Universiteit Leiden
The Netherlands

Master Thesis

Adversarial Detection and Defense in Deep learning

Yuxin Xiong

Supervisors:
Erwin Bakker & Michael Lew

Master Thesis

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

23/08/2021

Abstract

Adding invisible perturbations to the input of Deep Neural Networks(DNNs) arises critical impacts on the performance of DNNs. These adversarial attacks cause problems for many application scenarios of DNNs, for example, autonomous driving and facial recognition. In this paper, we propose a defense method that can protect Resnet101 when facing such attacks. Several aspects are studied, firstly, we take a deep look at adversarial attacks and generate adversarial examples as the setting of attacks in further experiments. Especially, adversarial examples produced by $shapeshifter$ are shown to be robust enough towards physical-world distortions such as different distances and different angles. Then, we focus on a framework UNMASK[15] to detect and defend against attacks. The idea of UNMASK is to extract features of certain classes by a semantic segmentation technique. By comparing extracted features, UNMASK framework can detect whether the input image is benign, and can counter against attacks by refining to the correct class. In addition, we propose a modification to the UNMASK model by adding 4 feature denoising blocks which is robust to various attacks. Finally, we evaluate the performance of the UNMASK in terms of defense. The new architecture significantly improves the robustness of UNMASK system on 4 unmask subdatasets by 6.53%. Especially, our new architecture greatly improves the defense towards Momentum Iterative Fast Gradient Sign Method(MI-FGSM) from 61.88% to 77.42%. The average accuracy of our new architecture under two attack strengths of MI-FGSM(L_∞) is shown to increase by 15.54% compared to the baseline UNMASK.

Key Words: deep learning, adversarial examples, adversarial attacks, adversarial training, feature extraction, feature denoising

Contents

1 Introduction

Deep learning has attained significant achievements on various Computer Vision(CV) tasks [27, 25]. Deep neural networks obtain useful features from raw data and require less professional knowledge. Due to these benefits, DNNs have been widely applied to our daily life [41, 52, 16]. However, recent researches figure out that almost all DNNs achieve poor performance against the well-perturbed input data [17, 45, 8, 24, 28, 35, 39], which would raise great risks in AI security scenes. For example, object detectors in autonomous driving recognize the STOP road sign with some stickers as other signs[13]. How we can make sure the results of Deep learning models are reliable under abnormal inputs becomes one of the most concerning questions to researchers when CV techniques are applied in the real world.

The risks caused by designed input instances in DNNs have aroused widespread attention in the field of CV, which in turn has made tremendous progress regarding adversarial attack method. A adversarial instance is generated by using a benign image and adding some small perturbations [26, 13, 6], which are usually difficult to be noticed by human beings. However, adversarial examples can successfully fool classifiers and detectors to get the wrong result. In general, adversarial attacks evaluated by using the baseline architectures in the ImageNet challenge [10]. Perturbations generated by the attack algorithms usually are so easily to be distorted that they can not achieve the expected attack performance in real-life circumstances [31]. In theory, the perturbations also need to be robust to resist different weather, light conditions, angles, distances, and other changes that may happen in the real world tests [13]. These adversarial attacks will not only lead to poor performance of classifiers, but also has many impacts to their applications in the physical world. For example, misclassfication will cause safety problems in autonomous driving scenarios.

Recently, many state-of-art attacks have been proposed to impose perturbations on DNNs [7, 9, 26, 11, 32]. In this project, we select some representative classic attack methods [11, 32, 26] as the main targets of defense and detection. Besides, we also explore and reproduce the state-of-art attack [9] that are still superior in mistracking the noise as encountered in the real world. In our robustness framework, there is a pivotal idea to be understood which is "robust features". For example, an image with class "bike" is supposed to have parts like wheel, frame, chains, etc. If this image is modified by an adversarial attack algorithm, these parts which are robust features should not be changed in both the eye as well as for the DNN models. This finding suggests that we can use robust features as the basis for judging the class of object in the defense and detection of adversarial learning.

Current studies indicate that there are some researches towards how to protect DNNs from various adversarial attacks. The current mainstream and effective strategy is the Adversarial Training(AT) [26, 46]. AT can solve the "minimax" problem very well since attack methods are usually maximization problems and the defense is mainly to minimize adversarial loss. However, according to [17], the defense method AT lacks the ability to leverage human subjective perception of objects in the learning process. To fill this gap, [15] proposes the knowledge extraction based framework "UNMASK" which highly improves the robustness of DNNs compared with Adversarial Training method, UNMASK shows the possibility to utilize

human subjective perception in adversarial defense and detection. Starting with [15], this paper presents the detection and defense of UNMASK for different attacks, and propose our new architecture of a DNN-protected framework based on UNMASK with better robustness.

Our contributions are as follows:

1. We have conducted in-depth researches on various state-of-art attack methods and algorithms which were also used as attacks in defense and detection systems. The attack algorithms involved in the project are: PGD, MI-FGSM, and Shapeshifter. We used 4 attack methods which are PGD-L_∞, PGD-L_2, MI-FGSM-L_∞, and MI-FGSM-L_2, each with two attack strengths as the tested attacks in our experiments.

2. With [15] as the starting point, we presented the unmask framework and implemented a robust classification task pipeline. The project used the manually generated original "unmask" dataset, which was obtained by a subset of imagenet, PASCAL-Part, PASCAL VOC 2020 and Flickr dataset after a series of preprocessing.

3. We proposed a new architecture by adding feature denoising blocks which highly improved the performance of UNMASK framework. Especially for the attack MI-FGSM(L_∞) on which the original UNMASK showed its worst performance , the new architecture increased the performance by 15.54%.

4. We conducted a series of experiments from many aspects to observe the performance and improvements. We evaluated the defense performance of UNMASK on different unmask subsets and analyzed how the number of classes and the rate of feature overlap contained in the dataset influence the results. We compared the improvements of the new architecture compared to UNMASK, showing an increase of the classification accuracy by 6.53%. Furthermore, the results showed that the feature denoising architecture also has significant effects on knowledge aligning except for adversarial training.

5. The code of our project can be referred to https://github.com/Yuxin33/unmask –which allows the reproduction of the results in this paper and contains details of how to run the project.

2 Background

In Section 2, we introduce various concepts involved in this paper in detail. Among them, the robustness of Deep Neural Networks(DNNs) is the main problem, and classifiers are the specific model we study. Adversarial examples provide attacks to our experiments. We conduct adversarial defense and detection to protect classifiers. Furthermore, we also provide Table 1 to explain all symbols appearing.

2.1 Deep Neural Networks

Deep Neural Networks(DNNs) have a long history in the development of computer science and algorithms. Nowadays, the concept of "Deep neural network" is massive and multidisciplinary. In our project, we study and apply so-called Artificial Neural Networks (ANNs). Typically, a DNN is an ANN or Multiple Perceptron with "deep" layers between input and output[23], main components are: neurons, layers, weights, biases, and functions. The increased layers bring more parameters like weights and thresholds which highly increase the learning ability of model and decrease the risk of overfitting in the meanwhile.

DNNs consist of various architectures of networks. These deep architectures have many classic and powerful implementations which perform best in certain specific domains. The performance of different architectures needs to be compared in the same dataset under the same metrics. In this paper, our study mainly concentrates on the Convolutional deep neural network (CNN) such as the Deep residual network (ResNet)[44] which have been proven to be very successful in the Computer vision domain. Besides, various networks offer a pre-trained DNN to tune for a new configuration. The tuning of a pre-trained ResNet model can be regarded as grouping the large number of parameters during the training. The global optimization is based on the local optimal of each group. In this way, the massive parameters concerned in these large DNN models can be trained with less computing resources.

2.2 Classification Tasks

Classification tasks are the fundamental of image-based machine learning, which refers to the prediction of class labels for instances[19]. A classification model calculates how to assign the most suitable class label by learning the input and output instances in the dataset. There are many types of classification tasks, different machine learning models are applied according to the objects to be classified. For a classification task, the goal is to achieve the best classification performance for certain problem domain. We usually choose classification accuracy to evaluate the performance.

As mentioned in Section 2.1, each layer in a CNN architecture processes the output of previous layers, where the whole model converts the relevance of initial inputs and outputs from non-correlated to highly correlated with each other. In other words, multiple layers convert low level features to high level features in CNN model. Therefore, complicated classification tasks are realized by feature learning. The classification task studied in our project is image classification. Recently, there are many DNNs that perform successfully on various classification

tasks. ResNet [44] and DenseNet[20] which have achieved very high performance on image classification problems. Algorithms have also developed to the limit level in performance that achieve the classification accuracy close to 100% on various benchmarks. However, there is a gap to bridge for real-world applications.

2.3 Adversarial Examples

Adversarial attacks were first proposed in 2014[17], here researchers found that ConvNets model will get completely wrong classification results when some well-designed images are inputted. Figure 1 [17] is an example of the first proposed adversarial attack method. The benign image is correctly recognized as "panda" by the given DNN model. However, when carefully constructed noise is added to the original image, a so-called adversarial example, then the DNN model predicts it as "gibbon". Figure 1 depicts the generation of an adversarial example in which the perturbation is very difficult to be observed by human beings' eyes. It has almost no visible differences compared to the clean one.

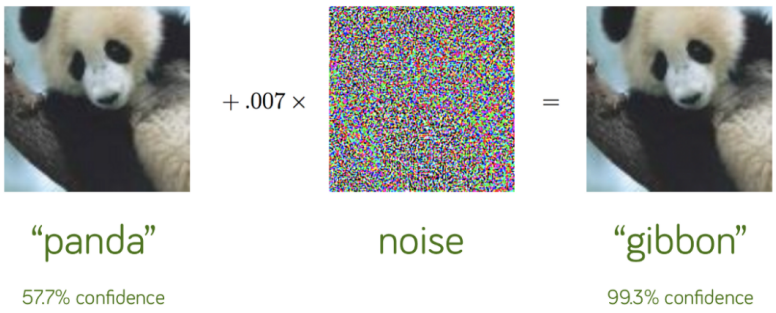


Figure 1: An example of adversarial example misclassified by DNN [1]

Adversarial examples can perform two kinds of attacks, which are targeted attacks and non-targeted attacks. Targeted attack means that the attacked instance should be recognized as a targeted class by a DNN model, while a non-targeted example should be recognized as a wrong class. Assume that a trained DNN model is represented by D, and the original image x in the training dataset $\mathcal{X}$ can be correctly classified by D. In case of a non-targeted attack, assume there exists some distance metric $m(x, x')$. The instance x', $x' \in \mathcal{X}$ performs a targeted attack if $m(x, x') \leq \epsilon$, $\epsilon > 0$, where ϵ is a given perturbation budget. In case of a targeted adversarial attack, the target class $D(x')$ is detected for the perturbed instance x' with class label D. The instance x and x' are required to meet the conditions: $\mathcal{L}_2$ bounded distance $m(x, x') = \|x - x'\|_2^2$, $\mathcal{L}_\infty$ bounded distance $m(x, x') = \|x - x'\|_\infty$.

Adversarial examples are effective for a variety of DNN models since many models are generally similar with respect to decision boundaries[9]. Therefore, the adversarial attack algorithm

[1] source:Explaining and Harnessing Adversarial Examples, Goodfellow et al, ICLR 2015.

is transferable, which also makes the black-box threat model in the physical world possible and arouses people's attention to DNNs security issues. Adversarial examples are used as a critical evaluation metric to the robustness of DNNs[11].

2.4 Adversarial Defense

Adversarial examples put forward unprecedented challenges to the DNN models. DNNs are expected to be robust against adversarial attacks. Figure 2 uses a binary classifier as an example to describe the problems caused by adversarial examples and the theoretically solutions. The set of green points and blue points represent the two classes which can easily be separated by a standard classifier. The boundary box of these points represents the space m for adversarial examples and perturbation budget. Standard classifiers fail to separate these points when including their boundary box. However, an improved robust classifier could cope with such adversarial attacks.

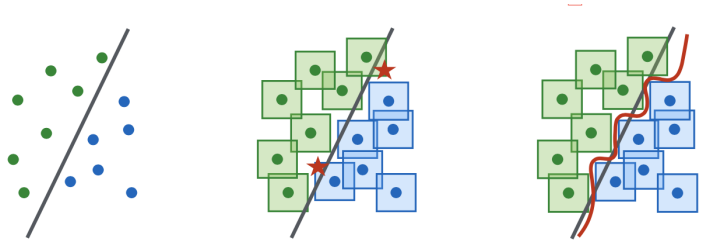


Figure 2: A conceptual description of adversarial examples for a binary classification model[2]. Left: A standard model classifies two set of points successfully. Middle: A standard model fails to classify points with adversarial perturbations. Right: A robust model successfully classifies points with adversarial perturbations.

Towards adversarial examples in DNNs, many methods have been proposed against adversarial attacks[34, 12, 26, 46, 40, 37]. Adversarial Learning (AT) [26, 46] is one of the most effective method. Adversarial learning is to add adversarial examples into the training procedure of DNNs, the model learns to defend with the participation of the attacks[11]. Theoretically, the model will be robust enough against adversarial examples in case there are suffficient attacks.

Adversarial training is a very powerful defense method in the white-box settings. However, it is not applicable in a black-box settings because of the coupling between adversarial attacks and the parameters in the training. Ensemble adversarial training uses multiple adversarial examples from various attacks which is regarded as data argumentation during the training of the target DNN model[46]. In this way, ensemble adversarial training [46] performs well in terms of both white-box settings and black-box settings[11].

[2]source: Towards deep learning models resistant to adversarial attacks.Madry et al, ICLR 2018.

Adversarial training describes a "minmax" optimization which can be expressed as in Equation 1. Here, X is adversarial example, $\delta \in \mathcal{S}$ is perturbations in a certain range, W the weights of the DNN model.

$$\min_{W} \left[E_{(X,y)\sim D} \left(\max_{\delta \in \mathcal{S}} L(W, X + \delta, y) \right) \right] \tag{1}$$

2.5 Adversarial Detection

Adversarial detection detects adversarial inputs in a model to prevent further influences of attacks. Adversarial detection can be realized through several means. Adversarial examples are regarded to have the same label as original examples in a high-dimensional search space[14]. According to this finding, attacks can be detected by statistical analysis like kernel density estimation(KDE) and Bayesian uncertainty estimation(BUE) [49, 14, 18]. Furthermore, adversarial detection can also be done by analysis of hidden layers[33, 29].

2.6 Semantic Segmentation

Semantic segmentation is a critical technique in CV field. The goal is to classify all pixels in an image. Different from instance segmentation, semantic segmentation is conducted according to the class of objects in the image. That is objects in the same class will be annotated with the same color. Semantic segmentation is the foundation of many tasks in CV for its excellent performance in various scenarios. Although semantic segmentation problems could also be solved by traditional Image processing techniques and machine learning, the development of deep learning brings new progress to semantic segmentation. CNN has the most related studies among DNNs. PASCAL VOC dataset is a famous challenge for semantic segmentation, object recognition and object detection. Figure 3 depicts examples of the PASCAL-Part dataset from which we can see that every small part of an object has its mask. In our project, the PASCAL VOC dataset is selected to train and evaluate the object segmented model.

Figure 3: Examples in PASCAL-Part dataset.

2.7 Symbols and Definition

This paper uses a lot of concepts that need special symbols and abbreviations. To avoid confusion, these symbols and abbreviations are explicitly listed and described in Table 1.

Table 1: Symbols and Definition

Symbol	Definition
AT	Adversarial Training
FGSM	Fast Gradient Sign Method
EoT	Expectation over Transformation
x	image instance
y	predicted class label
$\mathcal{X}$	training images dataset
X	input space
X^{adv}	adversarial examples
$m(x, x')$	the distance metric function of image x and x'
L_F	loss function of Faster R-CNN model
$M_t\left(x, \tanh\left(x'\right)\right)$	adding object image x to background image $\tanh(x')$ which is the MoT.
ϵ	perturbation budget, attack strength
$\delta \in \mathcal{S}$	a set of allowable adversarial perturbations
W	weights of model
M	the classification model
K	the extracted model/object detector
D	UNMASK framework
$\mathcal{F}$	a set of robust features
f_r	extracted features from model K
f_e	expected features by the expected class from model M
f_{unr}	useful but not robust features
JS	Jaccard similarity
d	Jaccard distance
$Clip$	clip function that conducts pixel clipping
$J\left(X, y\right)$	cross-entropy loss function of DNNs

3 Related Work

3.1 Adversarial Attacks

The FGSM adversarial attack was proposed in[17] as the first attack occurred. After that, many kinds of digital adversarial examples were generated by changing some critical pixels [7, 9, 11, 32, 24, 28, 35, 39]. In general, adversarial examples conduct attacks to DNNs as the input data, which is a threat model[26]. In this project, we choose 2 digital adversarial attacks which are PGD and MI-FGSM[11, 32] and implement the defense strategies against them in gray-box settings. As the foundation for many attacks, [17] proposed Fast Gradient Sign Method (FGSM). [32] proposed the Projected Gradient Descent (PGD) by adding iterative idea. [11] proposed MI-FGSM by adding momentum iteration. For classifiers, PGD and MI-FGSM are two typical adversarial attacks in a white-box setting. For object detectors, it is hard to attack all possible bounding boxes of objects precisely. Attacking a detector can be regarded as attacking multiple classification tasks at the same time. [50] proposed Dense Adversary Generation (DAG) by iterative optimizing loss function. DAG attack is extended from attacks from classifiers and it is proved to be effective for semantic segmentation. [30] proposed an attack algorithm that successfully attack Faster R-CNN and YOLO detectors in autonomous driving.

Compared to digital adversarial examples, physical adversarial examples[31, 13, 26, 9] provide more applications in real world by resisting interference of the physical world. In [26], the classifier is successfully fooled by printed adversarial examples caught by a smartphone. [26] produced physical adversarial examples simply by printing it out without any modifications. However, [13] proposed Robust Physical Perturbations(RP2) attack algorithm by adding stickers on road signs. RP2 performed 100% misclassification rate in a lab environment and 84.8% in a real-world driving test with different viewpoint angles and distances. Regarding physical adversarial attacks for object detection, [9] proposed the first detector attack algorithm Shapeshifter which is proved to be effective against the Fast R-CNN model. It is a robust method of attack to physical noises that achieves excellent performance to fool a detector in realistic conditions. Besides, Shapeshifter can provide both targeted attacks and non-targeted attacks.

3.2 Adversarial Defense

There are many defense methods proposed to protect DNNs from being attacked[34, 12, 26, 46, 40, 37, 21, 6, 4, 53]. Among them, the most popular defense method is Adversarial Training(AT)[21, 26, 46]. [21] proposed a defense method which added strong adversarial examples to the learning procedure of classifiers. This method can highly improve the robustness of classifier with a satisfied learning accuracy. [32] performed adversarial learning on classifiers in MNIST benchmark and significantly protected classifiers from white-box attacks. Apart from AT, [32] inspired that robustness of DNNs could also be achieved by a better architecture. [6] proposed GPDNN model combining Gaussian processes and DNNs that achieved robust performance with standard DNNs.

It is worth noting that the numbers of current defense methods towards object detectors are limited since and the performance of extending the defense method from classification tasks to object detection is poor. [53] provided very beginning attempt to protect detectors from a few adversarial attacks. It performed the multiple adversarial training method by utilizing various attacks. The results showed it improved detection accuracy by 16.48% on average across multiple models. [4] provided a new perspective to a defense in that DNNs should reject suspicious examples. [4] found that adversarial instances are usually designed to some common classes.

4 Fundamental Methods

In this section, we describe several of fundamentals studied and used. It starts with the introduction to the different threat models we study. Followed by the important type of attacks that are used in the robustness studies. And finally introduce several critical techniques concerning with the defense strategy we proposed.

4.1 Threat Model

Threat model is a process to identify possible threats and analyze how to provide a defense. For adversarial attacks, different attacks have different threat models, so not every model can be effectively attacked by any kind of attack. In Section 4.1, we will introduce all kinds of threat models for attacks and defense in more detail.

4.1.1 White-box

In case of a white-box threat model it means that attacks are able to understand the model and their defense methods. Thus, attack algorithms can generate adversarial examples by misguiding the classifier into a specific class. This called "targeted attack". For example, in Figure 1, the target is "gibbon" and the attacking object is "panda". Attacks in a white-box threat model can add perturbations by any methods. The attack will be regarded as "success" when the output of the model changes from the attacking object(panda) to the target(gibbon)[43]. A white-box threat model is the most difficult case to defend against because of the complete exposure of models and defense methods[15].

4.1.2 Black-box

A black-box threat model means that attacks have no information on the model and defense methods. Attacks can also obtain information from inputs and outputs from the model. In this case, attacks in black-box settings are still targeted[36]. Black-box threat model is the most common situation happens in the real world. It is not possible for attacks to grasp internal details of the defense system and model in most situations. Therefore, black-box model is the most challenging situation for the attack.

In case of a white-box threat model, attacks can clearly be designed after having a good knowledge of the model that is being attacked. Furthermore, it is possible to extend the attacks by using a so-called surrogate model of DNN model under attack[38].

4.1.3 Gray-box

A gray-box threat model means that the model information is exposed, but the defense method is unknown. A gray-box threat model is used in UNMASK system. The classification model in UNMASK is visible for attacks, while the detection and defense is not. Furthermore, attacks in the UNMASK system are non-targeted attacks which means perturbed outputs do not have to be classified as a certain class.

4.2 Adversarial Attacks

After the first adversarial attack method being proposed in [17], many researchers have begun to explore this field, and there have been many excellent attack algorithms been proposed. For more in-depth exploration of adversarial attacks, here we introduce two kinds of attack in detail gradient-based attacks and attacks towards object detectors. Gradient-based attacks are the strongest attacks for one-step models, such as classifiers, while Shapeshifter has a good performance on two-step object detectors such as Fast R-CNN[9]. Besides, Shapeshifter is also robust to real-world noise. In another words, adversarial examples generated by Shapeshifter can still maintain an excellent attack performance in physical world scenarios. It should be noted that not all attack methods can guarantee an attack effect. Please refer to Table 1 in Section 2.7 for the symbols used in the following introduction of the attack different methods.

4.2.1 PGD

Projected Gradient Descent (PGD) is extention of the Fast Gradient Sign Method (FGSM) as Equation 2 as proposed in [17]. Equation 2 is called fast attack since it does not involve any iteration but directly obtained by maximizing the loss function. PGD denoted by Equation3 generates an adversarial example X^{adv} by applying a fast attack iteratively with a small step ϵ. In this project, we have two kinds of attack strengths dependency on how many pixels will be change at each iteration.

$$X^{adv} = X + \epsilon sign\left(\nabla_X J\left(X, y\right)\right) \tag{2}$$

$$X_{t+1}^{adv} = X_t^{adv} + Clip_{X,\epsilon}\left\{X_t^{adv} + \epsilon sign\left(\nabla_X J\left(X_t^{adv}, y\right)\right)\right\} \tag{3}$$

Here, J stands for the cross-entropy loss function, where X represents the input data and y is the predicted class label. Each iteration generates a new adversarial image X^{adv} by the $sign$ and the gradient of the loss function. $Clip_{X,\epsilon}$ stands for a function that conducts the clipping of pixels of X with step size ϵ, which generates perturbations of PGD-L_∞ and PGD-L_2 as a result. In the UNMASK framework, PGD is one of the main attack methods used to measure the performance of detection and defense.

4.2.2 MI-FGSM

MI-FGSM adds the idea of momentum iteration to the FGSM method. Compared to PGD which iterates with gradients of the loss function $J\left(X, y\right)$ in order to achieve the maximum, MI-FGSM takes the momentum of the gradients of the loss function into account by accumulating the velocity factor over several iterations. The next gradient direction obtained by MI-FGSM can make the loss function $J\left(X, y\right)$ maximization problem jump out of a local optimum, thereby obtaining better attack results than PGD. In [11], MI-FGSM has been evaluated to perform better than first order attacks such as PGD in many threat models. MI-FGSM can successfully fool DNNs with defense method of adversarial training[46]. Adversarial training against MI-FGSM is also studied in our design of defense and detection towards UNMASK.

Detailed results for MI-FGSM attacking AT can be found in Section **??**. Here, we provide specific formulas for MI-FGSM in Equation4 and 5 as:

$$g^{(t+1)} = \mu \cdot g^{(t)} + \frac{\nabla_{X_t^{adv}} J\left(X_t^{adv}, y\right)}{\left\|\nabla_{X_t^{adv}} J\left(X_t^{adv}, y\right)\right\|_1} \tag{4}$$

$$X_{t+1}^{adv} = X_t^{adv} + Clip_{X,\epsilon}\left[\alpha \cdot sign\left(g^{(t+1)}\right)\right] \tag{5}$$

Here, g represents the gradient of loss function at the iterations, $g^{(0)} = 0$, $g^{(t+1)}$ is the accumulation of velocity vector, μ perform a certain decay, and ϵ stands for the strength of perturbation. Different from Equation 3, adversarial examples are generated using the sign of $g^{(t+1)}$.

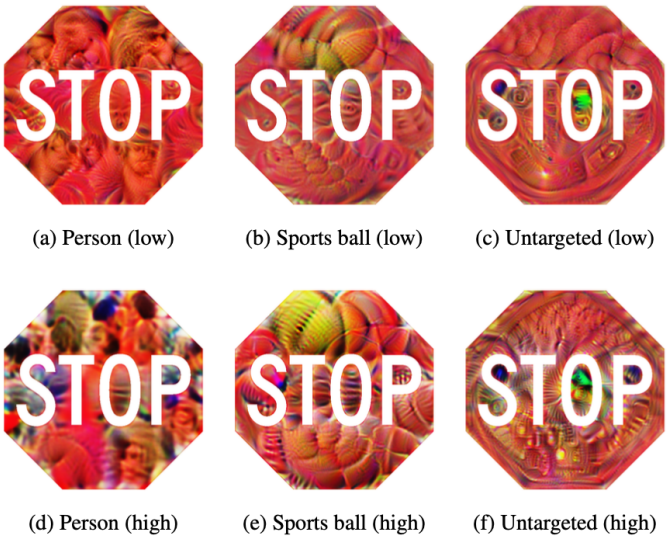
4.2.3 Shapeshifter

To explore the adversarial attacks more deeply and comprehensively, we also consider attack algorithms against object detectors. Attack algorithms for object detection have the following characteristics: (a) researches towards detectors are very rare and limited at present. Most existing attack algorithms are focused on classification. (b) attacks towards detectors are more difficult than classifiers, as object detection tasks actually consist of multiple classification tasks, and the attack algorithm should firstly determine the boundary box of multiple objects with different sizes. (c) attacks towards object detection tasks should be more robust. When we perform an object detection task in the real world, the camera captures images that have different resolutions, different angles and etc. Currently, most proposed attack algorithms for object detection are tested with images short-distances in a laboratory environment or obvious attacks visible to the naked eye, which is unsatisfactory and not applicable to the real world.

Shapeshifter is the first adversarial attack algorithm against object detectors that shows its robustness in real-world experiments[9]. Shapeshifter generates adversarial examples by adding changes to digital pixels that are unrecognizable to the naked eye. Further more, it can achieve both targeted and non-targeted attacks. The method of Shapeshifter is denoted in Equation 6. We extend the Expectation over Transformation(EOT)[5] from classification to object detection. EOT is a technique that demonstrate the real-world noise when generating adversarial examples, which enhance robustness.

$$\arg\min E_{x\sim X, t\sim T}\left[\frac{1}{m}\sum_{r_i \in rpn(M_t(x'))} L_{F_i}\left(M_t\left(x'\right), y'\right)\right] + c \cdot \left\|\tanh\left(x'\right) - x_o\right\|_2^2 \tag{6}$$

Here, L_{F_i} is the loss function of the object detector Fast R-CNN, i is the index of objects in a input instance, rpn(x) represents a set of object region proposals, $M_t\left(x, \tanh\left(x'\right)\right)$ is the EoT process adding x with transformation t to the background $\tanh\left(x'\right)$, $M_t\left(x, \tanh\left(x'\right)\right)$ is written as $M_t(x'$ in Equation 6 for simplicity, $\tanh\left(x'\right)$ is used to keep the result within the interval [-1,1] and c is a constant to keep the distance with the attacked image and the benign image.



(a) Person (low) (b) Sports ball (low) (c) Untargeted (low)

(d) Person (high) (e) Sports ball (high) (f) Untargeted (high)

Figure 4: Adversarial examples generated by Shapeshifter with "low" and "high" confidence(perturbation strength). Shapeshifter can perform both targeted attacks and non-target attacks.

We have reproduced Shapeshifter and obtained several attacked images. Shapeshifter can attack the object into a targeted class as Figure 4 shows. Among them, the image of high confidence is inevitably more obvious to the human. These adversarial examples can be printed out and placed in a real-world environment as attacks to road signs. It achieves good attack results according to [9]. Therefore, Shapeshifter is a very powerful state-of-art attack algorithm. Since the UNMASK framework can theoretically be applied to any DNN, the results of Shapeshifter attack algorithm provide us a reliable measurement to the detection and defense methods in the real world when unmask is expanded to application scenarios in the future. However, currently the dataset "unmask" involved in the existing UNMASK framework is completely manually designed and has multiple classes. It is not possible to focus on, for example, the road sign dataset like shapeshifter and it is also impossible to conduct the masking of each object in the dataset precisely.

4.3 Jaccard Similarity

Jaccard similarity is a statistical method generally used to calculate the similarity of different sets in machine learning. Jaccard similarity is obtained by dividing the cardinality of the intersection of two sets by the cardinality of the union of two sets, while Jaccard distance indicates the degree of dissimilarity between these two sets.

In this project, we choose Jaccard similarity as an important metric for adversary detection and defense in the UNMASK system. As shown in Section 5.1 shown, we can get a set of extracted features f_r from model K and get a set of expected features f_e according to the

expected class classified by model M. Then, the Jaccard similarity and Jaccard distance are calculated using Equation 7 which will be further used in the implement action of detection and defense of UNMASK.

$$d = 1 - JS(f_r, f_e) \tag{7}$$

4.4 Non-local Means

Non-local(NL) means is a denoising algorithm, which leverage the redundant pixels provided by the input data. Non-Local means filter can remove the noise without losing the features. It predicts the value of a certain pixels by comparing its neighborhood pixels with other pixels. Then, the average value of these pixels is the predicted pixel's value. Compared with other filters, for example, the local filter, the Non-local means filter take all possible pixels into consideration when predicting the value of a pixel. The Equation of NL means algorithm is as follows:

$$y_i = \frac{1}{\mathcal{C}(x)} \sum_{\forall j \in \mathcal{L}} f\left(x_i, x_j\right) \cdot x_j \tag{8}$$

Here $f(x_i, x_j)$ is used to calculate the similarity between x_i and x_j, and $\mathcal{C}(x)$ is used to normalize the pixels. The detailed calculation process of Non-local means is depicted in Figure 5.

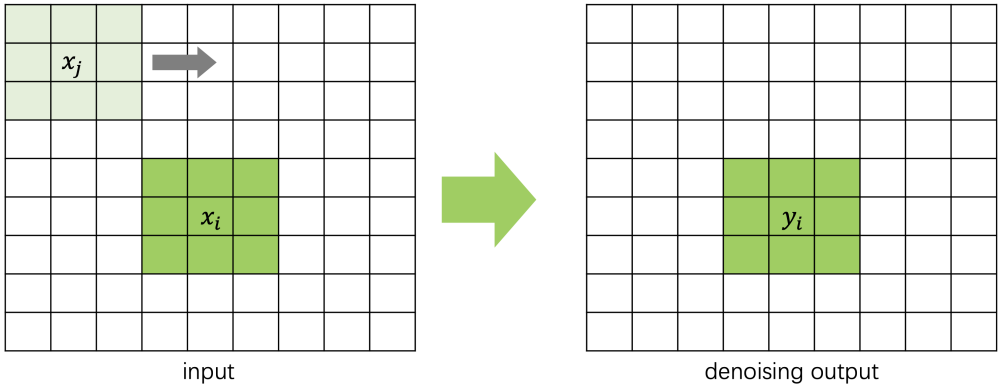


Figure 5: The detailed calculation process of Non-local means.

There are various kinds of forms of NL means, here we take the Gaussian form as an example which is also the form that we implemented in our method. In Gaussian embedded NL means, the weighting function is given by $f\left(x_i, x_j\right) = e^{\frac{1}{\sqrt{d}}\theta(x_i)^{\mathrm{T}}\phi(x_j)}$, and $\mathcal{C}(x)$ is $\mathcal{C} = \sum_{\forall j \in \mathcal{L}} f\left(x_i, x_j\right)$, where $\theta(x_i)$ and $\phi(x_j)$ are both obtained by a 1×1 convolution which represents the embedded x_i and x_j compared in Equation 8.

5 Methods

5.1 UNMASK Overview

The UNMASK framework is designed to protect the classification model and detect adversarial examples. In Figure 6, it is explained how UNMASK works with an example of "Bicycle" and "Bird". In the UNMASK system, the input image with class "Bicycle" has been attacked to be classified as "Bird". Therefore, for a vulnerable model, it will be misclassified as "Bird". Every image inputted will be analyzed by an Object Detector using semantic segmentation techniques to extract features. Here, we define the features that maintain the correct class even after being attacked by adversarial perturbations as robust features. When the attacked image was segmented by the object detector, small parts of this image(robust features) still remain useful although the whole image is misclassified. For example, the robust features of "Bicycle" are saddle, frame, handlebar, wheel, and pedal, while the robust features of "Bird" are expected to be head, beak, legs, etc. After comparing the extractions and the features that expected to have, the UNMASK system will give an answer to whether the input image is attacked, which is the **detection** method of this framework. In addition, the system compares extracted features with expected features of various classes, and selects the class label with the highest similarity as the final output of the UNMASK, which is regarded as the **defense** method.

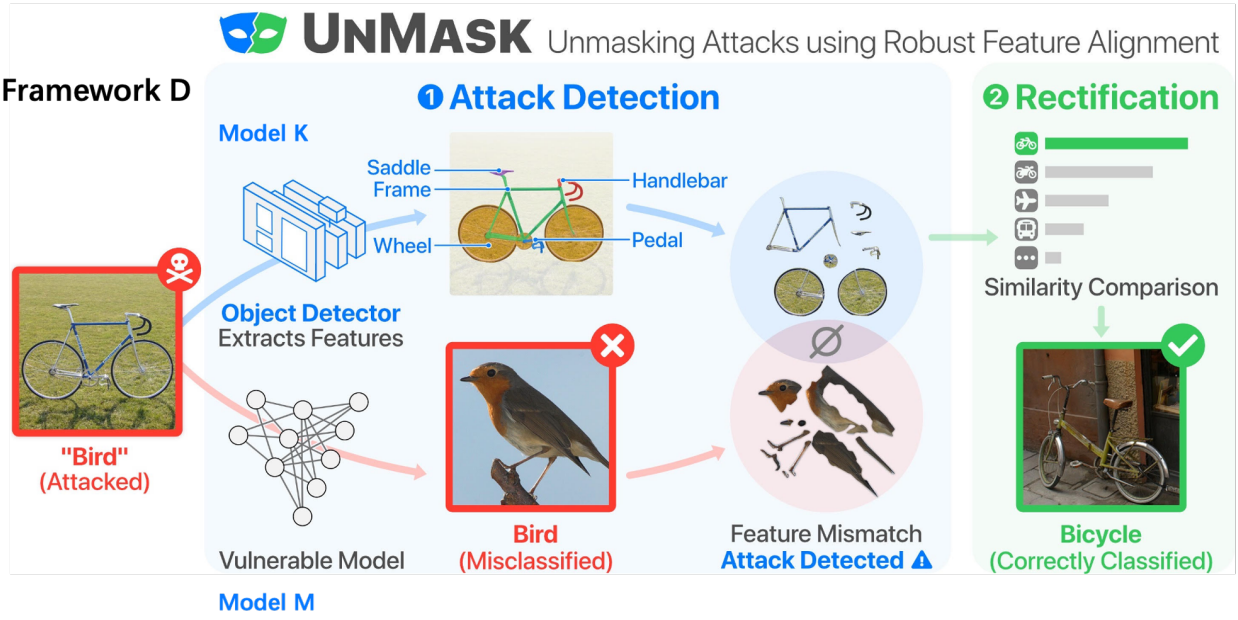


Figure 6: An overview of UNMASK framework. [3]

For better explanation, the object detector model in the upper part of Figure 6 is referred to

[3]source: https://github.com/safreita1/unmask/blob/master/images/unmask.jpg

model K in the following sections, the vulnerable model in the bottom is referred to model M, and the UNMASK framework is referred to framework D. Here, framework D can be regarded as a gray-box threat model in which model M is visible to attack methods while defense strategy is not visible. In the whole pipeline of UNMASK framework, there are three main steps to work through:

1. Extracting robust features: For each input image of input space X, model K extracts small parts of it as robust features.

2. Detection: For each image of input space X, we define adversarial example as "-1" and benign as "+1". Hence, detection is a binary classifier which maps robust features F to +1, -1.

3. Defense: leveraging robust features F extracted by model K to predict its class label y instead of original input image x.

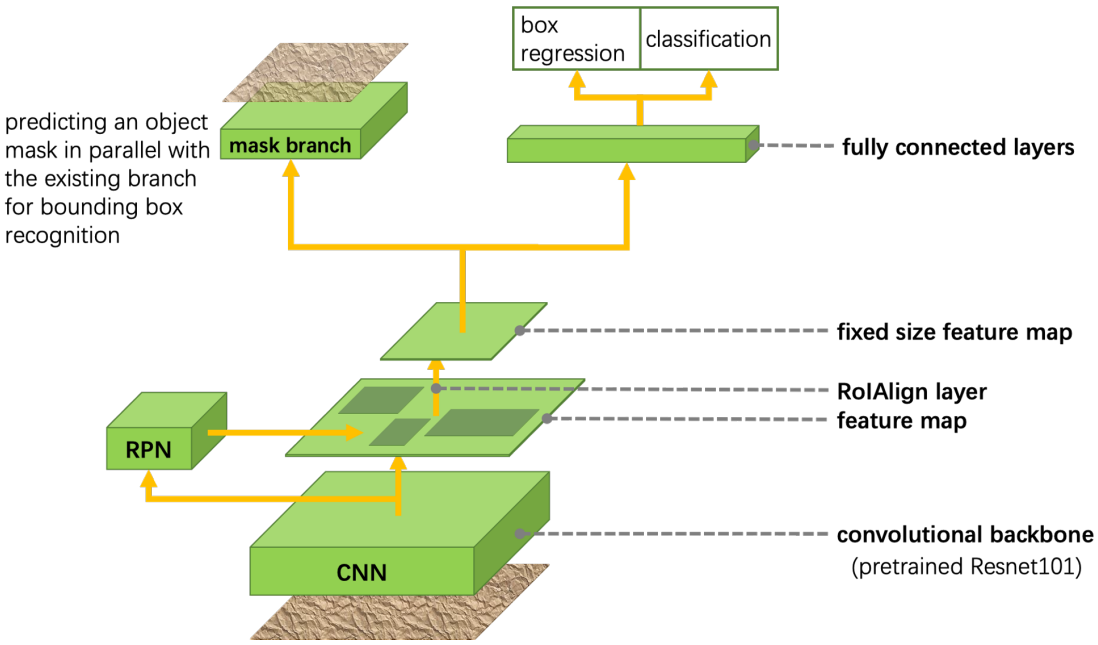


Figure 7: The architecture of Model K.

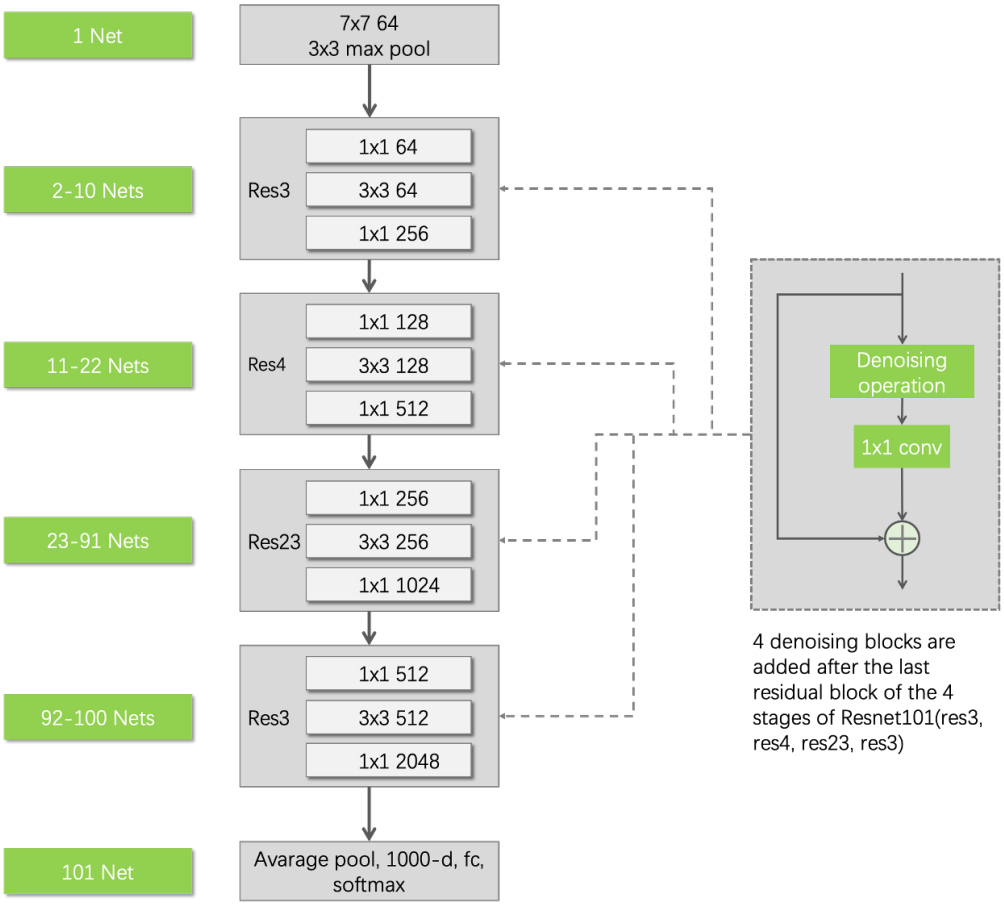


Figure 8: The architecture of model K.

5.2 Extracting Robust Features

To better utilize the features in the processing of model, here we divide features of an image into the following three types: a. p-useful features, b. robust features, c. useful but not robust features[22]. For example, the final class label is determined by p-useful features for classification task. As shown in Equation 9, for $p > 0$, a p-useful feature means the true predicted label exists with certain expectation.

$$E_{(X,y)\sim D}[y \cdot f(X)] \geq p \tag{9}$$

In the context of adversarial learning, p-useful features can be divided into two classes according to whether it is robust against adversarial attacks. As shown in Equation 10, for $\gamma > 0$, robust features are expected to remain the true predicted label when a perturbation δ was added, these features are defined as γ-robust features. Useful but not robust features are also very decisive features for classifiers. To some extent, these features satisfy Equation 9 but are not necessarily strongly related to the final decision of the classifier. However, useful but not robust features do not satisfy Equation 10 and are very unstable in the face of

perturbations which cause them to lose their original values and affect the final class label under adversarial attacks.

$$E_{(X,y)\sim D}\left[\inf_{\delta\in S} y \cdot f(X+\delta)\right] \geq \gamma \tag{10}$$

In the classification tasks, we believe the extraction of robust features in an image can be leveraged to defense against adversarial attacks. Therefore, before that, model K should extract the robust features of each class appearing in the dataset of model M, which is the main function of model K. In our method, we use Mask R-CNN as model K since it is powerful in learning and generating segmentation masks, and use the Pascal-Part dataset for training since it provides masks for parts of an object. The segmentation masks in the Pascal-Part dataset can be regarded as a kind of robust features that humans can perceive. In the training of model K, we only use these robust features to train model K so that model K can accurately extract robust features that are visible to the human eye. Therefore, in fact, for Mask R-CNN architecture, the training dataset inputted is no longer a whole object, but consists of the parts of each object. As Equation 11 shows, model K is trained on $\hat{D}_R$ which is a set of robust features, W stands for the weights of model K, L(X, y) denotes the loss function. Using Equation 11, we obtain the robust features extraction model K. In the UNMASK system, we use the information extracted by model K to achieve defense and detection.

$$\min_{W}\left[E_{(X,y)\sim\hat{D}_R} L(X,y)\right] \tag{11}$$

5.3 Detection and Defense

The UNMASK system provides the detection and defense of adversarial attacks. As shown in Figure 6, the unmask system has two models to process the input image in parallel. The trained extraction model K decomposes the object from the input image and obtains extracted parts of the object as described in Section 5.2 to obtain a set of robust features f_r. At the same time, model M, as a standard model, processes the input image and gets a predicted class label y. After learning all the above information, our UNMASK pipeline is able to detect and defend against multiple adversarial attacks.

Detection is to conduct depending on the Jaccard similarity(JS) between extracted features f_r from model K and the expected features f_e from model M. If JS exceeds a certain threshold, it means that the input image is adversarial and the output of model M has been attacked to the wrong class. The main idea of detection is to calculate JS between f_r and each class in the known class attribute matrix in turn, and output the class with the biggest value of JS. Details of Detection and Defense is elucidated in Algorithm 1.

Algorithm 1 UNMASK Algorithm.

1: **Input**: input image X, standard classification model M, feature extraction model K, class attribute matrix V, Jaccard Similarity(JS) threshold t, the set of possible class in dataset C.
2: **Output**: binary detection result of input image z, predicted class label of UNMASK system p.
3: **model M**:
$y = \mathrm{M}(X)$
4: **model K**:
$\mathrm{K} = \min_W \left[E_{(X,y)\sim\hat{D}_R} L(X, y)\right]$
$f_r = \mathrm{K}(\mathrm{X})$
$f_e = \mathrm{V}(y)$
5: **Detection:**
$s = JS\left(f_r, f_e\right)$
$d = 1 - s$
$z = \begin{cases} +1(benign), & if\ d < t \\ -1(adversarial), & if\ d \geq t \end{cases}$
6: **Defense**
$p = \begin{cases} y, & if\ z = +1 \\ argmin_{c\in C} JS\left(f_r, V[c]\right), & if\ z = -1 \end{cases}$
7: **return** z, p;

5.4 Data Augmentation

Data augmentation is performed by slightly modifying the original data or create new data from original data[2]. In the process of training the model, data augmentation can help to avoid overfitting and make the model perform better[42]. In the field of image classification, data augmentation is often implemented by two methods, the first one is by applying transformations such as rotation, flip, scale, and crop, and the cone method is by creating new synthetic images often mainly implemented by advanced machine learning algorithms like a Generative adversarial network (GAN).

As introduced in Section 5.1 and Section 5.2, there are two models to be trained in the UNMASK framework. For model K, to obtain the semantic segmentation ability, all objects in the dataset have segmentation masks of their small parts compared with object detection tasks. To improve the performance of model K, we perform a basic enhancement process by flipping 50% of the trainging images horizontally as Equation 12 shows:

$$Flip_{f_r} = Fliplr(0.5) * E_{(X,y)\sim D}\left[\inf_{\delta\in S} y \cdot f(X + \delta)\right] \tag{12}$$

For model M, the training image can be identified as robust features and non-robust features by model K. If only robust features are used for training model M, the model will most probably become more robust since the classification of robust features is hardly affected by adversarial attacks. However, the lack of other non-robust features will probably lead to a lower performance on the classification tasks. Therefore, the idea of augmentation as given in Equation 13 is to maintain the highest classification accuracy by training on training images as a whole and to achieve more robustness by augmenting robust features for training. Here, a,b,c are constants, $Flip_{f_r}$ represents the augmented f_r, f_{unr} represents useful but not robust features, these features are used as the input of X of model M by a certain proportion.

$$y = M(X); X = a * f_r + b * Flip_{f_r} + c * f_{unr} \tag{13}$$

However, data augmentation to improves robustness of a DNN only works for some specific situations. For example, data augmentation for the UNMASK framework in Section 2 in UNMASK framework. In the experiments, we observe that it takes more than 15 hours for the training of two models without data augmentation on the hardware described in Section 6. The training dataset of model K consists of about 7,000 images, and the training dataset of model M has more than 40,000 images which is too time consuming to conduct data augmentation. Therefore, the final data augmentation is performed for training model K only.

5.5 Architecture of Model M

In this Section, we propose our new architecture of Model M as our contributions, which has not been studied in the original UNMASK. The robustness of the UNMASK framework can also be achieved by improving the architecture of the DNN model[32]. Therefore, we explore different architectures of the DNN model from different aspects and observe its benefits in UNMASK. Our goal is to achieve better defenses by building a more powerful Model M. Specifically, we attempt to achieve improvements from the following methods: a.) replacing model M with other classifier models supported by UNMASK, b.) using better optimizers, c.) adding feature denoising blocks[51] to the DNN models to protect it from multiple attacks.

5.5.1 Classification Model

UNMASK is designed as a system composed of various modules. As Figure 6 shows, the standard model can actually be any kind of classifier model. In original version of UNMASK, model M is consisted of Densenet121. To select a model that is more robust, we explore better classifier models from the TORCHVISION.MODELS[3] package and perform preliminary experiments using different models. By comparing the performance(Acc@1 and Acc@5) of the different models in the classification tasks, we choose Resnet models as our ideal model. The detailed models involve are descrbed in Section 6.2.

5.5.2 Optimizer

An optimizer can improve the update strategy of weights and parameters during the training of model, so that the loss function can reach a better maximum or minimum. Regarding the

choice of optimizer, there is currently no theoretical consensus or rules. Using an optimizer that can perform adaptive learning in a DNN model should improve its robustness[47]. Therefore, we consider using different optimizers for training model M to find the best setting of optimizer and classification model for UNMASK. On this basis, we implement one further method by adding blocks to the architecture of model M. In this section, we mainly consider two optimizers: SGD and Adam.

SGD updates weights and parameters in each iteration. Frequent updates avoid being trapped in the local optimum and discover a new local optimum or even global optimum. SGD optimization algorithm converges fast but may be unstable. Adam stands for Adaptive Moment Estimation which can store the momentum of each parameter independently. The adaptive learning rate is assigned to each parameter so that parameters are controlled from a range in each iteration to make the optimization convergence more stable. When designing model M, we choose the best optimizer for our models and datasets to make the model quickly converge and learn precisely. However, it is worth noting that when adjusting UNMASK, we only make structural adjustments, no specific parameter adjustments.

5.5.3 Feature Denosing Block

Adding external blocks to the DNN model is an effective method to improve its robustness[47] in a defense method. The UNMASK[15] consists of standard classification model M. Therefore, here we propose a new architecture by adding feature denoising blocks to model M. Feature denoising blocks have been proven to be effective in an adversarial learning defense strategy in [51]. However, whether feature denoising block is also useful on improving robustness by robust feature aligning strategy is what we would like to verify. For noises caused by adversarial perturbations, it is hard to quantify between different DNNs. Besides, adding feature denoising blocks into UNMASK system will also have influences on features by modifying size and distribution. In spite of the above potential deteriorations, we still believe that feature denoising blocks as a novel architecture will improve the performance of UNMASK against adversarial attacks.

From [51], adversarial examples are found to have some noises in feature maps extracted from DNNs. Through a series of experiments, we observe that the adversarial perturbations cause some strange activations in feature maps of attacked images compared with feature maps of benign images. This is why the adversarial examples are eventually misclassified in the classifiers. Furthermore, the denoising filter is applied on the feature map to remove strange activations. This attempt succeeds in returning the feature map to what it was before being attacked so that activations in the feature map are concentrated on features that are used to predict the class label.

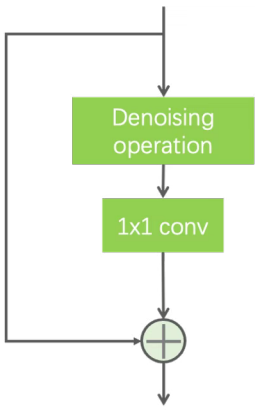


Figure 9: The architecture of a feature denoising block.

Motivated by the above findings, we add the feature denoising blocks to model M in the UNMASK system to improve its robustness against perturbations. Theoretically, feature denoising blocks are available to be added to any feature map layer in Resnet101[51]. In Figure 9, the complete architecture of a single feature denoising block consisting of a 1x1 convolutional layer and a denosing filter is depicted. In the denoising operation block, the feature map can be processed by any filter. According to empirical observations in [48], Non-local means filter is good at denoising processing for the image classification tasks. Therefore, here we choose a Non-local Means filter as the denoising method as described in Section 4.4. After the denoising operation, the 1x1 convolutional layer processes the feature map to balance the features remained and the noises removed. Then, the output signal from the feature denoising block is added together with the original signal. We add 4 denoising blocks into the end of each stage of Resnet101 as shown in Figure 8 since Resnets have 4 convolutional stages in architecture.

6 Experiments

To evaluate the effectivity of our proposed improvements and instances of the UNMASK architecture, an extensive series of experiments were conducted using several datasets and training scenarios. All codes of this project are implemented in Python 3.6. Many open-source libraries are imported, for specific package and version information we refer to the File environment.yml in https://github.com/Yuxin33/unmask. All experiments are run on Ubuntu 16.04 LTS 64-bit. The machine has a Geforce Titan X 12GB GPU, Intel Core i7-5820K 3.3 GHz 12 cores CPU and 64GB RAM. Detailed steps of how to run the project are given in the File README.md on Github.

6.1 Unmask Dataset

We manually generate the unmask dataset for the training and testing of the UNMASK system. Since the UNMASK system is mainly composed of two models in which model M processes the information extracted from model K, we carefully design the unmask dataset for the training, validation, and test of each model. For model K, the PASCAL-Part dataset which contains masks for parts of objects is used to train and test. For model M, PASCAL VOC 2010 and parts of the ImageNet dataset are selected as training dataset, and Flickr dataset are selected for evaluation.

In Table 2, there are 44 classes in the PASCAL-Part dataset for the training of extraction model K. We design 4 subdatasets of 3 or 5 classes each for model M, where CS3 means that there are three classes in this subset, and cs5 means there are 5 classes. "a" and "b" in the subdataset means there exists a different overlap of features. We use "a" to represent a lower overlap rate and use "b" to represent higher overlap rate. In Section 6.4, we compare the results in different datasets to analyze the performance of UNMASK system under different numbers of classes and different overlap rate. Regarding the validation dataset and the test dataset of model M, images are preprocessed by a perceptual hashing algorithm to avoid duplicating with the training dataset. Therefore, the unmask dataset attempts to provide more images so that models in UNMASK system can be better trained and evaluated.

Table 2: Number of images in Unmask dataset for model K and model M.

Setup		PASCAL-Part			VOC+ Net	Flickr	
Model	Classes	Train	Val	Test	Train	Val	Test
K	44	7,457	930	936	-	-	-
M	CS3a	-	-	-	7,780	1,099	2,351
	CS3b	-	-	-	9,599	1,399	2,867
	CS5a	-	-	-	11,639	1,477	3,179
	CS5b	-	-	-	13,011	1,928	4,129

6.2 Small Crafted Dataset

As introduced in Section 5.5, we select different configurations of architecture of the DNN model. It takes about 15 hours to run one experiment on the UNMASK dataset regardless of the defense methods. Considering practical feasibility, we manually created a separate small dataset to find the first-step best configuration with less time. The number of images in each subdataset of the small crafted dataset is listed in Table 3, sizes ranging from about 1/10 to 1/15 of the original unmask dataset. It only takes about 2 hours to run one experiment on the small crafted dataset which greatly reduces the time required for experiments and helps to find a initial configuration quickly.

Table 3: Number of images in the small crafted dataset.

Class	Train	Test	Val
Dog	150	50	20
Bird	150	70	30

We run several experiments with different configurations on the small crafted dataset and list the performance of defense against 4 strong attacks at 2 strength levels as Table 4. Here, we only consider the performance of defense since it is the core function of UNMASK. However, there are certain differences in the defense performance of different configurations against different attacks. Therefore, it is difficult to select the obvious best configuration from Table 4.

Table 4: The performance of UNMASK under different configurations of DNN models and optimizers.

Configuration (Model+optimizer)	PGD-L_∞ ϵ=8	PGD-L2 ϵ=300	PGD-L_∞ ϵ=16	PGD-L2 ϵ=600	MIA-L_∞ ϵ=8	MIA-L2 ϵ=300	MIA-L_∞ ϵ=16	MIA-L2 ϵ=600
Densenet121 SGD	78.40	88.40	68.00	82.00	74.40	86.00	61.60	82.00
Resnet50 SGD	80.80	85.20	70.00	84.00	77.60	86.40	62.80	86.40
Resnet101 SGD	81.60	87.20	70.00	88.40	77.20	89.20	67.60	84.80
Resnet152 SGD	80.00	86.80	67.20	84.80	74.00	86.80	62.00	84.40
Resnet101 Adam	78.80	87.60	68.00	85.20	76.40	88.80	52.00	81.60
Resnet152 Adam	78.80	88.40	72.00	84.80	74.40	86.40	62.40	81.60

To compare the performance under different configurations more intuitively, we calculate the average classification accuracy of UNMASK under different attack methods(PGD and MI-FGSM) and the total average defense accuracy in Table 5. It is obvious that the best accuracy against PDG is **81.8%**, the best accuracy against MI-FGSM is **79.7%**, and the best total accuracy is **80.8%**. It is obvious that all the best results are come from the configuration of Resnet101 and SGD. Therefore, the configuration(Resnet101 and SGD) was taken as the basic model for following experiments.

Table 5: The average classification accuracy of different configurations.

Model+Optimizer	PGD Acc	MI-FGSM Acc	Average Acc
Baseline	79.2	76.0	77.6
Resnet50+SGD	80.0	78.3	79.2
Resnet101+SGD	**81.8**	**79.7**	**80.8**
Resnet152+SGD	79.7	76.8	78.3
Resnet101+Adam	79.9	74.7	77.3
Resnet152+Adam	81.0	76.2	78.6

6.3 FD Blocks in Adversarial Training

To evaluate the performance of Feature Denoising block introduced in Section 5.5.3, we compare the robustness of the standard Resnet and the modified Resnet by adding a denoising block towards adversarial training. As first experiments, one feature denoising block of non-local means filter is added to the third layer of Resnet101. PGD-L_∞ with ϵ=10 is used as attack method, and Adversarial Trainging as defense method. The dataset we use here is CIFAR-10[1]. The learning rate is set to 0.005 and decayed 10% on every 30 epochs, with a batch size of 60. We use Top1 accuracy to evalaute these two Resnet model.

Table 6: Resnet101 with One Feature Denoising block compared with standard Resnet101 on Adversarial training.

Settings	Benchmark(top1)	AT- benign test	AT- adv test
Resnet101	95.400	81.490	59.210
Modified Resnet101	95.510	83.120	70.100

From Table 6, we can see that the modified Resnet is more robust than the standard Resnet in Adversarial training. Here benchmark means that the model is trained and tested using benign images in CIFAR-10. Although the classification accuracy decreases in adversarial training compared to the benchmark on benign tests, the results of adversarial tests show that feature denoising blocks are effective on Resnet101. Therefore, we determine modified Resnet 101 by adding feature denoising blocks is a more robust architecture.

6.4 FD blocks in UNMASK

After we have verified the results of FD blocks in adversarial training, we would like to further see its effects on robust feature aligning theory. In Section 5.5.3, we introduce the new architecture of Model M by adding four feature denoising blocks into Resnet101. The new architecture is described as "Ours" in the following parts, and the UNMASK system consists of Densenet121 and the standard architecture and is described as "Baseline". To compare these two systems better, we evaluate the defense system UNMASK by 4 widely used attacks with 2 kinds of strength. The results is shown in Table 7.

Table 7: The performance of our improved UNMASK compared to the baseline UNMASK.

Model	Dataset	No Defense (Average)	PGD-L_∞ (ϵ=8)	PGD-L2 (ϵ=8)	MIA-L_∞ (ϵ=8)	MIA-L2 (ϵ=8)	PGD-L_∞ (ϵ=16)	PGD-L2 (ϵ=16)	MIA-L_∞ (ϵ=16)	MIA-L2 (ϵ=16)
Ours	CS5b	11.50	83.97	90.80	83.60	90.26	72.66	90.14	72.61	89.61
Ours	CS5a	10.77	87.95	92.70	88.68	92.04	78.14	92.42	79.18	91.26
Ours	CS3b	29.89	85.42	90.97	85.11	90.51	72.86	90.97	72.62	90.34
Ours	CS3a	21.16	91.37	93.87	92.09	93.45	85.71	94.13	85.28	93.41
Baseline	CS5b	5.84	78.54	88.71	74.84	88.50	62.02	85.08	57.57	84.48
Baseline	CS5a	5.43	81.88	91.16	77.70	90.63	65.65	88.49	58.45	87.20
Baseline	CS3b	6.30	81.55	89.29	75.65	88.98	67.39	85.07	62.19	84.97
Baseline	CS3a	7.23	85.88	92.94	83.54	92.39	73.84	91.15	69.29	90.09

From Table 7, we can clearly see the defense function of UNMASK. The third column gives the average classification accuracy against 8 attacks of the two studied models (Ours and Baseline) with no defense algorithm. The results of two studied models against each attack on different datasets are also listed. The UNMASK system shows excellent defense performance compared to the no defense standard classifier. Ours model achieved higher classification accuracy than the Baseline model. Revealing that Resnet101 with feature denoising blocks in UNMASK system provides a more robust architecture compared with the baseline. When we compare the accuracy of these two models in the third column, the new architecture itself also shows a stronger robustness compared to the standard classifier. In terms of the dataset, as introduced in Section 2, there are 4 datasets with different classes and different overlap rates. The resulting of accuracies shows that the UNMASK system performs better on dataset with less classes and lower feature overlap rate.

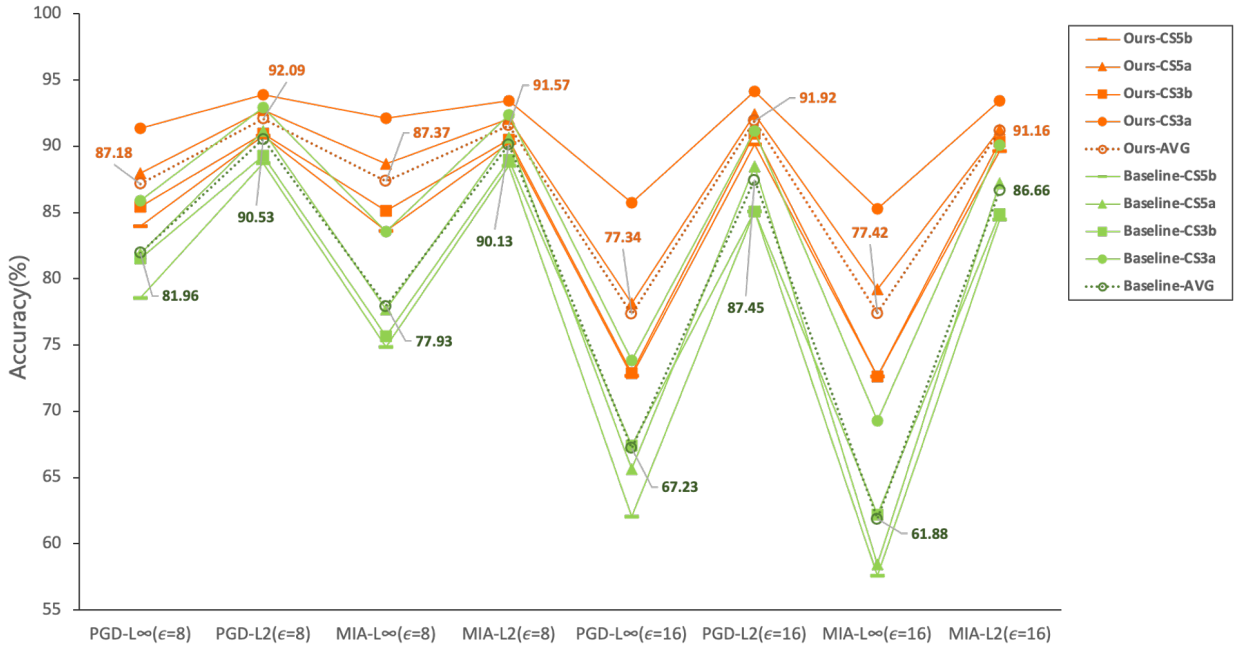


Figure 10: The performance of our improved UNMASK compared to the baseline UNMASK.

Analyzing from specific data perspective in Figure 10, the average accuracy of ours model in No defense state over 4 datasets is 18.33%, the average accuracy of "Ours" in defense state against 8 different attacks is **87.01%**, the average accuracy of Baseline model is 80.48%. Therefore, Ours model has improve the defense of classification task by **68.68%**, and improve the defense performance by **6.53%** over all attacks compared with Baseline model. Especially, as we can see from Figure 10, the Baseline UNMASK performs relatively poorly against MIA-L_∞ with strong perturbations which is only 61.88% on average in 4 datasets. Ours model fixes this problem of UNMASK by improving the accuracy to **77.42%** which is a acceptable and satisfied result that close to the average performance of Baseline model.

7 Conclusions

In this paper, we conducted a comprehensive and in-depth exploration of the application of adversarial learning for classification tasks which involves adversarial attacks, adversarial learning, feature knowledge aligning, feature denoising block, and Resnet101 architecture. After implementing theoretical methods and designing experiments, we evaluate the UNMASK system's defense performance on the unmask dataset we manually designed. In general, for the robustness of classifiers, we have the following conclusions:

1. The UNMASK system utilizing robust features knowledge aligning shows significant effects against adversarial attacks. As a result, the new architecture of UNMASK improves the accuracy of the classification task from 18.33%(no defense) to 87.01%. In our experiments, the UNMASK defense method performs better than adversarial training under 8 attacks on unmask dataset.

2. We propose a new architecture of the UNMASK consists of Resnet101, SGD optimizer and 4 feature denoising blocks. The results show that "Ours" model improves the classification accuracy by 6.53% compared with Baseline model. Furthermore, we verify that feature denoising blocks can provide protection to DNNs in the UNMASK defense method apart from adversarial training.

3. Our new architecture can provide a more stable and better protection than the baseline. For the worst performance of baseline UNMASK against MI-FGSM attacks, "Ours" model increases the classification accuracy by 15.54%.

4. By comparing the performance of UNMASK on 4 datasets, the UNMASK system achieves better defense effects on dataset with less classes or low feature overlap rate.

References

[1] The cifar-10 dataset. `https://www.cs.toronto.edu/~kriz/cifar.html`, accessed on 2020-04-08.

[2] Data augmentation. `https://en.wikipedia.org/wiki/Data_augmentation`, accessed on 2020-11-20.

[3] Torchvision.models. `https://pytorch.org/vision/stable/models.html`, accessed on 2021-02-13.

[4] Mahdieh Abbasi and Christian Gagné. Robustness to adversarial examples through an ensemble of specialists. *in Neural and Evolutionary Computing*, 2017.

[5] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[6] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *in Machine Learning*, 2017.

[7] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *in ICLR*, 2017.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[9] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2018.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[12] Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *in CVPR*, 2017.

[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[14] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *in ICML*, 2017.

[15] Scott Freitas, Shang-Tse Chen, Zijie Wang, and Duen Horng Chau. Unmask: Adversarial detection and defense through robust feature alignment. *in IEEE Big Data*, 2020.

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *in ICLR*, 2015.

[18] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, M. Backes, and P. Mcdaniel. On the (statistical) detection of adversarial examples. *ArXiv*, abs/1702.06280, 2017.

[19] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[21] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *in Machine Learning*, 2015.

[22] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *in NeurIPS*, 2019.

[23] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[24] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *2018 ieee security and privacy workshops (spw)*, pages 36–42. IEEE, 2018.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[26] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *in CVPR*, 2016.

[27] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR 2011*, pages 3361–3368. IEEE, 2011.

[28] Bo Li and Yevgeniy Vorobeychik. Scalable optimization of randomized operational decisions in adversarial classification settings. In *Artificial Intelligence and Statistics*, pages 599–607, 2015.

[29] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772, 2017.

[30] Jiajun Lu, Hussein Sibai, and Evan Fabry. Adversarial examples that fool detectors. *in CVPR*, 2018.

[31] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *in CVPR*, 2017.

[32] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[33] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 135–147, 2017.

[34] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *in ICLR*, 2017.

[35] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[36] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Exploring the Space of Black-box Attacks on Deep Neural Networks. *ArXiv*, page arXiv:1712.09491, December 2017.

[37] Tianyu Pang, Chao Du, and J. Zhu. Robust deep learning via reverse cross-entropy training and thresholding test. *ArXiv*, abs/1706.00633, 2017.

[38] Nicolas Papernot, P. Mcdaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv*, abs/1605.07277, 2016.

[39] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[40] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[41] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.

[42] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[43] Lu Sun, Mingtian Tan, and Zhe Zhou. A survey of practical adversarial example attacks. *Cybersecurity*, 1(1):9, 2018.

[44] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *in CVPR*, 2013.

[46] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *in ICLR*, 2018.

[47] Kedi WANG and Ping YI. A survey on model robustness under adversarial example. *Journal of Cyber Security*, 5(2):13–22, 2020.

[48] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[49] R. Wiyatno, Anqi Xu, Ousmane Amadou Dia, and A. D. Berker. Adversarial examples in modern machine learning: A review. *ArXiv*, abs/1911.05268, 2019.

[50] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017.

[51] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[52] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *in Proceedings of Australasian Conference on Robotics and Automation*, 2015.

[53] Haichao Zhang and Jianyu Wang. Towards adversarially robust object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 421–430, 2019.