Inconspicuous Naturalistic Adversarial Patches (INAPs) for Fooling Object Detectors

Audrey Ruckman
MITRE
aruckman@mitre.org

Abstract

As the use of machine learning becomes more common in our everyday lives, it is important to be aware of the limitations and vulnerabilities in these algorithms. In the computer vision domain, recent studies have shown models to be vulnerable to a particularly effective attack called the adversarial patch. The deployment of inconspicuous adversarial patches is of great concern as it may not be immediately apparent to human observers that the attack is taking place and may go unnoticed for some time. This paper introduces a generative adversarial network (GAN) based approach to generating inconspicuous naturalistic adversarial patches (INAPs), which are intended to degrade the performance of object detection algorithms while remaining difficult to detect visually. This approach is separate from other work on traditional adversarial patches for several reasons. First, this work targets object detection model failures rather than classification. Secondly, this work focuses on adversarial patches that are placed in the background of a scene, rather than on top of the subjects in a scene. Additionally, the GAN architecture utilized for INAP creation in this work only requires a single image input for training.

1. Introduction

In the past decade, the use of machine learning (ML) and deep neural networks (DNNs) capable of handling complex tasks has become more common in many domains, such as supporting Netflix recommendation algorithms or informing a semi-autonomous car whether it is within highway lanes. As the use of ML becomes more common in our everyday lives, it is important to be aware of the limitations and vulnerabilities in these algorithms to maintain security and reliability in our systems. Maintaining complete security, however, is difficult as the adversarial threats to ML systems are constantly evolving.

In the computer vision (CV) domain, a particularly effective attack demonstrated in recent studies is the adversarial patch [1], [2]. Adversarial patches can cause state-of-the-art image classification algorithms to misclassify an image by manipulating a small set of its pixels. More, adversarial patches can cause state-of-the-art object detection algorithms to change, add, or remove detections of multiple objects within a single image with a high attack success rate. They are a security concern for large-scale impact systems that rely on CV, such as autonomous vehicles or surveillance mechanisms that are used to localize and identify hundreds of objects in a single input frame. As a result of this, adversarial patches that can disrupt object detectors are the focus of this work.

The traditional adversarial patch is usually brightly colored and pixelated, which makes it easily identifiable. However, the concept of inconspicuous patches, which are strategically crafted to blend into a scene and avoid detection by humans or ML algorithms, are an emerging threat to CV systems [2]. Inconspicuous naturalistic adversarial patches are of great concern as it may not be immediately apparent that an attack is taking place and may go unnoticed for some time. Some work has been done investigating the feasibility of generating such inconspicuous patches [2], [3] ,[4]. However, much of the prior work has been done in the classification realm and/or generate relatively conspicuous patches.

This paper introduces a generative adversarial network (GAN) [5] based approach to generating inconspicuous naturalistic adversarial patches (INAPs), which are intended to degrade the performance of object detection algorithms. This work is different from other work on traditional adversarial patches for several reasons. First, this project targets object detection model failures rather than classification. Secondly, this work focuses on INAPs that are placed in the background of a scene, rather than on top of the subjects in a scene. Additionally, the GAN architecture utilized for INAP creation in this work only requires a single image input for training and encourages the generated patches to 'blend' into the background scene as much as possible while still disrupting the applied object detector. In this paper, we outline the related work in GANs and adversarial patches, discuss the methodology for INAPs, experimental results, limitations, and potential extensions to the work.

2. Background on GANs

2.1 Generative adversarial networks

GANs were introduced in 2014 by Goodfellow et al. [5]. The GAN framework entails the simultaneous training of two neural networks. One of the neural networks is a generative model, called the generator, G, that captures the data distribution from the training data. The second model, called the discriminator, D, estimates the probability that a sample produced by G was pulled from the training set rather than generated.

Generators in GANs learn the distribution of the training data through first receiving an input noise variable z with uniform distribution p_z and then map z to a learned data space. The process can be represented as $G(z;\theta_g)$, where θ_g represents the parameters of G. The discriminator produces its probability scalar that G's output data x is genuine, defined as $D(x;\theta_d)$, where θ_d represents the parameters of D [5].

As training iterations progress, D learns to maximize its probability of assigning the correct label to x, while G simultaneously learns to minimize $(1-D(G(z)))$. This two-player minimax game introduced in [5] can be represented with the cost function $V(D,G)$ as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

2.2 Periodic spatial GANs

An extension to Goodfellow et al.'s GAN was introduced by Bergmann et al. in 2017 called the Periodic Spatial GAN (PSGAN) [6]. PSGAN builds from the traditional GAN framework through the construction of a training dataset built of cropped patches from a single image, which allows training of the PSGAN with only one input image $X \in R^{HxWx3}$. Then, to account for the cropped subsets of training data that make up the entire input image X, the noise variable z that is fed to generator G is modified to have dimensions L and M, such that $z \in R^{LxMxd}$ and H > L and W > M [6]. Thus, the discriminator D is only ever receiving individual subsets of the data from the input image X, which is expressed as $D_{\lambda\mu}(X')$, where X' is a subset of X and $1 \le \lambda \le L$ and $1 \le \mu \le M$ [6]. The λ and μ represent the dimensions of the receptive field of the discriminator that runs convolutionally across input images, as shown in similar GAN studies [7], [8]. The losses of all image patches are averaged to provide the final loss of the discriminator, which is effective in capturing local high-frequency features of the input image, though less effective at capturing the global structure of the image [9]. Then, the cost function of PSGAN is expressed as:

$$\mathcal{L}_{PSGAN} = V(D,G) = \frac{1}{LM}\sum_{\lambda=1}^{L}\sum_{\mu=1}^{M}\mathbb{E}_{\mathbb{Z}\sim p_z(Z)}\left[\log\left(1-D_{\lambda\mu}(G(Z))\right)\right] + \frac{1}{LM}\sum_{\lambda=1}^{L}\sum_{\mu=1}^{M}\mathbb{E}_{X'\sim p_{data}(X)}\left[\log D_{\lambda\mu}(X')\right] \quad (2)$$

With the extended cost function $\mathcal{L}_{PSGAN}$, PSGAN allows for improved learning of the local and global features of a single input image, which is a capability that is important to the proposed solution in this paper. Without the addition of PSGAN to stabilize training, the INAP framework experienced instability more frequently with the traditional GAN architecture, though the ability to revert to the traditional GAN is a selectable option in the framework.

3. Background on adversarial attacks

3.1. Adversarial perturbation

The concept of adversarial attacks was introduced by Szegedy et al. in 2013 [10] and gained significant interest in academia in 2014 [11] when Goodfellow et al. discovered that by superimposing imperceptible noise (i.e., perturbation) to images, they could cause classifiers to report incorrect predictions. The demonstrated attack showed that the imperceptible perturbations could be produced through the maximization of the applied network's prediction error, a technique referred to as 'fast gradient sign method'(FGSM) [11]. Since the introduction of FGSM, many studies have been published such as Moosavi-Dezfooli et al.'s Deepfool in 2016 [12] which studied the robustness of classifiers against perturbation, and Madry et al.'s projected gradient descent (PGD) in 2017 [13], which suggested a stronger perturbation attack through iterative optimization.

3.2. Traditional, naturalistic, and inconspicuous adversarial patches

Inspired by adversarial perturbations, Brown et al. suggested the adversarial patch attack in 2017 [1], in which rather than applying imperceptible perturbations over an entire input image, the perturbations were constrained to a select region of the image. An intriguing property of the adversarial patch attack was its physical realizability in that a patch could be printed out and applied to a scene in a real-world attack. However, unlike imperceptible perturbation attacks, this type of attack was quite perceptible and appeared unnatural in the attacked scene.

To address this, an extension of the adversarial patch attack introduced in 2021 called the 'naturalistic' adversarial patch [14], [15], [16], in which the contents of the patch were inspired by real world objects or patterns and appear less artificial and less random than the traditional adversarial patch. The naturalistic patches

generated in [14], [15] were created using GANs paired with a classification model, and [16] paired a GAN with an object detector. Each of these studies relied on GANs that were trained on a large training dataset (between 600 and 900 samples) of the adversary's target object class.

To further disguise adversarial patch attacks, Bai and Jinqi [2] proposed the 'inconspicuous' patch attack in 2022; another GAN-based approach paired with a pre-trained classifier for patch creation. This work differs from previous GAN-based approaches however, in that the goal of the patch is to blend into scene as much as possible rather than attempting to mimic a real-world object. For this reason, their GAN only required one input image for training since it did not need a training dataset to learn a target class's feature distribution. Instead, their GAN was paired with a saliency model, a technique proposed by Liu et el. in [17], which detects the most salient region of the scene being attacked and selects that area as the single training image for the GAN. In [2], to generate the saliency map, the target model was used in white-box settings while an available pretrained model was used in black box settings. The resulting patches need not be completely imperceptible to the human eye to count as inconspicuous, but rather do not immediately raise human suspicion that the attacked scene has been altered.

4. Method

Building from the discoveries of other researchers on GANs and adversarial patches, this work proposes a GAN-based approach for generating inconspicuous and naturalistic adversarial patch attacks for fooling object detectors. This section discusses the problem statement, INAP framework architecture, and cost function details.

4.1. Problem statement

Given a victim image X with associated ground truth labels Y_{gt} and a pre-trained object detector f where $f_\theta: X \rightarrow Y$ in which Y is the associated annotation from f is represented by a list of tuples:

$$Y = [(c_0, s_0, b_0), \ldots, (c_{n-1}, s_{n-1}, b_{n-1})] \quad (3)$$

where each tuple i in the list Y represent an individual object detection with associated class label c_i, confidence score s_i, and bounding box coordinate b_i, and where n is the total number of detections. Then, INAP aims to generate a patch p such that when p is superimposed on X such that:

$$X' = X + p \quad (4)$$

and the resulting detection of object detector f on altered image X' is:

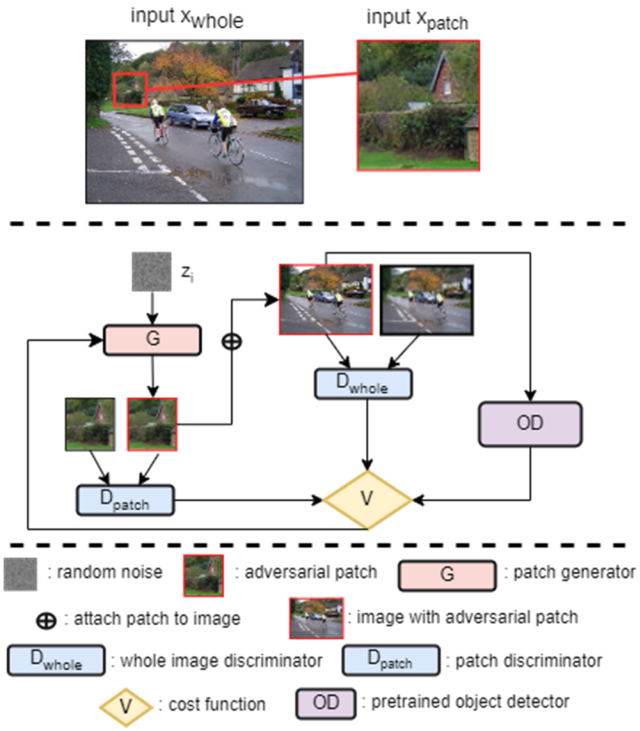


Figure 1: Overview of the INAP framework, in which one generator model $\boldsymbol{G}$ and three discriminator models $\boldsymbol{D_{patch}}$, $\boldsymbol{D_{whole}}$, and $\boldsymbol{OD}$ update cost function $\boldsymbol{V}$ throughout training.

$$Y_{pred} = f(X') \quad (5)$$

a prediction error arises from f such that:

$$Y_{pred} \neq Y_{gt}. \quad (6)$$

In the case that a prediction error arises, and a visual side by side comparison of image X to X' does not present an immediate obvious difference, the applied patch p is considered an inconspicuous adversarial attack on the object detector f. The two types of prediction errors that often occur are hallucinations, in which an object that is not present is detected, and a disappearance, in which an object that is present in the scene is not detected. The degree to which the adversarial patch alters the prediction of f varies widely, and as a result, some patches have more adversarial effect than others. Likewise, the degree of inconspicuousness of the generated patch varies and is likely to be correlated to its adversarial effect. The balance between adversarial effect and conspicuousness are discussed more in Section 4.3.

4.2. INAP framework

An overview of the INAP framework is presented in Figure 1. The framework consists of a traditional PSGAN

[6] generator and discriminator pair but includes an additional second GAN discriminator and third object detection model discriminator. The intention behind the second GAN discriminator is to ensure that the synthetic sample from the generator not only mimics its training data, but sufficiently blends into the entire content image, making it inconspicuous. Another benefit of the multi-discriminator GAN framework is its potential to help provide more reliable feedback to the generator model and may stabilize training [9]. The object detector that acts as the third discriminator in the framework is used to guide the adversarial aspect of the generated data.

Given an input image X_{whole} and user defined X_{patch} location, generator G receives the current training iteration's noise vector z_i and produces a synthetic sample, X'_{patch}. This sample is fed to the patch discriminator D_{patch}, as well as overlayed onto X_{whole}, producing X'_{whole}. Then, X'_{whole} is fed to the whole image discriminator, D_{whole}, as well as the pretrained object detector *OD*. The scalar predictions from D_{patch}, D_{whole}, and *OD* are fed to the cost function *V*, which in turn updates θ_g, the generator parameters, for the next iteration of training.

4.3. Cost function

Broadly speaking, the cost function in a GAN pipeline is used to provide feedback to the generator model on its synthetic output. It informs the generator of the error in its output based on the desired outcome as defined in the cost function. In the INAP framework specifically, as detailed in Figure 1, the cost function *V* expects three distinct terms, each with individual goals for the generated data. These terms are $\mathcal{L}_{patch}$, $\mathcal{L}_{whole}$, and $\mathcal{L}_{f_{classifier}}$.

The first term in *V* is the GAN loss from D_{patch}, which ensures that the generated patch mimics the input patch area selected for training. This term is expressed in equation (2), where a background on PSGAN was provided. It can be expressed specifically with D_{patch} as:

$$\mathcal{L}_{patch} = \sum_{\lambda=1}^{L}\sum_{\mu=1}^{M}(\mathbb{E}_{\mathbb{Z}\sim p_z(Z)}\left[\log\left(1 - D_{patch\lambda\mu}(G(Z))\right)\right] + \mathbb{E}_{X'\sim p_{patch}(X)}[\log D_{patch\lambda\mu}(X_{patch})]) \quad (7)$$

With the second discriminator in the GAN pipeline, cost function *V* is extended with the GAN loss of D_{whole} to ensure that the generated patch blends into the attacked scene. This loss value is responsible for guiding the generator to produce more inconspicuous patches that appear to blend into the scene. D_{whole} is trained on X_{whole}, whereas the generator only ever receives X_{patch}, and as such, does not require the PSGAN based cropping of training data that is done in $\mathcal{L}_{patch}$. Instead, it compares the image that has had the patch image superimposed on it, X'_{whole} to the benign image X_{whole}. The $\mathcal{L}_{whole}$ loss value can be represented as:

$$\mathcal{L}_{whole} = (\mathbb{E}_{(X'_{whole})}[\log D_{whole}(1 - X'_{whole})]) + \mathbb{E}_{(X_{whole})}[\log D_{whole}(X_{whole})]) \quad (8)$$

Lastly, to guide the adversarial aspect of the generated inconspicuous patch, the cost function incorporates the negative classification loss $\ell_{f_{classifier}}$ from the target object detection model OD, expressed as:

$$\mathcal{L}_{f_{classifier}} = \mathbb{E}_{(X'_{whole})}\ell_{f_{classifier}}(X'_{whole}, Y_{gt}) \quad (9)$$

where ℓ_f is the loss function that the *OD* model was trained with, and *Y* is the ground truth detection set for X_{whole}. This value is subtracted from the cost function, as the goal of the INAP framework is to guide the generated patch to being adversarial, and as such, an increased loss of the *OD* model should lower the total error of the resulting cost function value. The final cost function that incorporates the three individual loss term is expressed as:

$$\mathcal{L}_{INAP} = V(D_{patch}, D_{whole}, OD) = \boldsymbol{w_1}\mathcal{L}_{patch} + \boldsymbol{w_2}\mathcal{L}_{whole} - \boldsymbol{w_3}\mathcal{L}_{f_{classifier}} \quad (10)$$

where terms $w_1, w_2,$ and w_3 represent the respective weights placed on each term in cost function V. The weights can be adjusted to meet the desired INAP result. For example, w_3 may be increased to prioritize adversarial attack strength over inconspicuousness. More discussion of the weighting of these terms can be found in Section 5.2.

The training details are summarized in Algorithm 1.

Algorithm 1: INAP Generation

Input: images (X_{whole}, X_{patch}), pre-trained target model f
Output: set of adversarial patches $(X \mid X \in X'_{patch})$

for *each training iteration i* **do**
  sample $\boldsymbol{z_i} \sim \boldsymbol{P_z}$
  if *i = 1* **then**
    $Y_{gt} = f(X_{whole})$
  else
    $X'_{patch} = G(z_i)$
    $\theta_{D_{patch}} = D_{patch}(X'_{patch}, X_{patch})$
    $X'_{whole} = X_{whole} \oplus X'_{patch}$
    $\theta_{D_{whole}} = D_{whole}(X'_{whole}, X_{whole})$
    $Y_{pred}, \mathcal{L}_{f_{classifier}} = f(X'_{whole})$
  if $Y_{pred} \neq Y_{gt}$ **then**
    $X = X + X'_{patch}$
    $\theta_G = \theta_G + V(\theta_{D_{patch}}, \theta_{D_{whole}}, \mathcal{L}_{f_{classifier}})$
return *X*

5. Experimental results

A variety of experiments were conducted using the INAP framework to test its ability to generate INAPs against different scenes and settings. The experiments were conducted on a workspace that hosted Tesla V100-SXM2-32GB GPUs. The pre-trained object detection model used for INAP generation in white-box attacks was Torchvision's off-the-shelf implementation of the Faster-RCNN object detector architecture [18]. Due to the intensive computational requirements for patch generation, a limited number of images were selected for testing. In Table 1-5, an INAP is presented, alongside its adversarial impact, and a side-by-side visualization of the predictions on the image before and after the patch was applied.

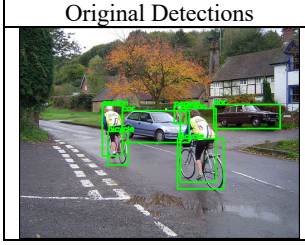
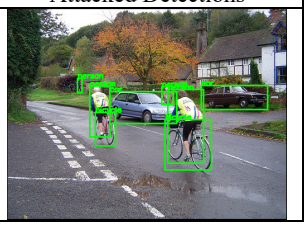
INAP	Adversarial Impact
	1 person hallucination of 64% confidence score
Original Detections	Attacked Detections
	

Table 1: Bikers

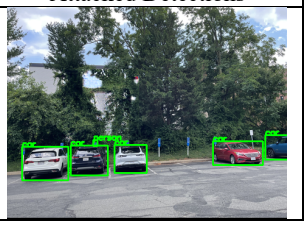
INAP	Adversarial Impact
	1 person hallucination of 88% confidence score
Original Detections	Attacked Detections
	

Table 2: Parking lot

INAP	Adversarial Impact
	3 person hallucinations of 93%, 93%, and 76% confidence scores

Original Detections	Attacked Detections
	

Table 3: Parking garage

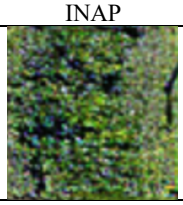
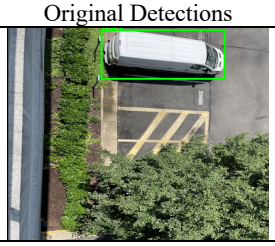
INAP	Adversarial Impact
	Dropped car detection confidence from 76% to 71%, which brought the confidence below the detector's threshold of 75%, causing a missed detection
Original Detections	Attacked Detections
	

Table 4: Overhead car

INAP	Adversarial Impact
	1 person hallucination of 47% confidence score
Original Detections	Attacked Detections
	

Table 5: Satellite

As shown in Tables 1-5, the generated INAPs successfully caused one or more hallucinations, which were much more common, or disappearance within the object detector's confidence threshold in the attacked scene while remaining inconspicuous at first inspection. The patches appear to capitalize on a select feature of the scene provided, and often alter the learned feature to cause a change in the applied model's prediction.

To test the transferability of the attack, the INAP example that caused a person hallucination in Table 1 was tested as an input to two other object detectors: YOLOv5

[19] and RetinaNet [20]. Table 6 shows the resulting detections:

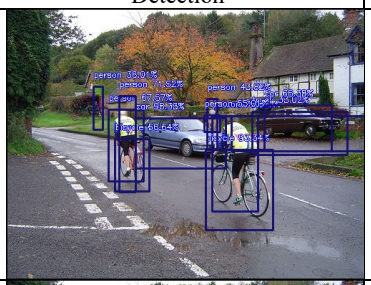
Model	Detection	Effect
YOLO v5		1 person hallucination of confidence 36.01%
Retina Net		1 person hallucination of confidence 17.64%

Table 6: INAP transferability

The detection scores on the hallucinated class in Table 6 show that the transferability of INAPs into a black-box attack are not as strong but do have potential to disrupt other models with varying strength.

5.1. Mode collapse and Wasserstein GANs

A common phenomenon that occurs with GANs called 'mode collapse' has been studied in recent works [21], [22]. It occurs when the generator in the GAN learns only a subset of the true distribution of the training data, and repeatedly produces outputs that only capture this subset. In mode collapse, the generator and discriminator no longer continue to learn, and are trapped in a cycle in which the generator produces the same limited data, and the discriminator repeatedly predicts the same value on the data. An example of a discriminator's prediction's during mode collapse is shown in Table 9. Training instability in GANs, like mode collapse, can occur for many reasons, some of which include the GAN architecture, cost function, or parameters. For INAPs, mode collapse was more frequently observed for larger patches and higher resolution images. It is up to the framework user to alter training parameters to reduce the risk of mode collapse. For example, increasing the batch size may introduce more diversity into the training data and improve stability. Additionally, a technique that has shown success in preventing mode collapse in other studies involving GANs [2], [15] is the Wasserstein generative adversarial network architecture [23], which was introduced by Arjovsky et al in 2017.

In WGANs, the discriminator no longer provides a prediction between 0 and 1 that the generator's output is authentic data. Rather, it produces an unbound value representing the Wasserstein distance between the real and generated data distributions it receives. This method can potentially provide the generator with more feedback on its output than a probability value.

Though WGAN improved stability in the INAP framework, it did not result in the production of adversarial results, except for in the first few epochs. This is likely a result of the WGAN's unbound loss values overpowering the bound object detector's loss values in later epochs. Table 7 reflects the cost function during training with and without the inclusion of WGAN, with the x-axis reflecting epochs and the y-axis reflecting generator error.

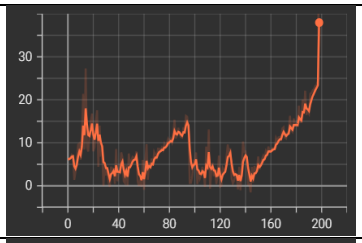
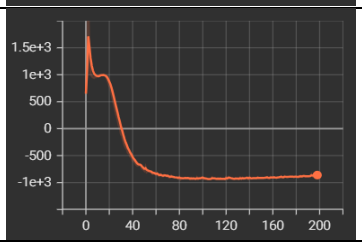
Bound GAN loss	
Unbound GAN loss (WGAN)	

Table 7: Cost function graphs with and without WGAN

Initial experimentation with a scaled object detector loss weight to compete with the unbound WGAN values did not show success in guiding the generator towards adversarial results.

5.2. Adversarial strength

As adversarial patches are more constrained and limited in their pixel values, it can be more difficult to use them to successfully fool an object detection model. In many applications of the INAP framework, per the nature of adversarial patches, it was unable to produce both inconspicuous *and* adversarial patches. The generated patches in these failure cases were typically one or the other – either well camouflaged into the scene, or a very noticeable but successful attack. A user of the INAP framework can experiment with the balance between inconspicuousness and adversarial strength through modifying the weight of the target object detector's loss value in the cost function during training (i.e., w_3 in equation (10)). Then the weight is increased, the INAP framework will be guided towards a stronger but less

inconspicuous patch, and vice versa. This impact is demonstrated in Table 8, where the weight of the object detector loss, or adversarial loss for the purpose of the framework, is modified. The off-the-shelf FRCNN model weights were used to determine adversarial impact. The first row shows the training data used to generate the INAPs shown in the second row.

Training Data:

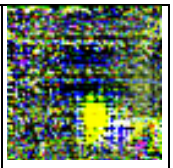
INAP			
OD Loss Weight	10	15	20
Adversarial Impact	None	Person hallucination of 72% confidence	Car hallucination of 88% confidence

Table 8: Attack strength versus inconspicuousness

Though the adversarial weight of the generated patches can be judged easily, there is much more difficulty in measuring their inconspicuousness. The term 'inconspicuous' is largely a judgement based on an individual human's eyes, which is not a value that can be measured quantitatively.

As a result, the INAP framework relies on the discriminators' predictions on the generated patches to determine its conspicuity and guide the patch generation. Table 9 demonstrates the difference in the patch discriminator's loss curve for each INAP in Table 9, where the x-axis represents epochs, and the y-axis represents the discriminator's prediction. The whole image discriminator produces very small loss values, as the changed data in the original and attacked images are quite small relative to the whole image, and as a result the whole image discriminator error curves are less visually informative. Note that these curves do not necessarily reflect the success of the INAP generation itself; rather, they show the predicted inconspicuousness of the generated patch from the patch discriminator over 200 epochs of training.

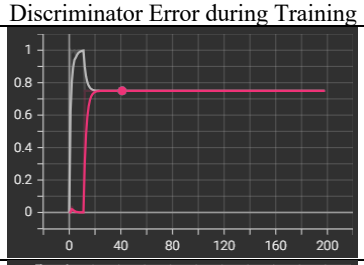
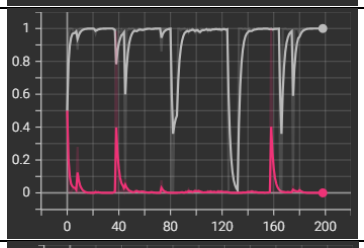
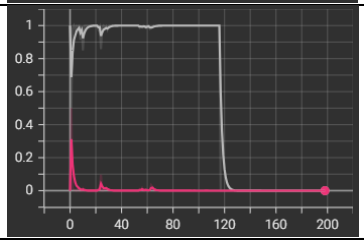
OD Loss Weight	Discriminator Error during Training
10	
15	
20	

Table 9: The lowest weight shows a stable loss curve of the discriminator (very inconspicuous), the middle weight shows an unstable loss curve (not very inconspicuous), and the largest weight shows a mode collapse of the GAN (not inconspicuous).

5.3. Printability

For physical deployment use cases, the INAP framework provides the option to incorporate a non-printability score (NPS) [24] into the cost function. Unlike digital pixel space that can produce a large range of values in the $[0, 255]^3$ RGB color space, physical printers have a more limited set of color values that can be printed onto paper, or they use different color space. When printing an adversarial patch onto paper, it may hinder the attack strength if the color values are not as close as possible to what was produced digitally. The NPS attempts to handle this pixel discrepancy in digital and physical space by measuring the distance between each pixel in the adversarial patch to color values that are known to be printable on physical paper. Note that using a NPS does not guarantee a perfectly printable output.

A side-by-side comparison of INAP examples from the same epoch in training that were generated under the same training conditions except the NPS is shown in Table 10.

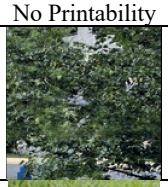
Original	No Printability	With Printability
		
		

Table 10: Non-printability scoring samples

A visual difference in the patches generated with printability is very subtle, but noticeable. An INAP framework user must consider the slight degradation in the GAN performance when incorporating NPS into training.

Additionally, incorporating a NPS into the INAP framework will inevitably prolong computation time, as stepping through each pixel in the generated patch can be an exponentially expensive task depending on the patch size. For the examples shown in Table 10, which are patches of size 128x128 pixels, computation time was extended by about 3 hours over 200 training epochs with the inclusion of printability scoring.

5.4. Computation Costs

The computation cost of the INAP framework depends on the input image size, the patch size, the number of training epochs, the batch size, the GPU capacity, the success rate of produced patches, and the decision to apply NPS to the training pipeline. During experimentation, when using patch sizes of 96x96 pixels, training time can take between 8-12 hours, but can take up to 3 days when patch sizes are increased past 256x256. Users who are looking to generate an INAP on a high-resolution image may save training time by lowering the image resolution, though the generated patch will also be of lower resolution and likely have different adversarial impacts.

To further reduce some computation costs, the INAP framework offers the ability to minimize the data passed to the whole image discriminator during training. Rather than passing the entire attacked image to the whole image discriminator, the attacked image is cropped to consist of only the region directly around the overlayed adversarial patch. This technique is meant to pass only the most necessary data to the whole image discriminator to determine if the patch appears to blend into its immediate surroundings in the scene, and in doing so, should lower overall training time. For example, a high-resolution image with dimensions 1280x848 and a selected INAP size 128x128 took 1 day and 21 hours to train. When cropped to 146x146 before feeding to the whole image discriminator, the immediate surrounding area of the patch location, training time took about 18 hours. Though there is less data being passed to the whole image discriminator, experiments have not suggested a difference in the framework's performance when this technique is in action.

Additionally, the number of times the generator produces adversarial results during training iterations will have a significant impact on training time as the added operations for plotting and saving annotated images with bounding boxes is costly. Users of the INAP frameworks may encounter varying results in adversarial patches and training time as the noise fed to the generator is randomly selected from a uniform distribution [6].

6. Future Direction

Future extensions of the pipeline could include a scaled weight of the object detector loss to compete with WGAN's extremely large or small loss values. Further alternative GAN architectures that were not tested in this work may produce stronger results, such as conditional GANs [25] which are capable of class conditional image generation. Additionally, diffusion techniques have shown great promise in image generation, especially with the combination of text prompt inputs as shown in [26] and may be a powerful alternative to GANs. Moreover, the application of saliency maps to the INAP framework may help identify vulnerable areas in an image in which to deploy a patch attack for stronger results. For attacks intending to cause a disappearance error, a semantic segmentation model could be used to identify relevant classes in a scene and generate patches *behind* the object.

Another extension could be the introduction of an attack detection technique such as entropy or anomalous pixel detection into the training pipeline such that the patch detector guides the generator to produce less detectable attacks. Finally, in generative AI based adversarial patches, it may be useful to trim the patch down to the minimum pixel space needed to maintain the attack. As seen in Tables 1-5, the hallucination attacks were only relevant to a portion of the synthetic patch and could likely be trimmed to reduce the total synthetic data in the attacked image.

7. Conclusion

This paper proposes a GAN-based approach to generating INAPs for fooling object detection algorithms in computer vision systems. With a single image for training data, the INAP framework can capitalize on existing features in a victim scene by slightly altering the feature in such a way that it is adversarial while remaining true to the original image. The novel INAP framework approach includes an additional discriminator in its GAN framework to ensure inconspicuousness. With experimental examples and discussion, this work demonstrates the plausibility of INAPs and their potential as a threat to computer vision systems, as well as possible

extensions to the work that may yield even stronger adversarial results.

References

[1] Brown, Tom B., et al. "Adversarial patch." arXiv preprint arXiv:1712.09665 (2017).

[2] Bai, Tao, Jinqi Luo, and Jun Zhao. "Inconspicuous adversarial patches for fooling image-recognition systems on mobile devices." IEEE Internet of Things Journal 9.12 (2021): 9515-9524.

[3] Pavlitskaya, Svetlana, Bianca-Marina Codău, and J. Marius Zöllner. "Feasibility of inconspicuous GAN-generated adversarial patches against object detection." arXiv preprint arXiv:2207.07347 (2022).

[4] Deng, Binyue, et al. "Rust-Style Patch: A Physical and Naturalistic Camouflage Attacks on Object Detector for Remote Sensing Images." Remote Sensing 15.4 (2023): 885.

[5] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[6] Bergmann, Urs, Nikolay Jetchev, and Roland Vollgraf. "Learning texture manifolds with the periodic spatial GAN." arXiv preprint arXiv:1705.06566 (2017).

[7] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[8] Li, Chuan, and Michael Wand. "Precomputed real-time texture synthesis with markovian generative adversarial networks." Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14. Springer International Publishing, 2016.

[9] Xu, Runze, et al. "Face transfer with generative adversarial network." arXiv preprint arXiv:1710.06090 (2017).

[10] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

[11] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).

[12] Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[13] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083 (2017).

[14] Hu, Yu-Chih-Tuan, et al. "Naturalistic physical adversarial patch for object detectors." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

[15] Doan, Bao Gia, et al. "Tnt attacks! universal naturalistic adversarial patches against deep neural network systems." IEEE Transactions on Information Forensics and Security 17 (2022): 3816-3830.

[16] Lapid, Raz, and Moshe Sipper. "Patch of invisibility: Naturalistic black-box adversarial attacks on object detectors." arXiv preprint arXiv:2303.04238 (2023).

[17] Liu, Aishan, et al. "Perceptual-sensitive gan for generating adversarial patches." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.

[18] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).

[19] G. Jocher, A. Stoken and J. Borovec, "Ultralytics/yolov5: V4. 0-Nn. SiLU () activations weights & biases logging PyTorch hub integration", Jan. 2021, [online] Available: https://zenodo.org/record/4418161.

[20] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proceedings of the IEEE international conference on computer vision. 2017.

[21] Lala, Sayeri, et al. "Evaluation of mode collapse in generative adversarial networks." High Performance Extreme Computing (2018).

[22] Zhang, Kaifeng. "On mode collapse in generative adversarial networks." Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30. Springer International Publishing, 2021.

[23] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.

[24] Sharif, Mahmood, et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." Proceedings of the 2016 acm sigsac conference on computer and communications security. 2016.

[25] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR, abs/1511.06434, 2015.

[26] Zhang, Lvmin, Anyi Rao, and Maneesh Agrawala. "Adding conditional control to text-to-image diffusion models." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.