Transferable Adversarial Perturbations

Wen Zhou(✉), Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang

Basic Research Group, Security Platform Department, Tencent, Beijing, China
wen8.zhou@gmail.com, hx173149@gmail.com,
{yongjunchen,mengyuntang,angelahuang,xenosgan,coolcyang}@tencent.com

Abstract. State-of-the-art deep neural network classifiers are highly vulnerable to adversarial examples which are designed to mislead classifiers with a very small perturbation. However, the performance of black-box attacks (without knowledge of the model parameters) against deployed models always degrades significantly. In this paper, We propose a novel way of perturbations for adversarial examples to enable black-box transfer. We first show that maximizing distance between natural images and their adversarial examples in the intermediate feature maps can improve both white-box attacks (with knowledge of the model parameters) and black-box attacks. We also show that smooth regularization on adversarial perturbations enables transferring across models. Extensive experimental results show that our approach outperforms state-of-the-art methods both in white-box and black-box attacks.

Keywords: Adversarial perturbations · Transferability
Black-box attacks

1 Introduction

Recently, deep neural networks achieve state-of-the-art performance in many fields such as computer vision [1,2], speech recognition [3], and machine translation [4]. However, recent works [5–9] show that deep neural networks are highly vulnerable to adversarial perturbations of the data. Adversarial examples are modified very slightly in a way that is intended to cause a classifier to misclassify them. Several methods have been proposed to generated adversarial examples using the gradient information of neural networks. Fast Gradient Sign Method (FGSM) [7] and Basic Iterative Method (BIM) [8] serve as two baseline methods to generate adversarial examples with different transfer abilities. FGSM generates adversarial examples by linearizing the cost function around the current parameters of models. It can be computed efficiently using back-propagation.

W. Zhou and X. Hou—Equal contribution.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01264-9_28) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11218, pp. 471–486, 2018.
https://doi.org/10.1007/978-3-030-01264-9_28

However, it usually has a low success rate with white-box attacks since it does not increase the cost function sufficiently with a single step. BIM extends FGSM by taking multiple steps of FGSM. It usually induces higher error rates than the FGSM for white-box attacks since it can produce more harmful adversarial examples without any approximation for the white-box model. However, BIM transfers across models at lower rates and it produces weaker black-box attacks than FGSM, which indicates that BIM tends to *overfit* on the white-box model. Because of the huge search space, both one-step and iterative methods can not search the transferrable perturbations efficiently, where the transferrable perturbations are insensitive to diverse parameters and architectures of models.

We propose a novel adversarial perturbations generation method which enables black-box transfer. We introduce two terms into cost function to guide the search directions of perturbations.

First, we maximize the distances between natural images and their adversarial examples in the intermediate feature maps which can address vanishing gradients for adversarial perturbations generation. Thus, it can search perturbations efficiently with back-propagation. Besides, since large distances in the intermediate feature maps correlate with the large distances in the predictions of neural networks, it will cause error predictions with high probability. We show that it also can increase the probability of successful black-box transfer.

Second, we introduce a regularization term into cost function to remove the high-frequency perturbations, which enables black-box transfer with high error rates. Because of the continuity of the neighboring pixels emerged in data, the convolutional kernels learned by deep neural networks also capture this property. Thus, the high-frequency perturbations are smoothed by these kernels layer by layer without effort, which makes no changes in the final predictions of neural networks. The regularization term reduces the variations of adversarial perturbations and makes them difficult to be smoothed by layer-by-layer convolutions, which enables black-box transfer. Figure 1 gives several adversarial examples generated by FGSM, BIM and the proposed method using Inception-V3 model.

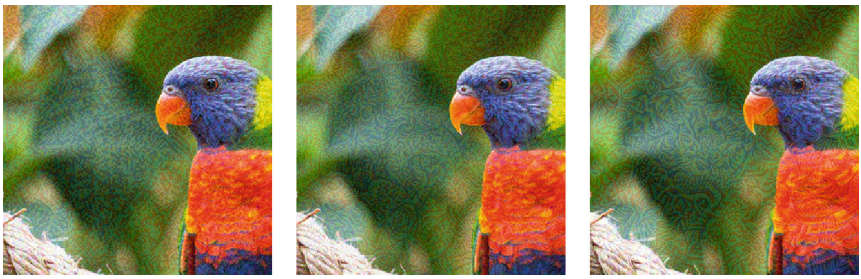

Fig. 1. Different adversarial examples for Inception V3 model using FGSM, BIM and our method respectively.

We evaluate the proposed method on two public datasets with various models including state-of-the-art classifiers [10] and defense models [11]. Experimental results show that our method outperforms state-of-the-art methods both in white-box and black-box attacks.

2 Related Work

In this section, we review some of the relevant work. Szegedy et al. [12] first introduce adversarial examples generation by analyzing the instability of deep neural networks. They show that the adversarial perturbations are more effective than random perturbations for deep neural networks despite of the larger magnitude of random perturbations, which indicates that adversarial examples expose fundamental blind spots of learning algorithms. Goodfellow et al. [7] further explain the phenomenon of adversarial examples by analyzing the linear behavior of deep neural network and propose a simple and efficient adversarial examples generating method: FGSM. It can generalize across models by taking advantage of the linear behavior of neural networks.

Kurakin et al. [8] investigate adversarial examples for large-scale dataset: ImageNet and they compare FGSM and BIM in terms of the robustness of black-box adversarial attacks. They show that BIM with multi-step of FGSM is less transferable than a single-step FGSM despite of the higher error rates for white-box attacks. Kurakin et al. [13] further explore the adversarial examples in the physical world and demonstrate BIM is also not robust to transformations caused by the camera. Both of these two works show multi-step optimization is less transferable than single-step optimization. However, we show that with properly guided gradients, multi-step optimization can achieve higher error rates for both white-box and black-box attacks.

It is natural to use ensemble methods for both inputs and architectures to enable transferring across models. Moosavi-Dezfooli et al. [6] use an ensemble of inputs to seek universal perturbations which generalize across both deep neural networks and inputs. Unlike prior works which compute perturbations for each example independently, they aggregate atomic perturbations to reduce the variations of perturbations. They show that such perturbations generalize well across different classification models. Another type of work uses an ensemble of different architectures to enable transferring across models which is widely used in the competitions. From the defensive point of view, Florian et al. [11] incorporate adversarial examples transferred from other pre-trained models to take the advantage of ensemble of different architectures, which improves the robustness of deep neural networks against black-box attacks. Both of these ensemble methods encourage algorithms to search a *shared* space to reduce the variations of perturbations with high computational complexity. We show that a smooth regularization on perturbations can reduce the variations of perturbations efficiently.

3 Transferable Perturbations

Let $f(x)$ denote an arbitrary deep neural network which takes x ($x \in R^n$) as input and outputs the probability of classes y ($y \in R^m$), we first define an adversarial example fooling the model $f(x)$ for a chosen p-norm and noise parameter ϵ as follows:

$$\tilde{x} = \underset{\|\tilde{x}-x\|^p \leq \epsilon}{\operatorname{argmax}} \, l(\tilde{x}, t) \tag{1}$$

where t and $l(\cdot, \cdot)$ denote the label of x and the loss function used to train the model respectively. In all our experiments, we use the cross entropy as the loss function. FGSM [7,8] and BIM [13] can be used to optimize above function and generate adversarial examples $\tilde{x}$. FGSM finds adversarial perturbations which increase the value of the loss function with one step. BIM extends FGSM which applies it multiple times with a small step size to increase the loss function further. It usually achieves higher success rate than FGSM for white-box attacks. However, it is less transferable than FGSM. To address this, we first maximize the distances between natural images and adversarial examples in feature space to increase transfer rates. Besides, we introduce the smooth regularization on perturbations which punishes the discontinuity of the neighboring pixels.

3.1 Maximizing Distance

Standard neural network architectures are deep hierarchy of convolutions and max-pooling with large capacity. Previous adversarial examples generation methods aim to increase the loss function as defined in (1) using gradient ascent. However, due to the deep hierarchy of architectures, the gradients of loss with respect to input x become extremely small (vanishing gradient problem). Thus, it is insufficient to maximize loss with few steps. To address this issue, we add an intermediate loss which measures the distance of intermediate feature maps between input x and adversarial examples $\tilde{x}$. The gradient of intermediate loss with respect to input x is sufficient to maximize intermediate loss in few steps. Besides, these gradients also provide good guides to maximize the loss function (1). Large distance of intermediate feature maps between x and $\tilde{x}$ will eventually result in large distance in final outputs of neural networks between x and $\tilde{x}$ with high probability, which will increase the loss function (1) and make error predictions.

Let $L(x, d)$ denote the intermediate feature map in layer $d \in D$, we maximize the $l2$-norm distance between $T(L(x, d))$ and $T(L(\tilde{x}, d))$ for all layers, where $T(L(\tilde{x}, d))$ denotes the power normalization [14] of $L(\tilde{x}, d)$. The power normalization is used to down-weight the contribution of large value in $L(x, d)$, which is defined as follows:

$$T(L(\tilde{x}, d)) = \operatorname{sign}(L(\tilde{x}, d)) \odot \operatorname{abs} L(\tilde{x}, d)^{\alpha} \tag{2}$$

where $0 \leq \alpha \leq 1$ is a parameter of the normalization and $\odot$ denotes element-wise production. In our experiments, we find above transformation is quite effective for black-box transfer.

By maximizing the distance of intermediate feature maps between input x and adversarial examples $\tilde{x}$, we generate adversarial examples as follows:

$$\tilde{x} = \underset{\|\tilde{x}-x\|^p \leq \epsilon}{\operatorname{argmax}} \; (l(\tilde{x}, t) + \lambda \sum_{d \in D} \|T(L(x, d)) - T(L(\tilde{x}, d))\|^2), \tag{3}$$

where λ denotes the trade-off between the loss (1) and the intermediate loss. To make the feature in each layer contribute equally, we normalize $L(x, d)$ to $[0, 1]$ using min-max scaling.

3.2 Regularization

Due to the diverse inputs and architectures of deep neural networks, the maximization of loss function used in one architecture does not guarantee the maximization of loss function used in other architectures. Thus, the transfer rate always degrades, especially when transferring to complex architectures such as Inception-Resnet-V2 and ResNet. An ensemble of diverse inputs [6] or architectures can partially solve this problem with high computational complexity. Both of these methods want to remove high-frequency perturbations and reduce the variations of adversarial perturbations to make adversarial examples more transferable. We introduce regularization on perturbations to reduce variations with much more efficiency:

$$\begin{aligned} \tilde{x} = & \underset{\|\tilde{x}-x\|^p}{\operatorname{argmax}} J(\tilde{x}, x, t, w_s) = \underset{\|\tilde{x}-x\|^p \leq \epsilon}{\operatorname{argmax}} \; (l(\tilde{x}, t) \\ & + \lambda \sum_{d \in D} \|T(L(x, d)) - T(L(\tilde{x}, d))\|^2 \\ & + \eta \sum_{i} \operatorname{abs} R_i(\tilde{x} - x, w_s)) \end{aligned} \tag{4}$$

where η controls the balance between regularization and loss function. $R_i(\tilde{x} - x, w_s)$ denotes the i-th element in the response map which is calculated by doing a convolution between the kernel w_s (with size s) and the perturbation $\tilde{x} - x$. w_s is designed to be a box linear filter which is a spatial domain linear filter in which each pixel in the resulting image has a value equal to the average value of its neighboring pixels in the input. It is a form of low-pass filter which enforces the continuity of the neighboring pixels and reduces the variations of adversarial perturbations.

3.3 Optimization

To optimize (4), we use Iterative FGSM (I-FGSM) [11] which iteratively applies FGSM k times with budget $\epsilon' = \epsilon/k$. First, we scale the input x into [−1,1] and

initialize $\tilde{x}_0 = x$. And then, we compute the gradients of loss (4) with respect to input x. After that, the adversarial examples are updated by multiple steps. In each step, we take the sign function of the gradients and clip the adversarial examples into [−1,1] to make valid images. Finally, adversarial examples are calculated by adding the pixel differences between the last updated adversarial examples and the input x with ϵ. Algorithm 1 gives the details of perturbations generation.

Algorithm 1. Computation of transferable perturbations

initialize: $\tilde{x}_0 = x,\ \epsilon' = \epsilon/k,\ i = 0,$
while $i < k$ **do**
$\tilde{x}_{i+1} = \text{clip}(\tilde{x}_i + \epsilon' \,\text{sign}(\nabla_x J(\tilde{x}_i, x, t, w_s)), -1, 1)$
end while
return $\tilde{x} = \text{clip}(x + \epsilon \,\text{sign}(x - \tilde{x}_k), -1, 1)$

4 Experiments

In this section, we describe the implementation details and experimental results on two public datasets. We first analyze the transferability of our method, FGSM, BIM, C&W [23], MI-FGSM [22] and Universal Adversarial Perturbations (**UAP**) on a ImageNet-compatible dataset[1]. This dataset contains 1000 images which are not used in the original ImageNet dataset. To avoid overfitting on above dataset, we perform hyper-parameter selection on a subset of ImageNet dataset [15] which contains 1000 images randomly selected from ILSVRC 2012 validation set [16].

We use diverse architectures including VGG16 [17], Inception V3 [18], Inception V4 [10], Inception-ResNet-v2 [10], ResNet V2 [2,19][2] as defense models. Besides, we average the predicted probabilities of above models as the ensemble model (**Ensemble**). We also use adversarially trained Inception v3 model [8] (**adv-v3**), adversarially trained Inception-ResNet-V2 model (**adv-res-v2**), and adversarially trained Inception v3 with an ensemble of 3 models (**ens3-inc-v3**) and 4 models (**ens4-inc-v3**) [11][3] respectively as defense models. In all our experiments, we report the **recognition accuracy** for comparison. Besides, to address "label leaking" problem [8], we use the prediction of the current model as the ground truth t (4). We use **TAP** to represent the proposed method for short (Table 1).

[1] This dataset can be download at https://github.com/tensorflow/cleverhans/tree/master/examples/nips17_adversarial_competition/dataset.
[2] https://github.com/tensorflow/models/tree/master/research/slim.
[3] https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models.

Table 1. The comparisons of transferability in terms of recognition accuracies. Perturbations are generated using VGG16 and Inception-V3 respectively.

	VGG16	InceptionV3	InceptionV4	Inception ResNet-V2	ResNet-V2	Ensemble
No perturbation	86.8%	96.4%	97.6%	100%	89.6%	99.8%
Random noise	81.3%	91.7%	94.6%	97.8%	84.5%	98.1%
VGG16-TAP	**3.2%**	**23.9%**	28.1%	32.3%	**23.9%**	26.7%
VGG16-FGSM	3.7%	34.9%	44.0%	50.0%	34.7%	46.4%
VGG16-BIM	4.0%	24.2%	**24.5%**	**28.5%**	**23.9%**	**22.7%**
VGG16-UAP	12.4%	31.2 %	32.8%	46.9 %	33.2%	43.7%
Inc-V3-TAP	**29.4%**	**0.0%**	**22.1%**	**24.7%**	**46.9%**	**30.8%**
Inc-V3-FGSM	57.7%	26.9%	70.2%	72.9%	65.7%	75.4%
Inc-V3-BIM	66.0%	0.01%	67.7%	70.2%	76.8%	73.6%
Inc-V3-UAP	39.8%	52.2%	56.4%	63.1%	50.5%	64.6%
Inc-V3-MI-FGSM	45.9%	0.1%	47.3%	50.7%	61.8%	62.5%
Inc-V3-CW	84.9%	24.5%	93.5%	98.6%	86.9%	96.9%

We use FGSM and BIM as our two baseline methods for comparison which are defined as follows:

$$\begin{aligned} FGSM:\ & \tilde{x} = x + \epsilon\,\mathrm{sign}\nabla_x l(\tilde{x}, t) \\ BIM:\ & \tilde{x}_0 = x, \tilde{x}_k = \mathrm{clip}(\tilde{x}_{k-1} + \epsilon\,\mathrm{sign}\nabla_x l(\tilde{x}_{k-1}, t)). \end{aligned} \tag{5}$$

We use implementations of FGSM, BIM and C&W with default parameters from CleverHans [20,21] in our experiments. We also modify CleverHans to implement our approach. We also use the implementation of MI-FGSM [22][4] for comparison. To fairly compared with these methods, the perturbation size is set to 16. We linearly normalize the perturbation size of C&W method [23] to 16 for fair comparison and use simple gradient descent to optimize the objective function defined in [23].

In our experiments, the perturbation size ϵ is set to 16 for all experiments. λ and η are set to 0.05 and 10^3 respectively to make each loss contribute equally. The parameter of the normalization α is set to 0.5 for best performance. The size of kernel w_s in (4) is set to 3 for balancing the smooth term and loss. The number of iterations k is empirically set to 5. We will analyze the effects of these parameters on transferability using the subset of ImageNet dataset as described above.

We first use above models to evaluate the clean images and those images added with random noise perturbations ($\epsilon = 16$). The results are listed in Table 2. Inception-Resnet-V2 (100%) achieves significantly better performance than VGG16 (86.8%) for clean images because of the higher capacity. Both of these models can resist random noise attacks and the models with higher capacity also perform better.

[4] https://github.com/dongyp13/Non-Targeted-Adversarial-Attacks.

Table 2. The comparisons of transferability in terms of recognition accuracies. Perturbations are generated using Inception-ResNet-V2, ResNet-V2 and ResNet-V1 respectively.

	VGG16	InceptionV3	InceptionV4	Inception ResNet-V2	ResNet-V2	Ensemble
Inc-ResV2-TAP	**37.0%**	25.9%	**33.2%**	4.8%	53.3%	48.2%
Inc-ResV2-FGSM	59.4%	69.0%	76.5%	57.2%	71.7%	78.7%
Inc-ResV2-BIM	48.9%	41.5%	51.5%	1.2%	60.4%	54.5%
Inc-ResV2-MI-FGSM	38.8%	**25.3**%	**33.2%**	**0.1%**	**51.6%**	**46.3%**
Inc-ResV2-CW	83.4%	91.7%	92.4%	49.0%	85.6%	93.5%
ResNet-V2-TAP	**31.8%**	48.2%	55.7%	55.5%	7.6%	**47.4%**
ResNet-V2-FGSM	37.3%	56.3%	64.8%	66.8%	14.6%	63.3%
ResNet-V2-BIM	44.8%	53.2%	62.0%	63.8%	**4.4%**	54.3%
ResNet-V2-MI-FGSM	46.3%	**45.2%**	**51.6%**	**55.2%**	24.1%	56.2%
ResNet-V2-CW	84.0%	94.5%	96.4%	99.5%	37.7%	98.5%
ResNet-V1-TAP	**20.2**%	**38.1**%	**48.7**%	**49.1**%	**25.3**%	**44.4**%
ResNet-V1-UAP	35.3%	41.6%	50.2%	57.8%	40.3%	56.8%
ResNet-V1-MI-FGSM	65.3%	74.3%	78.7%	82.0%	71.3%	86.8%
ResNet-V1-CW	86.9%	96.0%	97.5%	99.9%	89.4 %	99.6%

The Transferability Across Models. We first use VGG16, Inception-V3, Inception-V4, Inception-ResNet-V2, ResNet-V2 and an ensemble of these models as defense models. We iteratively generate perturbations using one model and report the recognition accuracy on all these models. For each iteration, we use all feature maps for the selected model except ResNet-V1 and ResNet-V2 because of the large number of layers. We use the feature maps: "block3/uint_23" $\sim$ "block3/uint_36" and "block4/uint_3" in ResNet-V2 and "block1" and "block2" in ResNet-V1 for adversarial examples generation. To demonstrate the superiority of the proposed method, we also compare our method with FGSM, BIM and UAP respectively. From Table 2 we can see that, our method achieves 3.2%, 0.0%, 4.8% and 7.6% accuracy for VGG16, Inception-V3, Inception-ResNet-V2 and ResNet-V2 respectively for white-box attacks, which are significantly better than FGSM, BIM, UAP, MI-FGSM and C&W except ResNet-V2-BIM, Inc-ResV2-MI-FGSM and Inc-ResV2-BIM. As for black-box transfer, our method achieves lowest recognition accuracy using Inception-V3 and ResNet-V1 to generate adversarial perturbations. Our method also get comparable or better results using Inception-ResNet-V2 and ResNet-V2 with MI-FGSM. Notably, using VGG16 model to generate perturbations has the highest transfer rate for all methods. We find the gradients of loss with respect to inputs for VGG model are several orders of magnitude larger than those gradients for more complex architectures because of the relatively small number of layers. Our method is unable to gain more benefits for VGG model.

Table 3. The robustness of TAP, FGSM, BIM and UAP to adversarially trained models.

	adv-v3	adv-res-v2	ens3-inc-v3	ens4-inc-v3
VGG16-TAP	**38.8%**	63.8%	**41.9%**	47.3%
VGG16-FGSM	50.9%	71.1%	56.1%	58.5%
VGG16-BIM	57.3%	73.6%	53.5%	55.4%
VGG16-UAP	39.4%	**57.3%**	47.4%	**43.9%**
Inc-V3-TAP	**52.8%**	**68.8%**	**60.9%**	**59.8%**
Inc-V3-FGSM	72.1%	93.6%	85.1%	86.4%
Inc-V3-BIM	82.4%	93.9%	88.2%	88.5%
Inc-V3-UAP	65.5%	82.4%	77.0%	76.9%
Inc-V3-MI-FGSM	74.3%	90.6%	80.7%	82.0%
Inc-V3-CW	93.0%	96.4%	92.3%	90.0%
Inc-ResV2-TAP	**60.5%**	**87.8%**	**79.1%**	**82.1%**
Inc-ResV2-FGSM	73.9%	92.7%	86.9%	87.3%
Inc-ResV2-BIM	70.8%	92.9%	84.8%	86.9%
Inc-ResV2-MI-FGSM	66.9%	83.6%	71.8%	73.4%
Inc-ResV2-CW	91.8%	94.9%	91.9%	89.3%
ResNet-V2-TAP	**49.2%**	**64.1%**	**57.8%**	**56.0%**
ResNet-V2-FGSM	62.1%	85.7%	77.4%	77.8%
ResNet-V2-BIM	64.7%	82.6%	72.3%	74.7%
ResNet-V2-MI-FGSM	71.1%	86.6%	76.9%	77.9%
ResNet-V2-CW	94.0%	96.3%	92.8%	90.5%
ResNet-V1-TAP	**50.2%**	**64.4%**	**55.5%**	**57.7%**
ResNet-V1-UAP	60.4%	77.9%	68.8%	66.1%
ResNet-V1-MI-FGSM	84.5%	93.4%	90.3%	90.2%
ResNet-V1-CW	95.0%	97.5%	94.2%	91.8%

We also evaluate the robustness of TAP to adversarially trained models as shown in Table 3. We show that VGG16 model still gives the highest transfer rate for black-box transfer. Kurakin et al. [8] show that adversarial training provides robustness to adversarial examples generated using one-step methods such as FGSM, but it can not help much against iterative methods such as BIM. They also show that adversarial examples generated by iterative methods are less likely to be transferred across models. However, we show that our method, which generates adversarial examples using iterative methods, still enables adversarial examples transferring across models because of the guided gradients.

We calculate the regularization term of (4): $\sum_i \text{abs}\, R_i(\tilde{x} - x, w_s)$ to inspect the low-frequency information captured by FGSM, BIM and TAP respectively. The values of regularization term in Fig. 1 are 3.72×10^5, 2.71×10^5 and 5.35×10^5

respectively. Since we use low-pass filter w_s for convolution, larger value of the regularization term means that the neighbor pixels vary more smoothly.

The Influence of Loss Functions. We remove two terms in the loss function (4) for each time to inspect the influence of these terms on performance. We remove the second and third terms by setting λ and η to 0 respectively. Figure 2 shows the performance of the proposed method which generates adversarial examples using Inception-V3 with different conditions. From Fig. 2 we can see that, adversarial training can resist adversarial examples. For example, adversarially trained Inception-ResNet-V2 (adv-res-v2) gets much higher recognition accuracy than original Inception-ResNet-V2.

With $\lambda = 0$ and $\eta = 0$, the proposed method degrades to BIM which has the highest recognition accuracy for all defense models, which indicates these defense models are robust to adversarial examples generated using BIM.

By adding the regularization term in (4) ($\lambda = 0$ and $\eta = 10^3$), we show that such operation reduces the recognition accuracy for all adversarially trained models (ens3-inc-v3, ens4-inc-v3, adv-res-v2, adv-v3) consistently. Such operation can also reduce the recognition accuracy for ResNet-V2 and the ensemble model, and it performs slightly worse on Inception-V4 and Inception-ResNet-V2 than BIM. Since it imposes restrictions on perturbations with smooth regularization, it removes the subtle and optimal perturbations for white-box attack (Inception-V3). Thus, it has higher recognition accuracy than BIM.

With $\lambda = 0.05$ and $\eta = 0$, the proposed method takes feature distances into consideration. We show that the proposed method achieves slightly lower accuracy on adversarially trained models than BIM. It reduces the recognition accuracy significantly on Inception-V3, Inception-V4 and Inception-ResNet-V2, ResNet-V2 and the ensemble model. Notably, for white-box attacks, it achieves extremely low recognition rate on Inception-V3.

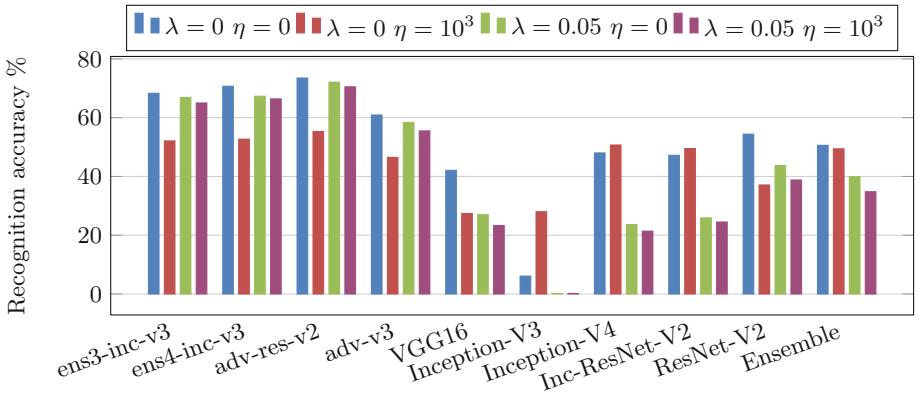

Fig. 2. Recognition accuracies of the proposed method with different configurations of λ and η. Inception-V3 is used to generate adversarial examples.

From above experiments we can see that, adversarially trained models (ens3-inc-v3, ens4-inc-v3, adv-res-v2 and adv-v3) and original models (Inception-V3, VGG16, Inception-V4, Inception-ResNet-V2, ResNet-V2 and Ensemble) are very complementary. Adversarial training injects adversarial examples from other models into the training set and it tends to solve the weaknesses of original models. Two terms in (4) behave differently on these two types of models. We set $\lambda = 0.05$ and $\eta = 10^3$ to balance the performance on these two types of models.

The Sensibility of Parameters on Transferability. To analyze the sensibility of parameters on transferability, we vary one parameter and fix other parameters to report recognition accuracies on defense models. λ varies from 5×10^{-4} to 100 ($\lambda \in \{5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 0.01, 0.02, 0.05, 0.1, 0.5, 1, 2, 5, 10, 50, 100\}$) and α and η are set to 0.5 and 10^3 respectively. The recognition accuracies on defense models are reported in Fig. 3. With increasing value of λ, the adversarial examples generated by our method perform differently on two groups of models (with and without adversarial training). Adversarial examples are more difficult to be transferred to adversarially trained models with larger λ while it is easier for those models without adversarial training.

Figure 4 shows the recognition accuracies with varied α ($\alpha \in \{0, 0.5, 1, 1.5, 2\}$). We observe that there is an optimal value $\alpha = .5$ of α yielding best robustness. Figure 5 presents the performance of the proposed methods with different size of w_s as in (4). With large s, the transfer rates of the proposed method degrades for all models slightly and consistently. Thus, we choose $s = 3$ for best performance.

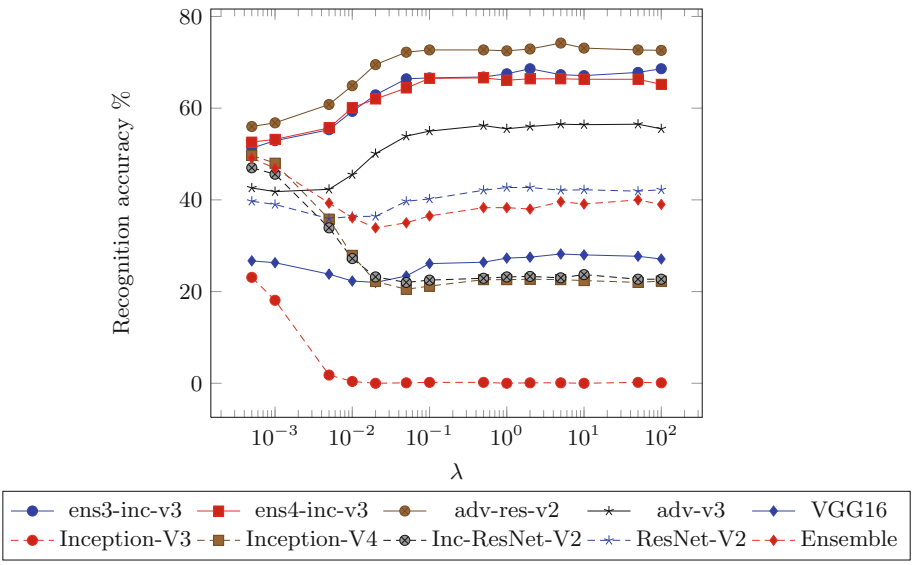

Fig. 3. Recognition accuracies with varied λ.

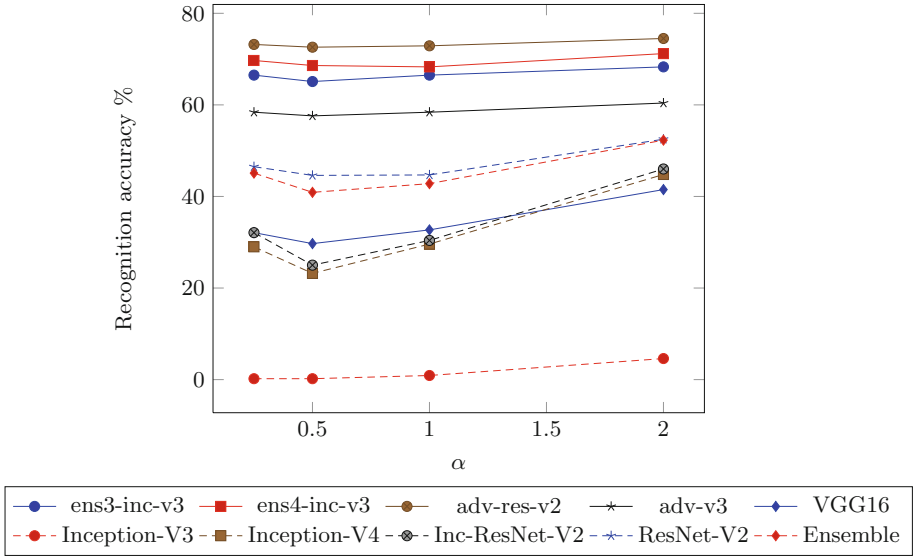

Fig. 4. Recognition accuracies with varied α.

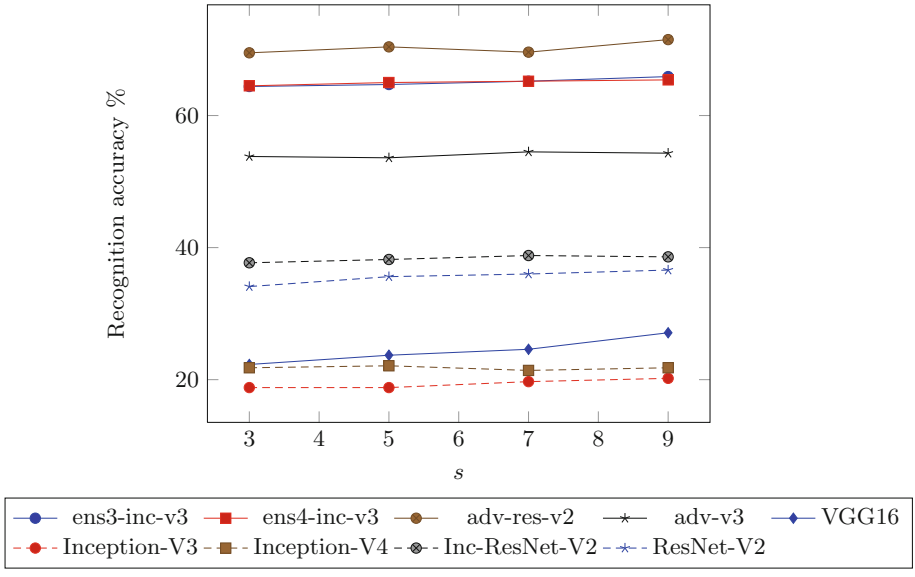

Fig. 5. Recognition accuracies with varied size of w_s.

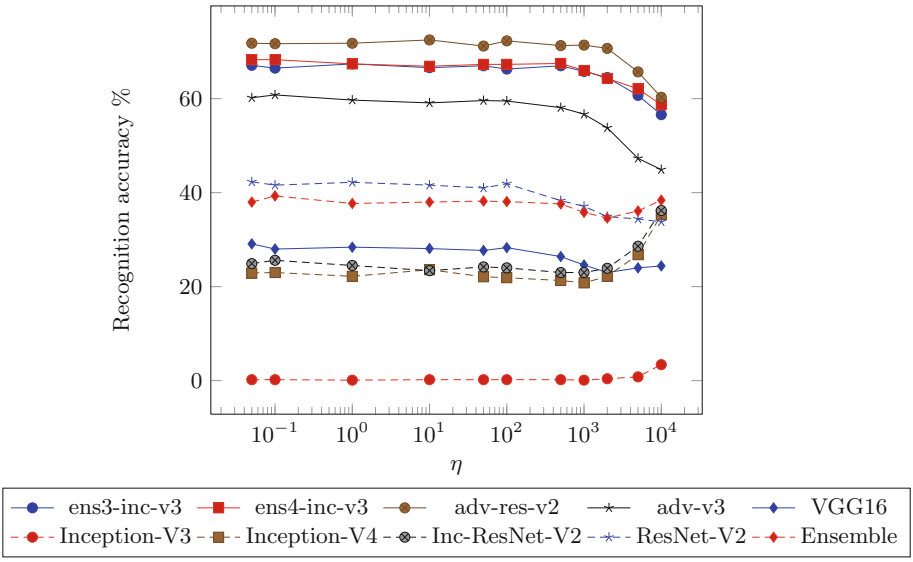

Fig. 6. Recognition accuracies with varied η.

We vary η from 0.05 to 10^4 ($\eta \in \{0.05, 0.1, 1, 10, 50, 100, 500, 10^3, 2 \times 10^3, 5 \times 10^3, 10^4\}$) and observe two distinct patterns for two types of models in Fig. 6. With increasing value of η, the recognition accuracies of adversarial examples on adversarially trained models decrease, which means that adversarially trained models are less robust to adversarial examples with larger value of η. However, with larger value of η, the proposed method performs better on those models without adversarial training.

t-SNE Visualization of Adversarial Examples. To demonstrate the influence of adversarial examples intuitively, we visualize the extracted features of black-box model as shown in Fig. 7. Specifically, we generate adversarial examples using Inception-V3 and extract 1536-dimensional features using Inception-V4 in the penultimate layer. We use t-SNE to compute a 3-dimensional embedding that respects the high-dimensional (L2) distances. Adversarial examples of our method and other two baseline methods are connected to the clean image by straight lines with red and blue color respectively.

By adding perturbations into clean images, it causes the perturbations in the penultimate feature space which will result in error predictions. For white-box attacks, it is clear that we optimize the objective function to find the perturbations in pixels to maximize the perturbations in the penultimate feature space directly. It will cause the extreme low accuracies on white-box models. For black-box attacks, such perturbations in pixels generated using one model can not transfer to the perturbations in the penultimate feature spaces of another model effectively because of different architectures.

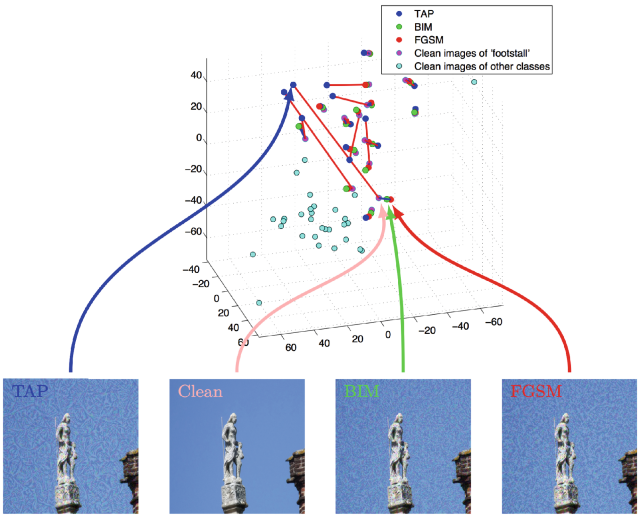

Fig. 7. t-SNE visualization of features which are extracted using Inception-V4 from clean images and adversarial examples (generated by FGSM, BIM and TAP (our method) on Inception-V3 respectively). The distances between adversarial examples generated by TAP and clean images in the embedding space are much larger than those distances corresponding to FGSM and BIM, which means the perturbations generated by TAP are more transferrable than FGSM and BIM.

Both FGSM and BIM can perturb adversarial examples in the penultimate feature space to make error predictions for Inception-V4 as shown in Fig. 7. However, Table 2 shows that Inception-V4 model can still achieve 70.2% and 67.7% on those adversarial examples generated using FGSM and BIM respectively. It means that most of the perturbations in pixels can not cause large enough perturbations in the penultimate feature space to make error predictions for Inception-V4.

As shown in Fig. 7, the distances between adversarial examples generated by our method and clean images are larger than those corresponding distances of FGSM and BIM. It will cause error predictions in Inception-V4 with high probability, which is validated by Table 2. We show that our method can generate the perturbations in pixels using Inception-V3 which can transfer to the perturbations in the penultimate feature space of Inception-V4 with high probability.

5 Conclusion

In this paper, we propose a novel transferable adversarial perturbations generating method to fool deep neural networks. We use two extra penalty terms to guide the search directions efficiently. We show that maximizing the distance of intermediate feature maps between inputs and adversarial examples makes adversarial examples transfer across models. In addition, we observe that

a smooth regularization can enable black-box transfer by reducing the variations of adversarial perturbations. We further use t-SNE to visualize the correlations between transferable ability and the distance in the penultimate feature space which also provides a insight for future research.

Acknowledgments. We thank Zhifei Yue and Shuisheng Liu for helpful discussions and suggestions.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
3. Hinton, G., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Sig. Process. Mag. **29**(6), 82–97 (2012)
4. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
5. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 427–436 (2015)
6. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. arXiv preprint arXiv:1610.08401 (2016)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
8. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. arXiv: Computer Vision and Pattern Recognition (2016)
9. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582 (2016)
10. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. In: AAA, pp. 4278–4284 (2017)
11. Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., McDaniel, P.: Ensemble adversarial training: attacks and defenses. arXiv preprint arXiv:1705.07204 (2017)
12. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
13. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. arXiv: Computer Vision and Pattern Recognition (2016)
14. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_11
15. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)

16. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015)
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
18. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
19. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
20. Nicolas, P., et al.: cleverhans v2.0.0: an adversarial machine learning library. arXiv preprint arXiv:1610.00768 (2017)
21. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
22. Dong, Y., et al.: Boosting adversarial attacks with momentum. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018
23. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. arXiv preprint arXiv:1608.04644 (2016)