A Four-Signal Learned Fusion for Near-Real-Time Phishing URL Detection

Abena M. Darko
Dept. of Computer Information Systems
Georgia State University
Atlanta, USA
adarko3@gsu.edu

Benjamin M. Ampel
Dept. of Computer Information Systems
Georgia State University
Atlanta, USA
bampel@gsu.edu

***Abstract*—Phishing URLs remain the primary vector for attackers to steal confidential user information. Cyber analysts triaging URLs during a live incident cannot wait for a page to render or for repeated reputation lookups. URL-only classifiers are fast, but they judge each link in isolation and ignore the surrounding context of the URL. Reputation-assisted systems add that context yet are rarely implemented in URL classifiers for real-time use. In this study, we treat detection as a multi-signal decision and score each URL with (1) a character-level BERT model, (2) a CatBoost lexical model, (3) a VirusTotal reputation lookup, and (4) a Tranco popularity lookup. A learned meta-model combines the four scores into a classification decision. Across seven meta-model configurations, the fusion achieves a mean F1 of 99.2%, compared with 96.5% for the best single detector and 63.5% for plain score averaging. Ablation suggests that additional signals do not necessarily help further. Our proposed pipeline labels a URL in about 0.1 s, or 0.01 s once the deep model is removed, with an F1 score of 99% on a held-out test set. Of the four signals, domain popularity carries by far the most weight and has important implications for cybersecurity.**

***Keywords*—phishing URL detection, ensemble learning, signal fusion, meta-learning, real-time security analytics**

I. Introduction

Phishing URLs are a common first step in modern cyberattacks. How quickly an analyst can classify a suspicious link determines how quickly an incident is contained [1]. In a security operations center (SOC), a detector that adds seconds per URL or relies on a rate-limited external lookup adds latency that could lead to a successful attack. Good phishing detection is therefore a latency problem in addition to an accuracy one.

Two families of detectors are commonly implemented within organizational settings. First, URL-only models implement lexical or character-level deep learning. These models are fast and do not require network calls. However, they read a URL in isolation and miss external trust and reputation context that an analyst would check [4], [5]. Second, reputation- and context-assisted systems add that context. Few of them, though, are evaluated in combination with URL-only deep learning models. Reputation- and context-assisted systems often treat reputation as a standalone lookup or stop at one ensemble layer. Further, they rarely report the inference latency, which determines whether a system can run in real time.

There is also the question of whether richer fusion is worth its cost. A learned fusion of a few signals may be enough for deployment, with lower latency and complexity and little loss in accuracy. This paper empirically measures that trade-off. We combine deep, lexical, reputation, and popularity signals in a learned meta-model and ask how few signals are still sufficient for strong detection. We pose three research questions:

RQ1. Does a learned meta-model that fuses complementary URL signals outperform the individual detectors and a naive averaging baseline?

RQ2. Are more signals necessarily better for real-time phishing URL detection?

RQ3. Which signal families drive the decision, and what accuracy–latency trade-off do they imply for deployment?

This work makes three contributions. First, to our knowledge, it is the first to integrate deep URL representation learning, structured reputation and popularity signals, and a learned meta-ensemble layer into in single pipeline. Second, a systematic ablation over signal sets, families, and meta-models suggests that only four signals optimize the model, and URL domain popularity contributes significantly to these results. Third, we find a deep-model-free setup (CatBoost with Tranco) that retains most of the performance of the four-signal model while removing slow and network-dependent components, making near-real-time operation practical.

II. Related Work

A. URL-Only Deep Learning Detectors

URL-only detection has shifted towards learning directly from the raw link rather than relying on the hand-engineered features used by earlier classical machine learning systems. In particular, the literature has implemented joint character- and word-level convolutional representations, thereby establishing a strong URL-only baseline learned end-to-end from the string itself [4]. In parallel, classical models with lexical URL features have been shown to remain competitive and to be computationally cheaper to run [5].

The field then moved towards sequence learning. Recurrent networks were applied to classify URLs based on their character order [14], and the rise of transformer architectures [12] and character/word hybrid models [13] brought richer contextual representations to the same problem. For example, attention-augmented convolutional networks offered additional representations to sequence-based approaches [18]. Subsequent work has continued to refine this. Embedding URLs with word2vec and stacking temporal convolutional network layers to learn patterns directly from the string achieved a 98% F1 score [2]. Converting URL strings into character embeddings and then classifying them with convolutional networks achieved similar results [3]. Across this progression, the

detectors became steadily more accurate. They remain fast but share the limitation that they only ever see a URL string, with no signal of whether a domain is reputable or widely trusted.

B. Reputation- and Context-Assisted Detection

A second line of work grew out of the recognition that the URL string alone misses the external trust context an analyst would consult. This direction matured into multi-stage ensembles that fold in heterogeneous evidence (e.g., lexical features, Google-search cyber-threat-intelligence signals, and WHOIS registration data combined through TF-IDF, a random forest, and a multilayer perceptron), reaching a 96.8% F1 score without even inspecting the URL page [6]. However, the same line of work exposed the cost of leaning on external intelligence. The threat-intelligence features are of uncertain reliability, and these studies often do not report inference time or the time cost of scraping search results. The reliance on reputation also inherited the long-standing weaknesses of blocklists, which fill slowly and miss zero-hour phishing. Early empirical work found blocklists caught only a fraction of phishing URLs at hour zero [15]. That gap is what drove the field toward predictive [16] and streaming [17] approaches. This persistent tension between reported accuracy and real latency necessitates a lightweight reputation signal.

C. Hybrid and Ensemble Detectors

As single classifiers plateaued, the field turned to ensembles of classical learners to achieve greater robustness. A representative step was stacking XGBoost, LightGBM, and CatBoost with a random-forest meta-learner, which lifted accuracy to 96.8% over standalone classifiers (which scored 93.6–95.7% on the same lexical features) [7]. These ensembles advanced robustness but stayed in a single view of the input by combining lexical features of the URL but not deep representations, third-party reputation, or popularity. A parallel branch of the evolution instead widened the input past the URL string itself, combining URL and HTML features under XGBoost (96.76% accuracy) [19] and, further still, modeling page content and the DOM to reach 99.05% accuracy (Web2Vec [20]). Those accuracy gains, however, require the page to be fetched and rendered, which again is a latency concern.

D. Research Gaps

To the best of our knowledge, no prior work combines (a) deep URL representation learning, (b) structured reputation and trust signals, and (c) a learned meta-ensemble fusion layer in one pipeline and then asks how few of these signals are necessary for near-real-time operation. Table I summarizes this landscape. No prior system spans deep representation, lexical features, reputation, popularity, and a learned fusion layer at once, and none reports the inference latency required for real-time use. This paper unifies these layers and evaluates that question directly.

TABLE I CAPABILITY OF PRIOR URL DETECTORS VS. OURS (LEX. = LEXICAL, REP. = REPUTATION, POP. = POPULARITY, FUS. = LEARNED FUSION, LAT. = LATENCY REPORTED; ~ = PARTIAL)

System	Deep	Lex.	Rep.	Pop.	Fus.	Lat.
Remmide [2]	✓	–	–	–	–	–
Aldakheel [3]	✓	–	–	–	–	–
URLNet [4]	✓	–	–	–	–	–
Sahingoz [5]	–	✓	–	–	–	–
CTI-MURLD [6]	–	✓	✓	–	~	–
Sankaranarayanan [7]	–	✓	–	–	✓	–
This work	✓	✓	✓	✓	✓	✓

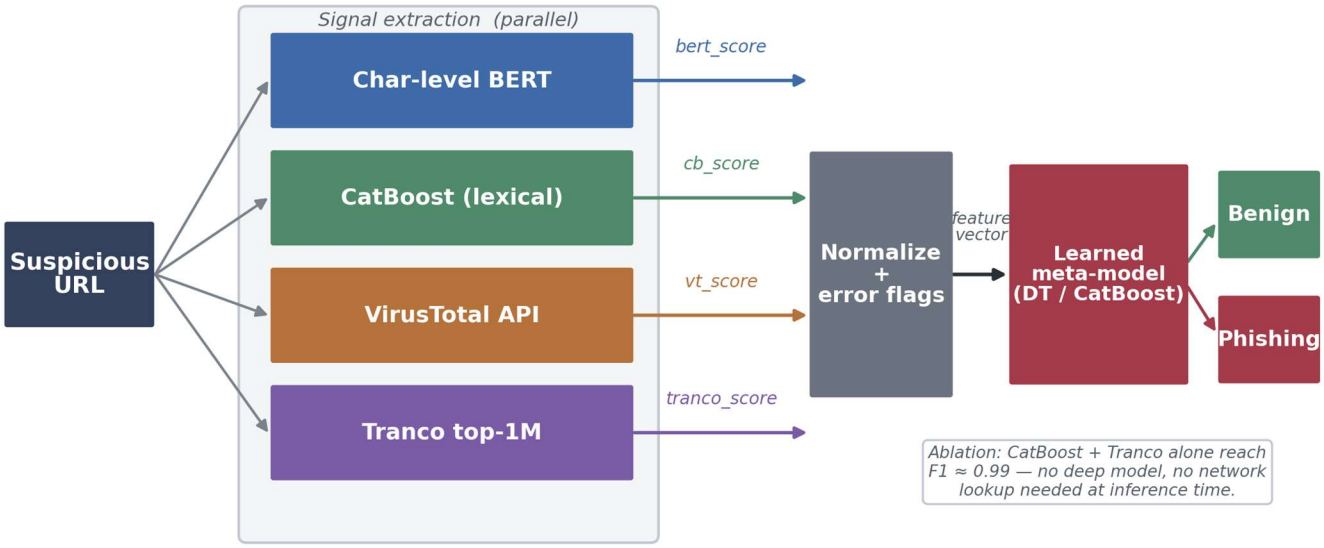

Fig. 1. Four-signal learned-fusion pipeline. A URL is scored in parallel by a character-level BERT model, a CatBoost lexical model, a VirusTotal reputation lookup, and a Tranco popularity lookup. Signals are normalized (with per-signal error flags) and fused by a learned meta-model. The ablation in Section V shows that the CatBoost + Tranco pair alone retains F1 ≈ 0.99, removing the deep model and the network lookup from the inference path.

III. Problem Formulation

Given a URL u, we extract a set of k scalar signals s(u) = [s_1(u), …, s_k(u)] produced by heterogeneous detectors (deep, lexical, reputation, popularity). Each signal is normalized to a common [0,1] benign-likelihood scale, and a binary error flag records whether its producing component executed successfully. Failed signals receive a neutral fallback. A learned meta-model f maps the signal vector to a benign/phishing decision. The objective is to learn f that maximizes detection quality (F1) while minimizing the size of the signal set and the end-to-end inference latency, since each additional signal may add a slow model or a network round-trip. The ablation in Section V searches over signal subsets to find the smallest signal vector that retains near-optimal F1.

IV. System Architecture and Method

We propose a five-step research design for this study. First, we collect recent and relevant benign and phishing URLs. Second, we train several base detectors on this dataset. Third, we calculate signals from these classifiers and external sources. Fourth, we implement a meta-model fusion framework to combine these signals. Finally, we conduct an ablation study to determine the best feature combination. Figure 1 demonstrates our pipeline.

A. Data and Preprocessing

We used two datasets for this study. First, we downloaded a large 808,042-row corpus of labeled benign and phishing URLs from Kaggle and PhishTank [8]. A second 48,812-URL Kaggle corpus [9] was used to supplement the dataset further. All data labels were normalized to a single convention (benign = 1, phishing = 0). For the meta-model corpus, we drew a roughly 60/40 benign-to-phishing mix to approximate realistic base rates and partitioned it into disjoint 40/30/30 training, validation, and test splits. There were 9623 URLs for training, and 7217 URLs for each of the validation and test splits. Signals were computed once per URL and cached so that every meta-model and ablation used identical inputs. Because reputation and popularity signals depend on external services, an error-flag mechanism governs missing-value handling (explained further below).

B. Base Detectors

Using our partitioned training dataset, we trained sixteen standalone classifiers (five character-level deep learning models and eleven classical machine-learning models) on lexical, structural, heuristic, and statistical URL features, then evaluated them on the validation split by F1, precision, recall, ROC-AUC, and inference time. From these, we picked one deep learning and one classical machine learning detector to anchor the fusion stage. Our deep learning model was the character-level BERT, which produces `bert_score`. Our classical machine learning model was CatBoost, which produces `cb_score`. We chose CatBoost as the lexical detector because it was the strongest classical model and the fastest to infer. We kept BERT as the deep learning model despite its higher latency to empirically evaluate its contribution to the latency tradeoff.

C. Signals and Signal Families

Our pipeline generates thirteen signals organized into four families: (1) *deep intrinsic* (BERT), (2) *lexical intrinsic* (CatBoost), (3) *external reputation* (VirusTotal), and (4) *external popularity* (Tranco). The reputation family is derived from VirusTotal's malicious, suspicious, and total engine counts. The popularity family is derived from a domain's presence and rank in the Tranco top-1M list [10], [11], generated on 18 October 2025 (models should always compare against the most up-to-date Tranco list), where a more popular domain is treated as close to benign (e.g., Amazon.com receives a 0.999 benign signal). Four additional signals are per-component error flags. If BERT, CatBoost, VirusTotal, or Tranco fails to produce a value, its flag is set, and the signal is replaced with a neutral fallback of 0.5, allowing the meta-model to learn how to handle missing evidence.

We define three nested sets of signals to assess the breadth required. The thirteen-signal set contains all signals and error flags. The nine-signal set drops the four error flags. The four-signal set retains one summary signal per family (Table II), which is the configuration we advocate for deployment.

TABLE II Four-Signal Feature Set: One Summary Signal Per Family

Signal	Family	Meaning (higher = more benign)
`bert_score`	Deep intrinsic	Benign-likelihood, character-level BERT classifier
`cb_score`	Lexical intrinsic	Benign-likelihood, CatBoost lexical classifier
`vt_score`	External reputation	Inverse VirusTotal detection rate
`tranco_score`	External popularity	Normalized popularity from Tranco rank

D. Fusion and Meta-Models

Each base component outputs a signal. We compared two fusion strategies. The first, *naive averaging*, averages the normalized signals and thresholds the mean, and cannot apply weights to the more informative signals. The second, *learned fusion*, trains a meta-model on the signal vectors to learn each signal's reliability and interactions. We evaluated seven meta-models to learn from the four signals: CatBoost (CB), Decision Tree (DT), Gradient Boosting (GB), Logistic Regression (LR), a Multilayer Perceptron (MLP), a Support Vector Machine (SVM), and XGBoost (XGB). The final meta-model is chosen based on F1, precision, and inference time.

E. Ablation Design

To answer RQ2 and RQ3, we retrained each meta-model on different signal sets and family combinations, then compared F1. The signal-set sweep contrasts the four-, nine-, and thirteen-signal sets to test whether the extra signals and error flags help. The combination sweep contrasts pairings of families (for example, intrinsic-only, intrinsic + popularity, reputation + popularity) to identify which families carry the decision and which can be dropped to cut latency.

V. Results and Discussion

A. Base Detectors (RQ1)

Among standalone models, the character-level recurrent detectors did best. Char-LSTM achieved F1 of 0.965, Char-BiLSTM of 0.963, and Char-RNN of 0.962. Overall standalone F1 ranged from 0.813 (Naïve Bayes) to 0.965 (Char-LSTM, Table III). The two anchors of our fusion stage sat lower on raw F1 but were chosen for different reasons. CatBoost reached F1 0.942 but scored the held-out set in only 0.019 s. Character-level BERT reached an F1 of 0.935 but required 86 s to score the same set. No single model achieved both top accuracy and low latency, which is why we fuse a fast lexical model, a deep learning model, and external context rather than trusting any one detector.

TABLE III REPRESENTATIVE STANDALONE BASE-DETECTOR PERFORMANCE

Model	Family	F1
Char-LSTM	Deep (char)	0.965
Char-BiLSTM	Deep (char)	0.963
Char-RNN	Deep (char)	0.962
CatBoost	Classical (lexical)	0.942
Char-BERT	Deep (char)	0.935
Naïve Bayes	Classical	0.813

B. Learned Fusion vs. Averaging (RQ1)

A learned meta-model beat averaging by a wide margin. Naive averaging reached only F1 of 0.635 on the four signals (and 0.613 on all thirteen). This result is because naïve averaging cannot down-weight noisy signals or use their interactions. Learned meta-models lifted F1 on the four-signal set to a mean of 0.992 across the seven meta-models, with CatBoost, Gradient Boosting, and XGBoost each at 0.995 (Table IV, top block). That is 3% above the best standalone detector (0.965) and more than 30% above naive averaging. Learned fusion thus clearly beats both the individual detectors and unweighted averaging.

C. How Many Signals Are Enough? (RQ2)

Expanding the four-signal set to the nine-signal and thirteen-signal sets changed the mean meta-model F1 by less than 0.01. The four error-flag signals added almost nothing for every meta-model except logistic regression, which is the most sensitive to feature scaling. The compact four-signal set is therefore enough. It matches the richer sets at lower complexity and faster inference. We use it for the combination analysis below.

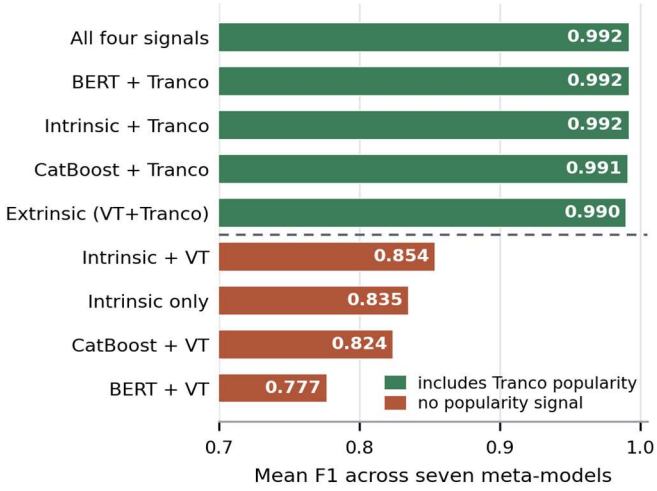

Fig. 2. Mean F1 by signal combination, colored by whether the combination includes the Tranco popularity signal. The mean F1 declines by about 15% along the dashed line once popularity is removed.

D. Which Signals Matter? (RQ3)

Table IV, visualized in Fig. 2, reveals a sharp structure. Every combination that includes Tranco popularity scores at or above F1 0.990 on average, whereas every combination without it falls to 0.78–0.85, a gap of about fifteen 15%. Domain popularity is therefore the most informative signal by far: pairing it with either intrinsic model (BERT + Tranco or CatBoost + Tranco) recovers nearly all of the four-signal performance. VirusTotal reputation, by contrast, adds little once popularity is present and is the weakest signal on its own; the BERT + VT and CatBoost + VT pairings are unstable, with logistic regression collapsing to F1 0.466 on CatBoost + VT. We read this as VirusTotal's uneven coverage of freshly registered phishing domains and its correlated engine verdicts, which add noise rather than independent evidence.

Across meta-models, CatBoost is the strongest learner (F1 0.995 with all signals, falling to 0.866 with intrinsic-only signals), with Gradient Boosting and XGBoost close behind (0.935 and 0.934 on average). CatBoost is, however, more expensive to train and to query than Decision Tree, Logistic Regression, or MLP, which were the fastest meta-models at inference; SVM was by far the slowest. Balancing accuracy against inference cost, we select *Decision Tree* and *CatBoost* as finalists, the most consistent models across the ablations, and drop SVM, whose latency does not justify its accuracy.

TABLE IV MEAN F1 BY SIGNAL COMBINATION ACROSS THE SEVEN META-MODELS

Signal combination	CB	DT	GB	LR	MLP	SVM	XGB	Avg
All four signals	0.995	0.990	0.995	0.991	0.991	0.991	0.995	0.992
BERT + Tranco	0.993	0.991	0.994	0.990	0.990	0.990	0.992	0.992
Intrinsic + Tranco	0.993	0.990	0.994	0.991	0.991	0.991	0.994	0.992
CatBoost + Tranco	0.994	0.991	0.993	0.987	0.989	0.989	0.994	0.991
Extrinsic (VT + Tranco)	0.991	0.991	0.992	0.987	0.987	0.989	0.991	0.990
Intrinsic + VT	0.897	0.874	0.895	0.689	0.885	0.844	0.893	0.854
Intrinsic only	0.866	0.853	0.867	0.708	0.854	0.836	0.862	0.835
CatBoost + VT	0.886	0.908	0.891	0.466	0.876	0.856	0.888	0.824
BERT + VT	0.799	0.747	0.798	0.692	0.800	0.805	0.796	0.777
Average	**0.935**	**0.926**	**0.935**	**0.833**	**0.929**	**0.921**	**0.934**	**0.916**

This answers RQ3. CatBoost + Tranco reaches an F1 of 0.991, within 0.001 s of all four signals (mean 0.992), so the deep model and the VirusTotal network call can be removed from the inference path entirely. CatBoost scores a URL in 0.019 s, and a Tranco lookup is an in-memory rank query. Dropping the BERT forward pass, which dominates per-URL latency, along with the rate-limited VirusTotal call, cuts end-to-end time from ≈0.1 s to ≈0.01 s per URL at almost no cost in accuracy. We recommend this setup for near-real-time triage and keep the full four-signal model for cases where accuracy matters more than latency.

E. End-to-End Pipeline Evaluation

We deployed the complete pipeline and evaluated the two finalist meta-models (Decision Tree and CatBoost, with Gradient Boosting as a strong third) under live signal enrichment on a held-out test set (N = 72), with a separate 150-URL sample for latency, robustness, and error analysis. All three models classified this small test set almost perfectly (Table V): CatBoost and Gradient Boosting achieved F1 1.000, and the lightweight Decision Tree achieved F1 0.988 (precision 1.000, recall 0.977).

TABLE V END-TO-END PERFORMANCE OF THE FINALIST META-MODELS ON THE FOUR-SIGNAL PIPELINE

Meta-model	Acc	P	R	F1	s/URL [a]
Decision Tree	0.986	1.000	0.977	0.988	0.099
CatBoost	1.000	1.000	1.000	1.000	0.127
Gradient Boosting	1.000	1.000	1.000	1.000	0.126

[a] *s/URL denotes end-to-end latency per URL with cached VirusTotal enrichment.*

Latency. The full four-signal pipeline classified a URL in 0.10–0.13 s end to end VirusTotal. This budget is dominated by the character-level BERT forward pass (≈0.09–0.11 s/URL); the cached VirusTotal lookup added only ≈0.001 s, the Tranco lookup was effectively free (≈0.00001 s), and the meta-model decision itself cost ≈0.001–0.005 s. Because the ablation showed the deep model is removable at almost no accuracy cost (CatBoost + Tranco, Section V), a BERT-free deployment cuts end-to-end latency to roughly 0.01 s/URL, well within real-time triage budgets. Running the pipeline uncached, with VirusTotal enrichment being fetched live introduces significantly higher end-to-end latency (≈14.909-15.009 s/URL).

Robustness. Removing signals at inference supported the ablation: dropping VirusTotal left F1 unchanged for all three meta-models (Decision Tree held F1 0.994 with and without VT), whereas dropping Tranco collapsed recall to zero: without the popularity signal, the meta-models caught no phishing URLs at all. Popularity does the heavy lifting; reputation is the most expendable signal.

Error analysis. On the 150-URL sample, CatBoost and Gradient Boosting produced no misclassifications, and Decision Tree produced a single false negative (0.67% error). Errors were rare, and the ones that occurred were conservative misses rather than false alarms. We still read these scores with caution: 72 URLs is too small a test set to separate a near-perfect detector from one that has saturated an easy sample. We treat the larger ablation (mean F1 0.92–0.99 across meta-models, Table IV) as the more conservative estimate of how well the pipeline generalizes, and Table V as evidence that the assembled system works, not as a claim of perfect accuracy.

F. Limitations and Threats to Validity

Several limitations qualify these results. First, the reputation and popularity signals depend on external services. The error-flag mechanism degrades gracefully, but sustained outages or rate limits would shrink the set of signals available at inference time. Second, popularity is double-edged. Because Tranco rank correlates with benignity, an attacker abusing a high-ranking hosting or shortening domain could theoretically exploit it. Thus, popularity should complement, not replace, content inspection. Third, the end-to-end evaluation used small held-out samples, so the strong scores should be treated as an upper bound. Finally, the public-feed datasets may not match a given deployment's label quality or time distribution, and Tranco ranks drift. Periodic re-evaluation on live traffic remains necessary.

VI. CONCLUSION

We presented a four-signal learned fusion for near-real-time phishing URL detection that combines character-level deep learning (BERT), lexical machine learning (CatBoost), external reputation (VirusTotal), and domain popularity (Tranco). Treating detection as a multi-signal decision and fusing the signals with a learned meta-model raised F1 to a mean of 0.992 (up to 0.995), well above the best standalone detector (0.965) and naive averaging (0.635). Ablation showed that more signals do not always help: the compact four-signal set matched a thirteen-signal version, domain popularity did most of the work, and a deep-model-free CatBoost + Tranco pairing held F1 0.991 while dropping the slowest and most network-dependent parts of the pipeline. Future work will validate the pipeline on live URL streams, measure end-to-end latency under operational load, and further reduce its dependence on external enrichment, ensuring the detector degrades gracefully when reputation or popularity lookups are unavailable.

. REFERENCES

[1] J. Hong, "The state of phishing attacks," Commun. ACM, vol. 55, no. 1, pp. 74–81, Jan. 2012, doi: 10.1145/2063176.2063197.

[2] M. A. Remmide, F. Boumahdi, N. Boustia, C. L. Feknous, and R. Della, "Detection of phishing URLs using temporal convolutional network," Procedia Comput. Sci., vol. 212, pp. 74–82, 2022, doi: 10.1016/j.procs.2022.10.209.

[3] E. A. Aldakheel, M. Zakariah, G. A. Gashgari, F. A. Almarshad, and A. I. Alzahrani, "A deep learning-based innovative technique for phishing detection in modern security with uniform resource locators," Sensors, vol. 23, no. 9, p. 4403, Apr. 2023, doi: 10.3390/s23094403.

[4] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," arXiv:1802.03162, 2018.

[5] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," Expert Syst. Appl., vol. 117, pp. 345–357, 2019, doi: 10.1016/j.eswa.2018.09.029.

[6] M. Alsaedi, F. A. Ghaleb, F. Saeed, J. Ahmad, and M. Alasli, "Cyber threat intelligence-based malicious URL detection model using ensemble learning," Sensors, vol. 22, no. 9, p. 3373, 2022, doi: 10.3390/s22093373.

[7] S. Sankaranarayanan, A. T. Sivachandran, A. S. Mohd Khairuddin, K. Hasikin, and A. R. Wahab Sait, "An ensemble classification method based on machine learning models for malicious uniform resource locators (URL)," PLOS ONE, vol. 19, no. 5, p. e0302196, 2024, doi: 10.1371/journal.pone.0302196.

[8] H. S, "Phishing and legitimate URLs dataset," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/harisudhan411/phishing-and-legitimate-urls/data

[9] T. Dhrubang, “Real-world phishing URL classification data,” Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/dhrubangtalukdar/real-world-phishing-url-classification-data/data

[10] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, “Tranco list ID 5XJGN,” Oct. 18, 2025. [Online]. Available: https://tranco-list.eu/list/5XJGN/full

[11] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, “Tranco: A research-oriented top sites ranking hardened against manipulation,” in Proc. NDSS, San Diego, CA, USA, 2019, doi: 10.14722/ndss.2019.23386.

[12] P. Maneriker, J. W. Stokes, E. G. Lazo, D. Carutasu, F. Tajaddodianfar, and A. Gururajan, “URLTran: Improving phishing URL detection using transformers,” in Proc. IEEE Mil. Commun. Conf. (MILCOM), 2021, doi: 10.1109/MILCOM52596.2021.9653028.

[13] F. Tajaddodianfar, J. W. Stokes, and A. Gururajan, “Texception: A character/word-level deep learning model for phishing URL detection,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), 2020, pp. 2857–2861, doi: 10.1109/ICASSP40776.2020.9053670.

[14] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, “Classifying phishing URLs using recurrent neural networks,” in Proc. APWG Symp. Electron. Crime Res. (eCrime), 2017, doi: 10.1109/eCrime.2017.7945048.

[15] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, “An empirical analysis of phishing blacklists,” in Proc. 6th Conf. Email Anti-Spam (CEAS), 2009.

[16] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, “PhishNet: Predictive blacklisting to detect phishing attacks,” in Proc. IEEE INFOCOM, 2010, doi: 10.1109/INFCOM.2010.5462216.

[17] S. Marchal, J. François, R. State, and T. Engel, “PhishStorm: Detecting phishing with streaming analytics,” IEEE Trans. Netw. Service Manag., vol. 11, no. 4, pp. 458–471, 2014, doi: 10.1109/TNSM.2014.2377295.

[18] M. Nanda, M. Saraswat, and P. K. Sharma, “Enhancing cybersecurity: A review and comparative analysis of convolutional neural network approaches for detecting URL-based phishing attacks,” E-Prime – Adv. Electr. Eng., Electron. Energy, vol. 8, p. 100533, 2024, doi: 10.1016/j.prime.2024.100533.

[19] A. Aljofey, Q. Jiang, A. Rasool, H. Chen, W. Liu, Q. Qu, and Y. Wang, “An effective detection approach for phishing websites using URL and HTML features,” Sci. Rep., vol. 12, no. 1, 2022, doi: 10.1038/s41598-022-10841-5.

[20] J. Feng, L. Zou, O. Ye, and J. Han, “Web2Vec: Phishing webpage detection method based on multidimensional features driven by deep learning,” IEEE Access, vol. 8, pp. 221214–221224, 2020, doi: 10.1109/ACCESS.2020.3043188.