Best-Effort Adversarial Approximation of Black-Box Malware Classifiers

Abdullah Ali and Birhanu Eshete(✉)

University of Michigan, Dearborn, USA
{aliabdul,birhanu}@umich.edu

Abstract. An adversary who aims to steal a black-box model repeatedly queries it via a prediction API to learn its decision boundary. Adversarial approximation is non-trivial because of the enormous alternatives of model architectures, parameters, and features to explore. In this context, the adversary resorts to a *best-effort strategy* that yields the closest approximation. This paper explores best-effort adversarial approximation of a black-box malware classifier in the *most challenging setting*, where the adversary's knowledge is limited to label only for a given input. Beginning with a limited input set, we leverage *feature representation mapping* and *cross-domain transferability* to locally approximate a black-box malware classifier. We do so with *different feature types* for the target and the substitute model while also using *non-overlapping data* for training the target, training the substitute, and the comparison of the two. Against a Convolutional Neural Network (CNN) trained on raw byte sequences of Windows Portable Executables (PEs), our approach achieves a 92% accurate substitute (trained on pixel representations of PEs), and nearly 90% prediction agreement between the target and the substitute model. Against a 97.8% accurate gradient boosted decision tree trained on static PE features, our 91% accurate substitute agrees with the black-box on 90% of predictions, suggesting the strength of our purely black-box approximation.

Keywords: Model extraction · Model stealing · Adversarial machine learning

1 Introduction

Recent advances in machine learning (ML), specially in deep learning, have led to significant improvement on the accuracy of image classification, machine translation, speech processing, and malware/intrusion detection. Despite their impressive accuracy, deep neural networks (DNNs) and other traditional machine learning models such as Logistic Regression, Support Vector Machines, and Decision Trees have been shown to be vulnerable to training-time poisoning [8,9], test-time evasion [7,14,25,33], model extraction [24,28,34], and membership inference attacks [11,31].

N. Park et al. (Eds.): SecureComm 2020, LNICST 335, pp. 318–338, 2020.
https://doi.org/10.1007/978-3-030-63086-7_18

With the advent of ML-as-a-Service (MLaaS), ML models are increasingly served via prediction APIs to allow remote submission of input samples to produce predictions or provide pre-trained models as foundations to build up on. While MLaaS enables new frontiers of ML use-cases such as serving models from the cloud with pay-per-query price model, it also exposes models behind prediction APIs to model extraction/approximation attacks via iterative input-output interactions [19,24,28,31,34]. An adversary aiming to game a prediction API of a model to avoid a pay-per-prediction bill or a competitor targeting the trade secret of a model have financial motivations to steal ML models. For instance, adversaries are motivated to steal (approximate) a remote black-box model trained on privacy-sensitive data (e.g., medical records), intellectual property (e.g., stock market trends), or inherently security-sensitive data (e.g., malware/intrusion traces). In general, so long as the cost of approximating an ML model is lower than the potential financial gain from obtaining a close-enough copy of it, MLaaS will continue to be a target of financially motivated adversaries.

More precisely, given a black-box model f_b (e.g., a malware detector) served via a prediction API, the adversary's goal is to perform *best-effort* approximation of f_b's decision boundary by locally training a substitute model f_s. Best-effort in this sense refers to relying on limited seed-set (e.g., 5%–10% of the training set for f_b) to probe f_b and leveraging publicly accessible resources (e.g., features, pre-trained models) for effective and efficient approximation of f_b.

Previous work explored black-box model approximation by leveraging the fidelity (e.g., probability scores) of predictions [19,34], feature and/or model architecture similarity between f_b and f_s [16,25,28,34], and cross-model transferability [25,26]. The approximation formulation spans equation solving [34], optimization methods [18], generative adversarial networks [16], and reinforcement learning [24].

This paper explores adversarial approximation of a black-box malware detector f_b in the *most challenging setting for the adversary*. In particular, we explore a threat model where the adversary aims for a close-enough approximation of f_b in the face of (*i*) access to limited inputs to f_b, (*ii*) for a given input sample no additional observations beyond prediction label, (*iii*) different feature representations for f_b and f_s, and (*iv*) disjoint training sets for f_b, f_s, and the similarity evaluation set. To that end, beginning with limited seed-set for the black-box classifier, we leverage *representation mapping* and *cross-domain transferability* to approximate a black-box malware classifier by locally training a substitute.

Our work complements prior work [16,19,25,26,28,34] in three ways. First, we do not assume any adversarial knowledge other than prediction label for a given PE. This is a strong adversarial setting, which effectively leaves the adversary no leverage except a best-effort strategy that relies on publicly available vantage points (e.g., input samples, pre-trained models). Second, we approximate f_b with *different feature types* for f_b (e.g., byte sequences) and f_s (e.g., pixel intensities). By mapping the representation of PEs from byte-space to pixel-space, our approach eliminates the need for manual feature engineering. The motivation behind using dissimilar feature representations for f_b and f_s

is to leverage publicly accessible pre-trained models (e.g., Inception V3 [1]) by means of transfer learning, while only training on the last layer. While prior work [25] demonstrated cross-model transferability for image classifiers, we show a different dimension of transferability in a cross-domain setting by re-purposing a pre-trained image classifier to approximate a black-box malware detection model trained on raw-byte sequences of PEs. Third, we deliberately use *non-overlapping data* for training f_b, training f_s, and comparison of similarity between the two. We note that some prior work [19,24–26,34] use disjoint data when f_b is hosted by MLaaS providers. It is, however, hard to verify the disjointedness of f_b's training data against f_s's or the comparison set, because in such a setting f_b's training data is typically confidential.

Against a black-box CNN [27] trained on byte sequence features of Windows PEs, our approximation approach obtained up to 92% accurate CNN on pixel features, and trained based on the Inception V3 [1] pre-trained model (details in 4.3). On a comparison dataset disjoint with the black-box's and the substitute's training sets, our approach achieved nearly 90% similarity between the black-box CNN and the substitute one. In a nutshell, the results suggest that, even if the target model is a black-box, an adversary may take advantage of a limited training data and the underlying knowledge of pre-trained models (Inception V3 in this case) to successfully approximate the decision boundary of a black-box model. An intriguing observation of our results is that, although the training samples used for approximation are disjoint with training samples used to train the black-box CNN, the adversary can still achieve an acceptable approximation of the black-box CNN with minimal efforts. Another intriguing observation is, despite the dissimilarity of the representation of the black-box (i.e., byte sequences) and that of the substitute model (i.e., pixels), our approximation approach still managed to achieve nearly 90% similarity between the target black-box and the substitute model.

We further evaluate our approach on a research benchmark dataset, EMBER [6], which is based on static PE features (details in 4.4). Our approach approximated the LightGBM [20] black-box model supplied with EMBER via a 91% accurate substitute model, which agrees with the black-box on 90% of test PEs. With EMBER, we also explore how our approximation strategy performs on different models by training multiple substitute models, confirming the validity of our approach across model architectures such as Decision Trees, Random Forests, and K-Nearest Neighbours. In summary, this paper makes the following contributions:

- By mapping byte sequence features to pixel representations, we eliminate the need for heuristics-based feature engineering, which significantly reduces adversarial effort towards feature guessing.
- Leveraging a pre-trained model as a foundation, we extend the scope of transferability from a cross-model setting by demonstrating the utility of cross-domain transferability for black-box approximation.

– We demonstrate the feasibility of close-enough adversarial approximation using different feature representations for the black-box and the substitute model, across multiple model architectures, and complementary datasets.

2 Background and Threat Model

2.1 Model Approximation Attacks

In this work, we focus on supervised learning for malware classification models, where input samples are Windows PEs and the output is the class label, i.e., `benign` or `malware`.

Let X be a d-dimensional feature space and Y be the c-dimensional output space, with underlying probability distribution $Pr(X, Y)$, where X and Y are random variables for the feature vectors and the classes of data, respectively. The objective of training an ML model is to learn a parameter vector θ, which represents a mapping $f_\theta : X \rightarrow Y$. f_θ outputs a c-dimensional vector with each dimension representing the probability of input belonging to the corresponding class. $l(f_\theta(x), y)$ is a loss of f_θ on (x, y), and it measures how "mistaken" the prediction $f_\theta(x)$ is with respect to the true label y. Given a set of training samples $D_{train} \subset (X, Y)$, the objective of an ML model, f_θ, is to minimize the expected loss over all $(x, y) : L_{D_{train}}(f_\theta) = \sum_{(x,y)\in D_{train}} l(f_\theta(x), y)$. In ML models such as DNNs, the loss minimization problem is typically solved using Stochastic Gradient Descent (SGD) by iteratively updating the weights θ as: $\theta \leftarrow \theta - \epsilon \cdot \Delta_\theta(\sum_{(x,y)\in D_{train}} l(f_\theta(x), y))$, where Δ_θ is the gradient of the loss with respect to the weights θ; $D_{train} \subset (X, Y)$ is a randomly selected set (*mini-batch*) of training examples drawn from X; and ϵ is the *learning rate* which controls the magnitude of change on θ.

Figure 1 depicts a typical setup for model approximation attacks against a MLaaS platform that serves a black-box model f_b via a prediction API. The goal of the adversarial client is to learn a close-enough approximation of f_b using as few queries (x'_i's) as possible.

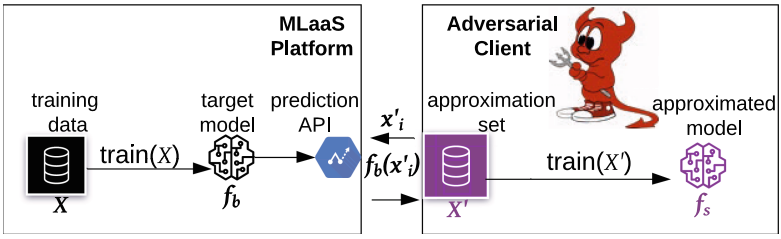

Fig. 1. Typical pipeline for model approximation attacks. MLaaS provider trains a model f_b on confidential/proprietary data X. It then serves f_b via a prediction API to allow clients to issue queries and obtain prediction results. An adversarial client uses f_b as an oracle to label $|X'|$ samples to train f_s such that $f_s \approx f_b$.

Typically, model approximation is used to steal a deployed model (e.g., intellectual property, security-sensitive business logic) [18,28,34], trick/bypass pay-per-prediction walls, perform reconnaissance for later attacks that go after the integrity of the victim model through adversarial examples [25], or mount membership inference attacks for models known to be trained on privacy-sensitive data [11,31].

The success of approximation depends on the adversarial client's knowledge (about details of f_b and its training set) and capabilities (e.g., access to inputs, maximum number of queries). In a *white-box* (full knowledge) setting, the adversary has complete knowledge of the target model architecture and its training data, making it the easiest for the adversary (but the most exposing for the target model). In a *grey-box* (partial knowledge) setting, the adversary has some knowledge about f_b's model architecture (e.g., DNN vs. SVM) and training data (e.g., a subset of the training set), which makes it moderately challenging. In a *black-box* (no/limited knowledge) setting, the adversary knows nothing but prediction labels (and possibly confidence probabilities) for a given input. This is the most challenging setting for the adversary. In this paper, we consider the black-box setting. Next, we describe our threat model.

2.2 Threat Model and Problem Statement

We consider the strongest threat model compared with previous work on adversarial approximation [25,28,34]. The adversary interacts with a deployed model f_b, for instance served from MLaaS malware detector. Without loss of generality, we assume that the ML model f_b is a malware detector.

Adversary's Goals: The adversary's goal is to approximate the decision boundary of f_b by training its substitute f_s, beginning with a limited seed-set of PEs.

Adversary's Knowledge: The adversary only knows that f_b accepts Windows PEs as input and returns labels ("benign" or "malicious") as output. The adversary doesn't know f_b's architecture, parameters, hyper-parameters, or features. Besides, the adversary has no access to the training data or the test data used to train and evaluate f_b.

Adversary's Capabilities: The adversary probes f_b with PEs to obtain prediction labels. We assume there is an upper bound on the number of queries the adversary can issue to f_b, but the adversary can workaround the query limit by probing f_b over an extended time-window or from different IP addresses. The adversary has access to a limited seed-set of PEs, but is unable to tell whether or not the PEs in the seed-set overlap with f_b's training set. The adversary is able to continuously collect more PEs to use f_b as an oracle and progressively train f_s with the goal of obtaining a close-enough approximation of f_b.

Problem Statement: Given a deployed malware detection model, f_b, under the threat model stated earlier, the adversary's goal is to find f_b's closest approximation, f_s, with best effort. By "best effort" we mean relying on limited seed-set to probe f_b, and leveraging publicly accessible resources (e.g., feature representations, pre-trained models) towards effective and efficient approximation of f_b.

More formally, given a black-box model f_b trained on dataset X, for a seed-set $|X'| < |X|$ and $X' \cap X = \emptyset$, the adversary's goal is to train f_s using X' such that when compared on dataset X'' disjoint with X and X', $f_b \approx f_s$. The $\approx$ quantifies the percentage of agreement (i.e., number of matching predictions between f_b and f_s on X''). The closer the agreement is to a 100% the more accurate the approximation and vice-versa. The real-life implication of successful approximation of f_b via f_s is that once the adversary obtains a close-enough (e.g., >90%) substitute of f_b, the intellectual property of f_b's owner (e.g., an IDS vendor) is jeopardized. Even worse, the adversary might as well emerge as f_b's competitor by tuning f_s's accuracy with additional training data.

3 Approach

Overview: Figure 2 depicts an overview of our approach. Given a black-box malware detector f_b, the adversary collects benign and malware PEs from public sources to obtain the substitute training set (X'). The adversary then uses f_b as an oracle to label samples in X' (Sect. 3.1). Next, samples in X' are mapped from a source representation: raw-bytes to a target representation: pixel intensities (Sect. 3.2). The progressive approximation step uses the mapped X' to iteratively approximate f_b (Sect. 3.3). It combines *representation mapping* and *cross-domain transferability* to obtain a close-enough approximation of f_b with a limited seed-set to f_b (i.e., $|X'| < |X|$), different feature representations for f_b and f_s, and disjoint training sets for f_b, f_s, and the comparison set used to evaluate similarity between f_b and f_s (i.e., $X \cap X' \cap X'' = \emptyset$). The approximation begins by training an initial substitute model f_s on a fraction (e.g., 20%–25%) of X', and f_s is refined until it achieves acceptable accuracy. Using a dataset X'', the last stage of the approach compares similarity between f_b and f_s (Sect. 3.4). A higher similarity score for f_b and f_s is an indication of the effectiveness of the approach at approximating the decision boundary of f_b.

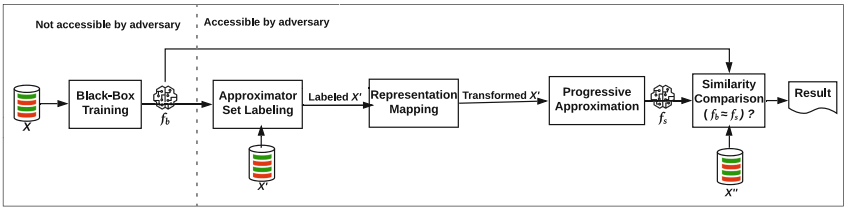

Fig. 2. Approach overview. f_b is accessible by the adversary via a prediction API only.

For the sake of presenting an end-to-end approximation framework, Fig. 2 includes f_b's training at the beginning. In practice, the adversary doesn't have access to the black-box model except through its prediction API. It is also noteworthy that the three datasets (X, X', and X'') shown in Fig. 2 are disjoint.

Again, in reality, with no access to f_b's training set, the adversary has no trivial way to determine if the substitute training set (X') or the model similarity comparison set (X'') have intersection among themselves or with the black-box training set (X). The only motivation of training our own black-box model f_b is to ensure X, X', and X'' are disjoint, and doing so lets us to deterministically examine the most challenging adversarial setting described earlier.

3.1 Approximation Set Labeling

Given a seed-set of Windows PEs collected by the adversary, a first-cut labeling strategy would be to take the ground truth labels that the samples come with (e.g., using VirusTotal [4] as an oracle). Had our goal been to train a malware detector, such an approach would suffice. Our goal, however, is to approximate the decision boundary of a black-box malware detector, and we expect our labeling method to serve this purpose.

Given a set of approximation samples $X' = x'_1, ..., x'_m$ and a learned hypothesis function f_b, we obtain $f_b(x'_i) = y'_i$. The y'_i's may or may not match the ground truth labels. If f_b misclassifies some x'_i's, the misclassified x'_i's will not match the ground truth counter-parts. What should be done with the misclassified samples in the substitute training set? The alternatives we have are (a) drop the misclassified x'_i's and explore approximation with the correctly labeled x'_i's, (b) reinstate labels to ground truth labels and proceed with approximation, or (c) take the labels assigned by f_b for what they are. Alternative (a) is no different from training the substitute without querying f_b. Alternative (b) entails "correcting" the "imperfections" of f_b (note "correcting" could as well mean lower accuracy because we are essentially changing the underlying distribution of f_b's training set). Alternative (c) is the most realistic for our threat model, because it takes f_b for what it is and uses its predictions (y'_i's) to populate the labeled approximation set, which is highly likely to result in a realistic approximation of f_b's decision boundary.

3.2 Representation Mapping

Under our threat model, the adversary doesn't know what features are used to train f_b. The adversary may then pursue different possibilities of features used to train malware detectors based on Windows PEs. However, the space of possible features is exponentially large. For example, if we only consider static analysis based features of a given PE, we end up with numerous possible features such as meta-data, DLL imports/exports, byte sequences, and so on. Similarly, considering the dynamic analysis-based features of PEs results in several candidates such as API/system call traces, instruction sequences, and call graphs. Therefore, given enough resources, while such a strategy of feature guessing may result in a close-enough approximation, it may not be the preferred avenue by the adversary whose goal is to effectively and efficiently approximate f_b.

In-line with the adversary's goals, we map the raw bytes representation of each PE to an image representation (pixel values), analogous to taking a photograph of the PE's raw byte sequences. The main rationale is, instead of searching for the best combination of a set of features to train the substitute, it is plausible to capture the whole PE's bytes via image representations such that the learning algorithm is able to "discover" distinguishing sub-structures from the image representation of PEs. We note that previous work ([15,21,23]) has also explored bytes-to-pixels conversion for malware detection, although not in the exact context of model approximation via cross-domain transferability that we explore in this work. Another equally important rationale for bytes-to-pixels mapping is the accessibility (to the adversary) of acceptably accurate pre-trained image classification models such as Inception V3 [1] which, by way of transfer learning [25], feed knowledge to the substitute model.

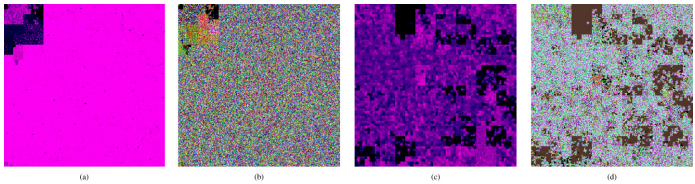

Fig. 3. (a) and (b) show EN and CH rendering, respectively, of a benign PE (small02Micro_Card_Reader_Driver_3.11.exe). (c) and (d) show EN and CH rendering, respectively, of a malware PE (Trojan.GenericKDZ.58985).

To realize the bytes-to-pixels mapping, the key intuition is to transform PEs into a colored canvas, such that the colors (pixel intensities) represent the bytes in the PEs, and the color intensities are used as features to train the substitute. To this end, we leverage two types of image representations, the Entropy (EN) representation [30] and the Color Hilbert (CH) representation [13]. CH scans the bytes of a PE and assigns color based on the value of each byte. The assigned pixel intensities are then mapped on a canvas of a chosen dimension. EN uses Shannon Entropy [30] to compute the randomness of bytes in a specific location of the executable as $E = -\sum_{i=1}^{n} \rho_i \log_2 \rho_i$, where ρ_i refers to the probability of appearances of a byte value i and n is the number of possible values ($n = 256$ for possible byte values). Depending on the computed value of E, the corresponding pixel is assigned color intensity in the range black (minimum/zero entropy) to bright pink (maximum entropy). An example that illustrates EN and CH representation for a benign PE is shown in Fig. 3 (a) and (b), respectively. Similarly, Fig. 3 (c) and (d) show EN and CH representations of a malware PE, respectively. Notice the clear visual difference between benign and malware PEs, which seems to support our intuition of mapping bytes to pixels. Looking at the CH representation, the combination of these colors in the images serves best to give the model discriminating features to put apart benign and malware PEs. With regards to the EN representation, we can see that the focus is more on

areas instead of specific pixels, and it is not as high-fidelity as the CH representation. In Sect. 4, we evaluate the utility of EN and CH representations via the substitute's accuracy. Next, we briefly describe how the canvas is filled with colors.

To paint the canvas of the image, we leverage a well-known mapping technique called the Hilbert curve [10] (implemented in BinVis [13]), which makes sure that if two bytes are close to each other in the PE, they should be close to each other in the image representation of the PE as well. This property of the Hilbert curve is essential to preserve the semantic structure of PEs when mapping bytes to pixels, and it provides the substitute model an accurate representation of the PE so that, during training, it explores the classification utility of all possible features. A natural question would be, how does the Hilbert curve function? Intuitively, the idea of Hilbert curve is to find a line to fill a canvas that will keep the points which are close to each other on that line at the same distance when it fills the needed space. This, in our case, keeps the features of the PEs intact since separating them would lead to breaking the semantics of the sequence of bytes that represents a part of our feature set in the images we would like to generate at the end of the mapping.

Note that although representation mapping is core to our approach, sometimes, feature guessing may fit the best-effort strategy when the adversary has access to publicly released datasets such as EMBER [6], which reveal details about features. To account for this possibility, in Sects. 4.3 and 4.4, we evaluate our approach with minimal relaxation on our threat model, i.e., the adversary knows features because of public disclosure as in [6]. As we will show, the purely black-box approximation is as strong as the minimally relaxed approximation.

3.3 Progressive Approximation

On the one hand, the adversary has access to limited input samples to the black-box model. On the other hand, the adversary has access to pre-trained and publicly accessible image classification models (e.g., Inception V3 [1]). To obtain an acceptably accurate approximation of the black-box model, the adversary takes advantage of the advances in image classification models to quickly and accurately train a candidate substitute on the mapped features of the PEs via *cross-domain transferability*. The motivation behind leveraging pre-trained models is threefold. First, the fact that our substitute model relies on image classification for which state-of-the-art benchmark datasets are readily accessible. Second, the prospect of using a widely available and acceptably accurate pre-trained model for anyone (including the adversary) is in favor of the constraints the adversary has on collecting enough training samples for the approximation and come up with an effective model architecture tuned to adversarial goals. Third, when using pre-trained models, we are only retraining the last layer, which not only saves us (the adversary) time, but also gives us confidence on the accuracy of the final model because of transferability. Moreover, taking advantage of image representations cuts the effort on feature engineering down to zero because the

candidate substitute (e.g., a CNN) would automatically learn the features from its mapped training data, i.e., image representations of PEs.

Algorithm 1: Progressive approximation of f_b.

τ_{acc}: accuracy threshold;
τ_{sim}: similarity threshold;
num_batches: number of approximation set batches;
for $i = 1 \rightarrow num\_batches$ **do**
  $acc_i, f_s \leftarrow TrainSubstitute(f_s, batch_i)$;
  $sim_i \leftarrow GetSimilarity(f_b, f_s)$;
  if $acc_i > \tau_{acc} \&\& sim_i > \tau_{sim}$ **then**
    $StopApproximation()$;
  end
end

Fig. 4. Progressive approximation of f_b.

In order to emulate an adversary who begins with a limited seed-set to bootstrap approximation, we assume that the adversary has $batch_1$ as the first batch (e.g., 20%) of substitute training samples X'. The adversary first trains f_s on $batch_1$ and evaluates its accuracy against a pre-determined threshold, which could be set close-enough to f_b's accuracy for our case. In real-life, one can safely assume that the adversary would estimate the accuracy threshold from public knowledge (e.g., state of the art model accuracy for malware classifiers). Similarly, the adversary can set a similarity threshold based on prediction matches between f_b and f_s over dataset X''. The adversary would then actively collect training data, and progressively re-train and re-evaluate f_s each time they obtain the next batch of approximation examples, until an acceptable accuracy and similarity score is obtained, or the training data is exhausted. Figure 4 shows the details of the progressive approximation.

3.4 Similarity Comparison

Once the substitute model f_s is trained with an acceptable accuracy, its effectiveness is assessed when compared with the black-box model on a separate dataset, disjoint with both the training set of the black-box and the substitute. Figure 5 shows the procedure for similarity comparison of f_b and f_s.

The similarity score is the percentage of matching predictions between f_b and f_s. The higher the similarity score, the closer f_s is to f_b, which means f_s effectively mirrors the decision boundary of f_b. The adversary then probes f_s for further attacks in a white-box setting. Attacks that succeed on f_s, would, by transitivity, succeed on f_b. This is the essence of having an accurate approximation which would be used as a substitute for the black-box. By crafting adversarial examples using methods such as FGSM [33] and CW [12], the adversary can transitively target f_b using f_s as a surrogate white-box model. Comparison of our candidate substitutes with the black-box model is discussed in Sect. 4.4.

Algorithm 2: Similarity comparison between f_b and f_s.

N: number of samples for comparison;
$matches \leftarrow 0$;
for $i = 1 \rightarrow N$ **do**
  $y_s^i \leftarrow f_s(x^i)$;
  $y_b^i \leftarrow f_b(x^i)$;
  if $y_b^i == y_s^i$ **then**
    $matches \leftarrow matches + 1$;
  end
end
$similarity\_score \leftarrow \frac{matches}{N} \times 100$;

Fig. 5. Similarity comparison between f_b and f_s.

4 Evaluation

We now describe our datasets, experimental setup, and results on progressive approximation and similarity comparison.

4.1 Datasets

Table 1 summarizes our datasets: Custom-MalConv and EMBER[6].

Custom-MalConv: We collected benign PEs from a Windows freeware site [2] and malware PEs from VirusShare [3]. These two sources are used by prior work [5,6,16,27,29] on adversarial examples and malware detection. Overall, we collect 67.81K PEs with 44% benign and 56% malware. We use 60% for training the black-box CNN, 23% as substitute training set, and 17% as similarity comparison set. Since we control the dataset, we use this dataset to evaluate the effectiveness of our approach on representation mapping and cross-domain transferability.

EMBER: EMBER [6] is a public dataset of malware and benign PEs released together with a Light gradient boosted decision tree model (LGBM) [20] with 97.3% detection accuracy. The dataset consists of 2351 features extracted from 1M PEs via static binary analysis. The training set contains 800K labeled samples with 50% split between benign and malicious PEs, while the test set consists of 200K samples, again with the same label split. The authors of EMBER used VirusTotal [4] as a labeling oracle to label samples in EMBER. We use this dataset to further evaluate the generalizability of our approach. We note, however, that EMBER releases only static analysis based features extracted from PEs, not the PEs themselves, which limits our scope of doing representation mapping. Despite the absence of original PEs, we still evaluate our approach on a different feature space and over multiple candidate substitute models. Of the 200K test set samples, we use 150K as substitution training set, and the remaining 50K (25%) as similarity comparison set.

Table 1. Datasets for Custom-MalConv and EMBER.

Dataset	Benign	Malware	Total
Custom-MalConv			
Black-Box training set	20,000	20,000	40,000
Substitute training set	8,000	8,000	16,000
Similarity comparison set	2,045	9,765	11,810
Total			**67, 810**
EMBER [6]			
Black-Box training set	400,000	400,000	800,000
Substitute training set	75,000	75,000	150,000
Similarity comparison set	25,000	25,000	50,000
Total			**1, 000, 000**

4.2 Experimental Setup

Black-Box Models: The Custom-MalConv black-box is trained on 20K benign and 20K malware PEs. It is based on MalConv [27], a widely used malware detection CNN in adversarial ML literature on malware [22,32]. We use the same architecture as the original MalConv [27] with slight modifications to fit our hardware (NVIDIA 1080 with 8 GB of RAM). MalConv is based on raw bytes of a PE and reads the first 1 MB to extract byte features. To fit our hardware limitations, we fed only $\frac{1}{3}$ MB of each PE to our custom CNN, and got an accuracy of 93% which is acceptable since MalConv [27] used 100K PEs to achieve 98% accuracy. The LightGBM black-box is trained on 400K benign and 400K malware PEs and is shipped with the EMBER [6] dataset with 97.3% accuracy.

Substitute Models: For Custom-MalConv black-box model, our f_s is based on Inception V3 (IV3) [1] and is trained on both CH and EN representations of the substitute training set. In addition, for sanity check, we trained a substitute Custom-MalConv model with exact same features and model parameters as our black-box. For EMBER, we explore 4 candidates for substitute model, namely: decision trees (DT), random forests (RF), k-Nearest neighbors (kNN), and gradient-boosted decision tree model (LGBM).

Progressive Approximation: For Custom-MalConv, we use 16K substitute training set and train f_s progressively on 4K, 8K, 12K, and 16K PEs. For LGBM, we use 150K of the 200K test set with a progression of 30K, 60K, 90K, 120K, and 150K PEs. The metrics we use are *validation accuracy* and *similarity score*. Validation accuracy is the prediction accuracy of the approximated substitute. Similarity score (as shown in Fig. 5) is the percentage of matching predictions between f_b and f_s.

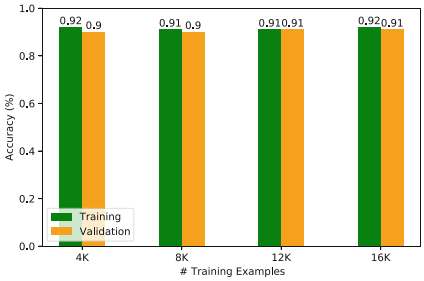

Fig. 6. Custom-MalConv: IV3 progressive accuracy with CH.

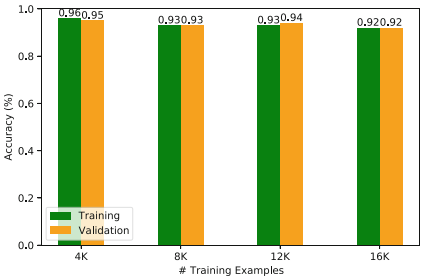

Fig. 7. Custom-MalConv: IV3 progressive accuracy with EN.

4.3 Progressive Approximation Results

Custom-MalConv: Figure 6 shows the training accuracy and validation accuracy of each progressive step in training f_s: IV3 with CH representation. It can be seen that the difference between training and validation accuracy narrows as the model progresses with more training data, which is an indication of the real-life effectiveness of the model on training examples it has never seen during training. Similarly, Fig. 7 shows the progressive training of f_s: IV3 with EN representation. Here, the trend persists, where we see a very high training accuracy across the board, but as we train on bigger data sets we get a more accurate model (validation accuracy is close to training accuracy).

For both CH and EN representations, it *takes only 10% the size of* f_b*'s training set to approximate it with* f_s that achieves 0.9 and 0.95 validation accuracy, respectively. On IV3 with CH, quadrupling the substitute training set (4K to 16K) improved its validation accuracy by 0.1 only, showing the *feasibility of approximation under data scarcity by taking advantage of transferability.* IV3 with EN shows a slightly different progress. On 16K approximation samples, it achieves 0.92 validation accuracy, which is 3% less than the IV3 with CH trained on 4K samples. Looking at the difference between the training and validation accuracies (green vs. orange bars in Figs. 6 and 7), we notice that the narrower the gap between training and validation accuracy, the more stable a model would be classifying unknown samples.

EMBER: Figure 13 shows the progressive validation accuracy of the four candidate substitute models against the LGBM black-box. Compared to the progress of Custom-MalConv substitutes (Figs. 6 and 7), the validation accuracies of LGBM, DT, and RF are relatively lower (in the range 0.85–0.88). However, kNN stands out as a clear contender (validation accuracy = 0.91) when compared with Custom-MalConv substitutes. Again, interestingly, the LGBM substitute, despite its architectural similarity to the target black-box LGBM, is not the best in terms of accuracy. From Figs. 8, 9, 10 and 11, it is noteworthy that the gap between training and validation accuracy for EMBER substitutes is comparatively bigger (on average 4%) compared to the narrow differences (on average 0.5%) for Custom-MalConv substitutes shown in Figs. 6 and 7.

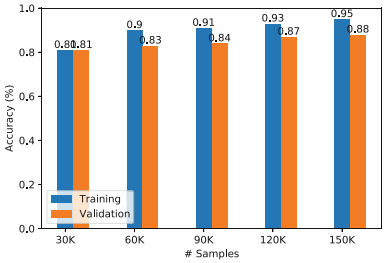

Fig. 8. Progressive accuracy of f_s: LGBM.

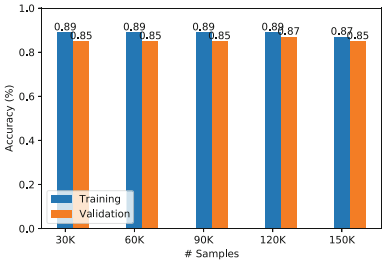

Fig. 9. Progressive accuracy of f_s: DT.

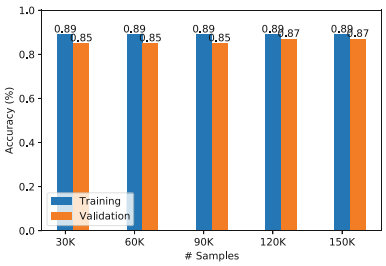

Fig. 10. Progressive accuracy of f_s: RF.

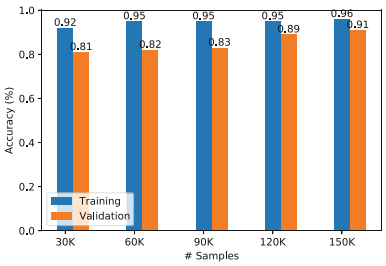

Fig. 11. Progressive accuracy of f_s: kNN.

Looking at the substitute training set size with respect to the LGBM black-box, *to obtain the best performing substitute (kNN), it takes only 18.8% (150K) of LGBM's training set size (800K).* Interestingly, from Fig. 8, we notice that the LGBM substitute, despite its architectural similarity with the target LGBM, performs relatively poorly, suggesting that *model architecture similarity may not always result in the best approximation.*

4.4 Similarity Comparison Results

Custom-MalConv: As shown in Table 2, we have 3 substitutes, a Custom-MalConv substitute, InceptionV3 with CH, and InceptionV3 with EN. The comparison of these substitutes with the black-box is done on a separate comparison set, disjoint with the b lack-box training set and the substitute training set.

On average, our approach achieved 83.7% similarity score, with the highest similarity score of 89%, on IV3 substitute trained on EN representation. The MalConv substitute that matches the architecture of the black-box Custom-MalConv model is the least accurate, which interestingly indicates that *model architecture similarity may not always result in a substitute that agrees well with a black-box target.* When we compare CH-based and EN-based substitutes, EN-based substitute outperforms the CH-based substitute by about 6.5%, which could be attributed to the canvas coloring schemes of CH and EN.

Table 2. Custom-MalConv: similarity comparison between f_b and f_s.

Substitute (f_s)	Validation accuracy	Similarity (f_b, f_s)
InceptionV3-ColorHilbert	0.91	82.19%
InceptionV3-Entropy	0.92	88.65%
Custom-MalConv-ByteSequence	0.90	80.11%

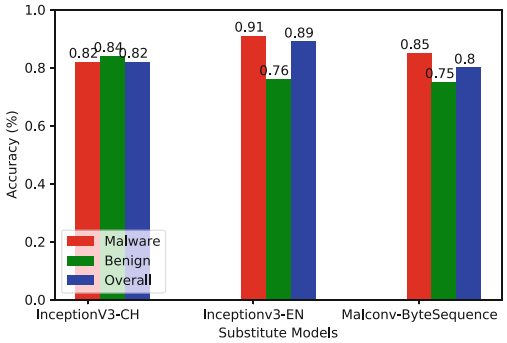

Fig. 12. Custom-MalConv: Benign vs. malware agreement split for f_b and f_s.

Figure 12 shows the details of similarity scores for the three substitute models split into malware and benign. To compute the split, for each label, we divide the number of agreements between f_b and f_s by the total number of samples for that label. Having a malware detection rate higher than the benign detection rate serves us best, since our goal is to be able to identify malware identified as so by the black-box. It can be seen that the EN-based substitute tends to agree with the black-box more on malware, while the CH-based substitute agrees with the black-box more on the benign predictions. Again, this variation is rooted in the canvas coloring methods of the two representations discussed earlier.

EMBER: Table 3 summarizes the similarity comparison between LGBM black-box and candidate substitute models. On average a substitute model agrees with the black-box on 90.3% of the comparison samples. kNN, as the closest approximation with a different model architecture from the black-box, agrees with it on almost 90% of the comparison samples. This result is comparable to the best substitute in Custom-MalConv: InceptionV3-Entropy with 89% similarity score. We also notice from Table 3 that, although it is the least accurate model from Fig. 13, LGBM substitute agrees with the LGBM black-box on over 97% of the comparison instances. This is not surprising, given the exactness of the model architectures. It is, however, in contrast to our observation on the similarity of Custom-MalConv-ByteSequence substitute with the same black-box model architecture, which scores just over 80% similarity (see Table 2).

Figure 14 shows the evolution of similarity scores showing how kNN substitute's similarity improves almost by 10% in the range 90K–150K of approximation instances. RF substitute also shows a similar pattern, but with a smaller

Table 3. EMBER: similarity comparison between f_b and f_s.

Substitute (f_s)	Validation accuracy	Similarity (f_b, f_s)
Decision tree	0.85	86.3%
Random forest	0.87	87.7%
k-Nearest neighbors	0.91	89.9%
LightGBM	0.88	97.1%

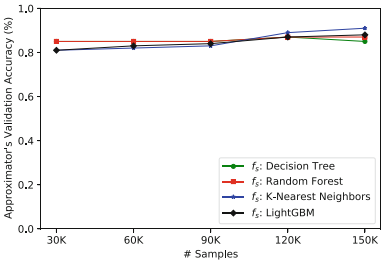

Fig. 13. EMBER: progressive validation accuracy of candidates for f_s.

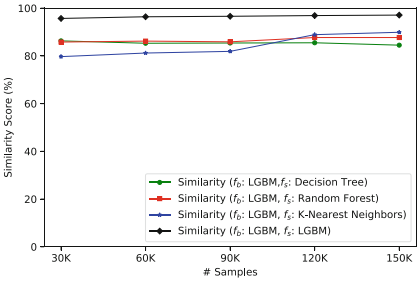

Fig. 14. EMBER: progressive similarity of f_b with f_s.

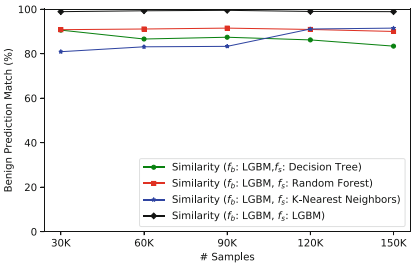

Fig. 15. EMBER: Benign prediction match between f_b and f_s.

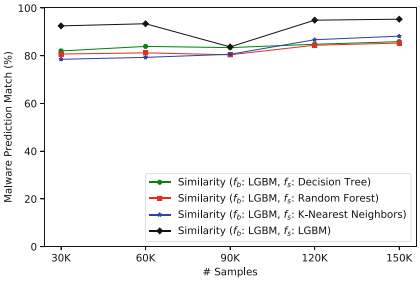

Fig. 16. EMBER: Malware prediction match between f_b and f_s.

improvement of 4% (83% to 87%). Note that the LGBM substitute remained stable across the progressive increment of the substitute training set.

Finally, Figs. 15 and 16 show the progression of benign and malware similarity matches as we train the substitutes. It can be seen that, overall, there is high degree of agreement between the LGBM black-box and the substitutes on benign predictions. On benign predictions, the LGBM substitute shows a stable and close to 100% agreement with its black-box counterpart. On malware predictions, it shows variations in the range 83%–95%.

5 Related Work

Tramer et al. [34] show equation-solving and optimization based attacks that approximate target ML models with near-perfect replication for popular learning algorithms including logistic regression, support vector machines, neural networks, and decision trees. They demonstrate their attacks against production MLaaS providers such as BigML and Amazon ML. They note that their most successful model approximations are for MLaaS services that return prediction labels with probability scores.

Papernot et al. [26] demonstrate an adversary controlling a remotely hosted DNN with no knowledge about the DNN and its training data. They train a local substitute against a target DNN using synthetically generated inputs labeled by the target DNN. They then use the local substitute to craft adversarial examples to evade DNNs hosted by MetaMind, and also train logistic regression substitutes against Amazon and Google prediction APIs.

Papernot et al. [25] train substitute models to approximate a target black-box model, and craft adversarial examples to evade it. They use the *Jacobian-based dataset augmentation* method to synthetically generate examples for training the substitute. Similar to [26], they demonstrate transferability within and between different classes of ML models such as DNNs, logistic regression, support vector machines, decision trees, nearest neighbors, and ensembles.

Hu and Tan [16] train a substitute model using the GAN framework to fit a black-box malware detector. In a follow-up work [17], they use the GAN-based substitute model training for a recurrent neural network. Both approaches ([16, 17]) assume the adversary knows the type of features and the architecture of the black-box model. In our case, we assume that the adversary knows nothing about the details of the black-box. In addition, our approach differs from [16] in the assumption about the underlying feature representations of training examples.

While [16] assumes the same (API calls precisely) feature representation of the black-box and the substitute, in our work, the black-box and the substitute have different feature representations. Rosenberg et al. [29] adopt the Jacobian-based augmentation [25] to synthesize training examples to approximate and evade a target black-box malware detector based on API call features. Like Papernot et al. [25] and Hu and Tan [16], Papernot et al. in a different work [26] craft adversarial examples to evade a black-box model.

Orekondy et al. [24] follow a similar threat model to ours where publicly available images from a domain different from the black-box model are used to train a substitute 'Knockoff' model. They show selecting samples from a totally different distribution using reinforcement learning is effective for an approximation attack, while using different model architectures for the victim black-box and the substitute knockoff. We note that, in the cross-domain transferability setting, this work is the closest to ours except that it uses the same pixel features for both f_b and f_s, while we use byte sequences for f_b and pixels for f_s.

Next, we make approach-level comparisons of our work and closely related prior work. Table 4 shows multi-criteria comparison of this work with the state-of-the-art. Since direct quantitative comparison is non-trivial given differences

in assumptions and dataset/task heterogeneity, our comparison is rather qualitative, based on feature representation, model architecture, f_s's training set size, training set overlap between f_s and f_b, and f_b's prediction output.

Features (f_b, f_s) compares assumptions about features of the black-box (f_b) and the substitute (f_s). While prior work [16,17,19,25,26,28,29] assumed similar features for f_b and f_s, we consider raw-bytes for f_b and pixels for f_s.

Model (f_b, f_s) captures assumptions about similarity of model architectures for f_b and f_s. Unlike most prior work [16,17,19,25,26,28] which assume the same model architecture for f_b and f_s, we assume the adversary is free to evaluate different model architectures for f_s (hence could match f_b's architecture or end up using a different one).

Table 4. Approach-level qualitative comparison with closely related work.

	[19,26]	[34]	[29]	[28]	[16,17]	[24]	[25]	This work
Features (f_b, f_s)	Same	Same	Same	Same	Same	Same	Same	Different
Model (f_b, f_s)	Same	Same/different	Different	Same	Same	Different	Different	Different
Seed-set (f_s)	Moderate	Moderate	Moderate	Limited	Moderate	Moderate	Limited	Limited
Data (f_b, f_s)	Disjoint	Disjoint	N/A	N/A	Disjoint	Disjoint	Disjoint	Disjoint
Output (f_b)	Label+conf.	Label+conf.	Label	Label	Label	Label	Label	Label

Seed-set (f_s) compares assumptions on the size (number of training examples) of the seed-set to train f_s. In [25,26,28], the adversary explores synthetic data generation techniques such as augmentation to generate more training samples to train f_s. In [16] and [17], the adversary collects enough samples to train f_s. In this work, we explore approximation with access to a limited seed-set that we extend with data augmentation techniques.

Data (f_b, f_s) examines whether or not there is overlap in training sets used for f_b, f_s, and comparison of f_b with f_s. In this work, we use disjoint datasets for training f_b, f_s, and comparison of f_b with f_s. Doing so enables assessment of the effectiveness of f_s approximated with a completely new dataset.

Output (f_b) captures whether label only or label with confidence score (conf.) is returned from f_b. While prior work used label only [16,17,24,28,29] and label with probability score [19,26,34], our black-box returns only the label of the PEs, i.e., "benign" or "malware".

6 Conclusion

We presented a best-effort adversarial approximation approach that leverages representation mapping and cross-domain transferability to obtain a close-enough approximation of a black-box malware detector. We show that an adversary can obtain nearly 90% similarity between a black-box model and its approximation beginning with a limited input-set to the black-box, different features and disjoint training sets for the black-box and its substitute. We further demonstrate that our approach generalizes to multiple model architectures across different

feature representations. This work broadens the scope of adversarial approximation of a black-box ML model in a strictly black-box setting. Our results shade light on the fact that different feature representations may not necessarily hinder close-enough approximation and disjoint datasets still result in successful approximation. More importantly, a pre-trained multi-class image classifier, such as Inception V3, can be re-purposed to approximate a binary malware classifier, demonstrating the scope of transferability beyond cross-model and extending it to a cross-domain setting.

References

1. Advanced guide to inception v3 on cloud TPU (2019). https://cloud.google.com/tpu/docs/inception-v3-advanced
2. Cnet freeware site (2019). https://download.cnet.com/s/software/windows/?licenseType=Free
3. Virus share (2019). https://virusshare.com
4. Virus total (2119). https://www.virustotal.com/gui/home/upload
5. Al-Dujaili, A., Huang, A., Hemberg, E., O'Reilly, U.: Adversarial deep learning for robust detection of binary encoded malware. In: 2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, 24 May 2018, pp. 76–82 (2018)
6. Anderson, H.S., Roth, P.: EMBER: an open dataset for training static PE malware machine learning models. CoRR abs/1804.04637 (2018)
7. Biggio, B., et al.: Evasion attacks against machine learning at test time. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, 23–27 September 2013, Proceedings, Part III, pp. 387–402 (2013)
8. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, 26 June – 1 July 2012 (2012)
9. Biggio, B., Roli, F.: Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recognit. **84**, 317–331 (2018)
10. Byrne, A., Hilbert, D.R.: Color realism and color science. Cambridge Univ. Press **26**(1), 3–64 (2003)
11. Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., Song, D.: The secret sharer: evaluating and testing unintended memorization in neural networks. In: 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, 14–16 August 2019, pp. 267–284 (2019)
12. Carlini, N., Wagner, D.A.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017, pp. 39–57 (2017)
13. Cortezi, A.: binviz (2019). https://github.com/cortesi/scurve/blob/master/binvis
14. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
15. Han, K., Lim, J.H., Kang, B., Im, E.G.: Malware analysis using visualized images and entropy graphs. Int. J. Inf. Sec. **14**(1), 1–14 (2015). https://doi.org/10.1007/s10207-014-0242-0

16. Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on GAN. CoRR abs/1702.05983 (2017)
17. Hu, W., Tan, Y.: Black-box attacks against RNN based malware detection algorithms. In: The Workshops of the the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, 2–7 February 2018, pp. 245–251 (2018)
18. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks (2020)
19. Juuti, M., Szyller, S., Marchal, S., Asokan, N.: PRADA: protecting against DNN model stealing attacks. In: IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, 17–19 June 2019, pp. 512–527 (2019)
20. Ke, G., et al.: Lightgbm: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA, pp. 3146–3154 (2017)
21. Khormali, A., Abusnaina, A., Chen, S., Nyang, D., Mohaisen, A.: COPYCAT: practical adversarial attacks on visualization-based malware detection. CoRR abs/1909.09735 (2019)
22. Kolosnjaji, B., et al.: Adversarial malware binaries: evading deep learning for malware detection in executables. In: 26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, 3–7 September 2018, pp. 533–537 (2018)
23. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S.: Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec 2011, pp. 4:1–4:7 (2011)
24. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: stealing functionality of black-box models. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019, pp. 4954–4963 (2019)
25. Papernot, N., McDaniel, P.D., Goodfellow, I.J.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. CoRR abs/1605.07277 (2016)
26. Papernot, N., McDaniel, P.D., Goodfellow, I.J., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. CoRR abs/1602.02697 (2016)
27. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K.: Malware detection by eating a whole EXE. In: The Workshops of the the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, 2–7 February 2018, pp. 268–276 (2018)
28. Reith, R.N., Schneider, T., Tkachenko, O.: Efficiently stealing your machine learning models. In: Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, WPES@CCS 2019, London, UK, 11 November 2019, pp. 198–210 (2019)
29. Rosenberg, I., Shabtai, A., Rokach, L., Elovici, Y.: Generic black-box end-to-end attack against state of the art API call based malware classifiers. In: Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, 10–12 September 2018, Proceedings, pp. 490–510 (2018)
30. Shannon, C.E.: A mathematical theory of communication. Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)
31. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22–26, 2017, pp. 3–18 (2017)

32. Suciu, O., Coull, S.E., Johns, J.: Exploring adversarial examples in malware detection. In: 2019 IEEE Security and Privacy Workshops, SP Workshops 2019, San Francisco, CA, USA, May 19–23, 2019, pp. 8–14 (2019)
33. Szegedy, C., et al.: Intriguing properties of neural networks. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014, Conference Track Proceedings (2014)
34. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 601–618 (2016)