Adversarial attack and Defense

1st 鄭幸怡 109062305
NTHU, CS
Hsinchu, Taiwan
sharren89776@gapp.nthu.edu.tw

2nd 段欣妤 109062313
NTHU, CS
Hsinchu, Taiwan
lily910502turn@gmail.com

3rd 葉明淳 062140
NTHU, CS
Hsinchu, Taiwan
emilyyeh0621@gapp.nthu.edu.tw

4th 何苡歆 109062306
NTHU, CS
Hsinchu, Taiwan
losyoply.km@gmail.com

***Abstract*—Machine learning models are widely used in real world. A false result may leads to fetal consequences. Thus, it's important for the models to be robust when facing hostile attacks. In this project, we use FGSM, I-FGSM method to apply adversarial attack on image classifying model MobileNet under white-box and black-box condition, and do further passive dense using Gaussian blurring and randomized defense against our attack.**

Keywords—FGSM, adversarial attack, passive defense, black box attack, white box attack, I-FGSM, machine learning, Gaussian blurring

I. Introduction

Machine learning is widely used in everyday life, like being put to use to save lives by Infervision or detecting fraud in near real-time, then saving millions in losses. Machine learning may be used to make important decisions. Thus, a false detection or prediction may lead to fatal results such as losing a serious amount of money and even making people die. For example, if a medical application with machine learning-based models generates a false result, it may lead to a false alarm or make the patient thinks there's no problem. Thus, machine learning models which are used to make important decisions should be robust, and shouldn't be vulnerable to hostile attacks.

We use adversarial attack to simulate an attack and try to defend accordingly. Adversarial attack is to use adversarial examples to make models generate the wrong result. Adversarial examples are some input designed on purpose to mislead the model, it is generated by adding some signals that cannot be detected by human eyes.

There are many types of attacking. We can divide them into the white-box attack and black-box attack according to what the attacker knows about the model details. Moreover, we can also divide them into the targeted attack and untargeted attack according to what the attacking goal is.

Then about the part of the defense, we use passive defense. Passive defense means not changing the model itself, but only adding a filter as a shield before data input to the model for resisting attacks. This Filter can reduce the influence of the signal added by the attack to the image, and does not affect those who are not attacked.

II. Methods

A. *Attack*

We are attacking MobileNet, an image-classifying model, trained on the dataset ImageNet. MobileNet is a model designed for mobile devices or embedded systems. Due to its compact size, it can generate a result with lower latency compared to other models such as ResNet50, AlexNet, VGG16, etc, which is good for applications such as self-driving-car pedestrian detection.

1) Fast Gradient Sign Method (FGSM)

[2]FGSM is a way to generate adversarial examples. The core concept is to maximize the loss function, which is on the contrary of backpropagation. The mathematic expression is listed as the following:

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right).$$

- θ : the parameters of a model
- X : the input to the model
- Y: truth label or targets
- J (θ, x, y) : the loss function with respect to the model
- ε: step size

The larger the step size is, the more obvious the generated perturbation is. The above method could be condensed into 4 steps:

✧ forward-propagate our image through our network
✧ calculate the loss
✧ back-propagate the gradients to the image
✧ nudge the pixels of the image in the direction that maximizes the loss value

An intriguing fact is that the gradients are taken with respect to the input image, since our objective is to create an image that maximizes the loss without modifying the parameters of the attacked model. Hence by applying chain rule, the according gradients of each pixel of input image could be calculated.

2) Iterative-Fast Gradient Sign Method (I-FGSM)

Iterative-Fast Gradient Sign Method is an extended version of FGSM. It was first introduced in[1], and was designed for boosting FGSM's performance. Basically, it is done by repeatly applying FGSM result on the input, and clip it to ensure the intermediate result is in input's ε-neighbourhood. The equation is as the following:

$$3)\ X^{adv} = X,\ X^{adv} = Clip\ (X^{adv} + \varepsilon * sign(\nabla J(X^{adv}, y)))\ _{0\ N+1\ X,\varepsilon\ N}X\ _{N}\ true$$

- X - Input. In our project, input is an image, which is a 3-D tensor (R, G, B). R, G, B are integers in range [0, 255].
- y_{true} - True class for the image X.
- J(X,y) - Cross-entropy cost function of the neural network, given image X and class y.
- $Clip_{X,\varepsilon}\{X'\}$ - Function which performs per-pixel clipping of the image X′, so the result will be in L_∞ ε-neighbourhood of the source image X. The exact clipping equation is as follows:
  $Clip_{X,\varepsilon}\{X'\}(x, y, z) = \min(255, X(x, y, z) + \varepsilon, \max(0, X(x, y, z) - \varepsilon, X'(x, y, z))$
  where X(x, y, z) is the value of channel z of the image X at coordinates (x, y).
- ε: step size

B. Defense

A straightforward method is to add the adversarial examples into the training set, fix the predicted label, and retrain the model again, which is called proactive defense. But since the model we are attacking is a huge model, it costs more computation resources to implement this method. Hence, we choose to implement passive defense instead. We perform passive defense using two methods: gaussian blurring and randomize defending.

1) Gaussian blurring

To reduce the impact of perturbation, we can add a Gaussian blurring filter on the input image. Gaussian blurring filter can make an image more smooth, blurring the perturbation, thus decreasing its influence on the input image. Applying gaussian blur is to convolve the image with a Gaussian function, Gaussian function on 2-dimension input is described as the following:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- x, y: input coordinates.
- σ: standard deviation of Gaussian distribution

2) Randomized defense

Randomized defense is first published in [5]. The defensing flow chart is as the following:

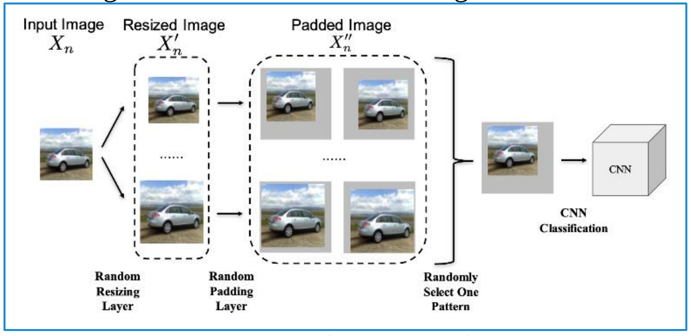

Fig 2.1 Flow chart of randomized defense

First, resize the input image randomly. Let the original input image I has the size W*H*3, and the resized image I' would have size W'*H'*3.

Next, randomly pad the resized image into the size that which classifying model takes in. The padded image should has size of W''*H''*3, where W'' and H'' are the size the classifying model takes in. We can choose to pad w pixels on the left, W''-W'-w pixels on the right, h zero pixels on the top and H'' − H'− h pixels on the bottom. This results in a total number of $(W'' - W' + 1) \times (H'' - H' + 1)$ different possible padding patterns.

After resizing and padding, we can get I'' with the corresponding size of which the model takes in.

III. Results

A. Attack

In this section, we conduct experiments on a carefully selected subset of the ImageNet dataset, which contains 100 pictures evenly of 100 top classes of ImageNet dataset classes. We try to attack the models using the method introduced in the prior sections and thus validate their effectiveness. Specially, we apply these methods under different conditions, namely white-box attack and black-box attack. In the widely explored white-box attacks, the attacker has full access to the architecture of the attacked model, while the black-box attack has limited information, which is much more practical for real-world applications. In our experiment, both methods are applied.

In the following article, we defined "change the result of prediction" as successful attack in untargeted FGSM, "change the result of prediction to the target" as successful attack in targeted FGSM.

1) White-box

a) One shot FGSM

untargeted

In the experiment of untargeted FGSM one-shot attack, we ran over 1000 images of a variety kind of objects and tried to attack it aiming at confusing the image recognition of MobileNet. In this experiment, the results are as the following graph:

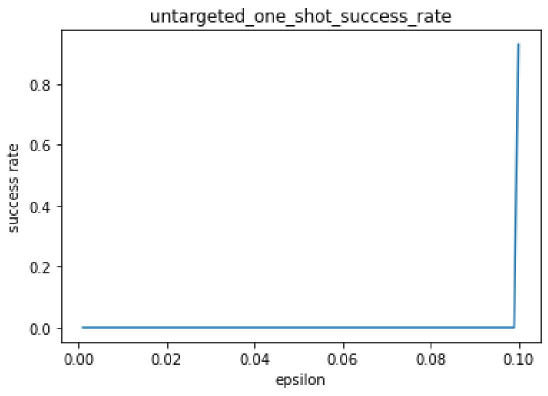

Fig 3.1 The success rate of untargeted one shot FGSM. X-axis: epsilon, y-axis: success rate

The success rate steeply rises to about a hundred percent when the epsilon closes to 0.1. This result indicates that the epsilon of this experiment shouldn't be too small which will always result in a failed attack. On the contrary, if we reached the epsilon that is enough for the one-shot attack, we can almost never fail to attack the model under this epsilon.

Targeted

In the targeted FGSM one-shot attack. We ran over 1000 images and computed the success rate of confusing the model with different Epsilon in this kind of attack. We specified the target as a highly aggressive object, an assault rifle. The following is the result:

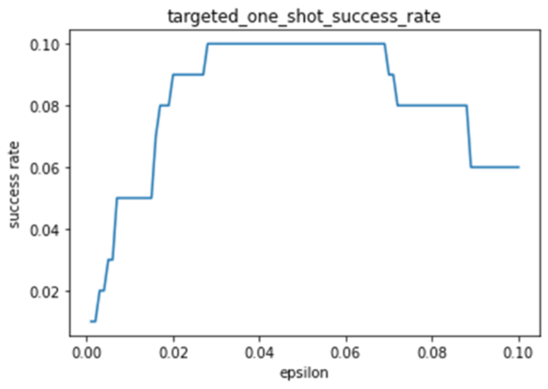

Fig 3.2 The success rate of targeted one shot FGSM. X-axis: epsilon, y-axis: success rate

This graph shows the variation through different epsilons from 0.001 to 0.1. We can see that the success rate of targeted one-shot attacks rises in the first half of the graph, where the epsilon is between 0.001 and 0.03. And dropped obviously after the epsilon is about 0.07.
Therefore, for the one-shot attack, we are most likely to confuse the model when the epsilon is from 0.03 to 0.07, where there is a ten percent possibility to make the result of image recognition to a highly aggressive object, an assault rifle.

Iterative FGSM

Untargeted

By the iterative FGSM attack, we defined we successfully attacked the model by making the attacked prediction different from the original prediction, in addition, the confidence of the misled prediction should be over 0.90 at the same time to make our attack more influential.
The following are the results:

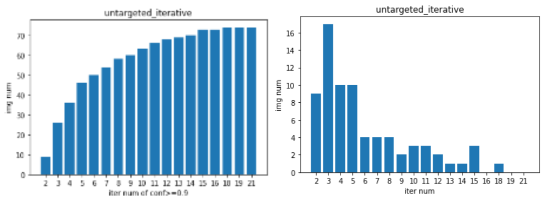

Fig 3.3 The success rate of untargeted I-FGSM. X-axis: iteration count, y-axis: success rate

They are respectively the number of images that can be successfully confused in different iterations, and the total number of images that can be successfully confused in different iterations. We set the max iteration number to 100, to make sure that it won't take too much time. From these results, we can find out that there are still about 20 images that we can't reach the demanded confidence in 100 iterations. Moreover, most images can be successfully attacked after merely 15 iterations, and in the fifth iteration, the success attack rate can reach 50 percent, which indicates that most attacks are already finished in the first five iterations.

Targeted

Similarly, same to the untargeted iterative FGSM attack, we defined the attack success by making the model predict the images as assault rifle, our specified target, to a confidence greater than 0.9. Then the results are as follows, the y labels of the graphs represent the iteration number requested, and the x labels represent the image number and the total image number respectively we successfully attacked under different iterations.

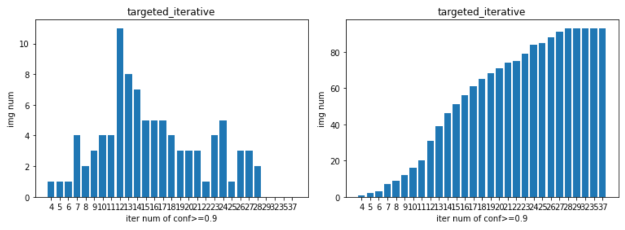

Fig 3.4 The success rate of targeted I-FGSM. X-axis: iteration count, y-axis: success rate

From the results, we can conclude that most images can be succesfully attacked after 28 iterations. Moreover, we can reach a 50 percent success rate when we do about 14 iterations.

2) Black-box

A black box attack is one where we only know the model’s inputs, and have an oracle we can query for output labels or confidence scores. An “oracle” is a commonly used term in this space that just means we have some kind of an opaque endpoint we submit our inputs to that then returns the model output(s). A number of black box attacks involve model extraction to create a local model (sometimes known as a substitute or surrogate model). However, a black box attack can also skip model extraction and directly query inputs against the target model. These attacks, where the internal configuration of the model is not needed at all, are what’s actually known as black box attacks in the academic literature. In our experiments, we choose to conduct the black-box attack that queries inputs images and accorded labels. Similarly, one-shot FGSM (ordinary FGSM) and I-FGSM (iterative FGSM) are both applied as white-box attacks.

One shot FGSM

Untargeted

With a pre-trained model(MobileNet V2), we first create a NumPy array to store the value of success rate under different epsilons, named Epsilons[ ]. Then we loop over the pictures in our dataset and for each picture, there are 100 epsilon values varying from 0.001 to 0.1. We are interested to find out which epsilon value works best for one-shot FGSM. For each epsilon value in the Epsilons array, the according to success rate indicates that there are more pictures been successfully perturbated into a different (untargeted) label. The following graph demonstrates the results:

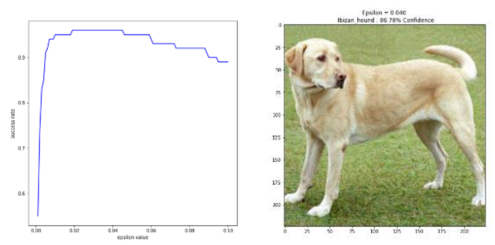

Fig 3.5 Shows that around Ɛ=0.04, the one-shot FGSM has relatively good success rate on attacking our dataset. Here we pick out the picture of golden retriever after perturbated with Ɛ=0.04, and misclassified as ibzian hound. The perturbation samples is hard to distinguish by human eyes.

Besides, we found that between Ɛ=[0.001, 0.1], the epsilon value must exceed a certain value so that the attack can achieve a certain success rate (larger than 0.9), in other words, there exists a threshold epsilon value for this dataset black-box attack. Additionally, the success rate falls after a certain epsilon value of around 0.05.

Targeted

In a similar way, we also record success rates for different epsilon values, however, this time we aim to misclassify those pictures into a target label, the “assault rifle gun”. The following graph shows that the success rate of each epsilon value range from 0.001 to 0.1, and the success rate is obviously lower than the targeted attack. The best success rate is only 0.44 in this experiment.

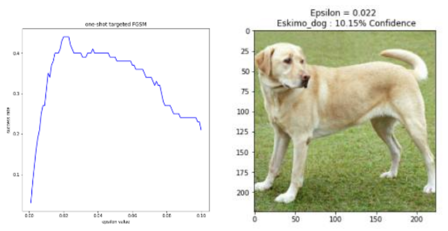

Fig 3.6 It is clear that Ɛ=0.022 has highest success rate of targeted attack. There are 44 pictures been mislabeled as the specific target. However, when we try to apply this Ɛ on the prevois golden retriever picture, it was misclassified into Eskimo dog, instead of a rifle gun.

Fig 3.7 An assault rifle gun

Clearly, the targeted attack is quite hard to achieve a good success rate, especially when the distribution of the dataset is random or unknown. As in our case, our dataset maintains diversity to a certain degree, thus leading to the poor success rate of the targeted “assault rifle gun”, which was decided by own preference without clues.

Iterative FGSM

Untargeted

With the pretrained model(MobileNet V2), we want to find out over what iteration times can our attack succeeds on most of our datasets, which means those datasets would be misclassified. Differing from one-shot FGSM, we set up an iteration time, say T. If the attacker could fool the model within T iterations, we consider the attack for a single picture to be successful. Here we choose epsilon as 0.001, in hope that small perturbations can contribute to good results with the lowest risks of being discovered by human eyes.

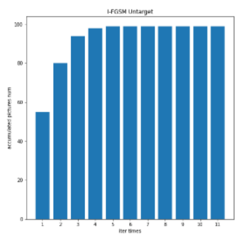

Fig 3.8 Bar graph of successfully misclassified pictures numbers(accumulated) / iteration times.

Fig. 3.9 A prayer rug that needs 5 iterations to be misclassified

This bar graph shows that over 50% of pictures of our dataset were mislabeled within 1 iteration (one-shot FGSM), and all of our datasets were mislabeled within exactly 5 iterations. The results coordinate with that in the one-shot untargeted FGSM experiment, since both of them show that under Ɛ=0.001, over 50% of the dataset classification would be distorted. Additionally, there are still some pictures that need more iterations, such as prayer rugs. In total, there are 14 pictures that need at least 3 iterations and 5 pictures that need at least 4 iterations. Through this experiment, we can figure out appropriate iteration times for each of the epsilon values.

Targeted

In I-FGSM (iterative FGSM), it provides a higher chance for the perturbations to fool the model, because the perturbations were accumulated on the original pictures, thus made the new image more distorted from the raw input image. However, to achieve 100% confidence of targeted attack is quite challenging, thus we set those misclassified with confidence over 90% as successfully attacked.

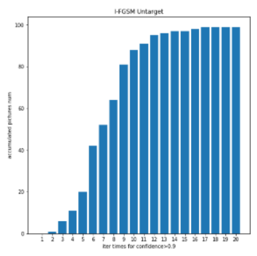

Fig 3.10 Bar graph of successfully misclassified pictures numbers(accumulated) / iteration times.

As we mentioned before, targeted attack is quite hard compared to untargeted attack. The implemention is no big deal, but what really costs time is, even if we loosen the constraint on the definition of "success", the iteration times is almost twice the times in untargeted attack if we want to obtain over 50% of our dataset to misclassified. (By observing that when iteration times=7, the accumulated mislabeled pictures numbers is near 50% but not 60%). Fortunately, every pictures of our dataset could be misclassified after 17 iterations, which implies our target label and I-FGSM is feasible on the dataset we choose.

B. *Defense*

We performed adversarial defense on our attacking results, using two methods: Gaussian blurring and randomized defense.

1) **Gaussian blur**

The success rate of Gaussian blur passive defense has a success rate of about 68%. In the function of gaussian blur, if the parameter sigma is larger, the picture will become more blurred.

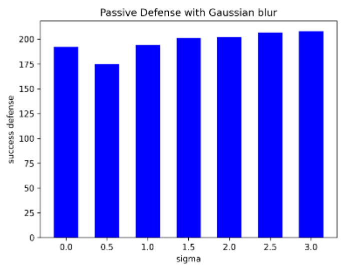

Fig 3.11 Bar graph of passive defense result with Gaussian blurring.

The above data is a statistical bar chart of the gaussian blur passive defense results generated according to the standard deviation (sigma) of the four attack models, which are iterative targeted black box attack, iterative untargeted black box attack, one shot targeted black-box attack, and one shot untargeted black box attack.

It can be seen from the data that the effect is relatively poor when the standard deviation (sigma) = 0.5. This is actually beyond our original expectations. We originally thought that the more blurred the image is, the more difficult it is for the original model to recognize what it was. But after experiment and observation, unless the original image is too low in quality and very blurred, most of them can still achieve a good defense success rate.

2) Randomized defense

We also tested the effect of the randomized padding function on passive defense under different resize low bounds. In the randomize padding function, if the resize low bound is smaller, it means that the padding range of the image will be larger.

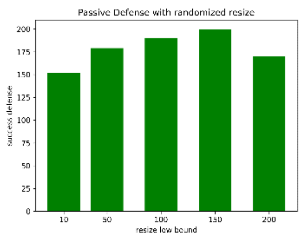

Fig 3.12 Bar graph of passive defense result with randomized defense

The above data is a statistical bar chart of randomized padding passive defense results generated according to different resize low bounds for four attack models, which are iterative targeted black-box attack, iterative untargeted black-box attack, one shot targeted black box-attack, and one shot untargeted black-box attack.

It can be seen from the data that when the low bound is at 10/224, the effect is the worst, and when the low bound is at 200/224, the effect is the second worst.

In fact, setting the low bound to the minimum of 10/224 is predictable for the worst effect, because it means that the picture may be randomly changed its size with a huge difference in length and width. Then it would result in serious deformation of the picture after padding, making the original model unable to correctly predict the original label. Surprisingly, under the condition of low bound 200/224, the picture can be said to be almost not changed, and the effect is not very obvious. We are speculated that because there are not too many changes that there is no way to achieve the effect of defense.

IV. Conclusion / Discussion

A. Conclusion

1) *Attack*

a) Comparison between white box and black box attack

For fairness, our white box attack and black box attack use the same dataset, as well as the same target label("assault rifle gun"). Here are some interesting results:

For the untargeted and targeted one-shot attack, we use different epsilons to implement and observe the results.

In the untargeted one-shot attack, the results of white-box attack and black-box attack differ apparently. In the white box attack, we need greater epsilon up to 0.1 to achieve the steep rise in success rate. In the contrast, the success rate can immediately reach 0.9 after the epsilon is greater than 0.001 in the black box attack, and remain a relatively stable success rate.

Similarly, in the targeted one-shot attack, the results of white-box attack and black-box attack are also significantly different. Although their success rate both rise and reached the max around the epsilon of 0.02, and drops after the epsilon reach 0.07, the maximum success rate of black-box attack is outrageously greater than the maximum success rate of black box attack, which the maximum success rate is 0.4 in black box attack, however, 0.1 in white box attack.

For the i-FGSM part, we can easily learn the relation between the different number of iterations and the change in the success rate.

In the untargeted i-FGSM attack, the white box attack requests 7 iterations to reach a 50 percent success rate to fool the input image, nevertheless, in the black box attack needs only one iteration to reach a 50 percent success rate. Moreover, almost 100 percent of images fooled the model in the fourth iteration of the black-box attack, however, the white-box requests up to 7 iterations, which seems to reveal that black-box attack has better efficiency.

The comparison of the results in the targeted I-FGSM attack is similar to the previous one, the white box attack requests 14 iterations to reach a 50 percent success rate to fool the input image to an assault rifle, nevertheless, the black box attack needs only 7 iterations to reach a 50 percent success rate. And it needs approximately 17 iterations for black-box to reach almost 100 percent of images successfully attacked, in contrast, the white box attack needs up to 29 iterations to make almost all input images fool the model.

b) Why we choose our target label as "rifle"?

In the hope our experiment could have a deeper meaning for practical use, we choose "assault rifle gun" as our targeted label in a targeted attack. Since guns are prohibited in Taiwan, if an object was misclassified as a gun, then this may lead to chaos in public areas, and even more, serious accidents may occur and endanger people's life.

c) Overall summary

We've tried much epsilon throughout our experiment of one-shot FGSM. While in iterative FGSM we set up constant epsilon values and explore the iteration times we need for our dataset. We understand that higher epsilon provides a better success rate along with the risk of being discovered by human eyes, and more iteration times mean more time cost. By combining the results of these two methods, we can find out the best epsilon as well as the best iteration times of certain datasets. For example, in a black-box attack, we can choose Ɛ=0.022 and iteration time=8 for targeted I-FGSM to accelerate the whole process. The above indicates one way to improve the attack when someone tempts to fool the model with all of the datasets. Therefore, if the goal is to succeed in a certain percentage of the dataset (70% for example), then the epsilon value and the iteration times could be further adjusted.

During the experiment of a one-shot untargeted attack (or targeted), the process of figuring out the best epsilons is similar to the way of iterating perturbations in I-FGSM. When we set Ɛ=0.003 in one-shot FGSM, this equals that we iterate 3 times in I-FGSM, starting with Ɛ=0.001.

It is clear that when implementing a targeted attack, the success rate was not as high as an untargeted attack. This might side-prove that our dataset ensures diversity to a certain degree because when a dataset is robust enough, the target label should be sophisticatedly designed in order to mischieve the model. If not, then the success rate of the whole dataset might be quite low. On the contrary, a mono dataset would turn out to be vulnerable since the distribution of class labels is too concentrated so that the attacker can easily predict the output label and changes it into another one.

2) *Defense*

a) comparison between gaussian blur filter and randomized padding filter

From the results, it seems that the success rate of gaussian blur passive defense is relatively higher, with a success rate of about 68%. Then, the success rate of randomized padding passive defense is relatively lower, and the success rate is only about 61%. However, considering if the attacker knows our defense strategy details, he can modify the attack method and make the defense useless. It is safer to use randomized padding for passive defense because its parameters are randomized.

b) Overall summary

In conclusion, gaussian blur passive defense is better than randomized padding passive defense, and sigma = 3 is the best one. The advantage of randomized padding passive defense is that it is not easy for the opponent to modify the attack strategy, and the best effect can be achieved by controlling the randomized padding within the range of 50%-100%. If the strengths of the two can be combined for defense, it must be able to achieve a more ideal defense effect.

B. Discussion

In our project, we not only implemented an adversarial example attack using FGSM and I-FGSM but also did passive defense against the attacks. We compared the performances using different parameters and gave some ideas for designing models to perform or prevent adversarial attacks. Due to time issues, we only implemented the above methods, but there are more methods to perform adversarial attack/defense in the literature that can achieve more performance. In the end, we know that it’s important to prevent adversarial attacks when designing models, and know that there are ways to crack other people’s model results. These knowledges will be useful for our future projects.

V. Authors and Contribution

鄭幸怡 (25%): study design, programming of the FGSM, figure design and writing.
段欣妤 (25%): study design, programming of the passive defense, figure design and writing.
葉明淳 (25%): study design, data collecting and analysis, statistical interpretation, figure design and writing.
何苡歆 (25%): study design, programming of the white box FGSM, figure design and writing.

VI. References

[1] Kurakin A, Goodfellow I, Bengio S. Adversarial machine learning at scale[J]. arXiv preprint arXiv:1611.01236, 2016.

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

[3] Junyu Lin, Lei Xu, Yingqi Liu, Xiangyu Zhang, Black-box adversarial sample generation based on differential evolution, Journal of Systems and Software, Volume 170, 2020,110767, ISSN 0164-1212

[4] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2017). Boosting Adversarial Attacks with Momentum, arXiv.1710.06081

[5] Xie, Cihang et al. “Mitigating adversarial effects through randomization.” *ArXiv* abs/1711.01991 (2017): n. pag.