SPML Final Project Report Group 6
Generation of Adversarial Patch: Variation and Application

Jun-Da Chen
Data Science Degree Program
National Taiwan University
R08946014
r08946014@g.ntu.edu.tw

Yu-Wei Chen
Graduate Institute of Communication Engineering
National Taiwan University
R09942066
r09942066@ntu.edu.tw

Abstract

Adversarial patch can be generated in many ways and types, for instance, gradient-based method and EOT on ensemble model. We are curious about generation of adversarial patch and its variation and application. In this report, we demonstrate the result of a novel adversarial patch generating method, which using image-to-image translation mechanism, and try to beautify and disguise adversarial patch using style transfer and special loss constraint.

1 Introduction

Adversarial patch is a type of attack that any image with adversarial patch will be recognized as target class. For example, no matter what image contains (apple, phone, human, etc.), when it adds a adversarial patch of class toaster, all of them will be classified as toaster, and this attack can even be used in physical world. Adversarial patch can be generated in many ways, for instance, gradient-based[1] and EOT[2] on ensemble model and types. However, most of adversarial patch appearance are weird and suspicious. Thus, in this work, we will try to use style transfer and add other constraint to beautify adversarial patch. Meanwhile, generating process of adversarial patch needs to consider many parameters, such as location, rotation, etc., so we take experiment of parameter-free adversarial generation method using image-to-image translation mechanism.

The summary of our contribution are shown following:

- Experiment and simple analysis of using image-to-image translation to generate adversarial patch.
- Use style transfer to beautify adversarial patch.
- Use dark channel prior loss to constraint and beautify adversarial patch.
- Optimize and compare implantation of most popular adversarial patch repository[1]

The slides of final presentation can be view in the link. [1]

[1] https://docs.google.com/presentation/d/1FUPcD4Rk2tNgiWKkZ89FMVmH4V6Eyx6nJOt4Ig6IOOO/edit?usp=sharing

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 Related Work

2.1 Style Transfer

Style transfer is a technique used to take two images, a content image and a style image, and blend them together so that the output image looks like the content image, but painted in style of the style image. In this work, we mainly refer to fast-neural-style-transfer[3]. The model contains two network, including image transform net and a loss network. The image transform net contains residual block and without any pooling layer, instead using stride and fractionally stride convolution; Loss network is VGG16, and the loss function are perceptual loss and feature reconstruction loss. That input will be a image, through image transform net, output will be style transferred image, and estimated perceptual loss of style image and content image for different layers, lower (bottom) layer will content more low-level feature, i.e, corner, edge, and higher (top) layer will contain high-level feature, i.e, texture, style, etc.

2.2 Image to Image Translation

Image-to-image translation is a technique that the goal is to learn the mapping between input image domain and output image domain. To accomplish this goal, JY Zhu et al. proposed CycleGAN with cycle-consistent loss[4], however, generalization of CycleGAN is not good, for many data, CycleGAN struggle to generate visual-pleasing result. Ming-Yu Liu et al. proposed UNIT[5], which is unsupervised image-to-image network to improve the result and become state-of-art. In this work, we will mainly refer to this paper. In UNIT[5], the network including 2 encoder and 1 coGAN[6], it can also be viewed as 2 VAE and 2 discrimintor. The encoder mapping input image into shared-latent space, which means the latent is shared in input image and output image domain. Then the input of coGAN will be the shared latent, and using domain generator and discriminator to generate domain image.

2.3 Natural Image Property

Natural image exist some property (statistics), for example, spatial correlation, that natural image follow $1/f^2$ power spectrum. Another natural image property is dark channel prior[7], which indicate that for natural images of outdoor, there are some local area will have at least one channel have very small intensity. In this work, we use dark channel prior to design a loss function that try to imitate adversarial patch as natural image.

3 Methodology

3.1 Beautify Adversarial Patch

After finishing training adversarial patch, we notice that the patch is too weird to human eyes. So we decide to beautify them with the following methods:

3.1.1 Style Transfer

In the beginning (before presentation), we naively applying style transfer to patch and found it is too destroying. We could not even tell any feature of target class in the transferred patch. Hence, we try to fuse the original patch and the transferred patch by weight sum, making the result patch still contain the important feature of target class and be painted in different style at the same time. Just like before, we treat it as a step of EOT. We apply style transfer after random location and rotation, before attacking on patch.

3.1.2 Add Another Constraint

To achieve the goal of beautifying patch, we further take experiment with another method. Contrary to make patch look like a artwork, we add another natural image statistics as constraint to patch to make it look naturally. we further add three losses, which are dark channel prior loss, total variation loss.

Dark channel prior loss To imitate natural image, we adopt dark channel prior as loss constraint, which formula as follow:

$$L_{dcp} = |\sum_{h,w \in H,W} min_{n \in R,G,B}(patch_{n,h,w})| \tag{1}$$

where H, W is hight, and width of patch.

Total variation loss We discover that smooth image make human feel visual pleasing , so we adopt total variation loss to smooth the patch, which formula as follow:

$$L_tv = 1/N \sum_{n=1}^{N} \sum_{c \in \theta} (| \bigtriangledown_x P_n^c| + | \bigtriangledown_y P_n^c|)^2, \theta = R, G, B \tag{2}$$

where N is iteration, P is adversarial patch, and $\bigtriangledown_x$ and $\bigtriangledown_y$ are the horizontal and vertical gradient operations.

then the total loss is formula as:

$$L_total = L_{logsoftmax} + W_\alpha L_{dcp} + W_\beta L_{tv} \tag{3}$$

where W_α and W_β are weight of additional loss.

3.2 Generating Adversarial Patch with Image to Image Translation

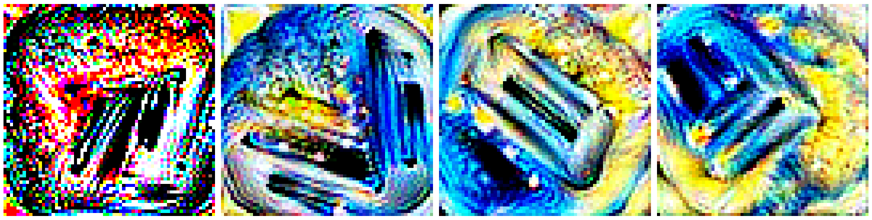
We simply set benign image as domain A and image with adversarial patch as domain B, then process image-to-image translation with UNIT model.

4 Experiments

4.1 Style Transfer

Here we choose *The Starry Night* as our style, because it contains only yellow and blue two colors basically, and we assume the lower number of colors, the more nature to human eyes. Besides, it is a famous painting, comparing to original adversarial patch, somehow human will not see its style as weird pattern but a kind of art.

Style Weight Comparison We try different style weight to fuse and see the change in Figure 1. Conditionally on not too large style weight, the patch tends to reduce weird multi-color pattern as the weight increases. On the other hand, the accuracy of attacking the image as target class would decrease. The test accuracy comparison is shown in Figure 2.



(a) 0% (b) 0.1% (c) 1% (d) 10%

Figure 1: Style transferred patch with different style weights.

Training Epoch and Attack Comparison Consider the difference of patch between training epochs, the comparison is shown in Figure 3. We take the style patches with weight 0.1% for example. The red pattern that commonly seen in original adversarial patch (Figure 1-(a)) still exists in the front of epochs, but it fades away as more training epochs. More specifically, change to the color of style. We then take the style patches with weight 1% for example in Figure 4. It reveals the change of patch

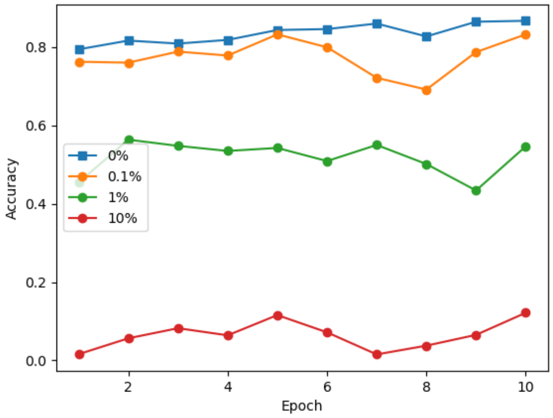


Figure 2: Test accuracy of attacking the image as target class with different style weights.

after attacking only one example. There is an obvious difference in the left bottom of the patch in the first epoch, and there are not perceptible variation in other epochs. Horizontally observing, the trend is just as the same as the Figure 3. Moreover, the perturbation of feature is more visible than the change in Figure 3 for more style weight.

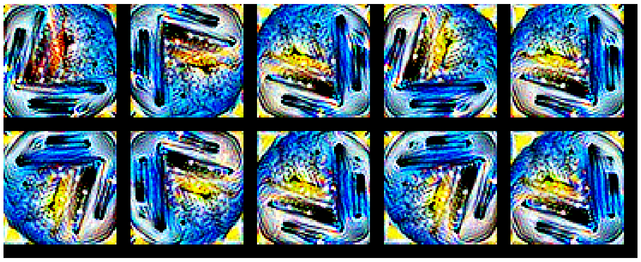


Figure 3: Comparison between training epochs. Row 1 are epoch 1 to 5, and row 2 are 6 to 10.

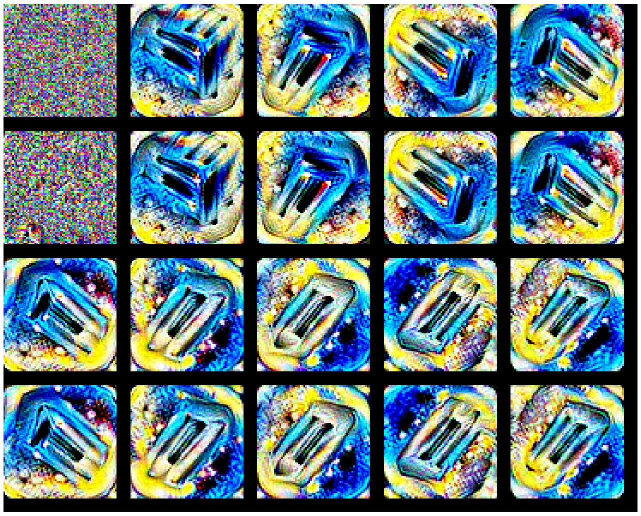
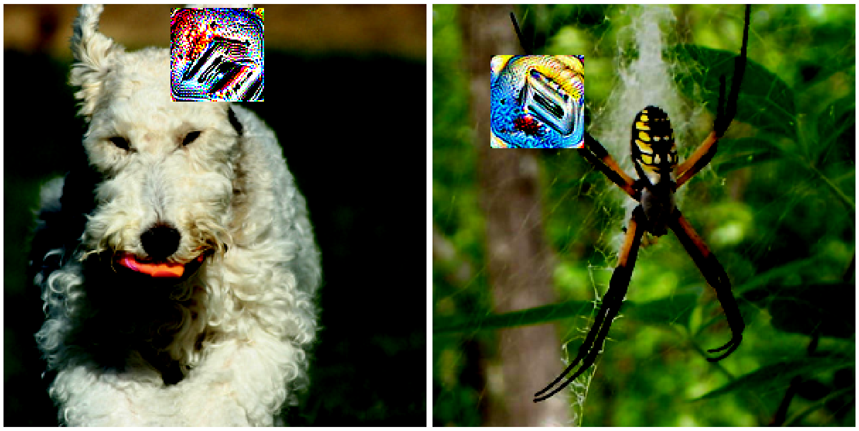


Figure 4: Comparison before and after attacking one example. Row 1 and row 3 are patches before attacking, row 2 and row 4 are results after attacking.

Successful Example Comparison The experiments above are trained and tested both in size of 2,000. We then train the patch in size of 40,000 and test in size of 10,000 for only two epochs, and the outputs are shown in Figure 5. The test accuracy of original adversarial patch is 96.6%, and the test accuracy of style transferred patch is 90.9%, in which the style weight is set to 0.1%.

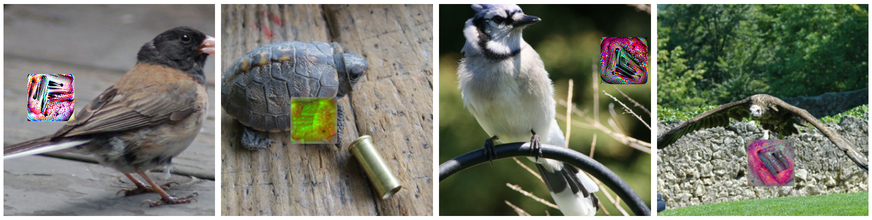


(a) With original adversarial patch (b) With style transferred patch

Figure 5: Successful examples: image with original adversarial patch and image with style transferred patch.

4.2 Add Another Constraint

The experiment result and ablation study for additional two loss constraint are shown in Fig 6. As the result shows, adversarial generated with total loss is closed to natural color and smooth; adversarial patch without total variation loss are very sharp and adversarial patch without dark channel prior constraint is present a weird color, and we discover that along with more additional constraint, top1 confidence of target drop down while evaluation.



(a) w/o L_{dc} and L_{tv} (b) L_{total} (c) w/o L_{tv} (d) w/o L_{dc}

Figure 6: ablation study of additional loss constraint

4.3 Generating Adversarial Patch with Image to Image Translation

The experiment result is shown in Fig 7, the result show that no matter from domain A to domain B or domain B to domain A, both of them are fail, we further use FID(Frechet Inception Distance) to estimate if these two domain are different distribution, and the result shows that they really are. We further refer these task to image inpainting task, although the author of [5] indicate UNIT can solve the type of problem, but we found UNIT is not suitable for block inpainting task, yet suitable for noise inpainting task. Thus we suppose this is the main reason why the experiment fail.

However, we found that even if the model have no capability to translate image through these two domain, but after translate image with patch to benign data domain, the result become benign data, and can use to adversarial patch defense, although the contour of adversarial patch do not erase.

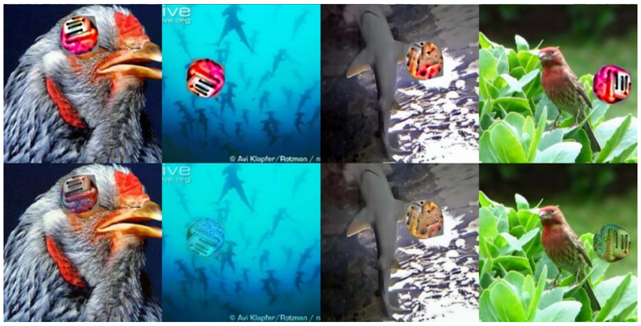


Figure 7: image to image translation result, row 1 is from adversarial patch image domain, and row 2 is translation result, which belongs benign image domain.

4.4 Optimize and compare

We found that the implementation of [1] calculate the gradient of whole image, instead of only patch part, so we re-implement it but found that only calculate the gradient of patch took long time to generate adversarial patch , and not as powerful as implementation of [1], which one possible reason is we only run one epoch for re-implementation version, since it took so much time, instead original version run 10 epoch.

We also try not to normalize the input image to [-1 ,1], and found that it is worse than normalization version, which is implementation version of [1].

5 Conclusion and Future Work

In this work, we try to beautify adversarial patch and using a novel way to generate adversarial patch, and we found although image-to-image translation cannot use to generate adversarial patch, but can use for adversarial patch image defense. Adversarial patch can use handcraft loss function and style transfer to constraint and beautify. we will try to explain why our re-implement version of [1] seems not as powerful as original implementation version as our future work.

References

[1] jhayes14, "adversarial-patch", In: *https://github.com/jhayes14/adversarial-patch*, 2018

[2] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, Justin Gilmer, "Adversarial Patch", In: *Conference on Neural Information Processing Systems(NIPS) Workshop*, 2017

[3] Johnson, Justin and Alahi, Alexandre and Fei-Fei, Li, "Perceptual losses for real-time style transfer and super-resolution", In:*European Conference on Computer Vision(ECCV)*, 2016

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", In:*IEEE International Conference on Computer Vision (ICCV)*, 2017

[5] Liu, Ming-Yu and Breuel, Thomas and Kautz, Jan, "Unsupervised Image-to-Image Translation Network", In:*Conference on Neural Information Processing Systems(NIPS)*, 2017

[6] Liu, Ming-Yu and Tuzel, Oncel, "Coupled Generative Adversarial Networks", In: *Advances in Neural Information Processing Systems (NIPS)*, 2016

[7] K. He, J. Sun and X. Tang, "Single Image Haze Removal Using Dark Channel Prior," In:*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[8]R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin and Y. Yang, "Adversarial Camouflage: Hiding Physical-World Attacks With Natural Styles," In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020