CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples

Honggang Yu*, Kaichen Yang*, Teng Zhang†, Yun-Yun Tsai‡,
Tsung-Yi Ho‡, Yier Jin*§
*University of Florida, {honggang.yu, bojanykc}@ufl.edu, yier.jin@ece.ufl.edu
†University of Central Florida, teng.zhang@ucf.edu
‡National Tsing Hua University, s107062548@m107.nthu.edu.tw, tyho@cs.nthu.edu.tw

***Abstract*—Cloud-based Machine Learning as a Service (MLaaS) is gradually gaining acceptance as a reliable solution to various real-life scenarios. These services typically utilize Deep Neural Networks (DNNs) to perform classification and detection tasks and are accessed through Application Programming Interfaces (APIs). Unfortunately, it is possible for an adversary to steal models from cloud-based platforms, even with black-box constraints, by repeatedly querying the public prediction API with malicious inputs. In this paper, we introduce an effective and efficient black-box attack methodology that extracts large-scale DNN models from cloud-based platforms with near-perfect performance. In comparison to existing attack methods, we significantly reduce the number of queries required to steal the target model by incorporating several novel algorithms, including active learning, transfer learning, and adversarial attacks. During our experimental evaluations, we validate our proposed model for conducting theft attacks on various commercialized MLaaS platforms hosted by Microsoft, Face++, IBM, Google and Clarifai. Our results demonstrate that the proposed method can easily reveal/steal large-scale DNN models from these cloud platforms. The proposed attack method can also be used to accurately evaluates the robustness of DNN based MLaaS classifiers against theft attacks.**

I. Introduction

Deep neural networks (DNNs) have become the most common architecture in machine learning, implemented in a variety of tasks across many disciplines [1], [2], [3], [4], [5]. However, creating a successful DNN model depends on the availability of huge amounts of data as well as enormous computing power, and the model training is often an arduously slow process. This presents a large barrier to those interested in utilizing a DNN. To meet the demands of users who may not have sufficient resources, cloud-based deep learning services arose as a cost-effective and flexible solution, allowing users to complete their machine learning (ML) tasks efficiently.

Cloud-based deep learning services generally provide end users with a prediction API for a DNN model trained to achieve performance beyond what users could create for themselves. Users query the API with their inputs (*e.g.*, images, audio, etc), pay for each individual query, and then receive predictions results (*e.g.*, labels, confidence) back from the API, without having to understand the methods in creating those predictions. Take the Microsoft Custom Vision Service as an example: this service helps users create high-quality custom deep learning classifiers by applying active learning along with neural network architecture search technology, and predicts the class of objects inside images supplied by the users with high accuracy.

Typically users access the API by querying it and receive the results. However, the DNN models and training data inside the prediction API are routinely inaccessible to the public due to economic and privacy concerns. The provider of the API may spend great effort collecting data and training models, and thus wants to keep them proprietary. The training data may also contain private information related to individuals, prohibiting disclosure of this data by law.

Though the DNN model and the training data behind the prediction API are not directly exposed to the public, recent research has demonstrated that information leakage is still possible through query operations. For example, F. Tramèr *et al.* were the first to develop a model extraction attack [6] in 2016, which extracts an equivalent or near-equivalent machine learning model by simply querying and obtaining the prediction results on input feature vectors. Since then, many following works have been proposed to improve model extraction attacks [7], [8], [9]. In addition to the model itself, the data used to train the model can also be leaked through querying. R. Shokri *et al.* proposed a membership inference attack to determine whether the training set contains certain data records [10]. Membership inference attacks are further studied in [11], which concludes that membership disclosure exists widely, not only in overfitting models, but also in well-generalized models.

Some defense mechanisms have been proposed to reduce the impact of information leakage during the querying process [12], [13], but none of them ensure effectiveness and efficiency at the same time. The defense against information leakage via DNN model queries thus still remains as an open problem.

Although recent DNN query and model extraction attack have made significant progress, they remain impractical for real-world scenarios due to the following limitations: 1) Current model stealing attacks against commercialized platforms mainly target small-scale machine learning models such as linear regression, logistic regression, support vector machine

§Corresponding Author.

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.24178
www.ndss-symposium.org

(SVM), and neural networks. The effectiveness of these attacks are not fully evaluated on complex DNN models with more layers and more parameters. 2) Current model stealing attacks require the number of queries to the target model to be proportional to the number of model parameters. This may be acceptable when the model is small and the number of parameters is limited. However, queries of this size will be impractical when targeting recent popular DNN models like VGGNet, ResNet and Inception that contain millions of parameters. The existing method for stealing large-scale deep learning models has reliable performance but requires massive amounts of prediction queries and incurs high costs [14].

In this paper we introduce a novel type of model stealing attack against popular MLaaS platforms hosted by Microsoft [15], Face++ [16], IBM [17], Google [18] and Clarifai [19]. Our hypothesis is that an adversary who targets these pay-as-you-go, commercialized, MLaaS platforms has no prior knowledge about the exact training data, architecture or parameter of the victim model, but can observe the classification outputs (*i.e.*, labels or confidence scores) when providing the prediction APIs with random inputs, *i.e.*, query operation.

The key idea of our attack approach is to use input-output pairs obtained by querying such black-box APIs with malicious examples to retrain the substitute models which are generally chosen from candidate Model Zoo (see Figure 1). Specifically, by applying a margin-based, adversarial, and active learning algorithm to search these malicious examples, we improve the efficiency of queries to the victim classifiers inside these MLaaS platforms. As a result, since the resulting images lie approximately on the decision boundary of the victim classifier, an attacker can greatly reduce the labeling effort when generating the synthetic data set for retraining the substitute model. Through detailed experimental evaluation and testing, we demonstrate that it is possible to replicate the functionality of victim classifiers by utilizing the well-trained substitute model. An adversary can use our attack framework to construct a free version of the victim model which bypasses the monetary costs involved in collecting data and training models.

A qualitative comparison between our work and existing works is shown in Table I. From Table I we can see that our method can steal large-scale deep learning models with high accuracy, few queries, and low costs simultaneously, while prior works fail in at least one or two of these aspects. We also provide detailed evaluations to clarify the quantitative analysis of our work and existing works in the paper.

In summary, we mainly make the following contributions to address the limitations of the existing works:

- We propose a new adversarial attack method named *FeatureFool* against local substitute models, that adopts internal representation for generating a subset of malicious samples (*i.e.*, synthetic dataset). These samples are used to query the victim model to efficiently learn the distance between decision boundaries of the victim model and the stolen model, significantly reducing the number of queries required to extract the victim model.
- We design a black-box model theft attack targeting large-scale DNN models provided by commercial platforms. Our attack accelerates the model theft process with adversarial active learning and transfer learning from existing well-trained models such as AlexNet, VGG19, VGGFace, ResNet50, etc.
- We evaluate the attack framework on a group of popular commercial platforms hosted by Microsoft, Face++, IBM, Google and Clarifai. The experimental results show that our model theft attack can successfully construct a local substitute model with performance similar to the victim model found in commercialized MLaaS platforms with much less queries than previous attacks.

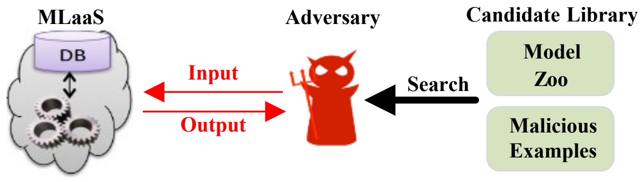

Fig. 1: Illustration of our MLaaS model stealing attacks.

Method	Parameter Size	Queries	Accuracy	Cost
F. Tramèr [6]	∼ 45K	∼ 102 K	High	Low
Juuti [20]	∼ 100M	∼ 111 K	High	-
Correia-Silva [14]	∼ 200M	∼ 66K	High	High
Papernot [21]	∼ 100M	∼ 7K	Low	-
Our Method	∼ 200M	∼ 3K	High	Low

TABLE I: A Comparison to prior works.

II. Related Work

Transfer Learning. Transfer learning aims to recognize and apply knowledge gained from previous tasks (*source domains*) to different but related tasks (*target domains*) [22]. For example, many researchers have recently shown that layers trained on a source task with large-scale labelled datasets can be reused to predict on a target domain that has substantially less available data [23], [24], [25], [26], [27]. Ge *et al.* [24] use special descriptors to search for a training subset and jointly fine-tune a pre-trained deep neural network for both source and target tasks. More similar to our work, Sun *et al.* [25] design a DeepID for learning a set of high-level feature representations and transfer joint Bayesian model from source domain to the target domain based on the DeepID. Unlike the work in [25], we fine-tune a VGG19 model [28] on a desired subset of training samples and use DeepID to further extract high-level image presentation. In this paper, we call this new model VGG_DeepID. Our transfer learning scheme additionally accelerates the model stealing process and provides performance gains by using well-trained models such as AlexNet, VGG19, VGG_DeepID, VGGFace and ResNet50.

Adversarial Attacks in Deep Learning. Adversarial attacks against DNNs generate adversarial examples by adding particular perturbations to the original inputs [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40]. In the image processing area, these carefully crafted images are often imperceptible to human eyes as the perturbations are slight, but can easily fool a classifier into predicting incorrect

TABLE II: MLaaS Services in Detail

Services	Products and Solutions	Customization	Function	Black-box	Model Types	Monetize	Confidence
Microsoft	Custom Vision	✓	Traffic Recognition	✓	Neural Nerwork	✓	✓
	Custom Vision	✓	Flower Recognition	✓	Neural Network	✓	✓
Face++	Emotion Recognition API	✗	Face Emotion Verification	✓	Neural Network	✓	✓
IBM	Watson Visual Recognition	✓	Face Recognition	✓	Neural Network	✓	✓
Google	AutoML Vision	✓	Flower Recognition	✓	Neural Network	✓	✓
Clarifai	Safe for Work (NSFW) API	✗	Offensive Content Moderation	✓	Neural Network	✓	✓

labels [41], [42], [43]. Szegedy *et al.* [34] propose the first algorithm to generate adversarial examples - L-BFGS to search for malicious examples that would be correctly classified by a human but successfully evade DNN model classifiers. Since then, more efficient algorithms [44], [35], [36] are proposed to trick DNN models into misclassifying inputs. Note that these described algorithms are all white-box attacks as they require internal information from the target model.

Prior works on black-box attacks rely on querying victim models and using the feedback from adversarial examples to guide the synthetic dataset crafting process [31]. In comparison, some others assume that the existing trained models (*i.e.*, substitute/local models) have boundaries similar to the victim models, and show that the adversarial examples generated for substitute models can transfer well to the non-targeted/targeted labels [31], [32], [33]. Several studies [45], [46] launch adversarial attacks on deep neural networks by manipulating their internal features to achieve better performance. We adopt a similar method of generating adversarial examples using the internal features of DNN models. As opposed to existing feature-level adversarial attacks, we mainly concentrate on the following two aspects: 1) the use of generated malicious features to craft visually imperceptible adversarial images against current state-of-the-art deep neural networks models. Note that existing feature-level attack methods only use the feature representation of guide images to generate adversarial examples rather than generating malicious feature representation calculated using salience maps; 2) solving for model parameters that minimize confidence scores for the target class.

Model Extraction Attacks. In these attacks [6], [47], a malicious entity aims to accurately extract model equivalent to a target model by querying the labels and confidence scores of model predictions to inputs. Papernot *et al.* [21] demonstrate that an attacker can use synthetic datasets to train a local substitute model for the victim models. Moreover, several studies [14], [20] present efficient algorithms to steal machine learning models. Unlike these prior works, this paper proposes a more efficient black-box attack method to steal deep learning models with millions of parameters by applying a special type of transfer learning scheme and specially crafted adversarial examples.

Active Learning. Generally, Active Learning (AL) is applied through iteratively selecting informative examples to present to users for labeling, while maximizing the performance of retrained deep learning classifiers [48], [49], [50], [51]. Previous uncertainty sampling methods tend to suffer from the problem of selected examples that lie approximately on the classification boundary being overly similar, resulting in poor classification performance, while users consider such examples as an ideal training set. In this paper, we address this challenge by leveraging a set of adversarial examples generation algorithms for increasing the diversity of useful examples lying on the classification boundary, improving the efficiency of query to the victim classifiers. As a result, with black-box access, an adversary can successfully replicate the functionality of the victim classifier by using a local substitute classifier with fewer queries compared to previous works on model extraction attacks.

MLaaS Platforms. Machine learning as a service (MLaaS) is a group of cloud computing services that provide end users machine learning products and solutions to data transformations, model training and ultimately, predictive analytics. We show the details of five popular (MLaaS) platforms in Table II, including the Microsoft Custom Vision, the Face++ Emotion Recognition API, the IBM Watson Visual Recognition, the Google AutoML Vision, and the Clarifai Not Safe for Work (NSFW) API. As shown in Table II, we can see that these services generally allow users to upload their well-labeled images to customize models by using the built-in algorithms or directly adopting the pre-trained models to create workflow specifically to meet their needs. Finally, MLaaS will provide APIs for users to leverage powerful tools built on top of powerful cloud computing resources. For these services, users can access the API provided by MLaaS and obtain corresponding classification results with chosen inputs. In general, users are incapable of accessing the details of the target model or the parameters used for optimization (*i.e.*, black-box), which makes it extremely difficult for an adversary to extract a black-box model. Based on the classification methods provided by those services, we can categorize them into two types: Non-neural-net based model and Neural-net based model. For the non-neural-net based models, small scale machine learning models (*e.g.* logistic regression, decision tree, and random forest) are widely used for general classification tasks. For the neural-net based model, MLaaS deep neural networks are used as the basic architecture for image classification or object detection tasks. Some of them also use transfer learning methods, allowing users to train high-quality customized models using a small labeled dataset. The providers monetize their services by charging users for training models or querying existing models through their APIs.

III. Background

A. Problem Formulation

Given a black-box victim model f_v that accepts the input $x \in \mathbb{R}^n$ and produces output $f_v(x) = [f_v^1(x), f_v^2(x), ..., f_v^m(x)] \in \mathbb{R}^m$, an adversary aims to use as few queries N_{query} as possible to extract a substitute model f_s

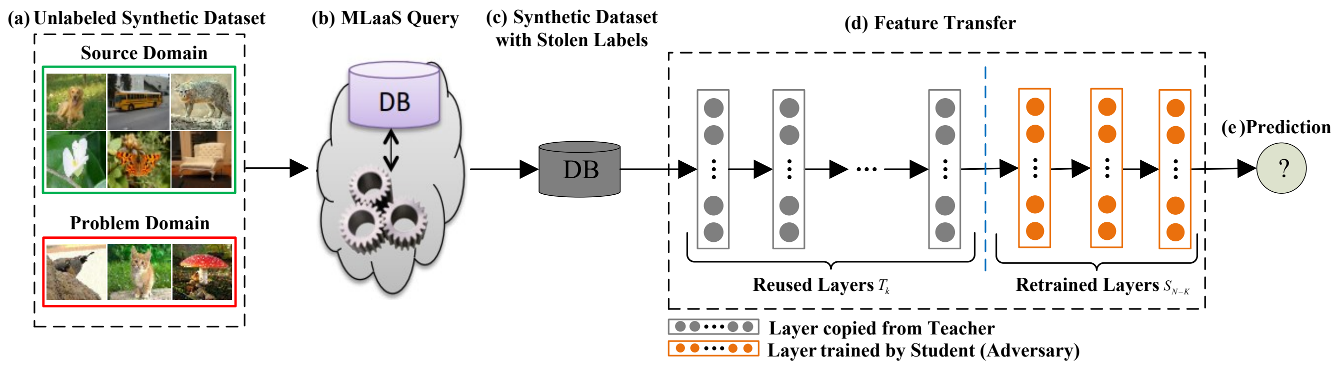
(a) Unlabeled Synthetic Dataset
Source Domain
Problem Domain
(b) MLaaS Query
DB
(c) Synthetic Dataset with Stolen Labels
DB
(d) Feature Transfer
Reused Layers T_k
Retrained Layers S_{N-K}
(e)Prediction
?
Layer copied from Teacher
Layer trained by Student (Adversary)

Fig. 2: Overview of the transfer framework for our proposed model theft attack. From left to right: (a) generate unlabeled adversarial examples as synthetic dataset. (b) query victim model using the generated synthetic dataset. (c) label adversarial examples according to the output of the victim model. (d) train the local substitute model using the synthetic dataset. (e) use the local substitute model for predictions. The local substitute model is expected to match the performance of the victim model.

with near-identical performance (*i.e.*, functionality) as a victim model f_v deployed on MLaaS platforms. Specifically, the adversary can launch the attack on a paid MLaaS to construct f_s that closely matches f_v even in black-box settings (*i.e.* the adversary has no internal knowledge of the victim model such as network architecture $\mathbf{A}$, exact training dataset $\mathbf{D}$, weights $\mathbf{W}$, etc.) The adversary's only capability is to collect a synthetic training dataset $T = \{(x, f_v(x))\}$ while providing informative input data $x \sim P_A(X)$ to retrain the substitute model f_s on such a dataset to replicate the functionality of victim model f_v.

B. Threat Model

The Machine learning as a service (MLaaS) provided by cloud-based platforms offer users a prediction API based on a DNN model pre-trained on the private dataset. The structures and/or designs inside API are usually inaccessible to the public due to economic and privacy concerns, *i.e.*, black-box. In our work, we assume an adversary targets such pay-as-you-go commercial machine learning services which provide cloud-based platforms to help users solve common deep learning problems such as data pre-processing, model training and model evaluation. The adversary will launch model theft attacks on a paid MLaaS to construct f_s that closely matches victim model f_v in black-box setting, meaning that the adversary has no inner knowledge of the victim model such as network architecture, exact training data, hyperparameter, weights, etc. The adversary's only capability is to query APIs with particular inputs (*i.e.*, malicious examples) and receive the resulting prediction or confidence scores. The substitute model f_s extracted by an adversary can be then arbitrarily used without incurring any query cost, *i.e.*, the adversary gains a free version of the victim model.

C. Transfer Architecture Construction

Figure 2 shows the overview of the transfer framework for our proposed model theft attack. Deep Neural Networks are made up of a cascade of computational layers which serves to learn automatic feature extraction and transformation. In general, these representations present different levels of abstraction in deep learning space. In deep neural networks, each hidden layer has a set of neurons connected to the neurons of the previous hidden layer. These neurons serve as computational units which transform input data into representations through particular activation functions. Many pre-trained models for various tasks are available for researchers to utilize directly, like AlexNet [52], VGGNet [53], VGGFace [54] and ResNet [55]. These models have given rise to classification accuracy in computer visual tasks with increasing computational complexity. Many techniques have been used to achieve image classification goals in practical applications of DNN. These pre-trained models can be used in transfer learning to apply the knowledge learned from source domains, $\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{N}$, to other different but related target domains, $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{M}$.

For the source task, we use four pre-trained networks including AlexNet, VGG19, VGGFace and ResNet50 as our basic architectures. In order to extract the multi-scale image representation, we remove the fully connected FC6 layer of the pre-trained VGG19 and add a DeepID layer formed by combining the features in the previous max-pooling layer and convolutional layer. Note that this extra DeepID layer is on top of the VGG19, followed by two fully connected layers (FC7 and FC8) which use the output of DeepID layer as input. The weights and bias in the previous convolutional layers are trained on the ImageNet dataset and shared by the source and target tasks. In comparison, both DeepID layer and two fully connected layers (FC7 and FC8) will be fine-tuned on synthetic datasets described in the following section. The dimensions of DeepID layer and fully connected layers FC7 will be fixed to 480 and 4096, respectively. The dimension of FC8 will be equal to the number of target classes it predicts. This network takes a fixed-size 224 $\times$ 224 RGB ConvNet images as input and boosts the performance of classification by pushing the depth to 19 weight layers.

IV. Model Theft Attacks

A. Adversarial Active Learning

1) Problem Analysis: By selecting an informative subset of unlabeled data $D_u(\mathbf{x})$ to present for labeling by a human expert, active learning (AL) aims to minimize the labeling cost in supervised learning while simultaneously maximizing performance of the classifier. The key idea of active learning is how an user can quantify the importance of each example in the active pool, for example, "useful" or "unusable". Motivated by the existing works on active learning [48], [49], [50], [51], we proposed a new learning methodology named margin-based adversarial AL for gathering a set of informative instances to train a substitute model with performance similar to the victim model f_v. We formally formulate this uncertainty sampling of margin-based adversarial AL as a querying function $\mathcal{Q}_{multiclass}$, which chooses a set of useful examples $\mathcal{D}_t(\mathbf{x}) \subseteq \mathcal{D}_u(\mathbf{x})$ from the given unlabeled data $\mathcal{D}_u(\mathbf{x})$, known as an active learning pool. The key idea of such margin-based active learning is that only a few examples from the pool of unlabeled data are useful or informative for determining the separating surface of the victim classifier, and all the other examples are superfluous to the classifier.

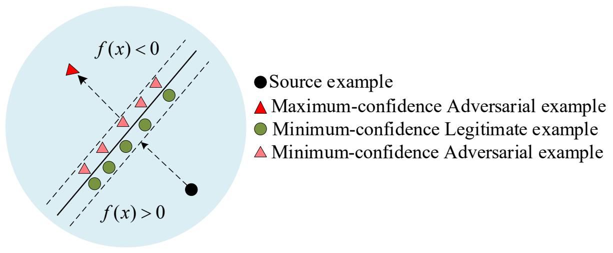
$f(x)<0$
$f(x)>0$
● Source example
▲ Maximum-confidence Adversarial example
● Minimum-confidence Legitimate example
▲ Minimum-confidence Adversarial example

Fig. 3: Illustration of the margin-based uncertainty sampling strategy.

Specifically, we apply a margin-based uncertainty sampling methodology as the adaptive strategy for boosting examples where the target classifier is the least confident, meaning that these selected adverserial examples lie on the global margin of target classifier. Since a multiclass classifier can be considered as a set of binary classifiers, we first propose the margin-based active learning algorithm for a linear binary classifier and provide a geometric illustration of the uncertainty sampling theory in Figure 3. Abstractly, we assume a learned affine classifier is a function $f : \mathcal{X} \to \mathcal{Y}$ which returns the prediction results (e.g., labels and confidence) within the range $\mathcal{Y}$ when given random input images $\mathbf{x} \in \mathcal{D}(\mathbf{x})$ (e.g., extracted from test dataset with the same distribution as the training dataset). We also denote the affine hyperplane as $\mathcal{H} = \{\mathbf{x} : f(\mathbf{x}) = 0\}$. We propose a new iterative attack procedure, named *FeatureFool* (The details will be demonstrated in the remaining part of this Section), to generate the adversarial examples with different confidence. Here, the adversary's goal is not to estimate the robustness of victim binary classifier $f_v(\mathbf{x})$, but rather to craft useful examples for the synthetic dataset which the domain $f_s(\mathbf{x})$ will be retrained on. In this work, the synthetic dataset generated by an adversary consists of two types of examples: one is minimum-confidence legitimate example, and the other is minimum-confidence adversarial example. In comparison to those examples with high confidence, the examples in synthetic dataset are more likely to provide useful information about affine hyperplane $\mathcal{H}$ of the binary classifier as a whole. For instance, as shown in Figure 3, we can see that the green circles (minimum-confidence legitimate examples) and pink triangles (minimum-confidence adversarial examples) are near the affine hyperplane $\mathcal{H}$, there is high uncertainty (i.e., the least confidence) and hence maximum performance with limited black-box queries.

We now extend the margin-based adversarial active learning algorithm to the multiclass case. The margin-based strategies in previous works are only effective in such a scenario where an adversary can determine the distance between the images of active learning pool and the affine hyperplane $\mathcal{H}$ of the target classifier. However, measuring such a distance is often intractable due to the high complexity of the geometrical shape of the affine hyperplane $\mathcal{H}$ in the multiclass models. We address this challenge by designing *FeatureFool* for exploring the useful examples where the target multiclass model has least confidence (LC). The proposed margin-based adversarial active learning methodology can be formulated as follows:

$$\mathcal{Q}^{LC}_{multiclass} : \mathbf{x}^{\star}_{\mathbf{s}} \in \underset{\mathbf{x}' \in \mathcal{D}_u(\mathbf{x})}{\arg\min}\, \kappa\left(\mathbf{x}', y, \mathbf{w}\right) \tag{1}$$

where y donates the predicted label corresponding to the first highest classification confidence, $\mathbf{w}$ donates the weights of victim classifier, κ denotes the output confidence while given random inputs $\mathbf{x}' \in \mathcal{D}_u(\mathbf{x})$. This approach chooses those informative examples from the given unlabeled dataset $\mathcal{D}_u(\mathbf{x})$ with the smallest margin (i.e., least confidence) and thus maximizes the uncertainty of instances. We further consider using these useful examples as synthetic dataset to retrain convolutional layers shared by the source domains. In the previous works, Silva *et al.* [14] directly make a large amount of superfluous queries to obtain the labeled data needed to generate the synthetic datasets and successfully train a local model with the near-perfect performance of the victim model. However, such large-scale queries would be expensive and make the attack easy to be detected by the MLaaS provider. To address these problems, we try the relevant queries by focusing on the two crucial objectives below: (1) Adopting *FeatureFool* to craft a basic informative dataset $\mathcal{D}_u(\mathbf{x})$ where each example $\mathbf{x} \in \mathcal{D}_u(\mathbf{x})$ has different classification confidence; and (2) Maximizing examples efficiency through uncertainty sampling strategy resulting in a subset of training examples $\mathcal{D}_t(\mathbf{x}) \subseteq \mathcal{D}_u(\mathbf{x})$. Our experimental results show that such adversarial examples would help considerably decrease the number of queries to victim models.

2) Synthetic Dataset Generation: We utilize the margin-based adversarial active learning algorithm to craft the informative examples and then query the victim model f_v for labels. Finally, the resulting image-prediction pairs can be viewed as a synthetic dataset to train the substitute model f_s for the purpose of replicating the victim model f_v inside the commercial API. We formally define the problem of finding an informative example x'_s selected by multiclass active function $\mathcal{Q}^{LC}_{multiclass}$ as follows:

$$x^{'}_{s} = \mathcal{Q}^{LC}_{multiclass}(x^{'}) \tag{2}$$

For the x', five generation strategies are considered in this paper (Due to vast majority of works on adversarial examples we focus only on those representative attacks here):

Random Sample (RS): For reference, we consider an extreme case where an attack randomly samples x from related domain and queries a victim API f_v as black-box in order to generate the synthetic dataset $T = \{x_i, f_v(x_i)\}_{i=1}^{N_{query}}$. In this case, an adversary can use all available images to obtain the best synthetic dataset and the resulting substitute model. However, lots of query operations make it easier to be detected by MLaaS providers.

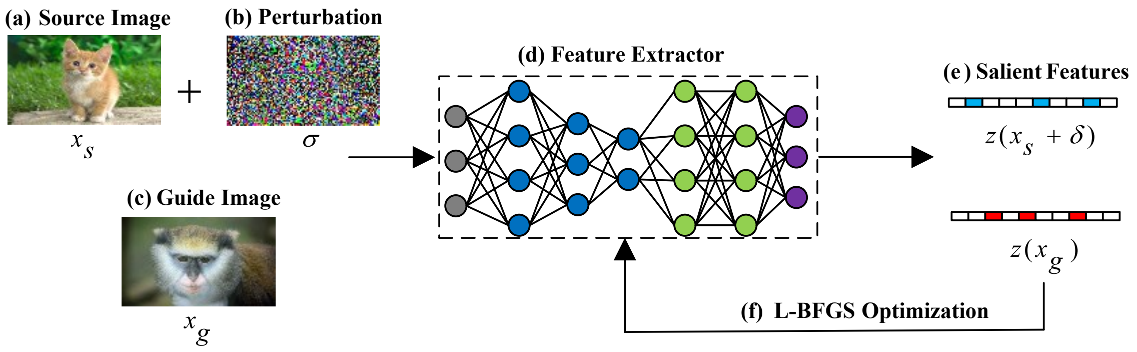

Fig. 4: Pipeline of the proposed *FeatureFool* attack method. We first input an image and extract the corresponding nth layer feature mapping by applying the non-linear filters to the output of last hidden layer. Then we compute the class salience map to decide which points of feature mapping should be modified. At last we search for the minimum distortion that satisfies the optimization formula.

Projected Gradient Descent (PGD): Madry et al. proposed the *Projected Gradient Descent* to effectively generate adversarial examples with multi-step iterations [37]. It exploits the first-order adversary information about the victim neural network and computes adversarial examples by using the following equation:

$$x_i' = \Pi_{x+\mathcal{S}}\left(x_{i-1}' + \alpha\, \text{sign}\left(\nabla_x J(F(x))\right)\right) \tag{3}$$

where ∇ denotes the gradient, $F(\cdot)$ denotes the network output and $J(\cdot)$ denotes the negative loss function. This attack can be viewed as a multi-step attack scheme which successfully solves the inner optimization problem. As such, errors on a legitimate input x can accumulate and eventually lead to an adversarial version of this given input that forces the victim network to output incorrect results, i.e., misclassification.

Carlini and Wagner Attack (CW): Carlini et al. [36] proposed new gradient-based attack algorithms using three different distance metrics (L_0, L_2 and L_∞). In the L_2 attack, they generate adversarial examples by solving the following optimization problem:

$$\begin{aligned} &\text{minimize } \mathcal{D}(x, x+\delta) + c \cdot g(x+\delta) \\ &\text{such that } x+\delta \in [0,1]^n \end{aligned} \tag{4}$$

where $D(x)$ denotes the L_2 distance function, $g(x)$ denotes the objective function which can be defined as:

$$g(x) = \max\left(\max\left\{Z(x)_i : i \neq t\right\} - Z(x)_t, -\kappa\right) \tag{5}$$

where $Z(x)$ denotes the input of the softmax function. As mentioned in [36], an attacker can easily control the confidence which adversarial image misclassification occurs by carefully selecting the parameter κ in Equation (5). This technique allows the L_2 attack to effectively craft those "informative" examples which lie approximately on the decision boundary of the victim classifier. Hence, we mainly consider using the L_2 attack mentioned in [36] to generate the synthetic datasets for retraining the substitute model.

FeatureAdversary (FA): Sabour et al. [45] introduce a new attack model by minimizing the L_p distance (*i.e.*, L_p norms) between the internal feature presentation of images pairs (source image x_s, target image x_t) of victim classifier f (In this paper, we call this attack **FeatureAdversary** (FA)). More precisely, we describe their problem as follows:

$$\begin{aligned} \text{minimize} \quad & D\left(\phi_K\left(x_s'\right), \phi_K\left(x_t\right)\right) \\ \text{such that} \quad & d\left(x_s', x_s\right) < \eta \end{aligned} \tag{6}$$

where $\phi_k(x)$ denotes the feature presentation of input image x at the kth layer of trained victim classifier $f(x)$, x_s denotes the source image, x_g denotes the guide image with expected target label l, *i.e.*, $f(x_g) = l$, the parameter η is the constraint that limits the bias of any single pixel color within budget η.

For the distance $D(.)$, if $\boldsymbol{a_k}$ and $\boldsymbol{b_k}$ are two feature vectors at kth layer of the classifier, then the distance $D(a_k, b_k)$ can be defined as $D(a_k, b_k) = \|a_k - b_k\|_p$, where the p-norm $\|\cdot\|_p$ of the vector $\mathbf{v} = (v_1, \ldots, v_n)$ can be denoted as:

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p} \tag{7}$$

They demonstrated that, with a tiny fixed value of η, **FA** attack can effectively craft adversarial examples which are generally imperceptible to humans, even in the case of targeting different intermediate layers $\phi_k(x)$, where $k = 1, 2, ..., m$.

FeatureFool (FF): As the classification models become more complex, normal adversarial attacks stop producing satisfied results. Therefore, these attack methods are not suitable for generating synthetic dataset with samples lying approximately on the decision boundary of victim classifier $f_v(x)$ in our scenarios. In this paper, we propose a novel adversarial attack method *FeatureFool* to improve the query efficiency of samples. It uses feature-based optimization algorithms for producing natural adversarial examples to mislead the large-scale deep learning models to output incorrect classification results.

Our proposed adversarial attack algorithm starts from the initial L-BFGS optimization problem: given an image x, targeted classifier $f(x)$ and targeted class l, the goal is to solve the following box-constraint optimization problem:

$$\begin{aligned} \text{minimize} \quad & d\left(x_s', x_s\right) \\ \text{such that} \quad & f(x_s') = l \\ & x_s' \in [0,1]^n \end{aligned} \tag{8}$$

where $d(.)$ denotes the loss function, x_s' denotes the examples crafted by adversarial attack methods, parameter η denotes the constraint that limits the amplitude of perturbation within a budget η. That is, the L-BFGS attack aims to find a particular perturbation by iteratively optimizing Equation (8), such that the perturbed image x_s' would be correctly classified by a human but incorrectly classified to the given target class l by a classifier $f(x)$.

Unfortunately, the highly non-linear constraint in Equation (8) makes it difficult to solve the optimization problem [34], [35], [36]. To address this challenge, we search for an alternative constraint to find an optimization solution as quickly as possible. Moreover, as mentioned in [45], [46], two random images can be classified into the same class by the victim model if their inner feature mappings are similar. This similarity controls whether the generated adversarial samples can be misclassified as the chosen malicious label l. Putting these ideas together, we apply the triplet loss as a new penalty method for solving the optimization problem in Equation (8) and rewrite the optimization as follows:

$$\begin{aligned} &\text{minimize } d\left(x_s', x_s\right) + \lambda \cdot \text{loss}_{f,l}\left(x_s'\right) \\ &\text{such that } x_s' \in [0,1]^n \end{aligned} \tag{9}$$

where $d(.)$ is the L_p norm distance which quantifies the similarity between two images in the 3-D space, λ is the suitably chosen coefficient that helps L-BFGS algorithm minimize both of the loss terms simultaneously. In our experiment, we empirically find that the smallest value of λ would be the best choice to generate more deceptive solutions x_s' while keeping the $\text{loss}_{f,l}\left(x_s'\right) > 0$, which makes sure that the perturbation budget in feature space can be maximized. For the triplet loss $\text{loss}_{F,l}\left(x_s'\right)$, we formally define it as:

$$\begin{aligned} \text{loss}_{f,l}\left(x_s'\right) = &\max(D(\phi_K(x_s^{'}), \phi_K(x_t)) - \\ &D(\phi_K(x_s^{'}), \phi_K(x_s)) + M, 0) \end{aligned} \tag{10}$$

Here $\phi_k(.)$ is the internal feature representation at k_{th} hidden layer of target classifier, $D(.)$ is a distance function that measuring the similarity between two internal representations under the constraint M, which defines the constant margin of triplet loss. Note that the reason why we choose the triplet loss as $\text{loss}_{f,l}\left(x_s'\right)$ in our attack scheme is that, compared to other loss functions like L_p loss (i.e., L_0 loss, L_2 loss and L_∞ loss) and VGG loss, the triplet loss can lead to faster convergence of Equation (9) and better performance of adversarial attacks.

For the purpose of simplicity, we define the M as follows:

$$M = \alpha - \frac{1}{n_{y_s}^2 - n_{y_s}} \sum_{i,j \in y_s} \left\|\phi_K(x_i) - \phi_K(x_j)\right\|^2 \tag{11}$$

Here α is a constant (Empirically, we set $\alpha = 0.5$), n_{y_s} is the number of input samples in the class y_s.

In order to solve the reformulated optimization problem above, we apply the box-constrained L-BFGS for finding a minimum of the loss function in Equation (9), which is considered particularly well-suited for parameter estimation in deep learning. The pipeline of the *FeatureFool* is shown in Figure 4.

3) Evaluation Metric: We use the Average Test Error (ATE) over test set D_{test} to evaluate the effectiveness of the proposed model theft attack. Given an input sample $x \in D_{test}$, ground-truth values $f(x)$ and prediction values $\hat{f}(x)$, then the ATE is given by:

$$ATE = \sum_{(x,y)\in D_{test}} \frac{d(f(x), \hat{f}(x))}{|D_{test}|} \tag{12}$$

In our experiment, the ATE refers to the extraction accuracy under the test set. A lower ATE is expected when an adversary aims to replicate the functionality of the victim model using the substitute model.

B. DNN Training

Our model stealing attack aims to retrain a substitute model in the target domain with near-perfect performance of the victim model. We adopt five synthetic dataset generation strategies, including *RS*, *PGD*, *CW*, *FA* and *FF*. For the *RS* strategy, we randomly sample a set of examples as the training dataset to re-train our substitute model. Different the *RS* strategy, the training procedure using adversarial examples generated by the these approaches is described in Algorithm 1.

First, we randomly sample a small set X_0 from target domain as the initial dataset S_0. We use the adversarial examples generation algorithms to launch adversarial attacks on the original substitute model and craft a small amount of malicious examples on this initial dataset. These macilious examples can easily mislead the local model to output incorrect results with 100% success rate.

Then we construct the synthetic datasets by querying the victim models with these malicious examples. In our implementations, the synthetic datasets differ from those used by the victim models for training. We use the pre-trained model in candidate Model Zoo (see Figure 1) as our transfer architecture and construct the corresponding substitute model. The synthetic dataset is applied to fine-tune this substitute model by only retraining the last few fully connected layers but leaving the previous convolution layers frozen. We significantly reduce the number of queries required to extract the victim model using transfer learning with adversarial examples.

Finally, we iteratively use the adversarial attack algorithms to generate synthetic dataset D_s and then retrain the local substitute model using transfer learning on this synthetic dataset. This helps us achieve higher accuracy and test agreement on the test set, which leads to an increase in the similarity of boundary between the substitute and victim models.

The key observation is that by applying the *FeatureFool* algorithm in step 4 of Algorithm 1, the output f_s is more similar to f_v. The main reason for this is that by adding the perturbation component to original images, an adversary can construct the data set D_s that lie approximately on the decision boundary of the classifier f_v. Since D_s and S_i' have the same set of images and D_s has labels from f_v, the data set D_s trains the classifier f_s better than random images.

Here we give some intuitive justification based on a simplified model: if the models f_s and f_v are linear classifiers in $\mathbb{R}^p$ with two classes, then the decision boundaries are

Algorithm 1 Training process of DNN substitute model: for victim model f_v, the adversarial examples generation algorithms $\mathcal{G}$ (*e.g.*, *PGD*, *CW*, *FA* and *FF*), the substitute model f_s, an adversarial set of examples S, a maximum number of iterations m, a synthetic dataset $D_s(x)$ and the random set of images with same distribution X_0, X_1, X_2, $\cdots$, X_m

```
Input: $f_v$, $f_s$, $X_0$, $X_1$, $X_2$, $\cdots$, $X_m$
Output: Retrained substitute model $f_s$
 1: Initialize $i \leftarrow 0$,
 2: while $i < m$ do
 3:     $S_i \leftarrow X_i$
 4:     $S_i' \leftarrow \{\mathcal{G}(f_s(x), x) | x \in S_i\}$
 5:     //Craft Adversarial Examples
 6:     $D_s \leftarrow \{(x, f_v(x)) | x \in S_i'\}$
 7:     //Generate synthetic dataset
 8:     $f_s \leftarrow$ Transfer $\{(f_s, D_s)\}$
 9:       //Transfer for the synthetic dataset
10:     $i \leftarrow i + 1$
11: end while
```

hyperplanes in $\mathbb{R}^p$. If in addition, all the points in S_i lie on the decision boundary, then as long as the number of points in S_i is larger than p, we can recover the decision boundary from $D_s = \{(f_v(x), x) | x \in S_i\}$ and the classifier f_v exactly: the hyperplane containing all p points is the decision boundary. On the other hand, if the points in S_i are randomly chosen from $\mathbb{R}^p$ and do not lie on the decision boundary, then it would requires much more points to recover the decision boundary (the hyperplane).

V. Experimentation

A. Experiment Setup

In this section, we discuss the experimental results of a large-scale evaluation on five popular MLaaS platforms, including those hosted by Microsoft, Face++, IBM, Google and Clarifai. We create three victim models ourselves, by uploading well-labeled training sets. These are the Microsoft Cloud Vision Service, IBM Watson Visual Recognition, and Google AutoML Vision, and they are trained for traffic sign recognition, flower recognition and face recognition, respectively. This simulates a user training the cloud models on private data sets using these services. Then other users can access the created cloud models by querying the resulting prediction APIs in a pay-as-you-go format, quickly fetching results without many restrictions. However, the DNN models behind these APIs and the data used for training the DNN models are at all points inaccessible to the public, or users other than the creator (*i.e.* they are treated as black box models during our attack). Additionally, we also consider two existing black-box models inside Face++ Emotion Recognition API and Clarifai Safe for Work (NSFW) API. Different from the previous MLaaS platforms provided by Microsoft, IBM and Google, the Face++ emotion recognition and Clarifai NSFW models only allow users to directly query the pre-trained models through MLaaS APIs to meet their needs. They do not allow fine tuning on thier models for individual needs.

Traffic Sign Recognition. We upload a well-labeled training set and train a victim model for traffic sign recognition using Microsoft Traffic Recognition API. Traffic sign recognition based on deep learning aims to classify different types of traffic signs from images, which can be used by self-driving cars to automatically recognize traffic signs. The dataset used to train the victim model online is the GTSRB dataset [56] with a training set containing 39000 images of 43 different traffic signs and a corresponding testing dataset containing 8000 images.

Flower Recognition. The victim model pre-trained on Microsoft Cloud Vision Service is for flower recognition. The flower recognition classifies images of flowers into different categories (e.g., Daisy, Sunflower, Fire lily, etc). This task is known to be difficult as the flower dataset is less uniform. Flowers in the same species may have various color appearances, and the target objects are influenced by several lighting conditions. Consequently, it is a well-known classification problem which people would like to use deep learning to solve. The victim classification model is trained on the VGG Flowers dataset [57], including 6146 images from 102 different flower types and the corresponding testing dataset which contains 1020 images with 10 images for each 102 classes.

Face Emotion Recognition. For the Face++ Emotion Recognition API [16], users cannot access the exact training sets or its distribution and only observe the outputs (*i.e.*, labels or confidence score) to chosen inputs by querying the API. The victim model pre-trained on Face++ Emotion Recognition API only allows an end user to upload a photo and fetch the response about the emotions of detected faces, *i.e.*, black-box. This API returns probability scores on the likelihood that an image contains emotions such as happiness, fear, surprise, anger, disgust, neutral and sadness. Different from previous victim models trained by users on their datasets, the Face++ Emotion Recognition API only provides the interface which can be queried by users, meaning that users cannot access the exact training process. Since the official test set of Face++ Emotion Recognition API are not provided, we create a test set which contains 1010 images in 7 categories roughly.

Offensive Content Moderation. The Clarifai Not Safe For Work (NSFW) API recognizes whether images contains various offensive contents which can be utilized by users to automatically filter these contents from their platforms. Typically users access the API by querying it with image inputs and receive resulting confidence scores for two output labels (NSFW - Not Safe For Work and SFW - Safe For Work). However, the details of the victim model inside the Clarifai NSFW API, such as training set and network architecture, are generally inaccessible to users. Here, we attempt to steal the black-box victim model provided by Clarifai. We also collect 1k images in two categories from Github as the test set to evaluate the victim/substitute model accuracy.

These models inside the MLaaS act as the black-box victim models of our model stealing attack. We launch the attack against these victim models without knowing the exact training sets and the internal information of these models. In our attack scheme, we re-train the local substitute model with the synthetic dataset which is generated from the examples by querying the victim model. Here, we use five different strategies (e.g., *RS*, *PGD*, *CW*, *FA* and *FF*)) to craft these query examples for the purpose of comparison. In particular,

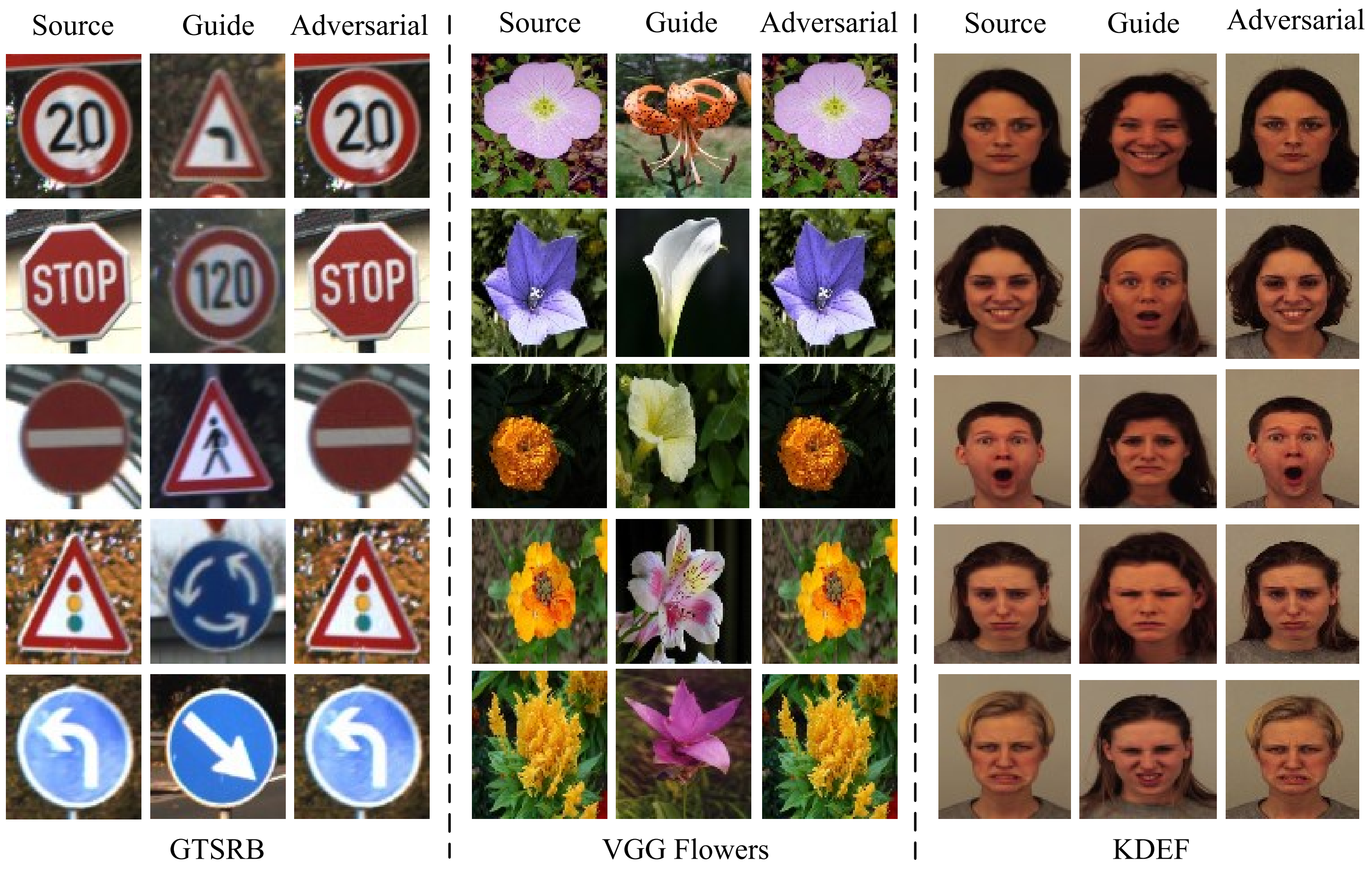

Fig. 5: Adversarial Examples generated by our *FeatureFool* algorithm.

the adversarial examples generated by our *FF* are shown in Figure 5.

We elaborate on the ablation study in two aspects: (a) We study the influence of different datasets and/or transfer architecture selections on the model stealing attack effectiveness; (b) We also show the comparison between our model stealing attack and existing attacks such as F. Tramèr attack [6], Correia-Silva attack [14] and Papernot attack [21] against commercialized MLaaS platforms in real world. The details of these comparison experiments will be demonstrated in the remaining sections.

We leverage open-source implementations of four popular pre-trained models: AlexNet, VGG19, VGGFace and ResNet50. All experiments were carried out on a server equipped with Intel E5-2623 v4 2.60GHz processor, 16GB of RAM, four NVIDIA GeForce GTX 1080Ti GPUs. The training starts from a relatively large learning rate and then the learning rate would decrease during training to allow for more fine-grained weight updates. The pre-trained weights are used to initialize our model extraction attack framework. We split the training vectors into two parts: a training dataset and a validation dataset. Then we use the stochastic gradient descent (SGD) method to minimize the cross-entropy loss while training the designed framework. We also apply some basic but powerful data augmentation techniques like flips, rotations, and scaling.

B. MLaaS Models Extraction Attacks

1) Case Study 1: Traffic Recognition Model: We train a model for the GTSRB dataset through Microsoft Custom Vision inference and set it up as the black-box victim model. The experimental results of our stealing attack on this victim model are shown in Table III. We use the designed VGG19_DeepID as the transfer architecture of the substitute model and generate five types of synthetic datasets for training this substitute model. With 0.43k queries, our substitute model achieves only 10.21% (13.10×) accuracy with random examples, 10.49% (13.16×) accuracy with *PGD* examples, 12.01% (15.53×) accuracy with *CW* examples, 11.64% (14.94×) accuracy with *FA* and 15.96% (20.48×) accuracy with *FF*, illustrating that too few queries fail to extract enough information from the victim model for model stealing attack. With 2.15k queries, our local substitute model achieves 70.03% accuracy with *RS* samples, 72.20% accuracy with *PGD* examples, 74.94% accuracy with *CW* examples, *71.30%* accuracy with *FA* samples and 76.05% accuracy with *FF* examples, which is similar to the 77.93% accuracy achieved by the victim model trained on Microsoft Traffic Recognition API. Our method can achieve the same level of accuracy with fewer queries.

The total cost for stealing victim model with 76.05% test accuracy is around $2.15 US dollars. Moreover, a local substitute model trained by adversarial examples always achieves higher accuracy and test agreement than the model trained by random samples, especially when the number of queries to the victim model is small.

Service	**Model**	**Queries**	**Dataset**					**Price ($)**
			RS	Non-Feature-based		Feature-based		
				PGD	CW	FA	FF	
Microsoft	Traffic	0.43k	10.21 (13.10×)	10.49 (13.16×)	12.10 (15.53×)	11.64 (14.94×)	**15.96 (20.48 ×)**	0.43
		1.29k	45.30 (58.13×)	59.91 (76.87×)	61.25 (78.60×)	49.25 (63.20×)	**66.91 (85.86×)**	1.29
		2.15k	70.03 (89.86×)	72.20 (92.65×)	74.94 (96.16×)	71.30 (91.49×)	**76.05 (97.63×)**	2.15
	Flower	0.51k	26.27 (28.97×)	27.84 (30.70×)	29.41 (32.43×)	28.14 (31.03×)	**31.86 (35.13×)**	1.53
		1.53k	64.02 (70.59×)	68.14 (75.14×)	69.22 (76.33×)	68.63 (75.68×)	**72.35 (79.78×)**	4.59
		2.55k	79.22 (87.35×)	83.24 (91.79×)	**89.20 (98.36×)**	84.12 (92.76×)	88.14 (97.19×)	7.65
Clarifai	NSFW	0.50k	65.10 (70.68×)	66.20 (71.88×)	71.50 (79.80×)	66.20 (71.88×)	**76.20 (82.74×)**	0.60
		1.00k	72.30 (78.50×)	74.90 (81.32×)	85.10 (93.60×)	75.00 (81.43×)	**87.10 (94.57×)**	1.20
		1.50k	76.10 (82.63×)	78.50 (85.23×)	89.70 (97.39×)	80.20 (87.08×)	**91.60 (99.46×)**	1.80
Face++	Emotion	0.68k	26.10 (36.20×)	30.19 (41.87×)	42.08 (58.36×)	37.05 (51.39×)	**44.00 (61.03×)**	0.34
		1.36k	43.14 (59.83×)	50.19 (69.61×)	**67.23 (93.25×)**	60.29 (83.62×)	65.33 (90.61×)	0.68
		2.00k	58.10 (80.58×)	62.00 (85.99×)	**71.19 (98.74×)**	64.10 (88.90×)	70.76 (98.14×)	1.00

TABLE III: Comparison of performance on victim models and their local substitute models. We report the accuracy on test sets in two forms: absolute ($x\%$) or relative to black-box victim model (×). Accuracy (%) of black-box victim models are: 77.93 (100×) for Microsoft traffic recognition model, 90.69 (100×) for Microsoft flower recognition model, 92.10 (100×) for Clarifai Not Safe For Work (NSFW) model and 72.10 (100×) for Face++ emotion recognition model, respectively.

Upon quantitative analysis, we observe that: (i) Adversarial perturbation increases the diversity of synthetic datasets, resulting in a more successful transfer set. As a result, adversarial examples help extract more decision information from the victim model than random samples and hence improve the query effectiveness, but this advantage shrinks as the number of queries increases. (ii) Compared to *RS*, *PGD* and *FA*, the substitute models trained on the *CW* and *FF* synthetic datasets achieve better performance on the same test dataset. The main reason for this is that, by solving optimization problems for generating "informative" examples, *CW* and *FF* strategies control the misclassification confidence by adjusting the parameter κ in Equation (5) and the parameter M in Equation (10), and thus effectively generate adversarial examples which lie approximately on the decision boundary of victim classifiers. Compared to *CW* method, the *FF* attempts to vary the contribution of different feature components and generate adversarial examples that can contribute with more boundary information about the victim model. Thus, our *FF* method can achieve the same level of accuracy with fewer queries than the *CW* method.

2) Case Study 2: Flower Recognition Model: The experimental results of this victim model are shown in Table III. We use the popular 50 layer ResNet50 model trained on the ImageNet dataset as the transfer architecture of our substitute model. As shown in Table III, we can see the accuracy of our self-trained victim model is 92.01% (100×). Our substitute model achieves 31.86% (35.13×) accuracy with 0.51k queries and 72.35% (79.78×) accuracy with 1.53k queries by using the *FF* training set. With few queries (e.g., 0.51k and 1.53k), the *FF* strategy leads to better performance compared to the other strategies such as *RS*, *PGD*, *CW* and *FA*. With 2.55k queries, substitute model trained on the *FF* synthetic dataset obtains 97.19× performance of the black-box victim model, which is comparable to the performance of the substitute model trained on the *CW* synthetic dataset (98.36×). In this case, compared to *CW* strategy, feature-based adversarial attacks such as *FA* and *FF* may add more perturbations to legitimate examples in order to maximize the uncertainty of these examples away from decision boundary of the victim classifier. Although adversarial examples with large perturbations pollute the synthetic training set, the substitute model trained on our *FF* still achieves strong performance on all test set, which is similar to the accuracy achieved by the substitute model trained on the *CW* synthetic set. These trends also appear while stealing other black-box models inside MLaaS platforms (We illustrate the details in the following sections). Moreover, with 2.55k adversarial queries to Microsoft custom vision service, it costs \$7.65 US dollars to extract a substitute model that achieves at least 91.76× performance of the victim model.

3) Case Study 3: Emotion Recognition Model: So far our attack framework assumes that the victim models inside APIs are trained by users themselves. Here we consider a more representative attack scenario where an adversary who targets the pay-as-you-go commercialized MLaaS platform has no knowledge about the exact training data or its distribution (we assume that the adversary has some knowledge of the training set but not the details, avoiding to use the irrelevant images in the test), model architecture and training strategy, but can observe the classification outputs. Specifically, we utilize the proposed attack algorithm to steal the Face++ Emotion Recognition API in the black-box setting. The transfer architecture of our substitute model is the VGGFace trained on VGG-Face dataset to recognize 2622 identities. The dataset utilized to train the substitute model is the KDEF dataset [58], which contains 4900 pictures of human facial expression. The set of pictures contains 70 individuals displaying 7 different emotional expressions, including happy, fear, sad, surprise, angry, neutral and disgust. Each expression is viewed from 5 different angles. The initial training data consists of random 224 ×

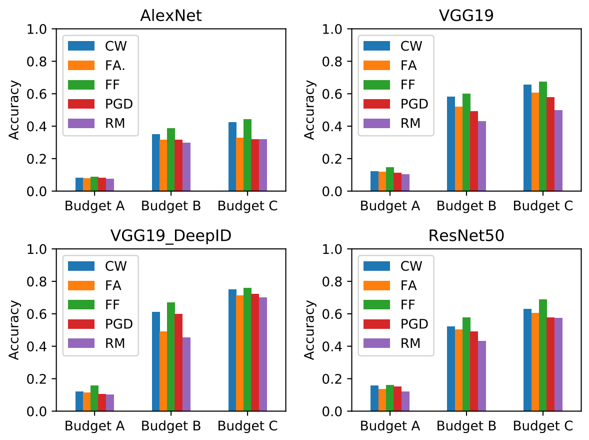

Fig. 6: Architecture Choice for stealing Microsoft **Traffic Recognition** API at various budgets ($A = 0.43$k$, B = 1.29$k$, C = 2.15$k)

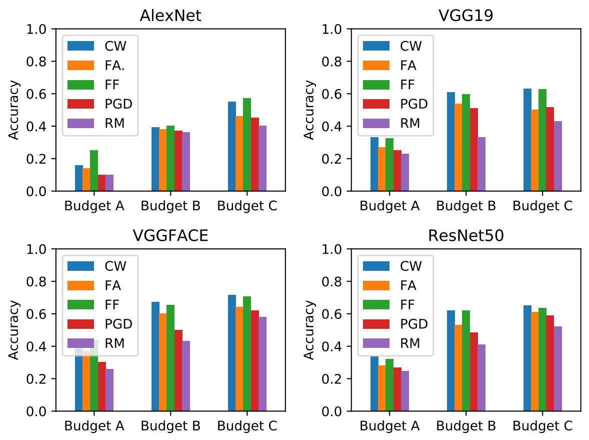

Fig. 8: Architecture Choice for stealing Face++ **Emotion Recognition** API ($A = 0.68$k$, B = 1.36$k$, C = 2$k)

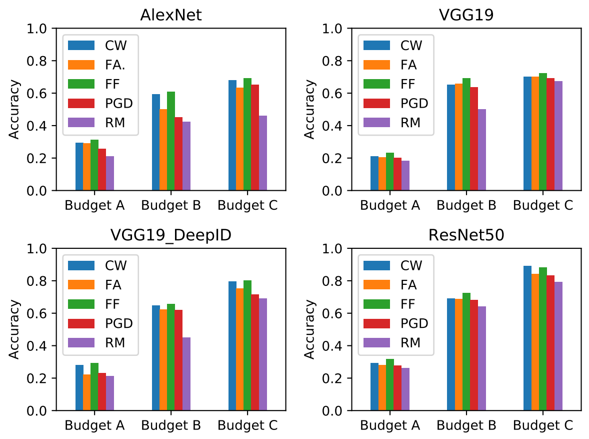

Fig. 7: Architecture Choice for stealing Microsoft **Flower Recognition** API at various budgets ($A = 0.51$k$, B = 1.53$k$, C = 2.55$k)

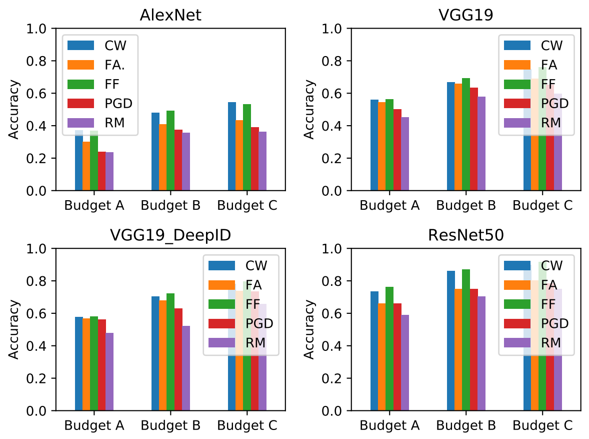

Fig. 9: Architecture Choice for stealing NSFW API ($A = 0.50$k$, B = 1.00$k$, C = 1.05$k)

224 pixel patches cropped from these images and it is further augmented by rotating 90 degree or transforming to gray scale with 50% probability of each image. The experimental results of our attack are shown in Table III. We can see our substitute model achieves 65.33% (90.61×) accuracy with 1.36k queries and 70.76% (98.14×) accuracy with 2k queries by using *FF* adversarial examples, which approaching the 71.17% (98.74×) accuracy achieved by the substitute model trained on the *CW* adversarial examples. The substitute model trained by adversarial examples always achieves better performance than the model trained by the random samples.

4) Case Study 4: Clarifai Safe for Work (NSFW) API: The victim model pre-trained on Clarifai Not Safe For Work (NSFW) API can recognize whether images include inappropriate contents on the Internet. In general, it is treated as Not Safe For Work if the NSFW probability is greater than 0.85. Similar to previous experiments, we apply the proposed attack algorithm to the black-box Clarifai NSFW API. The training dataset used to train our surrogate model (ResNet50) is randomly collected from Github opensource platform, which contained 1.5k pictures (half of NSFW and half of SFW). We then evaluated the accuracy of victim model by using 1k random images which are different with training data (1/2 for SFW and 1/2 for NSFW). The accuracy of our victim model is 92.10%. Experimental results show that our substitute model achieves 87.10% (94.57×) accuracy with 1k queries and 91.60% (99.46×) accuracy with 1.5k queries by using *FF* examples which is better than using random examples, i.e., 76.10% (82.63×) accuracy with 1k queries and 76.10% (82.63×). In all case, the substitute model trained on *FF* method achieves the best performance on the test set in comparison to other adversarial examples generation methods such as *PGD*, *CW* and *FA*.

C. Synthetic dataset and Transfer Architecture Selection

In the previous sections, we demonstrate that our attack framework can effectively replicate the functionality of an victim model inside the API with similar performance. This is achieved while concurrently applying the fixed substitute model architecture and dataset generation algorithm. In this section, we further evaluate how the attack effectiveness varies with different synthetic datasets D_s and transfer architectures f_s. In these experiments, we consider five strategies: *RM*, *PGD*, *CW*, *FA* and *FF*. In our attack scheme, these strategies are applied to generate the synthetic data set D_s for re-training the substitute model f_s chosen from the five popular network architectures, including AlexNet, VGG19, VGG19_DeepID, VGGFace and ResNet50.

Figures 6, 7, 8 and 9 summarize the influence of different datasets and/or architecture selections on the attack effectiveness. As shown in Figure 6, we find that the performance of the substitute model re-trained on the adversarial examples is usually better than the random exanples accross different network architectures when we increase the number of training examples from Budget A (0.43k) to Budget B (2.15k). The same trend appears in Figure 7, Figure 8 and 9. Our substitute models significantly recover the original performance of black-box f_v using the FF synthetic datasets, i.e., 97.19× for the flower recognition at $C = 2.55$k, 99.46× for NSFW at $C = 1.50$k and 98.14× for the emotion recognition at $C = 2.00$k.

We also analyze the performance of substitute models while using different pre-trained models as our transfer architectures. From Figures 6, 7, 8 and 9, we observe that performance of our substitute model can be influenced by both the model complexity and task relevance. Therefore, in order to extract a copy of the victim model, an adversary can focus on the following aspects: (i) Choosing a more complex/relevant network and the transfer architecture. In both cases, AlexNet networks achieve the lowest accuracies after stealing a victim model. A significantly more complex model VGGNet (VGG19, VGG19_DeepID and VGGFace)/ResNet50 is beneficial while stealing a victim model. Further, as seen in Figure 8, VGGFace, which is relevant to face recognition tasks, achieves the best accuracy across all choices of substitute model architectures while targeting the face emotion recognition API. This further indicates that, if the attacker does not know the exact architecture of the victim model, using a more complex and task relevant model as the transfer architecture is almost always beneficial for the adversary. (ii) Sampling images relevant to the classification problem (relevant queries). This is because irrelevant queries generally lead to noisy labels and hence impose additional difficulty for re-training the substitute model.

D. Comparison to Existing Attacks

As shown in Figure 10, we compare with the existing state-of-the-art attack methods proposed by previous works, including F. Tramèr attack [6], Correia-Silva attack [14] and Papernot attack [21]. In our implementations, we keep the architectures of the substitute models fixed and evaluate how the attack effectiveness varies with different synthetic dataset generation methods (e.g., Tramer, Papernot, and Correia-Silvia). The same trend appears while we use different transfer architectures to copy the black-box victim model. In our implementations, we launch these model stealing attacks on commercialized MLaaS platforms in the real world, including those hosted by Microsoft, Face++ and Clarifai. The substitute model architectures are chosen from five popular pre-trained models, including AlexNet, VGG19, VGG19_DeepID, VGGFace and ResNet50. We use the synthetic dataset generated by our FF algorithm to re-train these models in order to replicate the functionality of victim API. Experimental results demonstrate that our attack framework can steal large-scale deep learning models with high accuracy, few queries and low costs simultaneously, while previous works fail in at least one or two aspects. More detailed analysis of experimental results is given below.

- When compared, our attack is more effective for extracting large scale DNN model with few queries than F. Tramèr attack [6]. From Figure 10(a) We can see that, our substitute model, which uses the VGG19_DeepID as the transfer architecture, is trained on adversarial examples generated by our *FF* algorithm and achieves 74.25% accuracy with 2.15k queries, which is better than F. Tramèr attack's results (their substitute model achieves 15.97% accuracy with 2.15k queries). We find significant query efficiency improvements, e.g., while Tramèr reaches 25.17% test accuracy at $B = 5.00$k, our attack achieves this 3.9× quicker at $B = 1.29$k. Similar results are shown in Figure 10(b), Figure 10(c) and Figure 10(d). This is because Tramèr attack uses line-search to find those samples which are overly similar (i.e., the same class), resulting in poor training set and eventually degrading the performance of the substitute model.
- Different from the Correia-Silva attack [14], we vary the architecture of the substitute model and re-train the model on the synthetic dataset which is generated by adversarial examples labeled by querying the victim model. Take the Microsoft Flower Recognition as an example (Figure 10(b)), with 1.53k queries, our attack outperforms the Correia-Silva attack in terms of prediction accuracy on the same test set (up to 27.84 percentage points). Experimental results also demonstrate that our substitute model (ResNet50) achieves higher accuracy as the number of queries increases from 0.51k to 3.00k, which is better than the performance of stolen model by Correia-Silva attack (15.10% $\sim$ 78.92%). Since the model architecture is not complicated, we can conclude that the success rate of stealing black-box victim model depends not only on the network architecture of the substitute model, but also on the efficacy of the dataset augmentation method.
- In order to boost the performance of the substitute model, a Jacobian-based dataset augmentation method is explored in the Papernot attack [21]. We reproduce the exact setting of this attack reported in [21] and show the experimental results in Figure 10. These results show that adversarial examples help us improve the query effectiveness of examples augmented by Jacobian-based method. From Figure 10(a), we can see most of our substitute models trained by adversarial examples achieves higher performance (For example, the VGG19_DeepID achieves 62.75% accuracy with 1.29k queries and 74.25% accuracy with 2.15k queries) than the model trained on the Jacobian-based augmentation dataset. For the Flower Recognition, our attack achieves the same accuracy as Jacobian-based method with queries, e.g., the Jacobian-based method reaches 81.76% test accuracy at $B = 2.55$k, our attack achieves this 1.3 × quicker at $B = 2.00$k. While targeting the NSFW (Figure 10(c)) and the

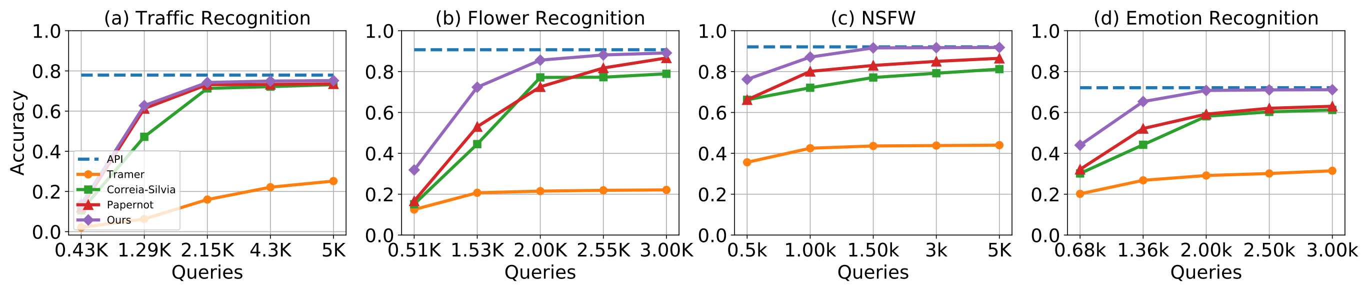

Fig. 10: Comparison of the performance over victim models between our method and previous works.

Emotion Recognition (Figure 10(d)), our substitute models always achieve the best accuracy on the test set with different sizes of queries. The main reason of this is that, compared to Jacobian-based method, our *FF* helps extract more decision information from the victim model and hence improve the query effectiveness.

E. More Commercial APIs

In addition to the commercial APIs we test our attack and show the comparison results in the previous sections, we further extend out attack to extra two commercial platforms: IBM Watson Visual Recognition [17], Google AutoML Version [18]. Specifically, the target model pre-trained on IBM Watson visual recognition is for face recognition. The dataset used to train the model is the PubFig83 dataset [59] containing 12502 images of 83 different individuals and a relating test dataset of 830 images (10 images per class). Experimental results show that our substitute model (VGG19_DeepID) achieves 78.43% accuracy with 2075 queries and 83.73% accuracy with 3320 queries, approaching the 84.94% accuracy achieved by the victim model on IBM Watson Visual Recognition API.

Similar to the Microsoft flower recognition, we use the same dataset to train and test the victim model on Google AutoML Version. The experimental results demonstrate that, for a synthetic dataset containing 2550 images, our local substitute model (ResNet50) achieves 60.10% accuracy with random samples, and 88.14% accuracy with *FeatureFool* examples, which is similar to the 89.22% accuracy achieved by the vicm model trained on the Google AutoML API.

VI. Discussions

A. Potential Defenses

We have shown in Section V that an adversary can successfully extract the victim models from MLaaS cloud platforms. As our findings undermine MLaaS platforms' privacy and integrity, defense mechanisms should be developed and applied to protect cloud-based MLaaS platforms from model stealing attacks. In this section, we evaluate one latest defense named *PRADA*, and further propose a novel defense mechanism that can effectively and adaptively defend against the malicious queries to MLaaS platforms. We discuss the details of these countermeasures.

1) Evasion of PRADA Detection: Juuti et. al [20] propose a new defense method, known as *PRADA*, to detect model extraction attacks. It analyzes the distribution of queries to victim models and rely on how these queries relate to each other for detecting model extraction attacks. In the experimental stage, we reproduce the exact setting of this defense reported in [20]. We evaluate the *PRADA* defense on all of the model theft attacks using different synthetic dataset generation strategies (e.g., *PGD*, *CW*, *FA* and *FF*) described in Section V and summarized in Table IV. We conduct experiments with five different victims including traffic, flower, NSFW and emotion recognition models, and find the similar conclusion. Taking the traffic and flower recognition models as examples, we randomly pick natural samples from a given data set and calculate the detection threshold value δ resulting in no false positives for the substitute models ($\delta = 0.91$ for the traffic recognition model and $\delta = 0.80$ for the flower recognition model). From Table IV we can see that our attacks can easily evade their defense by carefully selecting the parameters M from $0.1D$ to $0.8D$. This is because, by selecting the parameter M in Equation (10), we can simulate a normal distribution of query samples without significantly degrading the substitute models performance. By increasing δ (e.g, from $\delta = 0.87$ to $\delta = 0.94$ for flower adversarial examples), the PRADA produces a high false positives (up to 93%) while detecting our FF attack. Moreover, results demonstrate that other types of adversarial attacks such as *PGD*, *CW* and *FA* can also bypass the PRADA defense if δ is small.

Model (δ value)	Queries made until detection					
	PGD	**CW**	**FA**	**FF**		
				M=0.8D	M=0.5D	M=0.1D
Traffic (δ=0.92)	missed	missed	missed	missed	150	130
Traffic (δ=0.97)	110	110	110	110	110	110
Flower (δ=0.87)	110	missed	220	missed	290	140
Flower (δ=0.90)	110	340	220	350	120	130
Flower (δ=0.94)	110	340	220	350	120	130

TABLE IV: Adversarial queries made until detection. Here, the parameter D is the L_2 norm distance measuring the similarity between the legitimate example x_s and its adversarial example x'_s in the feature space.

2) The Proposed Defense: In order to reduce the impact of information leakage during the querying process, as demonstrated by our attack, we design and evaluate potential defense mechanisms. Unlike previous works, our goal is not to analyze the distribution of consecutive queries but rather focus our attention on the method for detecting adversarial examples, which can help providers offer MLaaS to monitor whether individual query inputs are malicious. In our defense scheme, we analyze the differences between the feature distributions of malicious and benign images, and further propose a novel defense mechanism that can effectively and adaptively defend against the *FeatureFool* adversarial attacks on MLaaS platforms. Specifically, we start with training the proposed feature distribution guided network, named *DefenseNet*, using a popular deep learning framework - PyTorch. The pre-trained *DefenseNet* will be adapted to extract each hidden layers output as the features of the input samples. A categorical mixture model is used as the prior probability to characterize these query samples distribution. Adversarial examples generated by attackers have a different characteristic distribution from the benign samples distribution. We also integrate an SVM classifier into our *DefenseNet* to distinguish benign samples and adversarial examples as opposed to prior works which may alter the decision boundaries. Our defense mechanism dramatically improves the average success rate (up to 91%) of detecting abnormalities in the input samples used for querying API.

B. Limitations

Though the experimental results show that the proposed attack framework is able to effectively steal the victim models inside the commercial APIs even in a black-box setting, there are some limitations that we may address in the future.

Further improvement of adversarial query methods. One main limitation is that, in order to maximize the uncertainty of examples away from decision boundary of the victim classifier, attackers may add more perturbations to these examples via adversarial attacks algorithms. For example, *Feature Adversary* and *FeatureFool* may be used to generate stronger datasets to train the substitute model. However, in this case, adversarial examples with large perturbations generally pollute the synthetic training set and consequently lower the accuracy of the substitute model. This problem can be addressed in the future by designing a more sophisticated algorithm that can trade-off between the perturbations of adversarial examples and the performance of the substitute model.

Extension to multi-label cases. Another problem is extending the attack method to multi-label cases. In comparison to classification tasks with a single label per image, where attackers aim at replicating the functionality of the multi-label model inside the API, we need to pay more attention to the synthetic data set generation algorithms and the substitute model architecture choices. That is, in order to launch a model stealing attack on the victim model, attackers need to first craft the adversarial examples with multi target labels and then generate the synthetic dataset to train a substitute model. Although the proposed attack framework is not evaluated on multi-label classification models, for example, Celebrity Recognition API [19] provided by Clarifai, the adversarial query method introduced in this paper can help an adversary obtain more crucial information about a victim model, such as decision boundary, label types, etc, which pose a great threat to the privacy of MLaaS platforms. Future work will also focus on developing an effective model extraction attack on the cloud-based multi-label classification model.

Extension to other domains. As adversarial examples are widely existed on various domains such as audio and text, the proposed attack can be easily extended to all DNN based MLaaS platforms. Furthermore, even in the case where appropriate pre-trained models may be harder to get from current model zoos, attackers can pre-train their basic "teacher" models from scratch (i.e., datasets related to target tasks) and then fine-tune these models using the adversary-query pairs proposed in this paper on the domains other images to steal black-box DNN models.

C. Responsible Disclosure

We have reported our findings to cloud providers including Clarifai, Microsoft, IBM, Google, and Face++. Among them, we contacted Clarifai and Face++ in December 2019 and contacted Microsoft, IBM, and Google in January 2020. Among them, Face++ replied to us in January 2020 and encourage us to apply the developed method on other Face++ APIs for security evaluations.

VII. Conclusions

Machine learning as a service (MLaaS) provided by cloud-based platforms, including Microsoft, Google, Face++ and Clarifai, has been widely applied in real-world applications. These services, however, tend to suffer from the model extraction attack launched by an adversary even with black-box access. Although previous works on model stealing attacks have shown good performance, their effectiveness is generally constrained by massive prediction queries and high costs. To address these challenges, we study the practicality of model stealing attacks against DNN models trained on commercial MLaaS platforms. Through local experiments and online attacks on commercialized MLaaS platforms we demonstrate that our model stealing attack can sufficiently train a local substitute model with near-equivalent performance as the target model. Our attack method requires significantly less queries to the target model compared to previous works of model extraction attack due to our novel design of architecture and training process of the local substitute model. The transfer learning helps us to utilize existing well-trained models in the source domain, and thus we only need to fine-tune a few layers of these models. The adversarial examples used for querying the target model help us to efficiently learn the distance between decision boundaries of the target model and the local model, thus accelerating the convergence in training. In the future, we will mainly focus on designing effective defense mechanisms against model stealing attacks, and therefore enhance the robustness of DNN based MLaaS image classifiers.

Acknowledgment

This work is partially supported by the Department of Energy through the Early Career Award. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Department of Energy.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[2] A. rahman Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 14–22, 2012.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. R. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[4] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[5] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[6] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 601–618, 2016.

[7] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," *arXiv preprint arXiv:1802.05351*, 2018.

[8] A. Dmitrenko *et al.*, "Dnn model extraction attacks using prediction interfaces," 2018.

[9] Y. Shi, Y. Sagduyu, and A. Grushin, "How to steal a machine learning classifier with deep learning," in *Technologies for Homeland Security (HST), 2017 IEEE International Symposium on*, pp. 1–5, IEEE, 2017.

[10] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 3–18, IEEE, 2017.

[11] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[12] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, "Prada: Protecting against dnn model stealing attacks," *arXiv preprint arXiv:1805.02628*, 2018.

[13] T. Zhang, "Privacy-preserving machine learning through data obfuscation," *arXiv preprint arXiv:1807.01860*, 2018.

[14] J. R. C. da Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data," *CoRR*, vol. abs/1806.05476, 2018.

[15] "Microsoft custom vision." https://azure.microsoft.com/en-us/services/cognitive-services/custom-vision-service/.

[16] "Face++ emotion recognition api." https://www.faceplusplus.com/emotion-recognition/.

[17] "Ibm watson visual recognition." https://www.ibm.com/cloud/watson-visual-recognition.

[18] "Google automl vision." https://cloud.google.com/vision/automl/docs/.

[19] "clarifai safe for work (nsfw)." https://www.clarifai.com/models/celebrity-image-recognition-model-e466caa0619f444ab97497640cefc4dc.

[20] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, "PRADA: protecting against DNN model stealing attacks," *CoRR*, vol. abs/1805.02628, 2018.

[21] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *AsiaCCS*, 2017.

[22] S. J. Pan and Q. Yang., "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[23] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pp. 36–45, 2015.

[24] W. Ge and Y. Yu, "Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[25] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[26] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, pp. 601–608, 2006.

[27] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4068–4076, 2015.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[29] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases* (H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, eds.), pp. 387–402, Springer Berlin Heidelberg, 2013.

[30] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *CoRR*, vol. abs/1511.07528, 2015.

[31] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1528–1540, ACM, 2016.

[32] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," vol. abs/1605.07277, 2016.

[33] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *CoRR*, vol. abs/1611.02770, 2016.

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *arXiv preprint arXiv:1312.6199*, 2013.

[35] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, (Saarbrucken), pp. 372–387, IEEE, 2016.

[36] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (S&P), 2017 IEEE Symposium on*, pp. 39–57, 2017.

[37] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR*, 2018.

[38] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[39] T. S. Sethi and M. Kantardzic, "Data driven exploratory attacks on black box classifiers in adversarial domains," *Neurocomput.*, vol. 289, no. C, pp. 129–143, 2018.

[40] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pp. 349–363, 2018.

[41] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *CoRR*, vol. abs/1511.04599, 2015.

[42] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[43] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," *CoRR*, vol. abs/1710.11342, 2017.

[44] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[45] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[46] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, "With great training comes great vulnerability: Practical attacks against transfer learning," in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1281–1297, 2018.

[47] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 36–52, 2018.

[48] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Advances in Neural Information Processing Systems 20* (J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, eds.), pp. 1289–1296, 2008.

[49] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active learning with statistical models,” *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 129–145, 1996.

[50] B. Settles and M. Craven, “An analysis of active learning strategies for sequence labeling tasks,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP ’08, pp. 1070–1079, 2008.

[51] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, 2002.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

[53] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.

[54] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2015.

[55] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.

[56] “The german traffic sign recognition benchmark.” http://benchmark.ini.rub.de/?section=gtsrb\&subsection=news.

[57] “102 category flower dataset.” http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html.

[58] “Kdef: A resource for studying face recognition in personal photo collections.” http://kdef.se/home/aboutKDEF.html.

[59] “Pubfig83: A resource for studying face recognition in personal photo collections.” http://vision.seas.harvard.edu/pubfig83/.