# TransPatch: A Transformer-based Generator for Accelerating Transferable Patch Generation in Adversarial Attacks Against Object Detection Models

Jinghao Wang[1†], Chenling Cui[1†], Xuejun Wen[2], and Jie Shi[2]

[1] School of Computer Science and Engineering, Nanyang Technological University
{C190209,CUIC0005}@e.ntu.edu.sg
[2] Singapore Digital Trust Lab, Singapore Research Centre, Huawei
{wen.xuejun,shi.jie1}@huawei.com

**Abstract.** Patch-based adversarial attack shows the possibility to black-box physical attacks on state-of-the-art object detection models through hiding the occurrence of the objects, which causes a high risk in automated security system relying on such model. However, most prior works mainly focus on the attack performance but rarely pay attention to the training speed due to pixel updating and non-smoothing loss function in the training process. To overcome this limitation, we propose a simple but novel training pipeline called **TransPatch**, a transformer-based generator with new loss function, to accelerate the training process. To address the issue of unstable training problem of previous methods, we also compare and visualize the landscape of various loss functions. We conduct comprehensive experiments on two pedestrian and one stop sign datasets on three object detection models, i.e., YOLOv4, DETR and SSD to compare the training speed and patch performance in such adversarial attacks. From our experiments, our method outperforms previous methods within the first few epochs, and achieves absolute 20% $\sim$ 30% improvements in attack success rate (ASR) using 10% of the training time. We hope our approach can motivate future research on using generator in physical adversarial attack generation on other tasks and models.

**Keywords:** adversarial attack, object detection, transformer

## 1 Introduction

Deep learning based object detection models [60] have been widely applied in automated surveillance cameras [21,47,23]. However, it has been proven that object detection models are very fragile to the adversarial attack. Digital adversarial attack such as perturbation [39,56], and digital footprint [33] could influence the object detection models in targeted or untargeted manners. Object detection

[†] This work was done during an internship at HUAWEI.

models will mis-classify or neglect some objects on the image that are manipulated by the adversarial pixels. With **digital adversarial attack** being largely unrealistic in automated surveillance camera scenes as it requires the attacker to have direct access to the inner camera system [42,22], **physical adversarial attack** directly on human body is more practical. Such attack requires the generation of adversarial patches held by the target person or printed on the clothes to fool the detector. It is therefore a more challenging task as the patch needs to be robust in various physical circumstances such as different camera angles and distances, distortion of color due to scene light and multiple possible noises.

Most of the previous works have two shortfalls: slow training process and relatively low attack success rate (ASR). Slow training process is common as most of the prior works involve directly updating of the pixels of patch from backpropagation. If we consider the pixel updating as a *zero-layer neural network*, we can see that its optimization is not efficient and its expression power is limited.

In this paper, we propose a novel adversarial attack patch generation pipeline denoted as *TransPatch*. Inspired by the recent progress in natural image synthesis by GAN-based deep generative model, we replace the pixel updating methods with a transformer-based generator and consider our target object detection model as a non-training 'discriminator'. Our generator relies on the loss function given by the target model to improve its ability. The design of loss function is also vital in the generation of the adversarial patches. Instead of using the maximum box objectness or class probability as the loss function, we propose a smoother version of the function and our subsequent visualization shows that our loss function makes gradient decent process easier. Extensive experimental results have shown the advantages of our approach in terms of faster convergence speed and higher attack success rate. Quantitative analysis on different physical scenario also proves the robustness of our patch. Overall, our contributions are summarized as follows:

1. We use a transformer-based generator instead of directly updating the patch to accelerate the process of patch generation.
2. We improve the ASR of adversarial hiding attack on three datasets including person and stop sign class and three target object detection models.

## 2 Related Work

### 2.1 Adversarial Attack on Object Detection

*Object detection.* Object detection model is an important component in the autonomous driving and security system. It locates the presence of objects with a bounding box and types or classes of the located objects in an image. Starting from 2-staged object detector models, object detection has gained much popularity in computer vision. 2-staged object detectors such as R-CNN [12] and Faster R-CNN [46] generate the region proposals in an image that could belong to a particular object. These particular regions are then classified into specific

object classes based on various pre-trained domains. Such models have achieved promising detection precisions but have also shown that the detection speed is relatively slow due to its 2-stage nature. One-stage object detection models, on the other hand, predict the presence and the class scores of the detected objects directly with a single pass. Well-known architectures such as YOLO [44,45,1], SSD [32], and DETR [3] are capable of high precision detection with faster speed as compared to 2-staged object detectors.

### 2.2 Adversarial attacks

Adversarial attack examples researches started from white-box based digital attack. Such attacks (i.e. FGSM [14], PGD [35]) assume the adversary has access to both the target model's gradient and pixel information of the input image. However, this assumption may not be applicable in real situations. In black-box based digital attack, attacker constantly inferences the model and uses the predicted results to update their pixel-level perturbations [39,56,8,9,54,62,52]. It has proven in [33,28] that, it is possible to attack the model without manipulating every pixel of the input image. Instead, a 40-by-40 pixel digital footprint is added at some specific location of the image to attack the model. Although such attack has less constraints, the only way to manipulate the pixels of the image captured by security system is through having direct access to the camera, which is largely difficult. Researchers have moved their interest to physical adversarial attacks which only allow direct physical changes to the environment or the target objects. By pasting adversarial patches onto physical stop signs, such attacks can achieve decently high ASR on stop sign detections [11,49,4]. Later works continued their interests to the more challenging person class hiding attacks [57,18,50,55,53]. Although person class has a higher in-class variation such as appearance, clothes and gesture, several successful researches [57,19,50,53,55] have proven the possibility to mislead the object detection by a patch pasted or printed on the t-shirt of the person.

### 2.3 Generative Models in Adversarial Attack

Generative Adversarial Network (GAN) is widely used in many generative tasks such as image generation [43,25,2,24], text generation [15], music generation [7] and image-to-image transfer [61]. Several works are inspired to use GAN in generating the adversarial patch. In [18], a pre-trained GAN is used and the loss given by the target model is only for updating and searching a specific latent vector input which makes the GAN generate a naturalistic adversarial patch to attack on person class. In [31,27], the GAN in the pipeline is trainable and designed to generate patch to attack road sign class on autonomous driving system. However, the discriminator proposed in these researches is used to make the patch-pasted stop sign as real as possible, which is not significantly applicable to person objects as a pasted patch naturally will not make the person 'unreal'. Therefore, this constraint is erased in our design of pipeline.

## 3 Methodology

We propose a novel Transformer-based generator pipeline, i.e., TransPatch, to accelerate the training and improve the ASR for the adversarial hiding attack on object detection model. In this section, we will first introduce the overall pipeline, and we will focus on two key components: the generator and the loss function.

### 3.1 Overall Pipeline

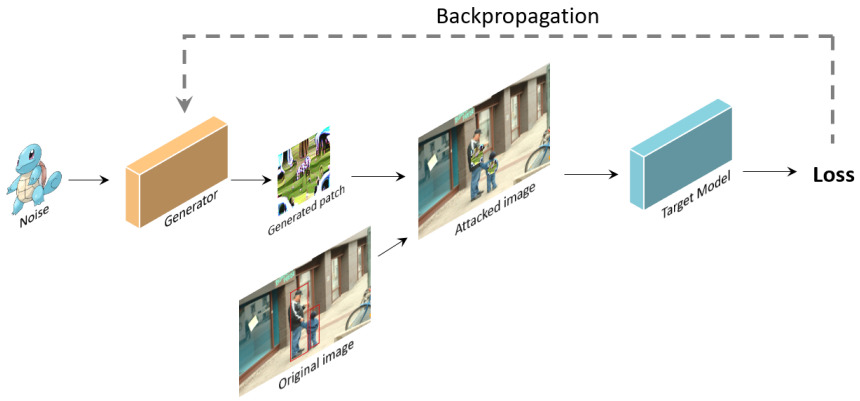



**Fig. 1. Overview of the training pipeline.**

*Components.* There are 5 components in the pipeline: input noise, generator, patch applier, target model and loss function. The generator takes a square noise $X \in [0,1]^{C\times L\times L}$ as input and output a same-size adversarial patch $P \in [0,1]^{C\times L\times L}$. We are not using any kinds of random noise such as Gaussian distribution or Bernoulli distribution, instead, we use images from pokemon dataset with random rotation as data augmentation as the input noise. The functionality of patch applier is to paste the generated patch correctly on the pedestrians with various sizes of bounding boxes from the original image. We resize the patch $P$ to $L' = f \times \sqrt{wh}$ ,where $f$ is a factor to control the size of the pasted patch and $w$ and $h$ represent the width and height respectively of the person bounding box. $f$ is a hyper-parameter and we fix it as 0.3 in all our experiments. The attacked image will be fed into target model. The object detection results (here we only focus on the person class) from the target model is used as the loss function for updating our generator.

### 3.2 Generator

*Architecture.* The generator takes input noise $X$ and output patch $P$ both with dimension $3 \times L \times L$. As shown in the left Stop Sign of Fig. 2, it contains a down-sampling block at the beginning which uses convolutional filters to transform the input noise to the feature map with $192 \times \frac{L}{4} \times \frac{L}{4}$. 5 similar transformer blocks are consequential to the down-sampling block which only change the number of channels innerly as $192 \rightarrow 384 \rightarrow 768 \rightarrow 768 \rightarrow 384 \rightarrow 192$ but do not change the size of the feature map. Finally, the feature map will be up-sampled by several transpose convolutional filters to the original size $3 \times L \times L$. Both up-sampling and down-sampling blocks are CNN-based and we apply spectral normalization [38] and batch normalization [20] for stable training.

*Transformer block.* As shown in the right side of Fig. 2, the number of channels will be adjusted by the point-wise convolutional filters kernel size 1. Following the discussion in [34] that fewer activation functions and fewer normalization layers are better, we directly apply the 25-head attention to the feature map following a layer normalization. We apply layer scale technique [51] in the residual connection where the output of the activation function will be multiplied by a learnable diagonal matrix where its diagonal value $\lambda$ will be initialized as $10^{-6}$. It is proven in our experiments that such technique can improve the stability of training. Following our experiments, we choose Mish [37] as the activation function since it has the best performance among ReLU [40], GELU [16] and Leaky ReLU of slope factor 0.01.

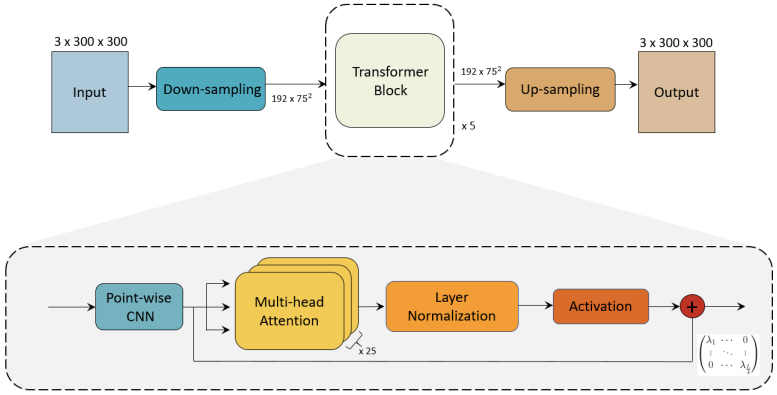



**Fig. 2. Generator architecture overview.** The top part shows that the generator is constructed by one down-sampling block, 7 transformer blocks and one up-sampling block. The bottom shows the residual connection and the forward flow inStop Sign the transformer block. All transformer blocks are identical to each other excluding the point-wise CNN block for channels adjustment.

### 3.3 Design of Loss Function

*Optimization objective.* We define our generating patch as $P$, the original image as $I$ and the process of applying patch to the correct location and size of pedestrians BB in the original image as $L(I, P)$. We further define our target model as $T$, the objectness and class probabilities of an arbitrary bounding box $B$ given by model $T$ as $T^B_{\mathbf{obj}}(x), T^B_{\mathbf{cls}}(x)$. We should notice that $T^B_{\mathbf{cls}}$ is a softmax output with dimension of number of classes (80 in our case).

For hiding person attack, we want our patch to minimize the objectness of bounding boxes whose person class has the highest class probability. Such conditional BB could be written as

$$\{B \mid \underset{\text{person}}{\mathbf{argmax}}\ T^B_{\mathbf{cls}}\} \tag{1}$$

So the objectness of these conditional BB, which we define it as $\mathbb{O}(x)$, could be written as

$$\mathbb{O}(x) = T_{\mathbf{obj}}^{\{B|\underset{\text{person}}{\mathbf{argmax}}\ T^B_{\mathbf{cls}}\}}(x) \tag{2}$$

Hence, our optimization objective is

$$\underset{P}{\mathbf{argmin}}\ \mathbb{O}(L(I, P)) \tag{3}$$

*Different loss functions.* Deriving from Eq. 3, we define our loss function $L_{\mathbf{atk}}$ as the mean of top-K of the objectness of these conditional BB, which is represented as *topK*:

$$L_{\mathbf{atk}} = \frac{1}{K}\sum^{K} \text{topK}(\mathbb{O}(x)) \tag{4}$$

where the value of $K$ is a hyper-parameter that is subjective to the size of batch, the average number of target object in one image of the dataset and the number of boxes in model's output. For DETR, we use the probability of its additional output class: no-object to calculate objectness. For SSD, we use the probability of background class to get the value of objectness.

Loss functions represented as *MaxObj* and *MaxCls* used by previous methods could be formulated respectively as:

$$L_{\mathbf{atk}} = \max(T^B_{\mathbf{obj}}(x)) \tag{5}$$

and

$$L_{\mathbf{atk}} = \max(T^B_{\mathbf{cls=person}}(x)) \tag{6}$$

We briefly summary the differences of the loss function in table 1.

It is clearly to see that our $L_{\mathbf{atk}}$ only focus on a subset of BB since it is intuitively to understand that influencing BB containing other classes with high confidence like the previous loss functions is very hard and not directly responds to our optimization objective Eq.3. And, we can see that ours is a smoother version to directly get the maximum of all BB. Quantitative analysis is conducted in the following ablation study.

**Table 1. Loss functions designs and explanations.** The same abbreviation will be used in the following sections

| Abbr. | Description |
|---|---|
| topK | Described as above. Formulated in Eq. 4. |
| MaxObj | The maximum of objectness of all bounding boxes. Formulated in Eq. 5. |
| MaxCls | The maximum of person class probability of all bounding boxes. Formulated in Eq. 6. |

## 4 Experiments

In this section, we evaluate our proposed TransPatch on pedestrian hiding attack on INRIA Person, CityPersons and Stop Sign datasets. We also show the effect of the transformer blocks and our new design of loss function to outperform the previous methods.

### 4.1 Datasets

*Different features.* The three pedestrians datasets we use have their own strong characteristic and reflect different scenarios of pedestrian detection, which could prove that our method has powerful adaptability. INRIA Person dataset [6] is captured by hand-holding camera with diversity in backgrounds such as square, mountain and grassland. CityPersons dataset [59] is a subset of Cityscapes [5] containing images of pedestrians recording by multiple driving recorders. Stop Sign dataset is generate from all images which is labeled as stop sign in COCO 2017 [30]. Fig. 3 shows example from three datasets and table 2 shows statics to numerically prove the above observation.

**Table 2. Quantitative analysis of different dataset predicted by YOLOv4.** Under the above nms and confidence setting, for each model (YOLOv4/DETR/SSD), there is a proportion of images with no target object could be detected and are considered as not valid image in our task. Such proportion could reflect the original performance of target models in our datasets

| Name | # Total image | # Valid image | # Avg object | Avg size of BB ($10^{-2}$) |
|---|---|---|---|---|
| INRIA Person | 902 | 902/898/823 | 4.24/6.14/1.92 | 6.43/4.80/10.40 |
| CityPersons | 5000 | 3363/2644/525 | 3.63/4.49/1.31 | 0.82/0.99/3.27 |
| Stop Sign | 1803 | 1408/1221/1114 | 1.12/1.09/1.03 | 10.47/11.18/13.82 |

*Labeling.* We randomly split the dataset into training and validation set in the ratio of 9 : 1 and all images are scaled to the same size before labeling. We use

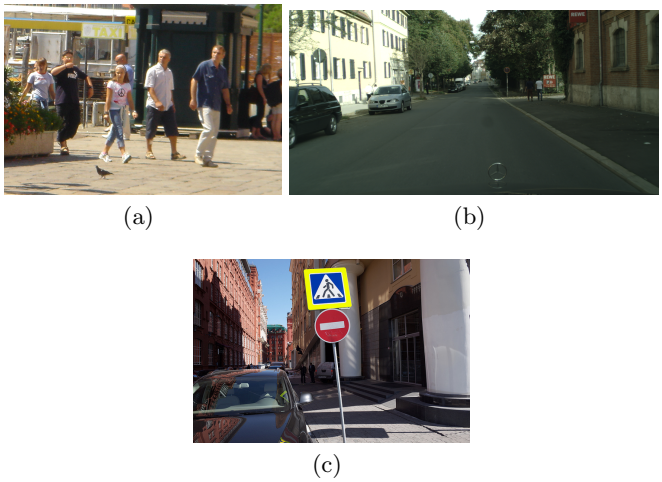


**Fig. 3.** Example from (a) INRIA Person (b) CityPersons (c) Stop Sign

the pre-trained target model to extract bounding boxes of all pedestrians in our dataset. We filter through all the boxes of person class using Non-Maximum Suppression (nms) [17] and keep only the boxes with output confidence (objectness $\times$ class probability) bigger than 0.5 and IoU threshold of 0.4. The reason why we use such label as the ground true labels instead of using the original annotation provided by the dataset is that our task is not evaluating the performance of the object detection model.

### 4.2 Adversarial Hiding Attack

*Evaluation metric.* In adversarial hiding attack, comparing with AP, ASR is a more suitable and more strict evaluation metric. We define the attack success rate (ASR) as the ratio of the number of hiding persons to the total number of persons in the images. We prove that, under the same circumstance, achieving high ASR is at least as hard as achieving low AP when we measure the adversarial attack result in Appendix A. The Convergence Time (CT) measures the training time cost. We multiply the number of epochs by the time taken of each epoch and use *hour* as the unit of results. We should notice that some methods will converge easily within a shorter time but with a relatively low ASR.

*Target models.* Three object detection models are chosen as target models. Their characteristics are summarized in Table 3.

*Comparison with baseline.* We compare TransPatch in both ASR and CT of all three datasets under all three target models with the baseline and the results are shown in Table 4. The baseline model is the combination of pixel updating and *MaxObj* loss function. It is shown that our method will take a longer time in

**Table 3. Comparison between different target models.** For fair comparison, the below statistics are based on the models used in our experiments. Release year is based on the year published on *arxir.org*. The source of pre-trained weights and Pytorch architecture implementation will be stated in Appendix B.

| Model | Release Year | # Parameter |
|---|---|---|
| YOLOv4 | 2020 | 64M |
| DETR | 2020 | 41M |
| SSD | 2015 | 22M |

each epoch due to its complexity in architecture; however, our methods uses significantly fewer number of epochs to converge, which makes our CT shorter than others. Also, our newly design of loss function also helps the training overcome the original local minima.

**Table 4. Adversarial attack results and training time consumption of INRIA Person, CityPersons and Stop Sign dataset on YOLOv4, DETR and SSD.** Due to the difference of target models and the lack of report of ASR, we train the baseline pixel-updating method [50] with tuning hyper-parameter settings adopting from their paper and official code. The random Gaussian patch and white patch are considered as trivial attack.

| Target Model | Method | INRIA | | CityPersons | | Stop Sign | |
|---|---|---|---|---|---|---|---|
| | | ASR↑ | CT(h)↓ | ASR↑ | CT(h)↓ | ASR↑ | CT(h)↓ |
| YOLOv4 | Random Gaussian | 5.4% | — | 22.1% | — | 9.7% | — |
| | White | 5.8% | — | 23.1% | — | 12.0% | — |
| | Baseline [50] | 43.8% | 6.4 | 24.4% | 37.0 | 25.8% | 10.8 |
| | **TransPatch (ours)** | **86.5%** | **0.7** | **68.2%** | **2.6** | **88.2%** | **0.1** |
| DETR | Random Gaussian | 24.7% | — | 43.4% | — | 33.3% | — |
| | White | 35.9% | — | 39.6% | — | 31.5% | — |
| | Baseline [50] | 37.9% | 6.4 | 62.5% | 16.5 | 84.6% | 8.7 |
| | **TransPatch (ours)** | **64.5%** | **0.8** | **92.7%** | **0.2** | **94.3%** | **0.1** |
| SSD | Random Gaussian | 10.7% | — | 41.2% | — | 10.4% | — |
| | White | 20.5% | — | 60.5% | — | 16.1% | — |
| | Baseline [50] | 64.3% | 3.2 | 20.3% | 1.1 | 27.3% | 1.9 |
| | **TransPatch (ours)** | **80.0%** | **0.2** | **84.4%** | **0.1** | **59.9%** | **0.5** |

### 4.3 Ablation Study

*Effect of transformer blocks.* In Table 5, we prove that the multi-head attention mechanism and the number of transformer blocks both have impact on the performance of adversarial attack by comparing with a CNN-based generator

and TransPatch with less number of transformer blocks. We fix *topK* as our loss function in this section.

*Effect of design of loss functions.* In Table 6, we prove that the design of loss functions has impact on the performance of adversarial attack. In some extreme cases with ill designed loss function, the training even does not converge. The difference of the computational cost is negligible between different loss functions.

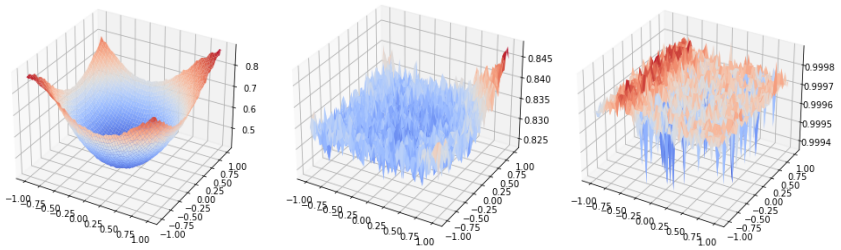
*Visualization of the smoothness of different loss functions.* To further illustrate the effect, we visualize the landscape of loss function to show that our design of loss function has a smoother gradient. Such method [29] first uniformly samples multiple steps $\alpha$ and $\beta$ given a closed range and gets two Gaussian random noise $\delta$ and $\eta$. Representing the weights of the network at epoch $k$ as $\theta_k$, the new value of the loss function $L_{\text{new}} = L(\theta_k + \alpha\delta + \beta\eta)$ will be given by the output of adjusted network (or adjusted patch in baseline model). The points of 3D loss landscape are formed by $(\alpha, \beta, L_{\text{new}})$. Due to the time constraint, we use 200 sample in the training set and 50 steps from $[-1, 1]$.

**Table 5. Adversarial attack results and number of parameters of different architectures of INRIA Person, CityPersons and Stop Sign on YOLOv4.** For *Void*, we only keep the up-sampling and down-sampling block, which could be considered as 0 transformer block. For *CNN*, in each residual-connection block, we connect 3 groups of convolutional filter with kernel size 3 and padding 1, batch normalization and activation function sequentially

| Architecture | # Parameters | ASR↑ | | |
|---|---|---|---|---|
| | | INRIA | CityPersons | Stop Sign |
| Void | 2M | 28.9% | 44.2% | 80.6% |
| 9 residual-connection CNN blocks | 8M | 43.7% | 44.8% | **88.9**% |
| 3 transformer blocks | 382M | 83.8% | 65.4% | 87.7% |
| **5 transformer blocks (ours)** | 636M | **86.5%** | **68.2%** | 88.2% |

**Table 6. ASR of of different loss functions on INRIA Person, CityPersons and Stop Sign datasets.** Our TransPatch is referred as *transfomer + topK* and the baseline method is referred as *pixel-update + MaxObj*.

| Target Model | Generation | Loss Function | ASR↑ | | |
|---|---|---|---|---|---|
| | | | INRIA | CityPersons | Stop Sign |
| YOLOv4 | pixel-update | MaxObj | 43.8% | 24.4% | **25.8%** |
| | pixel-update | MaxCls | 10.3% | 25.6% | 15.0% |
| | pixel-update | topK | **57.1%** | **65.7%** | 6.8% |
| | transformer | MaxObj | 49.6 % | 45.4% | 44.4% |
| | transformer | MaxCls | 14.4 % | 48.9% | 49.4% |
| | **transformer** | **topK** | **86.5%** | **68.2%** | **88.2%** |
| DETR | pixel-update | MaxObj | 37.9% | 62.5% | 84.6% |
| | pixel-update | MaxCls | 35.1% | 76.1% | 78.6% |
| | pixel-update | topK | **41.8%** | **81.1%** | **85.3%** |
| | transformer | MaxObj | 49.6 % | 79.9% | 70.6% |
| | transformer | MaxCls | 44.7 % | **93.2%** | **94.4%** |
| | **transformer** | **topK** | **64.5%** | 92.7% | 94.3% |
| SSD | pixel-update | MaxObj | 64.3% | 20.3% | 27.3% |
| | pixel-update | MaxCls | **87.8%** | 18.8% | 42.0% |
| | pixel-update | topK | 57.3% | **26.6%** | **61.0%** |
| | transformer | MaxObj | 76.1 % | 89.6% | **60.0%** |
| | transformer | MaxCls | 72.0 % | 79.7% | 57.2% |
| | **transformer** | **topK** | **80.0%** | **84.4%** | 59.9% |



(a) (b) (c)

**Fig. 4.** Loss function landscape of TransPatch at epoch 50 in training INRIA attacking YOLOv4 with (a) *topK* (b) *MaxObj* (c) *MaxCls*. Observing y-axis of the plots, *MaxObj* and *MaxCls* fluctuates intensively and the decrease in loss is insignificant. *topK*, on the other hand, ensures a smooth loss landscape with an obvious minima.

## 5 Physical Attack

In this section, we further quantitatively investigate our pipeline in the physical attack from two perspectives: additional constrains in the loss function to make our patch after printing become more robust and cycle consistency penalty to make our patch more naturalistic.

### 5.1 Addition Constrains

In order to perform such attack in a physical manner, we also need to consider the environmental influence to the patch. Two more objectives are introduced: the finite-difference approximation of total variants (**tv**) [36] which constrains the smoothness of the patch and non-printability score (nps) [48] which constrains the printability of the patch in our training.

$$L_{\mathbf{tv}} = \sum_{i,j} \sqrt{(p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2} \quad \text{and} \quad L_{\mathbf{nps}} = \sum_{p \in P} \min_{c \in C} \|\hat{p} - p\|$$

During training, we utilize all datasets we have to train the model by subsampling 1000 of them in each epoch. Fig. 5 shows the qualitative illustration.

(a) (b) (c) (d)

**Fig. 5. Qualitative results of physical attack.** The patch from our approach has the capability to attack the model on multiple circumstances such as different (a) distance, (b) view, (c) lighting condition and (d) posture.

## 6 Conclusion

In this paper, we propose a simple yet efficient pipeline, TransPatch, for physical adversarial attack on state-of-art object detection model. We have shown that the combination of the transformer-based generator and our loss function could significantly accelerate the training and improve the ASR as compared to previous methods. We experimentally validate that our TransPatch remains as

an effective method of generating adversarial patches in adversarial attacks on both human and stop signs in all three selected target object detection models.

This particular method comes with challenges as well regarding the need for computational power. We wish that future works can work on such improvements. We also believe that our result can be applied to attacks in automated surveillance camera scenarios and hope that our research could inspire more defense against such attacks in these circumstances.

## References


1. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: ICLR. OpenReview.net (2019)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (1). Lecture Notes in Computer Science, vol. 12346, pp. 213–229. Springer (2020)
4. Chen, S., Cornelius, C., Martin, J., Chau, D.H.P.: Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector. In: ECML/PKDD (1). Lecture Notes in Computer Science, vol. 11051, pp. 52–68. Springer (2018)
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. pp. 3213–3223. IEEE Computer Society (2016)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (1). pp. 886–893. IEEE Computer Society (2005)
7. Dong, H., Hsiao, W., Yang, L., Yang, Y.: Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In: AAAI. pp. 34–41. AAAI Press (2018)
8. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: CVPR. pp. 9185–9193. Computer Vision Foundation / IEEE Computer Society (2018)
9. Dong, Y., Pang, T., Su, H., Zhu, J.: Evading defenses to transferable adversarial examples by translation-invariant attacks. In: CVPR. pp. 4312–4321. Computer Vision Foundation / IEEE (2019)
10. Everingham, M., Eslami, S.M.A., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vis. **111**(1), 98–136 (2015)
11. Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., Song, D.: Robust physical-world attacks on machine learning models. CoRR **abs/1707.08945** (2017)
12. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. pp. 580–587. IEEE Computer Society (2014)
13. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. JMLR Proceedings, vol. 9, pp. 249–256. JMLR.org (2010)
14. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (Poster) (2015)
15. Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long text generation via adversarial training with leaked information. In: AAAI. pp. 5141–5148. AAAI Press (2018)
16. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
17. Hosang, J.H., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: CVPR. pp. 6469–6477. IEEE Computer Society (2017)
18. Hu, Y., Chen, J., Kung, B., Hua, K., Tan, D.S.: Naturalistic physical adversarial patch for object detectors. In: ICCV. pp. 7828–7837. IEEE (2021)

19. Huang, L., Gao, C., Zhou, Y., Xie, C., Yuille, A.L., Zou, C., Liu, N.: Universal physical camouflage attacks on object detectors. In: CVPR. pp. 717–726. Computer Vision Foundation / IEEE (2020)
20. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. JMLR Workshop and Conference Proceedings, vol. 37, pp. 448–456. JMLR.org (2015)
21. Javed, O., Shah, M.: Tracking and object classification for automated surveillance. In: ECCV (4). Lecture Notes in Computer Science, vol. 2353, pp. 343–357. Springer (2002)
22. Jeong, J., Kwon, S., Hong, M., Kwak, J., Shon, T.: Adversarial attack-based security vulnerability verification using deep learning library for multimedia video surveillance. Multim. Tools Appl. **79**(23-24), 16077–16091 (2020)
23. Kaliappan, M., Vimal, S., Vijayalakshmi, K., Lee, M.Y., Thangadurai, M.: OSDDY: embedded system-based object surveillance detection system with small drone using deep YOLO. EURASIP J. Image Video Process. **2021**(1), 19 (2021)
24. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: ICLR. OpenReview.net (2018)
25. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR. pp. 4401–4410. Computer Vision Foundation / IEEE (2019)
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (Poster) (2015)
27. Kong, Z., Guo, J., Li, A., Liu, C.: Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In: CVPR. pp. 14242–14251. Computer Vision Foundation / IEEE (2020)
28. Lee, M., Kolter, J.Z.: On physical adversarial patches for object detection. CoRR **abs/1906.11897** (2019)
29. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: NeurIPS. pp. 6391–6401 (2018)
30. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV (5). Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer (2014)
31. Liu, A., Liu, X., Fan, J., Ma, Y., Zhang, A., Xie, H., Tao, D.: Perceptual-sensitive GAN for generating adversarial patches. In: AAAI. pp. 1028–1035. AAAI Press (2019)
32. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. In: ECCV (1). Lecture Notes in Computer Science, vol. 9905, pp. 21–37. Springer (2016)
33. Liu, X., Yang, H., Liu, Z., Song, L., Chen, Y., Li, H.: DPATCH: an adversarial patch attack on object detectors. In: SafeAI@AAAI. CEUR Workshop Proceedings, vol. 2301. CEUR-WS.org (2019)
34. Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. CoRR **abs/2201.03545** (2022)
35. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (Poster). OpenReview.net (2018)
36. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR. pp. 5188–5196. IEEE Computer Society (2015)
37. Misra, D.: Mish: A self regularized non-monotonic activation function. In: BMVC. BMVA Press (2020)
38. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: ICLR. OpenReview.net (2018)

39. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: CVPR. pp. 86–94. IEEE Computer Society (2017)
40. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. pp. 807–814. Omnipress (2010)
41. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E.Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS. pp. 8024–8035 (2019)
42. Payne, B.R.: Car hacking: Accessing and exploiting the can bus protocol. Journal of Cybersecurity Education, Research and Practice **2019**(1), 5 (2019)
43. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR (Poster) (2016)
44. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
45. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7263–7271 (2017)
46. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
47. S., B.P., George, S.N.: An automated unified framework for video deraining and simultaneous moving object detection in surveillance environments. IEEE Access **8**, 128961–128972 (2020)
48. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: CCS. pp. 1528–1540. ACM (2016)
49. Song, D., Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Tramèr, F., Prakash, A., Kohno, T.: Physical adversarial examples for object detectors. In: WOOT @ USENIX Security Symposium. USENIX Association (2018)
50. Thys, S., Ranst, W.V., Goedemé, T.: Fooling automated surveillance cameras: Adversarial patches to attack person detection. In: CVPR Workshops. pp. 49–55. Computer Vision Foundation / IEEE (2019)
51. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: ICCV. pp. 32–42. IEEE (2021)
52. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I.J., Boneh, D., McDaniel, P.D.: Ensemble adversarial training: Attacks and defenses. In: ICLR (Poster). OpenReview.net (2018)
53. Wang, Y., Lv, H., Kuang, X., Zhao, G., Tan, Y., Zhang, Q., Hu, J.: Towards a physical-world adversarial patch for blinding object detection models. Inf. Sci. **556**, 459–471 (2021)
54. Wu, D., Wang, Y., Xia, S., Bailey, J., Ma, X.: Skip connections matter: On the transferability of adversarial examples generated with resnets. In: ICLR. OpenReview.net (2020)
55. Wu, Z., Lim, S., Davis, L.S., Goldstein, T.: Making an invisibility cloak: Real world adversarial attacks on object detectors. In: ECCV (4). Lecture Notes in Computer Science, vol. 12349, pp. 1–17. Springer (2020)
56. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.L.: Adversarial examples for semantic segmentation and object detection. In: ICCV. pp. 1378–1387. IEEE Computer Society (2017)

57. Xu, K., Zhang, G., Liu, S., Fan, Q., Sun, M., Chen, H., Chen, P., Wang, Y., Lin, X.: Adversarial t-shirt! evading person detectors in a physical world. In: ECCV (5). Lecture Notes in Computer Science, vol. 12350, pp. 665–681. Springer (2020)
58. Zhang, J., He, T., Sra, S., Jadbabaie, A.: Why gradient clipping accelerates training: A theoretical justification for adaptivity. In: ICLR. OpenReview.net (2020)
59. Zhang, S., Benenson, R., Schiele, B.: Citypersons: A diverse dataset for pedestrian detection. In: CVPR. pp. 4457–4465. IEEE Computer Society (2017)
60. Zhao, Z.Q., Zheng, P., Xu, S.t., Wu, X.: Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems **30**(11), 3212–3232 (2019)
61. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV. pp. 2242–2251. IEEE Computer Society (2017)
62. Zou, J., Pan, Z., Qiu, J., Liu, X., Rui, T., Li, W.: Improving the transferability of adversarial examples with resized-diverse-inputs, diversity-ensemble and region fitting. In: ECCV (22). Lecture Notes in Computer Science, vol. 12367, pp. 563–579. Springer (2020)

## A Comparison between AP and ASR

In this section, we will prove that ASR is a more strict metric than AP for illustration of the effects of adversarial attacks. Most of the prior works demonstrate how their attack lower mean average precision (if only person class is attacked, AP@0.5 is a common index to be reported) of the target model as it is consistent with the evaluation method in SOTA object detection benchmark datasets such as MS COCO [30] and PASCAL VOC [10]. The AP@0.5 is defined as area under the precision-recall curve where the true positive is defined as IoU $> 0.5$ between predicted BB and ground truth BB. Lower AP@0.5 implies more successful adversarial attack.

Reporting ASR is equivalent to reporting AP@0. AP@0 means that, the predicted BB having zero IoU with the ground truth BB will still be considered as a true positive prediction. In ASR, we consider the attack unsuccessful once the detector has detected the existence of person regardless of the ground truth label. ASR is the 1 - recall of AP@0 and lower AP@0 implies higher ASR.

It is therefore proven that AP@0 is at least more strict than AP@0.5.

$$\text{Recall} = \frac{\text{TP}}{\#\text{ object}} \quad \text{and} \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

We can retrieve from the definition that # person and FP are the same in AP@0 and AP@0.5 while $\text{TP}_{\text{AP@0}} \leq \text{TP}_{\text{AP@0.5}}$, which causes the recall and precision to be larger. It means that, given the same adversarial patch and the same target model, AP@0 $\leq$ AP@0.5 is always valid. Hence, we conclude that ASR is at least more strict than AP@0.5. We illustrate one example in Table 7.

**Table 7. ASR and AP@50 of INRIA Person, CityPersons and Stop Sign datasets on YOLOv4**

| Target Model | Method | INRIA | | CityPersons | | Stop Sign | |
|---|---|---|---|---|---|---|---|
| | | ASR↑ | AP@0.5↓ | ASR↑ | AP@0.5↓ | ASR↑ | AP@0.5↓ |
| YOLOv4 | Gaussian | 5.4% | 90.8% | 22.1% | 74.6% | 9.7% | 88.3% |
| | White | 5.8% | 84.7% | 23.1% | 74.6% | 12.0% | 85.2% |
| | Baseline | 43.8% | 52.9% | 24.4% | 77.4% | 25.8% | 74.5% |
| | TransPatch | 86.5% | 12.2% | 68.2% | 28.8% | 88.2% | 9.7% |

# B Hyper-parameter Details

## B.1 Training Circumstance

All experiments are conducted on a system with a single NVIDIA Tesla V100-PCIE-32GB with CUDA version 460 and dual Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz. We use Python 3.9.10 and Pytorch 1.11 [41]. We use automatic mixed precision training to save the memory consumption and utilize the Volta architecture. We dynamically adjust the brightness and contrast of the generated patch before applying it to the image to improve the robustness of the generation process.

## B.2 Target Models

- YOLOv4: We adopt its Pytorch architecture from Tianxiaomo's GitHub repository and download weights and configuration file from Alexey darknet.
- DETR: We adopt its Pytorch architecture from Facebook Research's Github repository and download weights file from Facebook model hub.
- SSD: We adopt its Pytorch architecture and weight from Pytorch Model Hub by Nvidia.

## B.3 Training Hyper-parameter

In training TransPatch, we use $1 \times 10^{-4}$ as the learning rate and use adam optimizer [26]. We use 1 as batch size for the generator and 16 for the target model. We initialize both the out and in projection metrics of the transformer using xavier [13] with gain 0.5. We implement no learning rate decay. Gradient clipping over 2 is also applied in the training to prevent exploding gradient problem [58]. In training the baseline model, we use the patience learning rate decay patience = 50.

## C Examples of Generated Patch

We illustrate several our result patches attacking YOLOv4.

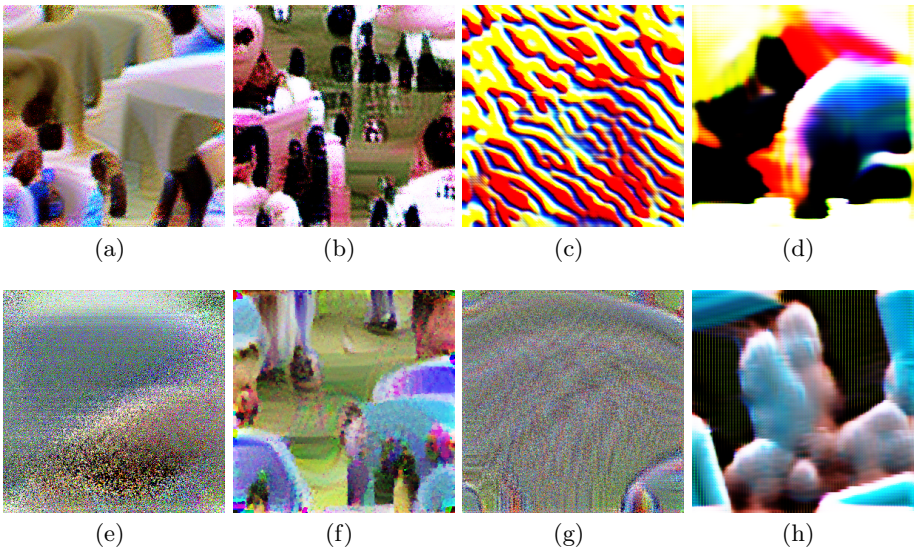



**Fig. 6. Examples of Generated Patch.** (a), (b) TransPatch training in INRIA with *topK* and *MaxObj*. (c) TransPatch training in Stop Sign with *topK*. (d) CNN training in INRIA with *topK*. (e) Pixel update training in CityPersons with *topK*. (f), (g) Pixel update training in INRIA with *topK* and *MaxObj*. (h) Void training in INRIA with *topK*.