Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

When Phishing Detectors Don't Generalize: A Diagnostic Study of BERT-Based Models Under Cross-Distribution Stress

Rahul Nayak
VIT-AP University
Shweta Sankhwar
shweta.sank@gmail.com
Maitreyi College (University of Delhi)
Shreyas Rajendra Hole
Symbiosis Institute of technology

Article

Keywords:

Posted Date: November 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11105894/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

When Phishing Detectors Don't Generalize: A Diagnostic Study of BERT-Based Models Under Cross-Distribution Stress

Rahul Nayak[1] and Dr. Shweta Sankhwar[2*]

[1]School of Computer Science and Engineering (SCOPE), VIT-AP University, Inavolu, Guntur, 522241, Andhra Pradesh, India.
[2*]Department of Computer Science, Maitreyi College, University of Delhi, New Delhi, 110021, India.

*Corresponding author(s). E-mail(s): ssankhwar@maitreyi.du.ac.in;
Contributing authors: rahuln9261@gmail.com;

Abstract

Phishing detectors built on BERT-style encoders often report macro-F1 above 0.95, but usually on a held-out slice of the corpus they were trained on. We wanted to know how much of such a score survives outside that corpus. We fine-tuned a DistilBERT model with 24 handcrafted features, reaching a macro-F1 of 0.97 on its own test split, and ran the same checkpoint through five diagnostic tests. Deleting exclamation marks cost 0.003 F1 and shuffling word order cost 0.044, so the model leans mainly on which words appear. On six other email sources F1 fell to 0.89, with most of the loss on CEAS-08 (0.87) and a merged file (0.86); on two phishing-only sources the model still caught 95–98% of phishing emails. On URLs it was close to chance (0.51), although fine-tuning on 500 labelled URLs brought it back to 0.94. SHAP on a tree-based proxy ranked exclamation density first, a cue the network barely uses. The handcrafted features, fed in unscaled, did not beat DistilBERT alone (0.97 versus 0.98, one seed). Everything here comes from one model and one training corpus, so we offer the tests as a protocol for checking other detectors.

Keywords: Phishing detection, BERT, Distribution shift, Cross-modal transfer, Domain adaptation, Explainable AI

1 Introduction

Transformer-based phishing detectors now routinely report accuracy or F1 between about 0.96 and 0.999 on public email and URL benchmarks. One hybrid of BERT with convolutional, recurrent and attention layers reports an F1 of 0.963 on a Kaggle email corpus [1]. A few-shot ModernBERT model reports 99.9% accuracy from just ten labelled examples per class [2], and in a comparison of fourteen models on ten datasets, BERT and RoBERTa reach 98.99% and 99.08% [3]. Most of these results follow the same recipe. Fine-tune a transformer, perhaps add a vector of handcrafted features, and test on a held-out part of the corpus used for training.

Phishing has not gone away. The Anti-Phishing Working Group counted 877,536 phishing attacks in the second quarter of 2024 alone [4]. A held-out split, though, comes from the same sources, the same period and the same writers as the training data, so a high score on it tells us little about messages from anywhere else. The authors of the PhreshPhish benchmark make a similar point: leakage and unrealistic base rates in public phishing datasets produce overly optimistic results [5]. Malware detection research ran into the same problem a few years earlier [6].

We set out to measure how much of a high in-distribution score survives simple, controlled changes to the test data. We trained one reference detector, a DistilBERT encoder with 24 handcrafted features that scores a macro-F1 of 0.97 on its own test split, and put the same checkpoint through five diagnostic tests. The surface tests, which delete exclamation marks or scramble word order, cost little. Other email corpora cost more: F1 fell to 0.89 overall, and most of that loss came from two of the six sources. On URLs the model was close to chance at 0.51, yet a few hundred labelled URLs were enough to win most of it back. Two results surprised us. A SHAP analysis on a tree-based proxy singled out a feature that the network hardly uses, and adding the handcrafted features did not improve on DistilBERT alone.

We should be clear about scope. Everything here comes from one backbone, one feature set and one training corpus, and most comparisons rest on a single random seed. The paper is therefore a worked example of a diagnostic protocol, not a verdict on published detectors. Its main contribution is the protocol itself: five tests (D1–D5) that can be run on any trained phishing text classifier. Applying them to one DistilBERT detector, we measure its behaviour under surface, corpus and modality shifts, show a concrete case in which proxy-based SHAP explanations point the wrong way, and find that 500 labelled URLs lift macro-F1 on PhiUSIIL from 0.51 to 0.94. We end with a short reporting checklist, which includes a warning we learned the hard way: macro-F1 means little on test subsets that hold only one class.

2 Related Work

2.1 Transformer-based phishing detection

Salloum et al. [7] reviewed 100 studies of natural language processing for phishing email detection, published between 2006 and 2022. TF-IDF and word embeddings were the most common text representations in those studies, and the Nazario corpus was the most common benchmark. Benavides-Astudillo et al. [8] classified the text of web

pages with GloVe embeddings and recurrent networks, reporting a mean accuracy of 97.4% for a bidirectional GRU. Eze and Shamir [9] looked at AI-generated phishing emails and found that fairly simple classifiers, an LSTM network among them, separate these from human-written emails with 97–100% accuracy.

Pretrained transformers push the numbers higher still. Hosseinzadeh et al. [1] stacked convolutional, recurrent and attention layers on BERT embeddings, tuned the whole model with the Mountain Gazelle Optimizer, and reported 96.8% accuracy and an F1 of 0.963. XF-PhishBERT [2] pairs a ModernBERT encoder with selected URL features and few-shot learning, and reports 99.9% accuracy from ten examples per class. Alhuzali et al. [3] compared fourteen machine-learning and deep-learning models on ten datasets, and BERT and RoBERTa came out on top. Most of these studies test on a held-out part of the corpus they train on. XF-PhishBERT is the exception, since it also reports a cross-dataset test.

2.2 Hybrid feature fusion

Adding other features to transformer embeddings is a popular idea. Chang et al. [10] joined TF-IDF vectors to BERT embeddings and, usefully, compared the hybrid with BERT alone on sets of 1,000, 5,000 and 10,000 emails. The gain was about one point of accuracy (0.981 against 0.972 on 10,000 emails). XF-PhishBERT likewise adds selected URL features to its encoder [2]. Our reference model does the same with handcrafted URL, persuasion, style and character features, and we train a DistilBERT-only baseline on the same split so that the features can be judged.

2.3 Evaluation under distribution shift in security

Pendlebury et al. [6] showed that random train–test splits can make Android malware classifiers look far better than they are, and proposed TESSERACT, an evaluation framework that keeps samples in time order. Closer to our topic, Gholampour and Verma [11] tested how phishing email detectors hold up against adversarial edits to the text, and Reda et al. [12] built a framework that monitors phishing detectors and retrains them as the data drift. Schmitt and Flechais [13] reviewed how generative AI is changing phishing and social engineering. The PhreshPhish authors [5] built a new website benchmark partly because older datasets gave overly optimistic results. We carry the same worry over to a text-based phishing detector and measure four kinds of shift (surface artifacts, word order, corpus and input type), plus a recovery curve.

2.4 Explainable AI for phishing classifiers

SHAP [14] and similar post-hoc methods are widely used to explain security models, phishing detectors included [15]. EXPLICATE [16], for instance, explains a feature-based phishing classifier with both SHAP and LIME and then has a large language model turn the explanations into plain text. For tree ensembles, SHAP values can be computed exactly and quickly [17], which makes tree models tempting stand-ins. The catch is that SHAP values describe whichever model they were computed on. Computed on a separate model fitted to handcrafted features, they may say little

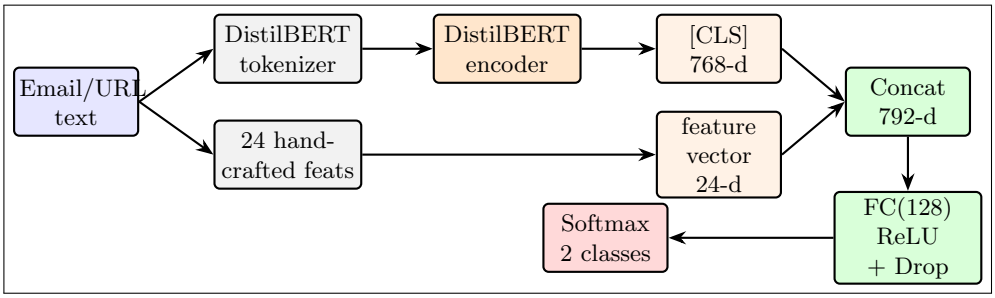


Fig. 1: The reference model. The DistilBERT `[CLS]` embedding is joined with 24 handcrafted features and passed through a small two-layer head.

about a transformer that reads the whole text. Section 4.7 shows a case where they point the wrong way.

3 Method

3.1 Reference model

Our reference detector, PhishModel, puts a small feature head on a DistilBERT encoder [18]. We picked this design because it resembles the hybrid detectors described above, so whatever the tests find should be relevant to them.

Figure 1 shows the architecture. Input text is tokenized with the DistilBERT tokenizer and cut to 128 tokens. The 768-dimensional `[CLS]` embedding is concatenated with the 24 handcrafted features, giving a 792-dimensional vector, which passes through a fully connected layer of width 128 (ReLU, dropout 0.3) and then a two-class output layer. The features enter the network as raw values. We did not scale them, and Section 4 shows why that matters.

3.2 Handcrafted features

The 24 features cover the cues phishing detectors usually look for: links, persuasion, writing style and character statistics. Seven of them describe brands and links. They count brand and service names (Microsoft, Google, PayPal, Apple and bank names, for example), URLs, URL-shortener links and links with a suspicious top-level domain; they flag URLs that use a raw IP address and text that contains Cyrillic letters, a simple homoglyph signal; and they give the share of links that use HTTPS. Seven more capture persuasion: urgency words per 100 words, flags for credential requests and generic greetings, and counts of threat words, deadline phrases, authority terms and reward words. Four describe style, namely the type–token ratio, Flesch reading ease, average sentence length and exclamation marks per 1,000 characters. The last six are character statistics: the share of uppercase letters and of special characters, the text length in characters, and the numbers of all-caps words, currency symbols and line breaks.

We compute the features with regular expressions, small word lists and the `textstat` library. One of them is broken. The all-caps word count is computed after the text has been lower-cased, so it is always zero, and in practice the model received 23 informative features.

3.3 Data

The training corpus is the Phishing Email dataset published on Kaggle (subhajournal/phishingemails). Once rows with empty text are removed, it holds 18,634 emails: 7,312 phishing and 11,322 legitimate. We kept all 7,312 phishing emails, drew the same number of legitimate emails at random (seed 42), and removed duplicates by the MD5 hash of each email's first 500 characters. That left 13,388 emails (6,295 phishing and 7,093 legitimate), which we split 70/15/15 with stratified sampling (seed 42) into 9,371 training, 2,008 validation and 2,009 test emails. The test split holds 1,064 legitimate and 945 phishing emails.

For the cross-corpus test (D3) we use the phishing email collection compiled by Al-Subaiey et al. [19] and released on Kaggle (naserabdullahalam/phishing-email-dataset). It contains six source files, namely Enron [20], CEAS-08, SpamAssassin [21], Ling, Nazario [22] and Nigerian Fraud, together with a merged file (`phishing_email.csv`) that combines the six sources into a single text field after removing punctuation and stop words [19]. Two details matter when reading D3. The Nazario and Nigerian Fraud files contain only phishing and fraud emails, and in several of the other sources the positive class covers spam in general rather than phishing alone. The MeAJOR corpus [23] gathers a similar set of sources and would make a natural next target.

For the URL tests (D4 and D5) we use PhiUSIIL [24, 25], which contains 134,850 legitimate and 100,945 phishing URLs; we took it from its Kaggle mirror (ndarvind/phiusiil-phishing-url-dataset). The model reads the URL string as its text, and the features are computed from that string.

3.4 Training

We fine-tune for three epochs with AdamW (learning rate 2×10^{-5}, batch size 32, weight decay 0.01), clip gradients at 1.0, and keep the checkpoint with the best validation macro-F1. All experiments ran on a Google Colab T4 GPU, where one epoch took about two minutes.

Three models are compared, each trained once with seed 42:

1. Baseline-LR, a logistic regression on the 24 handcrafted features after standardizing them to zero mean and unit variance;
2. Baseline-DistilBERT, the same network as PhishModel with the features removed, so that the head sees only the `[CLS]` embedding;
3. PhishModel, DistilBERT plus the 24 features, as described above.

We report macro-averaged F1 (simply F1 from here on), accuracy, and macro-averaged precision and recall, all computed with scikit-learn.

3.5 Diagnostic tests

All five tests use the same trained PhishModel checkpoint (seed 42). Whenever a test changes the text, we recompute the handcrafted features from the changed text.

D1 deletes every `!` from the test emails. Exclamation marks are a classic phishing cue, and we wanted to know whether the model depends on them.

D2 shuffles word order. Each test email is split on whitespace, its words are put in random order (fixed seed) and joined again. A model that keeps most of its F1 after this is relying mainly on which words appear, not on their order.

D3 moves to other email corpora. We pool the seven files of the cross-corpus collection (164,971 emails), draw 2,000 positive and 2,000 legitimate emails at random (seed 42), and report F1 on the whole sample and for each source file. For the two sources that contain only phishing emails we report the share flagged as phishing instead; Section 4.4 explains why.

D4 changes the input type. We draw 1,000 phishing and 1,000 legitimate PhiUSIIL URLs at random (seed 42) and give them to the email-trained model with no further training.

D5 measures recovery. We set aside a fixed test set of 1,000 PhiUSIIL URLs (500 per class) and, from the remaining URLs, draw training sets of $N \in \{50, 100, 500, 1000, 3000\}$ URLs, half phishing and half legitimate. Starting each time from the PhishModel checkpoint, we fine-tune for two epochs (AdamW, learning rate 5×10^{-6}, batch size 16) and evaluate on the fixed test set. This test set is a different sample from the one used in D4.

4 Results

4.1 Baselines and reference model

Table 1 gives the in-distribution results. DistilBERT on its own reached an F1 of 0.984; with the handcrafted features added, PhishModel reached 0.972. That is a gap of 0.013 from one training run per model, and fine-tuned language models can move by that much from a change of random seed alone [26]. So we do not read the gap as evidence that the features are harmful.

The training log points to a more specific culprit. The mean training loss of PhishModel was 6.3, 2.4 and 1.5 in its three epochs, and single batches went above 25, far beyond what a well-fitted two-class model produces. The unscaled features are the likely cause: text length alone runs into the thousands, while the `[CLS]` values are small. In our setup, then, naive fusion of unscaled features did not help and may have hurt. A fair test would scale the features and average over several seeds (Section 7). The features are not useless on their own, since logistic regression on the standardized features reached an F1 of 0.768.

Figure 2 shows how the same PhishModel checkpoint fares in tests D1 to D4.

4.2 D1: Removing exclamation marks

Deleting every `!` from the test emails moved F1 from 0.9715 to 0.9685, a drop of 0.003. Exclamation marks, in other words, hardly matter to this model. Keep that in

Table 1: In-distribution results on the held-out test split (2,009 emails). Each model was trained once, with seed 42.

Model	Accuracy	Macro F1	Precision	Recall
Baseline-LR (features only)	0.7715	0.7676	0.7766	0.7666
Baseline-DistilBERT (no features)	**0.9841**	**0.9840**	**0.9841**	**0.9840**
PhishModel (DistilBERT + 24 features)	0.9716	0.9715	0.9719	0.9711

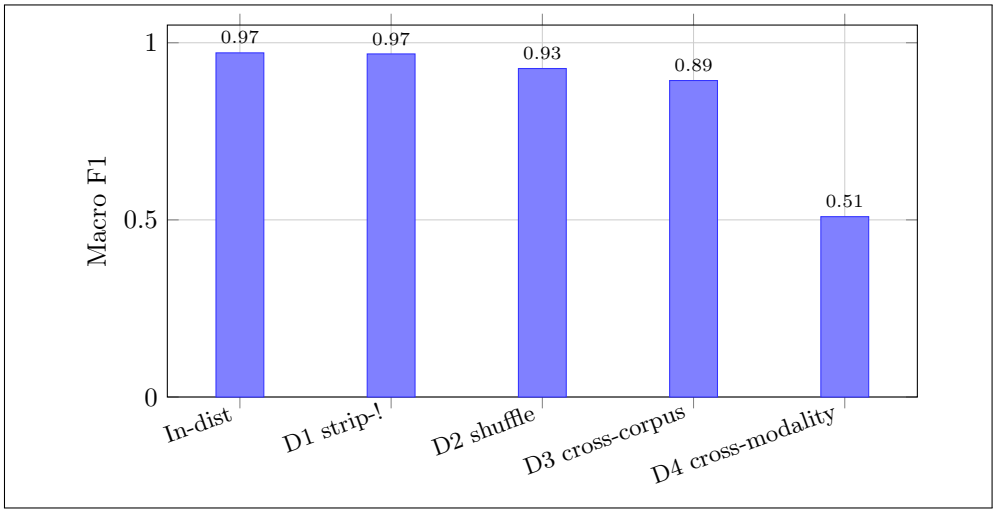


Fig. 2: Macro-F1 of the same PhishModel checkpoint on its own test split and in tests D1–D4. On URLs (D4) it falls to 0.51, close to chance for a balanced two-class task.

mind for Section 4.7, where a tree-based proxy ranks exclamation density as the most important feature of all.

4.3 D2: Shuffling words

Shuffling the words in each test email lowered F1 to 0.9273, a drop of 0.044. Most of the model's accuracy survives the loss of word order, which means it relies mainly on which words appear, much as a bag-of-words classifier would. Order is not irrelevant, though, or the drop would be zero.

4.4 D3: Other email corpora

On the 4,000 emails drawn from the cross-corpus collection, F1 fell to 0.8934, a drop of 0.078. Table 2 breaks this down by source file.

Two files account for most of the loss: CEAS-08 at 0.87 and the merged file at 0.86. The merged file stores its text without punctuation or stop words [19], so some of its

Table 2: D3 results by source file. For files with both classes we report macro-F1. The Nazario and Nigerian Fraud files hold only phishing emails, so for them we give the share of emails the model flagged as phishing; macro-F1 means little on a single-class subset.

Source file	N	Classes present	Result
Ling	75	both	F1 = 1.0000
Enron	706	both	F1 = 0.9681
SpamAssassin	133	both	F1 = 0.9642
CEAS-08	904	both	F1 = 0.8747
Merged file (`phishing_email.csv`)	2,065	both	F1 = 0.8614
Nazario	44	phishing only	43 of 44 flagged (97.7%)
Nigerian Fraud	73	phishing only	69 of 73 flagged (94.5%)
All sources	4,000	both	F1 = 0.8934

drop may come from the change in format rather than from the emails themselves. Enron, SpamAssassin and Ling stay between 0.96 and 1.00. We would not lean on those high scores, because the positive class in these files is mostly spam and we did not check how much material they share with our training corpus.

The two phishing-only files need a different yardstick. With no legitimate emails in the subset, the legitimate class gets an F1 of zero the moment the model makes a single mistake, and macro-F1 is then capped at 0.5 however many emails are caught. Counted directly, the model flagged 43 of the 44 Nazario emails (97.7%; 95% Wilson interval 88.2–99.6%) and 69 of the 73 Nigerian Fraud emails (94.5%; 86.7–97.8%). What these files cannot tell us is anything about false alarms, since they contain no legitimate mail.

4.5 D4: URLs

On 2,000 PhiUSIIL URLs (1,000 per class), F1 fell to 0.5091 and accuracy to 0.56. The predictions were lopsided: the model gave the same label to about 82% of the URLs. None of this is surprising. A URL has no body text, no greeting and no signature, so most of the handcrafted features read zero, and the token sequences look nothing like the emails the model learned from. The point is that the in-distribution score gives no hint of it. This detector should not be used on URLs.

4.6 D5: Recovery curve

Figure 3 shows the recovery curve. Fine-tuning on just 50 URLs (25 per class) lifted F1 to 0.62; 500 URLs took it to 0.94, and 3,000 to 0.99. Beyond about 1,000 URLs the gains are small. Each point is a single run on a single draw of training URLs, scored on a 1,000-URL test set, so the exact values carry some uncertainty. And because we ran no control, the curve cannot tell us how much the email training actually helps (Section 7).

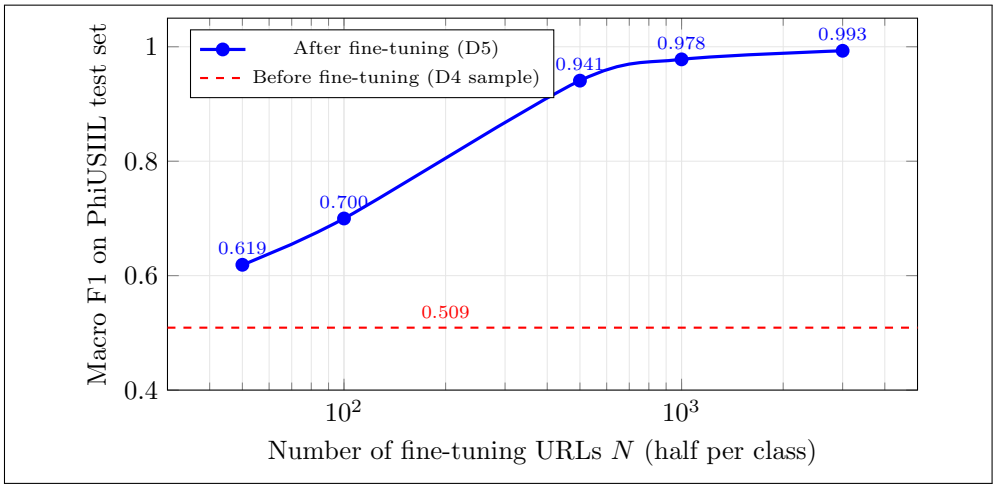


Fig. 3: D5 recovery curve. Fine-tuning the PhishModel checkpoint on 500 PhiUSIIL URLs brings macro-F1 to 0.94, and 3,000 URLs bring it to 0.99. The dashed line is the score before fine-tuning, measured on the separate 2,000-URL D4 sample. Each point is a single run.

4.7 SHAP on a tree-based proxy

We fitted a gradient boosting classifier (100 trees, maximum depth 4) to the same 24 features and the same training split as PhishModel. This proxy reached an accuracy of 0.85 on the test split, better than logistic regression (0.77) but well short of the neural models. We then computed exact TreeSHAP values [17] for 500 randomly chosen test emails. Figure 4 shows the ten features with the largest mean absolute SHAP value.

Exclamation density comes first, with a mean absolute SHAP value of 0.92, about 1.6 times that of the runner-up, the type–token ratio (0.58). By the proxy's account, exclamation marks are the strongest signal in the data. Yet removing them changed the F1 of the neural model by 0.003 (D1).

The SHAP values are not wrong. The gradient boosting model really does rely on exclamation density. The mistake would be to treat the proxy's ranking as an explanation of the fused transformer, which can draw its cues from the full text. SHAP values computed on a proxy describe the proxy, and we come back to this in Section 5.

4.8 Error breakdown

Figure 5 shows the in-distribution confusion matrix of PhishModel. Of the 1,064 legitimate test emails, 22 were flagged as phishing (2.1%); of the 945 phishing emails, 35 slipped through (3.7%).

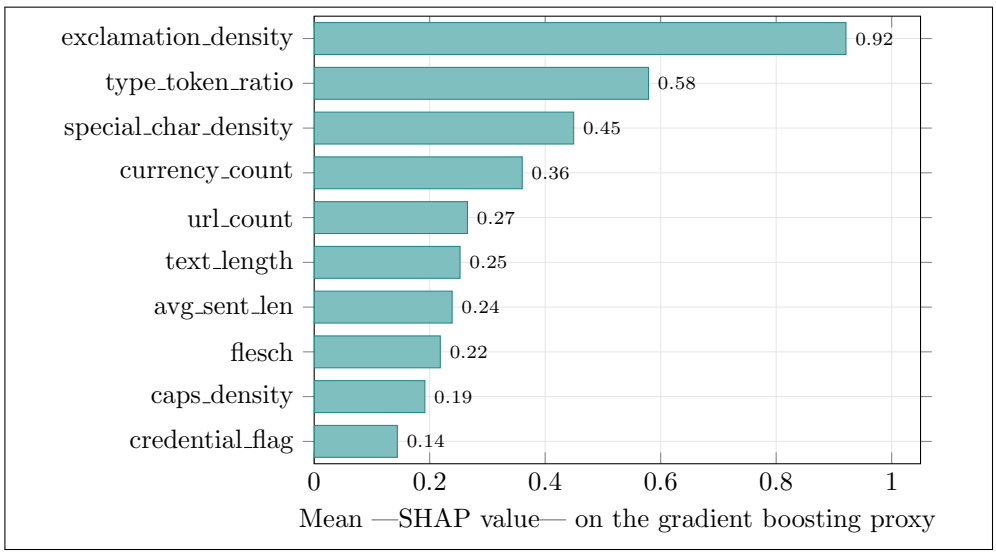


Fig. 4: The ten most important features according to TreeSHAP on a gradient boosting proxy fitted to the handcrafted features (500 test emails). The proxy puts exclamation density first, yet removing exclamation marks barely changes the neural model's predictions (D1).

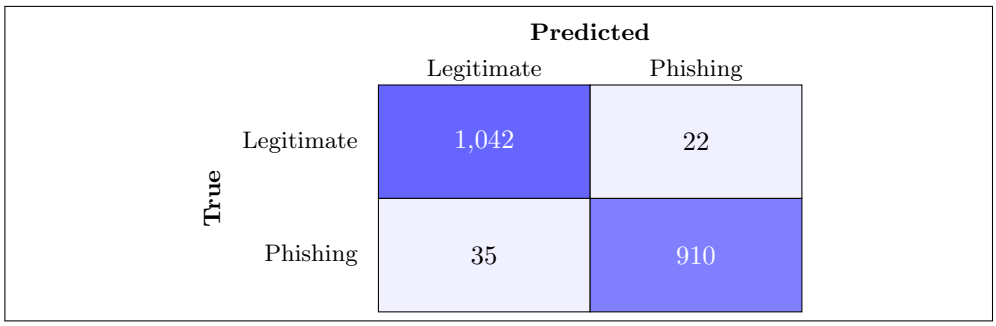


Fig. 5: In-distribution confusion matrix for PhishModel on the test split (1,064 legitimate and 945 phishing emails).

4.9 Inference latency

We timed single-message forward passes of PhishModel (batch size 1, 128 tokens, 100 timed runs after 5 warm-up runs). On a Colab T4 GPU the median was 7.5 ms, with a 95th percentile of 9.4 ms; on the Colab CPU the median was 156 ms and the 95th percentile 241 ms. These figures cover the model alone, not tokenization or feature extraction. They suggest that scanning messages one at a time is feasible, but we did not measure throughput under load or on end-user devices.

5 Discussion

Adding handcrafted features to DistilBERT did not help in our setup. The fused model scored 0.013 lower than DistilBERT alone, but that rests on one seed per model, and the high training loss suggests that the unscaled features got in the way of training. It is also possible that the encoder already picks up much of what the features measure, such as urgency words or links; our experiments cannot separate that from the scaling problem. What the comparison does show is how much a DistilBERT-only baseline on the same split matters. Without one, any gain or loss from the features cannot be interpreted. Chang et al. [10] include such a baseline, and we would like to see it become routine in hybrid studies.

The SHAP result has practical consequences. Tree-based proxies are attractive because TreeSHAP is exact and fast [17], but a proxy fitted to handcrafted features can rank them very differently from the way the full model uses them. Here the proxy's favourite feature has almost no effect on the network. A reader who trusted the proxy would come away with the wrong picture of the deployed model, and a defender who acted on it would be watching the wrong cue. If the aim is to explain a fused transformer, the attributions should be computed on that model, for instance with integrated gradients [27], or at the very least checked with an ablation such as D1.

The tests also make plain what an in-distribution score leaves out. A score of 0.97 says how well the model reproduces the labels of its own corpus. It does not say that the model loses about 0.08 F1 on other email corpora, mostly on particular sources, or that it is close to chance on URLs. None of this is surprising in hindsight, but none of it is visible in the in-distribution number either.

The recovery curve is the more encouraging part of the story. On PhiUSIIL, a few hundred labelled URLs and two epochs of fine-tuning on a T4 GPU took the email-trained model from 0.51 to 0.94. A team that needs a detector for a new kind of input may get a long way with a small labelled sample of its own data. One caution applies: published models already exceed 99% accuracy on PhiUSIIL [24], so the dataset may be easier than live traffic, and the same budget may not be enough elsewhere.

6 Recommendations: a reporting checklist

Papers that claim practical phishing detection should, in our view, report more than in-distribution scores. We suggest the following:

1. In-distribution macro-F1 as a mean and standard deviation over at least three random seeds. We fell short of this ourselves for most runs (Section 7), which is why it comes first.
2. The drop in F1 when word order is shuffled (D2). A small drop means the model relies mainly on which words appear.
3. An ablation of the top-ranked features on the model itself (D1): remove or neutralize each feature and evaluate again.
4. Results on at least one other corpus (D3), overall and per source. For test subsets that hold only one class, report the detection rate (or the false-alarm rate) rather

than macro-F1, which on such a subset cannot exceed 0.5 once the model makes a single mistake, however well it does otherwise.

5. Results on other input types (D4) for any paper that claims general phishing detection, or a plain statement that the model handles one input type only.

A recovery curve (D5) is a useful extra, ideally with a control that fine-tunes a model without the source-domain training. None of this is expensive. Once a model is trained, D1 to D4 need only inference, and D5 needs a handful of short fine-tuning runs.

7 Limitations

The study has real limits, and some of them bear directly on the results above. It covers one backbone (DistilBERT), one feature set and one training corpus. Other architectures, features or corpora may behave differently, and nothing here should be read as a statement about published detectors in general.

Each model and each diagnostic test was run once, with seed 42, and we report no confidence intervals for F1. Fine-tuning results can shift noticeably with the seed [26], so the 0.013 gap between PhishModel and DistilBERT alone lies within what seed variation alone could produce. This falls short of our own first recommendation (Section 6). Several D3 subsets are also small (44 Nazario, 73 Nigerian Fraud and 75 Ling emails), which makes their results uncertain.

The feature fusion has two problems of its own. The features went into the network unscaled, and the training loss of the fused model stayed high. On top of that, one of the 24 features, the all-caps word count, was always zero because of a coding error. Our comparison therefore cannot tell whether the features are useless or were simply fused badly. Settling that would take scaled features, a fixed feature extractor and several seeds.

The cross-corpus data are imperfect too. In several D3 sources the positive class is spam in general rather than phishing, and the merged file repeats the other six sources with punctuation and stop words removed. We did not check for overlap between our training corpus and the D3 sources, so the high scores on Enron, SpamAssassin and Ling may partly reflect shared material. The phishing-only sources, for their part, say nothing about false alarms.

On the URL side, the failure in D4 is what one would expect from a model trained on email text. The recovery curve in D5 has one run per point and a 1,000-URL test set, and it lacks a control that fine-tunes a fresh DistilBERT model on the same URLs. It therefore shows how quickly this checkpoint adapts to PhiUSIIL, not how much the email training helps, and PhiUSIIL itself may be easier than real traffic [24].

We also leaned on a tree-based proxy to explain feature importance, the very shortcut we caution against. Our analysis shows what the network does not rely on (exclamation marks) but not what it does rely on. Attributions computed on the transformer itself, for example with integrated gradients [27], would close that gap.

Finally, the five tests leave out adversarial perturbations, drift over time (training on older emails and testing on newer ones, for instance) and phishing written by large

language models [28]. Each would make a worthwhile extension. Latency was measured only on Colab hardware.

8 Conclusion

A DistilBERT phishing detector that scores a macro-F1 of 0.97 on its own test split scored 0.89 on other email corpora and 0.51 on URLs. A tree-based SHAP proxy put first a feature that the network barely uses, and adding handcrafted features, unscaled, did not improve on DistilBERT alone in a single-seed comparison. On two phishing-only sources the detector still caught 95–98% of phishing emails, a result that per-source macro-F1 would have hidden.

All of this comes from one model and one training corpus. It shows what the diagnostic tests can reveal, not how common such gaps are across the field. The tests are cheap once a model is trained, and we hope other phishing detection papers will start reporting them, so that readers can judge how much of an in-distribution score carries over to data the model has never seen.

Data availability

The datasets analysed in this study are publicly available: the Phishing Email dataset on Kaggle (https://www.kaggle.com/datasets/subhajournal/phishingemails), the phishing email collection of Al-Subaiey et al. on Kaggle (https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset), and the PhiUSIIL Phishing URL dataset in the UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset). No new data were generated. The notebooks used for all experiments, with the fixed random seeds and the saved outputs of our runs, are available at https://github.com/bluntlycoded/phishdiag.

Acknowledgements

We thank the Department of Computer Science at Maitreyi College, University of Delhi, and the School of Computer Science and Engineering at VIT-AP University for their support and facilities, and our colleagues in both departments for their comments on early drafts.

We also owe a debt to the people who keep public phishing data available: Jose Nazario, whose phishing email collection has run for many years, the Apache SpamAssassin project, the creators of PhiUSIIL, and the compilers of the phishing email datasets on Kaggle and in the UCI Machine Learning Repository. Our thanks go as well to the developers of PyTorch, Hugging Face Transformers, scikit-learn and SHAP.

Author contributions

R.N. designed the diagnostic tests, built the models, ran the experiments, analysed the data and wrote the manuscript. S.S. supervised the study, interpreted the results, and reviewed and edited the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare no competing interests.

References

[1] Hosseinzadeh, M. et al. Improving phishing email detection performance through deep learning with adaptive optimization. *Sci. Rep.* **15**, 36724; 10.1038/s41598-025-20668-5 (2025).

[2] Tawfik, M., Abu-Ein, A. A., Abdelhaliem, A. H., Al-Sharo, Y. M. & Fathi, I. S. Explainable few-shot learning with modern BERT for detecting emerging phishing attacks using XF PhishBERT. *Sci. Rep.* **15**, 42821; 10.1038/s41598-025-27500-0 (2025).

[3] Alhuzali, A., Alloqmani, A., Aljabri, M. & Alharbi, F. In-depth analysis of phishing email detection: Evaluating the performance of machine learning and deep learning models across multiple datasets. *Appl. Sci.* **15**, 3396; 10.3390/app15063396 (2025).

[4] Anti-Phishing Working Group. *Phishing Activity Trends Report, 2nd Quarter 2024*. https://docs.apwg.org/reports/apwg_trends_report_q2_2024.pdf (APWG, 2024).

[5] Dalton, T. et al. PhreshPhish: A real-world, high-quality, large-scale phishing website dataset and benchmark. Preprint at https://arxiv.org/abs/2507.10854 (2025).

[6] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J. & Cavallaro, L. TESSERACT: Eliminating experimental bias in malware classification across space and time. In *Proc. 28th USENIX Security Symposium* 729–746 (2019).

[7] Salloum, S., Gaber, T., Vadera, S. & Shaalan, K. A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access* **10**, 65703–65727; 10.1109/ACCESS.2022.3183083 (2022).

[8] Benavides-Astudillo, E., Fuertes, W., Sanchez-Gordon, S., Nuñez-Agurto, D. & Rodríguez-Galán, G. A phishing-attack-detection model using natural language processing and deep learning. *Appl. Sci.* **13**, 5275; 10.3390/app13095275 (2023).

[9] Eze, C. S. & Shamir, L. Analysis and prevention of AI-based phishing email attacks. *Electronics* **13**, 1839; 10.3390/electronics13101839 (2024).

[10] Chang, C. M., Al-Andoli, M. N. & Zheng, C. Hybrid phishing detection model: Integrating BERT with TF-IDF for enhanced email security. *Int. J. Robot. Autom. Sci.* **7**, 43–48; 10.33093/ijoras.2025.7.3.6 (2025).

[11] Gholampour, P. M. & Verma, R. M. Adversarial robustness of phishing email detection models. In *Proc. 9th ACM International Workshop on Security and Privacy Analytics* 67–76; 10.1145/3579987.3586567 (2023).

[12] Reda, A., Taie, S. A. & Shaheen, M. E. Hybrid MLOps framework for automated lifecycle management of adaptive phishing detection models. *Sci. Rep.* **15**, 38478; 10.1038/s41598-025-23600-z (2025).

[13] Schmitt, M. & Flechais, I. Digital deception: Generative artificial intelligence in social engineering and phishing. *Artif. Intell. Rev.* **57**, 324; 10.1007/s10462-024-10973-2 (2024).

[14] Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30* 4765–4774 (2017).

[15] Sharma, A., Rani, S. & Shabaz, M. A comprehensive review of explainable AI in cybersecurity: Decoding the black box. *ICT Express* **11**, 1200–1219; 10.1016/j.icte.2025.10.004 (2025).

[16] Lim, B., Huerta, R., Sotelo, A., Quintela, A. & Kumar, P. EXPLICATE: Enhancing phishing detection through explainable AI and LLM-powered interpretability. Preprint at https://arxiv.org/abs/2503.20796 (2025).

[17] Lundberg, S. M. et al. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2**, 56–67; 10.1038/s42256-019-0138-9 (2020).

[18] Sanh, V., Debut, L., Chaumond, J. & Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. Preprint at https://arxiv.org/abs/1910.01108 (2019).

[19] Al-Subaiey, A. et al. Novel interpretable and robust web-based AI platform for phishing email detection. *Comput. Electr. Eng.* **120**, 109625; 10.1016/j.compeleceng.2024.109625 (2024).

[20] Klimt, B. & Yang, Y. The Enron corpus: A new dataset for email classification research. In *Proc. European Conference on Machine Learning (ECML)* 217–226 (2004).

[21] The Apache SpamAssassin public corpus. Available at https://spamassassin.apache.org/old/publiccorpus/.

[22] Nazario, J. Phishing email corpus. Available at https://monkey.org/~jose/phishing/.

[23] Mendes, P., Maia, E. & Praça, I. MeAJOR corpus: A multi-source dataset for phishing email detection. Preprint at https://arxiv.org/abs/2507.17978 (2025).

[24] Prasad, A. & Chandra, S. PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Comput. Secur.* **136**, 103545; 10.1016/j.cose.2023.103545 (2024).

[25] Prasad, A. & Chandra, S. PhiUSIIL Phishing URL (Website) [dataset]. *UCI Machine Learning Repository*, https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset (2024).

[26] Dodge, J. et al. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. Preprint at https://arxiv.org/abs/2002.06305 (2020).

[27] Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. In *Proc. 34th International Conference on Machine Learning* 3319–3328 (PMLR, 2017).

[28] Chen, F., Wu, T., Nguyen, V. & Rudolph, C. SoK: Systematizing generation, characteristics, and defenses in LLM-generated phishing. Preprint at https://arxiv.org/abs/2508.21457 (2025).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SNAPPDataAvailabilityStatement.txt