Regional Homogeneity: Towards Learning Transferable Universal Adversarial Perturbations Against Defenses

Yingwei Li[1(✉)], Song Bai[2], Cihang Xie[1], Zhenyu Liao[3], Xiaohui Shen[4], and Alan Yuille[1]

[1] Johns Hopkins University, Baltimore, USA
yingwei.li@jhu.edu
[2] University of Oxford, Oxford, UK
[3] Kuaishou Technology, Palo Alto, USA
[4] ByteDance Research, Mountain View, USA

Abstract. This paper focuses on learning transferable adversarial examples specifically against defense models (models to defense adversarial attacks). In particular, we show that a simple universal perturbation can fool a series of state-of-the-art defenses.

Adversarial examples generated by existing attacks are generally hard to transfer to defense models. We observe the property of regional homogeneity in adversarial perturbations and suggest that the defenses are less robust to regionally homogeneous perturbations. Therefore, we propose an effective transforming paradigm and a customized gradient transformer module to transform existing perturbations into regionally homogeneous ones. Without explicitly forcing the perturbations to be universal, we observe that a well-trained gradient transformer module tends to output input-independent gradients (hence universal) benefiting from the under-fitting phenomenon. Thorough experiments demonstrate that our work significantly outperforms the prior art attacking algorithms (either image-dependent or universal ones) by an average improvement of 14.0% when attacking 9 defenses in the transfer-based attack setting. In addition to the cross-model transferability, we also verify that regionally homogeneous perturbations can well transfer across different vision tasks (attacking with the semantic segmentation task and testing on the object detection task). The code is available here: https://github.com/LiYingwei/Regional-Homogeneity.

Keywords: Transferable adversarial example · Universal attack

1 Introduction

Deep neural networks are demonstrated vulnerable to adversarial examples [66], crafted by adding imperceptible perturbations to clean images. The variants of

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58621-8_46) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12356, pp. 795–813, 2020.
https://doi.org/10.1007/978-3-030-58621-8_46

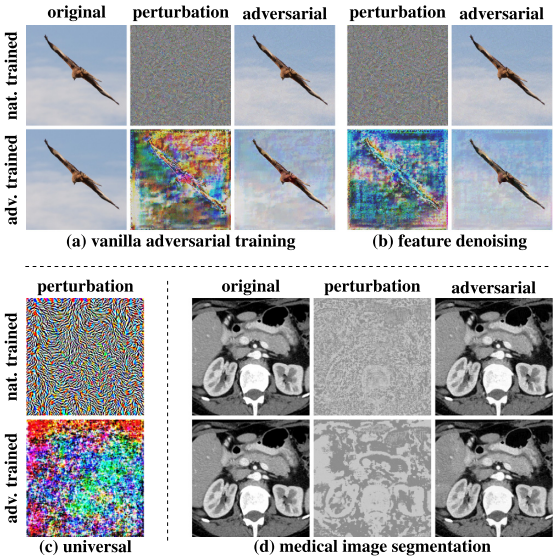

Fig. 1. Illustration of region homogeneity property of adversarial perturbations by white-box attacking naturally trained models (top row) and adversarially trained models (bottom row). The adversarially trained models are acquired by (a) vanilla adversarial training [48,75], (b) adversarial training with feature denoising [75], (c) universal adversarial training [60], and (d) adversarial training for medical image segmentation [41]

adversarial attacks [3,5,9,12,20,21,30,32,33,55,62,63,79] cast a security threat when deploying machine learning systems. To mitigate this, large efforts have been devoted to adversarial defense [7,34,47,71], via adversarial training [42,48, 67,68,73,75,76], randomized transformation [14,26,45,74] *etc.*.

The focus of this work is to attack defense models, especially in the transfer-based attack setting where models' architectures and parameters remain unknown to attackers. In this case, the adversarial examples generated for one model, which possess the property of "transferability", may also be misclassified by other models. To the best of our knowledge, learning transferable adversarial examples for attacking defense models is still an open problem.

Our work stems from the observation of *regional homogeneity* on adversarial perturbations in the white-box setting. As Fig. 1(a) shows, we plot the adversarial perturbations generated by attacking a naturally trained Resnet-152 [28] model (top) and an representative defense one (*i.e.*, an adversarially trained model [48,75]). It suggests that the patterns of two kinds of perturbations are visually different. Concretely, the perturbations of defense models reveal a coarser level of granularity, and are more locally correlated and more structured than that of the naturally trained model. The observation also holds when attacking different defense models (*e.g.*, adversarial training with feature denoising [75], Fig. 1(b)), generating different types of adversarial examples (image-

dependent or universal perturbations [60], Fig. 1(c)), or tested on different data domains (CT scans [57], Fig. 1(d)).

Motivated by this observation, we suggest that *regionally homogeneous perturbations* are strong in attacking defense models, which is especially helpful to learn transferable adversarial examples in the transfer-based attack setting. Hence, we propose to transform the existing perturbations (those derive from differentiating naturally trained models) to the regionally homogeneous ones. To this end, we develop a novel transforming paradigm (Fig. 2) to craft regionally homogeneous perturbations, and accordingly a gradient transformer module (Fig. 3), to encourage local correlations within the pre-defined regions.

The proposed gradient transformer module is quite light-weight, with only $12 + 2K$ trainable parameters in total, where a 3×3 convolutional layer (bias enabled) incurs 12 parameters and K is the number of region partitions. According to our experiments, it leads to under-fitting (large bias and small variance) if the module is trained with a large number of images. In general vision tasks, an under-fitting model is undesirable. However in our case, once the gradient transformer module becomes quasi-input-independent (*i.e.*, aforementioned large bias and small variance), it will output a nearly fixed pattern whatever the input is. Then, our work is endowed with a desirable property, *i.e.*, seemingly training to generate image-dependent perturbations, yet get the universal ones. We note our mechanism is different from other universal adversarial generations [50,54] as we do not explicitly force the perturbation to be universal.

Comprehensive experiments are conducted to verify the effectiveness of the proposed regionally homogeneous perturbation (RHP). Under the transfer-based attack setting, RHP successfully attacks 9 latest defenses [26,35,42,48,68,74,75] and improves the top-1 error rates by 21.6% in average, where three of them are the top submissions in the NeurIPS 2017 defense competition [39] and the Competition on Adversarial Attacks and Defenses 2018. Compared with the state-of-the-art attack methods, RHP not only outperforms universal adversarial perturbations (*e.g.*, UAP [50] by 19.2% and GAP [54] by 15.6%), but also outperforms image-dependent perturbations (FGSM [23] by 12.9%, MIM [16] by 12.6% and DIM [16,77] by 9.58%). The achievement over image-dependent perturbations is especially valuable as it is known that image-dependent perturbations generally perform better as they utilized information from the original images. Since it is universal, RHP is more general (natural noises are not related to the target image), more efficient (without additional computational power), and more flexible (*e.g.*, without knowing the target image, people can stick a pattern on the lens to attack artificial intelligence surveillance cameras).

Moreover, we also evaluate the cross-task transferability of RHP and demonstrate that RHP generalizes well in cross-task attack, *i.e.*, attacking with the semantic segmentation task and testing on the object detection task.

2 Related Work

Transfer-Based Attacks. Practically, attackers cannot easily access the internal information of target models (including its architecture, parameters and outputs). A typical solution is to generate adversarial examples with strong transferability. Szegedy *et al.* [66] first discuss the transferability of adversarial examples that the same input can successfully attack different models. Liu *et al.* [46] then develop a stronger attack to successfully circumvent an online image classification system with ensemble attacks, which is later analysed by [44]. Based on one of the most well-known attack methods, Fast Gradient Sign Method (FGSM) [23] and its iteration-based version (I-FGSM) [38], many follow-ups are then proposed to further improve the transferability by adopting momentum term [16], smoothing perturbation [80], constructing diverse inputs [77], augmenting ghost models [40] and smoothing gradient [17], respectively. Recent works [4,49,52–54,72] also suggest to train generative models for creating adversarial examples. Besides transfer-based attacks, query-based [5,8,11,25,78] attacks are also very popular black-box attack settings.

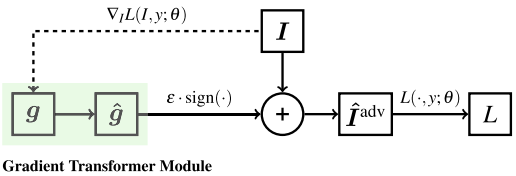

Fig. 2. Illustration of the transforming paradigm, where I is an original image with the corresponding label y, and the gradient g is computed from the naturally trained model θ. Our work learns a mapping to transform gradient from g to $\hat{g}$

Universal Adversarial Perturbations. Above are all image-dependent perturbation attacks. Moosavi-Dezfooli *et al.* [50] craft universal perturbations which can be directly added to any test images to fool the classifier with a high success rate. Poursaeed *et al.* [54] propose to train a neural network for generating adversarial examples by explicitly feeding random noise to the network during training. After obtaining a well-trained model, they use a fixed input to generate universal adversarial perturbations. Researchers also explore to produce universal adversarial perturbations by different methods [36,51] or on different tasks [29,54]. All these methods construct universal adversarial perturbations explicitly or data-independently. Unlike them, we provide an implicit data-driven alternative to generate universal adversarial perturbations.

Defense Methods. Xie *et al.* [74] and Guo *et al.* [26] break transferability by applying input transformation such as random padding/resizing [74], JPEG compression [18], and total variance minimization [59]. Injecting adversarial examples during training improves the robustness of deep neural network, termed as

adversarial training. These adversarial examples can be pre-generated [42,68] or generated on-the-fly during training [35,48,75]. Adversarial training is also applied to universal adversarial perturbations [1,60].

Normalization. To induce regionally homogeneous perturbations, our work resorts to a new normalization strategy. This strategy appears similar to some normalization techniques, such as batch normalization [31], layer normalization [2], instance normalization [69], group normalization [70], *etc.*. While these techniques aim to help the model converge faster and speed up the learning procedure for different tasks, the goal of our proposed region norm is to explicitly enforce the region structure and build homogeneity within regions.

3 Regionally Homogeneous Perturbations

As shown in Sect. 1, regionally homogeneous perturbations appear to be strong in attacking defense models. To acquire regionally homogeneous adversarial examples, we propose a gradient transformer module to generate regionally homogeneous perturbations from existing regionally non-homogeneous perturbations (*e.g.*, perturbations in the top row of Fig. 1). In the following, we detail the transforming paradigm in Sect. 3.1 and the core component called gradient transformer module in Sect. 3.2, respectively. In Sect. 3.3, we observe an underfitting phenomenon and illustrate that the proposed gradient transformer module becomes quasi-input-independent, which benefits crafting universal adversarial perturbations.

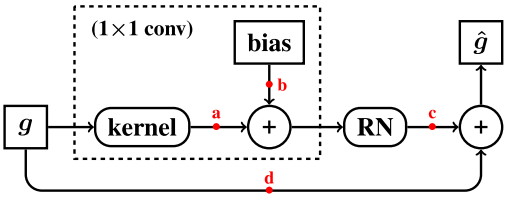

Fig. 3. Structure of the gradient transformer module, which has a newly proposed Region Norm (RN) layer, 1×1 convolutional layer (bias enabled) and identity mapping. We insert four probes (a, b,c and d) to assist analysis in Sect. 3.3 and Sect. 4.2

3.1 Transforming Paradigm

To learn regionally homogeneous adversarial perturbations, we propose to use a shallow network T, which we call gradient transformer module, to transform the gradients that are generated by attacking naturally trained models.

Concretely, we consider Fast Gradient Sign Method (FGSM) [23] which generates adversarial examples by

$$I^{\text{adv}} = I + \epsilon \cdot \text{sign}\left(\nabla_I L(I, y; \theta)\right), \tag{1}$$

where L is the loss function of the model θ, and sign$(\cdot)$ denotes the sign function. y is the ground-truth label of the original image I. FGSM ensures that the generated adversarial example I^{adv} is within the ϵ-ball of I in the L_∞ space.

Based on FGSM, we build pixel-wise connections via the additional gradient transformer module T, so that we may have regionally homogeneous perturbations. Therefore, Eq. (1) becomes

$$I^{\text{adv}} = I + \epsilon \cdot \text{sign}\left(T(\nabla_I L(I, y; \theta); \theta_T)\right), \tag{2}$$

where θ_T is trainable parameter of gradient transformer module T, and we omit θ_T where possible for simplification. The challenge we are facing now is how to train the gradient transformer module $T(\cdot)$ with the limited supervision. We address this by proposing a new transforming paradigm illustrated in Fig. 2. It consists of four steps, as we 1) compute the gradient $g = \nabla_I L(I, y; \theta)$ by attacking the naturally trained model θ; 2) get the transformed gradient $\hat{g} = T(g; \theta_T)$ via the gradient transformer module; 3) construct the adversarial image $\hat{I}$ by adding the transformed perturbation to the clean image I, forward $\hat{I}$ to the same model θ, and obtain the classification loss $L(\hat{I}^{\text{adv}}, y; \theta)$; and 4) freeze the clean image I and the model θ, and update the parameters θ_T of $T(\cdot; \theta_T)$ by **maximizing** $L(\hat{I}^{\text{adv}}, y; \theta)$. The last step is implemented via stochastic gradient ascent (*e.g.*, we use the Adam optimizer [37] in our experiments).

With the new transforming paradigm, one can potentially embed desirable properties via using the gradient transformer module $T(\cdot)$, and in the meantime, keep a high error rate on the model θ. As we will show below, $T(\cdot)$ is customized to generate regionally homogeneous perturbations specially against defense models. Meanwhile, since we freeze the most part of the computation graph and leave a limited number of parameters (that is θ_T) to optimize, the learning procedure is very fast.

3.2 Gradient Transformer Module

With the transforming paradigm aforementioned, we introduce the architecture of the core module, termed as gradient transformer module. The gradient transformer module aims at increasing the correlation of pixels in the same region, therefore inducing regionally homogeneous perturbations. As shown in Fig. 3, given a loss gradient g as the input, the gradient transformer module $T(\cdot)$ is

$$\hat{g} = T(g; \theta_T) = \text{RN}\left(\text{conv}(g)\right) + g, \tag{3}$$

where conv$(\cdot)$ is a 1×1 convolutional layer and RN$(\cdot)$ is the newly proposed region norm layer. θ_T is the module parameters, which goes to the region norm layer (γ and β below) and the convolutional layer. A residual connection [28] is also incorporated. Since RN$(\cdot)$ is initialized as zero [24], the residual connection allows us to insert the gradient transformer module into any gradient-based attack methods without breaking its initial behavior (*i.e.*, the transformed gradient $\hat{g}$ initially equals to g). Since the initial gradient g is able to craft stronger adversarial example (compared with random noises), the gradient transformer

module has a proper initialization. The region norm layer consists of two parts, including a region split function and a region norm operator.

Region Split Function splits an image (or equivalently, a convolutional feature map) into K regions. Let $r(\cdot,\cdot)$ denote the region split function. The input of $r(\cdot,\cdot)$ is a pixel coordinate while the output is an index of the region which the pixel belongs to. With a region split function, we can get a partition $\{P_1, P_2, ..., P_K\}$ of an image, where $P_k = \{(h,w) \mid r(h,w) = k, 1 \leq k \leq K\}$.

In Fig. 4, we show 4 representatives of region split functions on a toy 6×6 image, including 1) vertical partition $(h,w) = w$, 2) horizontal partition $r(h,w) = h$, 3) grid partition $r(h,w) = \lfloor h/3 \rfloor + 2\lfloor w/3 \rfloor$, and 4) slash partition (parallel to an increasing line with the slope equal to 0.5).

Region Norm Operator links pixels within the same region P_k, defined as

$$y_i = \gamma_k \bar{x}_i + \beta_k, \quad \bar{x}_i = \frac{1}{\sigma_k}(x_i - \mu_k), \tag{4}$$

where x_i and y_i are the i-th input and output, respectively. And $i = (n, c, h, w)$ is a 4D vector indexing the features in (N, C, H, W) order, where N is the batch axis, C is the channel axis, and H and W are the spatial height and width axes. We define S_k as a set of pixels that belong to the region P_k, that is, $r(h,w) = k$.

μ_k and σ_k in Eq. (4) are the mean and standard deviation (std) of the k^{th} region, computed by

$$\begin{aligned} \mu_k &= \frac{1}{m_k} \sum_{j \in S_k} x_j, \\ \sigma_k &= \sqrt{\frac{1}{m_k} \sum_{j \in S_k} (x_j - \mu_k)^2 + \text{const}}, \end{aligned} \tag{5}$$

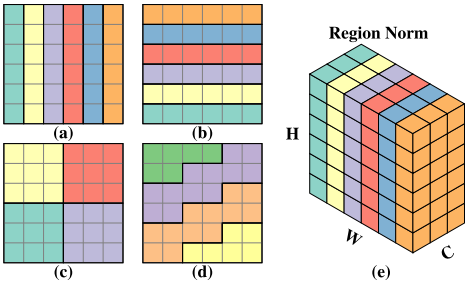

Fig. 4. Toy examples of region split functions, including (a) vertical partition, (b) horizontal partition, (c) grid partition, and (d) slash partition. (e) illustrates the region norm operator with the region split function (a), where C is the channel axis, H and W are the spatial axes. Each pixel indicates an N-dimensional vector, where N is the batch size

where const is a small constant for numerical stability. m_k is the size of S_k. Here $m_k = NC|P_k|$ and $|\cdot|$ is the cardinality of a given set. In the testing phase, the moving mean and moving std during training are used instead. Since we split the image to regions, the trainable scale γ and shift β in Eq. (4) are also learned per-region.

We illustrate the region norm operator in Fig. 4(e). To analyze the benefit, we compute the derivatives as

$$\begin{aligned} \frac{\partial L}{\partial \beta_k} &= \sum_{j \in S_k} \frac{\partial L}{\partial y_j}, \quad \frac{\partial L}{\partial \gamma_k} = \sum_{j \in S_k} \frac{\partial L}{\partial y_j} \bar{x}_j, \\ \frac{\partial L}{\partial x_i} &= \frac{1}{m_k \sigma_k} (m_k \frac{\partial L}{\partial \bar{x}_i} - \sum_{j \in S_k} \frac{\partial L}{\partial \bar{x}_j} - \bar{x}_i \sum_{j \in S_k} \frac{\partial L}{\partial \bar{x}_j} \bar{x}_j), \end{aligned} \tag{6}$$

where L is the loss to optimize, and $\frac{\partial L}{\partial \bar{x}_i} = \frac{\partial L}{\partial y_i} \cdot \gamma_k$. It is not surprising that the gradient of γ or β is computed by all pixels in the related region. However, the gradient of a pixel with an index i is also computed by all pixels in the same region. More significantly in Eq. (6), the second term, $\sum_{j \in S_k} \frac{\partial L}{\partial \bar{x}_j}$, and the third term, $\bar{x}_i \sum_{j \in S_k} \frac{\partial L}{\partial \bar{x}_j} \bar{x}_j$, are shared by all pixels in the same region. Therefore, the pixel-wise connections within the same region are much denser after inserting the region norm layer.

Comparison with Other Normalizations. Compared with existing normalizations (*e.g.*, Batch Norm [31], Layer Norm [2], Instance Norm [69] and Group Norm [70]), which aims to speed up learning, there are two main difference: 1) the goal of Region Norm is to generate regionally homogeneous perturbations, while existing methods mainly aim to stabilize and speed up training; 2) the formulation of Region Norm is splitting an image to regions and normalize each region individually, while other methods do not split along spatial dimension.

3.3 Universal Analysis

By analyzing the magnitude of four probes (a, b, c, and d) in Fig. 3, we observe that $|b| >> |a|$ and $|c| >> |d|$ in a well-trained gradient transformer module (more results in Sect. 4.2). Consequently, such a well-trained module becomes quasi-input-independent, *i.e.*, the output is nearly fixed and less related to the input. Note that the output is still a little bit related to the input which is the reason why we use "quasi-".

Here, we first build the connection between that observation and under-fitting to explain the reason. Then, we convert the quasi-input-independent module to an input-independent module for generating universal adversarial perturbations.

Under-Fitting and the Quasi-Input-Independent Module. People figure out the trade-off between bias and variance of a model, *i.e.*, the price for achieving a small bias is a large variance, and vice versa [6,27]. Under-fitting occurs

when the model shows low variance (but inevitable bias). An extremely low variance function gives a nearly fixed output whatever the input, which we term as quasi-input-independent. Although in the most machine learning situation people do not expect this case, the quasi-input-independent function is desirable for generating universal adversarial perturbation.

Therefore, to encourage under-fitting, we go to the opposite direction of preventing under-fitting suggestions in [22]. On the one hand, to minimize the model capacity, our gradient transformer module only has $(12+2K)$ parameters, where a 3×3 convolutional layer (bias enabled) incurs 12 parameters and K is the number of region partitions. On the other hand, we use a large training data set $\mathcal{D}$ (5k images or more) so that the model capacity is relatively small. We then will have a quasi-input-independent module.

From Quasi-Input-Independent to Input-Independent. According to the analysis above, we already have a quasi-input-independent module. To generate a universal adversarial perturbation, following the post-process strategy of Poursaeed *et al.* [54], we use a fixed vector as input of the module. Then following FGSM [23], the final universal perturbation will be $\boldsymbol{u} = \epsilon \cdot \text{sign}(T(\boldsymbol{z}))$, where $\boldsymbol{z}$ is a fixed input. Recall that $\text{sign}(\cdot)$ denotes the sign function, and $T(\cdot)$ denotes the gradient transformer module.

4 Experiments

In this section, we demonstrate the effectiveness of the proposed regionally homogeneous perturbation (RHP) by attacking a series of defense models. The code is made publicly available.

4.1 Experimental Setup

Dataset and Evaluation Metric. Without loss of generality, we randomly select 5000 images from the ILSVRC 2012 [15] validation set to access the transferability of attack methods. For the evaluation metric, we use the improvement of top-1 error rate after attacking, *i.e.*, the difference between the error rate of adversarial images and that of clean images.

Table 1. The error rates (%) of defense methods on our dataset which contains 5000 randomly selected ILSVRC 2012 validation images

Defenses	TVM	HGD	R& P	Inc_{ens3}	Inc_{ens4}	IncRes_{ens}	PGD	ALP	FD
Error Rate	37.4	18.6	19.9	25.0	24.5	21.3	40.9	48.6	35.1

Attack Methods. For performance comparison, we reproduce five representative attack methods, including fast gradient sign method (FGSM) [23], momentum iterative fast gradient sign method (MIM) [16], momentum diverse inputs iterative fast gradient sign method (DIM) [16,77], universal adversarial perturbations (UAP) [50], and the universal version of generative adversarial perturbations (GAP) [54]. If not specified otherwise, we follow the default parameter setup in each method respectively.

To keep the perturbation quasi-imperceptible, we generate adversarial examples in the ϵ-ball of original images in the L_∞ space. The maximum perturbation ϵ is set as 16 or 32. The adversarial examples are generated by attacking a naturally trained network, Inception v3 (IncV3) [65], Inception v4 (IncV4) or Inception Resnet v2 (IncRes) [64]. We use IncV3 and $\epsilon = 16$ in default.

Defense Methods. As our method is to attack defense models, we reproduce nine defense methods for performance evaluation, including input transformation [26] through total variance minimization (TVM), high-level representation guided denoiser (HGD) [42], input transformation through random resizing and padding (R&P) [74], three ensemble adversarially trained models (Inc_{ens3}, Inc_{ens4} and IncRes_{ens}) [68], adversarial training with project gradient descent white-box attacker (PGD) [48,75], adversarial logits pairing (ALP) [35], and feature denoising adversarially trained ResNeXt-101 (FD) [75].

Among them, HGD [42] and R&P [74] are the *rank-1 submission* and *rank-2 submission* in the NeurIPS 2017 defense competition [39], respectively. FD [75] is the *rank-1 submission* in the Competition on Adversarial Attacks and Defenses 2018. The top-1 error rates of these methods on our dataset are shown in Table 1.

Implementation Details. To train the gradient transformer module, we randomly select another 5000 images from the validation set of ILSVRC 2012 [15] as the training set. Note that the training set and the testing set are disjoint.

For the region split function, we choose $r(h, w) = w$ as default, and will discuss different region split functions in Sect. 4.4. We train the gradient transformer module for 50 epochs. When testing, we use a zero array as the input of the gradient transformer module to get universal adversarial perturbations, *i.e.* the fixed input $\boldsymbol{z} = \mathbf{0}$.

4.2 Under-Fitting and Universal

To verify the connections between under-fitting and universal adversarial perturbations, we change the number of training images so that the models are supposed to be under-fitting (due to the model capacity becomes low compared to large dataset) or not. Specifically, we select 4, 5k or 45k images from the validation set of ILSVRC 2012 as the training set. We insert four probes a, b, c, and d in the gradient transformer module as shown in Fig. 3 and compare their values in Fig. 5 with respect to the training iterations.

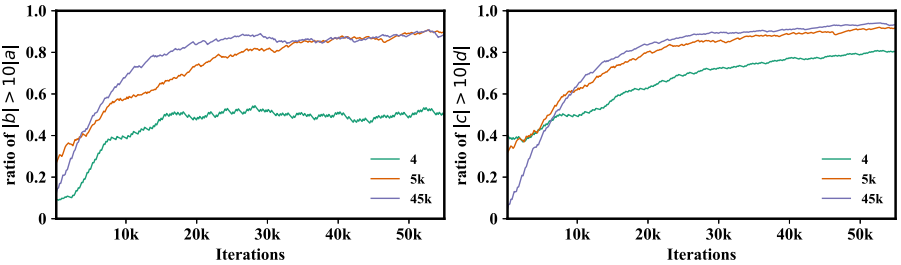

Fig. 5. Universal analysis of RHP. In (a), we plot the ratio of the number of variables in probe pairs (a, b) satisfying that $|b| > 10|a|$ to the total number of variables when training with 4, 5k or 45k images. In (b), we plot the case of $|c| > 10|d|$

When the gradient transformer module is well trained with 5k or 45k images, we observe that: 1) c overwhelms d, indicating the residual learning branch dominates the final output, *i.e.*, $\hat{g} \approx c$; and 2) b overwhelms a, indicating the output of the convolutional layer is less related to the input gradient g. Based on the two observations, we conclude that the gradient transformer module is quasi-input-independent when the module is under-fitted by a large number of training images in this case. Such a property is beneficial to generate universal adversarial perturbations (see Sect. 3.3).

When the number of training images is limited (say 4 images), we observe that b does not overwhelm a, indicating the output of the conv layer is related to the input gradient g, since a small training set cannot lead to under-fitting.

This conclusion is further supported by Fig. 6(a): when training with 4 images, the performance gap between universal inference (use a fixed zero as the input of the gradient transformer module) and image dependent inference (use the loss gradient as the input) is quite large. The gap is reduced when using more data for training.

To provide a better understanding of our implicit universal adversarial perturbation generating mechanism, we present an ablation study by comparing our method with other 3 strategies of generating universal adversarial perturbation with the same region split function. The compared includes 1) RP: Randomly assigns the Perturbation as $+\epsilon$ and $-\epsilon$ for each region; 2) OP: iteratively Optimizes the Perturbation to maximize classification loss on the naturally trained model (the idea of [50]); 3) TU: explicitly Trains a Universal adversarial perturbations. The only difference between TU and our proposed RHP is that random noises take the place of the loss gradient $\mathbf{g}$ in Fig. 2 (following [54]) and are fed to the gradient transformer module. RHP is our proposed implicitly method, and the gradient transformer module becomes quasi-input-independent without taking random noise as the training input.

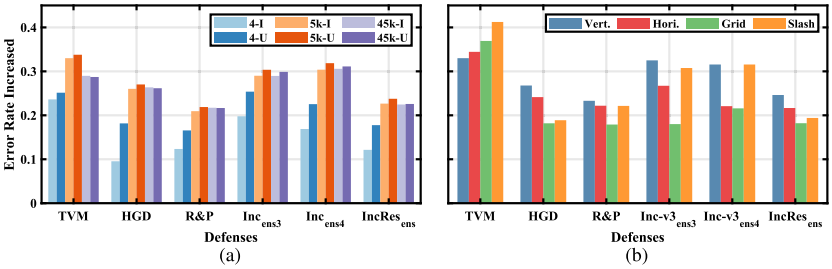

Fig. 6. (a) Performance comparison of universal (denoted by -U) inference and image dependent inference (denoted by -I) by varying the number of training images (4, 5k or 45k). (b) Performance comparison among four split functions, including vertical partition, horizontal partition, grid partition and slash partition

We evaluate above four settings on IncRes$_{ens}$, the error rates increase by 14.0%, 19.4%, 19.3%, and 24.6% for RP, OP, TU, and RHP respectively. Since our implicit method has a proper initialization (Sect. 3.2), we observe that our implicit method constructs stronger universal adversarial perturbations.

4.3 Transferability Toward Defenses

We first conduct the comparison in Table 2 when the maximum perturbation $\epsilon = 16$ and 32, respectively.

A first glance shows that compared with other representatives, the proposed RHP provides much stronger attack toward defenses. For example, when attacking HGD [42] with $\epsilon = 16$, RHP outperforms FGSM [23] by 24.0%, MIM [16] by 19.5%, DIM [16,77] by 14.9%, UAP [50] by 24.9%, and GAP [54] by 25.5%, respectively. Second, universal methods generally perform worse than image-dependent methods as the latter can access and utilize the information from the clean images. Nevertheless, RHP, as a universal method, still beats those image-dependent methods by a large margin. At last, we observe that our method gains more when the maximum perturbation ϵ becomes larger.

The performance comparison is also done when generating adversarial examples by attacking IncV4 or IncRes. Here we do not report the performance of GAP, because the official code does not support generating adversarial examples with IncV4 or IncRes. As shown in Table 3 and Table 4, RHP still keeps strong against defense models. Meanwhile, it should be mentioned that when the model for generating adversarial perturbations is changed, RHP still generates universal adversarial examples. The only difference is that the gradients used in the training phase are changed, which then leads to a different set of parameters in the gradient transformer module.

Table 2. The increase of error rates (%) after attacking. The adversarial examples are generated with IncV3. In each cell, we show the results when the maximum perturbation $\epsilon = 16/32$, respectively. The left 3 columns (FGSM, MIM and DIM) are image-dependent methods while the right 3 columns are (UAP, GAP and RHP) are universal methods

Methods	FGSM [23]	MIM [16]	DIM [16,77]	UAP [50]	GAP [54]	RHP (**ours**)
TVM	21.9/45.3	18.2/37.1	21.9/41.0	4.78/12.1	18.5/50.1	**33.0/56.9**
HGD	2.84/20.7	7.30/18.7	11.9/32.1	1.94/11.3	1.34/37.9	**26.8/57.5**
R& P	6.80/13.9	7.52/13.7	12.0/21.9	2.42/6.66	3.52/26.9	**23.3/56.1**
Inc_{ens3}	10.0/17.9	11.4/17.3	16.7/26.1	1.00/7.82	5.48/33.3	**32.5/60.8**
Inc_{ens4}	9.34/15.9	10.9/16.5	16.2/25.0	1.80/8.34	4.14/29.4	**31.6/58.7**
IncRes_{ens}	6.86/13.3	7.76/13.6	10.8/19.6	1.88/5.60	3.76/22.5	**24.6/57.0**
PGD	1.90/12.8	1.36/6.86	1.84/7.70	0.04/1.04	1.28/10.2	**2.40/25.8**
ALP	17.0/32.3	15.3/24.4	15.5/24.7	7.98/11.5	15.6/30.0	**17.8/39.4**
FD	1.62/13.3	1.00/7.48	1.34/8.22	-0.1/0.40	0.56/11.1	**2.38/24.5**

Table 3. The increase of error rates (%) after attacking. The adversarial examples are generated with IncV4. In each cell, we show the results when the maximum perturbation $\epsilon = 16/32$, respectively. The left 3 columns (FGSM, MIM and DIM) are image-dependent methods while the right 2 columns are (UAP and RHP) are universal methods

Methods	FGSM [23]	MIM [16]	DIM [16,77]	UAP [50]	RHP (**ours**)
TVM	22.4/46.3	20.1/40.4	22.7/42.9	6.28/18.2	**37.1/58.4**
HGD	4.00/21.1	10.0/23.9	16.3/37.1	1.42/9.94	**23.4/59.8**
R& P	8.68/15.1	10.2/17.4	14.7/25.0	2.42/6.52	**20.2/57.6**
Inc_{ens3}	10.1/18.3	13.4/20.3	18.7/28.6	2.08/7.68	**27.5/60.3**
Inc_{ens4}	9.72/17.4	13.1/19.0	17.9/26.5	1.94/6.92	**26.7/62.5**
IncRes_{ens}	7.58/14.7	9.96/16.6	13.6/22.1	2.34/6.78	**21.2/58.5**
PGD	2.02/12.8	1.50/7.54	1.82/8.02	0.28/2.12	**2.20/29.7**
ALP	17.3/32.1	14.8/25.1	15.2/24.8	10.1/15.9	**20.3/42.1**
FD	1.42/13.4	1.24/8.18	1.62/8.74	0.16/1.18	**1.90/31.8**

4.4 Region Split Functions

In this section, we discuss the choice of region split functions, *i.e.*, vertical partition, horizontal partition, grid partition and slash partition (parallel to an increasing line with the slope equal to 0.5). Figure 6(b) shows the transferability to the defenses, which demonstrates that different region split functions are almost equivalently effective and all are stronger than our strongest baseline (DIM). Moreover, we observe an interesting phenomenon as presented in Fig. 7.

Table 4. The increase of error rates (%) after attacking. The adversarial examples are generated with IncRes. In each cell, we show the results when the maximum perturbation $\epsilon = 16/32$, respectively. The left 3 columns (FGSM, MIM and DIM) are image-dependent methods while the right 2 columns are (UAP and RHP) are universal methods

Methods	FGSM [23]	MIM [16]	DIM [16,77]	UAP [50]	RHP (**ours**)
TVM	20.6/44.1	20.3/39.4	24.6/44.0	7.10/24.7	**37.1**/**57.4**
HGD	5.34/22.3	15.0/28.1	23.7/44.1	2.14/10.6	**26.9**/**62.1**
R& P	10.1/15.8	13.4/22.1	22.5/34.5	2.50/8.36	**25.1**/**61.4**
Inc_{ens3}	11.7/19.4	17.4/24.6	25.8/37.1	1.88/8.28	**29.7**/**62.3**
Inc_{ens4}	10.5/17.2	15.1/22.5	22.4/33.7	1.74/7.22	**29.8**/**63.3**
IncRes_{ens}	10.4/16.3	13.6/22.6	20.2/32.5	1.96/8.18	**26.8**/**62.8**
PGD	2.06/13.8	1.84/8.80	**2.36**/9.26	0.40/3.78	2.20/**28.3**
ALP	17.5/32.6	12.3/25.9	12.6/25.9	7.12/17.0	**22.8**/**43.5**
FD	1.72/14.7	1.62/9.48	1.78/10.1	-0.1/3.06	**2.20**/**32.2**

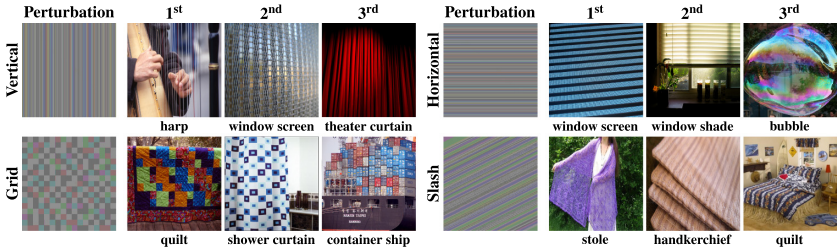

Fig. 7. Four universal adversarial perturbations generated by different region split functions, and the corresponding top-3 target categories

In each row of Fig. 7, we exhibit the universal adversarial perturbation generated by a certain kind of region split functions, followed by the top-3 categories to which the generated adversarial examples are most likely to be misclassified. For each category, we show a clean image as an exemplar. Note that our experiments are about the non-targeted attack, indicating the target class is undetermined and solely relies on the region split function.

As can be seen, the regionally homogeneous perturbations with different region split functions seem to be targeting at different categories, with an inherent connection between the low-level cues (*e.g.*, texture, shape) they share. For example, when using grid partition, the top-3 target categories are quilt, shower curtain, and container ship, respectively, and one can observe that images in the three categories generally have grid-structured patterns.

Motivated by these qualitative results, we have a preliminary hypothesis that the regionally homogeneous perturbations tend to attack the low-level part of a model. The claim is not supported by a theoretical proof, however, it inspires us

Table 5. Comparison of cross-task transferability. We attack segmentation model and test on the detection model Faster R-CNN, and report the value of mAP (lower is better for attacking methods). "–" denotes the baseline performance without attacks

Attacks	–	FGSM	MIM	DIM	RHP (**ours**)
mAP	69.2	43.1	41.6	36.2	**31.6**

to test the cross-task transferability of RHP. As it is a common strategy to share the low-level CNN architecture/information in multi-task learning systems [58], we conjecture that RHP can well transfer between different tasks (see below).

4.5 Cross-Task Transferability

To demonstrate the cross-task transferability of RHP, we attack with the semantic segmentation task and test on the object detection task.

In more detail, we attack a semantic segmentation model (an Xception-65 [13] based deeplab-v3+ [10]) on the Pascal VOC 2012 segmentation `val` [19], and obtain the adversarial examples. Then, we take a VGG16 [61] based Faster-RCNN model [56], trained on MS COCO [43] and VOC2007 `trainval`, as the testing model. To avoid testing images occurred in the training set of detection model, the testing set is the union of VOC2012 segmentation `val` and VOC2012 detection `trainval`, then we remove the images in VOC2007 dataset. The baseline performance of the clean images is mAP 69.2. Here mAP score is the average of the precisions at different recall values.

As shown in Table 5, RHP reports the lowest mAP with object detection, which demonstrates the stronger cross-task transferability than the baseline image-dependent perturbations, *i.e.*, FGSM [23], MIM [16], and DIM [16,77].

5 Conclusion

By white-box attacking naturally trained models and defense models, we observe the regional homogeneity of adversarial perturbations. Motivated by this observation, we propose a transforming paradigm and a gradient transformer module to generate the regionally homogeneous perturbation (RHP) specifically for attacking defenses. RHP possesses three merits, including 1) transferability: we demonstrate that RHP well transfers across different models (*i.e.*, transfer-based attack) and different tasks; 2) universal: taking advantage of the under-fitting of the gradient transformer module, RHP generates universal adversarial examples without explicitly enforcing the learning procedure towards it; 3) strong: RHP successfully attacks 9 representative defenses and outperforms the state-of-the-art attacking methods by a large margin.

Recent studies [42,75] show that the mechanism of some defense models can be interpreted as a "denoising" procedure. Since RHP is less like noise compared with other perturbations, it would be interesting to reveal the property of RHP

from a denoising perspective in future works. Meanwhile, although evaluated with the non-targeted attack, RHP is supposed to be strong targeted attack as well, which requires further exploration and validation.

Acknowledgements. We thank Yuyin Zhou and Zhishuai Zhang for their insightful comments and suggestions. This work was partially supported by the Johns Hopkins University Institute for Assured Autonomy with grant IAA 80052272.

References

1. Akhtar, N., Liu, J., Mian, A.: Defense against universal adversarial perturbations. In: CVPR (2018)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
3. Bai, S., Li, Y., Zhou, Y., Li, Q., Torr, P.H.: Metric attack and defense for person re-identification. arXiv preprint arXiv:1901.10650 (2019)
4. Baluja, S., Fischer, I.: Learning to attack: adversarial transformation networks. In: AAAI (2018)
5. Bhagoji, A.N., He, W., Li, B., Song, D.: Practical black-box attacks on deep neural networks using efficient query mechanisms. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11216, pp. 158–174. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01258-8_10
6. Bishop, C.M.: The bias-variance decomposition. In: Pattern Recognition and Machine Learning, pp. 147–152. Springer, Heidelberg (2006)
7. Borkar, T., Heide, F., Karam, L.: Defending against universal attacks through selective feature regeneration. In: CVPR (2020)
8. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: reliable attacks against black-box machine learning models. In: ICLR (2018)
9. Cao, Y., et al.: Adversarial sensor attack on lidar-based perception in autonomous driving. In: ACM SIGSAC CCS (2019)
10. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with Atrous separable convolution for semantic image segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 833–851. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_49
11. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (2017)
12. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
13. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: ICCV (2017)
14. Das, N., et al.: SHIELD: fast, practical defense and vaccination for deep learning using JPEG compression. In: KDD. ACM (2018)
15. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
16. Dong, Y., et al.: Boosting adversarial attacks with momentum. In: CVPR (2018)
17. Dong, Y., Pang, T., Su, H., Zhu, J.: Evading defenses to transferable adversarial examples by translation-invariant attacks. In: CVPR (2019)

18. Dziugaite, G.K., Ghahramani, Z., Roy, D.M.: A study of the effect of JPG compression on adversarial images. arXiv preprint arXiv:1608.00853 (2016)
19. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: a retrospective. IJCV **111**(1), 98–136 (2015)
20. Eykholt, K., et al.: Robust physical-world attacks on deep learning visual classification. In: CVPR (2018)
21. Gao, L., Zhang, Q., Song, J., Liu, X., Shen, H.T.: Patch-wise attack for fooling deep neural network. arXiv preprint arXiv:2007.06765 (2020)
22. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
23. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
24. Goyal, P., et al.: Accurate, large minibatch SGD: training ImageNet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
25. Guo, C., Frank, J.S., Weinberger, K.Q.: Low frequency adversarial perturbation. arXiv preprint arXiv:1809.08758 (2018)
26. Guo, C., Rana, M., Cissé, M., van der Maaten, L.: Countering adversarial images using input transformations. In: ICLR (2018)
27. Haykin, S.S.: Finite sample-size considerations. In: Neural Networks and Learning Machines, vol. 3, pp. 82–86. Pearson Upper Saddle River (2009)
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
29. Hendrik Metzen, J., Chaithanya Kumar, M., Brox, T., Fischer, V.: Universal adversarial perturbations against semantic image segmentation. In: ICCV (2017)
30. Huang, L., et al.: Universal physical camouflage attacks on object detectors. In: CVPR (2020)
31. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
32. Jia, R., Konstantakopoulos, I.C., Li, B., Spanos, C.: Poisoning attacks on data-driven utility learning in games. In: ACC (2018)
33. Jin, W., Li, Y., Xu, H., Wang, Y., Tang, J.: Adversarial attacks and defenses on graphs: a review and empirical study. arXiv preprint arXiv:2003.00653 (2020)
34. Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., Tang, J.: Graph structure learning for robust graph neural networks. arXiv preprint arXiv:2005.10203 (2020)
35. Kannan, H., Kurakin, A., Goodfellow, I.: Adversarial logit pairing. arXiv preprint arXiv:1803.06373 (2018)
36. Khrulkov, V., Oseledets, I.: Art of singular vectors and universal adversarial perturbations. In: CVPR (2018)
37. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
38. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: ICLR Workshop (2017)
39. Kurakin, A., et al.: Adversarial attacks and defences competition. arXiv preprint arXiv:1804.00097 (2018)
40. Li, Y., Bai, S., Zhou, Y., Xie, C., Zhang, Z., Yuille, A.: Learning transferable adversarial examples via ghost networks. In: AAAI (2020)

41. Li, Y., et al.: Volumetric medical image segmentation: a 3D deep coarse-to-fine framework and its adversarial examples. In: Lu, L., Wang, X., Carneiro, G., Yang, L. (eds.) Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics. ACVPR, pp. 69–91. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13969-8_4
42. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser. In: CVPR (2018)
43. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
44. Liu, L., et al.: Deep neural network ensembles against deception: ensemble diversity, accuracy and robustness. In: 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 274–282. IEEE (2019)
45. Liu, X., Cheng, M., Zhang, H., Hsieh, C.-J.: Towards robust neural networks via random self-ensemble. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 381–397. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_23
46. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: ICLR (2017)
47. Ma, X., et al.: Characterizing adversarial subspaces using local intrinsic dimensionality. In: ICLR (2018)
48. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
49. Mao, X., Chen, Y., Li, Y., He, Y., Xue, H.: GAP++: learning to generate target-conditioned adversarial examples. arXiv preprint arXiv:2006.05097 (2020)
50. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: CVPR (2017)
51. Mopuri, K.R., Garg, U., Babu, R.V.: Fast feature fool: a data independent approach to universal adversarial perturbations. In: BMVC (2017)
52. Naseer, M.M., Khan, S.H., Khan, M.H., Khan, F.S., Porikli, F.: Cross-domain transferability of adversarial perturbations. In: Advances in Neural Information Processing Systems, pp. 12905–12915 (2019)
53. Poursaeed, O., Jiang, T., Yang, H., Belongie, S., Lim, S.N.: Fine-grained synthesis of unrestricted adversarial examples. arXiv preprint arXiv:1911.09058 (2019)
54. Poursaeed, O., Katsman, I., Gao, B., Belongie, S.: Generative adversarial perturbations. In: CVPR (2017)
55. Qiu, H., Xiao, C., Yang, L., Yan, X., Lee, H., Li, B.: SemanticAdv: generating adversarial examples via attribute-conditional image editing. arXiv preprint arXiv:1906.07927 (2019)
56. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NeurIPS (2015)
57. Roth, H.R., et al.: DeepOrgan: multi-level deep convolutional networks for automated pancreas segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9349, pp. 556–564. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24553-9_68
58. Ruder, S.: An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098 (2017)
59. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D Nonlinear Phenomena **60**(1–4), 259–268 (1992)
60. Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L.S., Goldstein, T.: Universal adversarial training. In: AAAI (2020)

61. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
62. Sun, M., et al.: Data poisoning attack against unsupervised node embedding methods. arXiv preprint arXiv:1810.12881 (2018)
63. Sun, Y., Wang, S., Tang, X., Hsieh, T.Y., Honavar, V.: Adversarial attacks on graph neural networks via node injections: a hierarchical reinforcement learning approach. In: Proceedings of the Web Conference 2020, pp. 673–683 (2020)
64. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI (2017)
65. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
66. Szegedy, C., et al.: Intriguing properties of neural networks. In: ICLR (2014)
67. Tang, X., Li, Y., Sun, Y., Yao, H., Mitra, P., Wang, S.: Transferring robustness for graph neural network against poisoning attacks. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 600–608 (2020)
68. Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., McDaniel, P.: Ensemble adversarial training: attacks and defenses. In: ICLR (2018)
69. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: the missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
70. Wu, Y., He, K.: Group normalization. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_1
71. Xiao, C., Zhong, P., Zheng, C.: Enhancing adversarial defense by k-winners-take-all. In: ICLR (2020)
72. Xiao, C., Li, B., Zhu, J.Y., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. In: IJCAI (2018)
73. Xie, C., Tan, M., Gong, B., Yuille, A., Le, Q.V.: Smooth adversarial training. arXiv preprint arXiv:2006.14536 (2020)
74. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.: Mitigating adversarial effects through randomization. In: ICLR (2018)
75. Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: CVPR (2019)
76. Xie, C., Yuille, A.: Intriguing properties of adversarial training at scale. In: ICLR (2020)
77. Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L.: Improving transferability of adversarial examples with input diversity. In: CVPR (2019)
78. Yang, C., Kortylewski, A., Xie, C., Cao, Y., Yuille, A.: PatchAttack: a black-box texture-based attack with reinforcement learning. arXiv preprint arXiv:2004.05682 (2020)
79. Zhang, Z., Zhu, X., Li, Y., Chen, X., Guo, Y.: Adversarial attacks on monocular depth estimation. arXiv preprint arXiv:2003.10315 (2020)
80. Zhou, W., et al.: Transferable adversarial perturbations. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 471–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_28