Multiple Remote Adversarial Patches: Generating Patches based on Diffusion Models for Object Detection using CNNs

Kento Oonishi
Mitsubishi Electric Corporation
Onishi.Kento@ap.MitsubishiElectric.co.jp

Tsunato Nakai
Mitsubishi Electric Corporation
Nakai.Tsunato@dy.MitsubishiElectric.co.jp

Daisuke Suzuki
Mitsubishi Electric Corporation
Suzuki.Daisuke@bx.MitsubishiElectric.co.jp

Abstract

Adversarial patches can fool object detection systems, which poses a severe threat to machine learning models. Many researchers have focused on strong adversarial patches. Remote adversarial patches, placed outside the target objects, are candidates of strong adversarial patches. This study gives a concrete model of adversarial patches on convolutional neural networks (CNNs), namely diffusion model. Our diffusion model shows that multiple remote adversarial patches pose severe threats on YOLOv2 CNN. Our experiment also demonstrates that two remote adversarial patches reduce the average existence probability to 12.81%, whereas Saha et al.'s original single adversarial patch reduced the average existence probability to 50.95%. Moreover, we generate adversarial patches on SSD architecture. In SSD architecture, two remote adversarial patches also significantly reduce the average existence probability from 24.52% to 6.12%. By the above results, this paper provides a framework for analyzing the effect of adversarial patch attacks.

1 Introduction

1.1 Background

Object detection using a convolutional neural network (CNN) has been developed but suffers from severe threats posed by adversarial examples. Adversarial patches [2], an adversarial example type, have attracted attention in academia because they are physically feasible. Attackers place adversarial patches in an object detection system's field of view. For example, Thys et al.'s adversarial patch can effectively hide a person from an object detection system [19]. Many researchers have focused on generating strong adversarial patches [4, 6, 7, 9, 12, 15, 16, 22]. In parallel, many researchres proposed certified [10, 13, 20, 21] and empirical [5, 8] defenses against adversarial patches.

In proposed adversarial patches, remote adversarial patches [9, 12, 16] are threats to the real world. While remote adversarial patches are placed far from target objects, they can make an object detection system misrecognize the target objects. These target objects are not intended to be hidden, which is different from a case presented in Thys et al.'s research. Therefore, we must estimate threats of remote adversarial patches to protect target objects from misrecognition in object detection systems.

However it is widely known that remote adversarial patches are severe threats for CNNs, there are limited studies evaluating threats of remote adversarial patches. Different from an adversarial patch on target objects, it is unnatural that remote adversarial patches affect the target objects. Despite this unnaturalness, previous research only focuses on generating strong remote adversarial patches and does not focus on evaluating threats of remote adversarial patches. Therefore, we should consider how to evaluate threats of remote adversarial patches.

1.2 Our Contributions

This study proposes a novel model, a so-called diffusion model, for analyzing the effect of remote adversarial patches. In previous research, Araujo et al. [1] showed the calculation method of receptive fields of CNNs. Similar as this method, we propose a diffusion model in each convolutional layers.

This study aims to find a better remote adversarial patch layout using our diffusion model. During our estimation and experiments, we used PASCAL VOC dataset [3]. First, we calculated the effect of remote adversarial patches using our diffusion model on YOLOv2 [14] as an example. We then prepared single and multiple patch models under the total area of adversarial patch are almost the same. Then, our diffusion model demonstrates that multiple remote adversarial patches have a stronger effect than a single one. Next, we generated remote adversarial patches based on Saha et al's implementation [16]. Subsequently, we verified that a range of adversarial patch effect is almost the same with an estimated range by our diffusion model. Our multiple remote adversarial patches dramatically decrease existence probabilities. In our experiment, two remote adversarial patches reduce average existence probability to 12.81%, while Saha et al.'s single adversarial patch reduces average existence probability to 50.95%. We also generated multiple remote adversarial patches on SSD architecture [11]. In SSD architecture, two remote adversarial patches also significantly reduce the average existence probability from 24.52% to 6.12%.

2 Previous Adversarial Patches

Many researchers have proposed adversarial patch attacks since Brown et al.'s original work [2]. These patches have one of the following features, object hiding [19] or object misrecognition [12, 16]. Object hiding prevents targets from being detected. Different from object hiding, object misrecognition prevents targets from being detected correctly. Dpatch [12] and Saha et al.'s adversarial patch [16] are leading studies of placing a remote adversarial patch. These attacks virtually place a remote adversarial patch, whereas Lee and Kolter [9] physically realize a remote adversarial patch attack.

While the abovementioned attacks consider a single adversarial patch, researchers have been placing multiple adversarial patches on their images as well. For example, Zhu et al. [22] proposed a method for dynamically searching optimized locations of adversarial patches. This optimized location refers to the entire image; it is not limited to target objects' surroundings. This attack aims to minimize the area of adversarial patches. In addition, Huang et al. [7] proposed an attack finding key-pixels. However, this dynamical search is not physically realizable because the layout of adversarial patches depends on the input image.

Different from these attacks, Rossolini et al. [15] proposed multiple fixed adversarial patches for semantic segmentation models. Their experiment specifically demonstrated that multiple adversarial patches have a stronger effect than a single adversarial patch with approximately the same area. These adversarial patches expand a fooled area from themselves. Subsequently, the effect of multiple fixed remote adversarial patches on object detection systems must be assessed.

Based on the above discussion, many researchers have proposed strong adversarial patches. However, there are limited studies evaluating threats of remote adversarial patches. This study then aims to clarify how to evaluate threats of remote adversarial patches.

3 Our Core Model: Diffusion of Adversarial Patch Effect

This section presents our core model for adversarial patches. Our model demonstrates the effect of adversarial patches by calculating their diffusion. This study discusses our diffusion model on YOLOv2 as an example; however, but our diffusion model will be applicable to other CNNs. Based on our diffusion model, we show that multiple remote adversarial patches cause stronger an effect on YOLOv2's CNN compared with a single remote patch.

To evaluate the effect of remote adversarial patches, we introduce a coordinate system, such that (x, y) $(0 \leq x \leq 415, 0 \leq y \leq 415)$, on images. $(x, y) = (0, 0)$ represents the top left corner. x-coordinate increases from the left to the right cells, and the y-coordinate increases from the upper to the lower cells. Then, we make assumptions on adversarial patches. First, the size of adversarial patches remains approximately 10,000 pixels for comparison with previous results [16]. Next, we place adversarial patches over the range shown in Figure 1 to consider remote adversarial patches. Specifically, the x-coordinate between $0 \leq x \leq 104$ or $311 \leq x \leq 415$, while y-coordinate varies between $0 \leq y \leq 415$. Finally, we place adversarial patches in pre-fixed locations to consider physical attacks as well.

Next, we explain how diffusion occurs. The YOLOv2 network includes a convolutional layer with a 3×3 filter size. This convolutional layer calculates a cell's renewal value using nine cells, namely, itself and the surrounding eight cells, as shown in Figure 2. Adversarial patch information in these nine cells then spreads to adjoining cells. The detailed calculation method is shown in Appendix B.

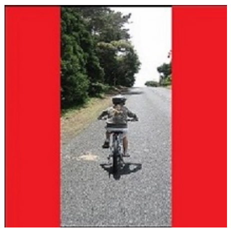


Figure 1: Placement range of adversarial patches. We place adversarial patches in the left and/or right painted range.

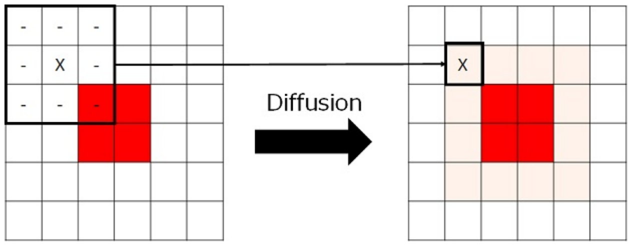
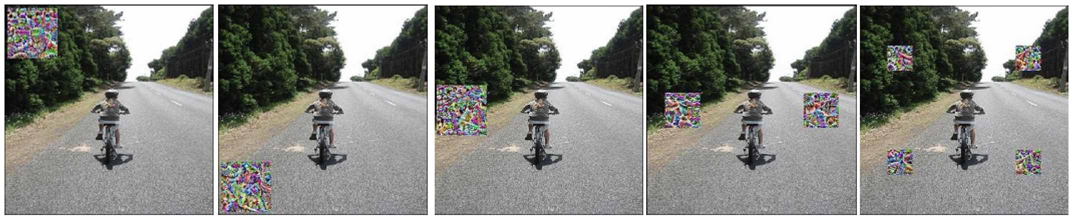


Figure 2: Our diffusion model at a convolutional layer with a 3×3 filter size. The effect of adversarial patches spreads to adjoining cells.

To evaluate the spreading range of adversarial patches, we simulated the effect of the remote adversarial patches shown in Figures 3a-Figure 3e. These placements are not optimized now, and we give detailed information of these adversarial patches in Appendix C.



(a) Top left corner (b) Bottom left corner (c) Middle left (d) Two patches (e) Four patches

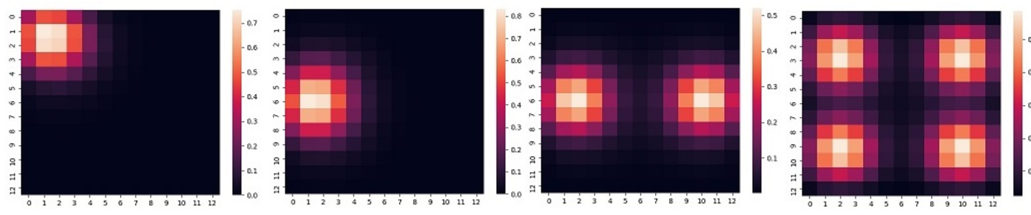
Figure 3: Layout of different remote adversarial patches. The area of adversarial patches are approximately 10,000 pixels. Figures (a)-(c) have a 100×100-size adversarial patch, Figure (d) has two 70×70-size adversarial patches, and Figure (e) has four 50×50-size adversarial patches.

Figures 4a-4d show the effect values calculated using different adversarial patch layouts depicted in Figures 3a,3c,3d, and 3e, respectively. We omit the result for Figure 3b because this layout is represented upside-down in Figure 3a. In Figure 4a, the effect of the patch is only noticeable near the adversarial patch. In Figure 4b, the effect of the patch spreads throughout the left part of the image. In Figures 4c and 4d, the effects of the patches spread throughout the image. Based on these calculations, we determine that the effect of the patch will be higher with multiple adversarial patches

Table 1: Average existence probabilities in YOLOv2 and SSD

# (Patches)	None	1			2	4
Area	-	100×100 =10,000 pixels			70×70×2 =9,800 pixels	50×50×4 =10,000 pixels
Layout	-	top left	bottom left	middle left	-	-
YOLOv2 [14]	64.77%	50.95%	44.89%	30.59%	**12.81%**	**16.76%**
SSD [11]	68.17%	24.52%	19.00%	15.25%	**6.12%**	**7.31%**

than a single adversarial patch. In a single patch, the patch placed middle left has a more prominent effect compared with the other single adversarial patch layouts. We also verified our calculation results based on the results obtained by the actual adversarial patches generated in the next section, which is shown in Appendix D.



(a) Top left corner (b) Middle left (c) Two patches (d) Four patches

Figure 4: Effect values calculated using our diffusion model. From the left, these figures correspond to the images depicted in Figures 3a,3c,3d, and 3e, respectively.

4 Experiments: Existence Probabilities with Adversarial Patches

In this section, the average existence probabilities with adversarial patches are calculated experimentally as depicted in Figures 3a-3e. We focused on YOLOv2 [14] and SSD [11] using the PASCAL VOC dataset [3]. We use YOLOv2 in Saha et al.'s implementation [16] and SSD in the PyTorch library [18]. Our experiment generated adversarial patches based on Saha et al.'s implementation [16], and adversarial patches were generated for each class in the PASCAL VOC dataset. Our implementation, however, learned adversarial patches without interruptions, while Saha et al.'s method interrupted the learning process when the existence probability was less than 0.35 in each image. Moreover, our implementation minimized the sum of $\Pr(target\ category|object)$ and the non-printability score [17] for each corresponding category, while Saha et al.'s original method minimizes only $\Pr(target\ category|object)$. Then, the average existence probability in each category was calculated by averaging the highest $\Pr(target\ category)$ for each image. Detailed information is given in Appendix E.

The experimental results are shown in Table 1. Table 1 shows the average existence probabilities of all 20 categories, and the average existence probability in each category is given in Apendix E. Table 1 demonstrates that our evaluation presented in the previous section is valid. The legitimacy of our evaluation stems from significantly lower average existence probabilities with 2 or 4 adversarial patches compared with that of a single adversarial patch. In conclusion, multiple adversarial patches have a stronger effect a single adversarial patch.

5 Conclusion and Future Work

In this study, we searched for a better remote adversarial patch layout on CNNs, using our proposed diffusion model. We defined our diffusion model, such that it underlines that multiple remote adversarial patches have a stronger effect compared with a single adversarial patch. We verified this by generating remote adversarial patches based on Saha et al's implementation [16].

In future work, we should consider our attack, "multiple" adversarial patches, leads to breaking some defending techniques. For example, well crafted multiple adversarial patch will evade certified defense using masking techniques [10, 13, 20, 21], because adversarial patches exist in multiple places. Moreover, our attacking method will be able to combine with attacking methods for breaking empirical defense [5, 8]. Therefore, we will evaluate our attacks on defended CNNs in future work.

Broader Impact

Our remote adversarial patches will cause misrecognition on CNNs, but we believe that our findings will pave the way for further improvements in countermeasures against adversarial patch attacks and further understanding of the mechanism behind CNNs.

Acknowledgments and Disclosure of Funding

This paper is based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Araujo, A., Norris, W., and Sim, J., Computing Receptive Fields of Convolutional Neural Networks. Distill (2019). `https://distill.pub/2019/computing-receptive-fields/`

[2] Brown, T. B., Mané D., Roy, A., Abadi, M., and Gilmer, J., Adversarial Patch. eprint arXiv 1712.09665 (2017).

[3] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A., The Pascal Visual Object Classes Challenge: A Retrospective. International Journal of Computer Vision, 111(1), 98–136 (2015).

[4] Gu, J., Tresp, V., and Qin, Y., Are Vision Transformers Robust to Patch Perturbations? eprint arXiv 2111.10659 (2021).

[5] Hayes, J., On Visible Adversarial Perturbations & Digital Watermarking. CVPRW 2018, pp. 1678–1685 (2018).

[6] Hoory, S., Shapira, T., Shabtai, A., and Elovici, Y., Dynamic Adversarial Patch for Evading Object Detection Models. eprint arXiv 2010.13070 (2020).

[7] Huang, H. Wang, Y., Chen, Z., Tang, Z., Zhang, W., and Ma, K.-K., Rpattack: Refined Patch Attack on General Object Detectors. ICME2021, pp. 1–6 (2021).

[8] Ji, N., Feng, Y., Xie, H., Xiang, X., and Liu, N., Adversarial YOLO: Defense Human Detection Patch Attacks via Detecting Adversarial Patches. eprint arXiv 2103.08860 (2021).

[9] Lee, M. and Kolter, J.Z., On Physical Adversarial Patches for Object Detection. eprint arXiv 1906.11897 (2019).

[10] Levine, A. and Feizi, S., (De)Randomized Smoothing for Certifiable Defense against Patch Attacks. NeurIPS 2020 (2020).

[11] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A.C., SSD: Single Shot MultiBox Detector. ECCV 2016. LNCS, vol. 9905, pp. 21-37 (2016).

[12] Liu, X., Yang, H., Liu, Z., Song, L., Li, H., and Chen, Y., DPatch: An Adversarial Patch Attack on Object Detectors. eprint arXiv 1806.02299 (2018).

[13] McCoyd, M., Park, W., Chen, S., Shah, N., Roggenkemper, R., Hwang, M., Liu, J.X., and Wagner, D., Minority Reports Defense: Defending Against Adversarial Patches. ACNS 2020, LNCS, vol. 12418, 564–582 (2020).

[14] Redmon, J. and Farhadi, A., YOLO9000: Better, Faster, Stronger. CVPR 2017, pp. 6517–6525 (2017).

[15] Rossolini, G., Nesti, F., D'Amico, G., Nair, S., Biondi, A., and Buttazzo, G., On the Real-World Adversarial Robustness of Real-Time Semantic Segmentation Models for Autonomous Driving. eprint arXiv 2201.01850 (2022).

[16] Saha, A., Subramanya, A., Patil, K., and Pirsiavash, H., Role of Spatial Context in Adversarial Robustness for Object Detection. CVPRW 2020, pp. 3403–3412 (2020). Avaliable source code at `https://github.com/UMBCvision/Contextual-Adversarial-Patches`

[17] Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K., Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. CCS2016, pp. 1528–1540 (2016).

[18] The Linux Foundation, SSD-Torchvision Main Documentation. `https://pytorch.org/vision/master/models/ssd.html` Last Access: September 21, 2022.

[19] Thys, S., Ranst, W.V., and Goedemé, T., Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection. CVPRW 2019, pp. 49–55 (2019).

[20] Xiang, C. and Mittal, P., PatchGuard++: Efficient Provable Attack Detection against Adversarial Patches. eprint arXiv 2104.12609 (2021).

[21] Xiang, C., Valtchanov, A., Mahloujifar, S., and Mittal, P., ObjectSeeker: Certifiably Robust Object Detection against Patch Hiding Attacks via Patch-Agnostic Masking. eprint arXiv 2202.01811 (2022).

[22] Zhu, Z., Su, H., Liu, C., Xiang, W., and Zheng, S., You Cannot Easily Catch Me: A Low-Detectable Adversarial Patch for Object Detectors. eprint arXiv 2109.15177 (2021).

A CNN networks

This section explains two CNNs used in this paper, namely YOLOv2 [14] and SSD [11].

A.1 YOLOv2

YOLOv2 [14] is an object detection system based on a CNN. YOLOv2 can detect objects by a single CNN running. We use a YOLOv2 CNN on the PASCAL VOC dataset [3], which is reported in Table 2. YOLOv2 outputs the position and size of a bounding box, and the existence probabilities of every VOC category in each bounding box.

We now explain the calculation process of the YOLOv2 CNN and how the existence probabilities are extracted from YOLOv2's output. As reported in Table 2, the YOLOv2 CNN consists of convolutional and pooling layers. YOLOv2's input is a 416×416-size image whose every cell has three elements representing RGB. Then, YOLOv2 outputs 13×13×125 elements. This output consists of 13 ×13 cells, where each cell contains five bounding boxes with 25 values. Each 13×13 cell represents offset of bounding boxes. In more detail, each 13×13 cell corresponds to 32×32 pieces of the original image, where the top left corner of each cell represents the offset of bounding boxes. Each bounding box has 25 values, which consist of the following:

- Position and size of the bounding box (4 values, combined with offset in each cell)
- Existence probability of an object (1 value, called as objectness score, represented as $\Pr(object)$)
- Conditional existence probabilities of every VOC category (20 values, called as category probabilities, represented as $\Pr(category|object)$)

In each bounding box, the existence probability of each VOC category in each bounding box is $\Pr(object) \times \Pr(category|object)$. The sum of 20 category probabilities is 1.

Table 2 shows the detailed YOLOv2's network for PASCAL VOC dataset. Each 13 ×13 cell in an output corresponds to 32×32 pieces of the original image as Figure 5, and the top left corner of each cell represents offset of bounding boxes.

A.2 SSD

SSD [11] can detect objects by a single CNN running, similar to YOLOv2. As described above, YOLOv2 outputs bounding boxes and existence probabilities on a 13×13-size image. Different from YOLOv2, SSD prepares multiple-size images and applies the same set of bounding boxes to them.

Table 2: YOLOv2 network for PASCAL VOC dataset

Layer	#(Filters)	Filter size	Stride	Input size	Output size
0 conv	32	3×3	1	416×416×3	416×416×32
1 max	–	2×2	2	416×416×32	208×208×32
2 conv	64	3×3	1	208×208×32	208×208×64
3 max	–	2×2	2	208×208×64	104×104×64
4 conv	128	3×3	1	104×104×64	104×104×128
5 conv	64	1×1	1	104×104×128	104×104×64
6 conv	128	3×3	1	104×104×64	104×104×128
7 max	–	2×2	2	104×104×128	52×52×128
8 conv	256	3×3	1	52×52×128	52×52×256
9 conv	128	1×1	1	52×52×256	52×52×128
10 conv	256	3×3	1	52×52×128	52×52×256
11 max	–	2×2	2	52×52×256	26×26×256
12 conv	512	3×3	1	26×26×256	26×26×512
13 conv	256	1×1	1	26×26×512	26×26×256
14 conv	512	3×3	1	26×26×256	26×26×512
15 conv	256	1×1	1	26×26×512	26×26×256
16 conv	512	3×3	1	26×26×256	26×26×512
17 max	–	2×2	2	26×26×512	13×13×512
18 conv	1024	3×3	1	13×13×512	13×13×1024
19 conv	512	1×1	1	13×13×1024	13×13×512
20 conv	1024	3×3	1	13×13×512	13×13×1024
21 conv	512	1×1	1	13×13×1024	13×13×512
22 conv	1024	3×3	1	13×13×512	13×13×1024
23 conv	1024	3×3	1	13×13×1024	13×13×1024
24 conv	1024	3×3	1	13×13×1024	13×13×1024
25 route 16					
26 conv	64	1×1	1	26×26×512	26×26×64
27 reorg	–	–	2	26×26×64	13×13×256
28 route 27 24					
29 conv	1024	3×3	1	13×13×1280	13×13×1024
30 conv	125	1×1	1	13×13×1024	13×13×125
31 detection					

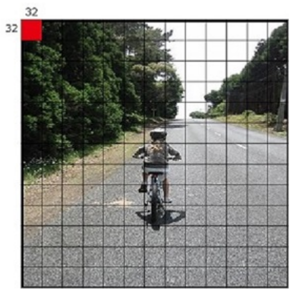


Figure 5: Position of output cells by YOLOv2 CNN. YOLOv2 CNN outputs 13 ×13 cells from a 416×416-size input image. Each cell's size is 32×32 as the piece at the top left corner.

We now explain SSD's network in more detail. In the original paper, the input of SSD is a 300×300-size image. SSD runs some layers in VGG-16 network on this image, and outputs the 38×38-size image. SSD calculates bounding boxes from output images obtained by applying convolutional and pooling layers on this 38×38-size image, namely a 38×38-size image, a 19×19-size image, a 10×10-size image, a 5×5-size image, a 3×3-size image, and a 1×1-size image. SSD applies bounding boxes, and tries to find an appropriate bounding box. The set of bounding boxes are the same between all-size image, and the ratio to an image is different. This causes SSD can detect larger objects in a small image. For example, we consider a 1×2-size bounding box. This bounding box has a very small part (0.14%) of a 38×38-size image, while this bounding box has a large part (22.2%) of a 3×3-size image. Therefore, SSD detects different-size objects in each-size image.

B Detailed Explanation of Our Diffusion Model

We evaluated the effects of these remote adversarial patches by calculating their respective effect values. Let P be the range of existing adversarial patches and l denote an index of the layer corresponding to the YOLOv2 network. The image size decreases as the calculation progresses. Consequently, let I_l be the range of an image at layer l, which is defined as follows:

$$I_l = \begin{cases} \{(x,y)\,|0 \le x \le 415, 0 \le y \le 415\} & (l = 0,1)\,, \\ \{(x,y)\,|0 \le x \le 207, 0 \le y \le 207\} & (l = 2,3)\,, \\ \{(x,y)\,|0 \le x \le 103, 0 \le y \le 103\} & (l = 4,5,6,7)\,, \\ \{(x,y)\,|0 \le x \le 51, 0 \le y \le 51\} & (l = 8,9,10,11)\,, \\ \{(x,y)\,|0 \le x \le 25, 0 \le y \le 25\} & (l = 12,13,14,15,16,17,26,27)\,, \\ \{(x,y)\,|0 \le x \le 12, 0 \le y \le 12\} & \text{Otherwise.} \end{cases} \tag{1}$$

Moreover, let $V_{l,x,y}$ be an effect value for cell (x,y) of layer l's input. We aim to calculate the effect values$V_{31,x,y}$ at $(x,y) \in I_{31}$. To calculate an output value, we fictitiously set zero effect values outside of an image, that is $V_{l,x,y} = 0$ at $(x,y) \in (\mathbb{Z} \times \mathbb{Z}) \setminus I_l$. These fictitious effect values are always zero, and we use these effect values when modeling of convolutional layer with a 3×3-size filter. Then, we calculate $V_{l,x,y}$ as follows:

- **Initial effect values:** Let $V_{0,x,y} = 1$, where $(x,y) \in P$, and let $V_{0,x,y} = 0$ for other cells.
- **Convolutional layer with a 3×3-size filter:** When l = 0,2,4,6,8,10,12,14,16,18,20,22,23, 24, and 29, we update effect values at $(x,y) \in I_{l+1}$ as $V_{l+1,x,y} = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1} V_{l,x+i,y+j}$. This equation calculates the average for $V_{l,x,y}$ and surrounding eight cells.
- **Convolutional layer with a 1×1-size filter:** When l = 5,9,13,15,19,21,26, and 30, we retain all values, that is, $V_{l+1,x,y} = V_{l,x,y}$.
- **Pooling layers with a 2×2-size filter:** When l = 1,3,7,11, and 17, we update effect values at $(x,y) \in I_{l+1}$ as $V_{l+1,x,y} = \frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1} V_{l,2x+i,2y+j}$. This equation calculates the average value of four cells' $V_{l,x,y}$ used in a new cell. In the "Reorg" layer, when $l = 27$, we apply the same update calculations.
- **The others:** We have not discussed layers 25 and 28 in the discussion above. In layer 25, we set $V_{26,x,y} = V_{17,x,y}$ where $(x,y) \in I_{17}$. In layer 28, we set $V_{29,x,y} = V_{25,x,y} + V_{28,x,y}$ where $(x,y) \in I_{28}$.

C Detailed Layout of Adversarial Patches Used in This Paper

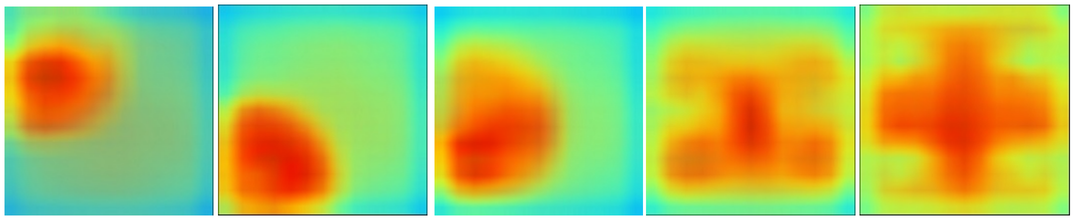
This section shows the detailed information of Figures 3a-3e. The following is the detailed information:

- **Figure 3a:** This figure shows a 100×100-size patch on the top left corner. This adversarial patch corresponds to $5 \le x \le 104, 5 \le y \le 104$. This layout is the same as Saha et al.'s placement.
- **Figure 3b:** This figure shows a 100×100-size patch on the bottom left corner. This adversarial patch corresponds to $5 \le x \le 104, 311 \le y \le 415$. This layout is upside down of Figure 3a.
- **Figure 3c:** This figure shows a 100×100-size patch on the middle left. This adversarial patch corresponds to $5 \le x \le 104, 158 \le y \le 257$. This adversarial patch is the center of left side.
- **Figure 3d:** This figure shows two 70×70-size patches. These adversarial patches correspond to $173 \le x \le 242$, ($35 \le y \le 104$ or $311 \le y \le 380$) We place these adversarial patches near the center of an image.

- **Figure 3e:** This figure shows four 50×50-size patches. These adversarial patches correspond to ($55 \leq x \leq 104$ or $311 \leq x \leq 360$), ($79 \leq y \leq 128$ or $287 \leq y \leq 336$). We place these adversarial patches near the center of image parted 2×2.

D Actual Diffusion of Adversarial Patches Effect

This section shows the verification of our diffusion model by the actual adversarial patches generated in Section 4. Figures 6a-6e shows actual averaged effect values. Especially, Figures 6a, 6c, 6d, and 6e correspond to calculation results depicted in Figures 4a-4d, respectively. We calculated effect values as the sum of every absolute value contained in each cell. Then, we obtained the results depicted in Figures 6a-6e by averaging effect values among all images. Figures 6a-6e showed that a range of adversarial patch effect is almost the same with an estimated range by our diffusion model.



(a) Top left corner (b) Bottom left corner (c) Middle left (d) Two patches (e) Four patches

Figure 6: A range of remote adversarial patch effect. The area of adversarial patches are almost 10,000 pixels. Figures (a), (c), (d), and (e) correspond to Figures 4a-4d, respectively.

E Detailed Experimental Result

This section shows the detailed experimental environment and results.

At first, we explain our experimental environment. We use the following two machines in this experiment:

- Ubuntu 20.04, 128GB memory, one GPU (NVIDIA GeForce RTX 2080 SUPER GPU) using Cuda 9.0. This machine has two PyTorch versions, PyTorch 1.1.0 for Cuda 10.0 (used in YOLOv2's experiment) and PyTorch 1.12.1 for Cuda 10.2 (used in SSD's experiment).
- Ubuntu 20.04, 32GB memory, two GPUs (NVIDIA RTX A5000), using Cuda 11.6 and PyTorch 1.12.1 for Cuda 11.6. This machine is used for SSD's experiment.

Our implementation was similar to that of Saha et al [16]. Our implementation, however, learned adversarial patches without interruptions, while Saha et al.'s method interrupted the learning process when the existence probability was less than 0.35 in each image. Other implementations functioned similarly. We adopted the standard l_∞ PGD attack. The learning process required 100 epochs with a learning rate of 0.01 and 10 iterations per image. The confidence, NMS, and IOU overlap thresholds used for the evaluations were 0.005, 0.45, and 0.5, respectively.

The detailed experimental results of YOLOv2 are shown in Table 3. Moreover, the detailed experimental results of SSD are shown in Table 4.

Tables 3 and 4 list the existence probabilities for our generated adversarial patches, which demonstrate that our evaluation presented in the previous section is valid. The legitimacy of our evaluation stems from significantly lower average existence probabilities with 2 or 4 adversarial patches compared with that of a single adversarial patch. In conclusion, multiple adversarial patches have a stronger effect than a single adversarial patch.

Table 3: Existence probabilities with respect to certain patch in YOLOv2

# (Patches)	None	1			2	4
Area	-	100×100 =10,000 pixels			70×70×2 =9,800 pixels	50×50×4 =10,000 pixels
Layout	-	top left	bottom left	middle left	-	-
aeroplane	75.92%	42.45%	44.52%	12.88%	4.04%	2.51%
bicycle	70.30%	70.41%	50.26%	26.60%	5.94%	25.17%
bird	74.25%	38.76%	56.05%	17.18%	4.78%	8.78%
boat	63.05%	38.13%	43.19%	23.50%	3.09%	7.39%
bottle	49.16%	45.11%	40.99%	35.18%	29.17%	25.54%
bus	60.16%	51.64%	43.36%	21.49%	1.55%	20.65%
car	72.00%	50.54%	44.01%	31.02%	18.93%	19.51%
cat	64.52%	34.33%	43.01%	16.65%	2.55%	0.14%
chair	47.78%	43.10%	32.06%	29.88%	21.18%	9.65%
cow	79.63%	70.48%	66.75%	68.45%	7.15%	10.56%
diningtable	53.27%	53.27%	26.26%	34.58%	45.53%	33.82%
dog	73.21%	63.97%	46.62%	20.92%	5.81%	6.89%
horse	73.04%	56.39%	41.28%	19.29%	0.59%	0.84%
motorbike	65.45%	62.73%	45.92%	35.81%	17.33%	38.84%
person	68.63%	59.79%	60.34%	56.01%	26.2%	29.98%
pottedplant	50.45%	45.30%	44.29%	36.03%	23.88%	27.59%
sheep	72.35%	71.07%	51.66%	41.46%	8.21%	14.08%
sofa	48.29%	44.50%	24.76%	17.09%	3.51%	14.37%
train	74.20%	33.88%	40.24%	20.27%	2.60%	4.96%
tvmonitor	59.83%	43.08%	52.16%	47.60%	24.2%	33.97%
Average	64.77%	50.95%	44.89%	30.59%	**12.81%**	**16.76%**

Table 4: Existence probabilities with respect to certain patch in SSD. The class name in SSD is given in parentheses.

# (Patches)	None	1			2	4
Area	-	100×100 =10,000 pixels			70×70×2 =9,800 pixels	50×50×4 =10,000 pixels
Layout	-	top left	bottom left	middle left	-	-
aeroplane (airplane)	76.65%	13.61%	13.24%	11.84%	5.62%	6.03%
bicycle	64.02%	28.54%	12.57%	13.85%	3.41%	5.42%
bird	68.65%	14.48%	14.67%	11.05%	5.09%	7.08%
boat	61.91%	19.33%	16.91%	14.05%	8.47%	10.21%
bottle	40.57%	30.84%	20.90%	19.52%	10.67%	11.76%
bus	76.06%	25.25%	20.72%	16.66%	2.81%	6.10%
car	68.79%	37.97%	31.87%	29.62%	16.92%	23.94%
cat	69.27%	13.21%	9.98%	4.36%	0.87%	0.49%
chair	53.75%	39.49%	24.89%	26.63%	8.71%	8.60%
cow	88.89%	36.85%	34.22%	20.02%	8.81%	8.16%
diningtable (dining table)	59.07%	17.88%	6.67%	5.77%	0.52%	0.44%
dog	72.77%	11.75%	9.58%	4.58%	0.86%	0.99%
horse	81.03%	13.83%	8.00%	8.96%	1.16%	2.43%
motorbike (motorcycle)	75.37%	32.77%	17.60%	15.71%	4.63%	4.83%
person	76.08%	44.59%	36.78%	30.95%	17.30%	20.19%
pottedplant (potted plant)	49.58%	25.90%	22.47%	18.81%	10.06%	11.02%
sheep	77.93%	34.60%	27.33%	20.65%	11.14%	11.96%
sofa (couch)	61.78%	15.71%	12.38%	7.84%	0.40%	0.81%
train	78.38%	12.29%	10.66%	7.03%	1.29%	1.01%
tvmonitor (tv)	62.94%	21.47%	28.61%	17.18%	3.57%	4.77%
Average	68.17%	24.52%	19.00%	15.25%	**6.12%**	**7.31%**