Model Extraction

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Data and Network Security Lab (DNSL)

May 23, 2023

Today's Agenda

Model Extraction

Model Extraction

Model extraction attacks target the **confidentiality** of a victim model deployed on a remote service.

- A model refers here to both the **architecture and its parameters**.
- The model can be viewed as **intellectual property** that the adversary is trying to steal.

Adversarial Motivations

There are two **primary intents** for adversaries to conduct model extraction attacks, **Stealing** and **Reconnaissance**.

Adversarial Motivations

There are two **primary intents** for adversaries to conduct model extraction attacks, **Stealing** and **Reconnaissance**.

- **Stealing**: Motivated by economic incentives. Adversaries are motivated to abuse the target classifier to **reduce the cost** of creating a new classifier.

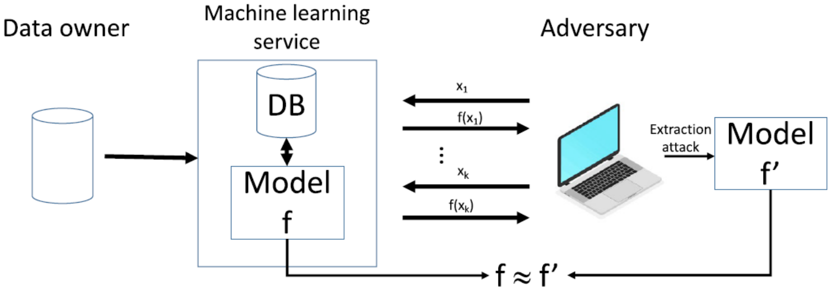
Adversarial Motivations

There are two **primary intents** for adversaries to conduct model extraction attacks, **Stealing** and **Reconnaissance**.

- **Stealing**: Motivated by economic incentives. Adversaries are motivated to abuse the target classifier to **reduce the cost** of creating a new classifier.
- **Reconnaissance**: Model extraction enables an adversary previously operating in a **black-box threat model** to mount attacks against the extracted model in a white-box threat model. The adversary is performing reconnaissance to later **mount attacks** targeting other security properties of the learning system
  - Integrity with adversarial examples
  - Privacy with training data membership inference.

Model Stealing Threat Model

The adversary has **black-box access** to the target model (Oracle)



(Trung ha, 2020)

Exact Extraction

Exact extraction is impossible.

- Functionality equivalent

Exact Extraction

Exact extraction is impossible.

- Functionality equivalent

Adversary’s Goal

- Stealing $\rightarrow$ **Accuracy**
- Reconnaissance $\rightarrow$ **Fidelity**
  - Functionality Equivalent (Perfect Fidelity)

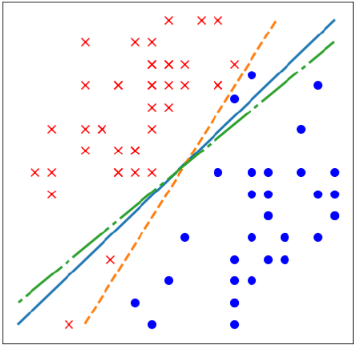


Figure 1: Illustrating fidelity vs. accuracy. The solid blue line is the oracle; functionally equivalent extraction recovers this exactly. The green dash-dot line achieves high fidelity: it matches the oracle on all data points. The orange dashed line achieves perfect accuracy: it classifies all points correctly.

(Jagielski, 2019)

Adversary's Goal

Accuracy

- For the true task distribution D_A over $\mathcal{X} \times \mathcal{Y}$, the goal of task accuracy extraction is to construct an $\hat{O}$ maximizing $Pr_{(x,y)\sim D_A}[argmax(\hat{O}(x)) = y]$.

Adversary's Goal

Accuracy

- For the true task distribution D_A over $\mathcal{X} \times \mathcal{Y}$, the goal of task accuracy extraction is to construct an $\hat{O}$ maximizing $Pr_{(x,y)\sim D_A}[argmax(\hat{O}(x)) = y]$.

Fidelity

- Given some target distribution D_F over $\mathcal{X}$, and goal similarity function $S(p_1, p_2)$, the goal of fidelity extraction is to construct an $\hat{O}$ that maximizes $Pr_{x\sim D_F}[S(\hat{O}(x), O(x))]$.

Adversary's Goal

Accuracy

- For the true task distribution D_A over $\mathcal{X} \times \mathcal{Y}$, the goal of task accuracy extraction is to construct an $\hat{O}$ maximizing $Pr_{(x,y)\sim D_A}[argmax(\hat{O}(x)) = y]$.

Fidelity

- Given some target distribution D_F over $\mathcal{X}$, and goal similarity function $S(p_1, p_2)$, the goal of fidelity extraction is to construct an $\hat{O}$ that maximizes $Pr_{x\sim D_F}[S(\hat{O}(x), O(x))]$.

Functionally Equivalent (Perfect Fidelity)

- The goal of functionally equivalent extraction is to construct an $\hat{O}$ such that $\forall x \in \mathcal{X}, \hat{O}(x) = O(x)$.

Adversarial Capabilities

Threat model

- **Label**: only the label of the most-likely class is revealed.
- **Label and score**: in addition to the most-likely label, the confidence score of the model in its prediction for this label is revealed.
- **Top-k scores**: the labels and confidence scores for the k classes whose confidence are highest are revealed.
- **Scores**: confidence scores for all labels are revealed.
- **Logits**: raw logit values for all labels are revealed.

Budget

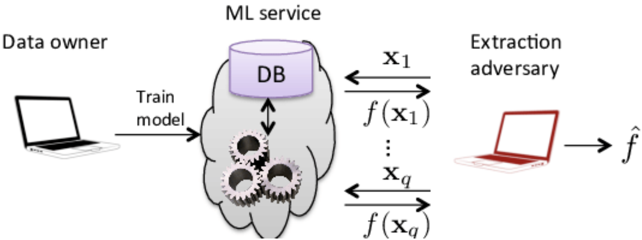
Adversaries want to **minimize the number of queries** to the target model in order to

- **Avoid detection** and prevention of the attack
- **Limit the amount of money** spent for predictions, in the case of MLaaS prediction APIs
- Minimize the number of **samples required** to query the model.

Attack Strategies

In-Distribution (ID) samples

- Transfer learning
- Semi-supervised learning
- Active learning
- Self-supervised learning



(Trammer, 2020)

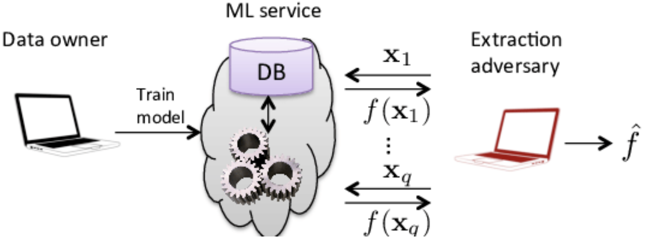
Attack Strategies

In-Distribution (ID) samples

- Transfer learning
- Semi-supervised learning
- Active learning
- Self-supervised learning

Out-Of-Distribution (OOD) samples

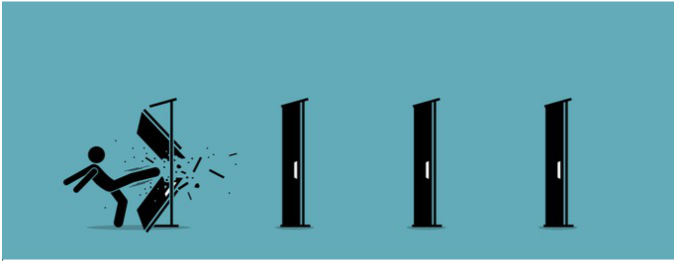
- Data Augmentation (Limited access to ID samples)
- Semantically similar natural samples
- Synthetic samples



(Trammer, 2020)

Defense Strategies

- Detection-based methods
- Perturbation-based methods
- Watermarking



(shutterstock.com)

Knockoff Nets

Knockoff Nets

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

It formulates model **functionality stealing** as a two-step approach

- **Querying** a set of input images to the blackbox model to obtain predictions
- Training a **knockoff** with queried image-prediction pairs.

Abstract

It formulates model **functionality stealing** as a two-step approach

- **Querying** a set of input images to the blackbox model to obtain predictions
- Training a **knockoff** with queried image-prediction pairs.

Remarkable observations

- Querying random images from a **different distribution** than that of the blackbox training data results in a well-performing knockoff
- This is possible even when the knockoff is represented using a **different architecture**
- **Reinforcement learning** approaches improve query sample efficiency in certain settings and provides performance gains.

Problem Statement

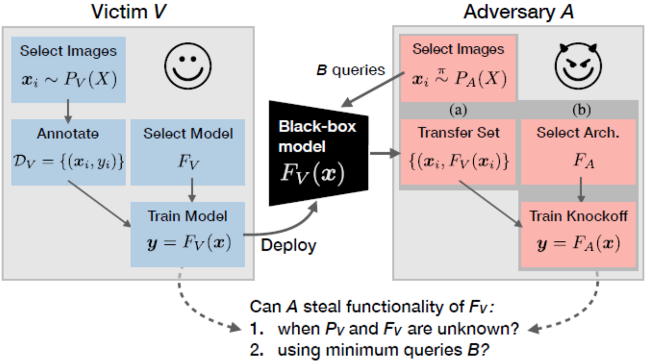


Figure 2: Problem Statement. Laying out the task of model functionality stealing in the view of two players - victim V and adversary A. We group adversary's moves into (a) Transfer Set Construction (b) Training Knockoff F_A.

Problem Statement

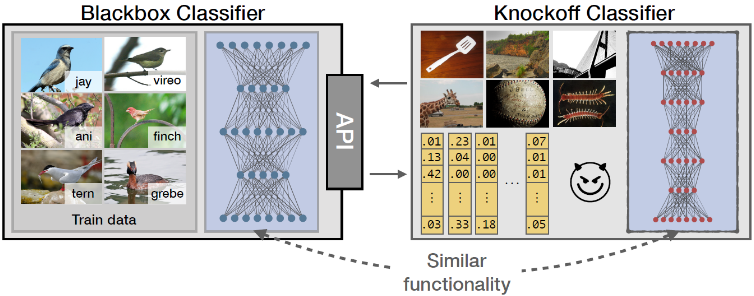


Figure 1: An adversary can create a "knockoff" of a blackbox model solely by interacting with its API: image in, prediction out. The knockoff bypasses the monetary costs and intellectual effort involved in creating the blackbox model.

Adversary's Attack

To train a knockoff, the adversary

- Interactively queries images $\{x_i \overset{\pi}{\sim} P_A(X)\}$ using **strategy** π to obtain a **transfer set** of images and pseudo-labels $\{(x_i, F_V(x_i))\}_{i=1}^{B}$
- **Selects an architecture** F_A for the knockoff and trains it to mimic the behavior of F_V on the transfer set.
- **Objective**
  - Maximizing performance within a budget of B blackbox queries

Transfer Set Construction

Selecting $P_A(X)$

- This can be a **large set of natural images**. For instance, one of the distributions P_A we consider is the 1.2M images of ILSVRC dataset

Transfer Set Construction

Selecting $P_A(X)$

- This can be a **large set of natural images**. For instance, one of the distributions P_A we consider is the 1.2M images of ILSVRC dataset

Sampling Strategy π

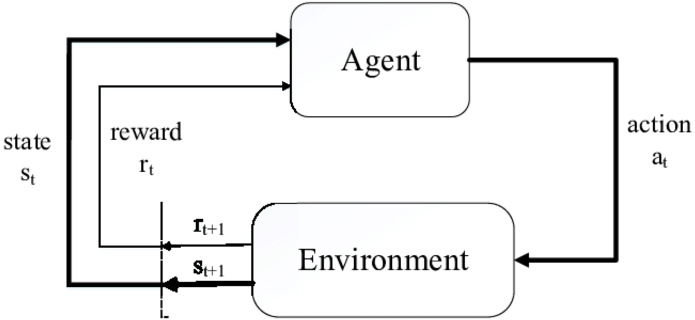
- Once the image distribution $P_A(X)$ is chosen, the adversary samples images $x \overset{\pi}{\sim} P_A(X)$ using a strategy π. There are two strategies.
  - **Random strategy**
  - **Adaptive strategy**

Random Strategy

In this strategy, the adversary **randomly samples images** (without replacement) $x \overset{iid}{\sim} P_A(X)$ to query F_V . This is an extreme case where adversary performs pure exploration.

- There is a risk that the adversary **samples images irrelevant to learning the task** (e.g., over-querying dog images to a birds classifier).

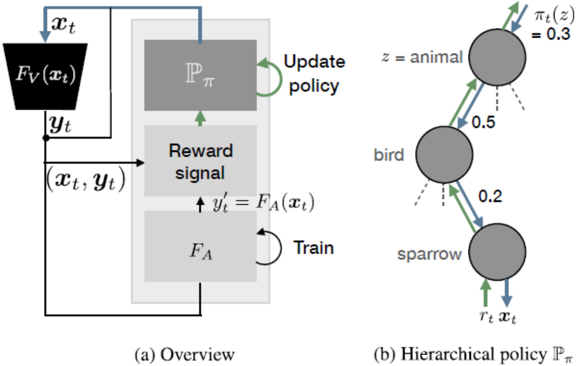
Adaptive Strategy - Reinforcement Learning



Supplementing P_A in Adaptive Strategy

- To encourage relevant queries, images in the adversary's distribution are enriched by associating each image 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 dataset, labels defined over 1000 classes are used.
  - These labels can be alternatively obtained by unsupervised measures e.g., clustering.
- 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.

Adaptive Strategy



(a) Overview

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

Figure 4: Strategy `adaptive`.

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).

Upon reaching a leaf-node, a sample of images is returned corresponding to label z_t.

Rewards

The quality of sampled image x_t is evaluated by **three reward functions**

Rewards

The quality of sampled image x_t is evaluated by **three reward functions**

- **Margin-based certainty** measure to encourage images where the victim is confident (hence indicating the domain F_V was trained on):

$$R^{cert}(y_t) = P(y_{t,k_1}|x_t) - P(y_{t,k_2}|x_t)$$

where $F_V(x_t) = y_t$ and k_i is the i^{th}-most confident class.

Rewards

The quality of sampled image x_t is evaluated by **three reward functions**

- **Margin-based certainty** measure to encourage images where the victim is confident (hence indicating the domain F_V was trained on):

$$R^{cert}(y_t) = P(y_{t,k_1}|x_t) - P(y_{t,k_2}|x_t)$$

  where $F_V(x_t) = y_t$ and k_i is the i^{th}-most confident class.

- To prevent the degenerate case of image exploitation over a single label, we introduce a **diversity reward** ($F_A(x_t) = y_t$):

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

Rewards

The quality of sampled image x_t is evaluated by **three reward functions**

- **Margin-based certainty** measure to encourage images where the victim is confident (hence indicating the domain F_V was trained on):

$$R^{cert}(y_t) = P(y_{t,k_1}|x_t) - P(y_{t,k_2}|x_t)$$

where $F_V(x_t) = y_t$ and k_i is the i^{th}-most confident class.

- To prevent the degenerate case of image exploitation over a single label, we introduce a **diversity reward** ($F_A(x_t) = y_t$):

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

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

$$R^{\mathcal{L}}(y_t, \hat{y}_t) = CE(y_t, \hat{y}_t)$$

Rewards

The quality of sampled image x_t is evaluated by **three reward functions**

- **Margin-based certainty** measure to encourage images where the victim is confident (hence indicating the domain F_V was trained on):

$$R^{cert}(y_t) = P(y_{t,k_1}|x_t) - P(y_{t,k_2}|x_t)$$

  where $F_V(x_t) = y_t$ and k_i is the i^{th}-most confident class.
- To prevent the degenerate case of image exploitation over a single label, we introduce a **diversity reward** ($F_A(x_t) = y_t$):

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

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

$$R^{\mathcal{L}}(y_t, \hat{y}_t) = CE(y_t, \hat{y}_t)$$

Individual rewards are **summed up** when multiple measures are used.

Learning the Policy

Gradient bandit algorithm (Reinforcement Learning: An Introduction - Sec. 2.7)

- We **use the received reward** r_t **for an action** z_t **to update the policy** π using the gradient bandit algorithm

Learning the Policy

Gradient bandit algorithm (Reinforcement Learning: An Introduction - Sec. 2.7)

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

$$\pi_t(z) = \frac{e^{H_t(z)}}{\sum_{z'} e^{H_t(z')}}$$

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

where $\alpha = \frac{1}{N(z)}$ 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.

Learning the Policy

Gradient bandit algorithm (Reinforcement Learning: An Introduction - Sec. 2.7)

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

$$\pi_t(z) = \frac{e^{H_t(z)}}{\sum_{z'} e^{H_t(z')}}$$

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

where $\alpha = \frac{1}{N(z)}$ 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 initialized such that reaching all leaf nodes in the hierarchy are equally probable.

Selecting Architecture F_A

F_A with a **reasonably complex architecture** e.g., VGG or ResNet.

- Existing findings in Knowledge distillation indicate robustness to choice of reasonably complex student models.

Training to Imitate

- To bootstrap learning, the training is initialized with a pretrained Imagenet network (**Transfer Learning**)
- Knockoff F_A is trained to imitate F_V on the transfer set by minimizing the cross entropy

Training the Black-boxes

All models are trained using a **ResNet-34 architecture**

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 F_V. Each blackbox is named in the format: [dataset][# output classes].

Choice of P_A

1. $P_A = P_V$
2. $P_A =$ ILSVRC
3. $P_A =$ OpenImages
   - OpenImages v4 is a large-scale dataset of 9.2M images gathered from Flickr. A subset of 550K unique images is gathered in 600 categories.
4. $P_A = D^2$: dataset of datasets

Datasets

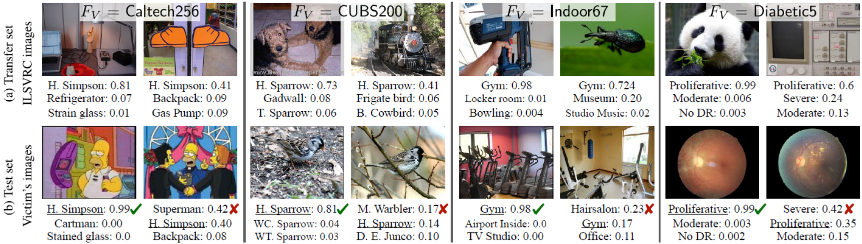


Figure 6: Qualitative Results. (a) Samples from the transfer set ($\{(\boldsymbol{x}_i, F_V(\boldsymbol{x}_i))\}, \boldsymbol{x}_i \sim P_A(X)$) displayed for four output classes (one from each blackbox): 'Homer Simpson', 'Harris Sparrow', 'Gym', and 'Proliferative DR'. (b) With the knockoff F_A trained 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.

Accuracy on test sets

	P_A	random Caltech256	random CUBS200	random Indoor67	random 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×)
Open	ILSVRC	75.4 (0.96×)	68.0 (0.89×)	66.5 (0.89×)	47.7 (0.82×)
Open	OpenImg	73.6 (0.93×)	65.6 (0.86×)	69.9 (0.93×)	47.0 (0.81×)

	P_A	adaptive Caltech256	adaptive CUBS200	adaptive Indoor67	adaptive Diabetic5
	$P_V (F_V)$	-	-	-	-
	P_V (KD)	-	-	-	-
Closed	D^2	82.7 (1.05×)	74.7 (0.98×)	76.3 (1.02×)	48.3 (0.83×)
Open	ILSVRC	76.2 (0.97×)	69.7 (0.91×)	69.9 (0.93×)	44.6 (0.77×)
Open	OpenImg	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. B=60k.

Performance of the knockoff at various budgets.

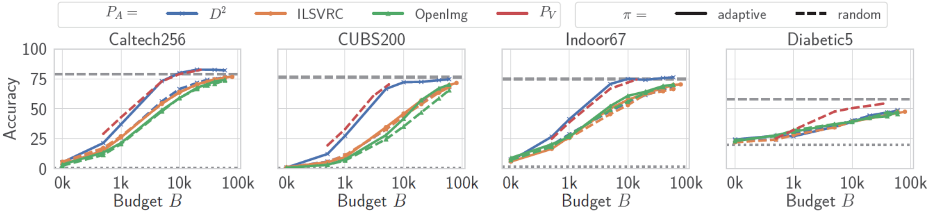


Figure 5: Performance of the knockoff at various budgets. Across choices of adversary's image distribution (P_A) and sampling strategy π. - represents accuracy of blackbox F_V and ···· represents chance-level performance. Enlarged version available in supplementary.

Policy π learnt by the adaptive approach

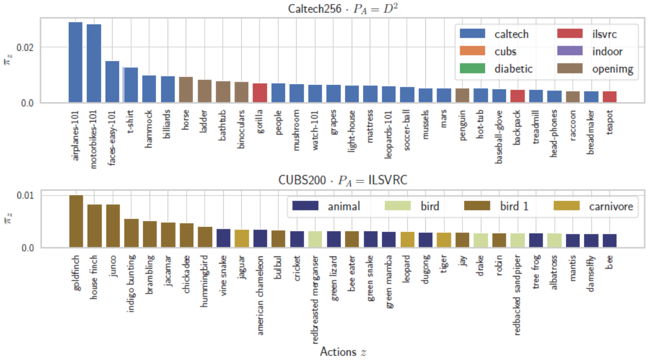


Figure 7: Policy π learnt by the `adaptive` approach. Each bar represents preference for action z. Top 30 actions (out of 2.1k and 1k) are displayed. Colors indicate parent of action in hierarchy.

Reward Ablation

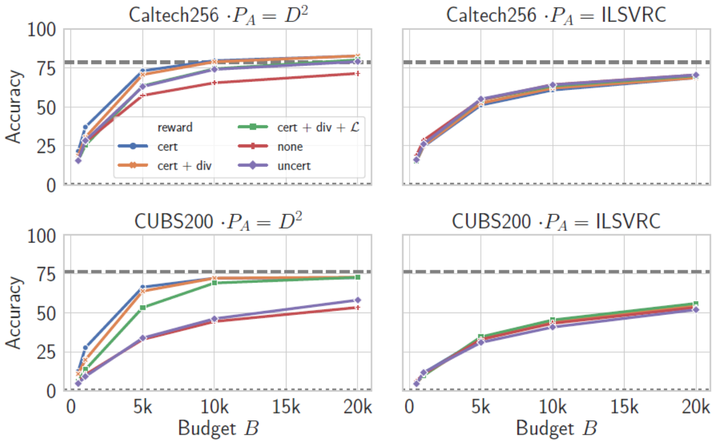


Figure 8: Reward Ablation. cert: certainty, uncert: uncertainty, div: diversity, $\mathcal{L}$: loss, none: no reward (`random` strategy).

Truncated Posteriors

1. top-k
   - Top-k (out of K) unnormalized posterior probabilities are retained, while rest are zeroed-out.
2. rounding r
   - Posteriors are rounded to r decimals e.g., $round(0.127, r = 2) = 0.13$.

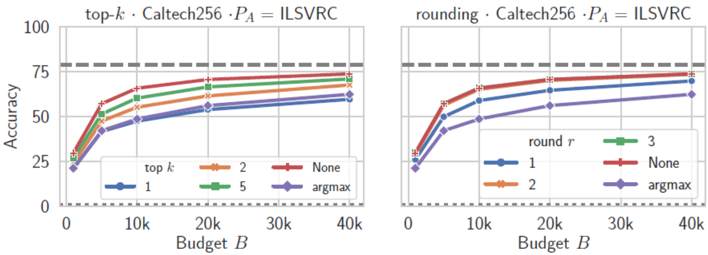


Figure 9: Truncated Posteriors. Influence of training knockoff with truncated posteriors.

Architecture choices

- Selecting a more complex model architecture of the knockoff is beneficial.

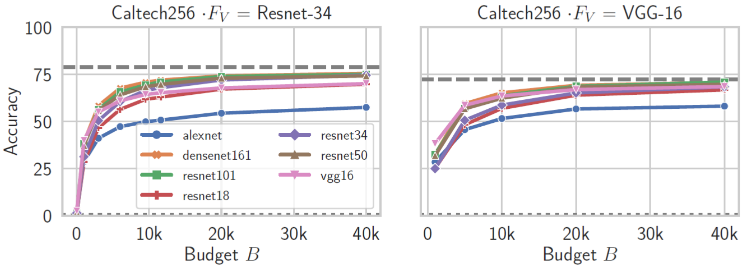


Figure 10: Architecture choices. F_V (left: Resnet-34 and right: VGG-16) and F_A (lines in each plot).

Effect of CNN Initialization

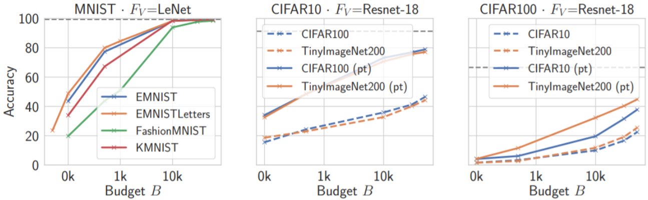


Figure S3: Training with non-ImageNet initializations of knockoff models. Shown for various choices of blackboxes F_V (subplots) and adversary's image distribution P_A (lines). All victim blackbox models are trained from scratch; test accuracy indicated by --- . All knockoff models are either trained from scratch, or pretrained on the corresponding P_A task (suffixed with '(pt)').