Perturbing Across the Feature Hierarchy to Improve Standard and Strict Blackbox Attack Transferability

Nathan Inkawhich, Kevin J Liang, Binghui Wang, Matthew Inkawhich,
Lawrence Carin and Yiran Chen
Duke University
nathan.inkawhich@duke.edu

Abstract

We consider the blackbox transfer-based targeted adversarial attack threat model in the realm of deep neural network (DNN) image classifiers. Rather than focusing on crossing decision boundaries at the output layer of the source model, our method perturbs representations throughout the extracted feature hierarchy to resemble other classes. We design a flexible attack framework that allows for multi-layer perturbations and demonstrates state-of-the-art targeted transfer performance between ImageNet DNNs. We also show the superiority of our feature space methods under a relaxation of the common assumption that the source and target models are trained on the same dataset and label space, in some instances achieving a $10\times$ increase in targeted success rate relative to other blackbox transfer methods. Finally, we analyze why the proposed methods outperform existing attack strategies and show an extension of the method in the case when limited queries to the blackbox model are allowed.

1 Introduction

The adversarial machine learning community has devised many ways to cause Deep Neural Networks (DNNs) to behave unexpectedly [32, 7, 2, 21, 19]. However, the knowledge assumptions and threat models considered by an adversary are critical to attack success. In settings where access to and familiarity with the the target model is restricted, there is significant room for improving adversarial methods in terms of potency and efficiency, which is the central motivation for this work.

We focus on the blackbox transfer-based adversarial threat model for DNN image classifiers. In the standard case, blackbox means the attacker does not have access to the gradients of the target model and makes no assumptions about its architecture. Transfer-based indicates that adversarial examples are created by computing adversarial perturbations using a substitute whitebox model and then attacking the target blackbox model with the resulting examples, leveraging the notion of *transferability* [23, 24, 33]. Within this threat model, our specific goal is targeted adversarial attacks, meaning the objective is to induce the target model to output a specific class. These are significantly more challenging than un-targeted attacks, which seek to simply cause an incorrect prediction. In addition, we consider a set of more "strict" blackbox threat models in which we make varying degrees of assumptions about access to the blackbox model's training data distribution. This includes cases when the label spaces of the whitebox and blackbox models differ, and when there is *zero* training data overlap.

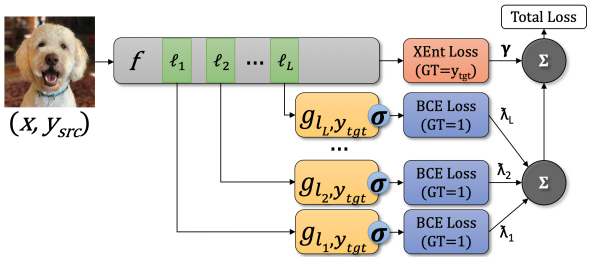

Figure 1: Visualization of forward pass construction to optimize our feature space attack objective.

Preprint.

A recent innovation in transfer attacks within the standard blackbox threat model is to craft perturbations based on intermediate layer representations, rather than simply optimizing the classification loss at the output layer [26, 13, 14, 18]. An example of such a method is the Feature Distribution Attack (FDA) [14], which computes adversarial examples using intermediate feature distributions at a single layer of the whitebox model. In this work, we propose to significantly improve the FDA method by extending it into a more flexible framework to allow for perturbations across the intermediate feature space, including the output layer. Our method relies on modeling the layer-wise and class-wise feature distributions of the whitebox model via auxiliary networks. We then optimize adversarial noise using the auxiliary models from across the deep feature space. The critical observation in this work is that by enforcing that a perturbed sample "looks-like" a target class sample *in multiple layers across the depth of the whitebox model*, the resulting sample is likely to have a much higher transfer rate than if we only considered a single intermediate layer or the output layer in isolation. For an intuition for how the attack works, consider Figure 1, which shows the forward pass required to generate adversarial noise given a pre-trained DNN f. Each $g_{\ell,y_{tgt}}$ is a feature distribution model that estimates the probability that the layer ℓ feature map $f_\ell(x)$ is from a sample of class y_{tgt}, i.e. $p(y_{tgt}|f_\ell(x))$. For a chosen y_{tgt} and set of layers, we accumulate the losses w.r.t. y_{tgt} at each intermediate layer and the output layer. By optimizing the sum, we are noising x with δ such that $x + \delta$ lies in high probability regions of the target class at several layers across feature space. We find that this method significantly improves transferability of the generated adversarial samples.

To evaluate our methods, we consider both standard and strict blackbox transfer cases. In the standard case, our attacks show state-of-the-art targeted transfer performance between popular ImageNet-1K [4] DNN models. In some cases, we improve the targeted success rate to over 55%, an absolute increase of about 50% over traditional output-layer-based methods. In the strict blackbox scenario, we evaluate our method under three separate relaxations of the common transfer assumption that the source and target models are trained on the same data distribution and share a label space. The results show that our feature-based attacks are significantly more potent than attacks generated at the output layer in all three situations. In an effort to explain why our methods yield high transferability, we also analyze the effects of our attacks and show that they cause significantly more disruption in the intermediate space than competing methods. Finally, we show that the noise generated with our methods provides a more useful prior direction for query-based attacking methods that incorporate prior information. With relatively few queries to the blackbox model, our method can be enhanced to yield over 90% targeted success rates.

2 Related work

In transfer-based blackbox adversarial attack research, many of the most popular methods build directly upon whitebox attacks [7, 15, 19], adding optimization tricks, regularization, and ensembling of whitebox models to boost transferability [15, 23, 5, 38, 17, 37]. The primary optimization goal of such methods is to cause the adversarial sample to cross the whitebox model's decision boundary through consideration of the classification loss. For example, Dong et al. [5] includes a momentum term in the optimization step of [15]; Wu et al. [36] weights the gradients through the residual connections and blocks differently; Liu et al. [17] and Tramèr et al. [34] use an ensemble of whitebox models; Xie et al. [37] use diverse inputs for more robust noise; and Sharma et al. [28] and Dong et al. [6] use low frequency constraints and translation-invariance respectively to generate un-targeted attacks for defended DNNs. These methods show reasonable un-targeted performance, but very limited targeted performance at scale, if any ([6, 34, 36, 37] only consider un-targeted attacks).

There are also some recent methods that focus on improving transfer attacks by considering the feature space of the source model, to develop noise that is less overfit to the generating architecture. Zhou et al. [38] develop a regularizer for a traditional un-targeted attack that encourages adversarial examples to have significantly different intermediate feature representations. Huang et al. [11] adjusts an existing un-targeted adversarial example to have a larger effect in a pre-specified layer of feature space. Rozsa et al. [26] and Inkawhich et al. [13] develop targeted transfer attacks in feature space, representing the "target" as a single point in a single layer of the feature space. The objective functions then minimize the L_2 distance between a source and target point in the feature space. With this limited modeling of the target class, scalability to larger models and datasets proves difficult, and success is sensitive to the choice of the target sample. To improve targeted transfer performance at scale, Inkawhich et al. [14] explicitly model the class-wise feature distributions at multiple layers of the whitebox model, so as to have a more descriptive representation of the target class when attacking. However, the method only considers attacking from a single layer (with no options for extensions),

and the primary result is that some individual layers are better than others to transfer from. As is common in all of the aforementioned methods, results are only shown for transfer scenarios where the whitebox and blackbox models are trained on the same dataset. For comparison, we develop a flexible framework that allows for perturbations of multiple layers simultaneously, evaluate in novel cross-distribution settings, and show an integration of the method with a query-based attack.

3 Methodology

Notation. We follow the notation conventions of Inkawhich et al. [14] for feature distribution modeling. To capture the layer-wise and class-wise feature distributions of a given pre-trained DNN f, we first specify a set of layers $\mathcal{L} = \{\ell_1, ..., \ell_J\}$ and classes $\mathcal{C} = \{c_1, ..., c_K\}$ of interest. Using the training dataset of f, we train binary classification auxiliary models on the feature maps at the layers in $\mathcal{L}$ for the classes in $\mathcal{C}$. Importantly, all weights of f stay fixed throughout this process. Let $f_\ell(x)$ be the layer ℓ feature map of f for input image x. Then, $g_{\ell,y}$ is a binary auxiliary model that inputs $f_\ell(x)$, and outputs the probability that the feature map is from an input of class y (i.e., $g_{\ell,y}(f_\ell(x)) = p(y|f_\ell(x))$). Note, the training of the necessary auxiliary models is done prior to the attacking process, and requires only a pre-trained model f and its corresponding training dataset. It is also worth noting that each auxiliary model is small and individually very inexpensive to train (see Appendix B). Finally, let $f(x)$ be the predicted probability distribution over the set of classes on which f was trained, and $F(x) = \text{argmax } f(x)$ be the classification prediction.

Preliminaries. Once trained, we can use the modeled feature distributions to attack. Given our interest in targeted attacks, we start from a source image x initially classified as y_{src} (i.e. $F(x) = y_{src}$) and generate a perturbation δ such that $F(x + \delta) = y_{tgt}$ for a pre-specified target class y_{tgt}, with $y_{src} \neq y_{tgt}$. To keep the noise approximately imperceptible and thus adversarial, for the following objectives we constrain the L_p norm of δ, such that $||\delta||_p \leq \epsilon$, and enforce that the perturbed input's pixel values exist in $[0, 1]$.

FDA. Our proposed attack framework builds on *FDA+fd*, the top performing variant of [14], which is noted here as *FDA* and described as

$$\max_{\delta} p(y = y_{tgt}|f_\ell(x + \delta)) + \eta \frac{\|f_\ell(x + \delta) - f_\ell(x)\|_2}{\|f_\ell(x)\|_2}. \tag{1}$$

This method has two main components, both of which are optimized at a *single* pre-specified layer ℓ. The first component is the feature distribution piece, which takes the form of $p(y = y_{tgt}|f_\ell(x + \delta))$. By maximizing this probability through optimizing δ, we are perturbing the input image such that the layer ℓ feature map $f_\ell(x + \delta)$ lies in a high probability region of the target class in feature space. In other words, the perturbed feature map resembles a feature map that will be classified as the target class. The second component is the feature disruption term, which enforces that the feature map of the perturbed input is significantly different than the feature map of the original input.

FDA+xent. A key contribution in this work is the extension of (1) to include *multi-layer* information. One way to do so is to incorporate the whitebox model's output prediction as part of the attack objective. Let $H(f(x), y)$ be the standard cross-entropy loss between the predicted probability distribution $f(x)$ and the target distribution y (commonly a one-hot distribution). We include this term in the *FDA* objective to create *FDA+xent*, as described by

$$\max_{\delta} p(y = y_{tgt}|f_\ell(x + \delta)) + \eta \frac{\|f_\ell(x + \delta) - f_\ell(x)\|_2}{\|f_\ell(x)\|_2} - \gamma H(f(x + \delta), y_{tgt}), \tag{2}$$

where $\gamma > 0$, weighting the contribution of cross-entropy term. This attack objective optimizes the noise such that the layer ℓ feature map is in a high-probability region of the target class, and that the output prediction of the whitebox model is of the target class. Addition of the cross-entropy term is motivated by the correlation results from Inkawhich et al. [14], which show that the probability distribution over the classes, as measured in the "optimal transfer layer," has low correlation with the probability distribution over the classes at the output layer. This indicates that the vanilla *FDA* in (1) does not reliably create targeted examples for the whitebox model, which may limit the effectiveness of the attack; we explore this further in Section 4.1.4.

***FDA*$^{(N)}$.** To further generalize the multi-layer framework, we extend the objective function to allow for optimization in *multiple intermediate layers*. We call this new attack *FDA*$^{(N)}$, with objective:

$$\max_{\delta} \sum_{\ell \in \mathcal{L}} \lambda_\ell \left[p(y = y_{tgt}|f_\ell(x + \delta)) + \eta \frac{\|f_\ell(x + \delta) - f_\ell(x)\|_2}{\|f_\ell(x)\|_2} \right], \tag{3}$$

where $\sum_{\ell \in \mathcal{L}} \lambda_\ell = 1$, $\lambda_\ell > 0$, and the *N* in the name refers to the number of layers in $\mathcal{L}$. Note that for $N = 1$, the attack objective is the same as (1). Both the *+xent* and *multi-intermediate-layer* extensions significantly increase the capability and flexibility of the method, while maintaining the intuition of attacking primarily based on feature space information. These two extensions can also be straightforwardly composed to form a *FDA*$^{(N)}$*+xent* attack.

Optimization. The final step in the methodology is the optimization of the above objective functions. Practically, we leverage the autograd feature of PyTorch by assembling the forward pass of the attack as shown in Figure 1. This step involves connecting the required auxiliary models to the corresponding layers of f and computing the final loss with proper weighting. We then use an iterative projected gradient descent with momentum procedure [5] to apply the adversarial noise to the input while maintaining the norm and natural image constraints.

4 Experiments

We evaluate the multi-layer *FDA* framework in a variety of settings. In Section 4.1, we consider the standard blackbox transfer case, where knowledge of the target model's training dataset is assumed, and test transferability between ImageNet-1K models. We examine single-source single-target model transfers, ensemble transfers, distal adversaries, and analyze why and how the attacks work through the lens of feature disruption. In Section 4.2 we evaluate transferability in cross-distributional settings, where the source and target model are trained on different data distributions and label spaces. Finally, in Section 4.3 we show an extension of the method when limited queries to the target model are allowed during attack generation.

4.1 ImageNet transfer experiments

Experimental Setup. In our analysis, we use the following ImageNet-1K models from the PyTorch Model Zoo: ResNet-34/50/101/152 (RN34, RN50, RN101, RN152) [9]; VGG16bn and VGG19bn [30]; MobileNetv2 (MNv2) [27]; and DenseNet-121/161/201 (DN121, DN161, DN201) [10]. The notation RN50→DN121 means attacks are generated on a RN50 whitebox (source) model and transferred to a DN121 blackbox (target) model. The primary metrics of attack success are the error rate and targeted success rate (tSuc) in the blackbox model, where all clean samples are correctly classified by both the source and target models. For the attack settings, we use the common L_∞ $\epsilon = 16/255$ and $\mathrm{perturb\_iters} = 10$. The architecture of each auxiliary model $g_{\ell,y}$ is a simple Conv-Conv-FC scheme, which is significantly more parameter efficient than the FC-FC in [14].

To find the best combinations of layers for *FDA*$^{(N)}$ attacks, we use an iterative greedy search on a held out part of the ImageNet validation set and find value in including up to 5 layers. Crucially, feature space attacks have been shown to be *blackbox model agnostic*: the optimal transfer layer for a whitebox model does not change for different target blackbox models [13, 14]. For this reason, we find the optimal layer sets for the RN50 and DN121 whitebox models only once, in the RN50→DN121 and DN121→RN50 transfer scenarios, respectively. When attacking any other blackbox model, we use the previously found layer combinations. Practically, an attacker may have two models in a sandbox environment, where one is treated as a whitebox and the other a blackbox. They may then find the optimal layer settings in this sandbox environment, which can be used to attack any other blackbox of interest. The layer decoding scheme and the sequence in which the layers are added to the attack is shown in Appendix A. Finally, we weight all intermediate layers equally, i.e. $\lambda_\ell = 1/N$. Additional details regarding experimental setup are in Appendix B.

4.1.1 Transfer results in standard blackbox settings

We perform single-source single-target model transfers for RN50→{DN121, VGG16bn, RN152, MNv2} and DN121→{RN50, VGG16bn, DN201, MNv2}. For baselines, we compare against the popular TMIM [5] and TMIM+SGM [36], both of which search for adversarial directions using *only output layer information* of the whitebox model. *FDA*$^{(1)}$ [14] is also considered a baseline. In an effort to emphasize the transferability of the noise generation technique when the blackbox model architecture may be unknown, we favor settings where the source and target model are from different architectural families to avoid potential biases.

The procedure for measuring the transfer results in Table 1 is as follows. For each source-target model pair, we randomly select 15,000 source images from the ImageNet validation set for which both models initially predict the correct source label. For each source sample, we then randomly select 5 target labels from the ImageNet label set, and execute a targeted attack towards each. Thus,

Table 1: Full-ImageNet Transfer Results (notation = error / tSuc, $\epsilon = 16/255$)

	Whitebox Model = RN50				Whitebox Model = DN121			
Attack \\ Target	DN121	VGG16bn	RN152	MNv2	RN50	VGG16bn	DN201	MNv2
TMIM	44.7 / 3.0	48.7 / 1.5	41.4 / 3.2	54.9 / 0.1	46.2 / 2.3	49.6 / 1.3	44.6 / 5.5	58.5 / 1.0
TMIM+SGM	47.4 / 4.5	51.9 / 2.4	43.1 / 4.2	61.1 / 2.2	54.8 / 4.7	53.8 / 2.8	51.4 / 10.0	66.1 / 3.2
FDA$^{(1)}$	90.3 / 19.4	86.6 / 13.6	90.8 / 17.1	84.7 / 7.1	91.8 / 20.4	90.6 / 20.6	94.8 / 35.3	88.4 / 10.7
FDA$^{(2)}$	93.1 / 28.4	91.5 / 22.9	92.7 / 22.0	89.8 / 10.7	**95.1** / 22.3	**94.7** / 26.3	96.6 / 38.9	**93.0** / 12.4
FDA$^{(3)}$	94.1 / 31.0	**93.0** / 26.1	93.2 / 22.9	**91.0** / 11.8	95.1 / 22.2	94.7 / 26.0	**96.7** / 37.1	92.8 / 12.6
FDA$^{(4)}$	**94.2** / 38.1	92.4 / 30.7	**93.6** / 30.6	90.2 / 14.9	92.9 / 43.7	92.4 / 42.6	96.3 / 67.5	88.9 / 20.9
FDA$^{(5)}$	94.2 / 38.6	92.7 / 31.0	93.5 / 29.7	90.9 / 15.1	93.8 / 44.4	93.4 / 44.4	96.6 / 68.2	90.6 / 22.4
FDA$^{(1)}$+xent	84.4 / 40.3	80.9 / 24.9	85.0 / 42.3	78.6 / 13.2	88.8 / 37.4	87.8 / 32.7	93.3 / 65.3	85.1 / 16.9
FDA$^{(2)}$+xent	89.0 / 51.7	86.8 / 37.2	88.2 / 48.1	84.3 / 18.7	92.7 / 40.5	92.5 / 40.5	95.5 / 70.3	90.1 / 19.4
FDA$^{(3)}$+xent	90.2 / 55.2	88.3 / 41.6	89.2 / 49.2	85.6 / 21.1	92.9 / 41.7	92.5 / 41.8	95.7 / 70.6	90.0 / 20.4
FDA$^{(4)}$+xent	90.5 / 57.3	88.2 / 42.8	89.7 / **53.2**	85.4 / 22.4	91.3 / 49.4	90.9 / 46.2	95.3 / 76.5	87.1 / 22.6
FDA$^{(5)}$+xent	90.9 / **57.9**	88.8 / **43.5**	89.7 / 51.6	86.4 / **22.9**	92.2 / **50.1**	92.1 / **48.0**	95.6 / **77.1**	88.8 / **24.4**

every table entry is an average computed over 75,000 targeted attacks. Both error rate and targeted success rate, as measured in the blackbox model, are reported as "error / tSuc".

Our first major observation is that the *+xent* component alone gives large gains in tSuc. If we compare *FDA*$^{(1)}$ to *FDA*$^{(1)}$*+xent*, in several cases it leads to a tSuc increase of over 20%; for example, in RN50→DN121 tSuc increases from 19.4% to 40.3%. Further, we observe *+xent* consistently helps, regardless of how many layers are used. The largest performance gain observed by *+xent* is in the DN121→DN201 case, where in the 1 attack layer case, tSuc increases by 30%. The next result of interest is the performance gain by adding multiple intermediate layers (*FDA*$^{(N)}$, $N > 1$). As N is increased, tSuc nearly doubles in all cases. For perspective, when we compare these results to the baseline *TMIM*-based methods, the targeted success rate in many cases improves by more than $10\times$.

We also observe that *+xent* and *multi-intermediate-layer* are complementary components: in all transfer scenarios the most powerful targeted attacks arise from using both. Specifically, consider the *FDA*$^{(5)}$*+xent* attack, which is the most effective in all but one case. When VGG16bn is the target model, on average this attack outperforms *TMIM+SGM* by 37% error / 43% tSuc and *FDA*$^{(1)}$ by about 2% error / 29% tSuc. An interesting observation is that the *+xent* term harms error performance, albeit less relevant because our focus is targeted attacks, and the methods have been optimized for tSuc. However, in all cases the best *FDA*$^{(N)}$ method nearly doubles the error induced by *TMIM+SGM*.

4.1.2 Comparison to ensemble methods

Ensemble-based approaches [17] form a separate family of methods to generate transferable adversarial examples, deviating from our previous single-source model assumption. In this setting, the attacker trains an ensemble of models (with whitebox access to each) and generates noise based on the output of the ensemble with traditional attacking techniques, e.g., *TMIM*. A natural iterate in ensemble methods is growing the numbers of models. We compare the effect of adding a layer within the *FDA*$^{(N)}$*+xent* framework versus adding a model to the generating ensemble using the same attack settings as in the previous experiment. To avoid conflicts with models already in use, in every transfer case we use the following sequence for adding models to the ensemble: [whitebox, DN161, RN101, VGG19bn, RN34]. For example, in the RN50→DN121 transfer case, using a 3-model ensemble means generating noise from RN50, DN161, and RN101.

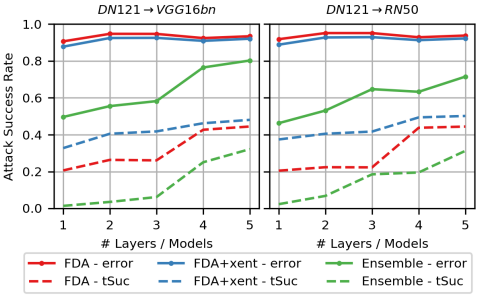

Figure 2: FDA versus ensemble attacks.

The results of these experiments are shown in Figure 2, with extensions in Appendix C. The *FDA* methods hold a wide margin over ensemble methods in almost all cases for both error and tSuc, especially *FDA*$^{(N)}$*+xent*. Also, although adding models to the whitebox ensemble tends to increase attack performance (as expected), large jumps in the ensemble's performance usually occur when a model from the same family as the blackbox model is added. For example, in the DN121→VGG16bn case, a performance spike happens when the fourth model gets added, which is the VGG19bn model. Similarly, in the DN121→RN50 case, the ensemble method has the greatest increases when the third and fifth models are included, which are the RN101 and RN34 models, respectively.

4.1.3 Distal transfers

Distal adversarial examples are generated by starting from random noise and optimizing for high prediction probability for some target class [22]. Transferring distals between ImageNet models in the standard blackbox setting can make for an interesting test of transferability. Although distals are not our main focus, they provide a compelling future direction to study, as they depend less on the source image/class than standard transfers. This eliminates one axis of variability that may better isolate the quality of the attacking algorithm. For these tests, we do not constrain ϵ and increase the perturbation budget to allow each attack to perturb for 200 iterations.

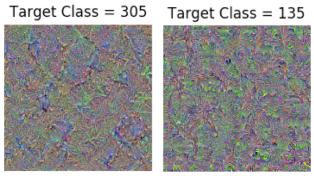

Figure 3: Samples of distals.

The results are shown as top-1 / top-5 tSuc rates, as averaged over 4,000 distals, each optimized towards a random target class. We use *TPGD* [19] as our baseline instead of *TMIM*, as momentum in the optimizer empirically harms performance in distal generation. Table 2 shows that results of the distal transfers align with the previous trends, and sample distals are shown in Figure 3. Distals generated with the *FDA*$^{(N)}$*+xent* method are significantly more transferable. On average, the *FDA*$^{(4)}$*+xent* causes increases of 48% top-1 / 62% top-5 tSuc over the baseline method.

Table 2: Distal transfer tSuc rates (top1 / top5)

attack	RN50→DN121	RN50→VGG16bn
TPGD	10.1 / 21.0	6.1 / 13.4
FDA$^{(1)}$+xent	46.9 / 68.1	30.6 / 50.6
FDA$^{(2)}$+xent	58.6 / 78.8	38.0 / 62.5
FDA$^{(3)}$+xent	64.0 / 83.6	44.6 / 69.9
FDA$^{(4)}$+xent	**65.8 / 85.3**	**48.1 / 73.0**

4.1.4 Analysis of *multi-intermediate-layer* and *+xent*

To analyze the effects of the *multi-intermediate-layer* and *+xent* components, we perform a feature disruption analysis [14]. Disruption measures the effect of an adversarial perturbation on the feature space of a model, w.r.t. the target class. In this setting, we measure the disruption caused by a targeted attack, at layer ℓ of model f as: $\text{disruption}_\ell = p(y = y_{tgt}|f_\ell(x+\delta)) - p(y = y_{tgt}|f_\ell(x))$. f can be either a whitebox or blackbox model and the attack that generates δ may be any method. Figure 4 shows the disruption caused by several attacks for RN50→DN121 (see Appendix D for more).

Firstly, the *TMIM* attack is shown to have drastically different effects in the whitebox and blackbox models. Since the noise is generated specifically for the output layer of the whitebox, it only causes high disruption in the last few layers of the whitebox, and very little disruption in the blackbox. For the *FDA*$^{(1)}$ method, we already see a significantly different pattern, namely a much higher disruption in the intermediate layers of both models. Now, consider the effects of adding multiple intermediate layers. As we look from *FDA*$^{(1)}$ to *FDA*$^{(2)}$ to *FDA*$^{(4)}$, the disruption is noticeably increased in the earlier and later layers, which is expected as the added intermediate layers are both earlier and later than the first layer. Finally, consider the *+xent* component. Without it, the *FDA*$^{(N)}$ methods have sharp decreases in the last few layers, despite high intermediate disruption. The inclusion of this term significantly boosts disruption in the final layers of both the whitebox and blackbox. Given the large effects of our feature space methods throughout the models, we say that the methods attack along the feature hierarchy.

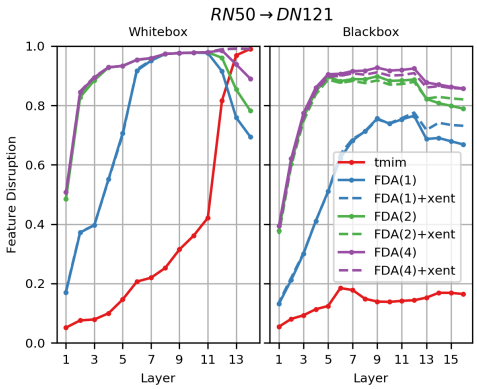

Figure 4: Disruption of features caused by transfer attacks.

4.2 Cross-distribution experiments

An often overlooked (but critical) detail in standard blackbox transfer attacks is the underlying assumption that the whitebox and blackbox models are trained on the exact same dataset [5, 17, 36, 13, 14, 19, 33]. This implies that the adversary can find such a pre-trained model in the open-source domain or actually has the blackbox model's dataset and can train their own whitebox model. In many cases, the dataset used to train the targeted model may not be publicly available due to proprietary [8, 31], security [16, 29], or privacy [1, 25] issues. Making too strong of an assumption about the availability of the target model's data may lead to overly optimistic transfer results. Here, we evaluate the performance of transfer attacks under several more "strict" blackbox threat models, where the whitebox model is not trained on the exact same data distribution or label space as the blackbox

model. We hypothesize that it is more challenging to create transferable adversarial examples if the source and target models are not trained on identical data.

We are primarily concerned with three "cross-distribution" transfer scenarios:

1. *No training data overlap between the whitebox and blackbox, but significant label space overlap.*
2. *The blackbox is trained on a subset of the whitebox's training dataset and label space.*
3. *The whitebox is trained on a subset of the blackbox's training dataset and label space.*

These three cases represent several common situations that may be encountered in reality. For example, scenarios 1 and 3 may arise when the attacker is generally aware of the types of classes that the target model is trained on, but has to go out and collect their own data for training the source model. Scenario 2 may arise when the attacker has access to a large model with a high variety of classes, and wishes to attack a smaller target model designed for a subset of the classes.

Experimental setup. To accommodate these scenarios, we create three new "Restricted-ImageNet" datasets (RINet), which are subsets of ImageNet-1k (Full-INet). We design *RINet-A* and *RINet-B* to each be 15-class datasets, sharing 10 label classes while each having 5 unique label classes. Each of the classes in *RINet-A* and *RINet-B* consist of WordNet [20] super-classes aggregating a related subset of ImageNet-1K classes. Critically, the classes chosen to comprise each of the shared classes of *RINet-A* and *RINet-B* are disjoint. For example, the "bird" class of RINet-A is composed of the ImageNet-1k classes: `[10:'brambling', 13:'junco', 16:'bulbul', 17:'jay', 18:'magpie']`, while RINet-B contains `[12:'housefinch', 14:'indigobunting', 15:'robin', 19:'chickadee', 20:'waterouzel']`. As a result, *RINet-A* and *RINet-B* have *zero training data overlap*. The *RINet-C* dataset is comprised of 20 classes and is meant to be a bigger subset of Full-INet than RINet-A and RINet-B. We train RN50 and DN121 models on each of the RINet datasets, then train the auxiliary models necessary to attack with the *FDA*$^{(N)}$*+xent* framework. For more details regarding setup and the dataset splits, see Appendix E. Hereafter, the notation RN50-A indicates the RN50 model trained on RINet-A.

Scenario 1. Table 3 shows the error / tSuc results for all three cross-distribution scenarios. First, consider the no training data overlap case as represented by RN50-A→DN121-B and RN50-B→DN121-A. For baselines, we include the RN50-A→DN121-A and RN50-B→DN121-B. As hypothesized, the cross-distribution transfer performance is worse than the within-distribution performance. However, the *FDA*-based methods still perform well, and significantly better than *TMIM*. For the DN121-B target model case, *FDA*$^{(5)}$*+xent* induces 77% error / 63% tSuc when transferred from the RN50-A whitebox, which is an improvement over *TMIM* of about 40% error / 51% tSuc. When compared to the *FDA*$^{(1)}$ method, our multi-layer extensions lead to improvements of over 9% error / 17% tSuc. Similar results are shown in RN50-B→DN121-A.

Scenario 2. Next, we analyze the case in which the blackbox model is trained on a subset of the whitebox model's training dataset, i.e., RN50-Full→DN121-A/B. Different from the previous experiment, the complexity of the whitebox and blackbox tasks is significantly different (1,000 class versus 15 class classification), where the whitebox model's task is more challenging. Remarkably, despite such a gap in the task and dataset complexity, the transfer results from the feature distribution attacks are quite high. In both cases, the error is near 60% and tSuc is over 45%. This is as compared to the near 16% error / 5% tSuc from the *TMIM+SGM* attack. Again, we also see large improvements using multi-intermediate-layer attacks. Interestingly, the *+xent* component is not helpful in this

Table 3: Transfer results for cross-distribution tests (notation = error / tSuc)

	Target Model = DN121-B			Target Model = DN121-A			Target Model = DN121-Full		
Attack \\ Source	RN50-A	RN50-B	RN50-Full	RN50-A	RN50-B	RN50-Full	RN50-A	RN50-B	RN50-C
TMIM	37.8 / 11.7	33.5 / 21.4	16.4 / 3.3	29.9 / 19.2	34.8 / 11.4	12.6 / 2.7	29.2 / 0.9	32.0 / 1.0	35.3 / 2.7
TMIM+SGM	43.0 / 16.6	38.6 / 27.8	19.2 / 5.2	36.6 / 27.0	38.3 / 14.6	16.2 / 4.6	30.4 / 1.2	32.9 / 1.4	37.8 / 3.3
FDA$^{(1)}$	67.7 / 45.8	70.9 / 60.8	38.6 / 21.0	66.0 / 57.3	67.0 / 40.8	34.8 / 18.9	53.8 / 6.9	58.1 / 8.1	75.8 / 14.1
FDA$^{(2)}$	72.3 / 54.1	75.5 / 68.0	53.9 / 37.1	72.5 / 65.9	71.3 / 48.4	52.4 / 36.9	59.3 / 9.4	61.5 / 9.2	80.6 / 16.7
FDA$^{(3)}$	73.9 / 58.0	77.4 / 72.4	58.3 / 42.1	76.9 / 72.8	70.7 / 50.4	57.0 / 42.3	55.2 / 11.1	57.4 / 10.7	77.9 / 22.1
FDA$^{(4)}$	76.1 / 60.7	79.4 / 74.9	56.0 / 41.5	79.1 / 75.3	73.0 / 53.2	55.0 / 42.1	58.1 / 12.1	60.4 / 11.4	80.1 / 22.5
FDA$^{(5)}$	76.5 / 61.9	79.7 / 75.4	**59.2 / 45.5**	79.4 / 75.9	73.5 / 54.0	**58.4 / 45.5**	58.8 / 12.2	60.7 / 11.7	80.5 / 22.4
FDA$^{(1)}$+xent	70.2 / 50.1	75.6 / 69.4	31.7 / 17.2	72.0 / 65.8	69.0 / 45.6	28.5 / 16.0	55.0 / 7.9	58.3 / 9.0	76.1 / 16.4
FDA$^{(2)}$+xent	74.5 / 58.3	79.8 / 75.2	43.9 / 29.8	77.2 / 73.0	73.3 / 53.0	42.4 / 30.7	**60.4** / 10.5	**62.1** / 10.7	**81.2** / 19.4
FDA$^{(3)}$+xent	74.3 / 59.3	79.6 / 76.1	47.0 / 33.3	78.7 / 75.4	71.7 / 52.4	46.1 / 35.0	55.6 / 11.5	57.9 / 11.4	78.3 / 23.8
FDA$^{(4)}$+xent	76.5 / 62.2	81.3 / 78.0	45.9 / 32.7	80.8 / 77.7	74.1 / 55.3	44.7 / 34.0	58.7 / 12.6	61.0 / 12.1	80.6 / **24.3**
FDA$^{(5)}$+xent	**77.0 / 63.1**	**81.8 / 78.5**	48.8 / 36.3	**80.9 / 77.8**	**74.4 / 56.2**	47.8 / 37.5	59.3 / **12.6**	61.3 / **12.2**	81.1 / 23.9

situation. This is likely because the decision boundary structure of the two models is sufficiently different that it is no longer relevant for generating attacks.

Scenario 3. In the last case, the whitebox model is trained on a subset of the blackbox model's dataset, which is represented in the RN50-A/B/C→DN121-Full columns. Not surprisingly, the transfer rates of all methods are lower in this setting, likely because the complexity of the whitebox model's task is much simpler than the blackbox model's task. Thus, the level of detail and granularity of the features learned in the whitebox model is less. However, as with the other tests, all *FDA*-based methods significantly outperform the *TMIM* method, and the multi-layer upgrades significantly advance the performance over the vanilla $FDA^{(1)}$. An interesting takeaway is that the size of the whitebox's subset matters greatly. Even as we look from RN50-A/B to RN50-C as whitebox models, the performance increases by over 20% error / 12% tSuc.

4.3 Query-based extension

A criticism of transfer-based blackbox attacks is that the success rate can be dependent on the whitebox-blackbox pair: certain combinations may have more limited transfer in some instances, as can be seen when MNv2 is the blackbox model in Table 1. In contrast, query-based blackbox attacks directly estimate the gradient of the blackbox model via repetitive querying during attack generation [35, 12, 3], and typically achieve higher success rates than transfer-based attacks. The principle drawback to query attacks, which does not afflict transfer methods, is that for each adversarial example the attacker may have to query the target model tens-of-thousands of times to properly estimate the gradient, which in some cases may not be feasible due to time and monetary constraints, or potential threat detection systems in the target model. Motivated by query efficiency, Cheng et al. [3] incorporate the adversarial direction from a transfer-based attack as a prior in the P-RGF attack. To show a potential way to extend our methods, in situations where limited queries to the blackbox model are acceptable, we investigate using $FDA^{(5)}$+*xent* and *TMIM* methods as *priors* in [3]. To integrate with P-RGF, we extend the transfer methods to perturb for 15 total iterations. In the first 10, the noise is solely optimized on the whitebox model (as usual). Only in the last 5 iterations do we use the transfer direction as the prior in the P-RGF estimator. We find warm-starting the transfer direction in this way to be effective. The tSuc results for several transfer scenarios, under different query budgets, for *TMIM* / $FDA^{(5)}$+*xent* priors, are shown in Table 4.

Table 4: tSuc of transfer-based attacks used as a prior with P-RGF (prior = *TMIM* / $FDA^{(5)}$+*xent*)

	Whitebox Model = RN50			Whitebox Model = DN121		
# Queries \\ Target	DN121	VGG16bn	MNv2	RN50	VGG16bn	MNv2
0	3.0 / 57.9	1.5 / 43.5	0.1 / 22.9	2.3 / 50.1	1.3 / 48.0	1.0 / 24.4
100	5.5 / 70.5	3.3 / 60.3	2.1 / 35.5	4.8 / 63.0	3.1 / 65.8	2.0 / 37.1
500	7.5 / 81.2	6.2 / 77.3	4.2 / 56.8	6.6 / 79.7	5.8 / 86.2	3.9 / 65.8
1000	10.2 / 87.3	9.9 / 86.5	6.9 / 71.9	9.0 / 88.9	9.3 / 93.5	6.4 / 82.3
2000	14.9 / **92.9**	16.9 / **93.4**	12.3 / **85.5**	13.0 / **94.7**	16.0 / **97.3**	11.9 / **93.2**

In all transfer cases, we observe that the $FDA^{(5)}$+*xent* attack yields a significantly better prior than *TMIM*, which is aligned with the performance gaps observed in previous experiments. As the number of queries to the blackbox model increases, the attack success rate increases, and the margin of performance between the two priors grows to nearly 80% tSuc. With only 100 queries, the attack success rate of $FDA^{(5)}$+*xent* increases by over 10%, and with 2,000 queries the average tSuc rate is over 90%. This result motivates future work on how to improve the feature space attack framework through the allowance of a limited budget of queries to the target model during attack generation.

5 Conclusion

We introduce a feature space-based adversarial attack framework that allows for perturbations along the extracted feature hierarchy of a DNN image classifier to achieve state-of-the-art targeted blackbox attack transferability. In the "standard" blackbox transfer case, where the source and target model share ImageNet-1K as a training dataset, our methods outperform output-layer-based attacks by a factor of $10\times$ and existing feature-space methods by a factor of $2-3\times$. These performance gains are attributed to the inclusion of *multi-layer* information, which leads to significantly higher disruption in the feature spaces of both the whitebox and blackbox. In a set of three "strict" blackbox transfer scenarios, where the training dataset and label spaces of the whitebox and blackbox models differ, we show that our methods maintain similar performance margins over the baselines, even when there is no training data overlap or a 985-class discrepancy in the label space. Finally, we show an extension of the method for situations when it is acceptable to query the target model during attack generation.

References

[1] US Census Bureau. Restricted-Use Microdata. *https://www.census.gov/topics/research/guidance/restricted-use-microdata.html*, 2018.

[2] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[3] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *NeurIPS*, 2019.

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

[5] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

[6] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*, 2019.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[8] Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's bing search engine. In *ICML*, 2010.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[11] Qian Huang, Isay Katsman, Zeqi Gu, Horace He, Serge J. Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *ICCV*, 2019.

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

[13] Nathan Inkawhich, Wei Wen, Hai Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. In *CVPR*, 2019.

[14] Nathan Inkawhich, Kevin J Liang, Lawrence Carin, and Yiran Chen. Transferable perturbations of deep feature distributions. In *ICLR*, 2020.

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017.

[16] Kevin J Liang, Geert Heilmann, Christopher Gregory, Souleymane O. Diallo, David Carlson, Gregory P. Spell, John B. Sigman, Kris Roe, and Lawrence Carin. Automatic Threat Recognition of Prohibited Items at Aviation Checkpoint with X-ray Imaging: A Deep Learning Approach. In *SPIE Anomaly Detection and Imaging with X-Rays (ADIX) III*, 2018.

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

[18] Yantao Lu, Yunhan Jia, Jianyu Wang, Bai Li, Weiheng Chai, Lawrence Carin, and Senem Velipasalar. Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction. In *CVPR*, 2020.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[20] George A Miller. *WordNet: An Electronic Lexical Database*. MIT press, 1998.

[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

[22] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

[23] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv*, abs/1605.07277, 2016.

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

[25] Dezsö Ribli, Anna Horváth, Zsuzsa Unger, Péter Pollner, and István Csabai. Detecting and Classifying Lesions in Mammograms with Deep Learning. In *Scientific reports*, volume 8. Nature Publishing Group, 2018.

[26] Andras Rozsa, Manuel Günther, and Terrance E. Boult. LOTS about attacking deep features. In *IJCB*, 2017.

[27] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[28] Yash Sharma, Gavin Weiguang Ding, and Marcus A. Brubaker. On the effectiveness of low frequency perturbations. In *IJCAI*, 2019.

[29] John B. Sigman, Gregory P. Spell, Kevin J Liang, and Lawrence Carin. Background Adaptive Faster R-CNN for Semi-supervised Convolutional Object Detection of Threats in X-ray Images. In *SPIE Anomaly Detection and Imaging with X-Rays (ADIX) V*, 2020.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[31] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *ICCV*, 2017.

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[33] Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. The space of transferable adversarial examples. *arXiv*, abs/1704.03453, 2017.

[34] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

[35] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aäron van den Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

[36] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. In *ICLR*, 2020.

[37] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *CVPR*, 2019.

[38] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In *ECCV*, 2018.

Appendix

A. Layer decoding tables

Layer #	RN50 Layers	Sequence
14	3_4_6_3	
13	3_4_6_2	
12	3_4_6_1	
11	3_4_6	4
10	3_4_5	1
9	3_4_4	
8	3_4_3	
7	3_4_2	3
6	3_4_1	
5	3_4	2
4	3_3	5
3	3_2	
2	3_1	
1	3_	
Input Layer		

Layer #	DN121 Layers	Sequence
16	6_12_24_16	
15	6_12_24_12	
14	6_12_24_8	4
13	6_12_24_2	
12	6_12_24	
11	6_12_22	
10	6_12_18	1
9	6_12_14	
8	6_12_10	3
7	6_12_6	
6	6_12_2	
5	6_12	2
4	6_10	5
3	6_6	
2	6_2	
1	6_	
Input Layer		

Figure 5: Layer notation of whitebox models and sequence in which layers get added to multi-intermediate-layer attacks.

Here we discuss the DNN layer notation used throughout the work. We use two whitebox models: ResNet50 (RN50) [9] and DenseNet121 (DN121) [10], which have been shown to be good sources for generating transferable adversarial examples.

The RN50 notation follows the implementation in `https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py`. RN50 has 4 layer groups, with {3, 4, 6, 3} Bottleneck blocks in each, respectively. Of the 16 possible layers we notate 14 of them in Figure 5 where "deeper" layers closer to the output of the model have higher layer numbers. This is the notation used in Figures 4 and 7.

The DN121 notation follows the implementation in `https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py`. DN121 also has 4 main layer groups, with {6, 12, 24, 16} layers in each, respectively. From the 58 possible layers, we sample 16 layers from across the depth as shown in Figure 5.

As an example of how the layers are used in the attacks, when generating an attack from a RN50 whitebox that uses layers 5 and 10, this means that we are "probing" the model to extract the feature map at the output of the {3,4} and {3,4,5} layers. The "Sequence" column is the order in which the 5 attacking layers get added to the multi-layer attack, as found by greedy optimization. See Appendix B for more information about this process.

B. Additional experimental setup details

Auxiliary models. Using pseudo-PyTorch notation, we use the following architecture for the auxiliary models: `[Conv(#kernels=128, kernel_size=3, stride=1, pad=1), ReLU, MaxPool(2,2), Conv(#kernels=128, kernel_size=3, stride=1, pad=1), ReLU, MaxPool(2,2), Dropout(p=0.3), linear(inputs=?, outputs=1)]`. Note that the number of nodes in the linear layer depends on the spatial size of the input feature map. We use this exact architecture regardless of layer and feature map size, for simplicity reasons. Customizing the auxiliary architecture to each layer in the whitebox model may result in better modeling of the feature distributions at each layer, but we find this architecture to perform well enough.

Each auxiliary model is trained for 8,000 iterations with `batch_size=64`, `momentum=0.9`, `weight_decay=5e-4`, `initial_learning_rate=0.001`, and a single learning rate decay step at the $6{,}000^{th}$ iteration to 0.0001. For each batch, we use a weighted sampling scheme to equalize the counts of positive and negative training samples. Note, the training of each auxiliary model can be done in a massively parallel fashion.

Greedy layer optimization. Selection of the whitebox model layers to target with *FDA*$^{(N)}$ is a combinatorial problem. Rather than try out every possible combination, we use a greedy optimization strategy to find the best layers in Section 4, which we elaborate on here. While this greedy approach may not find the absolute optimum set of layers, we find that it still performs quite well, with each added layer delivering significant gains in attack performance.

Note, in [14] the "optimal" attacking layer is found via sweeping across possible layers and using the layer that empirically has the best tSuc rate. We adopt a similar scheme here. We only optimize to find the attack layers once. When finding the attack layers for the RN50 whitebox model we use the DN121 as a blackbox, and when finding the attack layers for DN121 we use the RN50 as a blackbox. Using a random held-out set of 200 source images from the ImageNet validation set, we generate attacks from each layer to find the best single layer to attack from. To find the best two layers to attack from, we start with the best single layer and sweep the second layer to find the best complement to the first layer. This process of using the best previous layers and finding the most complementary new layer continues until we find the best 5 layers, where we notice a performance saturation. The sequence of how the attacking layers get added is shown in the "Sequence" columns of Figure 5. For example, for the *FDA*$^{(3)}$ attack on a RN50 whitebox, the layers used in the attack are {3,4,5}, {3,4}, and {3,4,2}. The hyper-parameters in the objective functions are found via line search. For the RN50 whitebox, $\eta = 1\mathrm{e}-5$ and $\gamma = 1\mathrm{e}-4$. For the DN121 whitebox, $\eta = 1\mathrm{e}-6$ and $\gamma = 1\mathrm{e}-4$. For simplicity, in a multi-intermediate-layer attack all N layers are equally weighted, so $\lambda_\ell = 1/N$.

Baseline attacks. The setup and configurations of the baseline attacks largely mirror the original papers. Since the results in the *TMIM+SGM* [36] paper are all tuned for un-targeted attacks, we do a similar line search on a held out dataset to find $\mathrm{decay} = 0.5$ for RN50 and $\mathrm{decay} = 0.4$ for DN121. For the ensemble attack [17], all models are weighted equally, and we include momentum in the optimization (although momentum is not discussed in the original paper) because it empirically improves the results.

All of the attacks, including new ones and the baselines use L_∞ $\epsilon = 16/255$, $\mathrm{step\_size} = 2/255$ and $\mathrm{perturb\_iters} = 10$.

C. Full ensemble comparison results

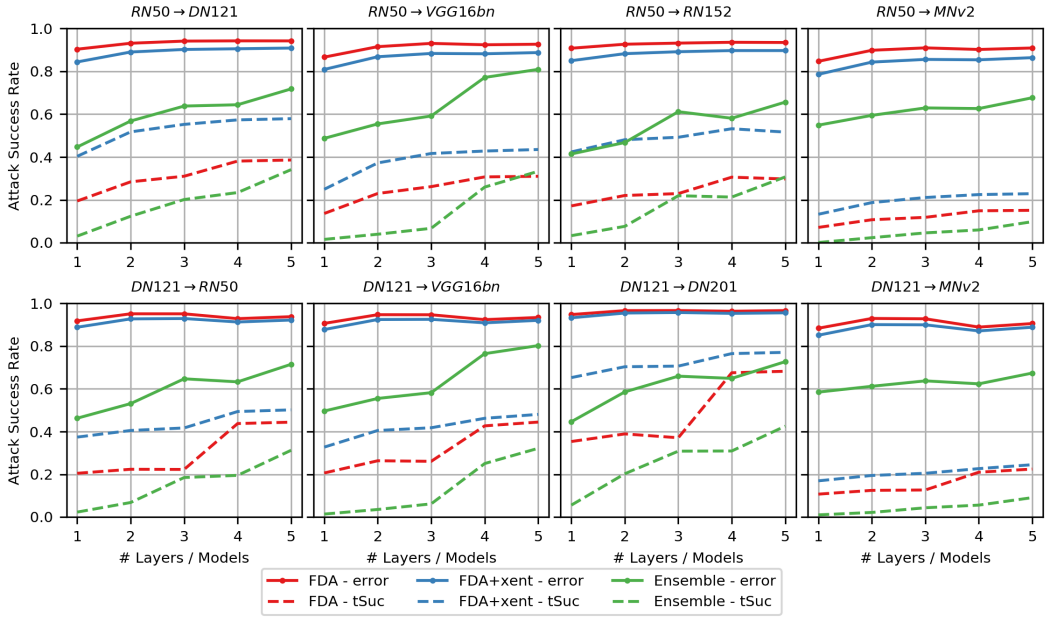

Figure 6: Comparison of FDA methods to ensemble attacks.

Figure 6 shown an extension of the results in Figure 2 to include all of the transfer scenarios considered in Section 4.1.1. The primary result is that our FDA methods, and particularly *FDA*$^{(N)}$+*xent*, outperforms the ensemble methods in all of these transfer scenarios in both error and tSuc. The critical detail is that in these plots, our method is still crafting noise using the feature space of a *single* source model, where the ensemble method is using output layer information from multiple source models of varying architectures and depths. Since FDA and ensemble are seemingly orthogonal methods, we leave it to future work to explore their combination.

D. Extended analysis of disruption

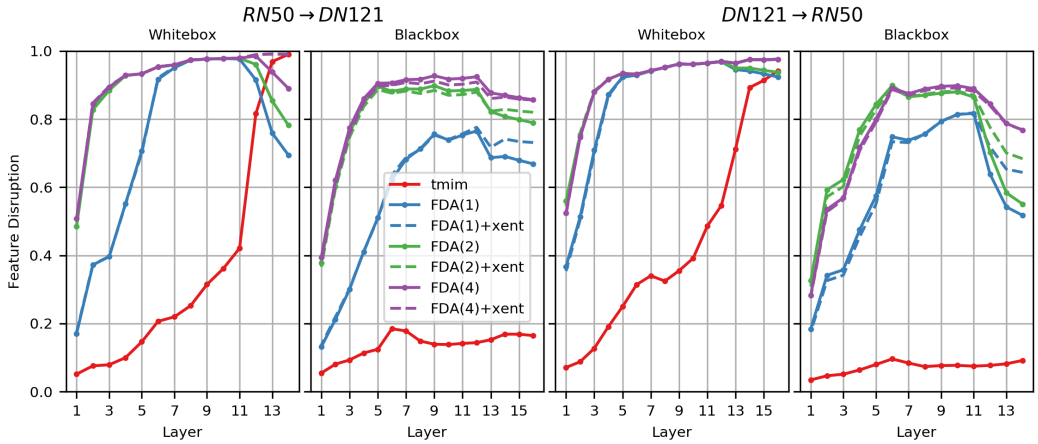

Figure 7: Disruption of features caused by transfer attacks.

Here we extend the disruption results from Section 4.1.4 to include both RN50→DN121 and DN121→RN50. The disruption caused by each attack is measured over 5,000 adversarial examples from each method. Adhering to the setup in Section 4, all source (clean) samples are correctly classified by both the whitebox and blackbox. The results shown here echo the conclusions drawn in Section 4.1.4. The main conclusions are as follows:

- The *TMIM* attack [5], which specifically leverages the output layer of the whitebox model, is only capable of causing high disruption towards the output of the whitebox model. In both the early layers of the whitebox and all throughout the blackbox model, *TMIM* causes very little disruption w.r.t. the target class.
- The baseline $FDA^{(1)}$ method [14] causes significantly more disruption in the intermediate feature spaces of both the whitebox and blackbox but has two undesirable behaviors. First, the intermediate disruption is significantly higher in the whitebox than in the blackbox, even for the layers that were not considered in the attack. Ideally, the intermediate space of the blackbox model would show higher disruption. Second, the disruption in the final few layers of both the whitebox and blackbox has a sharp decrease (as best seen in the RN50 whitebox and blackbox). This indicates that although the intermediate feature maps look to be the target class with nearly 100% probability, that does not automatically induce classification as the target class.
- The primary effect of the *+xent* component is to create higher disruption in the final few layers, which directly addresses the weakness discussed in the previous bullet. This behavior is perhaps best illustrated by comparing the $FDA^{(1)}$ and $FDA^{(1)}$*+xent* results in the rightmost plot (RN50 as a blackbox). From early through middle layers, the disruption is about the same, and in the final 3 layers the *+xent* version has higher disruption. As well as being quite intuitive behavior, this is evidence for why the attack performs better.
- The notable effect of using multi-intermediate-layers is significantly more disruption in the intermediate space of both whitebox and blackbox models. Looking at the whitebox models, as we go from $FDA^{(1)}$ to $FDA^{(2)}$, the disruption is noticeably increased in the earlier layers. This is not surprising, as from Figure 5, the second attack layer to be added in both cases is a layer closer to the input layer. This trend of increased disruption in earlier layers is also observed in the blackbox models. As we move from $FDA^{(2)}$ to $FDA^{(4)}$, from Figure 5 we have added 2 layers that are both deeper in the model than the second layer added, so we do not see increased disruption in the earlier layers. However, we do see increased disruption in the later layers, as expected because the fourth layer added is the closest to the output of the whitebox model. This pattern of how including early layers in the attack set causes early layer disruption and including late layers in the attack set causes late layer disruption matches intuition and provides insight into how the attack works.

E. Cross-distribution experiment

RINet-A	
class name	components
fish	[1, 2, 389, 392, 394]
bird	[10, 13,16, 17, 18]
lizard	[38, 40, 42, 45, 46]
snake	[52, 55, 57, 61, 67]
spider	[72, 75, 76]
dog	[161, 166, 179, 231, 249]
cat	[281, 284, 287, 288, 293]
insect	[302, 304, 308, 311, 313]
boat	[403, 472, 554, 814, 833]
car-truck	[436, 468, 609, 654, 817]
turtle	[33, 34, 35, 36, 37]
big-game	[347, 348, 349, 350, 351]
glassware	[572, 737, 898, 901, 907]
train	[466, 547, 565, 820, 829]
fungus	[992, 993, 994, 995, 996]

RINet-B	
class name	components
fish	[3, 391, 393, 395, 396]
bird	[12, 14, 15, 19, 20]
lizard	[39, 41, 43, 44, 47]
snake	[53, 60, 62, 64, 68]
spider	[73, 74, 77]
dog	[162, 167, 180, 232, 250]
cat	[282, 285, 289, 290, 292]
insect	[303, 305, 310, 312, 315]
boat	[510, 625, 628, 724, 871]
car-truck	[511, 627, 656, 705, 717]
crab	[118, 119, 120, 121, 125]
small-game	[356, 357, 359, 360, 361]
musical-instrument	[402, 420, 486, 546, 594]
computer	[508, 527, 590, 620, 664]
fruit	[948, 949, 950, 951, 954]

RINet-C	
class name	components
bird	[10, 11, 12, 13, 14]
turtle	[33, 34, 35, 36, 37]
lizard	[42, 43, 44, 45, 46]
snake	[60, 61, 62, 63, 64]
spider	[72, 73, 74, 75, 76]
crab	[118, 119, 120, 121, 122]
dog	[205, 206, 207, 208, 209]
cat	[281, 282, 283, 284, 285]
bigcat	[289, 290, 291, 292, 293]
beetle	[302, 303, 304, 305, 306]
butterfly	[322, 323, 324, 325, 326]
monkey	[371, 372, 373, 374, 375]
fish	[393,394, 395, 396, 397]
fungus	[992, 993, 994, 995, 996]
musical-instrument	[402, 420, 486, 546, 594]
sportsball	[429, 430, 768, 805, 890]
car-truck	[609, 656, 717, 734, 817]
train	[466, 547, 565, 820, 829]
clothing	[474, 617, 834, 841, 869]
boat	[403, 510, 554, 625, 628]

Figure 8: Class splits of Restricted-ImageNet subsets.

To evaluate the effect of differences in the training data distributions of the source and target model we propose three splits of the ImageNet-1k dataset [4], which we call RestrictedImageNet-A/B/C (RINet-A/B/C). To split the classes, we leverage the WordNet [20] hierarchical structure of the dataset such that each class in a RINet set is a superclass category composed of multiple ImageNet-1K classes, noted in Figure 8 as "components." For example, the "fish" class of RINet-A (both the train and val parts) is the aggregation of ImageNet-1k classes: `[1:'tench', 2:'goldfish', 389:'barracouta', 392:'rock-beauty', 394:'sturgeon']`. See `https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a` for the number to category translations.

RINet-A&B. One of the key experiments in this work is to measure the transfer performance when the source and target models are trained on similar categories, but have no training data overlap. This represents a more realistic attacking scenario when the adversary is aware of what classes the target model is trained on (or at least most of them) but must collect their own data as they do not have access to the target model's training dataset. To simulate this scenario, we create RINet-A and RINet-B, which each have 15 classes, of which 10 are shared between the two and 5 are unique, and have *zero* training data overlap.

To perform our experiments, we trained a RN50 and DN121 model on each of the splits using code from `https://github.com/pytorch/examples/blob/master/imagenet/main.py`. The accuracy of each model is shown in Figure 9. Since it is not meaningful to test an RINet-A model on the unique classes of RINet-B, we show the test accuracy on the "full" and "shared" classes only when appropriate. For the models trained and tested on the same splits, the accuracy is about 95%. When the models are tested on the shared class data of the other split, the test accuracy drops to about $81\% - 87\%$, which is not surprising given the difference of ImageNet classes used.

	Accuracy			
Model	A-Test-full	A-Test-shared	B-Test-full	B-Test-shared
RN50-A	0.9564	0.9546	*	0.8146
RN50-B	*	0.8387	0.9435	0.935
DN121-A	0.966	0.9658	*	0.8467
DN121-B	*	0.8788	0.9616	0.955

Figure 9: Standard accuracy of RINet models.

RINet-C. The other dataset considered is RINet-C. It has very similar construction to A & B in that each class is an aggregation of five ImageNet-1K classes. However, RINet-C has 20 classes which makes it a bigger subset than RINet-A/B. The purpose of including RINet-C is to measure how important the size of the subset is when attacking in the RN50-A/B/C→DN121-FullINet and RN50-FullINet→DN121-A/B/C scenarios. We train RN50 and DN121 RINet-C models in the same way as we do the RINet-A/B models. The test accuracy of RN50-C is 95.2% and DN121-C is 96.3%.

Cross-distribution attack settings. Lastly, we discuss how we setup and carry out the attacks in the cross-distribution experiments. Here, we assume the auxiliary models have been trained for RN50-A/B/C/FullINet (training details in Appendix B).

- *Constants across all tests:* The attacking parameters are still L_∞ $\epsilon = 16/255$, step_size $= 2/255$, and $perturb\_iters = 10$.
- *RN50-A/B→DN121-A/B (Scenario 1):* Since the source task is significantly different than Full-ImageNet, we re-search for the best 5 attacking layers for the RN50-A/B models. We use the same greedy search technique described in Appendix B only on the RN50-A→DN121-A transfer scenario. In the notation of Figure 5, the sequence of adding attacks layers in the *FDA*$^{(N)}$*+xent* framework is: `1:[3,4,6,1], 2:[3,4,6], 3:[3,4,6,2], 4:[3,4,5], 5:[3,4,4]`. We also find that increasing the weight of the cross-entropy term to $\gamma = 1e-3$ helps. Other attacking parameters stay the same: $\eta = 1e-5, \lambda_\ell = 1/N$. When attacking, we only consider source samples from the shared-classes that are correctly classified by both the source and target models. We then target each of the other 9 shared classes individually and do this for all source samples in the whitebox model's validation dataset. Given the classes are shared, the computation of error and tSuc is straightforward.
- *RN50-A/B/C→DN121-FullINet (Scenario 3):* In these tests, we use the exact layer-set and hyper-parameters used in Case 1. The only tricky part is how to define attack success because the class structures are significantly different. Here, since the whitebox model is RN50-A/B/C, we work in the label space of RINet-A/B/C. In the attack loop, for each source sample in the whitebox model's validation set that is correctly classified by both the whitebox and blackbox, we target each of the other 14 (or 19 in the case of RINet-C) classes individually. An attack is successful if the blackbox model's top-1 prediction is one of the components that makes up the RINet target class. For example, if we use RN50-A as the whitebox model, for the target label "spider" we count a tSuc iff the DN121-FullINet model's prediction is in $[72, 75, 76]$. This is also how we check that the blackbox model is initially correct for each source sample. Note, this is a somewhat restrictive definition of tSuc. If we attack with the target label "dog," we only count a tSuc if the blackbox model's prediction is a component that made up the RINet's dog class. However, ImageNet-1K has over 150 "dog" sub-classes but tSuc only accounts for 5 of these.
- *RN50-FullINet→DN121-A/B/C (Scenario 2):* In these evaluations, we use the attack configurations, layer set, and hyper-parameters from the main ImageNet transfer tests (i.e., we do not tune for this particular test). As in Case 2, since the label spaces of the whitebox and blackbox models are different, we must handle the conversion as to measure attack success. For each source image in the ImageNet-1K validation set, we first check that it is a component of the target model's dataset, then we check that it is correctly classified by both the whitebox and blackbox using their respective label spaces. Using images that passed both these checks, we then target each of the other 14 (or 19) classes in the RINet label space by randomly choosing a Full-ImageNet target label from the component set of the target RINet class. For example, if we want to target "bird" on the RN50-B blackbox model, we would randomly select a label from the component set $[12, 14, 15, 19, 20]$ and use that as the target label in the Full-ImageNet space. With this, computing error and tSuc is straightforward.

Lastly, we would like to emphasize that we did not re-tune all of the hyper-parameters and layer sets for each individual transfer. We mention this to illustrate that the method and results are not ultra-sensitive to these parameters (within reason). However, it would not be surprising if the transfer results improve if we tuned the hyper-parameters for each situation.