WHERE TO BE ADVERSARIAL PERTURBATIONS ADDED? INVESTIGATING AND MANIPULATING PIXEL ROBUSTNESS USING INPUT GRADIENTS

Jisung Hwang[*†]
University of Chicago
jeshwang92@uchicago.edu

Younghoon Kim[*†]
KC Machine Learning Lab
kyhoon@kc-ml2.com

Sanghyuk Chun[†], **Jaejun Yoo, Ji-Hoon Kim & Dongyoon Han**
Clova AI Research, NAVER Corp.
{sanghyuk.c, jaejun.yoo, genesis.kim, dongyoon.han}@navercorp.com

ABSTRACT

This paper addresses the robustness of deep neural networks (DNNs) in respect of the network input. When imposing an input perturbation by an adversarial attack, it is hard to tell which pixels in the input are weak to the adversarial perturbation. We conjecture that the pixels with large expected input gradient are common weak points regardless of the input images. Based on our observation, we propose a simple module referred to Pixel Robustness Manipulator (PRM). By adding a PRM module as the first layer of a base network, the pre-determined (or planned) pixels become general weak points against adversarial attacks. This is done by inducing the adversarial perturbations to the predictable and interpretable locations by PRM, we can easily manage the location, where adversarial perturbations will affect on. Additionally, to show the effectiveness of PRM, we propose a simple defense strategy under a weak attack scenario, where the adversary knows the full parameters while has no information about the defense strategy.

1 INTRODUCTION

Adversarial attacks based on iterative optimization (Carlini & Wagner, 2016; Madry et al., 2017) have been emerged to evaluate the robustness of a neural network model. By adding a small perturbation to the input image iteratively (i.e., using the gradients of the input), it has been shown that generated adversarial samples can successfully fool the targeted trained neural network. A popular method of Projected Gradient Descent (PGD) (Madry et al., 2017) iteratively generates adversarial examples as

$$x^{t+1} = \texttt{clip}_{\epsilon}\left[x^t + \alpha \frac{\nabla_{x^t} L(\theta, x^t, y)}{\|\nabla_{x^t} L(\theta, x^t, y)\|_p}\right], \tag{1}$$

where $\texttt{clip}_{\epsilon}$ denotes a clip operation with small ϵ, $\|\cdot\|_p$ denotes the vector ℓ_p norm (e.g., ℓ_2 norm and ℓ_∞ norm). Using equation 1 for imperceptible pixel perturbations, the robustness of a model including the adversarial training (Szegedy et al., 2013; Huang et al., 2015; Goodfellow et al., 2014; Madry et al., 2017; Cisse et al., 2017; Wong & Kolter, 2018) has been widely studied. On the other hand, understanding the model robustness with respect to the input domain has been overlooked. There have been a few works (Papernot et al., 2016; Su et al., 2017; Cao & Gong, 2017; Prakash et al., 2018) that directly tackle the model robustness in the input domain. However, they are difficult to be generalized to different model structures, datasets, and attack methods. In addition, they often require heavy pre-processing computations.

[*]Work done at Clova AI Research
[†]Equal contribution

In this work, we investigate whether the underlying pixels are vulnerable or not, which is determined solely by the model of interest. However, because adversarial perturbations are generated dependent on a given image, it is difficult to find pixels that are generally weak. Here, we conjecture that pixels with large expected input gradient are general weak points. We first measure the robustness of pixels by using the expected input gradients. Next, we empirically back up that the proposed expected input gradient is a good proxy of measuring the robustness of pixels regardless of model structures and attack methods. Moreover, we show that by adding a simple module called Pixel Robustness Manipulator (PRM), to the network input stage, the robust pixel domains are easily moved to the designated pixels under a weak attack scenario.

2 Measuring Pixel Robustness using Expected Input Gradients

In this section, we propose a simple technique to estimate the pixel robustness by using the expected input gradients for each pixel effectively. A simplistic way of doing this is to take the perturbed images by using equation 1 directly. However, perturbed pixels are highly related to the input gradient (i.e., gradient of x^t, $\nabla_{x^t} L(\theta, x^t, y)$) which has a dependency on the t-th input x^t. Hence, it is hard to find general pixels which are weak or robust to adversarial perturbations. Moreover, because it clips the perturbation with small ϵ after normalizing the gradient by its ℓ_p norm, the intensity of the gradients often becomes irrelevant to the adversarial attack especially when $p = \infty$.

Instead of using the input gradient directly, we compute the expectation with respect to the absolute value of the input gradient term so that measuring pixel robustness becomes image-agnostic:

$$g(i,j) = \frac{1}{K|\mathcal{X}|} \sum_{x \in \mathcal{X}} \sum_{k=1}^{K} \left| \frac{\partial L(\theta, x, y)}{\partial x_{ijk}} \right|, \tag{2}$$

where x_{ijk} is an input pixel value where i, j and k are the indices of xy-coordinates and channel, respectively. K denotes the number of image channels (i.e., for RGB image, $K = 3$), L denotes a loss function, and θ denotes model parameters. Note that $g(i,j)$ only depends on the model structure θ but not on the data distribution. We conjecture that pixels with large $g(i,j)$ tend to be weak against adversarial perturbations while pixels with small $g(i,j)$ tend to be robust.

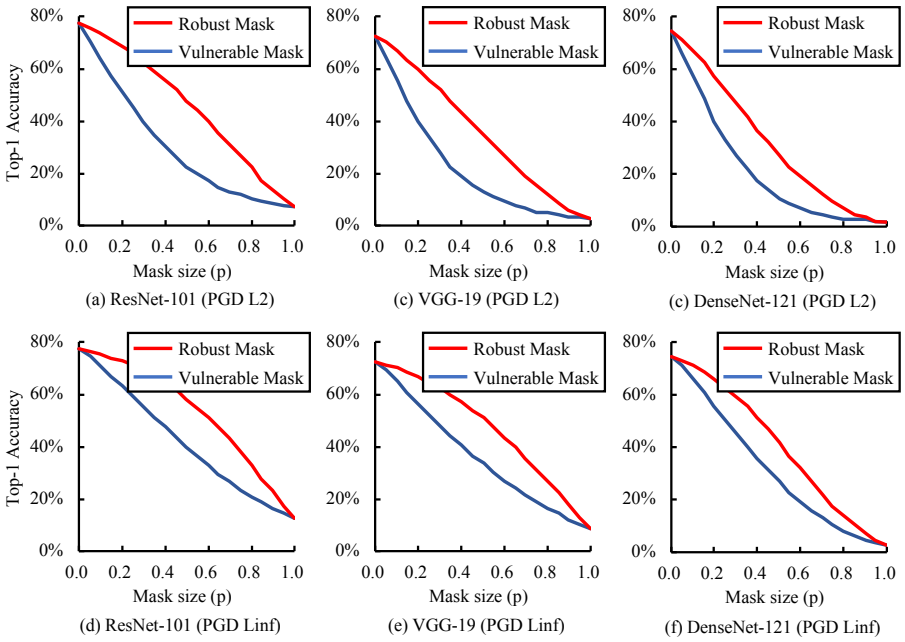

Figure 1: Top-1 accuracy (%) on ImageNet validation datasets after masked adversarial attacks using $M_t(p)$ (red line) and $M_b(p)$ (blue line) for ResNet-101, VGG-19 and DenseNet-121.

To show that the pixel robustness is related to the extent of $g(i, j)$, we generate a binary mask $M \in \{0, 1\}^{w \times h}$, where w and h denote width and height of an image x, respectively. Given M, we test the base network with the input $x = (1 - M) \odot x_o + M \odot x_a$, where x_o and x_a denote the original input and attacked input. We introduce two types of masks $M_t(p)$ and $M_b(p)$ by setting pixel values as 1 in the top p percentage of $g(i, j)$ and the bottom p percentage of $g(i, j)$, respectively. To generate the attacked image x_a, we use PGD using ℓ_2 and ℓ_∞ normalization with $1,000$ iterations with $\epsilon = 0.031$, which is the same setting used in Athalye et al. (2018). We report the top-1 accuracy on ImageNet validation dataset (Deng et al., 2009) after two types of masked attacks with varying p to ResNet-101 (He et al., 2016), VGG-19 (Simonyan & Zisserman, 2015) and DenseNet-121 (Huang et al., 2017) in Figure 1.

Figure 1 shows that the pixels with large $g(i, j)$ are vulnerable to adversarial perturbation while the pixels with small $g(i, j)$ are robust. The observation empirically supports that our proposed measure $g(i, j)$ is an appropriate proxy measure of pixel robustness that is independent of the input images, model structures, and the type of norm chosen by the PGD adversary.

3 Manipulating Pixel Robustness

3.1 Pixel Robustness Manipulator

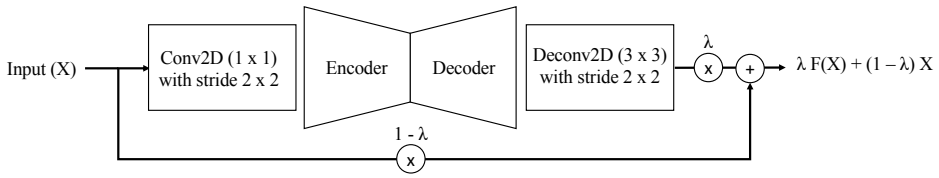

Figure 2: Overview of PRM manipulating the vulnerable pixels to align on checkerboard pattern. We use a standard U-Net (Ronneberger et al., 2015) structure with first layer with the kernel size of (1, 1) and the stride of (2, 2). The manipulated gradient patterns by the proposed module for different models are illustrated in figure 3.

In Section 2, we have shown that the pixel robustness can be estimated by equation 2. Interestingly, the derivative part in equation 2 is strongly correlated with sparsely connected layer in a network due to the chain rule in the back propagation. From the observation, we conjecture that the general weak points against adversarial attacks could be manipulated to the pre-defined locations regardless of network structures by manufacturing the connectivity between pixels and the network. To support this, we show that the pixels corresponding to large gradients (i.e., vulnerable pixels) become aligned to the target pixels by manufacturing the connectivity between pixels and the network.

Specifically, we propose a simple auxiliary module called Pixel Robustness Manipulator (PRM) that manipulates the locations of general weak points. By plugging a PRM module into the base network (i.e., just after the input), the pixel robustness can be manipulated. The details of our proposed PRM module are illustrated in Figure 2. We first train a convolutional auto-encoder with a skip connection where the first layer of the encoder is partially connected to the designated pixels. For example, if we set kernel size and the stride of the first convolution as $(1, 1)$ and $(2, 2)$, respectively, the first layer is only connected with the pixels of even coordinates. The weighting parameter $\lambda \in [0, 1]$ adjusts the sparse connection between input images and reconstructed images. By setting $\lambda = 0.0$, the output of PRM is exactly same as the original image while the output of PRM, when $\lambda = 1.0$ uses only sparse connections. For the experiments, we design a PRM module to have the kernel size and the stride of the first convolutional layer as $(1, 1)$ and $(2, 2)$ respectively. The PRM module is trained on ImageNet dataset with a reconstruction loss.

We report the top-1 validation accuracy on ImageNet dataset along with the intersection over union (IoU) between our designated pixels (i.e., checkerboard) and pixels of top 25% $g(i, j)$ by varying the weight of the skip connection λ in Table 1. Regardless of the base model, there is only a small decrease in the accuracy even with a large λ while the pattern of pixel robustness gradually coincide with the desired pixels. In addition, we report the original gradient heatmaps and pixels of top 25% $g(i, j)$ according to λ in figure 3. As reported in Table 1 and Figure 3, the robust and vulnerable pixels are almost aligned to the designated locations with small enough λ (e.g., 0.4).

Table 1: Top-1 accuracy (%) on ImageNet dataset and Intersection over Union (IoU) with the designated pixels (i.e., checkerboard) and 25% masks with varying λ's. Note that $\lambda = 0.0$ equals to the base network and $\lambda = 1.0$ equals to the base netowork with a PRM module without skip connection.

	λ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Acc (%)	ResNet	77.4	77.3	77.2	76.8	76.3	75.5	74.7	73.5	72.1	70.5	69.1
	VGG	72.4	72.3	72.1	71.8	71.3	70.4	69.1	67.5	65.8	63.7	61.8
	DenseNet	74.4	74.2	73.9	73.6	73.1	72.4	71.6	70.5	69.3	68	66.8
IoU (%)	ResNet	12.5	25.3	49.7	80.3	93.1	96.4	99.5	100	100	100	100
	VGG	14.3	25.1	52.6	91.8	94.8	96.5	99.0	100	100	100	100
	DenseNet	13.2	37.7	82.5	93.1	94.7	99.9	100	100	100	100	100

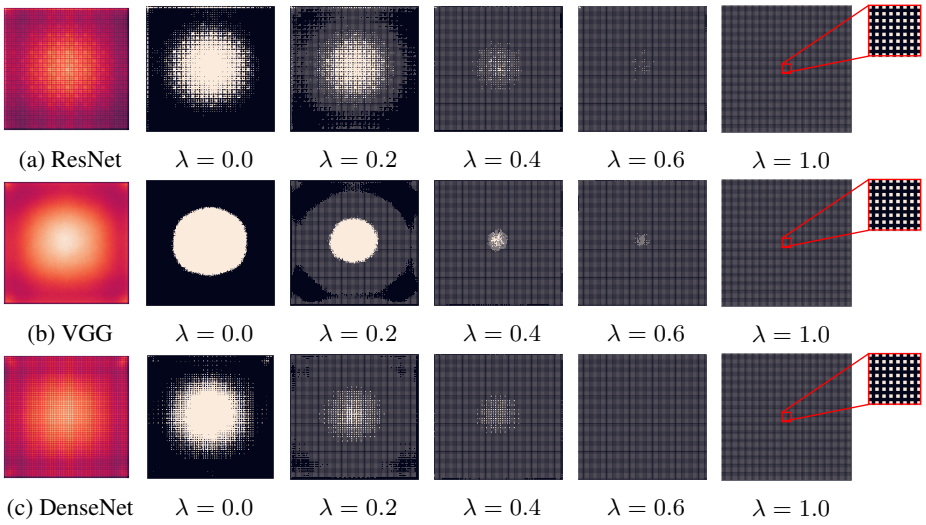

Figure 3: Input gradient and masks after plugging a PRM module in front of the network. Note that regardless of the base model, the PRM module almost aligns the robust and vulnerable pixels to the desired pixels even with a small enough λ (e.g., 0.4).

3.2 Evaluating Effectiveness of PRM in A Weak Attack Scenario

Here, we propose a simple defense method using PRM against adversarial attacks. We assume that the adversary knows the network parameter but has no information about the defense strategy. Similar scenarios were employed to a number of previous works (Cao & Gong, 2017; Xie et al., 2017; Prakash et al., 2018). Our scenario is neither white-box nor black-box because the adversary knows the full model parameter without knowing how the defense mechanism works.

Under the attack scenario, the adversary generates perturbations using the given network parameters. Interestingly, as a PRM module is plugged in front of the base network, the adversarial perturbations could be moved to the robust pixels by shifting the image just a single pixel in the evaluation stage. As illustrated in Figure 3, our module can manipulate the pattern of the pixel robustness to the desired pixels regardless of the base network.

We report the top-1 accuracy on CIFAR-10, CIFAR-100 and ImageNet dataset after performing various attack methods including One Pixel attack (Su et al., 2017), JSMA (Papernot et al., 2016), DeepFool (Moosavi-Dezfooli et al., 2016), C&W (Carlini & Wagner, 2016), and PGD (Madry et al., 2017) ℓ_2 attack with $1,000$ iterations to our proposed method and Region-based defense (Cao & Gong, 2017), Randomization defense (Xie et al., 2017), and Pixel Deflection defense (Prakash et al., 2018) in table 2. For the base network, we employ ResNet-18 for CIFAR experiments and ResNet-101 for ImageNet experiments. In ResNet-18 experiments, we finetune the model with a RPM module. All the experiments are done with NAVER Smart Machine Learening (NSML) GPU platform (Sung et al., 2017; Kim et al., 2018). The table shows that even in the weak attack scenario, the previous defense methods are easily broken while ours can defend the adversarial attacks successfully. Additionally, because our proposed module manipulates the pixel robustness in a

Table 2: Top-1 accuracy (%) of four defense methods against five attack methods on CIFAR-10, CIFAR-100 and ImageNet datasets. Boldface denotes the highest top-1 accuracy after defense.

Method	CIFAR-10					CIFAR-100					ImageNet
	OP	JSMA	DF	CW	PGD	OP	JSMA	DF	CW	PGD	PGD
Region.	65.1	6.3	78.0	7.6	0.0	50.0	17.3	**67.9**	6.5	0.1	0.0
Random.	76.9	72.6	84.4	74.7	1.5	51.3	21.7	56.9	39.1	0.3	11.1
Pixel Def.	60.7	48.7	71.9	66.7	10.6	41.1	22.0	54.0	48.4	7.2	20.9
Ours	**89.0**	**89.5**	**89.5**	**89.4**	**89.5**	**65.0**	**65.6**	65.7	**65.7**	**65.7**	**68.8**
Model (Acc.)	ResNet-18 (94.6)					ResNet-18 (73.6)					ResNet-101 (77.4)

network-agnostic way, our proposed module can be applied universally without any additional preprocessings or training techniques compared with the methods with additional computational costs of Region-based (Cao & Gong, 2017) and Pixel Deflection (Prakash et al., 2018). Our method evades adversarial perturbation with high interpretability while fully randomized resizing and padding by Randomization (Xie et al., 2017) is hard to interpret why it works.

Our future work is to make our proposed method be robust to adaptive attack scenarios. We will employ a randomization defense strategy by using various PRM modules, patterns generated by PRM modules (e.g., checkerboard, row-by-row, and column-by-column), and base networks.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 854–863. JMLR. org, 2017.

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

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

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

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

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

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. NSML: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

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

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 2016.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *CVPR*, 2018.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

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

Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. NSML: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*, 2017.

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

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5286–5295, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/wong18a.html`.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.