TinyLite: Edge-Based Spam Detection via Lightweight Transformer Compression

Xuanlin Zhu, Johora Akter Polin, Sidi Lu
College of William & Mary
Williamsburg, VA, USA
{xzhu09, japolin, slu02}@wm.edu

***Abstract*—Spam messaging inflicts billions of dollars in economic damage annually through phishing, malware distribution, and fraud. While Transformer-based classifiers such as BERT achieve state-of-the-art accuracy, their computational requirements (100M+ parameters, billions of FLOPs) render them impractical for edge deployment on smartphones and IoT devices. Traditional methods like Naïve Bayes and Decision Trees are lightweight but vulnerable to adversarial obfuscation and lack semantic understanding. This paper bridges the gap between high-accuracy Transformers and resource-efficient edge deployment. We systematically benchmark six lightweight BERT variants against classical baselines on SMS, email, and Twitter spam datasets, demonstrating that even minimal Transformers outperform traditional methods by over 8 percentage points in F1 score. We reveal severe overparameterization: models achieve near-peak accuracy with only 10–20% of training data. Motivated by these findings, we develop TinyLite, a heavily compressed BERT-Tiny derivative that applies head pruning and hidden dimension reduction to achieve 81.6% FLOP reduction while losing less than 1.1 percentage points in F1. We validate TinyLite's robustness across continual learning, federated learning, cross-domain transfer (email and Twitter), and INT8 quantization scenarios, confirming its versatility for diverse edge deployments.**

***Index Terms*—spam detection, edge computing, transformer compression, lightweight BERT, on-device NLP**

I. Introduction

The Spam Threat. Spam messaging, encompassing unwanted bulk communications sent across SMS, email, and social media platforms, accounts for billions of messages transmitted daily and inflicts significant economic and security costs on individuals, businesses, and society [1]–[3]. Financial fraud via phishing links, unauthorized promotional offers, and malware distribution campaigns all leverage spam as their primary attack vector. Beyond direct financial impact, spam campaigns frequently engage in data harvesting and sophisticated phishing attacks that can lead to identity theft. These mounting threats underscore the urgent need for effective, real-time spam detection systems that protect users at the point where messages are received.

Traditional Detection Methods. Traditional spam filters deployed at the network or application level typically rely on bag-of-words models [1], [4]. The most common approaches are Multinomial Naive Bayes classifiers and Decision Trees, operating on term frequency-inverse document frequency representations or n-gram features. While computationally lightweight and capable of processing messages quickly with minimal hardware resources, these methods suffer from inherent brittleness. They ignore semantic context by treating each word as completely independent, and they fail to capture long-range dependencies in message text.

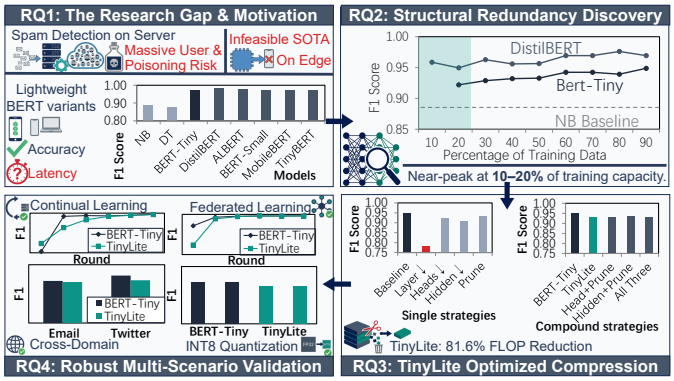


Fig. 1. Overview of TinyLite for edge-based spam detection. The framework illustrates the research pipeline: identifying the gap between heavyweight Transformer models and edge deployment requirements, discovering redundant capacity in lightweight variants, developing TinyLite through systematic compression, and validating robustness across diverse deployment scenarios.

Most critically, traditional approaches can be undermined by simple adversarial tactics called Bayesian poisoning [5]. Spammers deliberately insert misspellings, substitute special characters for letters, or employ other obfuscation techniques to evade keyword-based filters. Current mitigations often involve frequent manual rule updates, expert-curated blacklists, or costly retraining on freshly labeled data. These practices do not scale efficiently and raise privacy concerns when user messages must be aggregated on central servers for processing.

Transformer-Based Detection. In contrast, Transformer-based text classifiers such as BERT and its variants deliver state-of-the-art results on natural language processing tasks by modeling deep bidirectional context via self-attention mechanisms [6], [7]. These models learn rich representations that capture how each word relates to every other word in a passage. However, the original BERT models contain over 100 million parameters and require billions of floating-point operations per inference pass, making them impractical for deployment on resource-constrained edge devices such as smartphones, tablets, and IoT modules [3], [8], [9].

Challenges and Motivation. This paper bridges the gap

between high-accuracy Transformer models and resource-efficient on-device deployment. As illustrated in Figure 1, we develop TinyLite, a heavily compressed Transformer model specifically designed for edge-based spam detection. We benchmark TinyLite against both traditional methods and modern deep learning approaches across three distinct datasets covering SMS text messages, email, and Twitter posts. We also validate the robustness of our approach across realistic deployment scenarios including continual learning, federated learning, cross-domain transfer, and model quantization.

Physical Intelligence Perspective. Our work aligns with the emerging paradigm of physical intelligence, which considers how intelligent systems must operate within the physical constraints of their deployment environments [3], [10]–[12]. Edge devices such as smartphones and IoT sensors impose strict limitations on memory, computation, thermal dissipation, and battery consumption [13], [14]. Deploying spam detection directly on these devices offers significant advantages: reduced latency for real-time protection, enhanced privacy by keeping message content local, and independence from network connectivity. TinyLite embodies three principles of physically intelligent design: hardware-software co-design tailored to edge device constraints, resource adaptability for low-power and low-data environments, and system-level efficiency addressing thermal and battery limitations [15], [16]. The detailed analysis of how TinyLite achieves these goals is presented in Sections IV and V.

Research Questions. Our research methodology proceeds through four stages, each addressing a specific research question: i) **RQ1**: How much does a Transformer-based edge paradigm improve spam classification performance over traditional machine learning methods? ii) **RQ2**: To what extent can lightweight BERT models achieve high classification accuracy when fine-tuned on only a fraction of the available training set? iii) **RQ3**: What efficiency gains in model size and computational requirements does our proposed TinyLite model deliver relative to baseline architectures? iv) **RQ4**: How stable and generalizable is TinyLite when deployed across multiple spam domains and under various on-device learning scenarios?

Contributions. We develop TinyLite, a heavily compressed Transformer model purpose-built for edge-based spam detection, and systematically validate its effectiveness. Our specific contributions are:

- **C1**: We present a comprehensive study of edge-based spam detection that systematically evaluates lightweight Transformer models, identifies compression opportunities, develops an optimized compression strategy, and validates robustness across realistic deployment scenarios (Sections III and IV).
- **C2**: For RQ1, we confirm that lightweight BERT variants substantially outperform classical Naive Bayes and Decision Tree approaches, demonstrating superior robustness of contextual embeddings against lexical obfuscation tactics (Section IV-A).
- **C3**: For RQ2, we reveal that lightweight models reach near-peak accuracy with a small fraction of training data, demonstrating significant overparameterization and motivating aggressive compression (Section IV-B).
- **C4**: For RQ3, we develop TinyLite through systematic ablation studies, achieving substantial FLOP reduction while maintaining performance comparable to uncompressed baselines (Section IV-C).
- **C5**: For RQ4, we validate TinyLite in continual learning, federated learning, cross-domain transfer, and quantization scenarios, demonstrating versatility for diverse edge deployment settings (Section IV-D).

The rest of this paper is organized as follows. Section II provides background on spam detection methods and Transformer architectures. Section III details our methodology including hardware platform, datasets, models, and experimental protocols. Section IV presents empirical findings organized by research question. Section V discusses key observations explaining why Transformer models succeed where traditional methods fail. Section VI reviews related work. Section VII concludes with summary and future directions.

II. Background

This section provides background on traditional spam detection methods, the Transformer architecture, and lightweight BERT variants that make edge deployment feasible.

A. Traditional Spam Detection and Its Vulnerabilities

Early spam detection focused on classical machine learning applied to engineered text features. The most widely adopted approach uses Multinomial Naive Bayes classifiers operating on TF-IDF feature vectors [4]. These classifiers became ubiquitous due to simplicity, interpretability, and extremely low computational cost, training in seconds and classifying in microseconds. Decision Trees offer interpretability advantages, expressing classification logic as if-then rules that operators can inspect [1]. Support Vector Machines have also shown competitive performance [17].

Despite practical utility, these bag-of-words methods are inherently brittle. They treat each word as completely independent, ignoring semantic context. The word "free" in "free as in freedom" receives identical weight as "free" in "free money." They cannot capture long-range dependencies: when a message begins with innocent content but ends with spam, all words are weighted equally. These limitations make traditional classifiers vulnerable to adversarial attacks. Bayesian poisoning [5] inserts deliberate misspellings or character substitutions ("drvgsTo" instead of "drugs to") that classifiers treat as unknown vocabulary items. Good-word attacks [18] inject legitimate words to dilute spam indicators, tipping classification toward legitimate.

Countermeasures include heuristic rules matching obfuscation patterns, blacklists, and frequent retraining on new data. These impose ongoing maintenance burdens without addressing the fundamental lack of contextual understanding. Centralized retraining also raises privacy concerns when user messages must be collected and labeled.

B. Transformer Architecture and BERT

The Transformer architecture revolutionized NLP through self-attention mechanisms that allow every position in a sequence to attend to every other position [6], [7]. Unlike recurrent networks that process text sequentially, Transformers process all positions in parallel while capturing long-range dependencies through learned attention weights.

BERT (Bidirectional Encoder Representations from Transformers) applies the Transformer encoder to learn deep bidirectional representations through masked language modeling [6], [19], [20]. During pretraining on massive corpora (billions of words from Wikipedia and books), random tokens are masked and the model predicts them from bidirectional context. This produces representations capturing rich linguistic knowledge transferable to downstream tasks [21], [22]. After pretraining, BERT is fine-tuned on specific tasks by adding a classification layer.

A key component enabling robustness against obfuscation is WordPiece tokenization [23]–[25]. Unlike traditional tokenizers producing whole-word tokens, WordPiece breaks words into subword units based on frequency statistics. Common words remain intact, but rare or obfuscated words split into pieces overlapping with known vocabulary. When encountering obfuscated tokens, the tokenizer produces subwords that share representations with related words, providing inherent robustness against obfuscation attacks.

C. Lightweight BERT Variants

Full BERT's computational demands (110M parameters, billions of FLOPs per inference) motivated extensive compression research [26], [27]. **DistilBERT** [28] uses knowledge distillation [29] to train a smaller student network mimicking a larger teacher, achieving 40% size reduction while retaining 97% of BERT-base performance with 6 layers and 66M parameters. **ALBERT** [30] introduces cross-layer parameter sharing and factorized embeddings, achieving strong performance with only 12M unique parameters across 12 layers. **BERT-Small** and **BERT-Tiny** [31] systematically reduce depth and width: BERT-Small uses 4 layers with 512 hidden units (29M parameters); BERT-Tiny uses 2 layers with 312 hidden units (14M parameters). **MobileBERT** [32] introduces inverted-bottleneck modules optimized for mobile CPUs, delivering near-BERT accuracy with 4x faster inference. **TinyBERT** [33] employs two-stage distillation producing a 4-layer model with 14.5M parameters matching DistilBERT performance.

These variants exemplify major compression strategies: distillation (DistilBERT, TinyBERT), weight sharing (ALBERT), depth/width reduction (BERT-Small/Tiny), and architectural redesign (MobileBERT) [34], [35]. Pruning approaches remove redundant parameters, with research showing many attention heads are redundant [36], [37]. Quantization converts weights to lower precision for faster inference [38]–[40]. Our work builds on this foundation by identifying which strategies are most effective for spam detection and demonstrating that more aggressive compression is possible given task simplicity.

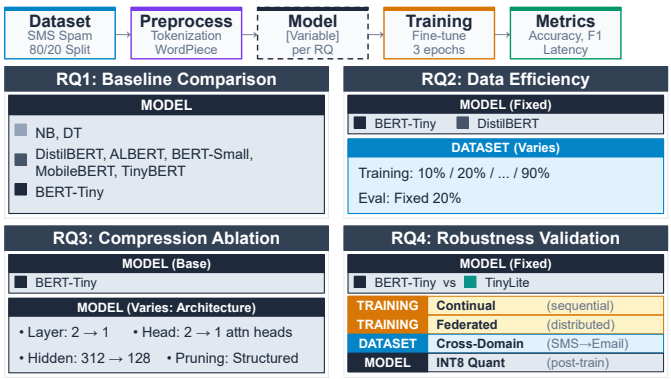


Fig. 2. Overview of the experimental pipeline from data input through model evaluation across four research questions.

III. Methodology

This section describes experimental methodology, covering the hardware configuration, datasets spanning different spam domains, the classification models under evaluation, metrics for assessing effectiveness and efficiency, and the experimental protocols designed to address each research question.

A. Hardware and Software Configuration

All experiments were conducted on a mid-range laptop representative of contemporary consumer devices that might serve as edge deployment targets [8], [10]. Table I summarizes the hardware specifications. We deliberately chose this configuration to demonstrate that effective Transformer-based spam detection is achievable without enterprise-grade hardware, as many edge deployment scenarios offer comparable or superior computational resources.

TABLE I. Hardware and Software Configuration

Component	Type	Specification
CPU	Model	Intel Core (10 cores, 16 threads)
	Clock	2.4 GHz base, 4.8 GHz turbo
	Memory	32 GB DDR4 @ 3200 MHz
GPU	Model	NVIDIA RTX 4060 Laptop
	VRAM	8 GB GDDR6
	CUDA Cores	3072 @ 115W TDP
Storage	SSD	1 TB NVMe (3500 MB/s)

We report results under two configurations: CPU-only inference, which simulates deployment on devices lacking dedicated accelerators, and GPU-accelerated inference, which represents deployment on devices with discrete or integrated graphics capable of running CUDA workloads.

B. Dataset Collection and Preprocessing

We conduct experiments on three publicly available spam classification datasets spanning the major domains where spam detection systems are deployed: SMS text messages, email communications, and social media posts [41]–[43].

UCI SMS Spam Collection [41] serves as our primary benchmark, containing 5,572 English-language messages (4,825 legitimate, 747 spam) collected from multiple sources. Messages average approximately 80 characters with vocabulary spanning over 8,700 unique tokens including abbreviations, emoticons, and deliberate misspellings characteristic of informal SMS communication. **Kaggle Email Spam Dataset** [42] contains 5,171 email messages (3,672 legitimate, 1,499 spam) with substantially longer messages averaging 1,048 characters and vocabulary exceeding 50,000 tokens. **Kaggle Twitter Spam Dataset** [43] contains 5,572 tweets with Twitter-specific conventions including hashtags, mentions, and URLs. For all datasets, we performed stratified 80/20 train/validation splits preserving class proportions.

C. Models Under Evaluation

We evaluate nine classification models that are organized into three categories.

1) Classical Baselines: To establish performance floors, we include two traditional bag-of-words classifiers [4]: (1) Multinomial Naive Bayes operating on TF-IDF feature vectors with vocabulary limited to 5,000 terms, and (2) Decision Tree classifier using the same TF-IDF features, implemented using scikit-learn with default hyperparameters.

2) Lightweight BERT Variants: We evaluate six lightweight BERT variants with publicly available pretrained checkpoints [26]: DistilBERT [28] (6 layers, 768 hidden, 66M parameters), ALBERT [30] (12 layers with parameter sharing, 12M parameters), BERT-Small [31] (4 layers, 512 hidden, 29M parameters), BERT-Tiny [31] (2 layers, 312 hidden, 14M parameters), MobileBERT [32] (24 layers with bottleneck, 25M parameters), and TinyBERT [33] (4 layers, 312 hidden, 14.5M parameters).

3) TinyLite (Proposed): Our proposed TinyLite model builds on BERT-Tiny and applies two compression strategies: reducing attention heads from 2 to 1 per layer, and reducing hidden dimension from 312 to 128 units. The resulting model contains approximately 4.32 million parameters while maintaining strong classification performance.

All BERT variants were fine-tuned from pretrained checkpoints using the Hugging Face Transformers library [20] with a classification head consisting of dropout (p=0.1) followed by linear projection to two output logits.

D. Evaluation Metrics

We assess model performance using effectiveness and efficiency metrics that together characterize the accuracy-efficiency tradeoff central to edge deployment [8], [14].

(i) **Accuracy:** Accuracy measures the proportion of messages correctly classified, computed as $(TP + TN)/N$ where TP, TN, and N denote true positives, true negatives, and total samples respectively.

(ii) **F1 Score:** F1 Score computes the harmonic mean of precision and recall: $F1 = 2 \cdot \frac{P \cdot R}{P+R}$, where precision $P = TP/(TP + FP)$ and recall $R = TP/(TP + FN)$. This metric ensures both precision and recall must be high to achieve a high score, making it preferred for imbalanced binary classification.

(iii) **Inference Latency:** Latency measures per-sample prediction time in milliseconds: $L = \text{Total Time}/\text{Samples}$. Prior research establishes delays under 100ms are perceived as instantaneous [44], 100 to 300ms are noticeable but tolerable [45], and delays exceeding 300ms interrupt cognitive flow.

(iv) **FLOPs:** Floating-point operations measure computational complexity, estimated for Transformers as FLOPs $\approx 2 \times L \times H^2 \times S$ where L is layers, H is hidden dimension, and S is sequence length (128 tokens in our experiments).

E. Experimental Protocols

We designed four experimental protocols corresponding to our research questions [46], [47].

RQ1 Protocol (Transformer vs. Baselines). All eight models trained on full SMS training set and evaluated on validation set. Classical baselines use TF-IDF with 5,000 terms; BERT variants fine-tuned for 3 epochs with batch size 16 (CPU) or 32 (GPU), AdamW optimizer with weight decay 0.01, learning rate 5×10^{-5} with linear warmup.

RQ2 Protocol (Data Efficiency). DistilBERT and BERT-Tiny fine-tuned on training fractions of 10%, 20%, through 90% using stratified sampling. F1 plotted against training fraction to identify diminishing returns.

RQ3 Protocol (Compression Ablation). Systematic ablation from BERT-Tiny evaluating: (1) layer reduction (2 to 1), (2) head pruning (2 to 1 heads), (3) hidden reduction (312 to 128), (4) structured 20% weight pruning [34], [37]. Compound strategies evaluated for viable single-factor approaches.

RQ4 Protocol (Robustness Validation). TinyLite and BERT-Tiny evaluated under: (1) continual learning with 5 sequential batches of 500 examples [46], (2) federated learning with 5 clients using Dirichlet non-IID partitions [47], (3) cross-domain transfer to email and Twitter without fine-tuning, (4) INT8 dynamic quantization using PyTorch utilities [39], [40].

IV. Results

This section presents empirical findings organized by research question, with tables and figures illustrating key performance comparisons.

A. RQ1: Transformer versus Traditional Baselines

Table II presents accuracy, F1 scores, and inference latency for all models on the UCI SMS Spam Dataset. The classical Naive Bayes baseline achieves 97.22% accuracy and 0.8848 F1, while Decision Tree achieves 96.86% accuracy and 0.8763 F1. All six BERT variants dramatically outperform these baselines, with DistilBERT achieving the highest performance (99.46% accuracy, 0.9799 F1) and the smallest model BERT-Tiny achieving 99.28% accuracy and 0.9730 F1, surpassing Naive Bayes by over 8 percentage points in F1 [6], [28].

On GPU, all models except ALBERT achieve latency under 100ms, meeting the instantaneous threshold from HCI research [44], [45]. On CPU, only BERT-Tiny (30.7ms) and TinyBERT (45.6ms) achieve sub-100ms latency. ALBERT's

TABLE II. RQ1: Model Performance and Inference Latency on UCI SMS Spam Dataset

Model	Accuracy	F1 Score	CPU (ms)	GPU (ms)
DistilBERT	0.9946	0.9799	78.9	4.2
ALBERT	0.9892	0.9586	1613.7	112.3
BERT-Small	0.9919	0.9697	51.2	3.1
BERT-Tiny	0.9928	0.9730	30.7	2.1
MobileBERT	0.9937	0.9764	89.4	5.8
TinyBERT	0.9910	0.9667	45.6	2.8
Naive Bayes	0.9722	0.8848	0.3	–
Decision Tree	0.9686	0.8763	0.2	–

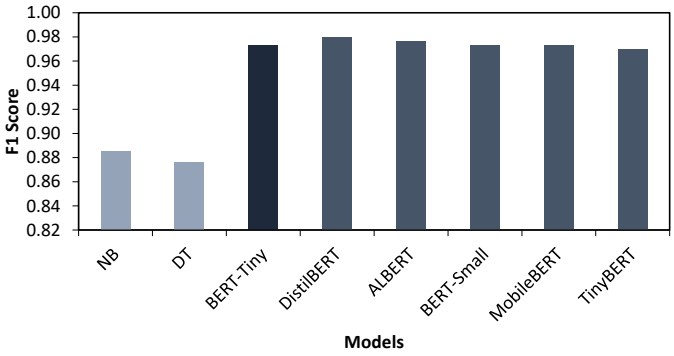


Fig. 3. RQ1 results showing F1 score comparison of traditional baselines (Naive Bayes and Decision Tree) versus six lightweight BERT variants on the UCI SMS Spam Collection. All Transformer models substantially outperform the classical methods.

exceptionally high CPU latency, despite having only 12 million parameters, results from its parameter sharing architecture: while memory-efficient, the repeated application of shared weights across 12 layers requires significant computation [30].

RQ1 Summary: All BERT variants substantially outperform traditional classifiers with F1 improvements exceeding 8 percentage points, confirming contextual embeddings provide far more robust spam detection [48], [49]. However, latency constraints limit viable CPU options to the smallest models.

B. RQ2: Data Efficiency

Figure 4 plots F1 score against training data fraction for DistilBERT and BERT-Tiny. Both models achieve near-peak classification accuracy with only 10 to 20 percent of available training data [27], [35]. Beyond this threshold, additional data provides marginal improvement as the learning curves flatten dramatically. This indicates severe overparameterization: models designed for complex GLUE benchmark tasks [21] have far more capacity than needed for binary spam classification on short messages with limited vocabulary.

RQ2 Summary: Near-peak performance with 10 to 20 percent training data reveals that most model capacity is arguably wasted, providing motivation for aggressive compression.

C. RQ3: Compression Ablation

Figure 5 presents ablation results [34], [36]. Layer reduction (2 to 1) causes catastrophic degradation (F1: 0.949 to 0.781, loss of 16.8 percentage points), demonstrating that both layers

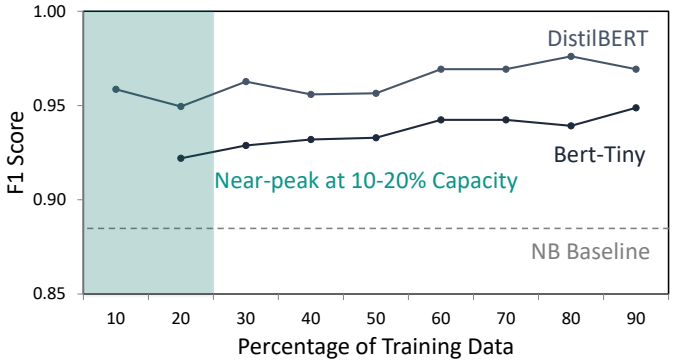
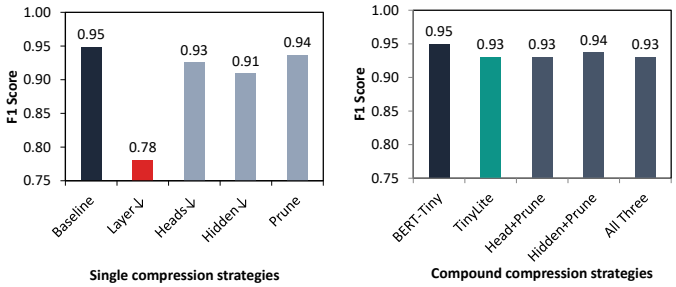


Fig. 4. RQ2 data efficiency results. Both DistilBERT and BERT-Tiny achieve near-peak F1 (greater than 0.97) with only 10 to 20 percent of training data, revealing overparameterization relative to task complexity.



(a) Single strategies (b) Compound strategies

Fig. 5. RQ3 ablation results. (a) Single compression strategies: layer reduction causes catastrophic F1 drop; head pruning, hidden reduction, and structured pruning maintain F1 above 0.90. (b) Compound strategies: TinyLite (Head plus Hidden) achieves optimal balance.

perform essential functions [50]. In contrast, head pruning (2 to 1 heads) reduces F1 by only 2.4 percentage points, hidden reduction (312 to 128) by 4.0 percentage points, and structured pruning by 1.3 percentage points [37]. Our TinyLite configuration combines head pruning with hidden reduction, achieving F1 of 0.930 (loss of 1.09 percentage points) with 81.6 percent FLOP reduction.

TABLE III. TinyLite versus BERT-Tiny Efficiency Comparison

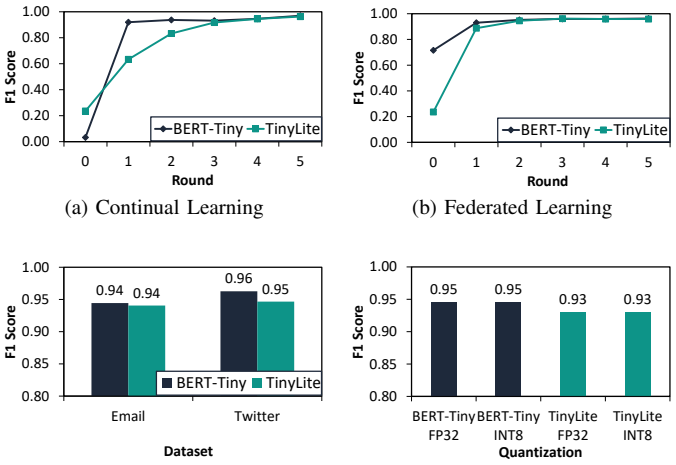
Model	F1 Score	GFLOPs
BERT-Tiny	0.949	0.639
TinyLite	0.930	0.117
Reduction	1.09pp	81.6%

RQ3 Summary: Layer depth is critical while width dimensions can be reduced [51]. TinyLite achieves 81.6 percent FLOP reduction with only 1.09 percentage points F1 loss.

D. RQ4: Robustness Validation

Figure 6 shows TinyLite's robustness across four deployment scenarios [46], [47].

In **continual learning** (Figure 6a), TinyLite catches up to BERT-Tiny by round 4, reaching 0.9630 vs 0.9697 F1



(a) Continual Learning (b) Federated Learning

(c) Cross-Domain Transfer (d) INT8 Quantization

Fig. 6. RQ4 robustness validation across four deployment scenarios.

by round 5. In **federated learning** (Figure 6b) with non-IID partitions, TinyLite catches up by round 2 and remains within 0.41 percentage points through round 5. In **cross-domain transfer** (Figure 6c), TinyLite achieves 0.9406 F1 on email (0.38 percentage points gap) and 0.9467 on Twitter (1.60 percentage points gap) without retraining. Under **INT8 quantization** (Figure 6d), both models maintain identical F1 scores with zero performance loss [39], [40].

RQ4 Summary: TinyLite demonstrates robust performance across all scenarios, tracking BERT-Tiny within 0.5 to 1.6 percentage points F1, confirming versatility for diverse edge deployment settings [14].

V. Key Observations

Beyond quantitative results, our experiments reveal important insights about why Transformer models outperform traditional methods for spam detection and how compression affects model behavior [6], [26]. These observations have implications for designing future edge-deployed classifiers.

⋆ **Observation$_1$:** *Subword tokenization defeats Bayesian poisoning attacks by decomposing obfuscated tokens into meaningful pieces that overlap with known vocabulary.*

Discussion: Spammers employ Bayesian poisoning by inserting typos, character substitutions, or obfuscation to evade keyword-based filters [5], [18]. Traditional classifiers treat obfuscated tokens as unknown vocabulary items with no relationship to their intended meaning. Consider the example: *"Low-cost prescripiton drvgsTo listen to email call 123."* Traditional TF-IDF with Naive Bayes predicts this as legitimate because "prescripiton" and "drvgsTo" are unrecognized. BERT variants succeed because WordPiece tokenization [24], [25] breaks "drvgsTo" into subword pieces ("dr," "vgs," "To") that partially overlap with known roots like "drugs." During fine-tuning, the model learns that such subword sequences in contexts discussing purchases signal spam intent.

⋆ **Observation$_2$:** *Self-attention captures long-range dependencies, enabling detection of spam indicators regardless of their position in the message.*

Discussion: Sophisticated spam often begins with innocent conversational content before revealing commercial intent [1]. Consider: *"Hi babe... missing me? SP visionsms.com Text stop to stop 150p/text."* Naive Bayes misclassifies this because benign opening tokens ("Hi," "babe," "missing") outnumber spam indicators at the end. BERT succeeds because self-attention is position-agnostic, meaning every token's representation incorporates information from all other tokens [7]. The model learns to assign high attention weights to diagnostic tokens like domain names and pricing information regardless of position, ensuring spam signals propagate through all layers.

⋆ **Observation$_3$:** *Contextual embeddings prevent false positives by distinguishing word meanings based on context.*

Discussion: Traditional filters trigger on words frequent in spam, causing false positives in legitimate contexts [4]. Consider: *"I liked the new mobile."* Naive Bayes flags this as spam because "mobile" appears frequently in spam offering phone deals. BERT correctly classifies it as legitimate because contextual embeddings reflect word meaning in context [19], [52]. The words "liked" and "new" provide cues that this is a personal opinion, not a commercial solicitation. The classification head learns to distinguish these contextual usages, associating spam with "mobile" co-occurring with commercial language ("free," "deal") versus legitimate with personal language ("liked," "my").

⋆ **Observation$_4$:** *Layer depth is critical for classification accuracy while attention width and hidden dimensionality can be aggressively reduced.*

Discussion: Our ablation studies reveal striking asymmetry: reducing layer depth causes catastrophic loss (16.8pp), while reducing attention heads or hidden dimensions causes only modest degradation (2.4 to 4.0pp) [36], [37]. The two Transformer layers perform qualitatively different functions. The first learns local token-level patterns while the second aggregates these into global sequence-level representations [50]. Neither can be eliminated. In contrast, multiple attention heads provide redundant parallel pathways; a single head suffices for binary classification. Similarly, 128 hidden dimensions encode sufficient features for spam detection. This insight directly informed TinyLite's design: maintain full depth while aggressively compressing width.

⋆ **Observation$_5$:** *Task simplicity enables aggressive compression. SMS spam detection requires far less model capacity than general NLP tasks.*

Discussion: The severe overparameterization in RQ2 (peak accuracy with 10 to 20 percent data) and successful 81.6 percent FLOP reduction in RQ3 reflect fundamental mismatch between model capacity and task complexity [27], [35]. BERT was designed for GLUE benchmark tasks requiring subtle linguistic understanding such as inference, similarity, and coreference [21]. SMS spam detection is comparatively simple: limited vocabulary (under 9,000 tokens), short mes-

sages (80 characters average), and often obvious features (URLs, phone numbers, promotional language). The unused capacity corresponds to dimensions essential for complex NLP but irrelevant for spam detection [51]. This suggests edge deployment should be approached through a task-specific lens, where compression level depends on task complexity [14].

VI. Related Work

This section reviews prior research relevant to our work, organized into four areas: traditional spam detection, BERT compression, applications of BERT to spam detection, and edge machine learning systems.

(1) Traditional Spam Detection. Classical machine learning approaches to spam detection have been extensively studied [1], [4]. The dominant paradigm uses bag-of-words feature representations where each message is converted to a sparse TF-IDF vector. Multinomial Naive Bayes classifiers became the standard due to simplicity, interpretability, and efficiency [4]. Decision Trees provide interpretable alternatives [1], while Support Vector Machines show competitive performance on benchmarks [17]. However, these methods share fundamental vulnerabilities to adversarial attacks. Bayesian poisoning inserts obfuscated tokens that classifiers cannot recognize [5]; good-word attacks inject legitimate words to dilute spam indicators [18]. Defenses impose ongoing maintenance costs without addressing the fundamental brittleness of keyword-based methods.

(2) BERT Compression. The computational demands of BERT have motivated extensive compression research [26]. Knowledge distillation trains smaller models to mimic larger ones [29]; DistilBERT [28] achieves 40% size reduction while retaining 97% performance, and TinyBERT [33] extends this with two-stage distillation. Parameter sharing approaches like ALBERT [30] reduce memory footprint by reusing weights across layers. Architectural redesign produces MobileBERT [32] with inverted-bottleneck modules optimized for mobile inference. Direct scaling yields BERT-Small and BERT-Tiny [31] with reduced depth and width. Pruning removes redundant parameters, with research showing many attention heads are redundant [36], [37], [50]. The lottery ticket hypothesis has been applied to BERT [35], and progressive module replacement offers another compression path [51]. Quantization converts weights to lower precision [38]–[40]. Hardware-aware optimization like MnasNet [11] and HAQ [53] jointly considers architecture and hardware constraints.

(3) BERT for Spam Detection. Several recent studies apply BERT to spam detection. Tida and Hsu [48] fine-tune full BERT models on email corpora achieving >95% accuracy, but assume GPU-accelerated server inference without considering edge constraints. Sahmoud and Mikki [49] adapt BERT to SMS spam, noting computational challenges but deferring edge deployment to future work. Studies on opinion spam and social media spam confirm benefits of contextual embeddings [4], [17] but focus on accuracy rather than deployment efficiency. Our work differs by systematically benchmarking lightweight variants, demonstrating overparameterization, developing optimized compression, and validating across edge deployment scenarios.

(4) Edge Machine Learning. Research on edge ML has produced frameworks and techniques for deploying neural networks on resource-constrained devices [3], [12], [13]. The emergence of edge computing has motivated extensive work on efficient inference [8], [10]. Mobile-optimized inference frameworks (TensorFlow Lite, PyTorch Mobile, ONNX Runtime) provide tools for model conversion and quantization. Neural architecture search produces models optimized for edge hardware [11]. TinyML has enabled deep learning on microcontrollers [9], [15], while TinyTL reduces memory requirements for on-device learning [16]. Vehicle edge computing demonstrates collaborative learning on connected devices [14], [54], [55]. Applications span keyword spotting [56], wake word detection, and smart sensor processing. Human-computer interaction research establishes critical latency thresholds: delays under 100ms are perceived as instantaneous [44], 100 to 300ms are tolerable [45], and greater than 300ms interrupt cognitive flow. Federated learning enables privacy-preserving distributed training [47], while continual learning addresses catastrophic forgetting when models adapt to new data [46]. Our work contributes by demonstrating that Transformer-based NLP, previously considered expensive for edge deployment, can be made practical through task-specific compression.

VII. Conclusion

This paper addressed deploying high-accuracy spam detection on resource-constrained edge devices. We systematically investigated the gap between traditional methods that are efficient but brittle, and Transformer methods that are accurate but computationally demanding, developing TinyLite as a practical solution combining benefits of both paradigms.

Our evaluation confirmed that lightweight BERT variants outperform traditional classifiers by over 8 percentage points in F1 score, demonstrating the value of contextual embeddings for spam detection. We revealed severe overparameterization, with models achieving peak accuracy using only 10 to 20 percent of training data. We developed TinyLite achieving 81.6% FLOP reduction with less than 1.1pp accuracy loss. Robustness validation across continual learning, federated learning, cross-domain transfer, and quantization confirms TinyLite's versatility for diverse edge deployments.

These findings establish that Transformer-based spam detection is feasible on edge devices, enabling privacy-preserving on-device inference while maintaining accuracy benefits. Future work will extend TinyLite to multilingual and multimodal spam detection.

Acknowledgment

The authors thank Professor Sidi Lu (College of William & Mary) for her guidance and supervision throughout this research, and Johora Akter Polin for her mentorship and feedback during the development of this work. This paper is being prepared for submission to the Physical Intelligence:

Systems and Applications (PISA) Workshop. All remaining errors and interpretations are the authors' own.

References

[1] M. Tusher *et al.*, "A survey on spam detection techniques," *Journal of Network and Computer Applications*, vol. 180, p. 103025, 2024.

[2] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 291–302, 2018.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] A. Mewada *et al.*, "Machine learning approaches for spam detection: A comprehensive review," *Expert Systems with Applications*, vol. 195, p. 116573, 2022.

[5] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012, pp. 1467–1474.

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

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[8] Y.-H. Chen *et al.*, "Benchmarking edge computing devices for machine learning applications," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4862–4874, 2019.

[9] J. Lin *et al.*, "Deep learning inference on microcontrollers," *Proceedings of NeurIPS*, 2020.

[10] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[11] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

[12] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[14] S. Lu and W. Shi, "Vehicle computing: Vision and challenges," *Journal of Information and Intelligence*, vol. 1, no. 1, pp. 23–35, 2023.

[15] P. Warden and D. Situnayake, "TinyML: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers," *O'Reilly Media*, 2019.

[16] H. Cai *et al.*, "TinyTL: Reduce memory, not parameters for efficient on-device learning," *Proceedings of NeurIPS*, 2020.

[17] M. Yurtseven *et al.*, "Deep learning based spam detection," *Neural Computing and Applications*, vol. 33, pp. 11 567–11 580, 2021.

[18] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2005.

[19] Y. Liu *et al.*, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[20] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," *Proceedings of EMNLP: System Demonstrations*, pp. 38–45, 2020.

[21] A. Wang *et al.*, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," *Proceedings of EMNLP Workshop BlackboxNLP*, pp. 353–355, 2018.

[22] C. Raffel *et al.*, "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.

[23] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[24] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *Proceedings of EMNLP: System Demonstrations*, pp. 66–71, 2018.

[25] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *Proceedings of ACL*, pp. 1715–1725, 2016.

[26] P. Ganesh *et al.*, "Compressing large-scale transformer-based models: A case study on BERT," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1061–1080, 2021.

[27] J. Li *et al.*, "Train large, then compress: Rethinking model size for efficient training and inference of transformers," *Proceedings of ICML*, 2020.

[28] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[29] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[30] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[31] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.

[32] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: a compact task-agnostic BERT for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.

[33] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[34] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[35] T. Chen *et al.*, "The lottery ticket hypothesis for pre-trained BERT networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[36] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[37] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *Proceedings of ACL*, pp. 5797–5808, 2019.

[38] S. Zhou *et al.*, "Incremental network quantization: Towards lossless CNNs with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[39] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[40] O. Zafrir *et al.*, "Q8BERT: Quantized 8bit BERT," *Proceedings of the Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

[41] T. Almeida and J. Hidalgo, "UCI SMS spam collection data set," https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection, 2012.

[42] Kaggle, "Email spam classification dataset," https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv, 2020.

[43] ——, "Twitter spam detection dataset," https://www.kaggle.com/datasets/, 2021.

[44] R. B. Miller, "Response time in man-computer conversational transactions," *Proceedings of the AFIPS Fall Joint Computer Conference*, vol. 33, pp. 267–277, 1968.

[45] S. C. Seow, "Designing and engineering time: The psychology of time perception in software," *Addison-Wesley Professional*, 2008.

[46] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[47] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of AISTATS*, pp. 1273–1282, 2017.

[48] V. Tida and S. Hsu, "BERT-based email spam detection," *IEEE Access*, vol. 10, pp. 26 582–26 592, 2022.

[49] T. Sahmoud and S. Mikki, "Spam detection using BERT and deep learning," *Journal of Big Data*, vol. 9, pp. 1–15, 2022.

[50] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *Proceedings of ICLR*, 2020.

[51] C. Xu *et al.*, "BERT-of-theseus: Compressing BERT by progressive module replacing," *Proceedings of EMNLP*, pp. 7859–7869, 2020.

[52] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," *Proceedings of EMNLP*, pp. 6894–6910, 2021.

[53] K. Wang *et al.*, "HAQ: Hardware-aware automated quantization with mixed precision," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

[54] S. Lu *et al.*, "Vehicle edge computing for enabling autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3688–3704, 2023.

[55] ——, "Collaborative learning on the edges: A case study on connected vehicles," *Proceedings of USENIX HotEdge*, pp. 1–7, 2022.

[56] Y. Zhang *et al.*, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.