2025 2nd International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM) | 979-8-3315-5542-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/NCIM65934.2025.11160230

TYMBERT: Tiny Yet Mighty– Fine-Tuned and Compressed TinyBERT for High-Accuracy, Low-Resource NLP

Md. Soyeb*, Mukit Mahdin*, Syed Samin Sadaf*, Mirza Mohammad Azwad*, Ajwad Abrar
Department of Computer Science and Engineering, Islamic University of Technology, Gazipur, Bangladesh
Email: {mdsoyeb, mukitmahdin20, saminsadaf, mirzaazwad, ajwadabrar}@iut-dhaka.edu
*These authors contributed equally

***Abstract*—Deploying large-scale NLP models on edge devices remains a challenge due to their high computational and storage requirements. This study aims to optimize TinyBERT by using knowledge distillation and quantization techniques to enhance its efficiency while maintaining high accuracy. We distill knowledge from BERT-base-uncased into TinyBERT and apply quantization (fp16) to reduce model size. Our proposed model, TYMBERT, achieves a 15x size reduction from the base BERT model while preserving high accuracy, with an AUC score of 0.999535 - outperforming MobileBERT by one-third its size and matching the performance of the original tinyBERT with half its size. Additionally, TYMBERT enables real-time NLP applications on low-power devices, particularly for SMS spam detection in resource-constrained environments. This research contributes to the development of efficient NLP models, suggesting future explorations in more aggressive quantization and pruning techniques to further optimize performance.**

***Index Terms*—TinyBERT, Knowledge Distillation, Quantization, Edge Computing, SMS Spam Detection**

I. Introduction

The last few years have seen an increasing need to deploy Natural Language Processing (NLP) models on edge computing devices, e.g., smartphones, IoT devices, and embedded systems, with the rising demand for real-time language processing with limited computational means. However, it is usually infeasible to run state-of-the-art transformer models such as Bidirectional Encoder Representations from Transformers (BERT) [1] on these devices because of their reasons for model size and computational requirements. The increasing demand for efficient NLP models in such environments calls for techniques that reduce the models' complexity while maintaining high performance.

Traditional transformer-based models like BERT and their variations have achieved state-of-the-art results in some of the challenging NLP tasks like sentiment analysis, named entity recognition, and question answering [2]–[4]. However, their deployments on edge devices are out of the question since they are too large and computationally expensive. For instance, the BERT-base has around 110 million parameters—an order of magnitude above what many of the edge devices allow for in terms of storage and processing power [1]. Hence, researchers are beginning to look into model compression techniques such as Knowledge Distillation [5] and Quantization [6] to enable such models to be more easily deployed on these critically resource-constrained devices.

In response to this challenge, we focus on optimizing TinyBERT [7], which is a smaller variant of BERT, using Knowledge Distillation and Quantization techniques. We propose ***Tiny Yet Mighty BERT (TYMBERT)***, which is designed to achieve a balance between model size and accuracy with its reduced number of parameters. Knowledge Distillation enables us to transfer knowledge from a large "teacher" model, like BERT-base, to a smaller "student" model, such as TinyBERT. This process effectively optimizes the performance of smaller models while maintaining their efficiency [5]. Additionally, applying Quantization, specifically reducing the precision of model weights to fp16, further reduces the model size without sacrificing the performance [6].

This paper aims to optimize TinyBERT via two techniques: knowledge distillation and quantization. The entire idea is to distill knowledge from bert-base-uncased into TinyBERT using soft tokens while applying Quantization to get the model in the order of fp16. Our contributions consist of:

- We introduce Tiny Yet Mighty BERT (TYMBERT), an optimized TinyBERT variant designed for edge computing, leveraging knowledge distillation and fp16 quantization to achieve a 15x size reduction while preserving high accuracy. To facilitate code reproducibility, the code has been made publicly available.[1]
- We demonstrate that further distillation beyond TinyBERT enhances performance, with TYMBERT achieving an AUC of 0.999535, outperforming MobileBERT while being one-third its size.
- We enable real-time NLP on low-power devices by delivering a 27.4 MB model that balances efficiency, accuracy, and deployability, making it ideal for SMS spam detection and resource-constrained applications.

[1]https://github.com/mirzaazwad/TYMBert/tree/master

II. Related Work

Researchers have proposed numerous models for SMS Spam Detection over the years. Earlier studies used traditional machine learning models, which required hand-crafted features. Duan et al. proposed a two-step filtering process for SMS spam classification [8]. At first, a rough set-based filtering was carried out. Then, the K-Nearest-Neighbor classifier was applied, which showed great improvement in recognizing spam messages. Gupta et al. used TF-IDF for word vectorization and achieved 95% accuracy in the test set [9]. Besides, Suleiman et al. used a Python-based h2o framework to evaluate the performance of the Random Forest, Deep Learning, and Naive Bayes Classifier on the UCI SMS Spam Classification dataset [10]. They concluded that the number of digits and the presence of URLs can be critical in the detection of spam. In evaluation, they showed Random Forest achieved the highest accuracy (97.7%), but Nayes Bayes showed superiority in runtime (0.6 seconds).

Unlike machine learning models, neural networks can extract complex features from input data, which are then used for prediction or result computation. Taheri et al. employed the Recurrent Neural Network(RNN), a variation of neural network that performed better than Support Vector Machines(SVM) and Bayesian Models on the UCI SMS Spam Classification dataset [11]. Roy et al. integrated CNN with LSTM for spam classification with a very high accuracy of 99.14% on the same dataset [12]. But the UCI SMS Spam Classification dataset [13] exhibited class imbalance where only 747 out of 5574 samples are spam. Ghourabi et al. utilized the CNN-LSTM model with a multilingual dataset in which limited Arabic samples are inserted into the UCI SMS Spam dataset [14].

As the transformer architecture [15] is proposed, many variants are developed both for particular tasks as well as for more generalized purposes. Encoder-based models such as BERT [1] achieved excellent results in classification tasks. But Liu et al. trained a variant of the transformer model known as SPAM Transformer, substituting the output sequence in the decoder with trainable parameters named memory and a single neuron after the feed-forward network to carry binary classification on UCI and UtkMI dataset [16]. Though the model achieved the highest accuracy, recall, and F-1 score, it had lower precision than Random Forest. Uddin et al. demonstrated that RoBERTa achieved state-of-the-art accuracy(99.84%) in the UCI dataset, surpassing SVM, XGBoost, and other deep learning models [17]. Authors employed back translation to address the class imbalance issue.

Despite the advancements in spam detection through machine learning and deep learning algorithms, Salman et al. discovered that traditional machine learning approaches and AI-based detectors continue to be vulnerable to adversarial perturbations [18]. Advanced spamming techniques can impact the performance of modern SMS Spam Detection models. The authors also presented a novel "Super Dataset" of 67,018 labeled SMS messages, comprising 40,837 benign messages (60.9%) and 26,181 spam messages (39.1%). Furthermore, Salman et al. presented SpaLLM-Guard, in comparing commercial and open-source LLMs in SMS spam detection on this dataset, demonstrated that fine-tuned models like Mixtral(8x7B) achieved 98.61% accuracy [19]. However, model size remains an issue for mobile applications.

III. Methodology

A. Dataset

Our work involved analyzing two datasets: the SMS Spam Collection Dataset with 5,572 instances (4,825 ham and 747 spam, approx. 17:3 ratio) [20], and the Super SMS Dataset with 67,010 instances (40,832 ham and 26,178 spam, approx. 3:2 ratio) [18].

TABLE I
Comparison of SMS Spam Datasets

Dataset	**Instances**	**Ham (%)**	**Spam (%)**
SMS Spam Collection	5,572	86.6%	13.4%
Super SMS Dataset	67,010	60.9%	39.1%

Upon comparing the two datasets, we decided to proceed with the Super SMS Dataset due to its greater number of instances and its well-balanced data.

B. Data Pre-processing

The dataset was prepared in a systematic way for training and evaluation. The raw texts were transformed into tokenized formats for model ingestion via the TinyBERT tokenizer. The maximum sequence length was set at 128 tokens, ensuring that input sequences were appropriately padded or truncated. Batch processing was utilized to tokenize the complete dataset for enhanced computing efficiency.

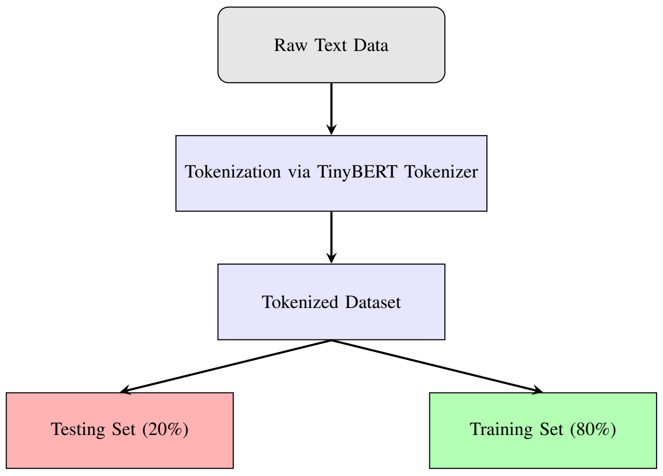

Fig. 1. Data Pre-processing Pipeline

Following tokenization, the dataset was split into two subsets: training (80%) and testing (20%). This ensured the model was trained on a variety of samples while preserving a significant proportion for evaluation purposes.

C. Knowledge Distillation

Our goal was to develop a high-accuracy model with minimal resource usage. While larger models offer better accuracy, they also increase in size, whereas smaller models reduce size at the cost of accuracy. To balance this trade-off, we applied knowledge distillation, a compression technique where a compact student model learns from a larger teacher model's soft outputs, preserving generalization [5]. We implemented a single Teacher-Student framework for efficient knowledge transfer.

*1) **Training Teacher Model**:* To facilitate knowledge distillation, we initially trained a high-capacity instructor model based on the BERT architecture, bert-base-uncased [1]. We used the Hugging Face Transformers architecture for training, employing the Trainer API for effective optimization. The training procedure was set with the following hyperparameters:

- **Batch size:** 16 (for both training and evaluation)
- **Number of epochs:** 3
- **Weight decay:** 0.01 (to mitigate overfitting)
- **Floating point precision (FP16):** set to true
- **Evaluation strategy:** Performed at the end of each epoch

We used `DataCollatorWithPadding`[2] to ensure consistent padding and formatting of sequences across batches, automatically padding input sequences to the length of the longest sequence in each batch.

*2) **Soft Label Generation**:* For knowledge distillation, we used the pre-trained bert-base-uncased model with a tokenizer set for binary classification. Training texts were validated as strings for proper tokenization, and data was processed in batches of 16 with padding and truncation (max length 128). The model generated logits, converted via softmax into class probabilities (soft labels), which were then stored for student model training.

*3) **Training Student Model**:* We executed knowledge distillation by training a smaller model TinyBERT (4L-312D) [7], leveraging soft labels supplied by the bert-base-uncased model as the instructor. The training process incorporated both Kullback-Leibler (KL) divergence for soft label alignment and cross-entropy loss for ground-truth supervision.

The distillation loss function was formulated as follows:

$$L_{\text{total}} = \alpha \cdot L_{\text{CE}} + (1 - \alpha) \cdot L_{\text{KL}} \tag{1}$$

where:

- L_{CE} is the cross-entropy loss with true labels.
- L_{KL} is the KL divergence[3] loss between the soft predictions of the teacher and student.
- α controls the balance between soft and hard labels.

Teacher and student models used the same hyperparameters.

D. Quantization

After training the student model, we performed quantization to shrink the model size and enhance inference performance. Quantization involved reducing model weights to a lower precision (fp16) than fp32.

[2]https://huggingface.co/docs/transformers/main_classes/data_collator#transformers.DataCollatorWithPadding

[3]https://hanj.cs.illinois.edu/cs412/bk3/KL-divergence.pdf

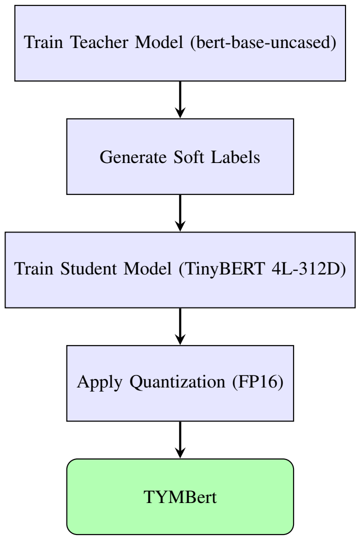
Train Teacher Model (bert-base-uncased)
Generate Soft Labels
Train Student Model (TinyBERT 4L-312D)
Apply Quantization (FP16)
TYMBert

Fig. 2. Overview of Knowledge Distillation and Quantization Process for TYMBERT

IV. Experimental Setup

A. Training Configuration

Training Framework and Hyperparameters: We utilized the Trainer API for model training, carefully selecting hyperparameters to balance performance and computational efficiency. The batch size determined the number of samples per training step, while the number of epochs controlled the total iterations over the dataset. Weight decay (0.01) was applied as a regularization technique to mitigate overfitting.

Computational Efficiency and Optimization: To enhance computational efficiency, floating-point precision (fp16) was employed, accelerating training while reducing memory consumption. Temperature scaling (set to 2) was used to smooth output probabilities for knowledge distillation, improving knowledge transfer. The learning rate was tuned to 0.5 to optimize convergence while maintaining stability.

Final Training Configuration: Through experimentation, we found that training for three epochs with a batch size of 16 and a weight decay of 0.01 yielded optimal results while adhering to hardware constraints. This configuration ensured a balance between model performance and resource efficiency.

B. Evaluation Metrics

In our study, we evaluated the performance of our experiments using the following evaluation metrics [21]. Here, TP denotes True Positives, TN True Negatives, FP False Positives, and FN False Negatives. Additionally, TPR represents the True Positive Rate, and FPR the False Positive Rate.:

1) **Accuracy:** Accuracy gives the measure of the properly classified samples relative to the total sample count.

TABLE II
PERFORMANCE METRICS OF BERT, ROBERTA, AND DISTILBERT MODELS

Class	BERT-base			RoBERTa			DistilBERT		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
0	0.993991	0.994723	0.994357	0.993025	0.995828	0.994424	0.994727	0.995460	0.995093
1	0.991805	0.990672	0.991238	0.993499	0.989149	0.991319	0.992948	0.991814	0.992381
Macro avg	0.992898	0.992698	0.992798	0.993262	0.992488	0.992872	0.993838	0.993637	0.993737
Weighted avg	0.993134	0.993135	0.993135	0.993211	0.993210	0.993207	0.994030	0.994031	0.994030
Accuracy	0.993135			0.993210			0.994031		
AUC Score	0.997453			0.999101			0.999333		
Model Size (MB)	417.65			475.49			255.41		

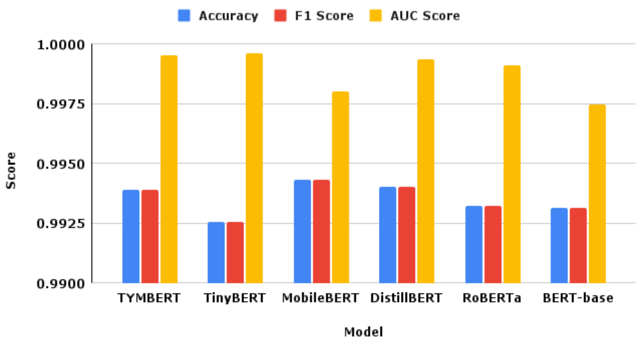

Fig. 3. Performance Comparison of the Models

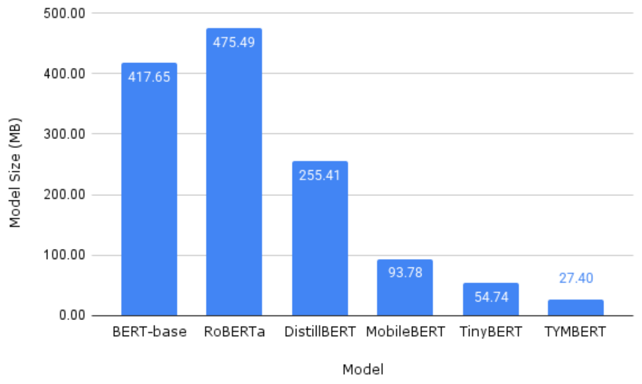

Fig. 4. Size Comparison of the Models

While this is straightforward, this may be misleading when it comes to imbalanced datasets. [21]

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2) **Precision:** Precision, which is also known as Positive Predictive Value, shows the number of true positive predictions relative to all the predictions. This metric is crucial for scenarios where minimizing the number of false positives is important. [21]

$$\text{Precision} = \frac{TP}{TP + FP}$$

3) **Recall:** Recall, also known as Sensitivity or True Positive Rate, measures the true positive predictions which are properly or correctly identified by the model. This becomes crucial when the the minimizing the number of false negatives become important [21].

$$\text{Recall} = \frac{TP}{TP + FN}$$

4) **F1-Score:** The F1-score is the harmonic mean of precision and recall, useful for evaluating models on imbalanced datasets [21].

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

5) **Receiver Operating Characteristics Curve (ROC Curve):** The ROC curve plots True Positive Rate against False Positive Rate at different thresholds, showing a model's ability to separate classes. The TPR and FPR are defined as:

$$\text{TPR} = \frac{TP}{TP + FN}$$

$$\text{FPR} = \frac{FP}{FP + TN}$$

A model that achieves a higher TPR while keeping a lower FPR is considered to be more effective. [21]

6) **Area Under the Curve Score (AUC Score):** The AUC score is a single value representing the area under the ROC curve, indicating how well a model distinguishes between positive and negative classes. A score near 1 implies excellent performance, while around 0.5 suggests random guessing [21].

$$\text{AUC} = \int_0^1 \text{TPR}\, d(\text{FPR})$$

7) **Model Size:** Consider the model's parameters W_i, where $\epsilon(W_i)$ denotes the total number of elements in a parameter and $\sigma(W_i)$ represents the memory occupied by each element in bytes, the model size is calculated as follows:

$$\text{Model Size (MB)} = \frac{\sum_{i=1}^{N} (\epsilon(W_i) \times \sigma(W_i))}{1024^2}$$

N stands for the total number of parameters in the model. The metric is important for evaluating the storage and deployment feasibility of the model.

TABLE III
PERFORMANCE METRICS OF TINYBERT, MOBILEBERT, AND TYMBERT MODELS

Class	TinyBERT			MobileBERT			TYMBERT		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
0	0.993986	0.993742	0.993864	0.994609	0.996073	0.995340	0.994726	0.995214	0.994970
1	0.990295	0.990672	0.990483	0.993894	0.991624	0.992758	0.992570	0.991814	0.992192
Macro avg	0.992140	0.992207	0.992173	0.994251	0.993848	0.994049	0.993648	0.993514	0.993581
Weighted avg	0.992539	0.992538	0.992539	0.994329	0.994329	0.994328	0.993881	0.993882	0.993881
Accuracy	0.992538			0.994329			0.993882		
AUC Score	0.999625			0.997992			0.999535		
Model Size (MB)	54.74			93.78			**27.40**		

V. RESULT ANALYSIS AND DISCUSSION

A. *Model Performance and Efficiency Trade-offs*

To evaluate the trade-offs between model performance and efficiency, we conducted a comparative analysis using Tables II and III. TinyBERT emerged as the most lightweight model within the BERT family, with an initial size of 54 MB. Given its compact nature, we explored optimization techniques to further enhance its efficiency without compromising performance.

In our experiments, we fine-tuned TinyBERT and applied precision reduction techniques. By enabling the fp16 flag during training and subsequently converting the model's precision to float16, we significantly reduced its size to 27.4 MB. This compression improved storage efficiency while maintaining computational performance, making the model more suitable for deployment in resource-constrained environments. Although MobileBERT achieved slightly higher accuracy (approximately 0.0008), it is more than three times larger than our proposed TYMBERT model, highlighting the efficiency of TYMBERT.

B. *Impact of Knowledge Distillation and Quantization*

Knowledge distillation, where a large model (bert-base-uncased) transfers knowledge to a smaller model (TinyBERT), proved highly effective in maintaining performance while reducing model complexity [5]. As shown in Figure 3, our distilled TinyBERT model achieved an overall accuracy of 0.993882 and an F1-score of 0.993881, outperforming the original TinyBERT, which had an accuracy of 0.992538 and an F1-score of 0.992539. This indicates that additional distillation can further enhance TinyBERT's effectiveness.

Although TinyBERT itself is a product of knowledge distillation [7], we extended this process specifically for the ”Super Dataset” of SMS spam classification [18], leading to improved classification accuracy. However, model quantization—reducing parameter precision to half—introduced a minor performance drop, a known trade-off in deep learning compression. Notably, the dataset was well-balanced, eliminating the need for additional techniques such as loss translation, weighted sampling, or cross-validation during training.

C. *Comparison with Teacher Model and MobileBERT*

Our distilled TinyBERT model (TYMBERT) performed comparably to its teacher model, bert-base-uncased, demonstrating that significant size reduction does not necessarily lead to a loss in accuracy. However, MobileBERT achieved the highest overall accuracy and F1-score, reaching nearly 0.995 in both cases. Despite its superior accuracy, MobileBERT requires 93.78 MB of storage, making it less suitable for deployment on resource-constrained devices.

While TinyBERT remains the smallest model in the BERT family at 54 MB, our distilled and quantized version (TYMBERT) achieved higher accuracy while reducing the model size by half. This balance of efficiency and performance makes it a strong candidate for real-world applications, particularly on low-power devices.

D. *Trade-offs in Quantization and Model Size*

The distillation process from bert-base-uncased to TinyBERT resulted in notable performance improvements. Despite reducing parameter precision from fp32 to fp16, our model performed exceptionally well on the “Super Dataset” [18]. This result is particularly noteworthy, as lower precision models typically suffer from performance degradation. One possible reason for this resilience is the nature of SMS samples, which are relatively short and do not require capturing long-range dependencies.

Our proposed model is approximately 15 times smaller than bert-base-uncased and one-third the size of MobileBERT. Additionally, TYMBERT achieved a higher AUC score (0.999535) compared to MobileBERT (0.997992), indicating greater confidence in classification. This suggests that our model maintains strong predictive capabilities even with quantization while being significantly more compact.

E. *Applicability in Edge Computing and Low-Resource Environments*

TYMBERT is designed for real-world deployment in low-resource and edge-computing environments. Devices such as smartphones, IoT devices, and embedded systems often have limited computational power and storage capacity. Deploying large-scale transformer models like BERT or RoBERTa on these devices is impractical due to their memory and processing requirements.

Through knowledge distillation and quantization, we reduced TinyBERT's size to 27.4 MB while maintaining competitive accuracy (Table III). This enables real-time inference on resource-constrained devices with minimal performance loss, making TYMBERT well-suited for applications such as SMS spam detection and other NLP tasks requiring efficient, low-power models.

F. Implications for Real-World Deployment

Compared with DistilBERT, MobileBERT, and other transformer-based models, TYMBERT offers significant advantages in terms of size, efficiency, and accuracy. DistilBERT, despite being a lighter alternative to BERT, still has 66 million parameters and requires 255 MB of storage. MobileBERT, though optimized for mobile devices, remains relatively large at 93.78 MB.

In contrast, TYMBERT achieves a balance of model size (27.4 MB), computational efficiency, and accuracy, with an AUC score of 0.9995. This makes it a compelling choice for real-world SMS classification and other NLP tasks on low-power devices. Unlike larger models that require substantial memory and processing resources, TYMBERT is well-suited for battery-operated devices with limited storage, offering a practical solution for deploying transformer-based models in constrained environments.

VI. Conclusion

This study optimized TinyBERT for edge devices using Knowledge Distillation and Quantization, balancing model size and performance. A teacher-student framework distilled knowledge from BERT to TinyBERT with soft-label training. Quantizing to fp16 halved the model size while preserving accuracy and F1-score, making the proposed TYMBERT ideal for real-time NLP on low-power devices. It outperformed existing compact transformers in efficiency.

While performance loss was minimal, future work will explore more aggressive quantization (e.g., 4-bit, 8-bit), pruning, and expansion to multilingual datasets like Bangla. Exploring lightweight alternatives beyond BERT may further boost efficiency for edge computing in low-resource settings.

References

[1] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] S. Alaparthi and M. Mishra, "Bert: A sentiment analysis odyssey," *Journal of Marketing Analytics*, vol. 9, no. 2, pp. 118–126, 2021.

[3] J. Arora and Y. Park, "Split-ner: Named entity recognition via two question-answering-based classifications," *arXiv preprint arXiv:2310.19942*, 2023.

[4] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage bert: A globally normalized bert model for open-domain question answering," *arXiv preprint arXiv:1908.08167*, 2019.

[5] G. Hinton, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[6] B. Jacob, S. Kligys, B. Chen, *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[7] X. Jiao, Y. Yin, L. Shang, *et al.*, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[8] L. Duan, A. Li, and L. Huang, "A new spam short message classification," in *2009 First International Workshop on Education Technology and Computer Science*, IEEE, vol. 2, 2009, pp. 168–171.

[9] S. D. Gupta, S. Saha, and S. K. Das, "Sms spam detection using machine learning," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1797, 2021, p. 012 017.

[10] D. Suleiman and G. Al-Naymat, "Sms spam detection using h2o framework," *Procedia computer science*, vol. 113, pp. 154–161, 2017.

[11] R. Taheri and R. Javidan, "Spam filtering in sms using recurrent neural networks," in *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, IEEE, 2017, pp. 331–336.

[12] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter sms spam," *Future Generation Computer Systems*, vol. 102, pp. 524–533, 2020.

[13] A. Frank, "Uci machine learning repository," *http://archive. ics. uci. edu/ml*, 2010.

[14] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, "A hybrid cnn-lstm model for sms spam detection in arabic and english messages," *Future Internet*, vol. 12, no. 9, p. 156, 2020.

[15] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[16] X. Liu, H. Lu, and A. Nayak, "A spam transformer model for sms spam detection," *IEEE Access*, vol. 9, pp. 80 253–80 263, 2021.

[17] M. A. Uddin, M. N. Islam, L. Maglaras, H. Janicke, and I. H. Sarker, "Explainabledetector: Exploring transformer-based language modeling approach for sms spam detection with explainability analysis," *arXiv preprint arXiv:2405.08026*, 2024.

[18] M. Salman, M. Ikram, and M. A. Kaafar, "Investigating evasive techniques in sms spam filtering: A comparative analysis of machine learning models," *IEEE Access*, 2024.

[19] M. Salman, M. Ikram, N. Basta, and M. A. Kaafar, "Spallm-guard: Pairing sms spam detection using open-source and commercial llms," *arXiv preprint arXiv:2501.04985*, 2025.

[20] T. Almeida and J. Hidalgo, "Sms spam collection," *UCI Machine Learning Repository*, 2011, Available: https://doi.org/10.24432/C5CC84.

[21] O. Rainio, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, 2024, Available: https://doi.org/10.1038/s41598-024-56706-x.