USING RANDOMNESS TO DEFEND AGAINST ADVERSARIAL EXAMPLES IN COMPUTER VISION

by

Huangyi Ge

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

December 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Ninghui Li, Chair

School of Computer Science

Dr. Elisa Bertino

School of Computer Science

Dr. Xiangyu Zhang

School of Computer Science

Dr. Zeynel B. Celik

School of Computer Science

Approved by:

Dr. Kihong Park

ACKNOWLEDGMENTS

I would like to thank the following people who have helped me to complete the dissertation:

My advisor Dr. Ninghui Li guided me how to do the research. The most important thing I learned from his was how to present the idea in the paper. Whenever he edited my writing, I felt I learned new things. In addition, he funded the GPU cluster which helped me so much to work on neural network research. Without him, I would not have made it through my Ph.D. degree.

Dr. Omar Chowdhury was the post doctor in my lab. Currently, he is an Associate Professor of CS at Stony BrookUniversity. He work with me during my first 2 year Ph.D. study and published 4 joint papers. He is my mentor, guided me not only doing the research but also how to become a mature researcher. Without him, I would not have passed my Ph.D. qualify exam. He is also like my father, shared with me all his experience, and always wishes me to grow and become success.

Dr. Sze Yiu Chau was my lab mate and is also one of my best friends. We had 6 joint works including 3 piece of works for this dissertation. We enjoyed discussing research ideas. Upon now, I still remember the moment that we discovered the MoNet. Both of us were amazed by the idea of using AES to pre-process the image and then feeding it to the neural network. We spent the whole winter break trying different approaches. Eventually, we found the experimental results were promising. Without him, I would not have come up with the second half of the dissertation.

I also would like to thank Purdue University's Department of Computer Science and Graduate School helped fund dissertation development.

Finally, I would to thank all my colleagues who had joint works with me: Jeremiah Blocki, Sze Yiu Chau, Jing Chen, Isis Chong, Omar Chowdhury, Sonia Fahmy, Victor E Gonsalves, Victor Gonsalves, Endadul Hoque, Aniket Kate, Huian Li, Ninghui Li, Hemanta K Maji, Scott R Moore, Cristina Nita-Rotaru, Robert W Proctor, Bruno Ribeiro, Tianhao Wang, Aiping Xiong, Weining Yang, Qi Zhong, Wanling Zou, Xukai Zou (Alphabetize by Last Name). Without you, I would not have made my Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Computer vision applications such as image classification and object detection often suffer from adversarial examples. For example, adding a small amount of noise to input images can trick the model into misclassification. Over the years, many defense mechanisms have been proposed, and different researchers have made seemingly contradictory claims on their effectiveness. This dissertation first presents an analysis of possible adversarial models and proposes an evaluation framework for comparing different more powerful and realistic adversary strategies. Then, this dissertation proposes two randomness-based defense mechanisms Random Spiking (RS) and MoNet to improve the robustness of image classifiers. Random Spiking generalizes dropout and introduces random noises in the training process in a controlled manner. MoNet uses the combination of secret randomness and Floyd-Steinberg dithering. Specifically, input images are first processed using Floyd-Steinberg dithering to reduce their color depth, and then the pixels are encrypted using the AES block cipher under a secret, random key. Evaluations under our proposed framework suggest RS and MoNet deliver better protection against adversarial examples than many existing schemes. Notably, MoNet significantly improves the resilience against transferability of adversarial examples, at the cost of a small drop in prediction accuracy. Furthermore, we extend the usage of MoNet to the object detection network and use it to align with model ensemble strategies (Affirmative and WBF (weighted fusion boxes)) and Test Time Augmentation (TTA). We call such a strategy 3MIX. Evaluations found that 3Mix can significantly improve the mean average precision (mAP) on both benign inputs and adversarial examples. In addition, 3Mix is a lightweight approach to migrate the adversarial examples without training new models.

1. INTRODUCTION

Modern society increasingly relies on classification models trained by machine learning (ML) techniques. Many ML techniques, however, were designed under the implicit assumption that both the training and testing data follow the same static (although possibly unknown) distribution. In the presence of intelligent and resourceful adversaries, this assumption no longer holds. Such an adversary can deliberately manipulate a test instance, and cause the trained models to behave unexpectedly. For example, it is found that existing image classifiers based on Convolutional Neural Networks (CNN) are highly vulnerable to adversarial examples [1, 2]. Often times, by modifying an image in a way that is barely noticeable by humans, the classifier will confidently classify it as something else. This phenomenon also exists for classifiers that do not use neural networks, and has been called "optical illusions for machines". Understanding why adversarial examples work and how to defend against them is becoming increasingly important, as machine learning techniques are a key component of transformative technologies such as autonomous cars, unmanned aerial vehicles, and so on.

Many approaches have been proposed to help defend against adversarial examples. Goodfellow et al. [1] proposed adversarial training, in which one trains a neural network using both the original training dataset and the newly generated adversarial examples. In region-based classification [3], one aggregates predictions on multiple perturbed versions of an input instance to make the final prediction. Some approaches attempt to train additional neural network models to identify and reject adversarial examples [4, 5].

We point out that since the adversary can choose instances and shift the test distribution *after* a model is trained, adversary examples exist so long as ML models differ from human perception on some instances. (These instances can be used as adversarial examples.) Thus adversarial examples are unlikely to be completely eliminated. What we can do is to reduce the number of such instances by training ML models that better match human perceptions, and by making it more difficult for the attacker to find adversarial examples.

While the research community has seen a proliferation in proposals of defense mechanisms, conducting a thorough evaluation and a fair head-to-head comparison of different mechanisms remains challenging. In Chapter 3, we analyze possible adversarial models,

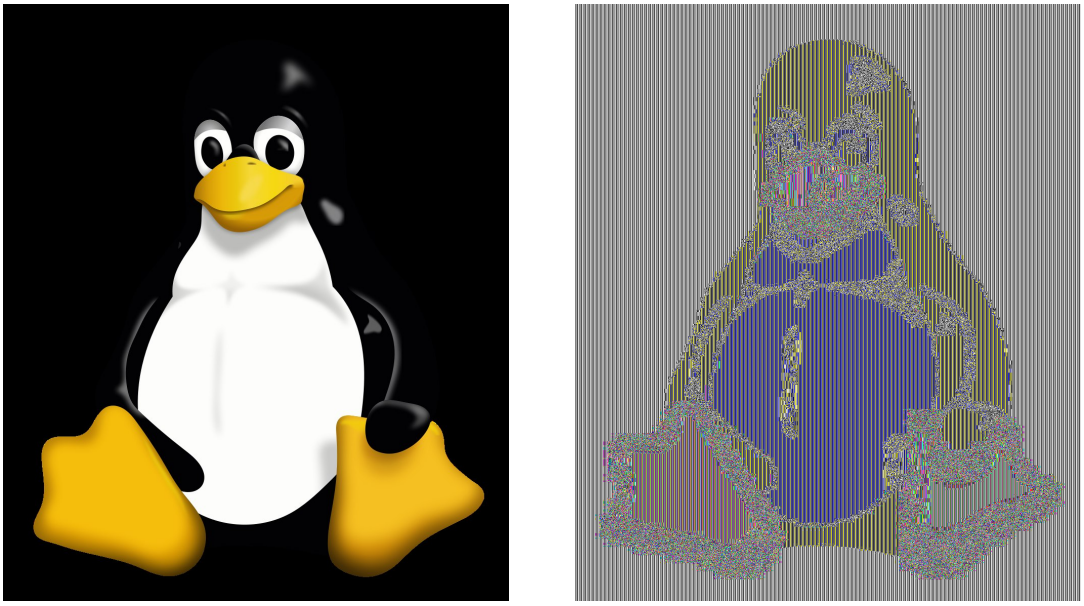
and propose to conduct evaluation in a variety of models, including both white-box and translucent-box attacks. In translucent-box attacks, the adversary is assumed to know the defense mechanism, model architecture, and distribution of training data, but not the precise parameters of the target model. With this knowledge, the adversary can train one or more **surrogate models**, and to generate adversarial examples leveraging such surrogate models.

While other research efforts have attempted to generate adversarial examples based on surrogate models and then assess transferability, existing application of this method does not fully exploit the potential of surrogate models. As a result, one can overestimate the effectiveness of defenses. We propose two improvements. First, one can train many surrogate models under the same configuration, and then generate adversarial examples that can simultaneously fool multiple surrogate models at the same time. Second, one can reserve some surrogate models as "validation models". These validation models are not used when generating adversarial examples; however, generated adversarial examples are first run against them, and only those examples that are able to fool a certain percentage of the validation models are used in evaluation against the target model. This models a more determined and resourceful attacker who is willing to spend more resources to find more effective adversarial examples to deploy, a scenario that is certainly realistic. Our experimental results demonstrate that these more sophisticated adversary strategies lead to significantly higher transferability rates.

Furthermore, in Chapter 4, we propose a new defense mechanism called Random Spiking, where random noises are added to one or more hidden layers during training. Random Spiking generalizes the idea of dropout [6], where hidden units are randomly dropped during training. In Random Spiking, the outputs of some randomly chosen units are replaced by a random noise during training.

In Chapter 5, we present extensive evaluations of several existing defense mechanisms and Random Spiking (RS) under both white-box and translucent-box attacks, and empirically show that RS, especially when combined with adversarial training, improve the resiliency against adversarial examples.

In addition to Random Spiking, which add random noises to one more hidden layers during the training, we explore using randomness to defend against a translucent-box at-



(a) (b)

Figure 1.1. Original image vs Encrypted output of AES-ECB

tacker, since it is natural to take advantage of randomness that is unknown to the adversary. We note that security of cryptographic schemes generally depends on randomness that is unknown to the adversary. An intriguing question is whether we can effectively use secret randomness to defend against the transfer of adversary examples.

Consider the images shown in Fig. 1.1a, which is an example used in some textbooks on cryptography. Fig. 1.1b is the result of directly encrypting the image of Fig. 1.1a using AES under the Electronic Code Book (ECB) mode. As can be seen, while the original color information of the penguin has been altered significantly, the overall outline of the penguin is still discernible in the encrypted version. This was used to illustrate that encryption using the ECB mode is insecure, because the same plaintext block will be encrypted to the same ciphertext block. This property, however, suggests that the "transformed" (or "encrypted") image can still be used to train classifiers.

We propose the following approach. Each CNN is trained with a secret key K. For each input image, it is first transformed using quantization so that the domain size of colors for each pixel is small. After that each image is transformed using the AES block cipher with a

secret key K. That is, each pixel with value v is replace with $\mathsf{AES}_K[v]$. Since the key K can be randomly chosen, it can be kept secret so long as the model does not need to be shared, which is the case when one provides access to the model as a service.

We have observed that directly applying AES results in very low test accuracy, because when adjacent pixels have a similar (but not the exact same) color, they will be mapped to random color values that are not similar. We thus propose to first reduce the color depth by quantization.

Simply applying quantization, however, results in color banding, where there are large patches of the same color, and visible sharp boundary where the color changes. In image processing, dithering is a technique for solving this problem. When using dithering, one intentionally adds noises to randomize quantization error, preventing large-scale patterns such as color banding in images. We use the Floyd-Steinberg dithering algorithm [7], which is the most widely used dithering algorithm. We find that using Floyd-Steinberg dithering alone (without using AES) can also improve the resistance to adversarial examples. We conduct extensive experiments to evaluate these defenses, and show that they improve resiliency against adversary examples with small reduction of accuracy and low computational cost.

Because we were able to train neural network models using transformed images that appears impressionistic, we named our approach MoNet as an homage to Claude MoNet, the father of impressionism.

In Chapter 6, we present the implementation detail of MoNet. In Chapter 7, we evaluate and compare MoNet with Random Spiking and other defense mechanisms. The evaluation shows that MoNet further improves the model robustness against adversarial examples comparing with Random Spiking More specifically, we show that CNNs for image data can be trained on images that are processed using Floyd-Steinberg dithering, and this provides some added protection against adversary examples. Also, combining Floyd-Steinberg dithering with applying pseudo-random permutation using a secret randomness help reduce transferability of adversarial examples.

In addition to image classification, Objection detection is an important task in computer vision, where semantic objects of interests are detected and classified to facilitate further processing. Existing object detection networks are mainly based on Convolutional Neural

Networks (CNN) and can be classified into two approaches: two-stage and one-stage. The two-stage approach, represented by the R-CNN family [8–10], first uses a region proposal network (RPN) to generate a set of region proposals. Then, for each region proposal, the classification network further refines the bounding box and predicts the class label. The alternative approach is to directly predict the bounding box and class label in one-stage, represented by the YOLO family [11–15] and SSD [16].

Similar to image classifiers, it is found that existing object detectors are highly vulnerable to adversarial examples [17–22]. For example, Unified and Efficient Adversary (UEA) [20] trains a Generative Adversarial Network (GAN) for generating adversarial examples that have high transferability. Robust Adversarial Perturbation (RAP) [21] and Dense Adversary Generation (DAG) [22] use iterative gradient-based approach to generate adversarial examples in order to attack the region proposal network (RPN) in two-stage models. Targeted Adversarial Objectness Gradient (TOG) Attacks [17–19] also uses the iterative gradient-based approach to generate adversarial examples, but the attack is applicable to both one-stage and two-stage models. In this dissertation, we consider the TOG attack as the TOG attack generates stronger adversarial examples comparing with other attacks.

To evaluate the adversarial examples for object detection, we consider the same adversarial models used in analyzing image classification: the white-box attack and the translucent-box attack.

We argue that defending against white-box attack for object detection is more challenging than image classification, because adversaries have more space to exploit object detection networks than image classification networks. First, object detection network outputs not only the label for each detection but also the position and the confidence of the detected bounding box. Therefore, adversaries can attack any combinations of these exploitable spaces. For example, the TOG attack [17–19] uses such property to proposed new attacks such as objection vanishing, fabrication, mislabeling, and untargeted which is the combination of the first three attacks. Second, the input resolution of images for object detection network is generally larger than that of image classification network. For example, the input dimension for YoloV3 [13] is 416×416, and the latest YoloV5 [15] uses 640×640 as the default input size and up to 1280×1280 for large scaled version. In contrast, image classification network

such as ResNet-50 [23], Inception-v3 [24], and VGG-19 [25] typically uses inputs of 224×224 resolution. The larger input size makes defending against adversarial examples even harder.

In this dissertation, we focus on defending against adversarial examples for object detection in translucent-box attacks. We explore three approaches to defend against transferability of adversarial examples in object detection: MoNet, Model Ensemble, and Test Time Augmentation (TTA).

In experiments, we found that MoNet models and standard models behave differently, in the sense that adversarial examples generated against one kind of models do not transfer well to the other. We thus explore the possibilities of combining their prediction outputs to further improve resistance to adversarial examples. Specifically, we explore different model ensemble strategies to determine which ones perform better. Note that simple model averaging, a strategy often used in image classification, is not applicable here, because object detection generates 0 or more predictions for a given input image. Instead, we evaluate model ensemble strategies designed for object detection network, such as Affirmative, Consensus, Unanimous [26], and Weighted Boxes Fusion (WBF) [27]. (See definitions in Sec 8.2.1.)

To better defend against adversarial transferability, ideally one should train multiple standard and MoNet models so that it becomes more difficult for adversaries to generate transferable adversarial examples, as each model behaves slightly differently. However, training an object detection network from scratch is more time consuming than image classification network due to the additional task complexity of objection detection and the significantly increased input dimension. Hence, we further explore other techniques that can imitate prediction models with different behaviours, but are cost-effective to users with limited computational resources. Instead of training multiple models from scratch, it has been suggested that Test Time Augmentation (TTA) can be used along with model ensemble without the needs of training new models [26]. Specifically, TTA generates additional inputs such as equalized and flipped images in test times and feed them into the network. We empirically confirm that in the context of object detection, combining MoNet and model ensemble with TTA can further improve the prediction accuracy.

In Chapter 8, we present the strategies to defend against transferability of adversarial examples in object detection. In Chapter 9 , we evaluate proposed strategies. We confirm

that the benefits of dithering and AES encryption can also apply to object detection network. In addition, we found the best strategy to defend against adversarial examples in object detection is Model Ensemble + TTA.

In this dissertation, we make five contributions. (1) The proposed evaluation methodology, especially the more powerful and realistic adversary strategy of attacking multiple surrogates in parallel and using validation models to filter. (2) The idea of Random Spiking, which is demonstrated to offer additional resistance to adversarial examples. (3) The idea of MoNet helps to reduce the transferability of adversarial examples by using the secret randomness. (4) The idea of using MoNet, Model Ensemble, and Test Time Augmentation (TTA) can reduce the transferability of adversarial examples in object detection (5) We provide a thorough evaluation of several defense mechanisms against adversarial examples, improving our understanding of them.

2. BACKGROUND

2.1 Adversarial Examples for Classification

We consider neural networks that are used as m-class classifiers, where the output of a network is computed using the softmax function. Given such a neural network used for classification, let $\mathbf{z}(x)$ denote the vector output of the final layer before the softmax activation, and $C(x)$ denote the classifier defined by the neural network. Then,

$$C(x) = \underset{i}{arg\,max}(\exp(\mathbf{z}(x)_i)/(\sum_{j=1}^{n}\exp(\mathbf{z}(x)_j))). \tag{2.1}$$

Oftentimes, $\exp(\mathbf{z}(x)_i)/(\sum_{j=1}^{n}\exp(\mathbf{z}(x)_j)$ is interpreted as the probability that the input x belongs to the i-th category, and the classifier chooses the class with the highest probability. Under this interpretation, the output $\mathbf{z}(x)$ is related to log of odds ratios, and is thus called the **logits output**.

2.1.1 Attacks for Generating Adversarial Examples

Given a dataset D of instances, each of the form (x, y), where x gives the features of the instance, and y the label, and a classifier $C(\cdot)$ trained using a subset of D, we say that an instance x' is an *adversarial example* if and only if **there exists an instance** $(x, y) \in D$ **such that** x' **is close to** x**,** $C(x) = y$**, and** $C(x') \neq y$.

Note that in the above we did not define what "x' is close to x" means. Intuitively, when x represents an image, by closeness we mean human perceptual similarity. However, we are unaware of any mathematical distance metric that accurately measures human perceptual similarity. In the literature L_p norms are often used as the distance metric for closeness. L_p is defined as

$$L_p(x, x') = \|x - x'\|_p = (\sum_{i=1}^{n}|x_i - x'_i|^p)^{1/p}. \tag{2.2}$$

The commonly used L_p metrics include: L_0, the number of changed pixels [28]; L_1, the sum of absolute values of the changes in all pixels [29]; L_2, the Euclidean norm [1, 30–32]; and

L_∞, the maximum absolute change [2]. In this dissertation, we use L_2 and L_∞ to measure the **distortion** of the adversarial example x'.

When generating an adversarial example against a classifier $C(\cdot)$, one typically starts from an existing instance (x, y) and generates x'. In an **untargeted attack**, one generates x' such that $C(x') \neq y$. In a **targeted attack**, one has a desired target label $t \neq y$ and generates x' such that $C(x') = t$. In this dissertation, we use the following two state-of-the-art attacks.

2.1.2 C&W Attack.

Carlini and Wagner [30] proposed an attack, which we call the **C&W** attack. Given a neural network with logits output $\mathbf{z}$, an input x, and a target class label t, the C&W attack tries to solve the following optimization problem:

$$\underset{x'}{arg\,min}\,(\|x - x'\|_p + c \cdot l(x')) \tag{2.3}$$

where the loss function l is defined as

$$l(x') = \max\left(\max\left\{\mathbf{z}(x')_i : i \neq t\right\} - \mathbf{z}(x')_t, -K\right). \tag{2.4}$$

Here, K is called the **confidence value**, and is a positive number that one can choose. Intuitively, we desire $\mathbf{z}(x')_t$ to be higher than any $\mathbf{z}(x')_i$ where $i \neq t$ so that the neural network predicts label t on input x'. Furthermore, we prefer the gap in the logit of the class t and the highest of any class other than t to be as large as possible (until the gap is K, at which point we consider the gap to be sufficiently large). In general, choosing a large value K would result in adversarial examples that have a higher distortion, but will be classified to the desired label with higher confidence. The parameter $c > 0$ in Eq. (2.3) is a regularization constant to adjust the relative importance of minimizing the distortion versus minimizing the loss function l. In the attack, c is initially set to a small initial value, and then dynamically adjusted based on the progress made by the iterative optimization process. The C&W attack

uses the Adam algorithm [33] to solve the optimization problem in Eq. (2.3). Adam performs iterative gradient-based optimization, based on adaptive estimates of lower-order moments.

2.1.3 Projected Gradient Descent (PGD) Attack

Goodfellow *et al.* [2] proposed the fast gradient sign (**FGS**) attack, which generates adversarial examples based on the gradient sign of the loss value according to the input image. The Projected Gradient Descent (PGD) Attack [34] generates adversarial examples by iteratively applying FGS attack and projecting the output to a valid constrained space. In each iteration, PGD attack computes:

$$x^{t+1} = \prod_{x+S} (x^t + \alpha \text{ sign}(\nabla_x L(\theta, x, y))) \tag{2.5}$$

where x^{t+1} denotes the output of $i+1$-th iteration, and $x+S$ denotes all images whose L_∞ distance to the input x is bounded by a given parameter).

2.2 Existing Defenses for Image Classification

Here we give an overview of some approaches that have been proposed to help defend against adversarial examples.

2.2.1 Adversarial Training.

Goodfellow *et al.* [2] proposed to train a neural network using both the training dataset and newly generated adversarial examples. In [2], it is shown that models that have gone through adversarial training provide some resistance against adversarial examples generated by the FGS method.

2.2.2 Defensive Distillation.

Distillation training was originally proposed by Hinton *et al.* [35] for the purpose of distilling knowledge out of a large model (one with many parameters) to train a more compact

model (one with fewer parameters). Given a model whose knowledge one wants to distill, one applies the model to each instance in the training dataset and generates a probability vector, which is used as the new label for the instance. This is called the soft label, because, instead of a single class, the label includes probabilities for different classes. A new model is trained using instances with soft labels. The intuition is that the probabilities, even those that are not the largest in a vector, encode valuable knowledge. To make this knowledge more pronounced, the probability vector is generated after dividing the logits output with a temperature constant $T > 1$. This has the effect of making the smaller probabilities larger and more pronounced. The new model is trained with the same temperature. However, when deploying the model for prediction, temperature is set to 1.

Defensive Distillation [36] is motivated by the original distillation training proposed by Hinton *et al.* [35]. The main difference between the two training methods is that defensive distillation uses the same network architecture for both initial network and distilled network. This is because the goal of using Distillation here is not to train a model that has a smaller size, but to train a more robust model.

2.2.3 Dropout.

Dropout [6] was introduced to improve generalization accuracy through the introduction of randomness in training. The term "dropout" refers to dropping out units, i.e., temporarily removing the units along with all its incoming and outgoing connections. In the simplest case, during each training epoch, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set to 0.5, which was suggested by the authors of [6].

There are several intuitions why Dropout is effective in reducing generalization errors. One is that after applying Dropout, the model is always trained with a subset of the units in the neural network. This prevents units from co-adapting too much. That is, a unit cannot depend on the existence of another unit, and needs to learn to do something useful on its own. Another intuition is that training with Dropout approximates simultaneous training of an exponential number of "thinned" networks. In the original proposal, dropout

is applied in training, but not in testing. During testing, without applying Dropout, the prediction approximates an averaging output of all these thinned networks. In Monte Carlo dropout [37], dropout is also applied in testing. The NN is run multiple times, and the resulting prediction probabilities are averaged for making prediction. This more directly approximates the behavior of using the NN as an ensemble of models.

Since Dropout introduces randomness in the training process, two models that are trained with Dropout are likely to be less similar than two models that are trained without using Dropout. Defensive Dropout [38] explicitly uses dropout for defense against adversarial examples. It applies dropout in testing, but runs the network just once. In addition, it tunes the dropout rate used in testing by iteratively generating adversarial examples and choosing a drop rate to both maximize the testing accuracy and minimize the attack success rate.

2.2.4 Region-based Classification.

Cao and Gong [3] proposed region-based classification to defend against adversarial examples. Given an input, region-based classification first generates m perturbed inputs by adding bounded random noises to the original input, then computes a prediction for each perturbed input, and finally use voting to make the final prediction. This method slows down prediction by a factor of m. In [3], $m = 10,000$ was used for MNIST and $m = 1,000$ was used for CIFAR. Evaluation in [3] shows that this can withstand adversarial examples generated by the C&W attack under low confidence value K. However, if one slightly increases the confidence value K when generating the adversarial examples, this defense is no longer effective.

2.2.5 MagNet.

Meng and Chen [4] proposed an approach that is called MagNet. MagNet combines two ideas to defend against adversarial examples. First, one trains detectors that attempt to detect and reject adversarial examples. Each detector uses an autoencoder, which is trained to minimize the L_2 distance between input image and output. A threshold is then selected using validation dataset. The detector rejects any image such that the L_2 distance between

it and the encoded image is above the threshold. Multiple detectors can be used. Second, for each image that passes the detectors, a reformer (another autoencoder) is applied to the image, and the output (reformed image) is sent to the classifier for classification.

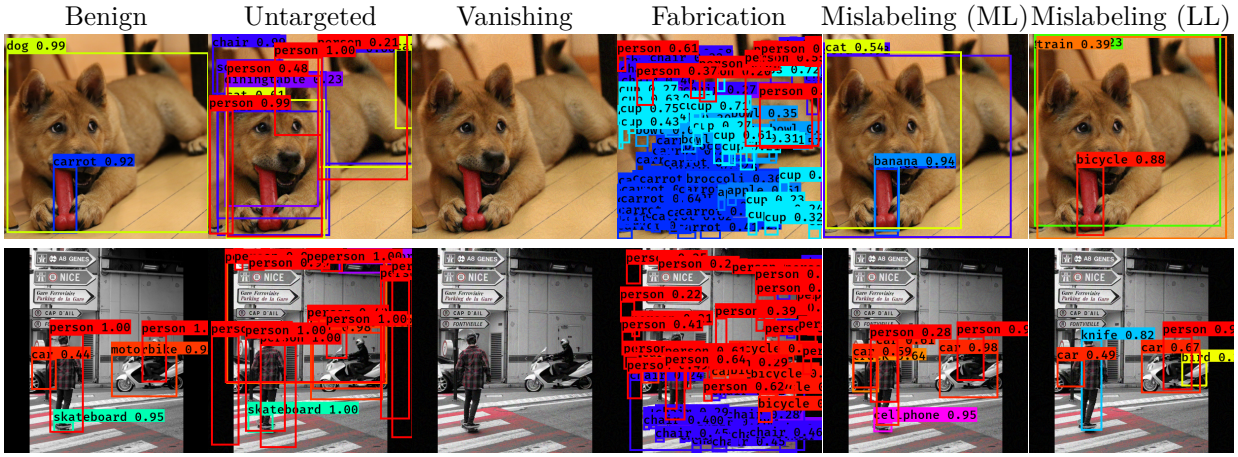
The evaluation of MagNet in [4] considers only adversarial examples generated without knowledge of the MagNet defence. Since one can combine all involved neural networks into a single one, one can still apply the C&W attack on the composite network. In [32], an effective attack is carried out against MagNet by adding to the optimization objective a term describing the goal of evading the detectors.

2.2.6 Feature Squeezing.

Xu *et al.* [5] proposed feature squeezing (FS) for detecting adversarial examples. Feature squeezing uses image pre-processing techniques such as bit-depth reduction as well as local and non-local image smoothing (Gaussian, median filtering etc.) to reduce the search space available to an adversary. By comparing the model prediction on the benign input and squeezed input, feature squeezing can detect adversarial examples with high accuracy and few false positives.

2.3 Adversarial Examples for Object Detection

Table 2.1. Visualization of generated TOG adversarial examples



In image classification, one assumes that each images has a single primary object and predicts the object's class label. In object detection, there may be multiple objects in one image, and one aims to identify, for each object, the bounding box containing the object and its class label. Specifically, given an image as the input, an object detection model M returns a list of detections $D = \{d_1, \ldots, d_n\}$. Each d_i represents a detected object using a 3-tuple (b_i, c_i, s_i), where b_i is the object's bounding box, c_i the corresponding class label, and s_i the confidence score which determines the existence of an object.

In object detection, the prediction output is more complex compared to classification. As a result, there are also more types of adversarial attacks. More specifically, the loss function for object detection network includes contributions from the location and the dimension of the bounding box $L_{bbox}(x, O)$, the class label $L_{class}(x, O)$, and the confidence score $L_{conf}(x, O)$, and can be described as the following formula:

$$L(x, O) = L_{bbox}(x, O) + L_{class}(x, O) + L_{conf}(x, O). \tag{2.6}$$

Chow et al. [17–19] proposed a suite of Targeted Adversarial Objectness Gradient Attacks, named the **TOG** family of attacks. The object-vanishing attack aims to generate adversarial examples have zero detected objects, and achieves this by manipulating $L_{conf}(x, O)$ to reduce the confidence values for all detected objects. The object-fabrication attack aims to create detection of non-existing object, increasing the confidence of such objects. The mislabeling attack aims to replaces the detected class label with a label chosen by the adversary. Two kinds of mislabeling attacks were considered: the most-likely (ML) and the least-likely (LL) attack, which adversaries chose the incorrect class label of an object detected on benign input with the highest and the lowest prediction confidence respectively. Mislabeling ML attack replaces the class label to the most likely targeted label, whereas mislabeling LL attack replaces the class label to the least likely targeted label.

Although each of proposed targeted attacks only exploits proportional loss function, adversary can exploit the whole loss function $L(x, O)$. We call such attack as TOG untargeted attack. Table 2.1 shows examples of TOG attack generated adversarial examples VS their benign images. To generate adversarial examples, both TOG targeted and untargeted attacks

iteratively apply the fast gradient sign (FGS) attack [2] and project the output to a valid constrained space. Specifically, in each iteration, the TOG attack first computes the gradient sign of the targeted loss value and then computes:

$$x^{t+1} = \prod_{x+S}(x^t + \alpha \text{ sign}(\nabla_x L(\theta, x, y))) \tag{2.7}$$

where x^{t+1} denotes the output of $i+1$-th iteration, and $x+S$ denotes all images whose L_∞ distance to the input x is bounded by a given parameter).

There are other attack algorithms such as UEA [20], RAP [21], and DAG [22]. UEA (Unified and Efficient Adversary) [20] trained a Generative Adversarial Network (GAN) framework based adversarial example generator and showed that the transfer attack is more effective. RAP (Robust Adversarial Perturbation) [21] and DAG (Dense Adversary Generation) [22] iterative gradient-based approach to generate adversarial examples in order to attack the region proposal network (RPN) in two-phase models.

2.4 Methods to Improve Model Robustness

There are various methods can be apply to improve the robustness of a neural network model and help defend against adversarial examples. Dropout was introduced to improve generalization accuracy through the introduction of randomness in training [6], which can help defend against adversarial examples [3]. [39–42] showed that dithering can be used in training to improve the robustness of image classifier and reduce the transferability of adversarial examples. Model ensemble combines the prediction results from multiple models in order to produce a final output [26, 27, 43]. Such methods have been widely used in machine learning (ML) applications for improving model accuracy and stability as the final output relies on multiple prediction results. [26] further proposed Test Time Augmentation (**TTA**) which can be used along with model ensemble and [44] showed this approach was effective in defending against adversarial examples for image classifiers.

3. SYSTEMATIC EVALUATION METHODOLOGY OF DEFENSES AGAINST ADVERSARIAL EXAMPLES

We discuss several important factors for evaluation, and introduce the translucent-box model to supplement white-box evaluation.

3.1 Adversary Knowledge

Adversary model plays an important role in any security evaluations. One important part of the adversary model is the assumption on adversary's knowledge.

3.1.1 Knowledge of Model (white-box).

The adversary has full knowledge of the target model to be attacked, including the model architecture, defense mechanism, and all the parameters, including those used in the defense mechanisms. We call such an attack a white-box attack.

3.1.2 Complete Knowledge of Process (translucent-box).

The adversary does not know the exact parameters of the target model, but knows the training process, including model architecture, defense mechanism, training algorithm, and distribution of the training dataset. With this knowledge, the adversary can use the same training process that is used to generate the target model to train one or more **surrogate models**. Depending on the degree of randomness involved in the training process, the surrogate models may be similar or quite different from the target model, and adversarial examples generated by attacking the surrogate model(s) may or may not work very well. The property of whether an adversarial example generated by attacking one or more surrogate models can also work against another target model is known as **transferability**. We call such an attack a translucent-box attack.

Technically, it is possible for a white-box adversary to know less than a translucent-box adversary *in some aspects*. For example, a white-box adversary may not know the distribution of the training data. However, for the purpose of generating adversarial examples,

knowing all details of the target model (white-box) is strictly more powerful than knowing the training process (translucent-box).

3.1.3 Oracle access only (black-box).

Some researchers have considered adversary models where an adversary uses only oracle accesses to the target model. That is, the adversary may be able to query the target model with instances and receive the output. This is also called "decision-based adversarial attack" [45, 46]. We call such an attack a black-box attack.

Some researchers also use black-box attack to refer to what we call translucent-box attacks. We choose to distinguish translucent-box attacks from black-box attacks for two reasons. First, an adversary will have some knowledge about the target model under attack, e.g., the neural network architecture and the training algorithm. Thus the box is not really "black". Second, the two kinds of attacks are very different. One relies on training surrogate models, and the other relies on issuing a large number of oracle queries.

Other researchers use "black-box attack" to refer to the situation that the adversary carries out the attack without specifically targeting the defense mechanism. We argue that such an evaluation has limited values in understanding the security benefits of a defense mechanism, as it is a clear deviation from the Kerckhoffs's principle.

3.1.4 Our Choice of Adversary Model.

We argue that defense mechanisms should be evaluated under both white-box and translucent-box attacks. While developing attacks that can generate adversarial examples using only oracle access is interesting, for a defense mechanism to be effective, one must assume that the adversary cannot break it even if it has the knowledge of the defense mechanism.

Evaluation under white-box attack can be carried out by measuring the level of distortion needed to attack a model. Effective defense against white-box attacks is the ultimate objective. Until defense in the white-box model is achieved, effective defense against translucent-box attacks is valuable and help the research community make progress.

Translucent-box is a realistic assumption especially in an academic setting, as published papers generally include descriptions of the architecture, training process, defense mechanisms and the exact dataset used in their experiments. Robustness and security evaluations under this assumption is also consistent with the Kerckhoffs's principle.

We also note that there are two possible flavors of attacks. Focusing on image classifiers, the goal of an *untargeted attack* is to generate adversarial examples such that the classifier would give any output labels different from what human perception would classify. A *targeted attack* would additionally require the working adversarial examples to induce the classifier into giving specific output labels of the attacker's choosing. In this dissertation, we consider only targeted attacks when evaluating defense mechanisms, as it models an adversary with a more specific objective.

3.2 Adversary Strategy

Even after the assumption about the adversary's knowledge is made, there are still possibilities regarding what strategy the adversary takes. For example, when evaluating a defense mechanism under the translucent-box assumption, a standard method is to train m models, and, for each model, generate n adversarial examples. Then for each of the m model, treat it as the target model, and feed the $(m-1)n$ adversarial examples generated on other models to it, and report the percentage of success among the $m(m-1)n$ trials.

Such an evaluation method is assessing the success probability of the following naive adversary strategy: The adversary trains one surrogate model, generates an adversarial example that works against the surrogate model, and then deploy that adversarial example. We call this a **one-surrogate attack**. A real adversary, however, can use a more effective strategy. It can try to generate adversarial examples that can fool multiple surrogate models at the same time. After generating them, it can first test whether the adversarial examples can fool surrogate models that are not used in the generation. We call this a **multi-surrogate attack**.

For any defense mechanism that is more effective against adversarial examples under the translucent-box attack than under the white-box model, the additional effectiveness must be

due to the randomness in the training process. When that is the case, the above adversary strategy would have much higher success rate than the naive adversary strategy. Evaluation should be done against this adversary strategy.

We thus propose the following procedure for evaluating a defense mechanism in the translucent-box setting. One first trains $t + v$ surrogate models. Then a set of t models are randomly selected, and adversarial examples are generated that can *simultaneously* attack all t of them; that is, the optimization objective of the attack includes all t models. For the remaining v models, we use leave-one-out validation. That is, for each model, we use $v-1$ model as validation models, and select only adversarial examples that can fool a certain fraction of the validation models. Only for the examples that pass this validation stage, do we record whether it successfully transfer to the target model or not. We call such an attack a **multi-surrogate with validation attack**. The percentage of the successful transfer is used for evaluation. In our experiments, we use $t = v = 8$, and an example is selected when it can successfully attack at least 5 out of 7 validation models.

3.3 Parameters and Data Interpretation

Training a defense mechanism often requires multiple parameters as inputs. For example, a defense mechanism may be tuned to be more vigilant against adversarial examples, at the cost of reduced classification accuracy. When comparing defense mechanisms, one should choose parameters in a way that the classification accuracy on test dataset is similar.

At the same time, when using the C&W attack to generate adversarial examples, an important parameter is the confidence value K. A defense mechanism may be able to resist adversarial examples generated under a low K value, but may prove much less effective against those generated under a higher value K (see, e.g., [3]). Using the same K value for different defenses, however, may not be sufficient for providing a level playing field for comparison. The K value represents an input to the algorithm, and what really matters is the quality of the adversarial examples. We propose to run the C&W attack against a defense mechanism under multiple K values, and group the resulting adversarial examples based on their distortion. We can then compare how well a defense mechanism performs

against adversarial examples with similar amount of distortion. That is, we group adversarial examples based on the L_2 distance and compute the average transferability for each group. When using the PGD attack to generate adversarial examples, we group adversarial examples based on the L_∞ distance, which is the important parameter to control the upbound of the distortion of generated adversarial examples.

4. PROPOSED DEFENSE:RANDOM SPIKING

From a statistical point of view, the problem with adversarial examples is that of classification under covariate shifts [47]. A covariate shift happens when the training and test observations follow different distributions. In the case of adversarial examples, this is clearly the case, as new adversarial examples are generated and added to the test distribution. If the test distribution with adversarial examples can be known, a simple and optimal way for dealing with covariate shifts is training the model with samples from the test distribution, rather than using the original training data [47–50], assuming that we have access to enough such examples. Training with adversarial examples can be viewed as a robust optimization procedure [34] approximating this approach.

Unfortunately, training with adversarial examples does not fully solve the defense problem. Adversaries can adapt the test distribution (a new covariate shift) to make the new classifier perform poorly again on test data. That is, given a model trained with adversarial examples, the adversary can find additional adversarial examples and use them. In this minimax game, where the adversary is looking for a covariate shift and the defender is training with the latest covariate shift, the odds are stacked against the defender, who is always one step behind the attacker [51, 52].

Fundamentally, to win this game, the defender needs to mimic human perception. That is, as long as there are instances (real or fabricated) where humans and ML models classify differently, these can be over represented in the test data by the adversary's covariate shift. Models that either underfit or overfit both make mistakes by definition, and these mistakes can be used in the adversary's covariate shift. Only a model with no training or generalization errors under all covariate shifts is not vulnerable to attacks.

4.1 Motivation of Our Approach

While it is impossible to completely eliminate classification errors, several things can be done to help defend against adversarial examples by making them harder to find.

One approach is to reduce the number of instances that the ML models disagree with human perception. Training with adversarial examples help in this regard. Using more

robust model architecture and training procedure can also help. When giving an image to train the model, intuitively we want to say that "all instances that look similar to this instance from a human's perspective should also have the same label". Unfortunately, finding which images humans will consider to be "similar to this instace" and thus should be of the same class is not a well-defined procedure. Today, the best we can hope for is that for some mathematical distance measure (such as L_2 distance) and with a smaller enough threshold, humans will consider the images to be similar. If we substitute "look similar to ... from a human's perspective" with "within a certain L_2 distance", this is a precise statement. This suggests that one training instance should be interpreted as a set of instances (e.g., those within a certain L_2 distance of the given one) all have the same given label. Our proposed defense is to some extent motivated by this intuition.

Another way is to make it more difficult for adversaries to discover adversarial examples, even if they exist. One approach is to use an ensemble model, wherein multiple models are trained and applied to an instance and the results are aggregated in some fashion. For an adversarial example to work, it must be able to fool a majority of the models in the ensemble.

If we consider defense in the translucent box adversary model, another approach is to increase the degree of randomness in the training process, so that adversarial examples generated on the surrogate models do not transfer well.

Our proposed new defense against adversarial examples are motivated by these ideas, which are recapped below. First, each training instance should be viewed as representatives of instances within a certain L_2 distance. Second, we want to increase the degree of randomness in the training process. Third, we want to approximate the usage of an ensemble of models for decision.

4.2 Random Spiking

As discussed in Section 2.2, Dropout has been proposed as a way to defend against adversarial examples. Dropout can be interpreted as a way of regularizing a neural network by adding noise to its hidden units. The idea of adding noise to the states of units has also been used in the context of Denoising Autoencoders (DAEs) by Vincent et al. [53, 54], where

noise is added to the input units of an autoencoder and the network is trained to reconstruct the noise-free input. Dropout changes the behavior of the hidden units. Furthermore, instead of adding random noises, in Dropout, values are set to zero.

Our proposed approach generalizes both Dropout and Denoising Autoencoders. Instead of training with removed units or injecting random noises into the input units, we inject random activations into some hidden units near the input level. We call this method **Random Spiking**. Similar to Dropout, there are two approaches at inference time. The first is to use random spiking only in training, and does not use it at inference time. The second is to use a Monte Carlo decision procedure. That is, at decision time, one runs the NN multiple times with random spiking, and aggregate the result into one decision.

The motivations for random spiking are many-fold. First, we are simulating the interpretation that each training instance should be treated as a set of instances, each with some small changes. Injecting random perturbations at a level near the input simulates the effect of training with a set of instances. Second, adversarial examples make only small perturbations on benign images that do not significantly affect human perception. These perturbations inject noises that will be amplified through multiple layers and change the prediction of the networks. Random Spiking trains the network to be more robust to such noises. Third, if one needs to increase the degree of randomness in the training process beyond Dropout, using random noises instead of setting activations to zero is a natural approach. Fourth, when we use the Monte Carlo decision procedure, we are approximating the behavior of a model ensemble.

More specifically, random spiking adds a filtering layer in between two layers of nodes in a DNN. The effect of the filtering layer may change the output values of units in the earlier layer, affecting the values going into the later layer. With probability p, a unit's value is kept unchanged. With probability $1-p$, a unit's value is set to a randomly sampled noise. If a unit has its output value thus randomly perturbed, in back-propagation we do not propagate backward through this unit, since any gradient computed is related to the random noise, and not the actual behavior of this unit. For layers after the Random Spiking filtering layer, back-propagation update would occur normally.

We use the Random Spiking filtering layer just once, after the first convolutional layer (and before any max pooling layer if one is used). This is justified by the design intuition. We also experimented with adding the Random Spiking filtering layer later in the NN, and test accuracy drops. There are two explanations for that. First, since units chosen to have random noises stop back-propagation, having them later in the network has more impact on training. Second, when random noises are injected early in the network, there are more layers after it, and there is sufficient capacity in the model to deal with such noises without too much accuracy cost. When random noises are injected late, fewer layers exist to deal with their effect, and the network lacks the capacity to do so.

4.2.1 Generating Random Noises.

To implement Random Spiking, we have to decide how to sample the noises that are to be used to replace the unit outputs. Sampling from a distribution with a fixed range is problematic because the impact of noise depends on the distribution of other values in the same layer. If a random perturbation is too small compared to other values in the same layer, then its randomization effect is too small. If, on the other hand, the magnitude of the noise is significantly larger than the other values, it overwhelms the network. In our approach, we compute the minimum and maximum value among all values in the layers to be filtered, and sample a value uniformly at random in that range. Since training NN is often done using mini-batches, the minimum and maximum values are computed from the whole batch.

4.2.2 Monte Carlo Random Spiking as a Model Ensemble.

For testing, we can use the Monte Carlo decision procedure of running the network multiple times and use the average. This has attractive theoretical guarantees, at the cost of overhead for decision time, since the NN needs to be computed multiple times for one instance. We now show that the Monte Carlo Random Spiking approximates a model ensemble. Let (x, y) be a training example, where x is an image and y is the image's one-hot encoded label. Consider a RS neural network with softmax output $\hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W})$, neuron

weights $\boldsymbol{W}$, and spike parameters $\boldsymbol{b}$ and $\boldsymbol{\epsilon}$, where bit vector $\boldsymbol{b}_i = 1$ indicates that the i-th hidden neuron of the RS layer gives out a noise output $\boldsymbol{\epsilon}_i \in \mathbb{R}$ sampled with density $f(\epsilon)$, otherwise $\boldsymbol{b}_i = 0$ and the output of the RS layer is a copy of its i-th input from the previous layer (i.e., the original value of the neuron). By construction, $\boldsymbol{b}_i = 1$ with probability $1 - p$ independent of other RS neurons. Let $L(y, \hat{y})$ be a convex loss function over $\hat{y}$, such as the cross-entropy loss, the negative log-likelihood, or the square error loss. Then, the following proposition holds:

Proposition 1. *Consider the ensemble RS model*

$$\overline{\hat{y}}(x, \boldsymbol{W}) \equiv \sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} \hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W}) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}, \tag{4.1}$$

where f is a density function, $p(\boldsymbol{b})$ is the probability that bit vector $\boldsymbol{b}$ is sampled, and $\hat{y}$ is a RS neural network with one spike layer. Then, by stochastically optimizing the original RS neural network $\hat{y}$ by sampling bit vectors and noises, we are performing the minimization

$$\boldsymbol{W}^\star = \operatorname*{argmin}_{\boldsymbol{W}} L(y, \overline{\hat{y}}(x, \boldsymbol{W}))$$

through a variational approximation model using an upper bound of the loss $L(y, \overline{\hat{y}}(x, \boldsymbol{W}))$.

Proof of Proposition 1

Proof sketch. The Monte Carlo sampling used in the RS neural network optimization gives an unbiased estimate of the gradient

$$\begin{aligned} &\sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} \frac{\partial}{\partial \boldsymbol{W}} L\left(y, \hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W})\right) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \\ &= \frac{\partial}{\partial \boldsymbol{W}} \sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} L\left(y, \hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W})\right) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}, \end{aligned}$$

with the above equality given by the linearity of the expectation and integral operators. That is, the RS neural network optimization is a Robbins-Monro stochastic optimization [55] that minimizes

$$\boldsymbol{W}' = \underset{\boldsymbol{W}}{\operatorname{argmin}} \sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} L(y, \overline{\hat{y}}(x, \boldsymbol{W})) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}.$$

L is convex on $\hat{y}$, then by Jensen's inequality

$$\sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} L\left(y, \hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W})\right) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \geq$$
$$L\left(y, \sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} \hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W}) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}\right) \equiv L\left(y, \overline{\hat{y}}(x, \boldsymbol{W})\right).$$

Thus, the RS neural network minimizes an upper bound of the loss of the ensemble RS model $\overline{\hat{y}}(x, \boldsymbol{W})$, yielding a proper variational inference procedure [56]. □

Definition 2 (MC Avg. Inference)**.** At inference time, we use Monte Carlo sampling to estimate the RS ensemble

$$\overline{\hat{y}}(x, \boldsymbol{W}) = \sum_{\forall \boldsymbol{b}} \int_{\boldsymbol{\epsilon}} \hat{y}(x, \boldsymbol{b}, \boldsymbol{\epsilon}, \boldsymbol{W}) p(\boldsymbol{b}) f(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon},$$

where f is a density function, $p(\boldsymbol{b})$ is the probability that bit vector $\boldsymbol{b}$ is sampled.

4.2.3 Adaptive Attack against Random Spiking.

Since Random Spiking introduces randomness during training, an adaptive attacker knowing that Random Spiking has been deployed but is unaware of the exact parameters of the target model can train multiple surrogate models, and try to generate adversarial examples that can simultaneously cause all these models to misbehave. That is, the multi-surrogate with validation is a natural adaptive attack against Random Spiking, and any other defense mechanisms that rely on randomness during training. In this attack, one uses probabilities from all surrogate models to generate the adversarial example. This is similar to the Expectation over Transformation (EOT) [57] approach for generating adversarial examples.

5. EXPERIMENTAL EVALUATION OF RANDOM SPIKING BY USING SYSTEMATIC EVALUATION METHODOLOGY

We present experimental results comparing the various defense mechanisms using our proposed approach **Random Spiking**.

5.1 Dataset and Model Training

For our experiments, we use the following 3 datasets: MNIST [58], Fashion-MNIST [59], and CIFAR-10 [60]. Table 5.1 gives an overview of their characteristics.

Table 5.1. Overview of datasets

Dataset	Image size	Training Instances	Test Instances	Color space
MNIST	28×28	60,000	10,000	8-bits Gray-scale
Fashion-MNIST	28×28	60,000	10,000	8-bits Gray-scale
CIFAR-10	32×32	50,000	10,000	24-bits True-Color

We consider 9 schemes equipped with different defense mechanisms, all of which share the same network architectures and training parameters. For MNIST, we follow the architecture given in the C&W paper [30]. Fashion-MNIST was not studied in the literature in an adversarial setting, and the model architectures used for CIFAR-10 in previous papers delivered a fairly low accuracy. Thus for Fashion-MNIST and CIFAR-10, we use the state-of-the-art WRN-28-10 instantiation of the wide residual networks [61]. We are able to achieve state-of-the-art test accuracy using these architectures. Some of these mechanisms have adjustable parameters, and we choose values for these parameters so that the resulting models have a comparable level of accuracy on the testing data. As the result, all 9 schemes result in small accuracy drop.

Table 5.2 gives the test errors, and Tables 5.3 and 5.4 give details of the model architecture, and training parameters. When a scheme uses either Dropout or Random Spiking, we consider 3 possible decision procedures at test time. By "Single pred.", we mean Dropout and Random Spiking are not used at test time. By "Voting", we mean running the network with

Table 5.2. Test errors (mean±std).

		MNIST	Fashion-MNIST	CIFAR-10
Standard	Single pred.	$0.77 \pm 0.05\%$	$4.94 \pm 0.19\%$	$4.38 \pm 0.21\%$
Dropout	MC Avg.	$0.67 \pm 0.07\%$	$4.75 \pm 0.09\%$	$4.46 \pm 0.25\%$
Distillation	MC Avg.	$0.78 \pm 0.05\%$	$4.81 \pm 0.18\%$	$4.33 \pm 0.27\%$
RS-1	MC Avg.	$0.88 \pm 0.09\%$	$5.34 \pm 0.10\%$	$5.59 \pm 0.22\%$
RS-1-Dropout	MC Avg.	$0.71 \pm 0.07\%$	$5.32 \pm 0.17\%$	$5.81 \pm 0.27\%$
RS-1-Adv	MC Avg.	$0.98 \pm 0.11\%$	$5.49 \pm 0.16\%$	$6.20 \pm 0.40\%$
Magnet	*Det. Thrs.*	0.001	0.004	0.004
	MC Avg.	$0.87 \pm 0.06\%$	$5.36 \pm 0.17\%$	$5.52 \pm 0.24\%$
Dropout-Adv	MC Avg.	$0.69 \pm 0.07\%$	$4.76 \pm 0.11\%$	$4.71 \pm 0.19\%$
RC	L_2 *noise*	0.4	0.02	0.02
	Voting	$0.77 \pm 0.11\%$	$5.39 \pm 0.23\%$	$5.72 \pm 0.46\%$

Dropout and/or Random Spiking 10 times, and use majority voting for decision (with ties decided in favor of the label with smaller index). By "MC Avg.", we mean using Definition 2 by running the network with Dropout and/or Random Spiking 10 times, and averaging the 10 probability vectors. For each scheme, we train 16 models (with different initial parameter values) on each dataset, and report the mean and standard deviation of their test accuracy. We observe that using Voting or MC Avg, one can typically achieve a slight reduction in test error.

5.1.1 Adversarial training

Two defense mechanisms require training with adversarial examples, which are generated by applying the C&W L_2 targeted attack on a target model, using randomly sampled training instances and target class labels.

5.1.2 Upper Bounds on Perturbation.

For each dataset, we generated thousands of adversarial examples with varying confidence values for each training scheme, and have them sorted according to the added amount of

Table 5.3. Mode Architectures. We use WRN-28-10 for Fashion-MNIST and CIFAR-10 ($k = 10, N = 4$).

MNIST		Fashion-MNIST			CIFAR-10	
		Group	Output Size	Kernel, Feature	Output Size	Kernel, Feature
Conv.ReLU	$3 \times 3 \times 32$	Conv1	28×28	$[3 \times 3, 16]$	32×32	$[3 \times 3, 16]$
Conv.ReLU	$3 \times 3 \times 32$					
Max Pooling	2×2	Conv2	28×28	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$
Conv.ReLU	$3 \times 3 \times 64$					
Conv.ReLU	$3 \times 3 \times 64$					
Max Pooling	2×2	Conv3	14×14	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times N$	16×16	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times N$
Dense.ReLU	200					
Dense.ReLU	200	Conv4	7×7	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$
Softmax	10	Softmax	10		10	

perturbation, measured in L_2. We have observed that from instances that have high amount of perturbation one can visually observe the intention of adversarial example. We thus chose a cut-off upper bound on L_2 distance. The chosen L_2 cut-off bounds are included in Table 5.5, and used as upper limits in many of our later experiments. With the bounds on L_2 fixed, we then empirically determine an upper bound for the confidence value to be used in the C&W-L_2 attacks for generating adversarial examples for training purposes. To diversify the set of generated adversarial examples, we sample several different confidence values within the bound, which are also reported in Table 5.5.

5.1.3 Magnet.

We use the trained *Dropout* model as the prediction model, and train the Magnet defensive models (reformers and detectors) [4] based on the publicly released Magnet implementation[1]. Identical to the settings[2] presented in the original Magnet paper [4], for MNIST, we use Reformer I, Detector I/L_2 and Detector II/L_1, with detection threshold set to 0.001. Since

[1]↑https://github.com/Trevillie/MagNet

[2]↑Regarding the Detector settings, a small discrepancy exists between the paper and the released source code. After confirming with the authors, we follow what is given by the source code.

Table 5.4. Training Parameters.

Parameters	MNIST	Fashion-MNIST & CIFAR-10
Optimization Method	SGD	SGD
Learning Rate	0.01	0.1 initial, multiply by 0.2 at 60, 120 and 160 epochs
Momentum	0.9	0.9
Batch Size	128	128
Epochs	50	200
Dropout (Optional)	0.5	0.1
Data Augmentation	-	Fashion-MNIST: Shifting + Horizontal Flip CIFAR-10: Shifting + Rotation + Horizontal Flip + Zooming + Shear

Table 5.5. Parameters used for generating adversarial examples. The values for K reported here were chosen so that the generated examples would fit a predetermined L_2 cut-off.

Dataset	L_2 **cut-off**	**Working confidence values** (K)	**Examples for each** K (n)
MNIST	3.0	$\{0, 5, 10, 15\}$	3000
Fashion-MNIST	1.0	$\{0, 20, 40, 60\}$	3000
CIFAR-10	1.0	$\{0, 20, 40, 60, 80, 100\}$	2000

Fashion-MNIST was not studied in [4], we use the same model architecture as CIFAR-10 presented in the original Magnet paper [4]. For Fashion-MNIST and CIFAR-10, we use Reformer II, Detector II/L_1, Detector II/$T10$ and Detector II/$T40$, and with a detection threshold (rate of false positive) of 0.004, which results in test error rates comparable to those of the other schemes.

5.1.4 Random Spiking with standard model (RS-1).

A Random Spiking (RS) layer is added after the first convolution layer in the *standard* architecture. We choose $p = 0.8$, so that 20% of all neuron outputs are randomly spiked.

5.1.5 Random Spiking with Dropout (RSD-1).

We add the RS layer to the *Dropout* scheme. All other parameters are identical to what we used for RS-1. We also use ***RSD-1*** as a shorthand to refer to this scheme.

5.1.6 Distillation.

We use the same network architecture and parameters as we did for the training of *Dropout* models. Identical to the configuration used in [30], we train with temperature $T = 100$ and test with $T = 1$ for all three datasets.

5.1.7 Region-based Classification (RC).

We use the *Dropout* models for *RC*. For each test example, we generate t additional examples, where for each pixel, a noise was randomly chosen from $(-r, r)$ and added to it. Prediction is then made with majority voting on the t input examples. Identical to the original RC paper [3], we use $t = 10,000$ for MNIST and $t = 1,000$ for CIFAR-10. We also use $t = 1,000$ for Fashion-MNIST. We choose values for r ($r = 0.4$ for MNIST, and $r = 0.02$ for Fashion-MNIST and CIFAR-10) so that the test errors would be comparable to the other mechanisms.

5.1.8 Adversarial Dropout (Dropout-Adv).

To use adversarial training with *Dropout*, we leverage the trained Dropout model from before as the target model for generating adversarial examples. We generated $12,000$ adversarial examples for each *Dropout* model by perturbing training instances. To ensure that the adversarial examples indeed should be classified under the original label, we sort the adversarial examples according to their L_2 distances in ascending order, and add only the first $10,000$ examples into the training dataset. These examples have L_2 distances lower than the cutoff mentioned earlier. We then apply the Dropout training procedure as described before on the new training dataset.

Table 5.6. C&W targeted Adv Examples *L2* (mean±std) when attacking a single model.

	MNIST	Fashion-MNIST	CIFAR-10
Standard	2.12 ± 0.69	0.12 ± 0.08	0.17 ± 0.08
Dropout	1.80 ± 0.52	0.14 ± 0.07	0.17 ± 0.08
Distillation	2.02 ± 0.63	0.13 ± 0.07	0.17 ± 0.07
RS-1	2.06 ± 0.76	0.31 ± 0.16	0.32 ± 0.14
RSD-1	1.79 ± 0.86	0.36 ± 0.21	0.32 ± 0.15
RS-1-ADV	2.36 ± 0.80	0.56 ± 0.30	0.39 ± 0.18
Magnet	2.22 ± 0.65	0.28 ± 0.15	0.29 ± 0.21
Dropout-Adv	2.44 ± 0.66	0.33 ± 0.15	0.18 ± 0.07

Table 5.7. C&W Adv Examples *L2* (mean±std) with *Multi 8* attack strategy.

	MNIST	Fashion-MNIST	CIFAR-10
Standard	2.50 ± 0.77	0.22 ± 0.15	0.25 ± 0.10
Dropout	2.29 ± 0.65	0.25 ± 0.13	0.26 ± 0.10
Distillation	2.37 ± 0.71	0.24 ± 0.14	0.33 ± 0.13
RS-1	2.77 ± 0.82	0.54 ± 0.25	0.49 ± 0.18
RSD-1	2.77 ± 0.93	0.61 ± 0.30	0.51 ± 0.18
RS-1-ADV	3.18 ± 0.88	1.04 ± 0.44	0.64 ± 0.23
Magnet	2.68 ± 0.75	0.54 ± 0.25	0.47 ± 0.24
Dropout-Adv	2.93 ± 0.70	0.57 ± 0.23	0.29 ± 0.10

5.1.9 Adversarial Random Spiking (RS-1-ADV).

For this adversarial training method, we use *RS-1* as the target model. The training parameters and procedure are largely identical to what were described for Dropout-Adv above.

5.2 White-box Evaluation

We first evaluate the effectiveness of the defense mechanisms under white-box attacks. We apply the C&W white-box attack with confidence 0 to generate targeted adversarial examples, and measure the L_2. distance of the generated adversarial examples. We consider

both single-model attack, where the adversarial example targets a single model, and multi-8 attack, where the adversarial example aims at attacking 8 similarly trained model at the same time. This can be considered as a form of ensemble white-box attack [62].

Tables 5.6 and 5.7 present the average L_2 distances of the generated examples for those generated adversarial examples. RS-1-Adv results in models that are more difficult to attack, requires on average the highest perturbations (measured in L_2 distance) among all evaluated defenses. Comparing to other methods, adversarial examples generated by RS-1 and RSD-1 have either higher or comparable amount of distortion. These again suggest RS offers additional protection against adversarial examples.

5.3 Model Stability

Given a benign image and its variants with added noise, a more robust model should intuitively be able to tolerate a higher level of noise without changing its prediction results. We refer to this property as model stability. Here we evaluate whether models from a defense mechanism can correct label instances that are perturbed. This serves several purposes. First, in [63], it is suggested that vulnerability to adversarial examples and low performance on randomly corrupted images, such as images with additive Gaussian noise, are two manifestations of the same underlying phenomenon. Hence it is suggested that adversary defenses should consider robustness under such perturbations, as robustness under such perturbations are also indications of resistance against adversary attacks. Second, evaluating stability is identified in [64, 65] as a way to check whether a defense relies on obfuscated gradients to achieve its defense. For such a defense, random perturbation may discover adversarial examples when optimized search based on gradients fail. Third, some defense mechanisms (such as Magnet) rely on detecting whether an instance belongs to the same distribution as the training set, and consider an instance to be an adversarial example if it does. However, when an input instance goes through some transformation that has little impact on human visual detection (such as JPEG compression), it will be considered as an adversarial example by the defense. This will impact accuracy of deployed systems, as the encountered instances may not always follow the training distribution.

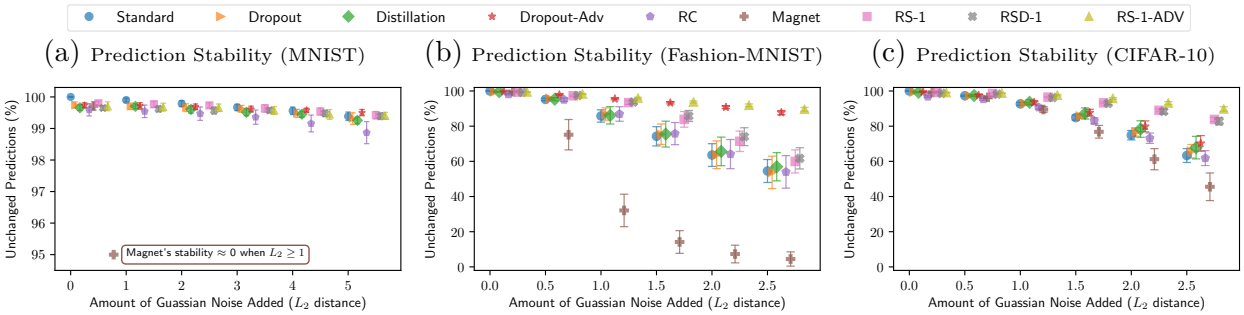


Figure 5.1. Evaluating model stability with Gaussian Noise

5.3.1 Stability with Added Gaussian Noise

We measure how many predictions would change if a certain amount of Gaussian noise is introduced to a set of benign images. For a given dataset and a model, we use the first $1,000$ images from the test dataset. We first make a prediction on those selected images and store the results as *reference predictions*. Then, for each selected image and chosen L_2 distance, we sample Gaussian noise, scale it to the desired L_2 value, and add the noise to the image. Pixel values are clipped if necessary, to make sure the new noisy variant is a valid image. We repeat this process 20 times (noise sampled independently per iteration).

Fig. **??** shows the effect of Gaussian noise on prediction stability for each training method (averaged over the 16 models trained in Sec. 5.1). Model stability inevitably drops for each scheme as the amount of Gaussian noise as measured by L_2 increases. However, different schemes behave differently when L_2 increases.

For MNIST, most schemes have stability above 99%, even when L_2 is as large as 5. However, Magnet has stability approaching 0 when the L_2 distance is greater than 1, because majority of those instances are rejected by Magnet.

For Fashion-MNIST, we see more interesting differences among the schemes. The two approaches that have highest stability are the two with adversarial training. When $L_2 = 2.5$, RS-1-ADV has stability 87.4%, and Dropout-Adv has stability 86%. Other schemes have stability around 60%; among them, RS-1 and RSD-1 have slightly higher stability than others.

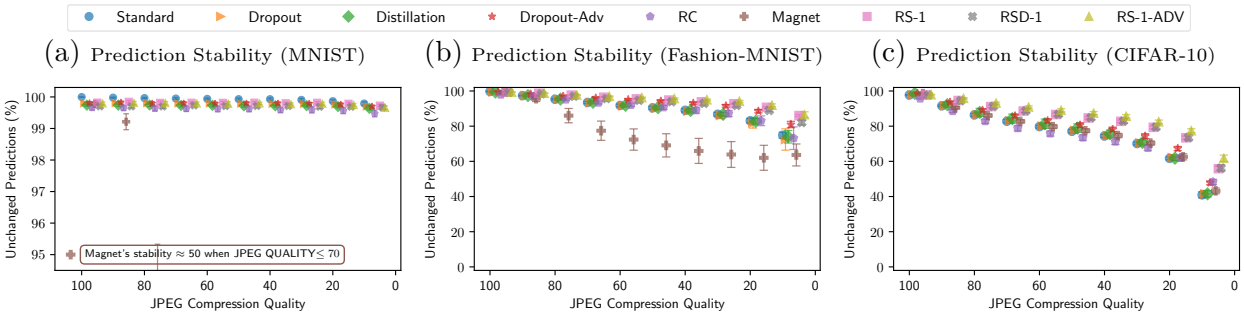


Figure 5.2. Evaluating model stability with JPEG compression

For CIFAR-10, we see that RS-1-ADV, RSD-1, and RS-1 have the highest stability as the amount of noise increases. When $L_2 = 2.5$, they have stability 87.9%, 81.7%, 83%, respectively. The other schemes have stability 70% or lower.

Furthermore, on all datasets, RS-1-ADV, RSD-1, and RS-1 give consistent results. Recall that we trained 16 models for each scheme, Fig. 5.1 also plots the standard deviation of the stability result of the 16 models. RS-1-ADV, RSD-1, and RS-1 have very low standard deviation, which in turn also suggest more consistent behavior when facing perturbed images.

5.3.2 Stability with JPEG compression

Given a set of benign images, we measure how many predictions would change if JPEG compression is applied to images. For a given test dataset and a model, we compare the prediction on the benign test dataset (*reference predictions*) with the prediction on JPEG compressed test dataset with a fixed chosen JPEG compression quality (**JCQ**). For the sake of time efficiency, for this particular set of experiments, we reduced the number of iterations used by RC to one-tenth of its original algorithm.

Fig. 5.2 shows the effect of JPEG compression on prediction stability for each training method (averaged over the 16 models trained). Model stability decreases for each scheme as the JCQ (ranges $10 - 100$) decreases.

For MNIST, most schemes achieve stability over 99, even if the JCQ is 10. Magnet is the outlier, which has a stability of around 50 when the JCQ is 70, and has a stability

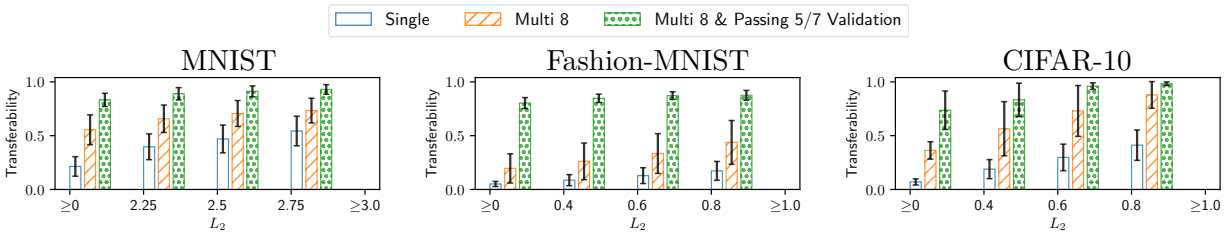


Figure 5.3. Transferability of adversarial examples found by the 3 attack strategies

of less than 20 when the JCQ is less than or equal to 40, because of the high rejection rate of MagNet. We believe that both of these results are related to the fact that MNIST images have black backgrounds that span most of the image. Noises introduced by JPEG compression result mostly in perturbations in the background that are ignored by most NN models. Since Magnet uses autoencoders to detect deviations from the input distributions, these noises trigger detection. Since Magnet aims at detecting perturbed images, this should not be considered as a weakness of Magnet.

For Fashion-MNIST, we see that RS-1-ADV, RSD-1, and RS-1 outperform other schemes on the stability as the JCQ decreases. When JCQ = 10, they have stability 85.2%, 80.9%, 84.4%, respectively. The other schemes have stability 80% or lower; The closest to the RS-class among other schemes is ADV.

For CIFAR-10, we see that RS-1-ADV, RSD-1, and RS-1 have the highest stability as the JCQ decreases. When JCQ = 10, they have stability 60.9%, 55.6%, 55.4%, respectively. The other schemes have stability 50% or lower; the highest among the other schemes is RC.

5.4 Evaluating Attack Strategies

Here we empirically show that our proposed attack strategy, as presented in Sec. 3.2, can indeed generate adversarial examples that are more transferable. In attacks like the C&W attack, a higher confidence value will typically lead to more transferable examples, but the amount of perturbation would usually increase as well, sometimes making the example noticeably different under human perception.

Intuitively, a better attack strategy should give more transferable adversarial examples using less amount of distortion. Hence we use *Distortion vs Transferability* to compare 3 possible attack strategies. Similar to previous experiments, we measure the amount of distortion using L_2 distance. In Fig. 5.3 we present the effectiveness of each attack strategy, averaged across the 9 schemes.

The first strategy we evaluated is a standard C&W attack which generates adversarial examples using only one surrogate model, dubbed *'Single'*. Recall that for each training/defense method, we have 16 models that are surrogates of each other (Sec. 5.1). For each surrogate model, we randomly select half of the original dataset as the training dataset, since the adversary may not have full knowledge of the training dataset under the transfer attack setting. For the *Single* strategy, we apply the C&W attack on 4 of the models independently to generate a pool of adversarial examples. The transferability of those examples are then measured and averaged on the remaining 12 target models. Regardless of the training methods and defense mechanisms in place, adversarial examples generated using the *Single* strategy often have limited transferability, especially when the allowed amount of distortion (L_2 distance) is small.

The second attack strategy that we evaluate is to generate adversarial examples using multiple surrogate models. For this, we use 8 of the 16 surrogate models for generating attack examples. The C&W attack can be adapted to handle this case with a slightly different loss function. In our experiments, we use the sum of the loss functions of the 8 surrogate models as the new loss function. We also use slightly lower confidence values than in Sec. 5.1.1 ($\{0, 10, 20, 30\}$ for Fashion-MNIST, $\{0, 20, 40, 60\}$ CIFAR-10). The transferability of the generated adversarial examples are then measured and averaged on the remaining 8 models as the target. We refer to this as *'Multi 8'*. As shown in Fig. 5.3, given the same limit on the amount of distortion (L_2 distance), a significantly higher percentage of examples generated using the *Multi 8* strategy are transferable than those found using the *Single* strategy.

Additionally, we evaluate a third attack strategy that is based on *Multi 8*. As discussed in Sec. 3.2, given enough surrogate models, one can further use some of them for validating adversarial examples. For those adversarial examples generated by the *Multi 8* strategy, we keep them only if they can be transferred to at least 5 of the 7 validation models, hence

we refer to this strategy *Multi 8 & Passing 5/7 Validation*. The remaining model is used as the attack target, and we measure the transferability of examples that passed the *5/7 Validation*. For this attack strategy, the measurements shown in Fig. 5.3 is the average of 8 rotations between target model and validation models. Comparing to *Multi 8* and *Single*, adversarial examples that passed the *5/7 Validation* are significantly more likely to transfer to the target model, even when the amount of perturbation is small.

This shows that simple strategies like *Single* are indeed not realizing the full potential of a resourceful attacker, and our proposed attack strategy of using multiple models for the generation and validation of adversarial examples is indeed superior. In the reset of this section, we will be using the most effective attack strategy of *Multi 8 & Passing 5/7 Validation*.

5.5 Translucent-box Evaluation

Here we evaluate the effectiveness of different schemes based on the transferability of adversarial examples generated using the *Multi 8 & Passing 5/7 Validation* attack strategy.

The results of our translucent-box evaluation are shown in Fig. 5.4. Adversarial examples are grouped into buckets based on their L_2 distance. For each bucket, we use grayscale to indicate the average validation passing rate for each scheme. Passing rate from 0% to 100% are mapped to pixel value from 0 to 255 in a linear scale. There are four rows, each correspond to adversarial examples with a certain L_2 range. Each column illustrates to what extent a target defense scheme resist adversarial examples generated from attacking different methods.

Examining the columns for Standard and Dropout, we can see that Standard and Dropout are in general most vulnerable. Distillation and RC are almost equally vulnerable. Magnet can often resist adversarial examples generated by targeting other defenses, but are vulnerable to ones generated specifically targeting it.

Overall, across the three datasets, RS-1-ADV performs the best, and is significantly better than Dropout-Adv. This suggests that Random Spiking offers additional protection against adversarial examples. RS-1 and RSD-1 also perform consistently well across the three

datasets. RC performs noticeably well on MNIST and Fashion-MNIST, likely because the images were all in 8-bit grayscale, and its advantages diminish on CIFAR-10 which contains images of 24-bit color.

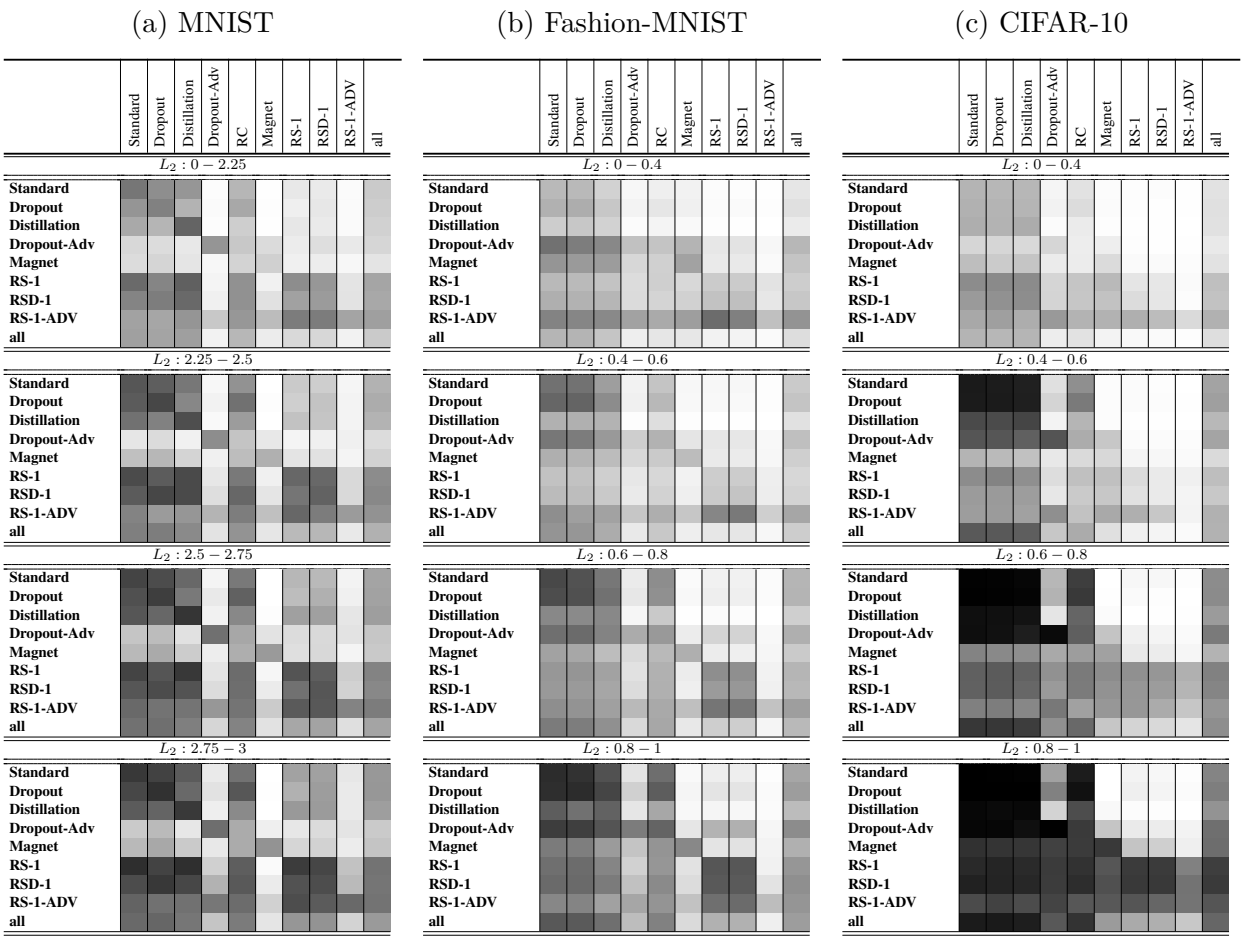
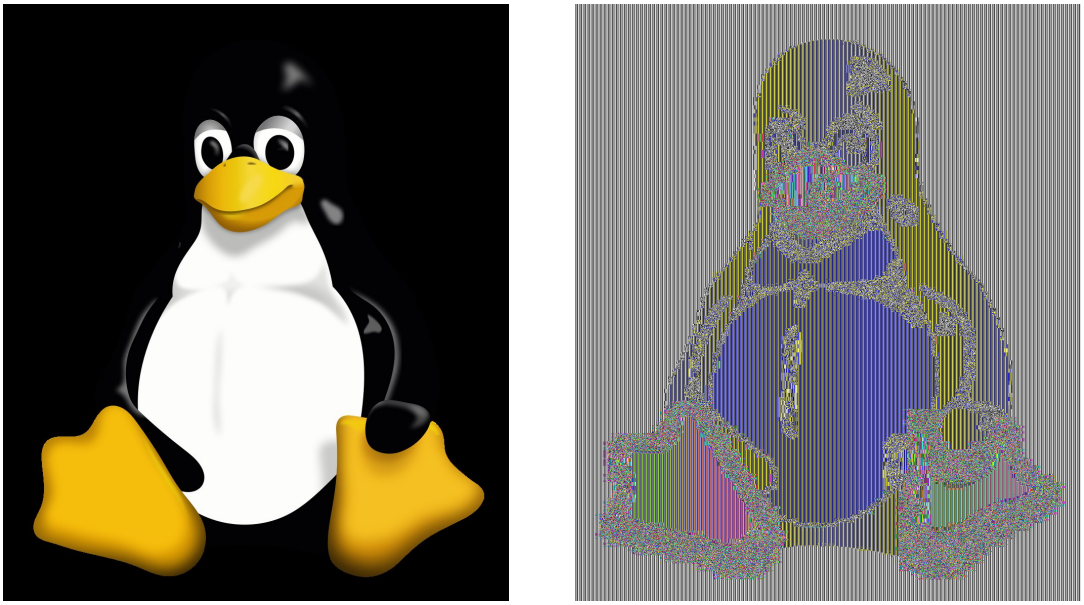


Figure 5.4. Average passing rate of 5/7 validation
This heat map shows the resilience of each scheme against the adversarial examples generated from all schemes, under a fixed allowance of L_2. Each column illustrates to what extent a target defense scheme resists adversarial examples generated from attacking different methods.

6. MONET: IMPRESSIONISM AS A DEFENSE AGAINST ADVERSARIAL EXAMPLES

In Chapter 4, we propose a new defense mechanism Random Spiking, which generalizes dropout and introduces random noises in the training process in a controlled manner. Random Spiking shows its effectiveness against a translucent-box attacker by injecting the randomness noise in the early layer of the network. When defending against a translucent-box attacker, it is natural to take advantage of randomness that is unknown to the adversary. We note that security of cryptographic schemes generally depends on randomness that is unknown to the adversary. In this chapter, we explore the idea of using **secret randomness** to defend against the transfer of adversarial examples. Security of encryption is based on a secret key that can be randomly chosen and kept secret.



(a) (b)

Figure 6.1. Original image vs Encrypted output of AES-ECB

6.1 Intuition

Consider the images shown in Fig. 6.1a, which is an example used in some textbooks on cryptography. Fig. 6.1b is the result of directly encrypting the image of Fig. 6.1a using

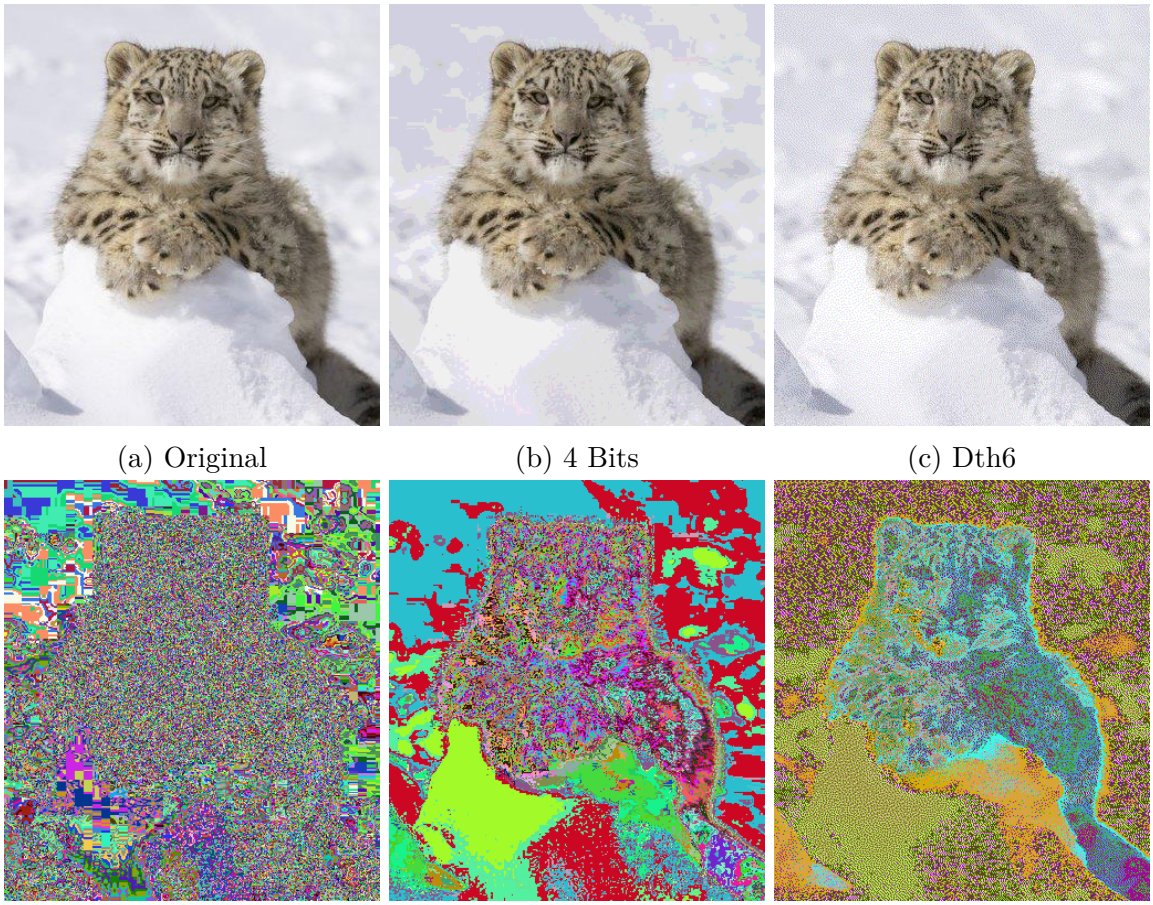
AES under the Electronic Code Book (ECB) mode. As can be seen, while the original color information of the penguin has been altered significantly, the overall outline of the penguin is still discernible in the encrypted version. This was used to illustrate that encryption using the ECB mode is insecure, because the same plaintext block will be encrypted to the same ciphertext block. This property, however, suggests that the "transformed" (or "encrypted") image can still be used to train classifiers.

6.2 Reducing color depth and quantization errors

While the example in Fig. 6.1 shows that a direct AES-ECB encryption can produce an intelligible ciphertext for images without fine-grained color information, this is not true in general. A direct application of AES-ECB to photographs, which are commonly used as training data and test inputs for classifiers, would often lead to unintelligible outputs. This is because typical photographs have a color depth of 24 bits, which can deliver high precision gradients of colors. Encrypt each pixel individually would have a better chance at preserving visible patterns in the image. However, the output quality would still be poor if the input has rich color depth.

An example of this can be found in Fig. 6.2, which is based on the photo of a snow leopard from the IMAGENET dataset. Here we apply the transformation to each pixel separately. Since each pixel in the bitmap has 24 bits of color level information, and the block size of AES-128 is 16 bytes, we add bits of zeros to the input as padding, and truncate the cipher output to keep only the first 24 bits, so that it can be plugged back to the bitmap as a pixel. As can be seen from Fig. 6.2a and 6.2d, even this more fine-grained transformation renders the image unintelligible. The fur of the snow leopard is completely replaced by pseudo-random noises, and the only discernible patterns remain are the blocks of neighboring pixels that are exactly the same in the background. This makes the transformed image hardly usable for an image classifier.

One possibility to deal with the challenge of fine-grained color levels is to reduce the color depth of the image. This can be easily implemented as setting the least significant bits of the value of each color channel to zero. Fig. 6.2b shows an example of this, with half of



(a) Original (b) 4 Bits (c) Dth6

(d) Orig AES (e) 4 Bits AES (f) Dth6-AES

Figure 6.2. IMAGENET Snow Leopard

the bits for each color channel being set to zero. While this helps to improve the overall intelligibility of the transformed image as shown in Fig. 6.2e, the snow leopard itself is still not very recognizable, and the background exhibits dramatic splashes of color blocks, due to the problem of color banding in the depth reduced image.

Color banding introduces inaccurate color presentation, and is a form of quantization error. To resolve color banding, we apply dithering, which intentionally adds noises to randomize quantization errors. Dithering is able to accurately present an image with a limited color palette. For those colors not available in the color palette, we choose the closest color with the palette by a diffusion of colored pixels. Floyd-Steinberg dithering [7] is an error diffusion technique to achieve the dithering effect by quantizing the pixel color

within the available palette and diffusing the error to the neighboring pixels. The algorithm scans the pixel from left to right, top to bottom, and quantizing pixel values one by one, for each color channel, and then diffusing the error to the neighboring pixels by multiplying the diffusion coefficient defined in the following matrix.

$$\begin{bmatrix} & & * & \frac{7}{16} & \cdots \\ \cdots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \cdots \end{bmatrix} \tag{6.1}$$

The star (*) pixel in Eq. 6.1 indicates the current scanned pixel, and the blank pixels indicates the previously-scanned pixels. For the scanned pixel, the algorithm rounds up the value of each of its color channels to the closest value in $\{0, 51, 102, 153, 204, 255\}$ (6 shades of the historic "web-safe" colors). In the end, the dithered image (Fig. 6.2c) uses at most $6^3 = 216$ colors, which is approximately $\log_2 6 \approx 2.58$ bits per channel. Comparing to the 4-bit depth-reduced image (Fig. 6.2b), this further reduces the bit depth but can more accurately present the image, because human eyes perceive diffusion as a mixture of the colors within it. As the result, Fig. 6.2f, an AES-ECB encrypted dithered image, intelligibly present the shape of the snow leopard. In Fig. 6.3, we present other selected IMAGENET examples, and MoNet (last two columns) consistently and intelligibly present the semantic information of the image.

6.3 The MoNet Approach

We specifically propose the following two approaches to be used for MoNet.

6.3.1 Dth$n$

We preprocess the image with Floyd-Steinberg dithering before feeding the image into the DNN. n refers to the number of discrete value chosen within each color channel, where the chosen values form an arithmetic sequence ranging from $[0, 255]$. **Dth**n uses dithered images in both training and testing. **Dth**n**-N** uses dithered images in training but not testing, that is, the network is trained using dithered images, but classifies non-dithered images.

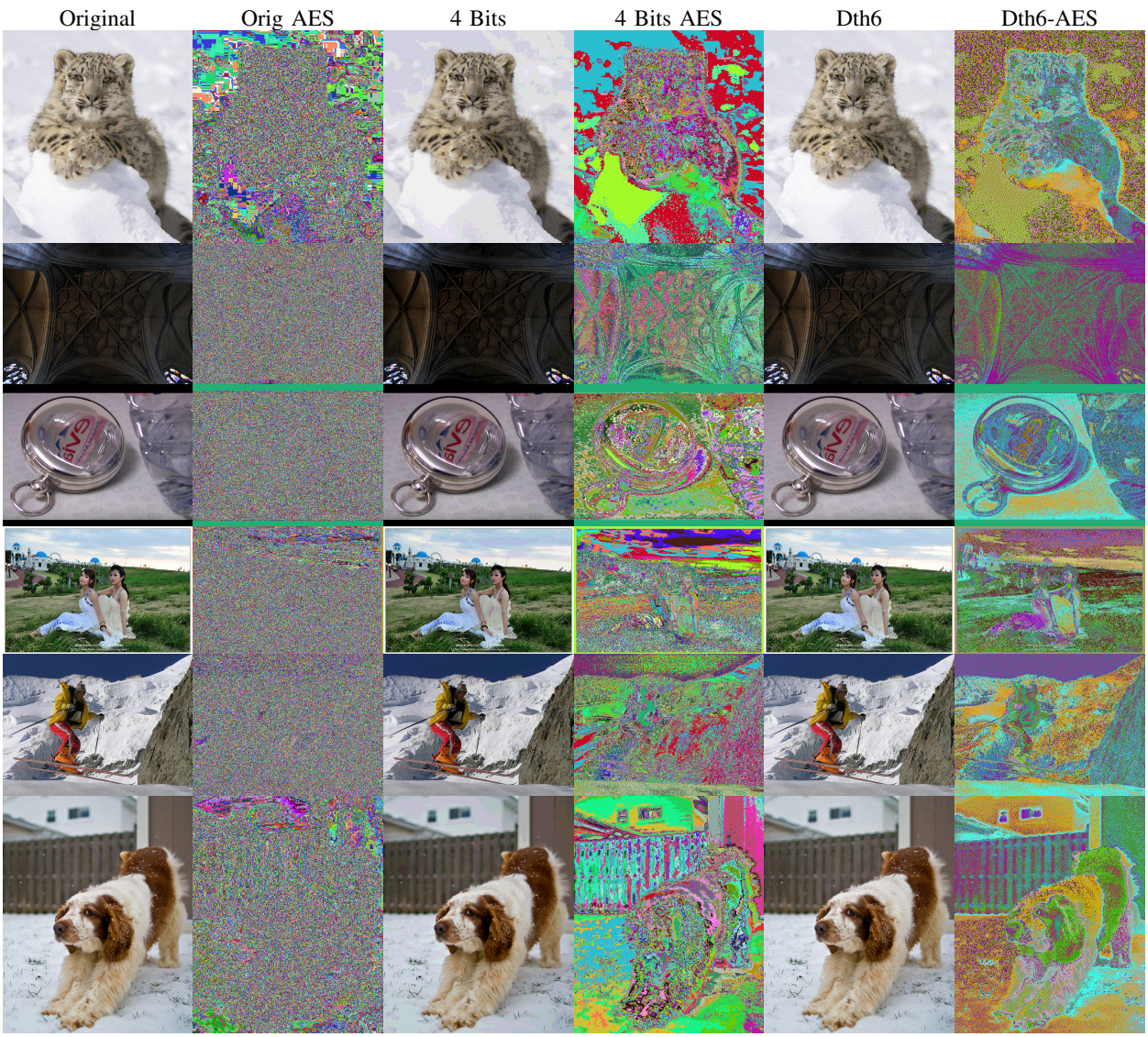


Figure 6.3. IMAGENET MoNet Examples

6.3.2 Dth$n$-AES

We apply the AES encryption on dithered image before using the image as input the CNN. For each pixel in the dithered image, we group the color value from three channels and then append zero to fill an AES plaintext block. Then, we apply the AES block cipher to the plaintext and use the first three bytes of the result as the pixel value. **Dth$n$-AES** requires the use of encrypted dithered image in both training and testing.

7. EVALUATION OF MONET

We evaluate and compare MoNet with Random Spiking and other existing defensive mechanism by using the systematic evaluation methodology presented in Chapter 3.

7.1 Dataset and Model Training

For our experiments, we use the following 2 datasets: CIFAR-10 [60] and IMAGENET [66]. Table 7.1 gives an overview of their characteristics.

Table 7.1. Overview of datasets

Dataset	Image size	Training Instances	Test Instances	Color space
CIFAR-10	32×32	50,000	10,000	24-bits True-Color
IMAGENET	Clip to 299×299	1,281,167	50,000	24-bits True-Color

For CIFAR-10, we use the state-of-the-art WRN-28-10 instantiation of the wide residual networks [61]. We consider 13 schemes equipped with different defense mechanisms, all of which share the same network architectures and training parameters. For 9 of 13 schemes, we reuse the training parameter in 5.1. The remaining 4 schemes include FS [5], and the 3 schemes from our MoNet proposal: Dth6, Dth6-N, and Dth6-AES. We choose $n = 6$ for MoNet since model trained with $n = 6$ delivers better testing accuracy and similar model stability comparing with $n = 4$ or 5. Some defensive mechanisms have adjustable parameters, and we choose values for these parameters so that the resulting models have a comparable level of accuracy on the test instances. As the result, all 13 schemes result in a small accuracy drop. Table 7.2 gives the test errors for CIFAR-10. The numbers in the parentheses for MagNet and FS are the detection threshold (rate of false positive) for detecting adversarial examples. The number in the parentheses for RC is the L_0 bound for adding random noise. Recall the evaluation of Random Spiking in Sec 5.1, when a scheme uses either Dropout or Random Spiking, we use MC Avg to make a prediction, which typically achieves a slight reduction in test error. By "MC Avg.", we mean using Definition 2 by running the network with Dropout and/or Random Spiking 10 times, and averaging the 10 probability vectors.

For each scheme, we train 16 models (with different initial parameter values), and report the mean and standard deviation of their test accuracy. We observe that Dth6 scheme achieves similar test error comparing to other defensive schemes, while Dth6-AES incurs an additional accuracy drop of 1.2%.

Table 7.2. Cifar Test Errors (mean±std).

Standard $4.38 \pm 0.21\%$	Dropout $4.46 \pm 0.25\%$	Distillation $4.33 \pm 0.27\%$	RS-1 $5.59 \pm 0.22\%$
RS-1-Dropout $5.81 \pm 0.27\%$	RS-1-Adv $6.20 \pm 0.40\%$	Magnet (0.004) $5.52 \pm 0.24\%$	Dropout-Adv $4.71 \pm 0.19\%$
RC (0.025) $6.52 \pm 0.58\%$	FS (0.008) $6.54 \pm 0.62\%$	Dth6 $6.63 \pm 0.21\%$	Dth6-N 6.33 ± 0.23
Dth6-AES $7.87 \pm 0.35\%$			

Table 7.3. ImageNet Test Error (mean±std), image size 299×299

	Method	Top-1 Err	Top-5 Error
Inception-V3	Standard	$22.04 \pm 0.09\%$	$5.99 \pm 0.08\%$
	Dth6-N	$23.45 \pm 0.14\%$	$6.74 \pm 0.07\%$
	Dth6	$22.55 \pm 0.07\%$	$6.30 \pm 0.06\%$
	Dth6-AES	$24.60 \pm 0.17\%$	$7.44 \pm 0.09\%$
	FS-0.014	$23.56 \pm 0.15\%$	$7.55 \pm 0.12\%$
ResNet-50	Standard	$22.59 \pm 0.11\%$	$6.23 \pm 0.07\%$
	Dth6-N	$24.58 \pm 0.11\%$	$7.31 \pm 0.08\%$
	Dth6	$23.09 \pm 0.01\%$	$6.53 \pm 0.07\%$
	Dth6-AES	$25.93 \pm 0.16\%$	$8.15 \pm 0.12\%$
	FS-0.022	$24.71 \pm 0.08\%$	$8.45 \pm 0.13\%$

For IMAGENET, we use two state-of-the-art model architectures: Inception-V3 [67] and ResNet-50 [23]. For each network architecture, we consider 5 schemes equipped with different defense mechanisms, all of which share the same training parameters. Table 7.3 gives the test errors on IMAGENET. For each architecture under each scheme, we train 16 models (with different initial parameter values), and report the mean and standard deviation of their test accuracy. We observe that all network architecture achieve a Top-5 test error

Table 7.4. C&W targeted Adv Examples $L2$ (mean±std).

CIFAR-10	Single	IMAGENET	Single
Standard	0.17 ± 0.08	Inception-V3	0.58 ± 0.27
Dropout	0.17 ± 0.08	Inception-V3-FS (0.014)	0.58 ± 0.27
Distillation	0.17 ± 0.07	Inception-V3-Dth6	1.51 ± 0.44
RS-1	0.32 ± 0.14	ResNet-50	0.64 ± 0.30
RS-1-Dropout	0.32 ± 0.15	ResNet-50-FS (0.022)	0.63 ± 0.29
RS-1-Adv	0.39 ± 0.18	ResNet-50-Dth6	1.49 ± 0.42
Magnet	0.29 ± 0.21		
Dropout-Adv	0.18 ± 0.07		
FS (0.008)	0.52 ± 0.23		
Dth6	0.45 ± 0.20		

of approximately 6% by using the standard training method. Using dithered images in testing (Dth6) achieves lower test error if we use dithered images in training. Dth6 has a 0.3% accuracy drop comparing to standard training. Dth6-AES has an additional 1.1-1.6% accuracy drop depending on the network architecture.

7.2 White-box Evaluation

We first evaluate the effectiveness of the defense mechanisms under white-box attacks.

7.2.1 C&W L_2 White-box Evaluation

We apply the C&W white-box attack with a confidence value of 0 to generate targeted adversarial examples, and measure the L_2 distance of the generated adversarial examples. We consider single-model attacks, where the adversarial example targets a single model.

For CIFAR-10, the left two columns of Table 7.4 present the average L_2 distances of the generated examples for those generated adversarial examples. Dth6 (gray rows) results in models that are more difficult to attack, requiring on average the highest amount of perturbations (measured in L_2 distance) among all evaluated defenses except for FS (0.008). For FS, since adversarial examples need to bypass multiple FS detectors in order to work, it

can be considered as attacking multiple models, which requires more noises to be added to the generated adversarial examples.

For IMAGENET, the right two columns of Table 7.4 present the average L_2 distances of the generated examples for those generated adversarial examples. Training with Dth6 (gray rows) results in models that are more difficult to attack, requiring on average the highest amount of perturbations (measured in L_2 distance) among all the evaluated schemes.

7.2.2 PGD L_∞ White-box Evaluation

We apply the PGD L_∞ white-box attack to generate targeted adversarial examples, and measure the bounded L_∞ distance of the generated adversarial examples against the attack success rate. We consider single-model attacks, where the adversarial examples target a single model.

Table 7.5. PGD targeted Adv Examples L_∞ VS attack success rate (mean±std) on CIFAR-10 dataset using **Single** attack strategy.

L_∞	1	2	3	4	5	6	7	8
Standard	13.40 ± 0.88	42.47 ± 2.45	68.08 ± 2.34	84.13 ± 2.19	92.88 ± 1.94	97.01 ± 1.50	98.87 ± 0.85	99.60 ± 0.45
Dropout	12.18 ± 0.62	41.38 ± 1.69	69.74 ± 1.95	87.38 ± 2.25	95.94 ± 1.56	98.83 ± 0.84	99.67 ± 0.31	99.89 ± 0.12
Distillation	10.55 ± 2.11	36.14 ± 4.85	62.47 ± 5.87	81.13 ± 4.85	91.81 ± 3.14	96.94 ± 1.72	98.83 ± 0.98	99.55 ± 0.54
Dropout-Adv	10.42 ± 1.06	36.69 ± 2.93	66.46 ± 3.37	87.21 ± 3.24	96.38 ± 1.80	99.09 ± 0.71	99.76 ± 0.26	99.94 ± 0.09
RS-1	4.06 ± 0.35	17.78 ± 2.02	43.68 ± 3.90	68.41 ± 4.11	84.61 ± 3.42	93.31 ± 2.30	97.25 ± 1.40	98.89 ± 0.72
RS-1-Dropout	3.65 ± 0.44	14.02 ± 2.12	36.48 ± 4.68	61.39 ± 4.52	79.42 ± 3.54	90.22 ± 2.35	95.80 ± 1.40	98.32 ± 0.75
RS-1-Adv	3.13 ± 0.36	10.86 ± 1.74	28.93 ± 5.00	53.25 ± 7.34	74.29 ± 6.79	88.13 ± 4.98	95.10 ± 2.73	98.11 ± 1.33
Dth6	2.94 ± 0.13	9.64 ± 0.30	24.27 ± 1.00	45.03 ± 1.21	63.73 ± 0.88	76.40 ± 0.64	83.67 ± 0.59	87.35 ± 0.63

For CIFAR-10, Table 7.5 presents the average attack success rate and its standard deviation under bounded L_∞. Dth6 has the lowest attack success rate among all defense methods. When $L_\infty = 8$, Dth6 has an attack success rate of only 87.4%, while the other defensive schemes all have an attack success rate of more than 98%.

For IMAGENET, Table 7.6 presents the average attack success rate and its standard deviation under bounded L_∞. When $L_\infty = 1$, models trained with Dth6 (gray rows) have a much lower attack success rate ($< 14\%$) than the others (attack success rate $> 80\%$).

Table 7.6. PGD targeted Adv Examples L_∞ VS attack success rate (mean±std) on ImageNet dataset using **Single** attack strategy.

L_∞	1	2	3	4	5	6	7	8
Inception-V3	87.29 ± 0.67	99.76 ± 0.10	99.95 ± 0.07	99.95 ± 0.07	99.98 ± 0.04	99.99 ± 0.03	99.99 ± 0.03	99.98 ± 0.07
Inception-V3-Dth6	13.44 ± 0.38	64.69 ± 0.78	77.75 ± 0.76	82.16 ± 0.42	84.51 ± 0.49	85.69 ± 0.45	86.61 ± 0.24	87.08 ± 0.37
ResNet-50	81.97 ± 0.72	99.55 ± 0.12	99.98 ± 0.04	100.00 ± 0.00	100.00 ± 0.00	99.99 ± 0.03	99.96 ± 0.05	99.98 ± 0.04
ResNet-50-Dth6	12.46 ± 0.73	64.66 ± 0.81	79.84 ± 0.55	84.30 ± 0.65	86.60 ± 0.44	87.85 ± 0.29	88.81 ± 0.24	89.86 ± 0.38

7.3 Model Stability

Given a benign image and its variants with added noise, a more robust model should intuitively be able to tolerate a higher level of noise without changing its prediction results. We refer to this property as model stability. Here we evaluate whether models of different defensive schemes can correctly label instances that are perturbed.

7.3.1 Stability with Added Gaussian Noise:

We measure how many predictions would change if a certain amount of Gaussian noise is introduced to a set of benign images. For a given dataset and a model, we use the first $1,000$ images from the test dataset. We first make a prediction on those selected images and store the results as *reference predictions*. Then, for each selected image and a chosen L_2 distance, we sample Gaussian noise, scale it to the desired L_2 value, and add the noise to the image. Pixel values are clipped if necessary, to make sure the new noisy variant remains a valid image. We repeat this process 20 times (noise sampled independently per iteration).

Fig. 7.1a shows the effect of Gaussian noise on prediction stability for each training method on CIFAR-10 (averaged over the 16 models trained in Sec. 7.1). Model stability inevitably drops for each scheme as the amount of Gaussian noise, measured by L_2, increases, though different schemes behave differently.

We see that Dth6, Dth6-N, and Dth6-AES have the highest stability as the amount of noise increases. When $L_2 = 2.5$, they have stability 89.9%, 93.1%, 89.7%, respectively. RS schemes have stability between 81-88%, and the other schemes have stability 70% or lower. Fig. 7.1a also plots the standard deviation of the stability result of the 16 models. Dth6,

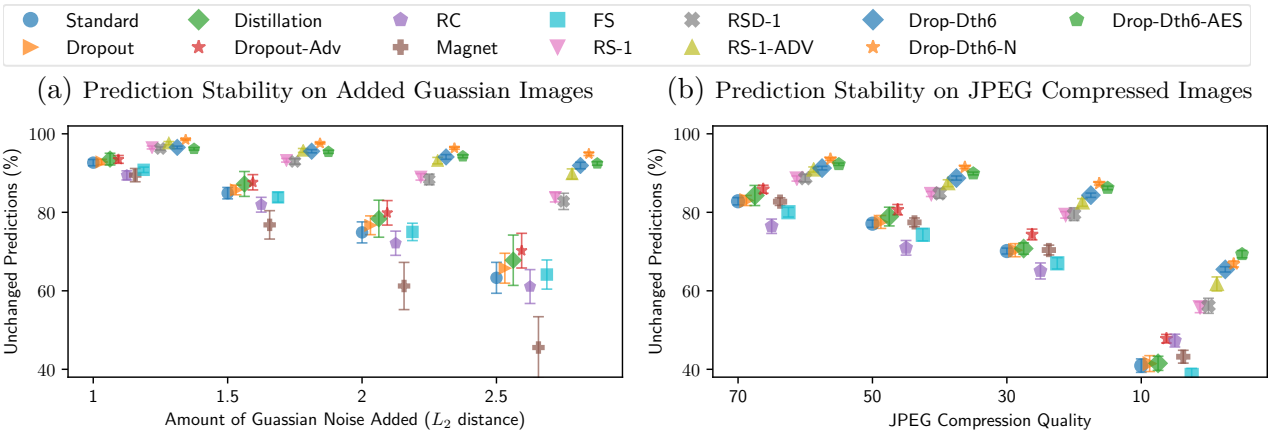


Figure 7.1. Evaluating CIFAR-10 model stability

Dth6-N, and Dth6-AES have very low standard deviation, which in turn also suggest a more consistent behavior when facing perturbed images.

7.3.2 Stability with JPEG compression:

Given a set of benign images, we measure how many predictions would change if JPEG compression is applied to images. For a given test dataset and a model, we compare the prediction on the benign test dataset (*reference predictions*) with the prediction on JPEG compressed test dataset under a chosen JPEG compression quality (**JCQ**). For the sake of time efficiency, for this particular set of experiments, we reduced the number of iterations used by RC to one-tenth of its original algorithm.

For CIFAR-10, Fig. 7.1b shows the effect of JPEG compression on prediction stability for each training method (averaged over the 16 models trained). As JCQ (ranges $70 - 10$) decreases, model stability decreases for all schemes, but we we see that Dth6, Dth6-N, and Dth6-AES have the highest prediction stability. When JCQ $= 10$, they have a stability of 64.6%, 65.9%, and 67.8%, respectively. The other schemes have a stability of $\leq 61\%$, and the most stable among them is RS-1-ADV.

For IMAGENET, Fig. 7.2 shows the effect of JPEG compression on prediction stability for each training method and model architecture (averaged over the 16 models trained).

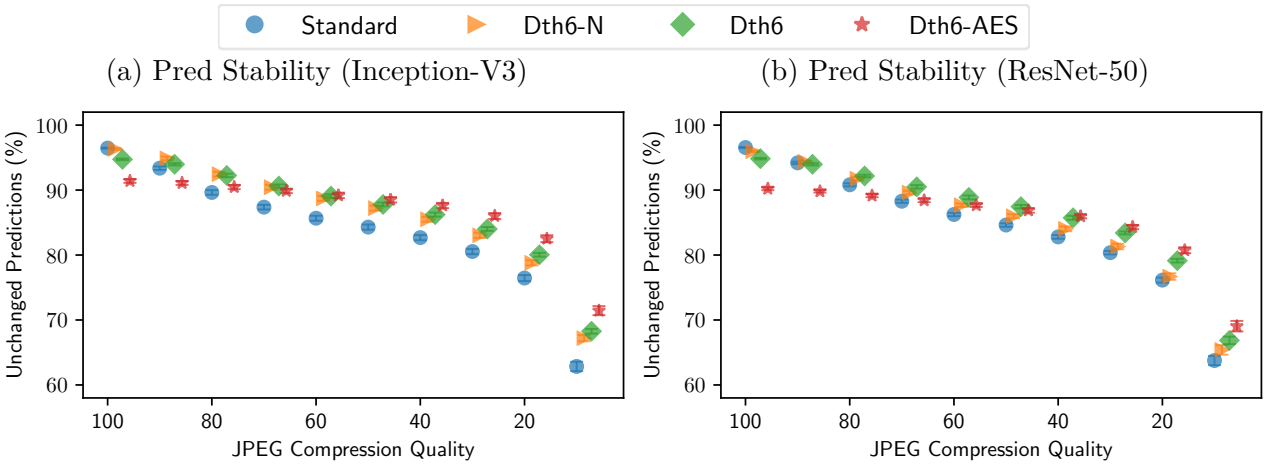


Figure 7.2. Evaluating IMAGENET model stability with JPEG compression

Model stability decreases for each scheme as the JCQ (ranges $100 - 10$) decreases. We see that although Dth6-AES has a lower model stability when JCQ $= 100$, it has a much slower drop in model stability as JCQ decreases. When JCQ ≤ 40, Dth6-AES has the highest model stability among all training methods, and this result is consistent across all model architectures.

7.4 Translucent-box Evaluation

Here we evaluate the effectiveness of different schemes based on the transferability of adversarial examples generated using the *Multi 8 & Passing 5/7 Validation* attack strategy presented in Chapter 5.4 . For each scheme, we train 16 surrogate models, each with half of the dataset randomly selected and randomly initialized model parameters. We use 8 of the 16 surrogate models for generating adversarial examples. To fool all 8 surrogate models, we use the sum of the loss functions of the 8 surrogate models as the new loss function to generate adversarial examples. To improve transferability, experimental result in Chapter 5.4 suggests to use additional surrogate models for validating adversarial examples. We use 7 of the remaining 8 surrogate models as validation models, and the last model as the targeted model to evaluate transferability. For those adversarial examples which can

fool all 8 surrogate models, we keep them only if they can be transferred to at least 5 of the 7 validation models. We measure the average of 8 rotations between target model and validation models.

For CIFAR-10, the results of our translucent-box evaluation are shown in Fig. 7.3. Adversarial examples are grouped into buckets based on their L_2 and L_∞ distance for adversarial examples generated by the C&W attack (Fig. 7.3a) and PGD attack (Fig. 7.3b) respectively. For each bucket, we use grayscale to indicate the average validation passing rate for each scheme. Passing rate from 0% to 100% are mapped to pixel value from 0 to 255 in a linear scale. There are four rows, each corresponds to adversarial examples within a certain L_p range. Each column illustrates to what extent a target defense scheme resist adversarial examples generated from attacking different methods. Comparing to non-dithered models, dithered models (column Dth6 and Dth6-N) achieve significantly lower transferability (close to 0) on adversarial examples excluding those generated on dithered model. Dithered models are vulnerable to adversarial examples targeting dithered models, but those are less likely to transfer to non-dithered models. AES-ECB models (column Dth6-AES) performs the best among all defense methods, which further reduce the transferability for adversarial examples generated on dithered models.

For IMAGENET, the results of our translucent-box evaluation are shown in Fig. 7.4. AES-ECB models (last 2nd and 3rd columns ending with AES) achieves near 0 transferability, which perform the best among all training methods and are consistent among different model architecture and attack algorithms.

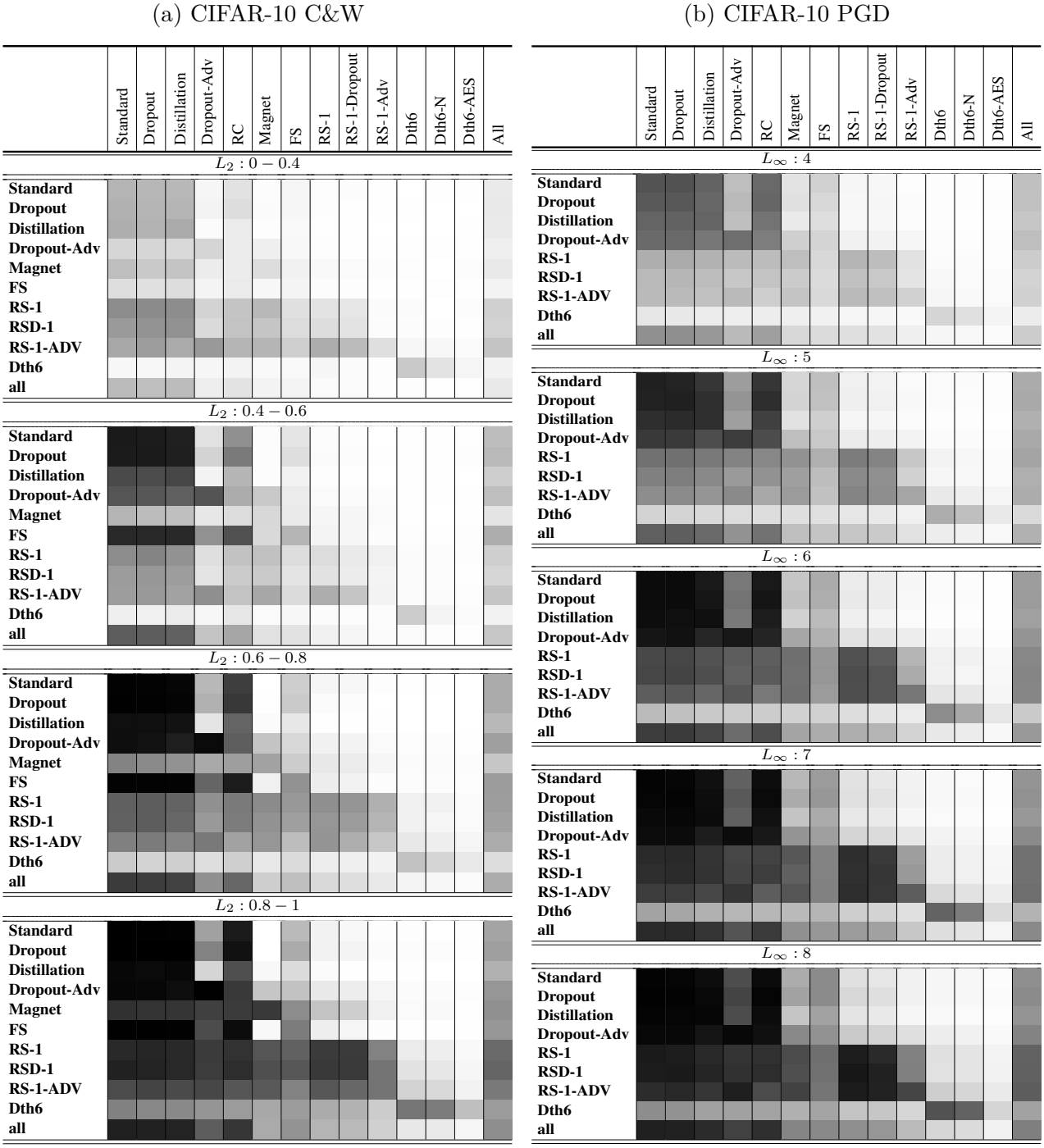


Figure 7.3. Average passing rate of 5/7 validation (CIFAR-10 dataset)
This heat map shows the resilience of each scheme against the adversarial examples generated from all schemes, under a fixed allowance of L_p. Each column illustrates to what extent a target defense scheme resists adversarial examples generated from attacking different methods.

(a) IMAGENET C&W

(b) IMAGENET PGD

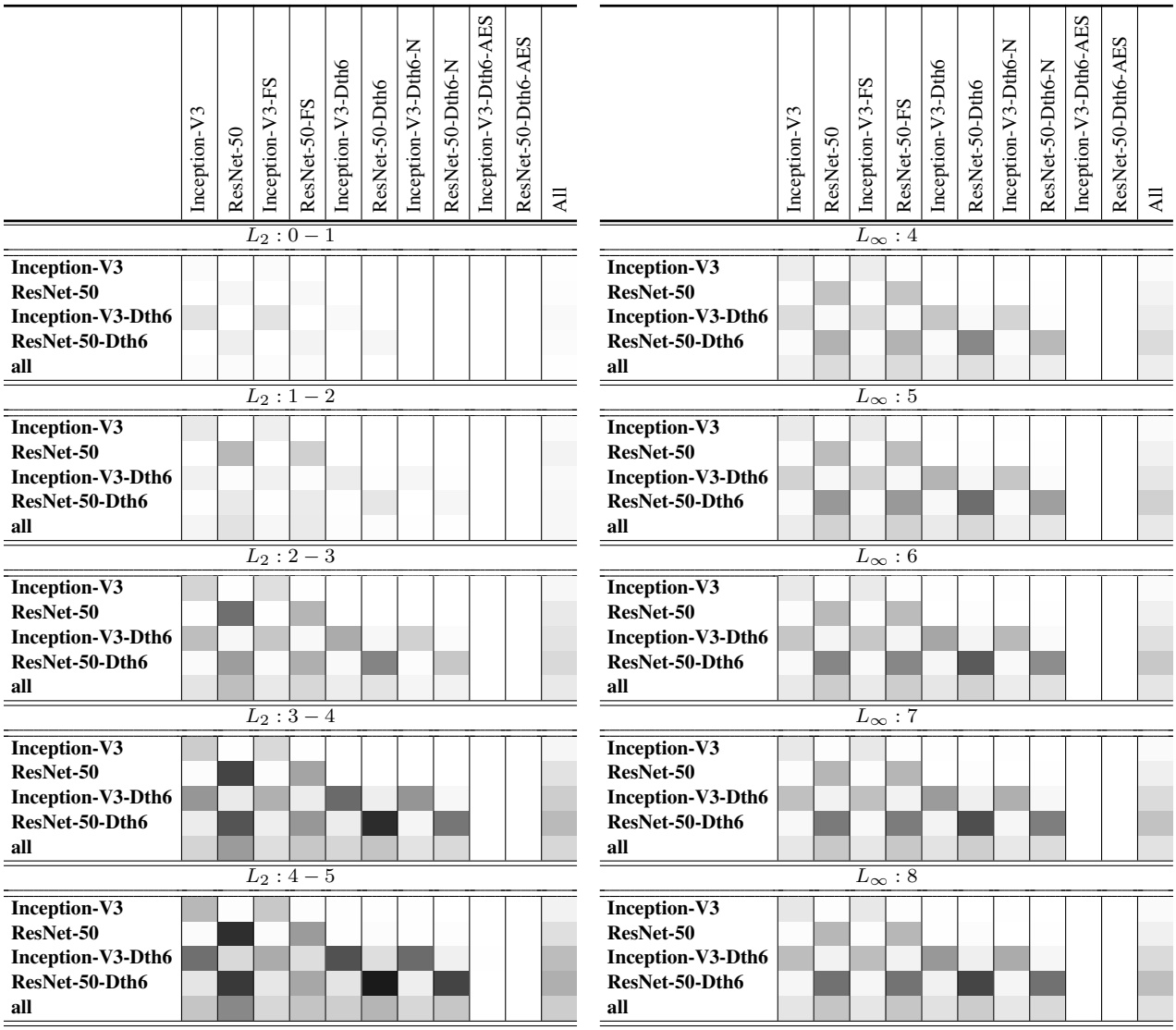


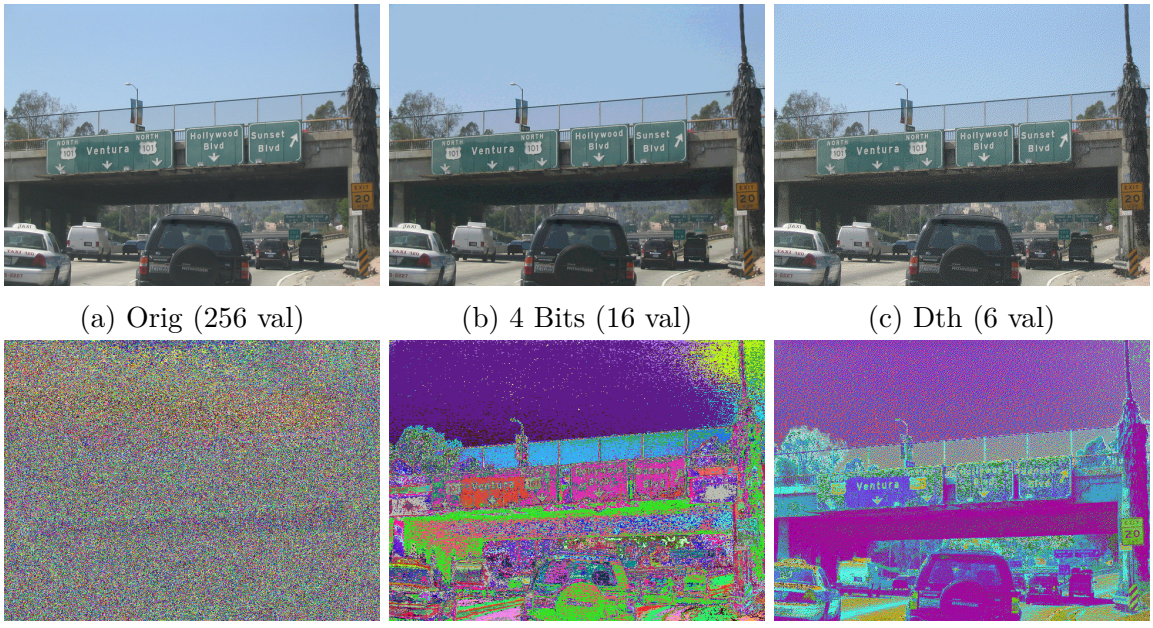
Figure 7.4. Average passing rate of 5/7 validation (IMAGENET dataset)
This heat map shows the resilience of each scheme against the adversarial examples generated from all schemes, under a fixed allowance of L_∞. Each column illustrates to what extent a target defense scheme resists adversarial examples generated from attacking different methods.

8. DEFEND AGAINST ADVERSARIAL EXAMPLES IN OBJECT DETECTION WITH MONET + MODEL ENSEMBLE + TTA

In Chapter 8, we consider the TOG adversarial attacks against object detection, and study different defense mechanisms. We first consider three existing defense methods, MoNet [39], Model Ensemble [26, 27], and Test Time Augmentation (TTA) [26], to reduce the transferability of adversarial examples for object detection. We also propose to combine these methods, which further reduces the transferability.

8.1 MoNet

In Chapter 7, we have shown that MoNet significantly reduces the transferability of adversarial examples for image classification network. In this dissertation, we study whether this property holds for the object detection network.



(a) Orig (256 val) (b) 4 Bits (16 val) (c) Dth (6 val)

(d) Orig AES (e) 4 Bits AES (f) Dth AES

Figure 8.1. MS-COCO-2017 Highway (number of discrete values used per channel)

We apply MoNet on MS-COCO-2017, a dataset for object detection, and compare the example with the original image, 4 bit depth reduction as well as their encrypted images (see Fig. 8.1). We found that Floyd-Steinberg dithering has to be used before applying AES encryption, because a direct application of AES produces unintelligible outputs, which is consistent with the result on image classification. Fig. 8.1d, the result of directly applying AES on Fig. 8.1a lost most of its intelligible information. Using such unintelligible images in training would give the object detector a very low testing accuracy. Although 4 bit depth reduction helps improve the intelligibility of encrypted image as shown in Fig 8.1e, the figure is not very recognizable for small cars. Comparing to two previous encrypted images, MoNet processed image as shown in Fig. 8.1f is significantly better in visualization as all cars and bridge can be clearly recognized. One can even read the text "`Ventura`" on the bridge. More examples of MoNet can be found in Fig. A.1 (See Appendix).

8.2 Model Ensemble

MoNet models and standard models behave differently, in the sense that adversarial examples generated against one kind of models do not transfer well to the other. We thus explore the possibilities of combining their prediction outputs to further improve the accuracy in the face of adversarial examples.

Model ensemble combines the prediction results from multiple models to produce a final output [43]. Ensemble methods have been widely used in machine learning (ML) applications to improve model accuracy and stability. For ML applications that generate a single output for each input (*e.g.* image classifiers), model ensemble can be viewed as model averaging [3, 68]. However, raw object detection generates 0, 1, or more predictions for a given input image. Therefore, simple model averaging is not directly applicable. Although there are methods that combine the output of detection models such as NMS (Non-Maximum Suppression) [69], Soft-NMS [70], NMW [71], and fusion [72], these methods are mainly to eliminate redundant bounding boxes and do not fully utilize the given information such as the label of each detected bounding box, the number of models that can be grouped by overlapped bounding boxes, and the labels. A better ensemble approach can use these dif-

ferent kinds of information to extract the relationship between prediction outputs so that it can make a better detection on bounding boxes and avoid false positives, and thus deliver a higher prediction accuracy on the validation dataset. In this dissertation, we consider two model ensemble approaches that fully utilize the information from multiple predictions: voting and Weighted Boxes Fusion (WBF).

8.2.1 Ensemble Object detection by Voting

Casado-Garcia and Heras proposed a four-step, voting-based approach to ensemble the prediction outputs of multiple models [26]. The input of the algorithm is a list $LD = \{D_1, \ldots, D_m\}$ where each D_i is a list of detections produced by model M_i (defined in Section 2.3), $i \in \{1, \ldots, m\}$. Given the input LD, the algorithm first flattens LD to a list of detections defined as $F = \{d_1, \ldots, d_k\}$. Then, detections $d_i \in F$ are grouped into clusters based on the overlapping of bounding boxes and their category. More specifically, this step outputs a list $G = \{D_1^G, \ldots, D_m^G\}$ where each D_i^G is a list of detections such that any pairs $\bar{d} = \{\bar{b}, \bar{c}, \bar{s}\}$, $\hat{d} = \{\hat{b}, \hat{c}, \hat{s}\} \in D_i^G$ have the same category $\bar{c} = \hat{c}$ and the overlapping of two bounding boxes, formally called the interaction of union (IoU), $IoU(\bar{b}, \hat{b}) > 0.5$, where

$$IoU(b_1, b_2) = \frac{area(b_1 \cap b_2)}{area(b_1 \cup b_2)}$$

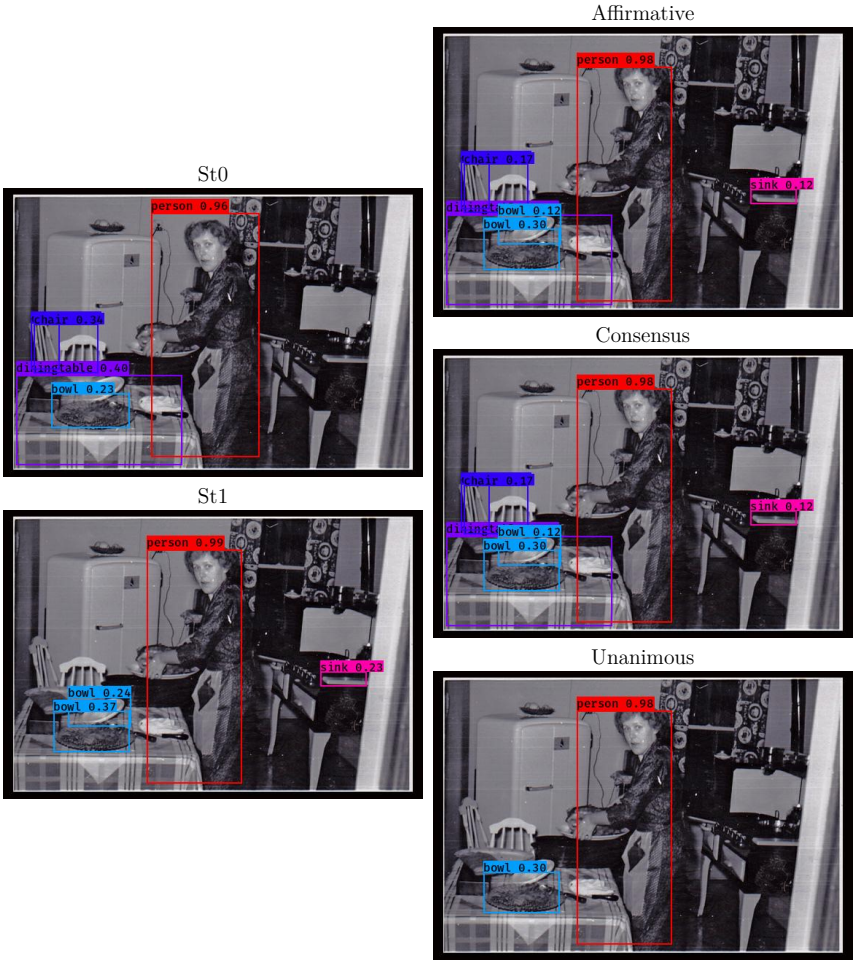
In short, we have a list of clusters G, where each cluster D_i^G is a list of detections $d_{i,j}^G$ that focus on the same region and predict the same category.

The third step is to vote whether the cluster will be included in the final prediction and outputs a valid list $G' \subseteq G$. Notice that each list $D_k^G \in G$ has at most m detections if the input is given by m models. The original paper considers $D_k^G \in G'$ is valid if D_k^G has at least m' elements given total $m/2$. Therefore, the original paper proposes three voting strategies:

- Affirmative implies at least one model produces the prediction ($m' \geq 1$).
- Consensus implies voting by the majority ($m' \geq m/2$).
- Unanimous implies all models have to agree to the prediction ($m' = m$).

The last step of the algorithm is to output the most appropriate bounding boxes among G' and their confidence score. We compute the non-maximum suppression (NMS) [69] for each valid $D_k^{G'}$ as NMS serves best for eliminating redundant bounding boxes for this case. The corresponding confidence score of $D_k^{G'}$ is the average of all elements in $D_k^{G'}$.

Table 8.1. Visualization of Predictions from Two Benign Models St0 & St1 (left) Vs. Voting Based Model Ensemble Strategies (right).



Regarding model ensemble strategies, the original paper suggests that the affirmative strategy performs better than two other strategies if detection models produce fewer numbers of false positives compared to the number of false negatives. For example, Table 8.1 shows Affirmative and Consensus include the predicted detections which are missed in one benign model and thus discarded by the Unanimous strategy. This suggests that the Affirmative and Consensus strategy can better utilize the prediction output from multiple models to

improve prediction accuracy, whereas the Unanimous strategy tends to worsen the prediction accuracy. In our experiments (see Sec. 9.1), we found that the affirmative strategy generally performs best on the validation dataset among the three strategies.

8.2.2 Ensemble Object detection by Weighted Boxes Fusion (WBF)

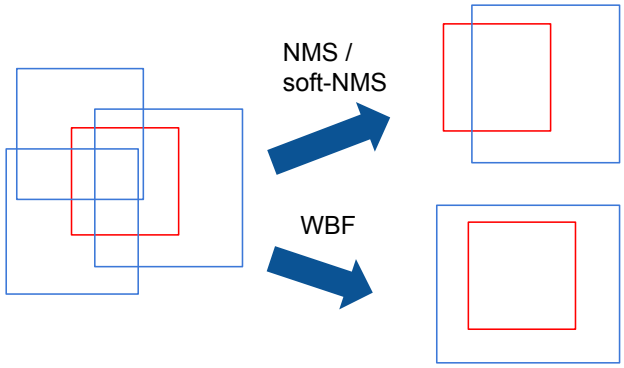


Figure 8.2. NMS/soft-NMS vs. WBF outputs. model predictions (blue), the ground truth (red)

While both NMS [69] and soft-NMS [70] exclude redundant bounding boxes and require that the final box must be among detections obtained from the model, Weighted Boxes Fusion (WBF) [27] uses information of all boxes to generate more accurate boxes. Therefore, WBF could find boxes that are closer to the ground truth even though all predicted boxes are inaccurate.

Similar to the model ensemble approach described in Section 8.2.1, WBF first clusters all detections and generates $G = \{D_1^G, \ldots, D_m^G\}$ where each detection in the cluster D_i^G has the same category and the box has significant overlaps with other boxes in the cluster (Note: WBF changes the overlap from $IoU > 0.5$ to $IoU > 0.55$). Then, for each cluster D_i^G, WBF computes the confidence score, box coordinates with the following formulas:

$$s = \frac{\sum_{j=1}^{T} s_j}{T} \times \frac{\min(T, M)}{M},$$

$$(X_1, X_2, Y_1, Y_2) = \frac{\sum_{j=1}^{T} s_j \times (X_{j1}, X_{j2}, Y_{j1}, Y_{j2})}{\sum_{j=1}^{T} s_i}$$

where the cluster D_i^G has T valid detections obtained from M model outputs and weight s_j is the confidence score from the j-th detection.

The main difference between the voting and WBF is that the voting approach chooses the most appropriate bounding boxes among all detections, whereas WBF uses a weighted function to compute new bounding boxes based on detections within the same cluster, (See Fig. 8.2). WBF tends to generate a bounding box that which is much closer to the ground truth. This suggests that WBF should perform better than Affirmative, which is confirmed by our evaluation. Despite the difference, both approaches use clustering to find the valid cluster for overlapping bounding boxes. This step can effectively eliminate false positives and redundant bounding boxes, thus leading to high prediction accuracy.

8.3 Test Time Augmentation

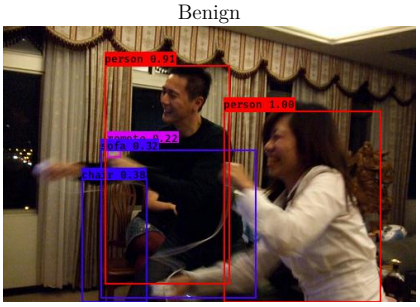
Although model ensemble can improve model accuracy and stability, the drawback is that it requires more resources to train different models from scratch. In addition, image augmentation such as flipping and changing the brightness and contrasts can change the model input. This can affect the prediction output, leading to model instability. To overcome these two issues, Casado-Garcia and Heras [26] further proposed Test Time Augmentation which can be used together with the model ensemble technique. The method first applies the model to both the original and augmented images (equalized and flipped), and then applies one of the model ensemble strategies described in Section 8.2.1. For example, Table 8.2 shows the model detects one remote controller in the woman's hands in the flipped image and three remote controllers in the equalized image (two in the woman's hands and one in the man's hands). However, all these detections are missed given the benign image. Therefore, using the Affirmative and WBF model ensemble strategy with TTA can immediately improve the model accuracy on existing models without training extra models from scratch.

8.4 Proposed Strategies

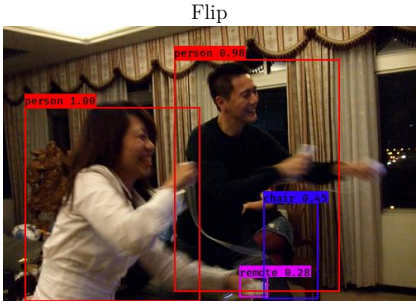
We consider several combinations of the ideas introduced so far to reduce the transferability of adversarial examples targeting object detection, and group them into four categories.

Table 8.2. Visualization of Prediction on Augmented Images (left) with. the Result of applying Model Ensemble Strategies (right).

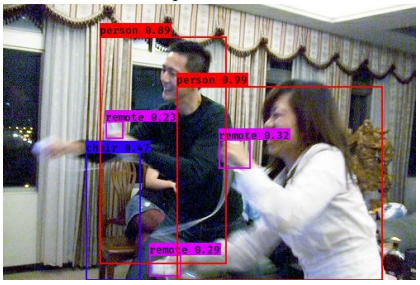
Benign



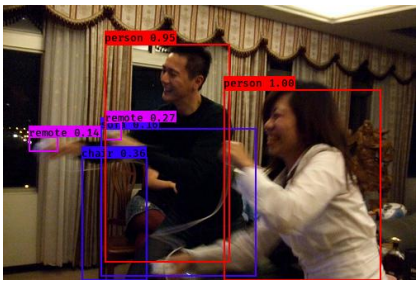
Flip



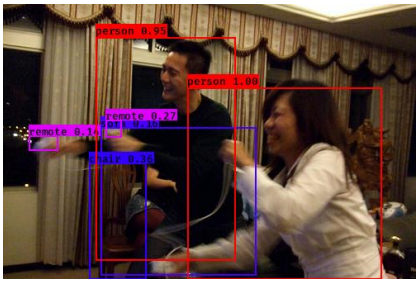
Equalization



Affirmative



WBF



8.4.1 Single models

We first consider training a single model without using TTA or model ensemble.

- **St** This is the baseline with no specific defense, and considers standard models trained with images without modification.
- **Dth** We preprocess the image with Floyd-Steinberg dithering and use such dithered images in both training and deployment.

- **AES** We apply the AES encryption to each pixel of the dithered image before feeding the image to a CNN. **AES** models requires the use of the encrypted dithered image in both training and testing.

8.4.2 Single Model with TTA

TTA generates additional input images. Using TTA with appropriate model ensemble methods such as Affirmative and WBF can immediately improve the model accuracy on existing models without training extra models. Therefore, we propose to apply TTA to the aforementioned single models.

8.4.3 Multi

Model ensemble is to improve model accuracy and stability as models usually behave slightly differently given the same input even though they are trained with the same procedure. This is because models are trained with different random initial parameters. Intuitively, model ensemble can defend against transfer attack if we have two or more models behaving differently on adversarial examples, but quite similar on benign examples. For example, one model is vulnerable to an adversarial example, but another is not vulnerable to the same example. Therefore, we propose to include at least two different types of models in an ensemble, resulting in 4 multi-model combinations: St + Dth, St + AES, Dth + AES, and 3Mix (St + Dth + AES).

8.4.4 Model Ensemble + TTA

As we mentioned in Sec. 8.3, model ensemble and TTA can be used together to improve the prediction accuracy of existing models without training extra models from scratch. We argue that such a strategy will perform even better than Model Ensemble strategy. Notice that TTA can generate multiple slightly different predictions as each input is slightly different but has the same semantic information. Therefore, one prediction has the potential of catching the prediction that the other model missed, which is the same reason we explained

in Sec. 8.3. Therefore, we propose to use Test Time Augmentation with 4 different Model Ensemble combinations described in Sec. 8.4.3.

9. EVALUATION OF MONET + MODEL ENSEMBLE + TTA TO DEFEND AGAINST ADVERSARIAL EXAMPLES FOR OBJECT DETETION

We present experimental results comparing proposed strategies described in Chapter 8 to defend against adversarial examples for object detection network.

9.1 Dataset and Model Training

For our experiments, we consider one-stage object detection network YoloV3 [13] and use ResNet-50 [23] as its backbone network. To train St, Dth, and AES models from scratch, we first duplicate the same training procedure in [73] to train the backbone network ResNet-50 on IMAGENET [66]. Then, we use an open source implementation[1] to train YoloV3 on MS-COCO-2017 [74] with a two-stage training procedure. The characteristics of training and testing datasets IMAGENET and MS-COCO-2017 are presented in Table 9.1.

Specifically, we train ResNet-50 with 90 epochs. All images are randomly clipped and resized to 416×416, and then sent into the network. The training procedure uses Stochastic Gradient Descent (SGD) as the optimizer. The reference learning rate is $\eta = 0.1$ per 256 mini-batch. To ensure the training is stable at the beginning, the learning rate starts with 0 and is linearly increased to the targeted reference learning rate over 5 epochs. Then, the learning rate is reduced by 1/10 at the 30-th, 60-th, and 80-th epoch. For each model type, we train 2 models respectively with different initial parameters and report the Top-1 and Top-5 test error in Table 9.2. We observe that St and Dth models achieve similar Top-1

[1] ↑https://github.com/Adamdad/keras-YOLOv3-mobilenet

Table 9.1. Overview of datasets

Dataset	Image size	Training Instances	Test Instances
IMAGENET	Clip to 416×416	1,281,167	50,000 (test)
MS-COCO-2017	Resize & Pad to 416×416	118,287	5,000 (validation)

Table 9.2. ImageNet Test Error (mean±std) for ResNet-50, image size 416 × 416

ResNet-50	St 0	St 1	Dth 0	Dth 1	AES 0	AES 1
Top-1 Err	25.27%	25.22%	25.59%	25.37%	27.46%	27.76%
Top-5 Error	8.22%	8.19%	8.33%	8.12%	9.30%	9.41%

Table 9.3. MS-COCO-2017 mAP (mean average precision) for YoloV3, image size 416 × 416

YoloV3	St 0	St 1	Dth 0	Dth 1	AES 0	AES 1
mAP	34.29	34.14	33.83	34.51	29.47	28.22

($\approx$ 25.3%) and Top-5 ($\approx$ 8.2%) error, whereas AES models drop roughly 2% Top-1 accuracy and 1% Top-5 accuracy.

Then, we use a two-stage training procedure[1] to train YoloV3 on MS-COCO-2017 [74]. We use ResNet-50 as the backbone network and the pre-trained ResNet-50 model as its initial model parameters. In the first training stage, we freeze the parameters of the backbone network and only train the parameter for the rest of the feature extraction network. We train the network 50 epochs with global batch size 64 and optimize the network with Adam optimizer [75]. The learning rate starts with 0.001 per 16 mini-batch, and is reduced by 1/10 if the average loss value does not get improved for 3 consecutive epochs when evaluating the validation dataset. In the second training stage, we repeat the same procedure as the first training stage. However, the backward propagation updates all parameters in the model, and the training procedure stops early if the average loss value does not get improved for 10 consecutive epochs when evaluating the validation dataset. In addition, for both training stages, we retain most of the default tuning parameters including parameters for image preprocessing and thresholds for feature extraction used in processing YoloV3 outputs (IoU, confidence score, and maximum valid bounding boxes).

After training, we evaluate YoloV3 models on MS-COCO-2017 validation dataset and report the Mean Average Precision (mAP) in Table 9.3. The evaluation uses IoU threshold 0.5 and confidence score threshold 0.45 to find valid bounding boxes. Specifically, the IoU

threshold requires each bounding box to have an over 50 percent overlap with a bounding box in ground truth detection. The Confidence score threshold ensures the model reports the existence of every valid bounding box over 0.45. We found that both St and Dth models have similar mAP on benign inputs, ranging from 33.83 to 34.51, whereas AES models experience an mAP drop of around 5 to 6.

9.2 TTA and Model Ensemble Improvements

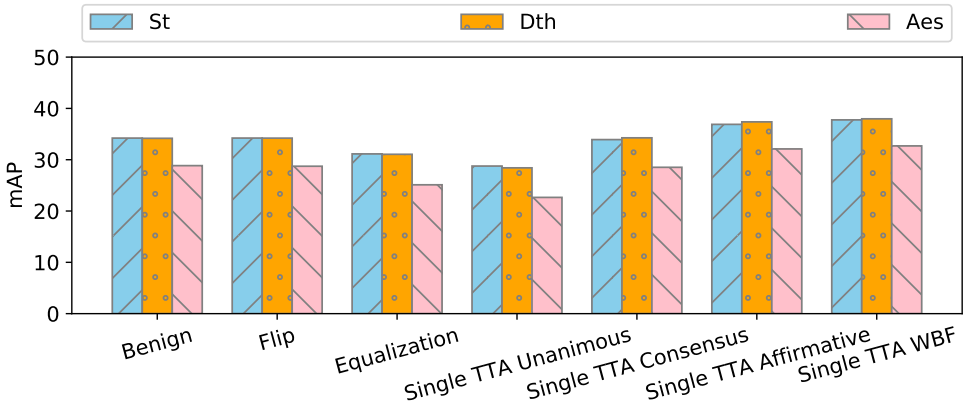
We are interested in how much improvement each proposed strategy described in Sec. 8.4 can deliver on benign examples. Thus, we first present how we measure each strategy and report the mAP (measured on MS-COCO-2017 validation dataset) in Fig. 9.1. Then, we present key findings.

9.2.1 Single TTA

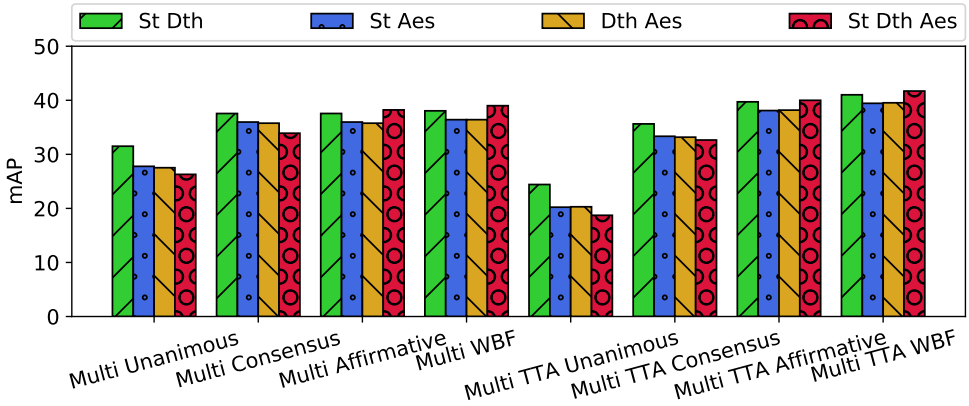
Given a single model, TTA can help to improve mAP but has to be used with model ensemble. Recall 4 model ensemble methods described in Sec. 8: Affirmative, Consensus, Unanimous, and WBF. We evaluate which model ensemble method is the best to use with TTA. To measure this, we first use TTA to generate predictions for benign, flipped, and equalized images. Then, we apply one of four model ensemble methods on TTA-generated predictions and plot the mAP to compare which ensemble strategy can better utilize TTA-generated predictions (See Fig. 9.1a). mAP for different ensemble methods is sorted in ascending order.

9.2.2 Ensemble with Multi-Models

Given multiple models, one can generate predictions for each model and then apply a model ensemble method. We compare 4 ensemble methods described previously where each strategy is measured on 4 different multi-model combinations described in Sec. 8.4.3: St + Dth, St + AES, Dth + AES, and 3Mix (St + Dth + AES). In total, we get 16 mAP data points which are then grouped by the model ensemble method. We sort the group by overall mAP in ascending order and plot them as the first 4 bar groups in Fig. 9.1b.



(a) mAP on Single Model



(b) mAP on Multi-Models

Figure 9.1. mAP for Different Proposed Strategies on MS-COCO-2017 Validation Dataset (Higher is Better)

9.2.3 TTA+Model Ensemble on Multi-Models

We repeat the same evaluation as we did above for Ensemble with Multi-Models. The difference is that, for each model in a multi-model combination, we use TTA to generate predictions for benign, flipped, and equalized images. Therefore, the total predictions that the model ensemble method needs to ensemble is the number of models in the combination multiple by 3. The final result is presented as the last 4 bar groups in Fig. 9.1b.

9.2.4 Key Findings:

1. **WBF and Affirmative are the best ensemble methods** (See Fig. 9.1a and Fig. 9.1b), whereas Unanimous is the worst model ensemble method, and Consensus is the second worst model ensemble method. This property is consistent for any given multi-model combination regardless of whether TTA is used. Because of this, we only use WBF and Affirmative in our later experiments.

2. **mAP drops when using TTA with Unanimous or Consensus.** Fig.9.1b shows that the group Multi TTA Unanimous and Multi TTA Consensus have lower mAP than the group Multi Unanimous and Multi Consensus. We attribute this to high agreement requirements imposed by the ensemble strategies of Unanimous and Consensus.

3. **AES model performs the worst on the validation dataset**. Fig. 9.1a shows that, when the prediction only uses a single model, the AES model always performs the worst, no matter which model ensemble method is used and whether TTA is used or not. Such property holds when the prediction uses two model combinations. Specifically, all 8 bar groups in Fig. 9.1b shows the combination having one AES model (St +AES and Dth +AES) always performs worse than the combination without an AES model (St +Dth)

4. **3Mix is the best model combination**. Fig. 9.1b shows that when using WBF and Affirmative, which are the best ensemble methods, the 3Mix combination always outperforms the other two model combinations regardless of whether TTA is used or not. This suggests including more diverse types of models can improve mAP, as their different behaviors lead to a better chance of covering the mistakes each other.

9.3 Evaluation of Single Model's Resistance to white-box and translucent-box attacks

Before evaluating the proposed strategies described in Sec. 8.4, it is important to understand how well single models can defend against adversarial examples. We use the TOG attack to generate adversarial examples and evaluate how well single model approaches can

defend against the white-box and translucent-box attack. Meanwhile, we evaluate how TTA helps a single model defend against adversarial examples as TTA can be used with the single model strategy to improve prediction accuracy.

9.3.1 white-box Evaluation

For each image in the MS-COCO-2017 validation dataset, we use 5 different TOG white-box attacks to generate adversarial examples on 2 St and 2 Dth models respectively as described in the original paper [18]. Therefore, we get 20 adversarial datasets. For each adversarial dataset, we measure the mean prediction accuracy mAP. Then, we group the mAP that is from the same model type and report the average mAP for each group. To understand how TTA helps defend against the white-box attack, we apply TTA Affirmative and TTA WBF in predicting generated adversarial examples and report the average mAP as we did previously.

9.3.2 translucent-box Evaluation

We assume the adversary knows the type of the targeted model type. We use the same adversarial examples generated in the white-box evaluation. For each model type, notice that we have trained two different models. To measure the transferability, we use one model to predict the adversarial examples generated by the other model and report the average mAP. Fig. 9.3 shows the mAP for a single model in predicting adversarial examples under the translucent-box setting by using different prediction strategies.

9.3.3 Key Findings

1. **Dithering improves model robustness and reduces the transferability**. Fig. 9.2 and Fig. 9.3 show the Dth model always delivers higher mAP in predicting adversarial examples for both white-box and translucent-box settings regardless of whether TTA is used or not. This suggests the Dth model is more robust against the white-box attack and transferability of adversarial examples.

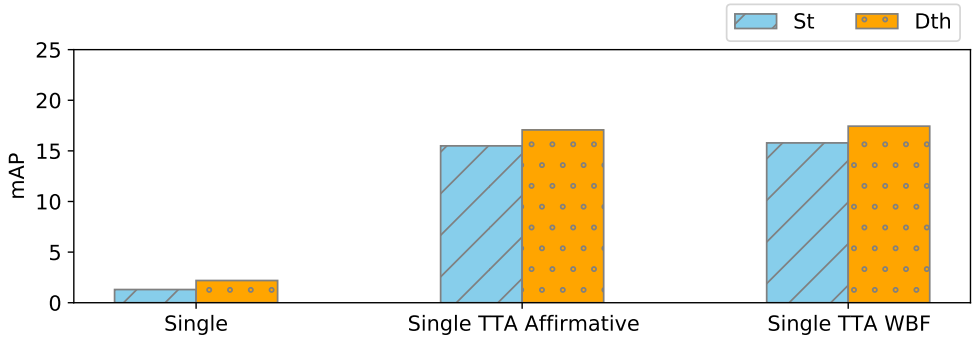


Figure 9.2. mAP for single model in predicting adversarial under the white-box setting. (Higher mAP is better.)

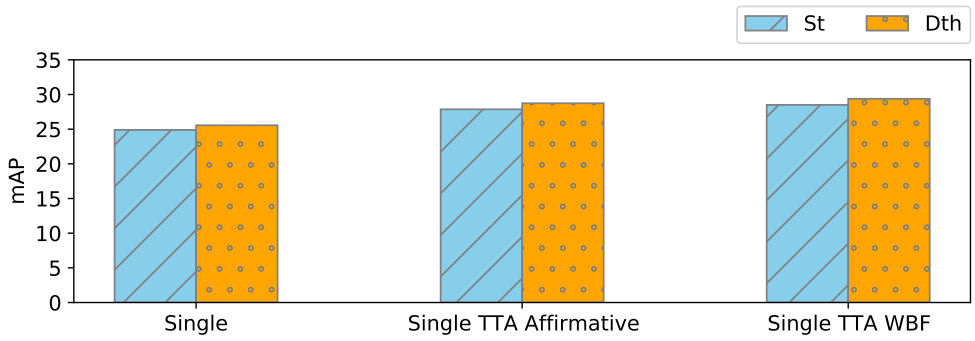


Figure 9.3. mAP for single model in predicting adversarial under the translucent-box setting. (Higher mAP is better.)

2. **TTA improves mAP in predicting adversarial examples**. The trend in Fig. 9.2 and Fig. 9.3 shows that Single TTA delivers higher mAP in predicting adversarial examples implying to reduce the transferability of adversarial examples.

The raw data we use in Fig. 9.1 is in Table A.1 which reports the mAP for every single model with it strategy in predicting adversarial examples (See Appendix). Columns are grouped by the prediction strategy and then the model type. Columns having the same model types are rendered with the same color. Each cell illustrates the mAP for an inference model in the column using different strategies to predict the adversarial examples generated from the attack method and targeted model on each row. The lower the number means the

attack is more effective. To get a sense of how effective the defense is, we add a baseline row "**Benign**" which is the reference mAP for a model using the strategy to predict the original MS-COCO-2017 validation dataset. In a column, if a number in a cell is close to the cell in the benign row, the model using the strategy is resistant to the corresponding adversarial dataset.

9.4 Tranferability Evaluation

We evaluate the effectiveness of the 4 proposed strategies described in Sec. 8.4. We use the same adversarial examples generated in the white-box evaluation. In addition, we assume the adversary does not know either the model type or the model parameter used in the prediction. Therefore, for each model combination (1-3 models) and prediction strategy, we evaluate the prediction accuracy on adversarial datasets that are not generated from any models used in the prediction. Then, we group the result by the same strategy and the model combination having the same model type, and compute the average mAP in the group. Notice that except for the Single Model strategy, the other 3 strategies use model ensemble which can be either Affirmative or WBF. Therefore, we have 7 total prediction strategies and sort the strategy by overall mAP in ascending order. The result is plotted in Fig. 9.4. The legend shows the actual types of models we use for the corresponding bar. The raw data we use in Fig. 9.4 can be found in Table A.1, Table A.2, and Table A.3. Table A.2 and Table A.3 represent the data reported by Multi and Model Ensemble + TTA strategies. Notice that The data format of Table A.2 and Table A.3 is consistent with Table A.1 as described in previous key findings.

9.4.1 Key Findings

1. **St < AES < Dth** First three column groups in Fig. 9.4 shows the Dth model achieves the highest mAP in predicting adversarial examples. This indicates the Dth model is more resistant to the tranferability of adversarial examples than the other two model type. AES performs second best. The reason that the model AES is better than the St model is because the AES model uses dithering in both training and testing,

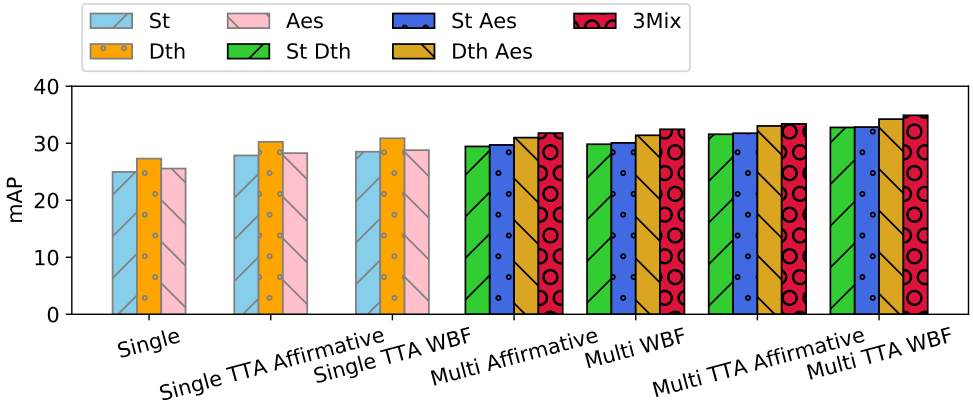


Figure 9.4. mAP for different strategies in predicting adversarial examples which are generated on models not used in predition (Higher mAP is better.)

indicating more robustness than the St model in defending against the tranferability of adversarial examples. The reason that AES performs worse than Dth is because the AES model has a lower prediction accuracy than Dth in predicting benign inputs which affect the AES in predicting adversarial examples.

2. **Single < Two Models < Three Models**. Fig. 9.4 shows that, in general, the ensemble that uses more diverse types of models achieves higher mAP in predicting adversarial examples. This is coherent with the previous key finding of 3Mix being the best model combination.

3. **Single < Single TTA; Multi < Multi TTA** Strategies using TTA is more robust against the tranferability of adversarial examples. Notice that TTA can generate two extra inputs which are different from the original input but have the same semantic information. This is coherent with the previous key finding on the benefits of including more models in an ensemble.

9.5 Attack the AES Model

We found it is infeasible to generate effective adversarial examples on AES models by attacking the gradient. Since the AES layer is non-differentiable, we studied attacks using

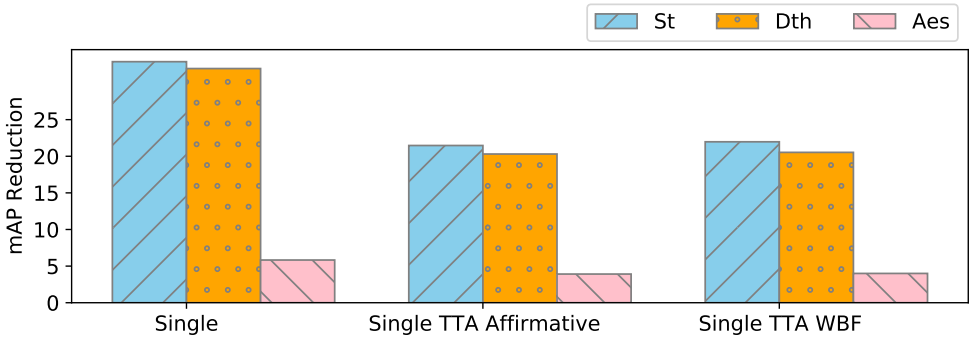


Figure 9.5. Difference of mAP for single model in predicting benign examples vs adversarial examples under the white-box setting. (Higher difference implies the stronger attack.)

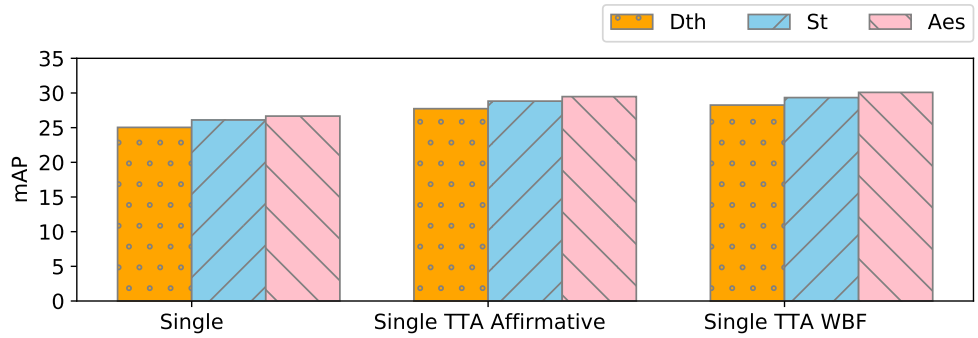


Figure 9.6. mAP for AES models in predicting adversarial examples generated on different targeted models. (Lower mAP imples adversarial examples are more effective)

the gradient approximation approach such as EOT (Expectation over Transformation) [57] and BPDA (Backwards Pass Differentiable Approximations) [65]. We found that EOT attack is not applicable on attacking the AES model since EOT assumes a random function $f(x)$ is used in model inference, which is contradict to the fact that the AES block cipher is a deterministic but pseudo-random permutation (on the mappings between plaintext and ciphertext). For the BPDA attack, our experiments found that BPDA attack on the AES model is not effective. We first measure the difference of accuracy (mAP) between predicting the benign example and adversarial example. Fig. 9.5 shows that BPDA attack on the AES model has the least reduction on mAP under the white-box setting comparing with attacking the St and Dth model, meaning that BPDA attack is not effective in attacking the AES

model. Then, we measure mAP of using the AES model in predicting adversarial examples generated on the St, Dth, and AES model respectively under the translucent-box setting. Fig. 9.6 shows that adversarial examples generated on the AES model achieves the highest mAP, implying those adversarial examples are less effective. The reason that BPDA attack is not effective is that BPDA method assumes the non-differentiable pre-processing function g has the property $g(x) \approx x$. Therefore, the derivative $g'(x) \approx 1$. However, such assumption does not hold for using AES encryption as the mapping from the original pixel value to the encrypted pixel value is pseudo-random. This implies that one has to figure out the right direction and the amount of noise to add to the image purely based on the gradient information instead of adding noise directly to the image. As the gradient approximation attack not applicable on attacking the AES model, the gradient free attack such as Zoo (Zeroth Order Adversarial Attacks) [76] needs to be studied.

10. CONCLUSION

In this dissertation, we present a careful analysis of possible adversarial models for studying the phenomenon of adversarial examples in computer vision. We first propose an evaluation methodology that can better illustrate the strengths and limitations of different mechanisms. As part of the method, we introduce a more powerful and meaningful adversary strategy.

Then, we present various approaches that use randomness to defend against adversarial examples for image classifiers. We introduce Random Spiking, a randomized technique that generalizes dropout. We have conducted extensive evaluation of Random Spiking and several other defense mechanisms, and demonstrate that Random Spiking, especially when combined with adversarial training, offers better protection against adversarial examples when compared with other existing defenses. In addition to Random Spiking, we present MoNet, an image pre-processing method, which uses the combination of using secret randomness and Floyd-Steinberg dithering. We use MoNet to generate input images which are first processed using Floyd-Steinberg dithering, and then each pixel is encrypted using the AES block cipher. The encrypted input images would protect the model parameters. Our evaluation shows that image classifiers trained with MoNet significantly improves model stability and robustness against adversarial examples, especially reduce the transferability of adversarial examples.

Moreover, we present the lightweight approach to defend against adversarial examples for object detectors by using the combination for existing methods. We explore three defense methods MoNet, Model Ensemble, and TTA. We present 4 strategies that combine these methods to further reduce transferability. Our evaluation shows that models trained with MoNet behave different from benign models in predicting adversarial examples and are robust against adversarial transferability. In addition, our evaluation shows that model ensemble strategies such as Affirmative and WBF not only improve mAP on benign input but also on transfered adversarial examples. We found ensemble more diverse models reduce more transferability, thus recommending 3Mix Affirmative or 3Mix WBF strategy. Moreover, our evaluation shows that Model Ensemble + TTA strategy further reduces transferability. Therefore, the best strategy is 3Mix TTA Affirmative or 3Mix TTA WBF.

REFERENCES

[1] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[3] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACM, 2017, pp. 278–287.

[4] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2017, pp. 135–147.

[5] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Network and Distributed Systems Security Symposium*, 2018. arXiv: 1704.01155.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[7] R. W. Floyd and L. Steinberg, "An Adaptive Algorithm for Spatial Greyscale," *Proceedings of the Society for Information Display*, vol. 17, no. 2, pp. 75–77, 1976.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[9] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91.

[12] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv e-prints*, arXiv:2004.10934, arXiv:2004.10934, Apr. 2020. arXiv: 2004.10934 [cs.CV].

[15] G. Jocher *et al.*, *ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference*, version v6.1, Feb. 2022. DOI: 10.5281/zenodo.6222936. [Online]. Available: https://doi.org/10.5281/zenodo.6222936.

[16] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 21–37, ISBN: 978-3-319-46448-0.

[17] K. H. Chow, L. Liu, M. E. Gursoy, S. Truex, W. Wei, and Y. Wu, "TOG: targeted adversarial objectness gradient attacks on real-time object detection systems," *CoRR*, vol. abs/2004.04320, 2020. arXiv: 2004.04320. [Online]. Available: https://arxiv.org/abs/2004.04320.

[18] K.-H. Chow *et al.*, "Adversarial objectness gradient attacks in real-time object detection systems," in *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications*, IEEE, 2020, pp. 263–272.

[19] K.-H. Chow, L. Liu, M. E. Gursoy, S. Truex, W. Wei, and Y. Wu, "Understanding object detection through an adversarial lens," in *European Symposium on Research in Computer Security*, Springer, 2020, pp. 460–481.

[20] X. Wei, S. Liang, N. Chen, and X. Cao, "Transferable adversarial attacks for image and video object detection," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 954–960. DOI: 10.24963/ijcai.2019/134. [Online]. Available: https://doi.org/10.24963/ijcai.2019/134.

[21] Y. Li, D. Tian, M. Chang, X. Bian, and S. Lyu, "Robust adversarial perturbation on deep proposal-based models," in *BMVC*, 2018.

[22] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *International Conference on Computer Vision*, IEEE, 2017.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. arXiv: 1512.00567. [Online]. Available: http://arxiv.org/abs/1512.00567.

[25] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. DOI: 10.48550/ARXIV.1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556.

[26] A. Casado-García and J. Heras, *Ensemble methods for object detection*, https://github.com/ancasag/ensembleObjectDetection, 2019.

[27] R. Solovyev, W. Wang, and T. Gabruseva, "Weighted boxes fusion: Ensembling boxes from different object detection models," *Image and Vision Computing*, pp. 1–6, 2021.

[28] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2016, pp. 372–387.

[29] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," *AAAI Conference on Artificial Intelligence*, 2018.

[30] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 39–57.

[31] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[32] N. Carlini and D. A. Wagner, "MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples," *CoRR*, vol. abs/1711.08478, 2017. arXiv: 1711.08478. [Online]. Available: http://arxiv.org/abs/1711.08478.

[33] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2015.

[34] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[36] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 582–597. DOI: 10.1109/SP.2016.41.

[37] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," vol. 48, Jun. 2016, pp. 1050–1059. arXiv: 1506.02142 [stat.ML].

[38] S. Wang *et al.*, "Defensive dropout for hardening deep neural networks under adversarial attacks," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '18, San Diego, California: ACM, 2018, 71:1–71:8, ISBN: 978-1-4503-5950-4. DOI: 10.1145/3240765.3264699. [Online]. Available: http://doi.acm.org/10.1145/3240765.3264699.

[39] H. Ge, S. Y. Chau, and N. Li, "Monet: Impressionism as a defense against adversarial examples," in *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2020, pp. 246–255. DOI: 10.1109/TPS-ISA50397.2020.00040.

[40] M. Aprilpyone, Y. Kinoshita, and H. Kiya, "Adversarial robustness by one bit double quantization for visual classification," *IEEE Access*, vol. 7, pp. 177 932–177 943, 2019. DOI: 10.1109/ACCESS.2019.2958358.

[41] M. AprilPyone, Y. Kinoshita, and H. Kiya, "Filtering adversarial noise with double quantization," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1745–1749. DOI: 10.1109/APSIPAASC47483.2019.9023341.

[42] S.-Y. Lo and V. M. Patel, "Error diffusion halftoning against adversarial examples," in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3892–3896. DOI: 10.1109/ICIP42928.2021.9506591.

[43] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated, 2012, ISBN: 1441993258.

[44] J. C. Perez *et al.*, "Enhancing adversarial robustness via test-time transformation ensembling," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 81–91. DOI: 10.1109/ICCVW54120.2021.00015. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCVW54120.2021.00015.

[45] W. B. *, J. R. *, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyZI0GWCZ.

[46] J. Chen and M. I. Jordan, "Boundary attack++: Query-efficient decision-based adversarial attack," *CoRR*, vol. abs/1904.02144, 2019. arXiv: 1904.02144. [Online]. Available: http://arxiv.org/abs/1904.02144.

[47] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[48] M. Sugiyama, N. D. Lawrence, A. Schwaighofer, *et al.*, *Dataset shift in machine learning*. The MIT Press, 2017.

[49] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in neural information processing systems*, 2008, pp. 1433–1440.

[50] J. J. Heckman, *Sample selection bias as a specification error (with an application to the estimation of labor supply functions)*, 1977.

[51] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *ICLR*, 2019.

[52] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 7472–7482. [Online]. Available: http://proceedings.mlr.press/v97/zhang19p.html.

[53] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland: ACM, 2008, pp. 1096–1103, ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294.

[54] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," in *Proceedings of the 27th International Conference on Machine Learning*, vol. 11, JMLR.org, Dec. 2010, pp. 3371–3408.

[55] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Springer, 2010, pp. 177–186.

[56] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[57] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, Oct. 2018, pp. 284–293. [Online]. Available: http://proceedings.mlr.press/v80/athalye18b.html.

[58] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits.," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[59] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *CoRR*, vol. abs/1708.07747, 2017. arXiv: 1708.07747. [Online]. Available: http://arxiv.org/abs/1708.07747.

[60] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[61] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[62] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proceedings of 5th International Conference on Learning Representations*, 2017.

[63] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk, "Adversarial examples are a natural consequence of test error in noise," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, Long Beach, California, USA: PMLR, Sep. 2019, pp. 2280–2289. [Online]. Available: http://proceedings.mlr.press/v97/gilmer19a.html.

[64] N. Carlini *et al.*, "On evaluating adversarial robustness," *CoRR*, vol. abs/1902.06705, 2019. arXiv: 1902.06705. [Online]. Available: http://arxiv.org/abs/1902.06705.

[65] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, Oct. 2018, pp. 274–283. [Online]. Available: http://proceedings.mlr.press/v80/athalye18a.html.

[66] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[68] O. Okun, G. Valentini, and M. Ré, Eds., *Ensembles in Machine Learning Applications*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 373, ISBN: 978-3-642-22909-1. DOI: 10.1007/978-3-642-22910-7. [Online]. Available: https://doi.org/10.1007/978-3-642-22910-7.

[69] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2017, pp. 6469–6477. DOI: 10.1109/CVPR.2017.685. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.685.

[70] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms improving object detection with one line of code," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2017, pp. 5562–5570. DOI: 10.1109/ICCV.2017.593. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.593.

[71] H. Zhou, Z. Li, C. Ning, and J. Tang, "Cad: Scale invariant framework for real-time object detection," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 760–768. DOI: 10.1109/ICCVW.2017.95.

[72] P. Wei, J. E. Ball, and D. T. Anderson, "Fusion of an ensemble of augmented image detectors for robust object detection," *Sensors*, vol. 18, no. 3, p. 894, 2018.

[73] P. Goyal *et al.*, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017. arXiv: 1706.02677. [Online]. Available: http://arxiv.org/abs/1706.02677.

[74] T.-Y. Lin *et al.*, *Microsoft coco: Common objects in context*, cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312.

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[76] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 15–26, ISBN: 9781450352024. [Online]. Available: https://doi.org/10.1145/3128572.3140448.

A. APPENDIX

The appendix includes more MoNet examples on MS-COCO-2017 dataset and the raw data used in Chapter 9. Each row in Table A.1, Table A.2, and Table A.3 represents the attack algorithm and the targeted model used in generating adversarial examples. Each column represents the prediction model. Each cell represents the mAP of using the tuple {attack algorithm, targeted model, prediction model}. Notice the first row in the table is the baseline result obtained on benign examples. If the number in the cell is closer to the baseline number, it implies the prediction model is more robust.

Table A.1. mAP for Single model and Single Model + TTA Predicting Adversarial examples Underlined numbers are for white-box attack.

		Single Model						Single TTA Affirmative						Single TTA WBF					
		St 0	St 1	Dth 0	Dth 1	AES 0	AES 1	St 0	St 1	Dth 0	Dth 1	AES 0	AES 1	St 0	St 1	Dth 0	Dth 1	AES 0	AES 1
Benign		34.29	34.14	33.83	34.51	29.47	28.22	36.89	36.89	37.18	37.57	32.77	31.43	37.74	37.78	37.71	38.23	33.39	31.99
untargeted	**St 0**	**1.33**	**19**	**23.87**	**24.46**	**25.23**	**24.94**	**5.33**	**21.88**	**26.44**	**26.87**	**27.85**	**27.53**	**6.01**	**22.52**	**27.12**	**27.72**	**28.56**	**27.82**
	St 1	**19.68**	**1.24**	**23.81**	**25.09**	**25.24**	**24.81**	**22.17**	**4.85**	**26.59**	**27.82**	**28.35**	**27.3**	**22.97**	**5.61**	**27.24**	**28.58**	**28.82**	**27.92**
	Dth 0	20.93	20.15	2.21	21.65	24.21	24.01	22.99	22.46	6.94	24.63	27.04	26.23	23.75	23.19	7.94	25.46	27.69	26.93
	Dth 1	20.04	20.27	20.18	2.33	23.77	23.57	22.06	22.62	23.14	7.05	26.62	26.21	22.89	23.52	23.79	7.93	27.16	26.68
vanish	**St 0**	**0.08**	**23.98**	**27.06**	**27.43**	**26.09**	**25.39**	**19.67**	**27.41**	**30.11**	**30.72**	**29.04**	**27.66**	**19.77**	**27.95**	**30.48**	**31.25**	**29.47**	**28.15**
	St 1	**23.98**	**0.11**	**26.78**	**27.51**	**26.15**	**25.45**	**27.41**	**20.7**	**30**	**30.75**	**29.04**	**27.79**	**27.82**	**20.8**	**30.42**	**31.22**	**29.57**	**28.37**
	Dth 0	24.09	23.8	0.18	24.86	24.68	24.31	27.21	27.27	20.23	28.57	27.67	26.9	27.57	27.77	20.33	28.99	28.08	27.26
	Dth 1	23.78	24.16	23.82	0.24	24.66	24.16	26.98	27.37	27.35	20.04	27.54	26.68	27.53	28	27.92	20.21	28.19	27.13
fabrication	**St 0**	**1.41**	**25.91**	**28.74**	**29.38**	**26.73**	**25.46**	**20.8**	**29.29**	**31.43**	**32.44**	**29.57**	**28.25**	**20.93**	**30**	**31.99**	**33.1**	**30.13**	**28.67**
	St 1	**26.31**	**1.2**	**28.81**	**29.78**	**26.42**	**25.48**	**29.18**	**21.53**	**31.57**	**32.56**	**29.33**	**28.18**	**29.79**	**21.71**	**32.06**	**33.15**	**29.99**	**28.74**
	Dth 0	26.08	25.8	3	27.42	25.29	24.64	28.8	29.15	21.73	30.54	28.4	27.14	29.55	29.74	22.01	31.18	28.76	27.66
	Dth 1	25.44	25.43	25.67	3.25	25.27	24.56	28.57	28.92	28.78	22.39	28.22	27.19	29.08	29.5	29.42	22.61	28.65	27.7
mislabelling ll	**St 0**	**0.29**	**25.86**	**28.78**	**28.83**	**26.64**	**25.73**	**14.5**	**28.82**	**31.47**	**31.85**	**29.42**	**28.36**	**14.64**	**29.45**	**32.06**	**32.53**	**30.03**	**28.91**
	St 1	**25.88**	**0.29**	**28.63**	**29.36**	**26.86**	**25.88**	**28.78**	**15.24**	**31.45**	**32.23**	**29.67**	**28.34**	**29.37**	**15.32**	**31.95**	**32.97**	**30.18**	**28.91**
	Dth 0	26.15	25.55	0.6	26.56	25.4	24.93	28.78	28.94	17.12	30.06	28.55	27.45	29.37	29.47	17.22	30.64	28.97	27.96
	Dth 1	25.49	25.9	26.02	0.69	25.39	24.91	28.4	29.21	29.14	17.63	28.39	27.26	29.11	29.85	29.84	17.82	28.94	27.72
mislabelling ml	**St 0**	**3.66**	**29.35**	**31.05**	**31.85**	**27.77**	**26.82**	**16.18**	**32.09**	**33.48**	**34.37**	**30.44**	**29.35**	**16.59**	**32.8**	**34.17**	**35.11**	**31.08**	**29.94**
	St 1	**28.94**	**3.41**	**30.8**	**31.64**	**28.07**	**26.68**	**31.6**	**16.15**	**33.25**	**34.51**	**30.73**	**29.09**	**32.31**	**16.48**	**33.94**	**35.22**	**31.32**	**29.72**
	Dth 0	29.31	29.46	4.64	30.05	27.26	26.26	31.83	32	18.24	33	30.02	28.85	32.63	32.81	18.74	33.69	30.53	29.33
	Dth 1	29.14	29.33	29.38	4.8	27.14	26.18	31.52	32.13	32.19	19.33	30.03	28.89	32.21	32.88	32.86	19.59	30.48	29.33

Table A.2. mAP for Multi Model Affirmative and WBF in Predicting Adversarial examples Underlined numbers means that adversarial examples are generated on one of the prediction models.

		Multi WBF								Multi Affirmative							
		StDth0	StDth1	StAES0	StAES1	DthAES0	DthAES1	3Mix0	3Mix1	StDth0	StDth1	StAES0	StAES1	DthAES0	DthAES1	3Mix0	3Mix1
Benign		37.95	38.15	36.64	36.19	36.62	36.21	39.11	38.89	37.49	37.62	36.3	35.65	35.96	35.57	38.31	38.12
untargeted	**St 0**	**9.84**	**25.31**	**9.35**	**26.85**	**29.16**	**29.86**	**20.17**	**29.42**	**8.97**	**24.97**	**8.86**	**26.29**	**28.69**	**29.3**	**18.99**	**28.59**
	St 1	**25.39**	**10.11**	**27.35**	**8.86**	**29.36**	**30.07**	**29.4**	**20.67**	**24.68**	**9.24**	**26.91**	**8.32**	**28.82**	**29.61**	**28.44**	**19.18**
	Dth 0	10.28	24.02	27.16	26.81	10.78	27.97	18.33	28.22	9.58	23.4	26.56	26.27	10.06	27.3	16.94	27.12
	Dth 1	23.13	10.19	26.18	26.78	26.24	10.89	26.96	18.03	22.54	9.57	25.83	26.15	25.86	10.19	26.23	16.83
vanish	**St 0**	**27.05**	**30.13**	**26.08**	**29.6**	**31.32**	**31.26**	**31.32**	**32.59**	**27.03**	**29.82**	**26.07**	**29.3**	**30.94**	**30.99**	**30.92**	**32.13**
	St 1	**29.68**	**27.5**	**29.89**	**25.44**	**31.23**	**31.43**	**32.61**	**31.41**	**29.45**	**27.47**	**29.77**	**25.41**	**30.81**	**31.06**	**32**	**31.03**
	Dth 0	24.08	28.34	29.19	28.74	24.68	29.35	29.19	31.12	24.06	28.15	28.88	28.5	24.65	29.14	28.88	30.62
	Dth 1	27.97	24.17	29.05	28.84	28.9	24.18	30.9	28.86	27.65	24.16	28.85	28.48	28.39	24.18	30.3	28.49
fabrication	**St 0**	**26.44**	**32.12**	**24.48**	**31.09**	**32.49**	**32.73**	**31.05**	**34.27**	**26.33**	**31.94**	**24.38**	**30.66**	**32.16**	**32.16**	**30.64**	**33.62**
	St 1	**31.81**	**27.2**	**31.39**	**23.38**	**32.46**	**32.8**	**34.1**	**31.26**	**31.53**	**27.05**	**31.26**	**23.29**	**32.06**	**32.5**	**33.52**	**30.89**
	Dth 0	24.07	30.71	30.72	30.35	23.65	31.45	29.41	33.16	24.03	30.45	30.45	29.97	23.58	31.12	29.09	32.52
	Dth 1	29.68	23.29	30.33	30.11	30.31	23.06	32.21	28.74	29.28	23.22	30.02	29.78	29.88	22.98	31.75	28.33
mislabelling ll	**St 0**	**23.26**	**31.59**	**20.13**	**30.77**	**32.46**	**32.21**	**29.92**	**33.78**	**23.21**	**31.2**	**20.1**	**30.37**	**32.13**	**31.94**	**29.58**	**33.12**
	St 1	**31.52**	**23.95**	**31.26**	**19.46**	**32.35**	**32.65**	**33.8**	**30.06**	**31.23**	**23.89**	**30.94**	**19.42**	**31.85**	**32.14**	**33.17**	**29.53**
	Dth 0	21.41	30.19	30.5	30.18	19.94	30.62	28.22	32.51	21.33	29.9	30.26	29.68	19.9	30.39	27.98	31.93
	Dth 1	29.74	21.01	30.28	30.35	30.36	19.57	32.21	27.94	29.39	20.96	30.07	29.96	29.98	19.55	31.67	27.56
mislabelling ml	**St 0**	**21.22**	**34.67**	**19.08**	**33.11**	**34.07**	**34.48**	**28.9**	**36.07**	**20.86**	**34.12**	**18.85**	**32.68**	**33.59**	**33.93**	**28.33**	**35.18**
	St 1	**33.68**	**21.36**	**33.49**	**18.12**	**34.07**	**34.24**	**35.69**	**28.99**	**33.37**	**21.05**	**33.05**	**17.87**	**33.76**	**33.73**	**35.03**	**28.16**
	Dth 0	20.87	33.57	33.13	32.8	19.58	32.95	28.21	34.96	20.62	33.07	32.71	32.33	19.45	32.8	27.65	34.43
	Dth 1	33.21	20.81	32.9	32.67	32.94	19.38	34.93	27.96	32.67	20.53	32.52	32.3	32.54	19.03	34.21	27.43

Table A.3. mAP for Multi Model TTA Affirmative and TTA WBF in Predicting Adversarial examples Underlined numbers means that adversarial examples are generated on one of the prediction models.

		Multi TTA WBF								Multi TTA Affirmative							
		StDth0	StDth1	StAES0	StAES1	DthAES0	DthAES1	3Mix0	3Mix1	StDth0	StDth1	StAES0	StAES1	DthAES0	DthAES1	3Mix0	3Mix1
Benign		40.9	41.13	39.66	39.22	39.73	39.34	41.75	41.67	39.68	39.75	38.4	37.8	38.41	37.95	40.07	39.92
untargeted	**St 0**	**17.48**	**28.2**	**18.19**	**29.59**	**31.74**	**32.26**	**24.55**	**31.61**	**15.53**	**26.78**	**16.54**	**28.54**	**30.47**	**30.91**	**22.15**	**29.95**
	St 1	**28.18**	**18.09**	**30.24**	**17.91**	**32.23**	**32.84**	**31.81**	**25**	**26.7**	**16.17**	**29.02**	**16.33**	**30.96**	**31.53**	**30.26**	**22.54**
	Dth 0	16.97	27.06	29.64	29.35	19.16	30.81	23.39	30.56	14.53	25.66	28.4	27.94	17.24	29.22	20.83	28.65
	Dth 1	25.82	16.71	28.86	29.25	29.32	18.88	29.41	23.03	24.56	14.52	27.54	27.75	28.14	17.16	27.9	20.45
vanish	**St 0**	**31.42**	**33.46**	**30.99**	**32.36**	**34.05**	**34.05**	**34.43**	**35.23**	**30.81**	**32.26**	**30.36**	**31.42**	**33.2**	**33.02**	**33.45**	**33.84**
	St 1	**32.76**	**32.17**	**33.04**	**30.39**	**34.25**	**34.33**	**35.22**	**34.63**	**31.75**	**31.52**	**32.27**	**29.61**	**33.16**	**32.96**	**33.87**	**33.29**
	Dth 0	28.82	31.78	32.02	31.68	29.87	32.45	32.58	33.86	28.18	30.94	31.01	30.92	29.09	31.6	31.27	32.75
	Dth 1	31.26	29.25	32.13	31.74	32.31	29.29	33.69	32.33	30.33	28.31	31.01	30.7	30.94	28.5	32.21	31.07
fabrication	**St 0**	**32.12**	**35.33**	**31.49**	**34.11**	**35.17**	**35.68**	**34.9**	**36.84**	**31.24**	**34.18**	**30.52**	**32.86**	**34.08**	**34.38**	**33.54**	**35.15**
	St 1	**34.46**	**33.11**	**34.37**	**30.8**	**35.29**	**35.6**	**36.4**	**35.36**	**33.31**	**32.11**	**33.21**	**29.76**	**34.1**	**34.44**	**34.97**	**33.82**
	Dth 0	30.41	34.1	33.56	33.39	30.79	34.3	33.6	35.86	29.54	33.05	32.56	32.44	30.17	33.07	32.51	34.33
	Dth 1	32.75	30.66	33.27	33.18	33.35	30.76	34.91	33.58	31.58	29.67	32.22	32.12	32.14	29.74	33.33	32.27
mislabelling ll	**St 0**	**30.99**	**34.6**	**29.53**	**33.5**	**35.1**	**35.14**	**34.24**	**36.21**	**30.24**	**33.35**	**28.73**	**32.39**	**33.98**	**33.99**	**33.03**	**34.58**
	St 1	**34.29**	**32.04**	**34.08**	**29.03**	**35.03**	**35.39**	**36.15**	**34.59**	**33.02**	**31.22**	**33.14**	**28.3**	**34.07**	**33.96**	**34.84**	**33.16**
	Dth 0	29.46	33.48	33.28	33.06	29.41	33.69	32.89	35.11	28.56	32.64	32.52	31.89	28.85	32.56	32.04	33.75
	Dth 1	32.87	29.72	33.27	33.14	33.42	28.75	34.9	32.71	31.7	28.76	32.31	32.2	32.4	28.13	33.6	31.68
mislabelling ml	**St 0**	**30.34**	**37.01**	**28.69**	**35.78**	**36.69**	**37.12**	**34.03**	**38.15**	**28.98**	**35.65**	**27.66**	**34.51**	**35.38**	**35.8**	**32.36**	**36.39**
	St 1	**36.16**	**30.93**	**35.92**	**28.24**	**36.62**	**37.09**	**37.69**	**34.23**	**34.88**	**29.7**	**34.88**	**27.3**	**35.57**	**35.76**	**36.22**	**32.6**
	Dth 0	30.12	36.09	35.36	35.32	29.36	35.75	33.44	37.16	28.72	34.8	34.35	34.09	28.39	34.65	31.94	35.71
	Dth 1	35.76	30.77	35.4	35.29	35.73	29.52	37.14	33.85	34.33	29.53	34.28	34.14	34.42	28.55	35.41	32.37

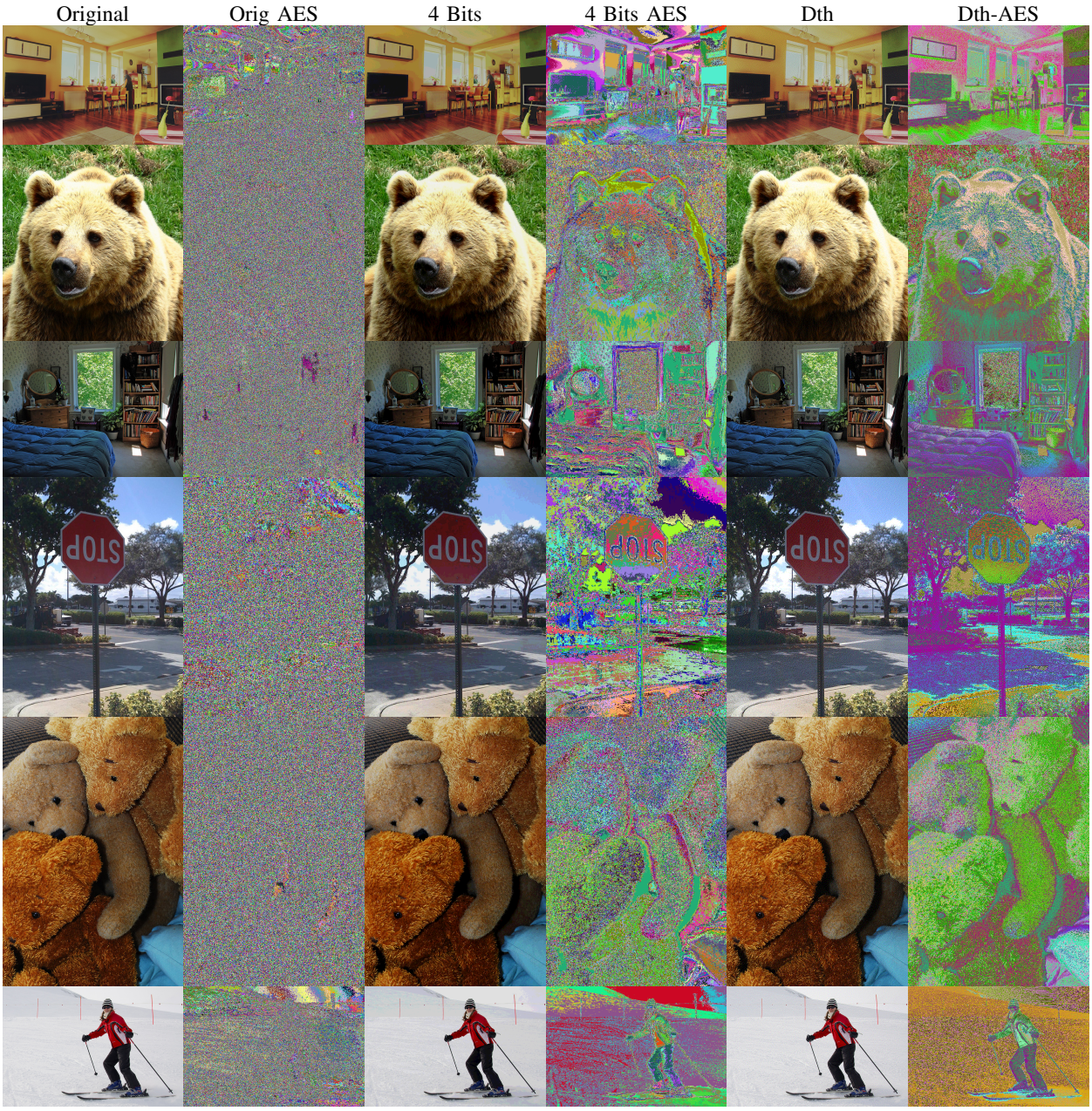


Figure A.1. MS-COCO-2017 MoNet Examples