabilizes are the top two transformer layers unfrozen and fine-tuned with a low learning rate – thus avoiding catastrophic forgetting of general linguistic representations while enabling the model to adapt more precisely to the rhetoric of the news domain.

The distilled nature of TinyBERT means that it retains nearly the full linguistic competence of its "parent" BERT-Base, yet contains 7.5 times fewer parameters and operates several times faster.

In summary, the shared TinyBERT provides the essential "alphabetical" foundation for the entire system, offering compact yet deep contextual representations of each token while maintaining resource efficiency – an indispensable requirement for building a practical, widely deployable tool for automated news verification.

3.5. LoRA Block

After both the headline and the body text pass through the shared TinyBERT, each token already carries a full 312-dimensional contextual vector. However, up to this point the two fragments have not "communicated" with each other: the transformer has processed them independently. To enable the model to capture either a semantic gap or, conversely, a confirmation between a sensational headline and the news body, a dedicated cross-attention layer is introduced.

The idea is as follows: the hidden-state matrix of the headline is treated as queries, while the hidden-state matrix of the news body is split into keys and values. Using the scaled dot-product mechanism, attention weights are computed to indicate how strongly each word in the headline "resonates" with each word in the body. The process is then mirrored: body tokens are placed in the query role, and the headline serves as keys and values. The result is two streams of mutual attention that literally "stitch" the two texts together. In practice, this bidirectional alignment is sufficient for the model to detect a classic fake-news tactic: a sensational "click-bait" headline that is not substantiated by the news content. The bidirectional attention mechanism can be described as: Block for the direction title ⟶ body (3)

$$\begin{aligned} Q_t &= H_t\left(W_{Qtb} + \Delta W_{Qtb}\right), \\ K_b &= H_b\left(W_{Ktb} + \Delta W_{Ktb}\right), \\ U_b &= H_b\left(W_{Utb} + \Delta W_{Utb}\right), \\ Attn_{t \Rightarrow b} &= softmax\left(\frac{Q_t K_b}{\sqrt{d_h}}\right) U_b \in R^{L_t \times d}, \end{aligned} \tag{3}$$

where $d_h = d/h = 156$, $h = 2$. In the specialized cross-attention block between the headline and the body text, the number of self-attention heads is reduced to two in order to lower computational costs. This reduction also decreases the risk of overfitting and gradient noise. In particular, with $d_h = 156$, each self-attention head models semantics more effectively due to the increased dimensionality of its representation.

The body → title direction is implemented symmetrically, after which the results are concatenated along the temporal axis $F = [Attn_{t \Rightarrow b}, Attn_{b \Rightarrow t}] \in R^{(L_t + L_b) \times d}$. The raw parameters of the *Q*, *K*, *U* layers and the output projection remain frozen – they were inherited from TinyBERT during distillation and already encode universal linguistic patterns. Adjusting the entire set of hundreds of thousands of weights for a single localized task would be inefficient; therefore, Low-Rank Adaptation (LoRA) is applied. The essence of LoRA is that each large matrix is augmented with a small rank-4 decomposition matrix, which becomes the only "trainable" component.

Each "large" matrix (for example W_{Qtb}) remains fixed, while only its low-rank decomposition is optimized: $\Delta W = \alpha AB$, where $A \in R^{d \times r}$, and $B \in R^{r \times d_h}$. Thus, with rank r=4, the number of trainable weights becomes $d \times r + r \times d_h = 1872$ instead of $d \times d_h = 48672$, yielding a parameter reduction of 96.2%. A scaling factor α=32 (the LoRA default) is applied to preserve gradient norms. In our case, only *A*, *B*, and the LayerNorm biases are trained, while all other weights remain frozen.

After computing attention, the two resulting tensors are concatenated into a single long sequence: now every word in the headline "knows" which sentences it aligns with, and vice versa. An additional LayerNorm and dropout (0.1) stabilize the statistics before passing the sequence to the Bi-GRU, which now processes not two separate fragments but a single, semantically coherent "dialogue" between them.

Thus, the cross-attention layer fulfills two critical functions: it makes the model sensitive to semantic incongruence – the core of most fake publications – and it enables interpretability, since the attention heatmap can be easily visualized to show an editor exactly which headline words "resonated" with which passages of the text. All of this is achieved without parameter explosion, thanks to LoRA-based lightweight fine-tuning.

3.6. Bi-GRU-64 Block

Once cross-attention has completed the "stitching" of the headline and body, we obtain a long sequence of tokens, where each vector now encodes not only its local context but also its semantic relation to the other part of the news item. However, classification requires a single fixed-length document representation. Reducing hundreds of tokens to just a few numbers can be done in several ways, but simple averaging or relying on the [CLS] vector often erases the causal and discursive nuances. For this reason, the next step employs a compact bidirectional GRU with 64 hidden units in each direction $h_k = [\vec{h}_{tb}, \vec{h}_{bt}] \in R^{128}$.

The GRU functions as a learnable pooling mechanism: by traversing the sequence both left-to-right and right-to-left, it progressively accumulates salient information, and through its built-in reset and update gates, it decides which contextual components to retain and which to discard. Each update requires only two parameter sets instead of the three used by LSTMs, thus avoiding unnecessary computation. The choice of 64 hidden units is a compromise: large enough to preserve long-term dependencies between phrases, yet small enough to remain within the bounds of a lightweight architecture.

At the final stage, global max-pooling is applied across the temporal axis $g_j = \max_{k=1,\dots,L} (h_{k,j}), g \in R^{128}$. Instead of relying on the last hidden state, this technique "selects" the strongest activations across all positions, ensuring that no critical sentence is lost, even if it appears far from the text's conclusion. An additional dropout layer (rate 0.2) is applied to the pooled vector, filtering out random spikes and mitigating overfitting.

In this way, the Bi-GRU fulfills several roles simultaneously. It introduces sequential context absent from the "flat" self-attention layers of the transformer, concentrates on the most significant phrases, and, most importantly, achieves this with minimal computational overhead. The result is a 128-dimensional vector that encodes both the linguistic and structural "profile" of the news item – this is the representation passed to the classification layer.

3.7. Classification Layer (Classification "Head")

After the bidirectional GRU compresses the entire news item into a compact 128-dimensional vector g, the final stage is decision-making: whether the publication should be classified as fake or considered credible. This role is fulfilled by a minimalist yet sufficient classification head, consistent with the overall "lightweight" design of the model.

First, the vector g is passed through a Dropout layer with a probability of 0.2. During training, this randomly zeroes out one-fifth of the components, forcing the network not to over-rely on a few "loud" features and preventing overfitting to isolated triggers – such as excessive exclamation marks or cliched words.

Next, a single linear layer produces the logit z, which reflects the cumulative "weight of evidence" for fakeness. To convert this logit into a probability, the sigmoid function $\hat{y}=\frac{1}{1+e^{-z}}$ is applied, mapping any real value into the (0,1) range. During inference, this is the value presented to the user: for example, 0.83 corresponds to an 83% confidence that the article is fake. By default, the decision threshold is set at 0.5; however, after training, the model is calibrated on the validation set to determine the threshold that maximizes the F1 score. This calibration allows tailoring the precision/recall trade-off to the needs of a specific newsroom: some may prioritize avoiding false accusations, while others may focus on catching every fake.

The entire model is optimized using Focal Binary Cross-Entropy (4).

$$\Phi=-\left((1-\hat{y})^{\gamma}\, y\log(\hat{y})+(\hat{y})^{\gamma}(1-y)\log(1-\hat{y})\right),\gamma=2 \qquad (4)$$

In summary, the classification head completes the work of the network while upholding three key principles: minimal resource footprint, the ability to provide detailed explanations for each decision, and flexibility in adapting thresholds or calibrating probabilities for new domains.

3.8. Model Training

The training process begins with preparing the Fake News Classification corpus. The CSV file is stratified into training, validation, and test subsets in a 70:15:15 ratio, preserving the fake/real class balance in each part. To verify metric stability on the same split, five-fold cross-validation is performed. Model training was conducted on a Tesla T4 GPU.

Optimization is carried out using the AdamW algorithm with classical moments (0.9, 0.999) and $\varepsilon=1\cdot10^{-8}$. All "heavy" Frozen-TinyBERT matrices remain excluded from updates – gradients do not pass through them – so memory required for gradient storage is significantly reduced compared to full fine-tuning. Training proceeds in two stages. In the first stage, lasting two epochs, the transformer remains a feature extractor: only the LoRA adapters in the cross-attention block, the projection before the GRU, the GRU itself, and the linear classifier are trainable. These layers use a relatively high learning rate of $2\cdot10^{-4}$, enabling the model to quickly adapt to the rhetoric of news without altering TinyBERT's deep linguistic base. Once the loss curve stabilizes, the second stage begins: the top two transformer blocks are unfrozen but trained with a lower learning rate of $1\cdot10^{-5}$, while other layers continue updating at the initial step. This freeze-schedule preserves TinyBERT's large knowledge base while giving the model just enough freedom to fine-tune context for the discourse domain.

Learning rates in both groups follow a short linear warm-up – cosine decay schedule. The first 10% of iterations serve as warm-up, preventing unstable gradient spikes; afterward, the step

gradually decays to zero, ensuring smooth convergence. At each step, losses are computed using focal binary cross-entropy: the scaling factor γ suppresses the influence of already correctly classified examples and concentrates gradients on hard, borderline cases; label smoothing (ε=0.05) filters out random spikes caused by data noise.

Several measures counter overfitting. Each vector after the GRU passes through dropout with probability 0.2; GRU gate weights undergo mix-out regularization (0.1), partially freezing past values and preventing the model from overfitting to dataset-specific artifacts. Gradients are clipped to a norm of 1.0 when necessary. Early stopping is applied if macro-F1 on validation does not improve for two consecutive epochs.

After the final epoch, the checkpoint with the best F1 score is retained. Stochastic Weight Averaging Gaussian (SWAG) is then applied over the last ten optimizer states: weights are smoothed, providing additional fractional improvements in stability. Finally, the decision threshold is calibrated: validation data determine the value that maximizes F1, which is saved in the configuration file.

In this way, the full data → model cycle remains feasible even on consumer hardware, ensures reproducible results, and allows LoRA adapters to be fine-tuned for new topics or local information landscapes within minutes – all without risking the loss of general linguistic knowledge distilled into TinyBERT.

4. Results and Discussion

We present the results of the experimental comparison between the proposed TinyBERT with Cross-Attention + LoRA + Bi-GRU model and classical "lightweight" machine learning methods, including logistic regression, linear SVM, Bernoulli Naive Bayes, k-nearest neighbors (k = 7), random forest, gradient boosting, and XGBoost. Comparing our model with full fine-tuning of large transformers – where training times are often measured in hours – was deemed impractical.

Experiments were conducted on the English-language Fake News Classification corpus [21], split into training, validation, and test subsets. Evaluation metrics included accuracy, weighted F1-score, and the area under the ROC curve (ROC AUC). The purpose of the study was not only to measure the absolute effectiveness of each approach in distinguishing fake from real news but also to assess the trade-off between classification quality and computational cost during fine-tuning. Interpretation of the results highlights the practicality of integrating a transformer backbone with lightweight adaptation and sequential context modeling to improve both accuracy and stability in text classification tasks.

In the comparative experiments, our model achieved the highest performance across all evaluated algorithms: Accuracy = 99%, weighted F1-score = 0.985, and ROC AUC = 0.998. The results of other models are summarized in Table 2. The class-specific accuracy metrics confirm the high reliability of the proposed architecture across both categories. For the real news class, the model achieved precision = 0.98 and recall = 0.99, while for the fake news class it reached precision = 0.99 and recall = 0.98. This balance ensures a minimal rate of both false positives and false negatives.

From the perspective of computational cost, training the LoRA adapter together with fine-tuning the Bi-GRU in the proposed model required approximately 778 seconds. While this is significantly longer than the training time for linear models (Logistic Regression and Linear SVM ≈ 105 s; Bernoulli NB and K-NN ≈ 88 s), our model delivers superior classification quality, as demonstrated in Table 2. In comparison to full fine-tuning of large transformers – where training can take hours – the choice of TinyBERT reduced the computational load and overall training time considerably (778 s).

Regarding inference, TinyBERT with cross-attention and Bi-GRU achieves an average latency of about 5 ms on a Tesla T4 GPU, making the model well-suited for integration into real-time services.

This compromise between accuracy and resource consumption positions TinyBERT + Cross-Attention + LoRA + Bi-GRU as an attractive solution for real-time fact-checking tasks.

Table 2
Results of the computer experiment

Model	F1-score		Precision		Recall		ROC AUC	Accuracy	F1 Score	Training Time (s)
	Real	Fake	Real	Fake	Real	Fake				
TinyBERT + Cross-Attention + LoRA + Bi-GRU	**0.99**	**0.99**	0.98	0.96	**0.99**	0.98	**0.998**	**0.99**	**0.985**	778.1
Logistic Regression	0.96	0.95	0.95	0.96	0.96	0.95	0.9907	0.96	0.954	106.2
Linear SVM	0.92	0.93	**0.99**	0.87	0.86	**0.99**	0.9955	0.93	0.9291	104.9
Bernoulli NB	0.9	0.89	0.87	0.92	0.93	0.86	0.9585	0.89	0.8892	**88.3**
K-NN (k=7)	0.84	0.84	0.88	0.81	0.8	0.89	0.9252	0.84	0.8446	**88.3**
Random Forest	0.96	0.96	0.96	0.97	0.97	0.95	0.9947	0.96	0.962	1983.7
Gradient Boosting	0.95	0.94	0.93	0.96	0.96	0.92	0.9851	0.94	0.9404	728.4
XGBoost	0.34	0.7	0.9	0.54	0.21	0.98	0.5919	0.58	0.6964	114.8

In the context of confusion matrices, it is evident that classical algorithms handle the trade-off between false positives and false negatives differently. For instance, in the case of logistic regression (Fig. 2), the metrics appear symmetrical; however, its recognition accuracy is approximately 3% lower than that of our proposed model. A drawback of logistic regression is that it performs effectively only when the data distribution is close to linearly separable and fails to capture complex n-gram interactions.

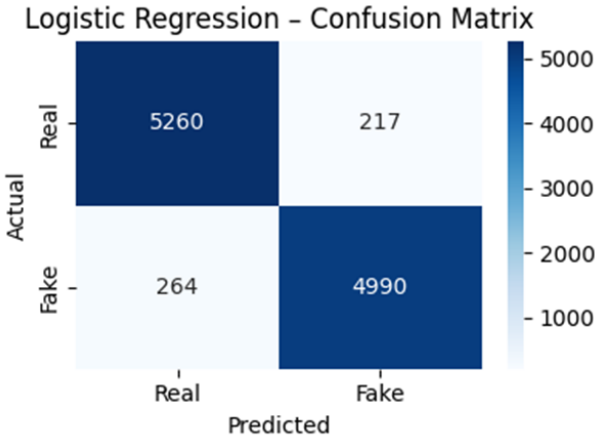

Figure 2: Confusion matrix for logistic regression.

For Linear SVM (Fig. 3), the model reliably avoids missing fake news but fails to identify approximately 14% of real cases, as reflected in the false-negative classifications shown in the upper-right corner of the confusion matrix.

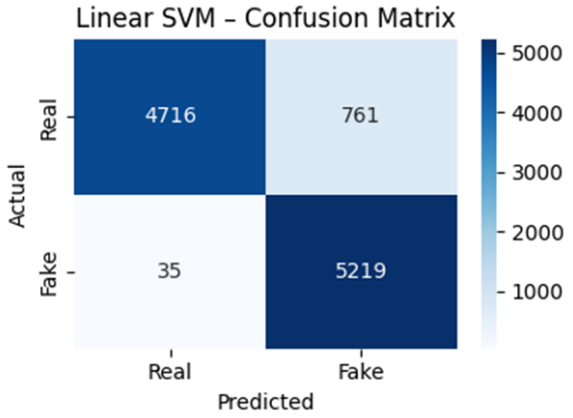

Figure 3: Confusion matrix for Linear SVM.

Bernoulli NB (Fig. 4) exhibits an almost uniform distribution of errors but overall suffers from overly generalized and partially excessive predictions. In contrast, K-NN with k = 7 (Fig. 5) frequently confuses feature-similar examples, resulting in a blurred diagonal in the confusion matrix.

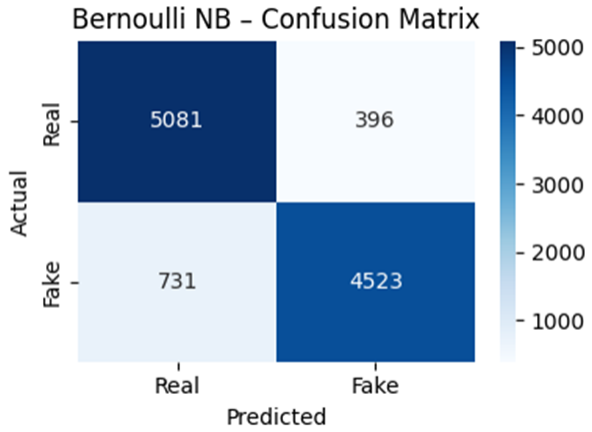

Figure 4: Confusion matrix for Bernoulli NB.

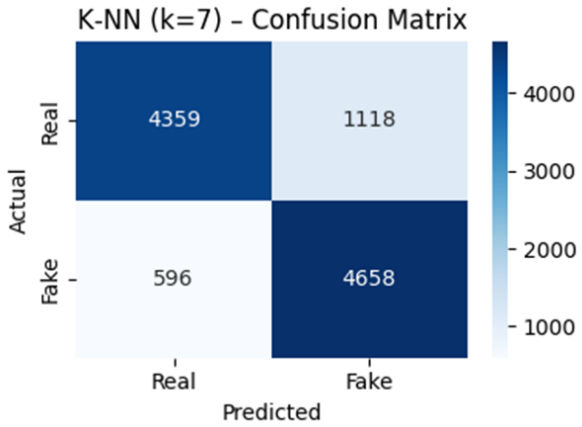

Figure 5: Confusion matrix for K-NN.

In the case of Random Forest (Fig. 6), a much clearer diagonal can be observed; however, some real posts are occasionally labeled as fake due to the model's excessive sensitivity to feature correlations. Overall, the model achieved an accuracy comparable to logistic regression, at around 96%. Nevertheless, it exhibited the longest training time among the classical approaches because of the large number of hyperparameters involved. Gradient Boosting (Fig. 7) demonstrates balanced behavior, although the model is not free from false-positive classifications.

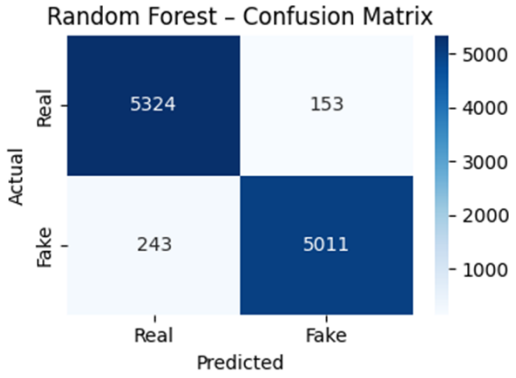

Figure 6: Confusion matrix for Random Forest.

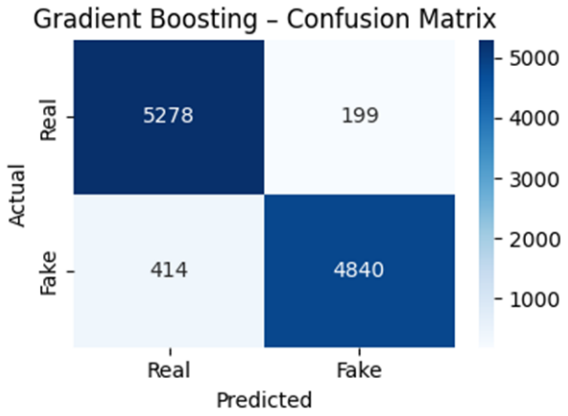

Figure 7: Confusion matrix for Gradient Boosting.

XGBoost (Fig. 8) produced the weakest result, classifying nearly all posts as fake. However, it should be noted that the training was conducted using default hyperparameters, and the model was not specifically tuned for this task.

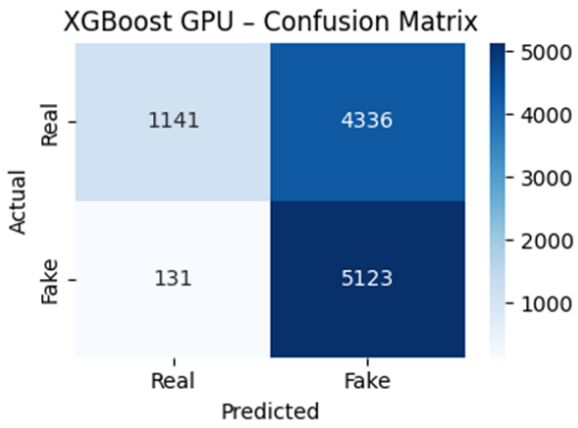

Figure 8: Confusion matrix for XGBoost.

In contrast, the confusion matrix for TinyBERT + Cross-Attention + LoRA + Bi-GRU (Fig. 9) is nearly perfect: both real and fake news are classified correctly in 99% of cases, with only a minimal number of false positives and false negatives.

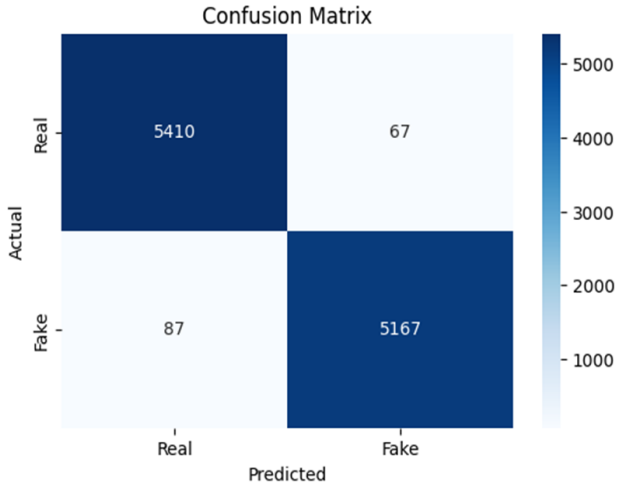

Figure 9: Confusion matrix for TinyBERT + Cross-Attention + LoRA + Bi-GRU.

Limitations and Future Directions. Despite the high classification performance, this study has several limitations that must be considered when interpreting the results and planning system deployment. First, the experimental evaluation relied exclusively on the English-language Fake News Classification corpus; therefore, the transferability of the model to texts in other languages or cultural contexts remains uncertain. Second, the classification task was limited to textual features only: multimodal data such as images, videos, or graphical metadata – elements that could provide critical cues for distinguishing fake from real – were not included. Although the TinyBERT architecture with cross-attention has strong potential for multimodal integration, combining text with images, videos, or graphical metadata could significantly improve fake news detection, albeit at the cost of higher computational demands. It should also be noted that training was conducted on a single Tesla T4 GPU, which poses challenges for scaling to larger models or datasets.

The further development of the proposed architecture envisions extending its capabilities and enhancing efficiency under real-world application scenarios. A promising direction involves experimenting with multimodal data by fusing textual information with additional sources such as images, videos, and graphical metadata. Integrating these modalities into the model (e.g., through cross-modal attention mechanisms) would improve classification accuracy and enable deeper analysis of content – particularly in cases where fake news is accompanied by manipulative visual material.

Another important direction is optimizing the model for portable devices and embedded systems. Planned research includes exploring different quantization techniques to reduce computational overhead and inference time without significant accuracy loss. Specifically, post-training INT8 quantization is considered for the entire model, while lower-precision formats such as FP8 or INT4 may be applied to computationally intensive components. Implementing these approaches with platform-specific tools – TensorFlow Lite with NNAPI for Android, Core ML for iOS, and ONNX Runtime Mobile for ARM devices – would accelerate model execution and reduce memory consumption.

5. Conclusions

The proposed compact neural network architecture – TinyBERT with Cross-Attention, LoRA, and Bi-GRU – demonstrated high effectiveness in the task of automatic fake news detection. The obtained results significantly outperformed classical machine learning models (logistic regression,

SVM, Bernoulli Naïve Bayes, k-NN, random forest, gradient boosting, and XGBoost), achieving 99% accuracy, a weighted F1-score of 0.985, and a ROC AUC of 0.998.

The main advantage of the developed model lies in its ability to explicitly detect inconsistencies between the headline and the body of a news item through the use of the cross-attention mechanism. This approach substantially improves classification accuracy, particularly in the context of widespread clickbait headlines, which traditional neural models are not always able to interpret correctly.

The application of Low-Rank Adaptation (LoRA) significantly reduced the computational resources required for fine-tuning the model while preserving performance – an essential advantage for practical deployment on devices with limited processing power.

The addition of a bidirectional recurrent Bi-GRU layer to the transformer backbone preserved sequential context and enabled effective aggregation of important semantic features, thereby improving both classification quality and interpretability of the results.

Practical benefits of the proposed model include its suitability for integration into real-world fact-checking services thanks to a low inference time (5 ms on a Tesla T4 GPU), the ability to be quickly fine-tuned for new topics and linguistic contexts, and a high level of decision interpretability via attention heatmaps and Grad-CAM.

Comparison of confusion matrices showed that the proposed model effectively balanced precision and recall, producing fewer misclassifications compared to other algorithms.

In conclusion, the TinyBERT + Cross-Attention + LoRA + Bi-GRU model is a highly effective and resource-optimized solution, well-suited for integration into real-time fake news detection systems, with considerable potential for further development and adaptation to new application scenarios.

Acknowledgements

The research was carried out with the grant support of the National Research Fund of Ukraine "Development of an Information System for Automatic Detection of Disinformation Sources and Inauthentic User Behavior in Chats", project registration number 33/0012 from 3/03/2025 (2023.04/0012).

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: grammar and spelling check. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Yoon, et al., Learning to detect incongruence in news headline and body text via a graph neural network, IEEE Access 9 (2021) 36195–36206. doi:10.1109/access.2021.3062029.

[2] X. Jiao, et al., TinyBERT: Distilling BERT for natural language understanding, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Stroudsburg, PA, USA, 2020. doi:10.18653/v1/2020.findings-emnlp.372.

[3] H. Ye, et al., LoRA-Adv: Boosting text classification in large language models through adversarial low-rank adaptations, IEEE Access (2025) 1. doi:10.1109/access.2025.3579539.

[4] Y. Taher, A. Moussaoui, F. Moussaoui, Automatic fake news detection based on deep learning, FastText and news title, Int. J. Adv. Comput. Sci. Appl. 13 (1) (2022). doi:10.14569/ijacsa.2022.0130118.

[5] R. Mishra, V. Setty, SADHAN: Hierarchical attention networks to learn latent aspect embeddings for fake news detection, in: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '19), New York, USA, 2019, pp. 197–204. doi:10.1145/3341981.3344229.

[6] K. Shu, et al., Hierarchical propagation networks for fake news detection: Investigation and exploitation, in: Proceedings of the International AAAI Conference on Web and Social Media, vol. 14, 2020, pp. 626–637. doi:10.1609/icwsm.v14i1.7329.
[7] Q. Chang, X. Li, Z. Duan, Graph global attention network with memory: A deep learning approach for fake news detection, Neural Netw. (2024) 106115. doi:10.1016/j.neunet.2024.106115.
[8] E. Qawasmeh, M. Tawalbeh, M. Abdullah, Automatic identification of fake news using deep learning, in: 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Granada, Spain, 22–25 October 2019. doi:10.1109/snams.2019.8931873.
[9] Y. Dun, et al., KAN: Knowledge-aware attention network for fake news detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35 (1), 2021, pp. 81–89. doi:10.1609/aaai.v35i1.16080.
[10] J. Alghamdi, Y. Lin, S. Luo, Enhancing hierarchical attention networks with CNN and stylistic features for fake news detection, Expert Syst. Appl. (2024) 125024. doi:10.1016/j.eswa.2024.125024.
[11] W. Zhang, et al., Cross-attention multi-perspective fusion network based fake news censorship, Neurocomputing (2024) 128695. doi:10.1016/j.neucom.2024.128695.
[12] S. Maham, et al., ANN: Adversarial news net for robust fake news classification, Sci. Rep. 14 (1) (2024). doi:10.1038/s41598-024-56567-4.
[13] D. Zhou, et al., GS2F: Multimodal fake news detection utilizing graph structure and guided semantic fusion, ACM Trans. Asian Low-Resour. Lang. Inf. Process. (2024). doi:10.1145/3708536.
[14] S. Kuntur, et al., Under the influence: A survey of large language models in fake news detection, IEEE Trans. Artif. Intell. (2024) 1–21. doi:10.1109/tai.2024.3471735.
[15] M.M. Kabir, et al., CUET-NLP_MP@DravidianLangTech 2025: A transformer and LLM-based ensemble approach for fake news detection in Dravidian, in: Proceedings of the Fifth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages, Albuquerque, New Mexico, USA, 2025, pp. 420–426. doi:10.18653/v1/2025.dravidianlangtech-1.75.
[16] V. Vysotska, O. Markiv, D. Svyshch, L. Chyrun, S. Chyrun, R. Romanchuk, Fake news identification based on NLP, big data analysis and deep learning technology, in: Proc. 2024 IEEE 17th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), IEEE, 2024, pp. 199–204.
[17] V. Vysotska, K. Przystupa, L. Chyrun, S. Vladov, Y. Ushenko, D. Uhryn, Z. Hu, Disinformation, fakes and propaganda identifying methods in online messages based on NLP and machine learning methods, Int. J. Comput. Netw. Inf. Secur. 16(5) (2024) 57–85.
[18] V. Vysotska, K. Przystupa, Y. Kulikov, S. Chyrun, Y. Ushenko, Z. Hu, D. Uhryn, Recognizing fakes, propaganda and disinformation in Ukrainian content based on NLP and machine-learning technology, Int. J. Comput. Netw. Inf. Secur. 17(1) (2025) 92–127.
[19] V. Vysotska, S. Mazepa, L. Chyrun, O. Brodyak, I. Shakleina, V. Schuchmann, NLP tool for extracting relevant information from criminal reports or fakes/propaganda content, in: Proc. 2022 IEEE 17th Int. Conf. on Computer Sciences and Information Technologies (CSIT), IEEE, 2022, pp. 93–98.
[20] V. Vysotska, L. Chyrun, S. Chyrun, I. Holets, Information technology for identifying disinformation sources and inauthentic chat users' behaviours based on machine learning, CEUR Workshop Proc. 3723 (2024) 466–483.
[21] Fake news classification, Kaggle dataset, 8 October 2023. Available at: https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification