ISNNI: 3007-3391

HEC Category: Y

Journal of Computers and Intelligent Systems

Vol. 3, No. 4, 2025, pages 319 - 332

Journal Homepage:

https://journals.iub.edu.pk/index.php/JCIS/

PHANTOM-DGA: Multi-Task Transformer With Cached Phonotactics, Dictionary Segmentation, and Meta-Learning for Robust DGA Domain Detection

Muhammad Asshad [1]

[1] Faculty of Computer Studies, Arab open university, Oman.

ARTICLE INFO

Article History:

Received:	November 05, 2025
Revised:	November 25, 2025
Accepted:	December 10, 2025
Available Online:	December 25, 2025

Keywords:

Domain generation algorithm (DGA), Malicious Domain Detection, Transformer Encoder, Self-Supervised Pretraining, Contrastive Learning, Meta-Learning, Phonotactics, Calibration.

Classification Codes:

Funding:

This research received no specific grant from any funding agency in the public or not-for-profit sector.

ABSTRACT

Domain generation algorithms (DGAs) enable malware to evade static blacklists by generating large volumes of pseudo-random candidate domains. While many published detectors achieve near-perfect performance under random i.i.d. splits, performance often degrades under temporal shift and when evaluating on previously unseen DGA families. This paper presents PHANTOM-DGA, a GPU-optimized, checkpointable training pipeline and feature-gated multi-task transformer that combines byte-level sequence modeling with lightweight side features: (i) classical lexical statistics, (ii) dictionary-based segmentation features, and (iii) phonotactic features learned via an n-gram consonant–vowel language model trained only on benign training data to avoid leakage. PHANTOM-DGA also supports self-supervised pretraining (masked modeling + contrastive alignment), and optional Reptile-style meta-learning to improve robustness in a held-out-family evaluation setting. Experiments on the public 'Domain Generation Algorithm' dataset (≈1.82M unique canonicalized domains; 52 DGA families) evaluate three protocols: random split, time-forward split, and unseen-family split. Under the random protocol, PHANTOM-DGA reaches ROC-AUC 0.9986 and F1 0.9916. Under the time-forward protocol, ROC-AUC decreases to 0.9697–0.9674 depending on ablations. Under the unseen-family protocol, the best ROC-AUC is 0.9359 with reduced false positives at 95% TPR. The results highlight that protocol choice dominates headline performance and that robustness gains require explicit distribution-shift evaluation and training strategies.

Corresponding Author's Email:

Citation:

1. Introduction

The Domain Name System (DNS) is a critical dependency for almost all Internet services. Malware operators frequently abuse DNS to locate command-and-control (C2) infrastructure, exfiltrate data, and coordinate botnets. A common evasion technique is the domain generation algorithm (DGA), which deterministically produces large sets of candidate domains from time- or seed-dependent inputs. Only a small subset of generated domains are registered by the attacker, but the high churn makes static blacklisting brittle and expensive to maintain. Early measurement work highlighted how DGA-driven traffic can be observed through characteristic DNS patterns and NXDOMAIN responses [1].

A large body of research has developed detectors for DGA domains using handcrafted lexical features and supervised learning [2], as well as deep learning models that treat domain strings as sequences of characters or bytes [3], [4]. However, the evaluation protocol has a large impact on conclusions. Random i.i.d. splits can overestimate performance because test samples share the same family distribution and time period as the training set. In contrast, time-forward splits and held-out family splits better approximate realistic deployments, where domain distributions drift and previously unseen malware families appear. Recent work has emphasized that 'easy' benchmarks can mask systematic weaknesses and that robust evaluation must explicitly model distribution shift and adversarial behavior [18].

This paper presents PHANTOM-DGA, a practical research prototype designed to (i) expose the robustness gap between random and shift-aware protocols, and (ii) explore a set of complementary modeling strategies that are cheap to compute at inference time. PHANTOM-DGA combines a byte-level transformer encoder [21] with a gated fusion mechanism that integrates classical lexical statistics, dictionary segmentation cues, and phonotactic regularities. The training pipeline is engineered for large-scale datasets in resource-constrained notebook environments through persistent memory-mapped caches, GPU-friendly data loading, mixed precision training [29], and checkpoint/resume support.

The main contributions of this paper are as follows: (1) a feature-gated multi-task transformer that jointly predicts maliciousness, DGA family identity (for known families), and coarse-grained time buckets, with uncertainty-based loss weighting; (2) a protocol-specific phonotactic feature extractor using an n-gram consonant–vowel language model trained on benign training data only to prevent leakage; (3) optional self-supervised pretraining that combines masked byte modeling (inspired by BERT-style objectives [22]) with contrastive representation alignment [23], [24]; (4) optional Reptile-style meta-learning [26] for the unseen-family setting; and (5) an empirical analysis on a public DGA dataset under three evaluation protocols with ablations that isolate each component.

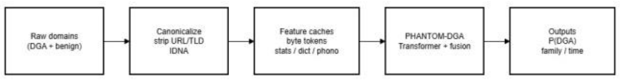

Fig. 1. *PHANTOM-DGA processing and training pipeline.*

2. BACKGROUND AND PROBLEM FORMULATION

A DGA maps an input seed (e.g., date, PRNG state, bot identifier, or malware configuration) to a set of candidate domain names. Bots iterate through the set until they reach an active C2 server. From the defender's perspective, DGA domains often exhibit unusual lexical characteristics compared to benign domains: higher character entropy, atypical vowel–consonant patterns, and lower alignment with natural language. Nevertheless, adversaries can imitate benign lexical statistics and exploit weaknesses of feature-based classifiers, as demonstrated by evasion methods such as CharBot [6] and later work on adversarial DGAs [7], [8].

Let x denote a canonicalized domain core string (e.g., 'example' from 'www.example.com'). We consider a binary classifier $f_\theta(x)\in[0,1]$ that estimates the probability of a domain being DGA-generated. In addition, we use auxiliary objectives to predict (i) a DGA family label for samples from families observed during training, and (ii) a discretized time-bucket label derived from observed timestamps. The auxiliary tasks are not assumed available at deployment for unknown families, but they can shape the learned representation and improve sample efficiency.

Robustness is assessed under three data partitioning protocols. (P1) Random: stratified i.i.d. split into train/validation/test. (P2) Time-forward: train on earlier timestamps and test on later timestamps, separately for benign and malicious samples. (P3) Unseen-family: hold out a subset of DGA families entirely for the test set, while benign samples follow a time-forward split. P2 and P3 approximate distribution shift and are strongly recommended for reporting realistic performance [18].

Protocol	**Split rule**	**Primary robustness factor**
Random	Stratified i.i.d. split	Measures in-distribution performance; often optimistic
Time-forward	Chronological split per class, merged	Evaluates temporal drift and concept shift
Unseen-family	Hold out DGA families for test	Evaluates generalization to novel malware families

Table I. *Evaluation protocols used in this study.*

3. Related Work

DGA detection has been studied for more than a decade, spanning network measurement, lexical analysis, and representation learning. Early operational systems relied on reputation, DNS graph structure, and behavioral signals. Notos proposed a dynamic reputation system for domains using multiple signals, while later systems such as Pleiades and Phoenix used DNS graph clustering and temporal features to identify DGA activity (surveyed in [2]).

Handcrafted lexical feature approaches model domains using character distributions, n-gram statistics, entropy, length, and other string properties. FANCI [2] formalized feature-based NXDOMAIN classification at scale and remains a strong baseline. Feature-based methods can be fast and interpretable, but they may require careful feature engineering, are sensitive to obfuscation, and can be brittle when adversaries adapt.

Deep learning methods reduced feature engineering by learning directly from raw strings. Woodbridge et al. demonstrated that recurrent neural networks can predict DGA families and detect DGAs from character sequences [3]. Character-level convolutional approaches such as eXpose [4] showed competitive performance for malicious strings in general (URLs, file paths, registry keys) and have been adapted for domain classification. Subsequent work explored hybrids of CNN/RNN architectures, attention mechanisms, and embedding strategies, often reporting strong ROC-AUC under random splits.

Transformer models have recently become attractive due to their parallelism and strong sequence modeling capabilities [21]. Several studies apply transformer-like encoders to DGA detection, including hybrid designs that combine transformers with additional modules or embeddings [10]–[17]. The key lesson across these studies is that evaluation under distribution shift is essential: performance under random splits can be near-perfect while time-forward and unseen-family performance can drop substantially, motivating training strategies that explicitly target robustness.

Adversarial DGAs and evasion techniques challenge detectors. DeepDGA introduced the idea of generating domains to fool detectors using adversarially tuned models, and CharBot proposed a simple but effective string-level perturbation method that degrades feature-based detectors [6]. More recent work proposes generative models for DGAs (e.g., DomainGAN [8]) and masked perturbation strategies (MaskDGA [7]) to study attack and defense under stronger threat models.

Complementary research considers deployment constraints and real-time detection. For example, programmable data planes have been used to implement DGA detection in P4 switches for high-speed networks [17]. Other work emphasizes reproducibility and meaningful benchmarks, highlighting pitfalls such as data leakage, unrepresentative benign sampling, and over-reliance on i.i.d. splits [18].

4. DATASET AND PREPROCESSING

We evaluate on the public 'Domain Generation Algorithm' dataset hosted on Kaggle, which provides a list of known DGA domains with family identifiers and timestamps, alongside a benign list derived from Alexa Top-1M domains and an English word list. After parsing and cleaning, we combine DGA and benign examples and apply canonicalization to obtain a 'domain core' string. Canonicalization removes URL scheme and paths, lowercases, applies IDNA/Punycode conversion for internationalized domains [30], collapses repeated separators, removes non-alphanumeric characters outside {a–z, 0–9, '.', '-', '_'}, and strips the top-level domain (TLD) to reduce trivial cues.

The resulting dataset contains 1,815,815 unique canonicalized domain cores, including 1,345,962 DGA samples across 52 families and 469,853 benign samples. The benign file lacks explicit timestamps in the provided dataset; to enable time-forward evaluation, we assign deterministic pseudo-timestamps to benign domains by mapping a stable hash of the domain string into the observed DGA timestamp range. This procedure avoids leakage across splits (the mapping is deterministic) but should be interpreted as an approximation of realistic benign temporal dynamics.

For sequence modeling, each domain core is encoded at the byte level. We prepend a special [CLS] token and pad/truncate to a fixed maximum length of 63 tokens (1 [CLS] + up to 62 bytes). Byte-level modeling avoids maintaining a character vocabulary and naturally handles digits and separators. In addition to the sequence input, we compute three groups of lightweight features: (i) classic lexical statistics (length, digit ratio, separator count, Shannon entropy), (ii) dictionary segmentation features (coverage by English words, number of segments, average segment length, residual length), and (iii) phonotactic features derived from a consonant–vowel stream and an n-gram language model trained on benign training data only.

Dataset attribute	**Value**
Total unique domains (canonicalized core)	1,815,815
DGA domains	1,345,962
Benign domains	469,853
Distinct DGA families	52
Max domain length used	63 characters (including CLS); 62 bytes payload
Dictionary size (words.txt)	416,255 words

Table II. *Dataset summary after preprocessing.*

Feature group	Type	Dim.	Description
Byte tokens	Sequence	63	CLS + bytes, padded
Classic statistics	Dense	4	length, digit ratio, separators, entropy
Dictionary segmentation	Dense	4	coverage, segments, avg len, residual
Phonotactics	Dense	6	ppl, log-prob, vowel ratio, alternation, longest consonant run, syllable estimate

Table III. *Input representations used by PHANTOM-DGA.*

5. PROPOSED METHOD: PHANTOM-DGA

PHANTOM-DGA integrates a byte-level transformer encoder with complementary side features and trains with a multi-task objective. This section details the feature construction, model components, and training objectives, matching the implementation shared in the accompanying notebook.

A. Byte-Level Transformer Encoder

Let $\mathbf{x} \in \{0,\dots,258\}^L$denote the tokenized input sequence of length L, including a prepended $[\mathrm{CLS}]$token, and let $\mathbf{m} \in \{0,1\}^L$denote the corresponding attention mask. Token and positional embeddings are summed and passed through a stack of Ntransformer encoder layers with multi-head self-attention [21]. The final hidden state associated with the $[\mathrm{CLS}]$token is used as the sequence-level representation, denoted by

$$\mathbf{h}_{\text{seq}} \in \mathbb{R}^d.$$

B. Side Features

Classic lexical statistics capture simple but informative cues such as length and character distribution. Dictionary segmentation features estimate the extent to which a domain can be decomposed into English words, which is often higher for benign domains. Phonotactic features are computed from the consonant–vowel stream $\phi(x)$ and an n-gram language model fit on benign training streams only. Using benign-only fitting prevents test leakage when protocols change.

C. Feature-Gated Fusion

Each feature group is first projected into a shared fusion space$\mathbb{R}^{d_f}$. Let

$$\mathbf{z}_{\text{seq}}, \mathbf{z}_{\text{ph}}, \mathbf{z}_{\text{di}}, \mathbf{z}_{\text{st}} \in \mathbb{R}^{d_f}$$

denote the projected representations of the sequential, phonotactic, dictionary, and statistical feature groups, respectively. These vectors are concatenated and passed through a multi-layer perceptron (MLP) followed by a softmax to produce feature gates,

$$\mathbf{g} = \mathrm{softmax}\Big(\mathrm{MLP}\big([\mathbf{z}_{\mathrm{seq}}\,;\,\mathbf{z}_{\mathrm{ph}}\,;\,\mathbf{z}_{\mathrm{di}}\,;\,\mathbf{z}_{\mathrm{st}}]\big)\Big),$$

where $\boldsymbol{g} \in \mathbb{R}^4$and $g_k$denotes the gate corresponding to feature group k. The fused representation is then computed as a weighted sum,

$$\boldsymbol{z} = \sum_{k \in \{seq,ph,di,st\}} g_k\ \boldsymbol{z}_k.$$

This gating mechanism enables the model to adaptively emphasize different feature groups under varying protocols and distribution shifts.

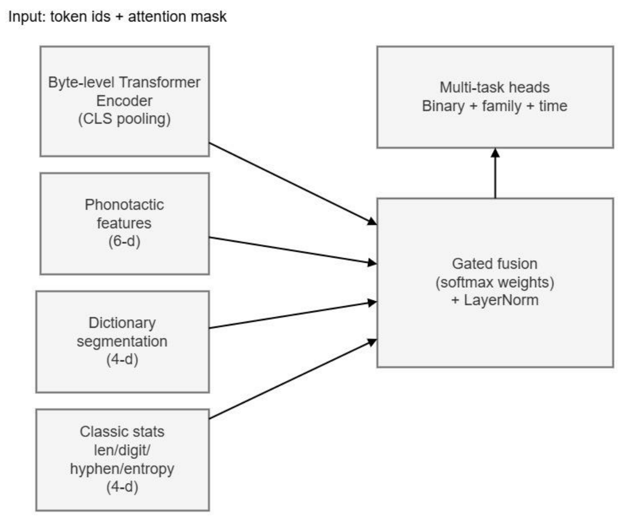
Input: token ids + attention mask
Byte-level Transformer Encoder (CLS pooling)
Phonotactic features (6-d)
Dictionary segmentation (4-d)
Classic stats len/digit/ hyphen/entropy (4-d)
Gated fusion (softmax weights) + LayerNorm
Multi-task heads Binary + family + time

Fig. 2. *PHANTOM-DGA model architecture with feature-gated fusion.*

D. Multi-Task Learning and Uncertainty Weighting

The primary task is binary DGA detection. We optionally include auxiliary classification heads for DGA family identity (restricted to families observed during training) and time-bucket prediction. The binary loss is the standard binary cross-entropy over logits.

$$(1)\quad \mathcal{L}_{bin} = -\frac{1}{B}\sum_{i=1}^{B}\left[y_i \log\sigma(\hat{z}_i) + (1-y_i)\log(1-\sigma(\hat{z}_i))\right]$$

Eq. (1). *Binary cross-entropy loss for DGA detection.*

To balance tasks with different scales, we use uncertainty-based weighting, learning one scalar s_t per task. The combined loss is:

$$(2)\quad \mathcal{L} = \sum_{t \in \{bin,\, fam,\, time\}} \exp(-s_t)\,\mathcal{L}_t + s_t$$

Eq. (2). *Uncertainty-weighted multi-task objective.*

E. Self-Supervised Pretraining

We pretrain the transformer encoder on the training set only using two objectives: (i) masked byte modeling (MLM), where a random subset of non-padding tokens is replaced by a [MASK] token and predicted from context, and (ii) a contrastive objective that aligns representations of two deterministic augmentations of the same domain. The contrastive loss is a batch-wise InfoNCE formulation [23], [24] with temperature τ.

$$(4)\quad \mathcal{L}_{con} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\mathrm{sim}(z_i, z_i^{+})/\tau)}{\sum_{j=1}^{B}\exp(\mathrm{sim}(z_i, z_j^{+})/\tau)}$$

Eq. (4). *Contrastive alignment loss (InfoNCE).*

F. Reptile Meta-Learning for Unseen Families

For the unseen-family protocol, we optionally apply Reptile meta-learning [26]. Each meta-episode samples a small set of DGA families from the training set, constructs a support set and query set (balanced with benign examples), performs a few inner optimization steps on the support set, and then updates the initialization parameters toward the adapted parameters using:

$$(3)\quad \theta \leftarrow \theta + \varepsilon(\theta' - \theta)$$

Eq. (3). *Reptile outer-loop parameter update.*

G. Probability Calibration

For deployment, calibrated probabilities are often more useful than raw scores. We apply temperature scaling [25] on the validation set and rescale test logits by a learned temperature parameter T:

$$(6)\quad \hat{z}^{(T)} = \hat{z}/T, \quad T > 0$$

Eq. (6). *Temperature scaling for calibration.*

For completeness, the phonotactic language model computes the perplexity of the consonant–vowel stream as:

$$(5)\quad \mathrm{ppl}(x) = \exp\left(-\frac{1}{N}\sum_{k=1}^{N}\log p(x_k \mid x_{<k})\right)$$

Eq. (5). *Perplexity used in phonotactic feature extraction.*

6. EXPERIMENTAL SETUP

All experiments were executed in a Kaggle notebook environment with a single CUDA GPU (≈15.9 GB memory). The implementation uses PyTorch and enables TF32 matmul on compatible GPUs for speed. Data loading is optimized using memory-mapped caches for tokens and engineered features, pinned memory, and multi-worker prefetching.

We run three evaluation protocols (Table I) and five model variants: full model, no phonotactics, no dictionary features, no meta-learning, and no pretraining. Each experiment trains on the protocol's training split, selects the best checkpoint by validation ROC-AUC, and evaluates once on the held-out test set. We report ROC-AUC, PR-AUC, F1 score at threshold 0.5, and FPR at 95% TPR (FPR@TPR=0.95), which is a common operating point for security screening.

Setting	**Value**
Transformer layers / heads	4 / 4
Model width (d_model)	128
Fusion projection dim	128
Dropout	0.1
Batch size	512
Optimizer	AdamW (fused if available)
Learning rate / weight decay	2e-4 / 0.01

Finetune epochs	2
Pretraining	1 epoch (400 steps), MLM + contrastive
Meta-learning	Reptile, 300 episodes (unseen-family only)
Mixed precision	AMP with GradScaler (CUDA)
Checkpointing	Every 400 steps (pretrain/finetune)

Table IV. *Core hyperparameters (full variant).*

7. Results and Discussion

Table V reports performance under the random split. As expected, all variants achieve very high ROC-AUC (≈0.9984–0.9986), indicating that the dataset is largely separable when train and test distributions match. However, this setting can obscure robustness issues relevant to deployment.

Variant	ROC-AUC	PR-AUC	F1	FPR@TPR=0.95
full	0.998415	0.999483	0.990473	0.001022
no_dict	0.998465	0.999497	0.990937	0.000809
no_meta	0.998543	0.999528	0.991526	0.000915
no_phono	0.998557	0.999531	0.991647	0.000873
no_pretrain	0.998459	0.999501	0.991133	0.000936

Table V. *Test performance under the random protocol.*

Table VI shows the time-forward results. Compared to the random protocol, ROC-AUC drops by ~3–4 percentage points and FPR@TPR=0.95 increases substantially. This behavior is consistent with temporal drift and changing domain naming patterns. Pretraining improves robustness in this setting: removing pretraining yields the worst FPR@TPR=0.95 (0.4195).

Variant	ROC-AUC	PR-AUC	F1	FPR@TPR=0.95
full	0.969730	0.990823	0.949480	0.207140
no_dict	0.964733	0.989451	0.948293	0.278849
no_meta	0.967407	0.990229	0.950085	0.233573
no_phono	0.964775	0.989446	0.939430	0.256474
no_pretrain	0.956747	0.987253	0.945496	0.419501

Table VI. *Test performance under the time-forward protocol.*

Table VII presents the unseen-family results. Performance declines further because the test set contains DGA families not observed during training. Under this protocol, the best ROC-AUC is 0.9359 (no-pretrain variant), while the best FPR@TPR=0.95 is 0.3983. Interestingly, disabling meta-learning (no-meta) harms F1 in this configuration, suggesting that the Reptile stage can help align the decision boundary for novel families even when the overall ROC-AUC gain is modest.

Variant	ROC-AUC	PR-AUC	F1	FPR@TPR=0.95
full	0.917486	0.969955	0.845497	0.563828
no_dict	0.928508	0.973719	0.846850	0.495978
no_meta	0.934488	0.975592	0.808507	0.419444
no_phono	0.929494	0.974002	0.854851	0.464287
no_pretrain	0.935908	0.976121	0.825253	0.398331

Table VII. *Test performance under the unseen-family protocol.*

Fig. 3 summarizes ROC-AUC across protocols and variants. The dominant factor is the evaluation protocol: random splits yield near-ceiling performance, while time-forward and unseen-family splits reveal a sizeable robustness gap. Fig. 4 and Fig. 5 provide the corresponding F1 and FPR@TPR=0.95 comparisons.

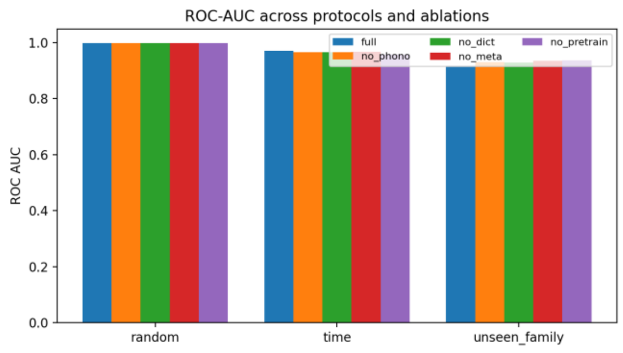

Fig. 3. *ROC-AUC across protocols and ablations.*

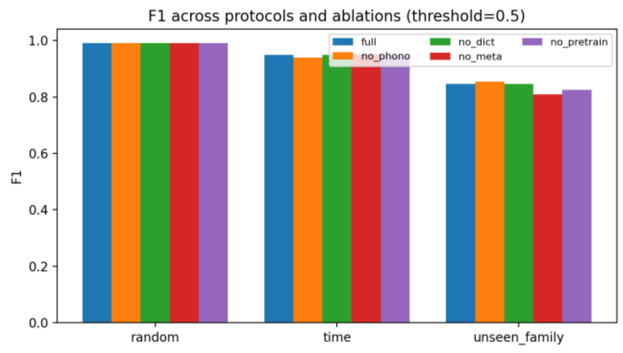

Fig. 4. *F1 score across protocols and ablations (threshold=0.5).*

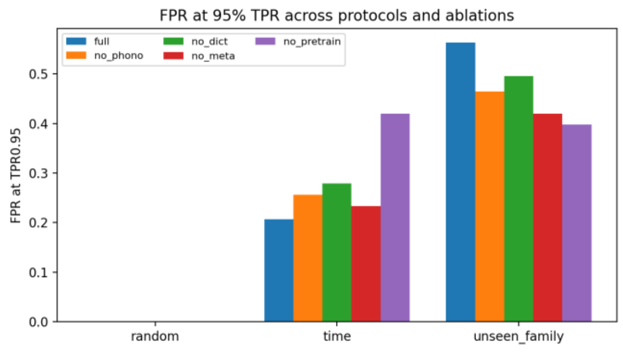

Fig. 5. *FPR at 95% TPR across protocols and ablations (lower is better).*

Ablation trends provide additional insight. Under the random protocol, removing phonotactic or dictionary features does not significantly affect ROC-AUC, and minor differences are within the expected variance of stochastic optimization. Under distribution shift, however, each component can affect operating characteristics, particularly the false-positive rate. These observations motivate reporting shift-aware metrics and operating points rather than relying solely on ROC-AUC under random splits.

8. ENGINEERING FOR LARGE-SCALE TRAINING

A distinctive aspect of PHANTOM-DGA is that it targets large datasets in commodity notebook environments. The implementation persists expensive preprocessing artifacts—token IDs, attention masks, classic statistics, and dictionary features—as memory-mapped arrays. This enables constant-memory streaming and avoids recomputation across protocols and ablations.

Training is checkpointed at a fixed interval and supports resume for both pretraining and finetuning. The pretraining stage uses GPU-side masking for masked language modeling and automatic mixed precision to accelerate throughput. The data loader is configured for GPU efficiency using pinned memory, multiple workers, and persistent workers when available. Together, these choices reduce wall-clock time and make exhaustive protocol-by-ablation evaluation feasible in practice.

A. Computational Complexity, Storage, and Throughput

PHANTOM-DGA is designed to support efficient experimentation on large-scale corpora. Persistent memory-mapped caches are used to store token identifiers and precomputed feature matrices, allowing subsequent protocol runs to reuse preprocessing outputs. This design significantly reduces end-to-end runtime by removing feature precomputation from the inner training loop.

The transformer encoder constitutes the primary computational cost. For an input sequence of length $L$and model dimension d, the self-attention mechanism incurs a per-layer complexity of

$$O(L^2 d).$$

With a fixed sequence length of $L = 63$and a modest hidden dimension of $d = 128$, the attention cost remains lightweight relative to typical natural language processing workloads.

Artifact	Datatype	Shape	Approx. size (GiB)
Token IDs (TOK_IDS)	uint16	1815815×63	0.213
Attention mask (TOK_AM)	uint8	1815815×63	0.107
Stats (STATS)	float32	1815815×4	0.027
Dictionary features (DICTF)	float32	1815815×4	0.027
Phonotactics (PHONO)	float32	1815815×6	0.041 per protocol

Table XIII. *Approximate cache footprint for the 1.82M-sample corpus (GiB).*

B. Deployment Guidance and Threshold Selection

In deployment, the classifier's output should be treated as a risk score rather than a binary decision. Operators can select thresholds based on operational capacity (alerts/day) and desired recall. Because FPR can change over time and differ across networks, thresholds should be calibrated and periodically revalidated on recent data. Time-forward validation provides a more realistic basis for threshold selection than random splits.

A common integration pattern is to combine PHANTOM-DGA with contextual signals (DNS response codes, passive DNS age, WHOIS registration features, certificate transparency, and endpoint telemetry). In such a fusion, the domain-string score acts as an early filter, prioritizing domains that are lexically suspicious and allowing downstream enrichment to focus on a smaller candidate set.

C. Security, Privacy, and Responsible Use

This work is intended to support defensive security monitoring. The model operates on domain strings and does not require user identifiers or payload data. Nevertheless, DNS telemetry can be sensitive; practitioners should minimize retention, apply access controls, and follow organizational privacy policies. When sharing datasets or trained models, care should be taken to avoid releasing personally identifiable information or internal domain naming conventions.

9. POSITIONING RELATIVE TO PRIOR WORK

A. Traditional Lexical and Feature-Based DGA Detection

Early DGA detectors relied on lexical statistics and DNS-side indicators such as NXDOMAIN ratios, query volumes, and reputation systems. Classic systems such as Pleiades and Notos leveraged DNS traffic characteristics and clustering to identify algorithmically generated domains and group them by campaign. Feature-based classifiers remain strong baselines: FANCI [2] systematically engineered features for

NXDomain classification and demonstrated that careful feature design can achieve high accuracy with moderate computational cost.

PHANTOM-DGA inherits the intuition of feature-based approaches by retaining interpretable lexical statistics and wordlist-derived features. However, rather than relying on a fixed feature set alone, it fuses these features with a learned byte-sequence representation, allowing the model to learn complementary patterns and compensate when individual feature groups fail under shift.

B. Deep Sequence Models and Representation Learning

Deep learning approaches treat the domain string as a character sequence. LSTM-based models demonstrated strong performance on DGA detection by learning sequential dependencies and morphology [3]. Character-level CNNs such as eXpose showed that convolutional architectures can capture discriminative n-gram patterns in URLs and file paths [4]. More recent work explores transformer-style encoders to model long-range dependencies and to support multi-task learning across malware families, time windows, or auxiliary labels.

Our encoder is a lightweight byte-level transformer inspired by the general success of attention-based architectures [21]. The byte-level choice avoids tokenization and supports arbitrary characters, which is important for internationalized domain names. Similar token-free modeling has been explored in NLP via byte-to-byte pretraining [27].

C. Adversarial DGAs and Robustness Evaluation

A growing body of work studies adversarial domain generation and evasion. DeepDGA introduced adversarially tuned generation to challenge detectors, while CharBot [6] and MaskDGA [7] highlight that simple perturbations can evade classifiers. DomainGAN [8] further showed that generative models can produce plausible domains that degrade detection. These studies emphasize that evaluation must incorporate distribution shift and adversarial settings.

In this paper, robustness is evaluated through time-forward and unseen-family protocols that mimic two common real-world failure modes: drift in benign and malicious distributions, and emergence of new DGA families. Our results show substantial degradation in high-recall operating points under these protocols, consistent with the SoK guidance that protocol selection significantly affects conclusions [17].

D. Calibration and Decision-Making

Beyond ranking metrics, calibrated probabilities are important for downstream decision-making, alert triage, and fusion with other security signals. Neural networks are often miscalibrated, motivating post-hoc calibration methods such as temperature scaling [25]. In our implementation, calibration is fit on the validation set and applied to test logits without leakage.

E. Summary Comparison

Method	Model family	Input	Shift-aware evaluation	Notes
FANCI [2]	Feature-based (RF/MLP)	Lexical + DNS features	Limited (dataset-dependent)	Strong NXDomain baseline; interpretable features.
LSTM DGA [3]	RNN	Character sequence	Often random split	Learns sequential morphology; sensitive to distribution shift.

eXpose [4]	Char-CNN	Characters	Task-dependent	General malicious string detector; efficient at inference.
MaskDGA/CharBot [6],[7]	Attack methods	String perturbations	—	Demonstrate evasion of learned classifiers.
PHANTOM-DGA (this work)	Byte-Transformer + gated fusion	Bytes + engineered features	Yes (time, unseen-family)	Multi-task, cached pipeline, calibration.

***Table XII**. Qualitative positioning of PHANTOM-DGA relative to representative prior work.*

10. LIMITATIONS AND FUTURE WORK

This study has several limitations. First, the benign set is derived from a popularity list and may not represent benign DNS traffic in enterprise or ISP settings. Second, benign timestamps are approximated using deterministic pseudo-timestamps, which enables time-forward splitting but does not capture real temporal dynamics of benign domains. Third, the model operates only on the lexical domain string; richer signals (DNS query behavior, IP reputation, WHOIS, certificate transparency) can significantly improve detection in production systems. Finally, adversarial robustness is evaluated only implicitly through shift-aware splits; explicit adversarial training against generative attacks such as DomainGAN [8] or MaskDGA [7] is left for future work.

Future work should evaluate PHANTOM-DGA on additional datasets and streaming DNS traces, incorporate contextual features while preserving deployment cost, and explore stronger self-supervised objectives for domain strings (e.g., byte-level pretraining inspired by token-free models [27]). In addition, the meta-learning setup can be extended to probabilistic adaptation and uncertainty estimation to better handle novel families and label noise.

11. CONCLUSION

We presented PHANTOM-DGA, a feature-gated multi-task transformer for DGA detection designed for rigorous evaluation and practical training at scale. Across three evaluation protocols, results highlight a pronounced robustness gap between random splits and shift-aware splits. While near-ceiling performance is achievable under random splits, time-forward and unseen-family protocols reveal substantially higher false-positive rates at high recall, underscoring the importance of realistic benchmarks. The proposed pipeline integrates efficient preprocessing caches, optional self-supervised pretraining, optional Reptile meta-learning, and temperature calibration, providing a flexible baseline for future research on robust and deployable DGA detection.

REFERENCES

[1] M. Antonakakis, R. Perdisci, Y. Nadji, N. V. II, S. Abu-Nimeh, W. Lee, and D. Dagon, “From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware,” in Proc. USENIX Security Symposium, 2012.

[2] J. Schuppen, D. Teubert, P. Herrmann, and U. Meyer, “FANCI: Feature-Based Automated NXDomain Classification and Intelligence,” in Proc. USENIX Security Symposium, 2018.

[3] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, “Predicting Domain Generation Algorithms with Long Short-Term Memory Networks,” arXiv:1611.00791, 2016.

[4] J. Saxe and K. Berlin, “eXpose: A Character-Level Convolutional Neural Network with Embeddings for Detecting Malicious URLs, File Paths, and Registry Keys,” arXiv:1702.08568, 2017.

[5] C. Patsakis and F. Casino, “Exploiting Statistical and Structural Features for the Detection of Domain Generation Algorithms,” arXiv:1905.02049, 2019.

[6] J. Spooren, D. Preuveneers, and W. Joosen, “CharBot: A Simple and Effective Method for Evading DGA Classifiers,” arXiv:1905.01078, 2019.

[7] D. Dawson, Q. Chen, B. Edwards, and A. Stupples, “MaskDGA: A Black-Box Evasion Attack on DGA Classifiers,” arXiv:1902.08909, 2019.

[8] W. Zhang, B. Li, H. Peng, and W. Wang, “DomainGAN: Generating Adversarial Domain Names for DGA Detection,” arXiv:1911.06285, 2019.

[9] S. Sayadi, J. Abawajy, N. Chowdhury, and E. Kelarev, “DGA Detection with Deep-Learning Through NXDomain,” in Proc. KSEM, 2019.

[10] R. Bilbo, “Advanced DGA Domain Detection using Deep Learning and Adaptive Ensemble,” arXiv:2108.06106, 2021.

[11] A. Maglia, M. Gritti, and F. Frati, “EXPLAIN: Explainable DGA Detection with Domain Sequence Modeling,” arXiv:2108.05496, 2021.

[12] B. Bezawada, P. Nambiar, and D. P. Rao, “A Collaborative Learning Approach for DGA Detection,” arXiv:2110.02519, 2021.

[13] M. M. Khaing, “DGA Classification With Deep Learning: Challenges and Opportunities,” IEEE Access, 2021.

[14] M. I. Al-hammdani et al., “Network Traffic Based DGA Detection using Hybrid Embedding and Transformer (NIOM-DGA),” Int. J. Adv. Eng. and Management, 2023.

[15] Z. M. Fakhr and M. A. Jaber, “TransFlexNet: A Hybrid Transformer–CNN Model for DGA Detection,” Electronics, vol. 13, no. 7, 2024.

[16] V. M. Ajila and A. R. Wani, “Programmable-Switch–Based DGA Detection in P4 Data Planes,” IEEE Access, 2024.

[17] B. Miller et al., “SoK: A ‘Down to Earth’ Approach for Evaluation Guidelines for Domain Generation Algorithm Detection,” in Proc. RAID, 2024.

[18] M. D. Tran et al., “UIT-DGADetector: Detect Domains Generated by Algorithms for Malware Detection,” Cluster Computing, 2024.

[19] C. Zhao, Z. Cao, and J. Tian, “PUFS: Partially Labeled Data with Uncertainty-Aware Sample Filtering for Malicious Domain Detection,” in Proc. ACM AsiaCCS, 2022.

[20] J. Yadav, A. K. Reddy, A. K. Reddy, and S. Ranjan, “Detecting Algorithmically Generated Malicious Domain Names,” in Proc. ACM IMC, 2010.

[21] A. Vaswani et al., “Attention Is All You Need,” in Proc. NeurIPS, 2017.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. NAACL-HLT, 2019.

[23] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation Learning with Contrastive Predictive Coding,” arXiv:1807.03748, 2018.

[24] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations,” in Proc. ICML, 2020.

[25] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On Calibration of Modern Neural Networks,” in Proc. ICML, 2017.

[26] A. Nichol, J. Achiam, and J. Schulman, “On First-Order Meta-Learning Algorithms,” arXiv:1803.02999, 2018.

[27] L. Xue et al., “ByT5: Towards a Token-Free Future with Pre-Trained Byte-to-Byte Models,” Trans. Assoc. Comput. Linguistics, 2022.

[28] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” arXiv:1711.05101, 2019.

[29] P. Micikevicius et al., “Mixed Precision Training,” arXiv:1710.03740, 2018.

[30] A. Costello, “Internationalized Domain Names in Applications (IDNA): Protocol,” RFC 5890, 2010.

[31] M. A. Kshirsagar, S. N. Mehetre, and R. A. Bhavsar, “A Machine Learning Approach to Detect Domain Names Generated by DGA Malware,” Procedia Computer Science, 2022.

[32] M. O. A. Elazab et al., “DGA Domain Detection based on Feature Extraction and Neural Networks,” PLOS ONE, 2022.