ACTIVETHIEF: Model Extraction Using Active Learning and Unannotated Public Data

Soham Pal,[1*] **Yash Gupta,**[1*] **Aditya Shukla,**[1*] **Aditya Kanade,**[1,2†]
Shirish Shevade,[1†] **Vinod Ganapathy**[1†]
[1]Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
[2]Google Brain, USA
{sohampal,yashgupta,adityashukla,kanade,shirish,vg}@iisc.ac.in

Abstract

Machine learning models are increasingly being deployed in practice. Machine Learning as a Service (MLaaS) providers expose such models to queries by third-party developers through application programming interfaces (APIs). Prior work has developed model extraction attacks, in which an attacker extracts an approximation of an MLaaS model by making black-box queries to it. We design ACTIVETHIEF – a model extraction framework for deep neural networks that makes use of active learning techniques and unannotated public datasets to perform model extraction. It does not expect strong domain knowledge or access to annotated data on the part of the attacker. We demonstrate that (1) it is possible to use ACTIVETHIEF to extract deep classifiers trained on a variety of datasets from image and text domains, while querying the model with as few as 10-30% of samples from public datasets, (2) the resulting model exhibits a higher transferability success rate of adversarial examples than prior work, and (3) the attack evades detection by the state-of-the-art model extraction detection method, PRADA.

Introduction

In recent years, machine learning (ML) models are being increasingly deployed in production software. Deep neural networks (DNNs) are a particularly successful and popular class of ML models. However, training DNNs is an expensive activity requiring access to data, compute and human expertise. Many companies provide paid access to models that are trained and hosted by them on the cloud, commonly referred to as Machine Learning as a Service (MLaaS) platforms. Third-party developers access these models through Application Programming Interfaces (APIs).

Even though this setup has the potential to democratize access to machine learning, it has been shown to be vulnerable to model extraction attacks (Tramèr et al. 2016). An attacker extracts a model by training a **substitute model** on labeled data obtained by repeatedly querying the service provider's **secret model**. The attacker can then freely use the substitute model, offer it as a competing service, or use it to generate adversarial examples against production software that depends on the provider's model.

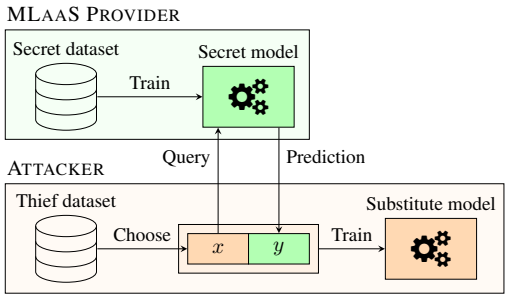

Figure 1: Overview of model extraction

Since the input-output format of the API is public, it is fair to assume that the attacker knows how to present data to the secret model and how to interpret its output. However, the attacker cannot access the **secret dataset** on which the secret model is trained (see Figure 1). In lieu of this, the attacker must obtain some alternative dataset, which we refer to as a **thief dataset**, using which the secret model can be queried.

The problem of obtaining thief datasets remains a challenge in making model extraction attacks more practical. Prior work has considered access to limited Problem Domain (**PD**) data (Papernot et al. 2017), which is drawn from a distribution that closely resembles the secret dataset, e.g., medical image data to extract models trained on medical images. However, getting such data even in small quantities could be difficult and expensive. Alternatively, researchers have used Synthetic Non-Problem Domain (**SNPD**) (Tramèr et al. 2016) and Natural Non-Problem Domain (**NNPD**) data (Correia-Silva et al. 2018; Orekondy, Schiele, and Fritz 2019). SNPD is sampled from standard probability distributions (such as uniform distributions) that do not necessarily model the problem domain distributions and hence, do not aid the attacker much. NNPD, on the other hand, is sampled from publicly available data of the same type of content as the secret model's input, e.g., image data for image models and text data for text models. NNPD has been shown to be more effective than SNPD in model extraction since it provides more natural samples to query the secret model with.

[*]All three authors contributed equally.
[†]All three authors contributed equally.

Another challenge for the attacker is to query the secret model as efficiently as possible with a limited **query budget**. This limitation arises from the cost of querying the secret model or due to rate-limited API access. Orekondy, Schiele, and Fritz (2019) exploit the hierarchical annotations of ImageNet (Russakovsky et al. 2015) data to fit within a query budget. However, the requirement of getting high-quality annotations limits the attacker. In this work, we show that model extraction can be accomplished using NNPD data. However, unlike Orekondy, Schiele, and Fritz (2019), our approach does *not* use annotated[1] data, thereby, overcoming a key limitation to the attacker. Instead of using the annotations to sample data points for efficiently querying the secret model, we use and combine pool-based active learning strategies. We call our approach ACTIVETHIEF.

Yet another challenge is limited knowledge about the architecture of the secret model. We show that ACTIVETHIEF can successfully extract models even when the choices differ between the secret and substitute models, e.g., RNN vs CNN. The success of model extraction can be measured in terms of the *agreement* of the substitute model with the secret model on the test data. Another measure is the *transferability* of adversarial examples crafted on the substitute model to the secret model. Our experiments show that ACTIVETHIEF achieves substantial agreement of the extracted substitute model with the secret model. Further, the adversarial examples crafted on the substitute models have higher transferability compared to a state-of-the-art adversarial attack based on model extraction (Papernot et al. 2017).

Considering the severity of model extraction attacks, many approaches to defend against them are proposed in the literature. They range from restricting the access to top-1 label instead of output class probabilities (Tramèr et al. 2016) or perturbing the output probabilities (Lee et al. 2018). We show that while ACTIVETHIEF can benefit from access to (unperturbed) class probabilities, it does very well even using only the top-1 label. Models can also be defended by observing the distribution of the queries made to the secret model by a third-party and detecting whether an attack is being launched. PRADA (Juuti et al. 2019) is a recent such technique which has been shown to detect and prevent model extraction attacks by checking normality of the distribution of distances between successive queries. Our experiments show that our use of natural thief data (NNPD) helps us defeat PRADA.

In summary, we make the following contributions:

- We present ACTIVETHIEF, a novel model extraction framework that exploits availability of *unannotated* public data to improve the attacker's ability to launch successful attacks. We show that using a *single* thief dataset, it is possible to extract multiple, separate deep classifiers. This is shown for both image and text domains. We refer to these datasets as **universal thief datasets**, since they do not depend on the specifics of the secret models (or datasets) and are successful across multiple secret models.
- We show that the use of active learning makes ACTIVETHIEF query-efficient. By using only 10-30% of the available data, it achieves 61.52-98.18% agreement with the secret models across image and text classification tasks. Note that unlike the usual active learning setup, the oracle here is itself another model, trained on secret data. The extracted models also enable better transferability of adversarial examples crafted on them compared to prior work.
- Finally, we show that since ACTIVETHIEF queries follow natural distributions, its attack cannot be detected by a state-of-the-art detection method, PRADA, that monitors the distribution of distances between queries for deviation from normal distributions.

The source code for ACTIVETHIEF will be made available at http://iisc-seal.net/ under an open source license.

[1]We use *label* to refer to the output of secret models, *approximate label* for the output of substitute models, and *annotations* for categorical information about samples available from curated datasets such as ImageNet (Russakovsky et al. 2015).

The ACTIVETHIEF framework

We start with a description of the ACTIVETHIEF framework with reference to Figure 2.

1. The attacker picks a random subset of initial seed samples S_0 of the training fold of the thief dataset.
2. In the i^{th} iteration ($i = 0, 1, 2, \ldots, N$), the attacker queries the samples in S_i against the secret model f and obtains the labeled[1] set $D_i = \{(x, f(x)) : x \in S_i\}$.
3. The substitute model $\tilde{f}$ is trained on $\bigcup_{t=0}^{i} D_t$.
4. The attacker queries the remaining samples against the substitute model $\tilde{f}$ to assign approximate labels[1] to them:
$$\tilde{D}_i = \{(x, \tilde{f}(x)) : x \notin S_1 \cup \cdots \cup S_i\}$$
Note that the substitute model predictions $\tilde{y} = \tilde{f}(x)$ are always obtained as full softmax probability vectors.
5. An active learning subset selection strategy is used to select the set of k samples S_{i+1} to be queried next, such that $x \in S_{i+1}$ only if $(x, \tilde{y}) \in \tilde{D}_i$.

This process is repeated for a fixed number of iterations, retraining the substitute model $\tilde{f}$ from scratch in each iteration. The number of samples to label in each iteration k, the number of iterations N, and the number of initial seed samples $|S_0|$ are hyperparameters.

Evaluation metric

The metric used for evaluation of the closeness between the secret model f and the substitute model $\tilde{f}$ is the **agreement** between them, evaluated on the test split of the secret dataset:

$$\text{Agreement}(f, \tilde{f}) = \frac{1}{|X^{\text{test}}_{\text{secret}}|} \sum_{x \in X^{\text{test}}_{\text{secret}}} \mathbf{I}\big(f(x) = \tilde{f}(x)\big)$$

where $\mathbf{I}(\cdot)$ is the indicator function. Note that we use the secret test data only for evaluation, and it is not made available to ACTIVETHIEF during the model extraction process. The secret model is queried with, and the substitute model is trained on samples only from the NNPD thief dataset.

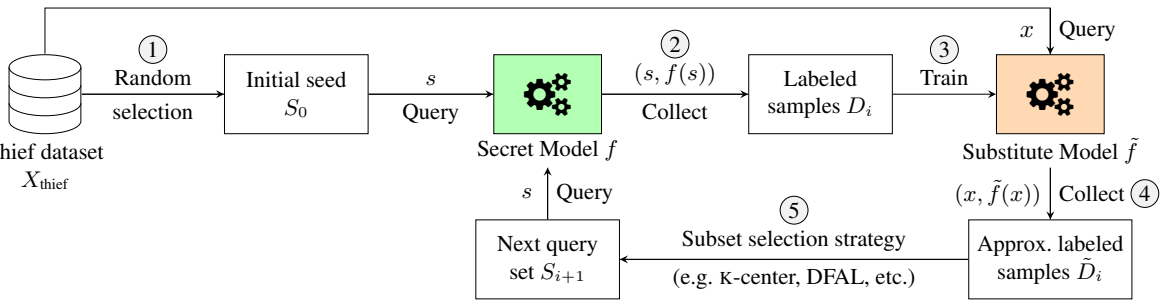

Figure 2: The ACTIVETHIEF framework for model extraction (see corresponding section for explanation of 1-5).

Active learning subset selection strategies

In each iteration, the attacker selects a new set S_i of k thief dataset samples to label by querying the secret model f. Each subset selection strategy takes as input the approximately labeled set $\tilde{D}_i = \{(x_n, \tilde{y}_n)\}$, and returns a set S_{i+1}:

Random strategy: A subset of size k of samples x_n is selected uniformly at random.

Uncertainty strategy: This method is based on uncertainty sampling (Lewis and Gale 1994). The entropy $\mathcal{H}_n = -\sum_j \tilde{y}_{n,j} \log \tilde{y}_{n,j}$ (where j is the label index) of predicted probability vectors $\tilde{y}_n$ is computed. The k samples x_n corresponding to the highest entropy values $\mathcal{H}_n$ (i.e. those that the model is least certain about) are selected.

K-center strategy: We use the greedy K-center algorithm of Sener and Savarese (2018). The predicted probability vectors $\tilde{y}_m$ are clustered as follows: The probabilities for the initial seed samples are marked as cluster centers. In each subsequent iteration, the strategy selects k samples x_n corresponding to the most distant $\tilde{y}_n$ from all existing centers:

$$(x_0^*, \tilde{y}_0^*) = \arg \max_{(x_n, \tilde{y}_n) \in \tilde{D}_i} \min_{(x_m, y_m) \in D_{i-1}} \|\tilde{y}_n - \tilde{f}(x_m)\|_2^2$$

The selected x_0^* is then labeled (the resulting pair is subsequently treated as a center, i.e. a member of D_{i-1}). This process is repeated until k samples $x_0^*, x_1^*, \ldots x_k^*$ are selected.

DFAL strategy: We use the DeepFool-based Active Learning (DFAL) algorithm of Ducoffe and Precioso (2018). DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016) is applied to every sample x_n to obtain a perturbed $\hat{x}_n$ that gets misclassified by the substitute model (i.e. $\tilde{f}(x_n) \neq \tilde{f}(\hat{x}_n)$). The perturbation $\alpha_n = \|x_n - \hat{x}_n\|_2^2$ is computed, and the k samples x_n with the lowest perturbation α_n are selected. Note that the secret model is never queried with the perturbed $\hat{x}_n$, but only their respective clean counterparts, x_n.

DFAL + K-center strategy: While the K-center strategy maximizes diversity, it does not ensure that each individual sample is informative. On the contrary, while the DFAL strategy ensures that each individual sample is informative (i.e. close to the decision boundary), it does nothing to eliminate redundancy. Inspired by this observation, we introduced this combined strategy. In this strategy, the DFAL strategy is first used to pick an initial subset of ρ informative samples (we choose ρ = the total budget). Of these, k points are selected, eliminating redundancy by the K-center strategy.

Experimental setup

Datasets

Secret datasets. For image classification, we use the following datasets: MNIST (LeCun et al. 1998), CIFAR-10 (Krizhevsky and Hinton 2009) and GTSRB (Stallkamp et al. 2012). For text classification, we use MR (Pang and Lee 2005), IMDB (Maas et al. 2011), and AG News[2]. Further details are presented in the supplement[3].

Thief dataset. An attacker could obtain a thief dataset by crawling the public internet for images and text. Here, we use a downsampled and unannotated subset of the training fold of the ILSVRC2012-14 dataset (Chrabaszcz, Loshchilov, and Hutter 2017) as a proxy for public image data. In our experiments, we also explored the use of the less diverse CIFAR-10 dataset, but we found agreement of the models to be consistently worse (we omit these results). Our training and validation splits are of size 100K and 20K respectively. For text, we use WikiText-2 (Merity et al. 2017). There are 80K training and 9K validation samples.

Model architectures

Image classification. The input is followed by l convolution blocks. Each convolution block consists of 2 repeated units of 2 convolution layers (3×3 kernel with stride 1) and 1 pooling layer (2×2 kernel with stride 2). Each convolution is followed by ReLU and batchnorm layers, and pooling by dropout. Convolution layers use $32, 64, \ldots 32 \times 2^{l-1}$ filters respectively. The output of the final pooling layer is passed through fully connected (FC) and softmax layers to obtain the vector of output probabilities. We use a default of l = 3.

Text classification. Word2vec (Mikolov et al. 2013) is first used to obtain the word embeddings (pretrained embeddings are used for the secret model, and are learned from scratch in the substitute model). We consider two architectures:

1. CNN: We use the CNN of Kim (2014) for sentence classification, with 100 filters each of width 3, 4 and 5, followed by a max-pool over time.

2. RNN: We use a single-layer GRU with 64 hidden units, operating in acceptor configuration. The final hidden state is passed through a FC layer (size 32, with a ReLU activation).

The outputs in either case are passed through FC and softmax layers to obtain the vector of output probabilities.

[2] https://di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[3] http://iisc-seal.net/publications/aaai2020_supplement.pdf

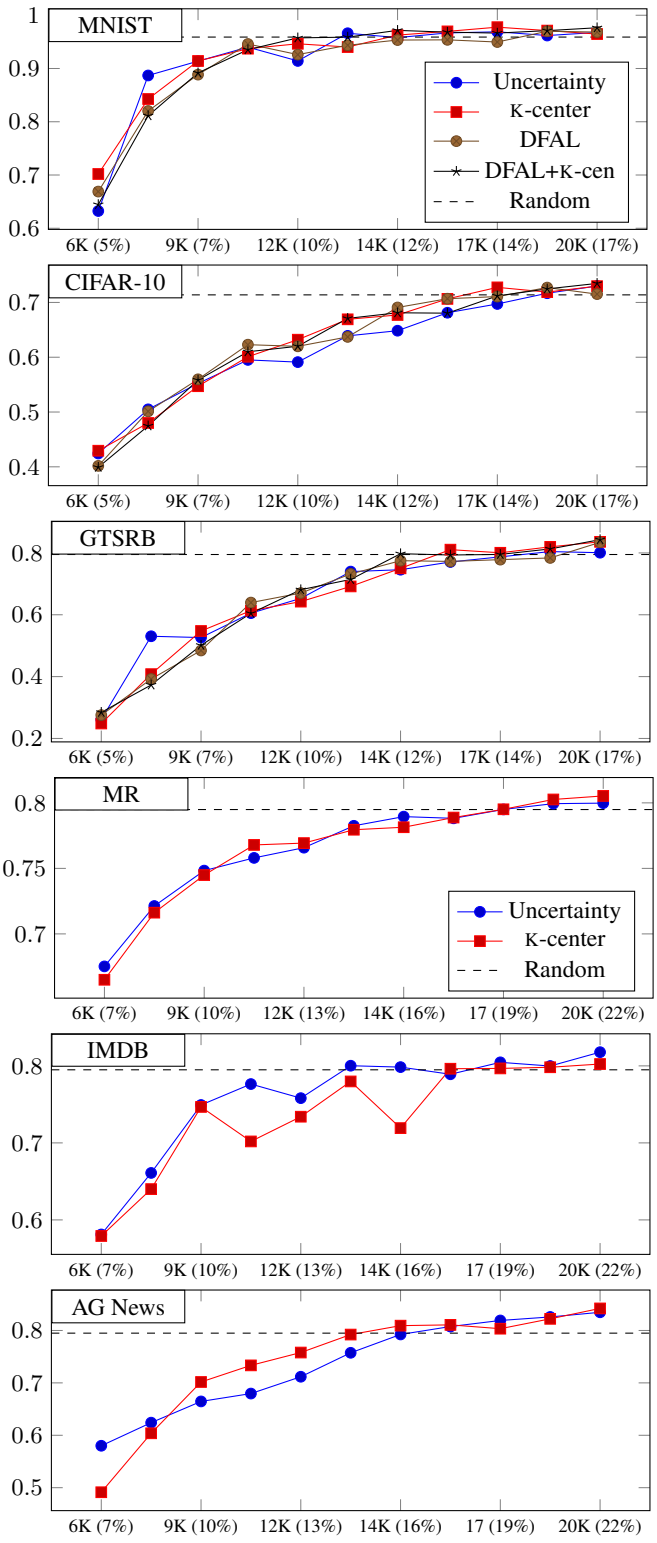

Figure 3: The improvement in agreement (%) over 10 iterations on the secret test set for image and text classification tasks, with a total budget of 20K. The X-axis indicates budget consumed (% of total dataset indicated in parenthesis). Since random is not run iteratively, it is indicated as a line parallel to the X-axis.

MNIST	10K (8%)	15K (12%)	20K (17%)	25K (21%)	30K (25%)
Random	91.64	95.19	95.90	97.48	97.36
Uncertainty	94.64	97.43	96.77	97.29	97.38
K-center	**95.80**	95.66	96.47	**97.81**	97.95
DFAL	95.75	95.59	96.84	97.74	97.80
DFAL+K-center	95.40	**97.64**	**97.65**	97.60	**98.18**
Using the full thief dataset (120K):					98.54
Using uniform noise samples (100K):					20.56

CIFAR-10	10K (8%)	15K (12%)	20K (17%)	25K (21%)	30K (25%)
Random	63.75	68.93	71.38	75.33	76.82
Uncertainty	63.36	69.45	72.99	74.22	76.75
K-center	**64.20**	70.95	72.97	74.71	78.26
DFAL	62.49	68.37	71.52	**77.41**	77.00
DFAL+K-center	61.52	**71.14**	**73.47**	74.23	**78.36**
Using the full thief dataset (120K):					84.99
Using uniform noise samples (100K):					10.62

GTSRB	10K (8%)	15K (12%)	20K (17%)	25K (21%)	30K (25%)
Random	67.72	77.71	79.49	82.14	83.84
Uncertainty	67.30	73.92	80.07	83.61	85.49
K-center	70.89	**81.03**	83.59	**85.81**	85.93
DFAL	**72.71**	79.44	83.43	84.41	83.98
DFAL+K-center	70.79	79.55	**84.29**	85.41	**86.71**
Using the full thief dataset (120K):					93.68
Using uniform noise samples (100K):					45.53

MR	10K (11%)	15K (17%)	20K (22%)	25K (28%)	30K (33%)
Random	76.45	78.24	79.46	81.33	82.36
Uncertainty	**77.19**	80.39	81.24	**84.15**	83.49
K-center	77.12	**81.24**	**81.96**	83.95	**83.96**
Using the full thief dataset (89K):					86.21
Using discrete uniform noise samples (100K):					75.79

IMDB	10K (11%)	15K (17%)	20K (22%)	25K (28%)	30K (33%)
Random	71.67	78.79	74.70	80.71	79.23
Uncertainty	73.48	78.12	**81.78**	**82.10**	82.17
K-center	**77.67**	**78.96**	80.24	81.58	**82.90**
Using the full thief dataset (89K):					86.38
Using discrete uniform noise samples (100K):					53.23

AG News	10K (11%)	15K (17%)	20K (22%)	25K (28%)	30K (33%)
Random	74.51	80.39	82.76	83.97	84.20
Uncertainty	75.47	**82.08**	83.47	84.96	**87.04**
K-center	**75.87**	79.63	**84.21**	**84.97**	85.96
Using the full thief dataset (89K):					90.07
Using discrete uniform noise samples (100K):					35.50

Table 2: The agreement (%) on the secret test set for image and text classification tasks. Each row corresponds to a subset selection strategy, while each column corresponds to a different query budget (% of total dataset indicated in parenthesis).

Training regime

We use the Adam optimizer (Kingma and Ba 2015) with default hyperparameters. In our experiments, for all but the random strategy, training is done iteratively. In each iteration, the model is trained for at most 1,000 epochs with a batch size of 150 (images) or 50 (text). Early stopping is used with a patience of 100 epochs (images) or 20 epochs (text). An L_2 regularizer is applied at a rate of 0.001, and dropout is applied at a rate of 0.1 for all datasets other than CIFAR-10, where a dropout of 0.2 is used. At the end of each epoch, the model is evaluated and the F_1 measure on the validation split is recorded. The model with the best validation F_1 measure is selected as the substitute model $\tilde{f}$ in that iteration. We set aside 20% of the query budget for validation, and use 10% as the initial seed samples.

Experimental results

In this section, we study two attack scenarios, differentiated by the attacker's knowledge of the secret model architecture, and also show how ACTIVETHIEF is able to evade PRADA.

Full knowledge of the secret model architecture

We first evaluate ACTIVETHIEF in a scenario where the attacker has full knowledge of the secret model architecture. We do this to study subset selection strategies in isolation, eliminating the influence of the substitute model architecture. For each dataset, we run ACTIVETHIEF across the following total query budgets: 10K, 15K, 20K, 25K and 30K. For 20K, we show the agreement at the end of each iteration for every strategy and each dataset in Figure 3. Clearly, the agreement of the model improves, on an average, with each iteration. As it is non-trivial to modify DeepFool to work for text classification tasks, we omit the two strategies that make use of it for evaluation in the context of the text datasets. These experiments are run assuming that only the top-1 prediction is returned by the MLaaS API (i.e. no probability scores are returned). We tabulate the agreement obtained at the end of the final iteration for each experiment in Table 2.

Effectiveness of active learning. The benefits of careful selection of thief dataset samples can be clearly seen: there is no dataset for which the random strategy performs better than all of the other strategies. In particular, K-center underperforms only once (for images) and once (for text), while DFAL underperforms twice (for images). Uncertainty underperforms 6 times (for images) and once (for text), but this is in line with the findings of Ducoffe and Precioso (2018). Running the random strategy 4 more times, we observe a standard deviation of 0.60, 0.56 and 1.12 for MNIST, CIFAR-10 and GTSRB. Encouraged by the performance of DFAL and K-center, we study the combination thereof.

Effectiveness of the combined strategy. The agreement of the models is improved by the combined strategy over the basic DFAL strategy in 10 out of 15 of the image classification experiments, and the combined strategy emerges as the winner in 8 experiments – a majority. This improvement in agreement bears evidence to the increased potential of the combined strategy in extracting information from the secret model. The other competitive method is the K-center method, which wins in 5 experiments. This is followed by the DFAL strategy, which won in 2 experiments.

Overall, we find that the K-center and DFAL+K-center strategies perform the best for image classification, while the K-center strategy performs the best for test classification.

Impact of the query budget. As evident from Table 2, there is an improvement in agreement when increasing the query budget. The attacker should thus make as many queries as possible. However, we observe that ACTIVETHIEF is able to achieve an agreement comparable to that of the full thief dataset, while using only 30% of it.

Effectiveness of universal thief datasets. We compare our results to *uniform noise* (multidimensional $U[0, 1]$) SNPD data, analogous to data used in the equation-solving attacks of Tramèr et al. (2016). For text classification, we use discrete uniform. It can be seen that the uniform baseline achieves a low agreement on all datasets. When queried with uniform noise, there are many labels which the secret model rarely predicts, and the substitute model fails to learn to properly identify such labels (further details in the supplement). This is overcome by the use of universal thief datasets, leading to a 3.4x improvement in agreement over uniform noise on an average.

Impact of the number of iterations. Table 3 shows that with an increase in the number of iterations, there is an improvement in agreement for the same budget. Thus, the substitute model agreement can be improved by running the ACTIVETHIEF framework for more iterations at the expense of increased training time, but with diminishing returns.

Impact of access to output probability scores. Table 4 demonstrates that access to the output probabilities of the secret model results in an improvement in agreement. We believe that this is because the substitute model receives a signal corresponding to every output neuron for each thief dataset sample that it is trained on. Consequently, the substitute model is able to learn a better approximation.

Table 3: Agreement of the winning strategy on the secret test set for each dataset (total budget of 10K), when using a different number of iterations.

	Substitute model agreement (%)	
Dataset	10 iterations	20 iterations
MNIST	95.80	**96.74**
CIFAR-10	64.20	**64.23**
GTSRB	72.71	**72.78**

Table 4: Agreement of the winning strategy on the secret test set for each dataset (total budget of 10K over 10 iterations), with and without access to output probability scores.

	Substitute model agreement (%)	
Dataset	Top-1 prediction	Probability scores
MNIST	95.80	**98.61**
CIFAR-10	64.20	**77.29**
GTSRB	72.71	**86.90**

Table 5: Transferability (%) of FGSM adversarial examples.

		ACTIVETHIEF			
Dataset	Papernot	Random	K-center	DFAL	DFAL+K-center
MNIST	40.28	49.77	47.75	**59.55**	53.08
CIFAR-10	82.61	**85.76**	85.26	84.25	84.30
GTSRB	84.83	93.41	93.45	**93.83**	93.34

Transferability of adversarial examples. Szegedy et al. (2013) introduce the concept of adversarial examples, where an imperceptible perturbation is introduced to input images that cause ML models to misclassify them. They show that these adversarial examples crafted for a particular model are *transferable*, i.e. they are also likely to be misclassified by other models. Papernot et al. (2017) present a technique for generating adversarial examples for black-box models, by first performing model extraction, and then using the substitute model to generate adversarial examples that transfer on to the secret model. Here, we compare the transferability of adversarial examples obtained using the ACTIVETHIEF substitute model, to that of Papernot et al. Details of how we train the substitute model of (Papernot et al. 2017) are presented in the supplement. We generate adversarial examples using the FGSM attack (Goodfellow, Shlens, and Szegedy 2014), at a rate of $\epsilon = 0.25$. We compute transferability as the fraction of perturbed secret dataset samples (which are PD data) that are misclassified by the secret model. We present our results in Table 5. It can be seen that while using only unannotated NNPD data for extraction, ACTIVETHIEF is able to achieve better transferability of perturbed PD samples than Papernot et al., which requires PD data during the extraction process.

Limited knowledge of the secret model architecture

Here, we perform extraction using the full thief dataset, to study the impact of substitute model selection.

Case study on text classification. We first consider the text classification scenario, with a CNN secret model architecture. With no prior information about the model, an attacker may guess the secret model architecture to be an RNN, given the prelevance of such models in natural language processing. Both this situation, as well as the converse, are illustrated in Table 6, where the architectures CNN and RNN are as described in the experimental setup. As it can be seen, the agreement achieved in each case, across all three datasets, is relatively robust to the attacker's choice of the substitute model architecture.

It is possible for an attacker to recover information about the model architecture using a related line of work, *model reverse-engineering* (Oh et al. 2018; Duddu et al. 2018), for instance, it may be possible to recover whether the model being used is an RNN or CNN, or the kind of activations used. We thus next consider a situation where the attacker has partial information about the model architecture:

Case study on CNNs for image classification. We now consider a situation where the attacker is privy to significant information about the secret model architecture, and study the impact of smaller discrepancies between the substitute and secret model architectures. We use different configurations of the CNN architecture for images introduced in the previous section, corresponding to three different values of l, viz., 2, 3 and 4. The results of these experiments is tabulated in Table 7. As is obvious from the table, the agreements along the principal diagonal (when the secret model and substitute model architectures are identical) are in general high. These results also corroborate the findings of (Juuti et al. 2019). We believe that the performance degradation from using a less or more complex substitute model than the secret model results from underfitting or overfitting, respectively. In any case, the agreement achieved is relatively robust to the choice of l for the substitute model.

Table 6: Agreement (%) for text classification tasks.

		Substitute model	
Dataset	Secret model	CNN	RNN
MR	CNN	**86.21**	85.18
	RNN	84.80	**89.12**
IMDB	CNN	**86.38**	85.53
	RNN	87.06	**90.22**
AG News	CNN	90.07	**92.62**
	RNN	90.96	**93.01**

Ability to evade a state-of-the-art detection method

Juuti et al. (2019) propose PRADA (**Pr**otecting **a**gainst **D**NN model stealing **a**ttacks), the most well known detection method for model extraction attacks. As PRADA is defined for image classification, we restrict our investigation to image classification tasks (MNIST, CIFAR-10 and GTSRB). Juuti et al. observe that the distribution of distances d_i:

$$d_i = \min_{j<i, y_j=y_i} \|x_i - x_j\|_2$$

between (benign) queries x_i (that are predicted to be in class y_i) closely fit a Gaussian distribution. Their detection method is based on their assumption of model extraction attacks having a *duplication phase*, where the attacker generates synthetic combinations or perturbations of benign samples, which causes the distribution of d_i to deviate from the Gaussian. To detect whether a potentially malicious client is attempting to extract an MLaaS model, they perform a *normality test* (the Shapiro-Wilk Test, refer to the supplement for more details). We run PRADA against our attack and that of Papernot et al. The histograms of d_i are plotted for each dataset in Figure 4. PRADA is able to stop the Papernot attacks, requiring 210, 380 and 710 queries for MNIST, CIFAR-10 and GTSRB respectively. It is unable to detect ACTIVETHIEF, even when using a budget of 30K, for the DFAL+K-center strategy (our results are similar for other strategies). For the Papernot attack, d_i is plotted only for samples up to the point of detection. It can be seen that the histogram for ACTIVETHIEF remains roughly normally distributed throughout, thus avoiding detection. This is as a result of ACTIVETHIEF querying the secret model with only natural NNPD dataset samples (and thus requiring no generation of synthetic samples during a duplication phase).

Table 7: The agreement (%) achieved for image classification tasks, using CNN architectures of differing complexity.

(a) MNIST dataset

Secret model	Substitute model l = 2	l = 3	l = 4
l = 2	**98.73**	98.15	97.63
l = 3	97.21	**98.81**	98.10
l = 4	96.75	98.05	**98.36**

(b) CIFAR-10 dataset

Secret model	Substitute model l = 2	l = 3	l = 4
l = 2	**78.34**	76.83	74.48
l = 3	80.66	81.57	**81.80**
l = 4	74.34	**79.17**	78.82

(c) GTSRB dataset

Secret model	Substitute model l = 2	l = 3	l = 4
l = 2	**95.02**	92.30	86.88
l = 3	90.08	**91.42**	91.28
l = 4	80.95	**86.50**	84.69

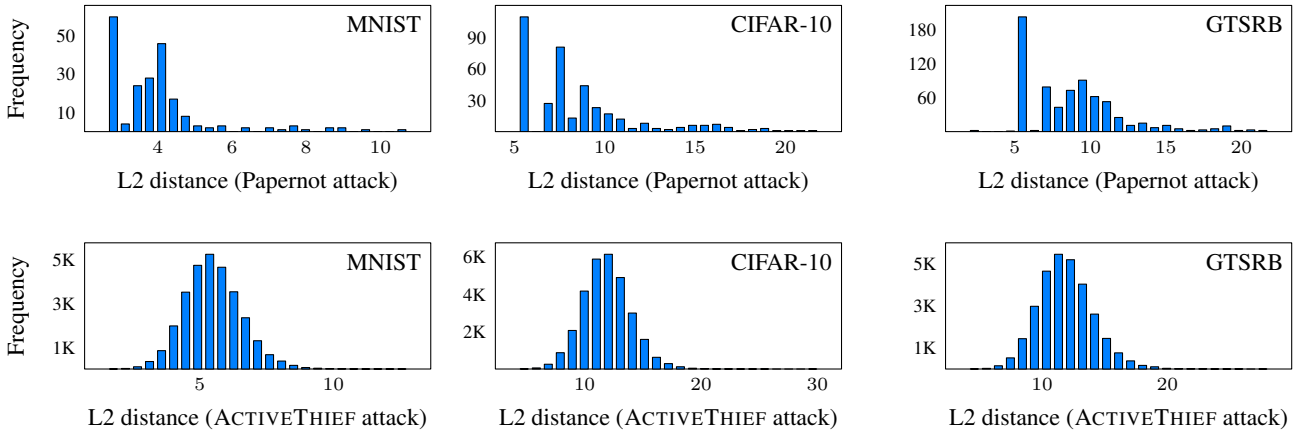

Figure 4: Distribution of distances for queries made by the attack of Papernot et al. (2017) and that of ACTIVETHIEF.

Related work

Model extraction is closely related to knowledge distillation (Hinton, Vinyals, and Dean 2015). The key difference is that in knowledge distillation, the attacker has full access to the secret dataset and secret model; no separate thief dataset is required. Apart from techniques described in previous sections, there are several other model extraction attacks and defenses in the literature. We discuss them briefly.

Attacks. Shi, Sagduyu, and Grushin (2017) show that DNNs can be used to extract the functionality of traditional ML models such as naïve Bayes and support vector machines, but not vice versa, using the test fold of the PD secret dataset. ACTIVETHIEF extracts DNNs using NNPD data. Sethi and Kantardzic (2018) present a framework for attackers attempting to bypass ML-based security mechanisms, e.g., CAPTCHA that uses click time to determine if users are benign. They use the extracted model to generate adversarial examples that allow attackers to bypass detection. Chandrasekaran et al. (2018) provide a theoretical treatment justifying the use of active learning in model extraction, and discuss the reduction in sample complexity for halfspace and decision tree-based learners. In contrast, we experimentally demonstrate active learning-based extraction with unannotated public data for more complex neural network models. Shi et al. (2018a) use active learning with PD data to extract a shallow feedforward network for text classification. Shi et al. (2018b) design an exploratory attack that uses a generative adversarial network trained on a small number of PD data samples, which is then used to generate synthetic samples with which the secret model is queried. Unlike ACTIVETHIEF, both of these approaches are reliant on PD data.

Defenses. Quiring, Arp, and Rieck (2018) show that when the secret model is a decision tree, defenses against model watermarking can also be used as defenses for model extraction attacks. Kesarwani et al. (2018) design a model extraction monitor that logs queries made to an MLaaS service. It uses total information gain and coverage of the input feature space to detect attacks. Both these defenses, however, assume the existence of a linear decision boundary in the secret model, and do not apply to the DNNs we extract. Lee et al. (2018) apply a perturbation to the predicted softmax probability scores to dissuade model extraction. Such a defense would still leave the secret model vulnerable to attacks that expect only the predicted label, such as ACTIVETHIEF.

Conclusion

In this paper, we introduce ACTIVETHIEF, a novel model extraction framework. We show that using only a single, unannotated public dataset, it is possible to extract models trained for classification tasks on different secret datasets. We also show that this is possible for both image and text domains with a limited query budget, with different architectures across secret and substitute models. ACTIVETHIEF is not detected by a state-of-the-art detection method for model extraction attacks. Models extracted using our method also possess good agreement with the secret models and improved transferability of adversarial examples.

Acknowledgments

We thank NVIDIA for computational resources; we thank Somesh Jha and the reviewers for their valuable inputs.

References

Chandrasekaran, V.; Chaudhuri, K.; Giacomelli, I.; Jha, S.; and Yan, S. 2018. Exploring connections between active learning and model extraction. *CoRR* abs/1811.02054.

Chrabaszcz, P.; Loshchilov, I.; and Hutter, F. 2017. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *CoRR* abs/1707.08819.

Correia-Silva, J. R.; Berriel, R. F.; Badue, C.; de Souza, A. F.; and Oliveira-Santos, T. 2018. Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Ducoffe, M., and Precioso, F. 2018. Adversarial active learning for deep networks: a margin based approach. *CoRR* abs/1802.09841.

Duddu, V.; Samanta, D.; Rao, D. V.; and Balas, V. E. 2018. Stealing neural networks via timing side channels. *CoRR* abs/1812.11720.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Juuti, M.; Szyller, S.; Dmitrenko, A.; Marchal, S.; and Asokan, N. 2019. PRADA: Protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*.

Kesarwani, M.; Mukhoty, B.; Arya, V.; and Mehta, S. 2018. Model extraction warning in MLaaS paradigm. In *Proceedings of the 34th Annual Computer Security Applications Conference*, 371–380. ACM.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lee, T.; Edwards, B.; Molloy, I.; and Su, D. 2018. Defending against model stealing attacks using deceptive perturbations. *CoRR* abs/1806.00054.

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3–12.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, 142–150.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer sentinel mixture models. In *ICLR*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. DeepFool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2574–2582.

Oh, S. J.; Augustin, M.; Fritz, M.; and Schiele, B. 2018. Towards reverse-engineering black-box neural networks. In *ICLR*.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019. Knockoff nets: Stealing functionality of black-box models. In *CVPR*.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

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

Quiring, E.; Arp, D.; and Rieck, K. 2018. Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 488–502.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Sener, O., and Savarese, S. 2018. Active learning for convolutional neural networks: A core-set approach. In *ICLR*.

Sethi, T. S., and Kantardzic, M. M. 2018. Data driven exploratory attacks on black box classifiers in adversarial domains. *Neurocomputing* 289:129–143.

Shi, Y.; Sagduyu, Y. E.; Davaslioglu, K.; and Li, J. H. 2018a. Active deep learning attacks under strict rate limitations for online API calls. In *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, 1–6.

Shi, Y.; Sagduyu, Y. E.; Davaslioglu, K.; and Li, J. H. 2018b. Generative adversarial networks for black-box API attacks with limited training data. In *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 453–458.

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

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32:323–332.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199.

Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M. K.; and Ristenpart, T. 2016. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*.