Learning Invariant Representation for Malicious Network Traffic Detection

Karel Bartos and **Michal Sofka** and **Vojtech Franc** [1 2]

Abstract. Statistical learning theory relies on an assumption that the joint distributions of observations and labels are the same in training and testing data. However, this assumption is violated in many real world problems, such as training a detector of malicious network traffic that can change over time as a result of attacker's detection evasion efforts. We propose to address this problem by creating an optimized representation, which significantly increases the robustness of detectors or classifiers trained under this distributional shift. The representation is created from bags of samples (e.g. network traffic logs) and is designed to be invariant under shifting and scaling of the feature values extracted from the logs and under permutation and size changes of the bags. The invariance is achieved by combining feature histograms with feature self-similarity matrices computed for each bag and significantly reduces the difference between the training and testing data. The parameters of the representation, such as histogram bin boundaries, are learned jointly with the classifier. We show that the representation is effective for training a detector of malicious traffic, achieving 90% precision and 67% recall on samples of previously unseen malware variants.

1 INTRODUCTION

Trained detectors of malicious traffic can become ineffective as a result of changes of the malware behavior. Formally, this means that a joint distribution of the observations computed from the malware samples and their labels differs for training (source) data and unseen testing (target) data. This can happen as a result of target evolving after the initial classifier or detector has been trained which is typically caused by a deliberate detection evasion strategy of an attacker. In supervised learning, this problem is solved by domain adaptation. Under the assumption that the source and target distributions do not change arbitrarily, the goal of the domain adaptation is to leverage the knowledge in the source domain and transfer it to the target domain. In this work, we focus on the case where the conditional distribution of the observations given labels is different, also called a conditional shift.

The domain adaptation (or knowledge transfer) can be achieved by adapting the detector using importance weighting such that training instances from the source distribution match the target distribution [16]. Another approach is to transform the training instances to the domain of the testing data or to create a new data representation with the same joint distribution of observations and labels [2]. The challenging part is to design an effective transformation that transfers the knowledge from the source domain and improves the robustness of the detector on the target domain.

We present a new optimized representation of network traffic that enables domain adaptation under conditional shift. The representation is computed for bags of samples, each of which consists of features (observations) computed from network traffic logs. The bags are constructed for each user and contain all network communication with a particular hostname. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. This is achieved by combining bag feature histograms with self similarity matrices. Unlike previous approaches [1], the parameters of the representation are learned along with the classifier decision rule in a joint optimization procedure.

The proposed representation is used when training a classifier to detect malicious HTTP communication in network traffic. We will show that the classifier trained on malware samples from one behavioral category can successfully detect new samples from a different category. This way, the knowledge of the malware behavior is correctly transferred to the domain of new category which improves the generalization of the classifier on unseen data. Compared to the baseline flow-based representation [7, 9, 14] or widely-used reputation-based security device, the proposed approach shows considerable improvements and correctly classifies new types of network threats that were not part of the training data.

2 DOMAIN ADAPTATION FOR NETWORK TRAFFIC

The paper proposes a representation of network communication that is invariant against malware modifications attackers typically implement to evade the detection systems. Formally, each sample is represented as an n-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^n$. Samples are grouped into bags, with every bag represented as a matrix $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{n \times m}$, where m is the number of samples in the bag and n is the number of features. A single malware category $y_i \in \mathcal{Y}$ can be assigned to each bag. The representation is used to train a classifier to label the categories as positive (malicious) or negative (legitimate). The categories found in the training data can be different from categories in the testing data. The classifier training is complicated by the fact that the sample distribution typically evolves in time, so the probability distribution of the training data differs from the probability distribution of the test data, i.e. there is a conditional shift [17]:

$$P^L(X|y_j) \neq P^T(X|y_j), \;\; \forall y_j \in \mathcal{Y}, \tag{1}$$

where given category y_j, $P^L(X|y_j)$ and $P^T(X|y_j)$ is the probability distribution on training and testing bags, respectively.

[1] Cisco Systems, Inc., Prague, Czech Republic, email:{kbartos, msofka, vfranc}@cisco.com

[2] Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic, email: bartokar@fel.cvut.cz, xfrancv@cmp.felk.cvut.cz

The purpose of the domain adaptation [3] is to apply knowledge acquired from the training (source) domain into test (target) domain. The relation between $P^L(X|y_i)$ and $P^T(X|y_i)$ is not arbitrary, otherwise it would not be possible to transfer any knowledge. Therefore there is a transformation τ, which transforms the feature values of the bags onto a representation, in which $P^L(\tau(X)|y_i) \approx P^T(\tau(X)|y_i)$. The goal is to find this new representation to classify individual bags represented as X into categories $\mathcal{Y}$.

Numerous methods for transfer learning have been proposed (since the traditional machine learning methods cannot be used effectively in this case), including kernel mean matching [8], kernel learning approaches [5], maximum mean discrepancy [10], or boosting [4]. These methods try to solve a general data transfer with relaxed conditions on the similarity of the distributions during the transfer. The downside of these methods is the necessity to specify the target loss function and availability of large amount of labeled data.

This paper proposes an effective invariant representation that solves the classification problem with a conditional shift (see Equation 1). Once the feature values are transformed, they do not rely on the original distribution and they are not influenced by the shift. The parameters of the representation are learned automatically from the data together with the classifier as a joint optimization process. The advantage of this approach is that the parameters are optimally chosen during training to achieve the best classification efficacy for the given classifier, data, and representation.

3 BAG INVARIANT REPRESENTATION

As stated in Section 2, the probability distribution of bags from the training set can be different from the test set. In the first step, the representation of bags is transformed to be invariant under scaling of the feature values. The traditional representation of a bag that consists of a set of m samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ can be written as $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)^T$, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{in})$ and x_{lk} denotes k-th feature value of l-th sample. This form of representation of samples and bags is widely used in the research community, as it is straightforward to use and easy to compute. However, in the network security domain, the dynamics of the network environment causes changes in the feature values and the shift becomes more prominent. As a result, the performance of the classification algorithms using the traditional representation is decreased.

In the first step, the representation is improved by making the matrix X to be invariant under scaling of the feature values. **Scale invariance** guarantees that even if some original feature values of all samples in a bag are multiplied by a common factor, the values in the new representation remain unchanged. To guarantee the scale invariance, the matrix X is scaled column-wise onto the interval $[0, 1]$.

In the second step, the representation is transformed to be invariant against shifting. **Shift invariance** guarantees that even if some original feature values of all samples in a bag are increased/decreased by a given amount, the values in the new representation remain unchanged. To ensure shift invariance, we create for each feature a self-similarity matrix. Self-similarity matrix is a symmetric matrix, where rows and columns represent individual samples and (i, j)-th element corresponds to the distance between i-th and j-th sample. Self-similarity matrix has been already used thanks to its properties in several applications (e.g. in object recognition [11] or music recording [15]). However, only a single self-similarity matrix for each bag has been used in these approaches. This paper proposes to compute a set of self-similarity matrices, one for every feature. More specifically, a per-feature set of self-similarity matrices $\mathcal{S} = \{S^1, S^2, \ldots, S^n\}$ is computed for each bag. The (i, j)-th element of S^k is a distance between k-th feature value of i-th and j-th sample. The matrices are further normalized by local feature scaling described above to produce a set of matrices $\tilde{\mathcal{S}}$.

The shift invariance makes the representation robust to the changes where the feature values are modified by adding or subtracting a fixed value. For example, the length of a malicious URL would change by including an additional subdirectory in the URL path. Or, the number of transfered bytes would increase when an additional data structure is included in the communication exchange.

Representing bags with scaled matrices $\{\tilde{X}\}$ and sets of locally-scaled self-similarity matrices $\{\tilde{\mathcal{S}}\}$ achieves the scale and shift invariance. **Size invariance** ensures that the representation is invariant against the size of the bag. In highly dynamic environments (such as network traffic), the samples (e.g. individual web requests) may occur in a variable ordering. **Permutation invariance** ensures that the representation should also be invariant against any reordering of rows and columns of the matrices. The final step of the proposed transformation is the transition from the scaled matrices $\tilde{X}, \tilde{\mathcal{S}}$ to normalized histograms. For this purpose, we define for each bag:

$$\boldsymbol{z}_k^X := \text{vector of values from } k\text{-th column of matrix } \tilde{X}$$

$$\boldsymbol{z}_k^{\mathcal{S}} := \text{column-wise representation of upper triangular matrix created from matrix } \tilde{S}^k \in \tilde{\mathcal{S}}.$$

This means that $\boldsymbol{z}_k^X \in \mathbb{R}^m$ is a vector created from values of k-th feature of $\tilde{X}$, while $\boldsymbol{z}_k^{\mathcal{S}} \in \mathbb{R}^r, r = (m-1)\cdot\frac{m}{2}$ is a vector that consists of all values of upper triangular matrix created from matrix $\tilde{S}^k$. Since $\tilde{S}^k$ is a symmetric matrix with zeros along the main diagonal, $\boldsymbol{z}_k^{\mathcal{S}}$ contains only values from upper triangular matrix of $\tilde{S}^k$.

A normalized histogram of vector $\boldsymbol{z} = (z_1, \ldots, z_d) \in \mathbb{R}^d$ is a function $\boldsymbol{\phi}\colon \mathbb{R}^d \times \mathbb{R}^{b+1} \to \mathbb{R}^b$ parametrized by edges of b bins $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_b) \in \mathbb{R}^{b+1}$ such that $\boldsymbol{\phi}(\boldsymbol{z}; \boldsymbol{\theta}) = (\phi(\boldsymbol{z}; \theta_0, \theta_1), \ldots, \phi(\boldsymbol{z}; \theta_{b-1}, \theta_b))$ where

$$\phi(\boldsymbol{z}, \theta_i, \theta_{i+1}) = \frac{1}{d}\sum_{j=1}^{d} [\![z_j \in [\theta_{i-1}, \theta_i)]\!]$$

is the value of the i-th bin corresponding to a portion of components of $\boldsymbol{z}$ falling to the interval $[\theta_{i-1}, \theta_i)$.

Each column k of matrix $\tilde{X}$ (i.e. all bag values of k-th feature) is transformed into a histogram $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ with predefined number of b bins and $\boldsymbol{\theta}_k^X$ bin edges. Such histograms created from the columns of matrix $\tilde{X}$ will be denoted as *feature values histograms*, because they carry information about the distribution of bag feature values. On the other hand, histogram $\boldsymbol{\phi}(\boldsymbol{z}_k^{\mathcal{S}}, \boldsymbol{\theta}_k^S)$ created from values of self-similarity matrix $\tilde{S}^j \in \tilde{\mathcal{S}}$ will be called *feature differences histograms*, as they capture inner feature variability within bag samples.

Overall, each bag is represented as a concatenated feature map $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})\colon \mathbb{R}^{n\times(m+r)} \to \mathbb{R}^{2\cdot n\cdot b}$ as follows:

$$\Big(\boldsymbol{\phi}(\boldsymbol{z}_1^X, \boldsymbol{\theta}_1^X), \ldots, \boldsymbol{\phi}(\boldsymbol{z}_n^X, \boldsymbol{\theta}_n^X), \boldsymbol{\phi}(\boldsymbol{z}_1^{\mathcal{S}}, \boldsymbol{\theta}_1^S), \ldots, \boldsymbol{\phi}(\boldsymbol{z}_n^{\mathcal{S}}, \boldsymbol{\theta}_n^S)\Big) \quad (2)$$

where n is the number of the original flow-based features, m is the number of flows in the bag, and b is the number of bins.

4 LEARNING OPTIMAL HISTOGRAM REPRESENTATION

The bag representation $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$ proposed in Section 3 has the invariant properties, however it heavily depends on the number of bins

b and their edges $\boldsymbol{\theta}$ defining the width of the histogram bins. These parameters that were manually predefined in Section 3 influence the classification performance. To choose the parameters b and $\boldsymbol{\theta}$ optimally, we propose to learn them automatically from the training data such that the classification separability between positive and negative samples is maximized.

When creating histograms in Section 3, the input instances are vectors $\boldsymbol{z}_k^X$ and $\boldsymbol{z}_k^{\mathcal{S}}$, where $k \in \{1, \ldots, n\}$. The algorithm transforms the input instances into a concatenated histogram $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$. To keep the notation simple and concise, we will denote the input instances simply as $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{n \times m}$ (instead of $\boldsymbol{z} = (\boldsymbol{z}_1^X, \ldots, \boldsymbol{z}_n^X, \boldsymbol{z}_1^{\mathcal{S}}, \ldots, \boldsymbol{z}_n^{\mathcal{S}})$), which is a sequence of n vectors each of dimension m.

The input instance $\boldsymbol{z}$ is represented via a feature map $\boldsymbol{\phi}\colon \mathbb{R}^{n \times m} \to \mathbb{R}^{n \cdot b}$ defined as a concatenation of the normalized histograms of all vectors in that sequence, that is, $\boldsymbol{\phi}(\boldsymbol{z}; \boldsymbol{\theta}) = (\boldsymbol{\phi}(\boldsymbol{z}_1; \boldsymbol{\theta}_1), \ldots, \boldsymbol{\theta}(\boldsymbol{z}_n; \boldsymbol{\theta}_n))$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ denotes bin edges of all normalized histograms stacked to a single vector.

We aim at designing a classifier $h\colon \mathbb{R}^{n \times m} \times \mathbb{R}^{n+1} \times \mathbb{R}^{n(b+1)} \to \{-1, +1\}$ working on top of the histogram representation, that is

$$\begin{aligned} h(\boldsymbol{z}; \boldsymbol{w}, w_0, \boldsymbol{\theta}) &= \operatorname{sign}(\langle \boldsymbol{\phi}(\boldsymbol{z}, \boldsymbol{\theta}), \boldsymbol{w} \rangle + w_0) \\ &= \operatorname{sign}\left(\sum_{i=1}^{n} \sum_{j=1}^{b} \phi(\boldsymbol{z}_i, \theta_{i,j-1}, \theta_{i,j}) w_{i,j} + w_0 \right) . \quad (3) \end{aligned}$$

The classifier (3) is linear in the parameters $(\boldsymbol{w}, w_0)$ but non-linear in $\boldsymbol{\theta}$ and $\boldsymbol{z}$. We are going to show how to learn parameters $(\boldsymbol{w}, w_0)$ and implicitly also $\boldsymbol{\theta}$ via convex optimization.

Assume we are given a training set of examples $\{(\boldsymbol{z}^1, y^1), \ldots, (\boldsymbol{z}^m, y^m)\} \in (\mathbb{R}^{n \times m} \times \{+1, -1\})^m$. We fix the representation $\boldsymbol{\phi}$ such that the number of bins b is sufficiently large and the bin edges $\boldsymbol{\theta}$ are equally spaced. We find the weights $(\boldsymbol{w}, w_0)$ by solving

$$\begin{aligned} \min_{\boldsymbol{w} \in \mathbb{R}^{b \cdot p}, w_0 \in \mathbb{R}} \Bigg[& \gamma \sum_{i=1}^{n} \sum_{j=1}^{b-1} |w_{i,j} - w_{i,j+1}| \\ & + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y^i \langle \boldsymbol{\phi}(\boldsymbol{z}^i; \boldsymbol{\theta}), \boldsymbol{w} \rangle\right\} \Bigg] . \quad (4) \end{aligned}$$

The objective is a sum of two convex terms. The second term is the standard hinge-loss surrogate of the training classification error. The first term is a regularization encouraging weights of neighboring bins to be similar. If it happens that j-th and $j+1$ bin of the i-the histogram have the same weight, $w_{i,j} = w_{i,j+1} = w$, then these bins can be effectively merged to a single bin because

$$\begin{aligned} & w_{i,j} \phi(\boldsymbol{z}_i; \theta_{i,j-1}, \theta_{i,j}) + w_{i,j+1} \phi(\boldsymbol{z}_i; \theta_{i,j}, \theta_{i,j+1}) \\ & \qquad = 2w\phi(\boldsymbol{z}_i; \theta_{i,j-1}, \theta_{i,j+1}) . \quad (5) \end{aligned}$$

The trade-off constant $\gamma > 0$ can be used to control the number of merged bins. A large value of γ will result in massive merging and consequently in a small number of resulting bins. Hence the objective of the problem (4) is to minimize the training error and to simultaneously control the number of resulting bins. The number of bins influences the expressive power of the classifier and thus also the generalization of the classifier. The optimal setting of λ is found by tuning its value on a validation set.

Once the problem (4) is solved, we use the resulting weights $\boldsymbol{w}^*$ to construct a new set of bin edges $\boldsymbol{\theta}^*$ such that we merge the original bins if the neighboring weights have the same sign (i.e. if $w_{i,j}^* w_{i,j+1}^* > 0$). This implies that the new bin edges $\boldsymbol{\theta}^*$ are a subset of the original bin edges $\boldsymbol{\theta}$, however, their number can be significantly reduced (depending on γ) and they have different widths unlike the original bins. Having the new bins defined, we learn a new set of weights by the standard SVM algorithm

$$\min_{\boldsymbol{w} \in \mathbb{R}^n, w_0 \in \mathbb{R}} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y^i \langle \boldsymbol{\phi}(\boldsymbol{z}^i; \boldsymbol{\theta}^*), \boldsymbol{w} \rangle\right\} \right] .$$

Note that we could add the quadratic regularizer $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ to the objective of (4) and learn the weights and the representation in a single stage. However, this would require tuning two regularization parameters (λ and γ) simultaneously which would be order of magnitude more expensive than tuning them separately in the two stage approach.

5 EXPERIMENTAL EVALUATION

The proposed approach was applied to classify unseen malware bags. We will show that the combination of learning the invariant representation and the classifier from the data, as described in Section 4, achieves significantly better efficacy results when compared to the baseline.

5.1 Specification of the Datasets

This section provides a detailed description of the datasets, labeled malware samples, and features used in the experimental evaluation. Malware samples were obtained from several months (January - July 2015) of real network traffic of 80 international companies in form of proxy logs [13]. Summary of the datasets used in the evaluation is described in Table 1.

The logs contain HTTP and HTTPS flows, where one flow is one connection defined as a group of packets from a single host and source port with a single server IP address, port, and protocol. As flows from the proxy logs are bidirectional, both directions of a communication are included in each flow. The malware samples were obtained from findings of several network security devices based on signatures (Cisco Cloud Web Security), blacklists (Shadowserver[3]), feeds (Collective Intelligence Framework CIF [6]), and static and behavioral analysis (Cisco Cloud Web Security). In many cases, human experts were involved to label novel previously undetected threats. Findings from VirusTotal[4] server were also used.

In the following, malware samples will be referred as **positive bags**, where one positive bag is a set of flows (connections) from proxy records with the same source host towards the same destination hostname. In other words, each bag contains the whole client-hostname communication for a given period of time. The bags not labeled as malicious are considered as legitimate/negative. Each bag should contain at least 5 flows to be able to compute a meaningful histogram representation from the input flows. Training dataset contains 5k malicious (8 malware families) and 27k legitimate bags, while testing dataset is consist of 2k malicious ($\gg$ 32 malware families) and 241k legitimate bags (more than 15 million flows).

Each flow consists of the following fields: user name, source IP address, destination IP address, source port, destination port, protocol, number of bytes transferred from client to server and from server

[3] www.shadowserver.org
[4] www.virustotal.com

to client, flow duration, timestamp, user agent, URL, referer, MIME-Type, and HTTP status. The most informative field is the URL, which can be decomposed further into 7 parts as illustrated in Figure 1. From the flow fields mentioned above, we extracted 115 flow-based features listed in Table 2. Features from the right column are applied on all URL parts, including the URL itself and a referer.

Category	Flows	Bags
Training Positives	132,756	5,011
Click-fraud mw	12,091	819
DGA malware	8,629	397
Dridex	8,402	264
IntallCore	17,317	1,332
Monetization	3,107	135
Mudrop	37,142	701
Poweliks	11,648	132
Zeus	34,420	1,275
Testing Positives	43,380	2,090
Training Negatives	862,478	26,825
Testing Negatives	15,379,466	240,549

Table 1: Number of flows and bags of malware categories and legitimate background traffic used for training and testing the proposed representation and SVM classifier.

5.2 Malicious Samples

More then 32 different malicious categories were found in the evaluation datasets. Note than most of the novel threats (typically not detected by most of existing devices) were found and confirmed manually and are placed into one malicious category called *other categories*. To illustrate the complexity of the classification problem caused by the large amount of malware families and their variability, a brief description of some categories is provided:

- **Asterope** – Threat related to Asterope click-fraud botnet. Asterope is a Trojan bot malware which performs click-fraud by imitating the action of a user clicking on an advertisement. The bot communicates with the command-and-control server using HTTP from which it receives the next website to visit.
- **Click-fraud, malvertising-related botnet** – The main distribution channel for this threat is fraudulent software such as antivirus, browser plugins, and software updates. The infection typically appears as a browser plugin that hijacks web browsers. It may then establish a command-and-control channel, track user activity, have rootkit capability, and perform click-fraud through the automatic loading and clicking of unsolicited advertisements. The attacker may obtain information about the infected device and attempt to further exploit the device with additional threats.
- **DGA** – Threat that uses domain generation algorithms (DGA) and Fast-Flux to establish its command-and-control communication. Fast-Flux is a DNS technique used by botnets to hide malicious devices behind a command-and-control infrastructure of compromised hosts. These hosts act as proxies that register and de-register their IP addresses. By using a short Time To Live (TTL) value, the hostname to IP address mapping for devices in the requested domain name space will change rapidly. This results in a constantly changing list of destination IP addresses for a single DNS name and allows the attacker to distribute information about the malicious devices.

Figure 1: URL decomposition into seven parts.

- **Dridex** – Threat related to Dridex banking trojan. Dridex is typically spread through spam campaigns and its main goal is to obtain confidential information from the user about his or her online banking and other payment systems. The trojan communicates with the command-and-control server using HTTP, P2P, or I2P protocols.
- **Monetization** – Malware monetization activity involving fake blog sites that serve as front ends for click-fraud.
- **Poweliks** – Threat related to the Poweliks Trojan which downloads other malware to the infected device and can steal information. The threat is persistent and uses several mechanisms to hide itself.
- **Zeus** – Threat related to the Zeus Trojan horse malware family which is persistent, may have rootkit capability to hide its presence, and employs various command-and-control mechanisms. Zeus malware is often used to track user activity and steal information by man-in-the-browser keystroke logging and form grabbing. Zeus malware can also be used to install CryptoLocker ransomware to steal user data and hold data hostage.
- **Others categories** – Other categories include Bedep, InstallCore, Mudrop, MultiPlug, Rerdom, Sality, Vawtrack, Vittalia, etc.

Features applied on URL, path, query, filename
length
digit ratio
lower case ratio
upper case ratio
ratio of digits
vowel changes ratio
ratio of a character with max occurrence
has a special character
max length of consonant stream
max length of vowel stream
max length of digit stream
number of non-base64 characters
has repetition of parameters
starts with number
Other Features
number of bytes from client to server
number of bytes from server to client
length of referer
length of file extension
number of parameters in query
number of '/' in path
number of '/' in query
number of '/' in referer
is encrypted

Table 2: List of selected flow-based features extracted from proxy logs. We consider these features as baseline (as some features were used in previously published work), and compare the baseline with the proposed representation.

The summary of the training and testing set is shown in Table 1. Positive bags from 8 categories with the highest number of bags were added to the training set, while the rest of the malware samples from the other categories (including novel threats) were included in the testing set. This means that training and testing data are composed of

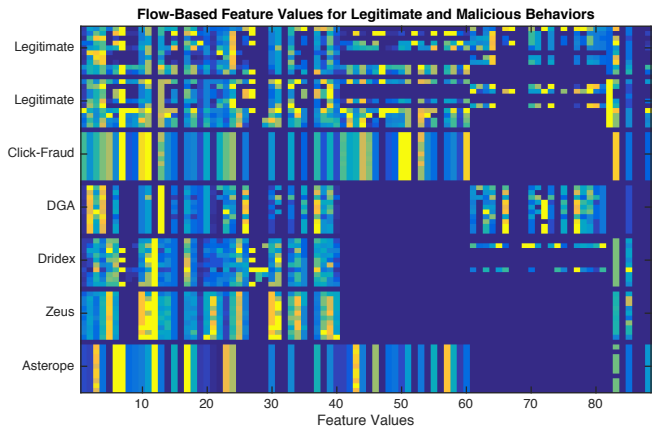

Figure 2: Graphical illustration of a baseline representation of two legitimate and five malicious bags. Each bag is represented with 10 flow-based feature vectors (rows). You can see that feature values of legitimate vectors have high range of values (represented by colors), which complicates the training of classifiers. However, they have also significantly higher variability of feature values than malicious bags, which cannot be utilized from flow-based representation.

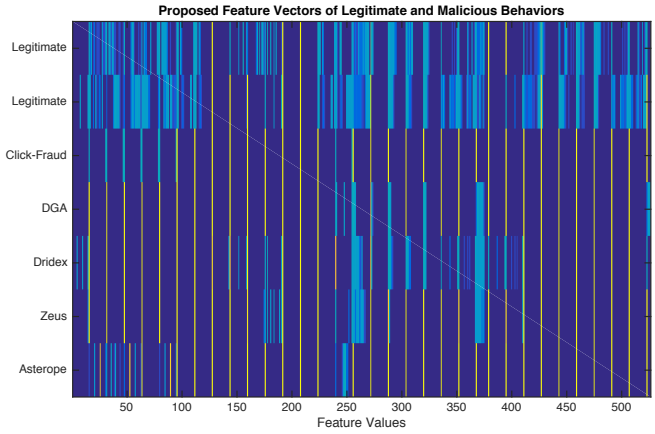

Figure 3: Illustration of the proposed invariant representation of bags. First two legitimate bags show high variability of feature values and can be easily separated from malicious bags. On the other hand, feature values of malicious bags describe the inner similarity of flows or URLs within each bag. In contrast to legitimate bags, malicious bags have a lot of common feature values equal to 0 and 1, which improves their separability.

completely different malware bags from different malware families. This makes the classification problem much harder, as the classifier is trained on 8 malware categories and then evaluated on malicious traffic of different categories and behaviors unseen in the training set. This scenario simulates the fact that new types of threats are created to evade detection. Anomaly detection system introduced in [12] automatically removes 38% of all malicious flows from the testing set, as they have empty URL query string. Training a classifier for each category separately is an easier task, however such classifiers are typically over-fitted to a single category and cannot detect further variations without retraining. On the other hand, training a classifier from multiple categories is more suitable for dynamic or polymorphic changes of new malware.

Negative bags were acquired from 10 hours of http network traffic of two companies. Negative bags from the first company were used for training, while bags from the second company were used for testing. Note that malicious bags found in the traffic were removed from the set of negative bags.

5.3 Evaluation on Real Network Traffic

This section shows the benefits of the proposed approach of learning the invariant representation for two-class classification problem in network security. Feature vectors described in Section 5.1 correspond to input feature vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ defined in Section 2. These vectors are transformed into the proposed representation of histograms $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$, as described in Section 3. We have evaluated two types of invariant representations. One with predefined number of equidistant bins (e.g. 16, 32, etc.) computed as described in Section 3, and one when the representation is learned together with the classifier to maximize the separability between malicious and legitimate traffic (combination of Section 3 and 4). For the representation learning, we used 256 bins as initial (and most detailed) partitioning of the histograms. During the learning phase, the bins were merged together, creating 12.7 bins per histogram on average.

Both approaches are compared with a baseline flow-based representation used in previously published work, where each sample corresponds to a feature vector computed from one flow. To provide a fair comparison of all representations, positive flow-based feature vectors are converted to positive bags. A bag is considered as positive if at least one flow from the bag is classified as positive. Maximum number of flows for each bag was 100, which ensures that the computational complexity is controlled and does not exceed the predefined limits.

First, we analyze the difference between baseline flow-based and the proposed invariant representation. Ten flow-based feature vectors of two legitimate and five malicious bags are displayed in Figure 2. Each row represents one flow-based feature vector. You can see that legitimate bags have higher diversity of flow-based feature values, which is a result of higher diversity of flows within a bag. This diversity makes it difficult for a flow-based classifier to learn more complex malicious behaviors, as they are not well separated from the legitimate traffic. On the other hand, feature values within malicious bags are more consistent, resulting in more bars with uniform color. This key property, which is shared across malware categories, is not visible from the flow-based feature point of view. It is visible only at the level of bags.

Figure 3 shows how the same positive and negative samples look in the proposed representation. Zero values are depicted with dark blue color, while maximum values (equal to 1) are depicted with yellow bars. Instead of ten flow-based feature vectors, each bag is represented with a single vector describing the inner dynamics of flow-based feature values within each bag. Malicious bags have a lot of values equal to zero as opposed to legitimate bags, which increases the separability of the two classes. Moreover, feature values equal to one are common for most of malicious bags across categories, which increases the descriptive value and robustness of the proposed representation. Such representation is more suitable for classification of frequently-changing malicious behaviors, as will be demonstrated further in this section.

Two-dimensional projection of the feature vectors for the flow-based and the proposed representation is illustrated in Figures 4 and 5 respectively. Bags from 32 malicious categories are displayed with red circles, while the legitimate bags are denoted with green circles. The projections show that the flow-based representation is suitable for training classifiers specialized on a single malware cat-

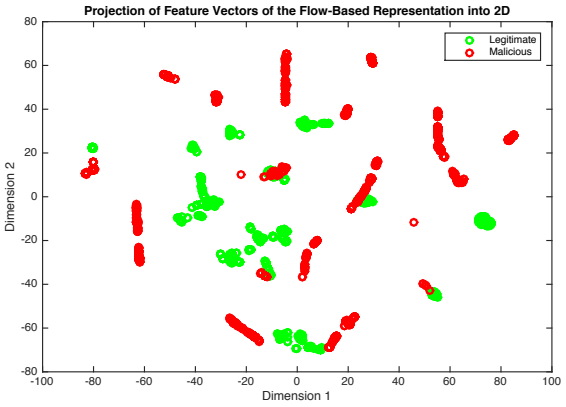

Figure 4: Projection of feature vectors of the baseline flow-based representation into two dimensions using t-SNE transformation. Due to high variability of flow-based feature values, legitimate (green) and malicious (red) samples are scattered without any clear separation. The results show that the flow-based representation is suitable for training classifiers specialized on a single malware category, which often leads to classifiers with high precision and low recall.

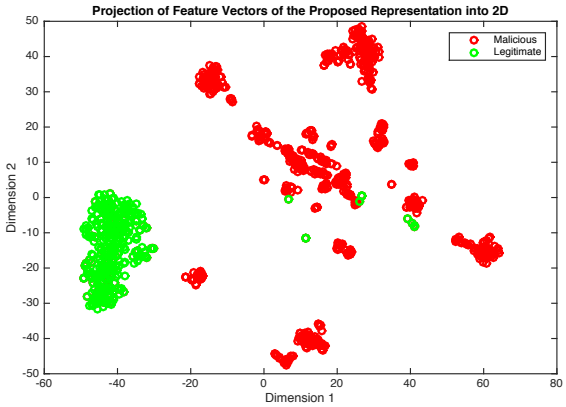

Figure 5: Projection of feature vectors of the proposed invariant representation into two dimensions using t-SNE (t-distributed Stochastic Neighbor Embedding) transformation. Thanks to the invariant properties, malicious bags from various categories are grouped together and are separated from the majority of legitimate samples. Classifiers trained from this representation will be more robust to any changes or new modifications of future malware samples.

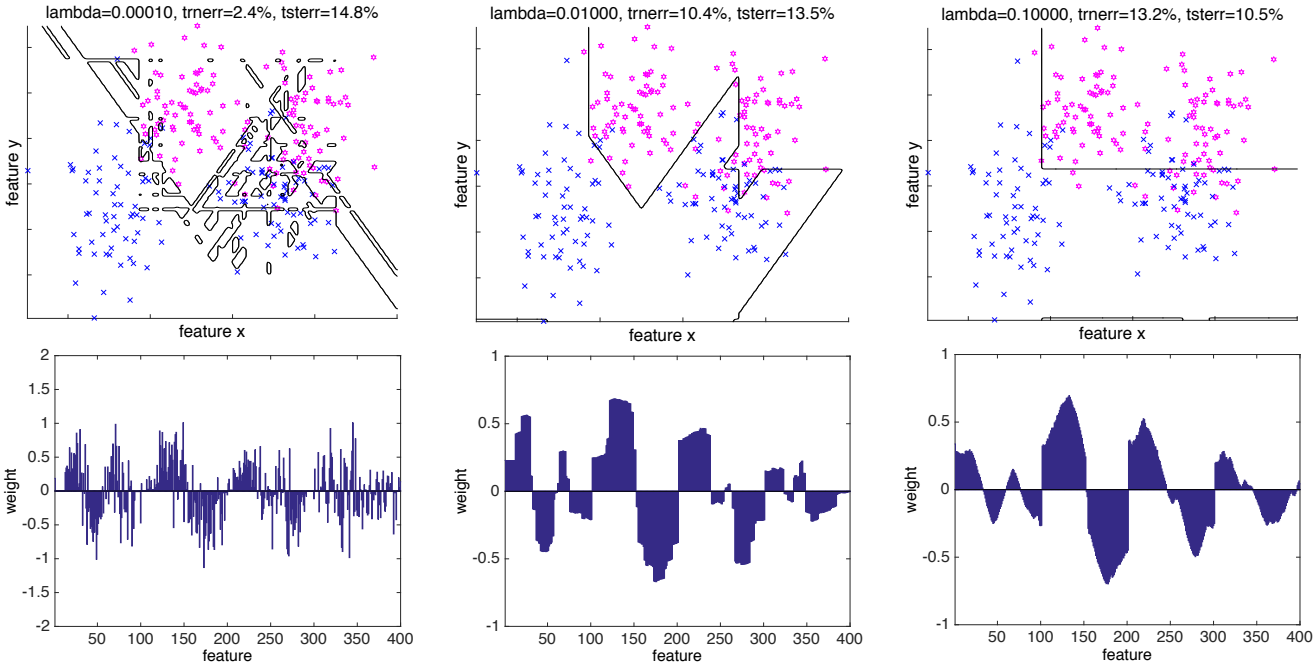

Figure 6: Visualization of the proposed method of learning the invariant representation on 2-dimensional synthetic data. Figures in the upper row show the decision boundaries of two class classifier learned from the bins for three different values of parameter λ (0.0001, 0.01, 0.1) which controls the number of emerging bins (the corresponding weights are shown in the bottom row). With increasing λ the data are represented with less bins and the boundary becomes smoother and less over-fitted to the training data.

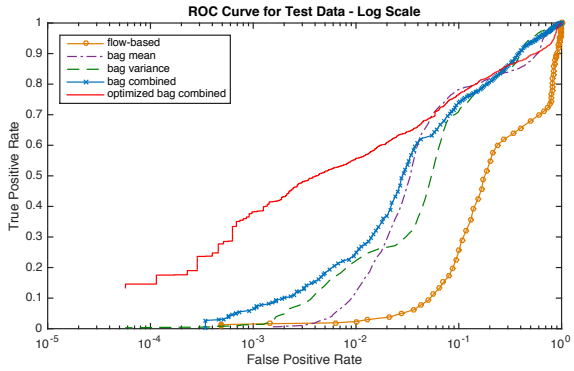

Figure 7: ROC curves of SVM classifier on test data for five types of representations (logarithmic scale). Flow-based representation shows very unsatisfactory results. The combination of feature values with feature differences histogram (bin combined) leads to significantly better efficacy results. The best results are obtained when the parameters are learned automatically from the data (optimized bag combined).

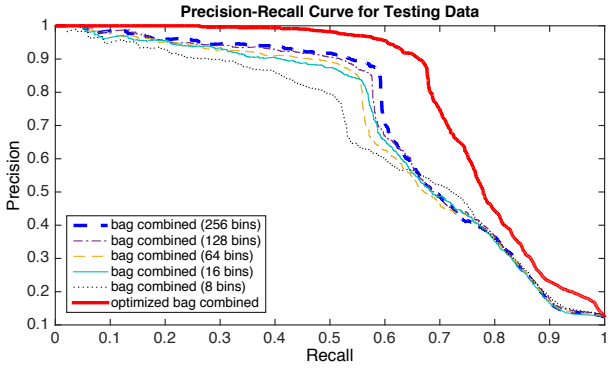

Figure 8: Precision-recall curve of SVM classifier trained on the proposed representation with different number of histogram bins for each feature. The efficacy of the classifiers improves as the number of histogram bins increases (creating more detailed representation), however all classifiers are outperformed by the classifier, where the parameters are learned automatically from the data (optimized bag combined).

egory. In case of the proposed representation, malicious bags from various categories are grouped together and far from the legitimate traffic, which means that the classifiers based on this representation will have higher recall and comparable precision with the flow-based classifiers.

Next, we will show the properties of the proposed method of learning the representation to maximize the separation between positive and negative samples (see Section 4 for details). Figure 6 visualizes the proposed method on synthetic 2-dimensional input data. The input 2D point $(x, y) \in \mathbb{R}^2$ is represented by 4-dimensional feature vector $(x^2, y^2, x + y, x - y)$. Each of the 4 features is then represented by a histogram with 100 bins (i.e. each feature is represented by 100 dimensional binary vector will all zeros but a single one corresponding to the active bin). Figures in the top row show the decision boundaries of two-class classifiers learned from data. The bottom row shows the weights of the linear classifier corresponding to the bins (in total 400 weights resulting from 100 bins for each out of 4 features). The columns correspond to the results obtained for different setting of the parameter λ which controls the number of emerging bins and thus also the complexity of the decision boundary. With increasing λ the data are represented with less bins and the boundary becomes smoother. Figure 6 also shows the principle of the proposed optimization process. The bins of the representation are learned in such a way that it is much easier for the classifier to separate negative and positive samples and at the same time control the complexity of the classifier.

Since the bins are equidistant and predefined at the beginning, the weights and the resulting histogram has complicated structure, leading most probably to complex boundary and over-fitted results (as shown in Figure 6 on the left hand side). As the bins are merged, the weights show a clear structure and the derived histogram has significantly less bins. The decision boundary is in this case smoother and the classifier trained from this representation will be more robust.

The results of the SVM classifier on testing data are depicted in Figure 7. Flow-based representation has worst results, mainly due to the fact that the classifier was based only on the values of flow-based features that are not robust across different malware categories. The SVM classifier based on the invariant bag representation (bag mean, beag variance, bag combined) performed significantly better. However, these results are furhter outperformed when using the proposed combination of the invariant representation with automatically optimized parameters according to Section 4.

Figure 8 compares the efficacy results of classifiers based on the proposed representation with predefined number of bins per feature (8, 16, 64, 128, and 256 bins) with the same representation, but when the parameters are learned from the training data (Section 4). You can see that increasing the number of bins in the histograms improves the precision-recall curve of the classifier. Finally, all classifiers trained from histograms with equidistant bins are outperformed by the proposed combination of learning the classifier and the representation parameters alltogether, achieving 90% precision and 67% recall on malware samples of previously unseen variants.

Overall, the results show the importance of combining both types of histograms introduced in Section 3 together, allowing the representation to be more descriptive and precise without sacrificing recall. But most importantly, when the parameters of the representation are trained to maximize the separability between malicious and legitimate samples, the resulting classifier performs in order of a magnitude better than a classifier with manually predefined parameters.

6 CONCLUSION

We proposed a robust representation suitable for classifying evolving malware behaviors. It represents sets of connections as bags based on the combination of invariant histograms of feature values and feature differences. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. The proposed optimization method learns the parameters automatically from the data. It solves the domain adaptation problem in a supervised learning, where the training and testing datasets have different probability distributions.

The proposed representation was evaluated on real HTTP network traffic with more than 43k malicious samples and more than 15M samples overall. The comparison with a baseline approach showed an order of magnitude better classification results in favor of the proposed approach. The proposed classifier trained on the optimized representation achieved 90% precision and detected 67% (29k) of malware samples of previously unseen variants.

REFERENCES

[1] Karel Bartos and Michal Sofka, *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III*, chapter Robust Representation for Domain Adaptation in Network Security, 116–132, Springer International Publishing, Cham, 2015.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al., 'Analysis of representations for domain adaptation', *Advances in neural information processing systems*, **19**, 137, (2007).

[3] John Blitzer, Ryan McDonald, and Fernando Pereira, 'Domain adaptation with structural correspondence learning', in *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128. Association for Computational Linguistics, (2006).

[4] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu, 'Boosting for transfer learning', in *Proceedings of the 24th international conference on Machine learning*, pp. 193–200. ACM, (2007).

[5] Lixin Duan, Ivor W Tsang, and Dong Xu, 'Domain transfer multiple kernel learning', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34**(3), 465–479, (2012).

[6] G. Farnham and K. Leune, 'Tools and standards for cyber threat intelligence projects', Technical report, SANS Institute InfoSec Reading Room, (10 2013).

[7] Vojtech Franc, Michal Sofka, and Karel Bartos, *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III*, chapter Learning Detector of Malicious Network Traffic from Weak Labels, 85–99, Springer International Publishing, Cham, 2015.

[8] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf, 'Covariate shift by kernel mean matching', *Dataset shift in machine learning*, **3**(4), 5, (2009).

[9] Luca Invernizzi, Stanislav Miskovic, Ruben Torres, Sabyasachi Saha, SJ Lee, Marco Mellia, C Kruegel, and Giovanni Vigna, 'Nazca: Detecting malware distribution in large-scale networks', in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, (2014).

[10] Arun Iyer, Saketha Nath, and Sunita Sarawagi, 'Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection', in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 530–538, (2014).

[11] Imran N Junejo, Emilie Dexter, Ivan Laptev, and Patrick Perez, 'View-independent action recognition from temporal self-similarities', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(1), 172–185, (2011).

[12] Christopher Kruegel and Giovanni Vigna, 'Anomaly detection of web-based attacks', in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pp. 251–261, New York, NY, USA, (2003). ACM.

[13] Wenwu Lou, Guimei Liu, Hongjun Lu, and Qiang Yang, 'Cut-and-pick transactions for proxy log mining', in *Advances in Database Technology EDBT 2002*, eds., ChristianS. Jensen, Simonas altenis, KeithG. Jeffery, Jaroslav Pokorny, Elisa Bertino, Klemens Bhn, and Matthias Jarke, volume 2287 of *Lecture Notes in Computer Science*, 88–105, Springer Berlin Heidelberg, (2002).

[14] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker, 'Learning to detect malicious urls', *ACM Trans. Intell. Syst. Technol.*, **2**(3), 30:1–30:24, (May 2011).

[15] Meinard Müller and Michael Clausen, 'Transposition-invariant self-similarity matrices.', in *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pp. 47–50, (2007).

[16] Hidetoshi Shimodaira, 'Improving predictive inference under covariate shift by weighting the log-likelihood function', *Journal of statistical planning and inference*, **90**(2), 227–244, (2000).

[17] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang, 'Domain adaptation under target and conditional shift', in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, eds., Sanjoy Dasgupta and David Mcallester, volume 28, pp. 819–827. JMLR Workshop and Conference Proceedings, (2013).