With Great Training Comes Great Vulnerability: Practical Attacks against Transfer Learning

Bolun Wang, ***UC Santa Barbara;*** **Yuanshun Yao,** ***University of Chicago;***
Bimal Viswanath, ***Virginia Tech;*** **Haitao Zheng and Ben Y. Zhao,** ***University of Chicago***

https://www.usenix.org/conference/usenixsecurity18/presentation/wang-bolun

This paper is included in the Proceedings of the 27th USENIX Security Symposium.

August 15–17, 2018 • Baltimore, MD, USA

978-1-939133-04-5

Open access to the Proceedings of the 27th USENIX Security Symposium is sponsored by USENIX.

With Great Training Comes Great Vulnerability: Practical Attacks against Transfer Learning

Bolun Wang
UC Santa Barbara

Yuanshun Yao
University of Chicago

Bimal Viswanath
Virginia Tech

Haitao Zheng
University of Chicago

Ben Y. Zhao
University of Chicago

Abstract

Transfer learning is a powerful approach that allows users to quickly build accurate deep-learning (Student) models by "learning" from centralized (Teacher) models pretrained with large datasets, *e.g.* Google's InceptionV3. We hypothesize that the centralization of model training increases their vulnerability to misclassification attacks leveraging knowledge of publicly accessible Teacher models. In this paper, we describe our efforts to understand and experimentally validate such attacks in the context of image recognition. We identify techniques that allow attackers to associate Student models with their Teacher counterparts, and launch highly effective misclassification attacks on black-box Student models. We validate this on widely used Teacher models in the wild. Finally, we propose and evaluate multiple approaches for defense, including a neuron-distance technique that successfully defends against these attacks while also obfuscates the link between Teacher and Student models.

1 Introduction

Deep learning using neural networks has transformed computing as we know it. From image and face recognition, to self-driving cars, knowledge extraction and retrieval, and natural language processing and translation, deep learning has produced game-changing applications in every field it has touched.

While advances in deep learning seem to arrive on a daily basis, one constraint has remained: deep learning can only build accurate models by training using large datasets. This thirst for data severely constrains the number of different models that can be independently trained. In addition, the process of training large, accurate models (often with millions of parameters) requires computational resources that can be prohibitive for individuals or small companies. For example, Google's InceptionV3 model is based on a sophisticated architecture with 48 layers, trained on $\sim$1.28M labeled images over a period of 2 weeks on 8 GPUs.

The prevailing consensus is to address the data and training resource problem using *transfer learning*, where a small number of highly tuned and complex centralized models are shared with the general community, and individual users or companies further customize the model for a given application with additional training. By using the pretrained *teacher* model as a launching point, users can generate accurate *student* models for their application using only limited training on their smaller domain-specific datasets. Today, transfer learning is recommended by most major deep learning frameworks, including Google Cloud ML, Microsoft Cognitive Toolkit, and PyTorch from Facebook.

Despite its appeal as a solution to the data scarcity problem, the centralized nature of transfer learning creates a more attractive and vulnerable target for attackers. Lack of diversity has amplified the power of targeted attacks in other contexts, *i.e.* increasing the impact of targeted attacks on network hubs [21], supernodes in overlay networks [54], and the impact of software vulnerabilities in popular libraries [71, 22].

In this paper, we study the possible negative implications of deriving models from a small number of centralized teacher models. Our hypothesis is that boundary conditions that can be discovered in the white box teacher models can be used to perform targeted misclassification attacks against its associated student models, even if the student models themselves are closed, *i.e.* black-box. Through detailed experimentation and testing, we find that this vulnerability does in fact exist in a variety of the most popular image classification contexts, including facial and iris recognition, and the identification of traffic signs and flowers. Unlike prior work on black-box adversarial attacks, this attack does not require repeated queries of the student model, and can instead prepare the attack image based on knowledge of the teacher model and any target image(s).

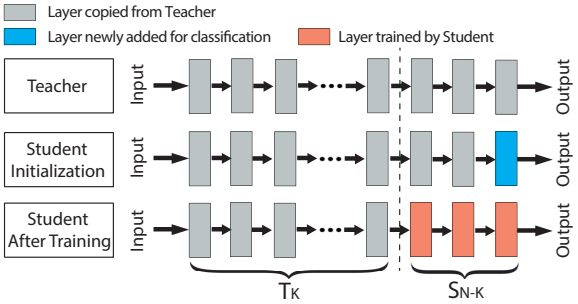


Figure 1: Transfer learning. A student model is initialized by copying the first N-1 layers from a teacher model, with a new dense layer added for classification. The model is further trained by only updating the last N-K layers.

This paper describes several key contributions:

- We identify and extensively evaluate the practicality of misclassification attacks against student models in multiple transfer-learning applications.
- We identify techniques to reliably identify teacher models given a student model, and show its effectiveness using known student models in the wild.
- We perform tests to evaluate and confirm the effectiveness of these attacks on popular deep learning frameworks, including Google Cloud ML, Microsoft Cognitive Toolkit (CNTK), and the PyTorch open source framework initially developed by Facebook.
- We explore and develop multiple defense techniques against attacks on transfer learning models, including defenses that alter the student model training process, that alter inputs prior to classification, and techniques that introduce redundancy using multiple models.

Transfer learning is a powerful approach that addresses one of the fundamental challenges facing the widespread deployment of deep learning. To the best of our knowledge, our work is the first to extensively study the inheritance of vulnerabilities between transfer learning models. Our goal is to bring attention to fundamental weaknesses in these models, and to advocate for the evaluation and adoption of defensive measures against adversarial attacks in the future.

2 Background

We begin by providing some background information on transfer learning and adversarial attacks on deep learning frameworks.

2.1 Transfer Learning

The high level idea of transfer learning is to transfer the "knowledge" from a pre-trained *Teacher* model to a new *Student* model, where the student model's task shares significant similarity to the teacher model's. This "knowledge" typically includes the model architecture and weights associated with the layers. Transfer learning enables organizations without access to massive datasets or GPU clusters to quickly build accurate models customized to their application context.

How Transfer Learning Works. Figure 1 illustrates transfer learning at a high level. The student model is initialized by copying the first $N-1$ layers of the Teacher. A new dense layer is added for classification. Its size matches the number of classes in the student task. Then the student model is trained using its own dataset, while the first K layers are "frozen", *i.e.* their weights are fixed, and only weights in the last $N-K$ layers are updated.

The first K layers (referred to as *shallow layers*) are frozen during training because outputs of those layers already represent meaningful features for the student task. The student model can reuse these features directly, and freezing them lowers both training cost and amount of required training data.

Based on the number of layers being frozen (K) during the training process, transfer learning is categorized into the following three approaches.

- *Deep-layer Feature Extractor:* $N-1$ layers are frozen during training, and only the last classification layer is updated. This is preferred when the student task is very similar to the teacher task, and requires minimal training cost (the cost of training a single-layer DNN).
- *Mid-layer Feature Extractor:* The first K layers are frozen, where $K < N-1$. Allowing more layers to be updated helps the student perform more optimization for its own task. *Mid-layer Feature Extractor* typically outperforms *Deep-layer Feature Extractor* in scenarios where the student task is more dissimilar to the teacher task, and more training data is available.
- *Full Model Fine-tuning:* All layers are unfrozen and fine-tuned during student training ($K = 0$). This requires more training data, and is appropriate when the student task differs significantly from the teacher task. Bootstrapping using pre-trained model weights helps the student converge faster and potentially achieve better performance than training from scratch [23].

We run a simple experiment to demonstrate the impact of transfer learning. We target facial recognition, where the student task is to recognize a set of 65 faces, and uses a well-performing face recognition model called VGG-Face [11] as teacher model. Using only 10 images per class to train the student model, we achieve 93.47% classification accuracy. Training the student with the same architecture but with random weights (no pre-trained weights) produces accuracy close to random guessing.

2.2 Adversarial Attacks in Deep Learning

The goal of adversarial attacks against deep learning networks is to modify input images so that they are misclassified in the DNN. Given a source image, the attacker applies a small perturbation so that it is misclassified by the victim DNN into either a specific target class, or any class other than the real class. Existing attacks fall into two categories, based on their assumptions on how much information attacker has about the classifier.

White-box Attacks. These attacks assume the attacker knows the full internals of the classifier DNN, including its architecture and all weights. It allows the attacker to run unlimited queries on the model, until a success adversarial sample is found [17, 36, 47, 41, 55]. These attacks often achieve close to 100% success with minimal perturbations, since full access to the DNN allows them to find the minimal amount of perturbations required for misclassification. The white-box scenario is often considered impractical, however, since few systems reveal internals of their model publicly.

Black-box Attacks. Here attackers do not have knowledge of the internals of the victim DNN, *i.e.* it remains a black-box. The attacker is allowed to query the victim model as an Oracle [46, 55]. Most black-box attacks either use queries to test intermediate adversarial samples and improve iteratively [55], or try to reverse-engineer decision boundaries of the DNN and build a replica, which can be used to craft adversarial samples [46]. Black-box attacks often achieve lower success than white-box attacks, and require a large number of queries to the target classifier [55].

Adversarial attacks can also be categorized into *targeted* and *non-targeted* attacks. A targeted attack aims to misclassify the adversarial image into a specific target class, whereas a non-targeted attack focuses on triggering misclassification into any class other than the real class. We consider and evaluate both targeted and non-targeted attacks in this paper.

3 Attacks on Transfer Learning

Here, we describe our attack on transfer learning, beginning with the attack model.

Attack Model. In the context of our definitions in Section 2, our attack assumes white-box access to teacher models (consistent with common practice today) and black-box access to student models. We consider a given attacker looking to trigger a misclassification from a Student model S, which has been customized through transfer learning from a Teacher model T.

- *White-box Teacher Model.* We assume that T is a white-box, meaning the attacker knows its model architecture and weights. Most or all popular models today have been made publicly available to increase adoption. Even if Teacher models became proprietary in the future, an attacker targeting a single Teacher model could obtain it by posing as a Student to gain access to the Teacher model.
- *Black-box Student Model.* We assume S is black-box, and all weights remain hidden from the attacker. We also assume the attacker does not know the Student training dataset, and can use only limited queries (*e.g.,* 1) to S. Apart from a single adversarial sample to trigger misclassification, we expect no additional queries to be made during the pre-attack process.
- *Transfer Learning Parameters.* We assume the attacker knows that S was trained using T as a Teacher, and which layers were frozen during the Student training. This information is not hard to learn, as many service providers, *e.g.,* Google Cloud ML, release such information in their official tutorials. We further relax this assumption in Sections 4 and 5, and consider scenarios where such information is unknown. We will discuss the impact on performance, and propose techniques to extract such information from the Student using a few additional queries.

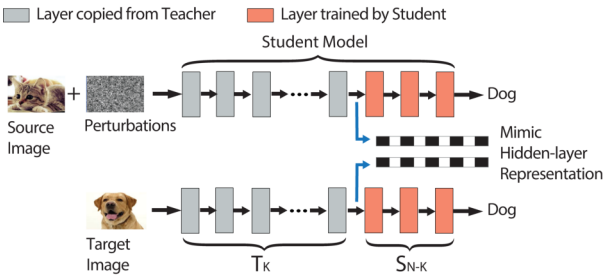


Figure 2: Illustration of our attack. Given images of a cat and a dog, attacker computes perturbations that mimic the internal representation of the dog image at layer K. If the calculations are perfect, the adversarial sample will be classified as dog, regardless of unknown layers in S_{N-K}.

Insight and Attack Methodology. Figure 2 illustrates the main idea behind our attack. Consider the scenario where the attacker knows that the first K layers of the Student model are copied from the Teacher and frozen during training. Attacker perturbs the source image so it could be misclassified as the same class of a specific target image. Using the Teacher model, *attacker computes perturbations that mimic the internal representation of the target image at layer K.* Internal representation is captured by passing the target image as input to the Teacher, and using the values of the corresponding neuron outputs at layer K.

Our key insight: is that (in feedforward networks) since each layer can only observe what is passed on from the previous layer, if our adversarial sample's internal representation at layer K perfectly matches that of the target image, it must be misclassified into the same label as the target image, regardless of the weights of any layers that follow K.

This means that in the common case of feature extractor training, if we can mimic a target in the Teacher model, then misclassification will occur regardless of how much the Student model trains with local data. We also note that some models like InceptionV3 and ResNet50, where "shortcut" layers can skip several layers, are not strictly feedforward. However, the same principle applies, because a block (consisting of several layers) only takes information from the previous block. Finally, it is hard in practice to perfectly mimic the internal representation, since we are limited in our level of possible perturbation, in order to keep adversarial changes indistinguishable by humans. The attacker's goal, therefore, is to minimize the dissimilarity between internal representations, given a fixed level of perturbation.

Targeted vs. Non-targeted Attacks. We consider both targeted and non-targeted attacks. The goal in targeted attacks is to misclassify a source image x_s into the class of a target image x_t. The attacker focuses on a specific layer K of the Teacher model, and tries to mimic the target image's internal representation (neuron values) at layer K. Let $T_K(.)$ be the function (associated with Teacher) transforming an input image to the internal representation at layer K. A perturbation budget P is used to control the amount of perturbation added to the source image. The following optimization problem is solved to craft an adversarial sample x'_s.

$$\begin{aligned} \min \quad & D(T_K(x'_s), T_K(x_t)) \\ \text{s.t.} \quad & d(x'_s, x_s) < P \end{aligned} \tag{1}$$

The above optimization tries to minimize dissimilarity $D(.)$ between the two internal representations, under a constraint to limit perturbation within a budget P. We use L_2 distance to compute $D(.)$. $d(x', x_s)$ is a distance function measuring the amount of perturbation added to x_s. We discuss $d(.)$ later in this section.

In non-targeted attacks, the goal is to misclassify x_s into any class different from the source class. To do this, we need to identify a "direction" to push the source image outside its decision boundary. In our case, it is hard to estimate such a direction without having a target image in hand, as we rely on mimicking hidden representations. Therefore, we perform a non-targeted attack by evaluating multiple targeted attacks, and choose the one that achieves the minimum dissimilarity between the internal representations. We assume that the attacker has access to a set of target images I (each belonging to a distinct class). Note that the source image can be misclassified to even classes outside the set I. The set of images I merely serves as a guide for the optimization process. Empirically, we find that even small sizes of set I (just 5 images) can achieve high attack success. The optimization problem is formulated as follows.

$$\begin{aligned} \min \quad & min_{i \in I}\{D(T_K(x'_s), T_K(x_{ti}))\} \\ \text{s.t.} \quad & d(x'_s, x_s) < P \end{aligned} \tag{2}$$

Measuring Adversarial Perturbations. As mentioned before, $d(x'_s, x_s)$ is the distance function used to measure the amount of perturbation added to the image. Most prior work used the L_p distance family, *e.g.,* L_0, L_2, and L_∞ [17]. While a helpful way to quantify perturbation, L_p distance fails to capture what humans perceive as image distortion. Therefore, we use another metric, called *DSSIM*, which is an objective image quality assessment metric that closely matches with the perceived quality of an image (*i.e.* subjective assessment) [65, 66]. The key idea is that humans are sensitive to *structural* changes in an image, which strongly correlates with their subjective evaluation of image quality. To infer structural changes, *DSSIM* captures patterns in pixel intensities, especially among neighboring pixels. The metric also captures luminance, and contrast measures of an image, that would also impact perceived image quality. *DSSIM* values fall in the range $[0, 1]$, where 0 means the image is identical to the original image, and a higher value means the perceived distortion will be higher. We include the mathematical formulation of *DSSIM* in the Appendix. We also refer interested readers to the original papers for more details [65, 66].

Solving the Optimization Function. To solve the optimization in Equation 1, we use the *penalty method* [43] to reformulate the optimization as follows.

$$\min \quad D(T_K(x'_s), T_K(x_t)) + \lambda \cdot (\max(d(x'_s, x_s) - P,\ 0))^2$$

Here λ is the penalty coefficient that controls the tightness of the privacy budget constraint. By gradually increasing λ, the final optimization result would converge to that of the original formulation. In our experiment, we empirically choose a λ large enough to ensure the perturbation constraint is tightly enforced.

We use Adadelta [69] optimizer to solve the reformulated optimization problem. To constrain input pixel intensity within the correct range ($[0, 255]$), we transform intensity values into *tanh* space [17].

4 Experimental Results

Next, we perform experiments across a number of classification tasks to validate the effectiveness of attacks on transfer learning. Given their wide adoption in a variety

of applications, we focus on image classification tasks, including facial recognition, iris recognition, traffic sign recognition and flower recognition.

4.1 Experimental Setup

Teacher and Student Models. We use four tasks and their associated Teacher models and datasets to build our victim Student models.

- **Face** Recognition classifies an image of a human face into a class associated with a unique individual. The Teacher is the popular 16 layer VGG-Face model [49] trained on a dataset of 2.6M images to recognize 2,622 faces. The Student model is trained using the PubFig dataset [8] to recognize a different set of 65 individuals[1]. The Student training dataset contains 90 faces belonging to each of the 65. The testing dataset for the Student model contains 650 images (10 images per class).
- **Iris** Recognition classifies an image of a human iris into one of many classes associated with different individuals. The Teacher model is a 16 layer VGG16 model trained on the ImageNet dataset of 1.3M images [56]. The Student model is trained on the CASIA IRIS dataset [2] containing 16,000 iris images associated with 1,000 individuals, and the testing dataset contains 4,000 images.
- **Traffic Sign** Recognition classifies different types of traffic signs from images, which can be used by self-driving cars to automatically recognize traffic signs. The Teacher model is again the 16 layers VGG16, trained on the ImageNet dataset. The Student is trained using the GTSRB dataset [1] containing 39,209 images of 43 different traffic signs. It also has a testing dataset of 12,630 images.
- **Flower** Recognition classifies images of flowers into different categories, and is a popular example of multi-class classification. It is also an example of transfer learning by Microsoft's CNTK framework [6]. The Teacher model is the ResNet50 model (with 50 layers) [28], trained on the ImageNet dataset. The Student is trained on the VGG Flowers dataset [9] containing 6,149 images from 102 classes, and comes with a testing dataset of 1,020 images.

These tasks represent typical scenarios users may face during transfer learning. First, the training dataset for building the Student model is significantly smaller than that of the Teacher's training dataset, which is a common scenario for transfer learning. Second, the Teacher and Student models either target similar tasks (Face Recognition) or very different tasks (Flowers and Traffic Sign Recognition). Finally, Face, Iris and Traffic sign recognition are security-related tasks. More details of training the Student models are listed in Table 2 in the Appendix.

[1] The original dataset contains 83 celebrities. We exclude 18 celebrities that were also used in the Teacher model.

Optimizing Student Models. We apply all three transfer learning approaches (discussed in Section 2) to each task, and identify the best approach. Table 1 shows the classification accuracy for different transfer approaches. For *Mid-layer Feature Extractor*, we show the result of the best Student model by experimenting with all possible K values. The results show that Face Recognition achieves the highest accuracy (98.55%) when using *Deep-layer Feature Extractor*. This is expected as the Student and Teacher tasks are very similar, leading to significant gains from transferring knowledge directly. The Flower classification task performs best with *Full Model Fine-tuning*, since the Student and Teacher tasks are different and there is less gain from sharing layers. Lastly, Traffic Sign recognition is a nice example for transferring knowledge from a middle layer (layer 10 out of 16).

Based on these results, we build the Student model for each task using the transfer method that achieves the highest classification accuracy (marked in bold in Table 1). The resulting four Student models cover all three transfer learning methods.

Attack Configuration. We craft adversarial samples using correctly classified images in the test dataset. These are images not seen by the Student model during its training and matches our attack model, *i.e.* the adversary has no access to the Student training dataset. To evaluate targeted attacks, we randomly sample 1K source and target image pairs to craft adversarial samples, and measure the attack success rate as the percentage of attack attempts (out of 1K) that misclassify the perturbed source image as the target. For non-targeted attacks, we randomly select 1K source images and 5 target images for each source (to guide the optimization process). Success for non-targeted attack is measured as the percentage of 1K source images that are successfully misclassified into any other arbitrary class.

For each source and target image pair, we compute the adversarial samples by running the Adadelta optimizer over 2,000 iterations with a learning rate of 1. For all the Teacher models considered in our experiments, the entire optimization process for a single image pair takes roughly 2 minutes on an NVIDIA Titan Xp GPU.

We implement the attack using Keras [19] and TensorFlow [12], leveraging open-source implementations of misclassification attacks provided by prior works [44, 17].

Student Task	Transfer Process		
	Deep-layer Feature Extractor	*Mid-layer Feature Extractor*	*Full Model Fine-tuning*
Face	**98.55**%	98.00% (14/16)	75.85%
Iris	**88.27**%	88.22% (14/16)	81.72%
Traffic Sign	62.51%	**96.16**% (10/16)	94.39%
Flower	43.63%	92.45% (10/50)	**95.59**%

Table 1: Transfer learning performance for different tasks when using different transfer processes. For each task, we select the model with the highest accuracy as our target Student model in future analysis. Numbers in parenthesis under *Mid-layer Feature Extractor* are the number of layers copied to achieve the corresponding accuracy, as well as the total number of layers of the Teacher.

Figure 3: Examples of adversarial images on Face Recognition ($P = 0.003$).

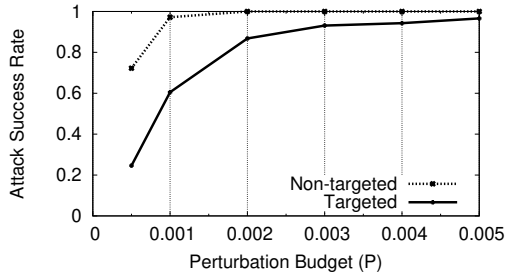


Figure 4: Attack success rate on Face Recognition with different perturbation budgets.

4.2 Effectiveness of the Attack

We first evaluate the proposed attacks assuming the attacker knows the exact transfer approach used to produce the Student model. This allows us to derive the upper bounds on attack effectiveness, and to explore the impact of the perturbation budget P, the distance metric $d(x'_s, x_s)$, and the underlying transfer method used to produce the Student model. Later in Section 4.3 we will relax this assumption.

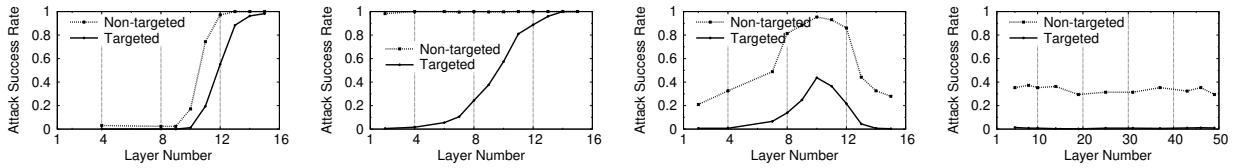
Consider the Face recognition task which uses *Deep-layer Feature Extractor* to produce the Student model. Here the attacker crafts adversarial samples to target the $N-1$ layer of the Teacher model. Even with a very low perturbation budget of $P = 0.003$, our attack is highly effective, achieving a success rate of 92.6% and 100% for targeted, and non-targeted attacks respectively. We also manually examine the perturbations added to adversarial images and find them to be undetectable by visual inspection. Figure 3 includes 6 randomly selected successful targeted attack samples for interested readers to examine.

It should be noted that an attacker could improve attack success by carefully selecting a source image similar to a target image. Our attack scenario is much more challenging, since the source and target images are randomly selected. Figure 3 shows that our attacks often try to mimic a female actress using a male actor, and vice versa. We also have image pairs with different lighting conditions, facial expressions, hair color, and skin tones. This significantly increases the difficulty of the targeted attack, given constraints on the perturbation level.

Impact of Perturbation Budget P. A natural question is how to choose the right perturbation budget, which directly affects the stealthiness of the attack. By measuring image distortion via the *DSSIM* metric, we empirically find that $P = 0.003$ is a safe threshold for facial images. Its corresponding L_2 norm value is 8.17, which is significantly smaller than/comparable to values used in prior work ($L_2 > 20$) [38].

Figure 4 shows the attack success rate as we vary the perturbation budget between 0.0005 and 0.005. As expected, smaller budget results in lower attack success rate, as there is less room for the attacker to change images and mimic the internal representation. Detailed comparison of images with different perturbation budgets is in Figure 10 in the Appendix.

Impact of Distance Metric $d(x'_s, x_s)$. Recall that we use *DSSIM* to measure perturbation added to input images, instead of the L_p distance used by prior works, *e.g.,* L_2. To compare both metrics, we also implement our attack using L_2 distance, and analyze the generated images ourselves. For a fair comparison, we choose an L_2 distance budget that produces a targeted attack success rate similar to using *DSSIM* with a budget of 0.003. Generated images are included in Figure 11 in the Appendix. We find that *DSSIM* generates imperceptible perturbations, while perturbations using L_2 are more noticeable. While *DSSIM* takes into account the underlying structure of an image, L_2 treats every pixel equally, and often generates noticeable "tattoo-like" patterns on faces.



(a) Face (b) Iris (c) Traffic Sign (d) Flower

Figure 5: Targeted and non-targeted attack success rate on Student models when targeting different layers. X axis indicates the layer being targeted. Face and Iris freeze the first 15 layers during training; Traffic Sign freezes the first 10 layers; Flower freezes no layers.

Impact of Transfer Method. We also test out attack on Iris, Traffic Sign, and Flower recognition tasks. Their perturbation budgets are set to 0.005 (L_2=9.035), 0.01 (L_2=7.77), and 0.003 (L_2=13.52), respectively. These values are empirically derived by the authors to produce unnoticeable image perturbations.

Overall, the attack is effective in Iris, with a targeted attack success rate of 95.9% and non-targeted success rate of 100%. Like Face recognition, the Iris student model was trained via *Deep-layer Feature Extractor*. On the other hand, the attack becomes less effective on Traffic Sign recognition, where the success rate of targeted and non-targeted attacks are 43.7%, and 95.35%, respectively. For Flower recognition, these numbers reduce to 1.1% and 37.25%, respectively. These results suggest that the attack effectiveness is strongly correlated with the transfer method: our attack is highly effective for *Deep-layer Feature Extractor*, but ineffective for *Full Model Fine-tuning*.

4.3 Impact of the Attack Layer

We now consider scenarios where the attacker does not know the exact transfer method used to train the Student model. In this case, the attacker needs to first select a Teacher layer to attack, which can be different from the deepest layer frozen during the transfer process. To understand the impact of such mismatch, we evaluate our attack on each of the Teacher layers in all four Student models. We organize our results by the transfer method.

Deep-layer Feature Extractor. The corresponding student models are Face and Iris. We set their perturbation budget P to 0.003, and 0.005, respectively (the same values used in the previous experiment). We launch attacks to each of the N-1 Teacher layers (N=16), *i.e.* computing adversarial samples that mimic the internal representation of the target image at layer K where $K = 1...N-1$. Figure 5(a) and Figure 5(b) show targeted and non-targeted success rates when attacking different layers.

For both Face and Iris, the attack is the most effective when targeting precisely the $N-1_{th}$ (15th) layer, which is as expected since both use *Deep-layer Feature Extractor*. As the attacker moves from deeper layers towards shallow layers (*i.e.* reducing K), the attack effectiveness reduces. At layer 13 and above, the attack success rates are above 88.4% for Face, and 95.9% for Iris. But when targeting layer 10 and below, the success rates drop to 1.2% for Face recognition, and <40% for Iris recognition. This is because shallow layers represent basic components of an image, *e.g.,* lines and edges, which are harder to mimic using a limited perturbation budget. In fact, both Face and Iris models are based on convolutional neural networks, which are known to capture such representations at shallow layers [70]. Therefore, given a fixed perturbation budget, the error in mimicking internal representations is much higher at shallow layers, resulting in lower attack success rates.

An unexpected result is that for Iris, the success rate for non-targeted attacks remains close to 100% regardless of the attack layer choice. A more detailed analysis shows that this is because Iris recognition is highly sensitive to input noise. The perturbations introduced by our attack behave as input noise, thus triggering misclassification into an "unknown" class. However, this is a unique property of the Iris recognition task, and does not apply to the other three tasks.

Mid-layer Feature Extractor. We then evaluate attack on Traffic Sign, where the first 10 layers are transferred from Teacher and frozen during training. Here the perturbation budget is fixed to $P = 0.005$. Results in Figure 5(c) show that the attack success rates peak at precisely the 10_{th} layer, where success rate for targeted attack is 43.7% and 95.35% for non-targeted attack. Similarly, the success rates reduce when the attacker targets shallow layers. Interestingly, the success rates also decrease as we target layers deeper than 10. This is because layers beyond 10 are fine-tuned and more distinct from the corresponding Teacher layers, leading to higher error when mimicking the internal representation.

Full Model Fine-tuning. For the Flower task, the Student model differs largely from the Teacher model, as all the layers are fine-tuned. Therefore, the attacker will always use incorrect information (from the Teacher) to mimic an internal representation of the Student. The resulting attack success rates are low and flat across the choice of attack layers (Figure 5(d) with $P = 0.003$).

How to Choose the Attack Layer? The above results suggest that the attacker should always try to identify if the Student is using *Deep-layer Feature Extractor*, as it remains the most vulnerable approach. In Section 5, we present a technique to determine whether *Deep-layer Feature Extractor* is used for transfer and to identify the Teacher model, using a few queries on the Student model. In this case, the attacker should focus on the $N-1^{th}$ layer to achieve the optimal attack performance.

If the Student is not using *Deep-layer Feature Extractor*, the attacker can try to find the optimal attack layer by iteratively targeting different layers, starting from the deepest layer. In the case of *Mid-layer Feature Extractor*, the attacker can estimate the attack success rate at each layer, using only a small set of image pairs and very limited queries. The attacker can observe the attack success rate increasing (or decreasing) as she approaches (or moves away from) the optimal layer.

4.4 Discussion

***Feature Extractor* vs. *Full Model Fine-tuning*.** Our results suggest that *Full Model Fine-tuning* and *Mid-layer Feature Extractor* lead to models that are more robust against our attacks. However, in practice, these two approaches are often not applicable, especially when the Student training data is limited. For example, for Face recognition, when reducing the training dataset from 90 images per class to 50 per class, pushing back by 2 layers (*i.e.* transfer at layer 13) reduces the model classification accuracy to 19.1%. Meanwhile, *Deep-layer Feature Extractor* still achieves a 97.69% classification accuracy. Apart from performance, these approaches also incur higher training cost than *Deep-layer Feature Extractor*. This is also why many deep learning frameworks today use *Deep-layer Feature Extractor* as the default configuration for transfer learning.

Can white-box attacks on Teacher transfer to student Models? Prior work identified the *transferability* of adversarial samples across different models for the same task [38]. Thus another potential attack on transfer learning is to use existing white-box attacks on the Teacher to craft adversarial samples, which are then transferred to the Student. We evaluate this attack using the state-of-the-art white-box attack by Carlini *et al.* [17]. Since Teacher and Student models have different class labels, we can only perform non-targeted attacks.

Our results show that the resulting attack is ineffective for all four tasks: only $< 0.3\%$ adversarial samples trigger misclassification in the Student models. Thus we confirm that the white-box attack on the Teacher will not be transferred to the Student. The failure of the attack can be attributed to the differences between the Teacher and Student tasks. The Student model has a different classification layer (and hence decision boundary) than the Teacher, so adversarial samples computed using decision boundary analysis (based on classification layer) of the Teacher model fail on the Student model.

5 Experiments with Real ML Services

So far our misclassification attacks assume that the teacher model is known to the attacker. Next, we relax this assumption by considering scenarios where the teacher model is unknown to the attacker. Specifically, today's deep learning services (*e.g.* Google Cloud ML, Facebook PyTorch, and Microsoft CNTK) already help customers generate student models from a suite of teacher models. In this case, a successful attack must first infer the teacher model given a student model. We address this challenge by designing a fingerprinting approach that feeds a few query images on the student model to identify the teacher model, allowing us to effectively attack the student models produced by today's deep learning services.

5.1 Fingerprinting the Teacher Model

Our design assumes that, given a student model, the attacker has access to the pool of candidate Teacher models where one of them is used to produce the student model. This is a practical assumption because for common deep learning tasks there are only a limited set of high quality, pre-trained models that are publicly available. For example, Google Cloud ML provides InceptionV3, MobileNets and its variants as Teacher models for image classification. Thus the attacker only needs to identify the Teacher from a (small) set of known candidates.

Methodology. We take a fingerprinting based approach. For each candidate Teacher model, the attacker crafts a fingerprint image that will intentionally "distort" the output of the student model, if and only if the student model is generated by the given Teacher model. By querying the student model with the fingerprinting images of all the candidates and comparing the model output, the attacker can quickly narrow down to the true Teacher model. In the following, we show that such fingerprinting method is highly effective when the student model is generated via *Deep-layer Feature Extractor*.

Consider the last layer of a student model (trained using *Deep-layer Feature Extractor*), which is a dense layer for classification. The prediction result (before softmax) of an input image x can be expressed as,

$$S(x) = W_N \times T_{N-1}(x) + B_N \quad (3)$$

where W_N is the weight matrix of the dense layer, B_N is the bias vector, and $T_{N-1}(.)$ is the function transforming the input x to neurons at layer $N-1$ [2].

[2] There will also be an activation function that further transforms

Given the knowledge of $T_{N-1}(.)$, our goal is to craft a fingerprinting image that nullifies the first term in Equation 3, *i.e.* an x that produces an all-zero vector $T_{N-1}(x) = \vec{0}$ so that the output vector $S(x) = B_N$. Since different Teacher models differ largely in $T_{N-1}(.)$, a fingerprinting image of a Teacher model A, when fed to a Student model derived from a different Teacher model B, is unlikely to produce an all-zero vector $T_{N-1}(x)$.

To decode the fingerprint, our hypothesis is that, without the contribution from x, the bias vector B_N (or $S(x)$ produced by the right fingerprint) will display much lower *dispersion* compared to normal $S(x)$ values. Thus by feeding candidate fingerprinting images into the student model and comparing the dispersion value of the corresponding $S(x)$, we can identify the Teacher model as the one that produces the minimum dispersion (below a threshold).

Assuming this hypothesis is true, we can craft fingerprinting images for each Teacher model following the same optimization process for our misclassification attack (see Section 3). The only difference is here the internal representation to mimic is a zero-vector.

Validation. To validate our approach, we produce five additional Student models using multiple popular public Teacher models [3]. These Student models are trained using the 17-class VGG Flower dataset [4], using *Deep-layer Feature Extractor*. Together with the Face and Iris models used in Section 4, we have a total of 7 Student models produced from different Teacher models. All of them achieve $> 83.1\%$ classification accuracy.

We measure the dispersion of $S(x)$ using the *Gini coefficient*, commonly used in economics to measure wealth distribution [26]. Its value ranges between 0 and 1, with 0 representing complete equality and 1 representing complete inequality.

We first measure the Gini coefficient of B_N, validating our hypothesis that B_N's dispersion level is very low. For each Student model, we set output neurons of $N-1_{th}$ layer as a zero vector, so that only B_N is fed into the final prediction. For all seven models, the corresponding Gini coefficient is below 0.011. We then feed 100 random test images into each model, where the Gini coefficient jumps to between 0.648 and 0.999, with a median value of 0.941. This confirms our hypothesis where B_N has a different statistical dispersion than normal $S(x)$.

Next, for each candidate Teacher model, we craft and feed 10 fingerprinting images to the target student model and compute the average Gini coefficient of $S(x)$. Figure 6 shows the average Gini coefficient as a function of the fingerprinting Teacher model and the Teacher model used to generate the Student model. The diagonal line indicates scenarios where the two Teacher models match. As expected, all the coefficients along the diagonal are small (< 0.058), suggesting that the fingerprinting images successfully nullify the neuron component in $S(x)$. All off-diagonal coefficients are significantly higher (> 0.443), since the Teacher model used to generate the fingerprinting image does not match that used to generate the student model.

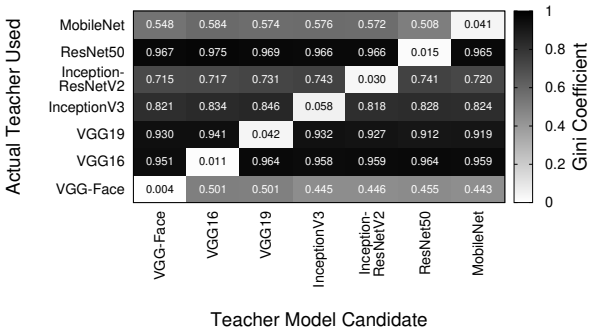


Figure 6: Gini coefficient of output probabilities of different teacher and student models.

It is worth noting that our fingerprinting technique can also identify different versions of Teacher models with the same architecture. To demonstrate this, we use Google's InceptionV3 model that has two versions (*i.e.* with different weights) released at different times.[5]. Our technique accurately distinguishes between these two versions, with a Gini coefficient < 0.075 when there is a match, and > 0.751 otherwise.

Overall, the above results confirm that our fingerprinting method can identify the Teacher model using a small set of queries. When crafting the fingerprinting image, a threshold of 0.1 on the Gini coefficient seems like a good cut-off to ensure successful fingerprinting.

Effectiveness on Other Transfer Methods. Our fingerprinting method is based on nullifying neuron contributions to the last layer of the Student model. It is effective when the student model is generated by *Deep-layer Feature Extractor*. The same set of fingerprinting images, when fed to student models generated by other transfer methods, will likely lead to higher Gini coefficients and fail to identify the Teacher model. For example, when fed to the Traffic Sign and Flower models, the Gini coefficient is always higher than 0.839.

On the other hand, when all the fingerprinting images

$S(x)$, but we ignore it for the sake of simplicity. Our methodology holds for any activation function.

[3] Our choice of Teacher models includes VGG16 [56], VGG19 [56], ResNet50 [28], InceptionV3 [59], Inception-ResNetV2 [58], and MobileNet [32].

[4] This is a smaller version of the full 102-class flower dataset we used in previous experiments [10].

[5] Version 2015-12-05 `http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz`, Version 2016-08-28 `http://download.tensorflow.org/models/inception_v3_2016_08_28.tar.gz`

lead to large Gini coefficient values, it means that either the Teacher model is unknown (not in the candidate pool), or the student model is produced by a transfer method other than *Deep-layer Feature Extractor*. For both cases, the misclassification attack will be less effective. The attacker can use this knowledge to identify and target student models that are the most vulnerable to the misclassification attack.

5.2 Attacks on Transfer Learning Services

Today, popular Machine Learning as a service (MLaaS) platforms [67] (*e.g.,* Google Cloud ML) and deep learning libraries (*e.g.,* PyTorch, Microsoft CNTK) already recommend transfer learning to their customers. Many provide detailed tutorials to guide customers through the process of transfer learning. We follow these tutorials to investigate whether the resulting Student models are vulnerable to our attacks. The adversarial samples generated on the three services are listed in Figure 13 in the Appendix.

Google Cloud ML. In this MLaaS platform, users can train deep learning models in the cloud and maintain it as a service. The transfer learning tutorial explains the process of using Google's InceptionV3 image classification model to build a flower classification model [5].

Specifically, the tutorial suggests *Deep-layer Feature Extractor* as the default transfer learning method, and the provided sample code does not offer control parameters or guidelines to use other transfer approaches or Teacher models (one has to modify the code to do so). We follow the tutorial to train a Student model on a 5-class flower dataset (the example dataset used in the tutorial), which achieves an 89.3% classification accuracy[6].

To launch the attack on the Student model, we first use the proposed fingerprinting method to identify that InceptionV3 (2015 version) is used as the Teacher model (*i.e.* the corresponding fingerprint image leads to Gini coefficient of 0.061 while the other fingerprint images lead to much higher values > 0.4063). The subsequent misclassification attack achieves a 96.5% success rate with $P = 0.001$.

Microsoft CNTK. The Microsoft Cognitive Toolkit (CNTK) is an open source DL library available on Microsoft's Azure MLaaS platform. The tutorial describes a flower classification task and recommends ResNet18 as the Teacher and *Full Model Fine-tuning* as the default configuration [6]. This creates a Student model similar to the Flower model used in Section 4. CNTK also provides control parameters to switch to *Deep-layer Feature Extractor* (*Mid-layer Feature Extractor* is unavailable) and other Teacher models hosted by Microsoft, including popular image classification models (*e.g.,* ResNet50, InceptionV3, VGG16) and a few object detection models. Following this process, we use VGG16 as the Teacher and *Deep-layer Feature Extractor* to train a new Student model using the 102-class VGG flower dataset (the example dataset used in tutorial). It achieves a classification accuracy of 82.25%.

Again, we were able to launch the misclassification attack on the Student model: our fingerprinting method successfully identifies the Teacher model (with a Gini coefficient of 0.0045), and the attack success rate is 99.4% when $P = 0.003$.

PyTorch. PyTorch is a popular open source DL library developed by Facebook. Its tutorial describes steps to build a classifier that can distinguish between images of ants and bees [3]. The tutorial uses ResNet18 by default and allows both *Deep-layer Feature Extractor* and *Full Model Fine-tuning*, but indicates that *Deep-layer Feature Extractor* provides higher accuracy. There is no mention of *Mid-layer Feature Extractor*. PyTorch hosts a repository of 6 image classification Teacher models that users can plug into their transfer process.

Again we follow the tutorial and verify that Student models trained using *Deep-layer Feature Extractor* on PyTorch are vulnerable. Our fingerprinting technique produces a Gini coefficient of 0.004, and targeted attack achieves a success rate of 88.0% with $P = 0.001$. We also test our attack on a student model trained using *Full Model Fine-tuning*. Surprisingly, our targeted attack still achieves an 87.4% success rate with $P = 0.001$. This is likely because the Student model is trained only for a short number of epochs (25 epochs) at a very low learning rate of 0.001, and thus the fine-tuning process introduces only small modification to the model weights.

Implications. Our experiments on the three machine learning services show that many Student models produced by these services are vulnerable to our attack. This is particularly true when users follow the default configuration in Google Cloud ML and PyTorch. Our attack is feasible because each service only hosts a small number of deep learning Teacher models, making it easy to get access to the (small) pool of Teacher models. Finally, by promoting the use of transfer learning, these platforms often *expose* their customers to our attack accidentally. For example, Google Cloud ML advertises customers who have successfully deployed models using their transfer learning service [4]. While we refrain from attacking such customer models for ethical reasons, such information can help attackers find potential victims and gain additional knowledge about the victim model. We discuss our efforts at disclosure in the Appendix.

[6] Instead of training the Student in the cloud, we build the model locally using Google TensorFlow using the same procedure [7].

6 Developing Robust Defenses

Having identified the practical impact of these attacks, the ultimate goal of our work is to develop robust defenses against them. Insights gained through our experiments suggest that there are multiple approaches to developing robust defenses against this attack. *First*, the effectiveness of attacks is heavily dependent on the level of perturbations introduced. Successful misclassification seems to be very sensitive to small changes made to the input image. Therefore, any defense that perturbs the adversarial sample before classification has a good chance of disrupting the attack. *Second*, attack success requires precise knowledge of the Teacher model used during transfer learning, *i.e.* the weights transferred to the Student model. Thus any deviations from the Teacher model could render the attack ineffective.

Here, we describe three different potential defenses that target different pieces of the Student model classification process. We discuss the strengths and limitations of each, and experimentally evaluate their effectiveness against the attack and impact on classification of non-adversarial inputs.

6.1 Randomizing Input via Dropout

Our first defense targets the sensitivity of adversarial samples to small changes. The intuition is that attackers have identified minimal alterations to the image that push the Student model over some classification boundary. By introducing additional random perturbations to the image before classification, we can disrupt the adversarial sample. Ideally, small perturbations could effectively disrupt adversarial attacks while introducing minimal impact on non-adversarial samples. In prior work, Carlini, *et al.* studied different defense mechanisms against attacks on DNNs [16], and found the most effective approach to be adding uncertainty to the prediction process [25].

Dropout Randomization. We add randomness to the prediction process by applying Dropout [57] at the input layer. This has the effect of dropping a certain fraction of randomly selected input pixels, before feeding the modified image to the Student model. We repeat this process 3 times for each image and use the majority vote as the final prediction result [7], or a random result if all 3 predictions are different.

We test this defense on all three tasks, Face, Iris, and Traffic Sign, by applying Dropout on test images as well as targeted and non-targeted adversarial samples [8]. The results for Face and Traffic Sign are highly consistent, so we only plot the results for Face in Figure 7, including classification accuracy on test images, and success rate of both targeted and non-targeted attacks. Results for Traffic Sign is in the Appendix as Figure 14. As the dropout ratio increases (*i.e.* more pixels dropped), both classification accuracy and attack success rate drops. In general, the defense is effective against targeted misclassification, which drops in success rate much faster than the corresponding drop in classification accuracy, *e.g.* at dropout ratio near 0.4, classification accuracy drops to 91.4% while targeted attack success rate drops to 30.3%. However, non-targeted attacks are less affected, and attack success consistently remains higher than classification accuracy of normal samples, *e.g.* 92.47% when the classification accuracy is 91.4%. Finally, as dropout increases, it eventually disrupts the entire classification process, reducing classification accuracy while boosting misclassification errors (non-targeted misclassification).

This defense is ineffective on the Iris task. Recall that this model is sensitive to noise in general. The inherent sensitivity leads classification accuracy to drop at nearly the same rate as attack success rate. When dropping only 2% pixels, model accuracy already drops to 51.93%, while targeted attack success rate is still 55.5% and non-targeted attack success rate is 100%. Detailed results are shown in the Appendix as Figure 14. Clearly, randomization as defense is limited by the inherent sensitivity of the model. It is unclear whether the situation could by improved by retraining the Student model to be more resistant to noise [72].

Strengths and Limitations. The key benefit of this approach is that it can be easily deployed, without requiring changes to the underlying Student model. This is ideal for Student models that are already deployed. However, this approach has three limitations. *First*, there is a non-negligible hit on model accuracy for any significant reduction in attack success. This may be unacceptable for some applications (*e.g.,* authentication systems based on Face recognition). *Second*, this approach is impractical for highly sensitive classification tasks like Iris recognition. *Finally*, this approach is not resistant to countermeasures by the attacker. An attacker can circumvent this defense by adding a Dropout layer into the adversarial image crafting pipeline [16]. The generated adversarial samples would then be more robust to Dropout.

6.2 Injecting Neuron Distances

The attack we identified leverages the similarity between matching layers in the Teacher and Student models to mimic an internal representation of the Student. Thus, if we can make the Student's internal representation deviate from that of the Teacher for all inputs, the attack would be less effective. One way to do that is by modifying weights of different layers of the Student. In this section, we present a scheme to modify the Student layers

[7] We tested and found little improvement beyond 3 repetitions.

[8] We choose adversarial samples from Section 4.3 that achieve the highest attack success rate.

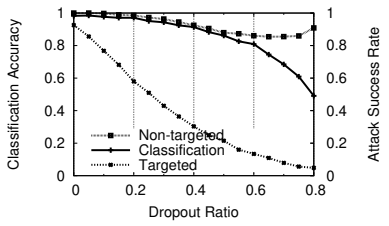


Figure 7: Attack success and classification accuracy on Face using randomization via dropout.

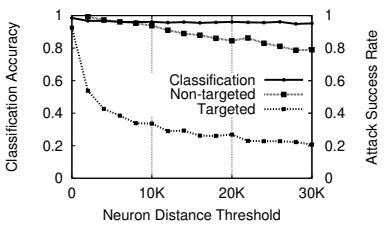


Figure 8: Attack success and classification accuracy on Face using neuron distance thresholds.

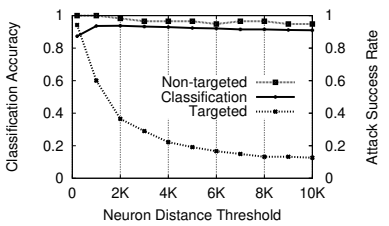


Figure 9: Attack success and classification accuracy on Iris using neuron distance thresholds.

(*i.e.* weights), without significantly impacting classification accuracy.

We start with a Student model trained using *Deep-layer Feature Extractor* or *Mid-layer Feature Extractor* [9]. This model lies in some local optimum of the model classification error surface. Our goal is to update layer weights and identify a new local optimum that provides comparable (or better) classification performance, and also be distant enough (on the error surface) to increase the dissimilarity between the Student and Teacher.

To find such a new local optimum, we unfreeze all layers of Student and retrain the model using the same Student training dataset, but with an updated loss function formulated in the following way. Consider a Student model, where the first K layers are copied from the Teacher. Let $T_K(.)$, and $S_K(.)$ be functions that generate the internal representation at layer K, for the Teacher, and Student, respectively. Let I be the set of neurons in layer K, and $|W_s|$ be a vector of absolute sum of outgoing weights from each neuron $i \in I$. Finally, let D_{th} be a dissimilarity threshold between two models. Then our objective is the following,

$$\begin{aligned} \min \quad & CrossEntropy(Y_{true}, Y_{pred}) \\ \text{s.t.} \quad & \sum_{x \in X_{train}} \| |W_s| \circ (T_K(x) - S_K(x)) \|_2 > D_{th} \end{aligned} \tag{4}$$

where $\circ$ is element-wise multiplication.

Here, we still want to minimize the classification loss, formulated as *cross entropy loss* over the prediction results. But, a constraint term is added to increase the dissimilarity between the Teacher and Student models. Dissimilarity is computed as the weighted L_2 distance between the internal representations at layer K, and is conditioned to be higher than a threshold D_{th}. The weight terms capture the importance of a neuron output for the next layer [10]. This helps make sure that distance between important neurons contribute more to the total distance between representations. We solve the above constrained optimization problem using the same penalty method used in Section 3.

Before presenting our evaluation, we note two other aspects of the optimization process. *First*, our objective function only considers dissimilarity at layer K. However, after training with the new loss function, the internal representations at the preceding layers also become dissimilar. Hence, our approach would not only reduce attack effectiveness at layer K, but also at layers before it. *Second*, a high value for D_{th} would increase defense performance, but can also negatively impact classification accuracy. In practice, the provider can incrementally increase D_{th} as long as the classification accuracy is above an acceptable level.

We evaluated this approach on all three classification tasks. Figure 8 shows how classification accuracy and attack success vary when we increase D_{th} in Face. Attacks are targeted at layer $N-1$, as Face uses *Deep-layer Feature Extractor*. Unlike the Dropout based defense (Figure 7), this method results in a steadier classification accuracy, while attack success rate drops. As classification accuracy drops from 98.55% to 95.69%, targeted attack drops significantly, from 92.6% to 30.87%. Non-targeted attacks are still hard to defend against, dropping from 100% to only 91.45% under the same conditions. We also analyze attack success rates at layers below $N-1$, and observe it to be lower than rates observed in Figure 8. This indicates that our retraining scheme makes the Student model more distinctive from the Teacher model across all layers. Result for Traffic Sign is in the Appendix in Figure 15, and is highly consistent with Face.

We plot the Iris results in Figure 9. Important to note that this defense works significantly better for the Iris task than the Dropout scheme. Sensitivity of the Iris model actually means classification accuracy increased from 88.27% to 91.0% (retraining found a better local optimum), while targeted attack success dropped from 100% to 12.6%. Unfortunately, non-targeted attacks remain hard to defend against. Attack success rate only

[9] Recall that models using *Full Model Fine-tuning* are generally resistant to the attack.

[10] The weight terms are not required for layers, where all neuron outputs are treated equally, *e.g.,* convolutional layers.

falls to 94.83% for Iris, and remains consistently above classification accuracy.

Finally, we note that retrained models are also robust against the Teacher fingerprinting technique. When using the true Teacher model as candidate, the fingerprinting attack results in an average Gini coefficient of > 0.9846 for both Face and Iris models, which effectively obfuscates the true identity of the Teacher model.

Strengths and Limitations. This scheme provides significant benefits relative to the randomized dropout scheme. *First*, we obtain improved defense performance, *i.e.* reduce attack success without significantly degrading classification accuracy. *Second*, unlike the dropout defense, this scheme has no clear countermeasures. Attackers do not have access to the Student training dataset, and cannot replicate the updated Student using retraining. *Third*, this approach successfully obfuscates the identity of the Teacher model, making it significantly harder to launch the attack given a target Student model.

Finally, the only limitation of this method is that all Student models must be updated using our technique, incurring additional computational cost. Compared to normal Student training, which takes several minutes to complete (for Face), our implementation that trains Student models with a fixed neuron distance threshold incurs training time that is an order of magnitude larger. For the example that corresponds to a reduced attack success rate of 30.87% on Face, our defense scheme takes 2 hours. As a one time cost, it is a reasonable tradeoff for significantly improving security against adversarial attacks. Also, we expect that other standard techniques for speeding-up neural network training (*e.g.,* training over multiple GPUs), can further reduce the runtime.

6.3 Ensemble of Models as a Defense

Finally, we consider using orthogonal models as a defense for adversarial attacks against transfer learning. The intuition is to have the provider train multiple Student models, each from a separate Teacher model, and use them together to answer queries (*e.g.,* based on majority vote). Thus even if an attacker successfully fools a single Student model in the ensemble, the other models may be resistant (since the adversarial sample is always tailored to a specific Student model). This can be an effective defense, while only incurring an additional one time computational cost of training multiple Students. This idea has been explored before in related contexts [13].

It is unclear whether an adversary with knowledge of this defense can craft a successful countermeasure, by modifying the optimization function to trigger misclassification in all members of the ensemble. One possibility is to modify the loss term that captures dissimilarity in internal representations (Equation 1), to account for dissimilarity in all models by taking a sum. In fact, a recent work in a non transfer learning setting, and assuming a white-box victim model shows that it is possible to break defenses based on ensemble models. He *et al.* , successfully crafted adversarial samples that can fool an ensemble of models, by jointly optimizing misclassification objectives over all members of the ensemble [29]. We are investigating this as part of ongoing work.

7 Related Work

Transfer Learning. In a deep learning context, transfer learning has been shown to be effective in vision [18, 52, 51, 15], speech [34, 63, 30, 20], and text [33, 40]. Yosinski *et al.* compared different transfer learning approaches and studied their impact model performance [68]. Razavian *et al.* studied the similarity between Teacher and Student tasks, and analyzed its correlation with model performance [50].

Adversarial Attacks in Deep Learning. We summarized some prior work on adversarial attacks in Section 2. Prior work on white-box attacks formulate misclassification as an objective function, and use optimization techniques to design perturbation [60, 17]. Goodfellow *et al.* further reduced the computational complexity of the crafting process to generate adversarial samples at scale [36]. Papernot *et al.* proposed an approach that modifies the image pixel by pixel to minimize the amount of perturbation [47]. Similar to our methodology, Sabour *et al.* proposed a method that manipulates internal representation to trigger misclassification [53]. Still others studied the physical realizability of adversarial samples [55, 24, 35], and attacks that generate adversarial samples that are unrecognizable to humans [42].

Prior work on black box attacks query the victim DNN to gain feedback on adversarial samples and use responses to guide the crafting process [55]. Others use these queries to reverse-engineer the internals of the victim DNN [46, 62]. Another group of attacks do not rely on querying the victim DNN, but assume there exists another model which has similar functionalities as the victim DNN [38, 45, 61]. They rely on the "transferability" of adversarial samples between similar models.

Defenses. Defense against adversarial attacks in DL is still an open research problem. Recent work showed that state-of-the-art adversarial attacks can adapt and bypass most existing defense mechanisms [16, 14]. One approach is adversarial training, where the victim DNN is trained to recognize adversarial samples [60, 39]. Others tried to detect certain characteristics of adversarial samples, *e.g.,* sensitivity to model uncertainty, neuron value distribution [64, 31, 27, 37, 25]. Another defense, called gradient masking, aims to enhance a model by remov-

ing useful information in gradients, which is critical to white-box attacks [48]. Most existing defenses have been bypassed in literature, or shown ineffective against new attacks.

8 Conclusion

In this paper, we describe our efforts to understand the vulnerabilities introduced by the transfer learning model. We identify and experimentally validate a general attack on black-box Student models leveraging knowledge of white-box Teacher models, and show that it can be successful in identifying and exploiting Teacher models in the wild. Finally, we explore several defenses, including a neuron distance threshold technique that is highly effective against targeted misclassification attacks while obfuscating the identity of Teacher models.

References

[1] http://benchmark.ini.rub.de/?section=gtsrb&subsection=news. The German Traffic Sign Recognition Benchmark.

[2] http://biometrics.idealtest.org/. CASIA Iris Dataset.

[3] http://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html. PyTorch transfer learning tutorial.

[4] https://cloud.google.com/blog/big-data/2017/08/how-aucnet-leveraged-tensorflow-to-transform-their-it-engineers-into-machine-learning-engineers. How Aucnet leveraged TensorFlow to transform their IT engineers into machine learning engineers.

[5] https://codelabs.developers.google.com/codelabs/cpb102-txf-learning/index.html#0. Image Classification Transfer Learning with Inception v3.

[6] https://docs.microsoft.com/en-us/cognitive-toolkit/Build-your-own-image-classifier-using-Transfer-Learning. Build your own image classifier using transfer learning.

[7] https://www.tensorflow.org/versions/r0.12/how_tos/image_retraining/. How to Retrain Inception's Final Layer for New Categories.

[8] http://vision.seas.harvard.edu/pubfig83/. PubFig83: A resource for studying face recognition in personal photo collections.

[9] http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html. 102 Category Flower Dataset.

[10] http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html. 17 Category Flower Dataset.

[11] http://www.robots.ox.ac.uk/~vgg/software/vgg_face/. VGG Face Descriptor.

[12] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[13] ABBASI, M., AND GAGNÉ, C. Robustness to adversarial examples through an ensemble of specialists. In *Proc. of Workshop on ICLR* (2017).

[14] ATHALYE, A., CARLINI, N., AND WAGNER, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of ICML* (2018).

[15] CAELLES, S., MANINIS, K.-K., PONT-TUSET, J., LEAL-TAIXÉ, L., CREMERS, D., AND VAN GOOL, L. One-shot video object segmentation. In *Proc. of CVPR* (2017).

[16] CARLINI, N., AND WAGNER, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. of AISec* (2017).

[17] CARLINI, N., AND WAGNER, D. Towards evaluating the robustness of neural networks. In *Proc. of S&P* (2017).

[18] CHEN, J.-C., RANJAN, R., KUMAR, A., CHEN, C.-H., PATEL, V. M., AND CHELLAPPA, R. An end-to-end system for unconstrained face verification with deep convolutional neural networks. In *Proc. of Workshop on ICCV* (2015).

[19] CHOLLET, F., ET AL. Keras. https://keras.io, 2015.

[20] CIREŞAN, D. C., MEIER, U., AND SCHMIDHUBER, J. Transfer learning for latin and chinese characters with deep neural networks. In *Proc of IJCNN* (2012).

[21] COHEN, R., EREZ, K., BEN AVRAHAM, D., AND HAVLIN, S. Breakdown of the internet under intentional attack. *Physical Review Letters 86* (2001), 3682–5.

[22] DELAMORE, B., AND KO, R. K. L. A global, empirical analysis of the shellshock vulnerability in web applications. In *Proc. of ISPA* (2015).

[23] ERHAN, D., MANZAGOL, P.-A., BENGIO, Y., BENGIO, S., AND VINCENT, P. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Proc. of AISTATS* (2009).

[24] EVTIMOV, I., EYKHOLT, K., FERNANDES, E., KOHNO, T., LI, B., PRAKASH, A., RAHMATI, A., AND SONG, D. Robust Physical-World Attacks on Deep Learning Models. In *arXiv preprint 1707.08945* (2017).

[25] FEINMAN, R., CURTIN, R. R., SHINTRE, S., AND GARDNER, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[26] GINI, C. Italian: Variabilità e mutabilità (variability and mutability). *Cuppini, Bologna* (1912).

[27] GROSSE, K., MANOHARAN, P., PAPERNOT, N., BACKES, M., AND MCDANIEL, P. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[28] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proc. of CVPR* (2016).

[29] HE, W., WEI, J., CHEN, X., CARLINI, N., AND SONG, D. Adversarial example defenses: Ensembles of weak defenses are not strong. In *Proc. of USENIX Workshop on Offensive Technologies* (2017).

[30] HEIGOLD, G., VANHOUCKE, V., SENIOR, A., NGUYEN, P., RANZATO, M., DEVIN, M., AND DEAN, J. Multilingual acoustic models using distributed deep neural networks. In *Proc. of ICASSP* (2013).

[31] HENDRYCKS, D., AND GIMPEL, K. Early methods for detecting adversarial images. In *ICLR Workshop Track* (2017).

[32] HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M., AND ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[33] JOHNSON, M., SCHUSTER, M., LE, Q. V., KRIKUN, M., WU, Y., CHEN, Z., THORAT, N., VIÉGAS, F., WATTENBERG, M., CORRADO, G., ET AL. Google's multilingual neural machine translation system: enabling zero-shot translation. In *Proc. of ACL* (2017).

[34] KUNZE, J., KIRSCH, L., KURENKOV, I., KRUG, A., JOHANNSMEIER, J., AND STOBER, S. Transfer learning for speech recognition on a budget. In *Proc. of RepL4NLP* (2017).

[35] KURAKIN, A., GOODFELLOW, I., AND BENGIO, S. Adversarial examples in the physical world. In *Proc. of ICLR* (2016).

[36] KURAKIN, A., GOODFELLOW, I., AND BENGIO, S. Adversarial machine learning at scale. In *Proc. of ICLR* (2017).

[37] LI, X., AND LI, F. Adversarial examples detection in deep networks with convolutional filter statistics. *arXiv preprint arXiv:1612.07767* (2016).

[38] LIU, Y., CHEN, X., LIU, C., AND SONG, D. Delving into transferable adversarial examples and black-box attacks. In *Proc. of ICLR* (2016).

[39] METZEN, J. H., GENEWEIN, T., FISCHER, V., AND BISCHOFF, B. On detecting adversarial perturbations. In *Proc. of ICLR* (2017).

[40] MIKOLOV, T., LE, Q. V., AND SUTSKEVER, I. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* (2013).

[41] MOOSAVI-DEZFOOLI, S.-M., FAWZI, A., AND FROSSARD, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proc. of CVPR* (2016).

[42] NGUYEN, A., YOSINSKI, J., AND CLUNE, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. of CVPR* (2015).

[43] NOCEDAL, J., AND WRIGHT, S. Numerical optimization, series in operations research and financial engineering. *Springer, New York, USA, 2006* (2006).

[44] PAPERNOT, N., CARLINI, N., GOODFELLOW, I., FEINMAN, R., FAGHRI, F., MATYASKO, A., HAMBARDZUMYAN, K., JUANG, Y.-L., KURAKIN, A., SHEATSLEY, R., GARG, A., AND LIN, Y.-C. cleverhans v2.0.0: an adversarial machine learning library. *arXiv* (2017).

[45] PAPERNOT, N., MCDANIEL, P., AND GOODFELLOW, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[46] PAPERNOT, N., MCDANIEL, P., GOODFELLOW, I., JHA, S., CELIK, Z. B., AND SWAMI, A. Practical black-box attacks against machine learning. In *Proc. of Asia CCS* (2017).

[47] PAPERNOT, N., MCDANIEL, P., JHA, S., FREDRIKSON, M., CELIK, Z. B., AND SWAMI, A. The limitations of deep learning in adversarial settings. In *Proc. of EuroS&P* (2016).

[48] PAPERNOT, N., MCDANIEL, P., WU, X., JHA, S., AND SWAMI, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. of S&P* (2016).

[49] PARKHI, O. M., VEDALDI, A., ZISSERMAN, A., ET AL. Deep face recognition. In *Proc. of BMVC* (2015).

[50] RAZAVIAN, A. S., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proc. of Workshop on CVPR* (2014).

[51] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You only look once: Unified, real-time object detection. In *Proc. of CVPR* (2016).

[52] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. of NIPS* (2015).

[53] SABOUR, S., CAO, Y., FAGHRI, F., AND FLEET, D. J. Adversarial manipulation of deep representations. In *Proc. of ICLR* (2015).

[54] SAROIU, S., GUMMADI, K., AND GRIBBLE, S. D. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (2002).

[55] SHARIF, M., BHAGAVATULA, S., BAUER, L., AND REITER, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. of CCS* (2016).

[56] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[57] SRIVASTAVA, N., HINTON, G. E., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *JMLR 15*, 1 (2014), 1929–1958.

[58] SZEGEDY, C., IOFFE, S., VANHOUCKE, V., AND ALEMI, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI* (2017).

[59] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., RABINOVICH, A., ET AL. Going deeper with convolutions. In *Proc. of CVPR* (2015).

[60] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[61] TRAMÈR, F., PAPERNOT, N., GOODFELLOW, I., BONEH, D., AND MCDANIEL, P. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453* (2017).

[62] TRAMÈR, F., ZHANG, F., JUELS, A., REITER, M., AND RISTENPART, T. Stealing machine learning models via prediction apis. In *Proc. of USENIX Security* (2016).

[63] WANG, D., AND ZHENG, T. F. Transfer learning for speech and language processing. In *Proc. of APSIPA* (2015).

[64] WANG, Q., GUO, W., ZHANG, K., ORORBIA II, A. G., XING, X., LIU, X., AND GILES, C. L. Adversary resistant deep neural networks with an application to malware detection. In *Proc. of KDD* (2017).

[65] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing 13*, 4 (2004), 600–612.

[66] WANG, Z., SIMONCELLI, E. P., AND BOVIK, A. C. Multiscale structural similarity for image quality assessment. In *ACSSC* (2003), vol. 2, IEEE, pp. 1398–1402.

[67] YAO, Y., XIAO, Z., WANG, B., VISWANATH, B., ZHENG, H., AND ZHAO, B. Y. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proc. of IMC* (2017).

[68] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Proc. of NIPS* (2014).

[69] ZEILER, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[70] ZEILER, M. D., AND FERGUS, R. Visualizing and understanding convolutional networks. In *Proc. of ECCV* (2014).

[71] ZHANG, L., CHOFFNES, D., DUMITRAS, T., LEVIN, D., MISLOVE, A., SCHULMAN, A., AND WILSON, C. Analysis of ssl certificate reissues and revocations in the wake of heartbleed. In *Proc. of IMC* (2014).

[72] ZHENG, S., SONG, Y., LEUNG, T., AND GOODFELLOW, I. Improving the robustness of deep neural networks via stability training. In *Proc. of CVPR* (2016).

Student Task	Dataset	# of Classes	Training Size	Testing Size	Teacher Model	Training Configurations
Face	PubFig83 [8]	65	5,850	650	VGG-Face [11]	epoch=200,batch=32,optimizer=adadelta,lr=1.0
Iris	CASIA Iris [2]	1,000	16,000	4,000	VGG16 [56]	epoch=100,batch=32,optimizer=adadelta,lr=0.1
Traffic Sign	GTSRB [1]	43	39,209	12,630	VGG16 [56]	epoch=50,batch=32,optimizer=adadelta,lr=1.0
Flower	VGG Flowers [9]	102	6,149	1,020	ResNet50 [28]	epoch=150,batch=50,optimizer=sgd,lr=0.01

Table 2: Detailed information about dataset, Teacher models, and training configurations for each Student task.

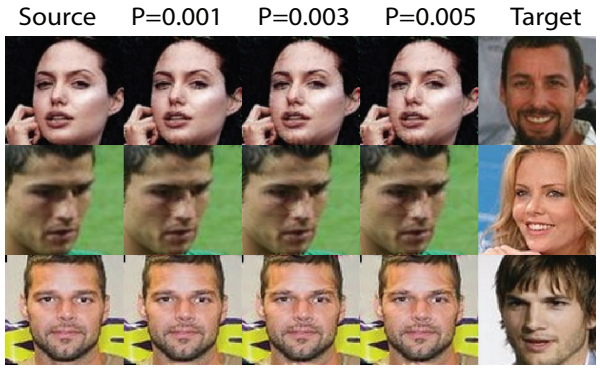


Figure 10: Adversarial examples generated from the same source image with different perturbation budgets (using *DSSIM*). Lower budget produces less noticeable perturbations.

Figure 11: Comparison between adversarial images generated using *DSSIM* perturbation budget ($P = 0.003$) and L_2 budget ($P = 0.01$). Budgets of both metrics are chosen to produce similar targeted attack success rate around 90%.

A Appendix

Disclosure

While we did not perform any attacks on deployed image recognition systems, we did experiment with publicly available Teacher models from Google, Microsoft and the open source PyTorch originally started by Facebook. Following their tutorials, our results showed they were vulnerable to this class of adversarial attacks. In advance of the public release of this paper, we reached out to machine learning and security researchers at Google, Microsoft and Facebook, and shared our findings with them.

Definition of *DSSIM*

DSSIM (Structural Dissimilarity) is a distance metric derived from *SSIM* (**S**tructural **SIM**ilarity). Let $x = \{x_1, ..., x_N\}$, and $y = \{y_1, ..., y_N\}$ be pixel intensity signals of two images being compared, respectively. The basic form of *SSIM* compares three aspects of the two image samples, luminance (l), contrast (c), and structure (s). The *SSIM* score is then described in the following equation.

$$\begin{aligned} SSIM(x,y) &= l(x,y) \cdot c(x,y) \cdot s(x,y) \\ &= \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \\ &\cdot \left(\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) \\ &\cdot \left(\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}\right) \end{aligned} \tag{5}$$

μ and σ are mean and standard deviation of pixel intensities of image samples. C_1, C_2, and C_3 are constants, and recommendation for choosing these constants is included in the original paper [65, 66].

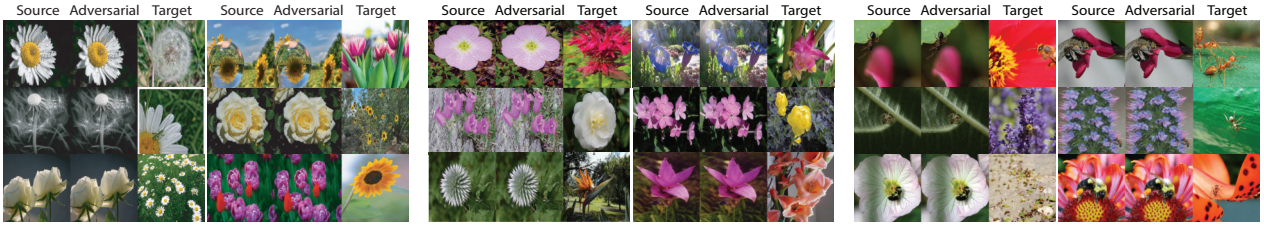
DSSIM is calculated as $\frac{1-SSIM}{2}$. It ranges from 0 to 1, where 0 represents two images are identical, and 1 represents two images are negatively correlated (often achieved by inverting the image).

In our experiments, we use an improved version of *SSIM*, referred as multi-scale *SSIM*, which also considers distortion due to viewing conditions (*e.g.,* display resolution). This is achieved by iteratively comparing the reference and distorted images at different scales (or resolutions) by applying a low-pass filter to downsample images. To compute *DSSIM*, we use the implementation of multi-scale *SSIM* from TensorFlow and follow the recommended parameter configuration [11].

[11] https://github.com/tensorflow/models/blob/master/research/compression/image_encoder/msssim.py

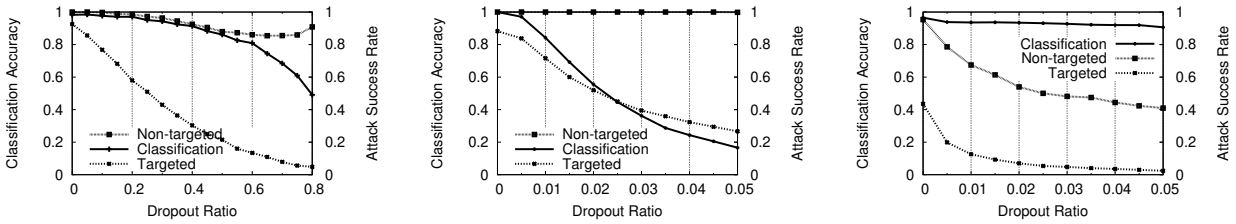
(a) Iris ($P = 0.005$) (b) Traffic Sign ($P = 0.01$) (c) Flower ($P = 0.003$)

Figure 12: Adversarial images generated in Iris, Traffic Sign, and Flower. Perturbation budgets selected result in unnoticeable perturbations. Iris attack targets at VGG16 layer 15 (out of 16 layers). Traffic Sign attack targets at VGG16 layer 10 (out of 16 layers), and Flower attack targets at ResNet50 layer 49 (out of 50 layers).



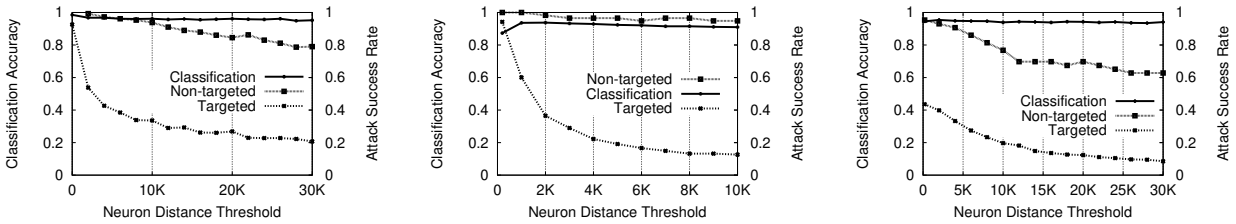
(a) Google Cloud ML ($P = 0.001$) (b) Microsoft CNTK ($P = 0.003$) (c) PyTorch ($P = 0.001$)

Figure 13: Adversarial images generated for Student models trained on Google Cloud ML, Microsoft CNTK, and PyTorch. Attacks using these samples achieve targeted success rate of 96.5%, 99.4%, and 88.0% in corresponding models.



(a) Face. (b) Iris. (c) Traffic Sign.

Figure 14: Performance of applying Dropout as defense with different Dropout ratio in Face, Iris, and Traffic Sign.



(a) Face. (b) Iris. (c) Traffic Sign.

Figure 15: Performance of modifying Student as defense with different distance thresholds in Face, Iris, and Traffic Sign.