Knockoff Nets: Stealing Functionality of Black-Box Models

Tribhuvanesh Orekondy[1] Bernt Schiele[1] Mario Fritz[2]

[1] Max Planck Institute for Informatics [2] CISPA Helmholtz Center for Information Security
Saarland Informatics Campus, Germany

Abstract

Machine Learning (ML) models are increasingly deployed in the wild to perform a wide range of tasks. In this work, we ask to what extent can an adversary steal functionality of such "victim" models based solely on blackbox interactions: image in, predictions out. In contrast to prior work, we study complex victim blackbox models, and an adversary lacking knowledge of train/test data used by the model, its internals, and semantics over model outputs. We formulate model functionality stealing as a two-step approach: (i) querying a set of input images to the blackbox model to obtain predictions; and (ii) training a "knockoff" with queried image-prediction pairs. We make multiple remarkable observations: (a) querying random images from a different distribution than that of the blackbox training data results in a well-performing knockoff; (b) this is possible even when the knockoff is represented using a different architecture; and (c) our reinforcement learning approach additionally improves query sample efficiency in certain settings and provides performance gains. We validate model functionality stealing on a range of datasets and tasks, as well as show that a reasonable knockoff of an image analysis API could be created for as little as $30.

1. Introduction

Machine Learning (ML) models and especially deep neural networks are deployed to improve productivity or experience e.g., photo assistants in smartphones, image recognition APIs in cloud-based internet services, and for navigation and control in autonomous vehicles. Developing and engineering such models for commercial use is a product of intense time, money, and human effort – ranging from collecting a massive annotated dataset to tuning the right model for the task. The details of the dataset, exact model architecture, and hyperparameters are naturally kept confidential to protect the models' value. However, in order to be monetized or simply serve a purpose, they are deployed in various applications (e.g., home assistants) to function as blackboxes: input in, predictions out.

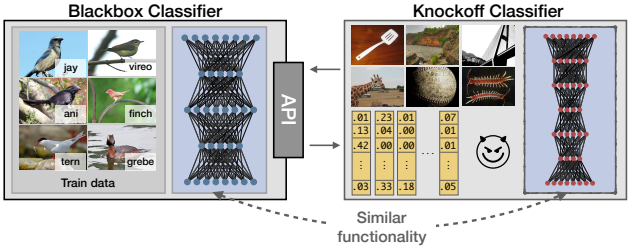


Figure 1: An adversary can create a "knockoff" of a blackbox model solely by interacting with its API: image in, prediction out. The knockoff bypasses the monetary costs and intellectual effort involved in creating the blackbox model.

Large-scale deployments of deep learning models in the wild has motivated the community to ask: can someone abuse the model solely based on blackbox access? There has been a series of "inference attacks" [9, 24, 29, 31] which try to infer properties (e.g., training data [31], architecture [24]) about the model within the blackbox. In this work, we focus on model functionality stealing: can one create a "knockoff" of the blackbox model solely based on observed input-output pairs? In contrast to prior work [19, 25, 35], we work towards purely stealing *functionality* of complex blackbox models by making fewer assumptions.

We formulate model functionality stealing as follows (shown in Figure 1). The adversary interacts with a blackbox "victim" CNN by providing it input images and obtaining respective predictions. The resulting image-prediction pairs are used to train a "knockoff" model. The adversary's intention is for the knockoff to compete with the victim model at the victim's task. Note that knowledge transfer [4, 14] approaches are a special case within our formulation, where the task, train/test data, and white-box teacher (victim) model are known to the adversary.

Within this formulation, we spell out questions answered in our paper with an end-goal of model functionality stealing:

1. Can we train a knockoff on a random set of query images and corresponding blackbox predictions?
2. What makes for a good set of images to query?

3. How can we improve sample efficiency of queries?
4. What makes for a good knockoff architecture?

2. Related Work

Model Stealing. Stealing various attributes of a blackbox ML model has been recently gaining popularity: parameters [35], hyperparameters [37], architecture [24], information on training data [31] and decision boundaries [25]. These works lay the groundwork to precisely reproduce the blackbox model. In contrast, as proposed by [35], we investigate stealing *functionality* of the blackbox independent of its internals. However, [35] addresses stealing extremely simple models by making additional assumptions (e.g., model family is known); we found chance-level performance using the approach in our experiments. [19, 25] extend the line of work by stealing models (slightly more complex than [35] e.g., shallow CNNs) to serve as a surrogate to craft adversarial examples, but by assuming knowledge and partial access to the training data distribution. In contrast to these works, we propose the first approach to steal complex vision models with high accuracy by making fewer assumptions and additionally under the constraint of minimizing queries to the blackbox.

Knowledge Distillation. Distillation [14] and related approaches [4, 5, 10, 38] transfer the knowledge from a complex "teacher" to a simpler "student" model. Within our problem formulation, this is a special case when the adversary has strong knowledge of the victim's blackbox model e.g., architecture, train/test data is known. Although we discuss this, a majority of the paper makes weak assumptions of the blackbox.

Active Learning. Active Learning [6, 34] (AL) aims to reduce labeling effort while gathering data to train a model. Ours is a special case of pool-based AL [30], where the learner (adversary) chooses from a pool of unlabeled data. However, unlike AL, the learner's image pool in our case is chosen without any knowledge of the data used by the original model. Moreover, while AL considers the image to be annotated by a human-expert, ours is annotated with pseudo-labels by the blackbox.

3. Problem Statement

We now formalize the task of functionality stealing (see also Figure 2).

Functionality Stealing. In this paper, we introduce the task as: given blackbox query access to a "victim" model $F_V : \mathcal{X} \rightarrow \mathcal{Y}$, to replicate its functionality using "knockoff" model F_A of the adversary. As shown in Figure 2, we set it up as a two-player game between a victim V and an adversary A. Now, we discuss the assumptions in which the players operate and their corresponding moves in this game.

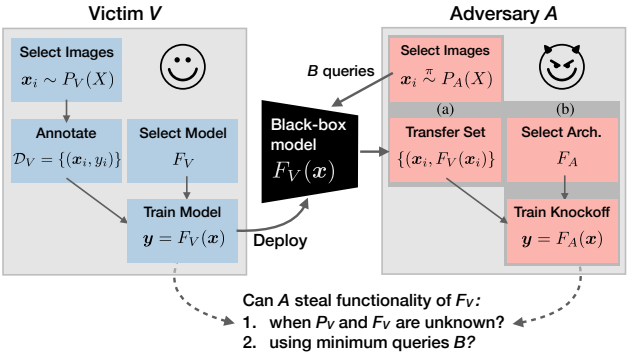


Figure 2: Problem Statement. Laying out the task of model functionality stealing in the view of two players - victim V and adversary A. We group adversary's moves into (a) Transfer Set Construction (b) Training Knockoff F_A.

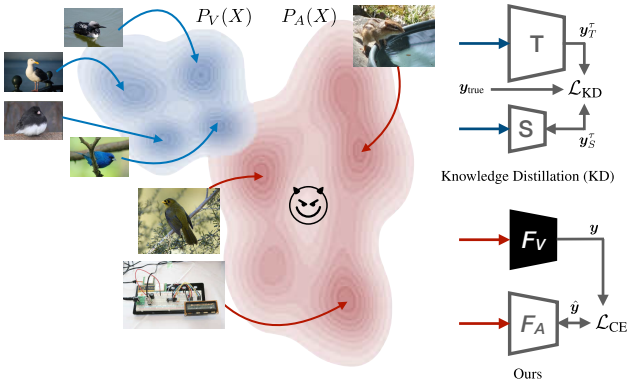
Victim's Move. The victim's end-goal is to deploy a trained CNN model F_V in the wild for a particular task (e.g., fine-grained bird classification). To train this particular model, the victim: (i) collects task-specific images $\boldsymbol{x} \sim P_V(X)$ and obtains expert annotations resulting in a dataset $\mathcal{D}_V = \{(\boldsymbol{x}_i, y_i)\}$; (ii) selects the model F_V that achieves best performance (accuracy) on a held-out test set of images $\mathcal{D}_V^{\text{test}}$. The resulting model is deployed as a blackbox which predicts output probabilities $\boldsymbol{y} = F_V(\boldsymbol{x})$ given an image $\boldsymbol{x}$. Furthermore, we assume each prediction incurs a cost (e.g., monetary, latency).

Adversary's Unknowns. The adversary is presented with a blackbox CNN image classifier, which given *any* image $\boldsymbol{x} \in \mathcal{X}$ returns a K-dim posterior probability vector $\boldsymbol{y} \in [0, 1]^K$, $\sum_k y_k = 1$. We relax this later by considering truncated versions of $\boldsymbol{y}$. We assume remaining aspects to be unknown: (i) the internals of F_V e.g., hyperparameters or architecture; (ii) the data used to train and evaluate the model; and (iii) semantics over the K classes.

Adversary's Attack. To train a knockoff, the adversary: (i) interactively queries images $\{\boldsymbol{x}_i \overset{\pi}{\sim} P_A(X)\}$ using strategy π to obtain a "transfer set" of images and pseudo-labels $\{(\boldsymbol{x}_i, F_V(\boldsymbol{x}_i))\}_{i=1}^B$; and (ii) selects an architecture F_A for the knockoff and trains it to mimic the behaviour of F_V on the transfer set.

Objective. We focus on the adversary, whose primary objective is training a knockoff that performs well on the task for which F_V was designed i.e., on an unknown $\mathcal{D}_V^{\text{test}}$. In addition, we address two secondary objectives: (i) sample-efficiency: maximizing performance within a budget of B blackbox queries; and (ii) understanding what makes for good images to query the blackbox.

Victim's Defense. Although we primarily address the adversary's strategy in the paper, we briefly discuss victim's counter strategies (in Section 6) of reducing informativeness of predictions by truncation e.g., rounding-off.



(a) (b)

Figure 3: Comparison to KD. (a) Adversary has access only to image distribution $P_A(X)$ (b) Training in a KD-manner requires stronger knowledge of the victim. Both S and F_A are trained to classify images $\boldsymbol{x} \in P_V(X)$

Remarks: Comparison to Knowledge Distillation (KD). Training the knockoff model is reminiscent of KD approaches [14, 27], whose goal is to transfer the knowledge from a larger teacher network T (white-box) to a compact student network S (knockoff) via the transfer set. We illustrate key differences between KD and our setting in Figure 3: (a) **Independent distribution $\boldsymbol{P_A}$**: F_A is trained on images $\boldsymbol{x} \sim P_A(X)$ *independent* to distribution P_V used for training F_V; (b) **Data for supervision**: Student network S minimize variants of KD loss:

$$\mathcal{L}_{\text{KD}} = \lambda_1 \mathcal{L}_{\text{CE}}(\boldsymbol{y}_{\text{true}},\ \boldsymbol{y}_S) + \lambda_2 \mathcal{L}_{\text{CE}}(\boldsymbol{y}_S^\tau,\ \boldsymbol{y}_T^\tau) \qquad (1)$$

where $\boldsymbol{y}_T^\tau = \text{softmax}(\boldsymbol{a}_T/\tau)$ is the softened posterior distribution of logits $\boldsymbol{a}$ controlled by temperature τ. In contrast, the knockoff (student) in our case lacks logits $\boldsymbol{a}_T$ and true labels $\boldsymbol{y}_{\text{true}}$ to supervise training.

4. Generating Knockoffs

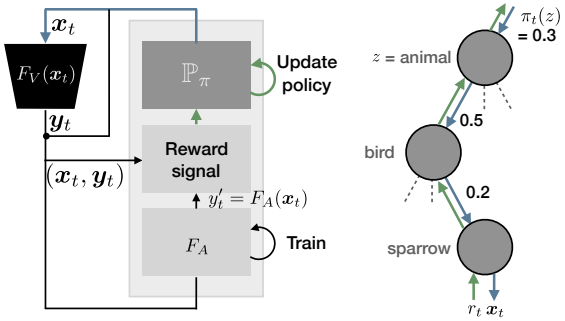
In this section, we elaborate on the adversary's approach in two steps: transfer set construction (Section 4.1) and training knockoff F_A (Section 4.2).

4.1. Transfer Set Construction

The goal is to obtain a transfer set i.e., image-prediction pairs, on which the knockoff will be trained to imitate the victim's blackbox model F_V.

Selecting $\boldsymbol{P_A(X)}$. The adversary first selects an image distribution to sample images. We consider this to be a large discrete set of images. For instance, one of the distributions P_A we consider is the 1.2M images of ILSVRC dataset [7].

Sampling Strategy $\boldsymbol{\pi}$. Once the image distribution $P_A(X)$ is chosen, the adversary samples images $\boldsymbol{x} \overset{\pi}{\sim} P_A(X)$ using a strategy π. We consider two strategies.



(a) Overview (b) Hierarchical policy $\mathbb{P}_\pi$

Figure 4: Strategy `adaptive`.

4.1.1 `Random` Strategy

In this strategy, we randomly sample images (without replacement) $\boldsymbol{x} \overset{\text{iid}}{\sim} P_A(X)$ to query F_V. This is an extreme case where adversary performs pure exploration. However, there is a risk that the adversary samples images irrelevant to learning the task (e.g., over-querying dog images to a birds classifier).

4.1.2 `Adaptive` Strategy

We now incorporate a feedback signal resulting from each image queried to the blackbox. A policy π to adaptively sample images ($\boldsymbol{x}_t \sim \mathbb{P}_\pi(\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{t-1})$) is learnt to achieve two goals: (i) improving sample-efficiency of queries; and (ii) aiding interpretability of blackbox F_V. The approach is outlined in Figure 4a. At each time-step t, the policy module $\mathbb{P}_\pi$ samples a set of query images. A reward signal r_t is shaped based on multiple criteria and is used to update the policy with an end-goal of maximizing the expected reward.

Supplementing $\boldsymbol{P_A}$. To encourage relevant queries, we enrich images in the adversary's distribution by associating each image $\boldsymbol{x}_i$ with a label $z_i \in Z$. No semantic relation of these labels with the blackbox's output classes is assumed or exploited. As an example, when P_A corresponds to 1.2M images of the ILSVRC [7] dataset, we use labels defined over 1000 classes. These labels can be alternatively obtained by unsupervised measures e.g., clustering or estimating graph-density [2,8]. We find using labels aids understanding blackbox functionality. Furthermore, since we expect labels $\{z_i \in Z\}$ to be correlated or inter-dependent, we represent them within a coarse-to-fine hierarchy, as nodes of a tree as shown in Figure 4b.

Actions. At each time-step t, we sample actions from a discrete action space $z_t \in Z$ i.e., adversary's independent label space. Drawing an action is a forward-pass (denoted by a blue line in Figure 4b) through the tree: at each node, we sample a child node with probability $\pi_t(z)$ (which sums to 1 over siblings). The probabilities are determined by a softmax distribution over the node potentials:

$\pi_t(z) = \frac{e^{H_t(z)}}{\sum_{z'} e^{H_t(z')}}$. Upon reaching a leaf-node, a sample of images is returned corresponding to label z_t.

Learning the Policy. We use the received reward r_t for an action z_t to update the policy π using the gradient bandit algorithm [33]. This update is equivalent to a backward-pass through the tree (denoted by a green line in Figure 4b), where the node potentials are updated as:

$$H_{t+1}(z_t) = H_t(z_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(z_t)) \quad \text{and} \quad (2)$$
$$H_{t+1}(z') = H_t(z') + \alpha(r_t - \bar{r}_t)\pi_t(z') \qquad \forall z' \neq z_t \quad (3)$$

where $\alpha = 1/N(z)$ is the learning rate, $N(z)$ is the number of times action z has been drawn, and $\bar{r}_t$ is the mean-reward over past Δ time-steps. $\pi_0(z)$ and $H_0(z)$ are intialized such that reaching all leaf nodes in the hierarchy are equally probable.

Rewards. To evaluate the quality of sampled images $\boldsymbol{x}_t$, we study three rewards. We use a margin-based **certainty** measure [18, 30] to encourage images where the victim is confident (hence indicating the domain F_V was trained on):

$$R^{\text{cert}}(\boldsymbol{y}_t) = P(\boldsymbol{y}_{t,k_1}|\boldsymbol{x}_t) - P(\boldsymbol{y}_{t,k_2}|\boldsymbol{x}_t) \quad (4)$$

where k_i is the ith-most confident class. To prevent the degenerate case of image exploitation over a single label, we introduce a **diversity** reward:

$$R^{\text{div}}(\boldsymbol{y}_{1:t}) = \sum_k \max(0, y_{t,k} - \bar{y}_{t:t-\Delta,k}) \quad (5)$$

To encourage images where the knockoff prediction $\hat{\boldsymbol{y}}_t = F_A(\boldsymbol{x}_t)$ does not imitate F_V, we reward high CE **loss**:

$$R^{\mathcal{L}}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = \mathcal{L}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) \quad (6)$$

We sum up individual rewards when multiple measures are used. To maintain an equal weighting, each reward is individually rescaled to [0, 1] and subtracted with a baseline computed over past Δ time-steps.

4.2. Training Knockoff F_A

As a product of the previous step of interactively querying the blackbox model, we have a transfer set $\{(\boldsymbol{x}_t, F_V(\boldsymbol{x}_t)\}_{t=1}^B$, $\boldsymbol{x}_t \overset{\pi}{\sim} P_A(X)$. Now we address how this is used to train a knockoff F_A.

Selecting Architecture $\boldsymbol{F_A}$. Few works [24, 37] have recently explored reverse-engineering the blackbox i.e., identifying the architecture, hyperparameters, etc. We however argue this is orthogonal to our requirement of simply stealing the functionality. Instead, we represent F_A with a reasonably complex architecture e.g., VGG [32] or ResNet [13]. Existing findings in KD [10, 14] and model compression [4, 12, 16] indicate robustness to choice of reasonably complex student models. We investigate the choice under weaker knowledge of the teacher (F_V) e.g., training data and architecture is unknown.

Blackbox (F_V)	$\lvert\mathcal{D}_V^{\text{train}}\rvert + \lvert\mathcal{D}_V^{\text{test}}\rvert$	Output classes K
`Caltech256` [11]	23.3k + 6.4k	256 general object categories
`CUBS200` [36]	6k + 5.8k	200 bird species
`Indoor67` [26]	14.3k + 1.3k	67 indoor scenes
`Diabetic5` [1]	34.1k + 1k	5 diabetic retinopathy scales

Table 1: Four victim blackboxes $\boldsymbol{F_V}$. Each blackbox is named in the format: [dataset][# output classes].

Training to Imitate. To bootstrap learning, we begin with a pretrained Imagenet network F_A (see § D.1 in supplementary for discussion on other initializations). We train the knockoff F_A to imitate F_V on the transfer set by minimizing the cross-entropy (CE) loss: $\mathcal{L}_{\text{CE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_k p(y_k) \cdot \log p(\hat{y_k})$. This is a standard CE loss, albeit weighed with the confidence $p(y_k)$ of the victim's label.

5. Experimental Setup

We now discuss the experimental setup of multiple victim blackboxes (Section 5.1), followed by details on the adversary's approach (Section 5.2).

5.1. Black-box Victim Models F_V

We choose four diverse image classification CNNs, addressing multiple challenges in image classification e.g., fine-grained recognition. Each CNN performs a task specific to a dataset. A summary of the blackboxes is presented in Table 1 (extended descriptions in appendix).

Training the Black-boxes. All models are trained using a ResNet-34 architecture (with ImageNet [7] pretrained weights) on the training split of the respective datasets. We find this architecture choice achieve strong performance on all datasets at a reasonable computational cost. Models are trained using SGD with momentum (of 0.5) optimizer for 200 epochs with a base learning rate of 0.1 decayed by a factor of 0.1 every 60 epochs. We follow the train-test splits suggested by the respective authors for **Caltech-256** [11], **CUBS-200-2011** [36], and **Indoor-Scenes** [26]. Since GT annotations for **Diabetic-Retinopathy** [1] test images are not provided, we reserve 200 training images for each of the five classes for testing. The number of test images per class for all datasets are roughly balanced. The test images of these datasets $\mathcal{D}_V^{\text{test}}$ are used to evaluate both the victim and knockoff models.

After these four victim models are trained, we use them as a blackbox for the remainder of the paper: images in, posterior probabilities out.

5.2. Representing P_A

In this section, we elaborate on the setup of two aspects relevant to transfer set construction (Section 4.1).

5.2.1 Choice of P_A

Our approach for transfer set construction involves the adversary querying images from a large discrete image distribution P_A. In this section, we present four choices considered in our experiments. Any information apart from the images from the respective datasets are unused in the `random` strategy. For the `adaptive` strategy, we use image-level labels (chosen independent of blackbox models) to guide sampling.

$\boldsymbol{P_A = P_V}$**.** For reference, we sample from the exact set of images used to train the blackboxes. This is a special case of knowledge-distillation [14] with unlabeled data at temperature $\tau = 1$.

$\boldsymbol{P_A}$ **= ILSVRC [7, 28].** We use the collection of 1.2M images over 1000 categories presented in the ILSVRC-2012 [28] challenge.

$\boldsymbol{P_A}$ **= OpenImages [21].** OpenImages v4 is a large-scale dataset of 9.2M images gathered from Flickr. We use a subset of 550K unique images, gathered by sampling 2k images from each of 600 categories.

$\boldsymbol{P_A = D^2}$**.** We construct a dataset wherein the adversary has access to all images in the universe. In our case, we create the dataset by pooling training data from: (i) all four datasets listed in Section 5.1; and (ii) both datasets presented in this section. This results in a "dataset of datasets" D^2 of 2.2M images and 2129 classes.

Overlap between $\boldsymbol{P_A}$ and $\boldsymbol{P_V}$. We compute overlap between labels of the blackbox (K, e.g., 256 Caltech classes) and the adversary's dataset (Z, e.g., 1k ILSVRC classes) as: $100 \times |K \cap Z|/|K|$. Based on the overlap between the two image distributions, we categorize P_A as:

1. $\boldsymbol{P_A = P_V}$: Images queried are identical to the ones used for training F_V. There is a 100% overlap.
2. **Closed-world ($\boldsymbol{P_A = D^2}$)**: Blackbox train data P_V is a subset of the image universe P_A. There is a 100% overlap.
3. **Open-world ($\boldsymbol{P_A \in \{}$ILSVRC, OpenImages$\boldsymbol{\}}$)**: Any overlap between P_V and P_A is purely coincidental. Overlaps are: `Caltech256` (42% ILSVRC, 44% OpenImages), `CUBS200` (1%, 0.5%), `Indoor67` (15%, 6%), and `Diabetic5` (0%, 0%).

5.2.2 `Adaptive` Strategy

In the adaptive strategy (Section 4.1.2), we make use of auxiliary information (labels) in the adversary's data P_A to guide the construction of the transfer set. We represent these labels as the leaf nodes in the coarse-to-fine concept hierarchy tree. The root node in all cases is a single concept "entity". We obtain the rest of the hierarchy as follows: (i) D^2: we add as parents the dataset the images belong to; (ii) ILSVRC: for each of the 1K labels, we obtain 30 coarse labels by clustering the mean visual features of each label obtained using 2048-dim pool features of an ILSVRC pre-trained Resnet model; (iii) OpenImages: We use the exact hierarchy provided by the authors.

6. Results

We now discuss the experimental results.

Training Phases. The knockoff models are trained in two phases: (a) *Online*: during transfer set construction (Section 4.1); followed by (b) *Offline*: the model is retrained using transfer set obtained thus far (Section 4.2). All results on knockoff are reported after step (b).

Evaluation Metric. We evaluate two aspects of the knockoff: (a) *Top-1 accuracy*: computed on victim's held-out test data $\mathcal{D}_V^{\text{test}}$ (b) *sample-efficiency*: best performance achieved after a budget of B queries. Accuracy is reported in two forms: absolute ($x\%$) or relative to blackbox F_V ($x\times$).

In each of the following experiments, we evaluate our approach with identical hyperparameters across all blackboxes, highlighting the generalizability of model functionality stealing.

6.1. Transfer Set Construction

In this section, we analyze influence of transfer set $\{(\boldsymbol{x}_i, F_V(\boldsymbol{x}_i)\}$ on the knockoff. For simplicity, for the remainder of this section we fix the architecture of the victim and knockoff to a Resnet-34 [13].

Reference: $\boldsymbol{P_A = P_V}$ (KD). From Table 2 (second row), we observe: (i) all knockoff models recover 0.92-1.05$\times$ performance of F_V; (ii) a better performance than F_V itself (e.g., 3.8% improvement on `Caltech256`) due to regularizing effect of training on soft-labels [14].

Can we learn by querying *randomly* from an independent distribution? Unlike KD, the knockoff is now trained and evaluated on different image distributions (P_A and P_V respectively). We first focus on the `random` strategy, which does not use any auxiliary information.

We make the following observations from Table 2 (`random`): (i) **closed-world**: the knockoff is able to reasonably imitate all the blackbox models, recovering 0.84-0.97$\times$ blackbox performance; (ii) **open-world**: in this challenging scenario, the knockoff model has *never* encountered images of numerous classes at test-time e.g., >90% of the bird classes in `CUBS200`. Yet remarkably, the knockoff is able to obtain 0.81-0.96$\times$ performance of the blackbox. Moreover, results marginally vary (at most 0.04$\times$) between ILSVRC and OpenImages, indicating any large diverse set of images makes for a good transfer set.

Upon qualitative analysis, we find the image and pseudo-label pairs in the transfer set are semantically incoherent (Fig. 6a) for output classes non-existent in training images

	P_A	random				adaptive			
		Caltech256	CUBS200	Indoor67	Diabetic5	Caltech256	CUBS200	Indoor67	Diabetic5
	P_V (F_V)	78.8 (1×)	76.5 (1×)	74.9 (1×)	58.1 (1×)	-	-	-	-
	P_V (KD)	82.6 (1.05×)	70.3 (0.92×)	74.4 (0.99×)	54.3 (0.93×)	-	-	-	-
Closed	D^2	76.6 (0.97×)	68.3 (0.89×)	68.3 (0.91×)	48.9 (0.84×)	82.7 (1.05×)	74.7 (0.98×)	76.3 (1.02×)	48.3 (0.83×)
Open	ILSVRC	75.4 (0.96×)	68.0 (0.89×)	66.5 (0.89×)	47.7 (0.82×)	76.2 (0.97×)	69.7 (0.91×)	69.9 (0.93×)	44.6 (0.77×)
	OpenImg	73.6 (0.93×)	65.6 (0.86×)	69.9 (0.93×)	47.0 (0.81×)	74.2 (0.94×)	70.1 (0.92×)	70.2 (0.94×)	47.7 (0.82×)

Table 2: Accuracy on test sets. Accuracy of blackbox F_V indicated in gray and knockoffs F_A in black. KD = Knowledge Distillation. Closed- and open-world accuracies reported at B=60k.

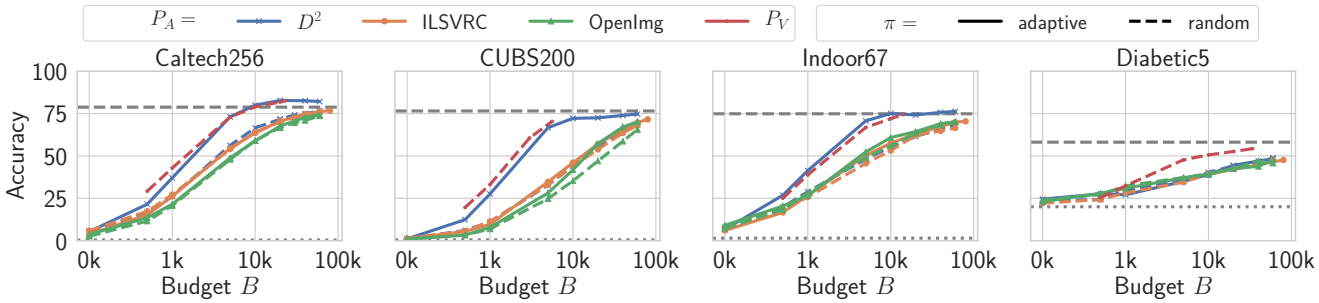


Figure 5: Performance of the knockoff at various budgets. Across choices of adversary's image distribution (P_A) and sampling strategy π. - represents accuracy of blackbox F_V and ····· represents chance-level performance. Enlarged version available in supplementary.

P_A. However, when relevant images are presented at test-time (Fig. 6b), the adversary displays strong performance. Furthermore, we find the top predictions by knockoff relevant to the image e.g., predicting one comic character (superman) for another.

How *sample-efficient* can we get? Now we evaluate the `adaptive` strategy (discussed in Section 4.1.2). Note that we make use of auxiliary information of the images in these tasks (labels of images in P_A). We use the reward set which obtained the best performance in each scenario: {certainty} (Eq. 4) in closed-world and {certainty, diversity, loss} (Eq. 4-6) in open-world.

From Figure 5, we observe: (i) **closed-world**: `adaptive` is extremely sample-efficient in all but one case. Moreover, we also find the label hierarchy result in better performance (see supp. §D.3). Its performance is comparable to KD in spite of samples drawn from a 36-188× larger image distribution. We find significant sample-efficiency improvements e.g., while `CUBS200-random` reaches 68.3% at B=60k, `adaptive` achieves this 6× quicker at B=10k. We find comparably low performance in `Diabetic5` as the blackbox exhibits confident predictions for all images resulting in poor feedback signal to guide policy; (ii) **open-world**: although we find marginal improvements over `random` in this challenging scenario, they are pronounced in few cases e.g., 1.5× quicker to reach an accuracy 57% on `CUBS200` with OpenImages. (iii) as an added-benefit apart from sample-efficiency, from Table 2, we find `adaptive` display improved performance (up to 4.5%) consistently across all choices of F_V.

What can we learn by inspecting the *policy*? From previous experiments, we observed two benefits of the `adaptive` strategy: sample-efficiency (although more prominent in the closed-world) and improved performance. The policy π_t learnt by `adaptive` (Section 4.1.2) additionally allows us to understand what makes for good images to query. $\pi_t(z)$ is a discrete probability distribution indicating preference over action z. Each action z in our case corresponds to labels in the adversary's image distribution.

We visualize $\pi_t(z)$ in Figure 7, where each bar represents an action and its color, the parent in the hierarchy. We observe: (i) **closed-world** (Fig. 7 top): actions sampled with higher probabilities consistently correspond to output classes of F_V. Upon analyzing parents of these actions (the dataset source), the policy also learns to sample images for the output classes from an alternative richer image source e.g., "ladder" images in `Caltech256` sampled from OpenImages instead; (ii) **open-world** (Fig. 7 bottom): unlike closed-world, the optimal mapping between adversary's actions to blackbox's output classes is non-trivial and unclear. However, we find top actions typically correspond to output classes of F_V e.g., indigo bunting. The policy, in addition, learns to sample coarser actions related to the F_V's task e.g., predominantly drawing from birds and animals images to knockoff `CUBS200`.

What makes for a good *reward*? Using the `adaptive` sampling strategy, we now address influence of three rewards (discussed in Section 4.1.2). We observe: (i) **closed-world** (Fig. 8 left): All reward signals in `adaptive` helps with the sample efficiency over `random`. Reward `cert` (Eq.

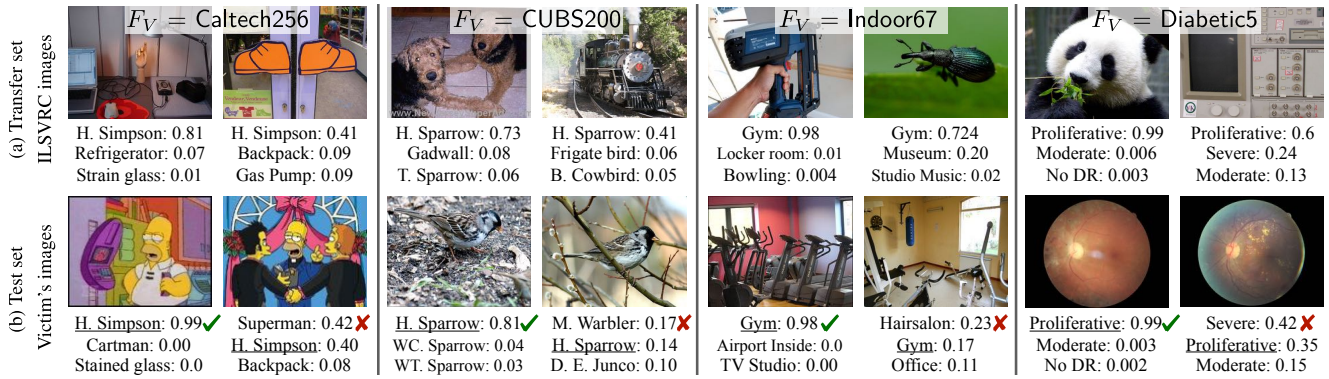


Figure 6: Qualitative Results. (a) Samples from the transfer set ($\{(\boldsymbol{x}_i, F_V(\boldsymbol{x}_i))\}, \boldsymbol{x}_i \sim P_A(X)$) displayed for four output classes (one from each blackbox): 'Homer Simpson', 'Harris Sparrow', 'Gym', and 'Proliferative DR'. (b) With the knockoff F_A trained on the transfer set, we visualize its predictions on victim's test set ($\{(\boldsymbol{x}_i, F_A(\boldsymbol{x}_i))\}, \boldsymbol{x}_i \sim \mathcal{D}_V^{\text{test}}$). Ground truth labels are underlined. Objects from these classes, among numerous others, were never encountered while training F_A.

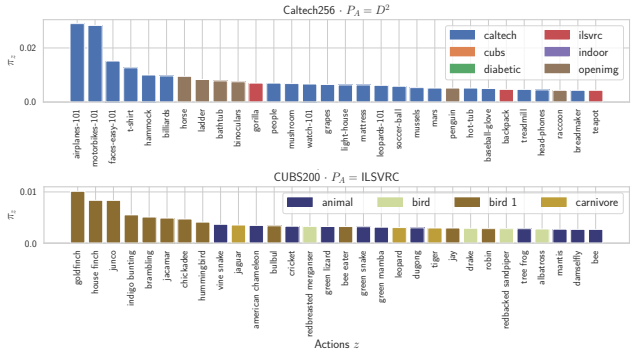


Figure 7: Policy π learnt by the `adaptive` approach. Each bar represents preference for action z. Top 30 actions (out of 2.1k and 1k) are displayed. Colors indicate parent of action in hierarchy.

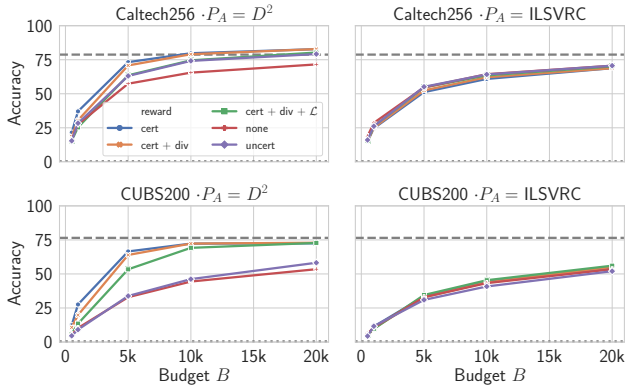


Figure 8: Reward Ablation. cert: certainty, uncert: uncertainty, div: diversity, $\mathcal{L}$: loss, none: no reward (`random` strategy).

4, which encourages exploitation) provides the best feedback signal. Including other rewards (Eq. 5-6) slightly deteriorates performance, as they encourage *exploration* over related or unseen actions – which is not ideal in a closed-world. Reward `uncert`, a popular measure used in AL literature [2, 8, 30] underperforms in our setting since it encourages uncertain (in our case, irrelevant) images. (ii) **open-world** (Fig. 8 right): Using all rewards (Eq. 4-6) display

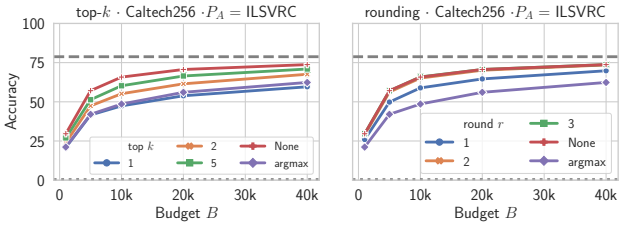


Figure 9: Truncated Posteriors. Influence of training knockoff with truncated posteriors.

only none-to-marginal improvements for all choices of F_V, with the highest improvement in `CUBS200`. However, we notice an influence on learnt policies where adopting exploration (`div + `$\mathcal{L}$) with exploitation (`cert`) goals result in a softer probability distribution π over the action space and in turn, encouraging related images.

Can we train knockoffs with *truncated blackbox outputs*? So far, we found adversary's *attack* objective of knocking off blackbox models can be effectively carried out with minimal assumptions. Now we explore the influence of victim's *defense* strategy of reducing informativeness of blackbox predictions to counter adversary's model stealing attack. We consider two truncation strategies: (a) top-k: top-k (out of K) unnormalized posterior probabilities are retained, while rest are zeroed-out; (b) rounding r: posteriors are rounded to r decimals e.g., round(0.127, r=2) = 0.13. In addition, we consider the extreme case "argmax", where only index $k = \arg\max_k y_k$ is returned.

From Figure 9 (with K = 256), we observe: (i) truncating $\boldsymbol{y}_i$ – either using top-k or rounding – slightly impacts the knockoff performance, with argmax achieving 0.76-0.84× accuracy of original performance for any budget B; (ii) top-k: even small increments of k significantly recovers the original performance – 0.91× at $k = 2$ and 0.96× at $k = 5$; (iii) rounding: recovery is more pronounced, with 0.99× original accuracy achieved at just $r = 2$. We find model

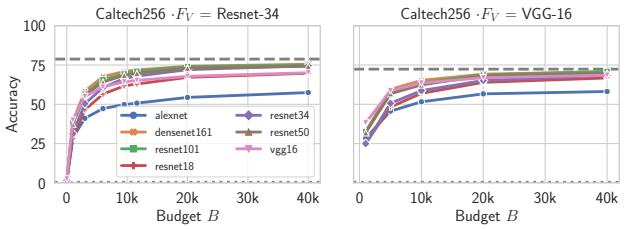


Figure 10: Architecture choices. F_V (left: Resnet-34 and right: VGG-16) and F_A (lines in each plot).

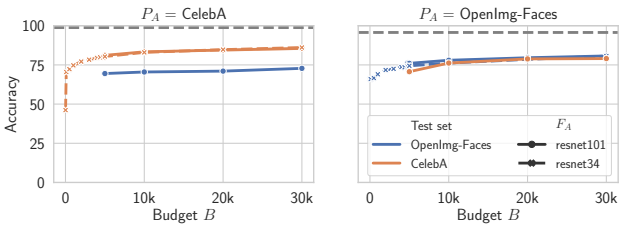


Figure 11: Knocking-off a real-world API. Performance of the knockoff achieved with two choices of P_A.

functionality stealing minimally impacted by reducing informativeness of blackbox predictions.

6.2. Architecture choice

In the previous section, we found model functionality stealing to be consistently effective while keeping the architectures of the blackbox and knockoff fixed. Now we study the influence of the architectural choice F_A vs. F_V.

How does the *architecture* of F_A influence knockoff performance? We study the influence using two choices of the blackbox F_V architecture: Resnet-34 [13] and VGG-16 [32]. Keeping these fixed, we vary architecture of the knockoff F_A by choosing from: Alexnet [20], VGG-16 [32], Resnet-{18, 34, 50, 101} [13], and Densenet-161 [15].

From Figure 10, we observe: (i) performance of the knockoff ordered by model complexity: Alexnet (lowest performance) is at one end of the spectrum while significantly more complex Resnet-101/Densenet-161 are at the other; (ii) performance transfers across model families: Resnet-34 achieves similar performance when stealing VGG-16 and vice versa; (iii) complexity helps: selecting a more complex model architecture of the knockoff is beneficial. This contrasts KD settings where the objective is to have a more compact student (knockoff) model.

6.3. Stealing Functionality of a Real-world Blackbox Model

Now we validate how our model functionality stealing attack translates to a real-world scenario. Image recognition services are gaining popularity allowing users to obtain image-predictions for a variety of tasks at low costs ($1-2 per 1k queries). These image recognition APIs have also been used to evaluate other attacks e.g., adversarial examples [3, 17, 22]. We focus on a facial characteristics API which given an image, returns attributes and confidences per face. Note that in this experiment, we have semantic information of blackbox output classes.

Collecting P_A. The API returns probability vectors per face in the image and thus, querying irrelevant images leads to a wasted result with no output information. Hence, we use two face image sets P_A for this experiment: CelebA (220k images) [23] and OpenImages-Faces (98k images). We create the latter by cropping faces (plus margin) from images in the OpenImages dataset [21].

Evaluation. Unlike previous experiments, we cannot access victim's test data. Hence, we create test sets for each image set by collecting and manually screening seed annotations from the API on ∼5K images.

How does this translate to the *real-world*? We model two variants of the knockoff using the `random` strategy (`adaptive` is not used since no relevant auxiliary information of images are available). We present each variant using two choices of architecture F_A: a compact Resnet-34 and a complex Resnet-101. From Figure 11, we observe: (i) strong performance of the knockoffs achieving 0.76-0.82× performance as that of the API on the test sets; (ii) the diverse nature OpenImages-Faces helps improve generalization resulting in 0.82× accuracy of the API on both testsets; (iii) the complexity of F_A does not play a significant role: both Resnet-34 and Resnet-101 show similar performance indicating a compact architecture is sufficient to capture discriminative features for this particular task.

We find model functionality stealing translates well to the real-world with knockoffs exhibiting a strong performance. The knockoff circumvents monetary and labour costs of: (a) collecting images; (b) obtaining expert annotations; and (c) tuning a model. As a result, an inexpensive knockoff can be trained which exhibits strong performance, using victim API queries amounting to only $30.

7. Conclusion

We investigated the problem of model functionality stealing where an adversary transfers the functionality of a victim model into a knockoff via blackbox access. In spite of minimal assumptions on the blackbox, we demonstrated the surprising effectiveness of our approach. Finally, we validated our approach on a real-world image recognition API and found strong performance of knockoffs. We find functionality stealing poses a real-world threat that potentially undercuts an increasing number of deployed ML models.

Acknowledgement. This research was partially supported by the German Research Foundation (DFG CRC 1223). We thank Yang Zhang for helpful discussions.

References

[1] Eyepacs. https://www.kaggle.com/c/diabetic-retinopathy-detection. Accessed: 2018-11-08.

[2] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *CVPR*, 2018.

[3] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*, 2017.

[4] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDD*, 2006.

[5] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *NIPS*, 2017.

[6] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *JAIR*, 1996.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[8] Sandra Ebert, Mario Fritz, and Bernt Schiele. Ralf: A reinforced active learning formulation for object class recognition. In *CVPR*, 2012.

[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 2015.

[10] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, 2018.

[11] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv:1602.07360*, 2016.

[17] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018.

[18] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, 2009.

[19] Mika Juuti, Sebastian Szyller, Alexey Dmitrenko, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *Euro S&P*, 2019.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[21] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018.

[22] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[23] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[24] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[25] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Asia CCS*, 2017.

[26] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[27] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[29] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NDSS*, 2019.

[30] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, 2008.

[31] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (S&P)*, 2017.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[33] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[34] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2001.

[35] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.

[36] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[37] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *Security and Privacy (S&P)*, 2018.

[38] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.