Toward Robust Deep Neural Networks

Thèse

Mahdieh Abbasi

Doctorat en génie électrique
Philosophiæ doctor (Ph. D.)

Québec, Canada

© Mahdieh Abbasi, 2020

Toward Robust Deep Neural Networks

Thèse

Mahdieh Abbasi

Sous la direction de:

Christian Gagné, directeur de recherche
Denis Laurendeau, codirecteur de recherche

Résumé

Dans cette thèse, notre objectif est de développer des modèles d'apprentissage robustes et fiables mais précis, en particulier les Convolutional Neural Network (CNN), en présence des exemples anomalies, comme des exemples adversaires et d'échantillons hors distribution – Out-of-Distribution (OOD).

Comme la première contribution, nous proposons d'estimer la confiance calibrée pour les exemples adversaires en encourageant la diversité dans un ensemble des CNNs. À cette fin, nous concevons un ensemble de spécialistes diversifiés avec un mécanisme de vote simple et efficace en termes de calcul pour prédire les exemples adversaires avec une faible confiance tout en maintenant la confiance prédicative des échantillons propres élevée. En présence de désaccord dans notre ensemble, nous prouvons qu'une borne supérieure de $0.5 + \epsilon'$ peut être établie pour la confiance, conduisant à un seuil de détection global fixe de $tau = 0,5$. Nous justifions analytiquement le rôle de la diversité dans notre ensemble sur l'atténuation du risque des exemples adversaires à la fois en boîte noire et en boîte blanche. Enfin, nous évaluons empiriquement la robustesse de notre ensemble aux attaques de la boîte noire et de la boîte blanche sur plusieurs données standards.

La deuxième contribution vise à aborder la détection d'échantillons OOD à travers un modèle de bout en bout entraîné sur un ensemble OOD approprié. À cette fin, nous abordons la question centrale suivante : comment différencier des différents ensembles de données OOD disponibles par rapport à une tâche de distribution donnée pour sélectionner la plus appropriée, ce qui induit à son tour un modèle calibré avec un taux de détection des ensembles inaperçus de données OOD? Pour répondre à cette question, nous proposons de différencier les ensembles OOD par leur niveau de "protection" des sub-manifolds. Pour mesurer le niveau de protection, nous concevons ensuite trois nouvelles mesures efficaces en termes de calcul à l'aide d'un CNN vanille préformé. Dans une vaste série d'expériences sur les tâches de classification d'image et d'audio, nous démontrons empiriquement la capacité d'un CNN augmenté (A-CNN) et d'un CNN explicitement calibré pour détecter une portion significativement plus grande des exemples OOD. Fait intéressant, nous observons également qu'un tel A-CNN (nommé A-CNN*) peut également détecter les adversaires exemples FGS en boîte noire avec des perturbations significatives.

En tant que troisième contribution, nous étudions de plus près de la capacité de l'A-CNN sur la détection de types plus larges d'adversaires boîte noire (pas seulement ceux de type FGS). Pour augmenter la capacité d'A-CNN à détecter un plus grand nombre d'adversaires, nous augmentons l'ensemble d'entraînement OOD avec des échantillons interpolés inter-classes. Ensuite, nous démontrons que l'A-CNN, entraîné sur tous ces données, a un taux de détection cohérent sur tous les types des adversaires exemples invisibles. Alors que la entraînement d'un A-CNN sur des adversaires PGD ne conduit pas à un taux de détection stable sur tous les types d'adversaires, en particulier les types inaperçus. Nous évaluons également visuellement l'espace des fonctionnalités et les limites de décision dans l'espace d'entrée d'un CNN vanille et de son homologue augmenté en présence d'adversaires et de ceux qui sont propres. Par un A-CNN correctement formé, nous visons à faire un pas vers un modèle d'apprentissage de bout en bout unifié et fiable avec de faibles taux de risque sur les échantillons propres et les échantillons inhabituels, par exemple, les échantillons adversaires et OOD.

La dernière contribution est de présenter une application de A-CNN pour l'entraînement d'un détecteur d'objet robuste sur un ensemble de données partiellement étiquetées, en particulier un ensemble de données fusionné. La fusion de divers ensembles de données provenant de contextes similaires mais avec différents ensembles d'objets d'intérêt (OoI) est un moyen peu coûteux de créer un ensemble de données à grande échelle qui couvre un plus large spectre d'OoI. De plus, la fusion d'ensembles de données permet de réaliser un détecteur d'objet unifié, au lieu d'en avoir plusieurs séparés, ce qui entraîne une réduction des coûts de calcul et de temps. Cependant, la fusion d'ensembles de données, en particulier à partir d'un contexte similaire, entraîne de nombreuses instances d'étiquetées manquantes. Dans le but d'entraîner un détecteur d'objet robuste intégré sur un ensemble de données partiellement étiquetées mais à grande échelle, nous proposons un cadre d'entraînement auto-supervisé pour surmonter le problème des instances d'étiquettes manquantes dans les ensembles des données fusionnés. Notre cadre est évalué sur un ensemble de données fusionné avec un taux élevé d'étiquettes manquantes. Les résultats empiriques confirment la viabilité de nos pseudo-étiquettes générées pour améliorer les performances de YOLO, en tant que détecteur d'objet à la pointe de la technologie.

Abstract

In this thesis, our goal is to develop robust and reliable yet accurate learning models, particularly Convolutional Neural Networks (CNNs), in the presence of adversarial examples and Out-of-Distribution (OOD) samples.

As the first contribution, we propose to **predict adversarial instances with high uncertainty through encouraging diversity in an ensemble of CNNs**. To this end, we devise an ensemble of diverse specialists along with a simple and computationally efficient voting mechanism to predict the adversarial examples with low confidence while keeping the predictive confidence of the clean samples high. In the presence of *high entropy* in our ensemble, we prove that the predictive confidence can be upper-bounded, leading to have a globally fixed threshold over the predictive confidence for identifying adversaries. We analytically justify the role of diversity in our ensemble on mitigating the risk of both black-box and white-box adversarial examples. Finally, we empirically assess the robustness of our ensemble to the black-box and the white-box attacks on several benchmark datasets.

The second contribution aims to address the detection of OOD samples through an end-to-end model trained on an appropriate OOD set. To this end, we address the following central question: **how to differentiate many available OOD sets w.r.t. a given in-distribution task to select the most appropriate one**, which in turn induces a model with a high detection rate of *unseen* OOD sets? To answer this question, we hypothesize that the "protection" level of in-distribution sub-manifolds by each OOD set can be a good possible property to differentiate OOD sets. To measure the protection level, we then design three novel, simple, and cost-effective metrics using a pre-trained vanilla CNN. In an extensive series of experiments on image and audio classification tasks, we empirically demonstrate the ability of an Augmented-CNN (A-CNN) and an explicitly-calibrated CNN for detecting a significantly larger portion of unseen OOD samples, if they are trained on the most protective OOD set. Interestingly, we also observe that the A-CNN trained on the most protective OOD set (called A-CNN*) can also detect the black-box Fast Gradient Sign (FGS) adversarial examples.

As the third contribution, we investigate more closely **the capacity of the A-CNN* on the detection of wider types of black-box adversaries**. To increase the capability of A-CNN* to detect a larger number of adversaries, we augment its OOD training set with

some inter-class interpolated samples. Then, we demonstrate that the A-CNN trained on the most protective OOD set along with the interpolated samples has a consistent detection rate on all types of unseen adversarial examples. Whereas training an A-CNN on Projected Gradient Descent (PGD) adversaries does not lead to a stable detection rate on all types of adversaries, particularly the unseen types. We also visually assess the feature space and the decision boundaries in the input space of a vanilla CNN and its augmented counterpart in the presence of adversaries and the clean ones. By a properly trained A-CNN, we aim to take a step toward a unified and reliable *end-to-end learning model* with small risk rates on both clean samples and the unusual ones, e.g. adversarial and OOD samples.

The last contribution is to show a use-case of **A-CNN for training a robust object detector on a partially-labeled dataset**, particularly a merged dataset. Merging various datasets from similar contexts but with different sets of Object of Interest (OoI) is an inexpensive way to craft a large-scale dataset which covers a larger spectrum of OoIs. Moreover, merging datasets allows achieving a unified object detector, instead of having several separate ones, resulting in the reduction of computational and time costs. However, merging datasets, especially from a similar context, causes many missing-label instances. With the goal of training an integrated robust object detector on a partially-labeled but large-scale dataset, we propose a self-supervised training framework to overcome the issue of missing-label instances in the merged datasets. Our framework is evaluated on a merged dataset with a high missing-label rate. The empirical results confirm the viability of our generated pseudo-labels to enhance the performance of YOLO, as the current (to date) state-of-the-art object detector.

Contents

List of Tables

List of Figures

List of Abbreviations

A-CNN	Augmented CNN
A-MLP	Augmented Multi Layer Perceptron
CNN	Convolutional Neural Network
CW	Carlini and Wagner
FGS	Fast Gradient Sign
GPU	Graphical Processing Units
I-FGS	Iterative Fast Gradient Sign
i.i.d.	independently and identically distributed
IoU	Intersection over Union
KL-div.	Kullback-Leibler divergence
MLP	Multi Layer Perceptron
MNIST	Modified National Institute of Standards and Technology
OOD	Out-of-Distribution
OoI	Object of Interest
PGD	Projected Gradient Descent
R-CNN	Region CNN
ROI	Region of Interest
SVHN	Street View House Numbers
SVM	Support Vector Machine

T-FGS	Targeted Fast Gradient Sign
UPI	Unlabeled Positive Instance
VOC	Visual Object Classes
YOLO	You Only Look Once

To memory of Batoul, my beloved mother, who taught me to live a meaningful life.

Acknowledgment

I'd like to first thank my mother for all the scarifies she made for me to have a fulfilling and meaningful life. Till her last breath, she did not stop to support and encourage me to achieve my education goals. I dedicate this thesis to my beloved mother, Batoul, who passed away a few months before the end of my doctorate.

My special thanks go to my supervisor, Christian Gagné, the director of Institute Intelligence and Data (IID), for providing me with an enriched research environment to accomplish this thesis. He has taught me to stand on my feet to do an original and valuable research, and gave me the liberty to explore the exciting field of Machine Learning. He has provided me with a generous financial support to attend several top-notch conferences to build collaborative links with other researchers and broaden my research perspective. Without his guidance, efforts, and time, I could not have written this thesis.

I'd love to thank my co-supervisor, Denis Laurendeau– the director of Computer Vision and Systems Laboratory (CVSL)– for not only his supports in the hardest moment of my Ph.D. but also his trust and investment in my research abilities. I owe a big thanks to Jean-François Lalonde, an amazingly inspiring professor at CVSL, who taught me some great techniques for giving a clear presentation.

I am thankful for my co-authors – Changjian Shui, Arezoo Rajabi, Rakesh Bobba, Christian Gagné, and Denis Laurendeau – for their valuable collaborations. Particularly, I am grateful Changjian Shui, a Ph.D. student at CSVL, who introduced me to some advanced theoretical topics in machine learning. I've had a lot of interesting and inspiring research discussions with him.

My Ph.D. was made possible with funding from NSERC-Canada, Mitacs, Prompt-Québec, E Machine Learning, and Thales Canada. I also thank Annette Schwerdtfeger for proofreading my papers presented in this thesis.

Last but not least, I am cordially appreciative of having an affectionate and supportive parent, who did their best to provide me with all the necessities to receive quality educations. I especially thank my dear sister, Marzieh, who is a computer engineer, for sparking my interest in Computer Science (CS). Inspired by her, I studied CS at my dream university (Sharif University of Technology). With all my sincere feelings, I want to thank my loving husband, Hadi, who has been next to me during all ups and downs of my life.

Introduction

The availability of very large-scale labeled data sets and the remarkable progress in computational power of Graphical Processing Units (GPU), allow modern deep learning models, e.g. Convolutional Neural Network (CNN), to thrive by achieving performance near to human-level performance on several machine vision tasks such as face detection [97], optical character recognition [24], object recognition [95], and object detection [81, 78]. Due to the outstanding performance of CNNs in solving very complex tasks, they are increasingly used in automatic decision-making in a wide range of sectors, from industry to medicine to intelligent homes, among others.

Despite drastically high generalization performance/accuracy of CNNs, they may still suffer from some small errors in their predictions. To make safe and reliable decisions, it is critical to recognize which predictions are likely incorrect in order to discard them and avoiding wrong decisions. Evidently, in security-sensitive applications, (e.g. self-driving cars, medical diagnostic systems [12, 103]), making any wrong decisions, even if they are rare-occurring, may lead to irreversible disaster. In spite of this critical demand, unfortunately, the modern machine learning models, particularly CNNs, have been shown to suffer from uncalibrated predictions [29], meaning their predictive confidence for any given inputs is usually high, whether the prediction is correct or not. Indeed, in addition to high generalization performance, it is crucial that the CNNs can predict hard-to-classify instances (i.e the samples difficult to process accurately) with low confidence, allowing to distinguish them. Then, we may use human-in-loop for properly handling such difficult samples. This, ultimately, can reduce the risk and enhance reliability of the decisions made by such state-of-the-art learners.

This challenge can be even more intense by knowing that the CNNs trained for a specific task can actually classify *confidently any given input instances* into one of its pre-defined categories, whether the input samples are related to the given task or not. For example, a CNN trained on MNIST (handwritten digits) classifies *confidently* any given instances, whether they are digits or not (e.g. random noise or even natural-scene images) into one of 10 digit classes. The samples that are statistically and semantically different from a given task's samples (i.e. in-distribution data generation) are called **Out-of-Distribution (OOD)** samples as they are drawn from unknown or irrelevant data distribution generations. The learner models that output high confidence predictions for the OOD samples can jeopardize human's security and

safety, especially when being deployed in real-world applications. In other words, the disability of the standard CNNs to output uncertainty for the unknown situations (equivalent to "I don't know") reveals a serious risk for the safety and security. For example, if an automatic check reader device, which designed to read digits on some checks, can be easily circumvented using the OOD samples (non-digit examples), then, one can perform illegitimate money-transfer transactions although they appear completely legitimate according to the predictions provided by the underlying intelligent model.

Moreover, it has been widely demonstrated that CNNs are susceptible to hostile but imperceptible changes to their clean inputs, a.k.a, **adversarial examples**. Indeed, CNNs have been widely shown to be susceptible to the adversarial modifications as they classify a pair of clean input and its adversarial counterpart into two distinct classes, regardless of their visually identical appearance [96, 25]. In other words, while human observers can not perceive any visible difference between an adversarial example and its clean counterpart, a CNN sees them completely different, i.e. as coming from two distinct classes. The invention of a wide range of algorithms for adversarial example generation [1, 70, 9], as well as their transferability to attack almost any ML models [96, 101] can reflect the severity of vulnerability of CNNs to the adversarial examples. The adversarial examples generated by a proxy model, which can be transferred and attack other unknown models, are known as black-box adversaries. While the adversarial examples generated directly using a victim model to fool itself are known as the white-box attacks[1]. Moreover, due to the uncalibrated nature of CNNs, unfortunately, they *confidently and incorrectly classify* adversarial examples. Regarding benign-looking appearance of the adversaries and their high confidence predictions by CNNs, identifying adversarial examples from their clean counterparts is one of the main open problems in the field of modern deep neural network.

In this thesis, our major goal is to develop robust and reliable yet accurate Convolutional Neural Networks (CNNs) in the presence of adversarial examples and Out-of- Distribution (OOD) samples. To alleviate the risk of adversarial examples, the first contribution is on calibrating the predictive confidence using an ensemble of diverse specialists. The second contribution is concerned with the metrics for characterizing and differentiating various OOD sets for training a well-generalized CNN, like an Augmented CNN [38] or a calibrated CNN [52, 36], that is also able to detect a large spectrum of unseen OOD sets, reducing the risk of OOD samples. Besides OOD samples, in the third contribution, we aim to further assess the ability of A-CNN (trained properly) to diminish the risk of adversaries, through detecting them. The fourth contribution is on developing a robust object detector using self-supervised learning, inspired by our second and third contributions. In the following, we elaborate on the research objectives and questions of this thesis:

- **Objective 1: Demonstrating the impact of diversity in the ensemble for adver-**

[1]To read more on black-box, white-box, and gray-box attacks, see Chapter 3

sarial detection. Traditionally, it has been widely shown that diversity in the ensemble of ML models has a definite impact on enhancing the generalization performance (i.e. accuracy) [30, 8, 18]. Different from boosting generalization performance by ensemble methods, our first objective is to propose a novel ensemble-based framework to provide calibrated predictions, particularly for a large spectrum of adversarial examples.
The main research questions associated with this objective are:

1. Does diversity in ensembles of CNNs encourage calibrated predictions?
2. For calibration purposes, how can we promote effective diversity in the ensemble in a computationally efficient way?
3. Using an ensemble of diverse CNNs (without adversarial learning), can we detect the black-box adversarial examples based on their low predictive confidence, resulting in minimization of their risk, while maintaining the generalization performance on the clean samples?
4. Is such an ensemble of diverse CNNs robust to white-box adversarial examples?

- **Objective 2: Differentiating various accessible OOD sets for effective OOD learning.** It has been shown that vanilla CNNs can be trained to output calibrated prediction on Out-of-Distribution samples with the aim of detecting (unseen) OOD samples [36, 5]. Alternatively, one can use an Augmented CNN (A-CNN)– a vanilla CNN with an extra class added to its output– in order to directly classify the (unseen) OOD samples to its extra class. However, there are several unaddressed questions about OOD learning of these CNNs that we aim to answer in this objective:
  1. Considering the availability of a large number of OOD sets, are they equal for training a well-generalized CNN which in turn can detect a wide range of unseen and unknown OOD samples?
  2 What property(ies) of OOD sets are important for differentiating and recognizing the most proper OOD set for the purposes of training?
  3. Inspired by these properties, how could we design a collection of cost-effective metrics to effectively identify the most proper OOD set, which can be used for training a well-generalized CNN, robust to OOD samples?

- **Objective 3: Investigating the capacity of Augmented-CNN (A-CNN) for increasing the robustness to adversarial examples.** Our goal here is to quantitatively and qualitatively investigate the robustness of an A-CNN, which is trained on a proper OOD set recognized by our previous proposed metrics, to the black-box and the white-box adversarial examples. To accomplish this objective, the following questions are answered:
  1. Are ODD learning and widely-known "adversarial learning" equally effective to make an A-CNN more robust to I) the OOD samples, II) the black-box adversarial examples?

2. By some qualitative results, observe where are the OOD samples and adversaries located in the feature space of a vanilla CNN? How are their locations in the feature space changed by OOD learning of an A-CNN?
3. How does OOD learning affects the learned decision boundaries and the fooling regions (adversarial examples region) in the input space?

- **Objective 4: Propose an application of OOD learning for training a robust object detector from a partially-labeled large-scale dataset.** Besides the application of OOD learning of an A-CNN for detecting the OOD samples, we aim to showcase another novel application of OOD learning, which is about training robustly an object detector on a merged (partially-labelled) dataset. Merging several datasets, where each has a disjoint set of Objects-of-Interest (OoIs), is a *cost-effective* way to create a unified large-scale dataset that includes a wider spectrum of objects of interest that appear in a larger spectrum of conditions, e.g. light conditions, styles, among others. Compared to learning several object detectors –each for one dataset–, another motivation for learning a unified object detector on a merged dataset is that training and testing a single unified object detector incur lower computational cost (in terms of computation and memory resources).
  However, in such a merged dataset, there are a large number of missing-label instances, and training on it leads to an object detector with an inferior performance. *Our main goals is to leverage the available labelled information in the constitutive datasets to automatically label the missing-label instances in the merged one.* This, in turn, allows us to train a unified robust object detector with higher performance. To this end, we need to answer the following questions:
  1. How do the missing-label instances in a dataset negatively impact on the performance of the state-of-the-art object detectors such as YOLO and faster R-CNN?
  2. How to handle the missing-label instances that are emerged through merging various datasets by the use of A-CNN trained on an OOD set?
  3. To generate accurate pseudo-labels for the missing-label OoIs, how to reduce the number of wrong predictions, particularly for hard-to-classify instances?

This thesis is organized as follows. In the first chapter, we provide some background materials, then review the renowned attack algorithms and the proposed defenses. The second chapter covers our proposed ensemble of diverse specialist CNNs and our extensive experimental evaluations. Then, in the third chapter, we propose three novel metrics for differentiating an array of OOD sets w.r.t a given in-distribution task/set. Our metrics are then evaluated using a diverse set of benchmarks and we show that A-CNN trained on our selected OOD set can also detect FGS (Fast Gradient Sign) adversarial examples. In the fourth chapter, we further investigate the capacity of different A-CNNs for detecting a wider range of black-box adversarial examples through quantitative and qualitative empirical analyses. To showcase an

application of the A-CNN, in chapter 5, we propose a training framework that integrates an A-CNN into the pipeline of YOLO (an object detector) in order to generate pseudo-labels for missing-label instances in a merged dataset. Finally, we conclude the thesis by reviewing our research findings in the conclusion chapter.

Chapter 1

Background Material

In this chapter, we briefly describe some preliminary concepts, particularly those used frequently in the thesis, then we review some widely-known adversarial example attack algorithms and some state-of-the-art defense strategies to adversaries. Finally, we close the chapter by discussing OOD samples and the proposed methods in the literature for detecting them.

1.1 Preliminary Concepts

1.1.1 Notions and Definitions

For a $K-$class classification problem, a classifier $h : \mathcal{X} \to \mathcal{Y}$ is learned to classify a given input instance $\mathbf{x}$ from input-space $\mathcal{X}$ to a label $\mathbf{y}$ in the label-space $\mathcal{Y}$. The samples of the given classification problem are generated from an unknown joint data distribution $\mathcal{D}$, i.e. $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$. A neural network, as a classifier, maps an input image $x \in \mathcal{X} = \mathbb{R}^D$ into the conditional probabilities over K classes, i.e. $h : \mathcal{X} \to [0, 1]^K$, such that $h_1(\mathbf{x}) + \cdots + h_K(\mathbf{x}) = 1$. More specifically, a neural network (e.g. CNN) is a series of L nested functions, represented by $h(\mathbf{x}; \mathcal{W}) = \text{softmax}(h^L(h^{L-1}(\dots h^1(\mathbf{x}))))$, where softmax function creates the final conditional probabilities and $\mathcal{W}$ represents compactly the parameters of the learner[1]. Formally, a softmax function computes the class-conditional probabilities for $\mathbf{z} \in \mathbb{R}^K$, which is the output of $L-$th layer for $\mathbf{x}$,i.e. briefly shown as $h^L(\mathbf{x})$, as follows:

$$\text{softmax}(\mathbf{z}) = \frac{1}{\sum_{i=1}^{K} e^{z_i}} \left[e^{z_1}, \dots, e^{z_K}\right]. \tag{1.1}$$

Each layer l of the network is indeed a function $h^l(h^{l-1}(.))$ that is taking the output of the preceding layer as its input.

Generally speaking, in machine learning, we aim at finding a classifier h with a small "true error", which is defined as the expected loss of h on the samples randomly drawn from the

[1] We may drop $\mathcal{W}$ from h for simplicity in the notation.

given data distribution $\mathcal{D}$:

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}(h(\mathbf{x}) \neq \mathbf{y}). \tag{1.2}$$

However, computing the true error is not possible as the data distribution $\mathcal{D}$ is unknown. Thus, we instead find a classifier through minimizing its loss over a given set of training samples. This is known as Empirical Risk Minimization (ERM).

Consider a training set $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, consisting of N independently and identically distributed (i.i.d.) training pairs of input sample $\mathbf{x}_i \in \mathbb{R}^D$ with its associated ground-truth label $k \in \{1, \ldots, K\}$, which is represented by a one-hot vector $\mathbf{y}_i \in [0, 1]^K$ with a single 1 at its k-th element. The i.i.d. assumption means that each training sample $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$ is freshly drawn from the joint data distribution $\mathcal{D}$. A neural network $h(\cdot; \mathcal{W})$ is trained through minimizing the cross-entropy loss function $\mathcal{L}(., .; \mathcal{W}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ over N i.i.d. training samples, formulated as follows:

$$\begin{aligned} \mathcal{W}^* = \underset{\mathcal{W}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(h(\mathbf{x}_i), \mathbf{y}_i; \mathcal{W}) &= \underset{\mathcal{W}}{\operatorname{argmin}} \left(-\frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i \log h(\mathbf{x}_i; \mathcal{W}) \right) \\ &= \underset{\mathcal{W}}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \log h_k(\mathbf{x}_i; \mathcal{W}), \end{aligned} \tag{1.3}$$

where $h_k(\mathbf{x}_i)$ is the estimated conditional probability of the k-th class corresponding to the ground-truth label of a given sample $\mathbf{x}_i$.

1.1.2 Deep v.s. Shallow Learning Models

Many of conventional learning models, such as linear and logistic regression as well as kernel-based models (e.g. kernel SVM) are regarded as shallow learning models as they have no more than 2 layers. For examples, Gaussian Processes (GP) as a shallow ML model has two layers: one layer is a kernel function and another is the expected responses of the kernel function for each pair of a training example and the given test instance (i.e. $K(\mathbf{x}_t, \mathbf{x}_i)$, where $\mathbf{x}_i$ represents the i-th training example and $\mathbf{x}_t$ is the given test example).

Human's brain–the most powerful and unique natural learning system– can be categorized as a **deep** rather than a shallow learning model. For example, the visual cortex in human's brain, which is responsible for visual recognition, has 5-10 levels [88], where each level in this hierarchy presents a level of abstraction with some distributed representations [3]. Inspired by such a brilliant natural learning model, the researchers devised Convolutional Neural Network (CNN) that have several layers of hierarchy to mimic the way that human's brain is operating (Fig 1.2).

It has been argued that many of the shallow learning models, especially those that are based on the kernel functions such as GP, kernel Support Vector Machine (SVM), and k-NN, are not truly able to model some *high-varying functions* from the input to the output space [3]

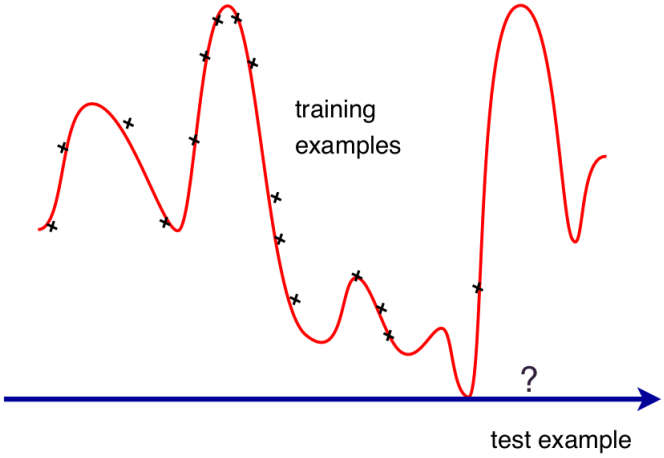


Figure 1.1: "local-smoothness" assumption by the kernel-based ML models (with one fixed generic kernel function) is not sufficient to estimate a highly-varying target function [3]

as these shallow models are based on the "local-smoothness" assumption, which means "very close" samples in the input space should be mapped nearby in the output space. Indeed, the main issue of these shallow methods is rooted from definition of "closeness" or proximity, i.e. how close two samples should be so that we expect their outputs should be similar? Can a fixed generic kernel function captures all the variations of a highly varying decision boundary or a target function? (Fig 1.1). In fact, one may require various kernel functions (with different kernel widths) to model different level of variations of a target function in the input space. On one hand, to attain a good generalization for such a high varying target function, e.g. a function with $2K$ "turning-points", a shallow learning model needs at least K training examples [3]. Hence, as the number of turning-points are increased, the number of training examples should be increased in order to attain a well-generalized learning model. On the other hand, the high volume of training data can be a computational burden for the kernel-based methods (e.g. Gaussian kernel SVM) as they need to compute a full kernel matrix $N \times N$ with N as the number of training samples.

Unlike the shallow models, deep models aim to define the closeness of two given sample by their similarity in the abstract level rather than in the raw input space or the conventional kernel space [3]. To capture such abstract-level similarities, deep learning models leverage the idea of hierarchical and distributed representation, where each layer represents a different level of abstract, i.e. the lower layers in this hierarchy represent the distributed and local spatial features of a given input while the higher layers offer more abstract representations of it (Fig. 1.2).

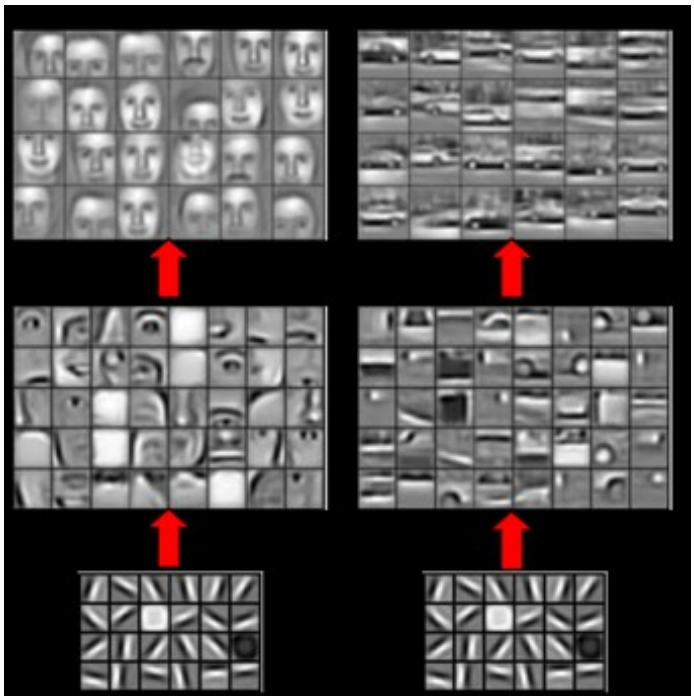


Figure 1.2: The hierarchical features learned by the first, second, and third layers of a deep learning model for human faces (1st column) and cars (2nd column). The early layer in this hierarchy represents the distributed and local spatial features of a given input while the upper layers offer more abstract representations of it. The image is adapted from [50].

1.2 Adversarial Example Attack Algorithms

Distributed and hierarchical representation learning empowered by GPUs and large-scale datasets allow deep learning models (e.g. CNNs) to outperform the conventional shallow counterparts on many vision tasks. Particularly, the deep models can perform nearly as good as human, this biologically smart creature, on some tasks such as digit recognition and face detection. However, Szegday et al. [96] revealed an intriguing phenomenon associated to deep neural networks as they show that CNNs recognize two visually alike images (one is clean image, another is its imperceptibly distorted one, named as an adversarial example) as two different objects. More precisely, while a classifier h can correctly classify a clean input, its adversarial counterpart, which is created by adding a *small and imperceptible distortion* to it, is **misclassified confidently** by this model. Nevertheless, human observers perceive the clean input and its adversarial equivalent as two identical instances (see Fig. 1.3). This finding emphasizes a serious gap between the human's perception system and that of its artificially intelligent counterpart, i.e. CNNs, and question the security and dependability of deep learning models in the presence of such potential threats. Later, Nguyen et al. [72] re-emphasized this

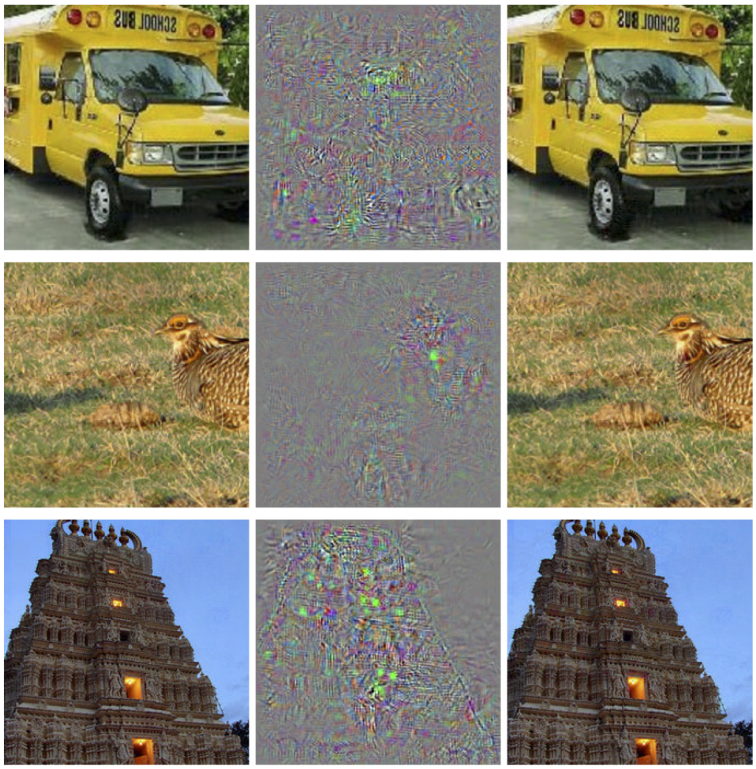


Figure 1.3: Some correctly classified clean images (left column) are distorted by some adversarial noises, magnified 10x, (middle column) to generate some benign-looking adversarial examples (right column). All the generated adversarial images are classified as "ostrich" by a victim classifier [96].

gap by showing that in fact CNNs recognize some unrecognizable (i.e. meaningless or noisy) synthetic images as perceptibly meaningful natural objects with high confidence.

Afterwards, a large body of researchers attracted to this new phenomenon, i.e. adversarial examples, to understand it, to investigate the vulnerability and robustness of machine learning models, and to defend them. Surprisingly, it has been shown that the machine learning models are vulnerable to a wide array of adversaries, generated by distinct attack algorithms (we elaborate later on these attacks). More interestingly, it has been also revealed that adversarial examples not only attack deep learners, but also other machine learning models such logistic regression [74] and K-nearest neighbors with small k [110].

It is worthwhile to note that while adversarial examples can break down a learning model at test-time, there are training-time threats, too. These type of attacks aim to a low-performing model at test-time by poisoning training data or illegally modifying the model itself. The train-time attacks are out of the scope of this thesis and the interested readers are referred to [73] for more details. In this thesis, our focus is on the test-time attacks, particularly adversarial example attacks.

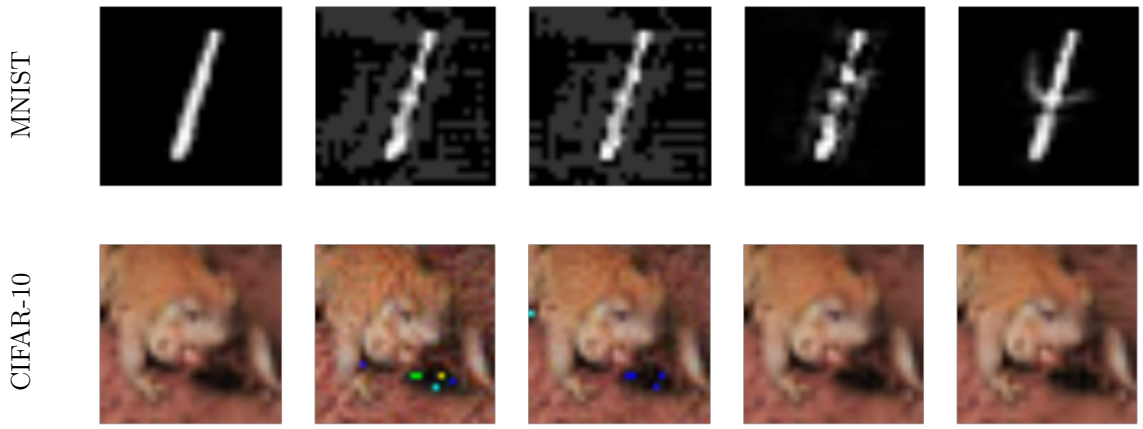


Figure 1.4: Various types of adversarial examples for an MNIST's instance (first row) and an CIFAR-10's sample (second row). From left to right: the original image, T-FGS [47], FGS [25], DeepFool [70], and C&W [10] (L_2) adversarial examples.

Originally, adversarial example generation for deep learning models was proposed in [96] through optimizing an objective function. More specifically, the following objective function is minimized for a given quadruple con siting of a clean sample $\mathbf{x}$, its true label (i.e. class) $k \in \{1, \ldots, K\}$, a target fooling class $k' \neq k$, and a classifier h:

$$\begin{aligned} & \text{minimize} \quad ||\epsilon||_2^2 \\ \text{s.t.} \quad & h(\mathbf{x} + \epsilon) = k' \\ & \mathbf{x} + \epsilon \in \mathbb{R}^D, \end{aligned} \tag{1.4}$$

where ϵ is the adversarial perturbation. As the generation of adversarial examples was initially introduced for image data, the input space can be represented by $[0, 1]^D$, instead of the unbounded space of $\mathbb{R}^D$. It is important to note that the generated adversarial example should be kept in the image space represented by $[0, 1]^D$, where D is the image space's dimensionality, i.e. $D = W \times H$ with W, H as width and height of the images, respectively. Minimizing exactly the above objective function is hard. To relax it, the following objective function is optimized instead, which is approximate the adversarial perturbation ϵ by:

$$\begin{aligned} \min_\epsilon \quad & c \cdot ||\epsilon||_2^2 + \mathcal{L}(\mathbf{x} + \epsilon, k'; h) \\ & \text{s.t. } \mathbf{x} + \epsilon \in [0, 1]^D, \end{aligned} \tag{1.5}$$

where c controls the magnitude of adversarial perturbation (using $l2$-norm) and $\mathcal{L}(., .; h)$, which is the loss of the classifier h the perturbed sample, i.e. $\mathbf{x} + \epsilon$.

Optimizing Eq. 1.5 does not necessarily generate an optimal adversary due to the non-convexity nature of deep neural networks. Furthermore, creating a single adversarial example using Eq. 1.5 is very slow. To generate adversarial examples with smaller time complexity, many novel adversarial example attack algorithms are proposed in the literature. However, for the

brevity reasons, we only review the widely-known ones in this chapter. The interested readers are referred to [76, 118] for an exhaustive review of a large spectrum of adversarial example attack algorithms and defence methods.

Fast Gradient Sign (FGS) [25] can generate adversarial examples very quickly in one-step. Inspired from the gradient ascend algorithm, the FGS algorithm modifies the clean image $\mathbf{x}$ to maximize the loss function on a given pair composed of a clean image $\mathbf{x}$ and its associated true label, represented by a one-hot vector $\mathbf{y}$. Formally, a FGS adversary is generated as follows:

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \text{sign}\left(\frac{\partial \mathcal{L}(h(\mathbf{x}), \mathbf{y})}{\partial \mathbf{x}}\right), \tag{1.6}$$

where ϵ, as the step-size, is large enough to achieve a misclassification in a single step. Note that $\frac{\partial \mathcal{L}(h(\mathbf{x}),\mathbf{y})}{\partial \mathbf{x}}$ and ϵ indicate the direction for generating an adversarial example and the magnitude of adversarial noise, respectively.

Targeted Fast Gradient Sign (T-FGS) takes a triple of $(\mathbf{x}, k, k')$ to perturb a clean sample $\mathbf{x}$ from the true class k such that the victim classifier misclassifies it as the selected target class $k' \neq k$. Therefore, contrary to FGS algorithm, a T-FGS adversarial example intents to fool the victim classifier to a target class, thus it generates a T-FGS adversary by minimizing the loss function of a classifier. Likewise [47], one can set the fooling target class as the least likely class $k' = \arg\min_{\{1,\dots,K\}} h(\mathbf{x})$, or the second most likely class $K' = \arg\max_{k' \neq k} h(\mathbf{x})$. Note that the target class k' is denoted by a one-hot vector $\mathbf{y}'$ with a single one at k'-th element.

Finally, a T-FGS adversary is produced by the gradient descent algorithm to minimizing the loss function of a classifier w.r.t. the input data for a given triple of a clean input image $\mathbf{x}$, its associated true class $\mathbf{y}$, and a fooling target class $\mathbf{y}'$ ($\mathbf{y} \neq \mathbf{y}'$):

$$\mathbf{x}' = \mathbf{x} - \epsilon \cdot \text{sign}\left(\frac{\partial \mathcal{L}(h(\mathbf{x}), \mathbf{y}')}{\partial \mathbf{x}}\right). \tag{1.7}$$

Iterative Fast Gradient Sign (I-FGS) is an iterative variant of the FGS algorithm [47]. The FGS algorithm finds an non-optimal perturbation as it takes a relatively large step-size, i.e. ϵ, in order to assure the generation of a successful attack in one step. Whereas, the I-FGS algorithm tends to create more optimal perturbations by iteratively distorting a given clean sample by a smaller noise magnitude ϵ'. Indeed, the step-size of I-FGS can be set to $\epsilon' = \frac{\epsilon}{T}$, where T and ϵ are the number of iterations and the total magnitude of noise, respectively. Formally, an I-FGS adversary sample is generated as follows:

$$\begin{aligned} &\mathbf{x}^0_{adv} = \mathbf{x}, \\ &\mathbf{x}^{t+1}_{adv} = \mathbf{x}^t_{adv} + \epsilon' \times \text{sign}\left(\frac{\partial \mathcal{L}(h(\mathbf{x}^t_{adv}), \mathbf{y})}{\partial \mathbf{x}^t_{adv}}\right), \end{aligned} \tag{1.8}$$

where $t = \{0, \dots T-1\}$. The iteration can be terminated once an adversarial example is found or allowing the number of iterations reach to T.

Projected Gradient Descent (PGD) [64] is another iterative variant of FGS, but unlike I-FGS, the total acceptable amount of distortion (which is denoted by α) can be larger than the multiplication of the step-size ϵ' and the number of iteration T, i.e. $\epsilon' \times T \geq \alpha$. It means that the generated sample can be perturbed more than the allowed perturbation α, but in order to keep the generated sample inside $\alpha-$ball around the clean sample $\mathbf{x}$, they used a project operation. This aims to project back the perturbed sample at each iteration t into a $\alpha-$ball around $\mathbf{x}$:

$$\begin{aligned} &\mathbf{x}_{adv}^{0} = \mathbf{x}, \\ &\mathbf{x}_{adv}^{t+1} = \pi_{\alpha,\mathbf{x}}\left(\mathbf{x}_{adv}^{t} + \epsilon' \times \text{sign}\left(\frac{\partial \mathcal{L}(h(\mathbf{x}_{adv}^{t}), \mathbf{y})}{\partial \mathbf{x}_{adv}^{t}}\right)\right), \end{aligned} \tag{1.9}$$

where $\pi_{\alpha,\mathbf{x}}$ is the project operation to keep the generated sample in a $\alpha-$ball around the clean sample $\mathbf{x}$. In addition, the initial samples (i.e. clean samples) can be perturbed by a random noise before starting perturbation process by PGD.

Carlini and Wagner (CW) [10] as another targeted attack attempts to find a sub-optimal perturbation through optimizing the following objective function for a given triple of an input image $\mathbf{x}$, its true label k, and a fooling target class k':

$$\begin{aligned} \min_{\epsilon} \quad & ||\epsilon||_p + c \cdot f(\mathbf{x}+\epsilon) \\ & \mathbf{x}+\epsilon \in [0,1]^D, \end{aligned} \tag{1.10}$$

where $f(\mathbf{x}') = max(0, \max_{i \neq k'} h(\mathbf{x}') - h_{k'}(\mathbf{x}'))$ with $\mathbf{x}' = \mathbf{x} + \epsilon$ and p as the parameter of the norm function can be set to 0, 1, 2, or ∞, reflecting different distance metrics (i.e. l_0, l_1, l_2, l_∞). Note that $f(.)$ can be replaced by a range of other functions, to read more about them, interested readers are referred to [10].

To remove the above explicit constrain ($\mathbf{x}+\epsilon \in [0,1]^D$), the authors used "change of variable" trick. More precisely, instead of optimizing over ϵ in Eq 1.10, they optimize over a new variable w, where $\epsilon = \frac{1}{2}(\tanh(w)+1) - \mathbf{x}$, satisfying automatically $\mathbf{x}+\epsilon \in [0,1]^D$. Therefore, they optimize the following objective function:

$$\min_{w} ||\frac{1}{2}(\tanh(w)+1) - \mathbf{x}||_2^2 + c \times f(\frac{1}{2}(\tanh(w)+1)), \tag{1.11}$$

where $f(\mathbf{x}') = \max\left(\max_{i \neq k'} h^L(\mathbf{x}') - h_{k'}^L(\mathbf{x}'), -\kappa\right)$ with $h_k^L(.)$ is the k-th element of the output vector by the last (i.e. L-th) layer of a CNN before the softmax activation. The hyper-parameter of κ can ensure achieving of a high confidence misclassification (by setting κ to a large value). A large κ creates instances with larger amount of distortions with high fooling confidence that can likely attack other learning models (i.e. high cross-model generalization).

DeepFool, as an untargeted attack, aims at efficiently generating (sub-)optimal adversarial perturbation for a given clean sample through orthogonal projection of it onto its nearest

decision hyper-plane. Initially, it is assumed that the decision boundaries are linear, thus in one single step an adversary can be found as shown in Fig. 1.5. However, since the assumption of linearity of hyper-planes does not hold true for the deep neural networks, the authors relaxed it and took several orthogonal-projection steps toward the solution, i.e. until an adversary is found. More details on DeepFool algorithm can be found in [70].

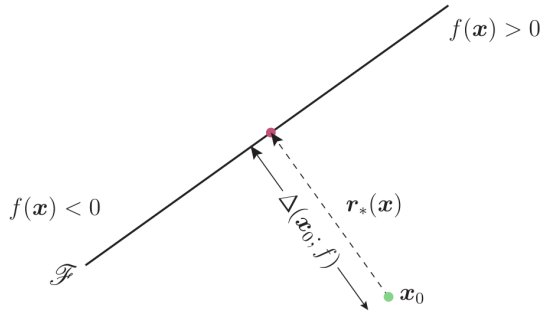


Figure 1.5: Deepfool creates an optimal adversarial example by orthogonal projection of a given clean sample x_0 to its nearest (linear) decision boundary (i.e. hyper-plane), where $\mathscr{F} = \{x : f(x) = 0\}$ represents the decision boundary of a binary classifier f [70].

1.2.1 Attack Models

As the aforementioned adversarial attack algorithms rely on a classifier/model for generating the adversarial examples, attackers should thus have access to a model. They might have a full access to the victim model (in white-box model), at all any access to it (in black-box model), or have a partial information about the victim model (in gray-box model). In the following, these attack models are briefly explained:

- **Black-box Attack**: In the black-box attack model, an attacker has no knowledge about the victim model of an intelligent system, including its structure, parameters, training set, etc, as the underlying ML models may be kept hidden from any external accesses. In this case, the end-users, including attackers, can only access the predictions (decisions) of a model such as online services (face recognition or image tagging) provided by online platform clarifai. Despite secrecy of the victim models, attackers can still attack and fool them by leveraging cross-model generalization property of adversarial example attacks [96, 60, 74]. This property explains that the adversarial examples crafted by a different local model can be easily transferred to mislead other learning models, showing the severity of vulnerability of deep neural networks to the adversarial examples. Another possible way to craft black-box adversaries, without knowing the exact information about the victim model, is to approximate the victim model by learning a local proxy model on a set of collected clean samples which are labelled by the victim classifier [74]. Most likely the proxy classifier and the victim one predict similarly, since the proxy is trained to behave like the victim

model. The adversarial examples crafted by such a proxy model, which is achieved through reverse engineering, can be considered as a kind of black-box adversarial examples. Since the attacker still does not know the real structure of victim model or its whole training set that being used for its training, such reverse engineering attacks can be regarded as black-box attacks.

- **Gray-box Attack**: In this model, it is often assumed that the attackers know some partial information about the underlying ML model, such as structure of the model, i.e. the number of layers and filters, filter size, and etc., or they are aware of the specific underlying defense mechanism used to defend the ML model. However, the attackers do not have access to the exact numerical values of the parameters of the (defended) ML model. Recall that in the black-box model, the attacker has *no knowledge* about the victim classifier (its structure and its parameter). In the gray-box attack model, by leveraging the available but partial information about the victim model, the attacker can increase the chance of generating successful attacks.
- **White-box Attack**: In the white-box model, we assume the attackers completely access to the underlying victim model, including its architecture, parameters and hyper-parameters. The adversaries generated directly from the victim model are called white-box adversarial examples. Indeed, in this model, we may assume that the attackers are the developers of an intelligent system, who have a full access to the underlying ML models, and they aim to assess the robustness of the underlying model under a worst-case scenario, i.e. when the attacker knows completely the victim model.

1.2.2 Defense Techniques

To defend the classifiers against adversarial attacks, two main tracks of approaches are active; in the first track, which we call "correct classification track", the major goal is to robustify the classifiers to adversarial example attacks such that any given adversarially perturbed input samples can be correctly classified. In other words, a robust model in this track should output the identical predictions for a clean sample and all of its adversarial counterparts, i.e. $h(\mathbf{x}) = h(\mathbf{x} + \epsilon)$. The other track, however, aims at reducing the risk of adversarial examples by identifying them. Compared to the former track, the main goal of the latter one is not to correctly classify the adversarial examples, but rather to detect and reject them to be classified. This allows the system to be absent from making automatic but wrong decisions in the presence of the misleading and hostile samples. Indeed, the cost of making wrong decisions in the face of adversaries by an intelligent system, e.g. a medical diagnostic system, can be remarkably higher than making no decision (by avoiding to make any decision). In the case of rejection (denying to classify a given sample), a human operator can keep the system under his control by safely handling such hostile situations. Carlini et al. [9] have meticulously shown that both detection and correct classification of adversaries are equally difficult to address.

1.2.3 Correct Classification of Adversaries

Adversarial Training

As the first and the most famous attempt for robustifying CNNs to adversarial example attacks, researchers have proposed the methods that ultimately enable CNNs to correctly classify virtually any kinds/types of adversaries. To this end, Goodfellow et al. [25] initially proposed the *"adversarial training"* method, where the model is simultaneously trained on both clean samples and their FGS adversarial counterparts. The rational behind adversarial training is that a model should be able classify two samples (i.e. a clean sample and its adversarial counterparts), which are locating very nearby in the input space, as the same output. Latter on, many researchers followed the adversarial training method, but they generate various types of adversarial examples in different settings (varying hyper-parameters) in order to enrich an adversarial training set. This training set can either be generated on-the-fly (during training time) by using the underlying model [47, 41, 64] or pre-generated from a single model [70, 82] (which is different from the underlying model) or from an ensemble of models [101].

Although adversarially training a CNN can increase its robustness to some types of adversarial examples, particularly those that are seen during training, the researchers demonstrated several limitations of adversarial training method. The first and most important limitation is the lack of guarantee of such adversarially trained models to the *unseen or undiscovered types of adversarial examples* [119, 100]. In other words, adversarially training a CNN on one type of adversaries, e.g. those generated by l_∞ or l_2 norms, does not necessarily bestow the models robustness to other unseen adversarial examples [87, 61]. For example, Tramer et al. [100] recently show a model robust to a specific type of adversaries, e.g. l_∞, is vulnerable to other types, e.g. l_2 type adversaries. Therefore, it seems that to achieve robustness to a large spectrum of types of adversaries, accessing to an all-inclusive training set of adversaries from different types is crucial. However, generation of such exhaustive adversarial training set is computationally very costly or even impossible, which can also result in an increment in the training time.

Denoising of Adversaries

Devising pre-processing methods [56, 27] to de-noise the adversarial instances is a sub-category of correct classification adversaries. In [56], using a training set of adversaries, the authors train a denoiser network for removing adversarial perturbation from the adversarial images before feeding them to some the target CNNs. However, training an effective and well-generalized denoiser network still requires a diverse set of adversaries.

1.2.4 Detection of Adversaries

It is argued in [119, 100] that dversarially training a CNN (for classifying them correctly) may cause a degradation in generalization performance as adversarial learning forces the model to learn two seemingly visually similar but statistically different samples as the same class. Despite of the visual resemblance of the adversaries to their clean counterparts, they are in fact statistically different [26], meaning that they can be drawn from two different data generation distributions. From manifold perspective, Tanay et al. [98] argued that the adversaries are not lying on data manifold even though they are locating near to it. Therefore, considering these arguments, it seems justified that the vanilla models (e.g. CNNs) classify clean samples and their benign-looking adversarial counterparts very differently.

As a result, with the aim of reducing the risk of adversaries, one can devise a detector for discriminating the adversaries from the clean ones, then rejecting to classify them [19, 63, 69, 67], instead of striving to train CNNs to correctly classify both adversaries and clean samples. The methods categorized under this track, i.e. adversarial detection, aim to learn a classifier that induces a minimum risk over the clean and adversarial samples. We can define the risk function $\mathcal{R} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ for a given input sample $\mathbf{x} \in \mathcal{X}$ and its true label $\mathbf{y} \in \mathcal{Y}$ w.r.t. a learning model h as follows:

$$\mathcal{R}(\mathbf{x}, \mathbf{y}; h) = \begin{cases} 0 & h(\mathbf{x}) = \mathbf{y} \\ 1 & h(\mathbf{x}) \neq \mathbf{y} \\ \delta & h \text{ rejects } \mathbf{x} \end{cases} \tag{1.12}$$

where $0 \leq \delta < 1$ ensures the risk of rejection is smaller than that of wrong prediction, which is one. The empirical risk rate on a given set of labeled samples, e.g. $\mathcal{S}$, w.r.t a classifier h is computed by $\frac{1}{N} \sum_{i=1}^{N} \mathcal{R}(\mathbf{x}_i, \mathbf{y}_i; h)$. Regardless of adversarial context, the rejection can be obtained by using a threshold on predictive confidence of a classifier [2, 53], an auxiliary model [20], or explicit rejection to an extra class, which is added to the output of the original classifier [14, 38].

Note that defining simply a threshold on the original predictive confidences of a CNN, with the aim of detecting adversaries, does not perfectly work since the CNNs are generally uncalibrated classifiers [29], meaning their predictive confidences on the adversaries are as high as those of clean samples, leaving them indistinguishable. Therefore, it is essential to somehow process the predictions of a CNN to create a gap between the predictive confidences of adversarial examples and those of clean ones.

A large body of research work [2, 19, 26, 62, 67, 63, 69, 53] has been proposed to train an auxiliary classifier/regressor on a joint training set of clean samples and their adversarial counterparts in order to have a detector automatically identifying and distinguishing adversaries

from their clean counterparts.

Given that the adversaries are *not lie* exactly on the sub-manifolds of clean samples [98], the located samples far away from the sub-manifolds should be identified as adversaries. Considering this hypothesis, Feinman et al. [19] proposed to compute the average kernel distance (using a Gaussian kernel) between a given sample and each of K sub-manifolds (associated to each class of a classification problem) in the feature space (i.e. the last convolution layer of a pre-trained CNN). To automatically discriminating the samples according to their distance to data manifolds, they trained a binary classifier on a joint set of adversarial examples (labeled as positive class), clean (as negative class), and noisy samples (as negative class).

Similarly, Ma et al. [63] trained a logistic classifier for identifying adversaries according to another metric, namely Local Intrinsic Dimentionality (LID). To train the classifier, the LID of all samples, including adversaries, clean and random noise samples, are computed using the penultimate layer of a pre-trained vanilla CNN. However, the LID metric can be computed using all or a number of the intermediate layers. Using the LID value(s) of a give sample, the trained classifier can discriminate adversaries from their clean counterparts.

Assuming multi-modal Gaussian distribution for the clean data in the feature space, Lee et al. [53] proposed to use Mahalanobis distance in the feature spaces, which is obtained from some intermediate layers of a pre-trained CNN. For each sample, to produce a final score by mixing all Mahalanobis distances (scores) achieved by the layers, a regressor is used. This regressor then is trained on a training set of clean samples and adversarial examples such that to create a gap between the final confidence scores of the clean samples and those of adversaries. Finally, using a threshold on these scores, the adversaries are separated from the clean ones.

Although these detectors (regressor/classifier) can detect some types of adversaries, the generalization ability of their detectors to detect unseen types of adversaries is not clearly investigated [2]. Thus, likewise adversarial training, training such auxiliary detectors might need a diverse set of adversaries, which is expensive to attain.

Without training an auxiliary detector, Hosseini et al. [38] used a CNN with an extra class added to its output, performing like an explicit rejection option. We call this CNN as augmented CNN (A-CNN). Inspired from adversarial learning, the authors trained an A-CNN using a set of adversaries such that they should be classified as the extra class (equivalent to rejection). However, this proposed method of training A-CNN still suffers from the limitations of the adversarial learning discussed in Section 1.2.3, such as the need of having a comprehensive set of adversaries to ensure achieving the generalization ability for a wide-range of adversaries, particularly the unseen ones.

[2] However, Lee et al. [53], have shown empirically the generalization ability of their regressor (mixing the Mahlanobis distance based scores) on a some types of unseen adversarial example attacks.

By the use of diversity in the ensemble of models, Sharif et al. [90] aimed to detect the adversaries. More precisely, the authors *explicitly* trained the CNNs in the ensemble on a set of training adversarial examples so that encourage the members to diversely predict these samples, leading to prediction with higher uncertainty for them. Likewise the aforementioned approaches, their approach is dependent on having access to a comprehensive set of adversaries. Moreover, Kariyappa et al. [44] proposed an ensemble of CNNs where they explicitly force each pair of CNNs to have dissimilar fooling directions for each given training samples, in order to promote diversity in the presence of adversaries. However, computing similarity between the fooling directions by each pair of members for every given training sample is computationally expensive, resulting in a drastic rise in training time.

1.3 Out-of-Distribution Samples

In supervised learning, it is generally assumed that a training set and a held-out test set are drawn independently from an identical data distribution, called an *in-distribution set*. While this assumption can be true for many controlled laboratory environments, it rarely holds for the real-world in which the samples can be drawn from both in-distribution and from any other distributions, called *out-of-distribution* (OOD) data [3]. The OOD dataset contains the samples that are semantically and statistically different from those in-distribution ones. For instance, a real-world image-based digit reader model may be exposed to an enormous types of images that do not contain any digit, in which we expect to deny to classify such OOD samples.

Unfortunately, it has been demonstrated that CNNs are not robust to OOD samples, as they are classifying them confidently to their pre-defined classes. For example, a three-layer CNN trained on MNIST confidently classifies the exhibited OOD samples in Fig 1.6 as 0-9 digit classes. Although OOD samples are not inherently adversaries or hostile samples, they still can impose a high risk on the CNNs. Indeed, the CNNs as uncalibrated classifiers [29, 34] are not generally able to predict OOD samples with low confidence, making them indistinguishable from the clean legitimate instances. Identifying the OOD samples, then rejecting them, is highly demanding for the security-sensitive applications in order to make reliable decisions as well as to avoid some illegal actions, which are arising from such illegitimate instances [105, 51].

Interestingly, in addition to CNNs (as discriminative models), it has been shown that deep generative models also suffer from the existence of OOD samples as they produce higher likelihood for OOD sets than the in-distribution set [13, 71, 80]. For example, Nalisnick et al. demonstrated that likelihood of a deep generative model trained on CIFAR-10 creates higher likelihood on SVHN than CIFAR-10. To explain this phenomenon, Ren et al.argued that deep generative

[3]There are the data distributions that are similar to the in-distribution but their samples are shifted, such as M-MNIST and SVHN datasets for MNIST (as in-distribution). In this thesis, we do not consider the shifted datasets as they are well-known in domain adaptation.

Figure 1.6: The OOD samples from NotMNIST (1st row) and CIFAR-100 (2nd row) are confidently classified as digit 0-9 (left to right) by a three-layer CNN trained on MNIST.

models focus heavily on background information rather than semantic information of a given sample for calculation of likelihood ($p(\mathbf{x})$). To cancel the effect of background in likelihood and focus on semantic information, they, instead, propose likelihood ratio; the ratio between likelihoods of semantic part of an input sample which is computed by two trained deep generative models –one is trained on clean in-distribution samples and the other is trained on the perturbed in-distribution samples. Then, regarding this ratio, they empirically shown that the OOD sets have lower likelihood ratio than that of in-distribution, leading to detect them. As the focus of this thesis is on discriminative models such as the variants of CNNs to detect OOD samples, we skip to review other OOD detection methods based on deep generative models(e.g. [13, 71].

1.3.1 Post-processing Methods

To tackle this challenge (i.e. very certain predictions for OOD samples), some post-processing approaches [16, 35, 53, 55, 43] proposed to transform the predictive confidence of the OOD samples, which are achieved by a vanilla CNN, to a lower confidence spectrum while keeping the in-distribution samples at a higher confidence spectrum. Using a threshold on the confidence, the OOD samples are then separated from the in-distribution ones.

Towards an open-set recognition [86, 85], where the unknown samples, e.g. adversarial examples and OOD data, can be detected, Bendale et al. [2] proposed a post-processing approach, where the softmax layer of a pre-trained vanilla network is replaced by OpenMax, a new output-layer that is incorporating an extra class (equivalent to a rejection option) to the usual class-outputs. By adopting the nearest class mean method [99], OpenMax computes a mean for each class in the feature space (achieved by the penultimate layer) using the training samples in order to compute the conditional probabilities of being outlier w.r.t. each class. These conditional probabilities are then used for detecting the unknown samples, thus took a step toward addressing the open-set recognition problem.

To encourage more calibrated predictions, the renowned temperature scaling [29, 37] multiplies the output of a CNN, before its softmax activation, by $\frac{1}{T}$ (a temperature factor) and then

re-computes new class conditional probabilities as follows:

$$p(k|\mathbf{x}) = \frac{\exp(h_k^L(\mathbf{x})/T)}{\sum_{j=1}^{K} \exp(h_j^L(\mathbf{x})/T)}. \tag{1.13}$$

Temperature scaling leads to a more smooth class probabilities for a large T, meaning that the larger T, the smoother the class probabilities will be. Therefore, by choosing a large T, the conditional class probabilities become less spiky (i.e. uncalibrated) as they were before the temperature scaling. By adopting the temperature scaling as a post-processing method as well as modifying the input samples by adding a FGS-type noise, Liang et al. [55] aim to create a gap between the predictive confidence of the OOD samples and that of in-distribution samples. Then using a threshold on the new confidence scores, the OOD samples are detected.

Using the Gaussian assumption, Lee et al. [53] assume the class-conditional probability as a Gaussian distribution function in the feature spaces, achieved by the output of each layer of a given CNN. Then, the Mahalanobis distance between the given sample and each class is computed for finding the nearest class to this sample (with the smallest Mahalnobis distance), the smallest distance, then, represents a confidence score for this sample. As a CNN has L layers, one can compute L confidence scores for each input sample. Combining these scores by using an auxiliary regressor results a final confidence score, which is used then for discriminating the OOD samples from the in-distribution ones.

Despite the simplicity and efficacy of the aforementioned post-processing approaches, their performances are highly depend on several additional hyper-parameters such as temperature, magnitude of additive noise, or the parameters of an auxiliary regression function, which should be carefully tuned for each OOD set separately. However, in the reality, we do not have access to all possible OOD sets apriori for tuning these hyper-parameters.

1.3.2 Calibrated Models

Unlike the post-processing methods, the end-to-end calibrated methods aims at training of either a single model [65, 36, 52, 107] or an ensemble of them [48] for producing low confidence predictions on any (seen and unseen) OOD samples while maintaining the confidence of in-distribution samples high.

Lakshminarayanan et al. [48] have shown the effectiveness of a simple ensemble of deep neural networks (initialized randomly with different values for their weights) as a calibrated method, where diversity in the ensemble leads to low predictive confidence for OOD samples while high confidence for the in-distribution samples. This creates a gap in their predictive confidences, then the OOD samples can be distinguished from the in-distribution samples by setting a threshold on the predictive confidence.

To train an end-to-end calibrated model, Masana et al. [65] have incorporated an OOD set to an in-distribution set to train a modified Siamese network in order to keep the in-distribution samples nearby while pushing the OOD training samples away from the in-distribution ones. Other researchers [36, 52, 107] proposed to train a vanilla CNN on an OOD set along with the given in-distribution set in order to explicitly force it to output the calibrated predictions. To attain such calibrated estimation for OOD samples, the common objective function of CNN (i.e. cross-entropy in Eq. 1.3) is modified by adding a KL-divergent term; measuring the distance between the predictive class-conditional probabilities and the expected class-conditional probabilities, i.e. $\mathcal{U} = (\frac{1}{K}, \ldots, \frac{1}{K})$, for a given OOD sample $\mathbf{x}' \in \mathcal{D}_O$ (where $\mathcal{D}_O$ is an OOD data distribution). Thus, the loss function of a calibrated CNN is defined as:

$$-\mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}[\mathbf{y}\log h(\mathbf{x})] + \beta\mathbb{E}_{\mathbf{x}'\in\mathcal{D}_O}[KL(\mathcal{U}||h(\mathbf{x}'))], \tag{1.14}$$

where $\beta > 0$ is a penalty hyper-parameter. To train such end-to-end CNN-based models, one can leverage a natural OOD set (as of $\mathcal{D}_O$) [5, 36, 65] or a set of synthetically-generated OOD samples [52, 117, 34].

Apart from the computational cost of generating a set of synthesized samples, Hendrycks et al. [36] have shown that a calibrated CNN trained on a proper naturalistic OOD set can outperform that of trained on GAN-generated synthetic samples [52]. Although the natural OOD sets can be more suitable for OOD training than the generated ones, it is not clear how one should select a proper natural OOD set among the available ones in order to induce a well-generalized model to detect unseen OOD samples that are drawn from any data distributions. In this thesis, particularly in chapter 3, we propose three metrics to differentiate natural OOD sets for the purposes of training such end-to-end calibrated models.

Chapter 2

Ensemble of Specialists

2.1 Introduction

CNNs are strongly vulnerable to minor and imperceptible adversarial modifications of input images a.k.a. adversarial examples or adversaries. While identifying such benign-looking adversaries from their appearance is not always possible for human observers, distinguishing them from their predictive confidences by CNNs is also challenging since these networks, as uncalibrated learning models [29], misclassify them with high confidence.

In this chapter, we aim at detecting adversarial examples by predicting them with high uncertainty (low confidence) through leveraging *diversity in ensemble of CNNs*, without requiring any form of adversarial training. To do this, we define an ensemble of specialists so as to encourage divergent predictions in the presence of adversarial examples, while making consistent predictions for clean samples.

Some ensemble-based approaches [44, 94] have been shown to be effective for mitigating the risk of being fooled by adversarial examples. Strauss et al. [94] demonstrated that the ensembles of CNNs that are created by bagging and different random initialization are less fooled by adversarial examples, compared to a single model. Recently, Kariyappa et al. [44] have proposed an ensemble of CNNs where they explicitly force each pair of CNNs to have dissimilar fooling directions for each given training samples, in order to promote diversity in the presence of adversaries. However, computing similarity between the fooling directions by each pair of members for every given training sample is computationally expensive, resulting in a drastic rise in training time. As another ensemble-based method for detection of adversaries, Sharif et al. [90] aimed to detect adversaries by use of an diverse ensemble, where its members are *explicitly* trained to classify a set of training adversarial examples diversely in order to have divergent predictions for adversarial examples. Although their approach is similar to ours in spirit (divergent predictions by the members of the ensemble), they achieve this through adversarial training of ensemble, need a training set of adversaries. However, our ensemble aims

to achieve divergent prediction for adversaries without an explicit use of adversarial training.

To build a diverse ensemble, we propose making *specialists ensemble*, where each specialist is responsible of classifying a different subset of classes. The specialists are defined so as to encourage divergent predictions in the presence of adversarial examples, while making consistent predictions for clean samples. In Fig. 2.1, using a toy example, we demonstrate the role of diversity in our specialist ensemble for driving the prediction of black-box adversaries with low confidence and hardening generation of high confidence white-box adversarial examples. We also devise a simple and computationally efficient voting mechanism to merge the specialists' predictions for computing final predictions.

As a result of our ensemble of diverse specialists and voting mechanism, we are enforcing a gap between the predictive confidences of adversaries (i.e., low confidence predictions) and those of clean samples (i.e., high confidence predictions). By setting a threshold on the prediction confidences, we can expect to identify the adversaries properly. Interestingly, we prove that the predictive confidence of our method in presence of disagreement (high entropy) in the ensemble is upper-bounded by $0.5 + \zeta$ (see corollary 1), where $\zeta = \frac{1}{2M}$ with M is the size of our ensemble (i.e. number of members of the ensemble). This upper-bound allows us to set a *global-fixed threshold (i.e., $\tau = 0.5$),* on the predictive confidence, eliminating the need to fine-tune the threshold.

Using standard benchmarks in image classification, we empirically show that several types of black-box attacks can be effectively detected with our proposal due to their low predictive confidence (i.e., ≤ 0.5). Also, we show that attack-success rate for generating white-box adversarial examples using the ensemble of specialists is considerably lower than those of a single generalist CNN and an ensemble of generalists (a.k.a. pure ensemble). These empirical results demonstrate how, without adversarial training and only by diversity in the ensemble, one may design more robust CNN-based classification systems.

2.2 Ensemble Construction

We define the expertise domain of the specialists (i.e. the subsets of classes) by separating each class from its most likely fooled classes. We later show in Section 2.4 how separation of each class from its high likely fooling classes can promote entropy in the ensemble, which in turns lead to predicting adversaries with low confidence (high uncertainty).

To separate the most fooling classes from each other, we opt to use the fooling matrix of FGS adversarial examples $\mathbf{C} \in \mathbb{R}^{K \times K}$ that reveals the clean samples from each true class have a high tendency to being fooled toward a limited number of classes rather than uniformly toward all of them (Fig. 2.2(a)). The motivation for using FGS adversaries is two-fold: their generation is computationally inexpensive, and they are known to be highly transferable to many other

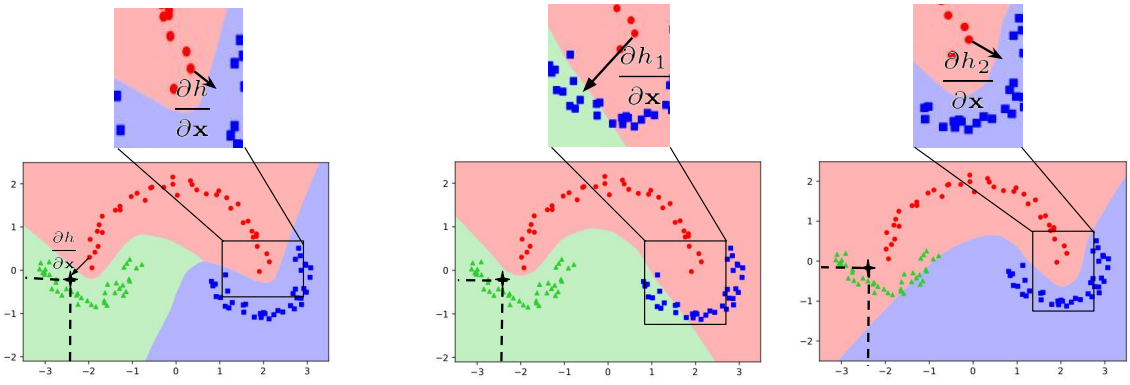


Figure 2.1: A schematic explanation of ensemble of specialists for a 3-class classification. On the left, a generalist Multi Layer Perceptron (MLP) ($h(\cdot)$) trained on all 3 classes. On the middle and the right, two specialist binary classifiers $h_1(\cdot)$ and $h_2(\cdot)$ are trained on different subsets of classes, i.e. respectively (red,green) and (red, blue). A black-box attack, shown by **a black star**, which fools a generalist classifier (left), can be classified as different classes by the specialists, creating diversity in their predictions. Moreover, generation of a white-box adversarial example by the specialists can create two different fooling directions toward two unlike fooling classes. The fooling directions (in term of derivatives) are shown by black arrows in zoomed-in figures. Such different fooling directions by the specialists can harden generation of high confidence white-box attacks (section 2.4). *Thus, by leveraging diversity in ensemble of specialists, without the need of adversarial training, we may mitigate the risk of adversarial examples.*

classifiers. In other words, other classifiers, which have different structures and parameters than a victim classifier, behave in a similar manner in the presence of FGS adversaries as they are fooled to the same classes [60, 96, 11].

Using each row of the fooling matrix (i.e. $\mathbf{c}_i$), we define two expertise domains for i-th true class so as to split its high likely fooling classes from its less likely fooling classes as follows (Fig. 2.2(b)):

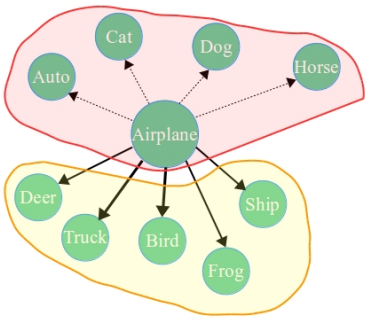
- The subset of high likely fooling classes of i: $\mathbb{U}_i = \cup\{j\}$ if $c_{ij} > \mu_i$, $j \in \{1, \dots, K\}$;
- the subset of less likely fooling classes of i: $\mathbb{U}_{i+K} = \{1, \dots, K\} \setminus \mathbb{U}_i$;

where $\mu_i = \sum_{j=1}^{K} c_{ij}$ (average of fooling rates of i-th true class). Repeating the above procedure for all K classes makes $2K$ subsets (expertise domains) for a K-class classification problem. We add another set, which includes all of K classes ($\{1, \dots, K\}$), to these $2K$ subsets (this is done to add a generalist CNN to this ensemble of specialists). Thus, the number of the expertise domains is $M = 2K + 1$. However, if there are some duplicated subsets among the $2K$ ones, we remove the duplicated subsets. Therefore, the number of expertise domains can be less than $2K + 1$, i.e. $M \leq 2K + 1$.

Afterwards, for each expertise domain (subsets), one specialist is trained in order to form

	Airplane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	0.00	1.60	22.00	3.88	15.20	0.22	18.46	1.70	17.70	19.24
Auto	2.70	0.00	7.20	1.94	5.46	0.06	7.86	0.90	8.50	65.38
Bird	3.70	0.58	0.00	13.34	32.24	0.98	33.02	2.78	3.28	10.08
Cat	0.82	0.30	12.58	0.00	20.92	6.86	43.38	6.24	1.46	7.44
Deer	0.90	0.24	21.00	7.30	0.00	0.58	51.94	8.94	0.94	8.16
Dog	0.66	0.26	16.94	14.92	15.88	0.00	36.96	5.86	1.74	6.78
Frog	0.60	0.76	28.02	16.94	41.14	0.52	0.00	3.48	0.74	7.80
Horse	0.92	0.36	6.40	5.66	32.44	1.62	47.16	0.00	0.98	4.46
Ship	11.10	4.12	20.94	4.42	7.12	0.10	14.28	0.84	0.00	37.08
Truck	3.88	23.92	7.24	2.64	13.44	0.22	42.06	1.74	4.86	0.00

(a) CIFAR-10 FGS fooling matrix



(b) The expertise domains of "Airplane" class

Figure 2.2: (a) Fooling matrix of FGS adversaries for CIFAR-10, which is computed from 5000 randomly selected training FGS adversaries (500 per class). Each row shows the fooling rates (in percentage) from a true class to other classes (rows and columns are true and fooling classes, respectively). (b) An example of forming two expertise domains for class "Airplane" according to the left fooling matrix. The bold (dotted) arrow shows that class "Airplane" is high (less) likely fooled to the class indicated at the end of the arrow. Collecting its high likely fooled classes (indicated in yellow zone) into one set forms one of the expertise domains and the rest of classes ,i.e. the less likely fooled classes (in red zone), are forming another expertise domain.

an ensemble of specialist CNNs. Note that a CNN (called generalist CNN), which is trained on all the classes, is also added to this ensemble. The ensemble involving of $M \leq 2K+1$ members is represented by $\mathcal{H} = \{h^1(\cdot), \ldots, h^M(\cdot)\}$, where $h^j(\cdot) \in [0,1]^K$ is the j-th individual CNN mapping a given input to the conditional probability over its expert classes. Thus, the probability of the classes out of its expertise domain is fixed to zero.

2.3 Voting Mechanism

To compute the final prediction out of our ensemble for a given sample, we need to activate the relevant specialists, then averaging their prediction along with that of generalist CNN. Note we can not simply use the generalist CNN for activating specialists since in the presence of adversaries it can be fooled, so this causes the selection (activation) of the wrong specialists. In Algorithm 1, we devise a simple and computationally efficient voting mechanism to activate those relevant specialists, then averaging their predictions.

Let us first introduce the following elements for each class i:

- The actual number of votes for i-th class by the ensemble for a given sample $\mathbf{x}$: $v_i(\mathbf{x}) = \sum_{j=1}^{M} \mathbb{I}\left(i = \text{argmax}_{\{1,\ldots K\}}\, h^j(\mathbf{x})\right)$, i.e. it shows the number of the members that classify $\mathbf{x}$ to i-th class.

Algorithm 1 Voting Mechanism

Input: Ensemble $\mathcal{H} = \{h^1, \ldots, h^M\}$, expertise domains $\mathbb{U} = \{\mathbb{U}_1, \ldots, \mathbb{U}_M\}$, input $\mathbf{x}$

Output: Final prediction $\bar{h}(\mathbf{x}) \in [0,1]^K$

1: $v_k(\mathbf{x}) \leftarrow \sum_{j=1}^{M} \mathbb{I}\left(k = \text{argmax}_{\{1,\ldots,K\}}\, h^j(\mathbf{x})\right),\ k = \{1, \ldots, K\}$

2: $k^* \leftarrow \text{argmax}_{k=1}^{K}\, v_k(\mathbf{x})$

3: **if** $v_{k^*}(\mathbf{x}) = \lceil \frac{M}{2} \rceil$ ▷ Agreement on k^* as the winner class

4: $\quad \mathcal{H}_{k^*} \leftarrow \{h^i \in \mathcal{H} \,|\, k^* \in \mathbb{U}_i\}$ ▷ Activating the members voting for k^*

5: $\quad \bar{h}(\mathbf{x}) \leftarrow \frac{1}{|\mathcal{H}_{k^*}|} \sum_{h^i \in \mathcal{H}_{k^*}} h^i(\mathbf{x})$

6: **else** ▷ No agreement on a winner class

7: $\quad \bar{h}(\mathbf{x}) \leftarrow \frac{1}{M} \sum_{h^i \in \mathcal{H}} h^i(\mathbf{x})$

8: **return** $\bar{h}(\mathbf{x})$

- The maximum possible number of votes for i-th class is $\lceil \frac{M}{2} \rceil \leq K + 1$. Recall that for each row, we split all the K classes into two expertise domains, where class i is included in one of them. Considering all K rows, we have K expertise domains that include class i. Given these K expertise domains and the generalist domain (with all the classes i.e. $\{1, \ldots, K\}$), we end up with a maximum of $K + 1$ subsets that involve class i.

As described in Algorithm 1, for a given sample $\mathbf{x}$, if there is a class with its actual number of votes equal to its expected number of votes, i.e. $v_i(\mathbf{x}) = \lceil \frac{M}{2} \rceil$, then it means all of the specialists, which are trained on i-th class, are simultaneously voting (classifying) for it. We call such a class *a winner class*. Then, the specialists CNNs voting for the winner class are activated to compute the final prediction (lines 3–5 of Algorithm 1), producing a certain prediction (with high confidence). Note that in the presence of clean samples, the relevant specialists in the ensemble are expected to do agree on the true classes since they, as strong classifiers, have high generalization performance on their expertise domains.

If no class obtains its maximum expected number of votes (i.e. $\nexists i,\ v_i(\mathbf{x}) = \lceil \frac{M}{2} \rceil$), it means that the input $\mathbf{x}$ leads the specialists to disagree on a winner class. In this situation, when no agreement exists in the ensemble, all the members should be activated to compute the final prediction (line 7 of Algorithm 1). Averaging of the predictions by all the members leads to a final prediction with high entropy (i.e. low confidence). Indeed, a given sample that creates a disagreement (entropy) in the ensemble is either a hard-to-classify sample or an abnormal sample (e.g. adversarial examples).

Using the voting mechanism for this specialists ensemble, we can create a gap between the predictive confidences of clean samples (having high confidence) and those of adversaries (having low confidence). Finally, using a threshold τ on these predictive confidences, the unusual samples are identified and rejected. In the following, we argue that our voting mechanism

enables us to set a global fixed threshold $\tau = 0.5$ to perform the identification of adversaries. This is unlike some threshold-based approaches [53, 2] that need to tune different thresholds for various datasets and their types of adversaries.

Corollary 1. *In a disagreement situation, the proposed voting mechanism makes the highest predictive confidence to be upper-bounded by* $0.5 + \epsilon'$ *with* $\epsilon' = \frac{1}{2M}$.

Proof. Consider a disagreement situation in the ensemble for a given $\mathbf{x}$, where all the members are averaged to create $\bar{h}(\mathbf{x}) = \frac{1}{M}\sum_{h^j \in \mathcal{H}} h^j(\mathbf{x})$. The highest predictive confidence of $\bar{h}(\mathbf{x})$ belongs to the class that has the largest number of votes, i.e. $m = \max[v_1(\mathbf{x}), \ldots, v_K(\mathbf{x})]$. Let us represent these m members that are voting to this class (for example k-th class) as $\mathcal{H}_k = \{h^j \in \mathcal{H} \,|\, k \in \mathbb{U}_j\}$. Since each individual CNNs in the ensemble is basically a uncalibrated learner (having very high confident prediction for a class and near to zero for the remaining classes), the confidence probability of k-th class by those excluded members (i.e. $\mathcal{H} \setminus \mathcal{H}_k$ those that do not vote for k-th class) can be negligible. Therefore, their prediction for k-th class can be simplified as $\bar{h}_k(\mathbf{x}) = \frac{1}{M}\sum_{h^j \in \mathcal{H}_k} h^j_k(\mathbf{x}) + \frac{\epsilon}{M} \approx \frac{1}{M}\sum_{h^j \in \mathcal{H}_k} h^j_k(\mathbf{x})$ (the small term $\frac{\epsilon}{M}$ is discarded). Then, from the following inequality $\sum_{h^j \in \mathcal{H}_k} h^j_k(\mathbf{x}) \leq m$, we have $\frac{1}{M}\sum_{h^j \in \mathcal{H}_k} h^j_k(\mathbf{x}) \leq \frac{m}{M}$ (I).

On the other hand, due to having no winner class, we know that $m < \lceil \frac{M}{2} \rceil$ (or $m < \frac{M}{2} + \frac{1}{2}$), such that by multiplying it by $\frac{1}{M}$ we obtain $\frac{m}{M} < \frac{1}{2} + \frac{1}{2M}$ (II).

Finally considering (I) and (II) together, it derives $\frac{1}{M}\sum_{h^j \in \mathcal{H}_k} h^j_k(\mathbf{x}) < 0.5 + \frac{1}{2M}$. For the ensemble with a large size, e.g. likewise our ensemble, the term $\epsilon' = \frac{1}{2M}$ is small. Therefore, it shows the class with the maximum probability (having the maximum votes) can be upper-bounded by $0.5 + \epsilon'$.

□

Using this corollary, we can fix a global threshold ($\tau = 0.5$) to identify the unusual samples that are causing entropy in the ensemble.

2.4 Analysis of Specialists Ensemble

Here, we first explain how adversarial examples give rise to entropy in our ensemble, leading to their low predictive confidence (with maximum confidence of $0.5 + \epsilon'$). As well, we examine the role of diversity in our ensemble, which harden the generation of white-box adversaries.

In a **black-box** attack, we assume that the attacker is not aware of our ensemble of specialists, thus generates some adversaries from a pre-trained vanilla CNN $g(\cdot)$ to mislead our underlying ensemble. Given a pair of an input sample with its true label, i.e. $(\mathbf{x}, k)$, an adversary $\mathbf{x}' = \mathbf{x} + \delta$ fools the model g such that $k = \text{argmax}\, g(\mathbf{x})$ while $k' = \text{argmax}\, g(\mathbf{x}')$ with $k' \neq k$, where k' is

one of those most-likely fooling classes for class k (i.e. $k' \in \mathbb{U}_k$). Among the specialists that are expert on k, at least one of them does not have k' in its expertise domain since we intentionally separated k-th class from its most-likely fooling classes when defining its expertise domains (Section 2.2). Formally speaking, denote those expertise domains comprising class k as follows $\mathcal{U}^k = \{\mathbb{U}_j \,|\, k \in \mathbb{U}_j\}$ where (I) $\mathbb{U}_j \neq \mathbb{U}_i \; \forall \mathbb{U}_i,\; \mathbb{U}_j \in \mathcal{U}^k$ and (II) $k' \notin \cap\, \mathcal{U}^k$. Therefore, regarding the fact that (I) the expertise domains comprising k are different and (II) their shared classes do not contain k', it is not possible that all of their corresponding specialist models are fooled simultaneously toward k'. In fact, these specialists may vote (classify) differently, leading to a disagreement on the fooling class k'. So, due to this disagreement in the ensemble with no winner class, all the ensemble's members are activated, resulting in prediction with high uncertainty (low confidence) according to corollary 1. Generally speaking, if $\{\cap\, \mathcal{U}^k\} \setminus k$ is a small or an empty set, harmoniously fooling the specialist models, which are expert on k, is harder.

In a **white-box attack**, an attacker attempts to generate adversaries to *confidently* fool the ensemble, meaning the adversaries should simultaneously activate *all* of the specialists that comprise the fooling class in their expertise domain. Otherwise, if at least one of these specialists is not fooled, then our voting mechanism results in adversaries with low confidence, which can then be automatically rejected using the threshold ($\tau = 0.5$). In the rest we bring some justifications on the hardness of generating high confidence gradient-based attacks from the specialists ensemble.

Instead of dealing with the gradient of one network, i.e. $\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}$, the attacker should deal with the gradient of the ensemble, i.e. $\frac{\partial \bar{h}(\mathbf{x})}{\partial \mathbf{x}}$, where $\bar{h}(\mathbf{x})$ computed by line 5 or line 7 of Algorithm. 1. Formally, to generate a gradient-based adversary from the ensemble for a given labeled clean input sample $(\mathbf{x}, \mathbf{y} = k)$, the derivative of the ensemble's loss, i.e. $\mathcal{L}(\bar{h}(\mathbf{x}), \mathbf{y}) = -\log \bar{h}_k(\mathbf{x})$, w.r.t. $\mathbf{x}$ is as follows:

$$\frac{\partial \mathcal{L}(\bar{h}(\mathbf{x}), \mathbf{y})}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \bar{h}_k(\mathbf{x})} \frac{\partial \bar{h}_k(\mathbf{x})}{\partial \mathbf{x}} = \underbrace{-\frac{1}{\bar{h}_k(\mathbf{x})}}_{\beta} \frac{\partial \bar{h}_k(\mathbf{x})}{\partial \mathbf{x}} = \beta \frac{1}{|\mathcal{H}_k|} \sum_{h^i \in \mathcal{H}_k} \frac{\partial h^i_k(\mathbf{x})}{\partial \mathbf{x}}. \tag{2.1}$$

Initially $\mathcal{H}_k$ indicates the set of activated specialists voting for class k (true label) plus the generalist for the given input $\mathbf{x}$. Since the expertise domains of the activated specialists are different ($\mathcal{U}^k = \{\mathbb{U}_j \,|\, k \in \mathbb{U}_j\}$), most likely their derivative are diverse, i.e. fooling toward different classes, which in turn creates perturbations in various fooling directions (Fig 2.1). Adding such diverse perturbation to a clean sample may promote disagreement in the ensemble, where no winner class can be agreed upon. In this situation, when all of the members are activated, the generated adversarial sample is predicted with a low confidence, thus can be identified. For the iterative attack algorithms, e.g. I-FGS, the process of generating adversaries may continue using the derivative of all of the members, adding even more diverse perturbations, which in turn makes reaching to an agreement in the ensemble on a winner fooling class even

more difficult.

2.5 Evaluation Criteria

Consider $h(\cdot) : \mathcal{X} \longrightarrow [0,1]^K$ a classifier model that maps an input sample to class conditional probabilities, i.e. $h(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_K(\mathbf{x})]$ for a given input sample $\mathbf{x}$. At inference time, the threshold-based approaches, like ours, define a threshold τ to reject the instances with lower predictive confidence than τ as an extra class $K+1$. Therefore, given a fixed threshold τ and the classifier model h, the decision function is defined:

$$d(\mathbf{x}|\tau, h) = \begin{cases} \text{argmax}_{\{1,\ldots,K\}}\, h(\mathbf{x}), & \text{if } \max_{\{1,\ldots,K\}} h(\mathbf{x}) > \tau \\ K+1, & \text{otherwise} \end{cases}. \tag{2.2}$$

To evaluate a predictor $h(\cdot)$ that includes a rejection option, we compute its risk rate $E_D|\tau$ on clean test set $\mathcal{D}_{test} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$ (containing M test samples with $\mathbf{x}_i$ as an input sample with its associated true label $y_i \in \{1, \ldots, K\}$) at a given threshold τ. This risk rate reflects the ratio of the (clean) samples that are *correctly classified but rejected* due to their confidence lower than the given confidence-threshold τ and those that are *misclassified but not rejected* due to their high confidence, i.e. larger than τ[1]:

$$\begin{aligned} E_D|\tau = \frac{1}{N}\sum_{i=1}^{N}\Bigg(& \underbrace{\left(\mathbb{I}[\underset{\{1,\ldots,K\}}{\text{argmax}}\, h(\mathbf{x}_i) \neq y_i] \;\times\; \mathbb{I}[d(\mathbf{x}_i|\tau) \neq K+1]\right)}_{\text{Non-rejected misclassified}} \\ & + \underbrace{\left(\mathbb{I}[\underset{\{1,\ldots,K\}}{\text{argmax}}\, h(\mathbf{x}_i) = y_i] \;\times\; \mathbb{I}[d(\mathbf{x}_i|\tau) = K+1]\right)}_{\text{Rejected correctly classified}}\Bigg). \end{aligned} \tag{2.3}$$

In addition, we report the risk rate $E_A|\tau$ on each adversaries set, i.e. $\mathcal{A} = \{(\mathbf{x}'_i, y_i)\}_{i=1}^{N'}$ that consists of the pairs of an adversarial example $\mathbf{x}'_i$ associated with its true label. This risk rate shows the percentage of misclassified adversaries that are not rejected due to their confidence larger than τ:

$$E_A|\tau = \frac{1}{N'}\sum_{i=1}^{N'}\left(\mathbb{I}[d(\mathbf{x}'_i|\tau) \neq K+1] \;\times\; \mathbb{I}[\underset{\{1,\ldots,K\}}{\text{argmax}}\, h(\mathbf{x}'_i) \neq y_i]\right). \tag{2.4}$$

2.6 Experimentation

2.6.1 Evaluation Setting

Using two standard benchmarks, namely MNIST [49] and CIFAR-10 [46], we investigate performance of our method for increasing detection rate of black-box attacks and reducing the

[1]Note that, for simplicity in notation, we drop the classifier h from $d(\mathbf{x}|\tau, h)$.

success rate of creating white-box adversaries. Two distinct CNN configurations are considered in these settings: a basic CNN (i.e. three convolution layers of respectively 32, 32, and 64 5×5 filters, each of these convolution layer being followed by a ReLU and 3×3 pooling with stride 2, followed by a final fully connected (FC) layer) for *MNIST* and a VGG-style CNN for *CIFAR-10* (details in [91]). For both CNNs, we use SGD with a nestrov momentum of 0.9, L2 regularization (with 10^{-4}), and dropout [93] ($p = 0.5$) in their FC layers. Experimental comparision is made regarding our ensemble of specialists approach with a vanilla (naive) CNN, and a pure ensemble approach, which involves five vanilla CNNs being different from the random initialization of their parameters.

MNIST. This dataset is comprised of gray scale images of size 28×28, where each image holds a hand-written digit. The training and test sets have 60K and 10K samples, respectively. All the images are scaled to $[0, 1]$. A CNN is trained on the training set (without data augmentation) for 150 epochs with batch size 128 and the initial learning rate 0.1 with momentum 0.9. The learning rate is decayed by a factor 10 at epoch 50, then epoch 100.

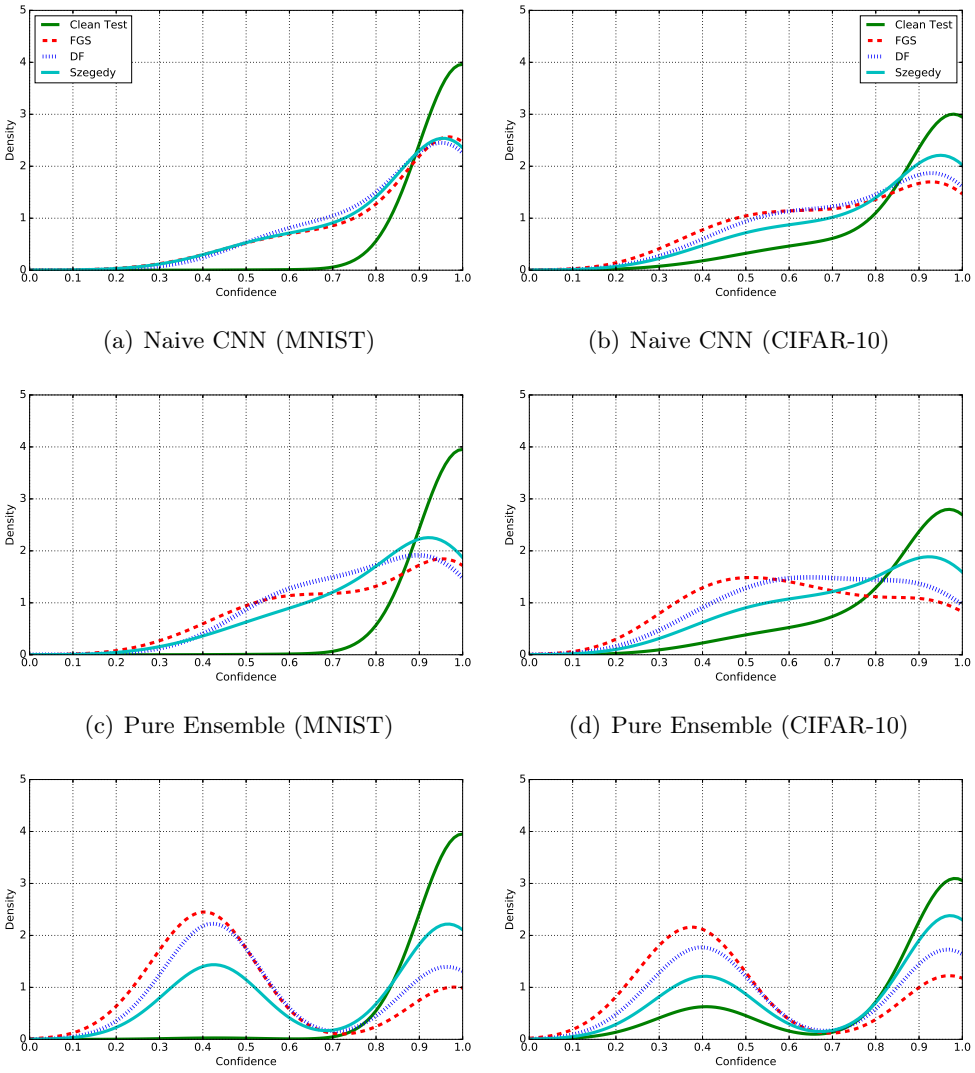
CIFAR-10. This dataset consists of 50K RGB images of size 32x32 as training set and 10K 32×32 RGB images as test set. Each image contains one object from one of 10 classes. All the images, either from train set or from test set, are scaled to $[0, 1]$, then normalized by mean subtraction, where the mean is computed over the training set. A CNN is trained on the training set (without data augmentation) for 150 epochs with batch size 128 and the initial learning rate is 0.01 with momentum of 0.9. The learning rate decays twice by a factor of 10 shortly before terminating training according to the training schedule $\{120, 130\}$.

We evaluate the detection rate of a vanilla CNN, a pure ensemble (consisting of 5 vanilla CNNs, i.e. generalist CNNs), and our ensemble on four well-known types of adversaries generated in the black-box model. Then, we investigate and compare the robustness of the mentioned methods to the white-box adversarial attacks including FGS, I-FGS, and T-FGS. Finally, we briefly investigate the capacity of our ensemble for detecting OOD samples.

2.6.2 Black-box Attacks

It is known that many types of adversaries generated from any models can be transferred to other victim models, where these adversaries still fool (misclassified) them confidently. Due to the transferability of the black-box attacks [25, 74], they are hard to be suppressed and detect. Moreover, the white-box attacks can occur less often than the black-box ones since the system designers can securely protect their model's information, such as its parameters and hyper-parameters, and their defending mechanism from outside accesses. Even in this case, such systems can still be threaten by the black-box adversaries. Therefore, as the black-box adversaries are more important for examination of robustness of CNNs, we first evaluate our deep specialists ensemble on them.

To do this, we generate FGS [25], TFGS [47], DeepFool (DF) [70], and CW attack [10] using a vanilla CNN, that has its random initialization of its weights different from that of all the naive CNNs used in our experiments. For two variants of gradient sign attack (namely FGS and T-FGS), we generate the adversaries with $\epsilon = 0.2$ and $\epsilon = 0.03$, for 2000 randomly selected correctly classified clean test samples from MNIST and CIFAR-10, respectively. For CW attack, due to its high computational cost, we generated 200 adversaries with $\kappa = 40$, for which larger κ ensures generation of high confidence and highly transferable CW adversaries.



(a) Naive CNN (MNIST)

(b) Naive CNN (CIFAR-10)

(c) Pure Ensemble (MNIST)

(d) Pure Ensemble (CIFAR-10)

(e) Ours (MNIST)

(f) Ours (CIFAR-10)

Figure 2.3: Density of the predictive confidence of clean test samples and their adversaries counterparts by a naive CNN, a pure ensemble and ours (ensemble of specialists) for MNIST (1st column) and CIFAR-10 (2nd column).

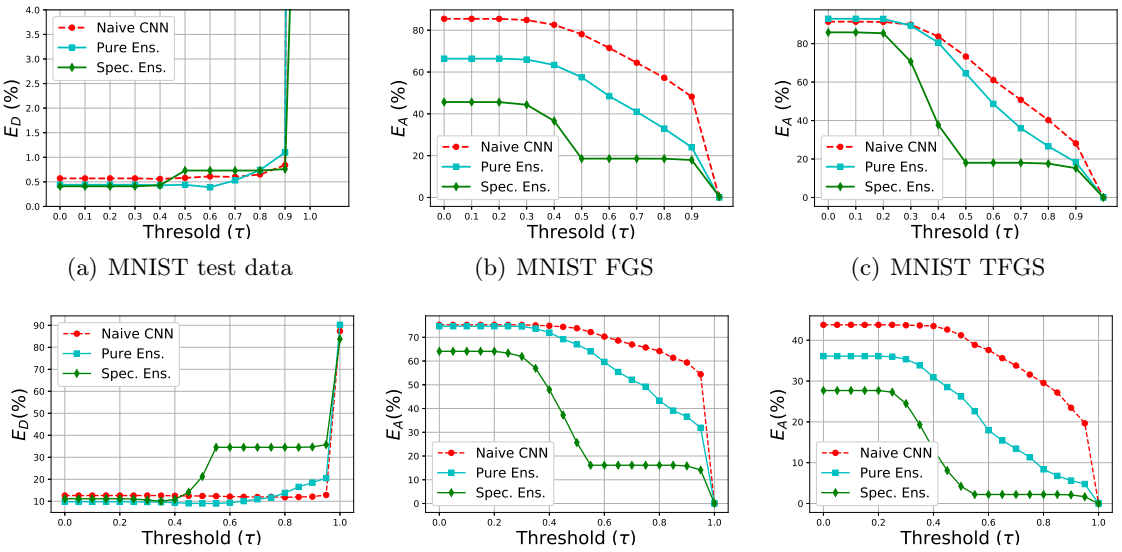
In Fig. 2.3, a naive CNN, the pure ensemble, and our ensemble are compared according to the distribution of their predictive confidences on the correctly classified clean test samples and the

black-box adversaries, including FGS, DeepFool (DF), and Szegedy [96] (Eq. 1.5) for MNIST (the first column) and CIFAR-10 (the second column). According to these observations for MNIST, our ensemble (2.3(c)) successfully provides significantly lower confidence for most of the adversaries of various types, in comparison to the naive CNN and the pure ensemble. Similarly, we observed the same behavior for CIFAR-10 test set and its adversaries. However, for the clean test samples of CIFAR-10, our ensemble shifts a small portion of these samples to lower confidence. This perhaps is rooted from our voting mechanism. More precisely, once at least one of the relevant experts misclassifies a clean sample, then the voting mechanism average the prediction of all the experts in the ensemble. One might reduce the number of the clean samples that are incorrectly predicted with low confidence through an adjustment in the voting mechanism such that it can tolerate having a few of the relevant experts (for example one or two of the relevant experts) misclassify the clean samples. In this case, we can still average the relevant experts even though a few of them can not provide correct prediction, resulting in a lower number of clean samples with low confidence. However, it should be noted that, by this adjustment, the performance of detection of adversaries might be reduced. Therefore, such adjustments in voting mechanism are a design choice and allow the user to adjust our ensemble to achieve his/her desire result (accuracy on clean sample v.s. robustness to adversaries).

As a result, *Although our ensemble is not trained on any types of adversaries and only by leveraging the diversity, it is able to automatically reduce the confidence of predictions for the adversaries, which results in a lower adversaries risk rate (in Fig 2.4), while preserving the confidence of the clean samples high.*

Fig. 2.4 presents risk rates ($E_D|\tau$) of different methods on clean test samples of MNIST (first row) and those of CIFAR-10 (second row), as well as their corresponding adversaries $E_A|\tau$, as functions of threshold (τ). As it can be seen from Fig. 2.4, by increasing threshold, more adversaries can be detected (decreasing E_A) at the cost of increasing E_D, meaning rejecting more the clean samples that are correctly classified.

To appropriately compare the methods, we find optimum threshold for the vanilla CNN and the pure ensemble that leads to small E_D and E_A collectively, i.e. $\text{argmin}_\tau\, E_D|\tau + E_A|\tau$. Recall that, as corollary 1 states, in our ensemble of specialists, we can fix the threshold of our ensemble to $\tau^* = 0.5$. In Table 2.1, we compare the risk rates of our ensemble with those of pure ensemble and vanilla CNN at their corresponding optimum thresholds. For MNIST, our ensemble outperforms naive CNN and pure ensemble as it detects a larger portion of MNIST adversaries while its risk rate on the clean samples is only marginally increased. Similarly, for CIFAR-10, our approach can detect a significant portion of adversaries at $\tau^* = 0.5$, reducing risk rates on adversaries. However, at this threshold, our approach has higher risk rates on clean samples than the two other methods.



(a) MNIST test data (b) MNIST FGS (c) MNIST TFGS

(d) CIFAR-10 test data (e) CIFAR-10 FGS (f) CIFAR-10 TFGS

Figure 2.4: Risk rates on clean test sets and black-box adversaries as functions of threshold on the predictive confidence.

Task	Adversaries / Methods	FGS E_A / E_D	TFGS E_A / E_D	CW E_A / E_D	DeepFool E_A / E_D
MNIST	Naive CNN	48.21 / 0.84	28.15 / 0.84	41.5 / 0.84	88.68 / 0.84
	Pure Ensemble	24.02 / 1.1	18.35 / 1.1	28.5 / 1.1	72.73 / 1.1
	Specialists Ensemble	**18.58** / **0.73**	**18.05** / **0.73**	**24** / **0.73**	**54.24** / **0.73**
CIFAR-10	Naive CNN	59.37 / **12.11**	23.47 / **12.11**	51.5 / **12.11**	28.81 / 12.11
	Pure Ensemble	36.59 / 18.5	8.37 / 13.79	4.0 / 13.79	7.7 / 18.5
	Specialists Ensemble	**25.66** / 21.25	**4.21** / 21.25	**3.5** / 21.25	6.02 / 21.25

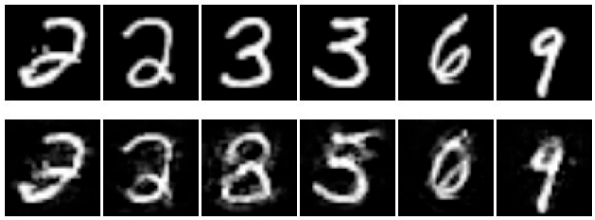
Table 2.1: Risk rate of clean test set ($E_D|\tau*$) along with risk rate of black-box adversarial examples sets ($E_A|\tau*$) are shown in percentage at the optimum threshold of each method. The methods with lowest collective risk rate (i.e. $E_A + E_D$) is underlined, while the best results for the two types of risk considered independently are in **boldface**.

Although many of FGS adversaries are rejected at threshold $\tau^* = 0.5$, some of them are hard to catch as adversaries for the specialists ensembles. Looking closely at some of non-rejected adversaries in Fig 2.5, it can be observed that recognizing their true labels are even difficult for the human observers. This is because some non-rejected adversaries are visually similar to a clean sample from the given fooling classes.

2.6.3 White-box Attacks

In the white-box model, we assume that the attacker has full access to a victim model. Using each method (i.e. naive CNN, pure ensemble, and specialists ensemble) as a victim model, we generate different sets of adversaries (i.e. FGS, Iterative FGS (I-FGS), and T-FGS). A

(a) FGS adversaries, clean samples (1st row) and their adversaries (2nd row)



(b) DeepFool adversaries, clean samples (1st row) and their adversaries (2nd row)

Figure 2.5: Illustration of some hard black-box adversaries that the specialists ensemble cannot identify them. (a) 1st and second rows are clean samples and FGS adversaries, respectively. Fooling labels of these FGS samples by our ensemble are (from left to right) 7, 8, 8, 5, 4, and 4. (b) 1st and 2nd rows are clean samples and DeepFool adversaries, respectively. Our ensemble predicts these DeepFool samples as 3, 3, 8, 5, 0, and 4, from left to right.

successful adversarial attack $\mathbf{x}'$ is achieved once the underlying model misclassifies it with a confidence higher than its optimum threshold τ^*. When the confidence for an adversarial example is lower than $\tau*$, it can be easily detected (rejected), thus it is not considered as a successful attack.

We evaluate the methods by their *white-box attacks success rates*, indicating the number of successful adversaries that satisfy the aforementioned conditions (i.e. a misclassification with a confidence higher than τ^*) during t iterations of the attack algorithm. Table 2.2 exhibits the success rates of white-box adversaries (along with their used hyper-parameters) generated by naive CNN ($\tau^* = 0.9$), pure ensemble ($\tau^* = 0.9$), and specialists ensemble ($\tau^* = 0.5$). For the benchmark datasets, the number of iterations of FGS and T-FGS is 2 while that of iterative FGS is 10. As it can be seen in Table 2.2, the success rates of adversarial attacks using ensemble-based methods are smaller than those of naive CNN since diversity in these ensembles hinders generation of adversaries with high confidence.

2.6.4 Gray-box CW Attack

In the gray-box model, it is often assumed that the attacker is aware of the underlying defense mechanism (e.g. specialists ensemble in our case) but has no access to its parameters and hyper-parameters. Following [33], we evaluate our ensemble on CW adversaries generated by another

Methods \ Adversaries	MNIST			CIFAR-10		
	FGS	T-FGS	I-FGS	FGS	T-FGS	I-FGS
	ϵ=0.2	ϵ=0.2	ϵ=2×10^{-2}	ϵ=3×10^{-2}	ϵ=3×10^{-2}	ϵ=3×10^{-3}
Naive CNN	89.94	66.16	66.84	86.16	81.38	93.93
Pure Ensemble	71.58	50.64	48.62	42.65	13.96	45.78
Specialists Ensemble	**45.15**	**27.43**	**13.63**	**34.1**	**7.43**	**34.20**

Table 2.2: The success rate of white-box adversarial examples (lower is better) generated by the naive CNN, the pure ensemble (5 generalists), and specialists ensemble at their corresponding optimum threshold. An successful white-box adversarial attack should fool the underlying model with a confidence higher than its optimum τ^*.

specialists ensemble, composed of 20 specialists and 1 generalist for 100 randomly selected MNIST samples. Evaluation of our specialists ensemble on these targeted gray-box adversaries (called "gray-box CW") reveals that our ensemble provides low confidence predictions (i.e. lower than 0.5) for 74% of them (thus able to reject them) while 26% have confidence more than 0.5 (i.e. non-rejected adversaries). Looking closely at those non-rejected adversaries in Fig. 2.6, it can be observed that some of them can even mislead a human observer due to adding very visible perturbation, where the appearance of digits are significantly distorted.

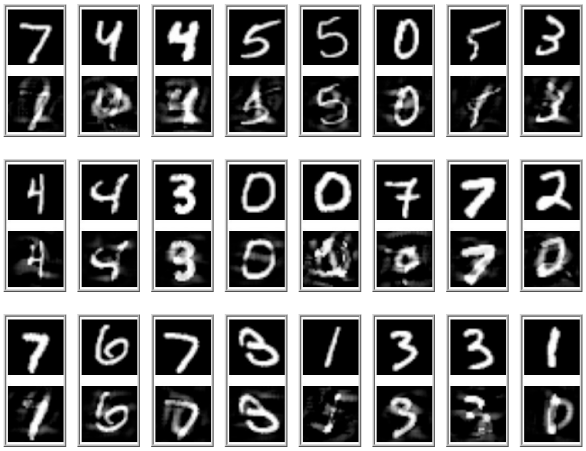
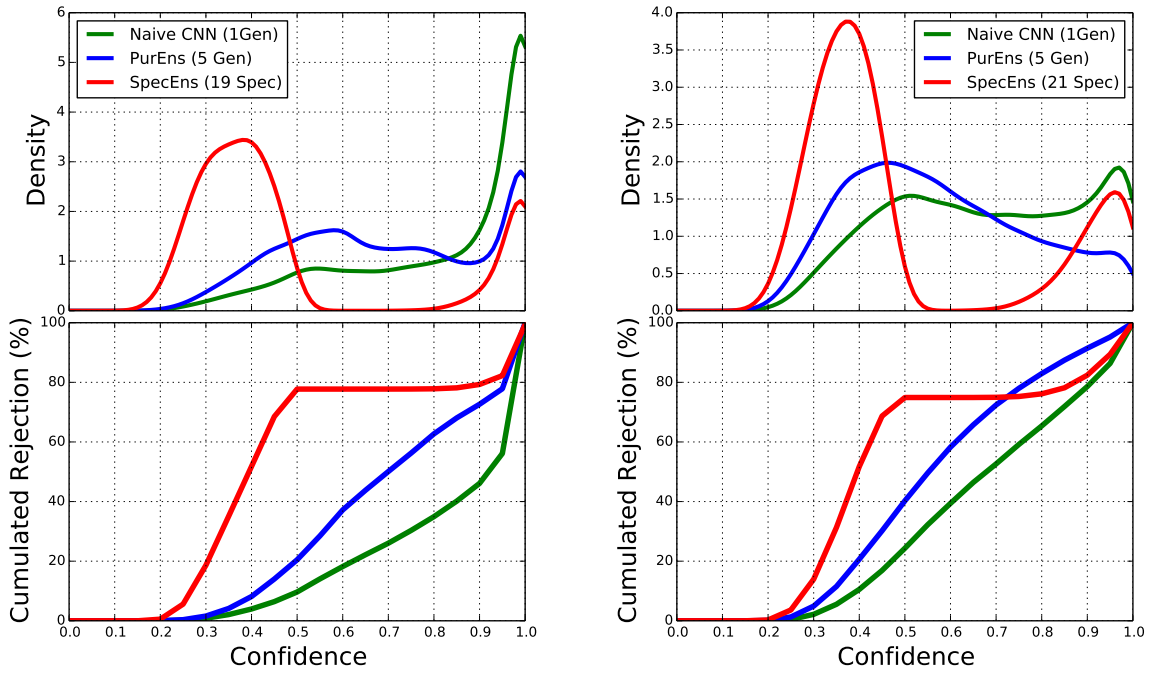


Figure 2.6: The gray-box CW adversaries that confidently fool our specialists ensemble. According to the definition of adversarial example, however, some of them are not actually adversaries due to their significant visual perturbations. Each pair represents a clean image (on top) and its CW adversaries (on bottom).

2.6.5 OOD Samples

In [48], the authors demonstrated the efficacy of generalists ensemble (an ensemble of vanilla CNNs) for detecting Out-of-Distribution (i.e. unknown) samples as it can potentially predict the OOD samples with higher uncertainty (i.e. low predictive confidence). Here, we also evaluate our specialist ensembles on OOD samples detection, where NotMNIST (gray-scale images



(a) NotMNIST as an OOD set for MNIST

(b) CIFAR-100 as an OOD set for CIFAR-10

Figure 2.7: The first row shows the distributions of predictive confidences for NotMNIST and CIFAR-100 samples, which are treated as Out-of-Distribution samples for MNIST and CIFAR-10, respectively. Their corresponding rejection rates as a function of confidence are exhibited at 2nd row.

of letters "A–J") and CIFAR-100 are regarded as the OOD sets for MNIST and CIFAR-10, respectively.

In Fig. 2.7, the ensemble methods (i.e. the specialists ensemble and the generalists one) are compared with a single naive CNN according to the distribution of their predictive confidences for Out-of-Distribution samples as well as their rejection rates as a function of threshold. As it can be seen in Fig. 2.7(a, top), while naive CNN confidently (confidence higher than 0.9) classifies a significant portion of NotMNIST samples, the ensembles predict many of them with lower confidence. Particularly, our ensemble shifts a major amount of the OOD samples to a lower confidence interval $[0, 0.5]$. Therefore, as shown in Fig. 2.7(a, bottom), our ensemble has a higher detection (rejection) rate of NotMNIST at $\tau^* = 0.5$ than that of the pure ensemble and the naive CNN at their optimal threshold ($\tau = 0.9$).

Similarly, for CIFAR-10, one can observe that the specialists ensemble provides low confidence predictions for a considerable amount of CIFAR-100 samples in Fig. 2.7(b, top) such that $\approx 78\%$ of them are rejected at threshold $\tau^* = 0.5$. However, the rejection rate of the pure

ensemble at $\tau^* = 0.9$ (i.e. $\approx 91\%$) is higher than ours (Fig. 2.7(b, bottom)).

Therefore, while a single CNN assigns high confidence prediction to the OOD samples, the ensemble approaches are more capable to predict them with considerably lower confidence (or higher uncertainty). This supports the crucial role of diversity in the ensemble for reducing certainty of the predictions, particularly for the unusual samples, such as the OOD samples and the adversaries ones.

2.7 Conclusion

In this chapter, we propose an ensemble of specialists, where each of the specialist classifiers is trained on a different subset of classes. We also devise a simple voting mechanism to efficiently merge the predictions of the ensemble's classifiers. Given the assumption that CNNs are strong classifiers and by leveraging diversity in this ensemble, a gap between predictive confidences of clean samples and those of black-box adversaries is created. Then, using a global fixed threshold, the adversaries predicted with low confidence are rejected (detected). We empirically demonstrate that our ensemble of specialists approach can detect a large portion of black-box adversaries as well as makes the generation of white-box attacks harder. This illustrates the beneficial role of diversity for the creation of ensembles in order to reduce the vulnerability to black-box and white-box adversarial examples.

Chapter 3

Metrics For Differentiating OOD sets

3.1 Introduction

In the presence of OOD samples, it is important for a model to distinguish them to make reliable decisions. However, due to the uncalibrated nature of (vanilla) deep neural networks (e.g. CNN), their predictions for the OOD samples are nearly as confident as that of in-distribution samples, making them indistinguishable from each other.

To tackle this challenge, apart from post-processing approaches,e.g. [55, 53], recently other researchers [5, 36, 52, 65, 66] have moved toward *end-to-end* calibrated CNNs, where they explicitly train the models to output uncertain and certain predictions for OOD samples and in-distribution samples, respectively. Then, by setting a threshold on these calibrated confidence predictions, the OOD samples with low predictive confidence are identified.

Moreover, the classical idea of adding an explicit rejection class [2, 15, 28] is another kind of end-to-end models for OOD detection task. Therefore, in addition to the calibrated vanilla CNN, we exploit A-CNN, i.e. a vanilla CNN with an extra class augmented to its pre-defined classes, as an end-to-end model for OOD detection task. Indeed, such augmented classifiers can directly reject the OOD samples by classifying them to the extra class, while correctly classifying the in-distribution samples to the rest of pre-defined classes. Interestingly, an A-CNN can perform the OOD sample detection without introducing any auxiliary hyper-parameters (e.g., threshold on the confidence score) as it explicitly rejects them by classifying them into the extra class.

However, considering the existence of an enormous number of OOD sets, the fundamental question here is: **which of the OOD sets should be selected and used for training well-generalized A-CNN and calibrated CNN?** Obviously, without having a principle for selecting an appropriate OOD set among those available, training *well-generalized A-CNNs and calibrated CNNs* can be a challenging task. Since using a randomly selected OOD set does not necessarily lead to a model with a high detection rate of unseen OOD sets (i.e., generalization

ability) as empirically shown in [36, 52] as well as we later show in our experiments. Particularly, Lee et al. [52] (in their appendix section) and Hendrycks et al. [36] argued that using SVHN as an OOD set for CIFAR-10 does not lead to an A-CNN and a well-calibrated CNN with high detection rates on unseen OOD sets, respectively. Moreover, simply using a union of an enormous number of OOD sets not only creates an extremely unbalanced dataset, but also makes training of these models computationally infeasible.

Without concretely answering the aforementioned question or providing a systematic basis for the selection of an OOD set for training purposes, these CNNs are trained by leveraging a naturalistic OOD set[1] that is selected *manually* from among many available ones [5, 36, 65, 66]. Although Hendrycks et al. [36] have conjectured a few clues (such as the role of diversity of OOD sets) to differentiate them and characterize a proper OOD set for training purposes, our main focus is to concretely answer this pivotal question: **how to differentiate among OOD sets to select a proper one for training a well-generalized calibrated model, which can induce high detection rate of *unseen* OOD sets?**

At first, we provide a formal criterion in the form of generalization errors of A-CNN for differentiating OOD sets and selecting the most effective one. *Using this, an OOD set is recognized as a proper (effective) if it leads to training of A-CNN with low generalization errors for both in-distribution and unseen OOD sets.* However, directly optimizing this selection criteria is computationally very expensive.

To overcome this computational burden, we, instead, propose that protection level of the in-distribution sub-manifolds by each OOD set (explained in section 3.3.1) as a factor to differentiate various OOD sets for a given in-distribution task. Indeed, under manifold hypothesis [3, 7][2], we assume that each class of the given in-distribution data has an intrinsic low-dimensional (sub)-manifold, which can be achieved by the last layer (before classifier layers) of a pre-trained deep neural network [7, 3]. Then, **we hypothesize that an appropriate (effective) OOD set is the one that covers (protects) the sub-manifolds of an in-distribution task**, such that training a CNN (e.g. A-CNN and calibrated CNN [12, 66]) on it allows to reject *automatically* the unseen OOD sets that are located relatively far away from the protected (in-distribution) sub-manifolds [3] (as shown in Figure 3.1).

With the aim of differentiating OOD sets, then selecting the most proper OOD set, we introduce three computationally inexpensive metrics (by the use of a vanilla pre-trained CNN) that are measuring the degree of protectiveness of the sub-manifolds by the OOD sets. These metrics are I) **Softmax-based Entropy** (SE) to measure how uniformly OOD samples are distributed

[1] We simply call OOD set instead of naturalistic OOD throughout the thesis.

[2] Bengio expresses in [4]: "manifold hypothesis states that natural classes present in the data (e.g., visual object categories) are associated with low-dimensional regions (i.e., manifolds) near which the distribution concentrates, and that different class manifolds are well-separated by regions of very low density"

[3] Throughout the text, by sub-manifold, we mean in-distribution sub-manifolds.

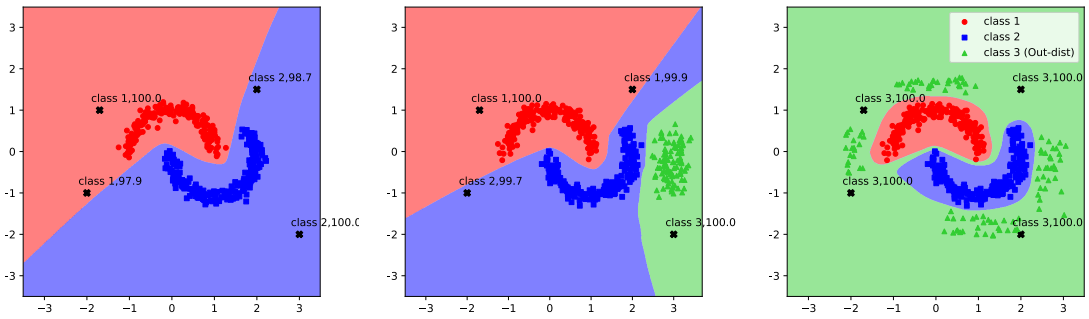


Figure 3.1: Illustration of properties of a partially-protective OOD set (middle) and a protective one (right) and their effect on training Augmented Multi Layer Perceptron (A-MLP) for two-moon classification dataset. (left) a vanilla MLP trained on two-moon dataset. The black-cross samples are some test OOD samples and their predicted class and confidence scores by each (A)-MLP are also indicated. All MLPs are made of three layers with ReLU activation function.

to the sub-manifolds; II) **Coverage Ratio** (CR), to measure how much of the sub-manifolds are covered by the OOD samples (of a given OOD set); III) **Coverage Distance** (CD), to quantify the distance between the OOD samples and the sub-manifolds.

Across various image and sound classification tasks, we assess our metrics for recognizing the most protective OOD set as an appropriate set. We then show that the most protective OOD set can lead to a calibrated CNN and an A-CNN (such A-CNN called as A-CNN*) with a drastically higher detection rate of unseen OOD sets than those trained on the least protective OOD set. Finally, we demonstrate that the A-CNN*, while never trained on any adversaries, can detect the FGS adversaries with a large amount of noise, as they are locating far away from the protected in-distribution sub-manifolds.

3.2 Characterizing a Proper OOD Set: A Formal Definition

Let us assume a hypothesis class $\mathcal{H}'$ for a $K+1$-class classification problem with K classes associated for a given in-distribution task and the extra class (i.e., the $(K+1)$-th class) reserved for OOD samples. Moreover, $\mathcal{S}_I = \{(\mathbf{x}_I^i, y_I^i)\}_{i=1}^N$ denotes an in-distribution training set consisting of N i.i.d. labeled samples drawn from data distribution $\mathcal{D}_I$, with true label $y_I^i \in \{1, \ldots, K\}$. As an OOD training set, take $\mathcal{S}_O = \{(\mathbf{x}_O^j)\}_{j=1}^M$ involving M i.i.d. unlabeled samples drawn from a data distribution $\mathcal{D}_O$, which we label as $(K+1)$-class.

The loss of a hypothesis $h' \in \mathcal{H}'$ for a given in-distribution sample can be defined as $\ell(h'(\mathbf{x}_I^i), y_I^i) = \mathbb{I}(h'(\mathbf{x}_I^i) \neq y_I^i)$ and its loss for an OOD sample is $\ell(h'(\mathbf{x}_O^j), K+1) = \mathbb{I}(h'(\mathbf{x}_O^j) \neq K+1)$ [4]. An augmented classifier $h' \in \mathcal{H}'$ is evaluated using the following losses:

[4]Indicator function $\mathbb{I}(p)$ returns 1 if condition p is true, and 0 otherwise.

The *empirical* loss computed on training set $\mathcal{S}_I$ and $\mathcal{S}_O$:

$$L_{\mathcal{S}_I}(h') = \frac{1}{N}\sum_{i=1}^{N} \ell(h'(\mathbf{x}_I^i), y_I^i);$$

$$L_{\mathcal{S}_O}(h') = \frac{1}{M}\sum_{j=1}^{M} \ell(h'(\mathbf{x}_O^j), K+1).$$

The *true* loss computed on the underlying data distributions $\mathcal{D}_I$ and $\mathcal{D}_O$:

$$L_{\mathcal{D}_I}(h') = \mathbb{E}_{(\mathbf{x}_I, y_I)\sim\mathcal{D}_I}\, \ell(h'(\mathbf{x}_I), y_I);$$

$$L_{\mathcal{D}_O}(h') = \mathbb{E}_{\mathbf{x}_O\sim\mathcal{D}_O}\, \ell(h'(\mathbf{x}_O), K+1).$$

Before presenting our definition, we remark that there is a set of B "out" data distributions $\mathcal{D}_O^b$ ($b = \{1, \ldots, B\}$) with their respective OOD training set $\mathcal{S}_O^b \sim \mathcal{D}_O^b$. Theoretically speaking, B can be infinitely large. Moreover, we assume that generalization error of a vanilla classifier (denoted by $h \in \mathcal{H}$), for the original K-class classification task, trained on $\mathcal{S}_I$ is small: $|L_{\mathcal{S}_I}(h) - L_{\mathcal{D}_I}(h)| \leq \epsilon$.

Definition 1. *: For a given OOD training set $\mathcal{S}_O^b \sim \mathcal{D}_O^b$ and in-distribution training set $\mathcal{S}_I$ w.r.t. hypothesis class $\mathcal{H}'$, $\mathcal{D}_I$ and B "out" data distributions, we define two kinds of gaps for the augmented classifier $h_b' \in \mathcal{H}'$ trained on $\mathcal{S}_I \cup \mathcal{S}_O^b$, i.e. $\min_{h_b'} L_{\mathcal{S}_I} + L_{\mathcal{S}_O^b}$:*

$$\boldsymbol{\mathcal{L}}_{\mathcal{S}_I} = |L_{\mathcal{S}_I}(h_b') - L_{\mathcal{D}_I}(h_b')|, \tag{3.1}$$

$$\boldsymbol{\mathcal{L}}_{\mathcal{S}_O^b} = \sup_{\mathcal{D}_O\in\{\mathcal{D}_O^1,\ldots\mathcal{D}_O^B\}} |L_{\mathcal{S}_O^b}(h_b') - L_{\mathcal{D}_O}(h_b')|. \tag{3.2}$$

The first term $\boldsymbol{\mathcal{L}}_{\mathcal{S}_I}$ represents the gap between the empirical loss of classifier $h_b' \in \mathcal{H}'$ on in-distribution training set $\mathcal{S}_I$ (i.e. $L_{\mathcal{S}_I}(h_b')$) and its true loss on $\mathcal{D}_I$ (i.e. $L_{\mathcal{D}_I}(h_b')$). The second term $\boldsymbol{\mathcal{L}}_{\mathcal{S}_O^b}$ concerns the largest (worst) gap between empirical loss of h_b' on the given OOD training set $\mathcal{S}_O^b$ (i.e. $L_{\mathcal{S}_O^b}(h_b')$) and its true loss on an "out" data distribution (i.e. $L_{\mathcal{D}_O}(h_b')$). By restricting B to a manageable (finite) large number, we re-define $\boldsymbol{\mathcal{L}}_{\mathcal{S}_O^b}$ by upper-bounding Eq. 3.2, i.e. sum of gaps on B finite "out" data distributions:

$$\boldsymbol{\mathcal{L}}_{\mathcal{S}_O^b} = \sum_{\mathcal{D}_O\in\{\mathcal{D}_O^1,\ldots\mathcal{D}_O^B\}} |L_{\mathcal{S}_O^b}(h_b') - L_{\mathcal{D}_O}(h_b')|. \tag{3.3}$$

As true data distributions are unknown, the aforementioned equations can be empirically computed using validation sets. Then, a proper OOD set is the OOD set such that training

a A-CNN on it should produce the lowest accumulation of generalization errors of both in-distribution task and (un)seen OOD sets:

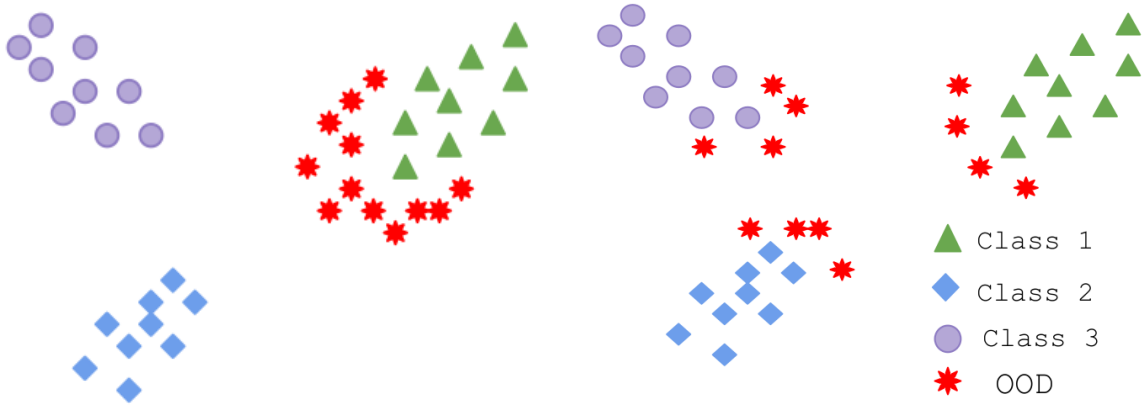
$$\mathcal{S}_O^{b^*} = \underset{\mathcal{S}_O^b \in \{\mathcal{S}_O^1, \dots \mathcal{S}_O^B\}}{\operatorname{argmin}} \boldsymbol{\mathcal{L}}_{\mathcal{S}_I} + \lambda \boldsymbol{\mathcal{L}}_{\mathcal{S}_O^b}, \tag{3.4}$$

where $\lambda > 0$ is a balancing hyper-parameter. Directly using Eq. 3.4 to find a proper OOD set is computationally inefficient as it involves training B individual augmented classifiers, i.e. train each h'_b on a pair of $\mathcal{S}_I \cup \mathcal{S}_O^b, b \in \{1, \dots, B\}$. Particularly for the case of CNNs, this incurs a huge computational overhead. To overcome this computational burden, we conjecture that a protective OOD set can also provide a well-generalized A-CNN on both in- and unseen OOD sets (an intuitive illustration is given in Sec. 3.1). Thus, instead of directly optimizing Eq 3.4, we develop some cost-effective metrics to assess the protectiveness level of OOD sets for identifying the most protective one.

3.3 Metrics for Characterizing a Protective OOD Set

3.3.1 Illustrative Example

To give a high-level intuitive explanation of our proposed metrics for recognizing a proper OOD set, we use an example based on the two-moon dataset (as in-distribution task), where each moon is considered as a sub-manifold. Fig. 3.1(a) exhibits the challenge of OOD samples for a vanilla MLP, which is trained on only in-distribution samples. As can be seen, this vanilla MLP *confidently* classifies OOD samples (indicated with black-crosses) as either "class 1" or "class 2" albeit they clearly belong to *none* of the in-distribution manifolds. In Fig. 3.1(b) we demonstrate a *partially-protective* OOD set whose samples are almost collapsed and only partially cover one of the sub-manifolds (i.e., the manifold with blue squares). An Augmented Multi Layer Perceptron (A-MLP) trained on two-moon dataset along with this OOD set leads to a classifier with a limited OOD detection performance. More precisely, OOD samples, e.g., the *unseen* black-cross samples, which are laying around *uncovered* parts of the manifolds, are still confidently misclassified by the underlying A-MLP. Whereas, in Fig 3.1 (c) a proper *protective* OOD set whose samples better cover the in-distribution's sub-manifolds (two-moon) is shown. As can be seen, training an A-MLP on such protective OOD set (along with in-distribution samples) leads to classifying *unseen* black cross OOD samples as class 3 (i.e., the extra class) as well as classifying automatically the regions out of the manifolds as class 3. This results in an A-MLP with high detection performance on unseen OOD sets (i.e., making the gap in Eq. 3.2 small). Therefore, the design of our metrics is driven according to this intuition that a proper OOD set should be more protective of (i.e., closely covers) all in-distribution sub-manifolds in the feature space. A similar intuition has been previously exploited by some researchers, e.g., [52, 117], with the aim of only generating synthetic OOD samples.



(a) An OOD set with the lowest SE

(b) An OOD set with the highest SE

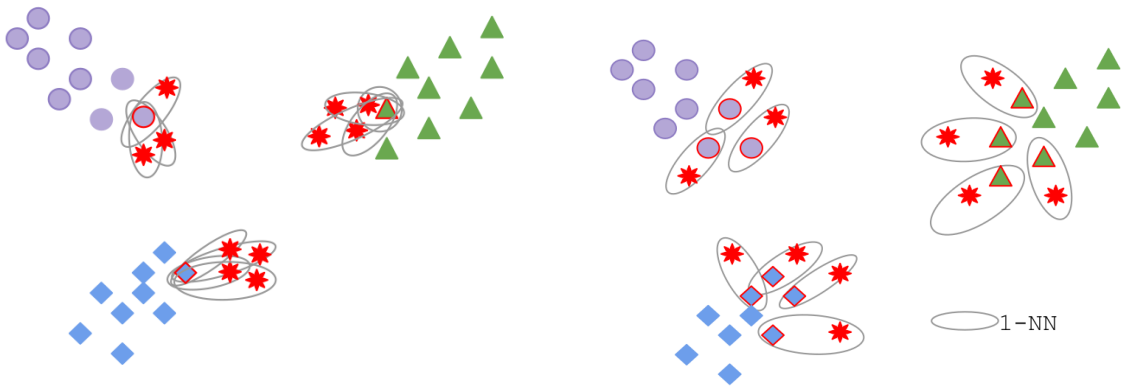
Figure 3.2: An illustrative example of two different OOD sets for a given in-distribution set with two different SE values; (left) an OOD set with the smallest SE (i.e. $\mathbf{H} = 0$) since the OOD samples are all collapsed to one of the sub-manifolds, the sub-manifold belong to class 1 represented by green triangles. Indeed the vanilla classifier $h(\cdot)$ classifies all of these samples to class 1. (right) an OOD set with the highest SE (i.e. $\mathbf{H} \sim \log 3$) as its samples are distributed evenly to all the sub-manifolds as the classifier classify an equal number of them for each of the classes.

Note that like other researchers [19, 40], we consider the penultimate layer of a vanilla CNN as a function that transfers samples from high-dimensional input space into a low-dimensional feature space, placing them on data (sub-)manifold(s) [4, 7]. Furthermore, using the manifold hypothesis, we assume that for a standard multi-classification problem (with K classes), each class has its own sub-manifold in the feature space where its associated in-distribution samples lie. In the following, we propose our metrics to assess which of the available OOD sets has a better and closer coverage of the sub-manifolds.

3.3.2 Softmax-based Entropy

We propose our first metric, namely Softmax-based Entropy (SE), to ensure the samples of a given OOD set are distributed evenly to all sub-manifolds (of a given in-distribution task) such that they have the chance of being covered by these OOD samples. For example, an OOD set, whose samples are misclassified by a given vanilla CNN into *only* a few of in-distribution classes (manifolds) instead of all of them, is deemed as non-protective OOD set. This is because the sub-manifolds, which have no OOD samples being misclassified to them, are still uncovered (e.g., Fig. 3.2 (a)), thus training A-CNN on such non-protective (or partially-protective) OOD set may lead to limited detection performance of unseen OOD sets. In contrast, the samples of a protective set are expected to be misclassified evenly to all the sub-manifolds, giving them a better chance of being covered (Fig. 3.2(b)).

To quantitatively measure this incidence for a given OOD set w.r.t. an in-distribution set



(a) An OOD set with low CR

(b) An OOD set with relatively higher CR than (a)

Figure 3.3: An illustrative example for two OOD set with the same high level of SE, but different CR values. The OOD set in (a) has a lower Coverage Ratio with k = 1-NN (i.e. $\mathbf{R} = \frac{3}{29}$) as it covers a lower number of in-distribution of samples, while the OOD set in (b) has a higher CR with k = 1-NN, $\mathbf{R} = \frac{9}{27}$.

and a vanilla CNN trained on it, we introduce Softmax-based Entropy (SE). First we define $p(c = k|\mathcal{S}_O)$ as the conditional probability of k-th class given $\mathcal{S}_O$ as follows:

$$p(c = k|\mathcal{S}_O) = \frac{1}{M} \sum_{j=1}^{M} \mathbb{I}_k \left(\underset{\{1,\dots,K\}}{\text{argmax}}(h(\mathbf{x}_O^j)) = k \right), \tag{3.5}$$

where h is softmax output of the vanilla CNN trained on $\mathcal{S}_I$ and $\mathbb{I}_k(.)$ is an indicator function for k-th class. It returns 1 if a given OOD sample $\mathbf{x}_O^j \in \mathcal{S}_O$ is (mis)classified as class k by h, otherwise it returns 0. Finally, SE is defined for an OOD set $\mathcal{S}_O$ as follows:

$$\mathbf{H}(\mathcal{S}_O) = -\sum_{k=1}^{K} p(c = k|\mathcal{S}_O) \log p(c = k|\mathcal{S}_O). \tag{3.6}$$

$\mathbf{H}(S_O)$ shall reflect how uniformly the samples of $\mathcal{S}_O$ are distributed to in-distribution sub-manifolds (i.e., corresponding to each in-distribution class). Thus, the highest $\mathbf{H}(\mathcal{S}_O)$ indicates that all the sub-manifolds have (nearly) equal number of OOD samples, whereas the smallest value of $\mathbf{H}(\mathcal{S}_O)$ indicates some sub-manifolds have a few (or no) OOD samples i.e., $x_O^j \in \mathcal{S}_O$ covering them. That is, a protective OOD set should have higher SE than non-protective ones.

3.3.3 Coverage Ratio

Although an OOD set with the high(est) SE confirms OOD samples are evenly distributed to all the sub-manifolds, using solely SE is not sufficient to assure the coverage of these sub-manifolds. Put differently, an OOD set with the highest SE might still be collapsed and only partially cover some parts of the sub-manifolds (Fig. 3.3).

Inspired by covering number notion [89], we introduce our second metric, named coverage ratio (CR), in order to measure coverage of the sub-manifolds. Recall the sub-manifolds are approximated using training in-distribution set in the feature space that is achieved by the penultimate layer of h. We denote $\mathbf{z}_I^i$ and $\mathbf{z}_O^j$ as the representations of $\mathbf{x}_I^i \in \mathcal{S}_I$ and $\mathbf{x}_O^j \in \mathcal{S}_O$ in the feature space, respectively.

To formally describe Coverage Ratio (CR), we form a rectangular weighted adjacency matrix $W \in \mathbb{R}^{N \times M}$ for a given pair $(\mathcal{S}_I, \mathcal{S}_O)$ with N in-distribution and M OOD samples, respectively. $W_{i,j} = \|\mathbf{z}_I^i - \mathbf{z}_O^j\|_2$ is the distance (l_2-norm) between in-distribution sample $\mathbf{z}_I^i$ and OOD sample $\mathbf{z}_O^j$ in the feature space. The distance between a pair of $(\mathbf{z}_I^i, \mathbf{z}_O^j)$ is computed only if $\mathbf{z}_I^i$ is among k-nearest in-distribution neighbors of $\mathbf{z}_O^j$, otherwise $W_{i,j} = 0$:

$$W_{i,j} = \begin{cases} \|\mathbf{z}_I^i - \mathbf{z}_O^j\|_2 & \text{if } \mathbf{z}_I^i \in \text{k-NN}(\mathbf{z}_O^j, S_I) \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

In other words, for each sample $\mathbf{z}_O^j$, we find its k nearest neighbors from the in-distribution set $\mathcal{S}_I$ in *the feature space*. Then if the given $\mathbf{z}_I^i$ belongs to k-nearest in-distribution neighbors of $\mathbf{z}_O^j$, we set $W_{i,j}$ to their distance. From matrix W, we derive a binary adjacency matrix A as $A_{ij} = \mathbb{I}(W_{ij} > 0)$. Now, using the matrix A, we define CR metric as follows:

$$\mathbf{R}(\mathcal{S}_I, \mathcal{S}_O) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\sum_{j=1}^{M} (A_{i,j}) > 0), \tag{3.8}$$

where $\mathbb{I}(\sum_{j=1}^{M}(A_{i,j}) > 0)$ assess whether i-th in-distribution sample $\mathbf{x}_I^i$ appears at least one time among the k-nearest neighbors of j-th OOD sample $\mathbf{x}_O^j$ in the feature space. Basically, this metric measures how many in-distribution samples (percentage) are covered by at least one OOD samples from $\mathcal{S}_O$ in the feature space. Finally, **we estimate an OOD set w.r.t. a given in-distribution set as protective if it has both high SE and high CR, compared those of the other OOD sets.**

It is important to note that SE and CR are complementary. As mentioned earlier, high SE of an OOD set without considering its CR is not sufficient for estimating the protective level of an OOD set. Similarly, from high CR alone without having high SE, an OOD set cannot be considered as protective. This is because, an OOD set with high CR but low SE is not distributed enough among all sub-manifolds and might cover a large portion of only a few sub-manifolds.

3.3.4 Coverage Distance

Furthermore, to measure the distance between OOD set $\mathcal{S}_O$ and the in-distribution data sub-manifolds, the following distance metric, named Coverage Distance (CD), can be driven:

$$\mathbf{D}(S_I, S_O) = \frac{\sum_{i,j} W_{ij}}{\sum_{i,j} A_{ij}} = \frac{1}{kM} \sum_{i,j} W_{ij}. \tag{3.9}$$

$\mathbf{D}(\mathcal{S}_I, \mathcal{S}_O)$ shows average distance between OOD samples of $\mathcal{S}_O$ and their k nearest neighbors from in-distribution set. Note between two OOD sets with the same level of high SE and high CR, the one with smaller CD, which indicates its samples are located nearer to the sub-manifolds, is preferable.

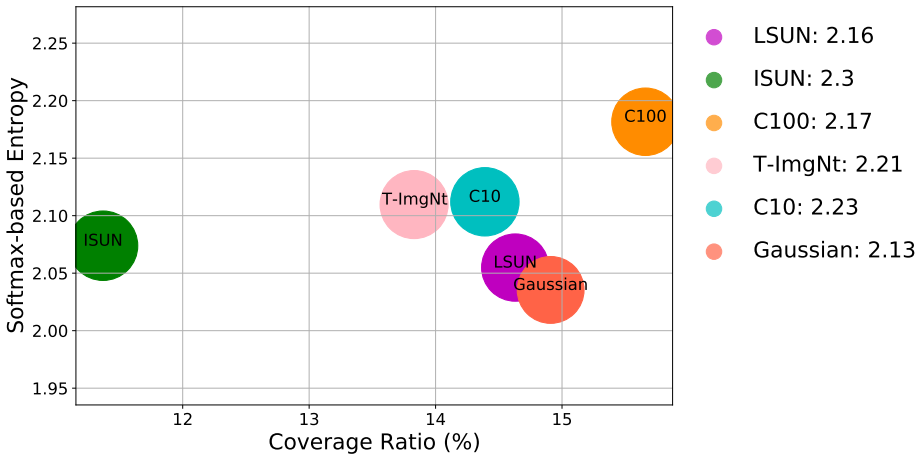
3.4 Experiments

We conduct a series of experiments on several classification tasks including two image benchmarks namely CIFAR-10 and SVHN and one audio benchmark, namely Urban-Sound [84]. In our experiments, we utilize VGG-16 and a CNN described in [83] (also precisely described in Appendix B) for image and audio benchmarks, respectively.

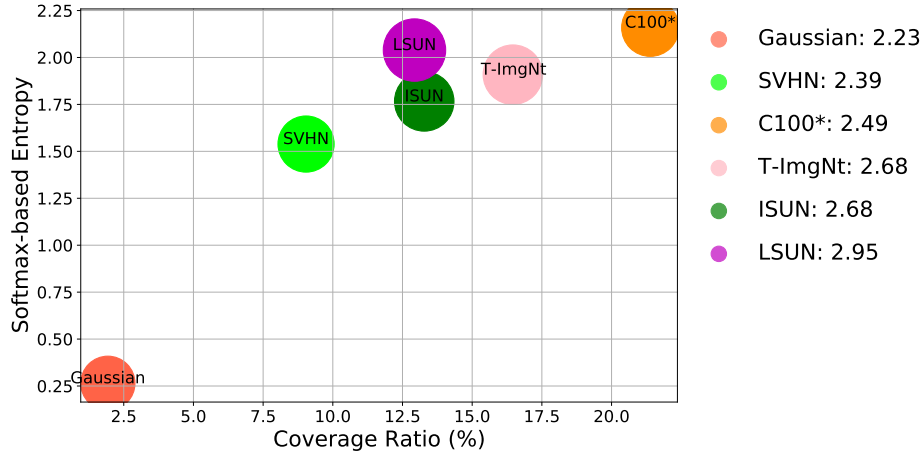
Like in [55], for each of these in-distribution task, various naturalistic OOD sets are considered; for image classification tasks, i.e., CIFAR-10 and SVHN, we regard LSUN, ISUN, CIFAR-100 and TinyImageNet as OOD sets and Gaussian noise as a synthetic OOD set. Note that we discard the classes from a given OOD set that have a semantic overlap with those from the in-distribution set, such as the overlap between the classes of CIFAR-100 and CIFAR-10 (as an in-dist. task). For audio classification task with 10 classes, i.e., Urban-Sound, OOD sets considered are TuT [68], Google Command [112] and ECS (Environmental Sound Classification) [75], as well as white-noise sound as a synthetic OOD set.

3.4.1 Empirical Assessment of Metrics

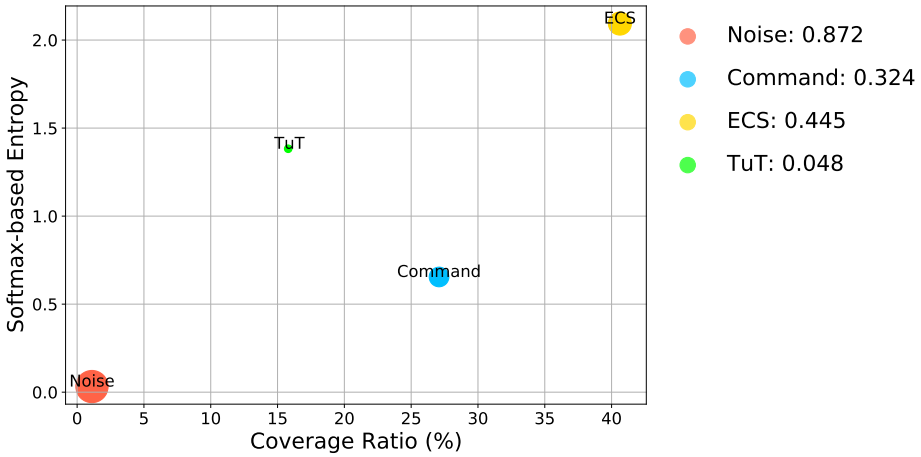
First, to obtain in-distribution sub-manifolds, if in-distribution training set has more than $10,000$ samples, we randomly select $10,000$ samples from it, otherwise, we use the whole in-distribution training set. Secondly, we pass the samples through the penultimate layer of a pre-trained vanilla CNN to map them into the feature space. The same procedure is done for the samples of an OOD set to transfer them to the feature space. To fairly compare OOD sets according to the metrics, we also randomly select equal number of OOD samples ($10,000$ samples) from each OOD set. For OOD sets with various sizes, we take the minimum size for making equal size OOD sets by randomly selecting from them. To compute CR and CD metrics, we set our only hyper-parameter, which is the number of nearest neighbors, to $k = 4$ for all our experiments. Note that among our metrics, CR and CD are dependent on k (the impact of k on our metrics is presented later).



(a) SVHN: ISUN/C100



(b) CIFAR-10: SVHN/C100*



(c) Urban-Sound: (Command, TuT)/ ECS

Figure 3.4: Differentiating OOD sets for SVHN, CIFAR-10, and Urban-Sound for the purpose of selecting the most protective one using our proposed metrics. Each sub-figure shows a bubble chart with SE and CR as y-axis and x-axis, respectively. The size of bubbles is determined by CD, also shown in the legend of the sub-figures. The least/most protective OOD sets are indicated in caption of the sub-figures.

Using the proposed metrics, we differentiate OOD sets for each in-distribution set to select the most protective OOD sets w.r.t. the given in-distribution. In Fig. 3.4, we demonstrate the difference between OOD sets according to their SE, CR, and CD, in order to identify the most and least protective OOD sets. The most and least protective naturalistic OOD sets identified by our metrics (particularly by SE and CR) are indicated in caption of sub-figures in Fig. 3.4. For *SVHN* task, for example, *ISUN*, among naturalist OOD sets, and *Gaussian noise*, as synthetic OOD set, are identified as the least protective sets. Note that despite the high CR of Gaussian noise, its SE is very small, indicating it as a collapsed OOD set, which thus causes it to be identified as the least protective. The most protective OOD set for SVHN is CIFAR-100 (i.e., C100) with the highest SE and CR. For Urban-sound, ECS is the most protective, while Command and TuT datasets can be regarded as non-protective OOD sets due to their significantly low SE with respect to the upper bound of SE ($\log 10$).

To assess the sensitivity of our metrics to the choice of k, we show the CR of OOD sets for varying k's values in Fig 3.5. In our experiments, we observe that the relative ranking of OOD sets according to their CR and CD is consistent with various values of k. Thus, CR and CD are not sensitive to the choice of k.

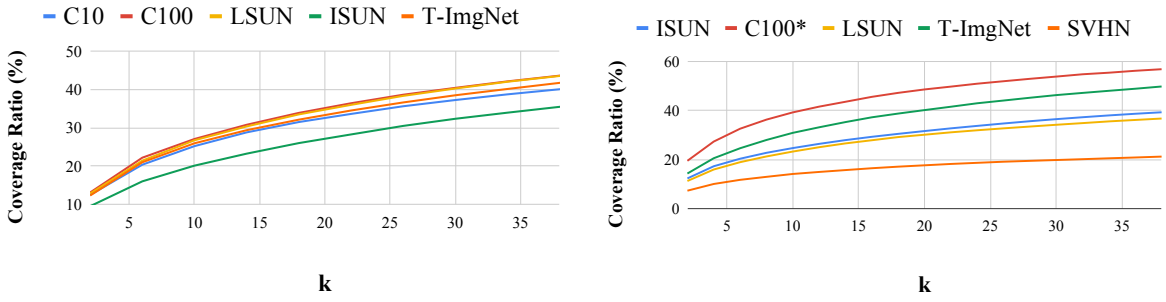
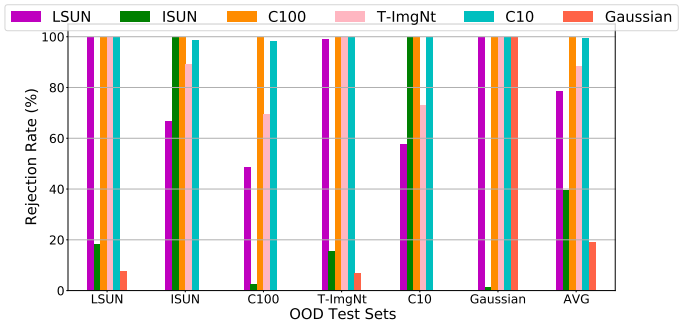


Figure 3.5: The effect of k (number of nearest neighbors) on CR of OOD sets for SVHN (left) and CIFAR-10 (right).

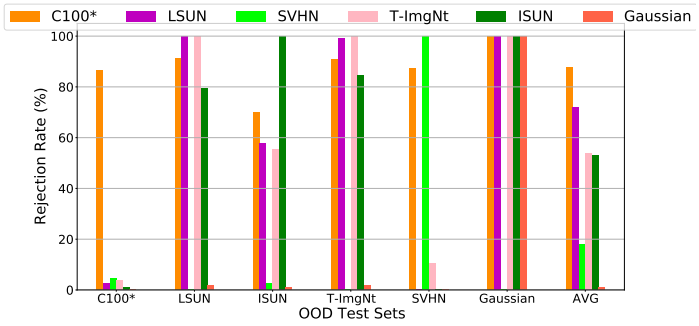
3.4.2 Impact of OOD Training Sets on Performance of A-CNNs

Performance criteria As A-CNNs have an explicit rejection option (used to explicitly classify OOD samples to the extra class), we consider three criteria to assess A-CNN performance on in-distribution test set; (I) **Accuracy rate (Acc.)** ↑: rate of samples classified correctly as their true associated label (↑ indicates higher is better), (II) **Rejection rate (Rej.)** ↓: rate of samples misclassified as dustbin (↓ indicates lower is better), (III) **Error rate (Err.)** ↓: rate of samples that are neither correctly classified nor rejected (Err. rate = 1- (Acc. rate + Rej. rate)). A-CNN **performance on OOD sets** is evaluated by (I) Rejection rate (Rej.) ↑: percentage (rate) of OOD samples classified as dustbin, and (II) Error rate (Err.) ↓: rate of OOD samples not classified as dustbin (Err. = 1 - Rej.)

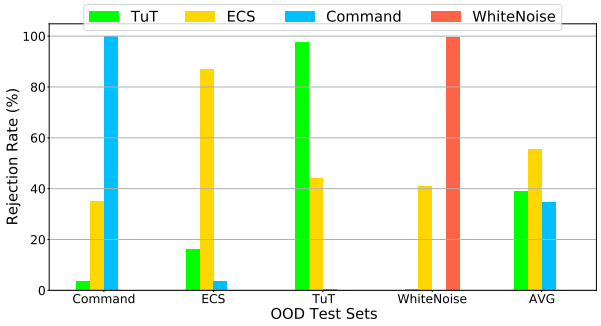
Recall that for OOD rejection (detection), we *are not using any threshold on the predictive confidence of A-CNN since it is trained to explicitly classifying OOD samples as the extra class.* Therefore, there are no AUROC and AUPR values as these are computed by varying the threshold. Note, A-CNN's OOD rejection rate and its in-distribution rejection rate are the same concepts as TNR (True Negative Rate) and FNR (False Negative Rate), respectively.



(a) SVHN



(b) CIFAR-10



(c) Urban-Sound

Figure 3.6: Rejection rates of different OOD sets (x-axis) by A-CNNs, where each A-CNN is trained on a single OOD set, e.g. for SVHN in figure (a) the violet color shown an A-VGG trained on SVHN as in-distribution and LSUN as the OOD set, then it is evaluated by different OOD test sets on the x-axis.

As Eq. 3.4 states, training an A-CNN on a proper OOD set should lead to a low average of [5] error rates on (un)seen OOD sets (or equivalently high average of OOD sample rejection rates). Therefore, for a given in-distribution task (e.g. SVHN), we train a separate A-CNN on each OOD set. Then, the error rates of these A-CNNs on all of the OOD sets are evaluated. Note a small error rate on a OOD set is equivalent to high detection rate.

In Fig 3.6, A-CNNs trained on the most protective set (identified by our metrics) across various in-distribution tasks consistently outperform the A-CNNs trained on other OOD sets, particularly the A-CNN trained on the least protective one. For instance, A-CNN trained for CIFAR-10 with CIFAR-100* (the non-overlapped version of CIFAR-100) as the most protective OOD set has 85% rejection rate on average while the least protective one, i.e. SVHN, delivers A-CNN with the lowest average of rejection rates (21%) of OOD sets.

It is also interesting to note that even though one may expect Gaussian noise (i.e. white noise) to have well distributed samples, SE metric shows that its samples are actually not evenly distributed over all of the in-distributed sub-manifolds (having lowest SE) and sometimes (for CIFAR-10 and Urban-Sound in-distribution sets) they even have a small coverage rate. As a result, an A-CNN trained on Gaussian noise as an OOD set has the lowest average OOD rejection rate.

Consequently, the results show that all of the OOD sets are not equal for training well-generalized A-CNNs as they do not equally protect in-distribution sub-manifolds. *Thus, we highlight that protectiveness can be an important factor for differentiating OOD sets to ultimately select the most proper one.* Moreover, to select an such OOD set, we remark that our metrics are computationally inexpensive than explicitly optimizing Eq. 3.4, which is equivalent to searching exhaustively all A-CNNs, where each is trained on an OOD set.

In Table 3.1, *in-distribution* generalization performance of two A-CNNs, one trained on the most protective OOD set (named A-CNN*) and another trained on the least protective one (named A-CNN‡), are compared with their standard (Vanilla) CNN. Although the accuracy rates of the A-CNNs* drop slightly, their error rates (i.e., risks) are considerably smaller than their counterparts, i.e., vanilla CNNs. This is because the A-CNNs are able to reject some “hard" in-distribution samples, instead of incorrectly classifying them (similar to [20]). Rejecting a sample rather than incorrectly classifying it is an important aspect, particularly for security and safety concerns.

Impact of OOD Training Sets on Performance of Calibrated Vanilla CNNs

In order to verify the importance of accessing to a proper OOD set for training a well-performed end-to-end CNN-based model on both in- and out-of-distribution samples, we perform another set of experiments by replacing A-CNNs with calibrated vanilla CNNs. A calibrated vanilla

[5] Instead of summation in Eq. 3.4, we take the average.

In-dist. task	Network	In-distribution Acc (↑) / Rej (↓) / Err (↓)	OOD sets Avg OOD Rej. (↑)
SVHN	Vanilla VGG	**95.53** / – / 4.47	–
	A-VGG‡ (ISUN)	95.11 / 0 / 4.89	47.23
	A-VGG* (C100)	95.38 / 0.34 / **4.28**	**99.88**
CIFAR-10	Vanilla VGG	**88.04** / – / 11.95	–
	A-VGG‡ (SVHN)	87.75 / 0.03 / 12.22	21.41
	A-VGG* (C100*)	85.37 / 5.65 / **8.97**	**85.10**
Urban-Sound	Vanilla CNN	**67.27** / – / 32.73	–
	A-CNN‡ (Command)	65.05 / 2.02 / 32.93	26.07
	A-CNN* (ECS)	63.13 / 12.02 / **24.85**	**55.40**

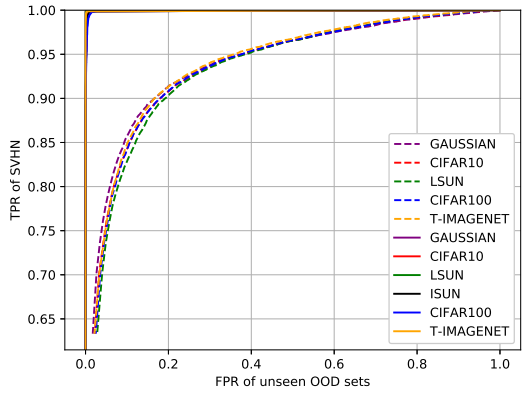
Table 3.1: The influence of selected most and least protective OOD sets on inducing well-generalized A-CNNs with high OOD detection rates.

CNN is trained to predict OOD training samples with uncertainty (i.e. uniform prediction) while confidently classifying correctly in-distribution training samples (we call this vanilla CNN as calibrated CNN). To achieve this, instead of cross entropy loss, we apply the modified loss function used in [36, 52]– defined in Eq. 1.14.

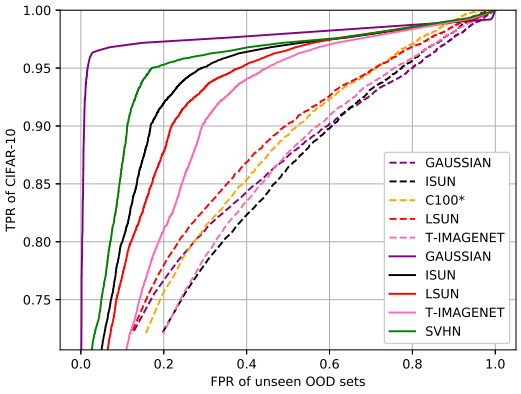
Evaluation criteria At inference time, by setting a threshold on the output predictions of the calibrated CNN, one can distinguish OOD sets from in-distribution samples since the predictive confidence of the calibrated CNN on OOD samples should be drastically lower than that of in-distribution samples. Likewise [55, 53, 36], for assessing performance of calibrated CNNs trained on various OOD sets, we report AUROC (Area Under Receiver Operation Characteristic), which obtains from the plot of FPR v.s. TPR at different threshold values, and FPR at 95% TPR.

As it can be seen (table 3.2) that the most protective OOD set recognized by our metrics is leading to a calibrated CNN with a considerable lower average of FPR at 95% TPR and highest AUROC. While the calibrated CNN training on the least protective one has the higher FPR and the lower AUROC[6]. The ROC curves of two calibrated vanilla CNNs (trained on the most and the least protective OOD sets) on unseen OOD sets are demonstrated in Fig 3.7.

Consequently, we highlight that efficiently recognizing proper (i.e. protective) OOD sets among the enormous available ones is a key for training a well-performed *end-to-end model* (either the underlying model is A-CNN or calibrated vanilla CNN) for addressing OOD detection challenge.



(a) Calibrated-vanilla CNNs on SVHN as in-dis. task



(b) Calibrated-vanilla CNNs for CIFAR-10 as in-dis. task

Figure 3.7: Comparison of the ROC curves of two calibrated CNNs in each plot. Evaluation (i.e. ROC curves) of the CNN trained on the *most protective OOD* set on varying unseen OOD sets are shown by the *solid lines*. While the curves related to the CNN trained on *the least OOD sets* are indicated by the *dashed line* (Sec. 3.4.2).

In-distribution	Seen OOD set	Unseen OOD sets Avg AUROC/ Avg FPR
SVHN	‡ISUN	94.73/31.97
	LSUN	99.25/ 4.39
	C10	99.75/0.41
	T-ImgNt	99.75/1.10
	⋆C100	**99.86/0.07**
CIFAR-10	‡SVHN	86.38 /75.04
	ISUN	86.20/77.03
	LSUN	93.31/ 38.59
	T-ImgNt	**93.89**/34.44
	⋆C100*	93.03/**26.13**
Urban-Sound	‡Command	59.15/63.06
	‡TuT	45.40/85.08
	⋆ECS	**71.41/60.67**

Table 3.2: The effect of OOD set selection on the performance of calibrated CNNs, where each trained on an OOD set, then evaluated on unseen OOD sets. We report the average of AUROC and FPR of calibrated CNNs on unseen OOD sets and test in-distribution set.

3.4.3 Comparison with Related Works

We compare A-CNN⋆ with three state-of-the-art methods, including baseline [35], ODIN [55], and Mahalanobis-distance-based detector [53] (briefly called Mahalanobis detector)[7]. These

[6]For brevity, we report the average of AUROCs and FPRs of *unseen* OOD sets.

[7]We use publicly-available Github codes of the state-of-the-art OOD detectors.

In-dist. Task	Methods	In-dis Rej. (FNR)↓ / Avg OOD Rej. (TNR)↑
SVHN	Baseline	4.06 / 58.07
	ODIN	4.71 / 69.34
	Mahalanobis Detector	6.40 / 97.41
	A-VGG*	**0.34** / **99.88**
CIFAR-10	Baseline	7.01 / 44.81
	ODIN	5.81 / 65.74
	Mahalanobis Detector	5.90 / 80.68
	A-VGG*	**5.65** / **85.10**
Urban-Sound	Baseline	22.83 / 40.96
	ODIN	27.38 / 47.45
	Mahalanobis Detector	16.13 / 45.42
	A-CNN*	**12.02** / **55.40**

Table 3.3: Comparison of A-CNN* with state-of-the-art approaches according to average rejection rates (TNR) of OOD sets and in-distribution rejection rate (FNR). Details are given in the supplementary.

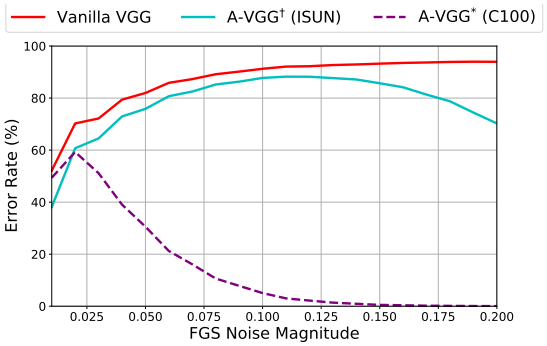
approaches attempt to detect OOD samples according to a specific threshold on the modified calibrated predictive confidence scores. As mentioned earlier, TNR and FNR of these methods are the same concept as OOD rejection rate and in-distribution rejection rate, respectively.

To set all hyper-parameters of these baselines, including threshold, we use two validation sets with 1000 samples each, for in-distribution and each OOD set. For ODIN, we find the optimal magnitude noise from $\{0.0, 0.0005, 0.0014, 0.001, 0.01\}$ and set temperature to 1000 for all experiments. For Mahalanobis-detector, the optimal magnitude of noise is selected among $\{0.0, 0.005, 0.002, 0.0014, 0.001, 0.0005\}$. To set threshold for detection, we obtain a threshold for each OOD task such that it achieves 95% TPR on the in-distribution *validation set*. Then, using the optimal tuned hyper-parameters (including threshold), we perform OOD detection process on OOD and in-distribution test sets for reporting TNR and FNR, respectively.

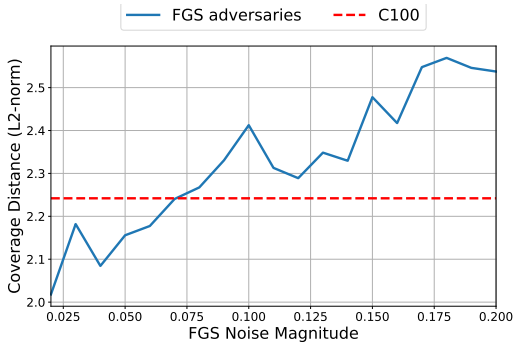
Note that Mahalanobis detector [53] and ODIN [55] have originally reported FPR (= 1-TNR) (on an OOD test set) at the threshold that produces 95% TPR on the in-distribution test set. However, more fair threshold setting would be finding the optimal threshold using separate a pair of validation set (the in-dist. set and the OOD set), then applying it on the test set of in-distribution and that of OOD task. Even by the use of a distinct validation set, we believe that setting a separate threshold for each OOD task is still unrealistic since in real-world scenarios, it is unknown the coming samples belong to which OOD task for setting its proper threshold. Therefore, the major challenge for the threshold-based approach is setting a global threshold, which shall lead to a global high detection rate on all OOD sets.

A comparison of the performances of the threshold-based approaches (which thresholds are fixed using validation sets) and A-CNN (as non-threshold-based approach) presents in Table 3.3. As it can be seen, A-CNN* performs *significantly* better than these threshold-based approaches

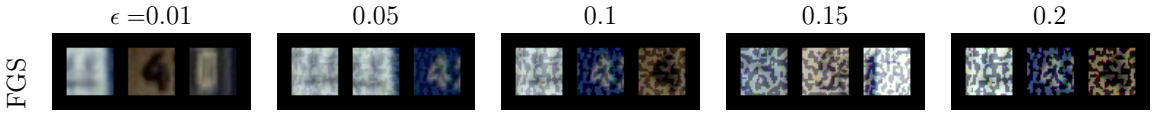
when comparing average OOD rejection and in-distribution rejection rates.

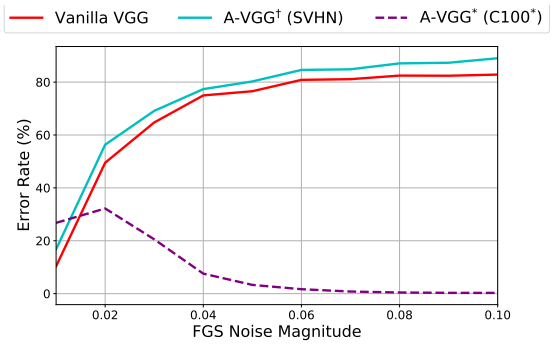


(a) Rejection rate of SVHN FGS adversaries

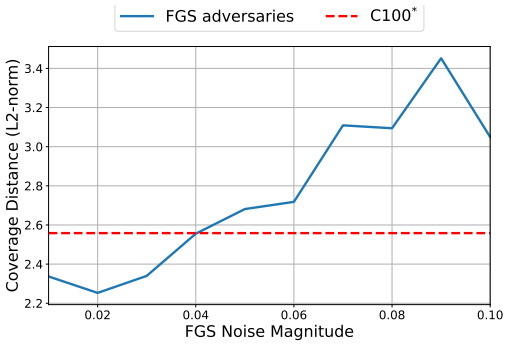


(b) CD of SVHN FGS adversaries





(c) Rejection rate of CIFAR-10 adversaries



(d) CD of CIFAR-10 adversaries

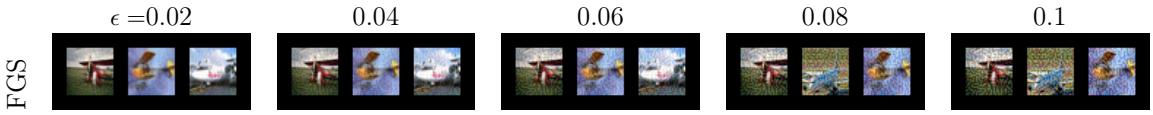


Figure 3.8: Performance of A-CNNs on FGS adversaries with various noise magnitudes for SVHN and CIFAR-10 as two in-distribution tasks. Sub-figures (a, c) show error rates of vanilla CNN, A-CNN* (trained on the most protective OOD set), and A-CNN† (trained on the least OOD set) on FGS adversaries as a function of noise magnitude for SVHN and CIFAR-10, respectively. Note Err rate = 1-(Acc rate +Rej rate). (b, d) Coverage Distance (CD) of FGS adversaries (their average distance to in-distribution sub-manifolds) for SVHN and CIFAR-10, respectively. In each sub-figure (b, d), the dotted red line is the Coverage Distance (CD) of the most protective OOD set for the given in-distribution.

3.4.4 Black-box Fast Gradient Sign (FGS) Adversaries

FGS adversaries with high noise level can be regarded as synthetic OOD samples. Even though such FGS adversaries contain perceptible noise, i.e., noticeable by human eyes, they can still fool vanilla CNNs easily [25, 101]. To explore the capability of A-CNN* in detecting such non-optimal adversaries, A-CNN*, A-CNN‡, and their vanilla counterparts are compared w.r.t. their error rates on FGS adversaries with varying amount of noise. We generated $5,000$ black-box FGS adversaries (from training in-distribution set) using another pre-trained vanilla CNN (different from the one evaluated here). Some samples are displayed in Fig. 3.8.

As evident from Fig 3.8, error rates (i.e., 1-Acc) of vanilla CNNs escalate as the amount of noise magnitude increases, showing the transferability of these black-box FGS adversaries. In contrast, the error rates (i.e., 1-(Acc+Rej)) of the A-CNNs* approach zero (Fig 3.8) as α increases since many of these FGS samples are rejected by A-CNNs*. Fig 3.8 (b) and (d) can explain this phenomenon; larger α causes generated FGS adversaries to be further away from the sub-manifolds of in-distribution classes (i.e., larger CD). When FGS adversaries enter the protected regions by A-CNN* (starting at the distance denoted by CD of the most protective OOD set, i.e., dotted red horizontal line), they are automatically rejected as OOD samples.

3.5 Conclusion

Our main goal is to differentiate OOD sets for recognizing a proper one for training an end-to-end augmented CNN with high detection rate on unseen OOD sets while maintaining in-distribution generalization performance. To this end, we feature an OOD set as protective if it can cover all of the in-distribution's sub-manifolds in the feature space. Then, we propose computationally efficient metrics as a tool for differentiating OOD sets for the purpose of selecting the most protective one. In practice, considering the increasing availability of new large-scale datasets for different applications, it is essential to have a tool for efficiently leveraging them for detecting unknown (OOD) samples.

Chapter 4

On the Capacity of Augmented CNN for Adversarial Examples Detection

4.1 Introduction

In [34, 36, 66], it is theoretically and empirically argued that a reliable CNN should reject the samples that are located in the regions far away from the in-distribution data generation distribution. In the previous chapter, we demonstrate that if an A-CNN is trained on a protective OOD set (i.e. a set with high coverage of all the in-distribution sub-manifolds) can detect a wide range of unseen unknown OOD samples as well as the FGS adversaries with a large adversarial noise magnitude. Recall that the detection is performed by classifying these samples into the extra class, which is called the "dustbin" class in this chapter. Using the Coverage Distance metric, we showed that, by increasing the adversarial noise, the generated FGS adversaries are distancing from the protected in-distribution sub-manifold, i.e. moving into the protected region classified as dustbin (extra class of A-CNN)(Fig 3.8).

Motivated by our previous observations, in this chapter, our goal is to further analysis and compare various A-CNNs, which are trained on different training sets for the dustbin class, in terms of detection rate of a wide array of black-box adversaries and their robustness to the white-box adversarial examples. Particularly, we aim to assess to what extent the A-CNN* (i.e. trained on a protective OOD set) is able to detect not only FGS adversaries but also other types of adversaries such as DeepFool and CW. *Our ultimate goal is to achieve an A-CNN, as a unified reliable model, that not only correctly classifies the clean samples but also identifies the anomaly samples, such as adversaries and unseen OOD samples, without the need to introduce an extra auxiliary regressor [53, 19, 63] or modifying its training objective function [52, 66].*

To enhance A-CNN*'s detection rates on larger types of adversarial examples, particularly those with the (sub)optimal amount of additive adversarial noises, e.g. CW, we augment the protective OOD set with a set of inter-class interpolated samples, created by interpolating

the samples from two distinct classes. Our motivation for interpolating the samples is that while they are conceptually similar to adversaries, generating them is straightforward and cost-effective. In other words, by imitating the adversaries, the inter-class interpolated samples contain simultaneously the features from two classes – like the adversaries that have the features from a fooling class and a true one. Other researchers have used the inter-class interpolated samples in the input space [120] or in the feature space [106] mainly to increase the accuracy rate of a vanilla CNN on the clean test samples. The latter approach, called manifold mixup [106], also showed that the vanilla CNN trained on the interpolated samples in the feature space can increase its robustness to adversaries (in terms of correctly classifying not rejecting them). However, we use the inter-class interpolated samples in the input space while the selection of the pair of the samples is guided by the feature space, with the goal of training an A-CNN that can reject a wider range of black-box adversaries. **The A-CNN trained on these interpolated samples along with the protective OOD set is called A-CNN*** throughout this chapter.

Instead of the inter-class interpolated and OOD data, it is also possible to train an A-CNN on an exhaustive and diverse set of adversarial examples (which includes all the known types of adversaries) in order to achieve a high detection rate on all adversaries, likewise [38]. However, generating such an inclusive and diverse set of adversaries, particularly for a large-scale dataset, can incur a significant computational burden. Moreover, we show later that the A-CNN trained on a set of Projected Gradient Descent (PGD) adversarial examples, i.e. a FGS-type adversaries, has lower detection rates on the *unseen* adversaries (particularly, on CW and DeepFool) than the A-CNN*. However, compared to A-CNN*, the latter A-CNN (i.e. PGD trained A-CNN) has higher detection rates on the other FGS variants (e.g. T-FGS and FGS).

Besides the quantitative analysis, we also aim to qualitatively investigate the impact of OOD training (by a protective OOD set) of A-CNN on the feature space and the decision boundaries in the input space. In other words, by visualizing the feature space (in section 4.3.5), we can observe how the A-CNN trained on a protective OOD set is be able to detect the adversaries, even never trained on any adversarial examples. We empirically exhibit that the A-CNNs have the capacity to learn more flexible feature space where some types of the black-box adversarial examples can be correctly disentangled from the in-distribution sub-manifolds and placed on a separated sub-manifold associated to the extra class (i.e. dustbin class). Furthermore, we show that some fooling regions (in the input space) of a vanilla CNN can be filled with dustbin (rejection) region in the A-CNN trained on a protective OOD set (Section 4.3.6).

4.2 Inter-class Interpolated Samples

While training A-CNN on a large and diverse set of adversaries can lead to adversarial detection by the A-CNNs [38], generating such a inclusive and diverse set, especially for a large-scale

clean dataset, is computationally expensive. To obviate the computational cost of generating a diverse set of adversaries, we use inter-class interpolation, a computationally efficient and simple procedure, for creating some synthetic samples that can conceptually mimic some behaviours of the adversaries. Since a CNN misclassifies an adversarial example into a fooling class, it can show that such an adversarial example simultaneously contain some features from two classes, i.e. its fooling and true classes, with the invisible features relevant to the fooling class and perceptually recognizable features related to the true class. Indeed, these invisible fooling features trigger the trained CNNs to misclassify the adversarial examples even though the fooling features can not be perceived by human eyes[1]. Conceptually similar to adversarial examples, the interpolated data contains the features from two classes (source and target classes).

Interestingly, Tramér et al. [102] argued that adding the inter-class mean difference (in input space) to clean samples can create a kind of model-agnostic adversarial examples which may fool some binary classifiers that satisfy a condition. Such agnostic-model adversaries (generated in the input space) can be regarded somewhat similar to our inter-class interpolated samples since both of them are adding the features (in input space) from a target class to a sample from the source class. We later show (in Sec. 4.3.2) that our interpolated samples (especially for MNIST task) augmented to a selected OOD set can significantly enhance the capacity of the A-CNN on detection of a larger variety of adversarial examples.

To create a set of synthetic samples, some pairs of correctly classified in-distribution samples from different classes are interpolated as follows:

$$\mathbf{x}_{k,k'} = \alpha \times \mathbf{x}_k + (1 - \alpha) \times \mathbf{x}_{k'}, \tag{4.1}$$

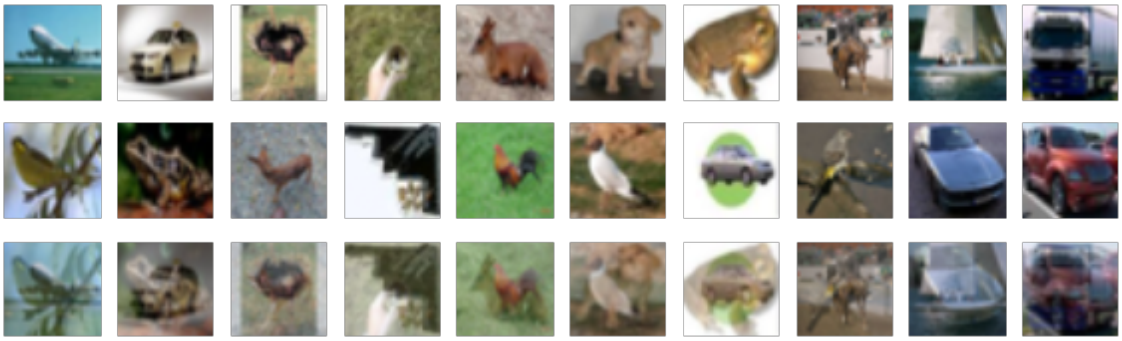
Where $\mathbf{x}_k$ and $\mathbf{x}_{k'}$ are the images from k and k' classes ($k \neq k'$), and α is the interpolation parameter. A large (or small) α produces less perturbed samples, that are more similar to a clean sample, and training an A-CNN on them (as the extra class) can result in unavoidable degradation of accuracy on the clean in-distribution samples. Therefore, we set $\alpha = 0.5$ in all the experiments.

4.2.1 Pair Selection

To create an interpolated sample, we need to define which pairs of samples, i.e. which classes and which two samples from the selected classes should be used. There is a tremendous number of combinations, particularly for a large-scale dataset with a large sample complexity and a

[1] In [42], the authors have shown that a clean sample contains robust and non-robust features, which the robust ones can be perceived by human observers while non-robust features can not. By training a CNN on the clean samples, it learns both robust and non-robust features. Because of learning non-robust features, the adversarial examples, which usually contain non-robust features of fooling classes, can trigger the CNN to misclassify them as the fooling classes.

(a) MNIST



(b) CIFAR-10

Figure 4.1: Some of our interpolated samples generated for all the classes of MNIST (a) and CIFAR-10 (b). The interpolated samples (shown in the 3rd rows of (a,b)) are composed of two images in the input (image) space: a randomly selected source sample (1st row) and its target sample (shown in 2nd row) is its nearest neighbor (from a different class) in the feature space.

high number of classes. To efficiently create an effective set of interpolated samples that are also conceptually similar to the adversarial examples, instead of randomly selecting the pairs, we interpolate each sample with its nearest neighbor (in the feature space) from a different target class. Two nearest samples (in the feature space) from two distinct classes are more semantically similar, thus, it is likely that an adversarial attack may easily perturbed them toward each other. Therefore, such an interpolated sample may mimic better their corresponding real adversarial examples. Moreover, finding the nearest neighbors in the low-dimensional feature space is more efficient, in terms of time and space complexity, compared to the high-dimensional raw input space. Likewise [2, 19], we use the penultimate layer of a vanilla CNN to represent the samples in the feature space.

Using a kd-tree, for a randomly selected training sample from a source class, we find its nearest neighbor from a different target class in the feature space. kd-tree's time complexity for finding the nearest neighbor is $O(\log N)$, with N as the number of training samples. For all the classes,

we generate the equal number of interpolated samples.

In Fig. 4.1, some interpolated samples for MNIST and CIFAR-10 are exhibited. As it can be seen, due to the selection of the nearest neighbor samples in the feature space, our MNIST interpolated samples are not visually too much perturbed. In other words, since the samples are semantically similar to each other, only a few features from a similar target sample perturbs the source one. We later also show (in Table 4.2) that these MNIST interpolated samples as a *single dustbin training source* can lead to an A-CNN with relatively high average of adversaries detection rates. Therefore, it seems that our interpolated samples are somewhat legit model-agnostic adversarial examples, like the model-agnostic adversaries generated by Tramér et al. [102]. However, we can not observe the similar behaviour for the CIFAR-10's interpolated samples as they are not only visually very perturbed, but also they can not be served as a single dustbin training source for producing an A-CNN with high expectation detection rate of adversarial examples.

4.3 Experimentation

4.3.1 Experimental Setting

MNIST with NotMNIST: As an protective OOD set for MNIST, we consider NotMNIST dataset that consists of 18,724 letters A-J printed with different font styles[2]. The size of images of MNIST and NotMNIST are equal, i.e. 28×28 pixels. We use a CNN (named Cuda-ConvNet) that has three convolution layers with 32, 32, and 64 filters of 5×5, respectively, and one Fully Connected (FC) layer with a softmax activation function[3]. In addition, dropout with $p = 0.5$ is used at the FC layer for regularization. To train an augmented version of Cuda-ConvNet, we utilize a training set comprising $50K$ MNIST training samples as in-distribution data as well as $10K$ randomly selected samples from NotMNIST dataset along with $15K$ interpolated samples generated from MNIST training samples. The remaining samples from NotMNIST ($\approx$8K) in conjugation with MNIST test samples are considered for the evaluation purposes.

CIFAR-10 with CIFAR-100: CIFAR-10 with $50K$ and $10K$ RGB images as training and test set are another in-distribution task and we used CIFAR-100* as its protective OOD set. To reduce overlap between the labels (classes) from CIFAR-10 and CIFAR-100, we ignore the super-classes of CIFAR-100 that are conceptually similar to CIFAR-10 classes. So, vehicle 1, vehicle 2, medium-sized mammals, small mammals, and large carnivores are excluded from CIFAR-100. Note that all the images are scaled to $[0, 1]$, then normalized by mean subtraction over the CIFAR-10 training set. For CIFAR-10, we use VGG-16 [91] architecture. To train

[2]Available at http://yaroslavvb.blogspot.ca/2011/09/notmnist-dataset.html.

[3]To read more about the configuration of this CNN, the readers should refer to https://github.com/dnouri/cuda-convnet/blob/master/example-layers/layers-18pct.cfg

an augmented VGG-16, $15K$ randomly selected samples from CIFAR-100* along with $15k$ interpolated samples from CIFAR-10 training set (labeled as dustbin class) are appended to CIFAR-10's training set.

The hyper-parameters of attack algorithms: Each adversarial generation algorithm comes with a few hyper-parameters. We summarize the details on the values of the hyper-parameters used for generating adversarial examples in Table 4.1. In addition, we present the error rate of another vanilla CNN, which differs from the one used for generating the adversaries, in order to show that transferability degree of these adversarial examples. Due to high time complexity of CW, we considered 100 randomly selected images for each dataset. For each selected image, likewise [115], two targeted adversarial samples are generated, where the target classes are the least likely and the second most likely classes according to the predictions provided by the underlying CNN. Thus, in total, 200 CW adversarial examples are generated per dataset. To increase the transferability of CW, we use $\kappa = 20$ for MNIST and $\kappa = 10$ for CIFAR-10. For other attacks (variants of FGS as well as DeepFool), we utilized 2K correctly classified test samples.

Dataset	Attacks	Hyper-parameters	Error rate (%)
MNIST	FGS	$\epsilon = 0.2$	65.86
	TFGS	$\epsilon = 0.2$	80.01
	PGD	$\epsilon = 0.02$, $\alpha = 0.2$, # of iterations $= 20$	74.10
	DeepFool	–	98.11
	CW	$\kappa = 20$	77.51
CIFAR-10	FGS	$\epsilon = 0.03$	63.84
	TFGS	$\epsilon = 0.03$	63.76
	PGD	$\epsilon = 0.003$, $\alpha = 0.03$, # of iterations $= 20$	48.81
	DeepFool	–	43.18
	CW	$\kappa = 10$	57.50

Table 4.1: The hyper-parameter setting of the adversarial attacks in our experiments and the error rate of another vanilla CNN on them to reflect the degree of transferability of these black-box adversaries

4.3.2 Black-box Adversarial examples

In [9, 74, 96], it has been empirically shown that many of adversarial examples generated from a learning model, e.g. an attacker CNN, can be transferred and attack different victim models that their architecture and parameters (e.g. weights and bias) are securely hidden from all the outside-viewers, including the attacker. These attacks are called *transferable black-box attacks*. Using a vanilla CNN, we generate assorted types of adversaries such as DeepFool, FGS, PGD, T-FGS and CW for the correctly classified clean test samples of each in-distribution task. We assess and compare the behaviour (in term of accuracy, error and rejection rates) of different A-CNNs on these range of test adversaries in Table. 4.2. Note that the A-CNNs are

	Test Set	Metrics	A-CNN Adve. (PGD)	A-CNN OOD*	A-CNN Interp.	A-CNN OOD* ∪ Interp.
MNIST / NotMNIST (CudaConv)	In-dist. test	Acc.	99.54	99.47	99.50	99.48
		Rej.	0.00	0.02	0.02	**0.08**
		Err.	0.46	0.51	0.48	**0.44**
	FGS	Acc.	0	19.15	10.47	0.34
		Rej.	100	65.19	83.31	99.59
		Err.	**0**	15.66	6.22	0.07
	PGD	Acc.	0	39.20	3.10	0.01
		Rej.	100	23.20	95.69	99.90
		Err.	**0**	37.60	1.21	0.09
	T-FGS	Acc.	0	1.17	1.05	0
		Rej.	100	95.92	98.58	100
		Err.	**0**	0.37	**0**	**0**
	DeepFool	Acc.	6.21	11.45	9.87	5.36
		Rej.	35.66	4.72	78.06	89.84
		Err.	58.13	83.83	12.07	**4.80**
	CW	Acc.	18.00	27.50	15.50	7.50
		Rej.	28.00	5.99	55.00	77.49
		Err.	54.00	66.51	29.50	**15.01**
	All Adv. (Avg)	Acc.(↑)	7.03	**20.19**	9.08	2.65
		Rej. (↑)	72.73	39.02	81.12	**93.36**
		Err. (↓)	20.24	40.79	9.8	**3.99**
CIFAR-10 / CIFAR-100 (VGG16)	In-dist. test	Acc.	91.66	88.58	90.38	86.65
		Rej.	0.10	5.69	1.55	**8.74**
		Err.	8.24	5.73	8.07	**4.61**
	FGS	Acc.	0	31.90	38.98	29.50
		Rej.	100	36.81	6.93	45.11
		Err.	**0**	31.29	54.09	25.39
	PGD	Acc.	0	54.27	59.22	50.28
		Rej.	100	12.94	5.06	24.76
		Err.	**0**	32.79	35.72	24.96
	T-FGS	Acc.	0	27.17	33.51	24.35
		Rej.	100	41.08	7.32	51.33
		Err.	**0**	31.75	59.17	24.32
	DeepFool	Acc.	46.52	44.22	54.48	42.81
		Rej.	14.12	33.20	5.16	40.26
		Err.	39.36	22.58	40.36	**16.93**
	CW	Acc.	44.50	46.50	48.50	39.00
		Rej.	1.50	18.50	8.00	39.50
		Err.	54.00	35.00	43.50	**21.50**
	All Adv. (Avg)	Acc.(↑)	18.21	40.82	**46.93**	37.19
		Rej. (↑)	**63.12**	28.50	6.50	40.19
		Err. (↓)	**18.67**	30.68	46.57	22.62

Table 4.2: Comparison of the performance of A-CNNs (in term of accuracy ("Acc."), rejection ("Rej.") and misclassification ("Err.") rates) on the black-box adversaries attacks for MNIST and CIFAR-10 in-distribution tasks. Underline, red color, and **boldface** denote the best accuracy (i.e. correctly classifying adversaries), the highest rejection rate, and **the lowest error rate** for each attack, respectively. OOD* denotes a protective OOD set.

identical in the structure but trained on different dustbin training set; either on I) training PGD adversaries, II) the selected (protective) OOD set, III) interpolated samples, or IX) the union of protective[4] OOD set and the interpolated samples. Comparing the error rate of these A-CNNs allows us to show the impact of each dustbin training set on reducing the risk of the black-box adversaries.

[4]For brevity reasons, we simply call a protective OOD set as OOD set in Table 4.2.

Regarding the test in-distribution set in Table 4.2, we observe a slight drop in test accuracy rates of the A-CNNs (except for that trained one on PGD adversaries). Despite the drop in the accuracy of A-CNN* (OOD*∪ Interp.) on the clean test samples, its error rate also is decreased (i.e. the smaller number of mistakes) as it rejects some the hard-to-classify clean samples instead of incorrectly classify them.

It is not surprising that the A-CNN trained on a set of **PGD** adversaries can perfectly reject almost all the variants of FGS adversaries (i.e. FGS and T-FGS) (as previously shown in [38]). However, it is not able to reject those non-FGS variants of adversaries, namely CW and DeepFool. **Thus, using one type of adversaries, e.g. PGD, as a single training source for the extra dustbin class of A-CNN can not necessarily lead to the high detection rate of unseen types of adversaries.** Although A-CNN* is never trained on an explicit set of adversaries, we can see that it is consistently rejecting a large portion of *unseen* adversaries generated by these five known attacks.

Consequently, if an A-CNN is trained on a proper dustbin training set, e.g. a protective OOD set along with a set of interpolated samples, our empirical results demonstrate that such an A-CNN has the capacity to detect different types of adversarial examples as well as unseen OOD sets (as empirically shown in Chapter 3)[5]. Indeed, *a properly trained A-CNN can be regarded as a unified method for correctly classifying the clean samples while detecting the anomaly samples, such as adversaries and OOD samples, without the need to introduce an extra auxiliary regressor [53] or the drastic modifications to the architecture of a vanilla CNN [65, 66] or its training procedure [52].*

The Impact of Protective v.s. Non-protective OOD Set on Adversaries Detection

In the preceding chapter, we show the extensive empirical evidences that an A-CNN trained with a protective OOD set has a superior performance over the one trained on a non-protective OOD set for the detection of unseen OOD sets. Here, in Table 4.3, we demonstrate that the performance of these two A-CNNs on detection of an array of adversaries for CIFAR-10 in-distribution task. The A-VGG* trained on CIFAR-100* (C100*) performs significantly better at rejecting of adversaries than the A-VGG‡ trained on SVHN (as a non-protective OOD set for CIFAR-10). Therefore, due to rejecting a greater number of the adversaries, A-VGG* obtains smaller error rates over the adversaries than A-VGG‡. Note that their accuracy rates, reflecting the number of correctly classified adversaries, are only slightly differ.

Consequently, to have an A-CNN with high detection rate of a vast range of adversaries, the protective OOD set is preferred over the non or less protective one. Considering these present results (in Table 4.3) along with those in Table 3.1, the A-CNN trained on a protective OOD set can outperform the A-CNN trained on a non (less) protective one for detecting a wide

[5]See comparison of A-CNN (PGD) with A-CNN* on detection of unseen OOD sets in Sec. 4.3.4

range of both unseen adversaries and OOD sets.

Test Set	Metric	A-VGG‡ (SVHN)	A-VGG* (C100*)
In-dist. Test	Acc.	**91.71**	88.58
	Rej.	0.07	5.69
	Err.	8.22	**5.73**
FGS	Acc.	37.65	31.90
	Rej.	0.0	36.81
	Err.	62.35	**31.29**
PGD	Acc.	52.50	54.27
	Rej.	0.0	12.94
	Err.	47.50	**32.79**
T-FGS	Acc.	31.28	27.17
	Rej.	0.05	41.08
	Err.	68.67	**31.75**
DeepFool	Acc.	53.88	44.22
	Rej.	0.05	33.20
	Err.	46.07	**22.58**
CW (L_2)	Acc.	47.00	46.50
	Rej.	7.00	18.50
	Err.	46.0	**35.0**

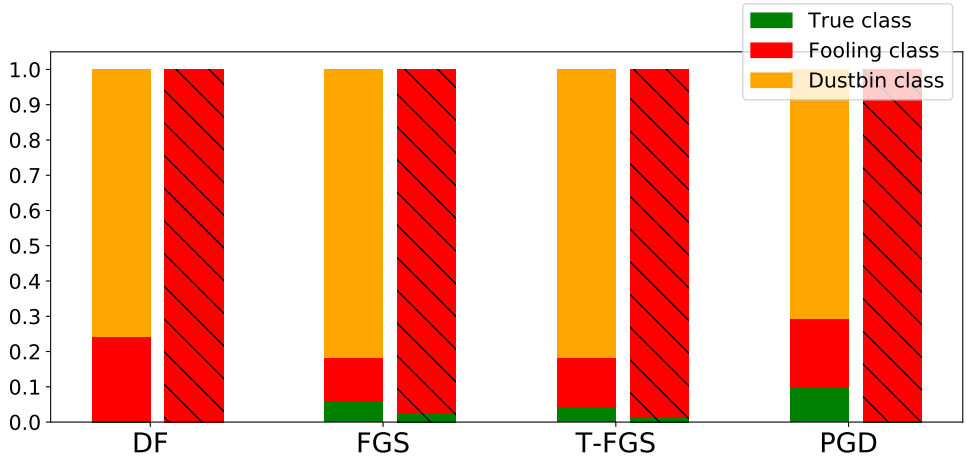
Table 4.3: Comparison of the performance of A-VGGs, which are trained on a protective v.s. non-protective OOD sets, on detection of adversaries for CIFAR-10 as in-distribution task.

4.3.3 White-box Adversarial Examples

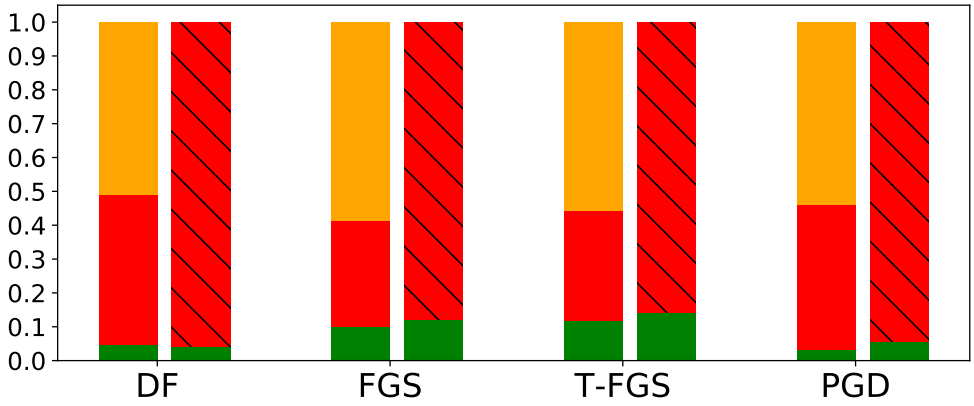
Using $1K$ test samples from MNIST (CIFAR-10), we generate various types of white-box adversaries (with the same hyper-parameters as the black-box ones) by an A-CNN* and a vanilla CNN. Since the augmented CNN has an explicit rejection class, the generated adversarial samples with fooling class *dustbin* are discarded as a threat (attack) to A-CNN*. Indeed, the A-CNN can already detect and reject such samples as it classifies them as dustbin. Therefore, a successful adversarial example should fool the A-CNN* toward a class different than the extra class (dustbin) and its associated true label.

In Fig. 4.2, we measure the success rate of adversarial algorithms that directly use either the A-CNN* or its vanilla counterpart. The success rate reflects the rate of the generated samples that are fooling their underlying model toward a fooling class (the success rate is represented by red color). The fail rate (the failure of the algorithm to generate adversaries) is the rate of the generated samples that are classified as their associated true class (represented by green color). Finally, the dustbin rate (only for A-CNN*), which is equivalent to rejection, is the rate of the samples that are classified as a dustbin class (exhibited by orange color).

Note that in the untargeted attack algorithms, such as FGS and PGD, we can not pre-determine



(a) MNIST



(b) CIFAR-10

Figure 4.2: The fooling (in red), dustbin (in orange), and failure (in green) rates of generating white-box adversaries using A-CNN* and a vanilla CNN (indicated by crosshatched pattern) for 1K test samples of MNIST (a) and CIFAR-10 (b).

the fooling class, i.e. we have no contorl to fool the model for a given sample toward a specific (selected) fooling class. Because their objective function for adversaries generation is to maximize the loss of the model on a given pair of a sample and its true label (class) Eq.1.6, meaning it can end up to any possible fooling classes (including the dustbin class for the A-CNN). However, for T-FGS and DeepFool, we can choose to not fool toward the dustbin class. Even by discarding the dustbin class from the set of candidate fooling classes for these targeted attacks, we observe (in Fig. 4.2) that the T-FGS and DeepFool are still generating the samples that the A-CNN* can already rejected.

As it can be seen, for MNIST (in sub-figure 4.2 (a)), a significant portion of adversaries generated by A-CNN* are fooled toward the dustbin class (orange color) and a smaller portion

of them are truly fooled (red color). However, a large portion of the generated samples by the vanilla CNN (the unprotected one) successfully fools it (red color). However, for CIFAR-10, the dustbin rate of A-CNN* is equal (or slightly larger) to its fooling rate. Consequently, we may consider such A-CNNs as a potentially more robust models (compared to their vanilla counterparts) to these types of adversarial white-box attacks as they can reject to create some fooling adversarial examples. Note that we are not claiming that these A-CNNs are *completely robust* to all types of adversaries with various values for their hyper-parameters. Rather, we empirically show that a properly trained A-CNN (which is trained on an appropriate dustbin training set, e.g. a protective OOD set and the interpolated samples) is more robust (compared to a vanilla CNN) to some types of adversarial examples. In fact, it is still possible that one can meticulously generate some adaptive adversaries to evade the dustbin class of a properly trained augmented CNN.

In-distribution	OOD set	A-CNN (PGD) Rejection (%)	A-CNN* Rejection (%)
MNIST	NotMNIST (unseen)	80.75	**99.98**
	Omniglot (unseen)	100	100
	CIFAR-10(gc) (unseen)	99.89	**100**
CIFAR-10	CIFAR-100† (unseen)	0.82	**96.21**
	DS-ImageNet†(unseen)	12.08	**87.49**
	SVHN (unseen)	0.04	**92.29**
	LSUN (unseen)	15.90	**84.80**

Table 4.4: The rejection rate of the *unseen* OOD sets by A-CNN (PGD) and A-CNN* for MNIST and CIFAR-10.

4.3.4 OOD Samples Detection

We argue in chapter 3 that the various OOD sets are not equal for training an effective A-CNN, which is able to detect a vast array of unseen OOD sets. Here we aim to show whether an adversarial training set can lead to training of an A-CNN with high detection rate of unseen OOD set. In table 4.4, we compare A-CNN (PGD) and A-CNN* for detecting various unseen OOD sets across MNIST and CIFAR-10. For MNIST, the A-CNN (PGD) can detect the unseen OOD sets as good as the A-CNN*. However, for a more complex task, i.e. CIFAR-10, the A-CNN (PGD) is not able to effectively reject/detect the unseen OOD sets, compared to its counterpart A-CNN*. Therefore, the adversarial training set consists of one type of adversaries, e.g. PGD, can not be a proper training set for training an A-CNN with high detection rates of unseen OOD sets, particularly for complex in-distribution tasks. Recall from Table 4.2 that A-CNN (PGD) can detect the variants of FGS algorithm, e.g. T-FGS and FGS, but its detection rates are inferior for the unseen adversaries such as CW and DeepFool, compared to the A-CNN*.

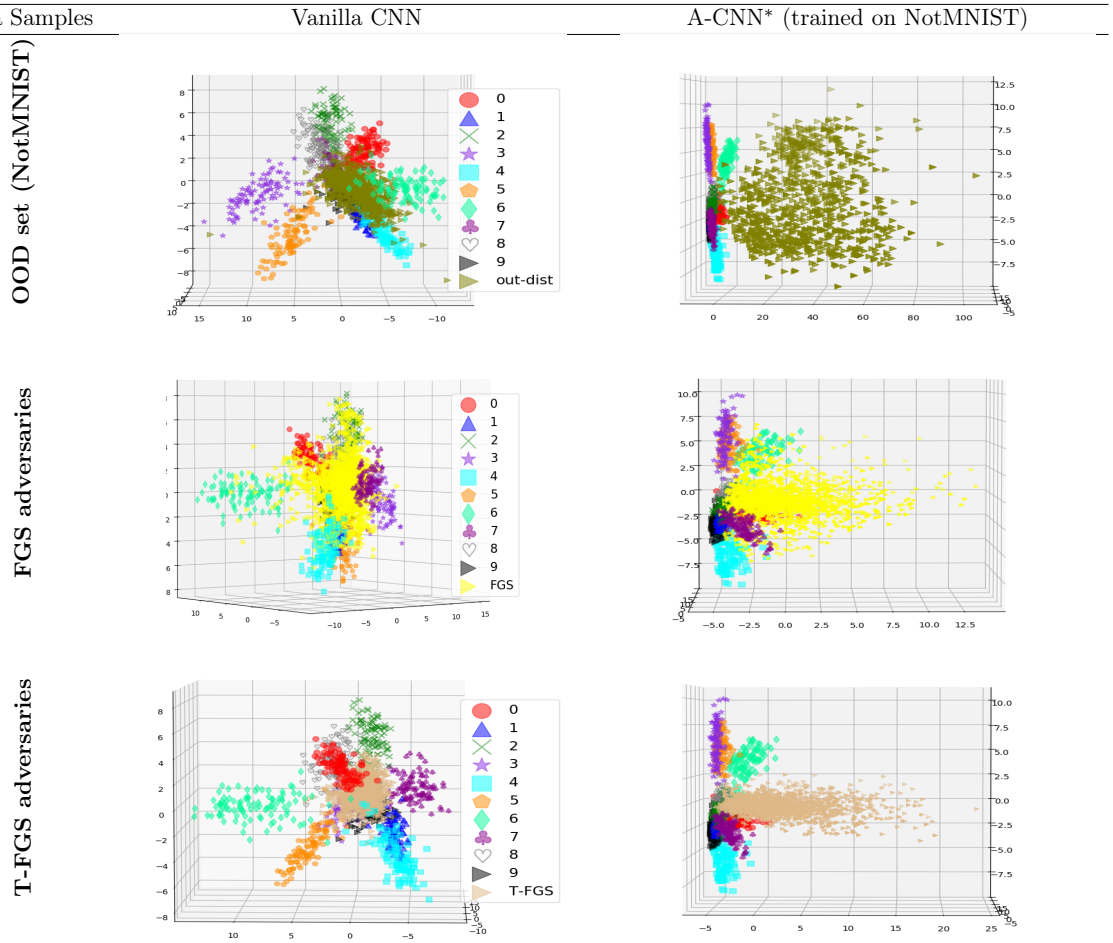


Figure 4.3: Visualization of MNIST randomly selected training samples, its adversarial examples (FGS and T-FGS), and the OOD samples (from NotMNIST) in the feature spaces achieved by a vanilla CNN and an A-CNN* for MNIST as the in-distribution task.

4.3.5 Visualization of the Feature Space

To represent the in-distribution instances, OOD samples, FGS and T-FGS adversaries in the feature space of a CNN (either the vanilla CNN or A-CNN*), we pass them to each CNN's penultimate layer. Using PCA, the dimensionality of data in the feature space is reduced to 3 for the visualization purposes.

In Figures 4.3 and 4.4, we visualize the feature spaces of A-CNN* (trained on a protective OOD set) and its vanilla counterpart for MNIST and CIFAR-10, respectively, in the presence of OOD samples and FGS types of adversaries[6]. Looking at the the feature space of a vanilla CNN, it can be observed that the OOD samples are mostly entangled with (or placed very nearby to) the in-distribution sub-manifolds. This may explain the behaviour of a vanilla CNN that produces very uncalibrated predictions (i.e. high confidence) for many OOD samples

[6]To see the feature space in 2D space, created using T-SNE, the interested readers can refer to Fig B.1 in Supplementary section.

as they are entangled with (or placed on or very nearby) the in-distribution sub-manifolds. While by training an A-CNN on a protective OOD set, i.e. a set with high coverage of all the in-distribution sub-manifolds, we force the network to learn a separate sub-manifold to disentangle the unseen anomaly samples from the in-distribution sub-manifolds.

While A-CNN* is never trained on any adversaries, we can observe that OOD training of an A-CNN (by a protective OOD set) can cause disentanglement of a relatively large portion of the black-box adversaries (FGS and T-FGS) from the in-distribution sub-manifolds by placing them on a separated extra sub-manifold (which belongs to dustbin class). Moreover, it can be seen that some portion of these adversaries, especially for CIFAR-10 in Fig. 4.4, are not separated from the in-distribution manifolds. This can happen as some of them are indeed correctly classified by the A-CNN* as reported in Table 4.2.

Although training A-CNN on the most protective OOD set results in disentanglement of unseen unusual samples (e.g. OOD and adversarial instances) in the feature space, if it is trained on the least protective one, we can not observe this interesting behaviour in the feature space (see Fig. 4.5). For example, SVHN, as an non-protective OOD set for CIFAR-10, can not cover some of the in-dist. sub-manifolds (Fig 4.4 in 2nd row of vanilla VGG (1st column)). Training an A-CNN on it (i.e. A-VGG‡) can not lead to properly separating the unseen OOD samples (and the unseen adversaries) from the in-dist. sub-manifolds as it can be seen in Fig 4.5. Because of that, the A-VGG‡ has a low detection rate of unseen OOD samples and the adversarial examples (in Table 4.3.)

4.3.6 Visualization of Classification Regions in the Input Space

We aim to visualize and compare the decision boundaries and classification regions (in the input space) that are learned by A-CNN and its vanilla counterpart. To this end, we plot 2-dimensional cross-section for several clean samples [111]. Using a trained model $h(.)$, each coordinate (x, y) in a 2-D cross-section reflects the model's class prediction (shown by a color) for a sample that is crafted by moving a clean sample x units in an adversary direction (denoted as $\mathbf{e}_1^{\mathbf{x}} \in \mathbb{R}^D$) and y units in a random direction (denoted as $\mathbf{e}_2^{\mathbf{x}} \in \mathbb{R}^D$) perpendicular to the adversary direction (i.e. $\mathbf{e}_1^{\mathbf{x}} \perp \mathbf{e}_2^{\mathbf{x}}$) (Eq. 4.2). In our results, the adversary direction $\mathbf{e}_1^{\mathbf{x}}$ is achieved by an adversarial algorithm such as FGS and DeepFool using a distinct vanilla CNN (i.e. this is the black-box adversarial direction).

We create a 2-D cross-section (denoted by $f : x \times y \longrightarrow \{1, \ldots, K\}$) for a clean sample $\mathbf{x} \in \mathbb{R}^D$ (shown by a black dot in the middle of the plot), its adversarial direction $\mathbf{e}_1^{\mathbf{x}}$, a random direction $\mathbf{e}_2^{\mathbf{x}}$ orthogonal to $\mathbf{e}_1^{\mathbf{x}}$, and a learning model $h : \mathcal{X} \longrightarrow \{1, \ldots, K\}$ that maps the crafted sample into one of K classes as follows:

$$f(x, y|\mathbf{x}, \mathbf{e}_1^{\mathbf{x}}, \mathbf{e}_2^{\mathbf{x}}) = \arg\max_k h(\mathbf{x} + x\mathbf{e}_1^{\mathbf{x}} + y\mathbf{e}_2^{\mathbf{x}}), \tag{4.2}$$

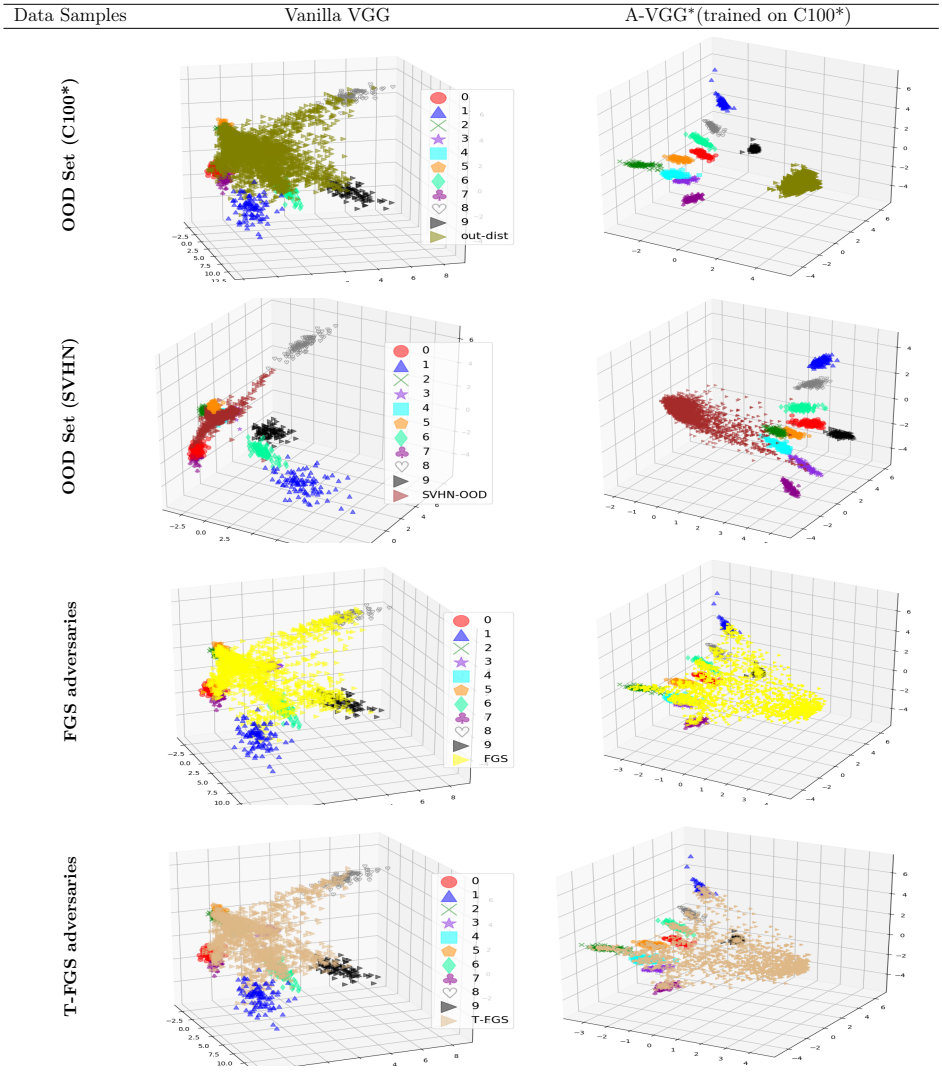


Figure 4.4: Visualization of some CIFAR-10's (as the given in-distribution set) training samples, its adversarial examples (FGS and T-FGS), and the OOD samples selected from C100* and SVHN in the feature spaces of a vanilla VGG (2nd column) and the A-VGG* (3rd column) for CIFAR-10 as the in-distribution task. To interact with these 3d plots, interested readers may refer to `https://github.com/mahdaneh/Out-distribution-learning_FSvisulization` for downloading the IPython notebook file (.ipynb).

where $x, y \in [-\epsilon, \epsilon]$ and h is either the vanilla CNN or A-CNN. We set $\epsilon = 0.2$ for MNIST and $\epsilon = 0.1$ for CIFAR-10 to create the cross-sections in Figures 4.6 and 4.7, respectively. As it can be seen, the fooling classification regions of the vanilla CNN are occupied (or shrunk) by dustbin regions (indicated by orange) in A-CNN*. In other words, while the vanilla CNN is fooled by the samples created through moving in the adversary directions, A-CNN* can detect and reject these adversaries by classifying them as dustbin.

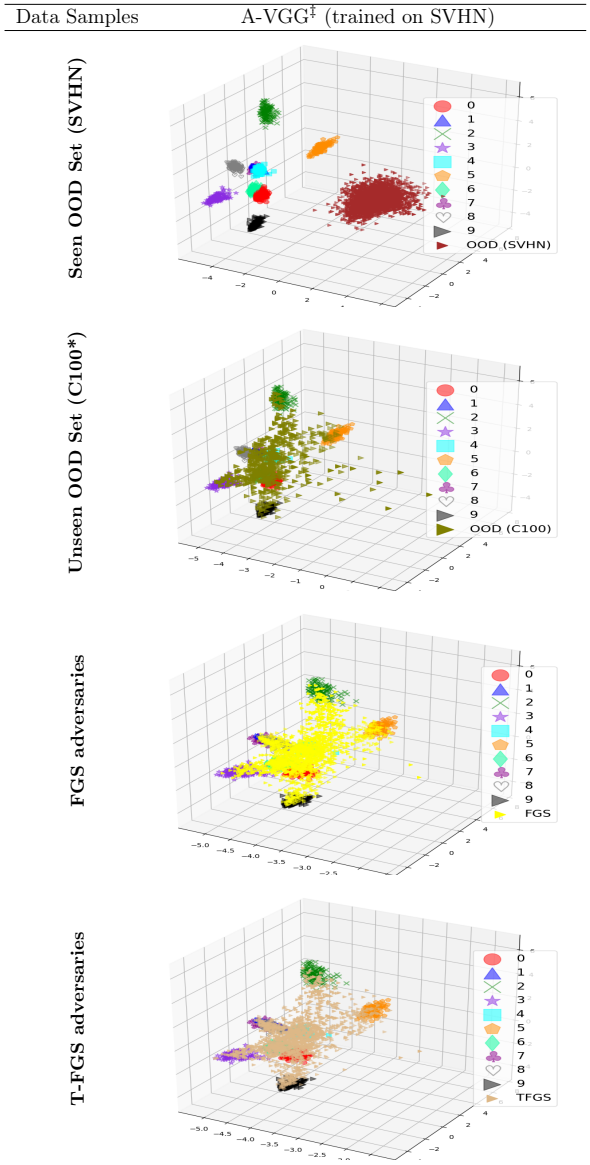


Figure 4.5: Visualization the impact of a non-protective OOD set on the feature space of A-VGG for CIFAR-10 (in-dist.). Training an A-CNN on SVHN (as the least protective OOD set for CIFAR-10) can not cause the separation of the unseen OOD samples (2nd row) and the adversaries (3rd and 4th rows) from the in-dist. sub-manifolds. Indeed, these samples are still entangled (placed nearby) to the in-dist. sub-manifolds. Thus, this A-CNN‡ exhibits low detection rates on both unseen OOD sets and the adversaries (Table 4.3).

4.4 Conclusion

The main goal of this chapter is to provide extensive empirical evidences to show the potential of A-CNN for detecting unseen adversaries, if it is trained on an appropriate training set. We compare the capacity of various A-CNNs on detection of an wide array of adversarial examples generated in the black-box and white-box models. There are several training source for training

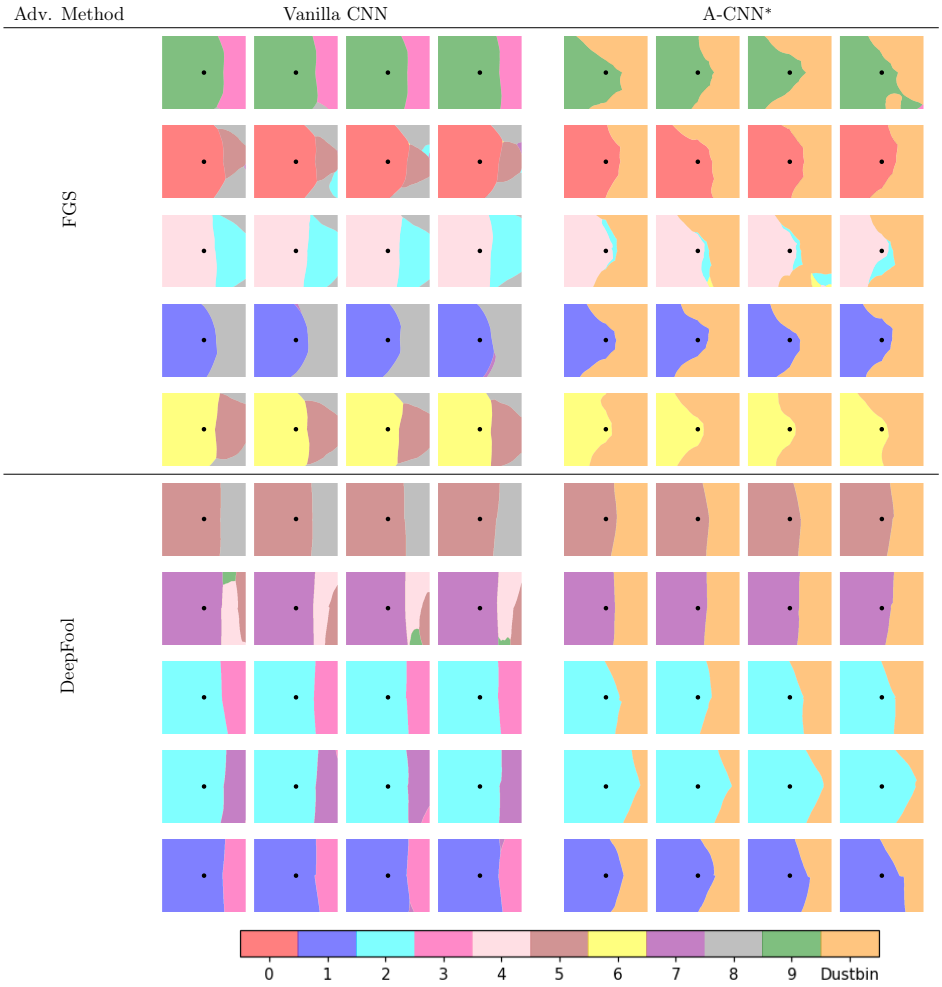


Figure 4.6: Illustration of cross-section (a.k.a. church-window) plots [111] for various MNIST clean instances in order to observe and compare the class predictions (classification regions) of the vanilla CNN and its augmented counterpart for the samples that are created by moving a clean sample in two directions i.e. $\mathbf{e}_2^{\mathbf{x}}$ and $\mathbf{e}_2^{\mathbf{x}}$. The adversary direction $\mathbf{e}_1^{\mathbf{x}}$ for the clean sample $\mathbf{x}$ is achieved by FGS or DeepFool using a distinct vanilla CNN. Also, for a fixed $\mathbf{e}_1^{\mathbf{x}}$, we choose four different random directions $\mathbf{e}_2^{\mathbf{x}}$ that are orthogonal to the given adversary direction to create four different cross-sections (each block of four cross-sections). The fooling classification regions of the vanilla CNNs are occupied by dustbin regions (indicated by orange) corresponding to the extra class in their augmented counterparts.

an A-CNN, including a set of adversarial examples, a protective OOD set, and synthetic samples (e.g. interpolated ones). Using these different training sets, we train several A-CNNs to compare their ability of detection of adversaries. Our experimental results reveal that A-CNN* (i.e. is trained on a protective OOD set along with a set of interpolated samples) can lead to relatively lower risk rate (in term of error rate) on a wide spectrum of *unseen* black-box adversaries, compared to the A-CNNs trained on a set of adversaries or a non-protective OOD set. Moreover, comparing a vanilla CNN, we show that the A-CNN* can be more robust to

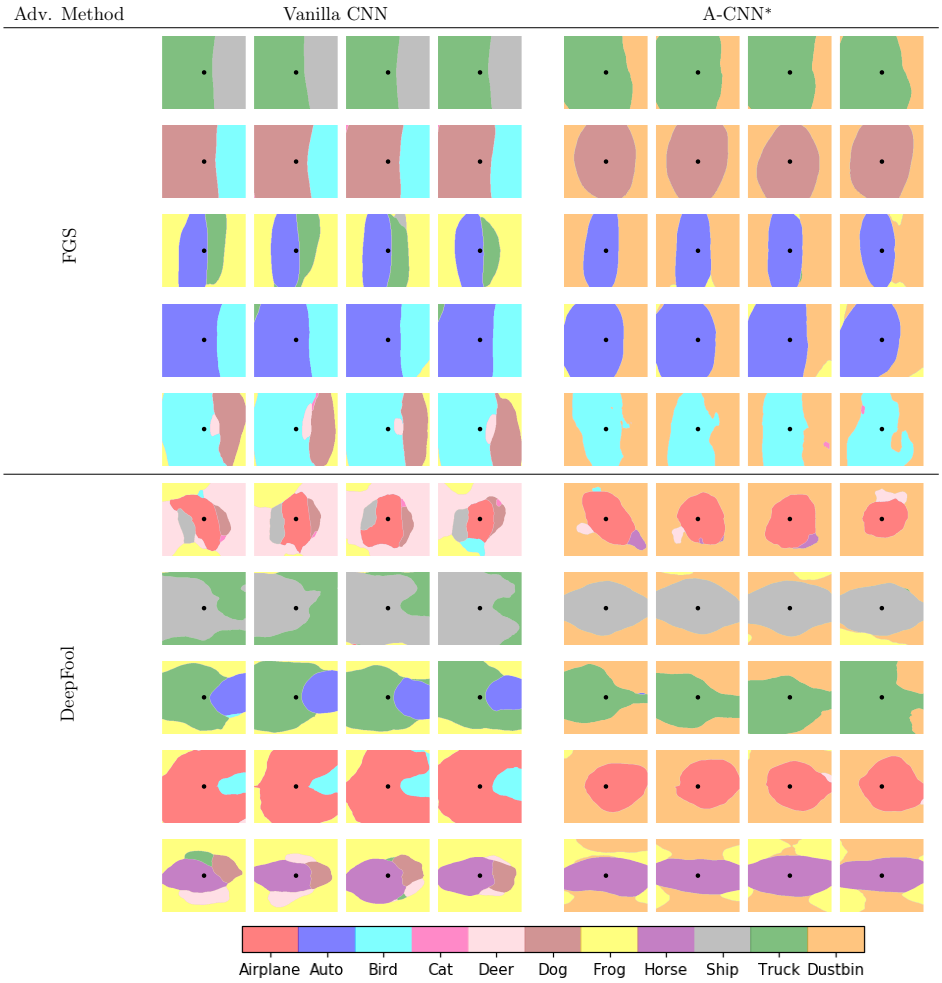


Figure 4.7: Illustration of cross-section plots for various CIFAR-10 clean test instances.

white-box FGS variants (with a specified hyper-parameters e.g. ϵ) and DeepFool as it generates a lower number of successful adversaries that can truly fool itself. This is because that the A-CNN* is able to reject many of these white-box adversaries, as they are classified into its extra class.

Furthermore, we visualize the feature space and the decision boundaries in the input space of A-CNN* in order to observe how OOD training impact these spaces, where the adversaries and OOD samples can be distinguished from the clean in-distribution ones. In the feature space of this A-CNN, an extra sub-manifold associated to the extra class allows some unseen adversaries to be disentangled from the in-distribution sub-manifold and be placed on this new sub-manifold. Visualizing the classification regions achieved by A-CNN* can show some of the fooling regions (found by FGS and DeepFool) that surrounded the clean samples can be occupied by the dustbin class, reaffirming the capacity of A-CNN* for rejecting adversaries. Finally, considering A-CNN's capability of rejecting a wide range of unseen OOD sets (showed in chapter 3) as well as the adversarial examples (experimentally demonstrated in this chapter),

we may hope to develop a unified and reliable model that not only correctly classifies the clean samples but also identifies the anomaly samples, such as adversaries and unseen OOD sets, without introducing a drastic modification of CNN's architecture or its training algorithm.

Chapter 5

Robust Object Detector for Partially Labelled datasets

5.1 Introduction

Modern CNN-based object detectors such as Faster Region CNN (R-CNN) [81] and You Only Look Once (YOLO) [78] achieve remarkable performance when trained on fully labeled large-scale datasets, which include both instance-level annotations, i.e. bounding boxes around each object of interest (a.k.a. Region of Interest (ROI)), and image-level labels, i.e. category of the object enclosed in a bounding box. On the one hand, collecting a dataset with full annotations, specifically bounding boxes, can be a tedious and costly process. On the other hand, the object detectors such as R-CNN and YOLO show that their performance is dependent on having access to such fully labeled datasets. In other words, training them on partially labelled datasets (i.e. containing instances with missing-labels) leads to a drop in generalization performance [113, 121].

To reduce annotation cost of large-scale datasets (e.g. OpenImagev3 [45]), partial-annotation policy considers to annotate only one instance of each object presented in a given image. For example, in an image containing a herd of sheep, only one sheep is annotated, instead of fully annotating all the sheep instances in the image. This policy causes some missing bounding box annotations but interestingly no missing image-level labels since at least one instance of each existing object in each image is annotated. Beside this, the datasets with partially-labeled instances can happen in other situations, including unintentional errors occurring in the annotation process and for the case of merged dataset.

The motivation of using merged datasets has been arouse in our industrial project (namely BRiTE), where we aim at training an object detector to detect both road's generic objects and traffic signs of Quebec province. To train such a unified model, there is no such a unified fully-labelled dataset from Quebec's traffic signs and road objects. In reality, we only have our

own unlabelled dataset that is gathered by filming from Quebec's roads. Unfortunately, fully labelling our own dataset is very expensive and tedious. To reduce the cost of labelling, we propose to label only the Quebec's traffic signs in our dataset and leave the other interesting objects unlabelled, likewise the German traffic sign dataset [39]. Then, by merging our partially-labelled dataset (annotated Quebec's traffic signs) into some publicly available labelled datasets (like Kitti [21] or BDD100k [116]), we can build a larger dataset with some labelled instances from all our objects of interest, then finally train a unified object detector to detect both Quebec's traffic signs and road's generic objects. Therefore, likewise [77], we aim at combining several datasets from similar (or the same) contexts but with disjoint (or partially disjoint) sets of Object of Interest (OoI), in order to **efficiently** construct a larger dataset that includes a wider range of objects of interest. Furthermore, merging several datasets, particularly those comprising the OoIs that are appeared in different poses, illuminations, styles, and etc (i.e. different domains), can possibly mitigate domain-shift challenge [77, 108].

In addition, such merged datasets can facilitate training of an integrated object detector, which in turn can potentially lead to a significant reduction of time and computational cost. Training and inferring from a unified object detector on a merged dataset is more efficient in terms of memory and computational resources, compared to training several disjoint object detectors; each for one of the constituting datasets. Particularly, inferring from a unified object detector is quicker, thus it is more appealing for the embedded devices that should make decisions in real-time but have limited computational resources (e.g. self-driving cars). In addition, training a unified model obviates the need to combine decisions made by the various models, which can be tricky and lead to sub-optimal solutions. Finally, merging datasets and training a unified object detector on it can pave the path toward the development of an universal object detector (e.g. [109]).

Despite the great potentials of the merged datasets for reduction of computational, time, and annotation costs, merging several datasets, unfortunately, results in missing-label instances (i.e. both image-level and instance-level labels are missing). Because the positive objects that are annotated in one constituent dataset are considered as non-interesting (negative) objects (thus not labeled) in the other constituent datasets. Unfortunately, training the modern object detectors, including YOLO and faster R-CNN, on such partially-labelled datasets induce inferior generalization performance [113, 114].

Regardless of the type of object detectors, the small number of labeled instances in a partially-labeled dataset is one reason for such performance degradation. Another reason is the false negative training signals arising from Unlabeled Positive Instance (UPI). Inspired by [113], we later (in Section 5.4) elaborate how these UPIs can mislead training of an object detector, particularly YOLO and faster R-CNN.

In this chapter, we aim at enhancing generalization performance of an object detector, when it

is trained on a merged dataset, through augmenting it with on-the-fly (online) pseudo-labels for the UPIs. For this purpose, we propose a generic self-supervised framework for training a single detector (e.g. YOLO and faster R-CNN) while simultaneously creating on-the-fly pseudo-labels for the UPIs using the object detector itself and a proxy network. Indeed, we deploy a pre-trained proxy CNN for flagging which YOLO's predicted bounding-boxes (a.k.a. Region of Interest (ROI)) contain UPIs so that their objectness and class pseudo-labels are generated. In other words, if the proxy network classifies the ROIs as one of the pre-defined positive (i.e. interesting) objects, their pseudo-labels are created and used for training the object detector. Otherwise, they are discarded to contribute to the training phase as the proxy network classifies them as the class which is associated with all negative or uninterested objects. In Fig. 5.4, the pipeline of our proposed method is illustrated.

We use a CNN with an explicit rejection option (i.e. A-CNN) as the proxy model so that either classify a given ROI into one of the pre-defined categories of OoIs or reject it as a uninteresting (negative) object. To train this proxy, we can leverage from readily accessible datasets that contain the samples from uninteresting objects (we call them Out-of-Distribution (OOD) samples) along with the labeled samples containing OoI (a.k.a. in-distribution samples). Recently, some promising results of OOD training have been reported for developing robust object recognition classifiers [36, 66] and semantic segmentation models [5], as well as for overcoming catastrophic forgetting [54].

5.2 Related Works

To augment the training size of a partially labeled dataset that is labelled by partial-annotation policy, e.g. OpenImagev3 [45], Weakly Supervised Learning (WSL) methods [6, 17, 114, 121] have been proposed to generate pseudo-labels for the UPIs by leveraging *image-level labels*. Indeed, by annotating a dataset using this policy, we know that the images contain which objects, although no bounding box annotations are precisely localizing them. Unfortunately, these WSL methods can not simply be employed for generating pseudo-labels for the merged datasets since both image-level and instance-level annotations are missed in such datasets.

With the aim of alleviating the performance degradation in faster R-CNN on partially-labelled datasets (e.g. OpenImagev3 [45]), Wu et al. [113] propose to control the false training signals arising from UPIs –they incorrectly force the model to learn them as negative instances (i.e. false negative signals). To this end, they discarded the gradients created by the ROIs that have small or no overlap with any ground truths (to read about how discarding such ROIs can lead to a smaller drop in performance, refer to Section 5.4). Although this simple approach can remove the false negative training signals by UPIs, correcting them, instead of ignoring them, can further improve the generalization performance, particularly for the merged dataset as comprising a large amount of missing-label instances. Indeed, by identifying and correcting

the false negative signals from UPIs (through pseudo-label generations), the size of the training set is also augmented, which in turn can lead to training of an object detector with higher performance. Moreover, labeling the UPIs in a merged dataset can expose the model to the OoIs from different domains, which can be satiable to alleviate domain-shift [108].

To generate a set of pseudo-labels for UPIs in the merged dataset, Rame et al. [77] proposed to use several object detectors, each is trained separately by each individual fully-labelled dataset included in the merged one. Using these trained models, a set of pseudo-labels is generated for the UPIs in the merged dataset. Finally, another unified object detector is trained on this offline set of pseudo-labels as well as the ground truth labels. However, their approach, i.e. creation of a set of pseudo-labels using several distinct object detectors, is computationally very expensive, compared to our self-supervised method that trains a single object detector on the merged dataset while generates pseudo-labels for the UPIs. Note that for a merged dataset comprising m datasets, for example, their approach needs to train $m + 1$ object detectors (m models for pseudo-labels generation and another for the final object detector), while ours performs by training a single object detector.

5.3 Background: Modern Object Detectors

Modern object detectors based on deep neural networks are generally categorized into two types; one-stage and two-stage object detectors. The latter detectors are the model that I) proposes a set of Regions of Interests (ROIs) through select search [23] or a regional proposal network [81], II) these ROIs are classified into a set of pre-defined object categories. The R-CNN variants [22, 23, 81], i.e. region-based CNNs, are two-stage object detectors. Contrary to the two-stage object detectors, the one-stage object detectors skip the first stage (i.e. no need to obtain a set of ROIs), and, instead, they tend to classify a dense sampling of possible locations that have a large overlap with ground truth labels. YOLO [78], SSD [59], and RetinaNet [57] are among the widely-known one-stage object detectors. In this chapter, we use YOLOv3 [79] and faster R-CNN [81] as the current (at the time that the thesis is written) state-of-the-art one-stage and two-stages object detectors, respectively.

While the precision of the two-stage object detectors is greater than their one-stage counterparts, their inference time is considerably slower. Thus, the latter object detectors, e.g. YOLO, are more suitable for real-time applications,e.g. self-driving cars. In the following, we review the architecture of YOLO and that of fast R-CNN as well as their corresponding loss functions.

5.3.1 YOLO

YOLO is a CNN-based object detector that predicts the bounding-boxes, in which the objects of interest from K classes are included. YOLO predicts all the bounding boxes in a given

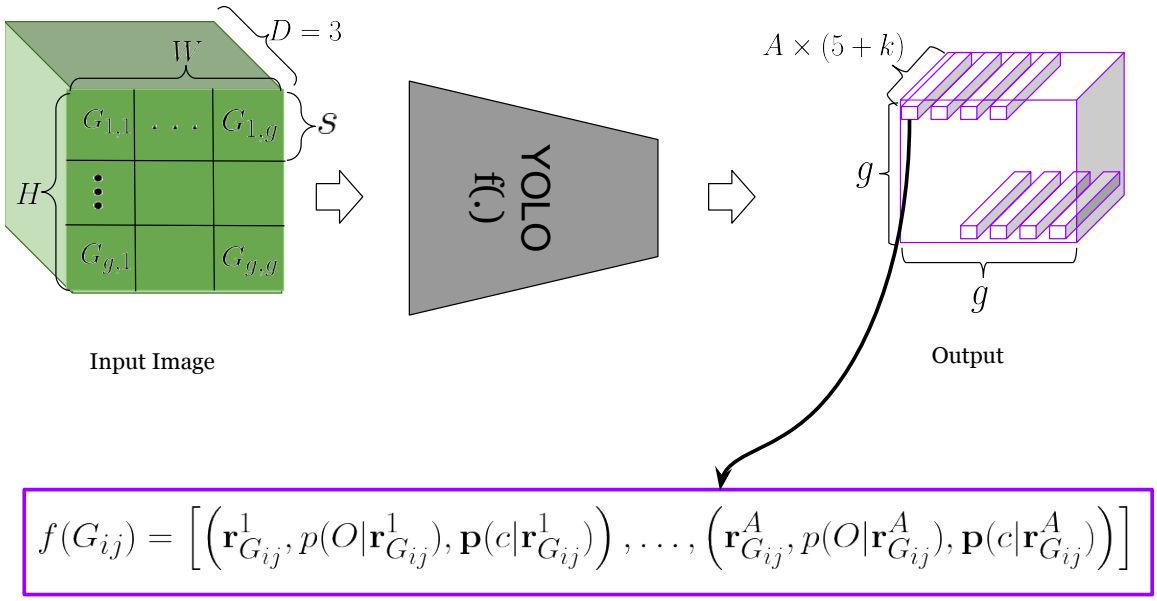


Figure 5.1: YOLO divides a given image to $g \times g$ grids, then for each grid estimates A different bounding-boxes, which each is a $5 + K$-dimensional vector, encompassing the coordinate information of the box (i.e. $\mathbf{r}^a_{G_{ij}} = \left[\hat{x}^a_{G_{ij}}, \hat{y}^a_{G_{ij}}, \hat{w}^a_{G_{ij}}, \hat{h}^a_{G_{ij}}\right]$), the objectness probability (i.e. $p(O|\mathbf{r}^a_{G_{ij}})$), and a K-dimensional vector as the probabilities over K object categorizes (i.e. $\mathbf{p}(c|\mathbf{r}^a_{G_{ij}}) \in [0,1]^K$) with $a \in \{1, \ldots A\}$.

image at once, leading to detection of objects of interest in real-time, i.e. significantly high Frame Per Second (FPS) rate.

To process the whole image at once, the input image is divided into $g \times g$ grids. Then, for each grid G_{ij} with $i, j \in \{1, \ldots, g\}$, YOLO predicts A bounding boxes, where each box is a $(5 + k)$-dimensional vector, comprising the coordinates of the box's center and its height and width w.r.t its corresponding grid (the bonding-box a belonging to G_{ij} is represented by $\mathbf{r}^a_{G_{ij}}$, $a \in \{1, \ldots, A\}$), objectness probability of the given $\mathbf{r}^a_{G_{ij}}$, i.e. $p(O|\mathbf{r}^a_{G_{ij}})$– this reflects the probability of having an object inside the given box–, and a K-dimensional vector of class probabilities, i.e. $p(c|\mathbf{r}^a_{G_{ij}})$. Therefore, the output of YOLO will be a tensor of size $[g, g, A, 5 + K]$ as demonstrated in Fig. 5.1. Moreover, for each grid G_{ij}, a set of pre-defined bounding-boxes (called anchors) with different aspect ratios and scales is considered. We represent a-th anchor of grid G_{ij} by $\mathbf{A}^a_{G_{ij}} = [w^a_{G_{ij}}, h^a_{G_{ij}}]$, where $w^a_{G_{ij}}, h^a_{G_{ij}}$ are its width and heights, respectively (later, we explain the role of anchors). YOLO then processes simultaneously each of these $g \times g$ grids for estimating the bounding-boxes that contain an object of interest.

In reality, the bounding-boxes do not have arbitrary scales and aspect ratios. For example, all bounding-boxes enclosing cars usually have very similar scales and aspect ratios. The same is true for many other objects such as pedestrians, trains, bicycles, and so on. Therefore, we can have an initial guess (in terms of anchors) for the size of A bounding-boxes. The adjustment

of the anchors to fit the ground truths is done during the training process by estimating the coordinate information of their corresponding bounding-boxes. To obtain the size of anchors, the whole training ground truths are clustered by their width and height into A clusters, then the center of each A clusters indicates the size of each A anchors.

Loss Functions of YOLO

Each training sample is denoted by $(I_i, \{\mathbf{t}_i^{*1}, \dots, \mathbf{t}_i^{*j}\})$, where I_i is the i-th training input image and $\mathbf{t}_i^{*j}$ is its j-th ground truth ($i \in \{1, \dots N\}$). Each ground truth $\mathbf{t}_i^{*j}$ consists of a bounding box coordinate information, i.e. $\mathbf{r}_i^{*j} = [x_i^{*j}, y_i^{*j}, w_i^{*j}, h_i^{*j}]$ and the category (class) of its enclosed object, i.e. $k_i^{*j} \in \{1, \dots, K\}$. The center of j-th ground truth and its corresponding height and width (w.r.t the image) are represented by $x_i^{*j}, y_i^{*j}, w_i^{*j}, h_i^{*j} \in [0, 1]$, respectively. From now on, we drop the indices from the ground truth and their estimations for simplicity reasons.

Contrary to the coordinate information of the ground truth, i.e. $\mathbf{r}^* = [x^*, y^*, w^*, h^*]$, that of estimated bounding-box by YOLO, i.e. $\mathbf{r} = \left[\hat{x}, \hat{y}, \hat{w}, \hat{h}\right]$, are relative to their corresponding gird (grid-orientation). To have the ground truths and the estimations in the same coordinate-system, the predicted bounding-box is transferred to image's coordinate system as follows:

$$b_{\hat{x}} = x_{G_{ij}} + \hat{x} \tag{5.1}$$

$$b_{\hat{y}} = y_{G_{ij}} + \hat{y} \tag{5.2}$$

$$b_{\hat{w}}^a = w_{G_{ij}}^a \exp^{(\hat{w})} \tag{5.3}$$

$$b_{\hat{h}}^a = h_{G_{ij}}^a \exp^{(\hat{h})}, \tag{5.4}$$

where $x_{G_{ij}}, y_{G_{ij}}$ are the coordinate of the top-left corner of grid G_{ij} w.r.t the image, and $w_{G_{ij}}^a, h_{G_{ij}}^a$ are the width and height of a-th anchor of the given grid.

For each grid G_{ij}, $i, j \in \{1, \dots, g\}$, we compute the class and coordinate loss functions only if the Intersection over Union (IoU) between a ground truth, e.g. $\mathbf{r}^*$, and at least one of the grid's anchors $\mathbf{A}_{G_{ij}}^a$ is larger than a pre-defined threshold τ, otherwise its class and coordinate losses are zero out (ignored). *Note if a grid has several anchors that have large IoU ($>\tau$) with a ground truth, then the class and coordinate losses are computed for the anchor with the largest IoU, and the rests are ignored.*

For a given G_{ij}, let $a' = \text{argmax}_{a=1}^{A} \left(\text{IoU}(\mathbf{r}^*, \mathbf{A}_{G_{ij}}^a)\right)$, its class loss computes the difference between the estimated class probabilities, i.e. $\mathbf{p}(c|\mathbf{r}_{G_{ij}}^{a'})$ and the true class k^*, which is encoded by $\mathbf{p}^*(c)$; that is a one-hot K-dimensional vector with only one at its k^*-th element

$(\mathbf{p}^*(c=k^*)=1)$ [1]:

$$\mathcal{L}_{cls}(\mathbf{p}^*(c),\mathbf{p}(c|\mathbf{r}^{a'}_{G_{ij}})|G_{ij}) = \begin{cases} \log p(c=k^*|\mathbf{r}^{a'}_{G_{ij}}) & \text{if} \quad \max_a \left(\text{IoU}(\mathbf{r}^*,\mathbf{A}^a_{G_{ij}})\right) \geq \tau \\ 0 & \text{Otherwise.} \end{cases} \tag{5.5}$$

$$\mathcal{L}_{coor}\left([x^*,y^*,w^*,h^*]\,,\left[b_{\hat{x}},b_{\hat{y}},b^{a'}_{\hat{w}},b^{a'}_{\hat{h}}\right]|G_{ij}\right) = \begin{cases} (x^*-b_{\hat{x}})^2+(y^*-b_{\hat{y}})^2 & \text{if} \quad \max_a \left(\text{IoU}(\mathbf{r}^*,\mathbf{A}^a_{G_{ij}})\right) > \tau \\ +(w^*-b^{a'}_{\hat{w}})^2+(h^*-b^{a'}_{\hat{h}})^2 & \\ 0 & \text{Otherwise.} \end{cases} \tag{5.6}$$

Contrary to coordinate and class losses, we compute the object loss for all the anchors of all the grids– regardless of the IoU of each anchor of grid G_{ij} with a ground truth. The object loss is computed by binary cross-entropy, measuring the loss on the estimated objectness probability. To label each anchor by its true object label, the anchor a' of a grid G_{ij} that has the highest IoU and is larger than τ ($a' = \text{argmax}_a \text{IoU}(\mathbf{r}^*,\mathbf{A}^a_{G_{ij}}) > \tau$) is labeled as 1 (meaning contain an object $t^{a'}_{G_{ij}}=1$), and the remaining anchors of this grid are labeled as zero ($t^a_{G_{ij}}=0$, where $a \neq a'$). There is another scenario for labeling the anchors; if an anchor has a large ($>\tau$) IoU with a ground truth, then its true object label is $t^a_{G_{ij}}=1$, whether its IoU is the highest or not, otherwise (if its IoU$<\tau$) its true label is $t^a_{G_{ij}}=0$. Using the latter labeling scenario, each anchor of a grid $\mathbf{A}^a_{G_{ij}}$ is labeled as follows:

$$t^a_{G_{ij}} = \begin{cases} 0 & \text{IoU}(\mathbf{r}^*,\mathbf{A}^a_{G_{ij}}) < \tau \\ 1 & \text{IoU}(\mathbf{r}^*,\mathbf{A}^a_{G_{ij}}) \geq \tau \end{cases} \tag{5.7}$$

The following object loss for the a-th anchor of grid G_{ij} is computed as follows:

$$\mathcal{L}_{obj}(t^a_{G_{ij}},p(O|\mathbf{r}^a_{G_{ij}})) = t^a_{G_{ij}}\log p(O|\mathbf{r}^a_{G_{ij}}) + (1-t^a_{G_{ij}})\log(1-p(O|\mathbf{r}^a_{G_{ij}})). \tag{5.8}$$

It should be re-emphasized that **the object loss is computed for all the anchors belonging to all the grids, whether they contain a ground truth or not**, while the coordinate and class losses are only computed for the grids that have a high IoU overlap and these losses are zero for the the girds with no such high IoU overlap. Finally, a weighted sum of the above loss functions are computed to define the total loss of YOLO. The weights are set so that the

[1] Note that, instead cross-entropy for the class predictions, the authors [79] originally used binary cross-entropy loss for each of K classes.

contributions of the losses are balanced.

$$\begin{aligned}\mathbf{L} =& \lambda_{cls} \sum_{G_{ij}} \mathcal{L}_{cls} \left(\mathbf{p}^*(c), \mathbf{p}(c|\mathbf{r}^{a'}_{G_{ij}}) | G_{ij} \right) + \\ & \lambda_{coor} \sum_{G_{ij}} \mathcal{L}_{coor} \left([x, y, w, h], \left[b_{\hat{x}}, b_{\hat{y}}, b^{a'}_{\hat{w}}, b^{a'}_{\hat{h}} \right] | G_{ij} \right) + \\ & \lambda_{obj} \sum_{G_{ij}} \sum_{a=1}^{A} \mathcal{L}_{obj} \left(t^a_{G_{ij}}, p(O|\mathbf{r}^a_{G_{ij}}) \right)\end{aligned} \tag{5.9}$$

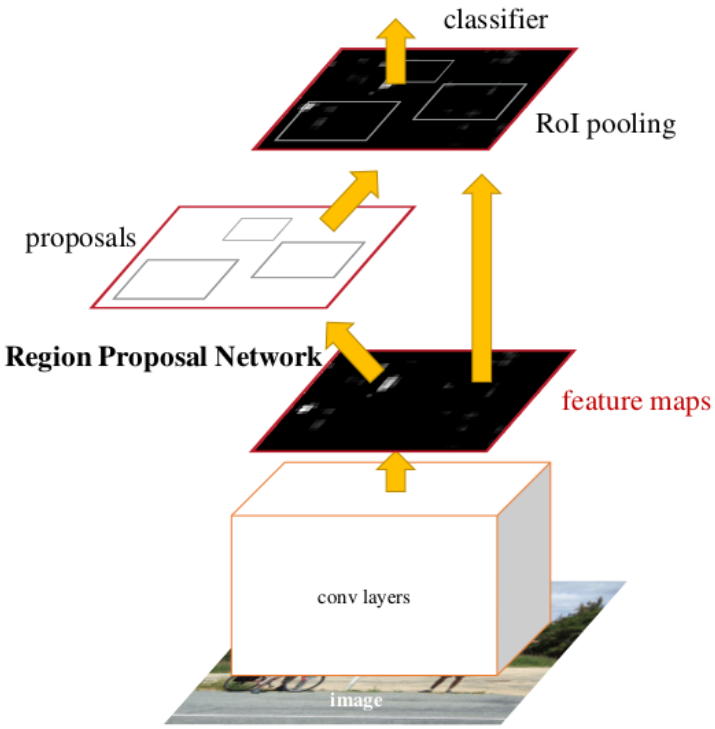


Figure 5.2: Faster R-CNN comprises two sibling sub-network, one for bounding-box prediction and another for predicting objectness score

5.3.2 Faster R-CNN

As the faster R-CNN [81] is the latest variant of the region-based CNNs, we briefly review it. Faster R-CNN relies on building a set of proposals (i.e. ROIs) using a Region Proposal Network (RPN), then classifying these ROIs into K classes. Taking a convolution feature map as an input, RPN slides a $n \times n$ (e.g. $n = 3$) window on the feature map, where for each window, a *reg* sub-network estimates the coordinate information of the A bounding boxes, which in total forms a $4A$ vector, and a *cls* sub-network predicts an objectness score (the probability of object or not object) for each of these bounding-box, which is a $2A$ vector. For the sliding-window $n \times n$,

there are A pre-defined reference boxes (called anchors) with different aspect-ratio and scales, which each anchor is centered at the center of its corresponding sliding-window. Therefore, the *reg* sub-network estimates the coordinates of the bounding-boxes through adjusting their corresponding anchors.

Loss Functions of Faster R-CNN

Unlike YOLO which defines the anchors for each grid, the faster R-CNN defines the A anchors for *each pixel* of the feature map with size $H' \times W'$ (represented by $F_{ij} \;\; i \in \{1, \dots H'\} \;\; j \in \{1, \dots, W'\}$) by sliding a $n \times n$ window on it. Thus, the total number of anchors for the feature map is $AW'H'$, while the number of anchors in YOLO is Ag^2 (g the number of grids), $g << W', H'$. We represent the anchor of each feature map's pixel by $\mathbf{A}^a_{F_{ij}}$. Each anchor of the sliding window is labeled either as **positive**, **negative** or **none of them** [2]:

$$t^a_{F_{ij}} = \begin{cases} 0 & \text{IoU}(\mathbf{r}^*, \mathbf{A}^a_{F_{ij}}) < \theta_1 \\ 1 & \text{IoU}(\mathbf{r}^*, \mathbf{A}^a_{F_{ij}}) > \theta_2 \\ \text{None} & \text{Otherwise,} \end{cases} \tag{5.10}$$

where θ_1, θ_2 are the hyper-parameters, they are set $\theta_1 = 0.3$ and $\theta_2 = 0.7$ in the faster R-CNN's paper. We tag the a-th anchor of a sliding window as positive ($t^a_{F_{ij}} = 1$) if it has a large IoU overlap (greater than a threshold, e.g. 0.7) with a ground truth bounding-box. Alternatively, this bounding box is labeled as negative (i.e. $t^a_{F_{ij}} = 0$) if its IoU overlap with a ground truth is lower than a threshold, e.g. $\theta_1 = 0.3$. The anchors neither positive nor negative are ignored to contribute in the training of the RPN.

For simplicity of the notation, let the total loss function of PRN be defined as $\mathbf{L} = \lambda_{obj}\mathcal{L}_{obj} + \lambda_{reg}\mathcal{L}_{reg}$, with $\lambda_{reg}, \lambda_{obj}$ as the balancing factors. The components of this total loss are defined as follows:

$$\mathcal{L}_{obj} = \sum_{F_{ij}} \sum_{a=1}^{A} t^a_{F_{ij}} \log p(O|\mathbf{r}^a_{F_{ij}}) + (1 - t^a_{F_{ij}}) \log(1 - p(O|\mathbf{r}^a_{F_{ij}})) \tag{5.11}$$

Note that the object loss **is not** computed for the anchors labelled as None.

$$\mathcal{L}_{coor} = \sum_{F_{ij}} \sum_{a=1}^{A} t^a_{F_{ij}} \cdot \left((x^* - \hat{x}^a_{F_{ij}})^2 + (y^* - \hat{y}^a_{F_{ij}})^2 + (w^* - \hat{w}^a_{F_{ij}})^2 + (h^* - \hat{h}^a_{F_{ij}})^2 \right). \tag{5.12}$$

Note that the negative anchors ($t_i = 0$) are only contributing to training through their object loss, i.e. their coordinate losses are zero-out (Eq. 5.12).

[2] In YOLO, the objectness label of the anchors is either zero or one.

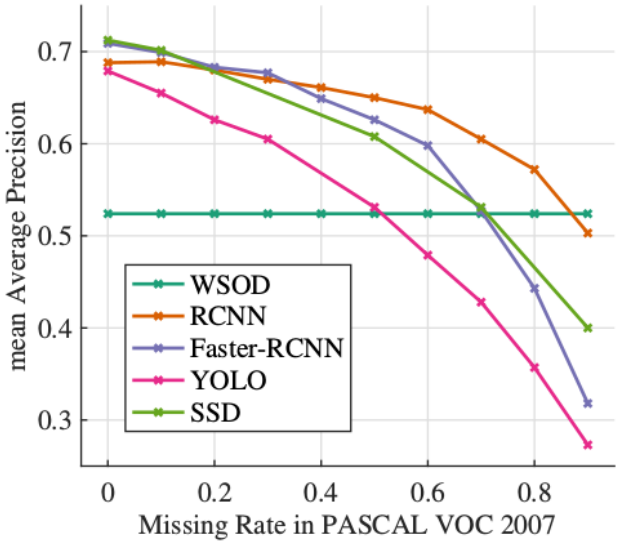


Figure 5.3: The influence of missing-label instances on the performance of several modern object detectors. The performance of the object detectors as a function of missing rate in PASCAL VOC2007. The higher missing rate, the higher drop in performance [114].

5.4 The Impact of Missing-label Instances on Performance

As stated earlier, the missing-label instances (called UPIs) can cause false negative signals during training, resulting in a drop in performance (i.e. mean average of precision) of object detectors, particularly YOLO and faster R-CNN.

Recall that YOLO computes the object loss in Eq. 5.8 for *all the anchors of all the grids, whether they contain any ground truths or not*. In other words, if an anchor has no high IoU overlap with a ground truth (IoU$\geq \tau$), its true label is zero, and is one, otherwise. This can cause the false negative signals by UPIs. More specifically, during the training of YOLO, the detector may be able to localize correctly a UPI[3], i.e. estimating the objectness score $p(O) \approx 1$, but since it has no associated ground truth label, the grid encompassing this UPI is labelled as negative, with true object label $t = 0$. This incorrectly forces the network to learn it as a negative or uninteresting object even thought the network can correctly localize such a UPI as a positive object ($p(O) \approx 1$). Ultimately such a false negative signal from a UPI can confuse the network as the LPIs (Labelled Positive Instances) of the same object force the network to learn it as an object of interest while the UPIs from this object incorrectly encourage the network to learn it as an uninteresting objects. Finally, such false negative signals can cause a drop in the performance of YOLO. *Note that the class and coordinate losses are not computed for the UPIs*

[3]The considered context is datasets consist of both labeled instances and missing-label instances for the objects of interest. Thus, in this context, the trained YOLO on the labeled OoIs may localize correctly the UPIs of these OoIs.

since their IoU conditions are not satisfied. Consequently, while the UPIs cannot contribute to the training through their class and coordinate losses, they may adversely contribute to the training through their object loss.

Similarly, in the faster R-CNN, a UPI can penalize the network incorrectly if its IoU is lower than a defined threshold $\theta_1 = 0.3$. More precisely, the true objectness label of an UPI (i.e. t) will be set to zero when it has an IoU overlap (with a ground truth) smaller than 0.3 (in Eq.5.10). Then, although the RPN may detect the UPI as a positive instance (i.e. $P(O) \approx 1$), the objectness loss defined in Eq. 5.11 is penalizing the network incorrectly by forcing it to learn such a unlabeled positive instance as a negative instance. Thus, these false negative signals can intervene with the true positive signals from LPIs, leading to a drop in performance of the faster R-CNN. However, interestingly, if the UPI has no high IoU overlap ($> \theta_2 = 0.7$) nor small IoU ($< \theta_1 = 0.3$), then such a UPI will be ignored (according to Eq. 5.10) and not contribute to the training. Compared to YOLO, this simple condition in the faster R-CNN may reduce the probability of false negative signals by UPIs as it discard to compute the object loss for the UPIs with $\theta_1 \leq$IoU$\leq \theta_2$. In other words, labeling the anchors by objectness score in the faster R-CNN can result in ignoring some of the UPIs (those that have $0.3 \leq$ IoU < 0.7) while in the YOLO no UPIs are discarded as they have IoU $< \tau = 0.7$. This main difference between YOLO and fast R-CNN in objectness-labeling may cause a smaller drop in the faster R-CNN than that of YOLO as can be seen in Fig. 5.3.

5.5 Proposed Method

We introduce our framework to handle missing-label instances when the underlying object detector is YOLO. However it can also be adapted for the faster R-CNN. During the training of YOLO, it is likely that some existing UPIs are localized correctly. However due to the lack of a ground truth label for them, they may adversely contribute to the training of YOLO, inducing a drop in performance. We propose to generate a pseudo-label for them in algorithm 2. The estimated ROIs by YOLO at training epoch e are evaluated to check whether they actually contain a positive unlabeled object or not. To achieve this, our framework incorporates a pre-trained proxy A-CNN, denoted by $h(\cdot)$, into the training process of YOLO. Indeed, the proxy network maps each current estimated ROI $\mathbf{r} \in \mathcal{R}^e$ of a given image I (denoted by $I^{\mathbf{r}}$), into a $K+1$-dim vector of probabilities over $K+1$ classes, i.e. $h(I^{\mathbf{r}}) \in [0,1]^{K+1}$, where $1, \ldots, K$ represent K classes (for interested objects) and the $K+1$-th (extra) class is for any uninterested (negative) objects. Note that to enable h to process these ROIs with different aspect ratios, we exploit a Spatial Pyramid Pooling (SSP) layer [31] after the proxy's last convolution layer.

To train this proxy, we can leverage from the readily accessible datasets that contain the samples of not-interesting-objects (we call them Out-of-Distribution (OOD) samples) along

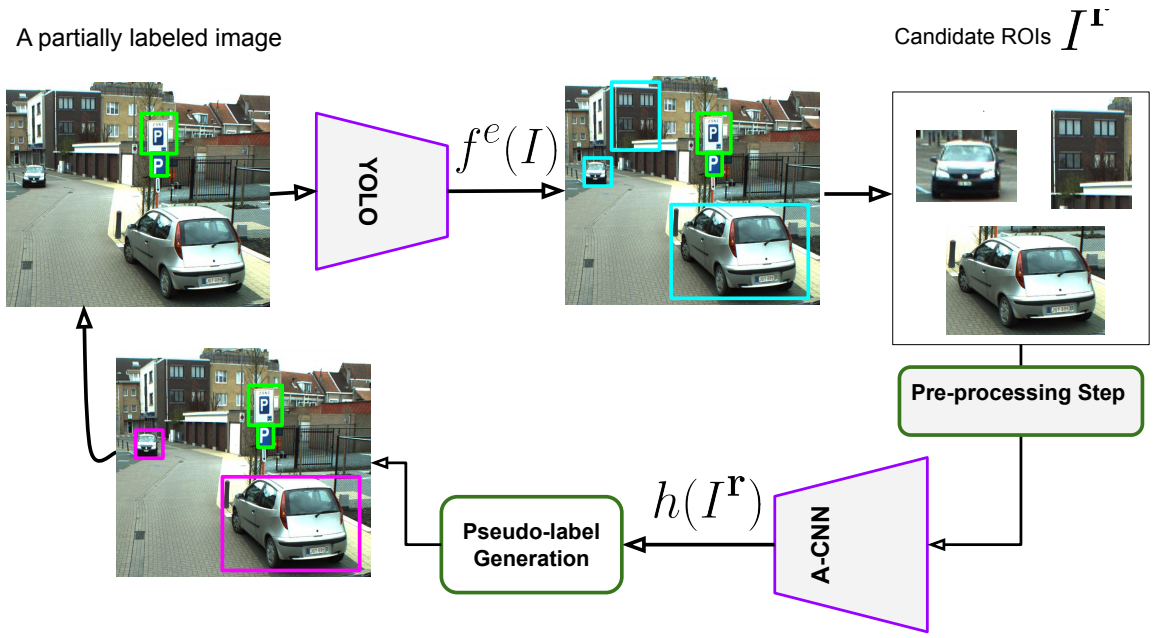


Figure 5.4: Schematic explanation of our proposal for generation of pseudo-labels for a partially labelled dataset, e.g. a merged dataset. For a given input I with some UPIs (Unlabeled Positive Instance), the bounding boxes (ROIs) estimated by YOLO at training epoch e (i.e. $f^e(I)$) are extracted for a pre-processing step, i.e. to prepare them for the proxy network. Using the proxy network's estimations for the given ROIs, we create pseudo-labels for UPIs allowing YOLO to be trained jointly with the pseudo-labels and the ground truths of the given input.

with the labeled samples from objects of interest (a.k.a. in-distribution samples).

Using the coordinate information $\mathbf{r}$ provided by YOLO (in line 2 of algorithm 2), an estimated ROI is extracted from an image I, denoted by $I^{\mathbf{r}}$. To avoid re-labeling the ROIs containing a ground truth, only those that have a small or no overlap with any of the ground truth annotations (line 6–8 of algorithm 2) are processed. In line 9 of the algorithm, before feeding these extracted ROIs to the proxy network, we pre-process them using the following procedure.

Pre-processing Step To allow $h(\cdot)$ processes the ROIs in a mini-batch style, we perform this pre-processing step (in line 9 of Algorithm2)[4]. Even thought the proxy model h is designed to accommodate the samples with varying aspect ratios using SSP [31], training and testing of h in a mini-batch style for the input sample with different sizes and aspect ratios, is challenging since Python libraries such as Pytorch do not allow the input samples with various sizes to be stacked in one batch. To address this issue, we can think of padding the inputs with the largest aspect ratio size in the batch, but this can destroy the information of the smallest inputs (since

[4]It is not our contribution, in fact the practitioners usually use this step to process a mini-batch of varying aspect ratios.

Algorithm 2 Pseudo-label Generation Algorithm

Input: $f^e(\cdot)$ object detector at training epoch e; $h(\cdot)$ pre-trained proxy network; I given input image with its associated ground truth bounding-boxes $\mathcal{R}^*$ (i.e. their coordinate information) ; θ_1, θ_2 and, β as hyper-parameters.

Output: S^e, pseudo-labels of I at time e

$S^e = \emptyset$

$\Big[[\mathbf{r}_1^e,\ p(O|\mathbf{r}_1^e),\ \mathbf{p}(c|\mathbf{r}_1^e)], \ldots, [\mathbf{r}_{Ag^2}^e,\ p(O|\mathbf{r}_{Ag^2}^e),\ \mathbf{p}(c|\mathbf{r}_{Ag^2}^e)]\Big] = f^e(I)$

$\mathcal{R}^e = \{\mathbf{r}_1^e, \ldots, \mathbf{r}_{Ag^2}^e\}$

$\mathcal{P}^e = \{\ \mathbf{p}(c|\mathbf{r}_1^e), \ldots\ \mathbf{p}(c|\mathbf{r}_{Ag^2}^e)\}$

$\mathcal{B} = \{\emptyset\}$

for $\mathbf{r} \in \mathcal{R}^e$

  if IoU($\mathbf{r}$, $\mathcal{R}^*$) $\leq \theta_1$ ▷ To skip generation of pseudo-labels for the estimated ROIs with a large IoU overlap with a ground truth from $\mathcal{R}^*$.

    $\mathcal{B} \longleftarrow \mathcal{B} \cup \{\mathbf{r}\}$

$\mathcal{B} \longleftarrow$ pre-processing step ($\mathcal{B}$) ▷ To cluster the ROIs with similar size, then zero-pad them to have identical size in a mini-batch for h.

for $\mathbf{r} \in \mathcal{B}$

  $\{I^{\mathbf{r}_1}, \ldots, I^{\mathbf{r}_m}\}$ = patch-drop($I^{\mathbf{r}}$) ▷ Create m copies of ROI extracted by $\mathbf{r}$, i.e. $I^{\mathbf{r}}$.

  $\bar{h}(I^{\mathbf{r}}) = \frac{1}{m+1}\left(h(I^{\mathbf{r}}) + \sum_{i=1}^{m} h(I^{\mathbf{r}_i})\right)$

  if $\arg\max \bar{h}(I^{\mathbf{r}}) \neq K+1$ & $\max_{\{1,\ldots,K\}} \bar{h}(I^{\mathbf{r}}) \geq \theta_2$

    $\tilde{\mathbf{p}}(c|\mathbf{r}) = \beta \cdot \mathbf{p}(c|\mathbf{r}) + (1-\beta) \cdot \bar{h}(I^{\mathbf{r}})$ ▷ $\mathbf{p}(c|\mathbf{r}) \in \mathcal{P}^e$is the class estimation for the given $\mathbf{r}$ by the YOLO at training epoch e.

    $\tilde{p}(O|\mathbf{r}) = \max_{\{1,\ldots,K\}} \bar{h}(I^{\mathbf{r}})$

    $S^e \longleftarrow S^e \cup [\mathbf{r},\ \tilde{\mathbf{p}}(c|\mathbf{r}),\ \tilde{p}(O|\mathbf{r})]$

these images can be dominated by an extremely large pad of zeros). To tackle this, in each training epoch of h, we load the samples with similar (close) aspect ratios in one batch and pad them with zeros, if needed, to achieve a batch of samples with equal aspect ratios. To implement this, all the training samples are clustered by their width and height using k-means. Then, the centers of these clusters serve as the pre-defined aspect ratios to load the batches accordingly. Similarly, *at the test time of* h during the training of the object detector, all the input instances to h (i.e. ROIs) should first be batched according to the achieved mean of clusters, then the input samples in each batch are zero-padded, if needed, in order to create a mini-batch with equal size.

Pseudo-label Generation Inspired by [92], we use patch-drop at the test time of h in order to estimate the true class of a given ROI more accurately. In the patch-drop, we divide a given ROI into $s \times s$ patches, then randomly drop one of them to create a new version of the ROI. In our experiments, we apply patch-drop with $s = 3$ for $m = 2$ times on the given ROI to create m versions of $I\mathbf{r}$, i.e. $\{I^{\mathbf{r}_1}, \dots I^{\mathbf{r}_m}\}$ (line 11 in Alg 2). We then feed them as well as the original ROI to the proxy network for estimating the probability over $K+1$ classes as follows:

$$\bar{h}(I^{\mathbf{r}}) = \frac{1}{m+1}\left(h(I^{\mathbf{r}}) + \sum_{i=1}^{m} h(I^{\mathbf{r}_i})\right). \tag{5.13}$$

This trick leads to more calibrated confidence predictions, especially for some hard-to-classify ROIs, as the proxy network h predicts each version of $I^{\mathbf{r}}$ differently (to different classes). This indeed allow us to reduce the number of false positive instances, leading to the creation of more accurate pseudo-labels. Moreover, using a threshold on the predictive confidence $\bar{h}(.)$ (i.e. θ_2 in the algorithm), the ROIs with low confidence prediction are dropped to continue the pseudo-label generation procedure. If the proxy network **confidently** classifies the given ROI into one of K classes, its pseudo class probability $\tilde{\mathbf{p}}(cls|\mathbf{r})$ is computed as follows:

$$\tilde{\mathbf{p}}(c|\mathbf{r}) = \beta \cdot \mathbf{p}(c|\mathbf{r}) + (1-\beta) \cdot \bar{h}(I^{\mathbf{r}}), \tag{5.14}$$

where $\mathbf{p}(c|\mathbf{r}), \bar{h}(I^{\mathbf{r}}) \in [0,1]^K$ are respectively the estimated class probabilities by YOLO at training epoch e and the proxy network h for the given ROI $I^{\mathbf{r}}$. Finally, we set the objectness score, i.e. probability of the presence of an object, for the given ROI $\mathbf{r}$ as $\tilde{p}(O|\mathbf{r}) = \max_{k=1}^{K} \bar{h}(I^{\mathbf{r}})$.

5.5.1 Modified Loss Functions

We use Kullback-Leibler divergence (KL-div.) for computing the loss between the generated class pseudo-label, i.e. $\tilde{\mathbf{p}}(c|\mathbf{r}) \in [0,1]^K$, and the class label estimation by YOLO, i.e. $\mathbf{p}(c|\mathbf{r}) \in [0,1]^K$,

$$\mathcal{L}_{\text{Pseudo-cls}}(\tilde{\mathbf{p}}(c|\mathbf{r}), \mathbf{p}(c|\mathbf{r})) = KL\left(\tilde{\mathbf{p}}(c|\mathbf{r}) \;||\; \mathbf{p}(c|\mathbf{r})\right). \tag{5.15}$$

The object loss for the object pseudo-label is computed as follows:

$$\mathcal{L}_{\text{Pseudo-obj}}(\tilde{p}(O|\mathbf{r}), p(O)) = \tilde{p}(O|\mathbf{r}) \log p(O|\mathbf{r}) + (1 - \tilde{p}(O|\mathbf{r})) \log\left(1 - p(O|\mathbf{r})\right). \tag{5.16}$$

We drive a new loss by adding the above two new losses (for the pseudo-labels (Eq. 5.16 and Eq. 5.15)) to the total loss by YOLO in Eq. 5.9. Finally, this new loss function is used to update the parameters of YOLO.

5.6 Experimentation

To simulate a merged dataset, we adapted two datasets with two disjoint sets of classes from VOC2007 (Visual Object Classes) with S_A={cat, cow, dog, horse, train, sheep} and VOC2012

with S_B={car, motorcycle, bicycle, aeroplane, bus, person}. One dataset, called D_{S_A}, gathers the samples from VOC2007 that contain one of the objects of interest in S_A (dropping the annotations from another set of classes S_B, if there are any in D_{S_A}). Similarly, another dataset D_{S_B} is made of the images from VOC2012 containing one of objects in S_B. Then, these two datasets are merged to produce dataset $D'_S = D_{S_A} \cup D_{S_B}$ with total classes of $S = S_A \cup S_B$. In addition, a fully labeled dataset D_S from the union of VOC2007 and VOC2012 is formed, where all the instances belonging to S are fully annotated. The missing-label rate of D'_S (the merged dataset) with respect to D_S is 48%.

As the proxy network (i.e. A-CNN), we use Resnet20 [32] structure, then adding an extra class to its output (corresponding to the rejection option) as well as placing a SPP (Spatial Pyramid Pooling) layer after its last convolution layer to enable it to process the inputs with various aspect-ratio sizes. To train this A-Resnet (A-CNN), we utilize the training set of MSCOCO [58] by extracting all the ground truth bounding boxes that comprise an object from one of the classes in $S = S_A \cup S_B$ (i.e. object of interest), and all other ground truth bounding boxes that do *not* belong to S are used as OOD samples (labeled as class $K + 1$ to train A-Resnet). The hyper-parameters of our algorithm are set to $\beta = 0$ (in Eq. 5.14), $\theta_1 = 0.5$ (to remove RoIs having a large overlap with ground truth, line 7 of our algorithm), and $\theta_2 = 0.8$ (the threshold on the prediction confidence of the proxy network for given RoIs).

In Fig. 5.5, we exhibit the pseudo-labels generated by our proposed method for some UPIs in D'_S. In Table 5.1, we compare mAP@0.5 of three YOLOs that have the same structure but trained differently on the fully-labelled test set of VOC2007. The YOLO (called baseline) is trained on the partially labelled dataset D'_S, another one is trained on the augmented D'_S by our pseudo-labels (called Ours), and the last one is trained with the fully labeled dataset D_S (called upper-bound). As it can be seen, training a YOLO on the partially-labelled dataset D'_S (with 48% missing rate) leads to a $\approx$17% drop in mAP@0.5 (on average), compared to the upper-bound YOLO. Augmenting D'_S with our generated pseudo-labels for some of the UPIs increases the performance (mAP@0.5) of YOLO by 4% (on average).

5.7 Conclusion

With the goal of training an integrated object detector with the ability to detect a wide array of OoIs, one can merge several datasets from similar contexts but with different sets of OoIs. While merging multiple datasets to train an integrated object detector has attractive potentials for reducing computational cost, many missing-label instances (Unlabeled Positive Instances) in the merged dataset cause a performance degradation of the object detector trained on it. To address this issue, we propose a general training framework for simultaneously training an object detector on the merged dataset while generating on-the-fly pseudo-labels for UPIs. Using a pre-trained proxy neural network, we generate a pseudo label for each

OoI	mAP@0.5 (%)			
	Baseline	Ours	Improvement	||Upper-bound
Cat	74.79	**77.2**	2.43	82.04
Cow	48.27	**55.6**	7.33	69.70
Dog	52.71	**62.0**	9.28	78.70
Horse	18.68	**23.7**	5.04	82.51
Train	**58.36**	57.7	-0.66	79.18
Sheep	57.77	**65.1**	7.33	72.45
Car	77.67	**78.3**	0.67	83.87
Motorbike	68.23	**72.4**	4.18	79.82
Bicycle	69.98	**72.1**	2.12	79.00
Aeroplane	59.96	**62.6**	2.68	71.29
Bus	65.26	**71.2**	6.00	78.83
Person	71.32	**72.0**	0.68	78.30
Avg	60.25	**64.2**	3.95	77.97

Table 5.1: Performance (i.e. mAP) of different YOLOs on the test set of VOC2007 with fully labeled instances from classes $\mathcal{S} = \mathcal{S}_A \cup \mathcal{S}_B$. Baseline is the trained YOLO on the merged dataset (VOC2007+VOC2012) with missing-label instances (D'_S), ours is YOLO trained on the merged dataset D'_S that is augmented with our generated pseudo-labels, and the upper-bound is the YOLO trained on voc2007+voc2012 with **fully** annotated instances (D_S).

estimated RoI if the proxy network confidently classifies it as one of its pre-defined classes of interest. Otherwise, we exclude it from contributing to the training of the object detector. By a simulated merged dataset using VOC2007 and VOC2012, we empirically show that YOLO trained by our framework achieves higher generalization performance than the YOLO trained on the original merged dataset. This achievement is the result of the augmentation of the merged dataset with the pseudo-labels for UPIs generated by our method.

Figure 5.5: The bounding boxes in violet are our pseudo-labels generated during training of YOLO while the green bounding boxes are ground truth labels in dataset D'_S (i.e. the merged dataset from VOC2007 and VOC2012 with disjoint sets of classes.)

Conclusion

Reliability and safety play a crucial role in successfully deploying machine learning models in real-world applications such as self-driving cars or automatic medial diagnostic systems. Besides generalization performance, it is required to develop the ML models that can reliably respond in rare (e.g. hard-to-classify samples) or hostile (e.g. adversarial examples) situations. Despite this pressing demand, the vanilla models (e.g. vanilla CNN) are generally unreliable and unsafe in the presence of adversarial examples and OOD samples as they confidently (mis)classify them, leading to making wrong decisions. In this thesis, we propose several original approaches to enhance the reliability of CNNs through identifying and rejecting (to classify) these odd samples. Indeed, detection of OOD samples and adversaries allows human intervention, leading to making right and controlled decisions in such potentially dangerous situations. In the following, we summarize the research findings regarding to each objective of this thesis.

On the role of diversity in ensembles for boosting reliability

In this objective, the main goal is to propose a diverse ensemble of CNNs to *predict the adversarial examples with high uncertainty while predicting the clean samples with certainty, which in turn it allows to distinguish (then reject) the adversarial examples and some hard-to-classify clean samples.* This ultimately can increase the dependability of the predictions provided by CNNs as the risk of the adversaries and some hard-to-classify samples are diminished. Our ensemble-based framework is among the early methods for detecting adversarial examples, when it was initially believed that detection can not mitigate the robustness issue of CNN. Later on, it has been shown that in fact detection and correct-classification of adversaries are equally hard to tackle [9], and the detection of adversaries can also robustify CNNs.

Our ensemble of specialists approach attempts to calibrate the predictive confidences produced by CNNs through encouraging diversity in the ensemble. Being the first ensemble-based framework to address adversaries issue allows our proposed framework to be recognized in the community for its originality and influential contributions, and finally it won "the best paper award" in Canadian Conference on Artificial Intelligence (2020). Indeed, we initiated a new direction toward adversarial robustness through ensemble of diverse classifiers, whereas the

ensemble methods were conventionally used only for generalization performance enhancement. We restate the research questions associate with this objective to provide our research findings to address these questions:

Q1. *Does diversity in the ensemble of CNNs encourage calibrated predictions?*
Answer: We show diversity in the ensemble of CNNs can implicitly influence on producing more calibrated prediction, so as to the members of the ensemble can not agree on a class for a given odd input sample, e.g. adversarial examples, inducing a prediction with high entropy (high uncertainty).

Q2. *For calibration purposes, how can we promote effective diversity in the ensemble in a cost-effective way?*
Answer: By proposing an ensemble of specialist CNNs, we successfully create a diverse ensemble, where each member, i.e. specialist CNN, is an expert on a distinct subset of the classes of a given classification task. Then, we propose an efficient voting mechanism to merge the predictions produced by the members. Finally, we prove that the prediction by our proposed ensemble in disagreement situations is upper-bounded by $0.5 + \epsilon'$. This important corollary allows us to set **a fixed global threshold** on the calibrated predictive confidence for the purposes of detection. Usually the threshold should be tuned for each type of adversarial examples separately. By having a fixed global threshold, we eliminate the need for locally tuning the threshold.

Q3. *Using an ensemble of diverse CNNs (without adversarial learning), can we detect the black-box adversarial examples based on their low predictive confidence, resulting in minimization of their risk, while maintaining the generalization performance on the clean samples?*
Answer: Through extensive experimental assessment, we demonstrate the ability of our diverse ensemble for producing calibrated predictions for clean and adversarial instances. This allows detection of a wide range of black-box adversarial examples while maintaining the generalization performance on the clean samples.

Q4. *Is such an ensemble of diverse CNNs robust to the white-box adversarial examples?*
Answer: We analytically show that generating white-box adversarial examples using our ensemble can be hard since the specialist CNNs have different fooling directions for an input sample, causing the ensemble ends up to a disagreement situation. We also empirically show that the rate of successful white-box adversarial examples by our ensemble can be lower than a single vanilla CNN, affirming its resistant to white-box adversarial examples.

Consequently, by our research finding, we confirm the positive influence of diversity on calibrating predictive confidence of CNNs. The list of peer-reviewed papers related to this objective is as follows:

Workshop **M. Abbasi**, C. Gagné, "Robustness to Adversarial Examples through an Ensemble of Specialists", International Conference on Learning Representations-Workshop, 2017, Toulon, France.

Conference **M. Abbasi**, A. rajabi, C. Gagné, Rakesh B. Bobba, "Toward adversarial robustness by diversity in an ensemble of specialized deep neural networks", Canadian Conference on Artificial Intelligence, 2020, Ottawa, Canada [best paper award].

Future Work

As a future work, our approach can be improved in the way of forming the subsets of classes. Training of our ensemble can be challenging for a very large-scale classification problems with hundred or thousand classes (e.g. ImageNet) since one should train at most $2K$ specialists for a K-class classification problem. Therefore, the number of the subsets should preferably smaller. Moreover, the subsets should have a small overlap with each other as this causes the likelihood of having a disagreement on a fooling class will be lower or even zero (to read more about this refer to Sec. 2.4). It is important to note that training time of a specialist CNN on a a small subset of the K classes is drastically smaller than that of generalist CNN trained on all the K classes, especially for a very large K (like ImageNet). Therefore, for the time complexity reasons, another constraint is to keep the size of expertise domains small.

On differentiating OOD sets

Recently, OOD learning has been shown as a technique for robustifying CNNs to (unseen) OOD samples. Even though this promising technique has been widely used by the researchers, a pivotal question in OOD learning has not been concretely addressed: *how to differentiate OOD sets with the goal of selecting the most proper one for OOD learning?*. Thus, we are first to provide some rigorous answer to this crucial question. The following research findings associated with each research question of the second objective are:

Q1. *Considering the availability of an enormous number of OOD sets, are they equal for training a well-generalized and robust CNN to OOD samples?*
Answer: In this research direction, we argue that all OOD sets are not equally effective for training a well-generalized CNN that is also robust to a large spectrum of OOD samples. To address this pivotal question, we provide a criterion based on generalization errors of Augmented-CNN on in-distribution and unseen OOD sets. We define an OOD set as a proper one if training of an A-CNN on it induces the lowest generalization errors on in- and unseen out-of- distribution data generations, which is achieved by optimizing our proposed criterion.

Q2. *What property(ies) of OOD sets are important to differentiate and then recognize the most proper OOD set for the purpose of training?*
Answer: Differentiating a large number of OOD sets w.r.t an in-distribution set through optimizing the above-mentioned criterion is computationally expensive. Instead of directly optimizing this criterion, we speculate the *protectiveness level* of OOD sets may be a good property for differentiating the OOD sets. We describe an OOD set as protective set of a

given in-distribution set if it can shield all in-distribution's sub-manifolds, each sub-manifold is associated with each class of the in-distribution task.

Q3. *Inspired by these properties, how we could design a collection of cost-effective metrics to effectively identify the most proper OOD set?*

Answer: We propose three metrics to measure the protectiveness level of an OOD set w.r.t a given in-distribution set in order to select the most proper OOD set among those available ones. These metrics are I) softmax-based entropy, II) coverage ratio, and III) coverage distance. Using these metrics, we successfully select the most proper OOD set for the purposes of effective OOD learning across different classification tasks, namely image and sound classification.

The followings are the peer-review papers associated with the second objective:

Workshop **M. Abbasi**, C. Shui, A. Rajabi, C. Gagné, Rakesh B. Bobba, Toward Metrics for Differentiating Out-of-Distribution Sets, Conference on Neural Information Processing Systems (NeurIPS), Workshop on Safety and Robustness in Decision Making, 2019, Vancouver, Canada.

Conference **M. Abbasi**, C. Shui, A. Rajabi, C. Gagné, Rakesh B. Bobba, Toward Metrics for Differentiating Out-of-Distribution Sets [an extension of the workshop paper], 24$^{\text{th}}$European Conference on Artificial Intelligence, 2020, Santiago de Compostela, Spain.

Future Work

This work can be extended by creating a proper OOD set by selecting the important OOD samples from *the union of all available OOD sets*, instead of selecting a single proper set out of the available OOD sets. Selecting the important OOD samples gives us the control to build an OOD set with a higher protection level of the in-dist. sub-manifolds. In other word, it is possible that even the most proper OOD set among the available ones can not still protecting completely all the sub-manifolds. But by the OOD sample-selection, we can assure the achievement of the maximum protection level. By this sample selection strategy, we can also create a diverse set of OOD samples that are selected from different OOD tasks. In the near future, we will devise an approach for OOD samples selection out of an union of OOD sets. Moreover, since natural OOD sets is not always available for all in-distribution tasks (e.g. for medical datasets), generating of synthesized OOD samples can be a vital alternative. Thus, as another future work, one can design a generative approach to synthesize OOD sets that protect the in-distribution sub-manifolds (i.e. inspired by the properties of most protective OOD set).

On the capacity of A-CNNs for adversarial detection

In the preceding objective, we empirically show that the OOD learning of an A-CNN (if it is trained on a proper OOD set) leads to promising results for the detection of wide spectrum of unseen OOD sets as well as a limited-range of adversarial examples (i.e. FGS). However, several questions should be answered in order to illustrate better the promising influence of OOD learning on adversaries detection. The answers to the research questions associated with this objective are summarized as follows:

Q1 *Are ODD learning and adversarial learning equally effective to make an A-CNN more robust to I) the OOD samples, II) the black-box adversarial examples?*

Answer: While an adversarially trained A-CNN can not detect OOD samples, an A-CNN trained on a proper OOD set–recognized by our proposed metrics– can detect both unseen OOD samples and some limited types of adversaries. This shows that OOD learning of A-CNN can induce more reliable (in the presence of unusual samples) models than adversarially training. We also propose to augment a proper OOD set with some interpolated data from in-distribution samples in order to promote the detection rate of adversarial examples.

Q2 *Where the OOD samples and adversaries are located in the feature space of a vanilla CNN? How their locations in the feature space can be changed by OOD learning of an A-CNN?*

Answer: Looking closely at the feature space can give us some visual insights to understand the strange phenomenon of high predictive confidence for OOD samples and adversaries. Indeed, by visually assessing the feature space of two tasks (MNIST and CIFAR-10), we found that OOD samples are indeed entangled with the sub-manifolds although they are not *evenly* distributed all over the sub-manifolds. This can provide some explanations for high confidence prediction of vanilla CNN for the OOD samples. That is, since they are entangled with (or placed very near to) the in-distribution sub-manifolds, the vanilla CNN classifies them with high confidence. We expect that the OOD samples should be disentangled (or placed far away) from the in-dist. sub-manifolds so that the model can predict them with lower confidence. We observe the same entanglement behaviour for the adversarial examples, i.e. FGS and T-FGS.

Q3 *How does OOD learning affect the learned decision boundaries in the input space in the adversarial directions?*

Answer: We create some cross-sections from the input space for some input samples in order to visualize the decision boundaries that are created by a vanilla CNN and by a A-CNN. Then, by comparing their cross-sections, we visually and empirically show that the fooling regions around a given sample that are obtained by a vanilla CNN are occupied with the extra class (named dustbin regions) of the A-CNN (the well trained A-CNN).

The results are published in a peer-reviewed workshop, which was co-located with a major international conference on computer security:

Workshop **M. Abbasi**, A. Rajabi, C. Gagné, R. B. Bobba, "Towards Dependable Deep Convolutional Neural Networks (CNNs) with Out-distribution Learning", Dependable and Secure Machine Learning, co-located with IEEE conference on Dependable Systems and Networks, 2018, Luxembourg.

Future Work

An interesting future work can be theoretically verifying that training an A-CNN on an appropriate dustbin training set (e.g. a protective OOD set along with some interpolated samples) is sufficient for achieving a high generalization ability for detecting of adversarial examples, particularly those locating far away from the in-distribution sub-manifolds.

An application of A-CNNs for a robust object detector

In this objective, we showcase a novel application of OOD learning for training a robust object detector on a partially-labeled dataset that includes many missing-label instances. Practically some large-scale datasets (e.g. OpenImagev3) may contain some missing-label instances, but our main focus here is on a *merged dataset* with many missing-label instances, which are emerged after merging several fully-labeled datasets from similar contexts but with disjoint sets of Object of Interest (OoI).

By merging datasets, one can create a large-scale dataset (large in terms of sample complexity and the number of classes) with the possibility of alleviating domain-shift. Compared to operating with several object detectors for the constitutive datasets (in the merged one), a unified object detector trained on the merged dataset is computationally more promising (both in terms of space and time complexity), if we can deal with the missing-label instances in the merged dataset. Using OOD learning, we propose a self-supervised framework for simultaneously training an object detector while creating accurate pseudo-labels for the missing-label instances.

The research questions of this objective are answered as follows:

Q1 *How do the missing-label instances in a dataset negatively impact on the performance of the state-of-the-art object detectors such as YOLO and faster R-CNN* **Answer**: The missing-label instances can create false training signals, causing a drop in precision of an object detector. These false training signals are mainly formed by the objectiveness loss of the anchors enclosing an unlabelled positive instance (missing-label instance).

Q2 *How OOD learning can be helpful for handling missing-label objects of interest in the merged dataset in order to achieve a robust and accurate object detector?*
Answer: To mitigate the decline in precision, we correct the false training signals through generating some accurate pseudo-labels for the missing-label instances with the help of OOD learning. More specifically, we propose to incorporate an A-CNN, as a proxy network, into the pipeline of object detector so as to classify the unlabelled positive instances that

are detected by the detector. If the proxy network classifies them as one of the pre-defined positive objects, their pseudo-labels (objectiveness and class probabilities) are generated to be included in the training set. Otherwise, they are discarded from contributing in training of the object detector.

Q3 *How to generate more accurate pseudo-labels for the missing-label OoIs, particularly the hard-to-predict ones?*

Answer: To avoid generation of high confident false positive signals, i.e. the objects that are wrongly classified by the proxy network, we applied patch-dropout on the given detected object before feeding it to the proxy network. More precisely, we create several copies of a given input by the patch-dropout, then compute an average of their predictions produced by the proxy network (A-CNN). This can create more calibrated predictions, which ultimately allows us to train a more robust object detector.

A paper for our proposed method, titled as "Self-supervised Robust Object Detectors from Partially Labelled datasets", which is available at `https://arxiv.org/abs/2005.11549,2020.`, will be submitted to Pattern Recognition Letter.

In overall, the ultimate goal of this thesis is to **mitigate the vulnerability of state-of-the-art discriminative deep neural networks**, particularly CNN, to adversarial examples and OOD samples. To this end, we approach these robustness challenges from different angles, from calibrating the predictive confidence of CNNs to training an effective end-to-end model with an explicit rejection option. All the methods in this thesis aim to identify the unusual input instances such as adversaries and OOD samples. In addition to effectiveness, computational efficiency and simplicity are the key factors that are taking into account in our proposed methods.

Appendix A

List of Publications

The thesis leads to the following peer-reviewed publications.

Canadian2020 **M. Abbasi**, A. Rajabi, C. Gagne, R. Bobba, “Toward Adversarial Robustness by Diversity in an Ensemble of Specialized Deep Neural Networks”, Long paper in Canadian Conference on AI, 2020.

ECAI2020 **M. Abbasi**, C. Shui, A. Rajabi, C. Gagne, R. Bobba, “Towards metrics for differentiating Out-of-Distribution sets”, NeurIPS Workshop on Safety and Robustness in Decision-Making, 2019, and accepted in European Conference on Artificial Intelligence (ECAI), 2020.

DSML2018 **M. Abbasi**, A. Rajabi, C. Gagné, R. B. Bobba, "Towards Dependable Deep Convolutional Neural Networks (CNNs) with Out-Distribution Learning", Dependable and Secure Machine Learning (DSML), co-located with IEEE conference on Dependable Systems and Networks (DSN), 2018.

ICLR-W2017 **M. Abbasi**, C. Gagné, "Robustness to Adversarial Examples through an Ensemble of Specialists." International Conference on Learning Representations (ICLR) Workshop, 2017.

To be submitted paper, available on Arxiv:

- **M. Abbasi**, D. Laurendeau, C. Gagné, "Self-supervised Robust Object Detectors from Partially Labelled Datasets", https://arxiv.org/abs/2005.11549, 2020.

The research papers collaborated during writing this thesis:

- C. Shui, **M. Abbasi**, L.E. Robitaille, B. Wang, C. Gagné, "A Principled Approach for Learning Task Similarity in Multitask Learning", International Joint Conference on Artificial Intelligence (IJCAI), 2019.

- F. Kiaee, C. Gagné, **M. Abbasi**, "Alternating Direction Method of Multipliers for Sparse Convolutional Neural Networks", Arxiv Preprint, 2017.

Appendix B

Supplementary Materials

In this appendix, we provide more detailed information on some experiments done in the thesis.

B.1 Chapter 3

B.1.1 Urban-Sound dataset

Like [83], we convert 3-seconds audio sounds to single channel image-like data with size 128×128 by extracting log-scaled mel-spectrograms with 128 bands, using a window size of 23 ms (1024 samples at 44100Hz) with the same stride size. If a given audio is more than 3 seconds long, we randomly clip a 3 second from its corresponding mel-spectrogram patch and if it is less than 3 seconds, its patch is padded to make all it 128. furthermore, we consider the CNN described in Table B.1 trained for 150 epochs using SGD with learning rate of 0.001 with momentum 0.9.

To train A-CNN for Urban-Sound on ECS (as an OOD set), we remove the overlap classes between ECS and Urban-Sound.

L#1	Conv: [1, 24, 5, 1, 2]; maxpool: [4, 2, 0]; relu
L#2	Conv: [24, 48, 5, 1, 2]; maxpool: [4, 2, 0]; relu
L#3	Conv: [48, 48, 5, 1, 2]; maxpool: [4, 2, 0]; relu:
L#4	Conv :[48, 128, 5, 1, 2]; maxpool: [4, 2, 0] ;relu
L#5	Fully Connected (FC): [4608, 128]; dropout: 0.5; relu
L#6	FC layer : [128, 10]; softmax

Table B.1: CNN used for Urban-Sound dataset. Conv: [#input channels, #out channels, kernel size, stride, padding], maxpool: [kernel size, stride, padding] and FC: [input dim, output dim]

B.1.2 Detailed Results

We report in Table B.2, the numeric values of our metrics computed for OOD sets of in-distribution sets as well as our metric values for *test in-distribution sets*. Although the latter set (in-distribution test sets) are not at all OOD, we just show their behaviour for covering

their own sub-manifolds, which are approximated by *in-distribution training samples in the feature space*. As expected, in-distribution test sets have the largest coverage ratio and SE as well as the smallest CD since they actually belong to their corresponding manifolds, where they are supposed to exactly lie.

In-dist.	OOD sets	CR (%) ↑	SE ↑	CD ↓
SVHN	Gaussian	14.91	2.035	2.13
	LSUN	14.63	2.054	2.16
	C100	15.66	2.181	2.17
	T-ImgNt	13.83	2.109	2.21
	C10	14.39	2.11	2.23
	ISUN	11.37	2.073	2.3
	Test in-dist	**70.67**	**2.302**	**0.55**
C10	Gaussian	1.93	0.264	2.23
	SVHN	9.04	1.538	2.39
	C100*	21.39	2.158	2.49
	T-ImgNt	16.46	1.908	2.68
	ISUN	13.28	1.766	2.68
	LSUN	12.93	2.039	2.95
	Test in-dist	**80.98**	**2.3**	**1.034**
Audio	WhiteNoise	1.110	0.031	0.87
	command	27.08	0.654	0.32
	ECS	40.62	2.093	0.44
	TuT	15.79	1.382	0.048
	Test in-dist	**28.70**	**2.21**	**0.36**

Table B.2: Numeric values of our proposed metrics for OO sets and in-distribution test set.

B.2 Chapter 4

B.2.1 Visualization of Feature Space by t-SNE

In section 4.3.5 (Fig. 4.4), we used PCA for visualization purposes. Here, we aim to visualize the feature space of A-CNN* and that of a vanilla CNN in 2D space by the use of t-SNE [104], instead of using PCA (Fig B.1).

B.2.2 Adaptive PDG adversarial examples by A-CNN*

To show the robustness of A-CNN* to the adaptive (i.e. white-box) adversaries, we report the success rate of generating PGD adversaries as a function of total iteration (T) and the overall noise magnitude (α) with the step size of $\epsilon' = 5 \times 10^{-3}$ for CIFAR-10 (refer section 1.2 to read the PGD algorithm). For generating adaptive adversaries, the generated samples that are classified as the dustbin class by A-CNN* are not considered as adversarial examples since the A-CNN* already rejects them. Therefore, generation of the adaptive adversaries becomes

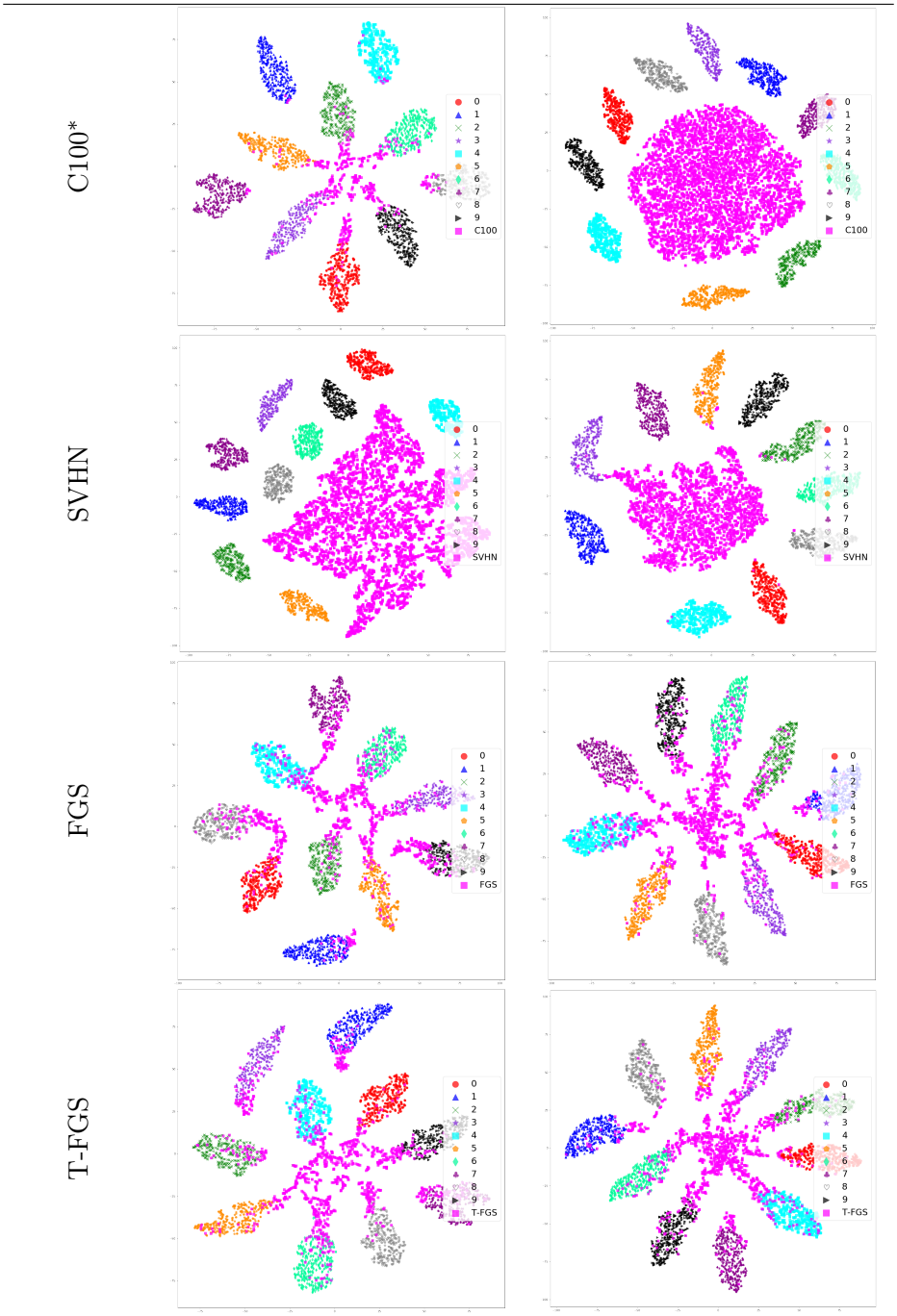


Figure B.1: Visualization of the feature space of vanilla CNN and A-CNN* for CIFAR-10 in presence of various OOD sets. We used t-SNE with perplexity=20.

harder as the PGD algorithm, which aims to increase the loss function of A-CNN*, causes the generated sample to be placed in the the dustbin region, where the loss is maximized. As a result, the success rate of generating PGD attack using A-CNN* is not significantly high (i.e. for the best iteration and noise magnitude, the success rate is $\approx 35\%$). As it can be seen in Fig B.2, the success rate of PGD is decreasing by increasing the noise magnitude. This is because the larger noise magnitude generates adversaries with large amount of noise, which are located further away from the protected in-distribution sub-manifolds (likewise the FGS adversaries in Fig 3.8).

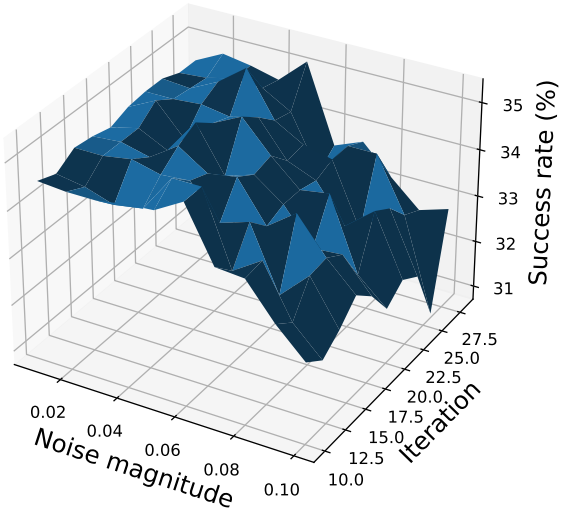


Figure B.2: The success rates of generating adaptive PDG adversaries as a function of iteration and noise magnitude.

Bibliography

[1] *How Google Translate squeezes deep learning onto a phone*, accessed February 24, 2014. http://googleresearch.blogspot.ca/2015/07/how-google-translate-squeezes-deep.html.

[2] Bendale, A., and Boult, T. E. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1563–1572.

[3] Bengio, Y. Learning deep architectures for AI. In *Foundations and Trends® in Machine Learning* (2009), vol. 2, Now Publishers Inc., pp. 1–127.

[4] Bengio, Y. Deep learning of representations: Looking forward. In *Statistical Language and Speech Processing*. Springer, 2013, pp. 1–37.

[5] Bevandic, P., Kreso, I., Orsic, M., and Segvic, S. Discriminative out-of-distribution detection for semantic segmentation. *arXiv preprint arXiv:1808.07703* (2018).

[6] Bilen, H., and Vedaldi, A. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2846–2854.

[7] Brahma, P. P., Wu, D., and She, Y. Why deep learning works: A manifold disentanglement perspective. vol. 27, pp. 1997–2008.

[8] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[9] Carlini, N., and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263* (2017).

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

[11] Charles, Z., Rosenberg, H., and Papailiopoulos, D. A geometric perspective on the transferability of adversarial directions. *arXiv preprint arXiv:1811.03531* (2018).

[12] Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., and Petrick, N. Calibration of medical diagnostic classifier scores to the probability of disease. vol. 27, SAGE Publications Sage UK: London, England, pp. 1394–1409.

[13] Choi, H., Jang, E., and Alemi, A. A. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392* (2018).

[14] Cortes, C., DeSalvo, G., and Mohri, M. Learning with rejection. In *International Conference on Algorithmic Learning Theory* (2016), Springer, pp. 67–82.

[15] Da, Q., Yu, Y., and Zhou, Z.-H. Learning with augmented class by exploiting unlabeled data. In *Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014).

[16] DeVries, T., and Taylor, G. W. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865* (2018).

[17] Diba, A., Sharma, V., Pazandeh, A., Pirsiavash, H., and Van Gool, L. Weakly supervised cascaded convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 914–922.

[18] Dietterich, T. G. Ensemble methods in machine learning. In *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[19] Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[20] Geifman, Y., and El-Yaniv, R. Selectivenet: A deep neural network with an integrated reject option. *International Conference on Machine Learning (ICML)* (2019).

[21] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research 32*, 11 (2013), 1231–1237.

[22] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1440–1448.

[23] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 580–587.

[24] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082* (2013).

[25] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *3th International Conference on Learning Representations (ICLR)* (2015).

[26] Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[27] Gu, S., and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. In *Workshop on International Conference on Learning Representations (ICLR)* (2015).

[28] Gunther, M., Cruz, S., Rudd, E. M., and Boult, T. E. Toward open-set face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 71–80.

[29] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR, pp. 1321–1330.

[30] Hansen, L. K., and Salamon, P. Neural network ensembles. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1990), no. 10, pp. 993–1001.

[31] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. vol. 37, IEEE, pp. 1904–1916.

[32] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[33] He, W., Wei, J., Chen, X., Carlini, N., and Song, D. Adversarial example defenses: Ensembles of weak defenses are not strong. *arXiv preprint arXiv:1706.04701* (2017).

[34] Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 41–50.

[35] Hendrycks, D., and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[36] Hendrycks, D., Mazeika, M., and Dietterich, T. G. Deep anomaly detection with outlier exposure. *Internation Conference on Representation Learning (ICLR)* (2019).

[37] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems (NIPS), Deep Learning and Representation Learning Workshop* (2015).

[38] Hosseini, H., Chen, Y., Kannan, S., Zhang, B., and Poovendran, R. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318* (2017).

[39] Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks* (2013), no. 1288.

[40] Huang, F. J., and LeCun, Y. Large-scale learning with svm and convolutional for generic object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition* (2006), vol. 1, IEEE, pp. 284–291.

[41] Huang, R., Xu, B., Schuurmans, D., and Szepesvári, C. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034* (2015).

[42] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems* (2019), pp. 125–136.

[43] Jiang, H., Kim, B., Guan, M., and Gupta, M. To trust or not to trust a classifier. In *Advances in Neural Information Processing Systems* (2018), pp. 5541–5552.

[44] Kariyappa, S., and Qureshi, M. K. Improving adversarial robustness of ensembles with diversity training. *arXiv preprint arXiv:1901.09981* (2019).

[45] Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., and Murphy, K. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *"https://github.com/openimages"* (2017).

[46] Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, April 2009.

[47] Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[48] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems* (2017), pp. 6405–6416.

[49] LeCun, Y. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/* (1998).

[50] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning* (2009), pp. 609–616.

[51] Lee, J. D., and See, K. A. Trust in automation: Designing for appropriate reliance. vol. 46, SAGE Publications Sage UK: London, England, pp. 50–80.

[52] Lee, K., Lee, H., Lee, K., and Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *5th International Conference on Learning Representations (ICLR)* (2017).

[53] Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems* (2018), pp. 7167–7177.

[54] Lee, K., Lee, K., Shin, J., and Lee, H. Overcoming catastrophic forgetting with unlabeled data in the wild. In *International Conference on Computer Vision* (2019).

[55] Liang, S., Li, Y., and Srikant, R. Principled detection of out-of-distribution examples in neural networks. *6th International Conference on Learning Representations (ICLR)* (2018).

[56] Liao, F., Liang, M., Dong, Y., Pang, T., Zhu, J., and Hu, X. Defense against adversarial attacks using high-level representation guided denoiser. *arXiv preprint arXiv:1712.02976* (2017).

[57] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2117–2125.

[58] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision* (2014), Springer, pp. 740–755.

[59] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. SSD: Single shot multibox detector. In *European Conference on Computer Vision* (2016), Springer, pp. 21–37.

[60] Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).

[61] Logan, E., Tsipras, D., Schmidt, L., and Madry, A. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779* (2017).

[62] Lu, J., Issaranon, T., and Forsyth, D. Safetynet: Detecting and rejecting adversarial examples robustly. In *The IEEE International Conference on Computer Vision* (2017).

[63] Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. *6th International Conference on Learning Representations (ICLR)* (2018).

[64] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *6th International Conference on Learning Representations (ICLR)* (2018).

[65] Masana, M., Ruiz, I., Serrat, J., van de Weijer, J., and Lopez, A. M. Metric learning for novelty and anomaly detection. *British Machine Vission Conference* (2018).

[66] Meinke, A., and Hein, M. Towards neural networks that provably know when they don't know. In *8th International Conference on Learning Representations (ICLR)* (2020).

[67] Meng, D., and Chen, H. Magnet: a two-pronged defense against adversarial examples. In *ACM SIGSAC Conference on Computer and Communications Security* (2017).

[68] Mesaros, A., Heittola, T., and Virtanen, T. TUT database for acoustic scene classification and sound event detection. In *24th European Signal Processing Conference* (2016).

[69] Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. *5th International Conference on Learning Representations (ICLR)* (2017).

[70] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[71] Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know?

[72] Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint arXiv:1412.1897* (2014).

[73] Papernot, N. *Characterizing the limits and defenses of machine learning in adversarial settings*. PhD thesis, The Pennsylvania State University, 2018.

[74] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (2017), ACM, pp. 506–519.

[75] Piczak, K. J. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia* (2015), ACM, pp. 1015–1018.

[76] Qiu, S., Liu, Q., Zhou, S., and Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences 9*, 5 (2019), 909.

[77] Rame, A., Garreau, E., Ben-Younes, H., and Ollion, C. Omnia faster r-cnn: Detection in the wild through dataset merging and soft distillation. *arXiv preprint arXiv:1812.02611* (2018).

[78] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 779–788.

[79] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[80] Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., and Lakshminarayanan, B. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems* (2019), pp. 14707–14718.

[81] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems* (2015), pp. 91–99.

[82] Rozsa, A., Rudd, E. M., and Boult, T. E. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2016), pp. 25–32.

[83] Salamon, J., and Bello, J. P. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters 24*, 3 (2017), 279–283.

[84] Salamon, J., Jacoby, C., and Bello, J. P. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM International Conference on Multimedia* (2014), ACM, pp. 1041–1044.

[85] Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boult, T. E. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence 35*, 7 (2013), 1757–1772.

[86] Scheirer, W. J., Jain, L. P., and Boult, T. E. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence 36*, 11 (2014), 2317–2324.

[87] Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on mnist. *6th International Conference on Learning Representations (ICLR)* (2018).

[88] Serre, T., Kreiman, G., Kouh, M., Cadieu, C., Knoblich, U., and Poggio, T. A quantitative theory of immediate visual recognition. *Progress in Brain Research 165* (2007), 33–56.

[89] Shalev-Shwartz, S., and Ben-David, S. *Understanding machine learning: From theory to algorithms*. CambridgeUniversity Press, 2014.

[90] Sharif, M., Bauer, L., and Reiter, M. K. n-ML: Mitigating adversarial examples via ensembles of topologically manipulated classifiers. *arXiv preprint arXiv:1912.09059* (2019).

[91] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[92] Singh, K. K., and Lee, Y. J. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *IEEE International Conference on Computer Vision* (2017), IEEE, pp. 3544–3553.

[93] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[94] Strauss, T., Hanselmann, M., Junginger, A., and Ulmer, H. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1709.03423* (2017).

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

[96] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[97] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708.

[98] Tanay, T., and Griffin, L. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690* (2016).

[99] Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences 99*, 10 (2002), 6567–6572.

[100] Tramèr, F., and Boneh, D. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems* (2019), pp. 5858–5868.

[101] Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).

[102] Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453* (2017).

[103] Van Calster, B., McLernon, D. J., van Smeden, M., Wynants, L., and Steyerberg, E. W. Calibration: the achilles heel of predictive analytics. *BioMed Central Medicine 17*, 1 (2019), 1–7.

[104] van der Maaten, L., and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605.

[105] Varshney, K. R., and Alemzadeh, H. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big Data 5*, 3 (2017), 246–255.

[106] Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning* (2019), pp. 6438–6447.

[107] Vyas, A., Jammalamadaka, N., Zhu, X., Das, D., Kaul, B., and Willke, T. L. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *The European Conference on Computer Vision (ECCV)* (September 2018).

[108] Wang, M., and Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing 312* (2018), 135–153.

[109] Wang, X., Cai, Z., Gao, D., and Vasconcelos, N. Towards universal object detection by domain attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 7289–7298.

[110] Wang, Y., Jha, S., and Chaudhuri, K. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning* (2018), pp. 5133–5142.

[111] Warde-Farley, D., and Goodfellow, I. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics* (2016), 311.

[112] Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).

[113] Wu, Z., Bodla, N., Singh, B., Najibi, M., Chellappa, R., and Davis, L. S. Soft sampling for robust object detection. In *British Machine Vision Conference (BMVC)* (2019).

[114] Xu, M., Bai, Y., Ghanem, B., Liu, B., Gao, Y., Guo, N., Ye, X., Wan, F., You, H., Fan, D., et al. Missing labels in object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2019).

[115] Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *Network and Distributed System Security Symposium* (2018).

[116] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 2636–2645.

[117] Yu, Y., Qu, W.-Y., Li, N., and Guo, Z. Open-category classification by adversarial sample generation. *International Joint Conference on Artificial Intelligence (IJCAI)* (2017).

[118] Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems 30*, 9 (2019), 2805–2824.

[119] Zhang, H., Chen, H., Song, Z., Boning, D., Dhillon, I. S., and Hsieh, C.-J. The limitations of adversarial training and the blind-spot attack. *7th International Conference on Learning Representations (ICLR)* (2019).

[120] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[121] Zhang, Y., Bai, Y., Ding, M., Li, Y., and Ghanem, B. Weakly-supervised object detection via mining pseudo ground truth bounding-boxes. *Pattern Recognition 84* (2018), 68–81.