Feasibility of Inconspicuous GAN-generated Adversarial Patches against Object Detection

Svetlana Pavlitskaya[1,*], Bianca-Marina Codău[2] and J. Marius Zöllner[1,2]

[1]FZI Research Center for Information Technology, 76131 Karlsruhe, Germany
[2]Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany

Abstract

Standard approaches for adversarial patch generation lead to noisy conspicuous patterns, which are easily recognizable by humans. Recent research has proposed several approaches to generate naturalistic patches using generative adversarial networks (GANs), yet only a few of them were evaluated on the object detection use case. Moreover, the state of the art mostly focuses on suppressing a single large bounding box in input by overlapping it with the patch directly. Suppressing objects near the patch is a different, more complex task. In this work, we have evaluated the existing approaches to generate inconspicuous patches. We have adapted methods, originally developed for different computer vision tasks, to the object detection use case with YOLOv3 and the COCO dataset. We have evaluated two approaches to generate naturalistic patches: by incorporating patch generation into the GAN training process and by using the pretrained GAN. For both cases, we have assessed a trade-off between performance and naturalistic patch appearance. Our experiments have shown, that using a pre-trained GAN helps to gain realistic-looking patches while preserving the performance similar to conventional adversarial patches.

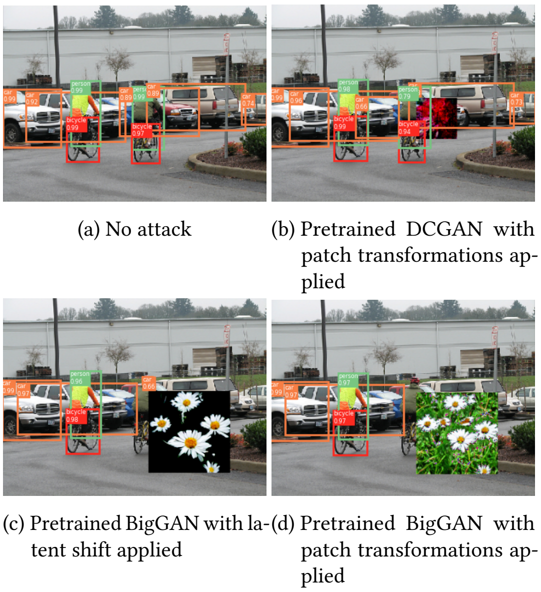
Keywords

adversarial attacks, object detection, GANs

Deep neural networks (DNNs) are vulnerable to adversarial attacks in which input data is deliberately modified [1]. In case of image data, adversarial noise is added to an input sample, affecting the entire image. Another type of attack is an adversarial patch, which can be positioned arbitrarily in a restricted region of an image. Patches can be applied to the input images digitally as well as in a real-world setting. But state-of-the-art research focuses on creating adversarial patches which are easily recognizable by the human eye. These are characterized by chaotic patterns, bright colors and do not resemble real-life objects but rather random noise [2, 3, 4]. A much harder problem is posed by creating inconspicuous patches as their purpose is to elude human detection while still being a threat to DNNs.

Recently, methods to enforce realistic appearance of adversarial patches have been proposed [5, 6, 7]. Existing approaches aim at deterring image classifiers or steering models as well as object detectors. In the latter case, however, an adversarial patch manages to attack only one large object in an input image.

In this work, we perform extensive literature research and identify promising approaches to generate inconspicuous adversarial patches. We further apply these methods to the object detection use case. Differently from the existing work on naturalistic patches against object detection, the focus of our work is to affect objects in the attacked image, which are located near the patch. We run experiments in a digital setting in per-instance and universal manner. We further analyse which ap-



(a) No attack

(b) Pretrained DCGAN with patch transformations applied

(c) Pretrained BigGAN with latent shift applied

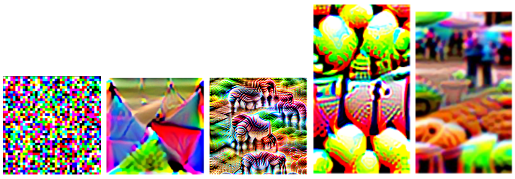
(d) Pretrained BigGAN with patch transformations applied

Figure 1: Overview of the patches generated with the evaluated methods

The IJCAI-ECAI-22 Workshop on Artificial Intelligence Safety (AISafety 2022), July 24-25, 2022, Vienna, Austria

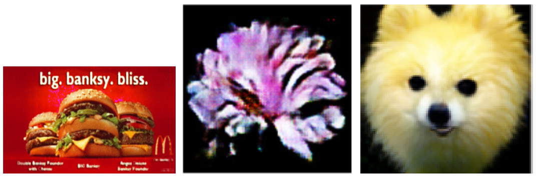
*Corresponding author.

pavlitskaya@fzi.de (S. Pavlitskaya); bianca.codau@student.kit.edu (B. Codău); zoellner@fzi.de (J. M. Zöllner)



(a) [9] (b) [3] (c) [10] (d) [13] (e) [14]

Figure 2: Examples of state-of-the-art **conspicuous** adversarial patches against object detection: (a-b) applied in a digital setting, (d-e) printed on a t-shirt



(a) PhysGAN [6] (b) TnT attack [7] (c) Naturalistic [5]

Figure 3: Examples of state-of-the-art **inconspicuous** adversarial patches against object detection

proach is the most suitable for the selected setting and discuss the trade-off between attack success and realistic appearance.

1. Related Work

1.1. Conspicuous Adversarial Patches

The idea of an adversarial perturbation restricted to a specific image area was first proposed by Brown et al. [2]. The first approaches focused on the image classification use case [8]. Later, patch-based attacks for object detection were also proposed [9, 3, 10]. A general approach consists in either maximizing the detector loss or, in case of an object vanishing attack, minimizing the detector loss for the empty label [11].

To enable attacks in the real-world setting, the non-printability loss component is usually added, which restricts pixel values to the set of printable colours. Furthermore, the total variation loss is usually applied in order to make colourful patterns of the generated adversarial patches appear smoother [12]. The patches then be printed, e.g. on a t-shirt to fool object detectors Examples of adversarial patch attacks against object detection in the real world are [13, 14]. Recently, a dataset of printable adversarial patches against object detection was introduced in [15]. However, adversarial patches generated in the conventional way still have a conspicuous character (see Figure 2).

1.2. Inconspicuous Adversarial Patches

An inconspicuous adversarial patch can be enforced either by using a specific loss function or a generative adversarial network (GAN). The first group of approaches maximizes the loss function to obtain patches that resemble a certain real image. *adv-watermark* [16], for instance, generates adversarial patches as image watermarks by performing a heuristic random search for the global minimum as an adaptation of the *Basin Hopping (BH)* optimization algorithm.

Recently, approaches of the second group, which rely on GANs, have gained popularity. We group GAN-based approaches into two categories: (1) methods which include patch generation directly into the GAN training process and (2) methods which generate an adversarial patch using a pretrained GAN.

A first attempt to use GANs to generate natural adversarial examples was performed by Zhao et al. [17]. Here, a pretrained Wasserstein GAN [18] is combined with an inverter, which maps data to the latent representation. The experiments, however, were restricted to the image classification on MNIST and LSUN datasets as well as on a text generation task.

1.2.1. Combined Patch-GAN Training

PhysGAN attack [6] is one representative of the first group of approaches. It is designed to generate patch attacks and place them in road side video footage to deter steering prediction models. For a given input video sequence, the algorithm learns a patch to be included into every frame. The PhysGAN model includes, next to a generator-discriminator pair, an encoder for extracting the features out of input video frames. The encoder output is then fed directly to the generator. The adversarial road sign, computed by the generator, and a real road sign are then sent to a discriminator. The resulting adversarial patch is then added to each frame of the original video sample creating an adversarial input. Finally, to obtain the perturbation, the generator is updated over the loss of the targeted model, calculated on the adversarial video slice, while taking the original frames as the ground truth. The resulting adversarial patch is indistinguishable from the roadside poster and leads to a noticeable prediction error.

Another approach designed to generate more realistic adversarial patches is the *Perceptual-Sensitive GAN (PSGAN)* [19]. It was evaluated on the traffic sign recognition as well as on general image classification use cases. It adapts existing patches, which are then placed in regions of an image in order to have the highest impact on final predictions. Similar to the Wasserstein GAN training [18], the PSGAN discriminator is updated several times in each epoch, whereas the generator is updated only once per epoch. Before each update, a minibatch of images and patches is sampled. The given minibatch of patches is fed to the generator to create the adversarial patches. Moreover, an attention model is included to determine a patch position that has the highest impact on the class prediction.

Closely related to PSGAN is the *Inconspicuous Adversarial Patches (IAP)* framework [20], which replicates the process of patch generation in PSGAN and repeats it for a series of generator-discriminator pairs. The goals is thus to reduce the conspicuousness of the patch by feeding it through the chain of GAN models. In the beginning, the background images are analyzed and an attention map indicating the best position for patch placement is calculated. Each GAN pair represents a step in the coarse-to-fine patch creation as it takes in the patch and background image at a different scale. The GAN training process remains the same as the generator aims to create realistic patches while the discriminator tries to distinguish them from the original images. IAP-generated patches aim to be indistinguishable from the background and thus resemble transparent masks.

1.2.2. Using a Pretrained Generator

In the second category, no full GAN training is performed. Instead, a pretrained GAN is used to improve patch appearance. The *Naturalistic Physical Adversarial Patch Attack*, developed by Hu et al. [5], aims to optimize for an adversarial patch in the GAN latent space directly. First, the patch is initialized as a noise vector. After the initialization, it performs a gradient update for each epoch and for each image, on which the patch is placed before the attack. For each iteration, the noise is fed to the generator to obtain the adversarial patch. The resulting patch is then added to the current image, which is then passed to the object detector. To perform an attack, a bounding box with the highest objectness probability and highest class probability is selected. The gradient descent is then used on the resulting loss, which also contains a total variation loss. Using the approach described above, Hu et al. performed several digital attacks, where they experimented with six different patch sizes, as well as physical attacks.

Universal NaTuralistic adversarial paTches (TnT) attack [7] is another approach relying on a pretrained GAN. This approach aims at attacking image classifiers with realistic universal patches. It uses Wasserstein GAN [18] with gradient penalty, which was pretrained on a dataset of flower images. For the background images, they used images from the ImageNet dataset to test the effectiveness of the attack in white-box and black-box setting. The TnT attack with high confidence scores on the pretrained image classifier had an up the three times higher attack success rate than the laVAN patch attack [8] while testing on the same image set. Finally, Doan et al. managed to create adversarial patches that resemble flowers, thus being less attention grabbing, but impacting the targeted classification model.

2. Approach

We identify two major groups of GAN-based approaches to generate inconspicuous patches and describe the proposed pipelines, adapted for the object detection use case. Our pipeline assumes using a white-box gradient-based approach for adversarial patch generation.

2.1. Combined Patch-GAN Training

In the first approach we incorporate adversarial patch training directly into the GAN training pipeline. This method attempts to map the processes of PhysGAN [6] and PSGAN [19] models from steering model prediction and image classification respectively to the object detector attack. We thus simultaneously train a GAN model to create a latent space of realistic-looking patches and an adversarial patch to deter the object detector.

An overview of the training pipeline in the case of the *combined Patch-GAN* attacks is presented in Figure 4. The patch is initialized randomly in the generator input format and undergoes two updates in each training epoch: one after the GAN training phase and one after the loss computation of the targeted object detector. Updating the patch after a GAN training step aims to restrict the patch to the latent space of realistic images developed by the GAN model.

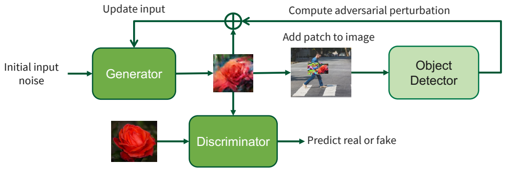


Figure 4: Overview of the combined Patch-GAN training

We further consider two extensions to the algorithm. First, we introduce a second generator update over the detector loss for the adversarial predictions. It takes into consideration the GAN loss for the generator, which gets the current patch as an input, and the loss of the object detector for the adversarial image. The current patch generation approach differs from PSGAN as the GAN loss is computed only over the patch and not over the entire adversarial image, similar to the PhysGAN.

Second, we use two different random noise vectors during the patch training. One noise vector is reinitialized with each epoch and background image as it is used to train the two GAN components, while the other is the actual patch noise, initialized as before and optimized with each epoch and background image with the goal of reducing the loss of the object detector under attack.

2.2. Patch Generation using a Pretrained GAN

The second approach focuses on restricting the trained patch to the images generated by a previously trained GAN model. Figure 5 shows the simplified pipeline for a *Pretrained GAN Patch Attack*. In this approach, random noise is fed into the generator to obtain a realistic image. Similar to the combined Patch-GAN approach, the patch is applied to a background image and the resulting adversarial image is passed to the object detector under attack. The patch is then optimized to change the loss of the object detector. However, the parameters of the generator are no longer updated during patch training as in the previous approach.

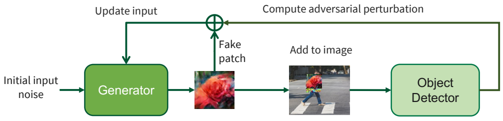


Figure 5: Overview of the patch training with a pretrained GAN generator

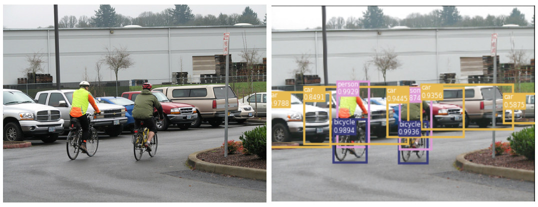
Our approach differs from the *Naturalistic Physical Patch Attack* [5] in the attack procedure. In particular, we no longer target the single *person* class and also focus on considering all objects in the image instead of a single object having the highest objectness and class probabilities.

3. Experiments and Evaluation

To evaluate the feasibility of the identified GAN-based approaches for inconspicuous patch generation for the object detection use case, we run experiments using YOLOv3 [21] as a model under attack.

3.1. Dataset and Models

We have performed experiments with YOLOv3 model [21], using an open source Python implementation[1], detection was performed at the resolution 416x416 pixels.



(a) Input image (b) YOLOv3 predictions

Figure 6: YOLOv3 predictions on an unattacked COCO image for the per-instance experiments

The images to be attacked come from the COCO dataset [22]. For the per-instance attacks, we use an exemplary COCO image (see Figure 6).

We use two GAN architectures: DCGAN [23] and BigGAN [24]. To train DCGAN, we have used the Flower Recognition dataset [25]. The dataset contains 4,242 flower images of 320x240 pixels equally split into the classes *daisy, dandelion, rose, sunflower, and tulip*. The dataset was built for image classification, not for unsupervised training for image generation as needed for the GAN models. Therefore, we performed dataset cleaning by manually removing the images containing scenarios such as a field of flowers or humans holding flowers, as these represent outliers from the intended GAN latent space, namely single flower generation. The clean Flowers Recognition dataset thus contains 1,385 images.

DCGAN was trained with the batch size of 64 with the Adam optimizer and learning rate 0.0002. The generated images have a size of 64x64 pixels and are further resized to reach the patch size. For BigGAN, we used the open source PyTorch re-implementation[2], pretrained on Imagenet.

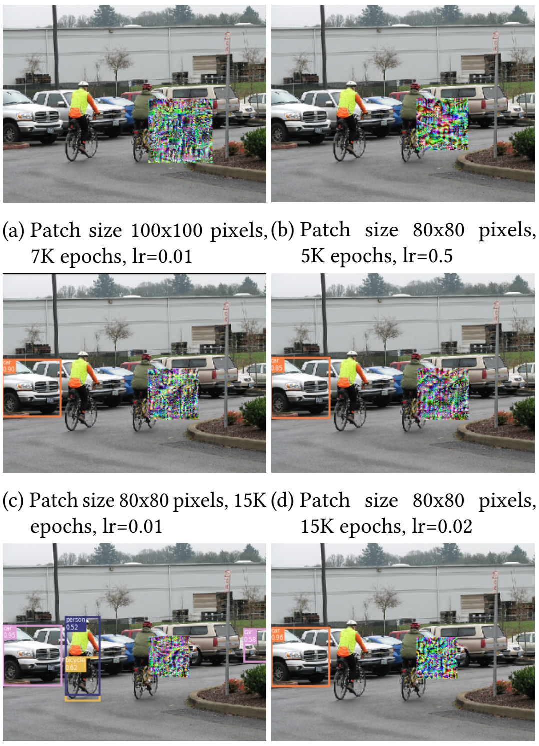
We use the PGD algorithm [26] for attacks. All trainings were performed on an NVIDIA RTX 1080 Ti GPU with 11GB VRAM.

3.2. Conspicuous Baseline Patches

To enable a fair comparison, we have first generated conventional adversarial patches using PGD. We have focused on the object vanishing attack, i.e. we have applied loss maximization using empty ground truth labels to enforce suppression of object detections.

Figure 7 demonstrates the PGD patches of different sizes, We have experimented with various training times and learning rates. The 100x100 pixels PGD patch requires 7K epochs at learning rate 0.01 to suppress all bounding boxes (see Figure 7a). The 80x80 pixels patch

[1] https://github.com/eriklindernoren/PyTorch-YOLOv3
[2] https://github.com/huggingface/pytorch-pretrained-BigGAN



(a) Patch size 100x100 pixels, 7K epochs, lr=0.01

(b) Patch size 80x80 pixels, 5K epochs, lr=0.5

(c) Patch size 80x80 pixels, 15K epochs, lr=0.01

(d) Patch size 80x80 pixels, 15K epochs, lr=0.02

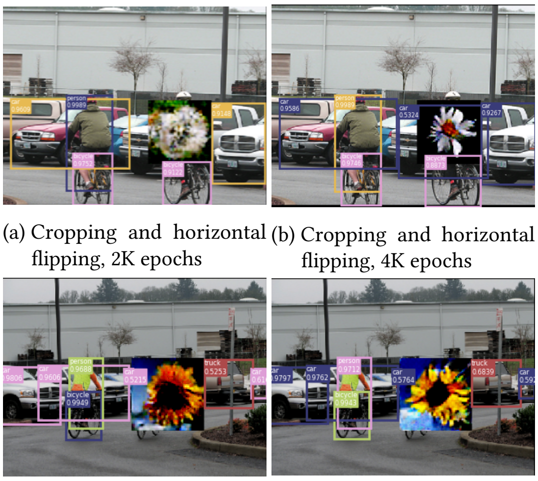
(e) Patch size 60x60 pixels, 10K epochs, lr=0.02

(f) Patch size 60x60 pixels, 10K epochs, lr=0.5

Figure 7: Attacks with conventional PGD patches

only achieves the same result in 5K epochs when using a learning rate of 0.5 as seen in Figure 7b. Training of the 80x80 pixels patch with learning rates of 0.01 and 0.02 did not manage to suppress all bounding boxes even after 15K epochs (see Figures 7c and 7d). Because of its smaller attack surface, we train the 60x60 pixels patch directly with a learning rate of 0.02. Figure 7e shows, however, that this patch does not manage to suppress four bounding boxes, which are placed towards the image margins. Using the learning rate of 0.5 for the 60x60 pixels patch gives better results. Only one bounding box remains in Figure 7f. The performance of the 60x60 pixels patch stagnates and the confidence score of the remaining bounding box does not decrease after 100K epochs, at which point the training is stopped.

Overall, the conventional PGD patches are able to completely supress all detections in an input images using a sufficiently large patch (at least 80x80x pixels, i.e. 3% of an input). The smaller the patch, the more it profits from a higher learning rate and longer training time.



(a) Cropping and horizontal flipping, 2K epochs

(b) Cropping and horizontal flipping, 4K epochs

(c) 2,5K epochs

(d) 5K epochs

Figure 8: Attacks with the combined PGD-GAN training using two generator updates per epoch

3.3. Combined PGD-GAN Training

For the combined PGD-GAN approach, we used the DCGAN architecture, while the PGD attack was implemented as done for the conspicuous baseline.

Following the baseline, a model with one discriminator and generator update per training step was first evaluated. After generator and discriminator parameters are updated at step t, the generator gets an updated patch at step $t+1$ and outputs a new patch. We then insert the new patch in the COCO image and produce predictions with the YOLOv3 object detector. After computing the YOLOv3 loss, the patch optimizer was run in order to update the current patch state. This approach led to highly distorted patches not resembling the dataset, whereas the patch itself had no impact on the surrounding bounding boxes.

We have achieved better results via introducing a second generator update. We thus updated generator twice per epoch: first during the GAN training step and then after the patched image is evaluated and the loss of YOLOv3 is calculated. Figure 8b shows, that, the patch images remain in the dataset distribution after 4K epochs. However, with each newly generated image, a different flower type is created (a dandelion at 2K epochs and a daisy at 4K epochs). At both stages the covered person is not detected and the confidence score for the car in the back decreases.

Figures 8c and 8d demonstrate how this version of the algorithm performs without horizontal flipping. The patch covers an entire cyclist, which prevents it from being identified. Moreover the adjacent cars are identified as such only with a 0.57 and 0.68 confidence score

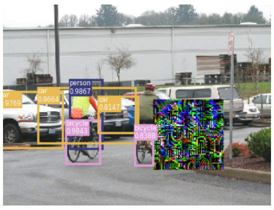
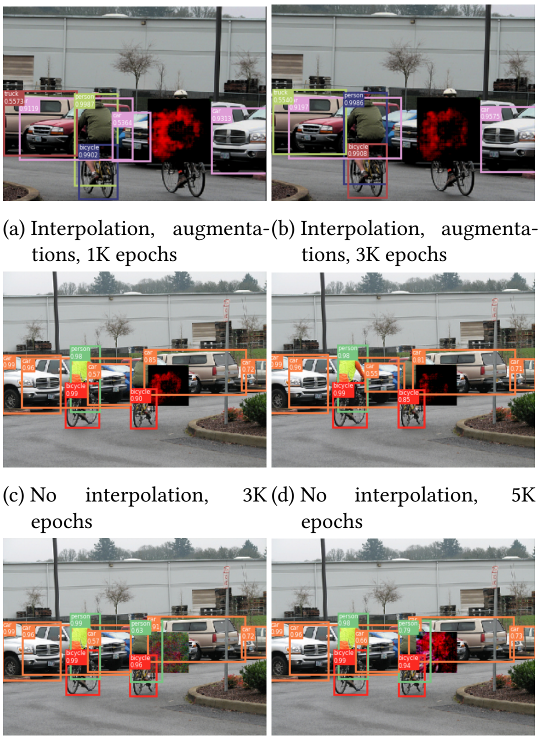


Figure 9: Attack with the combined PGD-GAN training with generator update after the discriminator and patch updates. Results are shown after 1K epochs



(a) Interpolation, augmentations, 1K epochs

(b) Interpolation, augmentations, 3K epochs

(c) No interpolation, 3K epochs

(d) No interpolation, 5K epochs

(e) With latent shift applied, 1K epochs

(f) With patch transformations applied, 5K epochs

Figure 10: Attacks with a pretrained DCGAN

respectively, which are lower than in the corresponding clean image. However, the confidence score does not decline linearly over the epochs. For instance, the red bounding box in Figure 8d displays a higher confidence score of 0.68 at epoch 5K compared to only 0.52 in epoch 2500 as shown in Figure 8c.

The last model that we have evaluated included updating the generator once per epoch, after both the discriminator and the patch were updated. Figure 9 shows that the patch developed with this method manages to suppress more bounding boxes in the neighbouring region. However, the generator obviously does not learn the distribution of the GAN training dataset.

In summary, we could generate realistic looking adversarial patches with the combined approach. The best performing version of the algorithm included two generator updates per epoch. The attack success, however, is worse than when conspicuous adversarial patches are used.

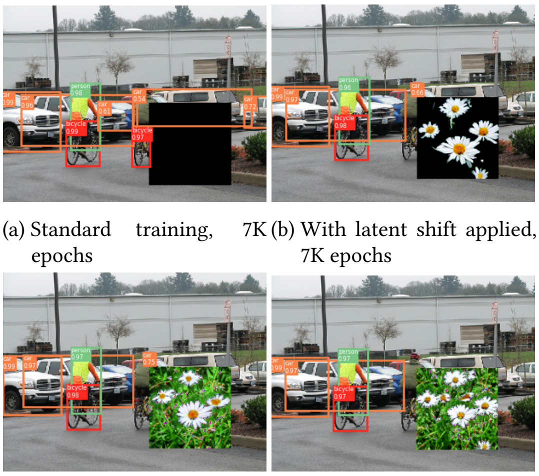
3.4. Using a Pretrained Generator

Next, we evaluate the usage of a pretrained GAN generator. We have experimented with two GAN models: DCGAN and BigGAN. DCGAN was trained for 2K epochs on the Flowers Recognition dataset, which was preprocessed as described above. For the BigGAN, we have used the pretrained model and set the chosen class to daisy (985). The patch optimization is the same as for the conspicuous baseline, the weight for the total variation is set to 0.01.

Figure 10 demonstrates the results for the experiments with the pretrained DCGAN. We have first experimented with patches resized from 64x64 as generated by DCGAN to 100x100 pixels using interpolation. As Figure 10a shows, the patch is placed on a cyclist, which deters the object detector from recognizing the person, the bicycle and the car behind them after 1K epochs using a learning rate of 0.01. Moreover, the car to the left of the patch has the reduced confidence score of 0.53 compared to the clean image score of 0.76. However, a major problem here is that the patch is getting darker with each training epoch (see Figure 10b).

Experiments without patch interpolation (i.e. using patches of size 64x64 pixels as generated by DCGAN) also show the same darkening effect (see Figures 10c and 10d). As expected, these patches also do not suppress the surrounding boxes as well as the previous experiment due to their smaller attack surface, but the confidence scores are decreased. This is also consistent with our conspicuous baseline experiments. The adversarial patch, generated with the pretrained generator, only covers part of the cyclist but the object detector cannot detect a person. In addition, the bounding boxes surrounding the patch have a lower confidence score. Affected are the detections of the cars to the right of the patch as well as the bicycle below it.

To mitigate the darkening patch effect, we further evaluate two countermeasures. First, we apply latent shift interpolation. For that, we initialize a patch mask of randomly distributed values and then apply it to the patch via interpolation. This procedure is repeated during each training epoch before applying the patch to the COCO image. Figure 10e shows results for this approach after 3K training epochs. In this case, the patch value does not



(a) Standard training, 7K epochs

(b) With latent shift applied, 7K epochs

(c) With patch transformations applied, 7K epochs

(d) With patch transformations applied, 10K epochs

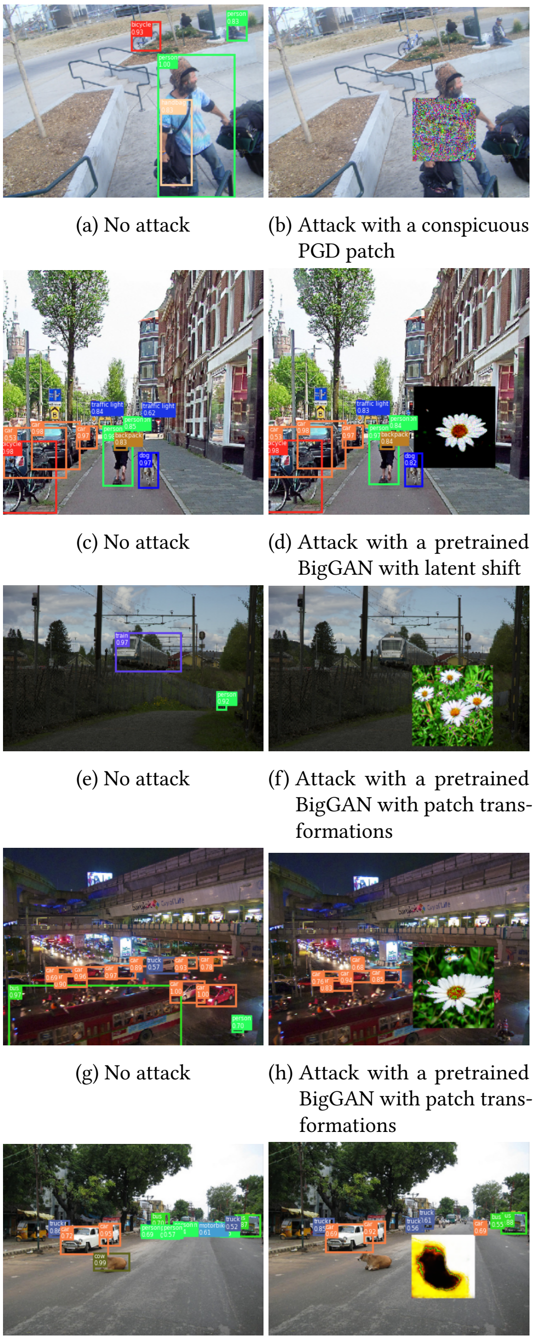
Figure 11: Attacks with a pretrained BigGAN

remain in the DCGAN image distribution, but resembles noise, which diverges from the flower images, and does not improve with longer training time. Moreover, the patch performs worse than the previous experiments during the evaluation. The person to the left is recognized by the object detector albeit with a lower score than in the clean image. The other surrounding bounding boxes do not have a considerably reduced confidence score.

A further attempt, aiming to improve the appearance of patches, is the usage of patch transformation, as suggested in [3]. This approach aims at making patches more robust and includes a number of transformations applied to a patch before it is added to an input image. In includes adding random noise to the patch as well as random changes in patch brightness and contrast. In particular, we first multiply the patch with a contrast mask and then add brightness and noise masks. In all cases, masks include randomly sample values, the contrast interval is restricted to [0.8, 1.2], the brightness interval is restricted to [-0.1, 0.1], the noise mask contains values in the interval [-0.1, 0.1]. As can be seen in Figure 10f, the patch stays in the latent space of the DCGAN model after 5K epochs. This, however, comes at a cost of small rise in the confidence of object detections near the patch.

As the figures demonstrate, in our DCGAN experiments we have no control over the generated flower class, so that patches may contain various flowers during the training.

We further repeat the experiments with the BigGAN model. The chosen class is 985 representing daisies. The experiments are performed with the patches of 128x128 pixels, which is the size of the original BigGAN generator



(a) No attack

(b) Attack with a conspicuous PGD patch

(c) No attack

(d) Attack with a pretrained BigGAN with latent shift

(e) No attack

(f) Attack with a pretrained BigGAN with patch transformations

(g) No attack

(h) Attack with a pretrained BigGAN with patch transformations

(i) No attack

(j) Attack with a pretrained DCGAN with patch transformations

Figure 12: Examples of universal patch attacks generated for a subset of COCO, all trained for 1K epochs

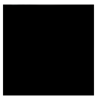
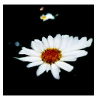
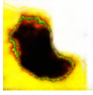
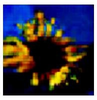
	No attack	Black	Conventional PGD	[5] with class=daisy	Pretrained BigGAN + latent shift	Pretrained BigGAN + transformations	Pretrained BigGAN + transformations	Pretrained DCGAN + transformations	Combined PGD-GAN training
									
mAP	43.8	4.2	**3.4**	3.8	3.8	4.4	4.1	3.8	3.9
AP$_{person}$	80.3	55.2	**28.4**	40.4	41.8	39.3	32.0	41.2	46.0
AP$_{bicycle}$	2.2	**0.0**	**0.0**	**0.0**	**0.0**	0.2	0.2	**0.0**	**0.0**
AP$_{car}$	7.9	3.1	**0.0**	1.2	0.3	0.4	0.3	0.3	0.3

Table 1

Mean average precision (mAP) and average precision (AP) for certain classes in % for universal patches, generated for a subset of COCO

output. Figure 11a displays the patch attack result after 7K epochs. It turns completely black, however it still manages to suppress the identification of the person to the left of the patch.

Next, we assess the effect of adding the interpolation with the latent value. Figure 11b shows the patch resulting from 7K epochs. In this case, only the background of the flower images turns black while the flowers remain clearly visible. Moreover, the patch manages to suppress the bounding boxes of the cars above and to the right of its position as well as the identification of the first cyclist and the first bicycle on the left.

Finally, we apply patch transformations. This helps to fully overcome the problem of the dark patch background, as the patch background is not longer black, but resembles a field. As Figure 11c shows, the patch achieves similar results to the previous BigGAN experiment from Figure 11b. It suppresses the same bounding boxes and shows a confidence score of 0.75 for the car bounding box in the upper left corner of the patch. This score is higher than in the previous BigGAN experiment but lower than in the clean image. Moreover, by training the pretrained BigGAN patch with transformation for 10K epochs on one COCO image, the bounding box in the upper left corner is suppressed as well (see Figure 11d).

In summary, the approach involving a pretrained generator leads to a significantly higher image fidelity. In a standard setting, the patch tends to get completely black, but the proposed latent shift and patch transformations help to overcome the problem. As expected, BigGAN led to significantly better patches due to larger capacity.

3.5. Universal Inconspicuous Patches

Finally, we evaluate whether the studied approaches to generate inconspicuous patches can also be applied in a universal manner. The goal of a universal attack is to fool all images with a single perturbation [27].

For the experiments, we create a subset of the COCO dataset, containing objects of classes *person, car, bicycle.* The resulting subset contains 1,146 images, which are further split according to the COCO protocol to 1,101 train and 45 test images. All universal patch training experiments are run for 1K epochs over the entire training dataset. The patch learning rate is set to 0.01 and the GAN learning rate for the combined PGD-GAN patch attack is set to 0.0002. The patch size during training is set to the original size of the GAN architecture output (i.e., 64x64 for DCGAN and 128x128 for BigGAN) to avoid information loss through resizing. The patch placement is fixed similar in the per-instance experiments.

Using the conventional PGD patches, we could suppress all bounding boxes in the test images. The universal patch generated using the pretrained BigGAN with patch transformations for brightness and contrast was also successful (see Figure 12). In comparison, the pretrained DCGAN patch attack has a reduced effect on the object detection (see Figure 12j). However, it reduces the confidence scores of the surrounding bounding boxes significantly. One major difference to the previous example is the quality of the image and of the generated object respectively. The daisy image in this case is distorted and no longer recognizable as a flower.

We have trained and evaluated several patches using the same settings (see Table 1). We have also evaluated a patch, generated using the approach by Hu et al. [5] using the open-source code[3]. Following the procedure in the paper, the training was performed on the INRIA dataset [28] for 1K epochs. We also set the class to daisy. Note, that direct comparison with the method by Hu et al. [5] is not possible due a different method to add patch to an image (see Figure 13). Instead of attacking object of a certain class by direct overlapping with a patch, we focus on a single patch at a fixed position in an image, which can attacks all objects.

Every approach managed to reduce the average mAP

[3] https://github.com/aiiu-lab/Naturalistic-Adversarial-Patch

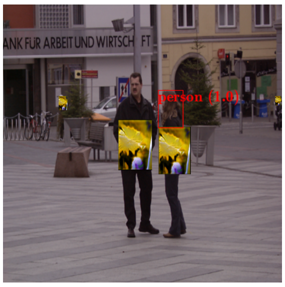


Figure 13: A naturalistic patch, created using the framework by Hu et al. [5], attacks objects of the class *person* via overlapping with the clothing area

drastically, whereas the best result was obtained with the conventional PGD attack, as expected. The patch generated according to [5] achieves the same mAP, as the pretrained BigGAN without transformations and the pretrained DCGAN with transformations. This patch also has the best results for the class *person*, but worst for the class *car*. Finally, the pretrained BigGAN patch with transformation scores the highest mAP for both images, being least effective overall. In the case of one of the pretrained BigGAN patches with patch transformations, the mAP score of 4.4% is even higher than the black square mAP value. The patches generated with the pretrained BigGAN demonstrate, however, the most naturalistic appearance out of all universal experiments, also compared to the results obtained with the framework by Hu et al. [5].

4. Conclusion

In this work, we have evaluated the existing GAN-based methods for inconspicuous patch generation on the object detection use case. Following the analysis of the state of the art, we have identified two groups of promising approaches: the first method focuses on combining the GAN training process with the training of the adversarial patch, while the second one relies on a pretrained GAN model during the patch training process. For each group, we have adapted the procedure to attack the object detector and ran the experiments on YOLOv3 as a model under attack both in per-instance and universal settings using the COCO dataset. All attacks were performed using the PGD algorithm. Differently from the state of the art, we focused on suppressing objects in the direct proximity of a patch, which is also a realistic attacks scenario.

Our experiments have demonstrated, that using the pretrained GAN generator leads to adversarial patches of higher visual fidelity. Better performing BigGAN led to more realistic looking patches compared to DCGAN. However, since BigGAN training on ImageNet is resource consuming, we have performed the experiments on combined PGD-GAN training only with a DCGAN model. Evaluating the combined training approach with a GAN of larger capacity might lead to even better results.

During evaluation of the universal attacks, we could observe an evident trade-off between the patch appearance and the attack performance. Our pretrained DCGAN and combined PGD-GAN have demonstrated attack performance comparable to the state-of-the-art approach by Hu et al [5], although no direct comparison is possible because of different patch placement approaches. The pretrained DCGAN approach as well as the PGD GAN approach led to a better attack success than the pretrained BigGAN method during evaluation. Overall, the performance on the test set under attack was significantly lower than on the clean images. Although the attack strength of the conspicuous patches could not be reached, the studied approaches present a promising trade-off between the attack success and naturalistic appearance.

Acknowledgments

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project "KI Absicherung" (grant 19A19005W) and by KASTEL Security Research Labs. The authors would like to thank the consortium for the successful cooperation.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing Properties of Neural Networks, International Conference on Learning Representations (ICLR) (2014).

[2] T. B. Brown, D. Mané, A. Roy, M. Abadi, J. Gilmer, Adversarial Patch, in: Advances in Neural Information Processing Systems (NIPS) - Workshops, 2017.

[3] S. Thys, W. V. Ranst, T. Goedemé, Fooling automated surveillance cameras: Adversarial patches to attack person detection, in: Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops, Computer Vision Foundation / IEEE, 2019.

[4] S. Pavlitskaya, S. Ünver, J. M. Zöllner, Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control, in: International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2020.

[5] Y.-C.-T. Hu, J.-C. Chen, B.-H. Kung, K.-L. Hua, D. S. Tan, Naturalistic Physical Adversarial Patch for Object Detectors, in: International Conference on Computer Vision (ICCV), Springer, 2021.

[6] Z. Kong, J. Guo, A. Li, C. Liu, PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving, in: Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision Foundation / IEEE, 2020.

[7] B. G. Doan, M. Xue, S. Ma, E. Abbasnejad, D. C. Ranasinghe, Tnt attacks! universal naturalistic adversarial patches against deep neural network systems, CoRR abs/2111.09999 (2021).

[8] D. Karmon, D. Zoran, Y. Goldberg, Lavan: Localized and visible adversarial noise, in: International Conference on Machine Learning (ICML), PMLR, 2018.

[9] X. Liu, H. Yang, Z. Liu, L. Song, Y. Chen, H. Li, DPATCH: an adversarial patch attack on object detectors, in: AAAI Workshop on Artificial Intelligence Safety, 2019.

[10] M. Lee, J. Z. Kolter, On physical adversarial patches for object detection, CoRR abs/1906.11897 (2019).

[11] K. H. Chow, L. Liu, M. Loper, J. Bae, M. E. Gursoy, S. Truex, W. Wei, Y. Wu, Adversarial objectness gradient attacks in real-time object detection systems, in: International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), IEEE, 2020.

[12] M. Sharif, S. Bhagavatula, L. Bauer, M. K. Reiter, Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition, in: Conference on Computer and Communications Security (CCS), ACM, 2016.

[13] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P. Chen, Y. Wang, X. Lin, Adversarial t-shirt! evading person detectors in a physical world, in: European Conference on Computer Vision (ECCV), Springer, 2020.

[14] Z. Wu, S. Lim, L. S. Davis, T. Goldstein, Making an invisibility cloak: Real world adversarial attacks on object detectors, in: European Conference on Computer Vision (ECCV), Springer, 2020.

[15] A. Braunegg, A. Chakraborty, M. Krumdick, N. Lape, S. Leary, K. Manville, E. M. Merkhofer, L. Strickhart, M. Walmer, APRICOT: A dataset of physical adversarial attacks on object detection, in: European Conference on Computer Vision (ECCV), Springer, 2020.

[16] X. Jia, X. Wei, X. Cao, X. Han, Adv-watermark: A novel watermark perturbation for adversarial examples, in: International Conference on Multimedia, ACM, 2020.

[17] Z. Zhao, D. Dua, S. Singh, Generating natural adversarial examples, in: International Conference on Learning Representations (ICLR), 2018.

[18] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN, CoRR abs/1701.07875 (2017).

[19] A. Liu, X. Liu, J. Fan, Y. Ma, A. Zhang, H. Xie, D. Tao, Perceptual-Sensitive GAN for Generating Adversarial Patches, in: AAAI Conference on Artificial Intelligence, 2019.

[20] T. Bai, J. Luo, J. Zhao, Inconspicuous adversarial patches for fooling image-recognition systems on mobile devices, IEEE Internet of Things Journal (2021).

[21] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, CoRR abs/1804.02767 (2018).

[22] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: common objects in context, in: European Conference on Computer Vision (ECCV), Springer, 2014.

[23] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, in: International Conference on Learning Representations (ICLR), 2016.

[24] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in: International Conference on Learning Representations (ICLR), 2019.

[25] A. Mamaev, Flowers Recognition, https://www.kaggle.com/datasets/alxmamaev/flowers-recognition, 2018. Accessed: 2022-01-12.

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks, International Conference on Learning Representations (ICLR) (2018).

[27] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in: Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision Foundation / IEEE, 2017.

[28] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision Foundation / IEEE, 2005.