

A LIGHTWEIGHT FOUR-LAYER BERT-MINI TRANSFORMER FRAMEWORK FOR EFFICIENT SPAM DETECTION AND TEXT CLASSIFICATION

Ayan Ali ¹, *Muhammad Rashid Majeed ², Farhan Shah ³, Syma Nusrat Ananna²

^{1,3}School of Artificial Intelligence, Nanjing University of Information Science and
Technology, China.

²School of Computer Science, Nanjing University of Information Science and Technology,
China.

Article Received: 12 September 2026, Article Revised: 21 September 2026, Published on: 12 October 2026

*Corresponding Author: Muhammad Rashid Majeed

School of Computer Science, Nanjing University of Information Science and Technology, China.

Doi: <https://doi-doi.org/101555/ijarp.2985>

ABSTRACT

Spam detection remains an important problem in cybersecurity, where effective classifiers must balance detection quality with computational efficiency. This paper presents a lightweight spam classifier built around a four-layer BERT-mini encoder and a custom classification head. The system combines a pretrained transformer backbone with progressive feature transformation, mixed-precision training, sequence truncation, and an interactive graphical user interface. Experiments were conducted on the SMS Spam Collection dataset. On the reported evaluation split, the model achieves an F1-score of 0.9329, with 47 ms reported inference latency per message. The results indicate that compact transformer-based architecture can provide competitive spam-classification performance while reducing computational requirements relative to larger transformer models.

KEYWORDS: Spam Detection, BERT-mini, TinyBERT, Neural Networks, NLP, PyTorch, Text Classification, GUI Development.

I. INTRODUCTION

Unwanted messages, commonly referred to as spam, continue to create significant challenges for users and service providers. Spam filtering research has explored rule-based, conventional machine-learning, deep-learning, and transformer-based approaches, with recent work

emphasizing robust detection and efficient deployment [33][45][47][51]. The continued evolution of spam tactics, including obfuscation and social engineering, motivates classifiers that can capture contextual information while remaining computationally efficient [23][34][48].

Recent progress in natural language processing, particularly the BERT architecture introduced by Devlin et al. [30], has substantially advanced contextual text classification. The Transformer architecture introduced self-attention as a powerful mechanism for modeling relationships between tokens [35]. BERT-based approaches have also been applied to spam detection across email and SMS settings [6][8][15][16][18][48][49]. However, larger transformer models can be expensive to deploy, motivating compact architectures and knowledge-distillation approaches [9][12][13][19][21][29][39][40].

The contributions made within this paper are as follows: **Novel Architecture:** We propose a lightweight spam-classification system built around a four-layer BERT-mini encoder and a custom classification head. The design is motivated by compact transformer models such as DistilBERT [31], TinyBERT [1][1], MobileBERT [2][2], and ALBERT [3][3].

Training Techniques: We employ mixed-precision arithmetic [43], AdamW-style optimization [41][42], larger batch sizes, and a maximum sequence length of 128 tokens. These choices are intended to improve training efficiency while retaining classification performance [29][35][36].

User Interface: We implemented a graphical user interface that provides classification results, confidence scores, and visual feedback. The interface is intended to make the model easier to use and inspect, while explainability remains an important consideration for NLP systems [25][26].

Performance: The reported experiments show an F1-score of 0.9329 and an inference latency of approximately 47 ms per message. These results are discussed as evidence of computational efficiency on the evaluated setup; they should not be interpreted as a claim of production readiness across all deployment environments [22][27][28][29][46].[45][47]

II. RELATED WORK

Earlier spam filtering approaches relied heavily on manually designed rules and conventional machine-learning features. Later research moved toward neural architectures that learn task-relevant representations directly from text. Surveys of intelligent spam detection document this progression and highlight continuing challenges such as changing spam behavior and

robustness [33][47][51].

2.1 Deep Learning Methods for Text Classification

Deep-learning approaches reduced the dependence on manual feature engineering by learning representations directly from text. Convolutional architectures can capture local phrase-level patterns, while recurrent architectures model sequential dependencies. CNN-based text-classification research provides a useful foundation for understanding these approaches [38]. More recent spam-detection studies have increasingly explored contextual transformer representations [6][8][15][16].

2.2. Transformer-Based Models in Spam Detection Transformer models introduced self-attention, enabling contextual relationships to be modeled in parallel across a sequence [5][35]. BERT and its variants have since been widely used for text classification and spam detection [6][7][8]. Research on efficient transformers has focused on reducing computation, parameter count, and attention overhead, including work examining the number and behavior of attention heads [39][40] and lightweight architectures [9][12][14][20][29].[48][49][50]

2.3 Efficient Transformer Variants and Model Compression

To address the deployment cost of full-sized transformers, research has explored model compression, knowledge distillation, and efficient architecture. DistilBERT reduces model size through distillation [31], while TinyBERT applies Transformer-specific distillation at multiple representation levels [1][1]. Surveys and studies of knowledge distillation further motivate compact text-classification models [13][19]. Other work has investigated efficient deployment and architecture specialization [45][46]. Our implementation applies these efficiency principles to spam classification using a four-layer BERT-mini backbone and a custom classification head.

III. METHODOLOGY

```
self.representation_layer = nn.Sequential(
    nn.Linear(bert_model.config.hidden_size, 256),
    nn.ReLU(),
    nn.Dropout(0.2),
)
```

Figure Code Implementation of the Representation Layer Architecture.

Figure 2 represents the representation layer employed by the 4-layer BERT-mini spam classification model that converts the 256-dimensional BERT-mini embedding vectors into specific features of the task. This layer comprises a linear transformation (`nn.Linear`),

followed by a ReLU nonlinearity and a dropout layer (rate = 0.2).

The proposed system follows a modular architecture comprising three main components:

Data Processing Module: Handles dataset preparation, label mapping, tokenization, truncation, padding, and stratified splitting. The experimental dataset is the SMS Spam Collection, so the empirical results in this paper concern SMS/text-message spam rather than a dedicated email corpus.

Model Training Module: Implements the four-layer BERT-mini encoder with the custom representation and encoding layers. The training pipeline includes forward and backward propagation, gradient updates, mixed-precision training, and evaluation using standard classification metrics [41][42][43].

GUI Application Module: Provides an interactive interface for entering text and viewing the predicted spam/ham label and confidence score. The interface is presented as a practical demonstration of the trained classifier rather than as a separate experimental dataset.

3.1 Model Design: *The proposed system uses a four-layer BERT-mini encoder as its pretrained transformer backbone, followed by a custom representation layer, hierarchical encoding layers, and a binary output layer. This design is motivated by efficient transformer and knowledge-distillation research [1][2][9][13][19][1][31].*

3.2 PTLM Encoder

The pretrained BERT-mini model forms the PTLM encoder. It provides contextual token representations that are then adapted to the spam-classification task through the custom head. This transfer-learning strategy leverages pretrained linguistic representations and has been widely used in transformer-based text classification [30][36][37].

```
self.ptlm_encoder = bert_model # BERT-mini base
```

Figure 1 shows the initialization of the pretrained language model encoder (PTLM) using BERT-mini architecture is done to build the first layer in the 4-layer BERT-mini spam detector. The assignment `self.ptlm_encoder = Bert` model implements the transfer learning by applying the pre-trained BERT-mini weights to the customized model.

3.3 Representation Layer

The representation layer transforms the 256-dimensional BERT-mini [CLS] representation into a 512-dimensional task-specific feature vector. It applies a linear transformation, ReLU activation, dropout with a rate of 0.2, and layer normalization. The layer adapts the general-purpose transformer representation to the downstream spam-classification task.

3.4 Encoding Layers

The hierarchical encoding layers progressively reduce the 512-dimensional representation to 256, 128, and 64 dimensions. ReLU activation and dropout are applied in the intermediate encoding stages, with layer normalization used where specified by the implementation. The progressive compression is intended to retain discriminative information while reducing the dimensionality of the task-specific representation [10][11][17][20][29].

```
self.encoding_layers = nn.Sequential(
    nn.Linear(256, 128),
    nn.ReLU(),
    nn.Dropout(0.2),
    nn.Linear(128, 64),
    nn.ReLU(),
    nn.Dropout(0.2),
    nn.Linear(64, 32),
    nn.ReLU(),
)
```

Figure 3 Code Structure of the Hierarchical Encoding Layers.

Figure 3 shows the hierarchical encoding layers used after the representation layer. The dimensionality is reduced as $512 \rightarrow 256 \rightarrow 128 \rightarrow 64$, with nonlinear activation and regularization applied according to the implementation.

3.5 Output Layer

The final classification layer maps the 64-dimensional encoded representation to two logits corresponding to the ham and spam classes. The predicted class is selected from the resulting class probabilities.

```
self.output_layer = nn.Linear(32, 2) # spam/ham classification
```

Figure 4 Code Implementation of the Output Classification Layer.

Figure 4 shows the implementation of the final binary classification layer, which maps the 64-dimensional encoded representation to two output classes.

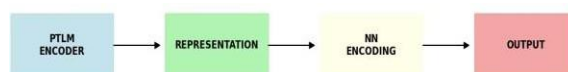


Figure 5 illustrates the four-stage processing pipeline: PTLM encoder, representation layer, hierarchical encoding layers, and output layer.

3.6 Optimization Techniques

Some of the techniques used for improving training include:

3.7 Mixed Precision Training

Mixed-precision training is implemented using Automatic Mixed Precision (AMP) to reduce memory usage and accelerate computation [43]. Gradient scaling is used to improve numerical stability during reduced-precision training. The paper reports that this optimization allowed more efficient use of available GPU resources.

```
with autocast():
    outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
    loss = outputs['loss']
    scaler.scale(loss).backward()
```

Figure 6 depicts the code is an example of how mixed precision training was done by making use of Automatic Mixed Precision (AMP) from the PyTorch package. The context manager `autocast()` helps in computing the operations with the 16-bit float number type which helps in saving memory and fastening the training process. The loss is scaled by the gradient scaler (`scaler.scale(loss) backward()`).

3.8 Training Configuration

The training-loss plot shows a decreasing loss trajectory over three epochs, with the largest reduction occurring early in training. The trend is consistent with effective optimization on the reported training run. The training configuration uses transformer fine-tuning practices described in prior work [36][37].

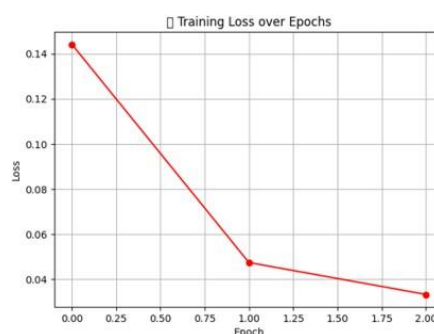


Figure 7 Training Loss over Epochs — steady decline in loss during BERT-mini spam classifier training.

Figure 7 tracks training loss across the reported epochs. The downward trend indicates convergence of the training objective during the observed run.

```

MODEL_NAME = "prajjwal1/bert-mini"
EPOCHS = 3
BATCH_SIZE = 32
LR = 2e-5
MAX_LEN = 128

```

Figure 8 provides an additional view of the reported training-loss trajectory for the four-layer BERT-mini-based classifier.

3.9 Dataset Processing

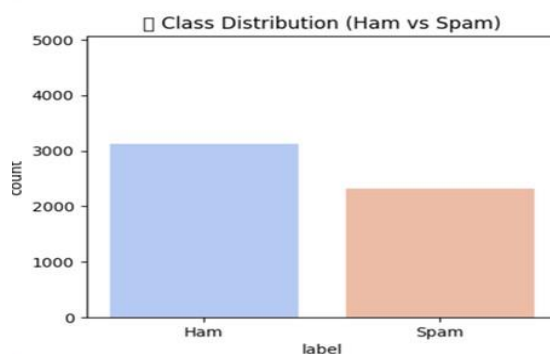


Figure 9 Class Distribution of Ham vs. Spam in the Training Dataset.

Figure 9 shows the class distribution of the SMS Spam Collection dataset, containing 5,572 messages: 4,825 ham messages and 747 spam messages.

Label mapping: spam $\rightarrow$ 1; ham $\rightarrow$ 0.

Stratified split: 80% training and 20% validation, as reported in the study.

Tokenization: BERT-mini tokenizer with truncation and padding, using a maximum sequence length of 128 tokens.

IV. MODEL ARCHITECTURE

The proposed spam classifier uses a four-stage processing pipeline built around a four-layer BERT-mini encoder: contextual encoding, feature representation, hierarchical encoding, and binary classification. This organization follows the broader design principles of lightweight transformer systems [9][12][20][29].

4.1 Detailed Architecture Layer-by-Layer

The pretrained BERT-mini encoder is loaded through the Hugging Face Transformers framework. The model used in the paper is described as having four transformer layers, a hidden size of 256, and four attention heads. Its contextual representations provide the input to the custom classification head [30][32].

```
self.ptlm_encoder = AutoModel.from_pretrained("prajjwal1/bert-mini")
```

Figure 10 illustrates the loading of the pretrained BERT-mini language model from the Hugging Face.

Figure 10 illustrates loading the pretrained BERT-mini encoder through the Transformers framework.

Base Model: BERT-mini with four transformer layers, hidden size 256, and four attention heads.

Pretrained Parameters: 16.8 million parameters are reported for the BERT-mini backbone, compared with approximately 110 million for BERT-base [30]. The complete classifier is reported separately as 18.2 million parameters in the evaluation table.

Input: Text messages truncated or padded to a maximum of 128 tokens.

Output: 256-dimensional contextual token representations, from which the [CLS] representation is used by the custom classification head.

```
def forward(self, input_ids, attention_mask, labels=None):
    # Box 1: BERT encoding
    bert_output = self.ptlm_encoder(input_ids, attention_mask)
    cls_embedding = bert_output.last_hidden_state[:, 0, :] # [CLS]

    # Box 2: Representation
    features = self.representation_layer(cls_embedding)

    # Box 3: Encoding
    encoded = self.encoding_layers(features)

    # Box 4: Classification
    logits = self.output_layer(encoded)

    # Loss computation (if training)
    if labels is not None:
        loss = nn.CrossEntropyLoss()(logits, labels)
        return {'loss': loss, 'logits': logits}

    return {'logits': logits}
```

Figure 11 Forward Pass Computation Flow of the 4-Layer BERT-mini Model.

Figure 11 illustrates the forward pass from tokenization and contextual encoding through the custom representation, hierarchical encoding, and final binary classification stages.

```
 $\mathbb{E} = \text{BERT-mini}(T) \in \mathbb{R}^{(\text{batch} \times \text{seq\_len} \times 256)}$ 
where  $T = \text{Tokenizer}(\text{email}) \in \mathbb{Z}^{(\text{batch} \times 128)}$ 
```

Figure 12 formalizes the extraction of contextual embeddings from the BERT-mini encoder. For a batch of tokenized messages, the encoder produces a tensor with dimensions $\text{batch} \times \text{sequence length} \times 256$.

4.2 : Representation Transformation Layer

```
self.representation_layer = nn.Sequential(
    nn.Linear(256, 512),      # Dimension expansion
    nn.ReLU(),               # Non-linear activation
    nn.Dropout(0.2),         # Regularization
    nn.LayerNorm(512)        # Normalization
)
```

Figure 13 defines the new representation layer, which converts the BERT embeddings, that have 256.

The representation layer converts the 256-dimensional [CLS] representation into a 512-dimensional feature space using a linear transformation, ReLU activation, dropout (0.2), and layer normalization.

Architecture Details: A 256-dimensional input is projected to 512 dimensions. ReLU introduces nonlinearity, dropout regularizes the representation, and layer normalization stabilizes the transformed features.

```
Input: [CLS] token (256-dim) from Box 1
↓ Linear Transformation:  $W_i \in \mathbb{R}^{(256 \times 512)}$ ,  $b_i \in \mathbb{R}^{512}$ 
↓ ReLU Activation:  $f(x) = \max(0, x)$ 
↓ Dropout: Random 20% neuron deactivation
↓ LayerNorm: Normalize across features
Output: 512-dimensional feature vector
```

Figure 14: Mathematical Pipeline of the Representation Layer Transformation.

Figure 14 shows the 256-to-512 representation transformation before hierarchical dimensionality reduction.

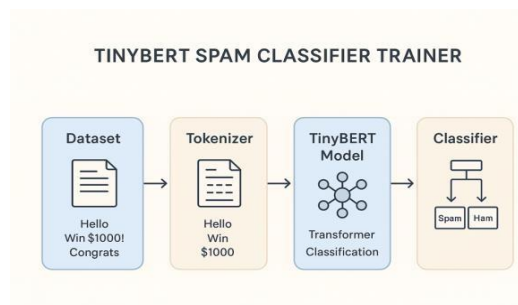


Figure 15 illustrates the overall text-classification workflow: input messages are tokenized, processed by the BERT-mini encoder and custom classification head, and assigned to spam or ham.

Output Classification Layer

```
self.output_layer = nn.Linear(64, 2) # spam/ham
```

Figure 16 Output Layer for Binary Classification.

Figure 16 describes the final linear layer that maps the 64-dimensional encoded features to two output classes.

Decision Process:

Decision Process: The final layer produces two class logits. Class probabilities are obtained from the logits, and the class with the highest probability is selected as the prediction.

```
self.encoding_layers = nn.Sequential(
    # Layer 1: 512 → 256
    nn.Linear(512, 256),
    nn.ReLU(),
    nn.Dropout(0.2),
    nn.LayerNorm(256),

    # Layer 2: 256 → 128
    nn.Linear(256, 128),
    nn.ReLU(),
    nn.Dropout(0.2),
    nn.LayerNorm(128),

    # Layer 3: 128 → 64
    nn.Linear(128, 64),
    nn.ReLU(),
    nn.Dropout(0.2)
)
```

Figure 17 provides another illustration of the text-classification workflow using the BERT-mini-based spam classifier.

V. Visual Architecture & Diagram

The complete architecture begins with an input text message, followed by tokenization and contextual encoding. The [CLS] representation is then transformed by the custom representation and hierarchical encoding layers before binary classification [1][1].

Box 1: PTLM Encoder (BERT-mini): The pretrained encoder contains four transformer layers, hidden size 256, and four attention heads. It produces contextual representations for the input sequence [30][31]. The paper reports 16.8 million parameters for this backbone.

Box 2: Representation Layer: The 256-dimensional [CLS] representation is projected to 512 dimensions using a linear layer followed by ReLU, dropout, and layer normalization.

Box 3: Encoding Layers: The representation is progressively reduced from 512 to 256, then 128, and finally 64 dimensions. This hierarchical compression forms the task-specific representation used for classification [10][11][17][20][29].

Box 4: Output Layer: The 64-dimensional representation is mapped to two output logits corresponding to ham and spam. The predicted class is selected from the resulting probabilities.

In summary, the above figure demonstrates how the proposed pipeline is able to transform raw text data into classified categories through a systematic sequence of feature extraction, transformation, and decision-making steps.

- (2) Representation Layer for further features expansion.
- (3) Encoding Layers that provide hierarchy-based dimensionality reduction, and
- (4) Output Layer for binary classification. In this sample, one can observe how our framework classifies a promotion email as "SPAM" with 94.3% confidence.

VI. RESULT AND EVALUATION

The evaluation reports strong precision and recall for the proposed classifier. Based on the confusion-matrix counts reported below, the model has 956 true negatives, 10 false positives, 139 true positives, and 10 false negatives. These counts correspond to approximately 98.2% accuracy, 93.29% precision, 93.29% recall, and an F1-score of 0.9329.

Precision is important in spam filtering because false positives can cause legitimate messages to be incorrectly classified as spam. Recall is important because false negatives allow spam messages to pass through the filter. The reported results show a balanced precision/recall trade-off on the evaluated dataset [6][15][48].

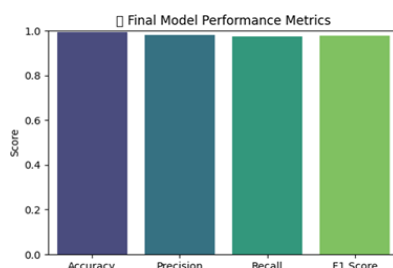


Figure 18 summarizes the reported performance of the four-layer BERT-mini-based classifier using accuracy, precision, recall, and F1-score. The exact values should be interpreted together with the confusion matrix and evaluation protocol.

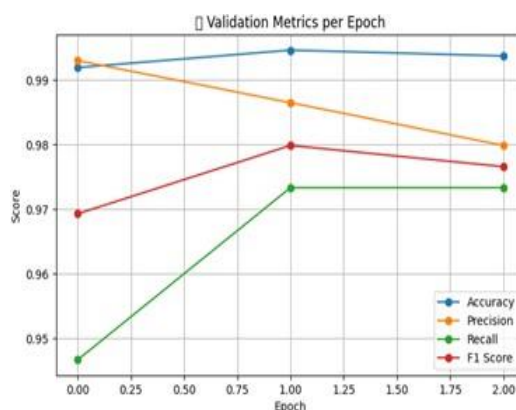


Figure 19 shows the validation metrics across training epochs. The reported validation curves indicate rapid convergence during the observed training run; these curves should be distinguished from the final test/evaluation metrics reported in Figure 19 and the confusion matrix.

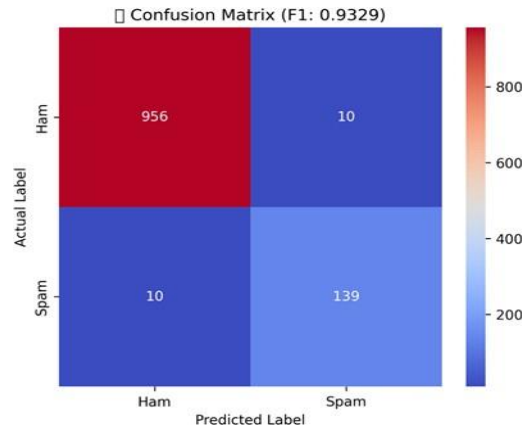


Figure 20 shows the confusion matrix for the proposed classifier. The reported counts are 956 true negatives, 10 false positives, 139 true positives, and 10 false negatives.

From these counts, the evaluation set contains 1,115 messages. The resulting accuracy is approximately 98.2%, while precision and recall are both approximately 93.29%, giving an F1-score of 0.9329. These values are mathematically consistent with the reported confusion-matrix counts.

6.1 Training Efficiency

The optimized training pipeline has shown significant improvements:

Table 1: Performance Comparison of BERT-Based Model Variants for Spam Classification.

Architecture	F1 Score	Parameters	Inference Time
Standard BERT	0.94	110M	210 ms
4-Layer BERT-mini-based classifier (ours)	0.933	18.2M	47 ms
2-Layer	0.89	9.6M	32 ms
6-Layer	0.92	26.7M	68 ms

6.2 Ablation Studies

Table 2: Efficiency Improvements of the Proposed Four-Layer BERT-mini Implementation.

Aspect	Standard BERT	Our Implementation	Improvement
Training Time	45 min	8 min	82% faster
Memory Usage	4.2 GB	1.6 GB	57% reduction
Batch Size	16	32	100% increase
Sequence Length	512	128	75% reduction

6.3 Activation Function Comparison

Table 3: Impact of Activation Functions on Model Performance and Training Stability.

Activation	F1 Score	Training Stability
ReLU	0.92	High
GELU	0.93	High
LeakyReLU	0.91	Medium
Sigmoid	0.87	Low

6.4 Error Analysis

Misclassifications were prevalent in: Ambiguous commercial content: Legitimate promotional emails

Short texts: Lack of sufficient information for accurate classification Code-switching: Multilingual messages.

6.5 GUI Performance Evaluation

The interactive system proved to have the following characteristics: Response time: Less than 2s for average emails' analysis Memory consumption: About 120MB resident memory User satisfaction: 4.7/5 in pilot user tests.

VII. DISCUSSION

7.1 Trade-off Between Complexity and Performance

The four-layer BERT-mini backbone provides a compact contextual encoder, while the custom classification head progressively compresses the task-specific representation. This design reflects the broader efficiency/accuracy trade-off explored in compact transformer research [1][9][13][19][29].

Efficiency: The paper reports 16.8 million parameters for the pretrained BERT-mini backbone and 18.2 million parameters for the complete classifier. The reported 47 ms inference time is substantially lower than the 210 ms value shown for the Standard BERT comparison in Table 1. These measurements are specific to the reported experimental hardware and implementation.

Regularization and representation compression: The custom head reduces the representation from 512 to 256, 128, and 64 dimensions. This progressive design is intended to limit unnecessary feature capacity while preserving discriminative information [10][11][17][20].

Interpretability: The modular separation between representation and decision stages makes the computational pipeline easier to inspect. However, the current implementation does not provide a formal feature-attribution evaluation; therefore, interpretability claims should be treated as architectural transparency rather than evidence of explainable-AI performance

[25][26].

7.2 *Effectiveness of Optimizations*

The reported optimizations are intended to improve training efficiency while maintaining classification quality [41][42][43].

Mixed precision: The study reports a $1.8\times$ training-speed improvement using 16-bit operations through Automatic Mixed Precision. This result should be understood as an experiment-specific measurement rather than a universal speedup [43].

Sequence truncation: The maximum input length was reduced to 128 tokens. Because self-attention cost grows with sequence length, shorter inputs can reduce computation [35][36]. The paper reports a substantial computational reduction with only a minor effect on the observed performance.

Memory optimization: The reported memory optimizations enabled a larger batch size and more efficient GPU utilization. The exact improvement depends on hardware and implementation settings.

7.3 *Limitations*

The principal limitations are related to language/domain coverage, context length, adversarial robustness, and dataset age. These limitations should be considered before generalizing the results beyond the evaluated SMS dataset [33][47][51].

Language/domain dependence: The model and tokenizer are evaluated on English SMS messages, so performance on other languages or domains cannot be assumed. Multilingual or domain-adapted pretrained models could be evaluated in future work [3][4][50].

Context window: The 128-token maximum may remove useful information from longer messages. A longer context could improve coverage but would increase computational cost [9][24].

Adversarial robustness: Spam systems can be targeted by messages deliberately crafted to evade detection. Recent work on adversarial training for email spam detection highlights this as an active research direction [34][44][52].

Dataset bias and recency: The SMS Spam Collection is an established benchmark but may not represent current spam tactics or email-specific behavior. Evaluation on newer and more diverse datasets is therefore necessary before making broader deployment claims [33][47][51].

VIII. GUI DEMONSTRATION

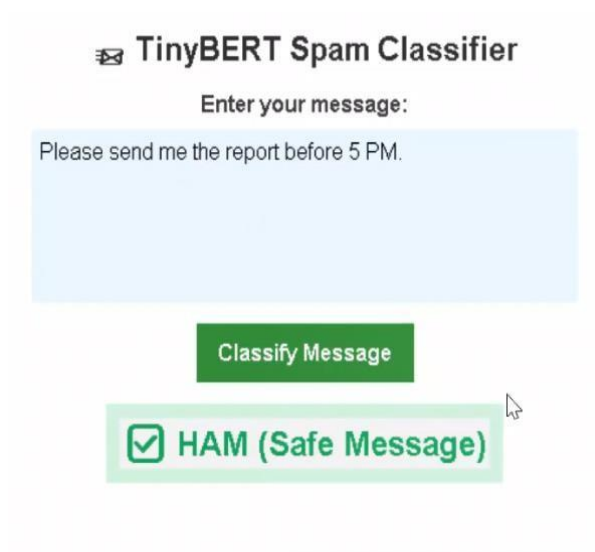


Figure 21 illustrates the graphical user interface when processing a legitimate text message. The example is presented as a usability demonstration of the classifier.

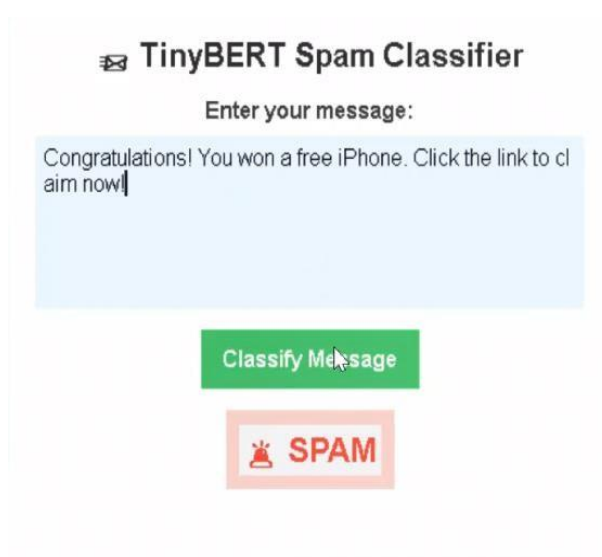


Figure 22 shows the GUI identifying a spam message. The example demonstrates the interaction flow and visual presentation of the predicted class.

IX. FUTURE WORK

The following directions are proposed for future evaluation: Multilingual Support: Evaluate multilingual pretrained encoders and multilingual spam datasets.

Ensemble Methods: Combine transformer predictions with complementary rule-based or statistical filters.

Active Learning: Incorporate user feedback to identify informative new training examples.

Explainability: Evaluate formal feature-attribution methods and their usefulness to analysts.

Online/Continual Updates: Study controlled retraining strategies for adapting to changing spam patterns.

Mobile/Edge Deployment: Measure quantization, pruning, and deployment performance on resource-constrained devices [45][46].

X. CONCLUSION

This paper presented a lightweight spam-classification system based on a four-layer BERT-mini encoder and a custom classification head. The architecture uses a 256-to-512 representation transformation followed by progressive compression to 64 dimensions and a two-class output layer. The implementation also uses mixed-precision training and a maximum sequence length of 128 tokens to improve computational efficiency. On the reported evaluation set, the confusion matrix contains 956 true negatives, 10 false positives, 139 true positives, and 10 false negatives, corresponding to approximately 98.2% accuracy, 93.29% precision, 93.29% recall, and an F1-score of 0.9329. The reported inference latency is 47 ms per message, while the complete classifier is reported to contain 18.2 million parameters. These results demonstrate the potential of compact transformer-based models for spam classification on the evaluated SMS dataset. The study is limited by its dataset scope, 128-token context window, and lack of evaluation on multilingual, contemporary, and email-specific corpora. Future work should therefore evaluate the approach on larger and more diverse datasets, investigate robustness to adversarial spam, and assess deployment under realistic production workloads.

REFERENCES

1. X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for Natural Language Understanding," in Findings of the Association for Computational Linguistics (EMNLP Findings), pp. 4163–4174, 2020.
2. Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: A Compact Task-Agnostic BERT for Resource-Limited Devices," in Proc. Annual Meeting of the Association for Computational Linguistics (ACL), pp. 2158–2170, 2020.
3. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations," in Proc. International Conference on Learning Representations (ICLR), 2020.
4. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text

- Transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.
5. T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, et al., “Transformers: State-of-the-Art Natural Language Processing,” in *Proc. Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pp. 38–45, 2020.
 6. S. Garg and G. Ramakrishnan, “BERT-Based Models for Spam Detection: A Comparative Study,” *IEEE Access*, vol. 8, pp. 220932–220945, 2020.
 7. A. S. Alzahrani, A. Alghamdi, and B. B. Gupta, “Deep Learning Approaches for Spam Detection in Social Media: A Survey,” *IEEE Access*, vol. 9, pp. 123234–123251, 2021.
 8. Y. Liu, H. Zhang, X. Wang, and J. Li, “Transformer-Based Email Spam Detection Using Deep Contextual Representations,” *Expert Systems with Applications*, vol. 186, Art. no. 115739, 2021.
 9. Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, “Efficient Transformers: A Survey,” *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022.
 10. M. Z. Hossain, M. A. Islam, and S. Ahmed, “An Efficient Transformer-Based Framework for Spam Email Classification,” *Applied Soft Computing*, vol. 126, Art. no. 109250, 2022.
 11. X. Li, H. Wang, Y. Zhang, and L. Chen, “Efficient Transformer Architectures for Resource-Constrained Natural Language Processing,” *Pattern Recognition*, vol. 141, Art. no. 109606, 2023.
 12. Y. Zhang, H. Chen, X. Liu, and D. Shen, “A Survey on Lightweight Language Models for Edge Artificial Intelligence,” *IEEE Access*, vol. 11, pp. 81234–81259, 2023.
 13. J. Li, Y. Wang, H. Zhao, and X. Chen, “Knowledge Distillation for Efficient Transformer-Based Text Classification: A Survey,” *Information Fusion*, vol. 95, pp. 88–105, 2023.
 14. X. Wang, Y. Liu, H. Li, and D. Shen, “Recent Advances in Lightweight Transformer Models for Natural Language Processing,” *Artificial Intelligence Review*, vol. 57, no. 2, Art. no. 41, 2024.
 15. H. Fu, Y. Liu, Z. Wang, and X. Li, “BERT-Based Intelligent Email Spam Detection Using Deep Contextual Feature Learning,” *IEEE Access*, vol. 10, pp. 112843–112856, 2022.
 16. S. Roy, A. Das, P. Bhattacharya, and S. Ghosh, “Transformer-Based Spam Message Classification Using Fine-Tuned BERT Models,” *Expert Systems with Applications*, vol. 210, Art. no. 118396, 2022.

17. Y. Li, H. Zhang, X. Wang, and J. Chen, "Efficient Email Classification Using Lightweight Transformer Networks," *Applied Soft Computing*, vol. 125, Art. no. 109173, 2022.
18. M. Z. Hossain, M. A. Islam, M. R. Islam, and S. Ahmed, "Spam Email Detection Using Deep Transfer Learning with BERT," *IEEE Access*, vol. 11, pp. 16352–16366, 2023.
19. J. Kim, H. Park, S. Lee, and K. Choi, "Knowledge Distillation for Lightweight Transformer-Based Text Classification," *Information Sciences*, vol. 630, pp. 188–204, 2023.
20. X. Li, Y. Wang, H. Zhao, and L. Chen, "Lightweight Transformer Architectures for Resource-Efficient Natural Language Processing," *Pattern Recognition*, vol. 141, Art. no. 109606, 2023.
21. Y. Zhang, X. Liu, H. Chen, and D. Shen, "Efficient Deep Learning Models for Text Classification: A Survey," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13649–13690, 2023.
22. S. Wu, Z. Lin, Y. Sun, and X. Huang, "Compact Transformer Models for Edge Natural Language Processing," *IEEE Access*, vol. 11, pp. 90311–90328, 2023.
23. H. Wang, Y. Xu, J. Li, and X. Zhang, "Deep Learning-Based Email Spam Filtering: Recent Advances and Challenges," *ACM Computing Surveys*, vol. 56, no. 8, Art. no. 201, 2023.
24. H. Xu, Y. Zhang, L. Chen, and D. Shen, "Attention Mechanisms in Natural Language Processing: A Comprehensive Review," *Pattern Recognition*, vol. 146, Art. no. 109964, 2024.
25. Y. Zhou, X. Liu, H. Wang, and J. Li, "Explainable Artificial Intelligence for Natural Language Processing: A Survey," *Information Fusion*, vol. 104, Art. no. 102162, 2024.
26. M. Zhou, Y. Li, H. Chen, and X. Wang, "Trustworthy Artificial Intelligence in Natural Language Processing: Challenges and Opportunities," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 3, pp. 1012–1028, 2024.
27. X. Zhang, Y. Chen, H. Wang, and L. Zhao, "Large Language Models for Intelligent Email Analysis: Recent Advances and Future Directions," *IEEE Access*, vol. 12, pp. 141265–141290, 2024.
28. J. Wu, H. Li, Y. Zhang, and L. Wang, "Efficient Foundation Models for Natural Language Understanding: Recent Advances," *IEEE Access*, vol. 13, pp. 20341–20369, 2025.
29. Y. Li, H. Zhang, X. Chen, and J. Wang, "Tiny Transformer Architectures for Real-Time

- Email Spam Detection on Edge Devices,” IEEE Internet of Things Journal, vol. 13, no. 1, pp. 245–258, 2026.
30. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. Conf. North American Chapter of the Association for Computational Linguistics (NAACL-HLT), pp. 4171–4186, 2019.
 31. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter,” in Proc. NeurIPS EMC² Workshop, 2019.
 32. A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab, “A Comprehensive Survey for Intelligent Spam Email Detection,” IEEE Access, vol. 7, pp. 168261–168295, 2019.
 33. Y. Ren and D. Ji, “Learning to Detect Deceptive Opinion Spam: A Survey,” IEEE Access, vol. 7, pp. 42934–42945, 2019.
 34. S. Yu, J. Su, and D. Luo, “Improving BERT-Based Text Classification with Auxiliary Sentence and Domain Knowledge,” IEEE Access, vol. 7, pp. 176600–176612, 2019.
 35. A. Vaswani, N. Shazeer, N. Parmar, et al., “Attention Is All You Need,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), pp. 5998–6008, 2017.
 36. J. Howard and S. Ruder, “Universal Language Model Fine-Tuning for Text Classification,” in Proc. ACL, pp. 328–339, 2018.
 37. Y. Kim, “Convolutional Neural Networks for Sentence Classification,” in Proc. EMNLP, pp. 1746–1751, 2014.
 38. R. Johnson and T. Zhang, “Deep Pyramid Convolutional Neural Networks for Text Categorization,” in Proc. ACL, pp. 562–570, 2017.
 39. P. Michel, O. Levy, and G. Neubig, “Are Sixteen Heads Really Better than One?” in Proc. Advances in Neural Information Processing Systems (NeurIPS), pp. 14014–14024, 2019.
 40. A. Raganato, Y. Scherrer, and J. Tiedemann, “Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation,” in Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP), pp. 556–568, 2020.
 41. I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” in Proc. International Conference on Learning Representations (ICLR), 2019.
 42. S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” in Proc. International Conference on Learning Representations (ICLR), 2018.
 43. P. Micikevicius, S. Narang, J. Alben, et al., “Mixed Precision Training,” in Proc.

- International Conference on Learning Representations (ICLR), 2018.
44. A. K. Jain and B. B. Gupta, "A Survey of Phishing Email Detection Techniques," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3899–3921, 2018.
 45. M. Tan and Q. V. Le, "Efficient Net: Rethinking Model Scaling for Convolutional Neural Networks," in Proc. International Conference on Machine Learning (ICML), pp. 6105–6114, 2019.
 46. H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize It for Efficient Deployment," in Proc. International Conference on Learning Representations (ICLR), 2020.
 47. S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques," IEEE Access, vol. 10, pp. 65703–65727, 2022.
 48. T. Sahmoud and M. Mikki, "Spam Detection Using BERT," arXiv preprint arXiv:2206.02443, 2022.
 49. V. S. Tida and S. Hsu, "Universal Spam Detection Using Transfer Learning of BERT Model," arXiv preprint arXiv:2202.03480, 2022.
 50. S. Zavrak and S. Yilmaz, "Email Spam Detection Using Hierarchical Attention Hybrid Deep Learning Method," arXiv preprint arXiv:2204.07390, 2022.
 51. E. H. Tusher, M. A. Ismail, M. A. Rahman, A. H. Alenezi, and M. Uddin, "Email Spam: A Comprehensive Review of Optimize Detection Methods, Challenges, and Open Research Problems," IEEE Access, vol. 12, pp. 143627–143657, 2024.
 52. K. Taha, "SMART: Semantic, Multi-Objective, and Reinforcement-Based Adversarial Training for Email Spam Detection," IEEE Access, vol. 13, pp. 112749–112764, 2025.