Journal of Ubiquitous Systems & Pervasive Networks
Volume 17, No. 1 (2022) pp. 35-42

Towards Performance of NLP Transformers on URL-Based Phishing Detection for Mobile Devices

Hossein Shirazi [a]*, Katherine Haynes [b], Indrakshi Ray [a]

[a]*Computer Science Department, Colorado State University, Fort Collins, CO, USA*
[b]*Cooperative Institute for Research in the Atmosphere, Colorado State University, Fort Collins, CO, USA*
{Hossein.Shirazi, Katherine.Haynes, Indrakshi.Ray}@colostate.edu

Abstract

Hackers are increasingly launching phishing attacks via SMS and social media. Games and dating apps introduce yet another attack vector. However, current deep learning-based phishing detection applications do not apply to mobile devices due to the computational burden. We propose a lightweight phishing detection algorithm that distinguishes phishing from legitimate websites solely from URLs to be used in mobile devices. As a baseline performance, we apply Artificial Neural Networks (ANNs) to URL-based and HTML-based website features. A model search results in 15 ANN models with accuracies >96%, comparable to state-of-the-art approaches. Next, we test the performance of deep ANNs on URL-based features only; however, all models perform poorly with the highest accuracy of 86.2%, indicating that URL-based features alone are not adequate to detect phishing websites even with deep ANNs. Since language transformers learn to represent context-dependent text sequences, we hypothesize that they will be able to learn directly from the text in URLs to distinguish between legitimate and malicious websites. We apply three state-of-the-art deep transformers (BERT, ELECTRA, and RoBERTa) for phishing detection. Testing custom and standard vocabularies, we find that pre-trained transformers available for immediate use (with fine-tuning) outperform the model trained with the custom URL-based vocabulary. In addition, we test a thinner BERT transformer which is suitable for lightweight devices like mobiles, called MobileBERT. Our results emphasize that evaluation metrics of this model are competitive to other models in this study, yet the testing time is significantly less, making this model a choice for embedding phishing detection algorithms in mobile phones. Using pre-trained transformers to predict phishing websites from only URLs has five advantages: 1) requires little training time (230 to 320 s), 2) is more easily updatable than feature-based approaches because no pre-processing of URLs is required, 3) is safer to use because phishing websites can be predicted without physically visiting the malicious sites, 4) is easily deployable for real-time detection and is applicable to run on mobile devices, and 5) using a mobile specific transformer yields comparable performance and predicts 3 times faster than the other transformer models in this study.

Keywords: · *Social Engineering Attack* · *Phishing Detection* · *Deep learning* · *Transformers* · *Mobile Application*

1. Introduction

Phishing is the top Internet crime by victim count as per the FBI [1], resulting in a $54 million loss in 2020. Phishing attacks are increasingly being launched via SMS text, social media, gaming, and dating applications. People tend to be less careful when using their mobile devices and are therefore more vulnerable [2]. A lightweight phishing detection approach that can be installed in mobile devices is very much needed.

Supervised deep learning appears to be a promising approach for phishing detection [3–5]. Machine learning requires a large volume of training data; such data may be unavailable and dataset creation is labor-intensive. For creating phishing datasets, one must visit the malicious websites, understand the code, and

*Corresponding author. Tel.: +19708898840
Fax: ; E-mail: Hossein.Shirazi@colostate.edu

DOI: 10.5383/JUSPN.17.01.005

extract the relevant features. Moreover, as attackers modify their approach, the process must be repeated. Phishing detection using only URL information alleviates these problems, thus recent development in language transformers has motivated us to look into phishing detection using only URLs.

In this work, we use a dataset developed by Shirazi et al. [6] which consists of 48 URL and HTML-based website features, and we test the ability of a deep classification Artificial Neural Network (ANN) to detect phishing websites. First, we test ANNs on URL-based features only. Even after an extensive architecture and hyper-parameter search, this method did not perform well. We next use URL and HTML-based features and obtained an accuracy of 97% with the top-performing model (ANNF), comparable to state-of-the-art approaches. We use this as a baseline for our transformer-based model.

Transformers are deep learning Natural Language Processing (NLP) models designed to handle sequential text for translation and summarization tasks. The well-known Bidirectional Embedding Representations from Transformers (BERT [7]) has been used in the past eight months to detect spam emails [8–10]. Following on the success of transformers in detecting phishing emails, we hypothesize that they can take URLs directly and parse out contextual information that indicates if a website is legitimate or malicious. Utilizing transformers to detect phishing leverages data to learn the textual features associated with websites, rather than relying on pre-determined features that are observed by experts and require manual efforts. We apply BERT and two other well-known transformers RoBERTa (A Robustly Optimized BERT Pretraining Approach) [11] and ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately [12]) to URL-based phishing detection.

To further reduce training time and computing resources while utilizing these state-of-the-art deep learning approaches, we use pre-trained models that are available for immediate use on language classification tasks, eliminating the need for extensive training and only requiring fine-tuning of the top model layers. In addition to applying transfer learning, we custom-train BERT using a URL-specific vocabulary. Our observations on testing pre-trained and custom models indicate that pre-trained transformers perform well, correctly identifying 96% of the websites tested at a cost of ∼8 minutes to fine-tune. Since these models take text strings as the input, they can be applied directly to URLs, eliminating pre-processing features and making them easily deployable for real-time detection on mobile devices.

Key Contributions. The contributions of this work are as follows:

(i) We compare the performance of seven state-of-the-art deep learning approaches.

(ii) We show that ANNs can detect phishing from URL and HTML-based features with high performance.

(iii) We demonstrate that using ANNs on URL-based features alone is not well-suited to detect phishing websites.

(iv) We illustrate that NLP models can be applied to website phishing detection solely using URL strings, revealing that pre-trained transformers provide phishing detection with similar performances to other approaches but with four distinct advantages: 1) it requires minimal training time (less than 8 minutes); 2) it is easily updatable as it does not require feature collection, determination, and pre-processing (thus, even if attackers change their strategy it will still work); 3) it is safer to use, as it removes the requirement of visiting malicious sites; and 4) it is easily deployable for real-time detection and can be used in mobile devices.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 provides an overview and Section 4 discusses the details of our approach: Section 4.1 defines Task 1, which uses both URL- and HTML-based features for phishing detection; Section 4.2 discusses Task 2, which exclusively uses URL-based features; and Section 4.3 elaborates on Task 3, using NLP transformers. Section 5 evaluates our results against state-of-the-art, and Section 6 concludes the paper.

2. Related Work

2.1. Content-Based Phishing Detection

Website phishing detection using deep learning typically focuses on content and/or URLs to define and obtain optimal features to eliminate overfitting. Zhu et al. [13] proposed the Optimal Feature Selection Neural Network (OFS-NN) to combine optimal feature selection into an ANN framework by introducing a feature validity value. Zhu et al. [5] combined decision trees and local search methods to select optimal features, clustering to remove duplicates, and parameter optimization in their model DTOF-ANN (Decision Tree and Optimal Features based Artificial Neural Network). Next, Saravanan et al. [4] used a genetic algorithm to reduce the dimensionality of selected features. Once the optimal features are determined, the proposed framework uses the Adaptive Resonance Theory Mapping (ARTMAP) neural network classifier to determine if a website is legitimate. Finally, Yang et al. [14] utilized synthetic samples in a novel approach to phishing detection based on the Non-Inverse matrix Online Sequence Extreme Learning Machine (NIOSELM), using an adaptive sampling algorithm to generate synthetic minority class samples in order to avoid biases and using an autoencoder to denoise the data and reduce the data dimensionality.

2.2. URL-based Phishing Detection

Another strategy to detect phishing websites directly uses URLs. While a number of methods have combined website content with URL strings (e.g. [15–19]), here we outline four models that solely utilize URLs. In 2017, Saxe et al. [20] proposed a Convolutional Neural Network (CNN) on raw character input embedded as multi-dimensional vectors to find patterns in characters, which are then fed into three fully connected ANN layers. In the past eighteen months alone, we have identified three more such studies. First, Huang et al. [3] proposed a deep learning approach that uses a CNN module for character-level extraction fused with a Recurrent Neural Network (RNN) for word-level extraction. Second, Xiao et al. [21] proposed a CNN combined with a multi-head self-attention approach that learns URLs' inner structures, exploiting character relationships. Third, Wei et al. [22] used one-hot encoding combined with an embedding layer in a CNN to focus on sensitizing the network to detect URL distortions. While solid performers, these methods are more computationally expensive than our proposed method.

Transformers have been used for phishing email detection, or detecting emails that are spreading phishing attacks. For example, Lee et al. [23] proposed a hybrid approach that considers both email's content and context features from email headers. In a different study, Thapa et al. [24] used federated learning in anti-phishing email detection by transformers.

Xu [25] introduced a transformer-based phishing detection model that is very similar to OpenAI's GPT model. This proposed model has outperformed six existing classification detection models with 97.3% of detection accuracy. Maneriker et al. [26] performed a comprehensive analysis of transformer models on the phishing URL detection task. Authors compared standard and domain-specific masked language models to fine-tuned BERT and RoBERTa models and proposed URLTran. This proposed model uses transformers to significantly improve the performance of phishing URL detection, and the authors considered classical adversarial black-box phishing attacks such as homoglyphs and compound word splits to improve the robustness of URLTran.

This work is an extension to our previous work [27] on URL-based phishing detection. Previously we used NLP transformers to detect phishing websites from legitimate ones solely based on the URL. Two transformer models of BERT and ELECTRA have been evaluated. In this work, we added two more transformers of RoBERTa and Mobile-BERT. The latest one is a lightweight specific transformer that is suitable for mobile devices.

3. Approach Overview and Methodology

While extracting both content and URL features result in high-performing phishing detection, it is laborious and time-consuming. Here we investigate using only URL-based information to yield reliable detection of malicious websites. Specifically, we determine if deep classification ANNs can be used for this task or if transformers are necessary to process the text and succinctly find appropriate features. In addition, we check whether mobile-specific transformers have lower testing time but similar performance. For this, we have defined four tasks:

- *Task 1:* Determine how deep classification ANNs perform on content and URL-based website features.

  We utilize both URL features and website content to provide a baseline top performance expected using a feature-based approach. Using a synthetically-extended dataset designed to make phishing detection more robust against adversarial attacks [28, 29], we perform a guided hyper-parameter search to find the best ANN.
- *Task 2:* Investigate how well deep classification ANNs perform on URL-only website features.

  Using only URL-based features, we again perform a guided search to find the best ANN and compare our results to Task 1, determining the importance of website content in feature-based deep learning approaches.
- *Task 3:* Explore the use of transformers to detect phishing directly on URL strings.

  We apply BERT, ELECTRA, and RoBERTa to website phishing detection using transfer learning, fine-tuning them on website URL strings. Because they learn patterns in text, we hypothesize that they will detect phishing websites solely from the URL, saving time in collecting and preparing website feature data as well as significantly reducing training time and resources.
- *Task 4:* Evaluate a light-weight version of the BERT transformer suitable for running on mobile devices.

  We apply MobileBERT, a thin version of the BERT transformer. Empirical studies show that MobileBERT is 4.3 times smaller and 5.5 times faster than the BERT-Base version. Similar to Task 3, we fine-tune this model on website URL strings and evaluate the performance. Additionally, we evaluate how much faster this mobile-specific transformer is in comparison with other models.

Table 1. Summarization of the datasets used in this study. The number of phishing (Phi.) and legitimate (Leg.) websites are shown, along with the number of features (Fea.). The URL-Only and URL-Vocab datasets are used only with transformers and thus do not require any feature extraction.

Name	Author	Phi.	Leg.	Fea.
DS-Cnt-Ftrs	Shirazi et al. [28, 29]	10K	10K	48
DS-URL-Ftrs	Shirazi et al. [28, 29]	10K	10K	31
DS-URL-Only	This study	10.9K	10.9K	N/A
DS-URL-Vocab	This study	0	1.7M	N/A

3.1. Datasets Used in Experiments

We use the following four datasets:

- **DS-Cnt-Ftrs:** In 2018, Tan et al. [30] created a dataset for phishing detection comprised of 48 URL-based and HTML-based features extracted from 5000 phishing webpages and 5000 legitimate webpages. The web pages included in this dataset were downloaded from January to May 2015 and from May to June 2017. The phishing webpage sources are *PhishTank.com* and *OpenPhish.com*, and the legitimate webpage sources are *Alexa* and *CommonCrawl.org*. In 2020, Shirazi et al. [6] extended this dataset using adversarial sample generation. Using the same features, they used an autoencoder to generate samples mimicking existing websites while containing features designed to bypass trained machine learning phishing detection models. This extended the dataset by adding 10,000 phishing web pages and 10,000 legitimate web pages, and we use this version of the dataset.
- **DS-URL-Ftrs:** We select the URL-related features from the DS-Cnt-Ftrs dataset and create a subset of it. This new dataset has 31 out of the 48 features available in the original dataset.
- **DS-URL-Only:** To investigate deep-learning approaches on URLs directly, we create a dataset of 10,955 phishing URLs from *PhishTank.com* and 10,955 legitimate URLs from *CommonCrawl.org*. The phishing URLs include sites discovered until November 27, 2020, when we downloaded the dataset. The legitimate URLs include websites from randomly selected files from the period of January 2018 to November 2020.
- **DS-URL-Vocab:** To use a URL-specific vocabulary, we create a dataset of 1,730,754 websites from *CommonCrawl.org*. Since the *CommonCrawl.org* corpus contains petabytes of data collected since 2008, to include a variety of websites, we randomly selected URLs from January 2018 until November 2020.

In all experiments, the training and testing subsets consisting of 80% and 20% of the data, respectively.

3.2. ANN Optimization for Feature-Based Phishing Detection

In order to optimize classification ANNs for feature-based phishing detection, we perform a guided model architecture and hyperparameter search using Hyperopt [31]. We conduct searches using the Tree of Parzen Estimators (TPE) algorithm, a sequential model-based optimization approach that sequentially constructs models to approximate the performance of hyperparameters based on historical measurements [32]. We divide the search space into two areas: input data options during pre-processing and model options. We search for the optimal number of features to include in the ANN, ranging from four to the total number of features. The features included per selected number are pre-calculated using the chi-square statistic on the original dataset. The ANN model architecture search includes options for the number of epochs (number of times the learning algorithm iterates through the entire training set), the optimizer (a method used to update model weights), the learning rate (the rate at which weights can change), the momentum (influence of previous changes), and the number of model layers.

We perform all ANN experiments with Python using TensorFlow on Google Colab with a GPU. For each optimization, we run 100 model searches. For each search, we use 3-fold cross-validation during training. One 100-model search takes $\sim$10 hours, with each model taking $\sim$3-5 minutes to train, depending on the size of the model.

3.3. Deep Language Processing Models

In order to thoroughly utilize current pre-trained transformers, we test four state-of-the-art deep language processing models: BERT, Mobile-BERT, ELECTRA, and RoBERTa.

3.3.1. BERT and Mobile-BERT

As described by Devlin et al. [7], BERT consists of a multi-layer bidirectional encoder that learns contextual relationships. BERT's framework consists of two steps: (i) pre-training, during which the model is trained using unlabeled data on masked language modeling and next sentence prediction tasks, and (ii) fine-tuning, during which the model is initialized with the pre-trained parameters that are adjusted using task-specific labeled data.

We focus on four pre-trained BERT models.[1] The first is BERT-Base (uncased), which is a 12-layer model with 110 million parameters. The second is BERT-BaseC (cased), which is the same as BERT-Base but uses case-sensitive tokens. Third is BERT-Large (uncased), which uses 24-layers and 340 million parameters. And last is MobileBERT, which is a thin version of BERT-LARGE and has 24 layers but with only 24.5 million parameters.

To compare using models pre-trained on a generic corpus versus a URL-specific vocabulary, we create a custom BERT model trained on an uncased URL-generated corpus. Using the DS-URL-Vocab dataset, we tokenize each URL using the SentencePiece tokenizer, breaking them apart by the forward-slashes and saving punctuation, numbers, and text groupings

[1] BERT pre-trained models and code are available in the Transformers library [33] and at https://github.com/google-research/bert.

Table 2. Transformer summarization, showing the pre-trained models tested along with their layers and number of parameters.

Transformer	# Layers	# Parameters
BERT_BASE	12	109M
BERT_C	12	109M
BERT_LARGE	24	334M
Mobile_BERT	24	24.5M
ELECTRA_Base	12	110
ELECTRA_Base	24	335M
RoBERTa	12	110M

(using lowercase text only). We use the maximum vocabulary size (128,000), a maximum sequence length of 128, and the same hyper-parameter setup as proposed by Devlin et al. [7]. We train the model for one million steps, which took 60 hours.

3.3.2. ELECTRA

ELECTRA is an efficient pre-training approach that uses replaced token detection described by Clark et al. [12]. ELECTRA consists of a generator and a discriminator. After mapping input tokens into contextualized sequences, the generator learns to predict masked-out tokens while the discriminator learns how to distinguish tokens in the data from tokens that have been replaced by generator samples. Like BERT, ELECTRA can be fine-tuned for specific tasks.

ELECTRA currently has three released pre-trained models: ELECTRA-Small with 12 layers and 14 million parameters, ELECTRA-Base with 12 layers and 110 million parameters, and ELECTRA-Large with 24 layers and 335 million parameters.[2] The vocabulary and pre-training datasets are the same as used for BERT.

3.3.3. RoBERTa

RoBERTa builds on BERTs language masking strategy, wherein the system learns to predict intentionally hidden sections of text within otherwise unannotated language examples. RoBERTa modifies key hyperparameters in BERT, including removing BERTs next-sentence pretraining objective and training with much larger mini-batches and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance. RoBERTa begins by training BERT models with the same configuration as BERT-BASE with 12 layers and 110M parameters.

3.3.4. Transformer Optimization

Table 2 summarizes the number of layers, parameters in each model we used in this study. We perform all of the transformer-based experiments with Python using TensorFlow on Google Colab with a TPU. For all of the transformers, we fine-tune the top

[2] ELECTRA pre-trained models and code are available in the Transformers library [33] and at https://github.com/google-research/electra.

model layers following the procedure outlined by TensorFlow[3], using the sequence classifier methodology and the URL training dataset (Section 3.1). The specific parameters used for fine-tuning each of the pre-trained models is discussed in Section 4.3.

4. Approach Details

4.1. Task 1: Website URL-Based and HTML-Based Feature Phishing Detection

After performing the model search using the feature dataset, the top 40 models all achieve testing accuracies $> 91\%$, with the top 15 models achieving accuracies $> 96\%$. As indicated by 0.98 normalized true-positive and 0.97 true-negative values of the top model (ANNF), the majority of the samples are correctly predicted. The 0.03 false-negative is slightly higher than the 0.02 false-positive, indicating a higher tendency to predict a website as legitimate when it is malicious.

Comparisons to the classifiers tested by Shirazi et al. [6] are shown in Figure 1. The ANNF model yields similar performances to the other top-performing classifiers, GB and RF. Interestingly, the ANNF precision is higher than the recall, while the opposite is true for the other two top performers. These relationships show that ANNF has a higher positive predictive value and thus has a lower tendency to predict a website is malicious when it is not; however, ANNF's lower recall indicates that it misses more phishing websites overall. These results show that using deep neural networks is comparable to both GB and RF classifiers for detecting phishing websites.

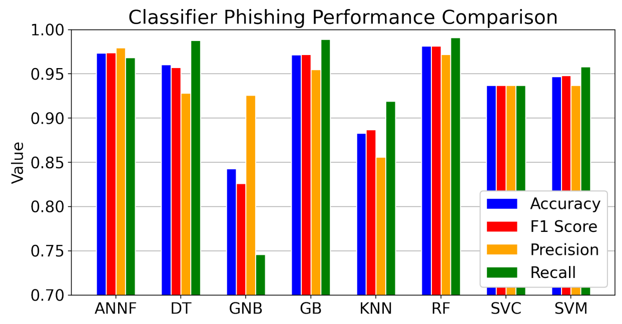

Fig. 1: Phishing detection score comparisons between the neural network in this study (ANNF) and other classifier models tested by [6]. DT=Decision Tree, GNB=Gaussian Naïve Bayes, GB=Gradient Boosting, KNN=k-Nearest Neighbor, RF=Random Forest, SVC=Support Vector Machine (linear), and SVM=Support Vector Machine (gaussian).

4.2. Task 2: URL-Based Feature Only Phishing Detection

To test how ANNs perform on features obtained only from URLs, we transform the URL dataset into 31 URL features. We use all of the URL features from the feature dataset and added features for the number of each different punctuation symbol and counts of the numbers, lowercase and uppercase letters. Even with a 100-model search, all models perform poorly. The performance is substantially lower than when website content is included, with the top model only achieves 86.2% accuracy and normalized true-positive and true-negative values of 0.84 and 0.95, respectively. While both the true-positive and true-negative predictions are lower compared to ANNF, the correct prediction of phishing websites suffers more, indicating that a substantial number of malicious sites will go undetected. While performance may increase with the selection of different URL features, it seems unlikely that any additional features based solely on URL will account for the drop in performance by removing website content. This result indicates that website content (via HTML-based features) is vital in feature-based phishing detection and that even deep-learning ANNs are not well-suited to detect phishing using pre-determined URL features alone.

4.3. Task 3: Transformer Phishing Detection

We perform five experiments using BERT, ELETRA, RoBERTa, and MobileBERT. First, we train a BERT base model using a URL-specific uncased vocabulary (BERT-URL). Next, we try a hybrid approach and add the BERT-URL predictions (0=legitimate, 1=phishing) into an ANN that utilizes only URL-based information as in Task 2, again performing a guided search to find the best model (BERT-ANN). Last, we perform experiments fine-tuning the BERT-Base, BERT-BaseC, and BERT-Large pre-trained models using the URL training dataset. For each, we run a series of experiments to tune the parameters. For BERT-Base, we achieve the highest accuracy using a learning rate of 8×10^{-5}, 2 epochs, and a batch size of 128, taking 319 s to fine-tune. For BERT-BaseC, we use the same learning rate and batch size but performed 4 training epochs, which took 491 s. Finally for BERT-Large, to compensate for a small batch size of 32, we use a learning rate of 5×10^{-6} and 3 epochs, which took 1867 s.

Figure 2 shows the accuracy, F1-Score, precision, and recall between the BERT models. The BERT-Base model is the most consistent model across all metrics, with the highest accuracy, F1-Score, and recall. Adding case sensitivity does not improve the model, but instead slightly lowers all four metrics. BERT-Large has the highest precision due to its low number of false positives; however, it is the most inconsistent across the metrics and has the lowest accuracy, F1-Score, and recall. The low recall means that more phishing websites go undetected with this model. The BERT-Large model may have the worst overall performance for two reasons. The first reason is the limited data used to fine-tune and test the model. Even the addition of a couple of hundred phishing and legitimate websites impacted the performance, indicating that this problem could benefit from creating adversarial-generated synthetic URL data as done for the website features. The second reason is the computational resources. Google Colab has TPU number and memory limitations, which we tried to exhaust; however, even with this, the small maximum batch size of 32 is likely insufficient to optimize this large model.

The custom BERT model trained on a URL corpus does not perform as well as the pre-trained models. There are three likely reasons for this. First, we use a dataset of 1.7 million URLs, whereas the BERT pre-trained models use a dataset of 2,500 million words. Second, because URLs are riddled with punctuation, we include these in the vocabulary; however, this could very likely be harming the performance. Third, due to computational constraints, we only use a maximum sequence

[3] https://www.TensorFlow.org/official_models/fine_tuning_bert

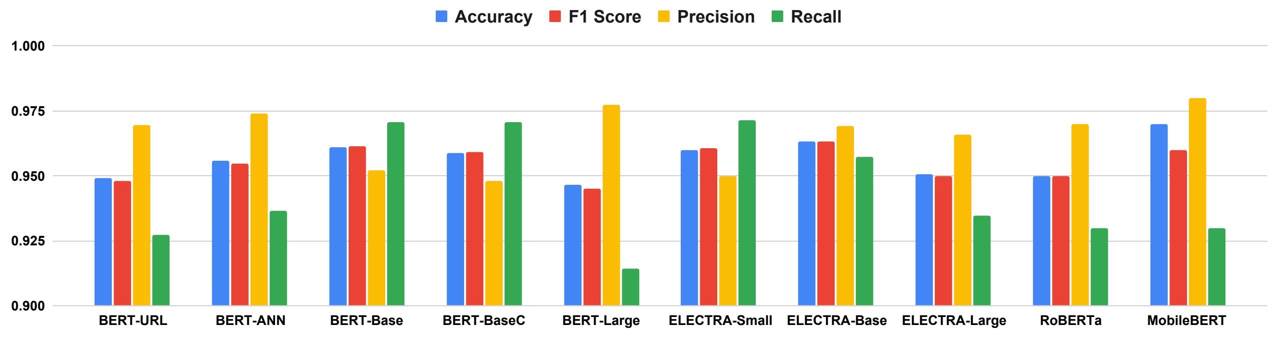

Fig. 2: Evaluation metrics for each of the five BERT models, three ELECTRA models, RoBERTa model, and MobileBERT model used in experiments.

length of 128 versus the 512 used in the pre-trained models. Since URLs are often longer than 128, it is possible that not including longer sequences degrades the performance.

Using a hybrid approach by combining BERT-URL predictions with an ANN optimized on pre-created features from the URLs (BERT-ANN) minimally improves the model performance. All four metrics increase slightly (< 0.005), indicating that adding URL-based features does not provide any additional information not already captured by the BERT methodology of language processing. This result shows promise for NLP approaches using URLs solely to obtain high phishing detection accuracy as these models continue to improve.

We perform three ELECTRA experiments, fine-tuning and optimizing three different pre-trained models. For ELECTRA-Small, we settle on a learning rate of 8×10^{-5} with 8 epochs and a batch size of 128, taking 358 s. ELECTRA-Base uses the same learning rate and batch size, but with 6 epochs, taking 489 s. ELECTRA-Large uses a learning rate of 8×10^{-6}, a batch size of 32, and 5 epochs, taking 2100 s.

Figure 2 shows ELECTRA's performances. ELECTRA-Base has the highest mean score across all metrics, with a value of 0.963 compared to ELECTRA-Small's value of 0.96. The ELECTRA-Base model has the highest accuracy, F1-Score, and precision, while the ELECTRA-Small model has the highest recall. Like BERT, ELECTRA-Large has the poorest performance, again likely due to data and computation limitations.

Figure 2 shows that BERT and ELECTRA have similar accuracy, with the highest values of 96.1% and 96.3%, respectively; both models have F1-Scores nearly identical to their accuracy. ELECTRA has slightly higher precision and lower recall, whereas the opposite is true for BERT. While ELECTRA has higher positive predictive values, it has a lower hit rate and is more likely to miss phishing websites than BERT. Reasonable scores demonstrates that ELECTRA and BERT have skills at predicting phishing websites directly using URLs.

We perform one RoBERTa experiment and report the results in Figure 2. Similar to previous models, RoBERTa, was pre-trained on general-purpose NLP tasks, and we fine-tuned it for the domain-specific task of phishing detection using the DS-URL-Only dataset and training for 3 epochs. Results show that RoBERTA's performance is not as good as other models. For example, the accuracy is 13% less than ELECTRA-Large and BERT-URL, two other competitor models.

Finally, we fine-tune MobileBERT for phishing detection and include the results in Figure 2. While this transformer has less parameters, this does not negatively impact the performance. In contrast, both the accuracy and precision are higher than those for BERT-Base and ELECTRA-Base. This proves that having more parameters does not necessarily improve the performance for a domain-specific task.

4.4. Task 4: MobileBERT Time Analysis

Comparing the prediction time of MobileBERT against BERT-Base, MobileBERT performs 3 times faster, taking only 2.8 s to predict 10 samples on Google Colab's TPU. Additionally, MobileBERT only took $\sim$230 s to fine-tune, which is less than the training time of the top ANN feature-based model ($\sim$300 s) and with the distinct advantage of using URLs directly rather than requiring feature specification and collection. This indicates that MobileBERT is an excellent choice for running on light-weight devices (like mobiles) for its training/updating simplicity, low computation cost, fast prediction rate, and high performance.

4.5. Evaluation Against Related Works

Evaluation metrics for our top models are shown in Figure 3 (bold), along with a compilation of results for the state-of-the-art models discussed in the related work (Section 2). To directly compare our results, for BERT and ELECTRA we show the model with the highest mean overall score, which was the pre-trained base configuration for both. We do not include Task 2's results due to its substantially lower performances, again emphasizing that website content is important in feature-based phishing detection.

Figure 3 shows that ANNF performs similarly to current models, outperforming the well-known phishing approach XGBoost. ANNF is also one of the most consistent models across all metrics. For example, both DTOF-ANN and OFS-NN have high accuracies but lower recalls than ANNF. This behavior is not optimal in phishing detection because it indicates that more phishing websites may go undetected using these approaches. Both BERT and ELECTRA have performances slightly lower than the top models but on par with XGBoost and higher than ARTMAP. MobileBERT outperforms both BERT and ELECTRA in terms of accuracy and precision, with values on par with ANNF and NIOSELM; however, it also has one of the lowest recall scores, performing only slightly better than ARTMAP for this metric.

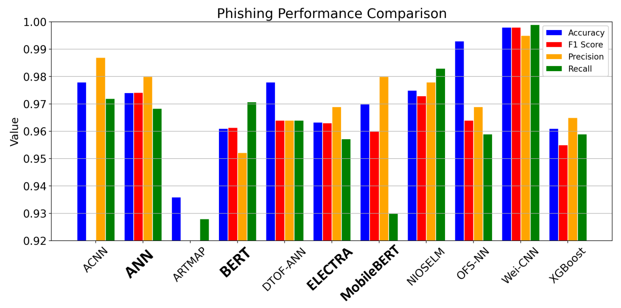

Fig. 3: Scores for the top ANN, BERT, ELECTRA, and MobileBERT models in this study (bold) against recently published model results. ACNN results are from Zhang et al. [19], ARTMAP results are from Saravanan et al. [4], DTOF-ANN results are from Zhu et al. [5], NIOSELM results are from Yang et al. [14], OFS-NN results are from Zhu et al. [13], Wei-CNN results are from Wei et al. [22], and XGBoost results are from Yang et al. [14].

5. Conclusions

Although using website content-based features and creating auxiliary data to train a deep ANN results in the highest-performing phishing detection model in this study, this methodology is very time-consuming and computationally expensive. Further, using only URL-based features in an ANN performs poorly, indicating that transformer-based models are required to achieve high-performance phishing detection solely using URLs directly.

To this end, all pre-trained transformers show promise compared to other current approaches. Although transformers currently have slightly lower performances than several of the top-performing models, they have the distinct advantage that they do not require pre-processing of URLs, making them more easily updatable than feature-based approaches. Finally, due to their ability to predict phishing websites solely from URLs, they are safer to use because malicious sites can be predicted without physically visiting the page. This combination of advantages makes pre-trained transformers easily deployable for real-time detection, indicating they are a viable option for website phishing detection. Specifically, we show that MobileBERT can perform nearly as well as other state-of-the-art models, yet it requires less training time and predicts phishing sites faster than the other transformer and feature-based approaches tested in this study, making it a strong candidate for light-weight and mobile devices.

For future work, we want to test our proposed approach on real mobile devices and evaluate the performance and running time to ensure it is applicable for such devices. In addition, since we had success using NLP transformers to detect phishing websites it is likely that transformers also can be used in spam detection problems, where spam emails can be used to train transformers on the downstream task of spam detection on mobile devices.

Acknowledgements

The work was supported in part by funding from NSF under Award Numbers CNS 1822118, IIS 2027750, Cyber Risk Research, Statnett, NIST, AMI, and ARL.

References

[1] Federal Bureau of Investigation (FBI) Internet Crime Complaint Center. (2021) "2020 Internet Crime Report." Washington, D.C., USA, https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf.

[2] Corrata. (2021) "Mobile phishing is becoming more prevalent and more difficult to follow." Available at https://corrata.com/the-many-faces-of-mobile-phishing/.

[3] Y. Huang, Q. Yang, J. Qin, and W. Wen. (2019) "Phishing URL detection via CNN and attention-based hierarchical RNN." *Proc. of IEEE International Conference on Trust, Security, and Privacy in Computing and Communications*, https://www.doi.org/10.1109/TrustCom/BigDataSE.2019.00024.

[4] P. Saravanan and S. Subramanian. (2020) "A framework for detecting phishing websites using GA based feature selection and ARTMAP based website classification." *Procedia Computer Science*, **171**, 1083-1092, http://doi.org:/10.1016/j.procs.2020.04.116.

[5] E. Zhu, Y. Ju, Z. Chen, F. Liu, and X. Fang. (2020) "DTOF-ANN: an artificial neural network phishing detection model based on decision tree and optimal features." *Applied Soft Computing Journal*, **95**(106505), https://doi.org/10.1016/j.asoc.2020.106505.

[6] H. Shirazi, S.R. Muramudalige, I. Ray, and A.P. Jayasumana. (2020) "Improved phishing detection algorithms using adversarial autoencoder synthesized data." *Proc. of IEEE Conference on Local Computer Networks*, Sydney, Australia.

[7] J. Devlin, M-W. Chang, K. Lee, and K. Toutanova. (2018) "BERT: Pre-training of deep bidirectional transformers for language understanding." Cornell University, New York, USA, https://arxiv.org/abs/1810.04805.

[8] I. AbdulNabi and Q. Yaseen. (2021) "Spam email detection using deep learning techniques." *Procedia Computer Science*, **184**, 853-0509, https://doi.org/10.1016/j.procs.2021.03.107.

[9] S. Kaddoura, O. Alfandi, and N. Dahmani. (2020) "A spam email detection mechanism for English language text emails using deep learning approach." *Proc. of IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 193-198, https://doi.org/10.1109/WETICE49692.2020.00045.

[10] Y. Lee, J. Saxe, and R. Harang. (2020) "CatBERT: Context-aware tiny BERT for detecting social engineering emails." Cornell University, New York, USA, https://arxiv.org/abs/2010.03484.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019) "Roberta: A robustly optimized BERT pre-training approach." *arXiv Preprint*, arXiv:1907.11692, https://arxiv.org/abs/1907.11692.

[12] K. Clark, M.-T. Luong, Q.V. Le, and C.D. Manning. (2020) "ELECTRA: Pre-training text encoders as discriminators rather than generators." *Proc. of International Conference on Learning Representations*, https://openreview.net/pdf?id=

r1xMH1BtvB.

[13] E. Zhu, Y. Chen, C. Ye, X. Le and F. Liu. (2019) "OFS-NN: an effective phishing websites detection model based on optimal feature selection and neural network." *IEEE Access*, **7**, https://10.1109/ACCESS.2019.2920655.

[14] L. Yang, J. Zhang, X. Wang, Z. Li, Z. Li, and Y. He. (2021) "An improved ELM-based and data preprocessing integrated approach for phishing detection considering comprehensive features." *Expert Systems with Applications*, **165**(113863), http://doi.org/10.1016/j.eswa.2020.113863.

[15] K.M.Z. Hasan, M.Z. Hasan, and N. Zahan (2019) "Automated prediction of phishing websites using deep convolutional neural network." *Proc. of International Conference on Computer, Communication, Chemical, Materials, and Electronic Engineering*, Rajshahi, Bangladesh, 1-4, https://doi.org/10.1109/IC4ME247184.2019.9036647.

[16] H. Le, Q. Pham, D. Sahoo, and S.C.H. Hoi. (2018) "URLNet: learning a URL representation with deep learning for malicious URL detection." Cornell University, New York, USA, https://arxiv.org/abs/1802.03162.

[17] P. Yang, G. Zhao, and P. Zeng. (2019) "Phishing website detection based on multidimensional features driven by deep learning." *IEEE Access*, https://doi.org/10.1109/ACCESS.2019.2892066.

[18] S.Y. Yerima and M.K. Alzaylaee. (2020) "High accuracy phishing detection based on convolutional neural networks." *Proc. of International Conference on Computer Applications & Information Security*, Riyadh, Saudi Arabia, 1-6, https://doi.org/10.1109/ICCAIS48893.2020.9096869.

[19] X. Zhang, D. Shi, H. Zhang, W. Liu, and R. Li (2018). "Efficient detection of phishing attacks with hybrid neural networks." *Proc. of IEEE International Conference on Communication Technology (ICCT)*, Chongqing, 844-848, https://doi.org/10.1109/ICCT.2018.8600018.

[20] J. Saxe and K. Berlin. (2017) "eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths, and registry keys." Cornell University, New York, USA, https://arxiv.org/abs/1702.08568v1.

[21] X. Xiao, D. Zhang, G. Hu, Y. Jiang, and S. Xia. (2020) "CNN-MHSA: a convolutional neural network and multi-head self-attention combined approach for detecting phishing websites." *Neural Networks*, **125**, 303-312, http://doi.org/10.1016/j.neunet.2020.02.013.

[22] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Wozniak. (2020) "Accurate and fast URL phishing detector: a convolutional neural network approach." *Computer Networks*, **178**(107275), http://doi/org/10.1016/j.comnet.2020.107275.

[23] Y. Lee, J. Saxe and Richard Harang, (2021) "CATBERT: Context-Aware Tiny BERT for Detecting Social Engineering Emails". *KDD '21 Workshop on AI-enabled Cybersecurity Analytics, Aug. 14-18, 2021, Virtual Conference*. ACM, New York, NY, USA, 7 pages, https://ai4cyber-kdd.com/KDD-AISec_files/CatBERT_for_ACM_KDD.pdf.

[24] C. Thapa, J. Tang, A. Abuadbba, Y. Gao, S. Camtepe, S. Nepal, M. Almashor, and Y. Zheng. (2021) "Evaluation of Federated Learning in Phishing Email Detection." *arXiv Preprint*, arXiv:2007.13300v3, https://arxiv.org/abs/2007.13300.

[25] P. Xu, (2021) "A Transformer-based Model to Detect Phishing URLs." *arXiv Preprint*, arXiv:2109.02138, https://arxiv.org/abs/2109.02138.

[26] P. Maneriker, J. Stokes, E. Lazo, E. Carutasu, F. Tajaddodianfar, and A. Gururajan. (2021) "URLTran: Improving Phishing URL Detection Using Transformers." *arXiv Preprint*, arXiv:2106.05256, https://arxiv.org/abs/2106.05256.

[27] K. Haynes, H.Shirazi, and I. Ray. (2021) "Lightweight URL-based phishing detection using natural language processing transformers for mobile devices." *18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC) 127-134*, https://doi.org/10.1016/j.procs.2021.07.040.

[28] H. Shirazi, B. Bezawada, I. Ray, and C. Anderson. (2019) "Adversarial sampling attacks against phishing detection." *Proc. of IFIP Annual Conference on Data and Applications Security and Privacy*, Charleston, SC, USA, https://doi.org/10.1007/978-3-030-22479-0_5.

[29] H. Shirazi, L. Zweigle, and I. Ray. (2020) "A machine-learning based unbiased phishing detection approach." *Proc. of International Conference on Security and Cryptography*, Paris, France.

[30] C.L. Tan. (2018) "Phishing dataset for machine learning: feature evaluation." Mendeley Data, V1, http://dx.doi.org/10.17632/h3cgnj8hft.1.

[31] J. Bergstra, D. Yamins, and D.D. Cox. (2013) "Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures." *Proc. of International Conference on Machine Learning*, Atlanta, Georgia, USA, http://proceedings.mlr.press/v28/bergstra13.pdf.

[32] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. (2011). "Algorithms for hyper-parameter optimization." In *Advances in Neural Information Processing Systems 24*, 2546-2554, https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

[33] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, . . . , and A.M. Rush. (2020) "Transformers: State-of-the-Art Natural Language Processing." *Proc. of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 38-45, https://www.aclweb.org/anthology/2020.emnlp-demos.6.