informatics

MDPI

Article

TERA: A Trade-Off Evaluation and Resource-Aware Framework for Spam and Phishing Email Detection

Chanankorn Jandaeng [1,†], Peeravit Koad [1], Mohamad Fadli Zolkipli [2] and Jurairat Phuttharak [3,*,†]

[1] Informatics Innovative Center of Excellence (IICE), School of Informatics, Walailak University, Nakhon Si Thammarat 80160, Thailand; cjundang@gmail.com (C.J.); harrykoad@gmail.com (P.K.)
[2] School of Computing, Universiti Utara Malaysia (UUM), Sintok 06010, Malaysia; m.fadli.zolkipli@uum.edu.my
[3] Faculty of Commerce and Management, Prince of Songkla University, Trang Campus, 102, Khuan Pring, Mueang Trang, Trang 92000, Thailand
* Correspondence: jurairat.b@psu.ac.th
† These authors contributed equally to this work.

Abstract

Email spam and phishing detection is typically evaluated using accuracy-centric metrics under implicitly unconstrained computational settings. However, in practical deployment scenarios—particularly in real-time and resource-constrained environments—models with comparable predictive performance may differ substantially in inference latency and resource usage, directly affecting their operational feasibility. This paper introduces TERA, a deployment-aware evaluation framework that formulates model assessment as a constraint-aware decision problem. Instead of aggregating performance and efficiency into a single objective, TERA treats predictive performance as a feasibility requirement that defines an admissible set of models. Within this feasible region, operational factors such as latency and resource usage are used to differentiate among candidates through structured, multi-dimensional analysis. Experiments on benchmark email datasets show that multiple models achieve comparable detection performance, forming a region of predictive equivalence. Within this region, significant variations in latency and resource consumption are observed, indicating that predictive equivalence does not imply deployment equivalence. These findings demonstrate that accuracy-based evaluation alone may provide limited guidance for deployment-oriented model selection. By explicitly separating feasibility constraints from preference-based trade-offs, TERA enables transparent and deployment-aligned model evaluation. The framework supports consistent comparison and selection among accuracy-comparable models without altering the role of detection effectiveness as a primary requirement, thereby complementing existing evaluation practices with a structured decision-oriented perspective.

Keywords: phishing detection; spam detectio; deployment-aware evaluation; constraint-aware model selection; multi-dimensional evaluation; inference latency; resource-constrained systems; cybersecurity; machine learning; pareto analysis

Academic Editor: Antony Bryant

Received: 17 February 2026
Revised: 24 April 2026
Accepted: 6 May 2026
Published: 12 May 2026

1. Introduction

Email remains a fundamental communication medium in modern information systems, supporting daily operations across enterprises, government agencies, and digital platforms [1]. Its widespread adoption, however, has also made it a persistent attack surface

in cybersecurity [2]. Spam and phishing emails continue to serve as primary vectors for credential theft, malware delivery, and initial access in multi-stage cyberattacks [3,4].

The threat landscape has further intensified with the emergence of AI-generated phishing emails [5]. By leveraging large language models, adversaries can produce linguistically fluent, context-aware, and highly personalized messages [6]. These advances significantly reduce the effectiveness of traditional detection mechanisms and increase the difficulty of distinguishing malicious content from legitimate communication [5,7].

In response, a large body of research has focused on improving phishing detection using machine learning and, more recently, transformer-based models [7]. These approaches have achieved high levels of predictive performance, and maximizing detection accuracy remains the primary objective in phishing detection research [8]. This work does not challenge the importance of accuracy, nor attempt to replace accuracy-centric evaluation [9]. Instead, it focuses on a complementary problem that arises once baseline detection performance becomes comparable across models: how to systematically select among models under real-world operational constraints [10].

To address this limitation, we propose TERA (Trade-off Evaluation and Resource-Aware), a deployment-oriented evaluation framework for phishing detection model selection. Unlike prior approaches, TERA introduces a fundamentally different perspective based on three key principles. First, model selection is explicitly formulated as a deployment-constrained decision problem, where feasibility constraints such as latency and resource limits define the admissible set of models. Second, the framework preserves the multi-dimensional structure of detection effectiveness, latency, and resource usage through a unified evaluation space, avoiding reduction to a single scalar objective. Third, TERA establishes a direct mapping from evaluation outcomes to deployment decisions, enabling context-aware selection aligned with operational requirements rather than leaderboard-style ranking.

This work does not aim to replace or challenge accuracy-centric model development. Instead, it addresses a subsequent decision layer that arises once multiple models achieve comparable predictive performance. In such settings, model selection is no longer determined solely by marginal differences in accuracy, but by the need to satisfy deployment constraints such as latency, memory usage, and system responsiveness.

TERA operates on top of accuracy-optimized models and assumes that candidate models already satisfy a high level of detection effectiveness. Rather than competing with accuracy optimization, the framework introduces a complementary perspective by structuring how models are selected under deployment constraints.

Importantly, the proposed framework does not redefine model quality in terms of computational cost, nor does it trade detection effectiveness for efficiency. Instead, predictive performance is treated as a feasibility requirement, ensuring that all selected models maintain competitive detection capability, while operational characteristics are used to differentiate among admissible candidates. In this sense, TERA complements accuracy-centric research by providing a structured mechanism for deployment-oriented decision-making.

The main contributions of this work are as follows:

- We formulate phishing detection model selection as a constraint-aware decision problem under deployment conditions, explicitly separating feasibility requirements from preference-based trade-offs.
- We introduce a multi-dimensional evaluation framework that jointly represents detection effectiveness, latency, and resource usage without reducing them to a single scalar objective.

- We demonstrate, through empirical evaluation, that models with comparable accuracy can exhibit substantial differences in operational cost, highlighting the importance of deployment-aware model selection.

To guide the analysis, this study addresses the following research questions:

- RQ1: How do classical and transformer-based models compare in detection effectiveness under identical conditions?
- RQ2: How do latency, memory usage, and model footprint influence model selection among similarly accurate candidates?
- RQ3: How does a constraint-aware evaluation framework improve model selection compared to accuracy-based ranking?

Unlike prior studies that focus on model evaluation, this work introduces a decision-oriented methodology that operates on top of multi-dimensional evaluation outputs. The contribution lies not in improving model performance, but in enabling structured model selection under explicit deployment constraints.

Overall, this work bridges the gap between predictive evaluation and deployment-oriented decision-making by providing a principled framework for selecting phishing detection models under realistic operational constraints.

2. Background and Related Works

2.1. Email Spam and Phishing as an Evolving Threat

Email systems remain a critical attack surface in modern cybersecurity, where spam and phishing attacks continue to serve as primary vectors for credential theft, malware delivery, and unauthorized access [2,3].

Recent advances in large language models have further intensified this threat by enabling the generation of linguistically fluent and context-aware phishing emails [5,6]. These developments challenge detection approaches that rely primarily on surface-level patterns and increase the demand for models capable of capturing deeper semantic characteristics.

At the same time, practical deployment environments introduce additional considerations beyond predictive performance. In operational systems such as secure email gateways and real-time filtering pipelines, factors such as inference latency, throughput, and resource availability influence the feasibility of detection models.

These observations highlight that phishing detection is not only a problem of improving predictive effectiveness, but also of ensuring that detection mechanisms remain applicable under realistic operational conditions. This dual perspective motivates the need for evaluation approaches that consider both detection performance and deployment feasibility in a structured manner.

2.2. Machine Learning Approaches for Phishing Detection

Phishing detection has traditionally relied on classical machine learning techniques, including Naïve Bayes, support vector machines, decision trees, and ensemble models, which are widely adopted due to their efficiency, scalability, and interpretability [11,12]. These methods typically employ sparse lexical representations, such as bag-of-words and TF–IDF, enabling effective pattern recognition with relatively low computational overhead [7].

With the increasing sophistication of phishing attacks, particularly those exploiting contextual and semantic cues, deep learning approaches have gained prominence. Models such as recurrent neural networks and transformer-based architectures provide enhanced capability in capturing contextual dependencies and semantic intent [8,13]. These advances have contributed to improved detection effectiveness, particularly in complex and context-aware attack scenarios.

However, the improved representational capacity of these models is often accompanied by increased computational cost, including higher inference latency, memory consumption, and deployment complexity [14]. These characteristics may affect their applicability in operational environments where real-time processing and resource constraints are critical.

Consequently, phishing detection methods exhibit diverse performance profiles across both predictive effectiveness and operational characteristics. This diversity becomes particularly relevant in scenarios where multiple models achieve comparable detection performance, yet differ substantially in their computational requirements. Such conditions highlight the need for evaluation perspectives that extend beyond predictive accuracy and support structured comparison under deployment constraints.

2.3. Accuracy-Centric Evaluation Practices

Accuracy-centric evaluation remains the dominant paradigm in phishing and intrusion detection research, where models are primarily assessed using metrics such as accuracy, F1-score, and AUC [9,15]. This approach emphasizes predictive correctness under controlled experimental conditions and serves as the foundation for benchmarking and comparative analysis across studies [10,16].

Prior work has shown that such evaluation protocols are typically conducted under unconstrained computational settings, where system-level factors such as inference latency, memory usage, and throughput are not explicitly incorporated into model comparison [17,18]. While this assumption is appropriate for assessing predictive performance, it provides limited insight into how models behave under deployment-oriented conditions.

Recent empirical studies further indicate that multiple models can achieve statistically comparable predictive performance under standardized benchmarks [9]. In such scenarios, accuracy-based ranking alone may offer limited discriminative power for selecting among candidate models, as it does not capture differences in operational characteristics.

Under deployment-oriented conditions, these differences become increasingly relevant. In resource-constrained and real-time environments, variations in latency and resource consumption can directly influence system feasibility and responsiveness [9]. In this context, accuracy remains a necessary requirement, but additional considerations are needed to differentiate among models with comparable predictive effectiveness.

These observations highlight a limitation not in accuracy as an evaluation criterion itself, but in the absence of a structured mechanism for incorporating system-level considerations into model selection. This gap motivates the need for evaluation approaches that support consistent and deployment-aware decision-making under such conditions.

2.4. Transformer-Based and Resource-Efficient Detection Models

Transformer-based models have become increasingly prominent in phishing detection due to their ability to capture semantic context and long-range dependencies [13,19]. Empirical studies show that these models improve robustness against context-aware and socially engineered attacks by modeling linguistic and semantic patterns beyond surface-level features [20]. These advances have contributed to improved detection effectiveness, particularly in complex and evolving threat scenarios.

However, the increased representational capacity of transformer-based models is typically accompanied by higher computational requirements. Prior work reports that such approaches often incur increased inference latency, memory consumption, and energy usage compared to classical methods, which may affect their applicability in real-time or resource-constrained environments [14,21].

To address these challenges, a range of efficiency-oriented strategies has been proposed, including model compression and distillation techniques (e.g., DistilBERT, TinyBERT),

lightweight architectures (e.g., ALBERT), and hybrid pipelines that decouple representation learning from classification [22–24]. These approaches aim to improve computational efficiency while maintaining strong detection performance.

Despite these advances, evaluation practices for transformer-based and resource-efficient models remain largely centered on predictive performance. System-level characteristics, such as latency and resource usage, are often reported independently rather than being systematically incorporated into model comparison.

This limitation becomes particularly relevant in scenarios where multiple models achieve comparable detection effectiveness but differ in their operational characteristics. In such cases, the absence of a structured evaluation mechanism makes it difficult to consistently assess deployment suitability across heterogeneous model architectures.

2.5. Multi-Objective Evaluation and Research Gap

Recent advances in machine learning have extended model evaluation beyond predictive accuracy to incorporate additional operational dimensions, including latency, computational cost, and resource utilization. This shift reflects the increasing importance of deployment-aware considerations in real-world systems, where model effectiveness alone is insufficient for practical adoption.

Existing approaches to multi-objective evaluation can be broadly categorized into three groups. First, cost-sensitive and multi-objective optimization methods integrate multiple performance criteria into a unified objective function, enabling joint optimization of accuracy and efficiency [25–27]. While effective under predefined priorities, these approaches rely on scalarization, implicitly assuming that heterogeneous performance dimensions are directly comparable within a single objective space. As a result, feasibility constraints—such as latency bounds or resource limits—are not explicitly enforced, but instead treated as tradeable quantities.

Second, latency-aware and resource-aware evaluation approaches extend conventional benchmarking by reporting system-level metrics alongside predictive performance [17,28,29]. Although these methods provide valuable insights into the operational characteristics of models, the reported metrics are typically presented independently and are not systematically integrated into the model selection process. Consequently, model comparison remains largely descriptive, without a structured mechanism for decision-making.

Third, deployment-oriented system designs address operational constraints through model adaptation techniques, such as compression, lightweight architectures, and hybrid inference pipelines [14,22,23,30]. While effective within specific contexts, these approaches are often tightly coupled to particular model families and do not provide a generalizable framework for evaluating heterogeneous models under unified criteria.

In addition, several recent studies in phishing detection focus primarily on improving predictive performance or conducting comparative evaluation [13,19,31]. These works remain largely accuracy-centric and do not explicitly incorporate deployment constraints into the model selection process.

From the perspective of this study, these limitations can be understood in relation to the three research questions. Prior work extensively addresses RQ1 by comparing predictive performance across models, and partially addresses RQ2 by incorporating operational metrics such as latency and resource usage. However, none of these approaches provide a systematic solution to RQ3, which concerns how such multi-dimensional information should be used to support model selection under deployment constraints as shown in Table 1.

Table 1. Comparison of prior work highlighting limitations in deployment-aware evaluation, decision support, and alignment with research questions. ("✓" and "×" are cover and discover the research question, respectively)

Study	Year	Domain	Multi-Dimensional Focus	Key Limitation	RQ1	RQ2	RQ3
[25]	2020	Intrusion Detection	Accuracy–cost trade-off via multi-objective optimization	Treats factors as jointly tradeable; lacks explicit feasibility constraints	✓	✓	×
[19]	2021	Phishing Detection	Transformer-based semantic robustness	Focuses on representation; ignores operational feasibility	✓	×	×
[26]	2021	Security ML	Cost-sensitive learning objective	Combines objectives; no separation of feasibility and optimization	✓	✓	×
[9]	2021	Security ML Evaluation	Dataset-driven evaluation	Highlights benchmarking issues; lacks decision support	✓	×	×
[17]	2022	Edge Computing Security	Latency-aware evaluation	Reports latency but not integrated into decision	✓	✓	×
[28]	2022	Intelligent Systems	Performance-oriented evaluation	Focus on performance; lacks decision formulation	✓	×	×
[14]	2023	Edge-based IDS	Resource-aware deployment constraints	Limited to specific models; lacks cross-model framework	×	✓	×
[27]	2023	ML Optimization	Multi-objective optimization (accuracy, cost, efficiency)	Joint optimization; no feasibility separation	✓	✓	×
[31]	2023	Intelligent Systems	Accuracy-driven evaluation	Evaluation-centric; no constraint-aware selection	✓	×	×
[13]	2024	Phishing Detection	Comparative deep learning evaluation	Remains accuracy-centric; ignores constraints	✓	×	×
[30]	2024	AI Applications	Model-centric optimization	No constrained decision formulation	✓	×	×
[29]	2024	Intelligent Systems	Multi-metric system evaluation	Descriptive metrics; no structured decision process	✓	✓	×
TERA	2026	Deployment-aware ML	Constraint-aware multi-dimensional evaluation	Separates feasibility from preference; structured decision formulation	✓	✓	✓

This limitation becomes particularly critical in performance-homogeneous regimes, where multiple models exhibit statistically comparable predictive effectiveness. In such scenarios, accuracy-based ranking alone is insufficient to guide model selection, as differences in latency, memory usage, and model footprint become decisive factors for deployment.

Existing state-of-the-art methods primarily focus on improving predictive performance or optimizing specific objectives such as accuracy, efficiency, or robustness. However, these approaches do not explicitly address the problem of model selection under deployment constraints, particularly in scenarios where multiple models exhibit comparable predictive performance.

More fundamentally, current methods conflate two distinct aspects of the decision process: (i) feasibility, which determines whether a model satisfies deployment constraints, and (ii) preference, which governs the selection among feasible candidates. Without a clear separation between these components, model selection remains under-specified and context-dependent.

To address this gap, a formulation is required in which feasibility constraints are explicitly enforced prior to trade-off analysis, thereby restricting the decision space to an admissible set of deployable models. Within this constrained space, preference-based criteria can then be applied in a structured and interpretable manner.

Building on this perspective, the following section introduces the TERA framework, which formulates model selection as a constraint-aware decision problem. By explicitly separating feasibility from preference while preserving predictive performance as a non-negotiable requirement, TERA provides a principled mechanism for integrating multi-dimensional evaluation into deployment-oriented decision-making.

3. TERA Framework

This section introduces TERA, a constraint-aware framework for deployment-oriented model evaluation and selection. The framework addresses the limitation identified in Section 2 by providing a structured mechanism for decision-making when multiple models exhibit comparable predictive performance under operational constraints.

We formally formulate model selection as a constraint-aware decision problem by explicitly defining a feasible set under deployment constraints and separating feasibility from preference-based ranking. This formulation distinguishes TERA from conceptual evaluation frameworks by providing a mathematically grounded and operational decision structure.

TERA formulates model selection as a constrained decision problem. Predictive performance is treated as a feasibility requirement that determines the set of admissible models, while operational characteristics—such as inference latency and resource usage—are used to guide selection among feasible candidates. Rather than performing model comparison, TERA provides a structured mechanism for decision-making based on evaluation outputs under deployment constraints.

To operationalize this formulation, TERA is organized into three layers: (1) constraint formulation for defining the feasible set, (2) multi-dimensional representation of model performance, and (3) decision-oriented selection based on deployment preferences.

This layered structure separates feasibility from preference, enabling transparent and deployment-aligned evaluation without altering the role of accuracy as the primary performance objective.

3.1. Layer 1: Constraint Formulation

This layer defines the admissible region of deployment by formalizing feasibility as a set of explicit constraints. Rather than evaluating models under unconstrained conditions, only models that satisfy predefined deployment requirements are considered for subsequent analysis.

Let M denote the set of candidate models. Each model $m \in M$ is characterized by its predictive performance and operational properties, including detection accuracy $A(m)$, inference latency $L(m)$, and resource usage $R(m)$.

Feasibility is defined relative to three deployment parameters: a latency constraint β, a resource constraint γ, and an allowable accuracy degradation threshold α. Detection effectiveness is enforced with respect to a reference model with accuracy A_{ref}. The accuracy degradation of a model m is defined as:

$$\Delta A(m) = A_{\text{ref}} - A(m),$$

and the feasibility condition requires:

$$\Delta A(m) \leq \alpha.$$

In addition, operational constraints are defined as:

$$L(m) \leq \beta, \quad R(m) \leq \gamma.$$

Together, these conditions define the feasible model set:

$$M_{\text{feasible}} = \{m \in M \mid \Delta A(m) \leq \alpha,\ L(m) \leq \beta,\ R(m) \leq \gamma\}.$$

Models that do not satisfy these constraints are excluded a priori. This formulation ensures that predictive performance is treated as a feasibility requirement rather than an

optimization objective, while latency and resource usage act as deployment constraints that determine admissibility. The resulting feasible set provides the foundation for structured evaluation and decision-making in subsequent layers.

The TERA framework assumes that candidate models have already achieved a comparable and sufficiently high level of predictive performance. Under this assumption, the objective is not to further optimize detection accuracy, but to differentiate among feasible models based on their operational characteristics within deployment constraints. Accordingly, predictive performance is treated as a feasibility condition, while latency and resource usage serve as decision variables for deployment-oriented selection.

3.2. Layer 2: Model Evaluation Space

Building on the feasible set defined in Layer 1, this layer introduces a multi-dimensional evaluation space for structured comparison of candidate models. Each model is represented by its predictive performance and operational characteristics, enabling comparison without reducing multiple dimensions into a single objective.

Each model $m \in M_{\text{feasible}}$ is characterized by three attributes: accuracy $A(m)$, latency $L(m)$, and resource usage $R(m)$. These attributes define a joint evaluation space in which models cannot be fully ordered by a single scalar metric.

For consistent representation, models are mapped into the TERA evaluation space:

$$T(m) = \langle A(m), 1 - L(m), 1 - R(m) \rangle,$$

where higher values indicate more desirable outcomes.

Model comparison is performed using Pareto dominance. A model m_i is dominated if there exists another model m_j such that:

$$A(m_j) \geq A(m_i), \quad L(m_j) \leq L(m_i), \quad R(m_j) \leq R(m_i),$$

with strict improvement in at least one dimension. The set of non-dominated models forms the Pareto frontier.

Within TERA, Pareto analysis serves as a filtering mechanism that removes strictly inferior models while preserving feasible alternatives. This process operates exclusively within M_{feasible}, ensuring consistency with the constraints defined in Layer 1.

This layer provides a structured representation of model performance that preserves multi-dimensional relationships and prepares the candidate set for decision-making in Layer 3.

3.3. Layer 3: Decision and Deployment Mapping

Building on the feasible set defined in Layer 1 and the structured evaluation space in Layer 2, this layer formulates model selection as a deployment-oriented decision process over admissible models.

Let M_{feasible} denote the set of models that satisfy all feasibility constraints. Model selection is performed in two stages:

1. Feasibility enforcement: Restrict consideration to models in M_{feasible}.
2. Preference-based selection: Rank and select among feasible models according to deployment-specific priorities.

To express preference within the feasible set, a context-dependent scoring function is defined:

$$S_{\text{TERA}}(m) = \alpha A(m) - \beta L(m) - \gamma R(m),$$

where (α, β, γ) are non-negative parameters reflecting deployment priorities.

This scoring function does not define a global optimization objective; instead, it induces a preference ordering within M_{feasible} without affecting feasibility. In this formulation, accuracy functions as a constraint that determines admissibility, while latency and resource usage guide selection among admissible candidates.

The weights (α, β, γ) in Table 2 are therefore not optimization parameters, but represent deployment-specific preferences. Importantly, they are applied only after feasibility constraints are enforced and thus do not influence which models are admissible. This design ensures a clear separation in which feasibility remains objective, while preferences remain configurable. For interpretability, the parameters may be normalized such that:

$$\alpha + \beta + \gamma = 1.$$

Table 2. Interpretation of preference weights in the TERA decision function

Weight	Interpretation	Deployment Implication
α (Accuracy)	Emphasizes detection effectiveness	Favors models with higher predictive performance; suitable for high-security scenarios
β (Latency)	Penalizes inference delay	Prioritizes low-latency models; suitable for real-time systems
γ (Resource)	Penalizes memory and model size	Prefers lightweight models; suitable for resource-constrained environments

The weight parameters (α, β, γ) provide an interpretable mechanism for expressing deployment preferences. Rather than defining a global optimization objective, these parameters act as context-dependent indicators that guide selection within the feasible set. Different deployment scenarios correspond to different weight configurations, enabling flexible adaptation without altering feasibility constraints.

By separating feasibility from preference, this formulation preserves detection effectiveness as a non-negotiable constraint, while allowing deployment preferences to be expressed explicitly through interpretable weighting within the feasible set.

3.4. Practical Interpretation and Framework Characteristics

This section summarizes the key characteristics of the TERA framework and its implications for deployment-oriented model selection.

TERA differs from existing evaluation approaches in how performance metrics are functionally interpreted. Prior methods typically treat accuracy and efficiency either as jointly optimizable objectives or as independent metrics reported for comparison. In contrast, TERA assigns distinct roles to each dimension: predictive performance defines feasibility, while operational characteristics guide selection within the feasible set.

This separation enables a structured evaluation process in which only admissible models are considered for comparison. As a result, model selection is performed over a well-defined set of candidates that satisfy deployment requirements, rather than over the entire model space.

The framework is inherently adaptive to deployment conditions. Changes in operational requirements can be incorporated through constraint parameters and preference parameters, allowing the evaluation process to remain aligned with system constraints and application contexts.

In practical settings, model selection follows the structure defined by the three layers. Feasibility constraints determine the admissible set (Layer 1), the evaluation space preserves multi-dimensional performance relationships (Layer 2), and preference-based selection resolves deployment-specific priorities (Layer 3). This process supports transparent interpretation of model differences under realistic operational conditions.

Overall, TERA provides a structured mechanism for reasoning about deployment-aware model selection, enabling consistent comparison among models with comparable predictive performance while maintaining the requirement for high detection effectiveness.

4. Research Methodology

This section presents the methodology used to operationalize the TERA framework for deployment-aware model evaluation. Building on the conceptual formulation introduced in Section 3, the methodology specifies how theoretical constructs are instantiated as measurable quantities and integrated into a unified evaluation pipeline.

The objective is to ensure that model evaluation reflects deployment-relevant conditions by aligning performance measurement, constraint enforcement, and decision-making within a consistent experimental setting. This alignment enables systematic comparison among models that exhibit comparable predictive performance, while preserving the role of accuracy as the primary evaluation criterion.

The following subsections describe the operationalization of framework variables, the system architecture used for evaluation, and the experimental setup supporting deployment-aware model selection.

4.1. Operationalization of the TERA Framework

To translate the theoretical formulation introduced in Section 3 into an executable evaluation process, the key performance variables are explicitly mapped to measurable quantities within a unified experimental pipeline. This operationalization ensures that feasibility constraints and decision criteria are grounded in system-level observations under deployment-consistent conditions.

Detection effectiveness $A(m)$ is computed using standard classification metrics (e.g., accuracy and F1-score) on the evaluation dataset. Inference latency $L(m)$ is measured as the end-to-end processing time per email instance, including preprocessing, feature encoding, and model inference. Resource usage $R(m)$ is quantified through peak memory consumption and model footprint observed during execution. These measurements are collected directly within the execution flow, ensuring consistency between evaluation and deployment behavior.

The deployment constraints are instantiated as thresholds over these measured quantities. Specifically, α defines the maximum allowable degradation from a reference detection performance, while β and γ represent upper bounds on latency and resource usage, respectively. This formulation enables direct enforcement of feasibility conditions and ensures that only deployable models are considered in subsequent analysis.

Figure 1 illustrates how this mapping is realized within the system architecture. The architecture functions as a methodological bridge between the theoretical framework and practical evaluation by integrating measurement, constraint enforcement, and decision support within a unified structure.

Within this structure, feasibility enforcement and preference-based selection are treated as distinct stages, consistent with the constraint-first formulation. A unified execution pipeline ensures that all models are evaluated under identical conditions, while a deployment-consistent measurement component captures latency and resource usage directly during inference.

The resulting measurements are used to enforce feasibility conditions ($\Delta A \leq \alpha$, $L \leq \beta$, $R \leq \gamma$), corresponding to Layer 1 of the TERA framework. Models that satisfy these constraints are passed to the decision stage (Layer 3), where deployment-specific preferences guide selection within the feasible set.

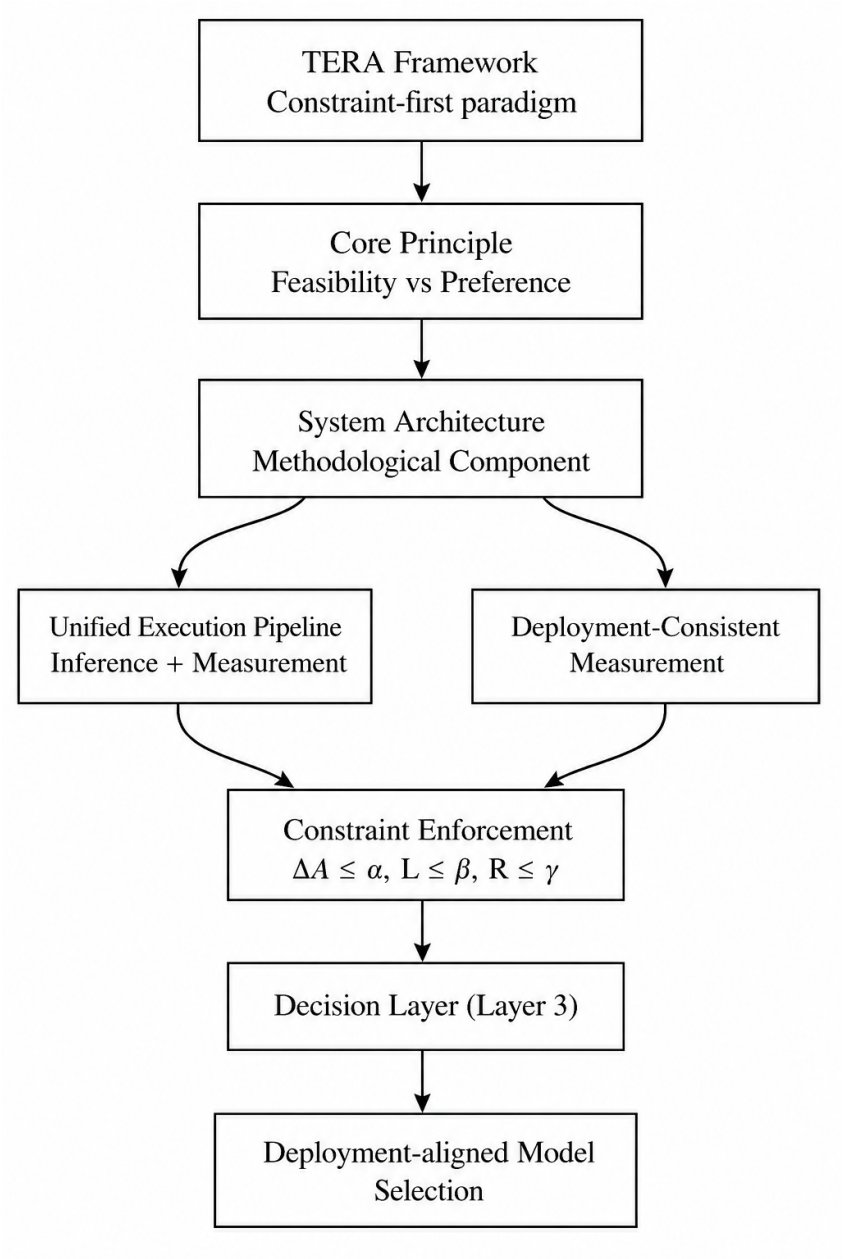

Figure 1. System architecture as a methodological bridge between the constraint-first TERA framework and deployment-aligned model selection.

Overall, this operationalization establishes a concrete link between theoretical constructs and system-level evaluation, enabling consistent, reproducible, and deployment-aware model selection without altering the role of detection effectiveness as the primary performance requirement.

4.2. System Architecture and Execution Model

This section describes the system architecture used to implement the TERA framework under deployment-aligned conditions. Building on the operationalization defined in Section 4.1, the architecture instantiates performance measurement, constraint enforcement, and model evaluation within a unified execution environment.

Figure 2 presents the overall architecture, which follows a single-pass execution model consistent with real-time email security systems. Each email instance is processed independently through a deterministic pipeline, ensuring that latency and resource usage remain bounded and directly measurable.

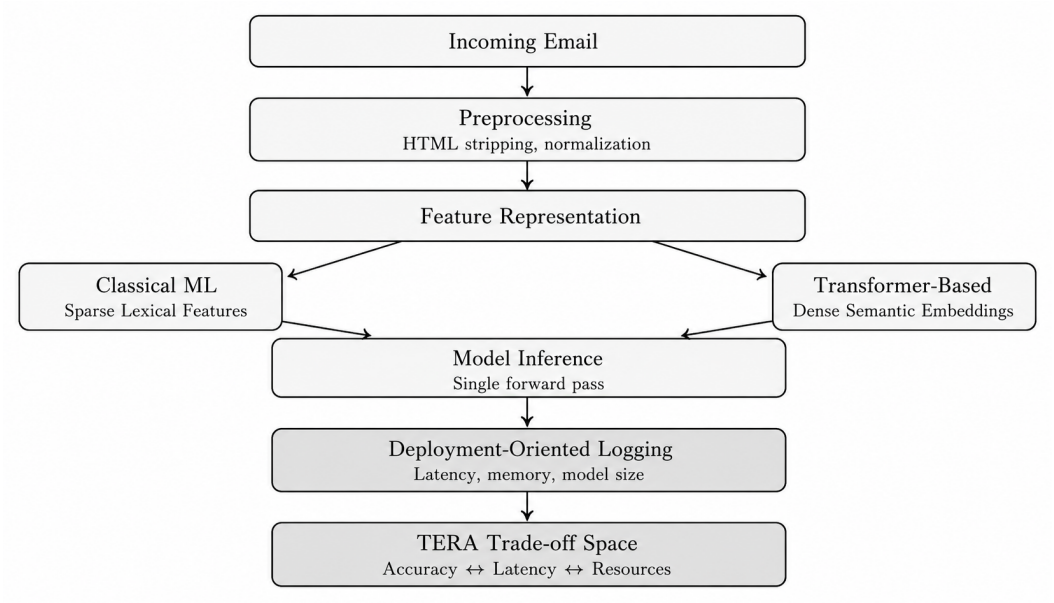

Figure 2. Unified system architecture illustrating the single-pass execution pipeline for both experimental evaluation and deployment, where feature representation is followed by parallel processing of the two proposed methods and comparative analysis of their experimental results.

To ensure deployment consistency, the system enforces a unified execution model in which batching, caching, external retrieval, and auxiliary optimization mechanisms are not employed. This design guarantees that all models are evaluated under identical conditions and that measured performance reflects deployable system behavior.

4.2.1. Unified Inference Pipeline

The architecture is implemented as a sequential pipeline consisting of preprocessing, feature encoding, model inference, and metric logging. Each email is processed using a single forward-pass execution, with feature encoding performed once per instance.

Operational metrics, including inference latency, peak memory usage, and model footprint, are recorded during execution. Integrating measurement directly into the pipeline eliminates discrepancies between evaluation and deployment, enabling reliable analysis of performance and operational cost.

4.2.2. Model Integration and Processing Flow

The system supports two complementary model pipelines that capture different regions of the deployment trade-off space.

Classical machine learning models prioritize efficiency. Emails are represented using sparse lexical features, enabling low-latency inference and predictable memory consumption. A diverse set of classifiers, including Multi-Layer Perceptron, XGBoost, and LightGBM, is used to represent different learning paradigms under resource-constrained conditions.

Transformer-based models emphasize contextual understanding. Email text is encoded using pretrained transformer encoders to produce dense semantic embeddings. To maintain deployment feasibility, representation learning is decoupled from classification, and lightweight classifiers operate on fixed embeddings. Although these models incur higher computational cost, their execution remains predictable within the unified pipeline.

4.2.3. Feature Representation Strategies

Feature construction is designed to balance representational expressiveness and computational efficiency while remaining compatible with real-time constraints. Within the

single-pass pipeline, features are generated once per instance without intermediate storage or retrieval, ensuring stable latency and resource usage.

For classical models, sparse representations such as count-based and TF–IDF vectors are employed. These representations produce high-dimensional but sparse feature vectors, significantly reducing memory usage and computational overhead.

For transformer-based models, dense embeddings are generated via a single forward pass of a pretrained encoder. These embeddings capture semantic and contextual information beyond surface-level features, improving robustness against context-aware phishing attacks. To preserve deployment consistency, embeddings are generated on-the-fly and directly consumed by downstream classifiers without additional infrastructure.

4.2.4. TERA-Based Model Selection Algorithm

Model selection is performed using a constraint-aware procedure based on the TERA framework. Given a candidate set $\mathcal{M}$, each model is evaluated to obtain $(A(m), L(m), R(m))$ under the unified execution environment as shown in Algorithm 1.

For example, given a candidate set of models with measured (accuracy, latency, memory), the algorithm filters infeasible models based on constraints and returns either a Pareto set or a single selected model depending on deployment preferences.

A feasible set $\mathcal{F}$ is constructed by enforcing deployment constraints on latency and resource usage. Pareto dominance is then applied to identify non-dominated models within $\mathcal{F}$. When deployment preferences are specified, a scoring function is used to select a final model from the resulting Pareto set.

This procedure operates offline and does not affect runtime inference behavior. Its computational complexity is $O(n^2)$ in the number of candidate models, which remains acceptable for model selection scenarios.

Algorithm 1 TERA-Based Constraint-Aware Model Selection

Require: Candidate model set $\mathcal{M}$; latency constraint λ; resource constraint ρ; optional preference weights (α, β, γ)
Ensure: Selected model set $\mathcal{M}^\star \subseteq \mathcal{M}$

for all $m \in \mathcal{M}$ **do**
  Evaluate detection effectiveness $A(m)$
  Measure inference latency $L(m)$
  Measure resource usage $R(m)$
end for
Construct feasible set
$\mathcal{F} = \{m \in \mathcal{M} \mid L(m) \leq \lambda \wedge R(m) \leq \rho\}$
Compute Pareto-optimal set
$\mathcal{P} = \{m \in \mathcal{F} \mid \neg\exists m' \in \mathcal{F} :$
  $A(m') \geq A(m),\ L(m') \leq L(m),\ R(m') \leq R(m)\}$
if preference weights (α, β, γ) are specified **then**
  for all $m \in \mathcal{P}$ **do**
    $S_{\text{TERA}}(m) = \alpha A(m) - \beta L(m) - \gamma R(m)$
  end for
  $\mathcal{M}^\star = \arg\max_{m \in \mathcal{P}} S_{\text{TERA}}(m)$
else
  $\mathcal{M}^\star = \mathcal{P}$
end if

return $\mathcal{M}^\star$

Following Algorithm 1, model selection is performed in three stages.

First, candidate models are evaluated and filtered based on deployment constraints to construct the feasible set $\mathcal{F}$. For example, all models satisfy the specified constraints and are therefore included in $\mathcal{F}$.

Second, Pareto dominance is applied to identify the Pareto-optimal set $\mathcal{P}$, consisting of models that are not dominated in terms of detection performance, latency, and resource usage. Finally, a single model is selected using the TERA scoring function:

$$S_{\text{TERA}}(m) = \alpha A(m) - \beta L(m) - \gamma R(m),$$

which requires user-defined preference weights (α, β, γ).

In the absence of explicit preferences, a balanced assumption ($\alpha = \beta = \gamma = 1$) can be adopted. Under this setting, TF-IDF with XGBoost is selected as the final model, as it provides the most favorable trade-off between predictive performance, latency, and resource usage.

Overall, the architecture provides a consistent and reproducible execution model that ensures alignment between theoretical formulation and empirical evaluation, supporting deployment-aware analysis of model performance.

4.3. Datasets and Preprocessing

This section describes the dataset construction process, preprocessing pipeline, and statistical characteristics of the resulting corpus. The design emphasizes reproducibility and alignment with deployment conditions, ensuring that both data preparation and evaluation reflect realistic operational environments.

4.3.1. Dataset Construction and Consolidation

The dataset used in this study is derived from publicly available email corpora, including the phishing email dataset proposed by [32], along with additional spam and legitimate email collections to capture diverse attack patterns and normal communication.

The final dataset consists of 23,498 email messages distributed across three classes: 11,322 ham samples (48.18%), 7328 phishing samples (31.19%), and 4848 spam samples (20.63%). This distribution reflects realistic class imbalance observed in real-world communication while maintaining sufficient representation for all categories.

All data sources are integrated through a standardized consolidation process, including: (i) merging into a unified corpus, (ii) label harmonization into three classes, (iii) removal of duplicate and near-duplicate entries to prevent data leakage, and (iv) filtering of corrupted or empty samples. All instances are represented as labeled text suitable for classification.

The dataset construction process, class distribution, and preprocessing steps are explicitly documented to ensure transparency and reproducibility. The processed dataset and experimental artifacts are available from the corresponding author upon reasonable request, and a public repository is being prepared.

4.3.2. Preprocessing Pipeline

Text preprocessing is designed to be lightweight and compatible with real-time deployment constraints. Each email is normalized by converting text to lowercase and removing leading and trailing whitespace. Non-textual elements, including URLs, email addresses, and numeric tokens, are replaced with placeholder tokens <URL>, <EMAIL>, and <NUM>) using regular expressions.

This masking strategy preserves structural cues relevant to phishing detection while reducing vocabulary sparsity and mitigating overfitting to instance-specific patterns. Consecutive whitespace is collapsed to maintain compact representations.

To ensure predictable latency and bounded memory usage, each email is truncated to a maximum length of 3000 characters. No stemming, lemmatization, or complex linguistic processing is applied, ensuring linear-time preprocessing consistent with the single-pass execution model defined in Section 4.2.

4.3.3. Dataset Characteristics

To characterize the dataset under deployment-relevant conditions, statistical properties of email length after preprocessing are analyzed. Table 3 summarizes class-wise statistics, and Figure 3 illustrates the corresponding distributions.

Table 3. Statistical summary of email text length after preprocessing.

Class	Records	Mean	Median	90th Percentile
Ham	11,322 (48.18%)	245.7	168	555
Phishing	7328 (31.19%)	206.6	142	538
Spam	4848 (20.63%)	123.0	58	371

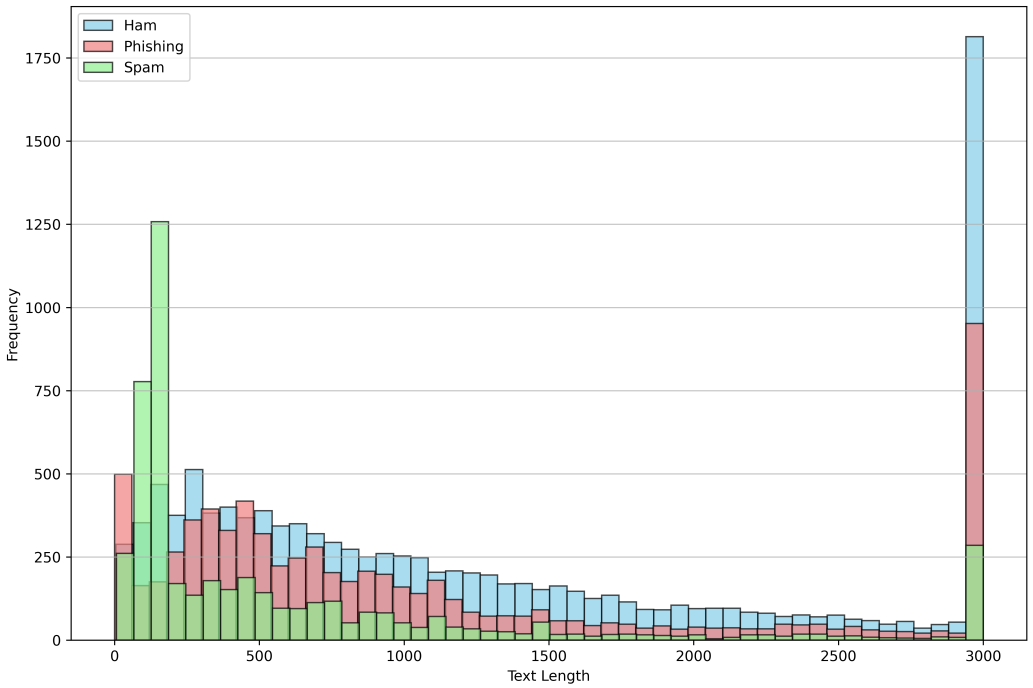

Figure 3. Distribution of email text length (in tokens) across ham, phishing, and spam classes after preprocessing, illustrating realistic variability in operational email traffic.

The results indicate heterogeneous length distributions across classes. Ham emails tend to be longer and more variable, phishing emails exhibit intermediate lengths, and spam emails are generally shorter.

These variations are particularly relevant in deployment settings, where input size directly influences inference latency and memory consumption. By explicitly quantifying these properties, the dataset provides a realistic basis for analyzing performance and efficiency trade-offs in subsequent evaluation.

4.3.4. Dataset Representativeness and Limitations

While the constructed dataset provides a standardized and reproducible benchmark for evaluating model behavior, it is important to acknowledge its limitations in representing evolving phishing threats. In particular, the dataset is derived from publicly available

corpora that may not fully capture recent advances in adversarial techniques, including AI-generated phishing emails.

Recent studies indicate that AI-generated phishing messages exhibit higher linguistic fluency and contextual adaptation, which may not be fully reflected in traditional datasets. As a result, the evaluation presented in this work primarily reflects model behavior under established attack patterns rather than emerging threat scenarios.

However, the objective of this study is not to optimize detection performance for a specific dataset, but to analyze model selection under deployment constraints. In this context, the use of established datasets ensures controlled comparison and reproducibility, while the proposed TERA framework remains applicable to more recent and domain-specific datasets.

Future work will extend the evaluation to incorporate datasets reflecting modern phishing techniques, including AI-generated and adaptive attack strategies, to further assess the generalizability of the framework.

4.4. Experimental Setup

This section describes the experimental configuration designed to ensure consistent, reproducible, and deployment-aligned evaluation. The setup enforces controlled execution conditions while explicitly incorporating system-level constraints to reflect realistic operational environments. In particular, the evaluation is conducted under deployment-oriented conditions, including single-instance inference, CPU-based execution, and real-time measurement of latency and resource usage. This design ensures that the reported results capture practical system behavior, rather than idealized benchmarking performance.

4.4.1. Execution Environment

Experiments were conducted on Google Colab using a CPU-only runtime (approximately 12–13 GB RAM, Linux-based system), without GPU acceleration or hardware-specific optimization. All models were evaluated under identical execution conditions to ensure fairness and comparability.

A fixed random seed (seed = 42) was applied across all stochastic components to support reproducibility. Performance metrics were recorded after steady-state execution, with warm-up runs excluded. Reported results correspond to the mean values over multiple runs to mitigate stochastic variability.

All experiments were conducted under a fixed configuration to ensure consistency and reproducibility. To avoid bias introduced by model-specific optimization, hyperparameters were not tuned individually. Instead, representative default configurations commonly used in prior studies and standard library implementations were adopted across all models. This approach is consistent with established experimental protocols in comparative machine learning studies, where fixed or default hyperparameter settings are used to ensure fairness and prevent overfitting to specific models or datasets [33–35].

For neural models, the Multi-Layer Perceptron (MLP) was configured with a single hidden layer of 100 units using ReLU activation. For ensemble-based methods, XGBoost was configured with 100 estimators and a maximum tree depth of 6, while LightGBM employed 31 leaves with a learning rate of 0.1. All remaining parameters followed default settings provided by their respective libraries.

This configuration ensures that the evaluation reflects intrinsic model characteristics and trade-offs, rather than performance gains resulting from hyperparameter tuning. Prior work has shown that extensive hyperparameter tuning can obscure fair comparison and reduce generalizability, particularly when comparing heterogeneous model families [34,35].

Consequently, the comparison focuses on deployment-relevant differences in predictive effectiveness and operational cost under consistent and unbiased conditions.

To support transparency and reproducibility, a public repository containing source code, configuration details, and instructions for reproducing the experiments will be released upon acceptance.

4.4.2. Deployment Constraints

The experimental design enforces deployment-oriented constraints consistent with real-time email security systems. Each email instance is processed independently using single-pass inference, ensuring bounded latency and predictable resource usage.

Batching, caching, and external retrieval mechanisms are not employed, as they may introduce variability and obscure true deployment behavior. Instead, all models are evaluated under a unified execution model aligned with the system architecture described in Section 4.2.

Operational metrics, including inference latency, peak memory consumption, and model footprint, are treated as primary evaluation criteria. These constraints reflect practical deployment environments such as email gateways and inline security systems, where computational resources and response time are strictly limited.

By enforcing these constraints at the experimental level, the evaluation ensures that measured performance is directly transferable to real-world deployment scenarios and supports reliable analysis of accuracy, latency, and resource trade-offs.

4.5. Evaluated Models

The evaluated models are selected to represent a diverse range of performance and operational characteristics within the deployment space considered by the TERA framework. The selection spans both classical machine learning approaches and transformer-based architectures, enabling systematic comparison across different levels of representational capacity and computational cost.

- Classical machine learning models are included as efficiency-oriented baselines. These models operate on sparse lexical representations and are characterized by low inference latency and minimal resource usage. Representative algorithms include Multi-Layer Perceptron (MLP), as well as ensemble-based methods such as XGBoost and LightGBM. These models reflect commonly deployed solutions in high-throughput and resource- constrained environments.
- Transformer-based models are incorporated to capture advanced semantic and contextual representations. Models from the BERT family, including both standard and lightweight variants, are used to evaluate the impact of increased representational capacity on detection effectiveness. These models typically incur higher computational cost but offer improved robustness against context-aware and linguistically sophisticated phishing attacks.

All models are evaluated under identical conditions within the unified execution framework described in Section 4.2. Hyperparameters are fixed prior to evaluation and applied consistently across all experiments. No model-specific optimization or tuning is performed, ensuring that observed differences reflect intrinsic model characteristics rather than configuration bias.

This selection strategy enables analysis of model behavior across a broad spectrum of accuracy, latency, and resource usage, providing a realistic basis for constraint-aware model selection under deployment requirements.

4.6. Evaluation Metrics

To support deployment-aware model evaluation, both predictive performance and operational characteristics are assessed. This dual perspective aligns with the TERA framework, where detection effectiveness defines feasibility, and system-level metrics inform selection among feasible models.

4.6.1. Predictive Performance Metrics

Detection effectiveness is evaluated using standard classification metrics:

- Accuracy, defined as

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

  where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.
- Precision, defined as

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

- Recall, defined as

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

- F1-score, defined as

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

These metrics provide a comprehensive assessment of classification performance under imbalanced and adversarial conditions commonly observed in phishing detection.

4.6.2. Operational Metrics

To assess deployment feasibility, system-level metrics are measured directly within the execution pipeline:

- Inference Latency: the end-to-end processing time required to classify a single email, including preprocessing, feature encoding, and model inference. Given N samples, average latency is computed as

$$\text{Latency}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} t_i,$$

  where t_i is the processing time of the i-th instance.
- Peak Memory Usage: the maximum memory consumption observed during inference, representing worst-case resource demand.
- Model Footprint: the serialized size of the model, reflecting storage requirements relevant to deployment and update scenarios.

4.6.3. Role in the TERA Framework

Within the TERA formulation, predictive performance metrics determine feasibility through the allowable degradation threshold ϵ, while operational metrics define deployment constraints (λ, ρ).

This separation ensures that accuracy is treated as a requirement for admissibility, whereas latency and resource usage are used to guide selection among feasible models. As a result, the evaluation framework enables multi-dimensional analysis without reducing

performance to a single scalar objective, supporting structured and deployment-aligned model selection.

4.7. Deployment-Aware Evaluation Protocol

This section defines the evaluation protocol used to assess model performance under deployment-aligned conditions. The protocol integrates predictive and operational measurements within a consistent execution framework, ensuring that evaluation outcomes reflect real-world system behavior.

All models are evaluated using the unified execution pipeline described in Section 4.2, where each email instance is processed independently under a single-pass inference model. Performance metrics are collected directly during execution, ensuring that latency and resource usage correspond to deployable conditions rather than idealized offline estimates.

Evaluation proceeds in three stages, consistent with the TERA framework. First, predictive performance is measured to establish baseline detection effectiveness. Second, operational metrics—including inference latency, peak memory usage, and model footprint—are recorded to characterize deployment cost. Third, constraint-aware filtering is applied using deployment parameters $(\lambda, \rho, \epsilon)$ to identify the feasible set of models.

Within the feasible set, models are compared using multi-dimensional performance analysis, including Pareto dominance, to eliminate strictly inferior candidates. When deployment preferences are specified, selection is performed using the TERA scoring function to identify the most suitable model under the given constraints.

To ensure fairness and reproducibility, all models are evaluated under identical experimental conditions, with no model-specific optimization or runtime adaptation. Measurements are repeated across multiple runs, and reported results correspond to mean values to mitigate stochastic variability.

This protocol ensures that evaluation outcomes are directly aligned with deployment requirements, enabling structured comparison among models with comparable predictive performance while preserving detection effectiveness as a primary constraint.

5. Results and Analysis

This section presents the empirical evaluation of the proposed framework under deployment-aligned conditions. The analysis follows the structure of the research questions, progressing from predictive performance comparison to operational cost analysis and constraint-aware model selection.

Rather than using accuracy as a sole ranking criterion, the evaluation treats predictive performance as a baseline reference for establishing feasibility. The primary objective is to examine how models with comparable detection effectiveness differ in terms of latency, resource usage, and deployment suitability.

5.1. Baseline Detection Performance

The purpose of this analysis is not to establish a final ranking, but to identify models with comparable detection performance that serve as the basis for subsequent constraint-aware evaluation. To this end, baseline classification results are summarized in Table 4, providing an accuracy-centric reference prior to incorporating deployment constraints.

The results indicate consistently high detection performance across model families, with several configurations achieving weighted F1-scores above 0.95. Notably, classical models combined with TF-IDF features perform comparably to transformer-based approaches, suggesting limited marginal gains from increased representational complexity under the evaluated datasets. This observation implies that classification performance has reached a near-saturation point under these conditions.

Table 4. Baseline classification performance of evaluated models.

Feature	Model	Accuracy	Precision	Recall	F1-Score
TF-IDF	LightGBM	0.9681	0.9685	0.9681	0.9682
TF-IDF	XGBoost	0.9598	0.9601	0.9598	0.9599
MPNet	MLP	0.9560	0.9565	0.9560	0.9561
TF-IDF	MLP	0.9438	0.9450	0.9438	0.9442
MPNet	LightGBM	0.9238	0.9256	0.9238	0.9241
MiniLM-L12	MLP	0.9194	0.9205	0.9194	0.9198
MPNet	XGBoost	0.9181	0.9194	0.9181	0.9180
MiniLM-L6	MLP	0.9006	0.9024	0.9006	0.9012
MiniLM-L12	LightGBM	0.8885	0.8898	0.8885	0.8883
MiniLM-L12	XGBoost	0.8783	0.8791	0.8783	0.8772

To assess whether these observed differences are statistically meaningful (RQ1), statistical validation was conducted under a paired evaluation setting with a significance level of $\alpha = 0.05$. The null hypothesis (H_0) assumes no statistically significant difference in predictive performance between models evaluated on the same test instances.

An iterative filtering process based on multivariate analysis of variance (MANOVA) was applied to remove structurally different, low-performing models. The initial model set exhibited statistically significant differences ($p < 0.05$), while the filtered subset satisfied $p \geq 0.05$, indicating statistical homogeneity among the remaining models.

To further validate pairwise differences, a bootstrap-based analysis was conducted to estimate confidence intervals of F1-score differences between representative models as shown in Table 5.

Table 5. Pairwise statistical comparison using bootstrap confidence intervals.

Model Pair	Metric	Δ	95% CI
TF-IDF + LightGBM vs. MPNet + MLP	F1	0.0121	[−0.0065, 0.0182]
TF-IDF + LightGBM vs. TF-IDF + XGBoost	F1	0.0083	[−0.0051, 0.0157]
MPNet + MLP vs. TF-IDF + XGBoost	F1	−0.0038	[−0.0102, 0.0069]

All confidence intervals include zero, indicating that performance differences among top-performing models are statistically insignificant. These findings are consistent with the MANOVA-based validation and confirm that classical and transformer-based models achieve comparable detection effectiveness under controlled experimental conditions.

Overall, the results establish the existence of a performance-homogeneous region, where models exhibit statistically indistinguishable performance. This finding directly answers RQ1 and motivates the transition from accuracy-centric evaluation to deployment-aware analysis in subsequent sections.

5.2. Deployment Cost and Trade-Off Analysis

Given statistically comparable performance, we examine operational cost as a determining factor for deployment feasibility (RQ2). Figure 4 presents inference latency characteristics across evaluated models, including both mean and tail (P95) latency.

The results reveal substantial variation in latency across model architectures. Lightweight classifiers consistently achieve low latency with stable tail behavior, whereas ensemble-based methods exhibit significantly higher latency and greater variability. Notably, classifier architecture exerts a stronger influence on latency than feature representation.

To further assess deployment feasibility, resource consumption is analyzed in terms of peak memory usage and model footprint, as shown in Figure 5.

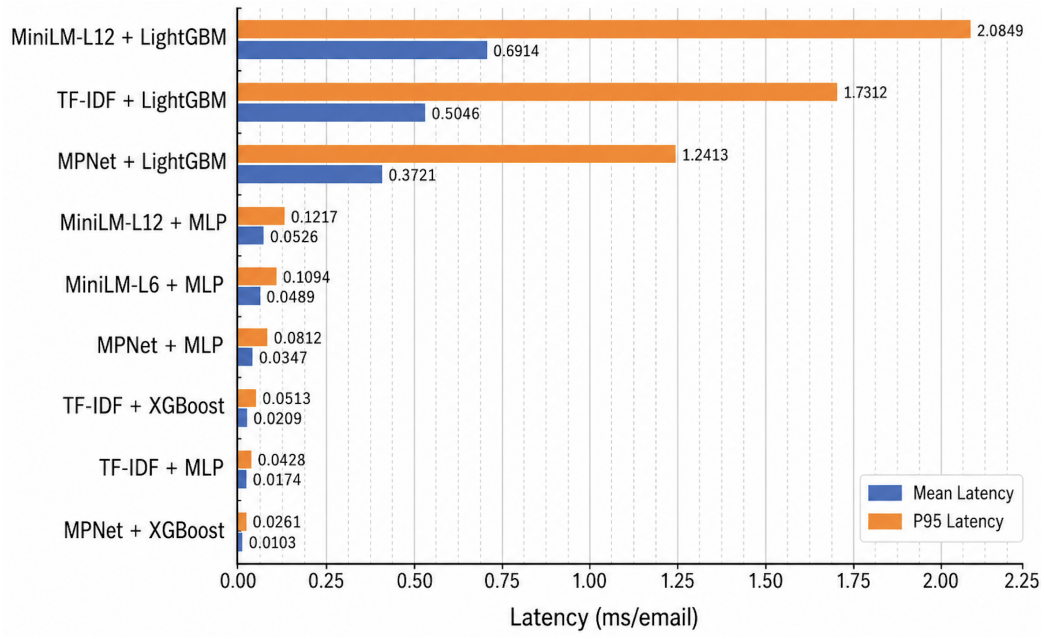

Figure 4. Comparison of mean and P95 inference latency across evaluated models under edge deployment constraints. The divergence between mean and tail latency highlights variability in real-time performance.

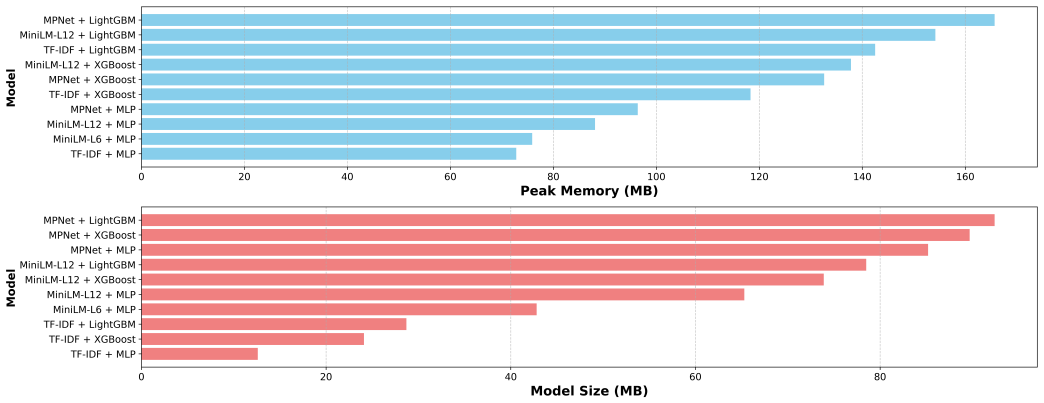

Figure 5. Comparison of peak memory consumption and serialized model size across evaluated models under edge deployment constraints.

The results indicate that resource usage varies by more than an order of magnitude across models. Lightweight configurations exhibit minimal memory consumption and compact model sizes, whereas ensemble-based models incur significantly higher overhead. While transformer-based embeddings introduce additional cost, their overall resource footprint depends strongly on the downstream classifier.

When considered jointly, latency and resource usage reveal a multi-dimensional trade-off space in which models exhibit fundamentally different operational profiles. Some configurations achieve low latency but higher memory usage, while others provide compact deployment footprints at the cost of increased inference time.

These observations indicate a regime shift in model selection. Once detection performance reaches a high threshold, further improvements yield diminishing returns while incurring disproportionately higher computational cost. Consequently, operational metrics become the dominant constraint governing deployment feasibility.

This analysis directly addresses RQ2 by demonstrating that incorporating latency and resource constraints fundamentally alters model prioritization compared to accuracy-only

evaluation, thereby motivating the constraint-aware selection framework introduced in the following subsection.

5.3. TERA-Based Model Selection and Comparison

To enable deployment-oriented model selection, the TERA framework formulates evaluation as a constraint-aware decision problem in which predictive performance defines feasibility, while operational characteristics guide selection among admissible candidates.

Based on the feasibility conditions defined in Section 3, Pareto analysis is applied to identify a set of non-dominated models that simultaneously balance detection effectiveness and operational cost. The resulting trade-off space is summarized in Table 6, which integrates predictive performance (Acc and F1-score) with system-level constraints, including tail latency (P95), peak memory usage, and model footprint.

Table 6. Trade-off analysis of Pareto-optimal models under the extended TERA decision space, integrating predictive performance and operational constraints.

Feature	Model	Acc ↑	F1 ↑	P95 Latency (s) ↓	Peak Memory (MB) ↓	Model Size (MB) ↓	Score ↑	Deployment Insight
TF-IDF	XGB	0.9615	0.9599	0.0365	0.0103	1.74	0.9130	Best overall trade-off under deployment constraints
MPNet	MLP	0.9580	0.9561	0.0513	0.1575	1.51	0.8710	Balanced alternative with moderate resource requirements
TF-IDF	MLP	0.9465	0.9442	0.0189	0.3138	1.95	0.8520	Efficient in latency but with reduced detection performance
MPNet	XGB	0.9203	0.9180	0.0108	0.0102	2.79	0.8420	Highly resource-efficient but lowest detection effectiveness
TF-IDF	LGBM	0.9701	0.9682	1.7196	22.01	4.91	0.5000	Highest predictive performance but dominated by latency and memory cost

As shown in Table 6, multiple models remain competitive in terms of detection effectiveness, yet exhibit substantial variation in operational characteristics. For example, TF-IDF with LightGBM achieves the highest predictive performance but incurs significantly higher latency and memory consumption, limiting its suitability under strict deployment constraints. In contrast, TF-IDF with XGBoost provides a more balanced trade-off, achieving near-equivalent predictive performance with substantially lower latency and resource usage, resulting in the highest composite score.

This observation highlights a key distinction between predictive equivalence and deployment equivalence. Models that are statistically comparable in accuracy are not necessarily interchangeable in practice, as operational constraints impose additional requirements that must be satisfied. Within the TERA framework, such constraints are explicitly enforced prior to comparison, ensuring that only deployable models are considered in the decision process.

Unlike conventional accuracy-based ranking, which produces a single global ordering, TERA yields a set of feasible candidates whose relative suitability depends on deployment priorities. As a result, model selection is no longer determined solely by marginal differences in accuracy, but by a structured evaluation of trade-offs between effectiveness and efficiency.

To further contextualize these results, we compare TERA with representative baseline strategies, including accuracy-only, latency-only, and cost-sensitive approaches. Accuracy-only selection prioritizes predictive performance at the expense of substantial resource overhead, often resulting in infeasible deployment configurations. Conversely, latency-only selection minimizes operational cost but significantly degrades detection effectiveness, limiting its practical utility. Cost-sensitive approaches provide a partial balance between performance and efficiency; however, their effectiveness remains dependent on manually tuned weighting parameters and does not guarantee feasibility across deployment scenarios.

In contrast, TERA explicitly enforces feasibility constraints prior to model selection, ensuring that all selected models satisfy deployment requirements while maintaining

competitive predictive performance. This constraint-aware formulation eliminates inconsistencies between ranking-based selection and real-world deployment constraints.

These findings confirm that model selection is inherently a multi-objective decision problem. By preserving the multi-dimensional structure of the evaluation space and incorporating feasibility constraints, TERA produces consistent, deployment-ready decisions that are not achievable through conventional single-objective or weighted approaches.

Overall, as evidenced by the trade-off analysis in Table 6, this approach directly addresses RQ3 by demonstrating that the TERA framework improves both the consistency and practical relevance of model selection under realistic deployment conditions.

5.4. Robustness and Sensitivity Analysis

To assess robustness, the TERA framework was extended to a four-dimensional decision space incorporating P95 latency and model footprint. Overall, the composition of the Pareto set remains largely consistent with the baseline formulation, with differences primarily confined to boundary cases involving models with disproportionately high resource consumption. Notably, the core set of deployment-recommended models remains stable across both configurations, indicating robustness to constraint refinement.

In addition to constraint extension, sensitivity analysis was conducted by varying the weighting parameters within the decision function. The results indicate that moderate variations in weights primarily affect ranking within the feasible set, while having minimal impact on the composition of the set itself. This suggests that model admissibility is determined predominantly by explicit feasibility constraints rather than by weight tuning.

This behavior is consistent with the interpretability of the weighting scheme, where weights function as preference indicators rather than strict optimization parameters. Consequently, the framework supports flexible, context-dependent selection without introducing instability due to parameter sensitivity.

Overall, these findings demonstrate that the TERA framework provides stable and reliable model selection under varying deployment conditions. By preserving feasibility constraints while allowing interpretable preference-based differentiation, TERA maintains consistent selection outcomes and reinforces its practical applicability in real-world deployment scenarios.

Model selection is inherently dependent on the deployment platform, as computational constraints such as latency and resource availability vary significantly across environments. To evaluate the robustness of the proposed framework, we compare model performance across three representative scenarios: a standard CPU environment (Colab), an edge CPU device (Raspberry Pi 5), and a resource-constrained edge GPU platform (Jetson Nano).

Figure 6 presents the corresponding S_{TERA} scores. The results show that XGBoost-based configurations consistently achieve the highest scores across all platforms, indicating strong robustness to varying deployment constraints. In contrast, MLP-based models exhibit performance degradation under edge conditions, particularly on Raspberry Pi 5, due to increased latency penalties.

Figure 6 further illustrates the shift in model ranking across platforms. While TF-IDF + MLP performs competitively in the Colab environment, its relative performance declines on edge devices. Conversely, MPNet + XGBoost becomes more competitive under constrained conditions, highlighting the impact of latency-sensitive evaluation.

These findings demonstrate that model ranking is not static but varies with deployment constraints. The proposed framework effectively adapts to different environments by selecting models that balance predictive performance with operational feasibility. This reinforces the necessity of platform-aware evaluation for real-world deployment scenarios.

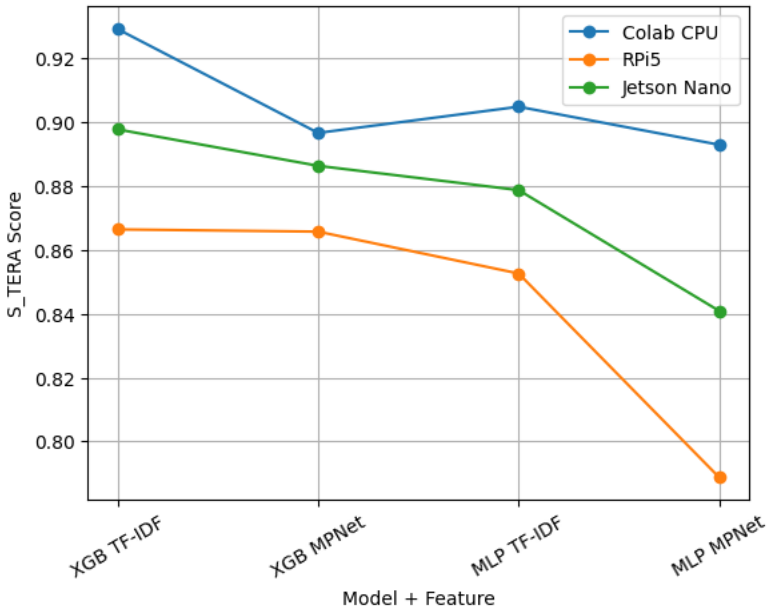

Figure 6. Platform-aware model ranking based on S_{TERA} scores.

5.5. Discussion and Deployment Implications

The results demonstrate that accuracy-centric evaluation alone is insufficient for deployment-oriented decision-making, as models with comparable detection performance can differ substantially in latency and resource consumption, resulting in significant variation in deployment feasibility.

From a system perspective, model selection must align with operational constraints. Lightweight models are suitable for CPU-based edge environments, whereas transformer-based models typically require GPU-enabled infrastructure. In practical settings, such systems operate under strict latency (e.g., sub-second per email) and memory constraints, reinforcing the need for constraint-aware selection despite similar predictive effectiveness.

The findings further indicate that once models reach a performance-homogeneous region, marginal accuracy gains offer limited practical value, while operational cost becomes the dominant factor in deployment decisions. This underscores the importance of constraint-aware evaluation strategies.

The proposed framework reflects real-world deployment conditions by integrating latency and memory constraints into the evaluation process, enabling practitioners to select models that satisfy operational requirements. It is directly applicable to systems such as email gateways and edge-based filtering platforms, and can be implemented following Algorithm 1 within a unified pipeline consisting of measurement, constraint filtering, and trade-off-based selection.

Although the evaluation is conducted on phishing detection datasets, the formulation is model-agnostic and extensible to other real-time cybersecurity tasks. This study focuses on simulation-based validation rather than live deployment; real-world implementation remains future work, including production-level validation under operational workloads. Evolving threats such as AI-generated phishing are also not explicitly addressed.

Overall, the results reinforce that model selection should be framed as a constraint-aware decision process rather than a purely accuracy-driven ranking task. The proposed TERA framework provides a practical and extensible approach for deployment-aware model selection.

6. Discussion

This section interprets the empirical findings in relation to the research questions (RQ1–RQ3) and examines their implications for deployment-aware model evaluation. Rather than reiterating established limitations of accuracy-centric assessment, the discussion focuses on how the observed results inform model selection under operational constraints.

Importantly, the proposed framework does not advocate reducing detection accuracy in favor of efficiency. Instead, it reflects a practical scenario in which multiple models already achieve comparable performance, necessitating deployment decisions under system-level constraints. In this context, TERA complements accuracy-driven research by introducing a structured decision layer, rather than redefining model quality or shifting the primary objective away from detection effectiveness. Rather than redefining what constitutes a good model, TERA provides a principled mechanism for selecting among equally strong models under real-world constraints.

6.1. From Predictive Equivalence to Deployment Differentiation

The results show that multiple models achieve statistically comparable detection performance (RQ1), forming a region of predictive equivalence under the evaluated datasets. Within this region, accuracy-based ranking provides limited discrimination among candidate models.

However, substantial differences are observed in inference latency and resource usage (RQ2), with variations exceeding an order of magnitude across models. This indicates that predictive equivalence does not imply deployment equivalence. Models with similar detection effectiveness may differ significantly in their ability to operate under real-time and resource-constrained conditions.

These findings suggest that, once a sufficient level of detection performance is achieved, operational characteristics become the primary determinants of deployment feasibility. As such, evaluation approaches that rely solely on predictive metrics may overlook practically viable alternatives.

The results reveal that models with comparable predictive performance can exhibit substantial differences in latency and resource consumption, highlighting a phenomenon of performance saturation. In such cases, accuracy alone becomes insufficient as a selection criterion, and operational characteristics emerge as the dominant factors influencing deployment decisions.

This observation underscores the importance of separating evaluation from decision-making. While traditional evaluation frameworks emphasize predictive performance, the findings suggest that deployment-aware considerations must be explicitly incorporated into the model selection process.

6.2. Implications for Constraint-Aware Model Selection

The empirical results further demonstrate that model selection is inherently a constrained decision problem (RQ3). When multiple models exhibit comparable predictive performance, selecting a model based solely on accuracy may lead to choices that are infeasible under deployment constraints.

The TERA framework addresses this issue by explicitly separating feasibility from preference. By enforcing constraints on latency and resource usage prior to comparison, the framework restricts the candidate space to deployable models. Within this feasible set, multi-dimensional analysis—such as Pareto dominance—preserves trade-off relationships without reducing them to a single scalar objective.

Importantly, the results show that Pareto-optimal models do not necessarily align with accuracy-based rankings. This highlights a systematic gap between conventional evaluation practices and deployment requirements. By structuring model selection as a constraint-aware process, TERA enables consistent identification of models that maintain competitive detection performance while satisfying operational limits.

6.3. Revisiting Model Complexity and Representation Trade-Offs

The findings also provide insight into the relationship between model complexity and practical performance. Classical machine learning models with sparse representations achieve detection effectiveness comparable to transformer-based approaches under the evaluated datasets.

Transformer-based models offer advantages in capturing semantic and contextual patterns, which are particularly relevant for sophisticated phishing attacks. However, these benefits are accompanied by increased computational cost. The results indicate that higher representational capacity does not automatically translate into superior deployment performance.

Within the TERA framework, transformer-based approaches remain viable when integrated with efficient classification strategies, suggesting that model design should emphasize balanced trade-offs between expressiveness and efficiency rather than maximizing complexity alone.

6.4. Generalization and Broader Implications

Although this study focuses on phishing detection, the observed patterns are not domain-specific. The distinction between predictive equivalence and deployment feasibility is likely to arise in other real-time cybersecurity applications, such as intrusion detection and anomaly detection systems.

More broadly, the results suggest that evaluation frameworks should explicitly incorporate system-level constraints when transitioning from benchmarking to deployment. The TERA formulation provides a structured approach for bridging this gap by aligning experimental evaluation with operational requirements while preserving the role of predictive performance as a feasibility condition.

6.5. Limitations and Future Work

Several limitations should be considered when interpreting these results.

First, the evaluation is conducted on benchmark datasets that do not explicitly include AI-generated phishing content. As a result, conclusions regarding robustness against emerging attack strategies remain limited. Future work should incorporate datasets reflecting modern adversarial techniques.

Second, the experiments are performed in a controlled edge-oriented environment. While relative comparisons remain valid, absolute performance values may vary across hardware configurations. Extending the evaluation to heterogeneous platforms would strengthen external validity.

Third, the analysis focuses on latency, peak memory usage, and model size as primary deployment constraints. Additional factors, such as energy consumption, adversarial robustness, and long-term adaptability, may further influence deployment decisions and warrant investigation.

Finally, while statistical validation supports the identification of performance-equivalent models, the study prioritizes practical significance over exhaustive statistical testing. This reflects real-world decision-making contexts, where operational constraints often outweigh marginal statistical differences.

6.6. Implications for Deployment-Aware Evaluation

The findings highlight the importance of aligning model selection with deployment constraints rather than relying solely on predictive performance. Lightweight models are well-suited for resource-constrained environments, while more complex architectures may be appropriate in settings with greater computational capacity.

Overall, the TERA framework provides a structured and reproducible mechanism for deployment-aware model selection. By integrating constraint-aware evaluation with multi-dimensional analysis, it enables transparent and defensible decision-making in scenarios where multiple models achieve comparable detection effectiveness.

6.7. Ablation Perspective on Evaluation Dimensions

To further understand the role of each evaluation dimension, we interpret the results from an ablation perspective by considering how model selection behavior changes when individual factors are omitted.

When only predictive performance is considered, multiple models fall within a region of statistical equivalence, resulting in limited discrimination among candidates. In this setting, model selection effectively reduces to arbitrary preference or marginal ranking differences, which may not reflect deployment feasibility.

When operational constraints such as latency and resource usage are introduced as filtering conditions, the candidate set is significantly reduced. Several models that are competitive in terms of detection effectiveness are excluded due to infeasibility under deployment constraints. This demonstrates that feasibility considerations alone can substantially alter the selection outcome.

Finally, when both predictive performance and operational characteristics are jointly considered within the TERA framework, model selection becomes a structured decision process. Predictive performance defines the feasible region, while latency and resource usage differentiate among admissible models. This combined perspective enables consistent identification of deployment-appropriate models without relying on scalar aggregation.

These observations indicate that each evaluation dimension plays a distinct and complementary role. Predictive performance ensures baseline detection capability, while operational metrics determine practical deployability. Removing either component leads to incomplete or potentially misleading selection outcomes, reinforcing the need for integrated, multi-dimensional evaluation.

7. Conclusions

This paper presented TERA, a constraint-aware framework for deployment-oriented evaluation and model selection in phishing detection. The framework addresses a key limitation of accuracy-centric evaluation by formulating model selection as a constrained decision problem that integrates detection effectiveness with operational factors such as inference latency and resource usage.

The empirical results show that multiple models achieve comparable detection performance, forming a region of predictive equivalence in which accuracy alone provides limited discrimination. Within this region, substantial differences in latency and resource usage emerge, indicating that predictive equivalence does not imply deployment equivalence. These findings highlight that model suitability must be evaluated in relation to system-level constraints rather than predictive performance alone.

To address this challenge, TERA introduces a structured evaluation process that separates feasibility from preference. By enforcing deployment constraints prior to comparison and preserving the multi-dimensional structure of performance, the framework enables consistent identification of models that are both effective and deployable. This formulation

provides a principled mechanism for aligning experimental evaluation with real-world deployment requirements.

The proposed framework provides practical value in deployment scenarios such as real-time phishing detection and edge-based systems, where selecting a feasible and efficient model is critical. By integrating predictive and operational metrics into a structured decision process, TERA enables more reliable and context-aware model selection in real-world environments.

Beyond phishing detection, the results suggest a broader implication for cybersecurity model evaluation. In real-time and resource-constrained settings, evaluation should move from accuracy-based ranking toward constraint-aware decision-making, where operational feasibility plays a central role alongside predictive performance.

Several limitations should be noted. The evaluation is conducted on benchmark datasets that do not explicitly include AI-generated phishing content, limiting conclusions regarding emerging attack scenarios. In addition, the observed trade-offs depend on the evaluated models and execution environment, and may vary across different deployment contexts.

Future work will extend this framework to incorporate additional operational dimensions, including energy consumption and adversarial robustness, as well as validation across diverse datasets and hardware platforms. These directions aim to further strengthen the applicability of deployment-aware evaluation in evolving cybersecurity systems.

Author Contributions: Conceptualization, C.J. and J.P.; methodology, C.J. and J.P.; software, C.J.; formal analysis, C.J., J.P. and P.K.; investigation, C.J.; resources, C.J.; data curation, C.J.; visualization, P.K.; writing—original draft preparation, C.J.; writing—review and editing, J.P., P.K. and M.F.Z.; supervision, J.P.; project administration, J.P.; funding acquisition, C.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The authors make available all experimental artifacts, including datasets, logs, and model predictions, at https://github.com/cjundang/Research-TERA (accessed on 31 April 2026) to support reproducibility and transparency.

Acknowledgments: During the preparation of this manuscript, the authors used SciSpace (https://scispace.com/) for literature search assistance and ChatGPT (OpenAI, GPT-5.2) for language refinement and grammatical improvement, and the final manuscript was reviewed and revised by the authors, who take full responsibility for its content.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bhowmick, A.; Hazarika, S.M. E-Mail Spam Filtering: A Review of Techniques and Trends. In *Advances in Electronics, Communication and Computing*; Kalam, A., Das, S., Sharma, K., Eds.; Lecture Notes in Electrical Engineering; Springer: Singapore, 2018; Volume 443, pp. 583–590. [CrossRef]
2. Kumar Birthriya, S.; Jain, A.K. A comprehensive survey of phishing email detection and protection techniques. *Inf. Secur. J. A Glob. Perspect.* **2022**, *31*, 411–440. [CrossRef]
3. Patra, C.; Giri, D.; Nandi, S.; Das, A.K.; Alenazi, M.J. Phishing email detection using vector similarity search leveraging transformer-based word embedding. *Comput. Electr. Eng.* **2025**, *124*, 110403. [CrossRef]
4. Verizon. *2023 Data Breach Investigations Report*; Technical Report; Verizon: Basking Ridge, NJ, USA, 2023.
5. Hasanov, I.; Virtanen, S.; Hakkala, A.; Isoaho, J. Application of Large Language Models in Cybersecurity: A Systematic Literature Review. *IEEE Access* **2024**, *12*, 176751–176778. [CrossRef]

6. Tooher, P.; Lallie, H.S. A Two-Stage Deep Learning Framework for AI-Driven Phishing Email Detection Based on Persuasion Principles. *Computers* **2025**, *14*, 523. [CrossRef]
7. Salloum, S.A.; Gaber, T.; Vadera, S.; Shaalan, K. A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques. *IEEE Access* **2022**, *10*, 65703–65727. [CrossRef]
8. Hosseinzadeh, M.; Ali, U.; Ali, S.; Abbaszadi, R.; Gharehchopogh, F.S. Improving phishing email detection performance through deep learning with adaptive optimization. *Sci. Rep.* **2025**, *15*, 20668. [CrossRef] [PubMed]
9. Chakraborty, P.; Arumugam, K.K.; Alfadel, M.; Nagappan, M.; McIntosh, S. Revisiting the Performance of Deep Learning-Based Vulnerability Detection on Realistic Datasets. *IEEE Trans. Softw. Eng.* **2024**, *50*, 2163–2177. [CrossRef]
10. Sharma, S.; Kumar, V.; Dutta, K. Multi-objective optimization algorithms for intrusion detection in IoT networks: A systematic review. *Internet Things Cyber-Phys. Syst.* **2024**, *4*, 258–267. [CrossRef]
11. Abedin, N.F.; Bawm, R.; Sarwar, T.; Saifuddin, M.; Rahman, M.A.; Hossain, S. Phishing Attack Detection using Machine Learning Classification Techniques. In *Proceedings of the 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*; IEEE: New York City, NY, USA, 2020; pp. 1125–1130. [CrossRef]
12. Brioua, H.; Siyambaş, H.; Şahin, D.Ö. Phishing E-mail Detection with Machine Learning and Deep Learning: Improving Classification Performance with Proposed New Features. *Bulg. J. Electr. Comput. Eng.* **2025**, *13*, 183–193. [CrossRef]
13. Altwaijry, N.; Al-Turaiki, I.; Alotaibi, R.; Alakeel, F. Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models. *Sensors* **2024**, *24*, 2077. [CrossRef] [PubMed]
14. Arshad, J.; Azad, M.A.; Abdeltaif, M.M.; Salah, K. An intrusion detection framework for energy constrained IoT devices. *Mech. Syst. Signal Process.* **2020**, *136*, 106436. [CrossRef]
15. Maseer, Z.K.; Robiah, Y.; Bahaman, N.; Mostafa, S.A. Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. *IEEE Access* **2021**, *9*, 3056614. [CrossRef]
16. Sommer, R.; Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*; IEEE: New York City, NY, USA, 2010; pp. 305–316. [CrossRef]
17. Sfaxi, H.; Lahyani, I.; Yangui, S.; Torjmen, M. Latency-Aware and Proactive Service Placement for Edge Computing. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 4243–4254. [CrossRef]
18. Noor, K.; Imoize, A.L.; Li, C.T.; Weng, C.Y. A Review of Machine Learning and Transfer Learning Strategies for Intrusion Detection Systems in 5G and Beyond. *Mathematics* **2025**, *13*, 1088. [CrossRef]
19. Otieno, D.O.; Siami Namin, A.; Jones, K.S. The Application of the BERT Transformer Model for Phishing Email Classification. In *Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*; IEEE: New York City, NY, USA, 2023; pp. 1303–1310. [CrossRef]
20. Ayodele, T.O. Impact of AI-Generated Phishing Attacks: A New Cybersecurity Threat. In *Intelligent Computing. CompCom 2025*; Arai, K., Ed.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2025; Volume 1424. [CrossRef]
21. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Modern Deep Learning Research. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 13693–13696. [CrossRef]
22. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. In Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, Vancouver, BC, Canada, 13 December 2019. [CrossRef]
23. Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. TinyBERT: Distilling BERT for Natural Language Understanding. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP*; IEEE: New York City, NY, USA, 2020. [CrossRef]
24. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*; IEEE: New York City, NY, USA, 2020. [CrossRef]
25. Milliken, M.; Bi, Y.; Galway, L.; Hawe, G. Multi-objective optimization of base classifiers in StackingC by NSGA-II for intrusion detection. In *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI)*; IEEE: New York City, NY, USA, 2016; pp. 1–8. [CrossRef]
26. Telikani, A.; Rudbardeh, N.E.; Soleymanpour, S.; Shahbahrami, A.; Shen, J.; Gaydadjiev, G.; Hassanpour, R. A Cost-Sensitive Machine Learning Model with Multitask Learning for Intrusion Detection in IoT. *IEEE Trans. Ind. Inform.* **2024**, *20*, 3880–3890. [CrossRef]
27. Abu Al-Haija, Q.; Al-Fayoumi, M. An intelligent identification and classification system for malicious uniform resource locators (URLs). *Neural Comput. Appl.* **2023**, *35*, 16995–17011. [CrossRef] [PubMed]
28. Abu Al-Haija, Q.; Al Badawi, A. URL-based Phishing Websites Detection via Machine Learning. In *Proceedings of the 2021 International Conference on Data Analytics for Business and Industry (ICDABI)*; IEEE: New York City, NY, USA, 2021; pp. 644–649. [CrossRef]

29. Al-Fayoumi, M.; Alhijawi, B.; Abu Al-Haija, Q.; Armoush, R. XAI-PhD: Fortifying Trust of Phishing URL Detection Empowered by Shapley Additive Explanations. *Int. J. Online Biomed. Eng. (iJOE)* **2024**, *20*, 80–101. [CrossRef]
30. Elqasass, A.; Aljundi, I.; Al-Fayoumi, M.; Abu Al-Haija, Q. Facilitating Secure Web Browsing by Utilizing Supervised Filtration of Malicious URLs. In *IoT Based Control Networks and Intelligent Systems*; Joby, P.P., Alencar, M.S., Falkowski-Gilski, P., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2024; Volume 789. [CrossRef]
31. Abu-Nimeh, S.; Nappa, D.; Wang, X.; Nair, S. A Comparison of Machine Learning Techniques for Phishing Detection. In Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit, Pittsburgh, PA, USA, 4–5 October 2007; pp. 60–69. [CrossRef]
32. Al-Subaiey, A.; Al-Thani, M.; Alam, N.A.; Antora, K.F.; Khandakar, A.; Zaman, S.A.U. Novel Interpretable and Robust Web-based AI Platform for Phishing Email Detection. *arXiv* **2024**, arXiv:2405.11619. [CrossRef]
33. Raschka, S.; Mirjalili, V. *Machine Learning and Deep Learning with Python*; Packt Publishing: Birmingham, UK, 2018.
34. Probst, P.; Wright, M.N.; Boulesteix, A.L. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *J. Mach. Learn. Res.* **2019**, *20*, 1–32.
35. Olson, R.S.; Bartley, N.; Urbanowicz, R.J.; Moore, J.H. Evaluation of a tree-based pipeline optimization tool for automating data science. In Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, USA, 20–24 July 2016; pp. 485–492.