An Exploration of Universal Adversarial Perturbation in Deep Learning

Yuan Gao Lingfeng Zhang Hongfan Mu Wei Li Peng Yu

Abstract

Given a state-of-art deep neural network classifier, we realize the existence of a universal(image-agnostic) and very small perturbation vector that causes natural images to be misclassified with high probability. Under this circumstance, the conditions of generating universal adversarial perturbation have been investigated:

- The existence of universal adversarial perturbation under different data complexity
- The existence of universal adversarial perturbation under different classifier model complexity

The goal of project is to find whether there exit relations among data complexity, classifier model complexity, and universal adversarial perturbation.

1. Introduction

With the rapid development of deep learning, it has shown its excellent performance in image classification, object recognition, image processing, natural language processing, etc. Deep learning has helped us solve many complex problems, such as the Go problem. The complexity of Go's game tree reached 10^{300}, and no one thought that the machine could completely crack it before Alphago came out. Alphago defeated Lee Sedol, Jie Ke and other masters one after another, showing its powerful strength.

Compared with traditional machine learning algorithms, deep learning has advantages in many aspects. First of all, deep learning does not require a lot of feature processing on the input data. Processing input data features is a very difficult task and requires extensive expert knowledge. This means that models designed using machine learning algorithms tend to focus on a specific field and have poor transferability. It is impossible to use the model used to analyze problems in the financial field to deal with problems in other fields. However, for deep learning, we do not need to perform feature processing on the input data. We can directly input the data into the deep learning model, and the model will learn the features of the data on its own. This allow the model to have some transferability. Secondly, deep learning is easier to improve the accuracy of the model compared to traditional machine learning algorithms. Sometimes increasing the amount of input data can improve the accuracy of deep learning models.However this is unrealistic for machine learning algorithms.

Because of these characteristics, deep learning is widely used in our daily lives. Face recognition unlocked first by Apple has become standard on today's phones. Voice assistants like Siri, Alexa, and Google Assistant have also gradually become an integral part of our lives. Unmanned driving will also become mainstream in the near future. Tesla's Autopilot has been made available on highways and will soon be available on urban roads.

When deep learning is widely used in life, its safety issues have gradually been paid attention to by people. No system can be said to be 100% safe, and deep learning models are no exception. (Szegedy et al., 2014) proposed an attack on deep learning-based image classification models in 2014. They generated small perturbations on the images and input changed images into state-of-art deep neural networks. The results showed that most of these pictures were misclassified with high probability. This means that small modifications can fool the classifier. This attack on the model is named as *Adversary Attack*

As people continue to study the Adversary attack, (Moosavi-Dezfooli et al., 2017) found that a general perturbation can be found in an image-net dataset. Any image in the data set will be misclassified by the classifier after adding this perturbation. They also found that the perturbation produced by their method also has a certain degree of transforbility, it can fool different classifier. This perturbation were named as *Universal adversarial Perturbation*.

Deep learning models are considered as a black box, we know the input data, output data and network architecture. And we also know it will bring good performance. However, we are difficult to know why its performance is so good. Taking image classification as an example, we cannot intuitively understand the classification criteria of deep learning models. This is also the reason why it is difficult to find the conditions for generating universal adversarial perturbation.

Adversarial perturbation makes errors in deep learning mod-

els that perform well. This will cause serious security threats to applications based on deep learning that have been applied to life. In 2017, (Kurakin et al., 2017) proposed that adversarial perturbation can be deployed in the physical world. As a simple example, if someone adds an adversarial perturbation to a stop sign, autonomous vehicles will have a high probability of misidentifying the stop sign and cause a traffic accident. Universal adversarial perturbation makes it easier to apply attacks to different scenarios without the need to customize perturbation examples for each scenario. So we want to do research on universal adversarial perturbation.

In this report, we investigate and study the conditions for generating universal adversarial perturbation, and look for the existence of universal adversarial perturbation under different data complexity, classifier model complexity and other relative parameters.And explore the relationship between these parameters and universal adversarial perturbation.

The remaining of the report is organized as fellow. Section 2 introduces the related works of the adversarial perturbation. Section 3 introduces the methods for generating adversarial perturbation. We use the method introduced in Section 3 to generate universal adversarial perturbation under different parameters in Section 4 and analyze results in Section 5. In Section 6 we discuss and analyze the problems found in the experiment and make simple plans for future work. Finally in section VII we try to draw simple conclusions based on our experimental results.

2. Related work

2.1. Notation

x, original images that do not be modified from the dataset. x', the adversarial images which are modified from the dataset. What should be noticed is that the adversarial images will not be misclassified sometimes. l, the label of class in the classification problem. l', the label of class in the adversarial class. η, the size of the adversarial perturbation. Goodfellow et al. (Goodfellow et al., 2014) proposed that, in most cases, the L_∞ of the perturbation should be less than η. η is also specified as pixel values in the range [0,255]. Based on the research from Szegedy et al. (Szegedy et al., 2014), 2014, there are also some work focus on minimizing the size of the perturbation directly instead of importing a constrain on the size of the perturbation. $J_f(x, l)$ represents the cost function of f.

2.2. Generating adversarial examples

In this subsection, multiple methods that are used to generate adversarial examples will be discussed.

1. L-BFGS Attack
   The adversarial examples attacks against deep neural networks is first introduced by Szegedy et al.(Szegedy et al., 2014) in 2014. L-BFGS, which aims to find an appropriate constant c, is used to solve the general targeted misclassified problem:

$$\min_{x'} \ c\|\eta\| + J_\theta(x', l')$$

$$s.t. \ \ x' \in [0, 1],$$

   where l denotes the output label of x.

2. Fast Gradient Sign Method(FGSM)
   L-BFGS is a typical linear-searching method requires large time consuming. In 2014, Goodfellow et al.(Goodfellow et al., 2014) proposed a more computationally efficient method named Fast Gradient Sign Method, updating gradient at each pixel within one step. The perturbation is:

$$\eta = \epsilon sign(\bigtriangledown_x J_\theta(x, l)),$$

   which can be generated by computing back-propagation. FGSM is a much more simple way to generate adversarial examples:

$$x' = x + \eta,$$

   where ϵ can be selected ranged [0,1] that represents the magnitude of the perturbation. Since the deep neural network in high dimension shows that it can not defence adversarial examples, Fast Gradient Value method, which was proposed by (Rozsa et al., 2016). The sign of the gradient was replaced by the raw gradient in this method that is different from previous work:

$$\eta = \bigtriangledown_x J(\theta, x, l).$$

3. Jacobian-based Saliency Map Attack (JSMA)
   Jacobian-based Saliency Map Attack was first proposed by Papernot et al. (Papernot et al., 2015)which computes Jacobian matrix of input x :

$$J_{F(x)} = \frac{\partial F(x)}{\partial x} = [\frac{\partial F_j(x)}{\partial x_i}]_{i \times j}$$

   F represents the second-to-last layer, while Carlini and Wagner (Carlini & Wagner, 2016) tends to replace it with the output of the softmax layer, from which, they observe that the most essential modification to the output was contributed by the input features of x. What was found as well is that a small perturbation on features can lead to the misclassification of neural network with inducing the large output changes.
   Because of the observation founded, two adversarial salience maps were created to select the features in

each iteration, which contributes to improving the success of misclassifying adversarial examples by modifying a slight changes on input images.

4. CW's Attack
Carlini and Wagner defined a new objective function g that is different from previous work (Carlini & Wagner, 2017a)(Carlini & Wagner, 2017b):

$$\min_{\eta} \ \|\eta\|_p + c \cdot g(\eta + x)$$

$$s.t. \ \ x + \eta \in [0,1]^n,$$

where $g(x) \geq 0$ if and only if $f(x) = l$. From their research(Carlini & Wagner, 2017a)(Carlini & Wagner, 2017b), C&W's Attack is effective for most of existing adversarial detecting defenses. In this method, seven different objective functions were proposed. In their research, the constant c will not be found in the gradient search, and the optimal result will not be obtained if the gradients of $\|\eta\|_p$ and $g(x+\eta)$ are not in the same scale.

5. One Pixel Attack
Adversarial examples can be generated by modifying on just one pixel in images, which is called one pixel attack(Su et al., 2017). The problem of finding one pixel adversarial examples can be defined as:

$$\min_{x'} \ J(f(x'), l')$$

$$s.t. \ \ \|\eta\|_0 \leq \epsilon_0,$$

where ϵ_0 = 1 is defined to modify only one pixel.

2.3. Adversarial examples in an image classification task

The original images will be trained by an established image classifier to obtain the correct class labels of each. Adversarial images are then modified through the original images with small perturbations that can not be recognized by human. These small perturbations will lead to the miss-classification and the miss-classified label will be obtained from the classier at the same time. Here, in our experiment, the original images are all generated by generating models which are based on deep learning architectures with different number of convectional layers and fully connected layers according to different data complexity.

3. Method For Generating Universal Adversarial Examples

In the experiment, Universal adversarial perturbations algorithm (Moosavi-Dezfooli et al., 2017) was used to find the universal attack, and in the algorithm, DeepFool (Moosavi-Dezfooli et al., 2016) is another method that we used to find perturbations for individual data point and to update the universal attack.

3.1. DeepFool

DeepFool is a non-targeted attack method, i.e. its objective is only to fool a classifier to a different class. In DeepFool, minimal adversarial perturbation is defined as the closest distance from the target data point to the correspondent decision boundary (a hyperplane) the attacker want to attack.

Formally, assume a multiclasses classification problem, we have data $x \in \mathcal{X}$. For each class, there is a hyperplane that separate the feature space where x living in. For a data point x_0, it is classified into the class which the small space it lies is separated by those hyperplanes.

Assume we have a classifier f, the minimal perturbation r that make $f(x_0 + r) \neq f(x_0)$ is the one that can send x_0 to its nearest decision boundary:

$$r = -\frac{f(x_0)}{\|w\|_2^2} * w$$

Where w is the normal vector of the nearest decision boundary.

What the algorithm do is finding the closest hyperplane of x_0, pushing it toward the hyperplane and then pushing it a little further, so that $f(x)$ will misclassify x_0 as another class. The following equation gives the equation that calculate the closest hyperplane:

$$\hat{l}(\boldsymbol{x}_0) = \underset{k \neq \hat{k}(\boldsymbol{x}_0)}{\arg\min} \frac{\left| f_k(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) \right|}{\left\| \boldsymbol{w}_k - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)} \right\|_2}$$

Where $f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0)$ the classifier's possibility of the true label of x_0, and $f_k(\boldsymbol{x}_0)$ is the possibility of the most likely label k that are not equal to $\hat{k}$. The DeepFool algorithm is given below Figure 1, line 6-10 calculates the closest hyperplane, line 11-12 calculate the minimal perturbation and update x_0.

3.2. Universal Perturbation

(Moosavi-Dezfooli et al., 2017) presented the Universal Perturbation method to find a single perturbation that can fool most of the data points in a data set. Assume μ is the distribution of data $\mathcal{X}$ in $\mathbb{R}^d$, and $\hat{k}$ is the classifier that classifier a data point $x \in \mathbb{R}^d$ into an estimated label $\hat{k}(x)$, an Universal perturbation v is the one that makes:

$$\hat{k}(x+v) \neq \hat{k}(x) \text{ for most x} \sim \mu$$

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

Figure 1. Deep Fool Algorithm

There are two constrains to the universal perturbation, firstly, it should be small enough (measured by ℓ_∞ norm). secondly, it should be general enough (i.e. to fool 'most' data points). So we have:

$$\|v\|_p \leq \xi$$

$$\mathbb{P}_{x \sim \mu}(\hat{k}(x+v) \neq \hat{k}(x)) \geq 1 - \delta$$

Where ξ controls the magnitude of the perturbation, and δ controls the percent of data points in the data set that are 'fooled'.

The universal perturbation algorithm iterates over data points $x_i \in \mathcal{X}$ and build up v gradually by calculation the DeepFool perturbation of x_i. Concisely, assume that we already have an universal perturbation vector v , for each data point $x_i \in \mathcal{X}$, as long as v cannot fool x_i on the classifer($\hat{k}(x_i + v) = \hat{k}(x_i)$), we proceed to calculate the extra minimal DeepFool perturbation r that after being added to the already perturbed x_i, the new $x_i + v + r$ can be misclassified. The extra perturbation r is saved as Δv_i to update v.

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ , s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i)$$

The Δv_i is then constrain into a vector of at most ℓ_∞ norm ξ by the following transformation, and the transformed perturbation vector is then used to update the universal perturbation v:

$$\mathcal{P}_{p,\xi}(v + \Delta v_i) = \arg\min_{v'} \|(v + \Delta v_i) - v'\|_2 \text{ , s.t. } \|v'\|_p \leq \xi$$

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i)$$

The algorithm iterates over data points in the data set, until the desired "fooling rate" is met. In other word, the algorithm terminates when the rate of misclassification exceeds the threshold $1 - \delta$.

4. Experiment Setup

In this experiment, relationships between the existence of university perturbation and the dataset complexity, as well as the training model complexity should be found. There are three main steps to process experiments:

- Generating images with different complexity levels
- Training models with different number of layers
- Generating the universal perturbation based on a certain complexity level dataset and a certain number of layers training model.

4.1. Generating images

Some assumptions setting:

- images size and channel: 28*28*1
- number of classes: 4
- images number for each class: 1000

Firstly, 4 class baseline images should be created. See Figure 2.To generate these images, setting 1 as the pixel value on separated locations and uniform distribution random number ranged [0,1) on other unfilled locations. Making sure that the training model can recognise different class images. In this case, setting 1 on upper left corner 7*7 pixels and random number on other pixels represents class 0 images. And by the same logic, images of class 1, 2 and 3 are created.

While getting more complex dataset, more convolutional layers in the generating images model(called "images generator") should be added. The relation between them see formula (1). The weights and bias in convolutional layers do not need be learned in the images generator. These parameters are initialized and fixed by random values, uniform distribution random value for weights and normal distribution for bias.

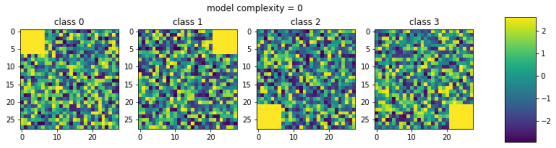

Figure 2. 4 class baseline images

$$number\ of\ layers\ in\ generator\ \propto dataset\ complexity\ level$$

After 2 layers convolutional computation, the baseline images are transferred into Figure 3. These images look like more complex than baseline images intuitively.

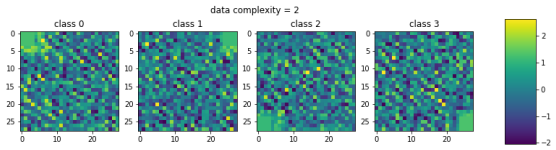

Figure 3. 4 class images after 2 convolutional computation

It should be mentioned that all pixel values of one image will be close to a certain number(in most cases, close to 0) after original images passing the images generator with many convolutional layers. This is probably because feature boundaries of different class images fades with increasing times of convolutional computations. So, these images pixel values should be scaled. Feature boundary in this experiment is represented by the location of pixels with value 1.

There are several comments and tricks about the images generator model

- To make sure that adding more layers in the images generator can get more complex dataset, weights in less layers of the images generator should be stored and used again on lower layers of the more layers images generator. Otherwise, if these weights are totally different random values, it will occur that the more layers generator may get lower complex dataset than the less layers generator. This is because all weights of the more layers generator are probably overall smaller than these weights of the less layers generator.
- This method to generate different complexity level dataset is reasonable because convolutional computation can gradually blur the decision boundary from baseline images dataset with increasing the convolutional layers.
- The number of convolutional layers in the images generator is in range from 0 to 9. When 10 convolutional layers are used in the generator, training model could not learn the decision boundary of these classes well, meaning testing accuracy is not good enough. This is probably because too many convolutional computations mix up the location information which can be recognised by the training model. In another word, too much layers in images generator will generate "garbage" images which can not be classified by training models well. After 9 layers convolutional computation, the baseline images are transferred into Figure 4. It is difficult to recognise which classes these images belong to by human eyes.
- To compare the size of universal perturbation generated from different complexity level dataset later, these images pixel values should be expanded or shrank to the same scale. So, these images pixels should be normalized after passing these convolutional layers.
- To avoid too much loss of baseline images' information, the activation function in each convolutional layers in images generator is LeakyRelu because other activation functions like ReLu, sigmoid, tanh or others, may induce values into saturation field, which cause the information loss.
- In the images generator, only convolutional layers are constructed because pooling layers may lead to the information loss from original images and dense layers may shuffle the location of representative pixels.

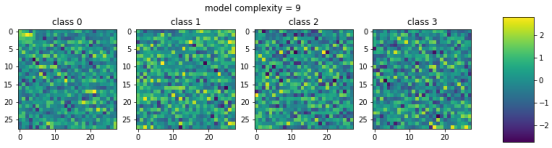

Figure 4. 4 class images after 9 convolutional computation

The structure of images generator with 4 convolutional layers(dataset complexity level 4) is shown in Table 1 and Figure 5.

To find the relation between the dataset complexity and the universal perturbation, the number of training model layers should be fixed on 4.

4.2. Training models

To get different complex training models, adding different number of layers can achieve this goal. Seven kinds of layers number are record: 3, 5, 8, 11, 14, 17 and 20. 3 layers training model includes 1 convolutional layer and 2

Height	Width	Depth	filter Height	filter Width
36	36	1	3	3
34	34	32	3	3
32	32	32	3	3
30	30	32	3	3
28	28	1	None	None

Table 1. The structure of images generator with 4 convolutional layers

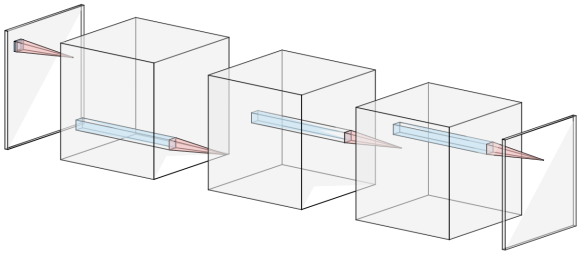

Figure 5. The structure of images generator with 4 convolutional layers

Layer	Height	Width	Depth	filter Height	filter Width
Convolution	28	28	1	3	3
Convolution	26	26	32	3	3
Dense	24	24	128	None	None
Dense	24	24	32	None	None
Output	4		1	None	None

Table 2. The structure of training model with 4 layers

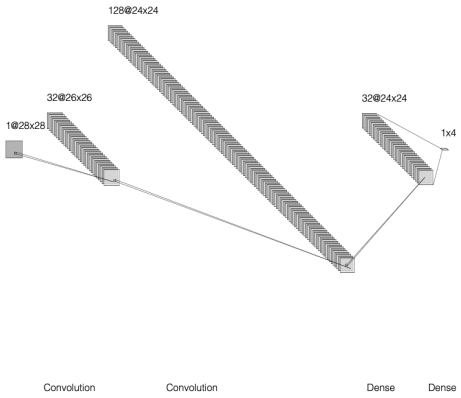

Figure 6. The structure of training model with 4 layers

fully connected layers. Similarly, for 3, 5, 8 and 11 layers training models, they include 2 fully connected layers and #layers-2 convolutional layers. With increasing too many convotional layers, the traning model can not perform well, but adding more fully connected layers based on sufficient number of convolutional layers can improve the training model performance again. So, for 14, 17 and 20 layers training models, they include 11 convolutional layers and #layers-11 fully connected layers.

In addition, with increasing the number of layers in the training model, smaller learning rate should be applied because large learning rate could not help complex training models converge.

After fine tuning, all training models can reach up 99% testing accuracy within small epochs. The activation function for all layers in all training models are ReLu.

The structure of training model with 4 layers(2 convolutional layers and 2 dense layers) is shown in Table 2 and Figure 6.

To find the relation between the training model complexity and the universal perturbation, the dataset complexity level(number of layers in images generator) should be fixed on 4.

4.3. Generating the universal perturbation

Some hyper-parameters setting

- Images used to train the universal perturbation: 400 and 100 for each class
- Baseline Method to generate perturbation: Deep Fool
- Fooling rate $> 70\%$
- Metric to calculate the magnitude of universal perturbation: Infinity Norm $||\xi||_\infty$
- Maximum number of iterations to train the universal perturbation: 10

5. Result Analysis

To visualize the Table 3, see Figure 7. The shallow line represents the original data points and solid line is the more smooth line generated from the shallow one by using Savitzky-Golay filter (Luo et al., 2005). As shown in table, for example, when the data complexity is 0 and the number of training models layers is 4, if adding the universal perturbation is smaller than 4.2, then the fooling rate will less than 70%. The value of ξ is the boundary to determine whether fooling rate is greater or smaller than 70%. So this value can show how easy the universal perturbation exists.

See Figure 8, when dataset complexity level is 1, adding universal perturbation with infinity norm 2.0 can fool training model to recognise one image of class 2 as class 3.

Data complexity	Model layers	ξ
0		4.2
1		2.0
2		1.8
3		1.8
4		1.8
5	4	1.6
6		1.6
7		1.6
8		1.1
9		0.9

Table 3. Relation between dataset complexity and universal perturbation

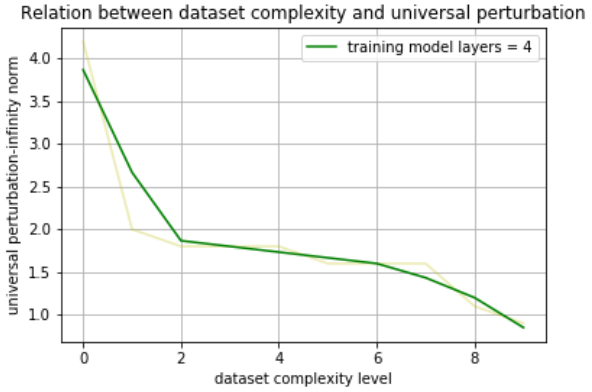

Figure 7. Relation between dataset complexity and universal perturbation

See Figure 9, when dataset complexity level is 9, adding universal perturbation with infinity norm 0.9 can fool training model to recognise one image of class 1 as class 3.

To compare these two universal perturbations, the universal perturbation with larger infinity norm changed the original images information more, like brute force to change one class image to another class. This kind of universal perturbation is relatively meaningless than slight perturbation in real world.

When the dataset become more complex, smaller universal perturbation can be added on images to fool the training model well.

To visualize the Table 4, see Figure 10.

Overall, when the training model become more complex, smaller universal perturbation can be added on images to fool the training model well. In this experiment, the universal perturbation generated from 20 layers training model is slightly larger than those generated from 14 and 17 layers training models. This is probably because some bias of this

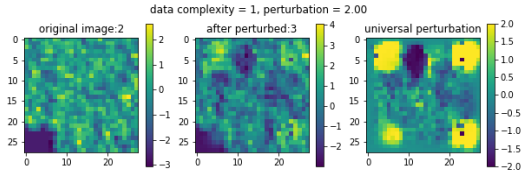

Figure 8. Example of universal perturbation with infinity norm 2.0

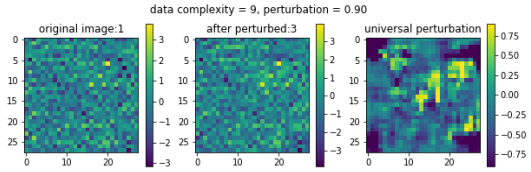

Figure 9. Example of universal perturbation with infinity norm 0.9

experiment occur.

6. Discussion and Future Work

6.1. An explanation to universal perturbation

In (Moosavi-Dezfooli et al., 2017), Moosavi-Dezfooli presented an explanation to the vulnerability of DNN classifiers to universal perturbation. Moosavi-Dezfooli compared Universal Perturbation with 1) random noise 2)adversarial perturbation computed by single sample 3)mean of images (images bias) 4)sum of adversarial perturbation over X, and the result shows that Universal Perturbation can obtain the highest fooling rate with small ξ(see Figure 11).

The result shows that by calculating DeepFool perturbation over all data points, universal perturbation must capture some of the common characteristics of the decision boundary, in other word, there is geometric redundancy in the decision boundary (different data points' direction to its nearest decision boundary is correlated), and the algorithm learned from the data about such a correlation. By calculating Deep-

Model layers	Data Complexity	ξ
3		2.7
5		2.4
8		1.5
11	4	1.5
14		1.3
17		1.2
20		1.5

Table 4. Relation between training model complexity and universal perturbation

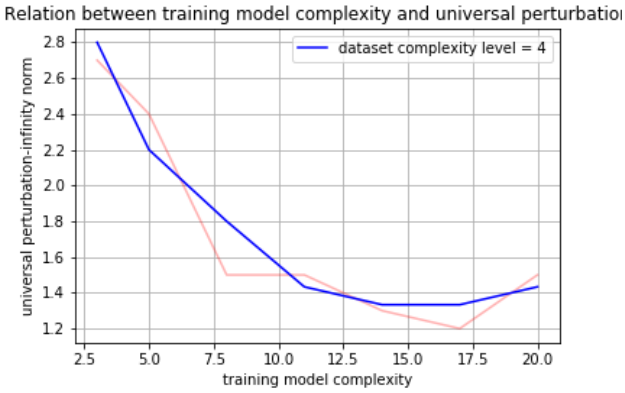

Figure 10. Relation between training model complexity and universal perturbation

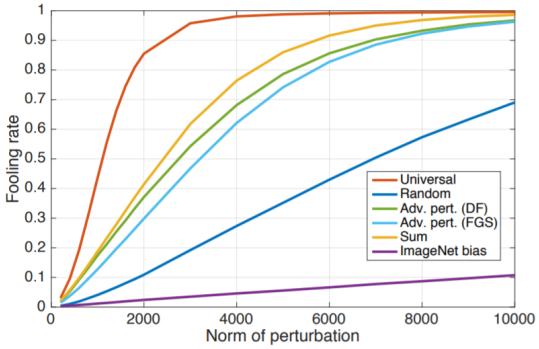

Figure 11. Comparison between fooling rates of different perturbations. Experiments performed on the CaffeNet architecture. (Moosavi-Dezfooli et al., 2017)

Fool perturbation, we are actually finding a normal vector of a data point to its nearest decision boundary. To quantify the linear correlation the structure of decision boundaries (to what extent there is a common direction vector which most data points share), Moosavi-Dezfooli calculated each data point's normal vector to its respective nearest decision boundary for data points in the validation set:

$$N = \left[\frac{r\left(x_1\right)}{\left\|r\left(x_1\right)\right\|_2} \cdots \frac{r\left(x_n\right)}{\left\|r\left(x_n\right)\right\|_2} \right]$$

Moosavi-Dezfooli calculated the singular values of matrix N and compare it with the singular values computed from uniformly sampled random noise matrix of the same shape(see Figure 12).

The singular values of N decay quickly, while the random noise's singular values decay gently, which indicate that

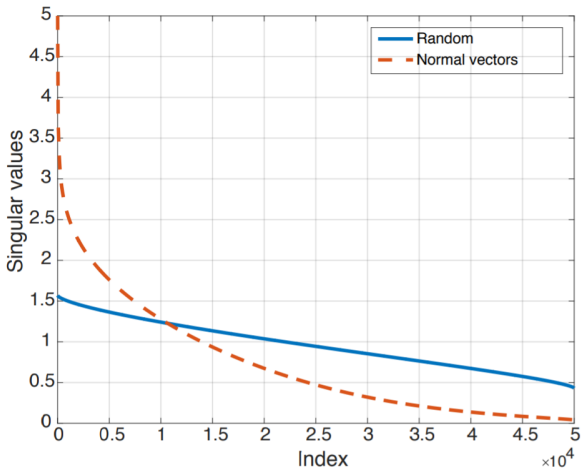

Figure 12. Singular values of matrix N containing normal vectors to the decision boundary (Moosavi-Dezfooli et al., 2017)

there exists linear redundancy in the decision boundary. In other word, there exists lower dimensional vector space $\mathcal{S}$ which contains most normal vectors of natural images in ImageNet to their nearest decision boundary. As a result, the existence of universal perturbation may due to the subspace that contains most normal vectors to the decision boundary. Actually, random vector that belong to a subspace that spanned by the first 100 singular vectors of N can fool near 38% natural images in ImageNet, significantly better than random noise, which is 10%.

6.2. Future works

For the bias in the experiment, more experiments should be done to analyse why this noise happen. For example, adding more than 20 layers in training model to analyze the infinity norm of the universal perturbation.

In this experiment, adding more convolutional layers in images generator can control the complexity level of the datasets. But in real world, some methods should be found or applied to evaluate semantic datasets complexity, such as spectral metric (Branchaud-Charron et al., 2019) or other complexity measures.

Calculate the difference of different class images in various complexity level datasets by using Kullback–Leibler divergence of other metrics. Find whether the difference of different class images in complex datasets is smaller than that in simple datasets.

Test the universality of universal perturbation and find whether there is a "global" universal perturbation. In this experiment, universal perturbation is "locally" universal,

meaning it is a pattern can fool the training model trained on a certain dataset and may not fool the training model trained on other datasets in a high fooling rate.

7. Conclusion

To summarise, when the dataset and training model become more and more complex, the decision boundary gradually become unclear and difficult to be learnt by classifiers. So, there may exist an universal perturbation or a pattern to disturb original dataset information, inducing one class images can be added a small perturbation to "skip" into other decision areas.

References

Branchaud-Charron, F., Achkar, A., and Jodoin, P.-M. Spectral metric for dataset complexity assessment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. URL http://arxiv.org/abs/1608.04644.

Carlini, N. and Wagner, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR*, abs/1705.07263, 2017a. URL http://arxiv.org/abs/1705.07263.

Carlini, N. and Wagner, D. A. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR*, abs/1711.08478, 2017b. URL http://arxiv.org/abs/1711.08478.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2014.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *ICLR Workshop*, 2017. URL https://arxiv.org/abs/1607.02533.

Luo, J., Ying, K., and Bai, J. Savitzky–golay smoothing and differentiation filter for even number data. *Signal Processing*, 85(7):1429 – 1434, 2005. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2005.02.002. URL http://www.sciencedirect.com/science/article/pii/S0165168405000654.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: A simple and accurate method to fool deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015. URL http://arxiv.org/abs/1511.07528.

Rozsa, A., Rudd, E. M., and Boult, T. E. Adversarial diversity and hard positive generation. *CoRR*, abs/1605.01775, 2016. URL http://arxiv.org/abs/1605.01775.

Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017. URL http://arxiv.org/abs/1710.08864.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.