为躲避探测器的衣服生成自然纹理的扩散模型

Diffusion Models for Generating Natural-looking Textures for Clothes that Evade Detectors

(申请清华大学工学硕士学位论文)

培 养 单 位 ： 计算机科学与技术系

学　　科 ： 计算机科学与技术

研 究 生 ： 奚　　墨

指 导 教 师 ： 胡 晓 林 副教授

二〇二四年二月

Diffusion Models for Generating Natural-looking Textures for Clothes that Evade Detectors

Thesis submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Computer Science and Technology

by

Bulatova Ekaterina

Thesis Supervisor: Associate Professor Hu Xiaolin

February, 2024

学位论文指导小组、公开评阅人和答辩委员会名单

指导小组名单

胡晓林　　副教授　　清华大学

公开评阅人名单

答辩委员会名单

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）按照上级教育主管部门督导、抽查等要求，报送相应的学位论文。

本人保证遵守上述规定。

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

(TODO: autotranslated) 最近，在构建物理上可实现的探测器规避攻击方面取得了巨大成功。尽管该领域取得了快速进展，但物理领域的对抗性攻击仍然常常不切实际：人眼很容易发现对抗性示例，因为它们不寻常的鲜艳颜色或过于显眼的怪异图案。

在这篇硕士论文中，我们提出了一种方法来创建真实的对抗纹理以逃避检测器，但在性能方面仍可与最先进的检测器竞争。受扩散概率模型强大生成能力的启发，我们采用预训练模型，通过指导生成物理上可实现的服装对抗纹理。

论文通过将结果与基线方法进行比较，证明了所提出的方法生成的模式的有效性、自然性和可控性。虽然模式的对抗效应并不高于最先进的方法，但我们的策略可以控制模式并能够以自然性来交换对抗效应。此外，由于该方法适用于预训练模型，因此它比基线要快得多。

关键词：对抗性攻击、物体识别、扩散模型、物理攻击、指导

ABSTRACT

Recently, huge success has been achieved in constructing physically realizable evasion attacks on detectors. Despite the rapid progress in the area, adversarial attacks in the physical domain still often suffer from being unrealistic: the human eye can easily spot adversarial examples due to their unusual vivid colors or overly conspicuous bizarre patterns.

In this master thesis, we propose a method to create realistic adversarial textures for evading detectors that would nevertheless compete with the state-of-the-art ones in terms of performance. Motivated by the strong generation abilities of diffusion probabilistic models, we employ pretrained models for generation of physically realizable adversarial texture for clothing using guidance.

The thesis demonstrates the effectiveness, the naturalness, and the controlability of the patterns generated with the proposed method by comparing the results against baseline methods. While the adversarial effect of the patterns is not higher than that of the state-of-the-art methods, our strategy gives control over the pattern and enable trading adversarial effects off for naturalness. Moreover, since the method works on pretrained models, it is significantly faster than the baselines.

Keywords: adversarial attacks; object detection; diffusion models; physical attacks; guidance

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ACRONYMS

AP	Average Precision
ASR	Attack Success Rate
bbox	Bounding BOX
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional Neural Network
COCO	Common Objects in COntext, Microsoft dataset for object detection
CV	Computer Vision
DETR	DEtection TRansformer, a transformer-based object detection model
DL	Deep Learning
DM	Diffusion Model
DNN	Deep Neural Network
IoU	Intersection over Union
LM	Language Model
ML	Machine Learning
R-CNN	Regions with CNNs, a region-based object detection model
RL	Reinforcement Learning
TPS	Thin plate splines, augmentation technique for imitating clothes
VAE	VAriational Encoder
YOLO	You Only Look Once, a grid-based object detection model

CHAPTER 1 INTRODUCTION

Deep Neural Networks (DNNs), an architecture of the large family of Deep Learning (DL) algorithms, are a powerful tool that is nowadays used for solving countless practical challenges.

In fact, almost every part of our everyday lives can be automatized to a certain degree with the help of an appropriate tool build upon a DNN. Cars and other vehicles can rely on Computer Vision (CV) systems for autonomous driving; the problem of international communication between people who do not know foreign languages had been significantly alleviated with the help of translation provided by Language Models (LMs); data analysis with the use of various basic Machine Learning (ML) techniques can now produce more accurate predictions for future trends in such important fields as finances, medicine, natural disaster prevention, and others; Reinforcement Learning (RL) enabled improvements in robotics, and so on.

A major problem that surfaces as the trend to deploying DNNs in all fields of our lives continues to gain momentum is the reliability of the said systems. In particular, even though the DNNs work exceptionally well on many tasks, they are known to be fragile to adversarial examples: an attacker can perturb input data in a way that the predictions of the model will be damaged.

This thesis addresses the aforementioned problem by demonstrating a way to construct such malicious examples. Thus, the work lies at the intersection of three major areas of DL: detection, adversarial attacks, and diffusion models. In this section, we delve a little deeper into these three areas as we introduce their importance to our modern fast-paced world.

1.1 Detection

CV tasks are one of the largest areas of research in DL. The human eye and brain are capable of acquiring and processing visual information easily and in an instant. By looking at a single picture, we immediately gain overall understanding of the situation, simultaneously retrieving a lot more data than we even realize.

In modern CV systems, however, the tasks are typically solved separately by applying different DL systems. For example, just as we can immediately “name” a picture

by analyzing the overall content and thus successfully categorizing the given image, a classification model can label images according to the classes that were introduced to the model during the training process. This particular task is called image classification, and, as far as the architectures of such models go, the task is relatively simple: in some cases, a single CNN (Convolutional Neural Network) that outputs probabilities of the classes for each given input is enough to achieve decent results.

Image classification, however, is far from being the only task in the large research area that CV is. Let us turn our attention to another ability our human vision provides: ability to detect and recognize various objects. Imagine driving down the street in a busy city. As a driver, you have to constantly monitor the situation outside your car windows: you have to quickly note and recognize road signs, watch over pedestrians that may want to cross the street, carefully keep your distance from other cars—after all, to err is human, and any driver must be ready to make swift decisions in response to other people's mistakes such as unskillful driving or crossing road in undesignated areas. For us, as for researchers of the CV area, this task is called object detection. If we describe the problem in a slightly more formal manner, it will sound like this: given an input image, our system is required to find all relevant to our task objects and classify them. Thus, the output of our system will more likely consist of three parts: firstly, the coordinates for every found object, secondly, what classes the objects are, and lastly, how confident the model is in these predictions. The last part is also easy to understand if you imagine being a driver that looks through a dirty window of their car trying to understand if the silhouette on the road in front of them is an animal or just a pool of water. If the driver is sufficiently confident that this is nothing but a pool, they can continue driving normally; otherwise, they need to stop the car.

Normally, unlike in a similar task of image segmentation where we have to provide class for each pixel of our input image, in case of object detection, we are only asked to provide a bounding box (bbox) around each object.

A large number of models with DNN backbones has been proposed to solve the object detection challenge, and the state-of-the-art methods in this field and have achieved high detection accuracy. For instance, the most famous works in this area include Faster R-CNN[1] (Faster Regions With CNNs), DETR[2] (Detection Transformer), and YOLO[3] end-to-end detectors.

1.2 Adversarial Attacks

As we have previously established, the fact that the DNNs are gradually getting into so many areas of our lives, gives rise to major concerns with the reliability of the DNNs.

Despite the exceptional results the DNNs show in most if not all of their applications, they have been proven to not be stable and consistent in their predictions, which can be exploited with malicious intents. The adverse effects of adversarial attacks are attracting a lot of attention now in light of the sheer amount of models produced for fault-sensitive practical tasks nowadays, such as detection modules of self-driving cars, chatbots, medicine, facial recognition systems and other applications, where a single mistake can cost lots of money or even human lives.

Unlike a DNN that is sensitive from the out-of-distribution input data to its inside architecture and training parts such as hyperparameters or initialization, the human eye is hard to fool. For example, let us consider the aforementioned task of image classification. When we see an image of a cat, we can immediately categorize it as such, even if the photo is perhaps a little bit dirty or even missing some parts. For a fragile CV system, however, a slight perturbation in the input data may be enough to give out completely incorrect prediction, going as far as to become absolutely sure that the provided image depicts a car, while for the human eye, the same perturbation will go undetected. It is easy to imagine an attacker changing a small set of pixels or adding unnoticeable noise to an input image; yet CV is not the only area vulnerable to such attacks. Even LMs that might seem more stable are susceptible to them: additional lines attached to the ending of an input line, or even seemingly unimportant and harmless typographical errors in given data may lead to abnormal behaviour of LMs. In recent years, many adversarial examples on different kinds of DNNs have been constructed, such as[4-6] etc.

In our case, however, as it has been mentioned previously, we are concerned with the safety a very specific area of DL: object detection. Attack on detection models are normally not as easy as attacks on less convoluted CV models like classifiers, yet it was shown that they are entirely plausible. Again, by introducing inconspicuous to the human eye changes to input data, one can cause the detectors to behave in bizarre ways. As we have established, a typical detection system output consists of three parts: bboxes that denote the position of elements, classes for each of them, and the confidence in found objects. Naturally, adversarial samples may aim to impair the operation of the detector with different corresponding attack types. For example, if an attack only causes the model to

assign an incorrect class, while still finding bboxes, this attack will be called a *misclassification* attack. As the title of the thesis shows, however, we are interested in avoiding detection altogether by performing *evasion* attacks. In practice, these attacks aim to lower the confidence of the detected bboxes. Let us imagine for a moment that we have come up with some kind of "perfect" evasion adversarial attack and our victim object detection model stopped detecting any objects of our interest, say, people. Now lets also assume that the victim detection model is actually a part of a CV system that dictates the movements to a car set on autonomous driving. Since the system is no longer able to detect humans at all, it will completely ignore any pedestrians, which will result in a car accident. Thus, it is easy to understand just how important it is to ensure the safety of our CV systems and protect them against adversarial attacks.

Given the importance of the stability of object detection systems, there is no wonder that the area does not lack active research. However, most of the attacks focus on achieving a higher attack success rate (ASR) and give little attention to how natural their adversarial perturbations seem to humans. Yet the lack of naturalness in adversarial examples lowers the value of the attack since such samples draw unwanted attention from human observers; moreover, it is easier to protect a CV system from such unnatural patterns than against those that seem harmless. Again, in practice, natural adversarial examples are important since they might be used for threatening models that absolutely need to be robust, like detectors in self-driving cars.

Furthermore, when it comes to practical applications in CV tasks, we become rather interested in physical scenarios instead of the digital ones. After all, the attacker rarely has access to the insides of a CV system, while changing the physical world by, for example, attaching a sticker to a road sign, is possible for everyone. While for the attacks in the digital space, it is fairly easy to introduce an inconspicuous change to the input data to fool the human eye, most of the physical attacks look unnatural since the task itself is way more difficult and requires more robust solutions compared to those in the digital space. In their turn, the adversarial samples also need to be way more robust than their digital analogies.

To address this issue, many natural-looking physical attacks have been introduced[7-9]. For the detection task, the adversarial attacks are mainly created as patches and stickers[10-12] or textures[13-14].

Stickers and patches are small pieces of paper with adversarial patterns, that can be

attached to something in the real world and disrupt the behaviour of their victim CV systems. Textures, in this case, are something that "envelopes" an entire object, like clothes to a person or colored surfaces to cars.

For the physical scenario, textures are preferable since a patch only retains its adversarial qualities at a limited range of angles of the camera in the real world. While it is possible to attach adversarial patches all over the object to avoid this problem at least partially, an object that has the same non-pattern patch repeated all over itself will look unnatural, while a texture would not attract as much attention. There are, of course, certain limitations to textures, even compared to patches. For instance, only a very limited range of objects does not look strange covered in a texture; normally, only personal belongings like clothes or car patterns that have been already mentioned before do not look suspicious when heavily personalized, unlike public objects like roads or signs. Patches, however, suffer from a similar problem since any foreign objects attached to things in public spaces are considered trash and removed as soon as the human eyes spots them.

While adversarial attacks by themselves have a negative societal impact in the sense that they might be used to fool DNNs with malicious purposes, we believe that this research can help with defending against attacks like this. Just like in cybersecurity where the people work either in the blue team coming up with defense techniques or in the red designing methods for attacks, we aim to push forward research that focuses on making the CV systems in all their forms more secure and robust.

1.3 Diffusion Models

Another research area that is highly relevant to this thesis is generative artificial intelligence, in particular, image generation in CV.

The first thought that comes to mind when it comes to applications of image generation is its artistic value, yet the usage of generative models go way beyond this. For this research, it is important to understand that they can play a big role in producing adversarial samples. For example, a correctly trained model can generate malicious out-of-distribution examples. While it is extremely useful in terms of how they can be used like augmentations to complement existing datasets for training more stable models, the models can also be used for attacking various systems.

For example, the patches, stickers, and textures we have discussed in the previous section can not only be constructed by perturbing existing patterns, but created from noise

with generative models.

IN particular, we are interested in generative abilities of diffusion models (DMs). DMs performed excellently on many tasks, and using them for generating adversarial samples is a relatively novel and promising way since overall DMs remain an underexplored research area even today.

A DM is usually given a text prompt as input and is required to produce a corresponding image. Overall, without going to deep into detail in the introduction part, while generating samples, pretrained DMs gradually remove noise from a random import, overall moving into the direction conditioned by the given prompt. Since the process is gradual and consists of many steps, one can alter the direction by bringing change to this trajectory. This particular property of DMs that is called *guidance* is important in this research since it means that there might be a way to alter that direction of a pretrained model to make it produce adversarial sample without even any additional training.

1.4 Thesis Goal

Now, armed with at least superficial understanding of all areas that are related to the research in this thesis, we can finally formulate the goal we are pursuing.

In short, we are to find a way to create physically realizable adversarial textures that are not immediately recognized as a threat by the human eye. The created framework should help to obtain such realistic adversarial textures for suppression of predictions of a detector by generating them with a diffusion model.

That is, we want to be able to generate patterns for clothes that make a person invisible to victim object detectors.

The created texture is to satisfy to three requirements: it has to be *effective*, *natural*, and *controllable*.

- *Effective* means that our clothes should hide the attacker from all angles, not like a patch that only works from a certain angle.
- *Natural* means that the clothes should not look suspicious to the human eye. That requirement prohibits us from simply generating an adversarial image and tiling the entire clothes with it since a repetition of a random image looks highly unnatural.
- *Controllable* means that we want to be able to generate a wide range of patterns depending on what the user (in our case—attacker) wants to see. This requirement is important because, as we will see in the literature review section, there are few

methods that provide an opportunity to control the generated adversarial pattern in the physical world.

1.5 Contribution

The pipeline proposed in this thesis satisfies the requirements defined above. The newly introduced method has the following main advantages:

- **Effectiveness and naturalness**. While this method does not beat the state-of-the-art ones in terms of how well the adversarial textures hide people from detection, this method allows to trade the adversarial effectiveness for achieving additional naturalness.
- **Speed**. Due to the fact that that we only use pretrained models, the method is significantly faster than other methods that require additional training.

CHAPTER 2 LITERATURE REVIEW

In this section, we will provide a broad overview of the existing methods and the references that motivated the project.

2.1 Digital adversarial examples

The first ones to discover adversarial examples were the authors of[4] who used gradients of the model to construct an imperceptible to the human eye noise that changed the prediction of the DNN.

Formally, a perturbation δ for a DNN f is adversarial if $f(x+\delta) \neq f(x)$ and δ meets certain requirements; most commonly, we aim to minimize δ and look for a perturbation in a certain radius ϵ with respect to norm L_p: $||\delta||_p < \epsilon$. In this case, $x + \delta$ is called an adversarial example.

After the initial discovery in[4], such adversarial examples have quickly gained popularity and have been studied in many areas and applications.

Over the years, many attacks have been created for all kinds of scenarios. The authors of[4] simply attacked a classifier by finding a small perturbation that causes a misclassification via minimizing the loss between the perturbed image and the target class. This method is called the L-BFGS Attack. Soon after, more advanced methods such as the popular Fast Gradient Sign Method (FGSM) from[5] that produces adversarial examples by moving against the gradients or the Projected Gradient Descent (PGD) attack from[6] that is basically an improved projected iterative version of FGSM. The DeepFool algorithm from[15] looks for an adversarial example with the smallest perturbation, and the Jacobian-based Saliency Map Attack (JSMA) from[16] finds the most influential pixels and changes their values to either maximum or minimum. [17] improved the L-BFGS method by switching to logits in loss and changing variables, the FGSM attack by a softer penalizing, and the JSMA method by shrinking the set of changeable pixels instead of growing it.

These attacks suggest that the attacker can get the gradients of the victim model and are called white-box attacks. This suggestion, however, does not necessarily hold true, as in a real life scenario, the attacker is rarely given access to the model. If the attacker can not get the gradients, they have to use different approaches; such scenarios are called black-

box attacks. Many methods were developed for this case as well. Ones of the first famous works in this field include[18] and[19] who studied adversarial sample transferability.

DNNs turned out to be so vulnerable to adversarial attacks that changing even one pixel might be enough to fool them[20].[21] explored a way to create a small perturbation vector that is able to fool the classifier for any input image. [22-23] created patches: localized adversarial noise that only covers a limited number of pixels concentrated in one area. Improvements to existing methods have been introduced, like the momentum-based iterative algorithm from[24] that boosts adversarial attacks, the translation-invariant attack method from[25] or Adam Iterative Fast Gradient Method from in[26] that improve the transferability between models, which is especially useful for the black-box scenarios.

Another classification of the attack according to their aims is the targeted versus untargeted classification. Untargeted attacks, like the classical versions of FGSM, PGD and L-BFGS, simply aim to thwart the predictions of the models in any way, while the targeted attacks aim to force the model to behave in a particular way. Normally, in order to create a targeted attack, one has to force the predictions to move towards the needed result instead of moving away from the true prediction, so instead of maximizing loss, one has to minimize it in a specific direction.

2.2 Physical attacks on detectors

In this work, we are mainly concerned with adversarial attacks on detectors. The final output of a detector consists of a bounding box, a confidence score and a class for every predicted object. By choosing different loss functions, an attacker can construct many attacks that aim at interfering with different output values of a detector. Mainly, there are two types of attacks on detectors: the ones that cause misclassification of boxes, like[27], and the ones that suppress predictions, like[28]. The adversarial examples that trigger misclassification were likely with a loss function that disturbed the classification loss of the detector, while the suppression attacks use losses that try to lower the confidence score of the predicted boxes.

There are also different ways to attack detectors depending on how an attacker constructs their adversarial structure. The most commonly used ones are adversarial patches and adversarial textures. In some cases, optical attacks are used as well. For this, an attacker has to change lighting or perform the attack with lasers. Even though such attacks are stealthy, they are hard to utilize in practice since they are susceptible to environmental

light.

Interestingly, since detector make their predictions partially based on the background, as shown in such papers as[29-31], there is a way to create an adversarial object that would suppress not only the closest boxes that overlap with the object, but also other predictions that might be far away from it. The authors of[32] claimed to have created a patch that can suppress all predictions on an input image. These results are, however, non reproducible since the authors have made several fatal mistakes such as not clipping their patch to the real color range and not freezing all the weights of the detector while training the patch. Other papers, however, corrected the errors and produced more realistic results. The authors of[28] went a step further and trained their patch with augmentations that allowed printing the patch and thus affecting a detector in the physical world. In order to produce such realistic augmentations, the authors applied the Expectations over Transformations approach from[33]. Unlike[32], who were mainly concerned with targeted attacks in a sense that they forced all predictions of the model leave their original positions to concentrate around the patch, the[28] created an untargeted attack that instead aims to hide as many objects as possible. Their patch is, however, pretty large as its side takes one fourth of the screen; moreover, as we have mentioned before, it neither takes into account the naturalness of its appearance nor retains its adversarial qualities when the camera angle shifts, unlike adversarial textures. Similarly, the authors of[34] perform a PGD attack on the confidence scores to hide all objects and produce real-life adversarial examples. They tried out different configurations and performed an ablation study that suggests that the classification loss, that was also used in[28], does not improve the patch.

Some other approaches aim to hide or suppress only a certain range of predictions. For example, the authors of[10] create a patch that makes the detector blind to all, even non-overlapping, objects of a certain class. Similarly,[11] create an adversarial patch that hides vehicles from automatic detectors.

A wide range of works is aimed at perturbing stop signs since they are crucial for self-driving cars[7,35-38]. Among them,[38] is an optical attack: the attacker casts a shadow over a road sign to distort the predictions of the detector. Another direction is to thwart facial recognition with glasses[39-40]. We are, however, mainly interested in using adversarial clothes[14,41-45].

2.3 Generating adversarial examples with GANs

Generative Adversarial Networks (GANs)[46] are a powerful tool that is often used for generating samples for all kinds of datasets. The framework is similar to a min-max two-player game, since the pipeline of a GAN consists of two models, the generative and the discriminative one. While the generative model is responsible for producing samples, the discriminative model trains alongside and tries to predict whether a given sample came from the true distribution or was generated by the other model. Such a game-like process encourages both models to improve.

Even though GANs have been applied to many areas in Deep Learning, for this project, we are only concerned with a very specific task for GANS—generation of adversarial examples.

To the best of our knowledge, the first ones to use GANs for generating adversarial samples were the authors of[47]. However, their work was focused on generating examples for malware. [48] created AdvGAN, a conditional GAN that can generate adversarial examples for images. AdvGAN directly generates an entire image. [49] proposed a similar GAN, but, unlike[48], they did not assume that the norm of the perturbation is small, so their solution is more general.

Closer to our task,[50] propose an algorithm that improves EoT and uses a GAN to generate printable adversarial images. [45,51-52] use GANs to generate naturalistic physically-realizable patches. The approach suggested in[45,52] is quite different from the other paper: they train the generators to map latent variables to a realistic patch and then traverse the latent space and find an adversarial example there. [53-54] also use GANs to generate a realistic patch, but specifically for autonomous driving tasks. More specifically,[53] focus on attaching patches to road signs, while[54] generate fake advertisements posters.

Another paper that is very important for this project is [8]. Not unlike the previously referenced[45,52], the authors of this paper traverse a certain range around the starting point in the latent space in order to generate an adversarial patch for suppressing the prediction closest to the patch. To be more precise, the authors generate a patch for attaching it to the clothes of a person and thus lowering both the confidence and the correct class probability for the box associated with the person who is wearing these clothes. Our work is different for two reasons: first, we generate textures instead of patches; second, we intend to use a diffusion model instead of the GAN models that were used in this paper.

2.4 Diffusion models

Diffusion probabilistic models (referred to as "diffusion models" for brevity) were originally introduced in[55] for the denoising task. During training, in the forward process, an input image is gradually turned into Gaussian noise. During the reversed process, a U-Net-based[56] model is trained to predict the noise for each step in the forward process, this denoising the result back to the original image. Thus, by passing a latent variable (Gaussian noise) to the reversed process of a trained model, one can generate a sample from the learned distribution.

Immediately after the paper was published, the models became very popular on all kinds of tasks. [57] developed a unified framework for four translation CV tasks: colorization, inpainting, uncropping, and JPEG restoration. Latent diffusion models from[58] work on a compressed latent space and achieve a state-of-the-art results on various generation tasks. [59] use text conditioning to produce high-quality samples. [60] presented a text-to-image photorealistic diffusion model. Diffusion models have also been applied to super-resolution[61-62], classification and regression[63], segmentation[64-65], video generation[66-67], and many other tasks. Despite the rapidly growing interest in diffusion models and their quick evolution, the field remains unexplored.

As far as we know,[68] were the first to apply diffusion models for generating adversarial examples. The authors use a diffusion model and optimize the latent variable instead of the generated picture, which makes their approach an unrestrained attack in a sense that they do not force the norm of the prediction be close to the unattacked sample. Their model, however, was only trained for the digital scenario, just like all earlier works on adversarial attacks, while we are interested in a realistic adversarial patch for the physical world.

2.5 3D modeling

Instead of optimizing a patch or a texture, one can turn to optimizing the 3D textures of objects. For example, the authors of[33], who were the first to create an adversarial 3D example—a turtle. In order to do this, they render multiple poses for the 3D object. The authors of[69] create 3D adversarial examples using the meshes from the ShapeNet dataset[70] that contains objects like table, chair and plane. [9,13,71-72] also render photorealistic vehicles covered with generated camouflages. However, all these approaches consider rigid objects, while we are interested in non-rigid objects like fabric.

While training the patch on the Inria Dataset[73], the authors of[14], the paper on adversarial textures, attach a patch to the center of the box corresponding to the person after applying EoT and TPS. In reality, however, the clothes and people are non-rigid, meaning that the pattern on the clothes will deform as they move, and the deformation is very difficult to describe. This problem was addressed by the authors of our most closely related work,[74]. In this paper, the authors use the Voronoi diagrams to parameterize camouflage clothings, then augment them to account for possible movements and distortion of the fabric and render the 3D models. We intend to leverage a diffusion model instead of the Voronoi diagram parameterization to broaden the range of the patterns for the clothings.

2.6 Closest works

Adversarial Textures The authors of[14] introduced Toroidal Cropping based Expandable Generative Attack (TC-EGA), a generative method for creating adversarial textures. The authors create a rather complicated pipeline for generating textures that includes two generative stages. The generative network, unlike those used in GANs, only consists of convolutional layers, which allows us to input latent variables of any size. In the first stage, this generative network is trained, while in the second stage, after searching for the best local pattern with applying toroidal cropping, we can tile this pattern and generate a texture of the required size. The loss function basically consists of three components: the usual expectation for lowering large confidence scores, a total variance (TV) loss[39], and the information objective loss. The last component is responsible for maximizing the mutual information between the latent variable and the generated patch. In addition to EoT, the authors of[14] employ the Thin Plate Spline technique[75]. It helps create realistic augmentations like wrinkles on the fabric.

3D zh paper

Guidance

CHAPTER 3 METHODOLOGY

3.1 Motivation

As we have established previously, it is difficult to control the generated adversarial patterns with existing methods. By control here we mean a possibility to lead our generation to a particular direction according to the attacker's will, thus being able to generate a wide range of possible patterns. So, the main idea of our methodology is to use the remarkable generative abilities of diffusion models to produce natural-looking adversarial clothes. Moreover, since we want our method to be zero-shot, that is, to use pretrained models only, ideally, we would like to be able to use the DM to find a trajectory that would lead us to adversarial samples hidden inside the areas of distribution of natural-looking images. Here, trajectory denotes the denoising path the DM choses when gradually producing the final result starting from random noise. So, we would like to be able to change the default generation trajectory of our DM to push it towards enhancing the adversarial effects of the image.

3.2 Pipeline

3.2.1 Introduction: Quick Overview

We have two models involved in the pipeline: a diffusion model and a detector model. As input, we receive a prompt from the user, and as output, we have to generate adversarial clothes with a pattern as close to the original generation of our stable diffusion model as possible. Again, in our case, adversarial means that the clothes can help to avoid being detected altogether instead of just making the detector assign an incorrect class to the detected bbox. At the same time, we also need to ensure that the generated pattern is still effective in the physical world, since our ultimate goal is to be able to print the clothes, put them on and demonstrate how they can be used to avoid detection systems.

First, we generate a latent that would correspond to the input image according to our diffusion model. This latent can be either random or somehow modified according to the user's wishes. Again, one of the main advantages of our method is that, unlike other methods, it is able to trade adversarial effectiveness off in order to create natural-looking patches that can be controlled by the user with the use of prompt and initial conditioning,

which is the chosen latent, our starting point in the noise distribution. It is essential that the prompt makes sense according to whatever the user is trying to generate. That is, even if the user was to only provide the initial latent, they still need to ensure that the prompt does not contradict the task, and rather help to push the generation into the right direction.

After that, we modify the sampling process of the diffusion model, adding "detector guidance" to the update in order to ensure that our patch helps us avoid the detector. In order to obtain this "detection guidance", we unfold the latents into the corresponding patch and create clothes that are used in a chosen pipeline to estimate the efficiency of our detector, and change the update direction based on the detector's output. Here, we have used two types of pipelines for calculating detection guidance: 2D and 3D.

Other guidances, like the "fixation guidance", are also used to ensure the naturalness of our final generation.

In the end, we once again obtain the final image from the latents and make adversarial clothes out of them using the same strategy.

3.2.2 Models

In order to understand the pipeline in more detail, first, we need to look at the component parts closely.

Two models are used in this pipeline: a Stable Diffusion model (SD) for generating the patch, and a YOLOv2 model (YOLO) to ensure that the generated patch is adversarial with respect to detection. Naturally, it is also possible to choose other diffusion and detection models. We stopped at SD mainly because of its speed and large variety of results, while YOLO was chosen to compare against other baselines as one of the most widely used models.

All weights of the models involved in the process are frozen, but the latents that are used and optimized by the SD have their gradients unlocked in order to enable adversarial guidance.

3.2.2.1 Stable Diffusion

Introduction On a high level of understanding, a pretrained SD model, just like any other DM, generates trajectories from the noisy space to the meaningful for us humans space by gradually removing noise from this random starting point. Figure 3.1 demonstrates three examples for such trajectories for the SD model that was used in this work. The prompt is fixed for all these examples: "A beautiful goldfish swimming through the

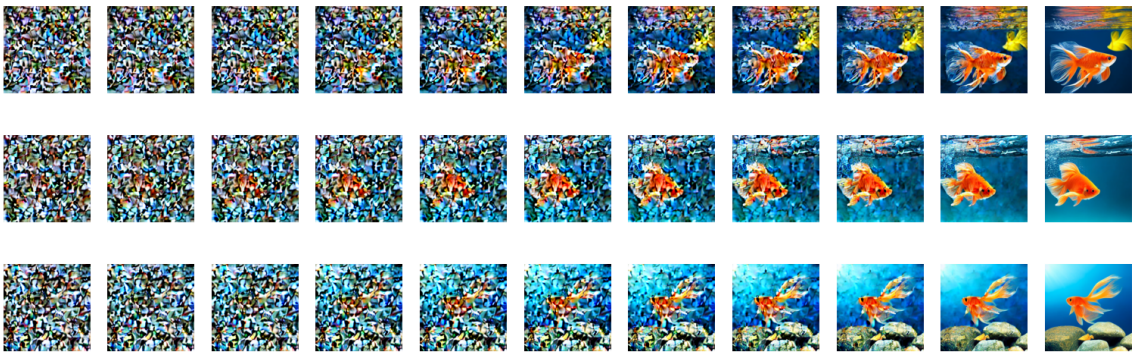


Figure 3.1 Example of a Stable Diffusion (MiniSD) generation. Prompt: “A beautiful goldfish swimming through the ocean”. 50 steps, each 5th step drawn, three random noises used as starting points for the rows.

ocean”. The first column depicts three random noises that were used as starting points for the corresponding three trajectories that show the denoising process. In this process, as we look at the pictures in each row from left to right, we observe the model gradually remove the noise to arrive at different finishing points. All three final generations in the last column, however, satisfy our conditioning requirement, which is the given prompt.

So, during the training process of a DM, noise is added to input images, and the main part of the model, normally its U-Net component, is training to predict the noise that was added at each step. That way, during sampling, we are able to use this pretrained U-Net to go in the opposite direction of removing noise.

Another important part of the process is conditioning, which in case of a SD model is usually added in the form of a text prompt, or more rarely a semantic map. The conditioning is inserted into the U-Net with the use of attention, and the text itself is encoded with a CLIP model.

Finally, a SD model is a latent diffusion model. Since we often want to work with large images that might take too much GPU memory, this class of models works on optimization in a latent space instead of the original pixel space of actual images. In order to work in this space, however, we have to be able to go into the space and leave it at will. Typically, a Variational AutoEncoder (VAE) model is used for this purpose: its encoder allows the transition into the latent space, while its decoder returns latents to the corresponding images in the real pixel space.

Formalization Now, let us turn to a little more low-level explanation with formulas and illustrations.

Figure 3.2 depicts the pipeline in the scheme from the original paper[58]. The leftmost

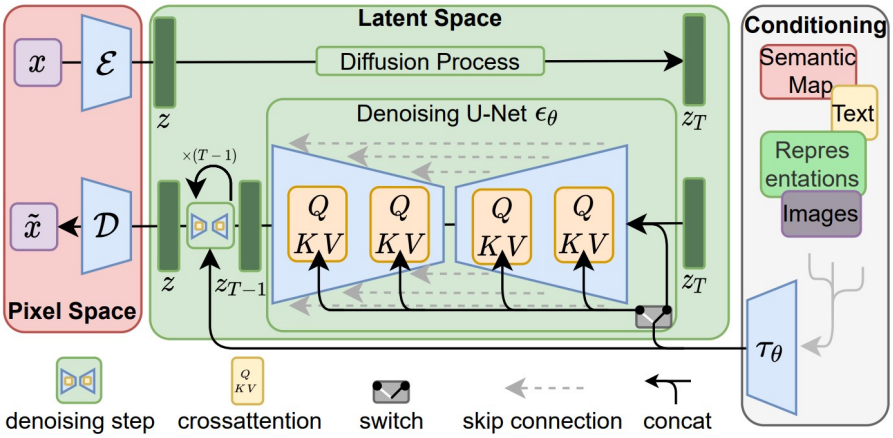


Figure 3.2 Stable Diffusion Training Pipeline.

red part indicates the pixel space with real images, and the green part—the training process that is happening in the latent space. The transitioning between the spaces is possible with the use of a VAE with encoder $\mathcal{E}$ and decoder $\mathcal{D}$. That is, a real image x can be translated into a latent z in the latent space by applying the encoder $\mathcal{E}$ of this VAE, while the reverse process is carried with the decoder $\mathcal{D}$:

$$z = \mathcal{E}(x), \tilde{x} = \mathcal{D}(z)$$

$\tilde{x}$ is a reconstructed original image x. During training of a SD, the weights of its VAE model are fixed.

The forward diffusion process gradually adds noise to our original latent z in T steps, and the model trains to predict the previous latent z_{t-1} for all z_t. So, our SD is actually learning the original data distribution by learning how to reverse a fixed Markov Chain of length T. A denoising U-Net model ϵ_θ is used to predict the noise that can be used to obtain z_{t-1}. The text prompt is included in the U-Net with help of cross-attention as shown on the picture: or conditioning y is turned into its representation $\tau(y)$.

After training is completed, we obtain a U-Net model that can denoise our latents. During sampling, we take a prompt and an initial latent, be it random or somehow conditioned, and then run the trained U-Net for several iterations to denoise it, apply our decoder $\mathcal{D}$ and obtain our final image $\tilde{x}$. Theoretically speaking, each iteration in the denoising sampling cycle improves our final result; in practice, however, for a simple generation without specific guidances, a relatively small number of iterations ($\lesssim 1e2$) is enough to obtain a meaningful realistic result. On the contrary, since generation is performed with a fixed scheduler, it is impossible to run the cycle for an infinite number of iterations without altering the sampling process of the given pretrained model.

Guidance Apart from conditioning, there exist other methods that push the generation towards a particular desired direction. Let us formally describe guidance, a method that can be used to ensure that our generated image is not random in the space of all generations that satisfy the prompt, but rather is somehow altered. Let us take a look at the update formula for the latents in the diffusion sampling cycle in the simplest form starting from $z_T \sim \mathcal{N}(0, 1)$:

$$\hat{\epsilon}_t = \epsilon_\theta(z_t; t, y)$$

where z_t is the latent at step t and y is our condition, typically a prompt. As we have previously stated, in this formula, we are predicting an estimate of the noise that was added to z_{t-1} in order to create z_t:

$$z_{t-1} = \text{scheduler}(z_t, \hat{\epsilon}_t, t)$$

where scheduler depends on our sampling method; our SD uses pseudo numerical methods for diffusion models.

It was, however, proven that for improving the quality of the image and for a better conditioning it it better to use classifier-free guidance[76]:

$$\hat{\epsilon}_t = (1 + s)\epsilon_\theta(z_t; t, y) - s\epsilon_\theta(z_t; t, \varnothing)$$

where s is a hyperparameter that determines the strength of classifier-free guidance and is typically set to 7.5. Without this guidance, the conditioning is rather poor and the result might not correspond to our prompt well. Let us also recall that since DMs in general are score-based, $\epsilon_\theta(z_t)$ estimates the score function for the intermediate noisy distributions between z_t and z_{t-1}:

$$\epsilon_\theta(z_t) \approx -\sigma_t \nabla_{z_t} \log p(z_t)$$

Now we can take a look at sampling even in a more general light and learn how to add the an arbitrary guidance to the formula:

$$\hat{\epsilon}_t = (1 + s)\epsilon_\theta(z_t; t, y) - s\epsilon_\theta(z_t; t, \varnothing) + \upsilon\sigma_t \nabla_{z_t} g(z_t; t, y)$$

here, g can be any energy function that guides our generation in a meaningful way, υ is self-guidance weight and σ_t converts the score function to a prediction of ϵ_t.

MiniSD It is also important to note that due to our GPU resources limitations, we were unable to use the standard SD and used MiniSD from Hugging face instead; compared to its standard version, the MiniSD can generate images at resolution 256 without a notice-

able loss in quality.

3.2.2.2 YOLO

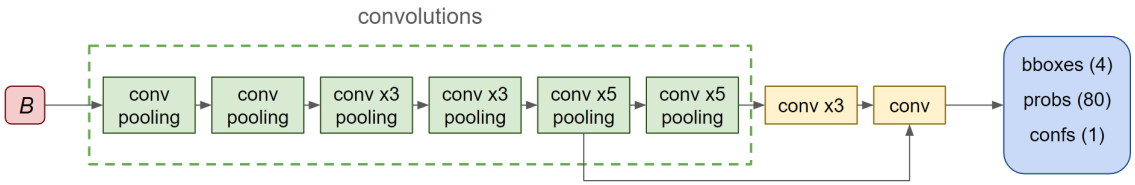


Figure 3.3 Darknet-19, detection pipeline.

Another important model that we use in our pipeline is YOLO. Figure 3.3 shows the architecture of the Darknet-19 that was introduced in the paper as a model that can be used for classification and detection. More specifically, in general, only the green part of the scheme is Darknet; the depicted architecture is a Darknet modification that is used for detection. When the yellow follow-up convolutions are replaced with another single convolution, the model becomes its classification Darknet-19 modification; they are trained jointly in the original paper.

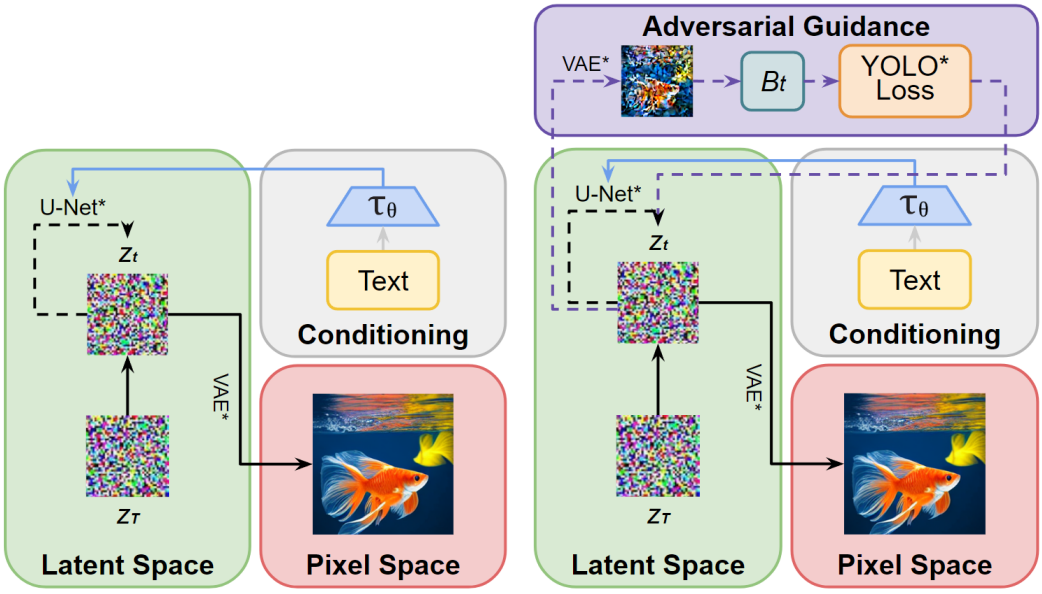
The pipeline itself is very simple. In the simplest case, for a detection Darknet, a batch of square input images $\boldsymbol{B}$ of size $3 \times 416 \times 416$ goes through the convolutions and poolings until the final predictions are obtained. As it has been mentioned previously, the output data of YOLO consists of three parts: bboxes that show the positions of the predictions, probabilities for all classes, and confidences in predictions. Since each bbox is defined through its position and size, all of them are predicted in four values. YOLOv2 specifically was trained for the COCO dataset[77], the Darknet model returns probabilities for 80 classes. The confidence, or the objectiveness score, predicts the IoU between the proposed bbox and the ground truth label. The architecture also contains a shortcut as shown on Figure 3.3 to make the model take fine-grain features into consideration. Each convolutional layer in the scheme is followed by batch normalization as regularization to speed up convergence and stabilize the training process.

Another trick that is used in YOLO is anchors. After the batch of images $\boldsymbol{B}$ goes through all the convolutions, we obtain feature maps for the images of size 13×13, which can be understood as positions on the images. For each of these positions, our model learns to predict offsets for 5 anchors that denote most cluster centers for object sizes of training dataset. That is, the authors chose the number of clusters for all object sizes in the training data as 5 (the number chosen empirically), and then, instead of actually

predicting the size of the object in the grid, we predict offsets to these anchors; centers of objects, that is, their positions, are also predicted as offsets from the central positions in the grid. So, to sum up everything that has been mentioned, the final output has size batch_size $\times 13 \times 13 \times 5 \times (4 + 1 + 80)$.

3.2.3 General Pipeline

In this section, we will introduce the general pipeline we used for our adversarial generations.



(a) Original SD (b) Modification: adversarial guidance

Figure 3.4 Stable Diffusion: original and modified pipelines.

Figure 3.4(a) depicts the original SD pipeline that was already introduced in Figure 3.2, albeit in a simplified form with the insides of U-Net hidden. The sampling cycle that goes on for T iterations is indicated by the gray dashed line: the U-Net model, taking into account the conditioning, predicts noise that should be removed from the latents at each step. After the cycle is completed, we leave the latent space for the pixel space with the help of VAE's decoder $\mathcal{D}$.

Figure 3.4(b) introduces changes to this process. Overall, the sampling pipeline stays the same, but we add a correction to our update direction in the form of adversarial detection guidance. That is, during the sampling process, we want to guide our latent not only towards generating a specific image according to our prompt conditioning, but also make it adversarial. Let us recall the guidance formula in general:

$$\hat{\epsilon}_t = (1 + s)\epsilon_\theta(z_t; t, y) - s\epsilon_\theta(z_t; t, \varnothing) + \upsilon\sigma_t\nabla_{z_t} g(z_t; t, y)$$

In order to define adversarial guidance, we have to understand what the energy function g looks like in this case. In essence, the energy function g is some kind of adversarial detection loss; more formally,

$$g(z_t; t, y) = s_t^{adv} \cdot \text{adv\_loss}(z_t, \mathcal{D}, \text{YOLO}, B_t)$$

where:

- s_t^{adv} is the adversarial guidance coefficient at step t. It makes sense to change the intensity of adversarial guidance through the sampling process, so the entire set of adversarial guidance coefficient, $S^{adv} = \{s_t^{adv}\}_1^T$, is an important hyperparameter that will be later referred to as adversarial guidance scheduler;
- z_t is intermediate latent that can be decoded into the corresponding intermediate noisy image $\tilde{x}_t$ in the pixel space with VAE decoder: $\tilde{x}_t = \mathcal{D}(z_t)$;
- YOLO is our object detection model that is the key to obtaining the adversarial guidance;
- $B_t = \{X_t, L_t\}$ is a batch of object detection images consisting of images X_t and their labels L_t;
- adv_loss is our current method to extract the component of the real YOLO loss that is relevant to the task. In our case, since we are working on evasion attacks, it will be the objectiveness score; in the general case, it can be any other meaningful part of the YOLO output, such as, for instance, the probabilities of predicted classes or bboxes.

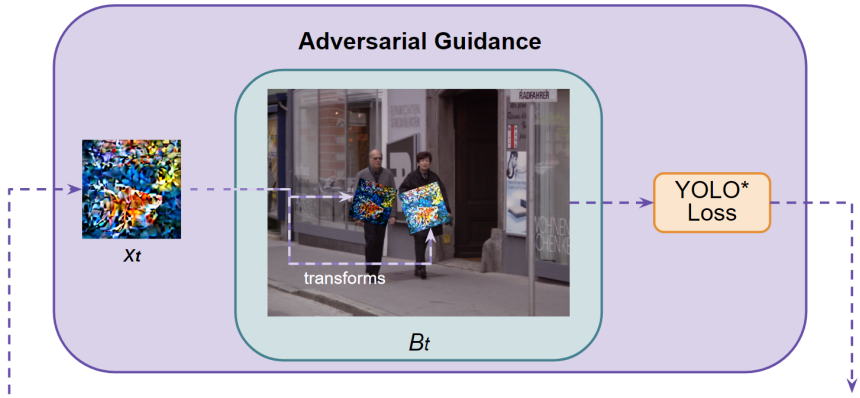
Since B_t mostly depends on the chosen pipeline type, further explanations of the formula will be introduced in the appropriate sections below.

Overall, the main idea is to alter the update direction of the conditioned update to arrive at an adversarial sample, much like classifier guidance from[78].

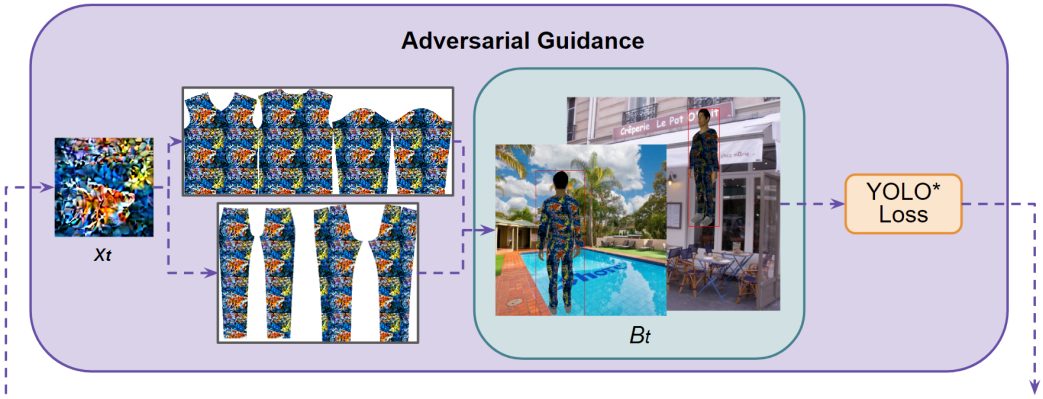
3.2.4 Adversarial Guidance Pipelines

We have tried two techniques for calculating YOLO loss based on different approaches to obtaining data B_t and applying x_t to this data. The pipelines are called 2D and 3D. In the first pipeline, we use images from an existing labeled dataset with people, Inria Person[73]. x_t are attached to the people from batch B_t as patches. With 3D pipeline, the background dataset[74] does not contain people; instead, human figures are dressed in clothes with tiled generations and rendered with certain augmentations.

The pipelines are shown on Figures 3.5(a) and 3.5(b).



(a) 2D



(b) 3D

Figure 3.5 Adversarial Guidance Pipelines

2D In the case of 2D pipeline, intermediate partially denoised images x_t are attached to all people from every image X_{tm} in batch $B_t = \{(X_{tm}, L_{tm})\}_1^M$ consisting of M images. We are trying to lower the highest confidence of relevant bboxes, so the energy function in this case is:

$$g(z_t; t, y) = s_t^{adv} \sum_{m=1}^{M} \max \left[\text{yolo\_loss}(\tilde{X}_{tm}, L_{tm})_{\text{obj}} \right]$$

where $\tilde{X}_{tm}$ is X_{tm} with attached augmentations of x_t, yolo_loss is the regular loss for the YOLO, and $_{\text{obj}}$ signifies that we only look at the objectiveness (confidence) scores of the predictions, since we are performing an evasion attack and want to not be detected.

In this case, augmentations consist of the standard simple transformations like adjustments of brightness, angle, contrast, and adding random noise, and also applying Thin plate splines (TPS, see Section 3.2.5 below). All of the applied transformations are differentiable.

3D In the case of 3D adversarial guidance pipeline, we first tile our generated texture x_t to cover a 3D model's clothes C_t. Afterwards, using a 3D rendering module of Python, we render a 3D model that we have prepared in advance with help of[74]. The models are rendered on random backgrounds from the dataset. We also sample light and camera angles to ensure the effectiveness of our attack in the real world. This, as well as the 3D TPS, TopoProj (see Section 3.2.5), and shifting of the initial position of x_t on C_t, are our augmentations.

The energy function is essentially the same as in the 2D case, albeit it is important to note that in this case, labels L_{tm} are not provided in the dataset, but rather calculated according to the current sampling and rendering strategy that changes over time, and images X_{tm} contain rendered figures as shown on Figure 3.5(b).

3.2.5 TPS and TopoProj

One of the most important augmentations we use in this task is TPS.

TODO

Another important technique that was introduced in[74] is TopoProj.

TODO

3.3 Appearance Restoration

As we have stated previously, we want our adversarial evasion clothes to be controllable: that is, we are able to obtain not just a random adversarial pattern, but rather be able to make it closer to something we want to see specifically. Let us say we have a simple non-adversarial sampling trajectory of our SD $\hat{z}_T \dots \hat{z}_0$ that leads us to our desired image $\hat{x} = \mathcal{D}(\hat{z}_0)$. Now we are trying to generate x, an adversarial texture for clothes that at the same time is close to our original $\hat{x}$. Firstly, we make a step according to our U-Net prediction; secondly, this direction is modified with adversarial guidance. The trajectory, however, can change significantly compared to the original one, so x can be very different from $\hat{x}$ if the adversarial scheduler S^{adv} consisted of large coefficients.

So, we have to propose a method to preserve the original appearance, even maybe at the cost of adversarial effectiveness. In order to do that, we insert additional appearance restoration guidances that push our generation in the direction of the original sampling trajectory. We have tried guidances dictated by two different losses that we will introduce in this section.

3.3.1 Appearance Constraint Loss

Since the attention maps of U-Net contain information about the position and the form of objects and the activations from appropriate layers store the appearance of the objects, it is possible to manipulate several properties of the objects using the representations stored in the U-Net layers.

Following[79], let us define the appearance of objects in terms of the information stored in U-Net. Let us denote an intermediate attention map from layer n on timestamp t as $\mathcal{A}_{t,n} \in \mathbb{R}^{P \times H_n \times W_n}$, where P is the length of the prompt (one word is one channel starting with position 1; empty tokens are added at the end until we reached the intended length P). Similarly, activation (features) will be denoted as $\mathcal{F}_{t,n} \in \mathbb{R}^{F_n \times H_n \times W_n}$.

According to[80], attention maps store information about the shape and the position of the objects from their corresponding channels. That is, by looking at the attention maps $\mathcal{A}_{t,:}$ at timestamp t, we can see the shape of all objects; if we want to see the shape of p-th object from layer n, we should look at $\mathcal{A}_{t,n,p}$. The bigger the value in the corresponding channel, the more related the pixels are to this word. Omitting timestamps and layering indexing, the following formula can be used to calculate the shape of object p:

$$\text{shape}(p) = \mathcal{A}_p$$

In practice, attention maps $\mathcal{A}$ are thresholded to account for background noise. It is possible to change the shape of object by guiding these maps towards those of another generation or provided by the user, but in our case, we are interested in the entire appearance of objects that can be expressed as:

$$\text{appearance}(p) = \frac{\sum_{h,w} \text{shape}(p) \odot \mathcal{F}}{\sum_{h,w} \text{shape}(p)}$$

This formula makes sense because shape(p) creates a mask that denotes the form of object p and activation maps represent local appearance.

To use appearance constraint as guidance, we need to define the energy function g. As we have mentioned in the introduction part of this section, we already have our original generation $\hat{x}$ and latent trajectory that leads to it. That is, we can run the SD to obtain this trajectory $\hat{z}_T \dots \hat{z}_0$. While running this process, we can store all the intermediate U-Net attention maps and activations to be able to reconstruct the original $\widehat{\text{appearance}}_t(p)$ for object p on timestamp t of this original run. With this, we can use attention constraint guidance as the gradient obtained from the l1-loss between the original appearance and

the current appearance of object p:

$$g(t, p) = s^{ap} \cdot \text{l1-loss}(\text{appearance}_t(p), \widehat{\text{appearance}}_t(p))$$

where s^{ap} is the corresponding guidance coefficient.

In our case, we want the entire generation to be close to the original, so we add the guidance to all words in the prompt as $\sum_{p \in P} g(t, p)$.

3.3.2 Attention Constraint Loss

Even though the method helps to fix the appearance of the generation, fixing all words is a prompt has little physical meaning. For example, for a prompt "a beautiful goldfish swimming through the ocean", it is easy to imagine what objects like "goldfish" and "ocean" do look like, but it fixing parts of prompt like "a"/"through" or "swimming" is unjustified. This is why we suggest to only fix self attention maps that restrict the affinities between the features. Self-attention maps control the layout of the image in the lower layers and capture details on higher, so guiding them will guide the entire image to the original $\hat{x}$.

Similarly to appearance constrain loss, we have to run the original trajectory $\hat{z}_T \dots \hat{z}_0$ and store all self-attention maps $\hat{\mathcal{A}}_t^{self}$ on all timestamps t.

The guidance can be defined as l1-loss for the original and newly generated self-attention maps:

$$g(t) = s^{at} \cdot \text{l1-loss}(\mathcal{A}_t^{self}, \hat{\mathcal{A}}_t^{self})$$

TODO for this and previous subsections: illustrations (?)

3.4 Final Algorithm for 3D Pipeline

In our case, for the g energy function, we chose YOLO loss. Again, the physical explanation of this guidance is that we are trying to maximize YOLO loss for the predicted people, so that ultimately our patch helps in avoiding detection by this YOLO:

$$\nabla_{z_t} g(z_t; t, y) = s_t^{adv} \sum_{m=1}^{M} \nabla_{z_t} \max \left[\text{yolo\_loss}\left(\tilde{X}_{tm}, L_{tm}\right)_{\text{obj}} \right]$$

So, compared to the standard SD sampling, we allow latent z_t to accumulate gradients and add this gradient to our update $\hat{\epsilon}_\theta$. In this section, we will provide the adversarial algorithm for our pipeline that is fully shown in Figure 3.6 on the example of the combination of two guidances: adversarial and attention constraint.

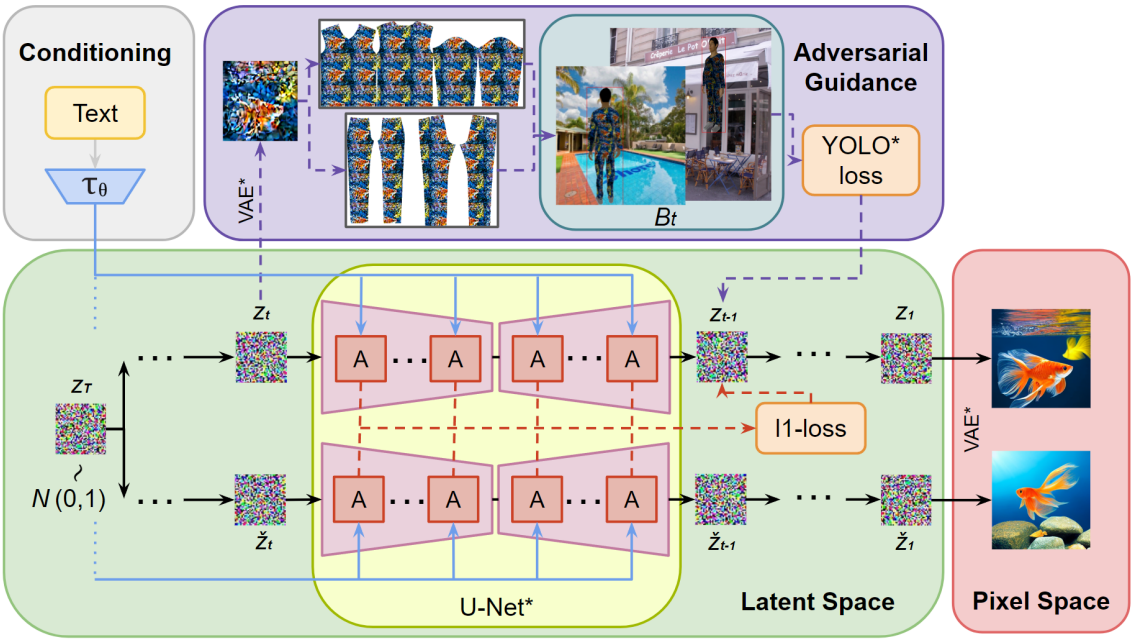


Figure 3.6 Final pipeline with both adversarial and fixation guidances.

Algorithm 3.1 Sample adversarial x based on non-adversarial $\hat{x}$

Inputs: Prompt $\mathcal{P}$, number of SD steps T, adversarial scheduler S^{adv}, attention constraint coefficient s^{at}, adversarial dataset DS

$z_T \sim \mathcal{N}(0,1)$ ▷ initial noisy latent
$P_{emb} \leftarrow \tau(\mathcal{P})$ ▷ CLIP prompt encoding
self_att_maps ← []
for $t \leftarrow T$ to 1 **do**
  self_att_maps.register(U-Net.self_attention) ▷ saving attention maps for non-adv sampling
  $\epsilon \leftarrow$ U-Net(z_t, P_{emb})
  $z_{t-1} \leftarrow$ update(z_t, ϵ)
end for
for $t \leftarrow T$ to 1 **do**
  $\epsilon \leftarrow$ U-Net(z_t, P_{emb})
  $x_t \leftarrow \mathcal{D}(z_t)$ ▷ decoding latents into intermediate images
  $X, L = DS$.next()
  $\tilde{X} \leftarrow X$.attach(x_t) ▷ obtaining augmented image for YOLO
  $\epsilon \leftarrow \epsilon + S_t^{adv} \nabla_{z_t}$ YOLO-loss$(\tilde{X}, L)$ ▷ adv guidance
  $\epsilon \leftarrow \epsilon + s^{at} \nabla_{z_t}$ l1-loss(U-Net.self_attention, self_att_maps[t]) ▷ appearance guidance
  $z_{t-1} \leftarrow$ update(z_t, ϵ)
end for
$x \leftarrow \mathcal{D}(z_0)$

Output: Adversarial x

The method is shown in algorithm 3.1 in pseudocode. Just as the usual SD pipeline, we start by generating the initial noisy latent z_T (in case it was not provided by the user in the input data). We also have to use our text encoder, typically CLIP, to create prompt embeddings P_{emb}.

After that, we run the first sampling that is non-adversarial. We need to do this since

we intend to save attention maps from all intermediate steps to be able to guide our adversarial trajectory in the direction of the original path. So, during the first cycle, we save all needed information from U-Net, which, in case of attention constraint guidance, are self-attention maps only. The cycle itself is standard for SD: we generate noise prediction ϵ with the U-Net model and then use it to update the current latent z_t. In the actual pipeline, we also use classifier-free guidance described in Section 3.2.2.1 by running the U-Net model for two inputs simultaneously, but this part is omitted in the pseudocode for its irrelevance to this method as it is used for improving results in general rather than its adversarial effects.

After the initial self-attention maps are stored, we can start running the adversarial sampling. Now, on each iteration of standard SD sampling, we perform the following additional operations after obtaining noise prediction ϵ on step t:

1. Use our frozen VAE model (decoder D from Figure 3.2) to obtain $\hat{x}_t$;
2. Transform this $\hat{x}_t$ into clothes $\hat{c}_t$ for our 3D model;
3. Render human figures dressed into the clothes $\hat{c}_t$ into scenes from the background dataset while performing augmentations to ensure effectiveness in physical attacks to create images $\tilde{X}$ for YOLO;
4. Calculate the YOLO detection loss for the scenes with our augmented $\tilde{X}$ and their labels L;
5. Perform adversarial guidance: add gradient ∇_{z_t} from this loss to the noise prediction ϵ with coefficient S_t^{adv};
6. Calculate the l1-loss between the current self-attention maps of U-Net and those we have saved from the original run;
7. Perform attention constraint guidance: add gradient of the loss in the previous step to the noise prediction ϵ with coefficient s^{at}.

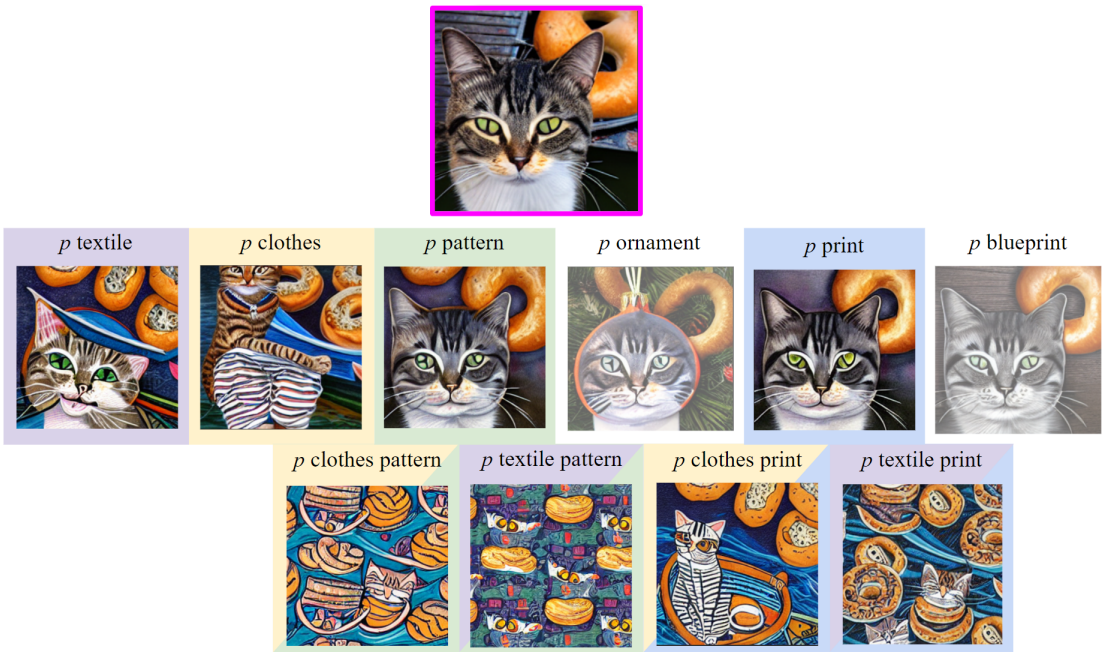
Each iteration of the cycle is finished with updating latent z_t with obtained guided noise ϵ.

After the sampling cycle is completed, we can obtain our final adversarial x as the decoded final latent: $x = D(z_0)$.

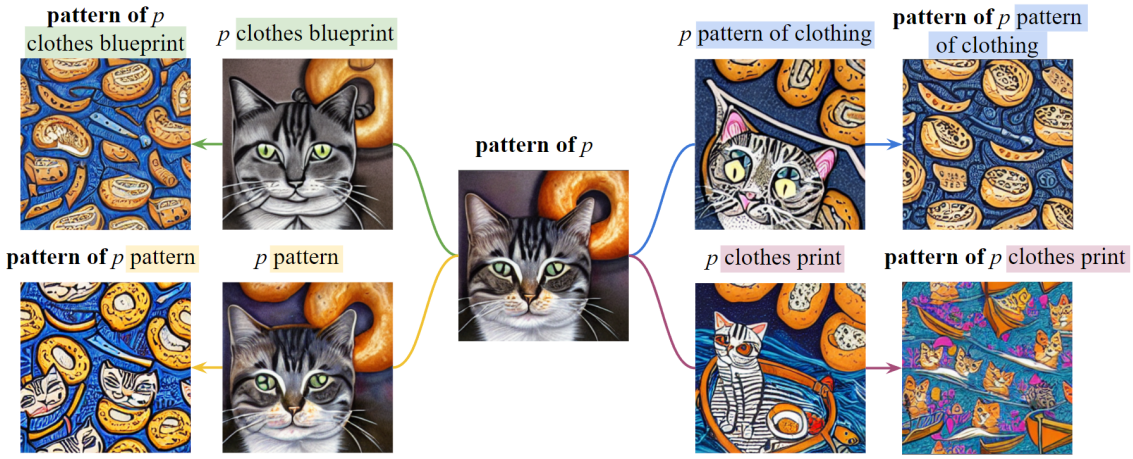
3.5 Tiling

So far, we have not discussed why our method is natural considering the fact that we simply tile generated pictures are not necessarily tileable. However, the section with ex-

periments will show examples of generated patches that can be tiles almost seamlessly. In this section, we describe our original method to achieve this effect without any additional guidance.



(a) Two levels of prompt extensions: the second level (the last row) contains patterns that can be tiled.



(b) Stacking specific lines increases the effect.

Figure 3.7 Prompt modifications for prompt p "A cat on a boat with a bagel" (the highest image in a dark pink box).

Prompt Modification The SD model is trained on a large dataset LAION[81] that consists of roughly 5 billion images, so the model manages to learn many nuances of the text-image relationship, which can be used by modifying the input prompt. It turns out that it is possible to modify the prompts to make the SD produce textures that can be tiled

almost seamlessly.

Figure 3.7(a) shows several such examples. In the upper row, we have the original generation for a random prompt — "A cat on a boat with a bagel". In the second row, we extend this prompt by adding one word that is related to our task, but do not obtain tileable textures just yet. In the last row, we show some of the combinations of the prompt extensions above. Three of them, "*p* clothes pattern", "*p* textile pattern", and "*p* textile print" achieve the needed effect: if we put any of these generations side by side with itself, the final pattern will be smooth. However, only a limited number of such combination works, and, for example, combining words "*p* textile" and "*p* pattern" with "*p* ornament", and "*p* blueprint" yield no needed results, and the generated patterns for these combinations in essence look like "*p* clothes print" in the last row, which can not be tiled.

Interestingly, stacking these lines, we can enhance the effect, turning generations even for the non-effective first level extensions tileable. Examples of this are shown on Figure 3.7(b). This set of examples for the same prompt demonstrates the effectiveness of the prompt command "pattern of *p*". By itself, as we can see from the image in the middle, the word does not make our random prompt generation continuous. The same goes for four other prompt extensions that are connected to the image in the middle: "*p* clothes blueprint" (green), "*p* pattern" (yellow), "*p* pattern of clothing" (blue), and "*p* clothes print" (purple). However, if we combine the original "pattern of *p*" with any of the four aforementioned extensions, we will obtain the outer set of four images that can be put side by side to create continuous patterns.

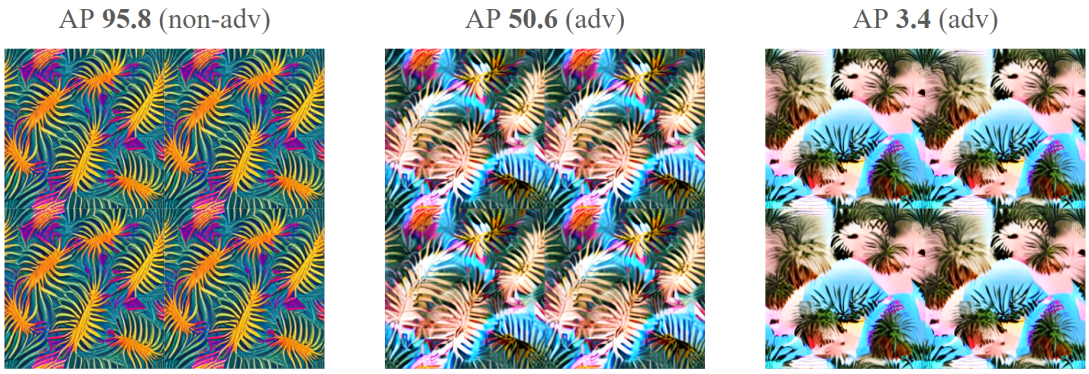


Figure 3.8 The generations are still tileable as we apply stronger adversarial guidances.

Tiling vs Adversarial Effect Importantly, this effect does not diminish as we add adversarial guidance to our generation. To demonstrate this, we show three generations for the same prompt, "palm clothes pattern", in Figure 3.8, which also shows what we mean

by “tileable” images. Here, in the left part, we have the original non-adversarial generation for the prompt repeated 4 times to demonstrate that the tiling has indeed almost negligible seams on the joints.

The next two parts of the figure show examples of adversarial and appearance-restorated generation for the same prompt. The middle one has little adversarial effect due to the weak adversarial scheduler and strong attention and appearance constraining guidance coefficients, so the average precision (AP) is rather high. At the same time, the rightmost part shows an example of a highly adversarial image with little resemblance to the original generation.

Yet all of these three generations can be tiled very well, and a stronger adversarial scheduler does not mitigate the tiling prompt extension, which, in this case, is “clothes pattern”.

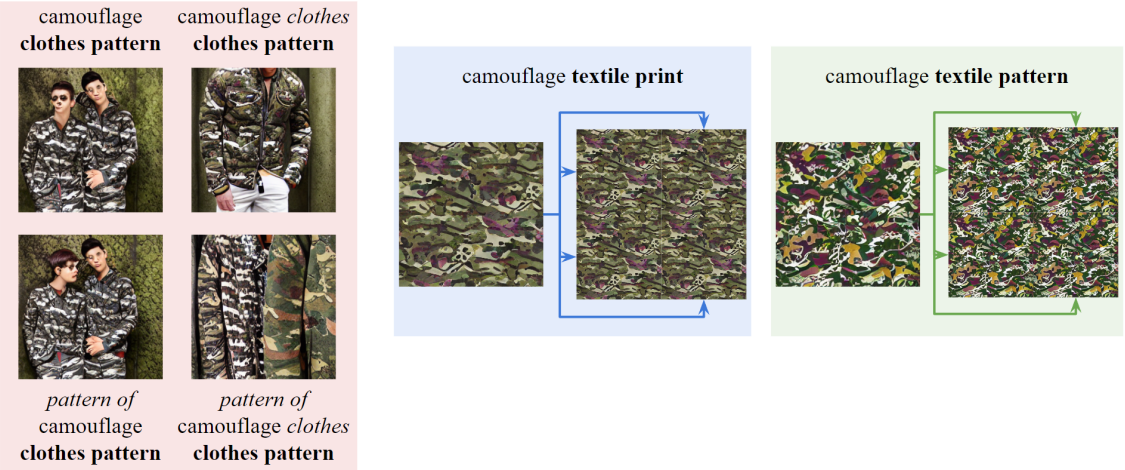


Figure 3.9 “clothes pattern” extension fails.

Problems The obvious problem of this approach is that there is nothing that guarantees that the chosen prompt extension will work as intended, creating a continuous pattern. For example, the “clothes pattern” extension fails on the “camouflage” prompt, as shown in the upper-left image of the red part of Figure 3.9. This can be explained by the fact that words like “camouflage” and “clothes” often go together, so the SD model has learned some additional dependencies and creates specific generations like the one we observe. Adding another “clothes” or enhancing the tiling effect with an additional “pattern of”, as suggested in the example of Figure 3.7(b), does not solve the problem.

Even though this particular problem can be solved by choosing another prompt extension like “textile print” or “textile pattern”, as demonstrated in the blue and green parts of

Figure 3.9, there is no guarantee that these other extension work for all possible prompts.

TV-loss TODO?

3.6 Datasets

In this section, we briefly describe the datasets that have been used during the research and our methods to add clothes to the generated data.

Inria Person TODO

Background TODO

3.7 Evaluation

In this section, we will briefly introduce the metrics that we are most likely to use for the evasion task.

IoU threshold Whether a box predicted by a detector is considered a valid object is decided based on the intersection over union (IoU) score. In case if there are many overlapping boxes on a test image like in the Inria Dataset that is often used for evaluation, a relatively high IoU threshold of 0.5 is usually chosen. However, a high threshold value may lead to overestimation of the effectiveness of the attack, so we intend to study different values.

AP TODO

ASR We define Attack Success Rate (ASR) as the ratio between the incorrectly predicted test images to the total number of test images. An object is considered as correctly identified if the objectiveness score for it is higher than 0,5.

Naturalness [8] suggested using subjective evaluation to get a naturalness score of a patch. For this evaluation, a number of participants is invited to estimate how natural the clothes in question look. A naturalness evaluation is not difficult to perform since the number of participants should not necessarily be very high—in [8], it was 24.

TODO describe problems with metrics

3.8 Experiments

In this section, we present the results of the method in different scenarios.

3.8.1 2D Pipeline

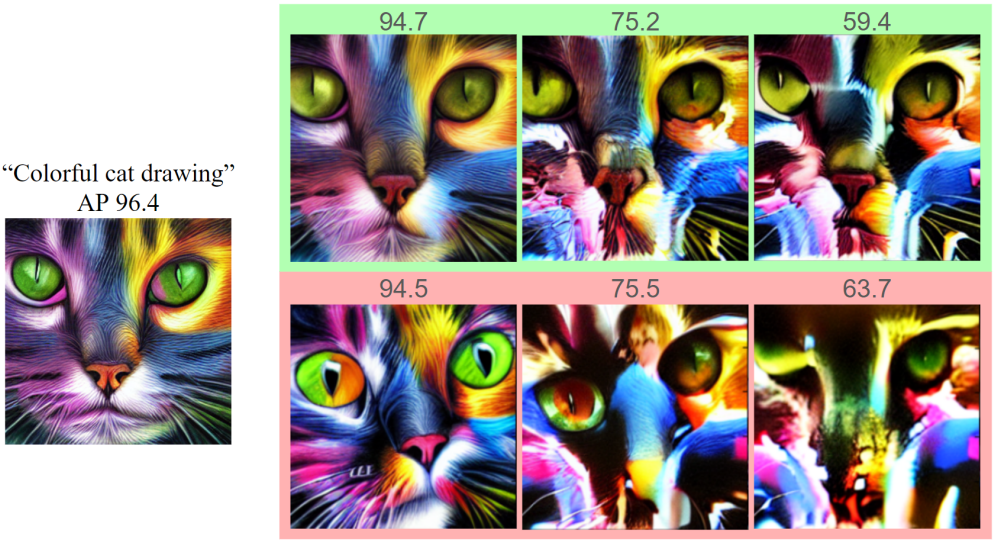


Figure 3.10 Experimental results for 2D pipeline, prompt "colorful cat drawing". Only adversarial guidance.

Figure 3.10 presents results obtained with the 2D pipeline for prompt "colorful cat drawing" for the original SD generation on the left and 6 adversarial generations with different schedulers on the right.

The upper green row shows relatively good results: the AP decreases as we trade naturalness of the picture for its effectiveness. The lower red row is significantly worse: we stray far away from our original generation on the left while achieving similar APs; moreover, the last two generations do not even appear natural. As a reminder, we also reiterate the reason why we want our adversarial generation to be close to the original image: we want to be able to control our final result, so, instead of just obtaining any image from the "colorful cat drawing" space, we want something close to provided image.

The first image in the lower world looks realistic, but has a high AP and is very far away from the original image. The reason behind this is that the scheduler that was used to create this generation had very high adversarial coefficients in the beginning and low by the end of the generations. Basically, we just started from a different noise and arrived at another natural image that just happened to be more adversarial. While optimizing latent as another part of the pipeline is promising in terms of adversarial effectiveness, we would have lost any control as soon as we switched to this method.

Figure 3.11 “A hamburger on a table”, SD. Different number of steps between 30 and 512.

Moreover, more often than not, a random noise that leads to higher adversarial effects also has absolutely unnatural non-adversarial generations. For example, for the “a hamburger on a table” prompt, SD generates very different results depending on the number of steps, as demonstrated in Figure 3.11. The same generally holds true for all prompts. It turns out that we can achieve higher adversarial effects if we guide the last generation; however, as we can see, this additional effectiveness comes at the cost of naturalness as well, so we do not look for the best starting latent and rather focus on controlling the generation process to produce something close to the original image. This also reinforces the theory that appearance constraint guidance is very important for our method.

The major problem of this approach is the fact that the APs are undeniably high, while the naturalness fades away rather swiftly; even more importantly, we are unable to achieve lower APs with this method, and higher coefficients in adversarial schedulers only lead to complete vanishing of naturalness and no decrease in adversarial effectiveness, like in the last generation with AP 63,7.

This is the initial pipeline we used, and it turned out to be so much less effective than the 3D pipeline that examples of the results are only presented in this subsection; the rest of this section is dedicated to the 3D pipeline.

3.8.2 Adversarial Guidance

After switching to the 3D pipeline, we immediately achieve much lower APs, as shown in Figure 3.12 on the example of the “palm clothes pattern” prompt adversarial generations for 12 different schedulers.

The generations on the upper green row are good and follow our expectations: as we apply stronger adversarial guidance coefficients, we go further away from the original non-adversarial generation on the right but also avoid detection better. Even though the last two images in this row do not resemble the original generation, we at least still observe “palms” in the resulting patches.

For the second blue row, while the adversarial effects are similar to the upper row, the generations are significantly worse: the first two have APs that are disproportionately

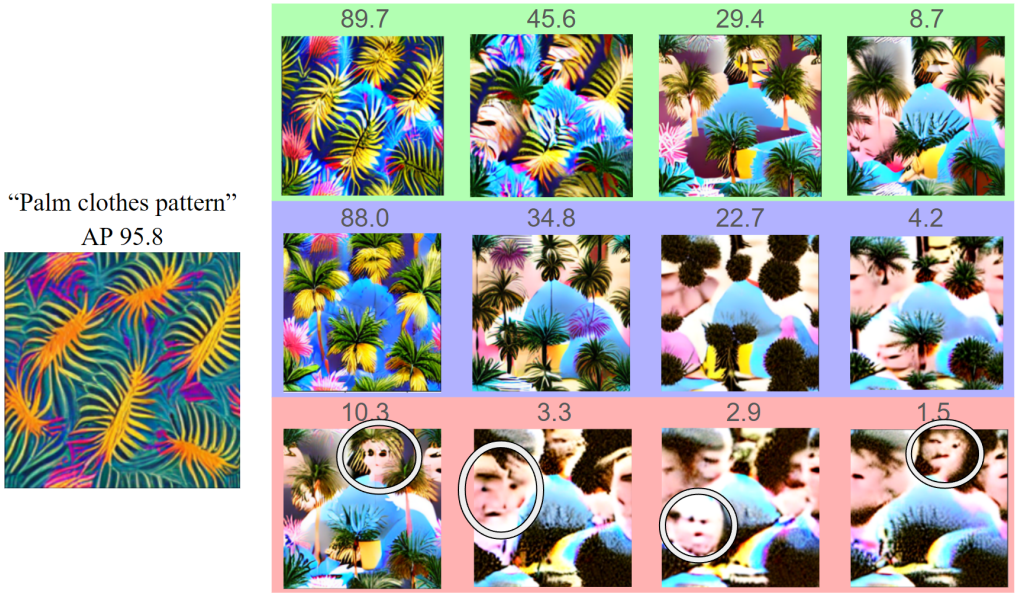


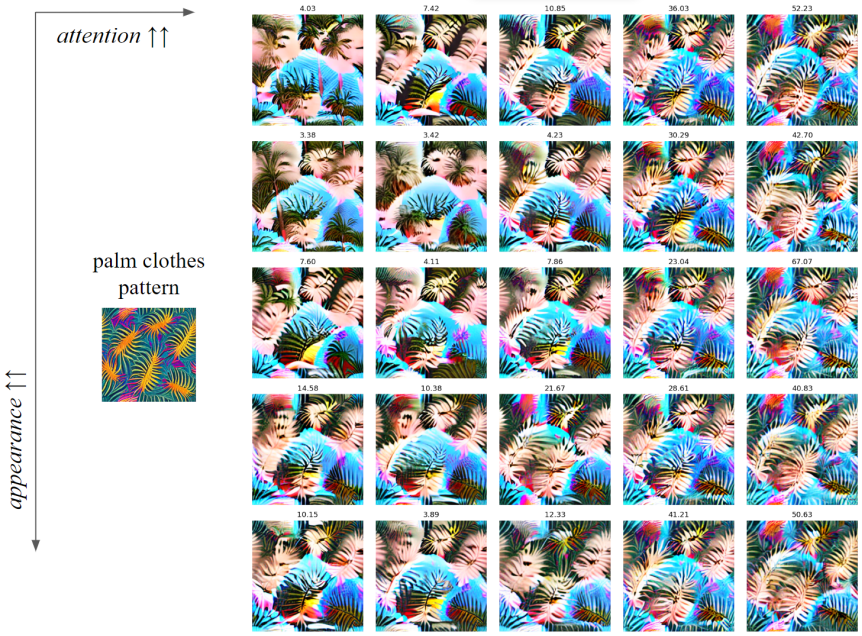
Figure 3.12 Experimental results for the 3D pipeline, prompt "palm clothes pattern". Only adversarial guidance.

large compared to how close they are to the original picture, and the last two are even more far away from the original compared to the corresponding images above and do not even seem to follow the prompt very well either. The scheduler used to generate the first image had adversarial coefficients that were high in the beginning and very low in the end, so, just like in similar experiments with the 2D pipeline, we obtain an image from the same space of "palm clothes pattern" generations but as if we started from another noise.

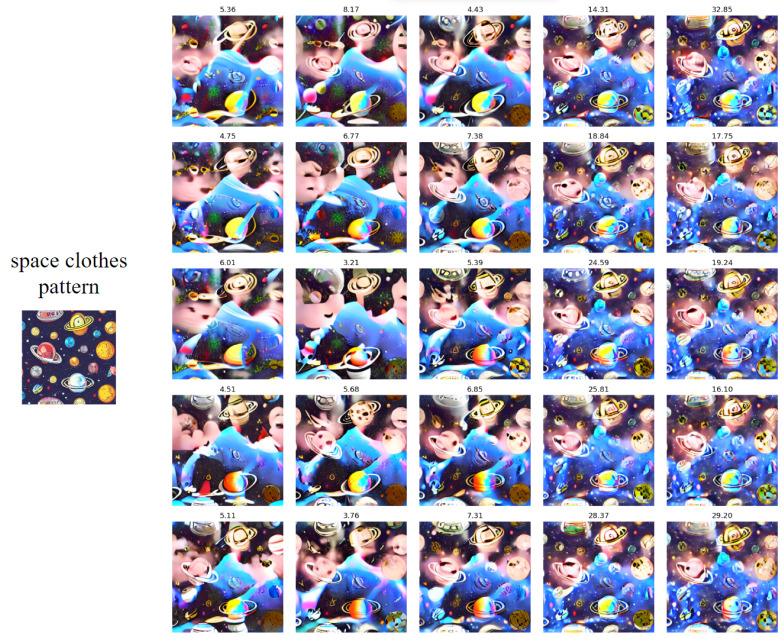
The last red row shows a common artifact that appears on the generations with low APs, that is, with very strong adversarial schedulers. First of all, the three last generations can be labeled as failures: they are neither natural nor even comply with the prompt conditioning. The artifact, people's faces, is a common problem not just in this method, but in detector evasion in general[14]. As a physical explanation, one might think of it as if after a person puts on the adversarial clothes, the model starts seeing many people in their silhouette and loses the big figure of the person themself. This artifact almost always accompanies low APs, and it is difficult to get rid of it, but appearance restoration does somewhat alleviate this problem.

3.8.3 Appearance Restoration

Figure 3.13 shows how the trade-off between adversarial effectiveness and resemblance to the original image by varying attention and appearance constraint coefficients



(a) “Palm clothes pattern”.



(b) “Space clothes pattern”.

Figure 3.13 Generations for the same adversarial scheduler with different strength of s^{at} and s^{ap} coefficients.

for two prompts, “palm clothes pattern” (Figure 3.13(a)) and “space clothes pattern” (Figure 3.13(b)).

In the left part of both figures, we can see the non-adversarial versions for the prompts.

Appearance constraint coefficient s^{ap} increases along the y axis (top to bottom direction), while the self-attention constraint coefficient s^{at} increases along the x axis (left to right direction), so the last images in the last rows are the closest to their respective non-adversarial images, but also have the lowest adversarial effects. While the resemblance to the non-adversarial pictures increases along both axes, we can see that self-attention constraint is indeed more effective for the same coefficients.

Sadly, we also notice that luck plays a visible part in the results since the AP does not change consistently. However, considering the fact that we only generated the images for 256 steps and used a weak SD model but managed to obtain good APs nevertheless, the method looks very promising. The AP is calculated in the digital world scenarios, but the patches are likely to be effective in the real physical world if the AP is at least lower than 7, and both prompts have relatively natural generations that satisfy that requirement.

3.8.4 Different Prompts

TODO maybe best results for around 5 random prompts, and also with different prompt extensions

3.8.5 Real Images

TODO we can give the model real images, use ddim inversion to obtain latents and work with them (or perhaps phantom trajectories like in self-guidance paper?)

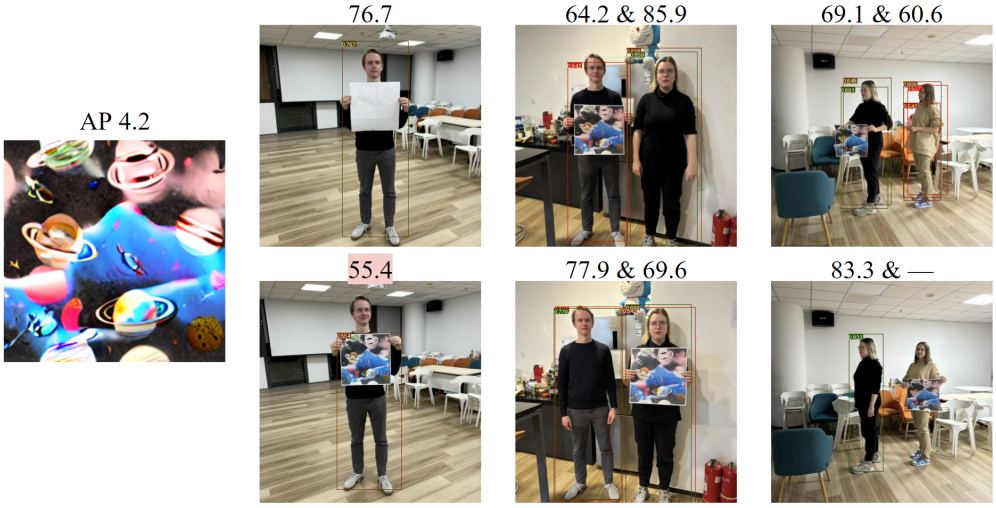
3.8.6 Physical Results

Patches Evaluation of physical results is rather challenging, mainly because printing actual clothes, taking videos with them, and then manually labelling all the data is very time-consuming and expensive. Firstly, we need to ensure that the generated patterns actually work by, for example, only printing them as patches and checking their effectiveness in several settings. Examples of this approach can be found in Figure 3.14.

TODO desc

Clothes TODO?

(a) “Space clothes pattern” prompt, AP 12.



(b) “Space clothes pattern” prompt, AP 4.

(c) “Flower clothes pattern” prompt, AP 57.

Figure 3.14 Physical results with generations as patches trained with the 3D pipeline.

3.9 Comparison Against Baseline Methods

TODO for this, we need to switch to yolo3

CHAPTER 4 CONCLUSION

4.1 Work and Contribution

TODO ([82])

4.2 Future Work

TODO

REFERENCES

[1] Ren S, He K, Girshick R B, et al. Faster R-CNN: towards real-time object detection with region proposal networks[J/OL]. CoRR, 2015, abs/1506.01497. http://arxiv.org/abs/1506.01497.

[2] Carion N, Massa F, Synnaeve G, et al. End-to-end object detection with transformers[A]. 2020. arXiv: 2005.12872.

[3] Redmon J, Farhadi A. Yolov3: An incremental improvement[A]. 2018. arXiv: 1804.02767.

[4] Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks[C/OL]// International Conference on Learning Representations. 2014. http://arxiv.org/abs/1312.6199.

[5] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[A]. 2015. arXiv: 1412.6572.

[6] Madry A, Makelov A, Schmidt L, et al. Towards deep learning models resistant to adversarial attacks[A]. 2019. arXiv: 1706.06083.

[7] Eykholt K, Evtimov I, Fernandes E, et al. Physical adversarial examples for object detectors [J/OL]. CoRR, 2018, abs/1807.07769. http://arxiv.org/abs/1807.07769.

[8] Hu Y C T, Chen J C, Kung B H, et al. Naturalistic physical adversarial patch for object detectors [C/OL]//2021 IEEE/CVF International Conference on Computer Vision (ICCV). 2021: 7828-7837. DOI: 10.1109/ICCV48922.2021.00775.

[9] Wang J, Liu A, Yin Z, et al. Dual attention suppression attack: Generate adversarial camouflage in physical world[A]. 2021. arXiv: 2103.01050.

[10] Saha A, Subramanya A, Patil K, et al. Role of spatial context in adversarial robustness for object detection[A]. 2020. arXiv: 1910.00068.

[11] Adhikari A, den Hollander R, Tolios I, et al. Adversarial patch camouflage against aerial detection[A]. 2020. arXiv: 2008.13671.

[12] Eykholt K, Evtimov I, Fernandes E, et al. Robust physical-world attacks on deep learning models[A]. 2018. arXiv: 1707.08945.

[13] Duan Y, Chen J, Zhou X, et al. DPA: learning robust physical adversarial camouflages for object detectors[J/OL]. CoRR, 2021, abs/2109.00124. https://arxiv.org/abs/2109.00124.

[14] Hu Z, Huang S, Zhu X, et al. Adversarial texture for fooling person detectors in the physical world[A]. 2022. arXiv: 2203.03373.

[15] Moosavi-Dezfooli S M, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks[A]. 2016. arXiv: 1511.04599.

[16] Papernot N, McDaniel P, Jha S, et al. The limitations of deep learning in adversarial settings [C/OL]//2016 IEEE European Symposium on Security and Privacy (EuroS&P). 2016: 372-387. DOI: 10.1109/EuroSP.2016.36.

[17] Carlini N, Wagner D. Towards evaluating the robustness of neural networks[A]. 2017. arXiv: 1608.04644.

[18] Papernot N, McDaniel P, Goodfellow I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples[A]. 2016. arXiv: 1605.07277.

[19] Papernot N, McDaniel P, Goodfellow I, et al. Practical black-box attacks against machine learning[A]. 2017. arXiv: 1602.02697.

[20] Su J, Vargas D V, Sakurai K. One pixel attack for fooling deep neural networks[J/OL]. IEEE Transactions on Evolutionary Computation, 2019, 23(5): 828-841. https://arxiv.org/abs/1710.08864. DOI: 10.1109/tevc.2019.2890858.

[21] Moosavi-Dezfooli S M, Fawzi A, Fawzi O, et al. Universal adversarial perturbations[A]. 2017. arXiv: 1610.08401.

[22] Brown T B, Mané D, Roy A, et al. Adversarial patch[J/OL]. CoRR, 2017, abs/1712.09665. http://arxiv.org/abs/1712.09665.

[23] Karmon D, Zoran D, Goldberg Y. Lavan: Localized and visible adversarial noise[A]. 2018. arXiv: 1801.02608.

[24] Dong Y, Liao F, Pang T, et al. Boosting adversarial attacks with momentum[A]. 2018. arXiv: 1710.06081.

[25] Dong Y, Pang T, Su H, et al. Evading defenses to transferable adversarial examples by translation-invariant attacks[A]. 2019. arXiv: 1904.02884.

[26] Yin H, Zhang H, Wang J, et al. Boosting adversarial attacks on neural networks with better optimizer[J/OL]. Security and Communication Networks, 2021, 2021: 1-9. https://arxiv.org/abs/2012.00567. DOI: 10.1155/2021/9983309.

[27] Shapira A, Bitton R, Avraham D, et al. Attacking object detector using a universal targeted label-switch patch[A]. 2022. arXiv: 2211.08859.

[28] Lee M, Kolter Z. On physical adversarial patches for object detection[A]. 2019. arXiv: 1906.11897.

[29] Divvala S K, Hoiem D, Hays J H, et al. An empirical study of context in object detection[C/OL]// 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009: 1271-1278. DOI: 10.1109/CVPR.2009.5206532.

[30] Kayhan O S, van Gemert J C. Evaluating context for deep object detectors[A]. 2022. arXiv: 2205.02887.

[31] Li J, Wei Y, Liang X, et al. Attentive contexts for object detection[A]. 2016. arXiv: 1603.07415.

[32] Liu X, Yang H, Song L, et al. Dpatch: Attacking object detectors with adversarial patches [J/OL]. CoRR, 2018, abs/1806.02299. http://arxiv.org/abs/1806.02299.

[33] Athalye A, Engstrom L, Ilyas A, et al. Synthesizing robust adversarial examples[A]. 2018. arXiv: 1707.07397.

[34] Pavlitskaya S, Hendl J, Kleim S, et al. Suppress with a patch: Revisiting universal adversarial patch attacks against object detection[A]. 2022. arXiv: 2209.13353.

[35] Lu J, Sibai H, Fabry E. Adversarial examples that fool detectors[A]. 2017. arXiv: 1712.02494.

[36] Chen S T, Cornelius C, Martin J, et al. ShapeShifter: Robust physical adversarial attack on faster r-CNN object detector[M/OL]//Machine Learning and Knowledge Discovery in Databases. Springer International Publishing, 2019: 52-68. https://arxiv.org/abs/1804.05810.

[37] Xue M, Yuan C, He C, et al. NaturalAE: Natural and robust physical adversarial examples for object detectors[J/OL]. Journal of Information Security and Applications, 2021, 57: 102694. https://arxiv.org/abs/2011.13692. DOI: 10.1016/j.jisa.2020.102694.

[38] Zhong Y, Liu X, Zhai D, et al. Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon[A]. 2022. arXiv: 2203.03818.

[39] Sharif M, Bhagavatula S, Bauer L, et al. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition[C/OL]//CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery, 2016: 1528–1540. https://doi.org/10.1145/2976749.2978392.

[40] Sharif M, Bhagavatula S, Bauer L, et al. A general framework for adversarial examples with objectives[J/OL]. ACM Transactions on Privacy and Security, 2019, 22(3): 1-30. https://arxiv.org/abs/1801.00349. DOI: 10.1145/3317611.

[41] Huang L, Gao C, Zhou Y, et al. Universal physical camouflage attacks on object detectors[A]. 2020. arXiv: 1909.04326.

[42] Thys S, Ranst W V, Goedemé T. Fooling automated surveillance cameras: adversarial patches to attack person detection[A]. 2019. arXiv: 1904.08653.

[43] Wu Z, Lim S N, Davis L, et al. Making an invisibility cloak: Real world adversarial attacks on object detectors[A]. 2020. arXiv: 1910.14667.

[44] Xu K, Zhang G, Liu S, et al. Adversarial t-shirt! evading person detectors in a physical world [A]. 2020. arXiv: 1910.11099.

[45] Doan B G, Xue M, Ma S, et al. Tnt attacks! universal naturalistic adversarial patches against deep neural network systems[A]. 2022. arXiv: 2111.09999.

[46] Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks[A]. 2014. arXiv: 1406.2661.

[47] Hu W, Tan Y. Generating adversarial malware examples for black-box attacks based on gan[A]. 2017. arXiv: 1702.05983.

[48] Xiao C, Li B, Zhu J Y, et al. Generating adversarial examples with adversarial networks[A]. 2019. arXiv: 1801.02610.

[49] Song Y, Shu R, Kushman N, et al. Constructing unrestricted adversarial examples with generative models[A]. 2018. arXiv: 1805.07894.

[50] Feng W, Wu B, Zhang T, et al. Meta-attack: Class-agnostic and model-agnostic physical adversarial attack[C]//2021 IEEE/CVF International Conference on Computer Vision. 2021: 7787–7796.

[51] Casper S, Nadeau M, Hadfield-Menell D, et al. Robust feature-level adversaries are interpretability tools[A]. 2023. arXiv: 2110.03605.

[52] Lapid R, Sipper M. Patch of invisibility: Naturalistic black-box adversarial attacks on object detectors[A]. 2023. arXiv: 2303.04238.

[53] Liu A, Liu X, Fan J, et al. Perceptual-sensitive gan for generating adversarial patches[C/OL]// AAAI'19/IAAI'19/EAAI'19: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. Honolulu, Hawaii, USA: AAAI Press, 2019. https://doi.org/10.1609/aaai.v33i01.33011028.

[54] Kong Z, Guo J, Li A, et al. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving[A]. 2021. arXiv: 1907.04449.

[55] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models[A]. 2020. arXiv: 2006.11239.

[56] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[A]. 2015. arXiv: 1505.04597.

[57] Saharia C, Chan W, Chang H, et al. Palette: Image-to-image diffusion models[A]. 2022. arXiv: 2111.05826.

[58] Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models[A]. 2022. arXiv: 2112.10752.

[59] Nichol A, Dhariwal P, Ramesh A, et al. Glide: Towards photorealistic image generation and editing with text-guided diffusion models[A]. 2022. arXiv: 2112.10741.

[60] Saharia C, Chan W, Saxena S, et al. Photorealistic text-to-image diffusion models with deep language understanding[C]//Koyejo S, Mohamed S, Agarwal A, et al. Advances in Neural Information Processing Systems: volume 35. Curran Associates, Inc., 2022: 36479-36494.

[61] Li H, Yang Y, Chang M, et al. Srdiff: Single image super-resolution with diffusion probabilistic models[A]. 2021. arXiv: 2104.14951.

[62] Kawar B, Elad M, Ermon S, et al. Denoising diffusion restoration models[A]. 2022. arXiv: 2201.11793.

[63] Han X, Zheng H, Zhou M. Card: Classification and regression diffusion models[A]. 2022. arXiv: 2206.07275.

[64] Amit T, Shaharbany T, Nachmani E, et al. Segdiff: Image segmentation with diffusion probabilistic models[A]. 2022. arXiv: 2112.00390.

[65] Wu J, Fu R, Fang H, et al. Medsegdiff: Medical image segmentation with diffusion probabilistic model[A]. 2023. arXiv: 2211.00611.

[66] Ho J, Salimans T, Gritsenko A, et al. Video diffusion models[A]. 2022. arXiv: 2204.03458.

[67] Harvey W, Naderiparizi S, Masrani V, et al. Flexible diffusion modeling of long videos[A]. 2022. arXiv: 2205.11495.

[68] Chen J, Chen H, Chen K, et al. Diffusion models for imperceptible and transferable adversarial attack[A]. 2023. arXiv: 2305.08192.

[69] Toheed A, Yousaf M H, Rabnawaz, et al. Physical adversarial attack scheme on object detectors using 3d adversarial object[C/OL]//2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2). 2022: 1-4. DOI: 10.1109/ICoDT255437.2022.9787422.

[70] Chang A X, Funkhouser T, Guibas L, et al. Shapenet: An information-rich 3d model repository [EB/OL]. 2015. http://arxiv.org/abs/1512.03012.

[71] Wang D, Jiang T, Sun J, et al. Fca: Learning a 3d full-coverage vehicle camouflage for multi-view physical adversarial attack[A]. 2021. arXiv: 2109.07193.

[72] Suryanto N, Kim Y, Kang H, et al. Dta: Physical camouflage attacks using differentiable transformation network[A]. 2022. arXiv: 2203.09831.

[73] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C/OL]//2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05): volume 1. 2005: 886-893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[74] Hu Z, Chu W, Zhu X, et al. Physically realizable natural-looking clothing textures evade person detectors via 3d modeling[M/OL]//Computer Vision and Pattern Recognition Conference. 2023. https://openaccess.thecvf.com/content/CVPR2023/papers/Hu_Physically_Realizable_Natural-Looking_Clothing_Textures_Evade_Person_Detectors_via_3D_CVPR_2023_paper.pdf.

[75] Bookstein F. Principal warps: thin-plate splines and the decomposition of deformations[J/OL]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989, 11(6): 567-585. DOI: 10.1109/34.24792.

[76] Ho J, Salimans T. Classifier-free diffusion guidance[A]. 2022. arXiv: 2207.12598.

[77] Lin T, Maire M, Belongie S J, et al. Microsoft COCO: common objects in context[J/OL]. CoRR, 2014, abs/1405.0312. http://arxiv.org/abs/1405.0312.

[78] Dhariwal P, Nichol A. Diffusion models beat gans on image synthesis[A]. 2021. arXiv: 2105.05233.

[79] Epstein D, Jabri A, Poole B, et al. Diffusion self-guidance for controllable image generation [A]. 2023. arXiv: 2306.00986.

[80] Tumanyan N, Geyer M, Bagon S, et al. Plug-and-play diffusion features for text-driven image-to-image translation[A]. 2022. arXiv: 2211.12572.

[81] Schuhmann C, Beaumont R, Vencu R, et al. Laion-5b: An open large-scale dataset for training next generation image-text models[A]. 2022. arXiv: 2210.08402.

[82] Routh B N, Johnston D, Harris K, et al. Anatomical and electrophysiological comparison of ca1 pyramidal neurons of the rat and mouse[J]. Journal of neurophysiology, 2009, 102(4): 2288-2302.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Associate Professor Hu Xiaolin, for his guidance and exceptional ability to motivate students.

This research was guided by Hu Zhanhao, without whom I would not have known how to start working in the area. His close ex-colleagues, other members of our laboratory, Zhang Wei and TODO, also gave priceless advice on multiple occasions in both technical and research directions.

I would also like to extend my gratitude to all the students in our laboratory for their enthusiastic help and support in weekly meetings and outside of them.

The GPU resources were provided by the laboratory under Associate Professor Hu's supervision. The experiments will be performed on the computational resources provided by the IDG/McGovern Institute for Brain Research at Tsinghua University.

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

RESUME

Education

2022 — 2024: **Tsinghua University**, Computer Science and Technology department, Master's Program "Advanced Computing Program". Supervisor — Associate Professor X. Hu, thesis on physical adversarial attacks.

2018 — 2022: **National Research University Higher School of Economics**, Faculty of Computer Science, Bachelor's Program "Applied Mathematics and Information Science". Supervisor — Associate Professor M. Rakhuba, thesis on theoretical ML. Minor in Bioinformatics.

Work Experience

2022 — 2024: **Tsinghua** Research Intern in the Tsinghua Laboratory of Brain and Intelligence.

2021 — 2024: **HSE, Moscow**: Teaching Assistant to Researcher E. Lobacheva.

2021 — 2022: **HSE, Moscow** Research Intern in the Laboratory of Stochastic Algorithms and High-Dimensional Inference.

2021: **HSE, Moscow** Research Intern in the Laboratory of Theoretical Computer Science.

Achievements

Achievements

[1] "Towards Practical Control of Singular Values of Convolutional Layers", **NeurIPS 2022**.

[2] First place in the "Student Research Paper Competition" for bachelor's students held by **HSE University**, 2021.

Scholarships

[3] 2022 — 2024: Tsinghua Scholarship for Graduate Students.

[4] 2022: Academic Scholarship & Travel Grant for winning the "Student Research Paper Competition".

[5] 2018 — 2022: Russian State Academic Scholarship.

[6] 2018 — 2020: Moscow Government Scholarship for Distinguished Achievements in Education.

Additional Information

Languages Russian (native), English (C1), Chinese (Intermediate).

Technical Skills Python, C/C++, LaTeX, Java, bash.

Other Skills Arts, Translation, Linguistics.

COMMENTS FROM THESIS SUPERVISOR

论文提出了……

RESOLUTION OF THESIS DEFENSE COMMITTEE

论文提出了……

论文取得的主要创新性成果包括：

1. ……

2. ……

3. ……

论文工作表明作者在 ××××× 具有 ××××× 知识，具有 ×××× 能力，论文 ××××，答辩 ××××。

答辩委员会表决，（× 票/一致）同意通过论文答辩，并建议授予 ×××（姓名）×××（门类）学博士/硕士学位。