# Propagated Perturbation of Adversarial Attack for well-known CNNs: Empirical Study and its Explanation

Jihyeun Yoon *
LG CNS
jihyeun.yoon@lgcns.com

Kyungyul Kim *
LG CNS
kyungyul.kim@lgcns.com

Jongseong Jang †
LG Science Park
j.jang@lgsp.co.kr

## Abstract

*Deep Neural Network based classifiers are known to be vulnerable to perturbations of inputs constructed by an adversarial attack to force misclassification. Most studies have focused on how to make vulnerable noise by gradient based attack methods or to defense model from adversarial attack. The use of the denoiser model is one of a well-known solution to reduce the adversarial noise although classification performance had not significantly improved. In this study, we aim to analyze the propagation of adversarial attack as an explainable AI(XAI) point of view. Specifically, we examine the trend of adversarial perturbations through the CNN architectures. To analyze the propagated perturbation, we measured normalized Euclidean Distance and cosine distance in each CNN layer between the feature map of the perturbed image passed through denoiser and the non-perturbed original image. We used five well-known CNN based classifiers and three gradient-based adversarial attacks. From the experimental results, we observed that in most cases, Euclidean Distance explosively increases in the final fully connected layer while cosine distance fluctuated and disappeared at the last layer. This means that the use of denoiser can decrease the amount of noise. However, it failed to defense accuracy degradation.*

## 1. Introduction

In the computer vision field, deep neural networks(DNNs) achieve successful performance across various areas such as image classification, object detection, and semantic segmentation. But even though DNN is well trained, it can be easily degraded when noise is added to input data. Especially, DNN models trained by gradient-descent and back-propagation can be deteriorated by gradient-based noise attack, so called adversarial noise [2, 17, 25]. In such an adversarial attack to classifier case,

noise is located near discriminant hyperplane of DNN models, which makes easy to deceive the classifier. Thus, it is accomplished by making noise in a vulnerable area of DNN [28, 14]. Briefly, adversarial noise is a practical method because it could perturb target DNN without involvement in the learning process, and it often happens that it is difficult to visually confirm the presence of noise.

As a defense method for the type of adversarial attack, it is very natural to consider gradient masking, which hides the gradient of DNNs. But, gradient-based noise could be easily generated by substituting a model to a target classifier called a black-box attack [18, 19]. Therefore, most differentiable DNN could be easily exposed to gradient-based attack.

One of basic defense method against adversarial noise is to remove adversarial perturbation before the classifier. Among many denoising methods, denoising Auto-Encoder(DAE) [29] can be designed as a convolutional neural network (CNN) that can reduce the number of parameters and improve calculation efficiency while it maintains the performance [4]. For example, encoder-decoder structure and lateral skip-connection for residual learning based methods such as U-Net [22], FusionNet [20] and stacked U-Net [24] were introduced. Though the schemes are designed for image segmentation, it could be trained as a denoiser using pixel distance based objective function.

As shown the table 1, the experiment addressed that the performance of classifiers was not significantly improved in spite of reduced noise by the denoiser(FusionNet). It means that the characteristics of adversarial noise affect classification performance. For a good understanding of this phenomenon, adversarial noise has to be observed, how it would be propagated while it passes through DNN.

In this study, we examined of propagation behavior from input to output using well-known classifiers, three gradient-based attack noise - fast gradient sign method(FGSM) [5], iterative fast gradient sign method(i-FGSM) [14] and momentum iterative fast gradient sign method(mi-FGSM) [3]. We analyzed why this kind of noise is difficult to defence using denoiser and DNN classifier.

---

*Equal contribution, alphabetical order
†Corresponding author

ICCV 2019 Workshop on Interpreting and Explaining Visual Artificial Intelligence Models

| adversarial mode | Perturbation of Val. set(pix) | | Top-1 Val. Acc. (%) | |
|---|---|---|---|---|
| | $\text{MSE}(x_{ori}^{val}, x_{adv}^{val})$ | $\text{MSE}(x_{ori}^{val}, \text{DN}(x_{adv}^{val}))$ | $\text{F}(x_{adv}^{val})$ | $\text{F}(\text{DN}(x_{adv}^{val}))$ |
| FGSM V1 | 0.166 | 0.268 | 80.24 | 80.31 |
| FGSM V2 | 6.018 | 2.703 | 27.06 | 45.47 |
| i-FGSM V1 | 1.071 | 0.689 | 59.37 | 66.77 |
| i-FGSM V2 | 3.621 | 1.415 | 29.6 | 45.68 |
| mi-FGSM | 1.262 | 0.574 | 76.43 | 78.41 |

Table 1: Adversarial perturbation measured by mean square error (MSE) and corresponding accuracy on a validation set of TinyImageNet. Classifier and denoiser are Inception-Resnet V2(train accuracy 86.67%, validation accuracy 82.0%) and FusionNet respectively. $x_{ori}^{val}$ is the original validation images and $x_{adv}^{val}$ is the corresponding adversarial examples. DN($\cdot$) is the output passing through the denoiser and F($\cdot$) is the classifier output. It seems that there is no exact dependency between the amount of noise and validation accuracy.

## 1.1. Contribution

In this study, we made the following contributions.

- We observed propagation of black-box and white-box adversarial attack noise from input to output layer for DNN models, and tried to explain how to generate hyperplane. We Also showed that efficacy of standard denoiser based on DAE is limited for adversarial defense. According to the observation, we found that although perturbation is reduced, propagation is amplified to a similar level. We also analyzed the difference between original and adversarial samples and provided understanding about defense against adversarial attack.

- We experimented using various CNNs which have different capacity. From the experiment, we provided insight about propagation behavior respect to capacity and architecture.

## 2. Preparations

For observing the propagation of adversarial perturbation through a classifier, we calculated some distance between feature maps of original data and perturbed one layer by layer. Before running into it, three essential components, 1)generation of adversarial examples, 2)trained classifiers by training the dataset, and 3)trained denoisers by perturbed datasets, are required.

## 2.1. Generation of adversarial examples

On generation adversarial examples, we used three types of gradient-based attack and generated five adversarial datasets. datasets are generated by using TinyImageNet [1] and black&white-box attack [18, 19] in this paper.

- *Fast gradient sign method(FGSM)* [5]: To generate adversarial example, the attacker accumulates perturbation to the direction of input-output gradient to the original image, as follows:

$$x_{adv} \leftarrow x + \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \qquad (1)$$

In the equation $x$ is an original input image, $y$ is the ground truth label, $\epsilon$ is the step size of the perturbation, $sign()$ is the sign function, $J(\theta, x, y)$ is the loss function when $\theta$ is the trained attacker's parameters, and $\nabla_x$ is the gradient function for $x$. The attacker continues to accumulate it until it successes to mis-classify for the classifier or gets to $\epsilon$'s step limitation.

- *Iterative fast gradient sign method(i-FGSM)* [14]: Similar to FGSM, this attack iteratively computes the direction of the gradient and accumulates it to the image perturbed just before, as follows:

$$x_{k+1} \leftarrow x_k + \epsilon \cdot sign(\nabla_x J(\theta, x_k, y)) \qquad (2)$$

Attack would be stopped when it can lead misclassification for the classifier or gets to $\epsilon$'s step limit.

- *Momentum iterative fast gradient sign method(mi-FGSM)* [3]: In this method, a gradient is updated from the previous version with the momentum term, then it is accumulated to the image perturbed just before, as follows:

$$g_0 = 0, \quad x_0 = \text{original image}$$
$$g_{k+1} \leftarrow \mu \cdot g_k + \frac{\nabla_x J(\theta, x_k, y)}{||\nabla_x J(\theta, x_k, y)||_1} \qquad (3)$$
$$x_{k+1} \leftarrow x_k + \epsilon \cdot sign(g_{k+1})$$

In the equation, $x_k$ is k times perturbed noise and $\mu$ is balancing coefficient to adjust the change of the gradient by using a previous one. It also iterates this procedure until the classifier misclassifies the input or it gets to $\epsilon$'s step limit.
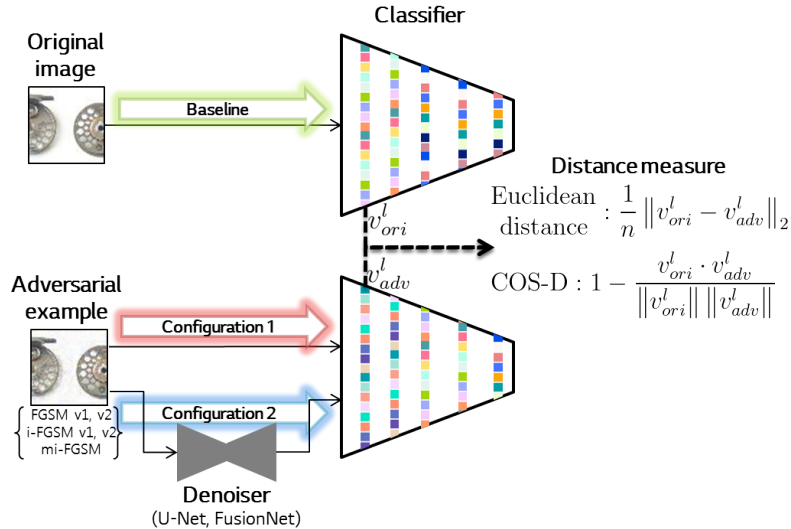
Figure 1: Concept of the experiment. We measure differences between feature maps of different inputs at the specified layer in the identical classifier. In the baseline setting, we feed original images to a classifier. Then, we also feed five types of adversaries and denoised adversaries to classifier for comparing the difference. Measuring difference is conducted by normalized Euclidean and cosine distance (NE-D & COS-D) between two feature maps, $v_{ori}^l$ and $v_{adv}^l$ which are the feature maps for the original and adversarial inputs at the $l$ th layer. $n$ is the number of elements in the feature map.

In general, the attacker is a substitute model of the classifier. In the black box attack, attack for input is iteratively conducted until a classifier returns a wrong label. During the attack, as an input-output gradient, adversarial perturbation is calculated in the attacker, which is another classifier trained on the same task. Based on these methods, we generated five adversarial examples sets by using Foolbox library [21] which is a python toolbox of large collection about the adversarial attack. The configurations for generating adversarial examples are shown in Table 2.

| | attacker (top-1 val. acc.) | classifier (top-1 val. acc.) |
|---|---|---|
| FGSM v1 | | ResNet-18 (60%) |
| FGSM v2 | | Inception-ResNet V2 (80%) |
| i-FGSM v1 | ResNet-18 (60%) [7] | ResNet-18 (60%) |
| i-FGSM v2 | | Inception-ResNet V2 (80%) |
| mi-FGSM | | Inception-ResNet V2 (80%) |

Table 2: Configurations for generating adversarial datasets.

## 2.2. Preparation of classifiers trained by training set

To observe the propagation of adversarial perturbation, 5 well-known CNNs with different capacity, i.e. VGG-19 [26], ResNet V2-50 [9], Inception-ResNet V2 [27], DenseNet-201 [11] and SENet-154 [10], were trained with TinyImageNet training set. For generalization, we trained them nearly perfectly on the training set. Before training, parameters of all classifiers were initialized by pre-trained model on ImageNet [23] and input images were resized to 128×128 (original size is 64×64) by bilinear interpolation. As an aside, we slightly modified their architecture (by adjustment stride size in the low level layers), as pre-trained models were optimized to the image size (299×299) of ImageNet. Training examples were sequentially transformed with random crop (range 0.85∼1.0) and horizontal flip (prob. 0.5) for each epoch, and normalized by a range of (-1.0, 1.0). Each of them was trained with Adam optimizer [12] with an initial learning rate of $1.0e^{-5}$ and L2 regularization coefficient of $1.0e^{-5}$. The prepared classifiers and their capacity are shown in Table 3.

## 2.3. Preparation of denoiser trained by adversarial example sets

As a denoising architecture, the authors selected U-Net [22] and FusionNet [20] as a denoiser. The reason is that U-Net methodology has proven to perform well in maintaining the robustness of models against adversarial attacks (in the NIPS2017 adversarial vision challenge [13]). So we have experimented with U-Net. Moreover, FusionNet which is an improved version of U-Net by skip-connections was used [20]. They were trained by a mean square error (MSE) objective between original images and adversarial examples. During training, each input batch consisted of all kinds of

| | top-1 training acc.(%) | Num. layers | Num. parameters |
|---|---|---|---|
| ResNet V2-50 | 99.905 | 50 | 25.56M |
| VGG-19 | 99.221 | 19 | 138.36M |
| SENet-154 | 99.334 | 154 | 113.45M |
| Inception-ResNet V2 | 99.598 | 234 | 55.97M |
| DenseNet-201 | 99.907 | 201 | 18.47M |

Table 3: well-known classifiers and their capacity. We used them to observe the propagation of adversarial perturbation. For generalization, all classifiers were nearly perfectly trained to the TinyImageNet training dataset.

adversaries with ratio of FGSM v1 : FGSM v2 : i-FGSM v1 : i-FGSM v2 : mi-FGSM = 0.05 : 0.30 : 0.05 : 0.30 : 0.30. FGSM and i-FGSM v1 are slightly perturbed dataset while FGSM and i-FGSM v2 are more heavily perturbed dataset. To compute MSE, we set the denoisers to generate the same output size ($64 \times 64$) to the adversarial input. The FusionNet was trained for 300 epochs with Adam optimizer with an initial learning rate of $1.0e^{-5}$ and L2 regularization coefficient of $1.0e^{-5}$. The U-Net was trained using fine-tuning with Adam optimizer with an initial learning rate of $1.0e^{-7}$ and L2 regularization coefficient of $1.0e^{-6}$. Primary and reduced errors between the training set and corresponding adversaries measured by MSE for each denoiser are shown in Table 4.

| | Primary $\mathrm{MSE}(x_{ori}^{tr}, x_{adv}^{tr})$ | FusionNet $\mathrm{MSE}(x_{ori}^{tr}, \mathrm{DN}_f(x_{adv}^{tr}))$ | U-Net $\mathrm{MSE}(x_{ori}^{tr}, \mathrm{DN}_u(x_{adv}^{tr}))$ |
|---|---|---|---|
| FGSM v1 | 0.051 | 0.230 | 0.168 |
| FGSM v2 | 13.439 | 7.395 | 8.537 |
| i-FGSM v1 | 1.224 | 0.829 | 0.738 |
| i-FGSM v2 | 4.220 | 1.632 | 1.538 |
| mi-FGSM | 1.466 | 0.612 | 0.572 |

Table 4: Primary and reduced errors between the training set and corresponding adversaries measured by MSE. $x_{ori}^{tr}$ and $x_{adv}^{tr}$ mean original images and adversarial examples for the training set. $\mathrm{DN}_f(\cdot)$ and $\mathrm{DN}_u(\cdot)$ are outputs from trained FusionNet and U-Net denoisers, respectively. Unit is pixels.

## 3. Experiments

We assume that the prepared classifiers are nearly generalized to the training set with their training accuracy. So, observing the propagation of perturbations for its corresponding adversarial sets is justified, since they can purely contribute to making the classifiers fool. In the remaining part of the paper, please note that we only use a training set and its corresponding adversarial sets.

As seen in Figure 1, to observe the propagation of adversarial perturbation, we should measure the difference between the feature maps extracted from the identical classifier, layer by layer. We can consider two feature maps; one is extracted by the original input (Baseline), and the other by the adversarial input (Configuration 1). The difference is measured by Euclidean and cosine distance (COS-D) between two feature vectors at the same layer, respectively. Euclidean value is normalized by the number of elements of the feature map, so we call it normalized Euclidean distance (NE-D). Additionally, as the classifiers show limited improvement even though passing through the denoising process (Table 5), we also need to observe the feature maps by denoised adversaries passed by denoiser (Configuration 2).

Practically, it is time-consuming to observe all of feature maps to all the datasets, due to tremendously large size of them and computational cost. Thus, we appointed some representative feature maps to observe for each CNN and Table 6 shows them. Because recently proposed DNNs have too much feature maps to observe all of them. So we only evaluated the last layers of each block as a representative layer. It is a common approach to analyze feature maps, such as [15]. Additionally, as extracting feature maps for all of the image set was burdensome work in the aspect of computational time and resources, we randomly sampled 1,000 images from the original training set and took adversarial examples corresponding to them. Consequently, at one observed position in one CNN, we measured the average distance of 1,000 feature map pairs of the originals (Baseline) and adversaries (Configuration 1) or denoised adversaries (Configuration 2).

In this experiment, PyTorch 0.4.1 and Python 3.5.2 were used in the Ubuntu 16.04 LTS.

## 4. Results and Discussion

Figure 2 and 3 show experimental results about averaged NE-D and COS-D between the feature maps of baseline and configuration settings for each classifier and adversarial set. In all classifier with/without denoiser, NE-D explosively increases in the fully connected (FC) layer while it tends to keep in small in the convolutional layers (i.e. all layers except for the last one). Generally, adversarial set which has relatively larger perturbation (FGSM v2 and i-FGSM v2) shows a bigger gap than others. Especially, VGG-19 shows an incredibly large jump in the gap after the last FC layer. On the other hand, Figure 3 shows dramatic directional identification at the end of each network by the FC layers, while the directional gap is gradually increased (ResNet V2-50, VGG-19, and SENet-154) or shows up-down-up (Inception-ResNet V2 and DenseNet-201) in the

| | ResNet V2-50 | | | VGG-19 | | | SENet-154 | | | Inception-ResNet V2 | | | DenseNet-201 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | w/o DN | w/ FN | w/ UN | w/o DN | w/ FN | w/ UN | w/o DN | w/ FN | w/ UN | w/o DN | w/ FN | w/ UN | w/o DN | w/ FN | w/ UN |
| FGSM v1 | 98.9 | 98.6 | 99.3 | 95.7 | 96.3 | 96.0 | 97.3 | 96.3 | 96.8 | 91.2 | 97.3 | 98.4 | 98.1 | 98.4 | 97.1 |
| FGSM v2 | 33.8 | 64.5 | 69.2 | 27.7 | 48.9 | 52.0 | 35.1 | 52.5 | 56.6 | 34.2 | 57.0 | 61.3 | 32.0 | 60.7 | 60.2 |
| i-FGSM v1 | 77.7 | 87.3 | 88.2 | 52.2 | 64.7 | 67.9 | 68.7 | 74.6 | 76.8 | 71.5 | 78.4 | 79.8 | 69.2 | 82.3 | 82.5 |
| i-FGSM v2 | 41.1 | 69.2 | 71.1 | 32.2 | 48.2 | 48.8 | 39.4 | 54.7 | 55.6 | 39.1 | 59.2 | 60.8 | 38.9 | 62.2 | 63.7 |
| mi-FGSM | 94.1 | 96.9 | 97.0 | 93.1 | 96.3 | 96.1 | 94.6 | 95.8 | 96.3 | 94.6 | 96.8 | 96.8 | 94.3 | 97.4 | 96.8 |

Table 5: Classification accuracy on the adversarial sets of each classifier with/without the denoiser. The classifiers are generalized to the training set. All classifiers show somewhat reasonable performance to the adversarial sets with small perturbation (FGSM v1, i-FGSM v1, and mi-FGSM). However, they show relatively lower performance to the bigger adversarial sets (FGSM v2, i-FGSM v2) even though inputs pass through the denoising process. FN: FusionNet, UN: U-Net
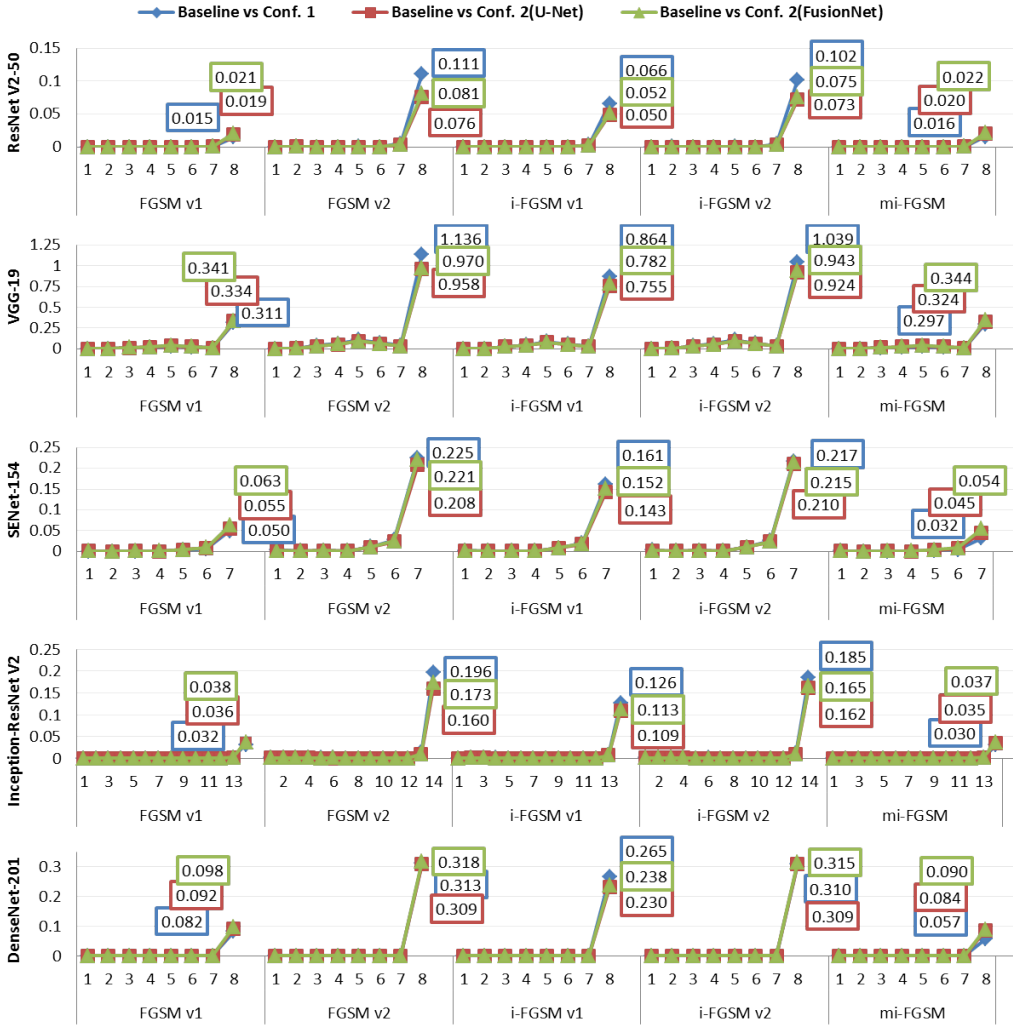


Figure 2: Average normalized Euclidean distance (NE-D) between the feature maps by baseline and configuration settings according to the adversaries and the classifiers. Each figure means the distance at the last observed feature map (i.e. the last layer).
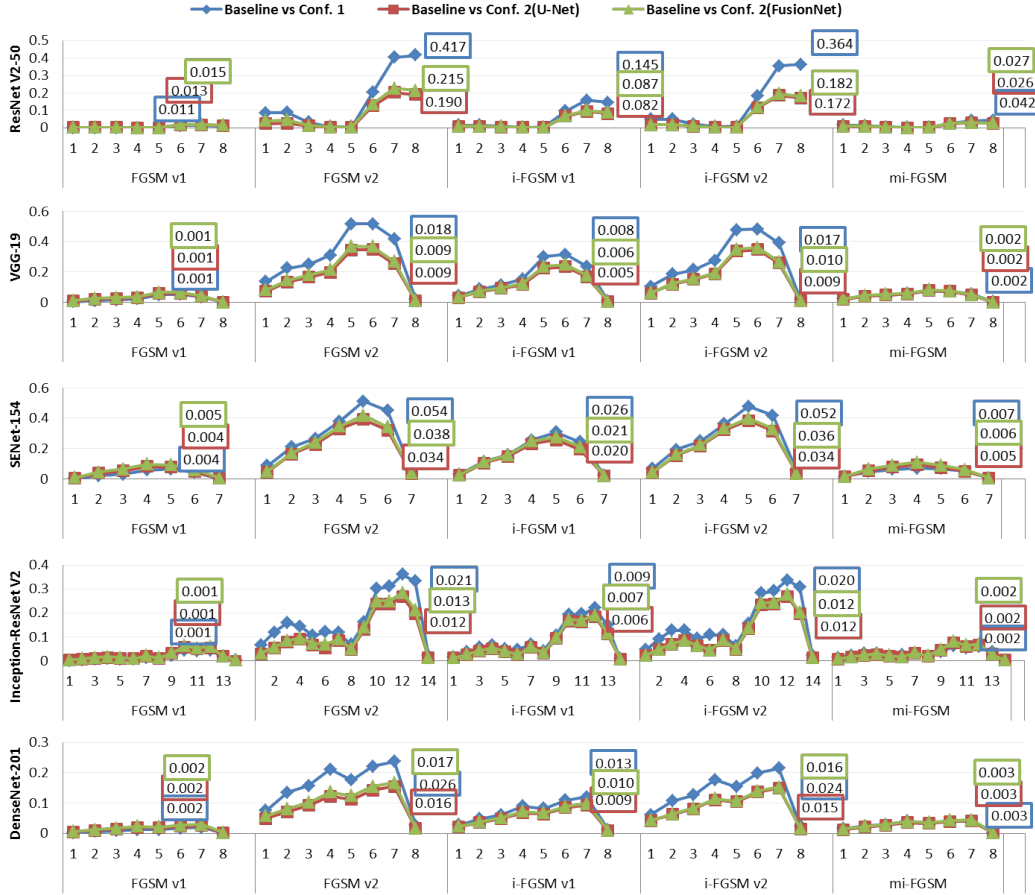
Figure 3: Average COS-D between the feature maps output by baseline and configuration settings according to the adversaries and the classifiers. Each figure is the distance at the last observed feature map (i.e. the last layer).

convolutional layers. Figure 3's result is consistent with [16], which confirmed that a feature vector of the middle layer shows the behavior of outlier.

In the aspect of the effect of denoisers, MSE was not perfectly eliminated by the denoisers (see Table 4), so that they gave not enough improvement in validation accuracy in spite of reduced adversarial noise (see Table 5). As seen the graph in Figure 2 and 3, NE-D is a little, but not enough, decreased compared to the case without the denoisers (Of course, in ResNet-V2 50, noise is reduced $20 \sim 30\,\%$ in the case of FGSM v2 and i-FGSM v2). COS-D also tends to be reduced during the convolutional layers, but it becomes identical eventually. Based on that observation, denoisers are somewhat effective in reducing COS-D in the middle layers. We should note that NE-D is much a little reduced at the last layers in SENet-154, Inception-ResNet V2 and DenseNet-201. This phenomenon seems that it relates to weaker improvement than the case of ResNet V2-50.

Overall, observations for the experiment follow below,

1. In convolutional layers, adversarial perturbation is not amplified with averaged NE-D, but with COS-D. However, in FC layers, it shows the opposite pattern. As an exceptional case, in ResNet V2-50, COS-D is not decreased when passing through the FC layer while it shows relatively lower NE-D than others.

2. As seen in Table 5, accuracy is most improved when a denoiser is combined with ResNet V2-50. Seemingly, ResNet V2-50 has successful noise suppression capacity than other classifiers when seen NE-D. However, it poorly controls the direction of noise in the aspect of COS-D.

3. Effect of denoiser is different depends on classifiers, but it is limited from the aspect of a logit vector. NE-D shows that the amount of change of ResNet V2-50 is larger than other classifiers, such as SENet-154, Inception-ResNet V2 and DenseNet-201. Denoiser also reduce COS-D in middle layers for all classifiers.
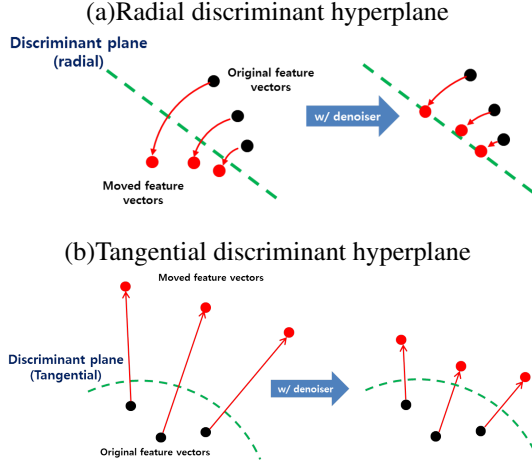
(a)Radial discriminant hyperplane

(b)Tangential discriminant hyperplane

Figure 4: Two types of discriminant hyperplane. Based on the observations, ResNet V2-50's plane is anticipated as radial and the others' as tangental.

| | Resnet V2-50 | VGG-19 | SENet-154 | Inception-ResNet V2 | DenseNet-201 |
|---|---|---|---|---|---|
| 1 | 1 convs | 2 convs | 3 convs | 1 conv | 1 Conv, 1 max pool, 6 FCs |
| 2 | 1 max pool | 2 convs | 3 SE block | 1 conv | 1 transition |
| 3 | 3 bottlenecks | 4 convs | 8 SE block | 1 conv | 12 FCs |
| 4 | 4 bottlenecks | 4 convs | 36 SE block | 1 conv | 1 transition |
| 5 | 6 bottlenecks | 4 convs | 3 SE block | 1 conv | 48 FCs |
| 6 | 3 bottlenecks | 1 FC | 1 avg pool | 1 maxpool, 1 mixed_b | 1 transition |
| 7 | 1 avg pool | 1 FC | 1 FC | 10 Block35s | 32 FCs |
| 8 | 1 FC layer | 1 FC | | mixed_a | 1 max pool, 1 FC |
| 9 | | | | 20 Block17s | |
| 10 | | | | mixed_a | |
| 11 | | | | 9 Block8s | |
| 12 | | | | 1 Block8 | |
| 13 | | | | 1 conv, 1 avg pool | |
| 14 | | | | 1 FC | |

Table 6: Selected feature map lists of each classifier. Feature maps are extracted right after each procedure, which is written in the table.

When seen the results of the last FC layer, adversaries result in angular (ResNet V2-50) or longitudinal perturbations (the others). This means that the different discriminant hyperplane might be constructed according to the classifiers. Since the adversarial examples are generated to make the classifiers fool, it is reasonable for us to infer where the discriminant hyperplane is, based on moved feature vectors. When inferring from relative small NE-D and large COS-D, a radial discriminant plane might be reasonable in the case of ResNet V2-50, as seen in Figure 4(a). On the other hand, in the case of the other classifiers, a tangential one is strongly suspected, as seen in Figure 4(b). With this inference, it is convincible that effectiveness of

| | DenseNet-201 | | | | | |
|---|---|---|---|---|---|---|
| | FC->Conv. | | | adding PRelu after FC | | |
| | w/o DN | w/ FN | w/ UN | w/o DN | w/ FN | w/ UN |
| FGSM V1 | 97.4 | 96.3 | 96.9 | 97.8 | 97.6 | 97.8 |
| FGSM V2 | 31.6 | 56.7 | 56.6 | 30.7 | 59.9 | 59.6 |
| i-FGSM V1 | 63.7 | 74.6 | 73.8 | 66.3 | 78.2 | 78.7 |
| i-FGSM V2 | 37.1 | 58.2 | 56.4 | 38.3 | 61.6 | 59.9 |
| mi-FGSM | 93.9 | 96.0 | 96.8 | 94 | 95.5 | 96.7 |

Table 7: Accuracy of two types of modified DenseNet-201s.

the denoisers is maximized in ResNet V2-50, since reduced COS-D (approx. 50% or more) and small NE-D are very effective to prevent misclassification. On the contrary, in the other cases which have a tangential plane, in spite of tiny COS-D, an improvement on performance cannot easily be achieved unless the denoiser reduces NE-D much a lot. Thus, ResNet V2-50 architecture is more efficient than other architectures to reduce NE-D in the experiment. However, because the adversarial perturbed image set is generated by ResNet only, additional experiments using various attacker have to be conducted for consistency.

Due to surprisingly amplified NE-D in the last FC layer, it is natural to consider replacement of it to convolutional one. Because DNN is generally trained to be overconfident [6], it is guessable that final FC layer is trained to make it's distance large. So, we additionally conducted the same experiment with modified DenseNet-201(i.e. the last layer is modified from avg. pooling(kernel size=4, stride=1, no padding) + FC to Conv. layer (kernel size=4, stride=1, no padding), so the network has only Conv. layers). But, there is no big difference (but, tiny improvement) when comparing to the previous result (See Figure 5). It is somewhat guessable because it just changes a linear operation from on $1\times1$ map to on $4\times4$ that is similar setting in that it refers to whole map, not local. Consequently, the FC layer itself, at least, is not a direct factor of amplification of NE-D.

In fact, except for the last FC layer, all feature maps were observed after Relu activation which reduces NE-D via rectifying the output. At that point, we wondered whether getting lower NE-D is possible or not if Relu is added after the FC. With that setting, we experimented again with DenseNet-201. However, generalization could not be achieved with the training set. So, we changed Relu to PRelu($\alpha = 0.25$) [8] instead.

As a result, NE-D of the model is decreased while COS-D is increased on the opposite side (Figure 6). The plot of NE-D and COS-D for the model follows a similar pattern to that of ResNet V2-50, but it little affects the performance (Table 7). Consequently, a classifier which is trained to reduce NE-D causes large COS-D logit, and training only using NE-D could not affect the performance. Thus, the performance of denoiser for the model is less than that of ResNet V2-50.
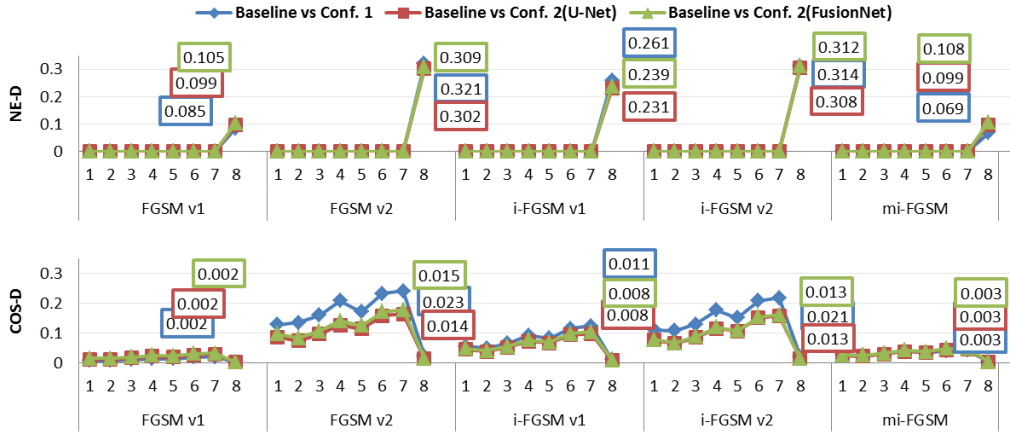
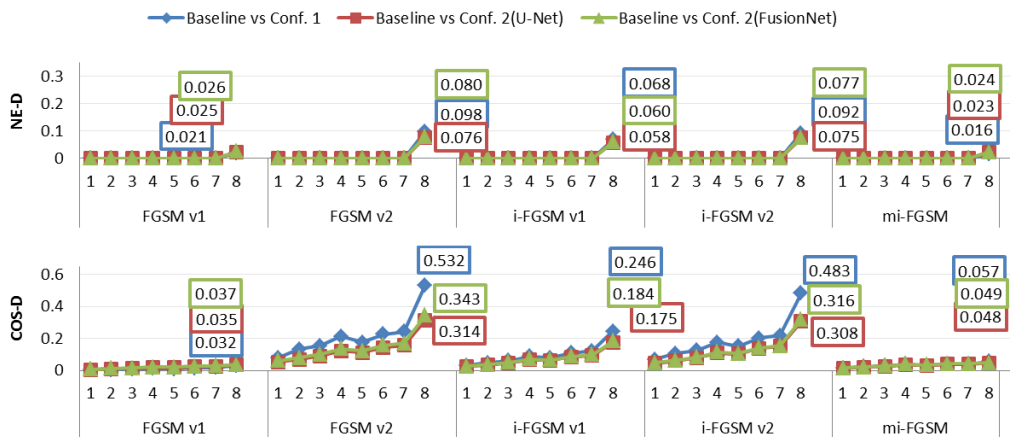Figure 5: Average NE-D and COS-D in the case of DenseNet-201 which replaced FC to Conv layer.



Figure 6: Average NE-D and COS-D in the case of DenseNet-201 adding PRelu after the FC layer.

# 5. Conclusion

To defense adversarial attack, the use of denoiser is the most widely used solution. It reduces the amount of noise, but the improvement of classification accuracy is marginal. In this paper, we aimed to examine the propagation of adversarial perturbation by measuring Euclidean distance and cosine distance in each CNN layer between each feature map of the original image and perturbed image passed through denoiser. We observed that Euclidean distance explosively increases in final FC layer while cosine distance fluctuated and disappeared at the last layer in most cases except the ResNet V2-50 classifier. In the case of ResNet V2-50, COS-D explosively increased in final FC layer while NE-D disappeared at the last layer. The accuracy improvement of ResNet V2-50 is more than that of other networks. This means that the two types of distance could be utilized to examine how noise is propagated through the network. It would be interesting future work to analysis why the ResNet V2-50 is robust for an adversarial attack.

# References

[1] Tiny ImageNet visual recognition challenge. https://tiny-imagenet.herokuapp.com/.

[2] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[3] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018.

[4] L. Gondara. Medical image denoising using convolutional denoising autoencoders. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*, pages 241–246. IEEE, 2016.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[10] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[11] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] A. Kurakin. Adversarial attacks and defences competition. *CoRR*, 2018.

[14] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[15] K. Lee. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS 31*, pages 7167–7177. 2018.

[16] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[17] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[18] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[19] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[20] T. M. Quan, D. G. Hildebrand, and W.-K. Jeong. Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics. *arXiv preprint arXiv:1612.05360*, 2016.

[21] J. Rauber, W. Brendel, and M. Bethge. Foolbox v0.8.0: A python toolbox to benchmark the robustness of machine learning models. *CoRR*, abs/1707.04131, 2017.

[22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[24] A. Sevastopolsky, S. Drapak, K. Kiselev, B. M. Snyder, and A. Georgievskaya. Stack-u-net: Refinement network for image segmentation on the example of optic disc and cup. *arXiv preprint arXiv:1804.11294*, 2018.

[25] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[27] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[29] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.