Toward More Generalized Malicious URL Detection Models

Yun-Da Tsai, Cayon Liow, Yin Sheng Siang, Shou-De Lin

National Taiwan University

f08946007@csie.ntu.edu.tw, cayon.1318.96@hotmail.com

b06902103@csie.ntu.edu.tw, sdlin@csie.ntu.edu.tw

Abstract

This paper reveals a data bias issue that can profoundly hinder the performance of machine learning models in malicious URL detection. We describe how such bias can be diagnosed using interpretable machine learning techniques and further argue that such biases naturally exist in the real world security data for training a classification model. To counteract these challenges, we propose a debiased training strategy that can be applied to most deep-learning based models to alleviate the negative effects of the biased features. The solution is based on the technique of adversarial training to train deep neural networks learning invariant embedding from biased data. Through extensive experimentation, we substantiate that our innovative strategy fosters superior generalization capabilities across both CNN-based and RNN-based detection models. The findings presented in this work not only expose a latent issue in the field but also provide an actionable remedy, marking a significant step forward in the pursuit of more reliable and robust malicious URL detection.

Introduction

Machine learning techniques have become indispensable in security-related applications like malware detection, URL reputation classification, and intrusion detection to combat the growing variety of Internet attacks (Gandotra, Bansal, and Sofat 2014; Tsai, Chen, and Lin 2021). Recent end-to-end solutions utilizing deep learning models have achieved state-of-the-art performance, reporting cross-validation AUC scores above 98% when trained on specific datasets.

However, our study reveals a significant vulnerability in these cutting-edge detection models: a serious deterioration in performance when applied to real-world daily queries. Through extensive testing of three leading malware detection models (URLNet, Malconv, and LSTM) using the ISCX-URL-2016 dataset, we observed a drop in performance at test time. The results in Table 1a show at least a 30.2% drop in performance when evaluating with open-domain data - large daily queries obtained from VirusTotal. We further evaluated the models trained on VirusTotal data from 2013-2017 and also tested on VirusTotal from 2018-2019. The results in Table 1b also showed at least a 30% drop in performance which indicates an inability to generalize to changes in data distribution over time.

Upon closer examination with interpretable machine learning techniques, we identified a severe data bias issue at the heart of this problem. This bias, caused by irrelevant but correlated relations between input tokens and output classes, misled the algorithm into making erroneous predictions. To tackle this bias issue, we introduce a debiasing feature embedding strategy. Trained in an end-to-end model using adversarial techniques and a blend of classification and debiasing loss, this method aims to eliminate target bias. Applicable to most end-to-end neural network models, our strategy demonstrated substantial improvements in generalizability (i.e., AUC from 6x% to 9x%) when tested on state-of-the-art CNN-based and RNN-based malware detection models with real-world queries.

In summary, this paper makes the following contributions:

- First, to our understanding, this is the first work that reports severe concern on the generalization of a malware classification model and further argues that we should carefully choose the validation and testing data, as opposed to a random k-fold validation, to prevent from overfitting the bias.
- We then perform an in-depth analysis to discover that such
performance degradation majorly comes from the token-level data bias issue in the security domain. Finally, we propose a debiasing solution based on the adversarial training technique allowing end-to-end classification models to learn a bias-free feature embedding. We demonstrate the proposed strategy can be applied to three state-of-the-art neural-based malware detection frameworks with significant performance boosts.

In the following sections, we will start by introducing the state-of-the-art neural network classification architecture for malicious URL detection in section . Next, we will describe how one can interpret such models to identify the data bias problem in section . Finally, we propose an adversarial debiasing architecture in section and demonstrate through several experiments how a more robust model can be generated under the proposed framework in section .

**Invariant Feature Learning**

The drive to learn representations invariant to specific factors has inspired research across machine learning fairness, debiasing learning, and domain adaptation, aiming to control the information encapsulated within features. For instance, studies such as (Edwards and Storkey 2015; Beutel et al. 2017) deploy adversarial training to cultivate fair representations. Works like (Louizos et al. 2015; Zafar et al. 2015) strive to attain a fair representation by minimizing the correlation between the latent representation and sensitive attributes. Other efforts, like (Xie et al. 2017), use domain-adversarial training to craft representations that remain invariant to specific factors, enhancing generalization. Meanwhile, studies like (Li et al. 2018; Akuzawa, Iwasawa, and Matsuo 2019) employ domain-adversarial training methodologies to achieve controllable invariant features regarding existing variables for domain generalization. Additionally, (Kim et al. 2019) suggests a regularization method that minimizes the mutual information between feature embedding and bias while training neural networks on biased images. These techniques predominantly leverage an adversarial training approach, aiming to either preserve or exclude certain information from the representation.

**Generalizability in Malicious URL Detection Models**

In security applications, where misclassification can lead to fatal consequences, ensuring that the trained model accurately learns meaningful patterns for decision-making becomes paramount. This section delves into how techniques from the realm of explainable machine learning can be utilized to discern what types of malicious patterns a classifier identifies. This understanding aids in explaining why a model may fail to generalize to out-of-domain data. We also uncover a significant data bias concern that, to the best of our knowledge, has never been raised in the context of malware detection tasks.

**Understanding the Malicious URL Classifier**

As illustrated in Table 1, we found that the trained classifiers exhibit instability when handling future daily queries. To
Figure 2: We demonstrate three URL samples with the visualized interpretation of URLNet. Red refers to the attribution of the malicious class, and Blue refers to the benign class. The first example shows unreasonable highlights on URL-encoded tokens. The second and third example highlights domain names. The highlights are strongly correlated with the tokens’ class-level distribution, which is shown in Table 2. These three samples can be found on VirusTotal.

We utilized the Integrated Gradient algorithm (Sundararajan, Taly, and Yan 2017) in conjunction with the SmoothGrad algorithm (Smilkov et al. 2017) to quantify the attribution of input features to gradients in the form of sensitivity maps. Here, we provide a brief overview of these techniques:

**Integrated Gradient:** This has emerged as an effective tool to probe the significance of each feature towards predictions in neural networks. Consider a deep network function $F : \mathbb{R}^n \rightarrow [0, 1]$, and an input $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. The attribution of the prediction at input $x$ relative to a baseline $x'$ is expressed as a vector $A_F(x, x') = (a_1, \ldots, a_n) \in \mathbb{R}^n$, where $a_i$ represents the contribution of $x_i$ to the prediction $F(x)$. Integrated gradients are calculated as the path integral of the gradients along the straight line from the baseline $x'$ to the input $x$. The attribution for the $i^{th}$ feature is given by:

$$
(x_i - x'_i) \times \int_0^1 \frac{\partial F(x' + \alpha(x-x'))}{\partial x_i} \, d\alpha
$$

**SmoothGrad:** To mitigate the effect of noisy gradients resulting from essentially inconsequential local variations in partial derivatives, a stochastic approximation of the local average is computed by random sampling in a neighborhood of an input $x$:

$$
\hat{M}_c(x) = \frac{1}{n} \sum_{i=1}^n M_c(x + N(0, \sigma^2))
$$

where $n$ is the number of samples, $M$ is the sensitivity map, $c$ is the class of input $x$, and $N(0, \sigma^2)$ symbolizes Gaussian noise with standard deviation $\sigma$.

**Discovering Data Bias**

There are several established patterns used to identify malicious URLs, such as random domain names (DGA algorithm), suspicious paths (LFI), embedded URLs (SSRF, 9779 1.0 info 0.93 www 0.98 facebook 0.89 B4 0.76 E7 0.85 php 0.81 apple 0.97 A1 0.28 99 0.39

Table 2: The class-level distribution of highlighted tokens in figure 2. 1.0 means the token only appears in malicious class and 0.0 for benign class.
The significance of low-entropy tokens lies in their association with specific classes; they tend to occur predominantly in either malicious or benign URLs, but not both. This property makes them attractive to machine learning models as representative signals for distinguishing between classes. However, relying heavily on such low-entropy tokens can lead to overfitting issues, especially when they lack generalization across different domains, datasets, or time spans. In particular, we found that such data bias issue has an even higher tendency to happen in URL or other sources for se-

Table 3: The top 5 least diverse tokens, with the smallest entropy in class distribution, reveal the substantial disparity between the training set and testing set. The datasets used include the ISCX dataset and daily queries collected from VirusTotal from 2013-2019. We create a three-fold dataset using a sliding window over time, illustrating the rapid changes in token distribution. Despite selecting a well-fitted model based on the validation set, the AUC score in the testing set on the VirusTotal dataset exhibits a significant drop.

To gain a better understanding, we conducted a thorough examination and introduced the concept of class-level distribution for tokens, representing their occurrence probabilities across different classes. For example, if a token appears 2 times in the malicious sample and 3 times in benign samples, the class distribution would be \([2/5, 3/5]\). We calculated the class-level entropy for the highlighted tokens in the malicious class (Table 2) and, interestingly, found that these tokens exhibited low entropy, making them crucial and distinguishable features among the classes. Figure 3 illustrates the class-level distribution of the tokens highlighted by the IG method, emphasizing the importance of tokens with extremely high or low distribution.

The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)
curity applications due to the following three reasons:

1. There are generally more low-entropy tokens in URL and malware detection tasks compared to other NLP tasks: Low-entropy tokens refer to those that are less informative to human experts, and experienced experts typically do not heavily rely on them for making judgments. Examples of such tokens include the scheme, path, and domain names in URL components. Despite their seemingly less informative nature, some of these tokens exhibit very low class-level token entropy, meaning they tend to occur predominantly in specific classes, rendering them distinguishable features. Figure 1 illustrates a histogram of class-level token entropy, comparing a malicious URL detection task with a regular NLP text classification task (i.e., sentiment analysis using the IMDB user comment dataset). Notably, URLs and malware data contain a significantly larger number of tokens with low class-level entropy compared to typical NLP data. Leveraging these low-entropy tokens, a powerful machine learning model can effectively capture critical signals, leading to high cross-validation accuracy.

2. Malicious patterns change rapidly: To avoid detection, malicious data usually mutate rapidly, causing the class-level token distributions to vary over time. Table 3 shows the more representative tokens (i.e., the tokens with the lowest class-level entropy) in different time span from 2013 to 2019. It reveals that the useful or representative tokens/features identified in the training data (from T-4 to T-1) vary significantly from the ones for testing data at time T, while such diversity is less obvious in NLP tasks. Such concept-drift phenomenon implies that a model cannot rely on the low-entropy tokens as critical features to perform classification though it may seem very effective during training and validation time.

3. Generating a training dataset that accurately reflects the real-world class distribution poses a significant challenge, particularly when collecting data through security intelligence services like VirusTotal, as it can introduce bias. In real-world scenarios, users typically encounter a far greater number of normal URLs (negative samples) than malicious ones (positive samples). However, in several existing malicious URL datasets, the negative data consists of suspicious URLs that are eventually marked as benign. Consequently, URLs that are obviously benign, such as those belonging to famous companies, are often excluded from negative examples. Even if some benign instances are included, their abundance can lead to an imbalanced dataset with a ultra-sparse representation of positive or malicious instances. For instance, tokens like ‘Facebook’ are frequently encountered in benign URLs, but in training data (e.g., ISCX and VirusTotal), such benign tokens may be absent, and worse, they could mistakenly appear in the malicious class due to mimicking phishing sites. Therefore, detection models must possess the ability to capture genuine malicious patterns while discerning and disregarding these strong yet biased statistical signals.

The phenomenon discussed above also reveals that to evaluate a malware detection model, simply splitting a given dataset for k-fold cross validation can be misleading. It is required to obtain a separate testing dataset through either different sources or collecting at different time-span to make the dataset more robust as possible.

Data bias can cause classifiers to behave like blacklists, memorizing low-entropy tokens rather than capturing complex malicious patterns. Such classifiers may fail to generalize well to new data, leading to performance degradation over time. This phenomenon is not unique to URL data and can be observed in other categories of malware files, such as HTML, executable binary (EXE), and JavaScript (Figure 3).

Adversarial Debiasing Feature Embedding

As we have previously identified, overfitting may occur when relying on individual tokens or features to identify malicious URLs. A straightforward way to alleviate this concern might be to remove individual tokens from the feature set. However, in the context of URL detection, all features are composed of tokens, so simply removing token features is not feasible. To address this challenge, we propose a strategy termed adversarial debiasing embedding (ADE), applicable to most end-to-end neural network-based solutions.

The fundamental idea of our proposed method is to eliminate the bias wherein a token is mistakenly correlated with specific classes. Once the biased information is removed, the embedding itself cannot be classified into any particular class. We accomplish this by introducing an adversarial component, called the Bias Predictor, to predict the class-level distribution of tokens. We then back-propagate the reverse gradient to update the embedding to remove information that is useful for the task of the Bias Predictor. Our aim is to minimize the mutual information between discriminative features and biased information so that the learned embedding can serve as inputs to the classifier to learn more
complex patterns without relying on biased tokens. The overall architecture is depicted in Figure 4. Unlike domain adaptation tasks in NLP/CV, which require explicit domain labels, we do not have bias labels for our objective function. A novel aspect of our approach is that we extract biased information directly from the data and use it as pseudo-labels in a manner similar to contrastive self-supervised learning.

To prevent the model from erroneously correlating biased tokens to a specific class, we define the class distribution of tokens as the biases that must be removed from the token embedding. For a $K$-class classification task with a dataset containing $N$ pairs of training data and a vocabulary dictionary $V = \{ v_i \}_{i=1}^M$ of $M$ tokens, we calculate a $K \times M$ class distribution matrix $B$, where $b_{ij}$ is the probability of token $v_i$ appearing in class $j$ and vector $b_j$ is a soft class distribution label. Assuming the bias information $B$ is only correlated with the training set:

$$ I(B(x_{\text{train}}); y) \gg I(B(x_{\text{test}}); y) \approx 0, \quad (1) $$

where $X_{\text{train}}$ and $X_{\text{test}}$ denote the training and testing datasets respectively, and $I(\cdot; \cdot)$ denotes the mutual information.

Given an input sequence $x$ and target label $y$, each token in $x$ is transformed into a token embedding after passing through $\theta_e$. The token embeddings are then forwarded to a classifier $\theta_c$ to minimize the classification loss:

$$ L_c(x, y; \theta_c, \theta_e) = - \sum_{k=1}^K y_k \log \hat{y}_k \quad (2) $$

After back-propagating from the classification loss, the token embedding $\theta_e$ learns discriminative features. However, since each individual token in training data are biased, the network $\theta_c$ can rely heavily on the misleading bias information:

$$ I(B(X); \theta_c(\theta_e(x))) \gg 0. \quad (3) $$

Next, we minimize the mutual information between the token embedding $\theta_e(x)$ and the bias information $B$:

$$ I(B(x); \theta_e(x)) = H(B(x)) - H(B(x)|\theta_e(x)) \quad (4) $$

Similar to (Kim et al. 2019), we ignore constant $H(B(x))$ and apply a bias predictor network that approximate the untraceable posterior distribution $P(B(x)|\theta_e(x))$ in eq 4 and relax it into minimizing the bias prediction cross-entropy loss which the biases $\hat{b}$ for each token are obtained from the class distribution matrix $B$. When back-propagating, the gradient is passed with gradient reversal technique (Ganin et al. 2017) in order to eliminate the learned biases in $\theta_e$. Due to the fact that the bias information vector $b$ is a soft class label, the bias prediction cross-entropy loss can be trained accordingly and characterized as a classification problem.

$$ L_b(x, b; \theta_e, \theta_b) = - \sum_{k=1}^K b_k \log \hat{b}_k \quad (5) $$

Together with the classification loss and bias prediction loss, the method minimizes the joint minimax loss function

$$ \min_{\theta_e, b} \max_{\theta_b} L_c(x, y; \theta_e, \theta_c) - \lambda L_b(x, b; \theta_e, \theta_b). \quad (6) $$

In each iteration of adversarial training, we first back-propagate the $L_c$ loss to update $\theta_e$ and $\theta_c$. With $\theta_e$ fixed, we then minimize the $L_b$ loss to update $\theta_b$. Next, with $\theta_b$ fixed, we maximize the $L_b$ loss to update $\theta_c$. After the last step, the bias information is removed from $\theta_c$. Eventually, $\theta_c$ learns feature embedding that is discriminative but does not rely on biased information.

<table>
<thead>
<tr>
<th>URL</th>
<th>AUC</th>
<th>bACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>URLNet</td>
<td>69.3</td>
<td>67.2</td>
<td>68.3</td>
</tr>
<tr>
<td>URLNet + BlindEye</td>
<td>75.1</td>
<td>65.5</td>
<td>70.3</td>
</tr>
<tr>
<td>URLNet + ADE</td>
<td>90.1</td>
<td>73.1</td>
<td>70.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTML</th>
<th>AUC</th>
<th>bACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malconv</td>
<td>80.7</td>
<td>76.2</td>
<td>81.6</td>
</tr>
<tr>
<td>Malconv + BlindEye</td>
<td>83.5</td>
<td>79.5</td>
<td>81.3</td>
</tr>
<tr>
<td>Malconv + ADE</td>
<td>86.7</td>
<td>81.0</td>
<td>84.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXE</th>
<th>AUC</th>
<th>bACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>61.6</td>
<td>54.0</td>
<td>56.4</td>
</tr>
<tr>
<td>LSTM + BlindEye</td>
<td>70.9</td>
<td>64.2</td>
<td>72.5</td>
</tr>
<tr>
<td>LSTM + ADE</td>
<td>90.6</td>
<td>75.5</td>
<td>75.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTML</th>
<th>AUC</th>
<th>bACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>80.8</td>
<td>59.4</td>
<td>73.6</td>
</tr>
<tr>
<td>LSTM + BlindEye</td>
<td>84.1</td>
<td>80.2</td>
<td>82.1</td>
</tr>
<tr>
<td>LSTM + ADE</td>
<td>87.9</td>
<td>81.2</td>
<td>84.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXE</th>
<th>AUC</th>
<th>bACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>62.5</td>
<td>62.5</td>
<td>85.5</td>
</tr>
<tr>
<td>LSTM + BlindEye</td>
<td>68.3</td>
<td>65.6</td>
<td>91.2</td>
</tr>
<tr>
<td>LSTM + ADE</td>
<td>68.6</td>
<td>68.6</td>
<td>92.3</td>
</tr>
</tbody>
</table>

Table 5: Performance of plain backbone models and backbone+BlindEye or backbone+ADE. The models are trained using ISCX-URL-2016 dataset and tested using VirusTotal daily queries. The best results in each column are typeset in bold.
Figure 5: The performance of URLNet trained on VirusTotal over time according to a sliding window on the timeline. URLNet with ADE surpasses the other by a large margin at nearly all times.

**Experiments**

Here we conduct experiments to evaluate whether ADE could lead to better generalization. We evaluate two scenarios: first, whether the proposed solution yields significant improvement over the out-of-distribution data, and second, whether the solution alleviates the degradation of the backbone classifier over time. In addition, we compared the state-of-the-art debiasing feature learning algorithm with the proposed solution. Note that, as described previously, the majority of the previous experiments divide training and testing data randomly, which are not suitable for evaluating the scenarios of out-of-domain data or data-distribution drift over time. Hence, we have designed two experiment scenarios to better test the generation capability of the proposed solution. In the first experiment, we intentionally choose data from other sources as the out-domain testing data. For example, we trained a model using ISCX-URL-2016 dataset and tested the model using VirusTotal daily queries. In the second experiment, we made the train/test split according to a sliding window on the timeline to evaluate whether the model can adapt to the data distribution changes over time. That is, a model is trained based on the previous records and evaluated on the future records. For example, the model to be tested using 2018 data is trained with data from 2013-2017.

**Evaluation Metrics and Datasets**

The experiment results are measured by three commonly used metrics, including Balanced accuracy, F1-score, and area under curve (AUC). Among them, AUC is commonly accepted as a critical measurement since it is desirable to have the most malicious files ranked highest and lower false positives.

We collect a large dataset of over one million files in three categories: URL, HTML, and EXE. The URL data contain ISCX-URL-2016 dataset and daily queries from VirusTotal. The HTML dataset is crawled according to the previously collected URL. The EXE data are obtained from VirusTotal. The statistics of the datasets are shown in Table 4.

**Backbone Models**

In our experiments, we chose three state-of-the-art end-to-end malware classification frameworks as our backbone, two based on CNN (i.e., Malconv (Raff et al. 2018) and URLNet (Le et al. 2018)) and one based on RNN (i.e., LSTM Malware Language Model (Athiwaratkun and Stokes 2017)) structures. We want to evaluate whether combining these models with ADE a better generalization capability can be observed. According to Figure 4, we can simply replace the classifier with either Malconv, LSTNet, or LSTM to conduct experiments. Note that URLNet is applied to only the URL dataset due to its inefficiency in longer sequences in HTML and EXE, thanks to the adoption of very small CNN kernels. The details of these models are mentioned in section.

Here we compare with the most relevant debiasing algorithm BlindEye (Alvi, Zisserman, and Nelläker 2018) since it also removes bias by learning invariant embedding. BlindEye explicitly removes biases by minimizing the entropy of the output class distribution on the testing set. The key difference is that the confusion loss in BlindEye minimizes predictive entropy, while ours minimizes mutual information. It bounds above our objective function so that we can achieve outperforming results. BlindEye is originally designed for image classification.

**Implementation Details**

In the experiments, we use URLNet, Malconv, and LSTM Malware Language Model as the backbone model. The parameters are exactly the same as the original papers. For the bias predictor, we used a two-layer fully connected network with soft-cross-entropy loss with Adam optimizer and learning rate 1/e.

**Results**

We show the performance of the first experiment in Table 5. The results show that the proposed algorithm outperformed
the original backbone model as well as the BlindEye alternative significantly. The boost in URL data is especially significant since the performance drop in a large held-out testing set is obvious due to data bias and high data variance. Our method improves the AUC score from 67.3% to 90.6%. Data in other categories is much more difficult to collect, which results in a relatively small testing set. This may cause the improvement in generalizability to be less significant, but still, we made improvements on all metrics in Table 5. Note that all scores are statistically significant with p-value < 0.001. Under the scenario of online daily query usage, we trained the model using all data collected before a certain time T and tested the results on data collected in time T. For predicting URLs in time T+1, we retrain the model using all data for the beginning to time T. The results are in Figure 5. Our proposed method shows obvious improvement in generalizability, which consistently outperform the original model without debiasing. In section, we applied URLNet with debiasing embedding and visualized it with an interpretation sensitivity map in Figure 6. The interpretation shows a significant difference from the original one in figure 2. The model no more tries to overfit the performance using domain names caused by data bias.

**Conclusion**

In this paper, we report the severe concern about the generalization of a malware classification model. We then perform an in-depth analysis to discover that such performance degradation majorly comes from the token-level data bias issue in the security domain. Finally, we propose a de-biasing solution based on the adversarial training technique allowing end-to-end classification models to learn a bias-free feature embedding. We demonstrate the proposed strategy can be applied to three state-of-the-art neural-based malware detection frameworks with significant performance boosts. We expect this study could be further extended to different usage of data and contributes toward more robust and interpretable ML models for security applications.

**References**


Beutel, A.; Chen, J.; Zhao, Z.; and Chi, E. H. 2017. Data decisions and theoretical implications when adver-


