2018 IEEE Symposium on Security and Privacy Workshops

Extending Detection with Privileged Information via Generalized Distillation

Z. Berkay Celik and Patrick McDaniel

SIIS Laboratory, Department of CSE, Pennsylvania State University
{zbc102, mcdaniel}@cse.psu.edu

***Abstract*—Detection systems based on machine learning models are essential tools for system and enterprise defense. These systems construct models of attacks (or non-attacks) from past observations (i.e., features) using a training algorithm. After that, the detection systems use that model for detection at run-time. In this way, the detection system recognizes when the environmental state becomes—at least probabilistically—dangerous. A limitation of this traditional model of detection is that model training is limited to features available at run-time. However, many features are either too expensive to collect in real-time or only available after the fact. In traditional detection, such features are ignored for the purpose of detection. In this paper, we consider an alternative detection model learning approach, generalized distillation, that trains models using privileged information—features available at training time but not at run-time—to improve the accuracy of detection systems. We use a deep neural network to implement generalized distillation for the training of detection models and making predictions. Our empirical study shows that detection with privileged information via generalized distillation increases precision and recall in systems of user face authentication, fast-flux bot detection, and malware classification over systems with no privileged information.**

***Index Terms*—Detection systems, privileged information, generalized distillation, deep learning**

I. Introduction

Detection systems use machine learning (ML) algorithms such as support vector machines and neural networks to learn detection models. These models aim at learning patterns to estimate an underlying dependency, structure or behavior of a system with a limited number of features (also referred to as inputs) from historical data (also referred as training data). For instance, in network malware detection, the features can be obtained from packets of incoming/outgoing traffic [1], and in mobile phone malware detectors, features can be obtained from user permissions [2]. Yet, a limitation of this traditional model of detection is that the detection systems use features that will be available at run-time. However, many features are either too expensive to collect in real-time or only available after the fact. Therefore, in traditional detection, such features are ignored for the purposes of detection.

Consider a security data repository that stores a myriad of information from packets, log files and other sources. For example, security information and event management (SIEM) systems capture raw data from web proxies, DHCP servers, VPN servers, and more at rates of up to 100K events per second stores up to 42 TB of data [3]. However, much of the useful information only becomes available after further investigation and human processing [4], [5]; thus it cannot be leveraged by detection systems in real-time. In other contexts, features may be available at run-time but infeasible or undesirable to collect because of environmental or system constraints. For example, a large number of features collection in mobile phones [2], Internet of Things (IoT) [6] is often too slow or requires too many resources to be feasible in practice.

We turn recent advances in learning theory that support learning models on a superset of features used at run-time [7], [8]. Our goal is to leverage these additional features, called *privileged information*, features available at training time, but not at run-time, to improve the accuracy of detection. More concretely, we explore an alternate approach *generalized distillation* [9] to train detection systems that exploit privileged information. Generalized distillation is an extension of distillation [10] or model compression [11] designed for Deep Neural Networks (DNNs). At training time, generalized distillation creates a novel *distilled* model that improves the model training by transferring knowledge from privileged features available at training time. At detection time, the distilled models trained with generalized distillation do not require all features; thus a new sample without privileged information is evaluated with a distilled model. The motivation behind the distilled models is to build a detection model with better generalization capabilities which retains the detection accuracy sufficiently close to the accuracy of a model trained and tested on the complete features. In this, we make the following contributions:

- We consider generalized distillation to train detection models with privileged information—features available at training time, but not at run-time.
- We present several privileged information-augmented detection systems. This highlights the inherent tension between information utilization, detection accuracy, and robustness of a system.
- We empirically validate generalized distillation in a range of detection domains. The use of privileged information via generalized distillation decreases the relative detection error on average 3.53% for fast-flux domain bot detection, 3.33% for malware classification, 11.2% for face authentication over benchmarked systems that do not use privileged information.

DOI 10.1109/SPW.2018.00021

II. Generalized Distillation

Model compression [11] or *distillation* [10] are techniques to reduce the size of statistical models. Model distillation compresses large models f_{large} by training a small model $f_{\text{small}}(x)$ that imitates the predictions of the large model $f_{\text{large}}(x)$. Remarkably, model distillation is often able to compress models without incurring any loss in accuracy [10].

Consider feature-target pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ is a vector of d features describing the i-th sample, and $y_i \in \mathbb{R}$ is the target defined for a sample. Distillation assumes that the large model f_{large} has been learned by minimizing a model, and proceeds to learn the small model f_{small} by minimizing

$$(1-\lambda)\mathcal{L}(\{(x_i, y_i)\}_{i=1}^n, f_{\text{small}}) + \lambda\mathcal{L}(\{x_i, s_i\}_{i=1}^n, f_{\text{small}}), \quad (1)$$

where $\lambda \in [0,1]$ is an *imitation parameter* trading-off how much does the small model imitate the big model, versus directly learning the data. In Equation 1, a second dataset is introduced with targets $s_i = f_{\text{large}}(x_i)/T$; these are the *softened predictions* made by the large model. Here, $T > 0$ is a temperature parameter scaling the predictions of the large model[1].

Lopez-Paz et al. recently introduced *generalized distillation*, an extension of model distillation. Generalized distillation compresses the models built on a set of features into models built on a different set of features [9]. It is one specific instance of *learning using privileged information* (LUPI) [7], [8], a learning paradigm assuming that some of the features used to train a machine learning model will be unavailable at run-time[2].

Generalized distillation assumes that the statistical model will be trained on some data $\{(x_i, x_i^\star, y_i)\}_{i=1}^n$, and that it will be tested on some data $\{x_j\}_{j=n+1}^{n+m}$. Therefore, the set of features $\{x_i^\star\}_{i=1}^n$ is available at training but not at test time. However, these features may contain important information that would lead to machine learning models of higher accuracy. Generalized distillation tackles the problem of learning using privileged information as follows (see Figure 1): First, it trains a privileged model $f_{\text{priv}}(x)$ on the complete feature-target set $\{(x_i, x_i^\star, y_i)\}_{i=1}^n$ which have an access to privileged features at training. Second, it trains a *distilled* model f_{dist} by minimizing Equation 1 where $s_i = f_{\text{priv}}(x_i)/T$.

The imitation parameter λ in a distilled model controls the trade-off between privileged features and accuracy. For $\lambda \approx 0$, the objective of the distilled model approaches the standard objective which amounts to learning a model solely on standard features. However, as $\lambda \to 1$, the objective transfers the knowledge acquired by the privileged model into the distilled model. The intuition here is that whenever the privileged model makes a prediction error on a sample, the distilled model should forget about getting the detection right on that sample, and focus on the rest of the samples. Therefore, the distilled model learns by simultaneously imitating the privileged predictions of the privileged model and learning the targets of original data.

[1]The temperature parameter was recently interpreted as a defense mechanism to adversarial data perturbations in DNNs [12].

[2]We refer the reader our technical report for implementation and formulation of LUPI paradigm through SVM+ [13] and our recent paper for application of LUPI paradigm of knowledge transfer and SVM+ in a security setting [14].

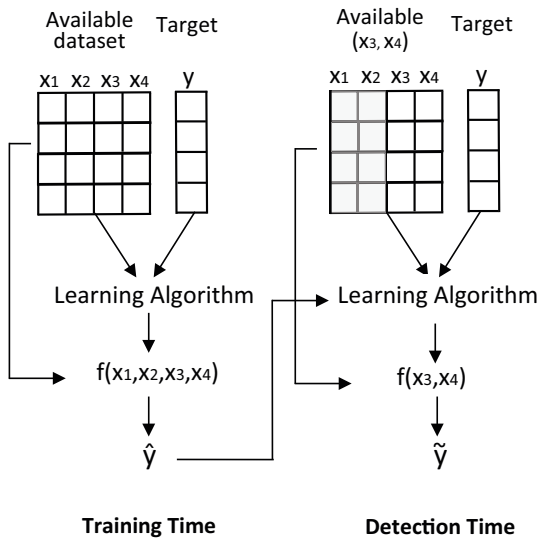

Fig. 1: Scheme of generalized distillation. The distilled model $f(x_3, x_4)$ learns a model using the standard features, since it only requires access to available features (x_3, x_4) at detection time. However, the distilled model was trained to imitate the predictions of a model $f(x_1, x_2, x_3, x_4)$, which have an access to the privileged features (x_1, x_2) available at training time. Through this transference of knowledge, the distilled model provides more accurate detection results than the models trained using only the standard features (x_3, x_4), and similar results to the models assuming access to the complete data (x_1, x_2, x_3, x_4).

These *teachings* from the privileged model do, in many cases, significantly help the learning process of the distilled model.

Privileged features– The first challenge to build detection models via generalized distillation is determining which features should be used as privileged. Asked another way, *given some potentially large set of features, which are the most likely not available at run-time and will improve detection?* To address this, we define the privileged features in a security domain that represent the relevant features of a detection task, yet they have constraints on obtaining them at run-time. The main constraints includes but not limited to the high resource consumption, computational overhead, and semantic information produced by human experts. Therefore, the acquisition of the privileged features are often too slow, available after the fact, or require too many resources to be feasible in practice. Using this definition, we determine the privileged features that include useful information for detection, yet they have one of the constraints above at run-time. We note that defining privileged features sometimes requires a level of domain expertise. However, trained security experts will find most of the privileged features straightforward after determining the run-time constraints on features. We believe that privileged information will change the way of building new datasets in complement with the existing datasets, and in turn, make it easier to construct systems with complete related

System	Goal	Standard features	Privileged features	Constraints on privileged features at detection time
User Face Authentication	Authenticate users from their face images	User face images	Bounding boxes, and cropped versions of the face images	(1) Human efforts and additional software is required to acquire features, (2) Processing overhead in low energy sensors prevents acquiring features
Fast-flux Bot Detection	Detect fast IP-changing domain names	A and NS record information from DNS packets, the processing time of servers	Features from domain names, spatial, and network features	Time-consuming and resource intensive operations (domain name processing, WHOIS processing, and IP coordinate database lookup) prevents real-time detection
Malware Classification	Classify malware binaries to their respective families	Frequency count of hexadecimal duos of the binary contents	Frequency count of meta-data information extracted from the assembler	Software-dependency of obtaining assembly source code introduces a computational overhead and error-prone feature extraction

TABLE I: Description of detection systems. Each system is depicted with its standard features and privileged features. We incorporate the privileged features into detection system via generalized distillation.

privileged features. A recent example is the Animals with Attributes dataset which has 30K images of 50 animal classes, annotated with 85 semantic attributes like color, texture, shape, and behavior among others [15]. The images are considered as standard, and semantic attributes are as privileged.

Motivating Example– We consider a malware detector designed for mobile phones that monitors system-level features such as capabilities of the hypervisor and the nonexecutable page table at run-time to find malicious rootkits [2]. However, some features have high energy costs and induce noticeable interface lag. This drains the battery and causes users disable the detection mechanism to save power. In the following, let us denote all the features required for rootkit detection by (x_1,x_2,x_3,x_4), being (x_1,x_2) the "system-level features". Thus, a system considers the "no system-level features" (x_3,x_4) *standard* as they are available at both at training and run-time of the detection system, and the features (x_1,x_2) *privileged* as they are not available at run-time. However, it is reasonable to assume that the system-level features $(x_i^\star)$ contain useful information to identify malicious rootkits, but such information will be unavailable at run-time. Once privileged features are defined, we built an ML model through generalized distillation. Generalized distillation incorporates privileged features into ML models without requiring them at run-time. The learning process proceeds as follows: a model $\hat{y} = f(x_1,x_2,x_3,x_4)$ is trained that uses *all* the features to estimate the malicious rootkits $\hat{y}$. Second, another model $\tilde{y} = f(x_3,x_4)$ is trained, this time only on "no system-level features". However, the model $\tilde{y} = f(x_3,x_4)$ is built to learn the true malicious rootkits available in our database, *as well as to imitate* $\hat{y} = f(x_1,x_2,x_3,x_4)$ *predicted by the privileged model that uses all the features*. Finally, when a new sample belonging to the "no system-level features" comes for detection, we can use the model $\tilde{y} = f(x_3,x_4)$ to predict the probability of a malicious rootkit while we preserve the low energy costs and prevent the interface lag.

III. Experimental Detection Systems

We present our efforts to build privileged-augmented systems from recently proposed systems for evaluation of distilled models. We show models of three detection systems through generalized distillation: (1) user face authentication, (2) fast-flux bot detection, and (3) malware classification. Table I summarizes the goal of detection systems, shows their standard, and privileged features, and the constraints on privileged features at run time. Our goal is evaluating the capacity of privileged features via generalized distillation to analyze efficacy on detection performance. In particular, a system's reverse engineering should let us process the privileged features in addition to its standard features. Thus, we consider the selection of privileged features is complete as long as we can obtain as many privileged features as we can from a system.

Feature definition	Run-time dependency	Type*
Number of unique A records Number of NS records	DNS packet headers	✓
Network delay[1] Processing delay[1] Document fetch delay[1]	HTTP requests	✓
Edit distance KL divergence[2] Jaccard similarity[2]	Whitelist of benign domain names	✗
Time zone entropy of A records Time zone entropy of NS records Minimal service distances[1]	IP coordinate database lookup	✗
Number of distinct autonomous systems Number of distinct networks	WHOIS processing	✗

* ✓: standard features, ✗: privileged features.
[1] Both mean and std. deviation are computed.
[2] Both unigrams and bigrams are computed.

TABLE II: Fast-flux bot detection system standard and privileged feature descriptions.

User Face Authentication– We implement a face authentication system to recognize a facial image corresponding to the individual depicted in the image. We use a subset of the images with three RGB channels in the Labeled Faces in the Wild dataset [16]. The dataset includes 1348 images with at least 50 images per person. We build the standard features from human facial images, and add bounding box of cropped faces and funneled face images as privileged features. These images aim at specifying the facial localization of the faces which gives useful information about each users face by eliminating statistical correlation from the background noise. However, the privileged features of the images can be obtained with the help of commercial software and human processing and may not be available in low energy and slow processing sensors [16]–[18].

Fast-flux Bot Detection– Fast-flux servers are employed by attackers to hide the actual IP addresses of the servers used for malicious activities. We built the dataset of a recent fast-flux detector [19] that includes features acquired from the domain names, DNS packets, packet timing intervals, WHOIS and

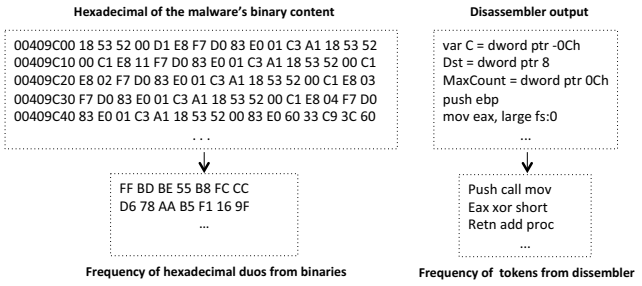

Fig. 2: An example of obtaining the standard (Left) and privileged features (Right) of the malware classification system.

IP coordinate database [20], [21] (see Table II). The dataset includes 4 GB benign and malicious fast-flux DNS requests collected in early 2013 [19]. However, processing WHOIS records, finding the KL-Divergence and Jaccard similarity of a domain name from a whitelist of domain names, and IP coordinate database lookup takes several minutes/hours to process and update. These features introduce time-consuming and resource-intensive operations at run-time and prevent real-time detection. Therefore, we define such features as privileged to assure real-time detection.

Malware Classification– We use Microsoft malware classification challenge dataset [22] to classify malware samples to their correct family. The dataset consists of 1746 malware samples with nine classes. Each malware sample includes byte and metadata files and a class label. The byte files include the raw hexadecimal representation of the malware binary contents. We use byte files to construct the standard features by counting the frequencies of each hexadecimal duos (i.e., byte bigrams) (see Figure 2). The byte bigrams provide discriminating characteristics between different families and low computational complexity for real-time detection [23]. The metadata manifest includes the assembly logs of IDA disassembler tool [24]. The distinct tokens such as `mov()`, `cmp()` from manifest files capture the different execution of malware families [23]. However, processing the manifest file may prevent real-time automated malware classification because of the disassembler overhead [25]. Additionally, different disassembler versions may interpret byte sequences of malware differently, and lead to inaccurate feature extraction. Therefore, similar to the byte files, we add tokens frequencies acquired from the malware manifest files as privileged features.

IV. Evaluation

Evaluation Setup– We obtain the standard features of the detection systems either by reverse engineering them or using their publicly released datasets. We then learn three models. First, *complete model* uses both standard and privileged features for training and validation. Second, the *standard model* trains and validates the model on standard features. The third model is the *distilled* which trains the model on both standard and privileged features via generalized distillation, yet it uses standard features for the purpose of detection. We train

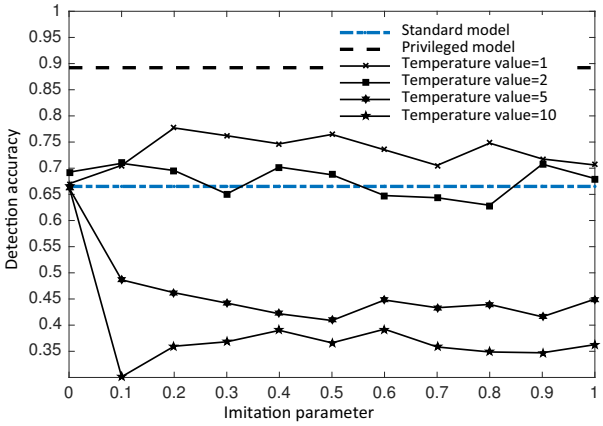

Fig. 3: Visualizing impact of privileged features on detection accuracy of user face authentication system: Accuracy of standard and privileged models are plotted as a baseline. The impact of temperature (T) and imitation (λ) parameters is quantified on various values.

privileged models using a deep neural network with two hidden layers of 10 rectifiers linear unit each using the Keras libary [26] running the Theano backend [27]. This type of architecture is commonly applied in security and privacy settings [28] and gives affordable results in analyzed datasets. We split the datasets into two cohorts; the training cohort is used to learn the models, and the validation cohort is used to evaluate the models. We show the experimental results of generalized distillation with an imitation parameter values of $\lambda \in [0,1]$ and $T \in \{1,2,5,10\}$. Our goal is to identify the optimal T and λ parameters leading to maximum detection gain in systems. The performance of the systems is presented with three metrics; accuracy, recall, and precision. Accuracy is the sum of the true positive and true negatives over a total number of samples. The recall is the number of true positives over the sum of false negatives and true positives, and precision is the number of true positives over the sum of false positives and true positives. Higher values of accuracy, precision, and recall indicates a higher quality of the detection.

Evaluation Results– We begin with the result of the generalized distillation on the user face authentication system. We note that in some cases background of the images may unrealistically increase the facial recognition of a user because of the distinguishing effect on the background regions of a face image. However, we manually verified that the images in our dataset do not suffer from this effect.

Generalized distillation improves the accuracy of the face authentication system compared to models built on standard features. Figure 3 plots the average model accuracy of standard, privileged models, and distilled model with varying imitation and temperature values. We note that all models are implemented using a DNN architecture defined in the evaluation setup. Learning with only privileged features achieves on average 89.2% correct classification rate, which is better than

		Fast-flux Bot Detection			Malware Classification		
	Model	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Complete Model	RF	98.99±0.88	99.4	99.4	96.6±1.2	99.3	95.2
	SVM	99.46±0.3	99.34	100	95.68±1	98.63	94.62
Standard Model	RF	96.5±2.57	98.68	96.75	91.2±0.97	91.34	94.35
	SVM	95.0±2.3	94.38	95.5	91.78±1.1	93.17	93.55
Distilled Model	DNN	**97.47 ±0.3**	97.35	99.32	**92.56 ±0.7**	92.57	95.31

TABLE III: Accuracy, precision, and recall of detection systems with optimal temperature (T) and imitation (λ) parameters.

the 66.5% of the standard model. We measure the distilled model accuracy with various temperature T and imitation λ parameters. Distilling the privileged features at specific T and λ parameters gives better detection accuracy over the standard model. The accuracy is maximized when T is between 1 and 3 for the most of the λ values. The accuracy improvement is on average 6.56% when $T = 1$ and it is maximal when $\lambda = 0.2$. This yields 11.2% increase in accuracy compared to the model built on standard features. However, the more increase in temperature parameter degrades the detection accuracy. This is because increasing the temperature parameter after a saturating point causes the distribution of the class probabilities be smoother, which, in turn, prevents the proper weighting between standard and privileged features.

We conduct similar experiments on the fast-flux bot detection (FF) and malware classification (MC). In these set of experiments, we use Random Forrest Classifier (RF) and Support Vector Machines (SVM) to train standard, and complete models because these classifiers give better results than other ML algorithms and also preferred by the authors of the detection systems. We apply grid search to find the optimal parameters of the SVM and RF models. Distilled models of FF and MC detection systems decrease both false positives and negatives over benchmark systems that do not use privileged features (i.e., standard models). Table III shows the generalized distillation results and compares it with the standard and complete models of SVM and RF. We observe that by distilling at the proper temperature and imitation parameters, we improve the detection results in both systems. In FF, the distilled model at T = 2 and λ = 0.6 yields 97.47% accuracy which is 1.99% less than the SVM complete model and 2.47% more than the SVM standard model. In MC, the distilled model at T = 5 and λ = 0.3 yields 92.56% which is 4.04% less than the RF complete model and 1.36% more than the RF standard model. We note that the impact of various temperature and imitation parameters in these datasets lower than the face authentication results. In the worst case, we observe on average 8.2% less accuracy than the standard model accuracy.

Our evaluation of three detection systems showed that distilled models reduce both false positives and negatives compared to the systems solely built with standard models. In a security setting, false positives make extremely difficult for the analyst that examines the reported incidents in order to identify the mistakenly triggered benign events correctly. On the other hand, false negatives may have a potential to cause dreadful damage to both users and organizations. For instance, in fast-flux bot detection, a false negative may allow a bot send private data to a malicious server. In malware classification and face authentication systems, it weakens the integrity of a system by authenticating the wrong user or misclassifying a malware family sample into an incorrect family. Thus, the use of privileged features via generalized distillation does matter in improving the false positives and negatives of these systems.

V. Discussion and Limitations

We applied generalized distillation, a technique of distilling the knowledge of privileged features as class probability vectors into the standard features of a detection system. The preceding analysis of generalized distillation demonstrated that privileged features enhance a detection model's generalization capabilities outside of their standard features. However, two crucial factors, model selection and parameter tuning, need to be addressed to maximize the detection accuracy. For the former, the objective of generalized distillation can be minimized using an arbitrary model that needs be picked carefully. For the latter, the imitation λ and temperature parameter T should be tuned carefully to find the optimal distilled model. We note that parameter tuning is not a limitation of generalized distillation, yet applies to all machine learning algorithms that require parameters. On the other hand, we validated generalized distillation in a supervised learning setting. However, its objective can be formulated in other learning models such as semi-supervised, transfer and universum learning [9], [28]. Indeed, Jonschkowski et al. have provided a discussion of how privileged information can be adapted to other learning settings [29].

VI. Related Work

We validated the performance of generalized distillation in detection systems of malware classification, fast-flux bot detection, and user face authentication. Feature cultivation in these detection systems has been a key effort within the security communities. For example, researchers have previously used specific patterns to group malware samples into families [23], [30], have explored using DNS information to understand and predict botnet domains [20], [31], [32], and have analyzed system and network level features to identify malware traffic [33]–[35]. Other works have focused on user authentication using the facial images [36], [37]. These detection systems can integrate privileged features into their standard features of their detection models to strike a balance between accuracy and run-time constraints.

The privileged information has recently used in a few others domains such as computer vision, healthcare, image processing, and finance. Wang et al. [38] and Sharmanska et al. [39] used privileged features of images in the form of manually derived attributes, textual descriptions and object bounding boxes. Celik et al. used privacy-sensitive features as privileged in a healthcare statistical model [28]. Niu et al. used privileged-augmented robust classifiers for action and event recognition [40]. Ribeiro et al. used annual turnover and global balance values as

a privileged in order to improve the financial models [41]. However, their approaches are not intended to work in a security setting, yet they determine whether there is a possibility of improvement using the domain-specific features.

VII. Conclusions

We explored the application of generalized distillation to learn a detection model that exploits privileged information—features available at training time, but not at run-time. Generalized distillation is a learning meta-algorithm which allow systems to learn models using features that are not available at run-time. By reusing the knowledge from privileged features available at training time, generalized distillation showed nearly optimal detection, competing with the idealized models that assume access to complete data required for detection. We showed that distilling privileged features into the recently proposed three detection models of user face authentication, fast-flux bot detection, and malware classification compete with the idealized models that assume access to complete features.

Our future efforts will attempt designing new systems that benefit from recent advances in learning theory to learn better detection models. At a higher level, we will provide a guideline for optimal temperature and imitation parameter selection in a principled way as well as feature cultivation in distilled models.

Acknowledgments

We thank Dr. David Lopez-Paz for his constructive comments on generalized distillation and Dr. Rauf Izmailov for his feedback on the application of Learning using Privileged Information (LUPI) paradigm. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Z. B. Celik, J. Raghuram, G. Kesidis, and D. J. Miller, "Salting public traces with attack traffic to test flow classifiers," in *Usenix CSET*, 2011.
[2] J. Bickford *et al.*, "Security versus energy tradeoffs in host-based mobile malware detection," in *Mobile systems, applications, and services*, 2011.
[3] ArcSight Data Platform, http://www8.hp.com/us/en/software-solutions/arcsight-logger-log-management/, 2017, [Online; accessed 9-May-2017].
[4] A. A. Cardenas, P. K. Manadhata, and S. P. Rajan, "Big data analytics for security," *IEEE System Security*, 2013.
[5] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, 2015.
[6] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, 2013.
[7] V. Vapnik and R. Izmailov, "Learning using privileged information: Similarity control and knowledge transfer," *Journal of Machine Learning Research*, 2015.
[8] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Networks*, 2009.
[9] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," *ICLR*, 2016.
[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[11] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014.
[12] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," *IEEE S&P*, 2016.
[13] Z. B. Celik, R. Izmailov, and P. McDaniel, "Proof and Implementation of Algorithmic Realization of Learning Using Privileged Information (LUPI) Paradigm: SVM+," NSCR, Department of CSE, Pennsylvania State University, Tech. Rep. NAS-TR-0187-2015, Dec. 2015.
[14] Z. B. Celik, P. McDaniel, and R. Izmailov, "Feature cultivation in privileged information-augmented detection," in *ACM CODASPY International Workshop on Security And Privacy Analytics*, 2017.
[15] C. H. Lampert *et al.*, "Learning to detect unseen object classes by between-class attribute transfer," in *CVPR*, 2009.
[16] G. B. Huang *et al.*, "Labeled faces in the wild: Updates and new reporting procedures," UMASS, Tech. Rep. UM-CS-2014-003, May 2014.
[17] L. Wolf, T. Hassner, and Y. Taigman, "Effective unconstrained face recognition by combining multiple descriptors and learned background statistics," *Pattern Analysis and Machine Intelligence*, 2011.
[18] V. J. and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," UMASS, Tech. Rep. UM-CS-2010-009, 2010.
[19] Z. B. Celik and S. Oktug, "Detection of Fast-Flux Networks using various DNS feature sets," in *ISCC*, 2013.
[20] S. Yadav, A. K. K. Reddy *et al.*, "Detecting algorithmically generated malicious domain names," in *ACM Internet measurement*, 2010.
[21] S. Huang *et al.*, "Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection," in *ASIACCS*, 2010.
[22] Microsoft Malware Classification Challenge, https://www.kaggle.com/c/malware-classification/, 2017, [Online; accessed 10-May-2015].
[23] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," *Data and Application Security and Privacy*, 2016.
[24] IDA Pro: Disassembler and Debugger, http://www.hex-rays.com/idapro/, 2017, [Online; accessed 9-Feb-2017].
[25] L. Nataraj, V. Yegneswaran *et al.*, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Security and Artificial Intelligence Workshop*. ACM, 2011.
[26] Keras: Theano-based deep learning library, https://github.com/fchollet, [Online; accessed 10-January-2018].
[27] J. Bergstra *et al.*, "Theano: A cpu and gpu math compiler in python," in *Python for Scientific Computing Conference (SciPy)*, 2010.
[28] Z. B. Celik, D. Lopez-Paz, and P. McDaniel, "Patient-driven privacy control through generalized distillation," *IEEE Symposium on Privacy-Aware Computing*, 2016.
[29] R. Jonschkowski, S. Hfer, and O. Brock, "Patterns for learning with side information," 2015.
[30] M. Z. Rafique and J. Caballero, "Firma: Malware clustering and network signature generation with mixed network behaviors," in *RAID*, 2013.
[31] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: detecting the rise of DGA-based malware," in *USENIX Security*, 2012.
[32] L. Bilge *et al.*, "Exposure: Finding malicious domains using passive DNS analysis." in *NDSS*, 2011.
[33] B. Rahbarinia *et al.*, "Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks," in *IEEE DSN*, 2015.
[34] Z. B. Celik, R. J. Walls, P. McDaniel, and A. Swami, "Malware traffic detection using tamper resistant features," in *IEEE MILCOM*, 2015.
[35] Z. B. Celik, P. McDaniel, and T. Bowen, "Malware modeling and experimentation through parameterized behavior," *Journal of Defense Modeling and Simulation*, 2017.
[36] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," *British Machine Vision*, 2015.
[37] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very dnns," *arXiv preprint arXiv:1502.00873*, 2015.
[38] Z. Wang and Q. Ji, "Classifier learning with hidden information," in *IEEE Computer Vision and Pattern Recognition*, 2015.
[39] V. Sharmanska *et al.*, "Learning to rank using privileged information," in *International Conference on Computer Vision (ICCV)*, 2013.
[40] L. Niu, W. Li, and D. Xu, "Exploiting privileged information from web data for action and event recognition," *Journal of Computer Vision*, 2016.
[41] B. Ribeiro *et al.*, "Enhanced default risk models with SVM+," *Expert Systems with Applications*, 2012.