MEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

One noise to fool them all

or

Computationally efficient black-box attacks against state-of-the-art real-time deep learning models for object detection and semantic image segmentation

Author:
Sixte de Maupeou

Supervisor:
Luis Muñoz González

June 17, 2019

Submitted in partial fulfillment of the requirements for the Masters of Engineering in Computing (Artificial Intelligence) of Imperial College London

Abstract

Over the past few years, real-time computer vision systems relying on deep convolutional networks have risen to previously unseen heights in terms of accuracy. Consequently, such models are becoming increasingly widespread in a wide range of applications, ranging from autonomous vehicles vision to inappropriate content detection on social media. However, such models have been proven to be vulnerable to adversarial samples, malicious inputs almost indistinguishable from their legitimate counterparts with the human eye but largely misinterpreted by computer vision models. In particular, it has been shown that such samples can be generated via query-efficient black-box attacks using procedural noise against image classification models. In this project, we demonstrate that these attacks transfer to more complex tasks thereby achieving the most query-efficient attacks on state-of-the-art object detection and semantic image segmentation models. Specifically, we show that within an L_∞ norm of 16 and just 25 queries of Bayesian optimisation per image, an image-specific black-box procedural noise attack can decrease the mean average precision of object detector YOLOv3 by 61% from 0.585 to 0.231 on a 5000 COCO images dataset. Besides, we introduce an image-generic Bayesian optimisation attack with only 25 queries which decreases the mean intersection over union of SwiftNetRN-18 segmentation model by 96% down from 75.45% to 3.18% on the Cityscapes validation dataset of 500 images. Alongside these, we design and evaluate various procedural noise attack techniques and compare the adversarial properties of different noise function as well as the impact of different L_∞ norm restrictions. Finally, we explore and assess the suitability and effectiveness of different potential defences against procedural noise attacks in an effort to improve upon the robustness of the existing models.

Acknowledgements

Thank you very much Luis for your enthusiasm, support and helpful guidance throughout the project. More importantly, thank you for communicating me your passion for adversarial machine learning and instilling into me the desire to pursue a research-orientated career.

I would also like to thank Kenneth T. Co for his insights on procedural noise and for offering me the chance to give my best and contribute to an exciting paper.

Finally, I want to give credit to my family for supporting me in my desire to pursue my studies at Imperial and to Charlotte for encouraging me to always go one step further than I would on my own.

Contents

List of Figures

Chapter 1

Introduction

In recent years, the exponential increase in computational power and storage space of computers following Moore's law, as well as the implementation of optimised hardware has supported the development of extremely accurate artificial intelligence systems leveraging the power of deep learning.

These powerful models have already gained significant attention and are already widely and successfully used in highly critical real-world scenarios ranging from cancer detection on MRI images to environment assessment in autonomous vehicles. However, the deep neural networks underpinning these machine learning systems have been proven to be highly vulnerable to quasi-imperceptible perturbations in their inputs dramatically impacting their accuracy [13].

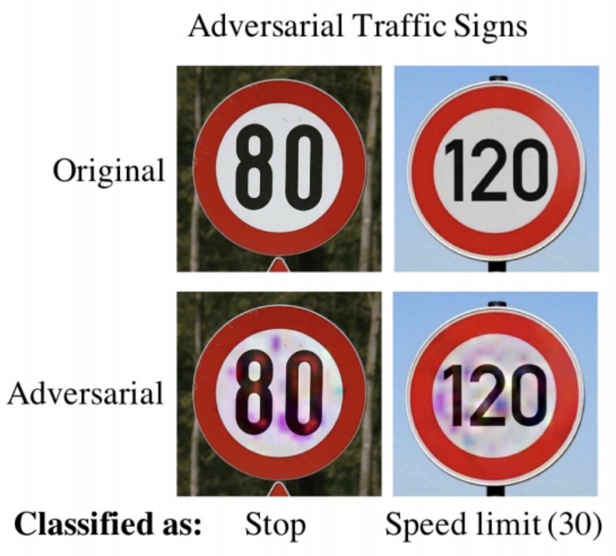


Figure 1.1: Example of an evasion attack against a road signs classifier. [1]

Those findings have revealed that systems taking their inputs from the internet or the outside world such as self driving cars are particularly insecure to these so-called evasion attacks. As shown in Figure 1.1, a speed limit sign can be slightly modified to be treated as a stop or another speed limit sign by a well-trained road signs classifier.

Considering the importance of the existence of vulnerabilities in so widely and blindly used deep learning models, an arms race has emerged over the past five years between evasion attacks aiming to fool these machine learning systems using adversarial input samples[13], [14], [16] and defenses enhancing their robustness by detecting[21], [22], [23], [24] or countering[25] such attacks. Most of the attacks to date were presented against image classifiers, deep neural networks taking an image as an input and outputting a class label or probabilities for the image to belong to each class label considered. Besides, some of these attacks have successfully been adapted and tested for different tasks such as object detection[26], [27], [28] which consists in identifying the boundaries of the instances of a specific object in an image, as well as semantic image segmentation[15], [28], which generates a color map of the different objects in an image and their associated labels.

Recent breakthroughs, in particular the procedural noise attacks by Kenneth Co which are detailed in our paper [16] have proven inexpensive while very effective against state of the art image classifiers in the black-box setting, that is with no knowledge of the target classifier. It achieves significant results by leveraging the resemblance of perturbations induced in images by specific noise functions with the low-level features convolutional networks are most sensitive to and optimising the noise parameters. However, these attacks have yet to be tested against other types of machine learning systems in order to assess their transferability but also to better understand their strengths and develop effective defences.

In the following we implement and test the first black-box procedural noise attacks against machine learning models beyond image classification. In particular, we focus on compromising object detection and real-time semantic image segmentation models which are widely used in very sensitive applications such as autonomous vehicles and robotics. We further evaluate the performance of these attacks in order to assess the transferability of procedural noise attacks from one task to another, as well as the robustness of the state of the art models for these tasks to such attacks. Thanks to our work on those different tasks, in particular semantic image segmentation we provide a better understanding of the adversarial properties of Perlin noise.

On the state-of-the-art object detection model YOLOv3, we explore different strategies to cause poor detection on our adversarial samples in terms of both the intersection over union and mean average precision metrics by preventing true detections and generating false detections. We further assess the potential threat of targeted attacks by generating attacks aiming to fool the model on a specific class. We also compare the results to the current performance of the target models on original samples in order to measure the impact of our attacks on this object detection system. We then proceed in a similar manner with SwiftNetRN-18, a state-of-the-art semantic image segmentation model, evaluating its performance on legitimate and adversarial samples in various settings. Finally, we show a potential defence that is rather effective against procedural noise adversarial examples while slightly reducing the accuracy of both models on legitimate samples.

Chapter 2

Background

This section lays out the principles of machine learning and adversarial machine learning techniques and defences as well as procedural noise functions. These will serve as a fundamental basis for a better understanding of the research carried out throughout this project.

2.1 Machine Learning

2.1.1 Principles of machine learning

Machine learning is a field of applied artificial intelligence in which machines learn by experience to perform tasks they were not explicitly programmed to achieve. It is particularly relevant for tasks such as image classification that are too complex and involve too many parameters for the machine to be efficient following explicit algorithms.
We can distinguish three main paradigms in machine learning: supervised learning, unsupervised learning and reinforcement learning.

2.1.1.1 Supervised learning

In supervised learning a mathematical model is built by learning from a training set containing known associations of inputs and outputs. Leveraging the training data, the mathematical model acts a function that maps any unseen input to an estimated output.
Supervised learning tasks are divided in two categories: classification and regression problems. In classification problems, the output, usually called a label, belongs to a discrete set of values and the model's role is to attribute the right label to any input. Regression tasks on the other hand map an input to an output that may be any value in a specified continuous range.
As illustrated in Figure 2.1, a classification task involves identifying the boundaries between classes in order to associate the right label to new data points while a regression model aims to fit the underlying function mapping inputs to outputs in order to produce an estimation of the output given an unseen input. The simple classifier on the left-hand side of the figure represents a linear classifier, where the boundary determined by the model is a line in a two-dimensional space formed by the variables "Gene 1" and "Gene 2" and separates data that falls in the

“Disease” class from data that falls in the “Healthy” class. However, a classifier may also be non linear, operate in a multidimensional space and classify data into more than two classes. The regression illustrated on the right is also linear in a two-dimensional space although regression is often non linear in real world problems and the space may be more than two-dimensional with a multivariable function. In general, to solve a regression problem a model is chosen and a cost function is defined that typically measures how far training data points are from the model. The model can then be improved via gradient descent, an iterative process reducing the cost function by adjusting its parameters in the direction opposite to the gradient of the cost function.

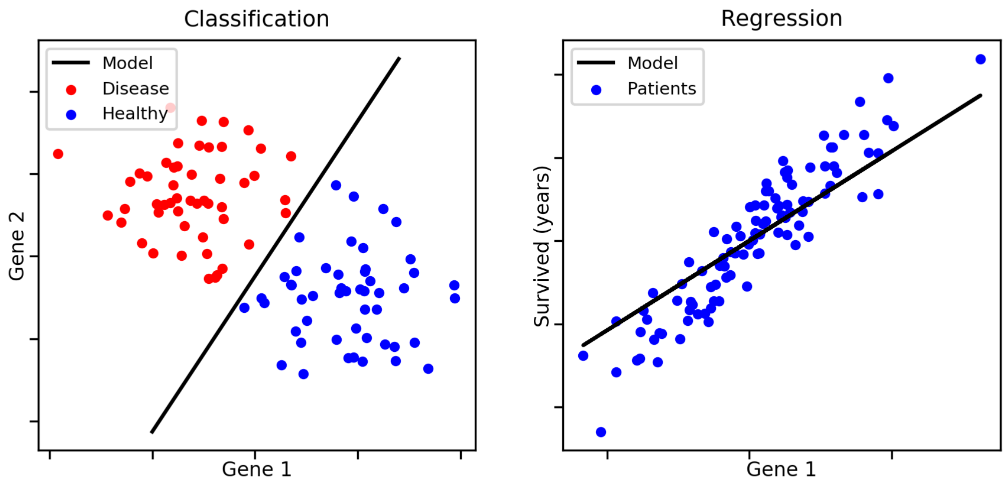


Figure 2.1: Classification and regression tasks examples [2]

In some cases the corresponding output is not available for all inputs in the training set and the task is considered semi-supervised.

2.1.1.2 Unsupervised learning

In unsupervised learning, the training data does not have associated labels or values. The task is then about inferring a structure describing the distribution of the data. The applications of unsupervised learning include clustering, outlier detection, dimensionality reduction and generative adversarial networks.
An example of a clustering algorithm is k-means data clustering, used to identify three groups of data in Figure 2.2. Given the number of desired clusters n and a set of data points, it finds k centers such that the sum of the squared distance between each data point and the closest center is minimal.

Dimensionality reduction is the mapping of high dimensional data to a lower dimensional space. This is often often performed using Principal Component Analysis (PCA) which performs a selection of k eigenvectors corresponding corresponding to the k largest eigenvalues of the data and maps the data to the space spanned by those k eigenvectors or principal components.

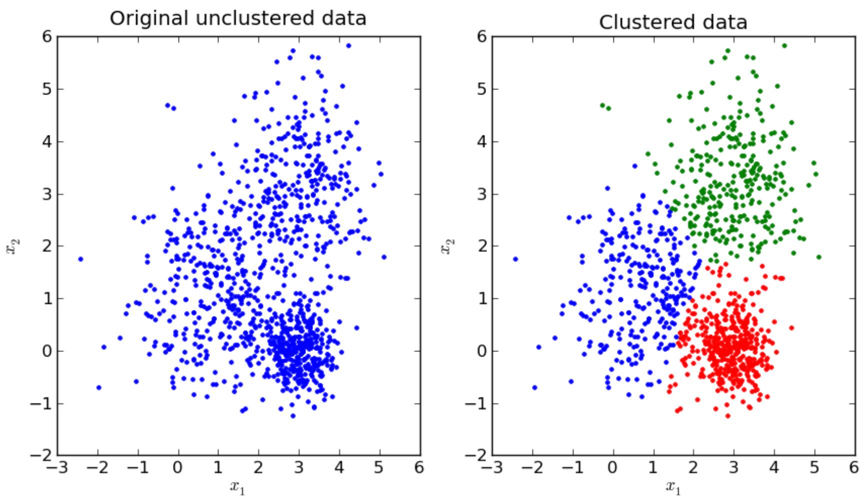


Figure 2.2: Unsupervised learning using the k-means data clustering algorithm [3]

2.1.1.3 Reinforcement learning

Reinforcement learning is an area of machine learning in which agents interact with an environment by taking actions and moving between states with the aim to maximise rewards as shown in Figure 2.3.

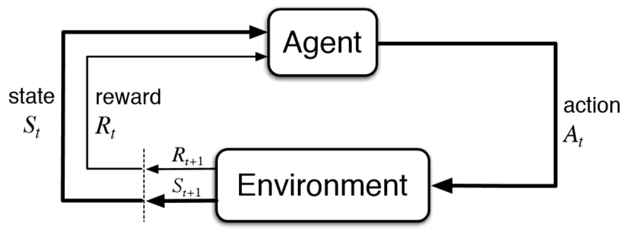


Figure 2.3: Basic Reinforcement Learning Framework [4]

It is based on the notion of Markov Decision Process (MDP) described by Sutton and Barto [29] and defined by:

- A state space $\mathcal{S}$ and an action space $\mathcal{A}$.
- A transition probability matrix $\mathcal{P}^a_{ss'}$ for each action a containing the probability transitioning to state *s'* when action *a* is taken in state *s*.
- An instant reward matrix $\mathcal{R}^a_{ss'}$ for each action a listing rewards. $r(s, a, s')$ obtained by transitioning from state *s* to *s'* as a result of taking action *a*.
- A discount factor $\gamma \in [0, 1]$.
- A policy π determining how the agent behaves in a given state. It may be stochastic, in which case $a \sim p_\pi(a|s)$ or deterministic and $a = \pi(s)$.

In an MDP the total return is $R_t = \sum_{k=0}^{\infty} \gamma^t r_{t+k+1}$ and serves as a basis to define the value of a state given a policy: $V^\pi(s) = E_\pi(R_t|S_t = s)$. The optimal policy for an agent is the policy that yields the optimal (largest possible) state value function in the starting state.

Given full knowledge of an MDP it can be solved by planning optimisation using dynamic programming, a set of algorithms that use bootstrapping, breaking a problem into overlapping subproblems and finding the optimal solutions to the subproblems to solve the original problem. In a model free environment, an agent does not know the transition probabilities rewards but it can learn estimates of the state value function via other methods using sampling such as Monte-Carlo or relying on bootstrapping such as temporal difference learning.

Some of the major tasks in reinforcement learning are estimating the state value function $V_\pi(s)$ given a policy π, $q_\pi(s,a)$ the value of taking action a in state s given a policy π and $q_*(s,a)$, the value of taking action a in state s and then following the optimal policy. These can be solved using full backup in a fully known MDP with respectively iterative policy evaluation, Q-policy iteration and Q-value iteration. However, in a model free MDP, this is done using sample backup with temporal difference learning, Sarsa ($Q(s,a) \leftarrow Q(s,a)+\alpha(r+\gamma\cdot Q(s',a')-Q(s,a))$) and Q-learning ($Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a))$).

The most important challenge with sample backup techniques is to balance the tradeoff between exploration and exploitation of the environment, which can be controlled with the α, appearing in the Sarsa and Q-learning updates above.

Deep neural networks are also increasingly used as policy networks and are subject to adversarial machine learning attacks [30].

2.1.2 Artificial Neural networks

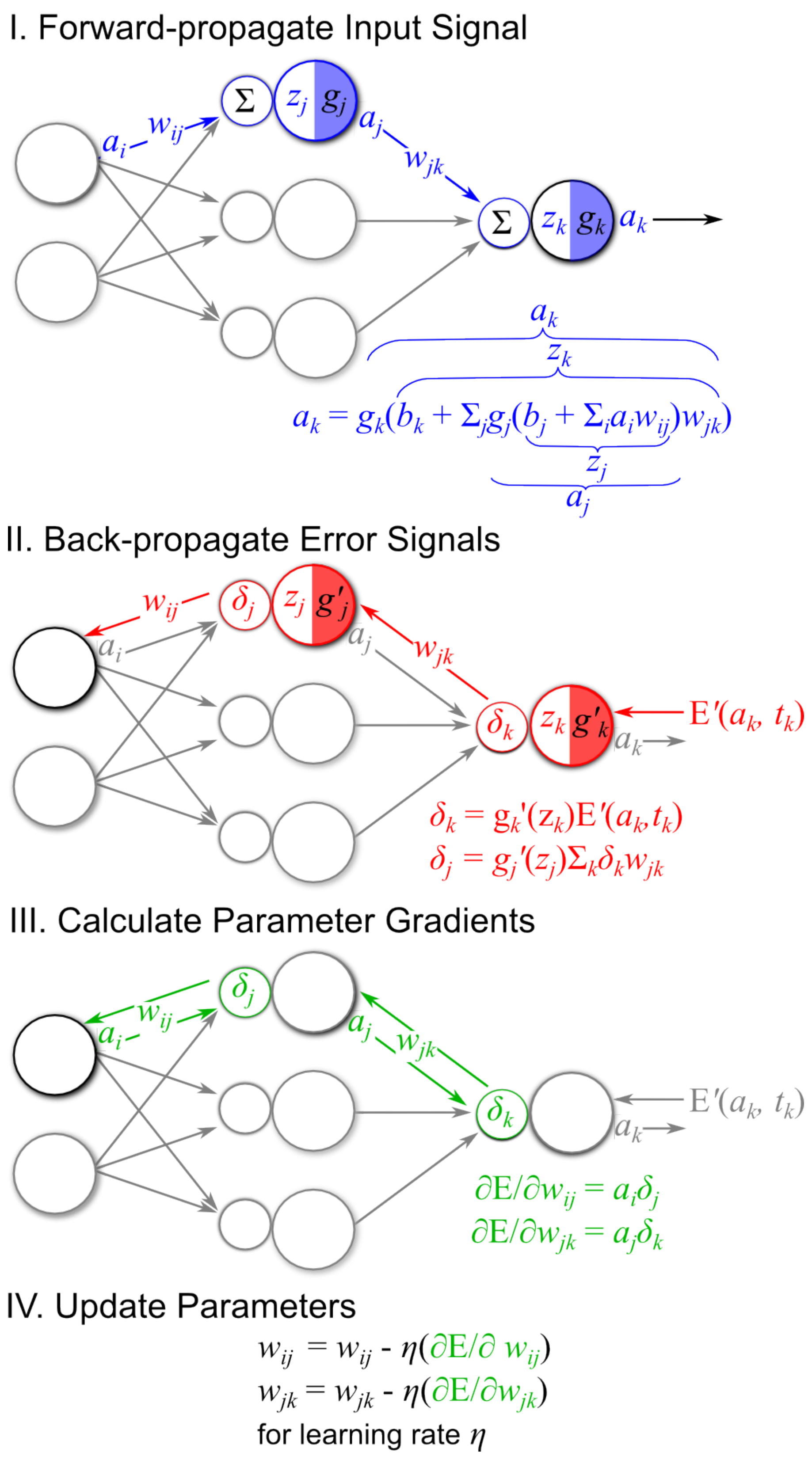
Artificial neural networks (ANN) are a framework used in supervised, unsupervised and reinforcement learning. An ANN consists of a collection of interconnected nodes called neurons. Originally, the purpose of a neural network was to model learning in a brain but nowadays ANNs are often built with an architecture to support one specific learning task such as image classification.

2.1.2.1 Basic Neural Networks

A simple neural network consists of an input layer, a number of hidden layers and an output layer. The data to learn or process enters the input layer, is processed through the hidden layers an an output is produced by the output layer. If the graph formed by the neurons connections is acyclic then the network is feedforward, otherwise it is recurrent or feedback and possesses an internal state that can act as a memory.

Each neuron i receiving an input $p_i(t)$ at time t has an activation $a_i = g_i(t)$, a threshold θ_i, an activation function $g_i(t+1) = f_{act}(p_i(t), g_i(t), \theta_i)$ and an output function $o_i(t) = f_{out}(g_i(t))$. Common activation functions include the sigmoid $f(x) = \frac{1}{1+e^{-x}}$, ReLU $f(x) = max(0,x)$ and softmax $S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$ functions while the identity function is often used as output function.

Then, each connection from a neuron i to a neuron j has a weight w_{ij} that adjusts how much the output of neuron i will impact the input of neuron j, and potentially a bias b_j or w_{0j}. Then, the input of neuron j with predecessors i is defined by the propagation function:



I. Forward-propagate Input Signal
a_i
w_{ij}
Σ
z_j g_j
a_j
w_{jk}
Σ
z_k g_k
a_k
$a_k = g_k(b_k + \Sigma_j g_j(b_j + \Sigma_i a_i w_{ij}) w_{jk})$
z_k
z_j
a_j
II. Back-propagate Error Signals
w_{ij}
δ_j
z_j g'_j
a_i
a_j
w_{jk}
δ_k
z_k g'_k
a_k
$E'(a_k, t_k)$
$\delta_k = g_k'(z_k)E'(a_k,t_k)$
$\delta_j = g_j'(z_j)\Sigma_k \delta_k w_{jk}$
III. Calculate Parameter Gradients
δ_j
a_i
w_{ij}
a_j
w_{jk}
δ_k
$E'(a_k, t_k)$
a_k
$\partial E/\partial w_{ij} = a_i \delta_j$
$\partial E/\partial w_{jk} = a_j \delta_k$
IV. Update Parameters
$w_{ij} = w_{ij} - \eta(\partial E/\partial w_{ij})$
$w_{jk} = w_{jk} - \eta(\partial E/\partial w_{jk})$
for learning rate η

Figure 2.4: Artificial neural network training [5]

$p_j(t) = b_j + \sum_i a_i(t) \cdot w_{ij}$. This forward propagation of the input signal can be visualised in Figure 2.4 (I).

During the learning phase, a learning rule is followed to adjust parameters of the network such as the weights and thresholds in order to improve its accuracy. The most popular is backpropagation, in which the error or loss function is propagated backwards from the output layer in order to adjust the weights of the network by gradient descent using the gradient of the loss with respect to the weight considered. This technique illustrated in Figure 2.4 (II)-(IV) allows reducing the loss until a local minimum of the loss function is reached.

According to the universal approximation theorem, any real valued function can be approximated by a neural network with one hidden layer and a finite number of neurons. Nevertheless, in recent years deep neural networks, ANNs containing a large number of layers have been widely adopted as they appear to perform better and with a smaller total number of neurons than shallow networks.

2.1.2.2 Autoencoders

Autoencoders are neural networks with the ability to perform dimensionality reduction with an encoder and reconstruct an approximation of the original data by minimising the reconstruction error with a decoder. A particularly interesting variant is the denoising autoencoder (DAE) illustrated in Figure 2.5.

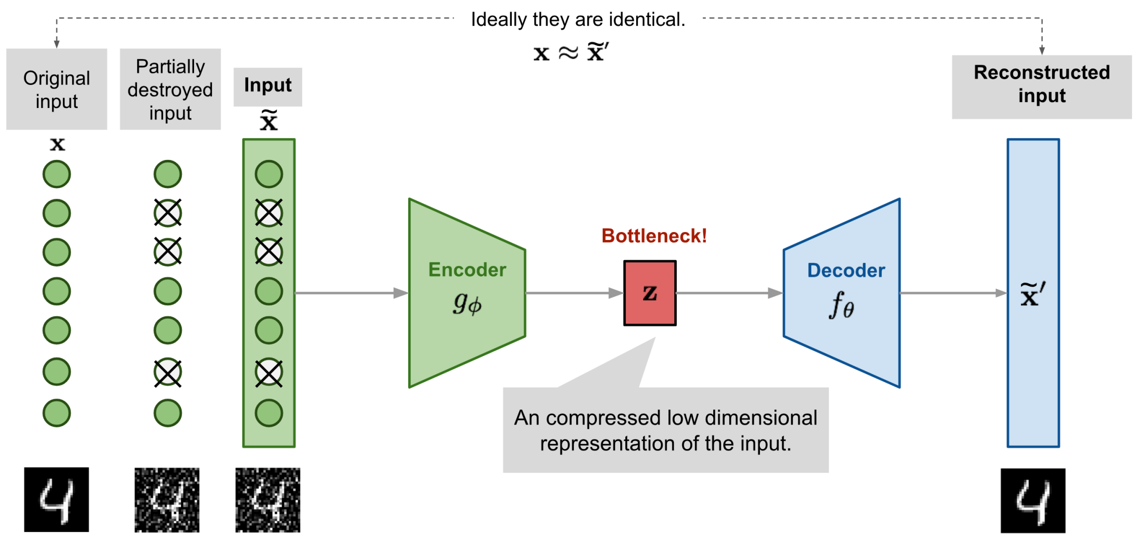

Ideally they are identical.
$\mathbf{x} \approx \tilde{\mathbf{x}}'$
Original input
$\mathbf{x}$
Partially destroyed input
Input
$\tilde{\mathbf{x}}$
Reconstructed input
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\tilde{\mathbf{x}}'$
An compressed low dimensional representation of the input.

Figure 2.5: Denoising autoencoder [6]

Leveraging a collection of data samples x corrupted by noise addition to generate noisy samples $\widetilde{x}$, the DAE with input $\widetilde{x}$ and output $\widetilde{x}'$ uses the distance between x and $\widetilde{x}'$ $L(x, \widetilde{x}')$ as a loss function to minimise. N such denoising autoencoders can be stacked such that the output of DAE k, $0 \leq k \leq N-1$ is the input of DAE k+1.

2.1.2.3 Convolutional Neural Networks

Convolutional neural networks (CNN) are a class of deep neural networks particularly suited for image [31] and video [32] classification and recognition but may also perform other tasks such as natural language processing. A CNN usually contains a large number of layers with different roles among which the most popular are convolutional, pooling, fully connected and ReLU layers.

Convolutional layers In a convolutional layer, each neuron is connected to a small subset neurons from the previous layer forming a specific region of that layer called the receptive field of the neuron as shown in Figure 2.6.

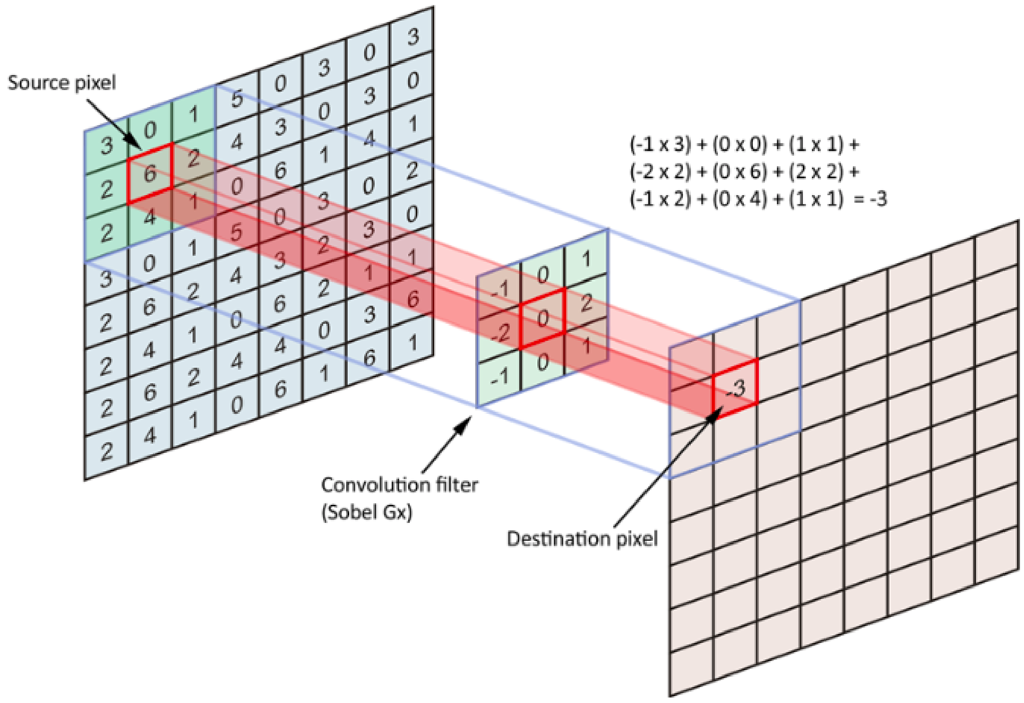

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 2.6: Example of a convolution in a CNN [7].

These layers simplify the learning process by vastly reducing the number of edges and associated weights, hence reducing the number of free parameters to be adjusted in the training phase as opposed to fully connected networks. Yet, the entire area of the input is covered by overlapping receptive fields. This is particularly useful in tasks with a large input space such as the pixels of an image. In a convolutional layer, each neuron is connected to a small subset neurons from the previous layer forming a specific region of that layer called the receptive field of the neuron.

Unlike layers of regular neural networks, convolutional layers for image classification are in three dimensions, two spatial dimensions and one for the different filters supported by the layer, usually corresponding to the three colour channels (red, green and blue).

Pooling layers are commonly placed between two convolutional layers in a CNN architecture. Their role is reduce the size of the two spatial dimensions typically by applying the max function on a small region while keeping the filters dimension. This downsampling is performed with a given stride which is the size of the side of the squares of points considered per pooling operation and the most common method is max pooling where the maximum single point value from the square is passed to the next layer as shown in Figure 2.7.

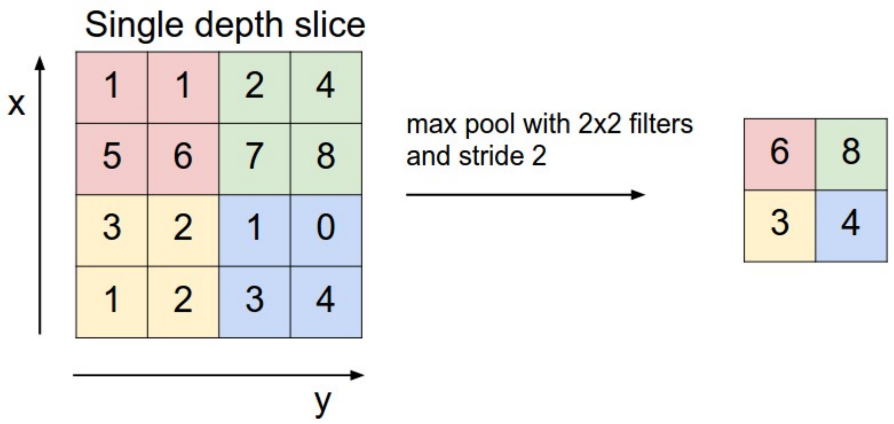


Figure 2.7: Pooling in CNNs [8]

ReLU layers apply the rectified linear unit (ReLU) function defined as $f(x) = max(0, x)$ to each element of the previous layer. This layer produces non-linearity and can also be treated as the activation function of the previous layer.

Fully connected layers contain nodes that are all connected to every single node from the previous layer. These often serve as the final layer, computing the probabilities for each class in classifiers with a softmax function.

Convolutional networks for object detection Object detection consists in locating and classifying the different objects contained in an image. An example of an this task performed with a real-time model is given in figure 2.8. In order to fulfill this function, simple convolutional networks built for image classification need to be extended to identify the regions in which the objects are located so that the output indicates the bounding box for each instance of the object class.
Such a region-based convolutional neural network (R-CNN) was introduced by Girshick et al. [33] in 2014. It implements the selective search algorithm by Uijlings et al. [34] to generate region proposals and feeds each resulting region to a standard CNN as shown on 2.9.

Later improvements over this architecture were presented as Fast R-CNN by Girshick et al. in 2015 [35] which only uses one CNN generating a feature map on which the selective seach for regions of interest is performed, and Faster R-CNN by Ren et al. [36] which combines Fast R-CNN with a region proposal network outputting a bounding box together with a score.
Further improvements were achieved in 2016 with the region-based fully convolutional network (R-FCN) by Dai et al. [37], merging region proposals detection and object recognition in a fully convolutional network.
Besides, simpler models relying on a single network and a single evaluation were also introduced in 2016, in particular the You Only Look Once (YOLO) [38] and Single-shot detector (SSD) [39] architectures. These architectures are among the most suitable for applications such as

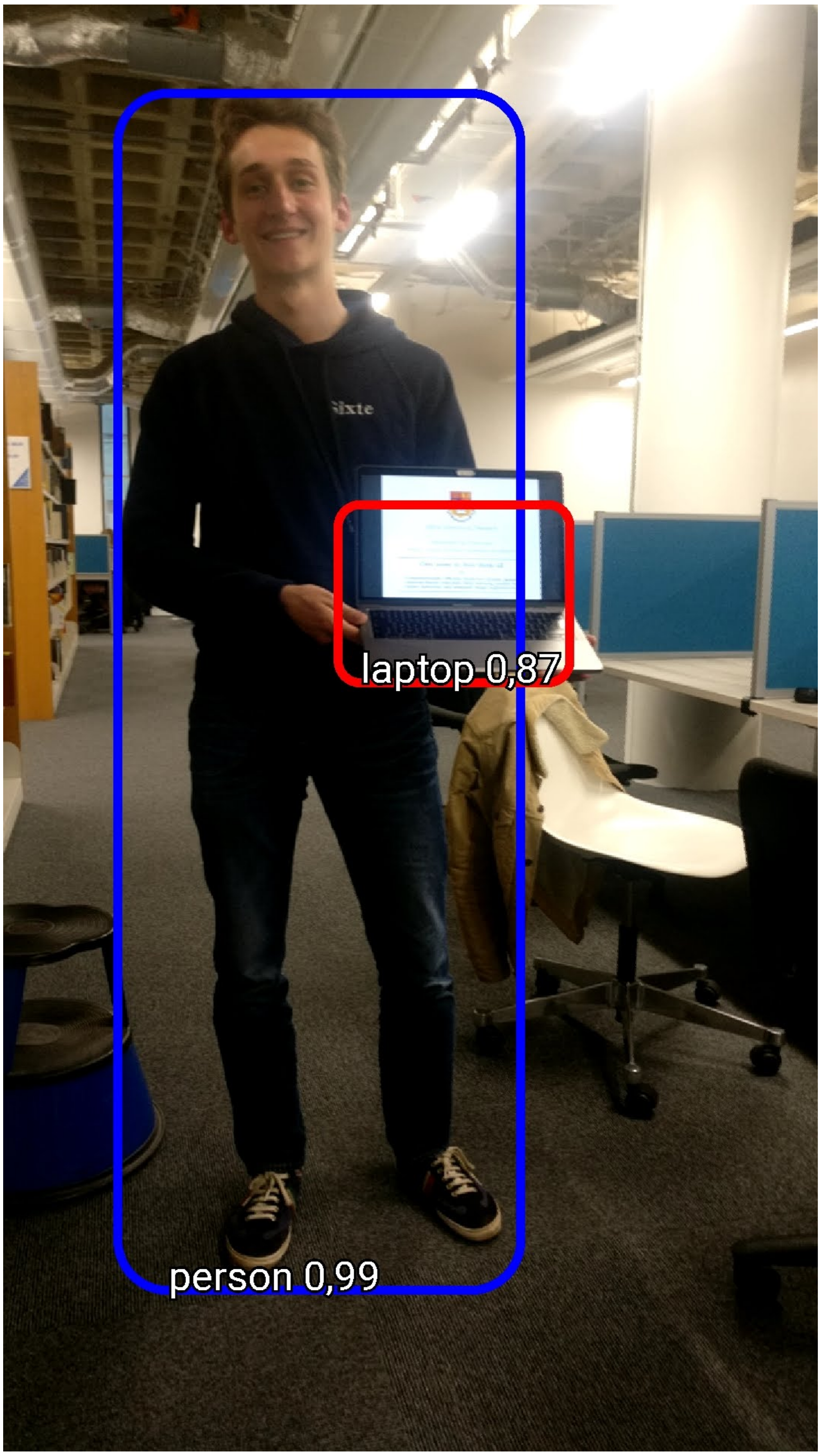


Figure 2.8: Object detection example

autonomous vehicles vision as they allow real-time object detection. Considering it currently achieves the best real-time performance on real-time object detection, we choose YOLOv3 as our target model for object detection related experiments.

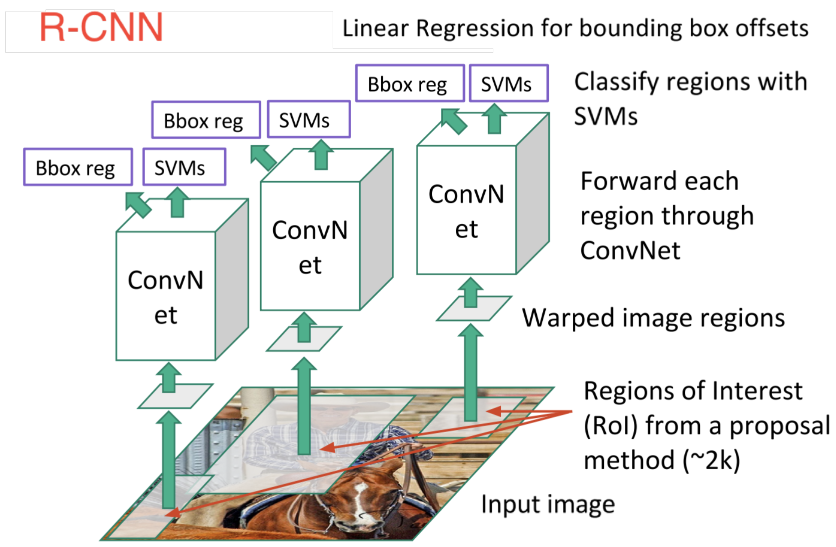

R-CNN
Linear Regression for bounding box offsets
Bbox reg
SVMs
Classify regions with SVMs
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Figure 2.9: R-CNN architecture [9]

Convolutional networks for semantic image segmentation In the case of semantic image detection, the model output should contain a full reconstruction of the image, segmented by objects as well as the corresponding labels as shown in figure 2.10. However, it is computationally expensive to preserve the dimensionality of an image throughout an entire deep neural network. Therefore, a common techique used for such tasks is to perform downsampling followed by upsampling in a network resembling autoencoders. In 2014, Long et al. developed efficient fully convolutional networks [10] based on traditional classification CNNs where the final layer with the softmax function is replaced with an upsampling layer with 32 pixel stride. However, due to the coarse coding of the image produced by downsampling in the first layers and the 32 pixel stride upsampling, the output was not fine-grained. As a countermeasure Long et al. introduced a DAG with skipping edges from the first layers to the last layers, bringing fine grain in otherwise coarse layer (Figure 2.11).

Improvements over the fully convolutional network were introduced in 2015 by Ronneberger et al. with the introduction of the U-Net architecture [11]. It comprises a first half of layers downsampling to support accurate identification of the objects in the image and a second half upsampling in order to support the localisation of these objects. Additionally, connections between two symmetric layers ensure fine-grain of the output.

Several variants of the U-Net have later been proposed, in particular DenseNet by Jégou et al. in which each layer is not only connected to the following and its symmetric but to every other layer in a feedforward fashion [40].

Some of the most recent advances in the field have been about the design of real-time semantic image segmentation models such as SwiftNetRN-18 that address the trade-off between high accuracy and inference speed. Being among the most accurate real-time semantic image segmentation models as of Spring 2019, SwiftNetRN-18 is chosen as our target model for all experiments related to this task. Its architecture is further detailed in section 3.2.2.1.

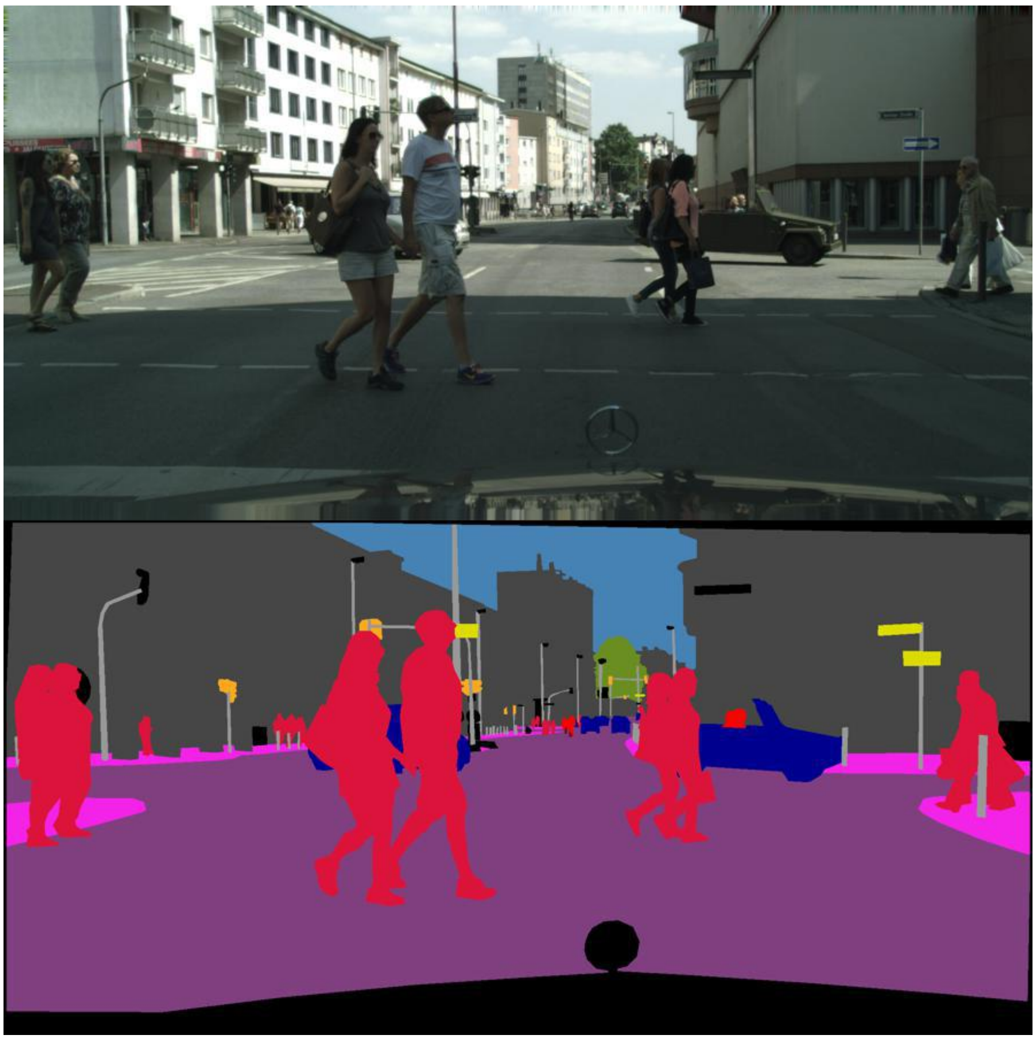


Figure 2.10: Semantic image segmentation example

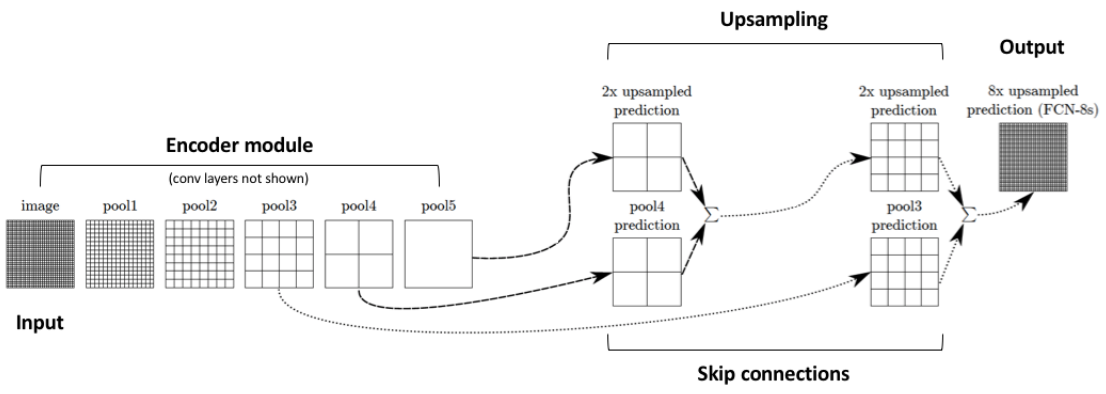


Figure 2.11: Skip connections in a fully convolutional network [10]

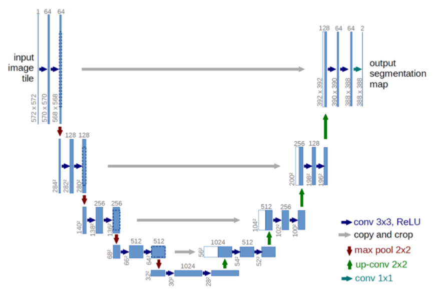


Figure 2.12: U-Net architecture [11]

2.2 Adversarial Machine Learning

Originally machine learning algorithms were designed to work on genuine inputs and assumed training with legitimate samples. However, as machine learning systems become widely deployed in the real world with universal access over the internet, it is becoming crucial to make them resistant to all sorts of attacks that can be crafted by adversaries. The set of possible attacks against machine learning systems was clearly defined in [41] by elaborating a taxonomy of attacks based on the following properties: influence, security violation and specificity. These attacks are part of the field of adversarial machine learning, the set of methods used to compromise machine learning models [42].

2.2.1 Threat model

In 2018, Luis Muñoz González and Emil C. Lupu introduced a threat model [43] improving on the taxonomy of Ling Huang et al. [41] to characterise attacks against machine learning systems. It is based on the attacker's capability, goal and knowledge.

2.2.1.1 Attacker's capability

The attacker's influence may be causative or exploratory.

- A **causative** attacker alters the training data to generate vulnerabilities to be exploited at training time. This is commonly referred to as a data poisoning attack.
- An **exploratory** attacker may exploit vulnerabilities in the machine learning system at test time, for example by querying it, in order to fool the model and achieve an evasion attack.

Data manipulation constraints are often present in an attack scenario and limit the capabilities of an attacker. In the case of evasion attacks against image classifiers, an adversarial sample is crafted to evade its true label but there is a constraint on its appearance: it should still resemble legitimate images that do not evade their true label. These constraints can be formalised by the following: given an initial set of data samples $\mathcal{D}_p$, these can only be mapped to a space $\phi(\mathcal{D}_p)$ to create the adversarial samples. This is often represented by distance constraints between original data and malicious data.

2.2.1.2 Attacker's goal

The attacker's goal is described in terms of security violation, attack specificity and, in the case of attacks against models such as classifiers, error specificity.

A security violation may belong to one of three types:

- An **integrity violation** is achieved by an attack that evades detection by the system and does not affect its normal behaviour. This is typically the case in evasion attacks studied in the next section.
- An **availability violation** is the result of an attack deteriorating the functionality of the system. For classifiers this happens when the attack induces a large increase in the overall error rate.
- A **privacy violation** occurs when an attacker gains sensitive information about the users, the training data or the machine learning system.

Attack specificity may range from indiscriminate to targeted.

- An **indiscriminate attack** is sample agnostic: it aims to degrade the performance of the system on a wide variety of samples.
- A **targeted attack** focuses on a set of specific samples on which the objective is to degrade the system the performance.

Error specificity establishes the distinction between error-specific and error-generic attacks. The former aim to produce any kind of error while the latter generate a specific type of error.

2.2.1.3 Attacker's knowledge

Perfect knowledge attacks assume complete knowledge of the model and its parameters, the data and feature sets and the objective function used to train the system. Although rather unlikely in practice, such attacks are useful for research purposes as they model the worst-case scenario from the defender's point of view.

Limited knowledge attacks on the other hand assume some knowledge is not available to the attacker:

- In limited knowledge attacks with surrogate data the attacker does not have access to the actual training data but may use a surrogate dataset with a similar distribution.
- Limited knowledge attacks with surrogate model may exploit the actual training set of the system. However, the attacker does not know anything about the model itself and so uses a surrogate model to design the attack.

Poisoning Training time attacks enter the category of aforementioned causative attacks on classifiers. Their general threat model may assume white-box or black-box setting based on the attack. It is also generally assumed that the size of the inputs of the system is known and that knowing an input the true label associated with that input is known. In a training time attack, the goal of the attacker may be to compromise integrity and availability of the target classifier. Its main specificity is the ability to craft a limited number of training samples that will be learned by the target system. Such an attack is realistic as it is common practice in real world machine learning systems to exploit user produced data to retrain and improve a classifier. These causative attacks are generally denoted as data poisoning attacks [44]. Poisoning attacks form an active field of study and several countermeasures have been developed in order to detect [45] or prevent them [46].

Some poisoning attacks have already occurred in the real world, one of the most famous casualty being Tay, a Microsoft chatbot released on Twitter that was supposed to interact in a positive way with Twitter users but turned racist in less than 24 hours due to poisoning attacks [47].

Evasion Classification time attacks on the other hand are exploratory attacks that usually exploit vulnerabilities of the learning algorithm to achieve any of the three possible security violations: integrity or privacy. For the purpose of this report we will focus on integrity attacks, also known as evasion attacks which we discuss more in depth in the following section.

The crafting of adversarial samples can be formulated as an optimisation problem. Thus, in order to formalise the problem, we define the threat model with its constraints and objectives from the adversary's point of view: **knowledge**, **capabilities** and **goals**.

Just like poisoning, evasion attacks have also been implemented in the past in real world applications. An iconic example is the Cerber ransomware which possesses the ability to evade detection from the machine learning component of a number of antivirus softwares [48].

2.2.2 Evasion attacks

Evasion attacks are exploratory integrity attacks: the attacker's capabilities are limited to queries to the machine learning system, the number of which is often limited in order to avoid detection.
In a white-box attack the attacker's knowledge includes some access to the target model and its parameters. Such information is supposedly not accessible without insider knowledge or reverse engineering but it has been shown that black-box attacks can be exploited by crafting adversarial samples on an adversary trained surrogate model and then applying them to the target model using the **transferability** property [49]: adversarial samples that impact one model often affect other models trained to perform the same task, even if their architectures and training sets differ.
In a pure black-box attack on the other hand, the attacker assumes no knowledge of the target model and its parameters and does not build a surrogate model but only learns about the target model by querying it. Finally, in an evasion attack the goal of the attacker is to create one or more malicious samples that will be classified as legitimate as shown in Figure 2.13. This figure shows how evasions happen: since a model only has a finite number of training samples, in general its decision boundary does not exactly fit the task decision boundary. Thus, it is possible to alter test points to generate closely related malicious test points that lie in between the task decision boundary and the model decision boundary and will be misclassified.

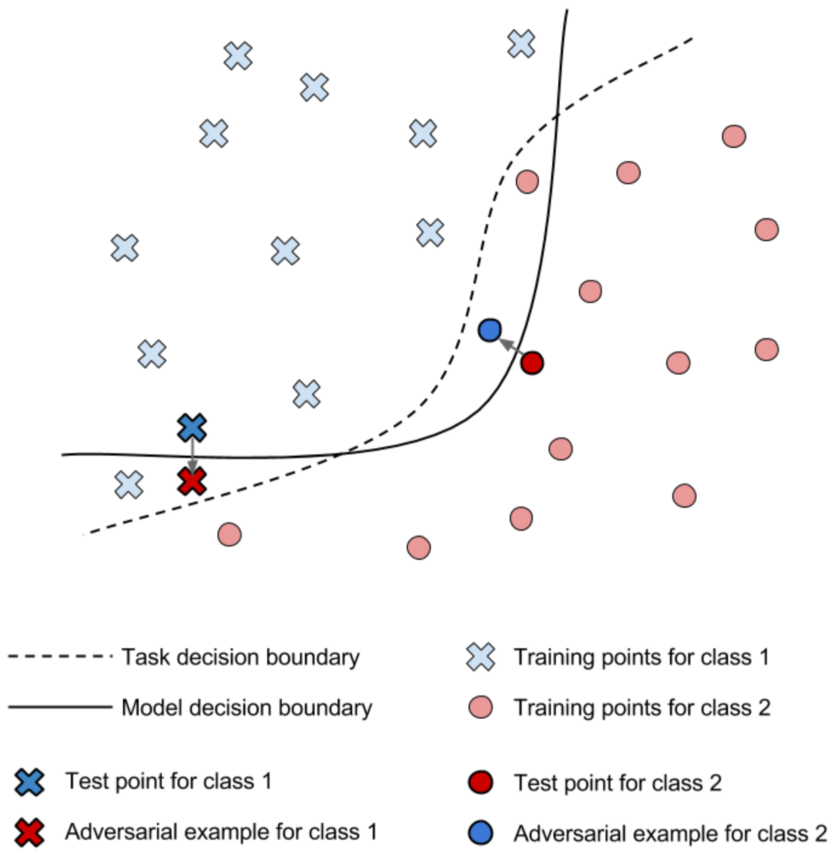

Task decision boundary
Model decision boundary
Training points for class 1
Training points for class 2
Test point for class 1
Adversarial example for class 1
Test point for class 2
Adversarial example for class 2

Figure 2.13: Evasion attack [12]

2.2.2.1 Findings by Szegedy et al.

In 2013, Szegedy et al. introduced the very first evasion attack based on slight perturbations almost imperceptible to the naked eye [13]. Their aim was given an image to generate another image similar under the L_2 distance classified under a different label, which yields the optimisation problem in the non trivial case where $f(x) \neq l$:

$$\min ||r||_2$$
$$\text{s.t. } f(x+r) = l \text{ and } x + r \in [0,1]^m$$

where $x \in \mathbb{R}^m$ is an image, $f : \mathbb{R}^m \rightarrow \{1...k\}$ is a classifier mapping from a feature vector to a label and r is the perturbation to add to x in order to produce the most similar adversarial sample classified under l.
However this optimisation is a hard problem and was thus approximated by a box-constrained L-BFGS using line search to find the minimum $c > 0$ such that $f(x+r) = l$:

$$\min c|r| + loss_f(x+r, l) \text{ s.t. } x + r \in [0,1]^m$$

This proved particularly effective as illustrated in Figure 2.14 and paved the way to a series of evasion attacks using distance metrics (L_p norm) to craft hard to detect adversarial samples.

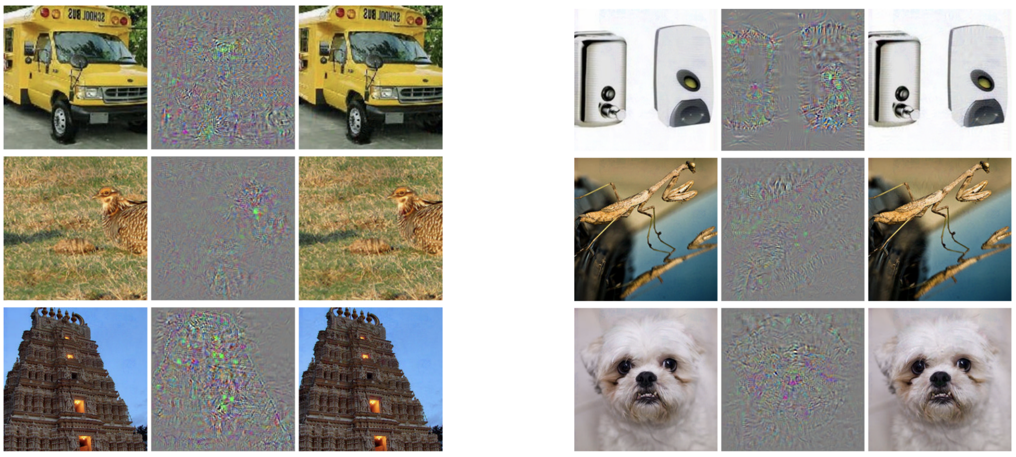


Figure 2.14: Attack by Szegedy et al. with original samples (left), adversarial perturbations (center) and adversarial samples (right). All adversarial samples on the right were classified as “ostrich, Struthio camelus” [13]

.

2.2.2.2 Fast Gradient Sign Method (FGSM)

The fast gradient sign method was introduced in 2014 by Goodfellow et al. [50]. It is a method to produce adversarial samples that focuses on fast creation of samples rather than the distance between original and adversarial sample. It was successfully tested on the MNIST dataset of handwritten digits and given an image x it creates the adversarial example x' as follows:

$$x' = x - \epsilon \cdot sign(\nabla loss_{F,t}(x))$$

where ϵ is small enough to avoid detection and t is the target label for the adversarial sample. This method changes the value of each feature based on the sign of the gradient of the loss function. This was later improved by Kurakin et al. who developed iterative gradient sign [51]. The FGSM was tested by Arnab et al. [52] on semantic segmentation models. For $\epsilon = 32$, it proved very effective on the Cityscape dataset with intersection over union (I ratios of 2.5% on PSPNet and 2.8% on Dilated Context. However the attack was less successful on Pascal VOC with respectively 27.9% and 12.2%.

2.2.2.3 L_2 attack algorithm of Carlini and Wagner

In 2016, Carlini and Wagner introduced various attacks [53] using different distance metrics: L_2, L_0 and L_∞ to limit the visual difference between original samples and adversarial samples. Furthermore, their research showed that a defense defeating their L_2 attack would also defeat the L_0 and L_∞ attacks which makes the L_2 attack a particularly interesting case.
Considering a full neural network F with a softmax activation function where $Z(x)$ is the output of all layers except the softmax function, ie $F(x) = softmax(Z(x))$, the attack is formulated as the following optimisation problem:

$$min||\tfrac{1}{2}(tanh(w) + 1) - x||_2^2 + c \cdot f(\tfrac{1}{2}(tanh(w) + 1))$$

with the loss function f defined as: $f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t - \kappa)$
where:

- $max\{Z(x')_i : i \neq t\} - Z(x')_t$ is the distance between t and the most likely other class.
- κ controls the strength of the adversarial examples which increases with κ.

This produces the adversarial sample: $x' = \frac{1}{2}(tanh(w) + 1)$.

This attack is particularly innovative in that it looks for minimum perturbations to achieve successful evasion.

2.2.2.4 Jacobian-based Saliency Map Attack

The JSMA, introduced in 2016 by Papernot et al. [54] is an evasion attack to achieve targeted misclassification: given a classifier F, it aims to make the output $F_c(x)$ of input x for class c greater than the output $F_{c'}(x)$ of input x for any class $c \neq c'$. The method iteratively perturbs a specific feature until the targeted class c achieves $c = argmax_i(F_i(x))$. This attack achieved a high degree of success on the MNIST digits dataset.

2.2.2.5 Universal Adversarial Perturbations

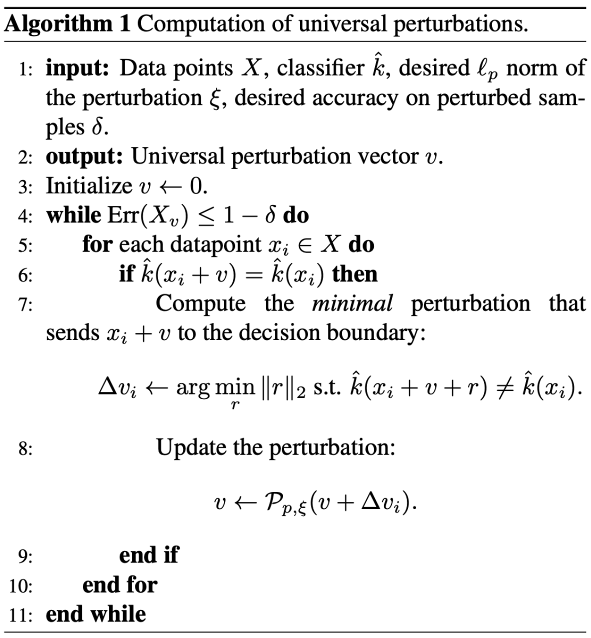
In 2017 were developed universal adversarial perturbations [14]: single perturbations effective to generate adversarial samples on a large number of test samples for a given classifier. Given a distribution μ of images in $\mathbb{R}^d$ and $\hat{k}$ a classification function, the attackers aims to find $v \in \mathbb{R}^d$ such that:

$$\hat{k}(x+v) \neq \hat{k}(x) \text{ for most } x \sim \mu$$

Formally, the attacker needs to meet two constraints:
- $||v||_p \leq \xi$
- $\mathbb{P}_{x\sim\mu}(\hat{k}(x+v) \neq \hat{k}(x)) \geq 1-\delta$

where ξ controls the magnitude of the perturbation and δ the error rate of the classifier to be obtained on images from μ.

Satisfaction of these constraints can be reached using the algorithm in Figure 2.15.



Algorithm 1 Computation of universal perturbations.

```
1:  input: Data points $X$, classifier $\hat{k}$, desired $\ell_p$ norm of
    the perturbation $\xi$, desired accuracy on perturbed sam-
    ples $\delta$.
2:  output: Universal perturbation vector $v$.
3:  Initialize $v \leftarrow 0$.
4:  while Err($X_v$) $\leq 1 - \delta$ do
5:      for each datapoint $x_i \in X$ do
6:          if $\hat{k}(x_i + v) = \hat{k}(x_i)$ then
7:              Compute the minimal perturbation that
    sends $x_i + v$ to the decision boundary:
                $\Delta v_i \leftarrow \arg\min_r \|r\|_2$ s.t. $\hat{k}(x_i + v + r) \neq \hat{k}(x_i)$.
8:              Update the perturbation:
                $v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i)$.
9:          end if
10:     end for
11: end while
```

Figure 2.15: Universal adversarial perturbations algorithm [14]

Note that the constraint on ξ is met by projecting $v + \Delta v_i$ on the l_p sphere of radius ξ in order to update the perturbation.

This algorithm was tested using both L_2 and L_∞ norms on state of the art deep neural networks. Both types of perturbations achieved significantly high error rates, above 77% on both the set used in the algorithm and the validation set on all DNNs tested (CaffeNet, VGG-F, VGG-16, VGG19, GooGleNet and ResNet-152). The universal perturbations developed against these architectures can be seen in Figure 2.16. Besides, the perturbations obtained were shown to exhibit cross-model universality: universal perturbations obtained for a specific architecture

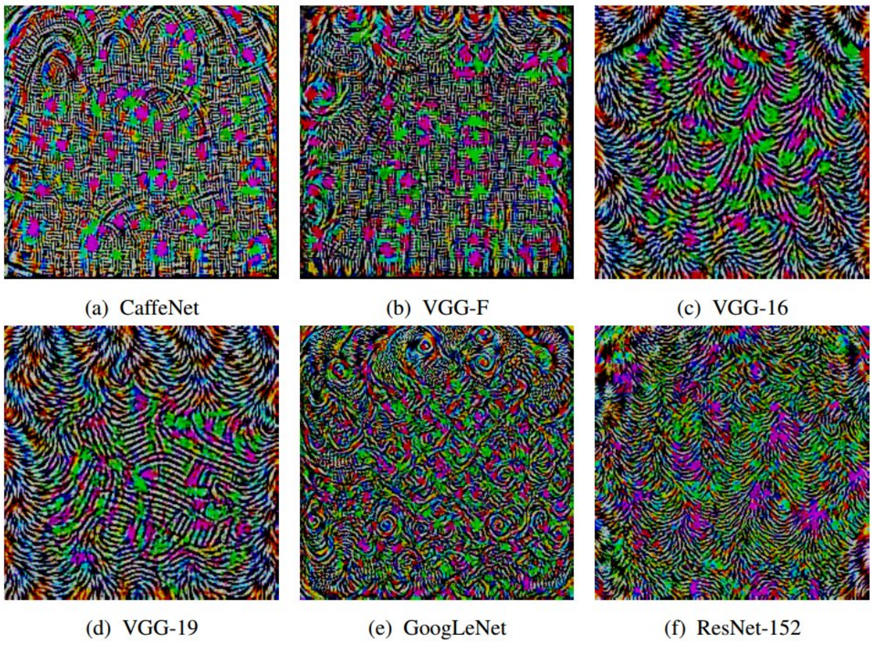


Figure 2.16: Universal adversarial perturbations developed against various neural networks using L_∞ and $\xi = 10$ [14]

also have relative effectiveness against other architectures.

In 2018 Khrulkov and Oseledets introduced a different technique using the singular vectors of the Jacobian matrices of the hidden layers of a neural network to produce universal adversarial perturbations [55]. The visual similarity between these patterns and procedural noise perturbations such as Gabor and Perlin noise motivated our work relying on those noise functions to generate adversarial patterns.

More recently, Co et al. introduced a different attack based on Gabor noise perturbations [56]. By generating noise patterns that act as universal adversarial perturbations on various classifiers, they showed the sensitivity of these models to Gabor noise and suggested that further research should be conducted to better understand the resulting misbehaviour of deep convolutional networks.

The universal adversarial perturbations by Moosavi-Dezfooli et al. [14] were also successfully adapted by Metzen et al. [15] to craft attacks against semantic image segmentation systems targeting a specific segmentation. Metzen et al. also revealed the existence of perturbations resulting causing a similar segmentation on arbitrary samples as well as attacks removing an entire target class as shown with the pedestrian class in Figure 2.17. This attack first applies semantic segmentation on the original image to identify the instances of the target class to be hidden and the background classes. An adversarial sample is then built by keeping the background unchanged and making pixels of the target class classified in the nearest neighbouring class.

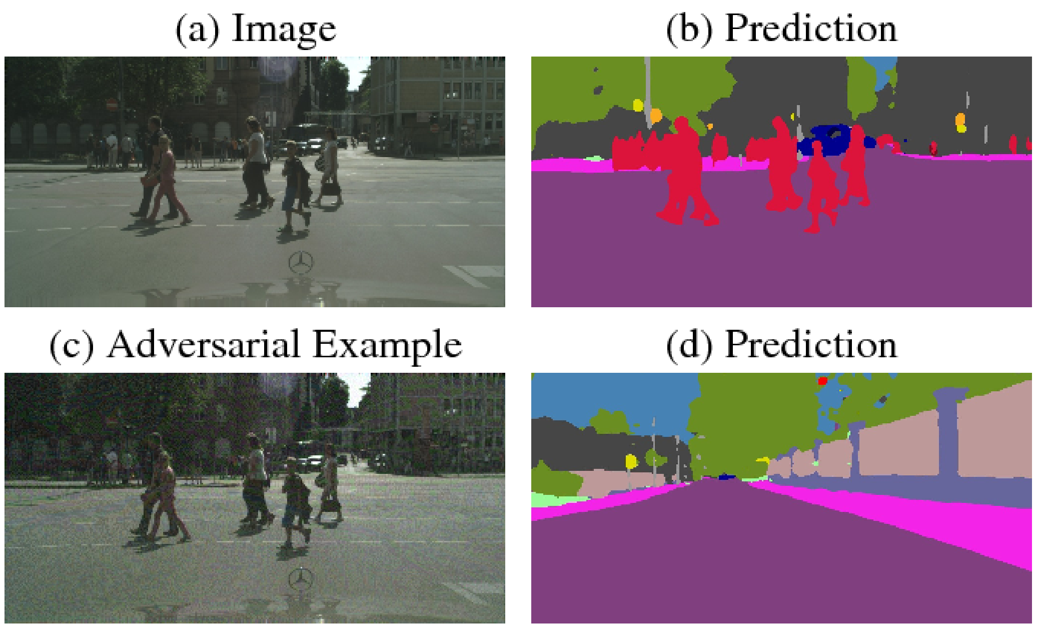


Figure 2.17: Universal adversarial perturbations against semantic image segmentation [15]

2.2.2.6 Procedural Noise attacks

In [16], Kenneth T. Co et al. developed attacks using the pre-existing Perlin noise function; as this noise is widely used to create natural looking textures it was thought likely to be able to fool a classifier on a wide variety of images as shown in Figure 2.18.

These attacks target a black-box classifier, with only the data types of the inputs, outputs and

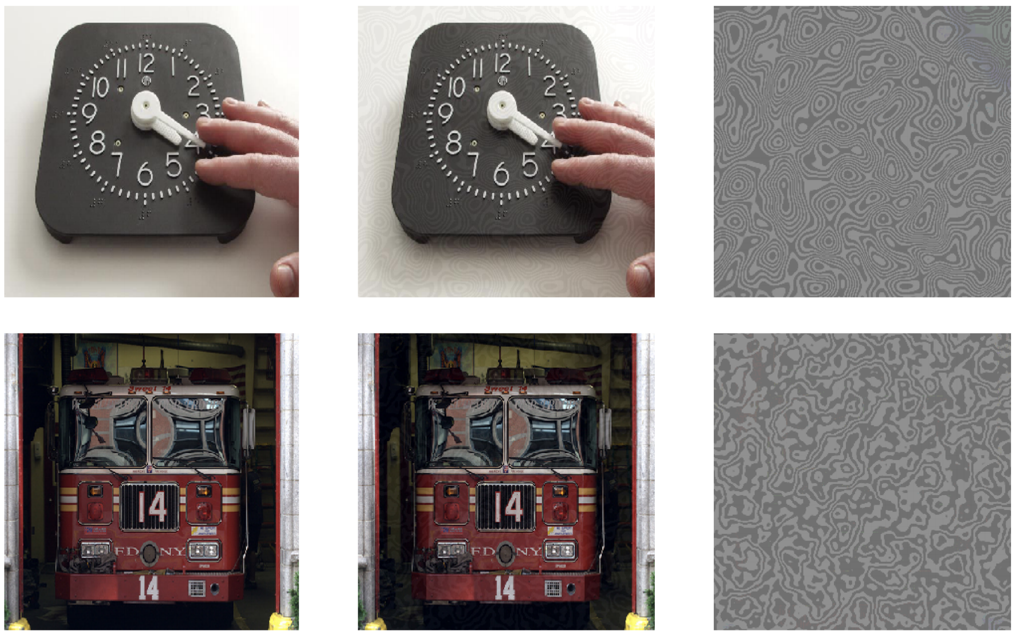


Figure 2.18: Perlin noise attack adversarial samples. From left to right and top to bottom: original image classified as analog clock (28.53%), adversarial sample classified as barbell (29.84%) and perturbation used (magnified), clean fire truck image (92.21%), malicious image classified as wall clock(18.32%) and corresponding perturbation [16]

.

class labels of the test samples being known by the attacker. They assume query access to the classifier with possibly a probability vector output and knowledge of the true class labels $\tau(x)$. Here, the attacker aims to achieve top n evasion, meaning the true label is not among the n most likely labels appearing in the probability output vector, while limiting both the norm of the perturbation and the number of queries.
Let $F_i(x)$ be the probability that classifier F attributes to x belonging to class i. Let $T_n(x)$ be the nth highest probability for any class by the classifier on input x. Given that top n evasion is achieved for $F_{\tau(x')}(x') < T_n(x')$, the optimisation problem is

$$\begin{aligned} &\min_{\theta} F_{\tau(x)}(x + G(\theta)) - T_n(x + G(\theta)) \\ \text{s.t. } & x + G(\theta) \in [0,1]^d,\ ||G(\theta)|| < \epsilon,\ q < q_{max} \end{aligned}$$

To reach top n evasion, it is sufficient to have $\min_{\theta} F_{\tau(x)}(x + G(\theta)) - T_n(x + G(\theta)) < 0$. Thus, it is the stopping condition of the algorithm.
This black-box attack leverages the efficiency of Bayesian optimisation to find the optimal parameters θ while limiting the number of queries to avoid detection. This attack requires to choose the domain of the parameters θ, boundaries (ϵ and q_{max}) and distance metrics. Using a grid search, boundaries on the parameters θ (frequency of the Perlin noise function ν, number of octaves ω, lacunarity κ and frequency of the sine colour map function ν_{sine}) were chosen as follows: $\nu \in [20, 80]$, $\omega \in 1, 2, 3, 4$, $\kappa \in [1.7, 2.2]$ and $\nu_{sine} \in [2, 32]$. Based on experiments $q_{max} = 100$ appears to be appropriate considering the complexity $O(q^3)$ of exact inference in Gaussian process regression for Bayesian optimisation and the experiments showing that the error rate comes to a plateau before q reaches 100. Following previous studies, an ϵ of $\frac{16}{256}$ is chosen together with the l_∞ norm.

Two different Perlin noise attacks were conducted: Perlin-R selecting the parameters of the noise generating function at random and Perlin-BO using Bayesian optimisation. These achieved a high error rate against simply trained classifiers v3 and IRv2 as well as adversarially trained $IRv2_{adv}$ and $IRv2_{adv-ens}$ using the ImageNet dataset. Random noise addition caused a top 1 evasion rate of at least 40.2% on all considered classifiers, Perlin-R 70.9% and Perlin-BO 89.5%. On top 5 evasion, random noise achieved at least 9.6% error rate, Perlin-R 40.8 and Perlin-BO 45.2. It was also noted that the error rate achieved by the Perlin-BO attack reaches a plateau after fewer than 20 queries.

The results demonstrate a major vulnerability of the current classifiers to Perlin noise attacks especially with parameter optimisation achieving a high error rate with very few queries. However, Bayesian optimisation appears to be less effective for top 5 evasion than for top 1 evasion when compared to random selection of noise parameters. It was hypothesised that the diminished impact of Bayesian optimisation together with the fast convergence of the error rate in terms of the number of queries were most likely due to the simplicity of the Perlin noise function used. Therefore, an attack efficiently exploiting the color map or based on a more complex noise function is likely to show an improved effectiveness.

2.2.3 Evasion countermeasures

Evasion attacks being a serious threat to deep neural networks, several countermeasures against them have been developed. They rely on a variety of mechanisms, from adversarial training and generative adversarial networks to detectors and denoising. In the following, we will cover some of the major defenses against evasion attacks that have been introduced in the past few years.

2.2.3.1 Defensive distillation

Defensive distillation, introduced in 2016 by Papernot et al. [25], consists in training a deep neural network on the softmax outputs of another neural network itself trained on the data. This technique reduces the network dimensionality to defend against adversarial perturbations which produced encouraging results, reducing the success of adversarial samples to less than 0.5% on the MNIST dataset and 5% on the CIFAR10 dataset without increasing the error rate on legitimate samples. Nevertheless, defensive distillation was proven not robust to adversarial samples by Carlini & Wagner [57].

2.2.3.2 Classifier based adversarial samples detection

In 2017 Grosse et al. showed that adversarial samples are not drawn from the same distribution as original data and developed a statistical test to detect adversarial inputs and trained a model with an additional output identifying adversarial samples with more than 80% accuracy or causing the adversarial inputs to require a 150% increase in the perturbation [58]. This method is a variant of retraining, analogous to adding an additional class labeled "malicious sample". Zhitao Gong et al. [59] introduced a similar defense relying on a secondary classifier trained with the training data. Both defenses achieve a near-100% success rate in classifying adversarial samples generated by the fast gradient sign method and JSMA on the MNIST dataset. However, they appear to perform poorly on more complex data set such as CIFAR and are not robust against more sophisticated attacks such as the L_2 attack by Carlini and Wagner (C&W). Another detection neural network based defense by Jan Hendrik Metzen et al. [21] exploits the convolutional layers in a ResNet architecture achieves 99% accuracy in the detection of adversarial samples on CIFAR against fast gradient sign and JSMA but is also vulnerable to the C&W attack as shown by Carlini and Wagner [60].

2.2.3.3 Principal Component Analysis based defenses

Other defenses are based on principal component analysis (PCA) to differentiate adversarial from legitimate samples. PCA is a method to perform a transformation from N to k dimensional data $k < N$ by preserving the k principal components.
Hendrycks & Gimpel [22] introduced such a method exploiting the observation that most adversarial samples apply a larger weight on the first principal components than original samples. Another method by Bhagoji et al. [61] performs dimensionality reduction on the training data and uses the reduced data set to train the classifier. Both methods are effective on MNIST but

have been proven to be specific to that data set and not robust to perfect knowledge attacks. [60]. The defense by Li et al. [23] applies PCA to the outputs of each convolutional layer to serve as input to a classifier. The input of the convolutional network is considered malicious if at least one of the classifiers rejects its input. This defense was tested on ImageNet but does not appear to be robust to the C&W attack in the zero-knowledge model, even on simpler data sets.

2.2.3.4 Distribution based detection and dropout randomisation

Distribution based detection methods rely on the potential difference between the distribution of original data and the distribution of adversarial samples in order to estimate the likelihood of a sample being malicious. This is used in Kernel density estimation by Feinman et al. [24] which is powerful on MNIST but insufficient on CIFAR against the C&W attack in the zero-knowledge setting.

Beside distribution based detection, in order to use Bayesian neural network uncertainty, a detection technique to measure the uncertainty of a network on an input, Feinman et al. [24] add randomisation by dropout, the act of removing a number of nodes from the network at random. The dropout randomisation strategy is based on the assumption that under different dropout conditions an original sample is repeatedly classified under the same label while an adversarial sample may be classified under different labels. This defense is effective against the traditional C&W attack in the zero-knowledge and full-knowledge threat models. However, in the full-knowledge scenario it is vulnerable to a modified version of C&W with using a different loss function on both MNIST and CIFAR.

2.2.3.5 Obfuscating gradient and adversarial training

Most recent defenses mostly belong the masking gradient category defined by Papernot et al. in 2016. It consists of all the attacks constructing a model with impractical gradients, which includes defensive distillation [62] and adversarial training based on single-step techniques to maximise a linear approximation of the loss function of the model. However Tramèr et al. [63] have shown that such simple adversarial training learn weak perturbations but remain vulnerable to strong ones. Besides, as pointed out by Athalye et al. [64], robustness to preexisting attacks, in particular gradient iteration attacks such as fast gradient sign is not sufficient. These defenses relying on shattered gradient, stochastic gradient or vanishing and exploding gradients should also be robust against adaptive attacks in order to be considered as strong defenses.
A particularly robust defense against black-box attacks that was also proposed by Tramèr et al. [63] is ensemble adversarial training, a method that transfers perturbations from different models in order to enhance the training data. However, this technique was proven not robust to Perlin noise attacks by Co et al. [16].

2.3 Procedural noise

As shown by Kenneth T. Co et al. [16], procedural noise functions may constitute a strong basis for evasion attacks in a zero-knowledge scenario. Procedural noise function possesses a strong set of properties that supported its use in computer graphics and motion pictures that may also be useful in adversarial samples generation: it is continuous, multi-resolution, non-periodic, parameterised and randomly accessible [17]. Thanks to its continuousness and multi-resolution properties, procedural noise can be applied to various tasks on different data sets with varying dimensions. Besides, non-periodicity and parameterisation make it flexible, hence hard to detect and subject to optimisation. Finally, random accessibility provides constant time evaluation of the function which is a useful guarantee for a small complexity attack. Procedural noise is composed of three main categories: lattice gradient, sparse convolution and explicit noise. The latter requiring pre-processing and storage, it is not adapted to cost-efficient attacks. Its main constituents are Wavelet and anisotropic noise. Different noise functions are shown in Figure 2.19

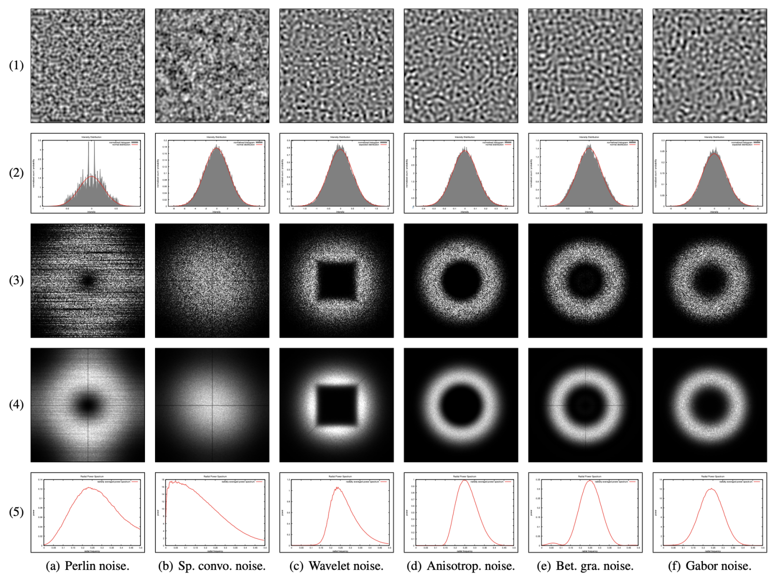


Figure 2.19: Comparison of noise function by noise (1), amplitude distribution (2), periodogram (3), power spectrum estimate (4) and radially averaged power spectrum estimate (5) [17]

2.3.1 Lattice gradient noises

Lattice gradient noises generate noise by interpolating or convolving random values and/or gradients defined at the points of the integer lattice [16]. The most widely used is Perlin noise, which was the basis for the attack by Kenneth T. Co et al. [16]. It was first developed by Perlin in 1985 [65] and improved in 2002 [66]. In order to compute the value at point (a, b), the lattice points $(i, j), i \in |a|, |a| + 1, j \in |b|, |b| + 1$ are considered. Then, the gradient $g_{ij} = V[Q[Q[i] + j]]$ is computed using pre-computed arrays Q and V. The four resulting functions are then bilinearly interpolated using $s(t) = 6t^5 - 15t^4 + 10t^3$ to create the Perlin noise p(a, b) at (a, b). To further control the noise pattern, it can be transformed into $S(a, b)$ using frequencies ν_a and ν_b and number of octaves ω: $S(a, b) = \sum_{n=1}^{\omega} p(a \cdot 2^{n-1}\nu_a, b \cdot 2^{n-1}\nu_b)$
Other lattice gradient noise have been presented such as the Simplex noise, a variation of the Perlin noise based on a simplex grid [67] and the lattice convolution noise by Wyvill and Novins [68] that uses a denser grid. Some lattice gradient noise produce nature inspired patterns which may be relevant to some evasion tasks. It is the case of flow noise by Perlin & Neyret [69] and curl noise by Bridson et al. [70]. Finally, better gradient noise, an improvement on Perlin noise

was introduced by Kensler et al. in 2008 [71] and may be worth considering for adversarial machine learning as it has less directional preference and better quality on surfaces.

2.3.2 Sparse convolution noises

Sparse convolution noises are formed by the summation of randomly positioned and weighted kernels. The development of sparse convolution noise was motivated by increased spectral control. Its main constituents include sparse convolution [72], spot [73] and Gabor noise [74].

Gabor noise is constructed using sparse convolution with a Gabor kernel, the product of a Gaussian and a two-dimensional cosine $g(a,b) = \kappa e^{-\pi\alpha^2(a^2+b^2)} \cos[2\pi\nu(x\cos\omega + y\sin\omega)]$ to generate the noise $S(a,b)$ at point (a, b):

$$S(a,b) = \textstyle\sum_{i=1}^{N} w_i g(a-a_i, b-b_i; \kappa, \alpha, \nu, \omega)$$

where (a_i, b_i) are random points and w_i random weights [74]. The parameters κ, α, ν and ω are respectively the magnitude and inverse width of the Gaussian used in the Gabor kernel, the frequency and orientation of the two-dimensional cosine [17].
As Gabor noise contains more parameters than Perlin noise, it appears more expensive to optimise using Bayesian optimisation for evasion attacks but might be more flexible.

Chapter 3

Design

3.1 Threat model

3.1.1 Attacker's capability

All the attacks conducted as part of this project attempt to evade the target model by leveraging its weaknesses at inference time without altering the inner behaviour of the system. As such they can all be classified as exploratory.

Furthermore, these attacks are all subject to the same data manipulation constraints: crafting of adversarial samples may only be achieved by superimposition of an adversarial pattern within the L_∞ norm of $\frac{16}{256}$ over a legitimate samples. This means that while typical pixel values are within the interval [0, 255] for each of the red, green and blue colour channels, the noise added to any channel of any pixel of the image has to be in the interval [-16, 16]. This L_∞ norm is the most suitable metric to clip a generated procedural noise pattern in order to guarantee visual similarity between the legitimate samples x and their malicious counterparts $x + G(\theta)$. Indeed, unlike other norms such as L_0 and L_2 often used in the literature, it is applied pixelwise, which makes it a very simple constraint to integrate in a procedural noise attack. The point of such a norm constraint is to ensure the attack does not alter the ground truth, i.e. the perfect detections or segmentations that a human could make. In some instances we even apply a stricter norm restriction in order to make the perturbations quasi-imperceptible to the human eye.

3.1.2 Attacker's goal

The adversarial methods and experiments described in this report constitute security violations of the target models with a focus on integrity violation. However, these attacks do not challenge the performance of the model on legitimate samples and do not attempt to expose information about the training data or the inner architecture of the model.

Then, concerning the attack specificity, we attempt both indiscriminate and targeted attacks on both models. An indiscriminate or untargeted attack aims at affecting the overall performance

of the target model while a targeted attack focuses on decreasing the performance on a particular class.
Some aim at crafting a specific adversarial sample for each individual image (image-specific) while others aim to find universal adversarial patterns (image-generic or image-agnostic) that can be successfully applied to a wide array of legitimate images.
Finally, both error specific and error generic attacks are explored with various degrees of success in this project.

3.1.3 Attacker's knowledge

The various noise attacks described here follow strict limitations concerning the attacker's knowledge. In particular, the number of queries of a model in order to craft a specific pattern is limited to $q = 25$ with reports of the results obtained for smaller query budgets. This reduces the amount of knowledge that can be extracted from the target models and ensures fast and cheap computation of the optimisation.
Besides, all image specific attack assume no knowledge of the ground truth with the crafting of adversarial samples, only relying on distance metrics between the output of the model on a legitimate sample and its malicious variant. This is made possible by the fact that in most cases the model output on benign inputs is a good approximation of the ground truth considering its high performance. On image-generic attacks on the other hand, the evaluation metric is based on the entire dataset and therefore we simply use the ground truth as the basis for metrics to estimate the effect of the attacks.

3.2 Project setup and objectives

3.2.1 Object detection: YOLOv3

For the task of object detection the target object detector we chose for our experiments is YOLOv3 [75], as it was until April 2019 the most accurate real-time open source object detection model with a reported mean average precision of 57.9 on the COCO dataset with an inference time of 50 milliseconds per frame of 608x608 pixels.

3.2.1.1 YOLOv3 architecture

Unlike prior object detection models [35] [36], it does not perform several evaluation of a simple classifier at different region proposals. Instead, it divides the image into a grid of $13 * 13 = 169$ regions, each of which is responsible for predictions of the class, bounding box and confidence score of objects whose center lies within the region. Then, only the boxes with the highest confidence scores are predicted thanks to non-max suppression in order to avoid duplicates. For feature extraction YOLOv3 uses Darknet-53, which is composed of 53 convolutional layers (1x1 and 3x3), shortcut connections, average pooling and a fully connected layer with a softmax. Then, several convolutional layers are added on top of the feature extractor at 3 different scales with downsampling by 32, 16 and 8 in order to generate predictions

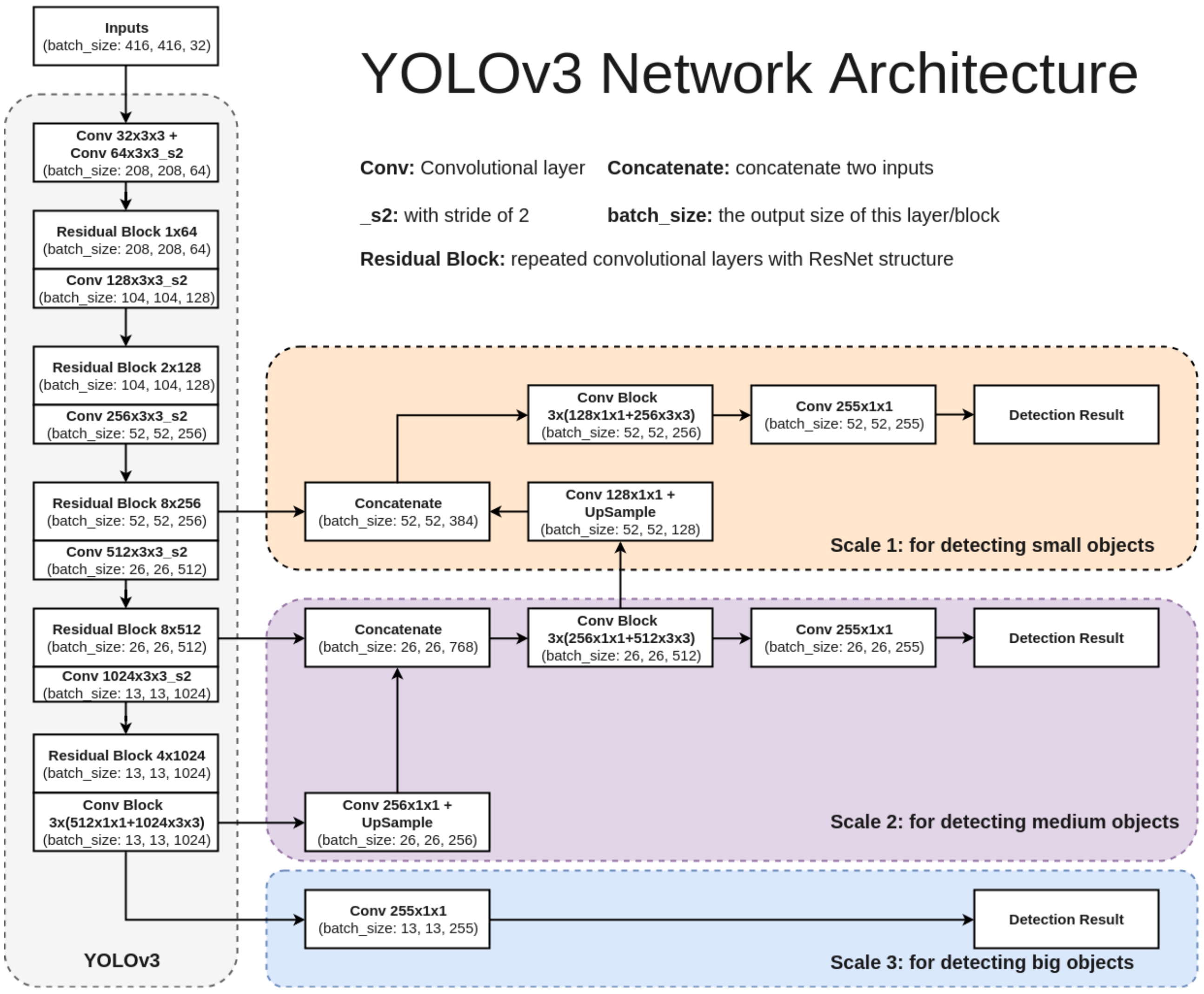

YOLOv3 Network Architecture
Conv: Convolutional layer
Concatenate: concatenate two inputs
_s2: with stride of 2
batch_size: the output size of this layer/block
Residual Block: repeated convolutional layers with ResNet structure
Inputs (batch_size: 416, 416, 32)
Conv 32x3x3 + Conv 64x3x3_s2 (batch_size: 208, 208, 64)
Residual Block 1x64 (batch_size: 208, 208, 64)
Conv 128x3x3_s2 (batch_size: 104, 104, 128)
Residual Block 2x128 (batch_size: 104, 104, 128)
Conv 256x3x3_s2 (batch_size: 52, 52, 256)
Residual Block 8x256 (batch_size: 52, 52, 256)
Conv 512x3x3_s2 (batch_size: 26, 26, 512)
Residual Block 8x512 (batch_size: 26, 26, 512)
Conv 1024x3x3_s2 (batch_size: 13, 13, 1024)
Residual Block 4x1024 (batch_size: 13, 13, 1024)
Conv Block 3x(512x1x1+1024x3x3) (batch_size: 13, 13, 1024)
YOLOv3
Concatenate (batch_size: 52, 52, 384)
Conv Block 3x(128x1x1+256x3x3) (batch_size: 52, 52, 256)
Conv 255x1x1 (batch_size: 52, 52, 255)
Detection Result
Conv 128x1x1 + UpSample (batch_size: 52, 52, 128)
Scale 1: for detecting small objects
Concatenate (batch_size: 26, 26, 768)
Conv Block 3x(256x1x1+512x3x3) (batch_size: 26, 26, 512)
Conv 255x1x1 (batch_size: 26, 26, 255)
Detection Result
Conv 256x1x1 + UpSample (batch_size: 26, 26, 256)
Scale 2: for detecting medium objects
Conv 255x1x1 (batch_size: 13, 13, 255)
Detection Result
Scale 3: for detecting big objects

Figure 3.1: YOLOv3 architecture [18]

of small, medium and big objects as illustrated in 3.1. The output tensor at each scale is of shape $N * N * (B * (4 + 1 + C))$ where N is the width of the regions grid mentioned above, B corresponds to the number of predictions per cell at each scale, 4 is the number of scalars needed to indicate the coordinates of the center of each box as well as its width and height. 1 additional probability score represents the objectness prediction and C others correspond the prediction for every class. The architecture we are using relies on a $13 * 13$ grid, hence $N = 13$, it makes $B = 3$ predictions per cell at each scale and the COCO dataset contains $C = 80$ classes. Thus, our output tensor is $13 * 13 * 255$ at each scale.

While YOLOv3 has a fixed architecture, it also has a number of settings that can alter its behaviour at inference time. For those we decide to keep the original parameters. In particular, the IoU threshold is set 0.5, which implies that only boxes that have an intersection over union of 0.5 or more with the ground truth. We also apply a confidence threshold of 0.3 which discards any detections with a predicted object confidence of less than 30%. The non maximum suppression threshold which defines the minimum PC (probability of an object being present in the bounding box) for each bounding box is set to 0.45. Finally, we keep the original YOLOv3 input image size of 416.

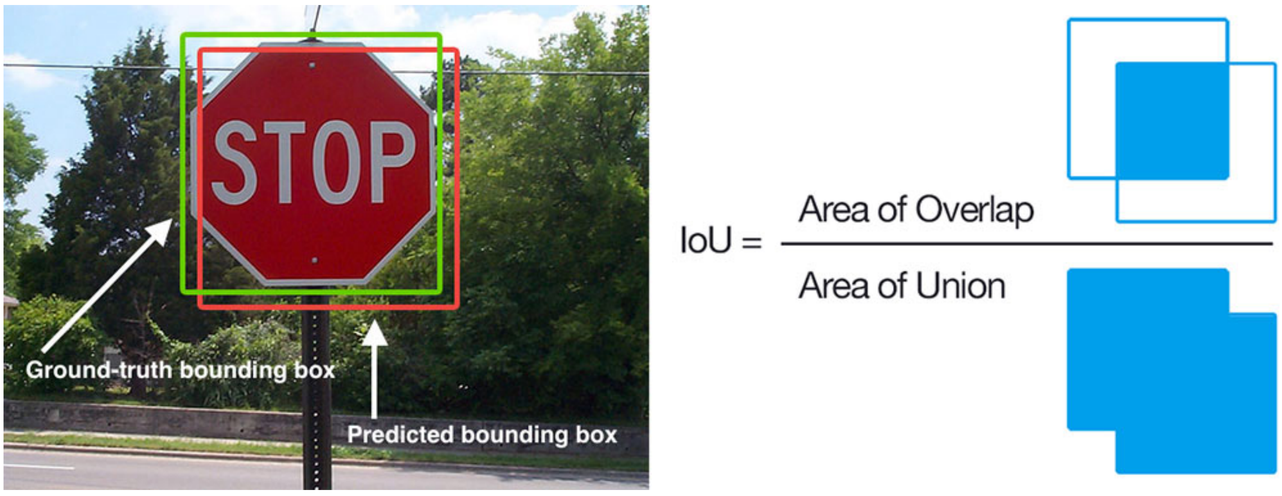


Figure 3.2: Intersection over Union measure [19]

For it is pretrained on ImageNet, we expect Darknet-53 to learn similar low level features to those of deep convolutional ImageNet classifiers vulnerable to procedural noise attacks such as Inceptionv3 [16]. Therefore it appears very plausible for YOLOv3 to also be vulnerable to noise attacks.

3.2.1.2 Object detection evaluation and objectives

Evaluation The most basic building block for the evaluation of object detection models is the intersection over union or IoU. It corresponds for a ground truth bounding box and its corresponding detection bounding box to the area of their intersection divided by the area of their union. The IoU is a measure of the accuracy of a detection between 0 (no overlap or different labels) and 1 which is depicted in figure 3.2.

From the IoU, we can define the mean average precision (mAP) which is the main metric to measure the performance of an object detection model and will serve as a reference to evaluate the effectiveness of our various attacks. It is computed as Throughout this project we will refer to the mAP at IoU 0.5 simply as mAP. This metric is computed as indicated in algorithm 1.

Beside the mAP used to assess the overall performance of an object detection model, individual class average precision may also be used to evaluate the performance of a model on a specific class.

Objectives Our attacks rely on the Pytorch implementation from ultralytics as the black-box target model. This implementation which achieves a mAP of 0.585 on the COCO validation 2014 dataset with pretrained weights, was adapted for the purpose of this project to make it easy to query in order to conduct a wide range of black-box attacks. We introduce untargeted and targeted but also image-specific and image-generic attacks. All of those aim to find optimal noise parameters $\hat{\theta}$ to compromise the model under a specific metric by adding noise $G(\hat{\theta})$ to every original sample x. In general, we define the objectives of the attacks as follows:

Algorithm 1 mAP at IoU 0.5 computation

```
 1: function MAP(gtDetections, yoloDetections)
 2:     for class in classes do
 3:         yoloClassDetections ← filter(yoloDetections) on class
 4:         yoloClassDetections ← sort(yoloClassDetections) on confidence desc
 5:         tp ← 0
 6:         fp ← 0
 7:         for yoloClassDetection in yoloClassDetections do
 8:             match ← false
 9:             image ← yoloClassDetection.image
10:             gtClassDetections ← filter(gtDetections) on class, image
11:             for gtClassDetection in gtClassDetections do
12:                 $iou \leftarrow \frac{overlap(gtClassDetection,yoloClassDetection)}{union(gtClassDetection,yoloClassDetection)}$
13:                 if $iou$ >= 0.5 and detected(gtClassDetection) != true then
14:                     match ← true
15:                     detected(gtClassDetection) ← true
16:                     tp ← tp+1 break
17:             if match == false then
18:                 fp ← fp+1
19:             precision ← $\frac{tp}{tp+fp}$
20:             recall ← $\frac{tp}{len(gtClassDetections)}$
21:             recallToPrecision(recall) ← precision
22:         classAP(class) ← $\frac{1}{11}\sum_{r=0,0.1,...1} \max_{\tilde{r}\geq r} recallToPrecision(\tilde{r})$
23:     return $\sum_{class\in classes} \frac{classAP(class)}{len(classes)}$
```

- Untargeted attack: $\hat{\theta} = arg \min_{\theta} mAP$ such that $||G(\theta)|| \leq \epsilon$
- Targeted attack against class c: $\hat{\theta} = arg \min_{\theta} AP_c$ such that $||G(\theta)|| \leq \epsilon$

In the case of image-specific attacks, a different $\hat{\theta}$ is to be found for every single image while image-generic attacks on a single objective for the entire dataset.

3.2.2 Semantic image segmentation: SwiftNet

In the context of semantic image segmentation, our target model is SwiftNetRN-18 [20], which is the most accurate real-time model on the Cityscapes datasets with a mean intersection over union of 75.5% on the Cityscapes test dataset and 39.9 frames per second as of May 2019. It was trained on the 2975 images of the Cityscapes training set and is evaluated on the 500 images validation set.

3.2.2.1 SwiftNetRN-18 architecture

In a similar fashion to YOLOv3, Swiftnet achieves state-of-the-art results in real-time by relying on an ImageNet pre-trained feature extractor, making it a good candidate for procedural noise attacks experimentations.
It comes in two different architectures: a single scale model and an interleaved pyramidal fusion model. Throughout our experiments, we use the latter whose structure is shown in diagram 3.3. SwiftNet uses an ImageNet pre-trained ResNet-18 encoder as it supports real-time performance and the pre-training on a large dataset together with transfer learning act as regularisation, which is beneficial considering the small size of open-source segmentation datasets. Thus, the SwiftNetRN-18 encoder consists of two series of four convolutional groups or residual blocks (RB in 3.3). The input image is fed to the encoder part of the network at two different scales (HxW and H/2xW/2), one for each series of convolutional groups. Within these groups, the first one downsamples the image by a factor or four and the following ones each by a factor of two. Also, the final residual unit within each block produces a lateral connection by element-wise summation between its input and output prior to the ReLu activation pass as depicted in 3.4. These encoded features of the two series are concatenated at equivalent scale (green cat) and passed through 1x1 convolutions to be projected on the decoder feature space to be incorporated into the decoding module by element-wise summation with the output of the previous upsampling block followed by a 3x3 convolution. A series of 4 upsampling blocks finally restores the original resolution of the image.

3.2.2.2 Semantic image segmentation evaluation and objectives

Evaluation There exists a number of metrics to evaluate the performance of a semantic image segmentation model. In order to do so, we compute by pixelwise comparison between the ground truth and the model segmentation output for each class c the number of true positives TP_c, false positives FP_c and false negatives FN_c. These allow us to define the following metrics:

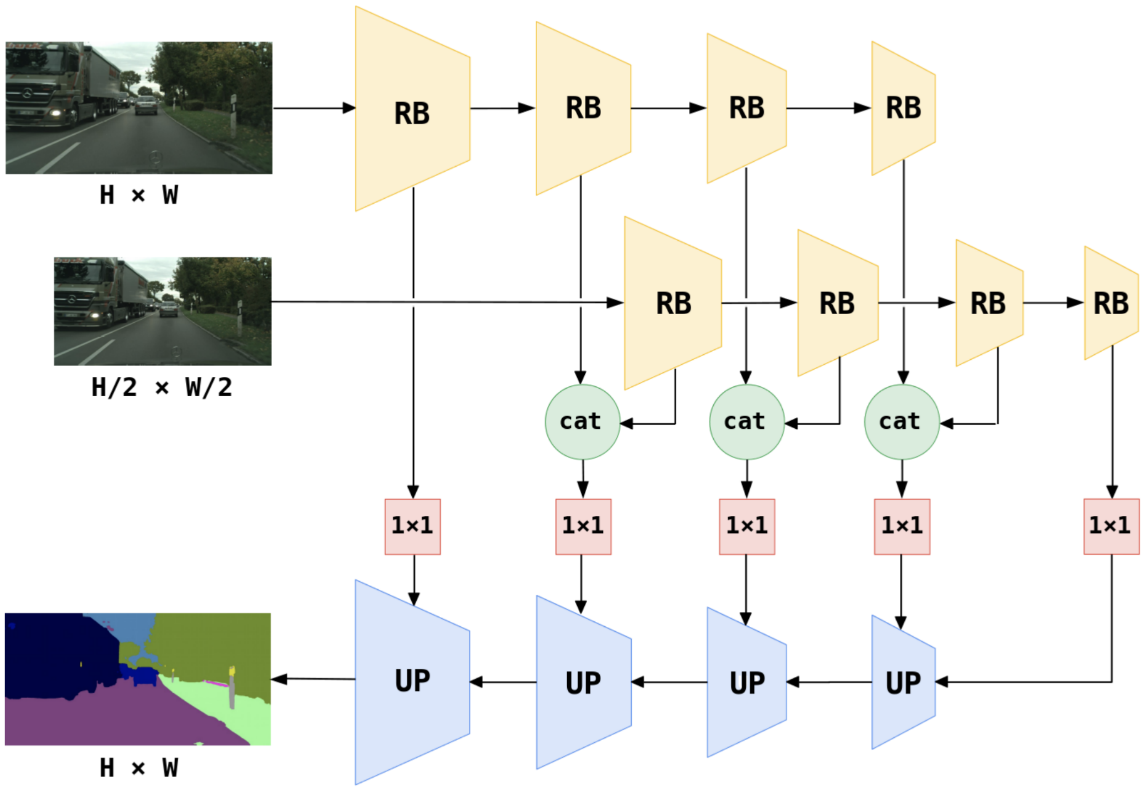


Figure 3.3: SwiftNetRN-18 interleaved pyramidal fusion architecture [20]

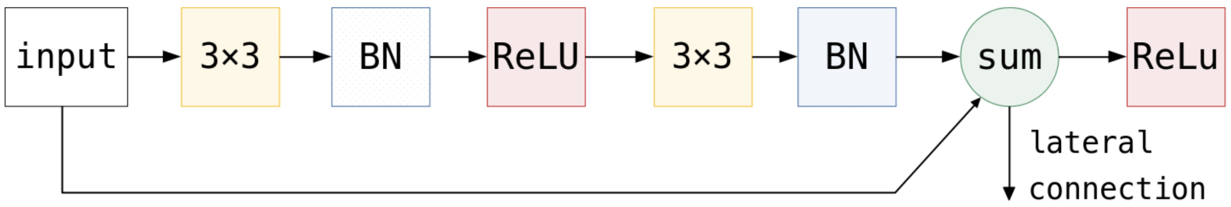


Figure 3.4: SwiftNetRN-18 final residual unit of each encoder block [20]

- Mean class IoU accuracy or Jaccard Index: $mIoU = \frac{1}{|C|} \sum\limits_{c \in C} \frac{TP_c}{TP_c + FP_c + FN_c}$
- Overall Pixel accuracy: $OP = \frac{TP}{TP+FP+FN}$
- Mean class Recall: $R = \frac{1}{|C|} \sum\limits_{c \in C} \frac{TP_c}{TP_c + FN_c}$
- Mean class Precision: $P = \frac{1}{|C|} \sum\limits_{c \in C} \frac{TP_c}{TP_c + FP_c}$

The reference metric on the Cityscapes dataset used in most modern semantic image segmentation challenges is the mean IoU (mIoU) as it is robust to class imbalance even though it does emphasize the accuracy of the class boundaries.

Objectives Because it is the main evaluation metric for modern semantic image segmentation models, the mean IoU is the first metric targeted by our experiments on SwiftNetRN-18. Indeed, if we achieve a large decrease in the value of the mIoU on adversarial samples it would imply a poor performance of the model on those inputs. In order to investigate whether such a behaviour can easily be generalised to all images, we attempt to find an image-generic pattern that can be applied by simple pixel-wise addition to any benign input to generate an adversarial sample.

This is a minimisation problem and requires to find the optimal noise parameters $\hat{\theta}$ within L_∞ norm ϵ that minimise the mIoU on the entire target dataset where every image x is summed with the noise pattern $G(\theta)$:

$$\hat{\theta} = arg \min_{\theta} \frac{1}{|C|} \sum_{c \in C} \frac{TP_c}{TP_c + FP_c + FN_c}$$
$$\text{such that } ||G(\theta)|| \leq \epsilon$$

Given that such a minimisation is intractable, only an approximation of the solution can be obtained via different computationally techniques. In this project, we experiment with two minimisation methods, which are random search and Bayesian optimisation and we compare the performance of the SwiftNetRN-18 model on a dataset of legitimate samples x and the adversarial dataset of samples $x + G(\hat{\theta})$.

We will additionally introduce a series of untargeted image-generic attacks focusing with different objectives that also aim to approximate the optimal parameters $\hat{\theta}$ for their objective. These are defined as follows:

- Pixel accuracy: $\hat{\theta} = arg \min\limits_{\theta} \frac{TP}{TP+FP+FN}$ such that $||G(\theta)|| \leq \epsilon$
- Mean class precision: $\hat{\theta} = arg \min\limits_{\theta} \frac{1}{|C|} \sum\limits_{c \in C} \frac{TP_c}{TP_c + FP_c}$ such that $||G(\theta)|| \leq \epsilon$
- Mean class recall: $\hat{\theta} = arg \min\limits_{\theta} \frac{1}{|C|} \sum\limits_{c \in C} \frac{TP_c}{TP_c + FN_c}$ such that $||G(\theta)|| \leq \epsilon$

3.3 Optimisation framework

In a black-box setting with limited adversarial capabilities and knowledge, the key to a cost-efficient design of powerful adversarial patterns is the optimisation algorithm.

3.3.1 Bayesian Optimisation

In a similar fashion to the work done by Co et al. [16] we use Bayesian optimisation, which is a sequential optimisation algorithm for black-box functions. It relies on a Gaussian Process $\mathcal{GP}(m, k)$ which consists of a prior mean function $m : \mathcal{X} \rightarrow \mathbb{R}$ and a positive definite kernel $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. In the following we use the Matérn 5/2 kernel, which is rather general and well suited for black-box optimisation: $k_{5/2}(x, x') = (1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2})e^{-\frac{\sqrt{5}r}{l}}$.

We use the expected improvement (EI) as our acquisition function. Given the mean $\mu(x)$ and standard deviation $\sigma(x)$ of $g(x)$, and $\gamma(x) = \frac{g(x_{best})-\mu(x)}{\sigma(x)}$ the EI function is defined as $\alpha_{EI}(x) = \sigma(x)(\gamma(x)\phi(\gamma(x)) + \mathcal{N}(\gamma(x)|0, 1))$.

All procedural noise attacks essentially consist in adding a noise pattern $G(\theta)$ to a genuine image x. Therefore, the first step to setting up Bayesian optimisation is to define the bounds of the search space for the Gaussian process.

For Perlin noise we define the parameters subject to the optimisation as $(\nu, \omega, \nu_{sine}) = \theta$ such that: $\nu \in [\frac{1}{160}, \frac{1}{20}]$, $\omega \in \{1, 2, 3, 4\}$ and $\nu_{sine} \in [4, 32]$ where ν is the frequency, ω is the number of octaves and ν_{sine} is the sine wavelength.

In the case of Gabor noise the Bayesian optimisation parameters are $(\sigma, \omega, \lambda, \xi) = \theta$ subject to: $\sigma \in [0, 1]$, $\sigma \in [0, \pi]$, $\lambda \in [0, 1]$ and $\xi \in \{1, 2, ...8\}$ where σ is the width of the underlying Gaussian, ω the orientation, λ the bandwidth and ξ the isotropy.
Although by default Gabor noise is anisotropic, we modulate the isotropy of the noise pattern by using varying numbers of Gabor kernels with different orientations to increase the isotropy. The number of these kernels is the isotropy parameter and its effect can be seen in figure 3.5.

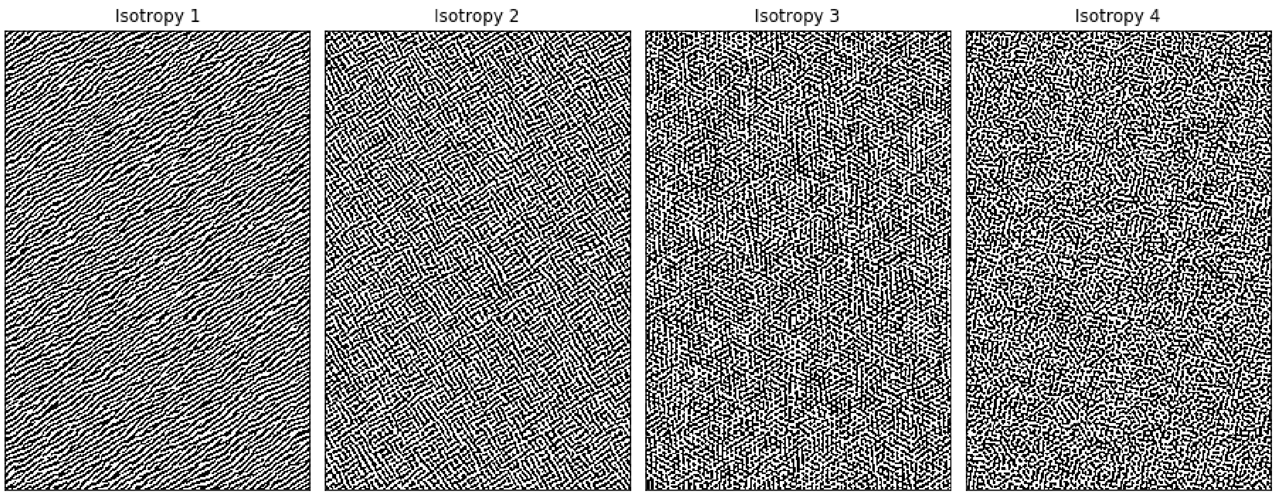


Figure 3.5: Gabor noise patterns with varying isotropy

Chapter 4

Experiments

4.1 Object detection

On the task of object detection, our target model is YOLOv3, which as of January 2019 was the most accurate real-time object detection system with a mAP on the COCO dataset of 57.9% at 0.5 IoU and an inference time of 50ms per frame. Its first class performance and speed among open source models has made the YOLO architecture a standard in object detection, which makes it the ideal candidate to evaluate the quality of a set of evasion attacks. Besides, the YOLOv3 paper [75] details its performance on the MS COCO dataset, which is our choice for training and evaluation of the model. Indeed, with more than 200000 labeled images and 80 object classes, it is among the most abundant and popular for this task. In this section, we detail our experiments, results and interpretations on both untargeted and targeted attack against YOLOv3 on the COCO dataset as well as potential defences to improve the model's robustness against those.

4.1.1 Untargeted attacks

Before any experiments, we generate random Perlin noise patterns with varying number of octaves on a few images of the COCO dataset to observe whether procedural noise attacks may have a significant impact on the performance of YOLOv3. An example is shown in figure 4.1 where the bounding boxes indicate the object detected, its location and the confidence of the model about the detection.

This simple experiment proves that Perlin noise can generate adversarial patterns causing false positives (couch in the second image from the left) as well as false negatives (dining table, potted plant, wine glass and chair in the third image). Besides, different patterns yield different detections with various degrees of accuracy. This tends to indicate that adversarial samples using procedural noise may also be optimised to fool object detection models, and in particular YOLOv3.

In our first experiments, we implement and evaluate the impact of untargeted attacks in a wide range of settings in an effort to make a comparative study of black-box procedural noise attacks. Given that untargeted procedural noise attacks were shown [16] to be much more effective than

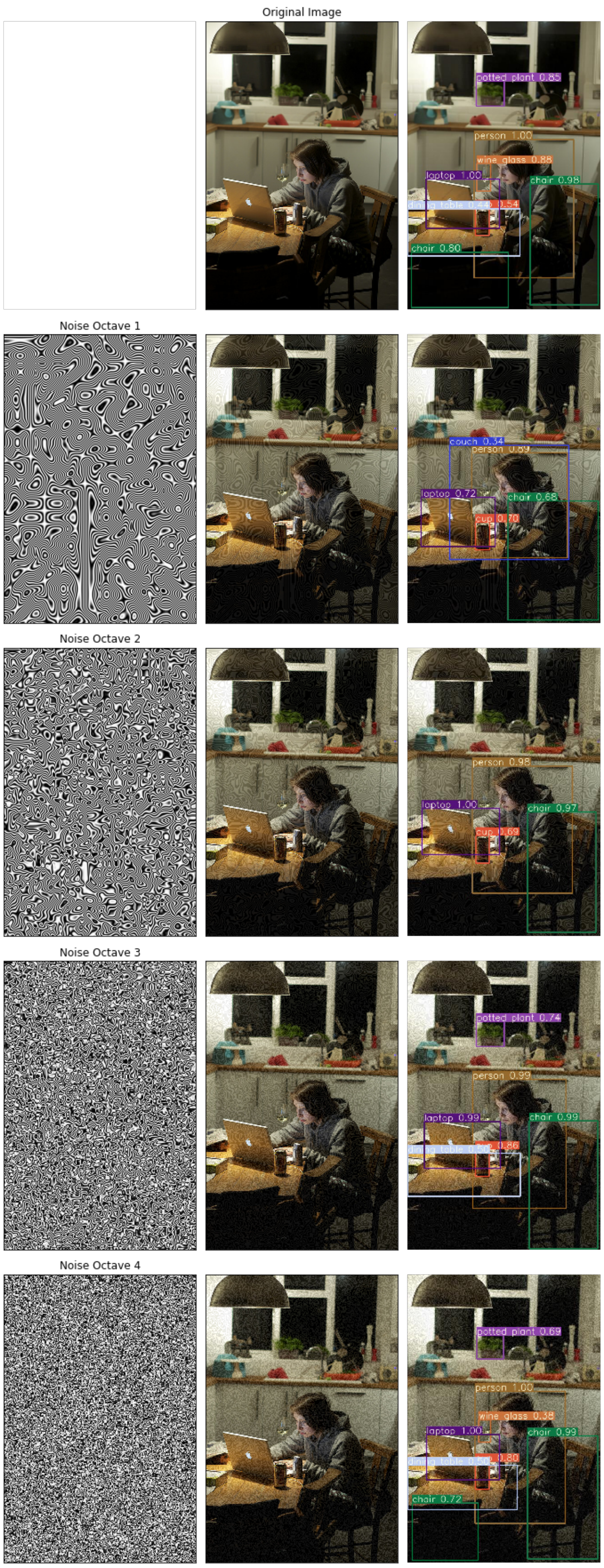


Figure 4.1: YOLOv3 detections on a COCO image perturbed with random Perlin noise with varying numbers of octaves and L_∞ norm 16.

targeted ones on various image classification models, we hypothesize that similar results will arise on object detection. This is because the structure of procedural noise patterns appears not to be flexible enough to capture class specific adversarial patterns. Besides, the Gaussian processes behind Bayesian optimisation does not seem to have sufficient learning capabilities to fool a specific class with just a few queries of the target model.

4.1.1.1 Image-specific average IoU attack

Firstly, we want to perform a number of input-specific procedural noise attacks on YOLOv3. This requires setting up the Bayesian optimisation framework described in the design chapter 3 and choosing an appropriate objective function for the attack. In general, this is the function of the metric to minimise in terms of the parameters to optimise, in this case the procedural noise parameters. The main metric used to assess object detection, the mean average precision (mAP) is defined as the mean over all classes of the AP for each class. The AP per class itself is computed the whole test dataset and as such is not suited for an input-specific attack. Therefore, in the case of input-specific attacks on object detection, the objective function needs to rely on a different metric that should ideally be positively correlated to the mAP in order to minimise it on the aggregate of all the target images. Thus, we conducted a series of small scale experiments with different candidate metrics for the objective function in order to find the most effective one to use throughout our experiments. The best fit appeared to be the average IoU with a threshold of 0.5 or the sum of intersection of unions above 0.5 of the original detection and the attacked detection averaged over the number of objects in the original detection as shown in algorithm 2.

Algorithm 2 Average IoU computation

```
1: function GETAVGIOU($originalDetection, perturbedDetection$)
2:     $iousSum \leftarrow 0$
3:     for $originalObj$ in $originalDetection$ do
4:         for $perturbedObj$ in $perturbedDetection$ do
5:             $iou \leftarrow getIOU(originalObj, perturbedObj)$
6:             if $iou >= 0.5$ then
7:                 $iousSum \leftarrow iousSum + iou.$
8:     return $iousSum/len(originalDetection)$
```

Evaluation of input-specific Perlin-R and Perlin-BO on YOLOv3 In our first series of attacks, a different noise pattern is applied to every single image in the COCO val2014 5k dataset, which consists of 5000 images. For both Perlin-R and Perlin-BO attacks, the objective function is the average intersection over union as defined in the above algorithm 2. In Perlin-R, 25 random Perlin noise patterns are applied to each image and the resulting image with the lowest average IoU when passed to YOLOv3 is chosen as the best perturbation. For Perlin-BO on the other hand, the perturbation corresponding to each image is optimised with Bayesian optimisation using 25 queries. This uses Gaussian processes to model the function that takes the noise parameters and returns the average intersection over union and try to find its minimum. The success of the attack is then measured over the entire dataset by computing the mAP at different stages of the random and Bayesian optimisations. This experiment is repeated for four different values of the norm of perturbations, namely 2, 6, 12 and 16. In the following we display the results of the Perlin-BO for different norm values on an image sampled from the target dataset (4.2). The impact on the mean average precision of the different attack settings is also reported in figure 4.3.

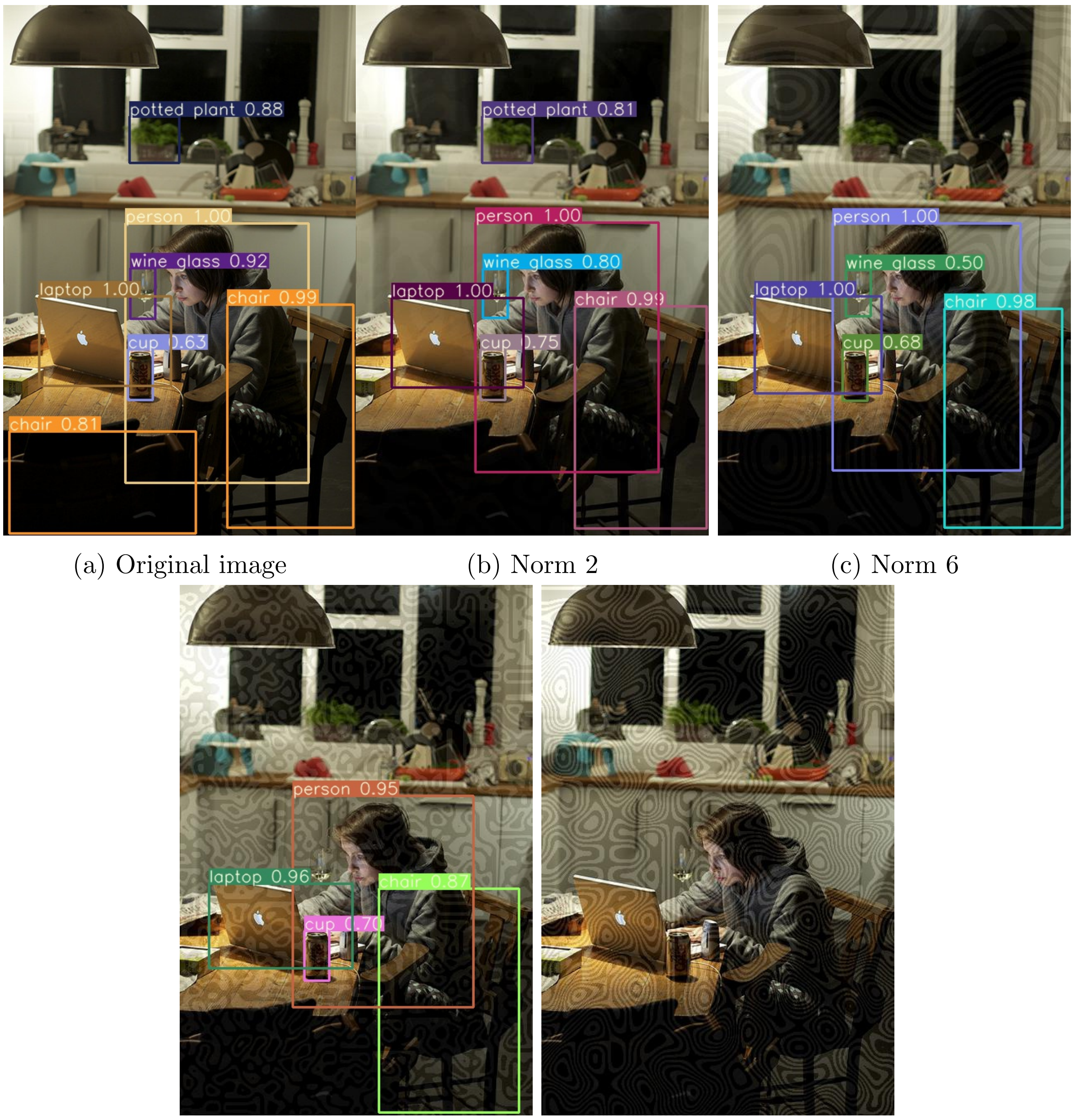
These results very clearly show that given a large enough norm for the perturbations, Perlin-BO performs better than Perlin-R on YOLOv3, which highlights that an appropriate optimisation algorithm contributes to making procedural noise attacks more effective. However, given a small norm such as 2, the Bayesian optimisation does not appear any better than random search. This may be explained by the limitations on the values of the objective functions implied by such a small norm.

This graph also highlights that both these optimisation methods converge very quickly. Indeed, after just 5 queries for Perlin-BO and 9 for Perlin-R the improvement following each query seems to largely decrease. Besides, even without any optimisation and queries, a Perlin noise attack with norm 16 divides the mAP by 2 (0.425 as opposed to 0.585) whereas further experiments showed that salt and pepper noise only causes to decrease to 0.545. This demonstrates that Perlin noise has natural properties making it a function of choice for evasion attacks but does not offer full flexibility.

In order to provide a more comprehensive understanding of the role played by Perlin noise in the context of object detection, we assess whether it is able to hide or mimic some specific objects. By taking the original COCO val 2014 5k dataset and its adversarial variant built with our Perlin-BO image-specific attack, we measure the ratio of the number of detections for each class on the adversarial versus the original dataset.
As a result, we observe that every single class apart from two yield fewer detections on the adversarial set than on the original one. This insight shows that the effectiveness of Perlin is due to its ability to decrease the confidence of object detections or the probability of an objects being present at specific locations. In some particular cases however, namely the bed and zebra classes, we observe a ratio of 2.33 and 1.23. Such outstanding results indicate that Perlin noise also has mimetic properties with respect to these classes from a deep convolutional network's point of view. We believe that this is due to the resemblance Perlin patterns may bare with bed sheets folds and zebra stripes. Moreover, we investigate the relationship between this ratio and the number of training samples for each class by plotting the corresponding points for each class in figure 4.4.

There does not appear to be any clear pattern in this relationship, an observation confirmed by the correlation coefficient which is only 3.2%. Indeed, the decrease in detections in adversarial conditions is impacting most classes, regardless of their frequency in the COCO training dataset. In particular, the person class only offers a ratio of 0.39 while the average ratio over all classes is 0.22 although it is by far the most common in the training dataset with over 187000 occurrences compared to an average of 7608 per class. This suggests that a large increase in the training dataset size may only provide a small contribution in terms of robustness of the model to Perlin noise perturbations.

Perlin noise extension: coloured patterns As an extension, we attempted to develop better adversarial patterns based on Perlin noise by modifying its set of parameters. In particular, we used three different sine frequencies, one for each colour channel (red, blue and green) instead of a single one broadcast across all three channels. In order to do so, we introduce extend the basic Perlin noise, using a separate sine frequency parameter for each colour channel.



(a) Original image (b) Norm 2 (c) Norm 6

(d) Norm 12 (e) Norm 16

Figure 4.2: Comparison of object detection for different values of the L_∞ norm under the average IoU attack

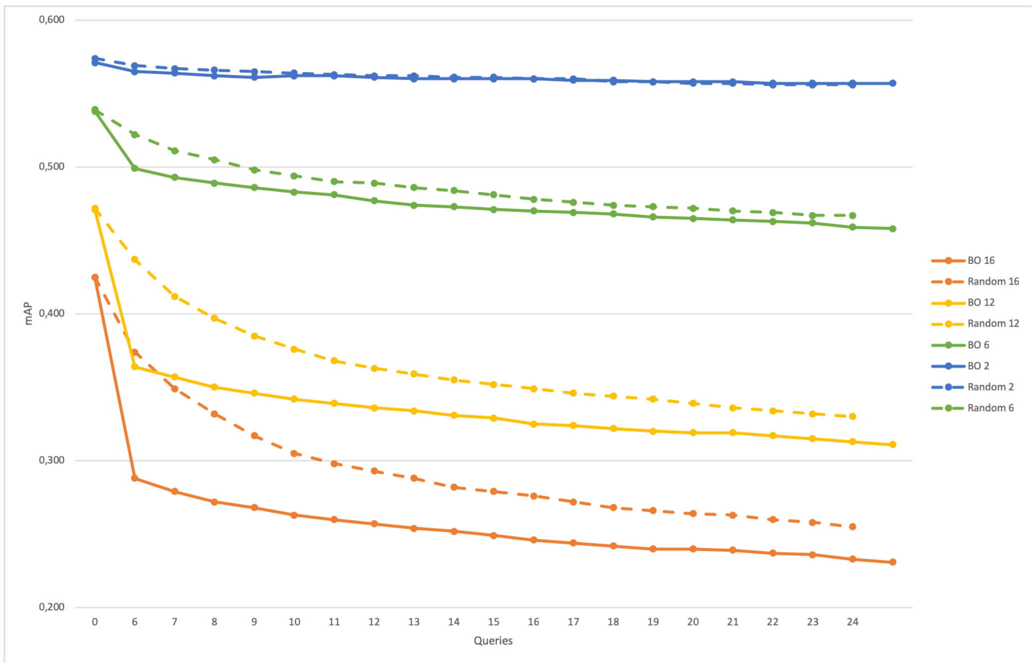


Figure 4.3: mAP results for Perlin-R and Perlin-BO for norm 2, 6, 12 and 16

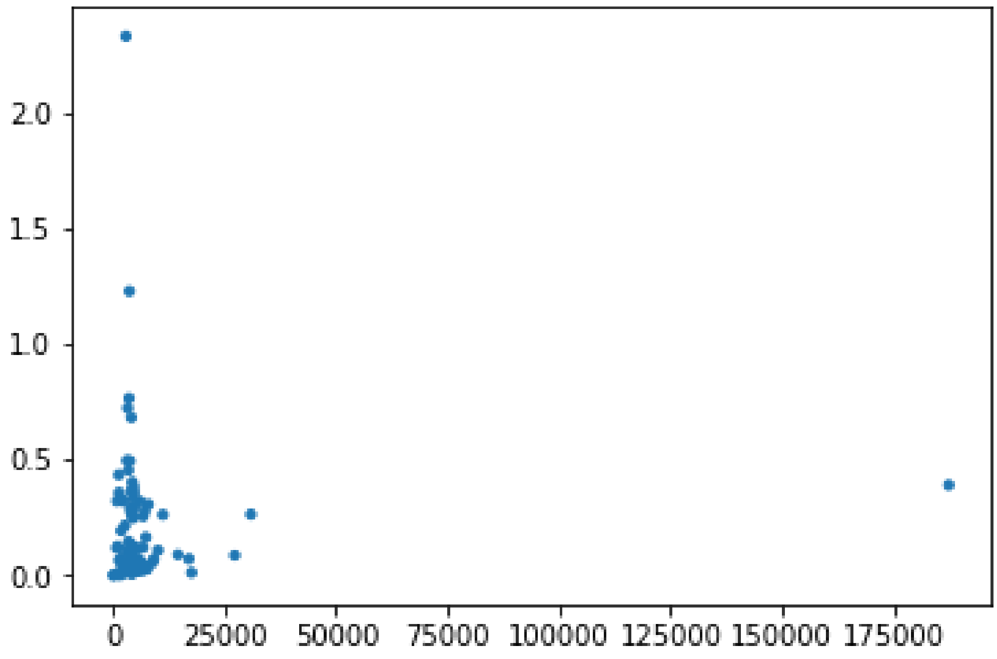
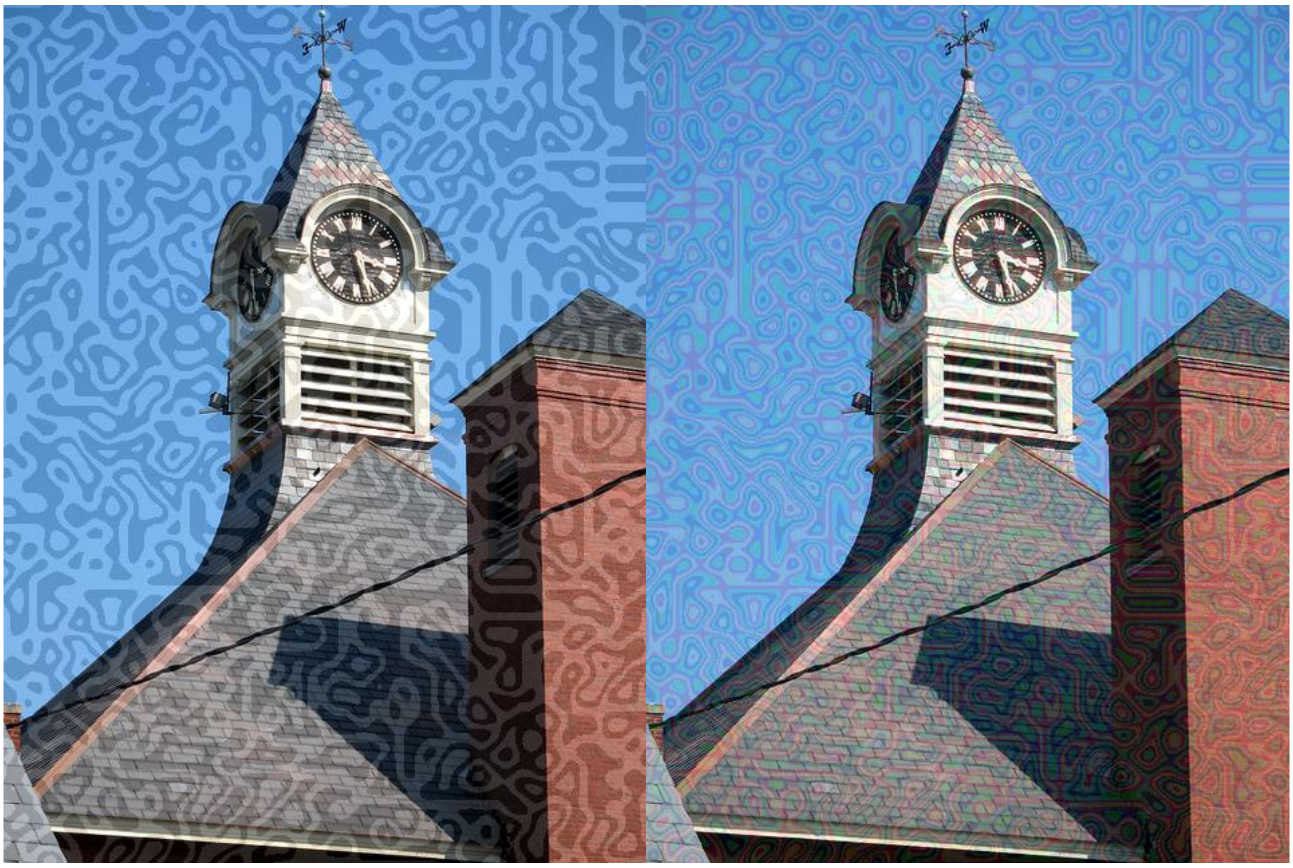


Figure 4.4: Ratio of the number of detection per class on the COCO val 2014 5k dataset following a Perlin-BO attack versus the original dataset with respect to the number of training samples



(a) Grayscale (b) Coloured

Figure 4.5: Grayscale and coloured Perlin noise obtained with the average IoU attack

Instead of having just one common grayscale sine frequency, we introduce a red, a blue and a green sine frequency which are optimised via Bayesian optimisation along with the number of octaves and the frequency. This has the effect of producing rainbow-like coloured patterns instead of the previously used grayscale pattern. The difference can be seen in figure 4.5.

Note that on these images not only do the sine frequency parameters differ between the two images but also the sine and number of octaves. Indeed, the two result from the same attack but given the difference in the parameters, the Bayesian optimisation differed in the two cases.

In table 4.1, we report the results of the Perlin-BO experiments with norm 16 on the COCO val 2014 5k dataset for both grayscale and coloured noise, as well as salt and pepper noise (no optimisation) for reference:

This table shows a significantly higher mAP, precision and recall for the coloured Perlin-BO attack compared to its grayscale counterpart, meaning it is not as effective in the general case. This can be explained by the larger number of parameters (5 versus 3) which makes Bayesian optimisation slower to converge. Indeed, the number of parameters of the noise function used in the optimisation algorithm corresponds to the number of optimal values that need to be learned in the process. Besides, although both noise patterns carry the same L_∞ norm, a single pixel will have the same noise applied consistently across all three colour channels in the grayscale attack, which induces a larger change in brightness and may in turn affect to a greater extent the low-level features learnt by the model.

Table 4.1: Grayscale and coloured Perlin-BO comparison (norm 16)

Attack	mAP	Precision	Recall
Original	0.585	0.628	0.594
S & P noise	0.545	0.601	0.552
Grayscale	0.231	0.283	0.233
Coloured	0.334	0.398	0.337

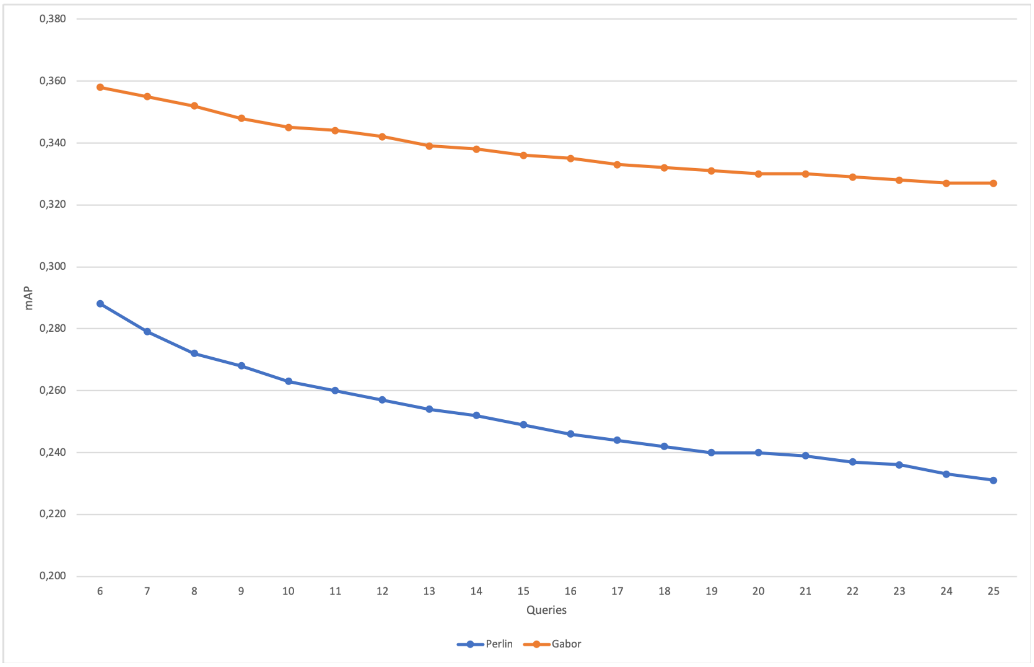
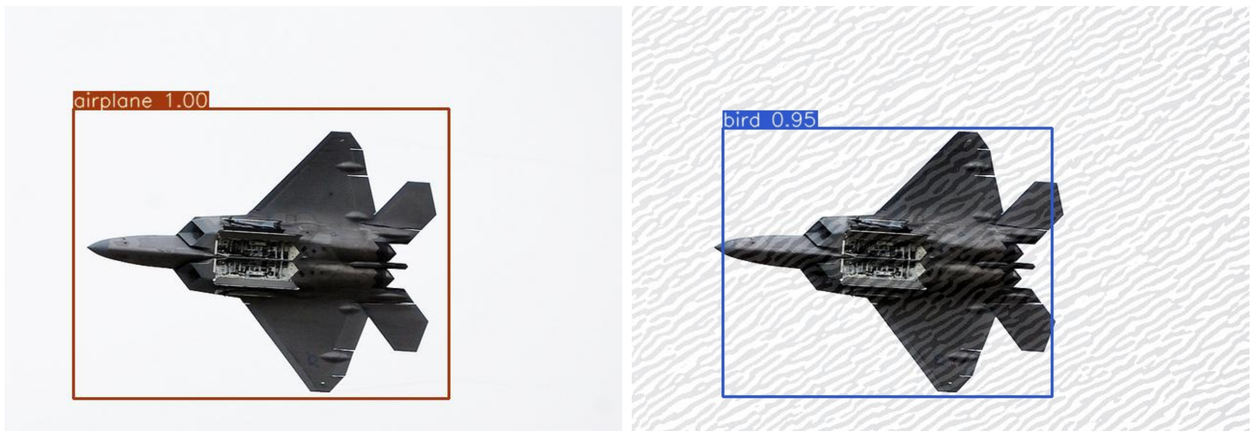


Figure 4.6: mAP for Perlin-BO and Gabor-BO with norm 16

However, on some classes, the coloured Perlin-BO attack performed better than the grayscale one. For instance it decreased the person class AP from 0.7412 to 0.4025 versus 0.4688 following the grayscale attack. This result appears to be due to the greater flexibility offered by the increased number of parameters offered by the coloured noise.

We conclude that for query-efficient untargeted attacks grayscale Perlin noise is more suitable than coloured Perlin noise and that it should be used as the reference implementation in the following experiments.

Comparison between procedural noise patterns In an effort to compare different procedural noise functions to assess their relative effectiveness as adversarial pattern generators, another experiment was realised in the same conditions using Gabor as well as Perlin noise. Gabor is a sparse convolution noise function which has already been tested in the context of classification evasion attacks by Co et al. [16] and shown less effective than Perlin noise on that specific task. This experiment aims to evaluate whether or not this conclusion remains valid across different computer vision tasks. The results of these attacks expressed as the mAP in terms of the number of queries are displayed on graph 4.6.



(a) Original image detection (b) Gabor noisy image detection

Figure 4.7: Example of the effect of Gabor noise perturbations on YOLOv3

Considering that the mAP resulting from Gabor noise attack is consistently higher than the one induced by Perlin-BO, it appears that Perlin noise is a better vector for untargeted image-specific procedural noise attacks against object detection. This appears to have two main reasons. Firstly, after 5 random queries and just one query of Bayesian optimisation, Perlin noise is already more effective than Gabor noise with a mAP 0.29 as opposed to 0.36. This suggests that Perlin noise patterns, even random, are in general more suitable as adversarial patterns than Gabor noise patterns. Besides, we observe that given a larger number of queries Perlin sees a larger improvement with decrease of 6 points on the mAP from the 6th to the 25th query while the Gabor noise attack only sees a decrease of 3.5 points in the same interval. This implies that Perlin noise also offers more flexibility for Bayesian optimisation to improve with respect to a random choice of initial perturbations.

Given these results, it appears that Perlin noise Bayesian optimisation attacks are more devastating than Gabor noise on unprotected models in an untargeted setting. This raises the question of whether Gabor noise may be more suitable in a different context and highlights that Perlin noise should serve as a reference for untargeted procedural noise attacks. Nevertheless, Gabor noise still appears to have a substantial negative impact on the target model. A visual example of this is given in figure 4.7.

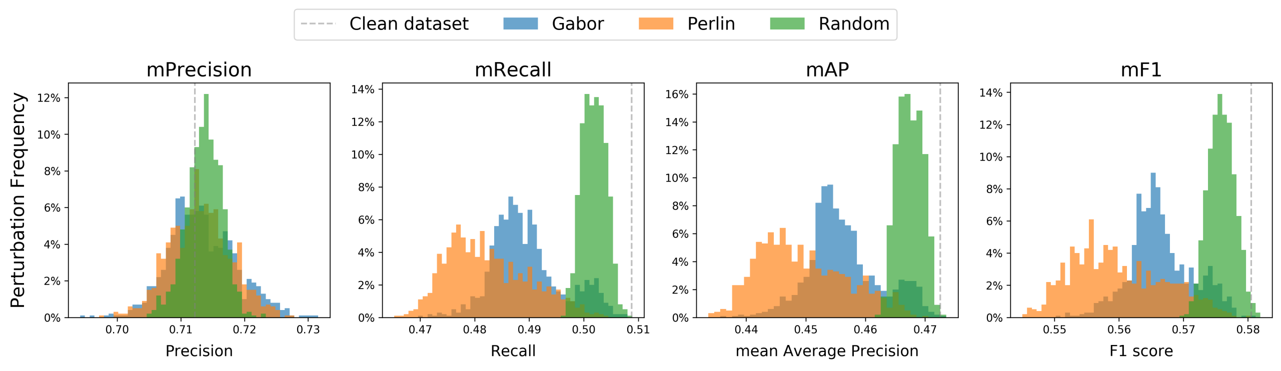
4.1.1.2 Image-agnostic mAP attacks

In some instances, in particular in the context of real-time object detection, image-specific attacks may not constitute a threat as it typically takes several seconds to generate. In this scenario, it is particularly interesting to consider image-agnostic adversarial patterns. In the following, we conduct experiments to measure the impact of procedural noise and random noise as image-generic adversarial perturbations on YOLOv3 for different norm values (2, 6, 12 and 16). First, we randomly generate 1000 different Perlin, 1000 Gabor and 1000 random noise patterns. Then we apply each of those patterns to 1000 images from the COCO dataset and evaluate the performance of YOLOv3 on the resulting noisy images using the mean class precision, mean class recall, mAP and mean F1 score. This will allow us to compare the adversarial properties of different noise functions and study their effectiveness with respect to their maximal norm. The results are displayed in the form of histograms and reported in figure 4.8.

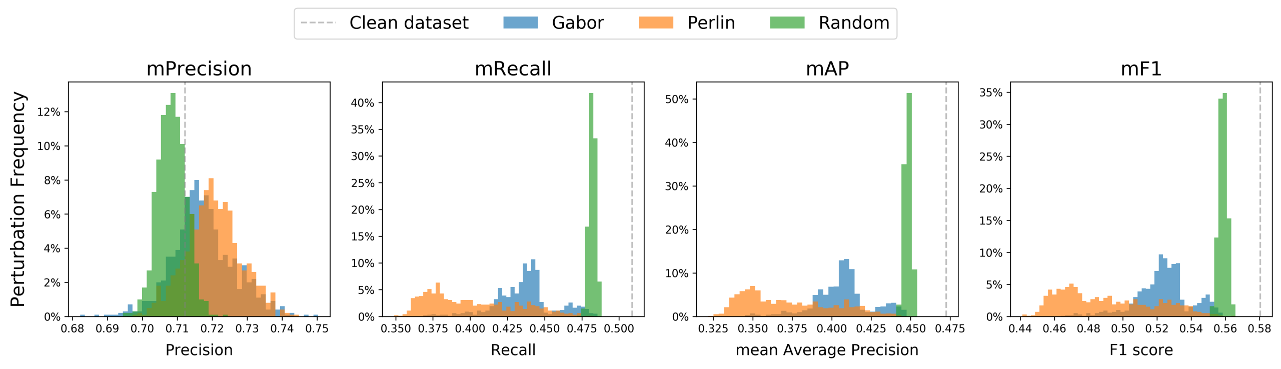
These histograms provide a number of insights regarding the role the different noise patterns play in the context of object detection. Firstly, although every single noise pattern leads to a decrease in mean recall, in some instances the addition of noise to every image yields an increase in mean precision compared to that on the clean dataset. This phenomenon can be explained by a comparatively smaller number of predictions or positives on those noisy samples. Indeed this effect will lower the number of true positives and false positives, hence decreasing the recall and increasing the precision respectively. Such a behaviour can be expected by YOLOv3 as in our settings the model only outputs predictions for which it has at least a 30% confidence. Thus, any perturbation causing the confidence of a prediction to drop below 30% will result in a negative instead of a positive. This effect could partly be addressed by lowering the detection confidence threshold of the model but this would result in a precision drop on legitimate samples. In other cases however, we also measure a drop in the precision of the model on a noisy dataset. This proves that noisy samples are not only able to decrease the overall confidence of the model but also to generate false positives, an effect that can not be countered by lowering the confidence threshold.
Besides, every single noise perturbation negatively affects the mAP and the mean F1 score (mF1) across all norms (except for some statistically insignificant random noise patterns for a norm of 2). This result suggest that every pattern generated with the noise functions considered can deteriorate the overall performance of the model. Nevertheless, there appears to be a great disparity between the adversarial properties of the different noise functions. In particular, Perlin noise appears to be the most flexible of the three with a wide spread in terms of resulting performance as well as the one with the most effective patterns on the mAP and mF1, followed by Gabor and finally random noise. This result concurs with the results observed on image-specific attacks and illustrates why Perlin noise may be more suited for Bayesian optimisation attacks than its rivals.

In order to understand more precisely the impact of each noise parameter on the target model in different norm settings, we generate correlation matrices based on the results of applying 1000 random Perlin and 1000 Gabor noise patterns, each to a set of 1000 COCO val 2014 images in an image-generic fashion. The Perlin noise correlation matrices appear in figure 4.9 while the Gabor noise results are displayed in figure 4.10.

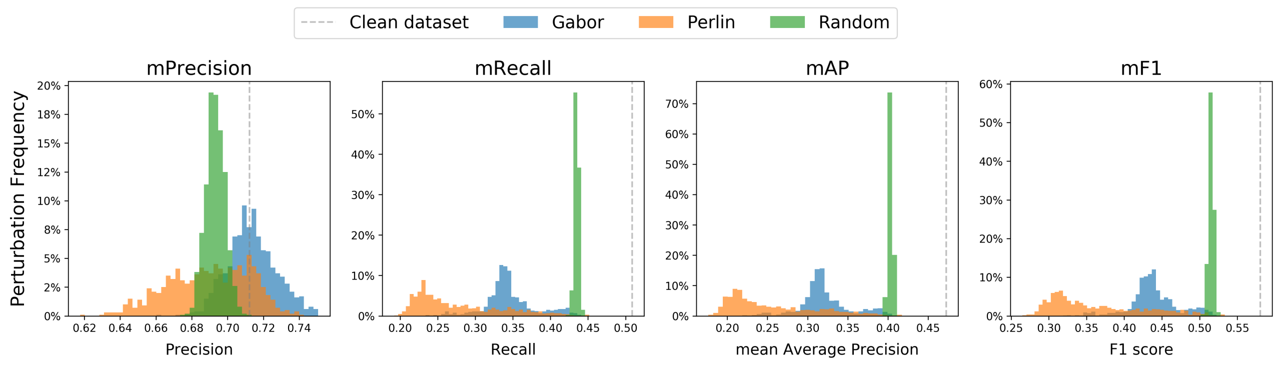
Both figures indicate a tight correlation between the mean class recall, the mean average precision and the mean F1 score while the mean precision appears less correlated with those three metrics, in particular if the norm of the perturbations is small. Overall, we observe smaller variations in the mean precision, which makes its variations have a smaller impact on the mean average precision and the F1 score. This first result is in line with our previous observations that the mean precision does not consistently decrease in the presence of adversarial noise patterns while the mean recall, mAP and mean F1 score do and can be explained by the appearance of false positives in the dataset induced by the perturbations. This appears to be particularly true in the case of significant Gabor noise perturbations with a norm of 16 which indicates that Gabor noise is more appropriate than Perlin noise to generate false positives. We hypothesize that the similarity between Gabor noise and naturally occurring patterns such as those seen on zebras may be responsible for this property of Gabor perturbations.



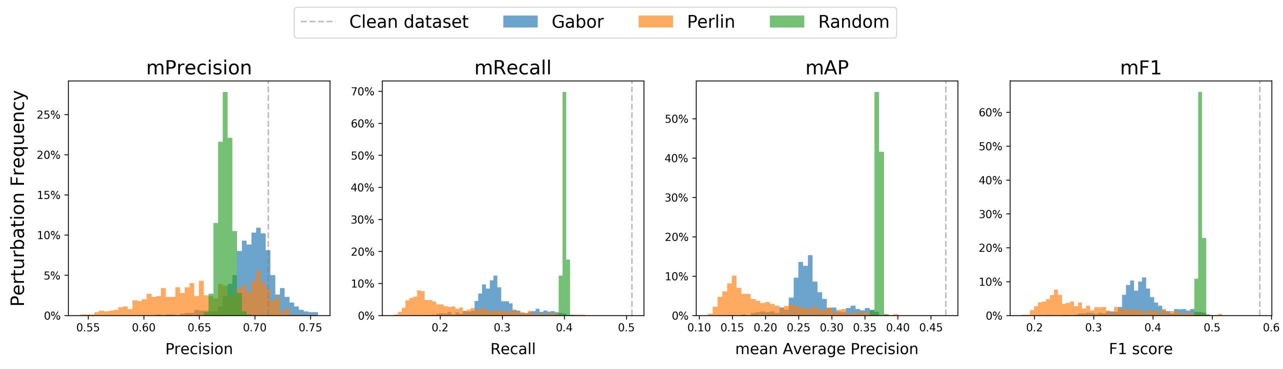
(a) Norm 2



(b) Norm 6

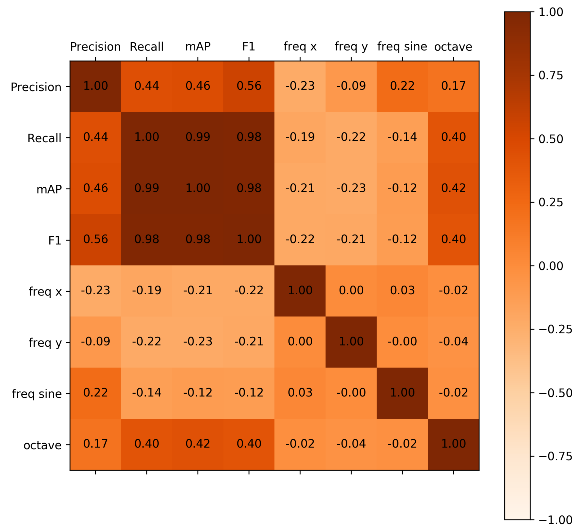


(c) Norm 12

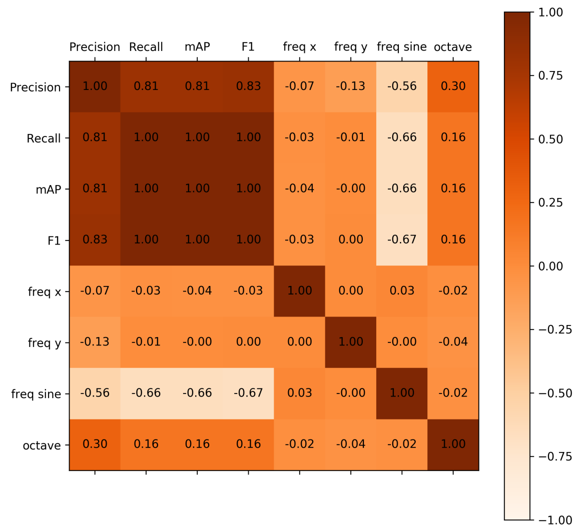


(d) Norm 16

Figure 4.8: YOLOv3 performance on 1000 noisy COCO images for norm 2, 6, 12 and 16

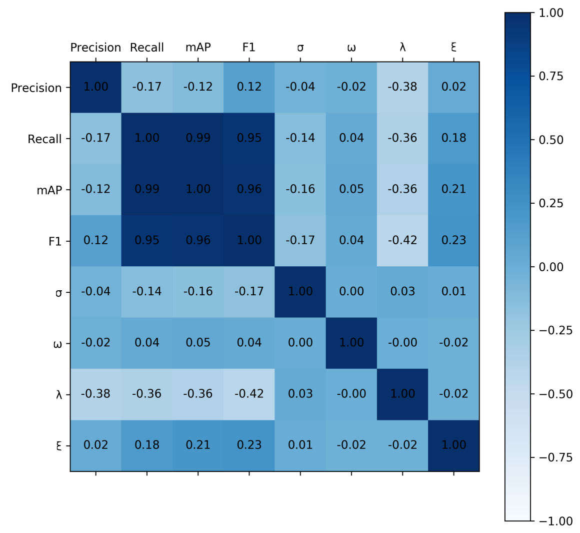


(a) Norm 2

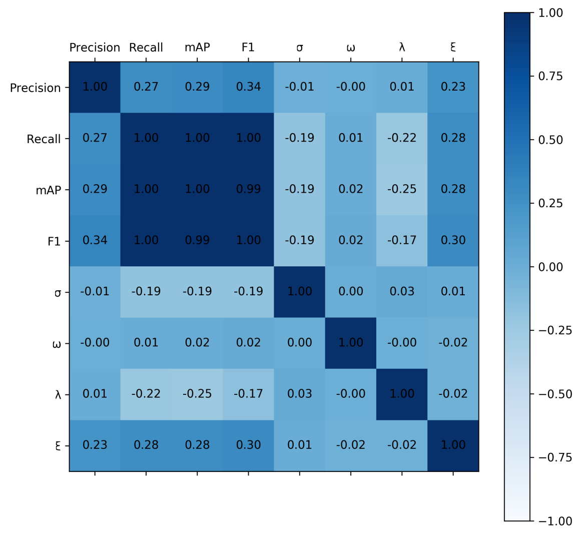


(b) Norm 16

Figure 4.9: Perlin noise correlation matrices



(a) Norm 2



(b) Norm 16

Figure 4.10: Gabor noise correlation matrices

Relationship between noise parameters and model performance Looking at the correlation matrices, we can establish a relationship between the procedural noise parameters and the model evaluation metrics. Such a relationship can indicate to which extent each parameter contributes to the performance drop of the model on adversarial samples. This could in turn allow us to reduce the number of parameters involved in Bayesian optimisation attacks to only optimise those that play a major role in the attack, thereby further improving the query-efficiency and effectiveness of these attacks.

In the case of Perlin noise, we used four parameters for this experiment: horizontal frequency (freq x), vertical frequency (freq y), sine frequency (freq sine) and number of octaves (octave). Considering norm 2 perturbations, all four parameters seem to have a significant impact on the performance of YOLOv3, with a correlation going from a minimum absolute correlation with the mAP of 0.12 for the sine frequency up to 0.42 for the number of octaves. Therefore, in the case where the norm of the Perlin perturbations is strictly limited, all parameters should remain considered in the optimisation process.
In a more flexible scenario where the L_∞ norm of the noise can reach up to 16 however, we observe that the only the sine frequency (0.66) and the number of octaves (0.16) have a significant correlation with the mAP value. Therefore, in such a setting, the horizontal and vertical frequencies could be arbitrarily set and ignored in the optimisation process to increase the convergence rate of the parameters. Nevertheless, their correlation with the precision is not completely insignificant (-0.07 and -0.13 respectively) and therefore those parameters should be optimised in an attack aiming at generating false positives.

In this experiment, the Gabor noise parameters are the width of the Gaussian σ, the orientation ω, the bandwidth λ an the isotropy ξ. Given a norm of 2, we observe that all four parameters are correlated to the mAP although the correlation between the orientation and the mAP is rather small (0.05). Besides, we observe that the only the bandwidth enjoys a relatively strong correlation with the precision which makes it the most powerful parameter to optimise to generate false positives.
Finally under norm 16 perturbations, the correlation between the orientation and the mAP, recall and F1 score appears insignificant, suggesting the orientation could be omitted in the optimisation. Besides, we observe that only the isotropy clearly plays a role in terms of the generation of false positives with a 0.23 correlation with the precision.

For both Perlin and Gabor noise, we have shown that given a norm significant enough to offer flexibility to the procedural noise patterns, some parameters appear to be irrelevant in an image-generic adversarial attack. Discarding these parameters from the Bayesian optimisation offers the possibility to develop more effective attacks, both in terms of query-efficiency and impact on the target model.

4.1.2 Targeted attack

Following the claims by Co et al [16] that procedural noise attacks were not particularly suited for targeted attacks on image classification, we assess whether this conclusion matches our results and applies to object detection. Considering that the person class is the most common object in both the training set and the validation set with 187437 and 10888 occurrences respectively, it appeared that it would be of particular interest to perform a targeted input-specific attack and compare the results with the untargeted attack conducted in the same

Table 4.2: Untargeted and targeted attacks comparison

Attack \ Metric	Person AP	mAP	Precision	Recall
None	0.7412	0.585	0.628	0.594
Targeted	0.4453	0.338	0.404	0.341
Untargeted	0.4688	0.231	0.283	0.233

conditions.

We conducted this targeted attack with a norm $L_\infty = \frac{16}{256}$ using Perlin noise and the Bayesian optimisation settings described in chapter 3. The objective function used here is the average intersection over union for the person class only and the results are compared with those of the untargeted image-specific average IoU attack in 4.1.1.1 and displayed in the table below:

According to the results in table 4.2, the untargeted attack performs slightly better than the targeted Perlin-BO attack with a slightly lower AP for the person class (0.4453 as opposed to 0.4688). However, its person AP remains above the person AP of 0.4025 resulting from the coloured Perlin-BO attack. This indicates that although Perlin noise may be used for targeted attacks, it is not particularly well suited for that usage. The reason for this is that targeted attacks require perturbations that accurately capture the specificities of the targeted class in order to increase the number of false negatives by hiding its features or increase the number of false positives by replicating its features. Given the complexity of such a task, Bayesian optimisation does not have a sufficient learning power to fit it in just a few queries. Besides, Perlin noise does not appear to offer sufficient flexibility to exploit the specific vulnerabilities of every single class.

Besides, the lower mAP, precision and recall resulting from the untargeted attack show, as expected, that the untargeted attack is more effective in the general case. The recall and precision results in particular show that the perturbations obtained in the targeted scenario do not behave any differently from those obtained in the untargeted attack.

4.1.3 Defences

4.1.3.1 Total Variation Denoising

It has been shown by Guo et al. [76] that some input transformations such as total variation (TV) minimisation reduce the impact of a number of evasion attacks including FGSM and the L_2 attack by Carlini and Wagner on ImageNet. This indicates that TV denoising may also prove relevant as a countermeasure against procedural noise attacks on object detection models, which we investigate in the following.

Given an input image x, TV denoising aims to approximate it with a close image $\hat{y}$ with respect to the L_2 norm and such that its total variation norm is minimised. This minimisation problem is defined as:

Table 4.3: TV denoising effects on mAP, precision and recall

Model input \ Metric	mAP	Precision	Recall
Original	0.585	0.628	0.594
Original + TV Chambolle	0.542	0.596	0.549
Original + TV Bregman	0.557	0.608	0.565
Perlin-BO	0.231	0.283	0.233
Perlin-BO + TV Chambolle	0.408	0.467	0.412
Perlin-BO + TV Bregman	0.374	0.437	0.378

$$\hat{y} = arg \min_{y} [||x - y||_2 + \lambda V(y)]$$
$$\text{where: } V(y) = \sum_{i,j} \sqrt{|y_{i+1,j} - y_{i,j}|^2 + |y_{i,j+1} - y_{i,j}|^2}$$

We apply TV Chambolle denoising with weight 0.1 as well as TV Bregman denosing with weight 10 to both legitimate samples and Perlin-BO adversarial samples with norm 16 and report the results.

Our experiments simulate a transferability attack whereby the attacker generates adversarial samples with Perlin noise and Bayesian optimisation on an unprotected model to attack a model equipped with total variation denoising.

Indeed, the above results show that both TV Chambolle and TV Bregman denoising improve the performance of YOLOv3 on Perlin-BO adversarial samples in terms of mAP, precision and recall. In particular, the mAP reaches as high as 0.408 when using TV Chambolle and 0.374 with TV Bregman as opposed to 0.231 without any denoising. However, TV denoising also has a negative impact on the model on legitimate samples, with a mAP of 0.542 for Chambolle and 0.557 for Bregman, while the original mAP is 0.585 on the COCO 5k validation set. Although smaller than the positive impact on malicious samples, it may not be worth using TV denoising if only a small fraction of inputs are expected to be malicious procedural noise samples. Nevertheless, it shows that TV denoising may be used as an effective measure to address the vulnerability to black-box procedural noise attacks and that there exists a trade-off between the performance of the system comprised of a TV denoiser and an object detection model in a benign environment and its robustness to adversarial samples. Currently TV denoising also has a major impact on the speed performance of SwiftNetRN-18, which is a real-time model. Although some optimisations are possible to increase its computation time as shown in [77] it is most likely that this defence mechanism will for now remain only suitable for applications where real-time performance is not necessary.

4.2 Semantic image segmentation

Following the successful use of procedural noise as untargeted adversarial perturbations on real-time object detection, we decided to further evaluate its impact on real-time semantic

Table 4.4: SwiftNet IoU mean class accuracy attack results for general metrics (in percentage)

Norm	Attack	Mean class acc.	Mean class recall	Mean class precision	Pixel acc.
0	None	75.45	83.09	87.87	95.68
6	Perlin-BO	26.06	31.81	67.46	40.31
	Perlin-R	25.11	30.86	66.69	39.73
12	Perlin-BO	6.07	10.26	63.21	26.06
	Perlin-R	5.95	10.12	61.75	26.06
16	Perlin-BO	3.18	7.29	60.45	23.56
	Perlin-R	3.0	7.11	59.58	23.58

segmentation. On this task, our target model is the state-of-the-art SwiftNet described in more detail in 3.2.2, which we evaluate on the Cityscapes fine validation dataset.

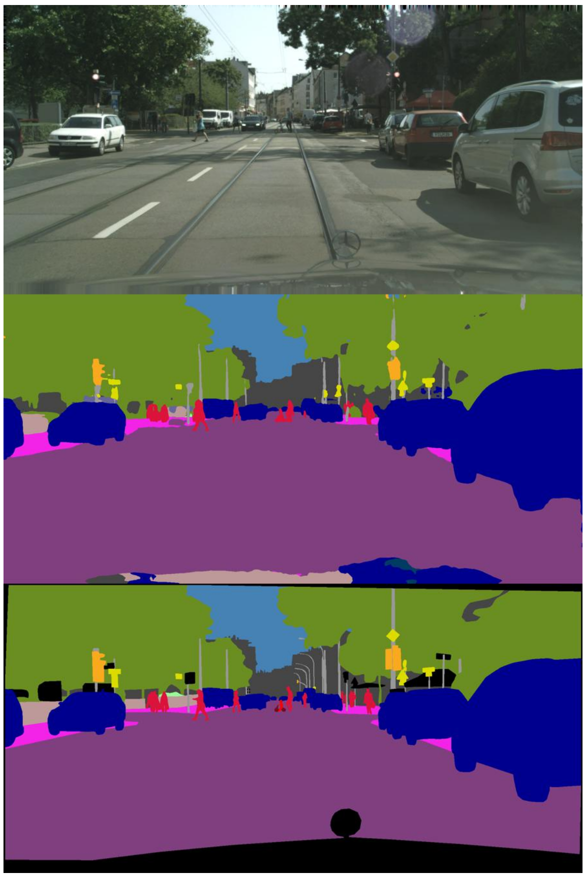
4.2.1 Untargeted attacks

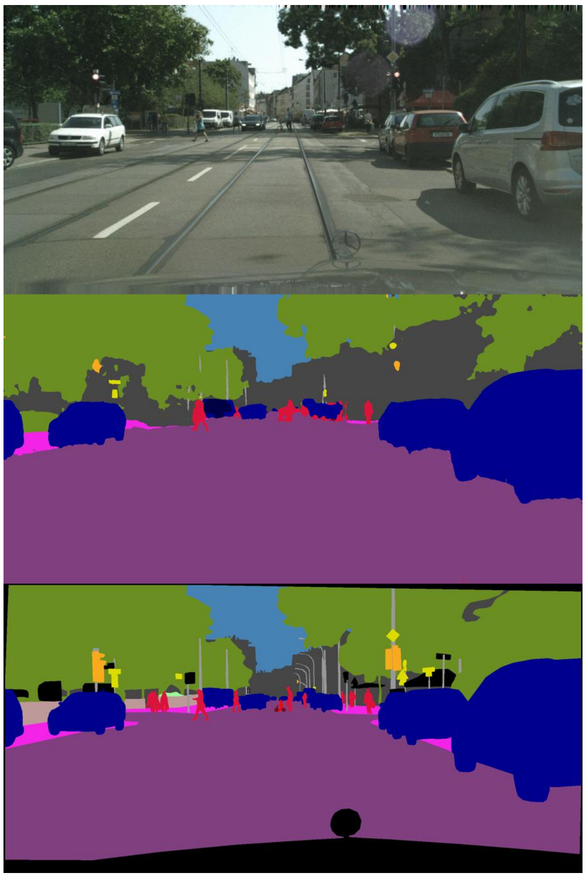
In this section we introduce and evaluate a series of procedural noise attacks against SwiftNet on the Cityscapes dataset in different settings using both Bayesian optimisation as well as random search to evaluate whether those methods are suited to find a universal adversarial pattern in a limited number of queries. These attacks are all image-generic as they consists in applying a single noise pattern to the entire image dataset causing the largest possible decrease in the target metric. This type of attack is suitable for a wider range of applications than image-specific attacks as the adversarial noise pattern can be precomputed and simply applied to any original image in a live system. In the scope of this project however, we introduce this attack with the purpose of evaluating the universality of Bayesian optimisation generated Perlin noise patterns.

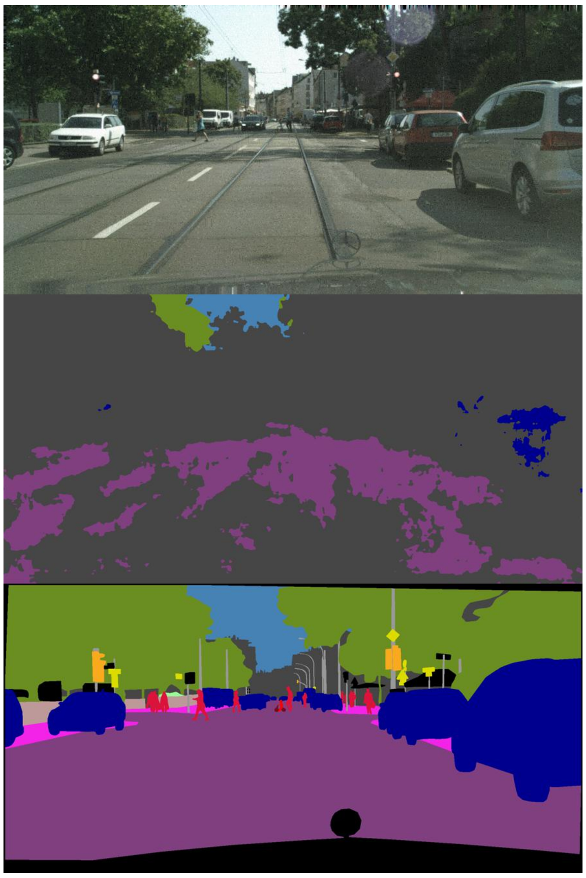
4.2.1.1 IoU mean class accuracy attack

The IoU mean class accuracy aims to evaluate the impact of procedural noise attacks on the IoU mean class accuracy metric with it taken as the objective function for the optimisation of the noise pattern. Below are the random and Bayesian optimisation attacks results on different metrics for norm values 6, 12 and 16 (4.4) and an example of segmentation of the same image in a legitimate setting and under attack with these different norms (4.11). The main class labels visible in these images are defined as follows: green: vegetation, light blue: sky, dark blue: car, red: person, orange: traffic light, yellow: traffic sign, pink: sidewalk, purple: road

Given the results in table 4.4, it appears that both Perlin-BO and Perlin-R attacks largely reduce the mean class accuracy, down from 75.45% on legitimate samples to 3.0% on Perlin-R adversarial samples with norm 16. The mean class recall is also severely impacted by the attacks and to a lesser extent the pixel accuracy, down from 95.68% to 23.56% on Perlin-BO samples with norm 16. With a decrease from 87.87% to 59.58% due to Perlin-R with a norm of 16, the mean class precision is the least affected metric. The explanation for this behaviour can be visualised in figure 4.11: strong Perlin noise perturbations (norm 12 and 16) reveal a significant


(a) Original


(b) Norm 6


(c) Norm 12

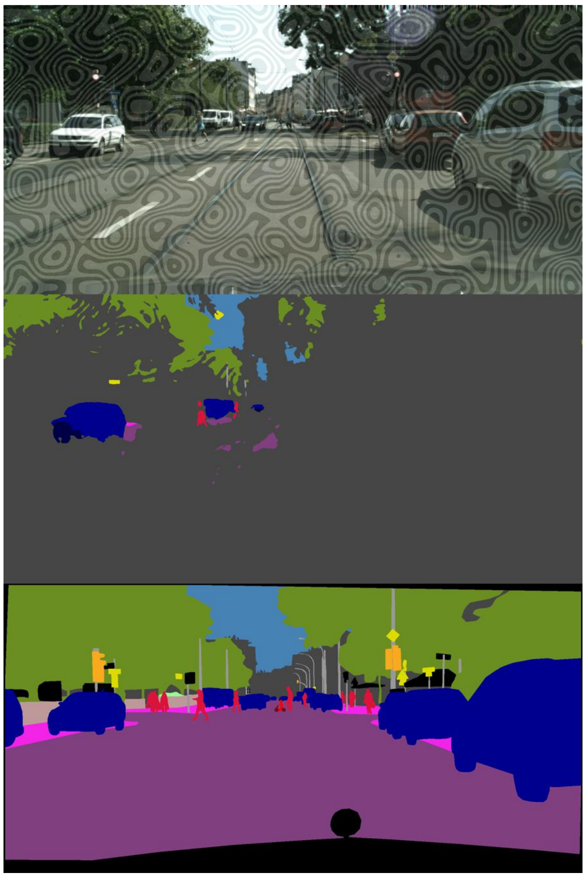

(d) Norm 16

Figure 4.11: Input image, segmentation output and ground truth on the Cityscapes dataset under the mean class IoU accuracy Perlin-BO attack for different L_∞ norm values

Table 4.5: SwiftNet performance on clean and noisy inputs (norm 16)

Noise	Mean class acc.	Mean class recall	Mean class precision	Pixel acc.
Clean	75.45	83.09	87.87	95.68
Salt and pepper	37.90	44.39	75.25	80.98
Perlin	12.64	17.63	62.43	38.76

bias towards the building class. Indeed these attacks cause a large fraction of the image to be classified as building. Thus, the building class tends to have a very high recall (99.73% for Perlin-BO norm 12) while the 18 other classes have a very low recall (all below 16%), which yields a low mean class recall as this metric is computed by averaging the recall across all classes. On the other hand, the precision of most classes remains relatively high despite the attack while the building class precision is consistently very low (22.91% for Perlin-BO norm 12) in these circumstances which explains why the mean class precision remains so high.

It is also worth noting that in this image-generic scenario Perlin-BO does not outperform Perlin-R. This is most likely due to the comparatively harder function to model for the Gaussian processes in the Bayesian optimisation. Indeed, this task involves only one Bayesian optimisation with 25 queries to optimise a noise pattern for 500 images whereas an image-specific attack would have required 25 queries for each of the 500 images. Furthermore, the high vulnerability of the model seems to contribute to large gradients on the objective function, which in turn reduce the chances of the Bayesian optimisation to converge to a global minimum.

Figure 4.11 seems to confirm the vulnerability of the SwiftNetRN-18 model to black-box Perlin noise attacks, especially with L_∞ norm greater than 6, such as 12 or 16. Moreover, as we can see on the segmentation resulting from the attack in 4.11d, the patterns typical of Perlin noise tend to create artificial edges in the image which are sometimes interpreted as such by the model. This may in part explain how Perlin noise acts as a basis for adversarial examples. However, it does not appear to be the main reason. Indeed, much more than new edges, what we observe on the adversarial samples is a shrinkage of all labels except for the building label which becomes prevalent in almost every segmentation. Considering the results on object detection which indicated that Perlin noise caused a decrease in the confidence of the model, this bias towards the building class on adversarial samples seems to suggest that the SwiftNet model is biased towards classifying uncertain pixels into the building class.
Finally, these images also illustrate that Perlin noise patterns with a high frequencies such as on 4.11b and 4.11c are much less visible (and even more so when resized) than patterns with smaller frequencies such as in 4.11d. This effect is exacerbated when resizing large images into a smaller format as it is the case here with originally 1024x2048 pixels images. Such examples lead us to challenge the suitability of the L_∞ norm of 16 as a standalone standard to limit the visibility of adversarial perturbations to the human eye.

As a means to evaluate how effective Perlin noise attacks are on semantic image segmentation models, we compare in table 4.5 the impact on the model of one random Perlin noise pattern and random salt and pepper noise added to every image with a maximum L_∞ norm of 16.

Table 4.5 indicates that SwiftNet is somewhat vulnerable to random salt and pepper noise

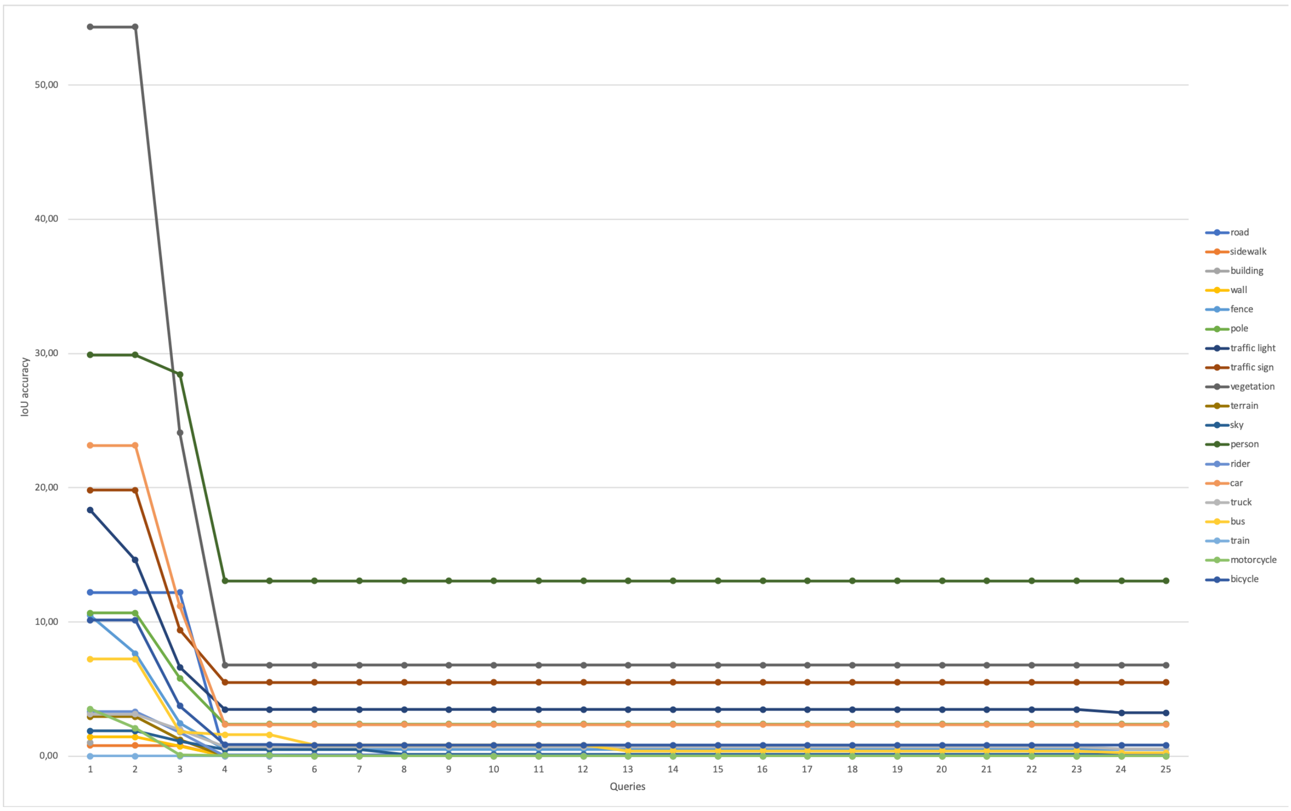


Figure 4.12: Decrease of SwiftNet IoU accuracy per class under Perlin-BO attack with norm 16

given an L_∞ norm of 16. Although we observe a large drop from 75.5% to 37.90% in terms of mean class accuracy, the decline in pixel accuracy is only from 95.68% to 80.98%. This result indicates that this attack is most likely very impactful on some little represented classes while less effective overall as the mean class acurracy attributes the same weight to every class. This appears mostly due to a lack of recall on these smaller classes considering the significant drop in recall, meaning a number small classes are disappearing from the segmentation. Meanwhile, a few larger classes may be taking over and absorbing the impact on the precision which explains why the mean precision does not decrease nearly as much. While a very similar metrics behaviour is observed following Perlin noise perturbations, the decrease in performance of the model is much more significant across all metrics, even in terms of pixel accuracy which reaches a low of 38.76%. These results suggest that although the behaviour of the model with respect to random perturbations is similar to the one it shows on Perlin noise perturbations, Perlin noise has a much more severe impact, inaccurately segmenting a large part of the pixels by not detecting most pixels of a large number of classes.

We also report the decrease of the IoU accuracy per class in terms of the number of queries in the case of Perlin-BO with noise 16 in order to gain a more comprehensive understanding of the potential vulnerability of the model to Perlin noise (figure 4.12).

Given the results in figure 4.12, SwiftNetRN-18 appears to be extremely vulnerable to adversarial perturbations generated with procedural noise. However, Bayesian optimisation does not seem to yield any improvement over the random search performed with the first 5 queries. We also observe that the segmentation of a large majority of objects is very brittle, but most of those that are resistant to the first 4 queries remain robust even for 25 queries. This indicates

Table 4.6: SwiftNet untargeted attacks results per objective

Objective	Mean class acc.	Mean class recall	Mean class precision	Pixel acc.
Original	75.45	83.09	87.87	95.68
Mean class acc.	3.18	7.29	60.45	23.56
Pixel acc.	3.18	7.29	56.81	23.56
Mean class recall	3.04	7.16	60.32	23.48
Mean class precision	12.11	16.93	60.43	46.14

that the Perlin noise attack can only slightly be improved through optimisation and underlines the natural effectiveness of Perlin noise patterns as universal adversarial perturbations. Besides, we also notice that the person class is the most robust class of Cityscapes against this attack, with 13.06% IoU accuracy at norm 16 under a Perlin-BO attack, almost double that of the second most robust class (vegetation at 6.79%). An example of this relative robustness of the person person class is given in figure 4.13 where it appears much more resilient than the sidewalk class in particular which is entirely absorbed by the building class. It is important to note that the person class is the most common one within the training data, which may very well have contributed to its relative robustness. Furthermore, the surprisingly high vulnerability of SwiftNet to this attack may in part be due to the small size of its training set. Indeed, despite its fine-grain pixel-level annotations, the Cityscapes training set only contains 2975 images which is rather low to train a deep neural convolutional neural network.

4.2.1.2 Alternative untargeted attacks with different objectives

Following the success of the IoU mean class accuracy attack, we decided to target different metrics, namely the pixel accuracy, mean class recall and mean class precision to investigate whether different objectives can be achieved by procedural noise attacks. These attacks are based on the same methodology as the IoU mean class accuracy ones, only the objective function differs, with the aim to minimise a different metric. The results are shown in table 4.6.

Given its results, this experiment allowed us to very effectively target the mean class recall and the mean class accuracy. However the pixel accuracy produces very similar results as in some cases mean class accuracy and pixel accuracy are correlated and the 25 queries of Bayesian optimisation are not sufficient for the Gaussian processes to better model the objective function. We also observe that the optimisation for the mean class precision does not converge to the minimum as the pixel accuracy attacks boasts a lower precision (56.81% versus 60.43%). Such a result could be due to the granularity of that objective function as well as the small number of queries and the generic objective of the attack which affect the statistical significance of this experiment. However, in order to investigate that hypothesis, we implemented a Bayesian optimisation with 110 queries including 10 initial queries under the mean class precision objective which resulted in a mean class precision of 61.57. This shows that the mean class precision is particularly hard to optimise for with Bayesian optimisation. This is most likely due to the fact that the natural properties of Perlin noise support a high precision by erasing with a tendency to erase all class labels apart from the building labels in the segmentation as mentioned previously.

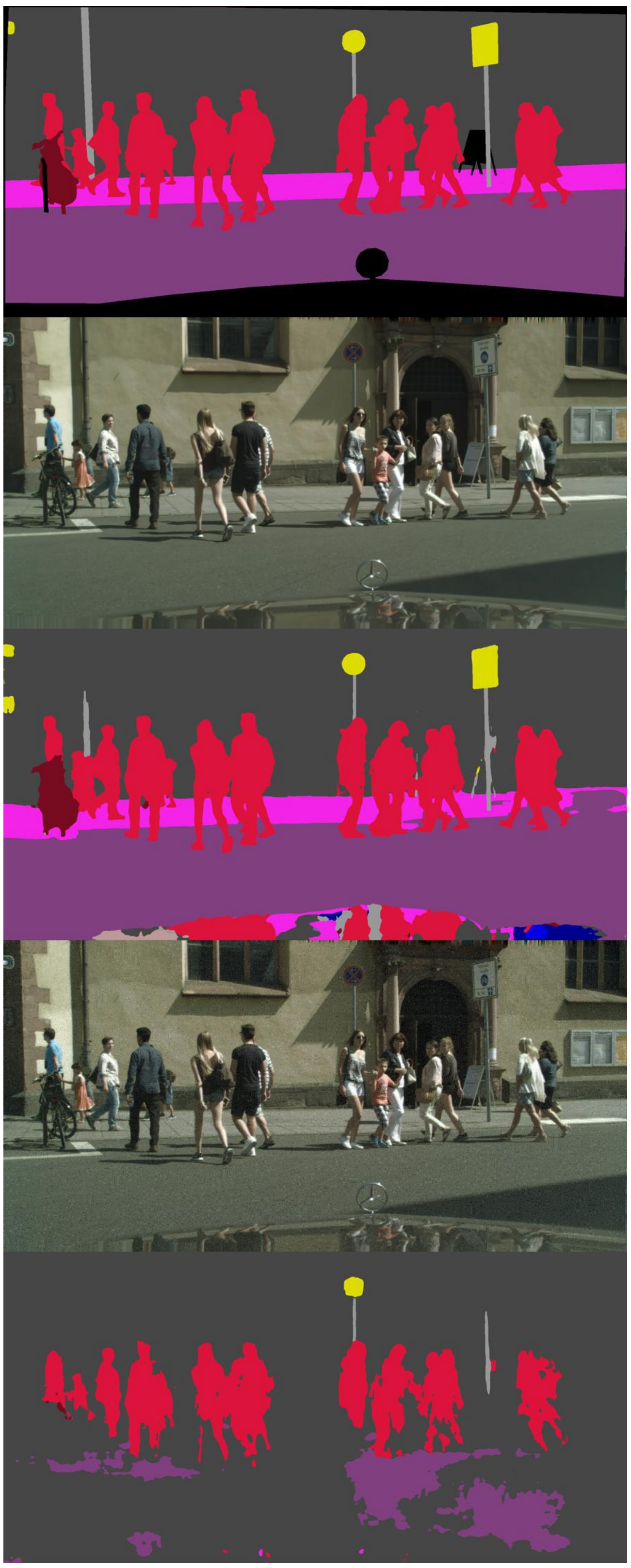


Figure 4.13: Example of the relative robustness of the person class. From top to bottom: ground truth, original image, original SwiftNet segmentation, Perlin-BO norm 12 adversarial example and resulting segmentation. The class labels are the following: red: person, dark gray: building, purple: road, pink: sidewalk, yellow: traffic sign, light gray: pole, dark red: bicycle.

4.2.2 Targeted attack

Beside the general metrics used above as objectives for untargeted attacks, we implemented a targeted attack against SwiftNetRN-18 with the aim to learn whether the Bayesian optimisation strategy with procedural noise may be used as a query-efficient targeted attack vector against semantic image segmentation models. Once again, we limit ourselves to 5 initial queries and 20 additional optimisation queries. In order to make this attack comparable to the mean class accuracy one and reproducible, we set the same random seed to 0 as we have done in the previous cases. To ensure our results are not biased by the possible weakness of the targeted class, we run it in the worst case scenario from the attacker's point of view, that is against the most robust class, namely the person class.

This results in a person IoU accuracy of 13.06% after only 4 queries as the model appears very fragile. This is also the case in the untargeted attack with the IoU mean class accuracy as objective, which means that the Bayesian optimisation is unable to generate more adequate Perlin noise parameters to lower the person IoU accuracy specifically in a targeted attack than in an untargeted attack. This phenomenon highlights some of the limitations of Perlin noise which has limited flexibility and cannot fool SwiftNetRN-18 with respect every single class individually. Considering that procedural noise is also poorly suited to targeted attacks on object detection as shown above and classification as per [16], it seems generally inappropriate for untargeted attacks against computer vision models.

4.2.3 Defences

Considering the above results, both YOLOv3 on COCO and SwiftNetRN-18 on Cityscapes appear very brittle to query efficient black-box procedural noise attacks. Those models, being the state-of-the-art in their respective domains - real-time object detection and semantic image segmentation -, it raises grave concerns about the security of real-time deep learning based computer vision systems. Therefore, we decided to take our research one step further in search for mitigations of these vulnerabilities.

4.2.3.1 Total Variation Denoising

As a simple initial method consists in applying TV denoising to inputs prior to a pass through the segmentation model with the aim to attenuate the noise, thus increasing the performance of the system. Using the same functions and parameters as for object detection, we measure the impact of this measure on key metrics of SwiftNetRN-18. The results are indicated in table 4.7.

In a similar fashion to the simulation conducted on YOLOv3, we evaluate here the impact of a transferability attack from an unprotected segmentation model to model that has a total variation denoising preprocessing module.

It appears from these results that TV Chambolle and TV Bregman denoising both decrease the performance of the model on original images from 75.45% mean class accuracy to respectively 61.11% and 68.51%. Nevertheless, it also drastically improve the performance in terms of mean class accuracy, precision, recall and mean class accuracy, notably with a more than 500%

Table 4.7: SwiftNetRN-18 performance on adversarial samples (norm 12) with and without prior TV denoising

Model input	Mean class acc.	MC recall	MC precision	Pixel acc.
Original	75.45	83.09	87.87	95.68
Original + TV Chambolle	61.11	74.20	75.71	92.62
Original + TV Bregman	68.51	79.40	81.32	94.40
Perlin-BO	10.42	14.43	79.40	40.92
Perlin-BO + TV Chambolle	61.46	70.98	80.78	92.51
Perlin-BO + TV Bregman	57.57	67.07	79.27	91.31

improvement on the mean class accuracy for both TV Chambolle and Bregman denoising, from 10.42% up to 61.46% and 57.57% respectively.

Chapter 5

Evaluation

5.1 Key achievements

In this project, we evaluated the impact of black-box procedural noise attacks against state-of-the-art object detection and semantic image segmentation models. We successfully showed that such attacks prove effective against different computer vision models and tasks while remaining query-efficient. We also implemented a number of extensions mentioned in the interim report and compared the results of a wide range of such attacks, including input-specific and generic, grayscale and coloured noise, Perlin and Gabor noise, untargeted and targeted attacks. Furthermore, our results showed that Perlin is generally more suitable as an adversarial pattern than Gabor noise and that these attacks are more effective in the untargeted setting. Our findings also revealed that optimisation based procedural noise attacks can be improved by reducing the number of parameters involved in the optimisation process. Besides, we demonstrated that some image denoising methods can be applied to attenuate the impact of such attacks, even though these measures have limitations and tend to decrease the accuracy of the model on legitimate samples.

5.2 Limitations

Despite the insightful results of the the aforementioned experiments, we have been facing a lack of meaningful standards to properly assess how visible adversarial patterns truly are and whether they prevent humans from seeing perceiving some features in an image. Indeed, the l_p norms traditionally used in the literature to limit the distance between original images and adversarial samples do not offer a tight relationship with visual similarity as for example they are unable to capture visual transformations such as rotation and translation. This has been highlighted by Sharif et al. [78] who have shown the insufficiencies of L_0, L_2 and L_∞ norms and suggested the use of different similarity measures such as SSIM but also pointed out their limitations. In the current state of research, there is not a single measure that can properly evaluate the visual similarity between two images.

Besides, the standard performance measure of object detection models, the mAP is not ideal as well pointed out by Redmon et al [75] because it takes into account the per class ordering

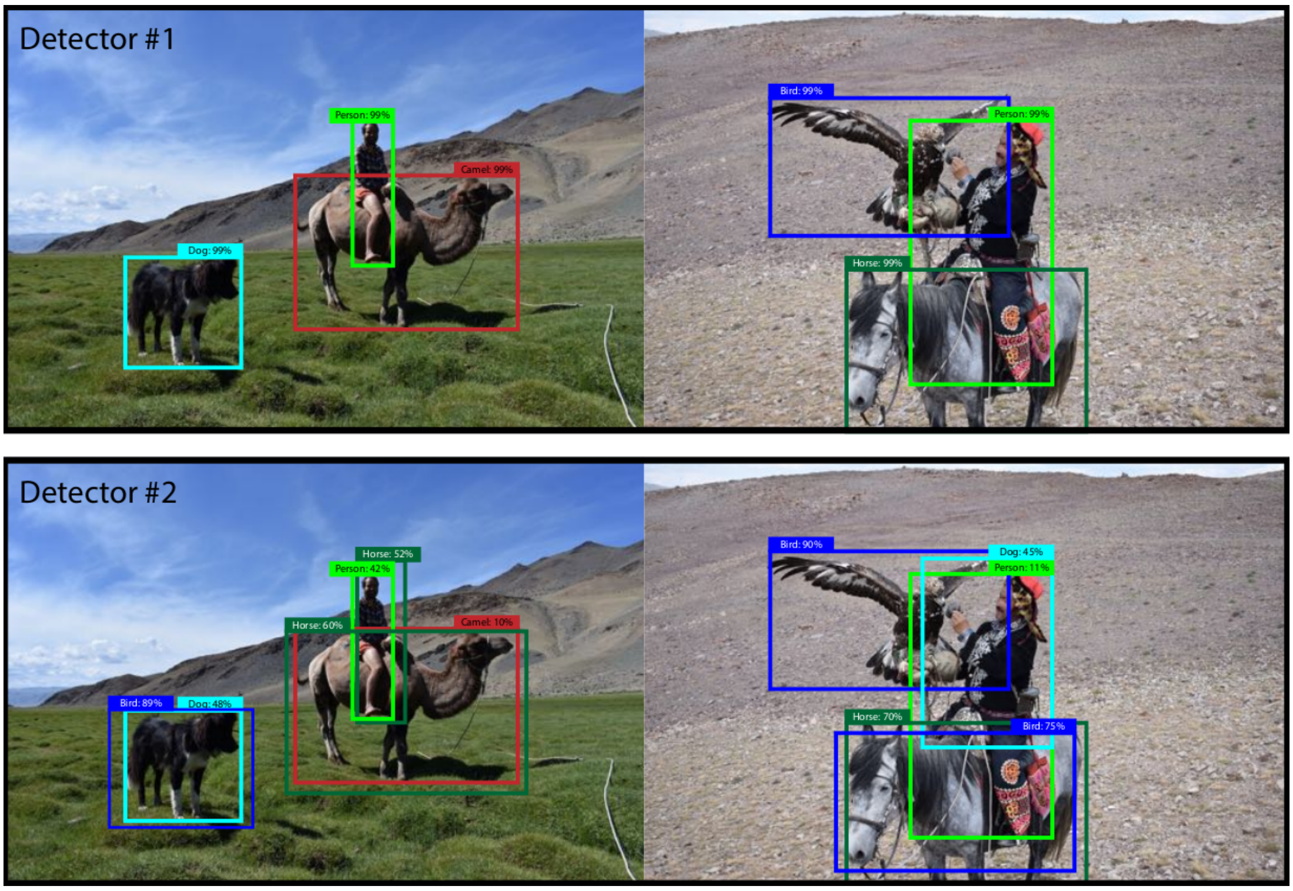


Figure 5.1: Comparison of two detectors scoring a perfect mAP score highlighting the drawbacks of the mAP

of detections in its computation. This leads to surprising results in some cases as shown in the example below 5.1 where the two detectors yield the same mAP on the given images even though the second detector produces a number of false positives while the first one does not. Such results suggest that a more appropriate measure could be developed to evaluate object detection models and the attacks against those.

5.3 Future research

Following our findings on procedural noise attacks against a different computer vision models and tasks, it appears that many questions remain to be elucidated on the matter of black-box procedural noise attacks on deep learning computer vision models and many others stem from the findings of this project. In particular, we have shown that Perlin noise is a natural fit for adversarial perturbations and has the power decrease the accuracy of the target model even without any optimisation and within a strict L_∞ norm. Besides, it vaguely resembles universal adversarial perturbations. As such, it would be interesting to study whether Perlin noise patterns could be used as an initial perturbation vector to speed up the convergence of more expensive and effective attacks.
Another potential area of research is the usage of denoising neural networks to attenuate the procedural noise perturbations, thereby cleaning the data prior to a pass through a computer

vision model. We hypothesize that a denoising autoencoder trained on images containing Perlin and Gabor noise could act as a firewall in front of another model increase the robustness of the system. Such a tailored solution may prove more effective than TV denoising. Nevertheless, further research should investigate the most appropriate weight for the different TV minimisation functions with respect to the norm of the adversarial perturbation. That should be used along with estimator of the norm of the noise in an input image in order to improve the success of the denoising process.

Finally, the number of different functions used thus far in the context of evasion attacks relying on Bayesian optimisation with only Perlin and Gabor procedural noise having been tested. This is in part due to the constraints implied by the the use of Bayesian optimisation and the context of evasion attacks. Indeed, such a function requires a limited number of parameters and should most likely produce low level patterns similar to the ones learned by deep convolutional networks or resemble objects in order to fool the target model. This makes fractals a suitable candidate, as several fractal functions take a restricted number of parameters and have a similar appearance to natural objects such as trees or flowers. Another method we could apply is to generate a significant number of random adversarial patterns using a procedural noise function with a large parameter space and identify the parameters that play the most important role on the evaluation metrics of the target model by means of a correlation matrix. This could allow us to perform Bayesian optimisation on a restricted set of parameters.

Chapter 6

Conclusion

In this project we have shown that state-of-the-art convolutional neural networks used in real-world applications are highly vulnerable to procedural noise, enabling successful black-box attacks with a limited number of queries. As such, some of the work carried out in this project and described in this paper has contributed to the paper "Procedural Noise Adversarial Examples for Black-Box Attacks on Deep Convolutional Networks" [16] which has recently been submitted for publication at a security conference. Indeed, we have introduced and implemented a wide range of black-box procedural noise attacks against real-time object detection and semantic image segmentation models which are the most query-efficient ever achieved with substantial results. In particular, we have tested the performance of these attacks against two state-of-the-art deep convolutional neural networks: YOLOv3 and SwiftNetRN-18 on the COCO and the Cityscapes dataset respectively. The extensiveness of our experiments can be summarised as follows:

YOLOv3:

- Perlin-BO and Perlin-R image-specific average IoU attack for different number of queries (up to 25) and L_∞ norm (2, 6, 12 and 16)
- Image-specific salt and pepper average IoU reference attack (25 queries, norm 16)
- Image-specific coloured Perlin-BO average IoU attack (25 queries,norm 16)
- Image-specific Gabor noise average IoU attack from 6 to 25 queries (norm 16)
- Image-generic attack for 1000 different patterns for each of Perlin, Gabor and random noise and norm 2, 6, 12 and 16.
- Image-specific Perlin-BO targeted attack (25 queries, norm 16)
- TV denoiser defence against Perlin-BO image-specific average IoU attack transferability attack (norm 16) to model with TV denoiser (Bregman and Chambolle).

SwiftNetRN-18

- Perlin-BO and Perlin-R image-generic IoU accuracy attack (norm 2, 6 12 and 16)

- Image-generic salt and pepper reference attack (norm 16)
- Perlin-BO image generic attacks with different objectives, namely mean class precision, mean class recall and pixel accuracy (norm 16)
- Perlin-BO image-generic targeted IoU accuracy attack (norm 16)
- TV denoiser (Chambolle and Perlin) defence against Perlin-BO image-generic IoU accuracy transferability attack (norm 12)

Alongside these implementations we have provided comparisons between the various experiments as well as in-depth comments of the results and their implications, which we summarise here.

On YOLOv3, we have shown that image-specific attacks with norm 16 can decrease the mean average precision of YOLOv3 by 60% from 0.585 to 0.231 on the COCO validation set with only 25 queries. We also measured that random salt and pepper attack with the same norm only decreases the mAP by 7% down to 0.545, showing that Perlin noise has inherent properties as a basis for adversarial patterns. Besides, it also appeared that coloured Perlin noise does not perform as well as its grayscale counterpart with such a low number of queries with a resulting mAP of 0.334. We have also conducted targeted experiments, in particular against the most common class in the dataset, which have revealed that very little improvement can be gained on the targeted objective compared to an untargeted attack. This highlighted that procedural noise attacks are in general more suited for untargeted attacks. Then, our comparative study of image-generic untargeted attacks across different norm values has revealed that Perlin noise tends to be more effective and flexible than Gabor noise and even more so compared to random noise. This study also revealed that the norm of the noise plays an overwhelming role in the success of an attack with a decrease in mAP of up to 10%, 32%, 62% and 75% respectively for an L_∞ norm of 2, 6, 12 and 16. This observation places the trade-off between attack efficiency and imperceptibility at the heart of the design of such attacks and should be considered in the development of adversarial samples detectors to ensure their sensibility is high enough to detect any attack likely to affect the usability of the model.

SwiftNetRN-18, one of the state-of-the-art real-time semantic image segmentation models, also appears extremely brittle against black-box Perlin noise attacks with a mean class accuracy dropping from 75.45% to 3% and pixel accuracy decreasing from 95.68% to 23.56% in just 25 queries under an image-generic attack with norm 16. However, the image-generic attack does not appear to perform better with Bayesian optimisation than with random search within such a low number of queries. This is most likely due to the lack of robustness of the models which produces extremely poor results under both kinds of attacks revealing a strong bias, segmenting almost the entirety of many adversarial samples as buildings.

Finally, we have shown that procedural noise attacks with Bayesian optimisation can be largely mitigated by applying TV denoising on adversarial samples in the case those samples were crafted against a model without any sort of defence. For example, an image-generic Perlin-BO attack within $L_\infty \leq 12$ decreases the 75.45% mean class accuracy of SwiftNetRN-18 by 86% on the bare model while that metric only decreases by 19% if prior TV Chambolle denoising is applied. However the use of such preprocessing implies a trade-off between the performance of the model on legitimate samples and its performance on adversarial samples. Besides, total variation denoising also impacts the speed of the model which may be critical in some applications of the real-time target models. Therefore, despite being a rather effective defence mechanism for object detection and semantic image segmentation models, TV denoising exhibits drawbacks preventing it to be widely used in a number of real-world applications. This implies that further defences will need to be developed to cover those cases in which TV denoising is not appropriate.

Bibliography

[1] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "DARTS: deceiving autonomous cars with toxic signs," *CoRR*, vol. abs/1802.06430, 2018.

[2] A. Drouin, *Classification vs regression.* Alexandre Drouin, 2017.

[3] M. Davis, *Example Clustering.* Apr 2018.

[4] M. Barnes, *The reinforcement learning framework.*

[5] D. Stansbury, *Key steps of the backpropagation algorithm.* Sep 2014.

[6] L. Weng, *Denoising Autoencoder.* Lilian Weng, Aug 2018.

[7] D. Cornelisse, *Convolutional layer.* freeCodeCamp, Apr 2018.

[8] T. Kothari, *Max Pooling with a 2x2 Kernel and 2 Stride.* Nov 2017.

[9] J. Xu, *R-CNN architecture.* Towards data science, Sep 2017.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[12] V. Behzadan, *Adversarial examples visualisation.* Offsec research, Jun 2017.

[13] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[14] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *CoRR*, vol. abs/1610.08401, 2016.

[15] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2774–2783, 2017.

[16] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep neural networks," *CoRR*, vol. abs/1810.00470, 2018.

[17] A. Lagae, S. Lefebvre, R. L. Cook, T. DeRose, G. Drettakis, D. S. Ebert, J. P. Lewis, K. Perlin, and M. Zwicker, “A survey of procedural noise functions,” *Comput. Graph. Forum*, vol. 29, no. 8, pp. 2579–2600, 2010.

[18] “A closer look at yolov3,” Aug 2018.

[19] A. Rosebrock, *Intersection over Union (IoU).* May 2018.

[20] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, “In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images,” *CoRR*, vol. abs/1903.08469, 2019.

[21] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, “On detecting adversarial perturbations,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[22] D. Hendrycks and K. Gimpel, “Early methods for detecting adversarial images,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

[23] X. Li and F. Li, “Adversarial examples detection in deep networks with convolutional filter statistics,” *CoRR*, vol. abs/1612.07767, 2016.

[24] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, “Detecting adversarial samples from artifacts,” *CoRR*, vol. abs/1703.00410, 2017.

[25] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” *CoRR*, vol. abs/1511.04508, 2015.

[26] X. Wei, S. Liang, X. Cao, and J. Zhu, “Transferable adversarial attacks for image and video object detection,” *CoRR*, vol. abs/1811.12641, 2018.

[27] A. J. Bose and P. Aarabi, “Adversarial attacks on face detectors using neural net based constrained optimization,” in *20th IEEE International Workshop on Multimedia Signal Processing, MMSP 2018, Vancouver, BC, Canada, August 29-31, 2018*, pp. 1–6, 2018.

[28] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. L. Yuille, “Adversarial examples for semantic segmentation and object detection,” *CoRR*, vol. abs/1703.08603, 2017.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[30] E. Tretschk, S. J. Oh, and M. Fritz, “Sequential attacks on agents for long-term adversarial goals,” *CoRR*, vol. abs/1805.12487, 2018.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[32] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR ’14, (Washington, DC, USA), pp. 1725–1732, IEEE Computer Society, 2014.

[33] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[34] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[35] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[36] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[37] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016.

[38] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[39] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[40] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *CoRR*, vol. abs/1611.09326, 2016.

[41] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec 2011, Chicago, IL, USA, October 21, 2011*, pp. 43–58, 2011.

[42] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 399–414, 2018.

[43] L. Muñoz González and E. Lupu, *The Security of Machine Learning Systems*, p. 47–79. Intelligent Systems Reference Library, 2019.

[44] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," *CoRR*, vol. abs/1708.08689, 2017.

[45] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of adversarial training examples in poisoning attacks through anomaly detection," *arXiv preprint arXiv:1802.03041*, 2018.

[46] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," *CoRR*, vol. abs/1706.03691, 2017.

[47] P. Lee, "Learning from tay's introduction," Mar 2016.

[48] J. Vijayan, "Cerber ransomware now evades machine learning," Mar 2017.

[49] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016.

[50] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[51] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

[52] A. Arnab, O. Miksik, and P. H. S. Torr, "On the robustness of semantic segmentation models to adversarial attacks," *CoRR*, vol. abs/1711.09856, 2017.

[53] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016.

[54] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pp. 372–387, 2016.

[55] I. Oseledets and V. Khrulkov, "Art of singular vectors and universal adversarial perturbations," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[56] K. T. Co, L. Muñoz-González, and E. C. Lupu, "Sensitivity of deep convolutional networks to gabor noise," 2019.

[57] N. Carlini and D. A. Wagner, "Defensive distillation is not robust to adversarial examples," *CoRR*, vol. abs/1607.04311, 2016.

[58] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel, "On the (statistical) detection of adversarial examples," *CoRR*, vol. abs/1702.06280, 2017.

[59] Z. Gong, W. Wang, and W. Ku, "Adversarial and clean data are not twins," *CoRR*, vol. abs/1704.04960, 2017.

[60] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, pp. 3–14, 2017.

[61] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *CoRR*, vol. abs/1704.02654, 2017.

[62] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pp. 506–519, 2017.

[63] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[64] A. Athalye, N. Carlini, and D. A. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 274–283, 2018.

[65] K. Perlin, “An image synthesizer,” in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1985, San Francisco, California, USA, July 22-26, 1985*, pp. 287–296, 1985.

[66] K. Perlin, “Improving noise,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 681–682, 2002.

[67] M. Olano, J. Hart, W. Heidrich, and M. McCool, *Real-Time Shading.* CRC Press, 2002.

[68] G. Wyvill and K. Novins, “Filtered noise and the fourth dimension,” in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999, Los Angeles, CA, USA, August 8-13, 1999, Abstracts and Applications*, p. 242, 1999.

[69] K. Perlin and F. Neyret, “Flow Noise.” 28th International Conference on Computer Graphics and Interactive Techniques (Technical Sketches and Applications), Aug. 2001.

[70] R. Bridson, J. Houriham, and M. Nordenstam, “Curl-noise for procedural fluid flow,” in *ACM SIGGRAPH 2007 Papers*, SIGGRAPH ’07, (New York, NY, USA), ACM, 2007.

[71] A. Kensler, A. Knoll, and P. Shirley, “Better gradient noise,” 2008.

[72] J.-P. Lewis, “Texture synthesis for digital painting,” in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’84, (New York, NY, USA), pp. 245–252, ACM, 1984.

[73] J. J. van Wijk, “Spot noise texture synthesis for data visualization,” in *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991, Providence, RI, USA, April 27-30, 1991*, pp. 309–318, 1991.

[74] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré, “Procedural noise using sparse gabor convolution,” *ACM Trans. Graph.*, vol. 28, no. 3, p. 54, 2009.

[75] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *CoRR*, vol. abs/1804.02767, 2018.

[76] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, “Countering adversarial images using input transformations,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

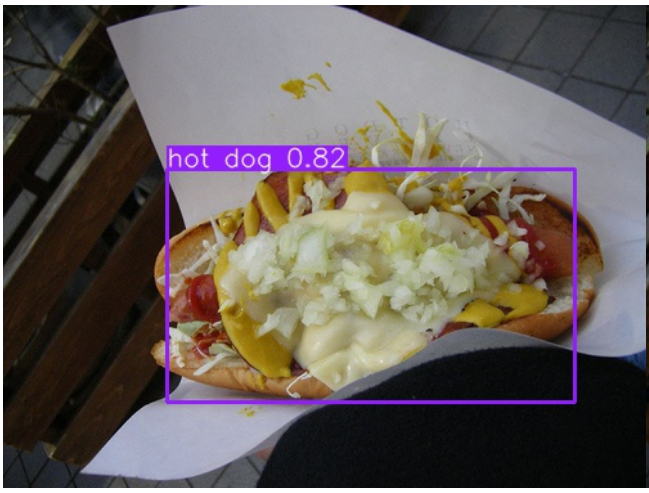
[77] M. Sakurai, S. Kiriyama, T. Goto, and S. Hirano, “Fast algorithm for total variation minimization,” in *2011 18th IEEE International Conference on Image Processing*, pp. 1461–1464, Sep. 2011.

[78] M. Sharif, L. Bauer, and M. K. Reiter, “On the suitability of l_p-norms for creating and preventing adversarial examples,” *CoRR*, vol. abs/1802.09653, 2018.

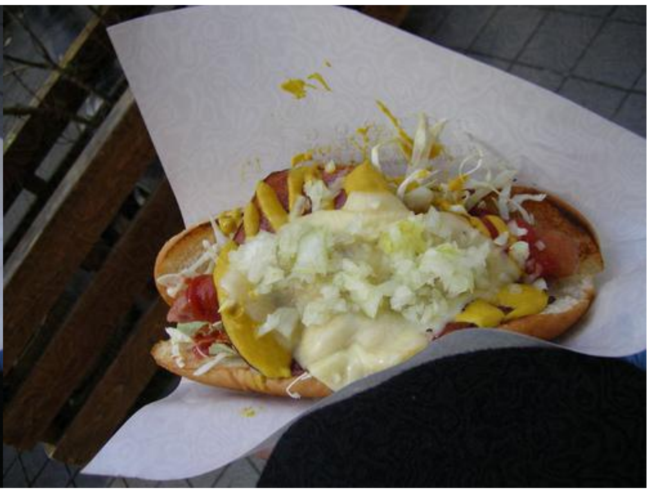
Appendix A

YOLOv3 image-specific Perlin-BO results from MS COCO

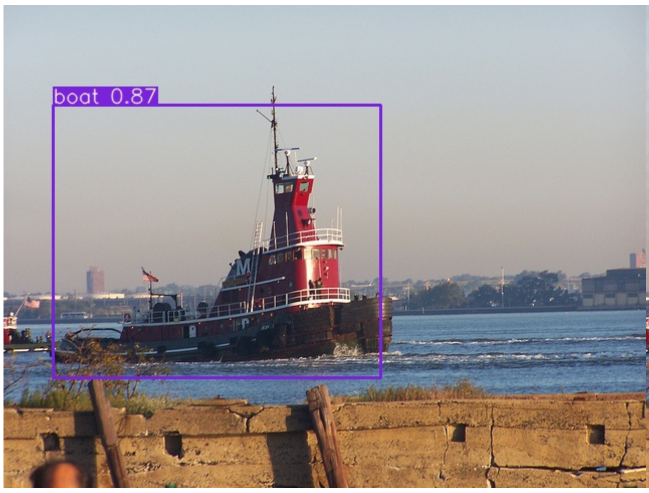
All of the following are examples of detections on legitimate and adversarial samples where none of the original detections persist.



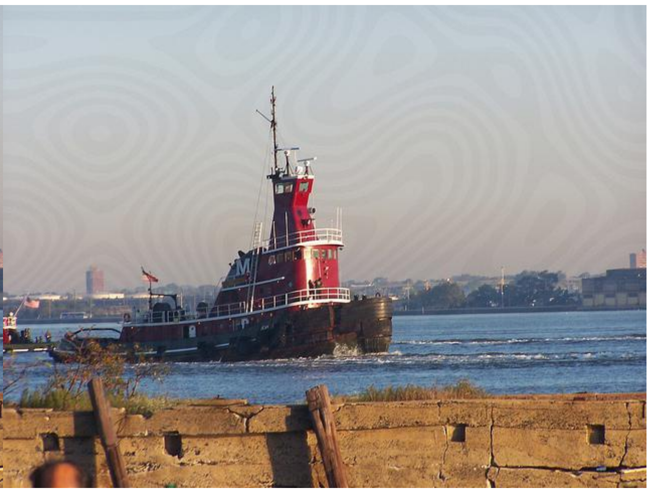
(a) Original detection



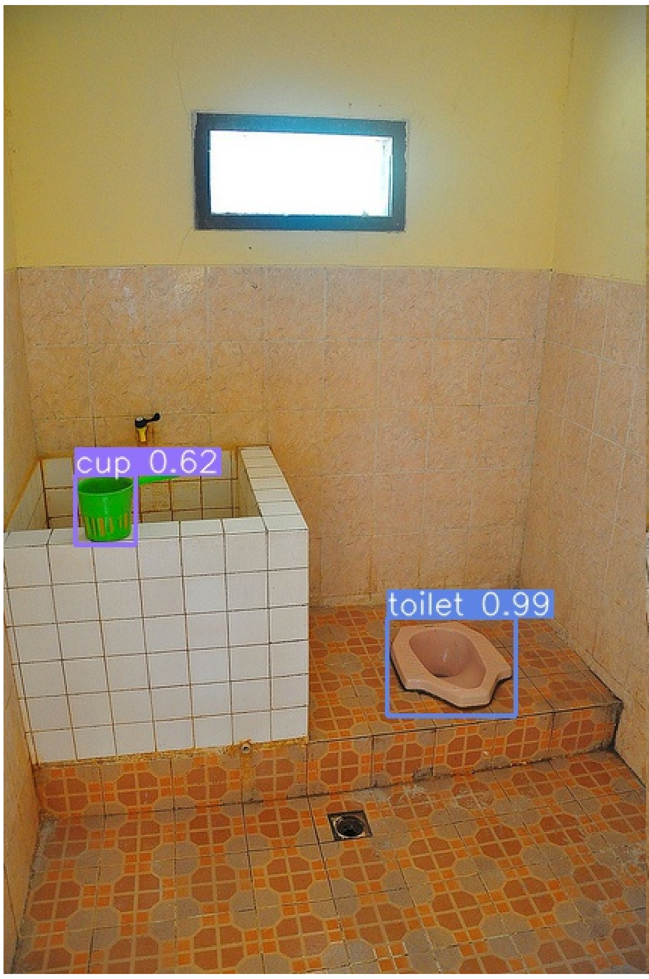
(b) Adversarial sample detection (norm 2)



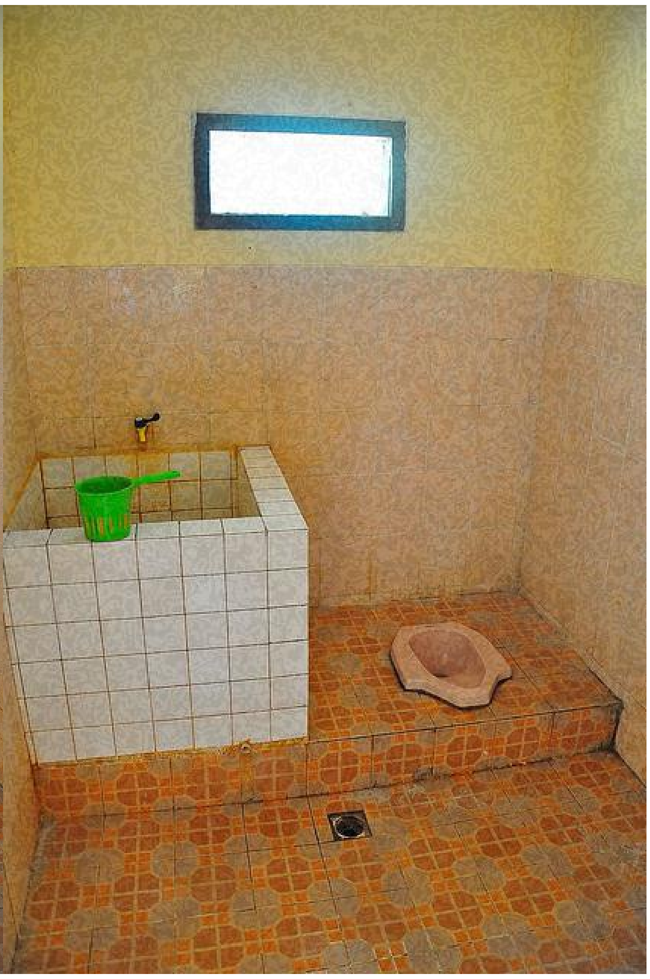
(a) Original detection



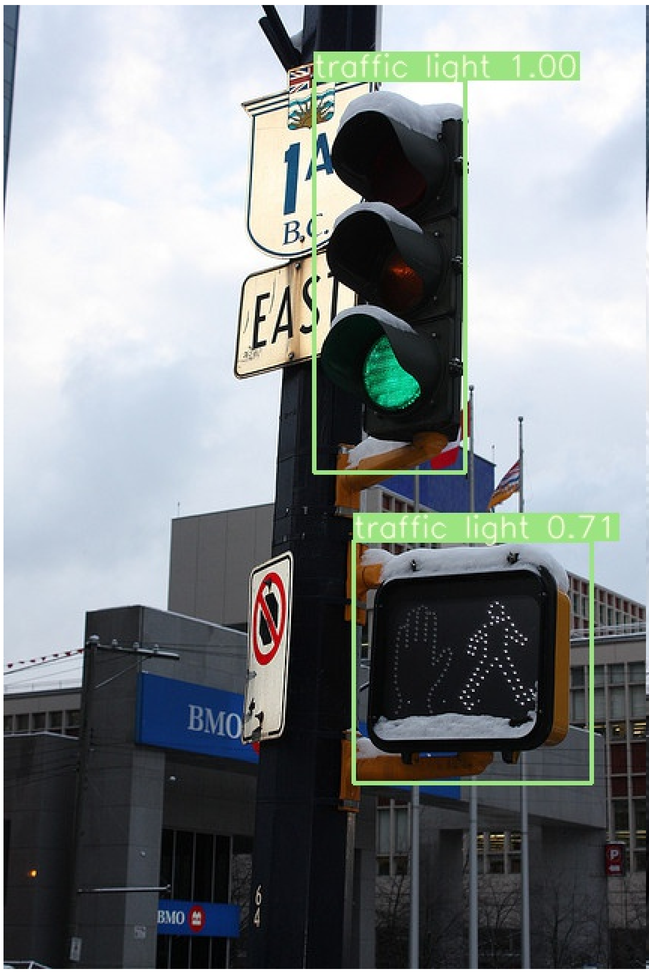
(b) Adversarial sample detection (norm 2)



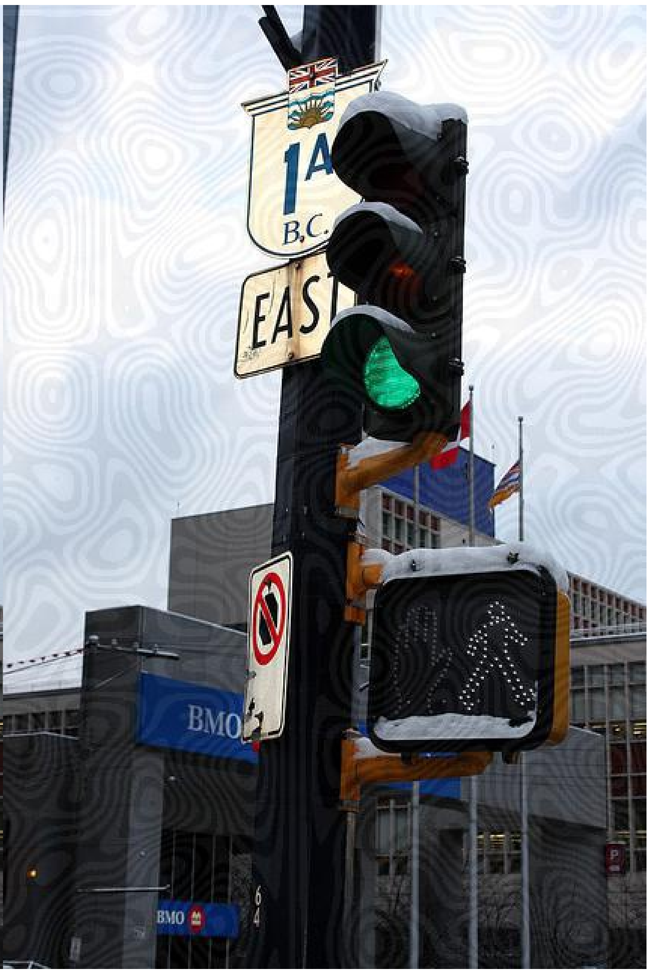
(a) Original detection



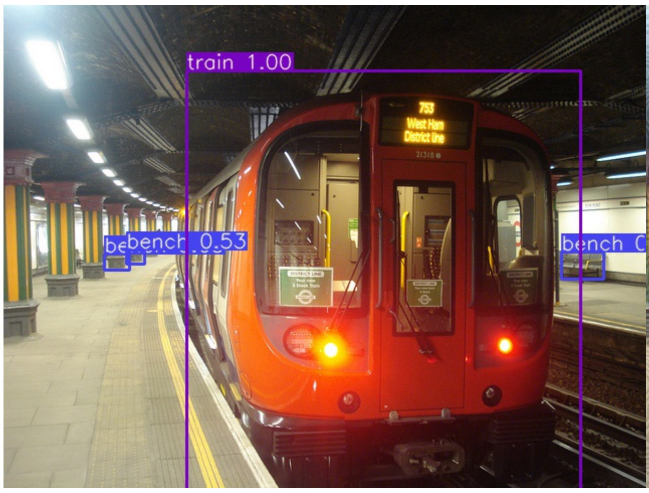
(b) Adversarial sample detection (norm 6)



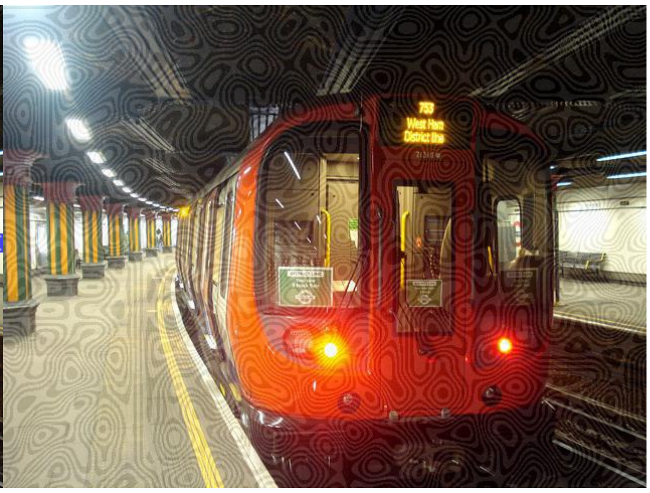
(a) Original detection



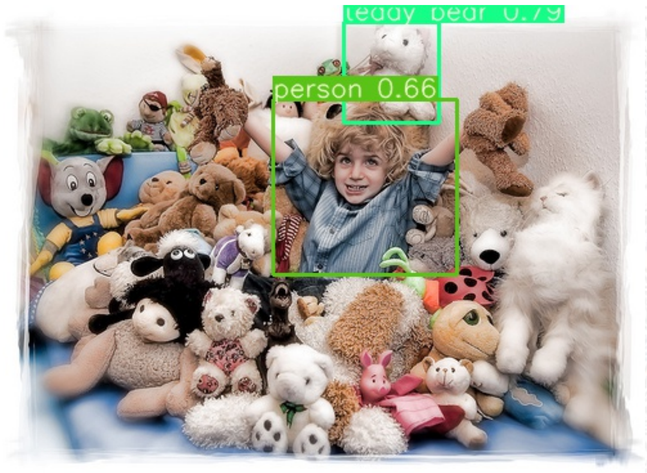
(b) Adversarial sample detection (norm 6)



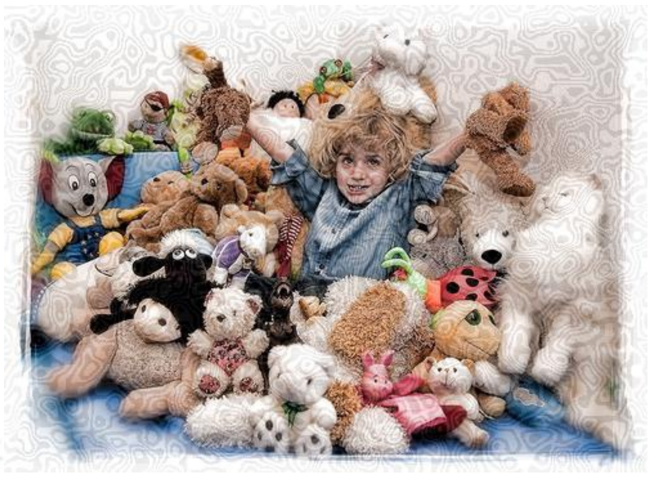
(a) Original detection



(b) Adversarial sample detection (norm 12)



(a) Original detection



(b) Adversarial sample detection (norm 12)

Appendix B

SwitNetRN-18 image segmentation examples on Cityscapes

This appendix exhibits additional examples of Perlin-BO image-generic attacks against SwitNetRN-18, in particular situations revealing the potential threat of such attacks applied if applied in the real world against autonomous vehicles using semantic image segmentation. Each figure contains 5 images, from top to bottom: ground truth, original image, original segmentation, adversarial example under norm 12 and resulting segmentation

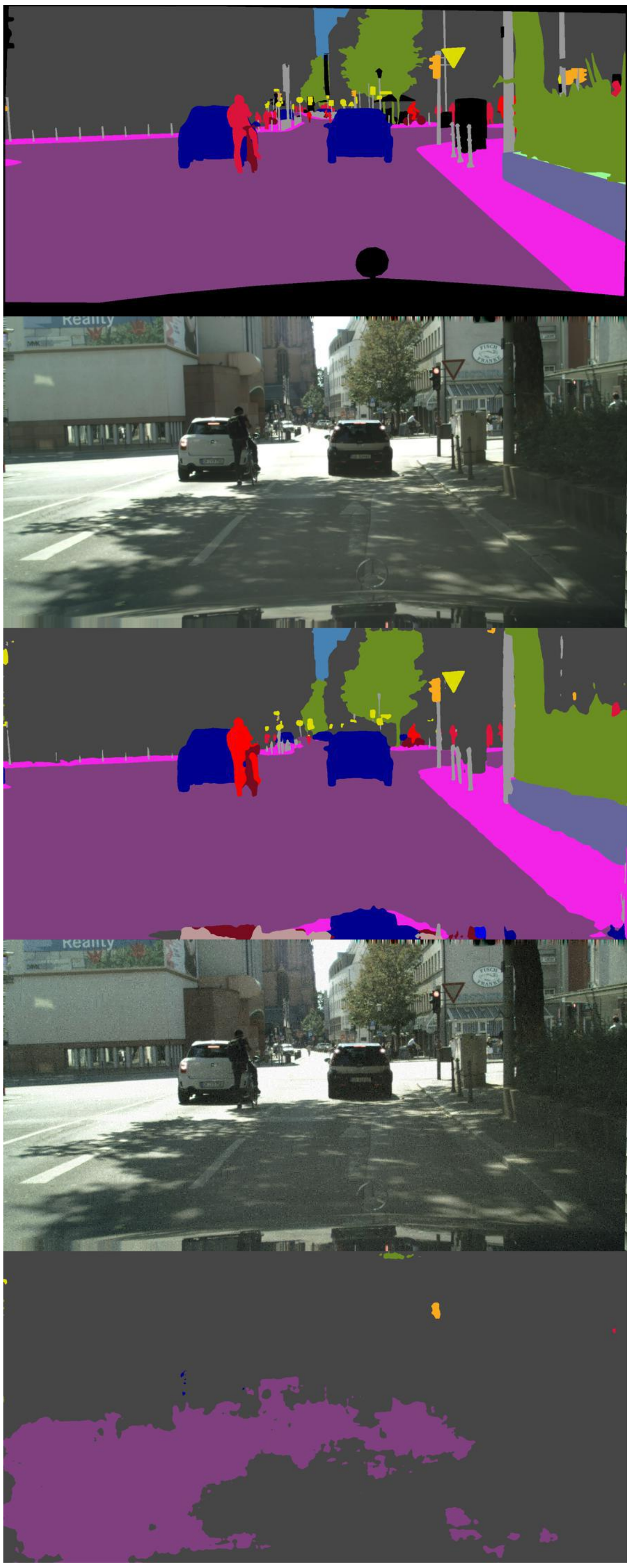


Figure B.1: Example of situation where the model does not perceive cars, a cyclist and a give way sign in front of it.

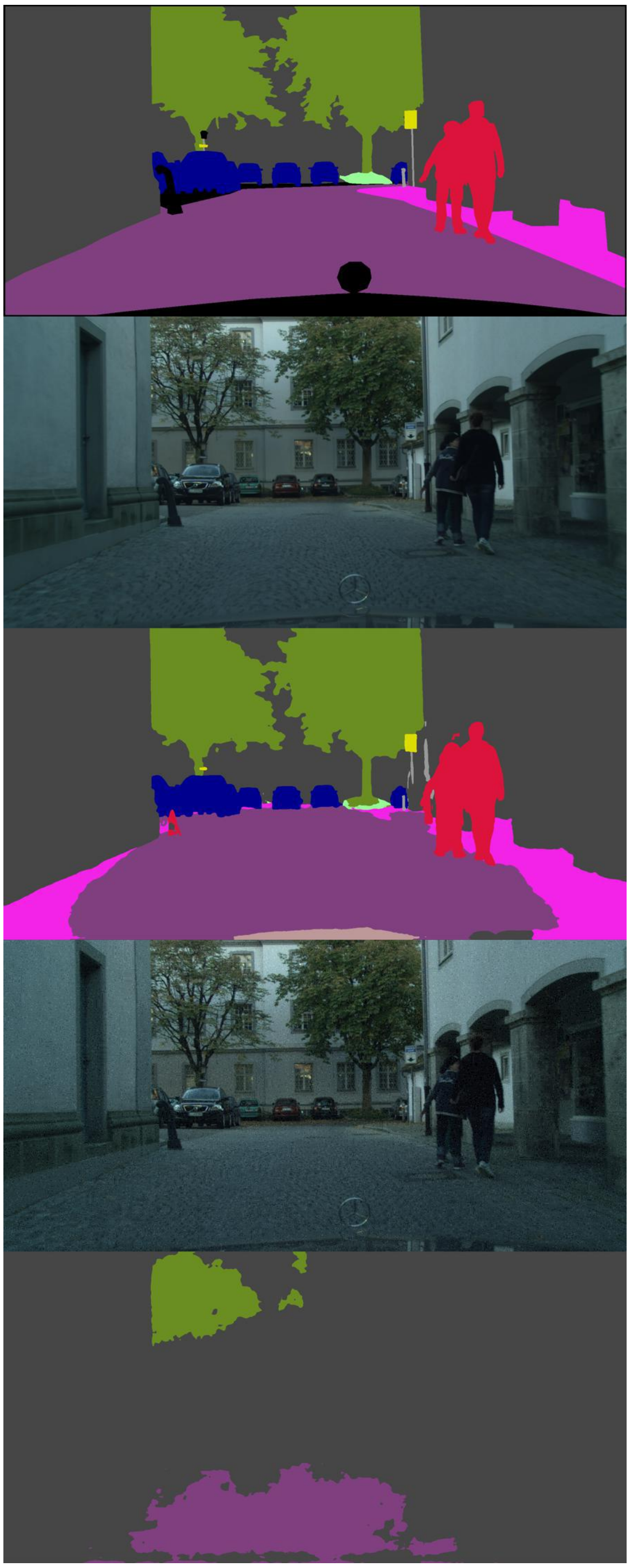


Figure B.2: Life-threatening attack where the model does not see two pedestrians on the road but still perceives the road right in front of it.