Generating Adversarial Examples with Controllable Non-transferability

Renzhi Wang\textsuperscript{1}, Tianwei Zhang\textsuperscript{2}, Xiaoifei Xie\textsuperscript{2}, Lei Ma\textsuperscript{3}, Cong Tian\textsuperscript{1}, Felix Juefei-Xu\textsuperscript{4}, Yang Liu\textsuperscript{2}
\textsuperscript{1}Xidian University \textsuperscript{2}Nanyang Technological University \textsuperscript{3}Kyushu University \textsuperscript{4}Alibaba Group
\textsuperscript{1}wangrenzhi@stu.xidian.edu.cn,\textsuperscript{2}ctian@mail.xidian.edu.cn \textsuperscript{3}(tianwei.zhang,xfxie,yangliu)@ntu.edu.sg \textsuperscript{4}juefei.xu@gmail.com

\textbf{Abstract}—Adversarial attacks against Deep Neural Networks have been widely studied. One significant feature that makes such attacks particularly powerful is \textit{transferability}, where the adversarial examples generated from one model can be effective against other similar models as well. A large number of works have been done to increase the transferability. However, how to decrease the transferability and craft malicious samples \textit{only} for specific target models are not explored yet.

In this paper, we design novel attack methodologies to generate adversarial examples with controllable non-transferability. With these methods, an adversary can efficiently produce precise adversarial examples to attack a set of target models he desires, while keeping benign to other models. The first method is \textit{Reversed Loss Function Ensemble}, where the adversary can craft qualified examples from the gradients of a reversed loss function. This approach is effective for the white-box and gray-box settings. The second method is \textit{Transferability Classification}: the adversary trains a transferability-aware classifier from the perturbations of adversarial examples. This classifier further provides the guidance for the generation of non-transferable adversarial examples. This approach can be applied to the black-box scenario. Evaluation results demonstrate the effectiveness and efficiency of our proposed methods. This work opens up a new route for generating adversarial examples with new features and applications.

\section{I. INTRODUCTION}

The past decade has witnessed the revolutionary development of Deep Learning (DL) technology. A variety of DL algorithms and Deep Neural Network (DNN) models are designed to enable many artificial intelligent tasks, especially in the safety- and security-critical domains, \textit{e.g.,} autonomous driving, security authentication, intrusion detection, \textit{etc}. As a result, DL models are expected to meet high requirements of robustness and security. Unfortunately, DNNs are well-known to be vulnerable to adversarial examples (AEs): with imperceptible modifications to the input sample, the model will be fooled to give wrong prediction results. Since its discovery in 2013 \cite{1}, a large amount of efforts have been spent to launch and hunt the arms race between the adversarial attacks and defenses.

Adversarial examples enjoy one important feature: \textit{transferability} \cite{1}. AEs generated from one model has certain probability to fool other models as well. Due to this transferability, it becomes feasible for an adversary to attack black-box models, as he can generate AEs from an alternative white-box model and then transfer it to the target model. As a result, extensive works have been focusing on increasing AE’s transferability \cite{2}, \cite{3}, \cite{4}, explaining the mechanism and effects of transferability \cite{5}, \cite{6}, \cite{7}, \cite{8}, and trying to attack in real world\cite{9}.

Different from those studies, we aim to explore how to decrease the transferability and generate AEs only for certain DL models desired by the adversary. Generating AEs with such controllable non-transferability can enable the adversary to precisely attack his targets without affecting other irrelevant or his own applications. For instance, Autonomous Vehicles (AVs) adopt state-of-the-art DNN models for objection recognition and motion planning. A malicious AV vendor may want to attack its competitor’s products. He can generate AEs only for the target model, and physically install them in the real-world scenarios. Then the vehicles from its rival company will malfunction when meeting the AEs, while its own vehicles can behave correctly.

There are two challenges towards this goal. First, transferability is a nature feature for AEs. It is hard to efficiently craft AEs to minimize their damages on specific protected models, while still maintaining high success rate on specific target models. Second, we need to consider different attack settings. When the adversary has little knowledge about the details and parameters of target or protected models, it is also difficult for him to generate accurate AEs to achieve his goal. To the best of our knowledge, there is currently only one study about this non-transferability feature \cite{10}. However, this work only considers the white-box setting. The generated AEs are not controllable: the attack aims to decrease the transferability over all models instead of some specific protected ones. Besides, it is unknown whether this approach is applicable to models with different structures.

In this work, we propose two attack methodologies to generate adversarial examples with controllable non-transferability. The first one is \textit{Reversed Loss Function Ensemble}, for the white-box or gray-box attack settings. We design a new loss function which considers the impacts of both the target models, and the protected models (in a reversed fashion). Optimization of this loss function can give very precise AEs with success rates of higher than 95\% on the target models, and lower than 1\% on the protected models. The second solution is \textit{Transferability Classification}, for the black-box attack setting. We find that the adversarial perturbations with different transferability results have distinct characteristics. Based on this observation, we build a transferability-aware classifier, and generate AEs.
efficiently from it to meet our demands. The adversary can reduce the non-transferability across different models as low as 12%, even he has very limited knowledge about those models.

II. RELATED WORKS

A. Generating AEs with High Transferability

The transferability feature of AEs was first identified with the L-BFGS method in [1]. Since then, a number of works have been done to increase the transferability of the gradient-based methods for AE generation. MI-FGSM [2] adopted momentum optimization in the process of gradient generation to enhance the transferability. This helps it win the 1st place in both targeted and untargeted adversarial attack competitions of NeurIPS2017. Di²-FGSM [4] applied random transformations to the input at each iteration of AE generation to achieve high success rate on black-box models. TI-FGSM [3] enhanced the transferability of AEs towards the models with adversarial defenses. It identified the region which is less sensitive to white-box models but more sensitive to the target model, and adopted a pretrained matrix kernel to handle the gradients. These three methods are all based on FGSM, and can be combined to further improve the transferability. TAP [11] maximized the distance between clean samples and the corresponding AEs in the intermediate feature map, and adopted smooth regularization to improve the success rates on both white-box and black-box models.

B. Explanation and Understanding of Transferability

There are different points of view to understand the mechanism of transferability. The first one is from the model decision boundary. Liu et al. [12] proposed that transferability is caused by the similarity of decision boundaries of different models, even they have different network structures. An AE that can cross the boundary of one model has certain probability to cross the boundaries of other models as well. Tramer et al. [13] extended this hypothesis with the concept of adversarial subspace. Models with a higher-dimensional adversarial subspace have higher probabilities to coincide, thus the transferability is higher. The second perspective is from the data features. Ilyas et al. [14] proposed the concept of non-robust features that can be leveraged for AE generation. Different models for the same task can utilize similar non-robust features. As a result, AEs constructed from those features can be transferred across each other. The third direction is to identify the factors that can affect the transferability. Wu et al. [7] discovered that models with similar architecture, lower capability and high test accuracy have higher transferability. Papernot et al. [15] pointed out that transferability is not only limit to neural networks. They exist in statistic machine learning models (e.g., logistic regression, SVM, KNN, decision tree) as well.

III. PROBLEM STATEMENT AND THREAT MODEL

We aim to generate adversarial examples that can accurately fool a specific set of DL models, while remaining inoffensive to another set of models. Formally, we consider two sets of DL models: \(M_{\text{attack}}\) and \(M_{\text{protect}}\). All models in these two sets are used for the same task, but with different parameters, network structures or configurations. Given an input sample \(x\), our goal is to identify imperceptible perturbations \(\delta\) to satisfy the following properties:

\[
\begin{align*}
  f_i(x + \delta) &\neq f_i(x), \quad \forall f_i \in M_{\text{attack}} & (1a) \\
  f_i(x + \delta) &= f_i(x), \quad \forall f_i \in M_{\text{protect}} & (1b)
\end{align*}
\]

For the models in \(M_{\text{attack}}\), we consider an untargeted attack, where the misclassified label can be an arbitrary one different from the correct one. Targeted attacks can be conducted in a similar way. Our solutions are general-purpose and can be combined with various AE generation algorithms (e.g., FGSM [16], C&W [17], DeepFool [18]). How to design novel attack algorithms to evade state-of-the-art defenses is out of the scope of this paper.

We consider three typical attack scenarios [19]. For each scenario, we propose a attack method to meet our goal. (1) White-box attack: the adversary knows every detail about each model including all the parameters. We introduce a Reversed Loss Function Ensemble (RLFE) method to generate adversarial examples (Section IV-A). (2) Gray-box attack: the adversary knows all details of the models (e.g., training algorithms, network topology, hyper-parameters) except the parameters. He can train shadow models with the same configurations, and then apply the RLFE method on the shadow models to generate adversarial examples and attack the target models (Section IV-B). (3) Black-box attack: the adversary does not have any knowledge about the target models. This is the most restricted adversarial setting. We introduce a novel method, Transferability Classification (TC) to generate high-quality adversarial examples only by querying the black-box models (Section V).

IV. METHOD 1: REVERSED LOSS FUNCTION ENSEMBLE

A. Attacking White-box Models

We first consider a white-box setting, where the adversary has full knowledge of the models in \(M_{\text{attack}}\) and \(M_{\text{protect}}\). This scenario occurs when the model owners adopt the DL models published online for free use. The adversary can calculate the gradients of the model parameters to generate precise non-transferable adversarial examples.

Recall that the generation of adversarial examples for one target model \(f\) can be formulated as an optimization problem in Equation 2(a). To solve this problem, one typical approach is to maximize the loss function in Equation 2(b), where \(\langle f(x)\rangle_j\) denotes the confidence score of the \(j\)-th output, and \(y\) denotes the correct label of \(x\).

\[
\begin{align*}
  \text{minimize } \|\delta\|, \quad \text{subject to } & f(x + \delta) \neq f(x) & (2a) \\
  L(f, x, y) = -\langle f(x)\rangle_y + \log \left( \sum_{j \neq y} \exp \langle f(x)\rangle_j \right) & (2b)
\end{align*}
\]

We introduce Reversed Loss Function Ensemble (RLFE) to handle multiple models from the attacked model set \(M_{\text{attack}}\)
and protected model set \( M_{\text{protect}} \). Specifically, we ensemble the loss function from each model into a combined function (Equation 3): the attacked models positively contribute to the total loss while the protected models have negative impacts on this function. By maximizing this loss function, the identified adversarial examples are guaranteed to mislead the attacked models while hardly affecting the outputs of the protected models.

\[
L^* = \sum_{f_i \in M_{\text{attack}}} \lambda_i \cdot L(f_i, x, y) - \sum_{f_j \in M_{\text{protect}}} \lambda_j \cdot L(f_j, x, y)
\] (3)

It is worth noting that RLFE is very accurate since the adversary has white-box accesses to the models' parameters and generates the precise perturbations on those models. So even the attacked and protected models have very minor differences, the adversary is still able to discover examples that can distinguish them. This is hard to achieve in gray-box or black-box scenarios, due to the limited adversarial capabilities.

### B. Attacking Gray-box Models

Next we consider the gray-box setting, where the adversary knows the training details and configurations of the models, but not the parameter values. This is the scenario when the model owner adopts the public DL algorithms or cloud services to train his own copy of models.

In this case, the adversary cannot directly employ RLFE to calculate the adversarial perturbations due to the lack of model parameters. Instead, he can first construct shadow models with the same algorithms and configurations to mimic the behaviors of the actual models. Then he can generate the adversarial examples using RLFE over the shadow models. These examples can be used to attack or protect the original models as well. Note that this solution can also be applied to the case where some of the models are white-box to the adversary while the rest are gray-box. The adversary only needs to construct the shadow models of the gray-box ones and use the parameters of white-box models directly when generating adversarial examples.

It is interesting that we utilize the transferability between shadow and actual models to generate non-transferable AEs between attacked and protected models. As a result, the quality of adversarial examples depends on two factors. The first factor is the similarity between the shadow models and actual models. A higher similarity can improve the transferability between the constructed shadow and actual models, making the results closer to the whit-box setting. Various model extraction techniques [20], [21], [22], [23] have been proposed, which can be leveraged by the adversary to build the shadow models. The adversary can use the same or similar training set to train replicas of the actual model to achieve the same performance. He can also synthesize a dataset by querying the actual model with certain samples, and use the confidence scores as the output. This has been demonstrated more effective and efficient [23].

The second factor is the distinction between the attacked and protected models. If the attacked and protected models have very similar behaviors and features (e.g., originating from the same training algorithms and settings), it is hard to construct shadow models to differentiate them. This is one limitation in the gray-box setting, due to the restricted capability. The more distinct are the structures of the attacked models from the protected ones, the more effective our approach will be.

### V. Method 2: Transferability Classification

Finally we consider the black-box scenario. Now the adversary does not have knowledge about the protected and attacked models, and cannot construct shadow models and then apply RLFE. We only assume the adversary can query those black-box models, i.e., sending arbitrary input and retrieving the corresponding output. One straightforward way is to keep generating adversarial examples and testing their transferability until a qualified one is discovered. This brute force method is not efficient since for each sample, the adversary has to try a lot of sessions before obtaining a good one. Instead, we aim to build a generator, that can produce the desired adversarial perturbations corresponding to a given input easily and promptly. To achieve this goal, we propose a novel method, Transferability Classification, which could precisely capture the transferability information between two models.

It has been demonstrated that the transferability of AEs is caused by the common non-robust features that are learned by different models [14]. Inspired by this, we further find that the non-transferability could also be controlled by identifying some specific features learned by one model but not by others. The key insight of our solution is that the adversarial perturbations exhibit certain inherent characteristics that can represent their transferability across different models. So we can build a transferability-aware classifier (TC) to identify the relationship and facilitate the generation of new perturbations based on our demands. Our approach consists of two steps: (1) training a TC; (2) generating non-transferable adversarial examples by attacking on this classifier. Fig. 1 illustrates the methodology overview.

#### A. Step 1: Training Transferability-aware Classifier

Without loss of generality, we consider two models: one is attacked model \( f_{\text{attack}} \) and another one is protected model \( f_{\text{protect}} \). So there are four categories of adversarial examples: (I) succeed in attacking both models; (II) succeed in attacking \( f_{\text{attack}} \) but fail to attack \( f_{\text{protect}} \); (III) succeed in attacking \( f_{\text{protect}} \) but fail to attack \( f_{\text{attack}} \); (IV) fail to attack both models. The goal of this step is to train classifiers, such that given an
adversarial perturbation, they are able to predict which type of attack results this perturbation belongs to without querying the models.

**Dataset preparation.** We need a dataset to train the TC. We prepare an assisted white-box model \( f_{\text{assist}} \) which performs the same task as \( f_{\text{protect}} \) and \( f_{\text{attack}} \). Then we generate a large quantity of adversarial examples from \( f_{\text{assist}} \) using conventional methods (We use BIM [24] to generate AEs in this paper, but other methods can be used in the same way). For each example, we label them as one of the four categories defined above by querying \( f_{\text{protect}} \) and \( f_{\text{attack}} \). Then we are able to synthesize a four-class dataset with the added perturbation \( \delta \) as the feature and the attack result as the label. Note that this process can give unbalanced samples and make the classifier biased. We prune the dataset to guarantee each class has the same number of samples.

**Classifier training.** We split this dataset as training set and validation set, and train classifiers over them. We adopt a DNN model, Resnet18 as the structure of our classifier. Other networks can be used as well. Detailed experimental configurations can be found in Section VI-A. We consider two strategies: the first one is to train a unified classifier to predict the samples from all classes of the target models. The second one is to train a label-specific classifier for the samples from each class. Table I reports the training and testing accuracy of different combinations of white-box and target models. We can observe that each type of classifier can achieve a very satisfactory accuracy to differentiate four categories. The label-specific classifiers have better performance. So we will adopt this strategy in the follow evaluations.

**TABLE I: Training and testing accuracy of our Transferability-aware Classifier**

| White-box \( f_{\text{assist}} \) | Target \( f_{\text{attack}} \) | Target \( f_{\text{attack}} \) | Classifier Type | Train acc | Test acc |
|---|---|---|---|---|---|
| Resnet50 | VGG16BN | Densenet121 | Label-specific | 0.946 | 0.906 |
| VGG16BN | Resnet50 | Densenet121 | Label-specific | 0.908 | 0.922 |
| Densenet121 | Resnet50 | VGG16BN | Label-specific | 0.929 | 0.922 |
| | | | Unified | 0.791 | 0.185 |

**B. Step 2: Generating Adversarial Examples from Transferability-aware Classifier**

With the trained TC we can easily generate adversarial examples can only attack \( f_{\text{attack}} \) other than \( f_{\text{protect}} \). Specifically, given an input \( x \), we use conventional methods to generate the corresponding perturbation \( \delta \) which the TC can give correct prediction. If the perturbation can meet our demand \( \delta \), then we will select it as the final result. Otherwise, we treat this perturbation \( \delta \) as the input sample of the TC, and use conventional methods to generate a second order of perturbation \( \delta \) that can shift the prediction result to category II. Then we are able to get the final result \( x + \delta + \delta \) to satisfy our goal. We can use this adversarial sample to query the two models for validation. We repeat the above process in case the validation indicates the generated AE does not work as expected.

Note that once we obtain the TC, normally we need to run two generation sessions to identify the desired samples. The first session is to generate the perturbation \( \delta \) from the assisted model \( f_{\text{assist}} \), which can converge in a few iteration. The second session is to generate the perturbation \( \delta \) from TC. This process needs dozens of iterations with a smaller \( \epsilon \), but takes much shorter time due to the simple network structures of TC. In contrast, the brute force needs to run a large number of the first sessions to identify the satisfied AE. This makes our solution much more efficient.

We use two models to describe our methodology. It can be extended to multiple models: if there are \( n \) models in the attacked and protected model sets, we need to build TC with 2\( n \) labels to distinguish different sorts of perturbations, and generate the perturbations in the category of attacking all attacked models and bypassing all protected models.

**VI. Evaluation**

**A. Implementation and Experimental Configuration**

**Datasets and DL models.** We mainly consider three CNN models with different network structures: Resnet50 [25], VGG16 [26] and Densenet121 [27]. We select two representative datasets for image classification: Cifar10 [28] and Imagenet [29]. Other deep learning models and datasets can be attacked in the same way. Cifar10 dataset contains 50,000 training samples and 10,000 testing samples of 10 classes. Each image has the size of 32 \times 32 \times 3. For Imagenet, we select ILSVRC-2012 dataset, which contains more than 1.2 million training samples, 50,000 validation samples and 10,000 testing samples of 1,000 classes. The original images vary in size, and we reshape them to 224 \times 224 \times 3.

**Adversarial examples generation.** We adopt Basic Iterative Method (BIM) [24] to generate adversarial examples. It is the basis of new transferable adversarial attack techniques, e.g., MI-FGSM [2], TI-FGSM [3] DI\(^2\)-FGSM [4]. We set a high noise level \( \epsilon = 32 \) to increase the transferability of the generated adversarial examples. This can better reflect the effectiveness of our solutions.

We implemented our attacks with PyTorch 1.3.1. All the experiments in this paper were conducted on a server equipped with 2 Intel Xeon Sliver 4214 CPUs and 1 NVIDIA 2080Ti GPU.

**B. White-box Attacks**

**Different structures.** As the first study, we consider the attacked and protected models with different network structures. Table II shows the attack success rates of the attacked and protected models with combinations of various structures. We abbreviate Resnet50 as R50, VGG16 as V16, and Densenet121 as D121. For each entry, the left number denotes the success rates of the two models when we generate AEs from the attacked model and transfer them to the protected model (baseline). The right number (bold) denotes the success rates after we apply RLFE.

We get three observations from this table. First, RLFE gives very satisfactory results: it can significantly decrease the attack
success rates of protected models, while still maintaining high attack impacts on attacked models. Second, the attack results are better for Imagenet than Cifar10. The non-transferability is easier to achieve for more complicated images, as the search space is larger. Third, RLFE can effectively handle multiple models. We included two different models in the protected model set, and the adversary can generate the expected adversarial examples for both of the two models with very low success rates.

TABLE II: Attack success rates of different network structures under the white-box setting

(a) Cifar10

| Attacked Model Structure | Protected Model Structure | Succ. Rate (%) |
|--------------------------|--------------------------|---------------|
| R50                      | V16                      | 74.78 → 1.24  |
| R50                      | D121                     | 79.83 → 1.61  |
| R50                      | V16                      | 74.78 → 1.05  |
| R50                      | D121                     | 79.83 → 1.59  |
| V16                      | R50                      | 43.18 → 0.28  |
| V16                      | D121                     | 46.13 → 0.31  |
| V16                      | R50                      | 43.18 → 0.17  |
| V16                      | D121                     | 46.13 → 0.14  |
| D121                     | R50                      | 77.54 → 2.77  |
| D121                     | V16                      | 79.60 → 2.50  |
| D121                     | R50                      | 77.54 → 2.47  |
| D121                     | V16                      | 79.60 → 1.85  |

(b) Imagenet

| Attacked Model Structure | Protected Model Structure | Succ. Rate (%) |
|--------------------------|--------------------------|---------------|
| R50                      | V16                      | 50.28 → 0.58  |
| R50                      | D121                     | 51.31 → 0.19  |
| R50                      | V16                      | 50.28 → 0.70  |
| R50                      | D121                     | 51.31 → 0.20  |
| V16                      | R50                      | 37.93 → 0.22  |
| V16                      | D121                     | 39.20 → 0.11  |
| V16                      | R50                      | 37.93 → 0.20  |
| V16                      | D121                     | 39.20 → 0.13  |
| D121                     | R50                      | 52.83 → 0.29  |
| D121                     | V16                      | 51.42 → 0.05  |
| D121                     | R50                      | 52.83 → 0.28  |
| D121                     | V16                      | 51.42 → 0.04  |

Adversarial training. One possible defense against adversarial attacks is adversarial training [30], [31], [32], where AEs are used with normal samples together to train DNN models to recognize and correct malicious samples. However, the new model is still vulnerable to white-box attacks as the adversary can generate new AEs based on this model. We further show that with our RLFE method, the adversary can generate AEs to only attack the models with adversarial training protection, and protect the ones without AE protection.

Table III shows the transferability across attacked and protected models. We can see that even after the model is retrained with AEs, we can still generate new AEs to attack the new model with very high success rate. These new AEs have around 20% to 30% chances to affect the original model as well. After we apply RLFE, we can decrease the success rate on the original models to below 0.5%, without affecting the attack effects on the retrained models. This confirms that RLFE has the capability to attack the models with defenses and protect the models without defenses.

TABLE III: Attack success rates of models w/ and w/o adversarial protection under the white-box setting

(a) Cifar10

| Structure | Attacked Model (w/ adv. training) | Protected Model (w/o adv. training) |
|-----------|-----------------------------------|-------------------------------------|
| R50       | 99.94 → 98.31                     | 18.25 → 0.05                        |
| V16       | 98.93 → 98.77                     | 28.00 → 0.41                        |
| D121      | 99.95 → 99.45                     | 22.19 → 0.13                        |

(b) Imagenet

| Structure | Attacked Model (w/ adv. training) | Protected Model (w/o adv. training) |
|-----------|-----------------------------------|-------------------------------------|
| R50       | 99.98 → 97.09                     | 26.87 → 0.02                        |
| V16       | 97.30 → 73.04                     | 31.07 → 0.16                        |
| D121      | 95.89 → 96.33                     | 24.96 → 0.01                        |

C. Gray-box Attacks

Same model structure. Next we consider the gray-box attacks. We assume the adversary has white-box access to the attacked model, but gray-box access to the protected model. He needs to train a shadow model with the same network structure. In our experiment, we reset the logits layer’s weights and retrain the model for 100 epochs as the shadow model. Then the adversary can generate non-transferable AEs using RLFE. Table IV reports the attack success rate of the attacked model, and the protected model (not the shadow model).

We can observe that this RLFE with shadow model construction can significantly reduce the non-transferability as expected. The attack effects are not as good as the white-box attacks, as we generate the AEs from the shadow model instead of the protected model. Besides, we also find that the attack effects on Cifar10 is better than that on Imagenet. The reason is that the Imagenet models are more complicated than the Cifar10 models, so the difference between constructed shadow and original models is larger, making it harder to precisely protect the original models against AEs.

Same model family. We consider another case of gray-box scenario: the adversary does not know the network structure of the protected model, but he knows which family this model originates from. Then he can pick a random structure from the same family to train the shadow model. In our experiment, when the protected model is VGG16BN, the adversary can use its variant, VGG11BN or VGG19BN for shadow model construction. When the protected model is Resnet50, the adversary can choose Resnet18 or Resnet34 from the same family. Table V displays the attack results.
TABLE IV: Gray-box attacks: shadow model has the same structure as the protected model

(a) Cifar10

| Attacked Model Structure | Attacked Model Succ. Rate (%) | Protected Model Structure | Protected Model Succ. Rate (%) |
|--------------------------|-------------------------------|---------------------------|-------------------------------|
| R50                      | 99.61 → 99.44                | V16                       | 57.78 → 57.55                |
| R50                      | 99.61 → 99.28                | D121                      | 81.94 → 81.11                |
| V16                      | 97.98 → 97.67                | R50                       | 43.18 → 42.99                |
| V16                      | 97.98 → 97.71                | D121                      | 47.65 → 10.29                |
| D121                     | 99.15 → 94.16                | R50                       | 80.70 → 5.26                 |
| D121                     | 99.15 → 96.97                | V16                       | 82.96 → 8.33                 |

(b) Imagenet

| Attacked Model Structure | Attacked Model Accuracy | Protected Model Structure | Protected Model Accuracy |
|--------------------------|-------------------------|---------------------------|--------------------------|
| R50                      | 99.87 → 99.28           | V16                       | 50.28 → 26.34            |
| R50                      | 99.87 → 99.13           | R50                       | 81.50 → 31.17            |
| V16                      | 96.68 → 97.67           | D121                      | 37.93 → 22.48            |
| V16                      | 96.68 → 97.71           | R50                       | 60.40 → 28.58            |
| D121                     | 99.78 → 98.49           | R50                       | 84.05 → 20.61            |
| D121                     | 99.78 → 99.71           | V16                       | 86.55 → 27.05            |

We can still observe the effectiveness of our solutions for both high success rate of attacked models, and low success rate of protected models. The results are worse than the attacks with the same network structure. This is expected as different structures can give larger discrepancy between the shadow and original models, even they are from the same family.

TABLE V: Gray-box attacks: shadow model and protected model are from the same family

(a) Cifar10

| Attacked Model Structure | Attacked Model Succ. Rate (%) | Original Structure | Original Shadow Structure | Original Succ. Rate (%) | Original Shadow Succ. Rate (%) |
|--------------------------|-------------------------------|-------------------|--------------------------|-------------------------|-----------------------------|
| R50                      | 99.61 → 99.48                | V16               | V16                      | 57.78 → 41.82           |
| R50                      | 99.61 → 99.53                | V16               | V19                      | 54.78 → 30.74           |
| R50                      | 99.61 → 99.36                | V16               | V11&V19                  | 74.78 → 20.03           |
| V16                      | 97.98 → 97.77                | R50               | R18                      | 43.18 → 19.65           |
| V16                      | 97.98 → 97.87                | R50               | R34                      | 43.18 → 22.49           |
| V16                      | 97.98 → 97.74                | R50               | R18&R34                  | 43.18 → 12.65           |

(b) Imagenet

| Attacked Model Structure | Attacked Model Succ. Rate (%) | Original Structure | Original Shadow Structure | Original Succ. Rate (%) | Original Shadow Succ. Rate (%) |
|--------------------------|-------------------------------|-------------------|--------------------------|-------------------------|-----------------------------|
| R50                      | 99.87 → 95.50                | V16               | V11                      | 50.28 → 39.92           |
| R50                      | 99.87 → 99.64                | V16               | V19                      | 50.28 → 37.78           |
| R50                      | 99.87 → 99.42                | V16               | V11&V19                  | 50.28 → 32.96           |
| V16                      | 99.65 → 99.28                | R50               | R18                      | 37.93 → 32.23           |
| V16                      | 99.65 → 99.21                | R50               | R34                      | 37.93 → 31.98           |
| V16                      | 99.65 → 98.92                | R50               | R18&R34                  | 37.93 → 28.51           |

D. Black-box Attacks

We evaluate the black-box attacks with the Transferability Classification method. We set Densenet121 as the attacked black-box model and VGG16BN as the protected black-box model. The adversary adopts Resnet50 as the assisted white-box model to generate AEs and train the TC. As mentioned in section V-A, the classifiers are based on the Resnet18 structure. We train one classifier for each class.

Table VI lists the attack results. We show all 10 classes of Cifar10 and the first 10 classes of Imagenet. We observe that the success rate of the attacked model has certain decrease compared to white-box and gray-box attacks. This is because the adversary cannot access the attacked model directly. He has to identify the perturbation from the TC. This is already a good attack performance for black-box setting. Besides, the success rate of the protected model is reduced a lot, and there are big gaps between the attacked model and protected model. This proves the effectiveness of our Transferability Classification method, and the possibility of non-transferability adversarial attacks even all the models are black-box access to the adversary.

TABLE VI: Results of black-box attacks using Transferability-aware Classifier

(a) Cifar10

| category | Attacked Model | Protected Model |
|----------|----------------|-----------------|
| 0        | 82.86 → 54.08  | 83.24 → 13.86   |
| 1        | 87.72 → 52.90  | 87.54 → 11.99   |
| 2        | 76.98 → 47.21  | 75.70 → 15.04   |
| 3        | 71.90 → 53.57  | 69.04 → 14.93   |
| 4        | 75.60 → 52.14  | 70.72 → 14.94   |
| 5        | 75.10 → 60.27  | 72.12 → 14.46   |
| 6        | 83.58 → 53.28  | 79.80 → 13.77   |
| 7        | 85.38 → 53.90  | 82.56 → 14.05   |
| 8        | 89.12 → 53.59  | 86.90 → 15.05   |
| 9        | 80.18 → 48.48  | 85.52 → 18.19   |

(b) Imagenet

| category | Attacked Model | Protected Model |
|----------|----------------|-----------------|
| 0        | 53.22 → 47.24  | 61.39 → 23.31   |
| 1        | 49.73 → 51.99  | 48.08 → 18.20   |
| 2        | 65.52 → 54.29  | 58.45 → 21.92   |
| 3        | 49.73 → 44.21  | 48.68 → 19.24   |
| 4        | 47.38 → 49.10  | 51.10 → 17.77   |
| 5        | 58.03 → 52.91  | 46.99 → 18.97   |
| 6        | 62.31 → 57.31  | 58.33 → 25.05   |
| 7        | 48.46 → 42.91  | 59.32 → 18.28   |
| 8        | 55.89 → 50.45  | 48.49 → 19.49   |
| 9        | 55.11 → 57.91  | 66.66 → 21.44   |

VII. Conclusion

In this paper, we propose a new form of non-transferable adversarial examples, which can accurately attack a set of models without compromising another sets of models, controlled by the adversary. We introduce two methods to generate such samples under different attack scenarios. The first one is gradient-based approach: we identify a new loss function considering both attacked and protected models. Then AEs
can be generated by maximizing this function and calculating its gradients. The second one is perturbation-based approach: we build a transferability-aware classifier to predict the attack results of adversarial perturbations as well as generate desired AEs. Our methods can efficiently produce accurate AEs with the specified non-transferability. They are generous and can be integrated with existing AE generation approaches.

VIII. Broader Impact

In this work, we present the first systematic study about the non-transferability feature of adversarial examples. Although transferability makes AEs more severe and dangerous, non-transferability can also enhance adversarial attacks, by increasing the accuracy of target identification. This provides a brand new attack scenario, particularly useful for target-specific attacks between business competitors. Considering deep learning has been widely commercialized into many safety- and security-critical products (e.g., autonomous driving, face authentication, anomaly detection, home automation, etc.), we expect our methods can bring new challenges to deep learning applications in real world.

For research, we expect our work can open a new direction and draw researchers’ interests, just like the study of transferability over the past years. Specifically, we suggest some topics and research questions, which deserve in-depth exploration in the near future:

- Physical attacks: how can we generate physical perturbations with such non-transferability to attack real-world deep learning products?
- Advanced attacks: can we design advanced techniques to further decrease the transferability across specific models in a more efficient and scalable way?
- Defenses: can we prevent such highly targeted attacks, and thwart the generation of non-transferable AEs?
- Explanation and quantification: how can we explain the non-transferability feature? What properties of DNN models can impact non-transferability? What is the relationship between non-transferability and model similarity?

REFERENCES

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[2] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jiaming Li, Boosting adversarial attacks with momentum. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 9185–9193. IEEE Computer Society, 2018.

[3] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4312–4321, 2019.

[4] Chihang Xie, Zhishuai Zhang, Yuxin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2730–2739, 2019.

[5] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204, 2017.

[6] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. CoRR, abs/1802.00420, 2018.

[7] Lei Wu, Zhaxing Zhu, Cheng Tai, et al. Understanding and enhancing the transferability of adversarial examples. arXiv preprint arXiv:1802.09707, 2018.

[8] Xiuyue Zhang, Xiaolei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. arXiv preprint arXiv:2004.11577, 2020.

[9] Qing Guo, Felix Juefei-Xie, Xiaofei Xie, Lei Ma, Jian Wang, Wei Feng, and Yang Liu. Abba: Salient-regularized motion-based adversarial blur attack. arXiv preprint arXiv:2002.03500, 2020.

[10] Preetum Nakrani. A discussion of ‘adversarial examples are not bugs, they are features’. Adversarial examples are just bugs, too. Distill, 2019. https://distill.pub/2019/advers-exugs-discussion/response-5.

[11] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In Proceedings of the European Conference on Computer Vision (ECCV), pages 452–467, 2018.

[12] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. arXiv preprint arXiv:1611.02770, 2016.

[13] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. arXiv preprint arXiv:1704.03453, 2017.

[14] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In Advances in Neural Information Processing Systems, pages 125–136, 2019.

[15] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277, 2016.

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57. IEEE, 2017.

[18] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2574–2582, 2016.

[19] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 135–147, 2017.

[20] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. CoRR, abs/1609.02943, 2016.

[21] Binghui Wang and Niel Zhenqiang Gong. Stealing hyperparameters in machine learning. In 2018 IEEE Symposium on Security and Privacy (SP), pages 36–52. IEEE, 2018.

[22] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In Proceedings of Network and Distributed Systems Security Symposium (NDSS), 2020.

[23] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural network weights. In 29th {USENIX} Security Symposium ({USENIX} Security 20), 2020.

[24] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. CoRR, abs/1607.02533, 2016.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

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

[27] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical Report, 2009.
[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

[30] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016.

[31] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. arXiv preprint arXiv:1511.03034, 2015.

[32] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. Neurocomputing, 307:195–204, 2018.