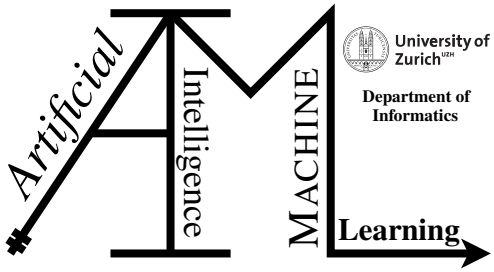
Adversarial Training for Improved Adversrial Stability in Open-Set Networks

Bachelor Thesis

Uros Dimitrijevic
15-936-834

Submitted on
August 22 2023

Thesis Supervisor
Prof. Dr. Manuel Günther



Bachelor Thesis

Author: Uros Dimitrijevic, uros.dimitrijevic@uzh.ch

Project period: 8 February 2023 - 22 August 2023

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

I sincerely thank Prof. Manuel Günther for guiding me through this complex research field and making this thesis feasible. Additionally, I would like to express my graditude to my family and friends for providing me support throughout these times.

Abstract

Deep neural networks have found great success in various recognition tasks. While their performances speak for themselves, they are still not fully understood. In particular, deep neural networks are susceptible to adversarial attacks. Research has found ways to defend against these attacks, one of the strategies being adversarial training, where networks are introduced to adversarial samples during training time. Another field where deep neural networks face problems are open-set recognition tasks, where the neural network has to address samples that do not belong to any known class. Some of the approaches addressing this problem incorporate samples, not belonging to any known class, whilst using a specific loss functions like the entropic open-set loss. The question remains if these two problems are somehow related to each other. Prior work suggested that open-set performance can be achieved by utilizing adversarial training. In this thesis we perform adversarial training on different types of loss functions, research these networks for adversarial stability, and evaluate their open-set recognition performances.

Zusammenfassung

Deep neural networks haben bei verschiedenen Erkennungsaufgaben große Erfolge erzielt. Obwohl ihre Leistungen für sich sprechen, sind diese Netzwerke nicht vollständig erkundet. Insbesondere sind deep neural networks anfällig für "adversarial" Angriffe. Die Forschung hat Wege gefunden, sich gegen diese Angriffe zu verteidigen. Eine dieser Strategien ist das adversarial Training, bei dem Netzwerke während der Trainingszeit mit adversarial Proben vertraut gemacht werden. Ein weiterer Bereich, in dem deep neural networks auf Probleme stoßen, sind Open-Set-Erkennungsaufgaben, bei denen das Netzwerk Proben adressieren muss, die keiner bekannten Klasse angehören. Einige der Ansätze zur Lösung dieses Problems beziehen Stichproben ein, die keiner bekannten Klasse angehören, und verwenden gleichzeitig spezifische Verlustfunktionen wie den entropic open-set loss. Es bleibt die Frage, ob diese beiden Probleme irgendwie miteinander zusammenhängen. Frühere Arbeiten legten nahe, dass relativ gute Open-Set Leistung durch adversarial Training erreicht werden kann. In dieser Arbeit führen wir adversarial Training für verschiedene Arten von Verlustfunktionen durch, untersuchen diese Netzwerke auf adversarial Stabilität und bewerten ihre Open-Set Leistung.

Contents

Chapter 1

Introduction

The research revolving around deep neural networks (DNNs) is steadily growing in size and the fields such networks are applied to are constantly expanding. Image recognition is one of those fields where DNNs have found major success. The networks are able to learn how to classify the content of images fairly accurately based on examples of available classes. Despite achieving high accuracy in correctly classifying images, research has shown that deep neural networks are susceptible to adversarial attacks Goodfellow et al. (2015). Adversarial attacks can be regarded as optical illusions to DNNs, their purpose mainly being to deceive the network while retaining visual form for humans. An adversarial sample is commonly characterized as the perturbed counterpart of a correctly classified clean sample. The perturbation should be done in such a way, that the network misclassifies the sample, while ideally, it remains visually indistinguishable from the previous clean image by the human eye. The misclassification is frequently performed with a high assumed certainty, by the network, making this phenomenon a very interesting field of study. While there are conflicting explanations for the vulnerability of DNNs against adversarial attacks, research has shown that defenses such as adversarial training can improve the robustness against them (Wang et al., 2019). The training method essentially consists of involving adversarial attacks during the training phase of the networks, making them less prone to these attacks.

Another issue deep neural networks face is handling unknown classes. Generally during training the networks receive solely information from the known classes, as such it is later able to distinguish between them. The problem arises when a network receives an image not belonging to the known class pool, but only has the option to classify it as one of the known classes. These images, not belonging to any of the classes, are misclassified frequently with a high certainty, the same as the adversarial images. Open-set recognition (OSR) tasks address this problem, one of the approaches being the entropic open-set (EOS) training technique introduced by Dhamija et al. (2018). Contrary to the closed-set approaches where the networks are only trained with known classes as positives, EOS incorporates unknown samples as negatives into the training phase, as such the networks are able to distinguish known classes from the unknown ones. The selection of unknown samples for the training phase is still a relevant question. The question is, which samples are suitable to represent the whole spectrum of the unknown classes?

An interesting idea is to regard adversarial images as unknown samples and incorporate them as negatives. While adversarial training has been done in closed-set scenarios, where the adversarial samples are used as positives, adversarial training in the open-set scenario still remains a rather new field. Both Schnyder (2021) and Palechor (2022) used adversarial samples as negatives in combination with the EOS loss during their experiments. Their findings indicated the reasonable effectiveness of such adversarial training. While their experiments mostly revolved around evaluating the open-set capabilities of such trained networks, their adversarial stability remains to be evaluated.

In this thesis, we focus on researching different aspects. Firstly, we want to analyze the ad-

versarial stability of standard open-set models and compare them to their standard closed-set trained counterparts. The overall OSR capabilities shall also be compared to each other. We use three different models for this purpose to address this question:

- Are standard open-set trained networks more robust than the standard closed-set trained network against adversarial attacks?

Secondly, we search for the most suitable adversarial samples to train open-set with, so that they are resistant to adversarial attacks. The main variable we impose is that the open-set networks still remain relatively viable for OSR tasks. Similarly, we search suitable adversarial samples for the closed-set network, to achieve adversarial stability. The adversarial trained networks are tested on their adversarial stability to address the question:

- What adversarial samples are suitable to train open-set and closed-set networks with to achieve adversarial stability?

Lastly, we want to find how the adversarial trained networks compare to their standard trained counterpart. We compare them based on their OSR capabilities and adversarial stability.

- Are open-set adversarial trained networks viable?

Chapter 2

Background and Related Work

This chapter covers the relevant topics on which the thesis is based while omitting the fundamental theories of machine learning and rather focusing on its subtopics. Section 2.1 describes the open-set scenario and discusses open-set recognition difficulties. Section 2.2 introduces different approaches tackling the OSR problem. In Section 2.3 adversarial attacks and their taxonomy are presented. Finally, Section 2.4 discusses the methods and related work on how adversarial training is done.

2.1 The open-set scenario

As the name entails, open-sets have no exact limit points, while closed-sets have clear limit points. In the context of machine learning, one tries to establish boundaries between classes to properly differentiate between them, for example through features. While in the closed-set scenario there are only *known* classes whose boundaries are known and defined, the open-set scenario introduces *unknown* classes whose boundaries are open and thus invite limitless possibilities and problems. In the field of machine learning, open-set recognition (OSR) tackles these problems. The concept of OSR was formally introduced to the machine learning world by Scheirer et al. (2013), where the authors defined the OSR problem as minimizing the open-set risk. In the following years, researchers sought different approaches to tackle these *unknown* classes, by developing networks suitable to minimize the risk of misclassification for an open-set scenario. Various approaches have been developed, which will be elaborated upon in the next section and although the field of OSR is compromised by a wide range of machine learning tasks, this thesis focuses mainly on the visual multiclass classification problem of it.

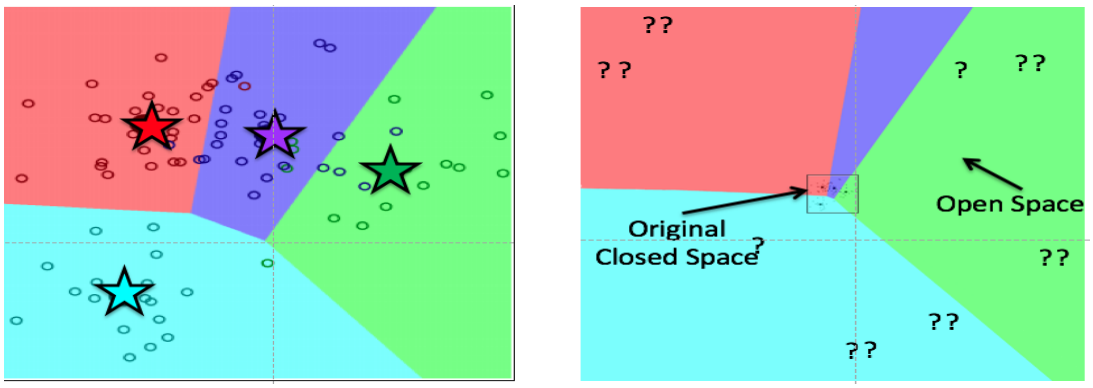
Contrary to the open-set scenario, in the traditional closed-set scenario, multiclass recognition tasks focus exclusively on the classification of known classes. In the closed-set recognition (CSR) setting all classes are known during the training time and generally the goal of a CSR-trained network would be to differentiate between these known classes accurately during testing. Whilst OSR operates with the same goal in mind, it also has the task of handling unknown classes. Although these unknown classes may occur during training time, their inclusion and handling during testing are the main aspects of OSR tasks. As such, the goal of an OSR trained network would be to differentiate between known classes, while also having the ability to discern unknown classes as unknown. The objective of OSR is only a slight extension of the CSR one, but the increase in complexity is immense due to the limitless possibilities of the unknown classes. Albeit the stark rise in complexity, the open-set scenario embodies real-world recognition cases and their problems more accurately and its research is a worthwhile endeavor. Likewise, many closed-set scenario approaches are desired to be extended to an open-set scenario for this rea-

son. As mentioned by Scheirer et al. (2013), open-set recognition faces the problem of incomplete knowledge of the limitless variety in unknown samples and the finite training time, thus one goal is to balance out the risk of the unknown in the open space with the known risk. The task of finding suitable representative samples of the unknown classes, which are to be used during training and testing, is of great importance. The general goal is to find samples that represent a vast amount of possible and probable unknown classes that are present during testing time. Otherwise, an OSR network would either need to be trained endlessly due to the limitless number of unknown classes, or it may only be specialized for certain unknown classes. To tackle this problem and to put it in the multiclass classification context for a better understanding, this thesis is relying on the notation proposed by Dhamija et al. (2018), which defines the following:

- $\mathcal{Y} \subset \mathbb{N}$: The infinite label space of all classes. These are all the classes we can define with our language.
- $\mathcal{C} = \{1, ..., C\} \subset \mathcal{Y}$: The set of the *known* classes of interest that the network shall identify. These are the classes the model should be able to differentiate between. Their samples are usually referred to as *positives* in the OSR setting.
- $\mathcal{U} = \mathcal{Y} \setminus \mathcal{C}$: The *unknown* classes containing all types of classes the network needs to reject. These are the classes the model should not be able to differentiate between, but be able to reject as not being part of known classes $\mathcal{C}$. Furthermore, $\mathcal{U}$ can be divided into:
  1. $\mathcal{B} \subset \mathcal{U}$: The set of *known unknown* classes. These are the classes the network has some knowledge of. Generally they will be introduced during training and can also occur during testing of the network. Their samples are also usually referred to as *negatives*, as they stand in opposition to the positive/known samples during training.
  2. $\mathcal{A} = \mathcal{U} \setminus \mathcal{B} = \mathcal{Y} \setminus (\mathcal{C} \cup \mathcal{B})$: The set of *unknown unknown* classes. These are the classes that are not used during training of the network and are only used during testing. Their samples are also commonly referred to as *unknown samples*, as they are truly unknown to the network.

As such, $\mathcal{D}'_c$ are samples belonging to the known classes $\mathcal{C}$ seen during training and $\mathcal{D}_c$ are the samples only seen during testing. Likewise, the samples belonging to the *known unknown* classes $\mathcal{B}$ during training and testing are depicted as $\mathcal{D}'_b$ and $\mathcal{D}_b$ respectively. The samples seen only during testing, representing *unknown unknown* classes are depicted as $\mathcal{D}_a$. Thus, the overall unknown test samples are represented as $\mathcal{D}_u = \mathcal{D}_a \cup \mathcal{D}_b$.

To illustrate the open-set recognition problem in the context of multiclass classification tasks, observe Figure 2.1 The first Figure 2.1(a) shows a closed-set model in a two-dimensional field. The points represent samples, and their coordinates represent some sort of feature that can be divided into four separate classes. In our case, these samples would belong to $\mathcal{D}_c$. Here it is assumed that only samples from known classes are present, and the boundaries are defined by the edges and lines of the figure. The model is hence only trained and tested with these boundaries and possibilities in mind. Going over to the second Figure 2.1(b), the space is opened up. While the samples $\mathcal{D}_c$ remain in the predefined boundary, unknown samples $\mathcal{D}_u$ represented by '?' are invited into the plane field. These unknown samples are classified as one of the known classes because the current classifying method of the model can only operate on the CSR setting. As can be seen, a closed-set trained model is an inadequate solution to solve an open-set problem.



(a) Closed-set Scenario (b) Open-set Scenario

Figure 2.1: OPEN SPACE. *The subfigure (a) shows a closed-set view and the subfigure (b) shows an open-set view of the same classification problem. Source: (Boult et al., 2019)*

2.2 OSR Approaches

Open-set recognition approaches aim to correctly classify known samples while rejecting unknown samples. For this objective, several approaches have been proposed. In a recent survey by Geng et al. (2020) a multitude of open-set recognition approaches have been comprehensively covered and were also shown on how they link to each other. The authors divide the approaches between discriminative and generative perspectives and remark that there might be an unknown hybrid method still to be researched for.

Generative models are separated into Non-Instance generation and instance generation. Non-instance generation seems to be in its infancy and Geng et al. (2020) only lists one approach, which applies the hierarchical Dirichlet process to the OSR setting Geng and Chen (2020). A noteworthy example of an instance generation model is the Generative OpenMax (G-OpenMax) by Ge et al. (2017). The approach combines generative adversarial networks (GANs) (Goodfellow et al., 2014) with the OpenMax algorithm (Bendale and Boult, 2016a), by producing synthetic samples with their own new label and using them besides the known label samples during training. Although this method uses deep neural networks it is categorized as an instance generation-based method, as it generates unknown samples during the learning phase.

The function of discriminative models is, as the name suggests, to establish some form of boundaries based on the determined probability scores of the samples. This boundary formation is mostly done through a thresholding scheme. Once a sample reaches a certain probability score it will either be classified to a known class or if it falls below it, it is marked as unknown. The approaches are furthermore subdivided into traditional machine learning (TML) based and DNN-based models. A few notable TML-based methods are the "1-vs-set machine" by Scheirer et al. (2013), which adds another plane to traditional support vector machine-based models to minimize the open risk, the sparse representation open-set recognition model (SROSR) by Akhtar and Mian (2018), which utilizes the Extreme Value Theory (EVT) modeled error distributions, Bendale and Boult (2016b) proposed the Nearest Non-Outlier (NNO) method, which extends on the Nearest Class Mean classifier by adding object categories incrementally for novelty detection and Rudd et al. (2018) introduced the Extreme Value Machine (EVM), which also uses EVT to extend upon margin distribution based recognition algorithms. **DNN-based models** mainly focus on modifying the softmax layer and formulating a thresholding scheme to generalize a CSR

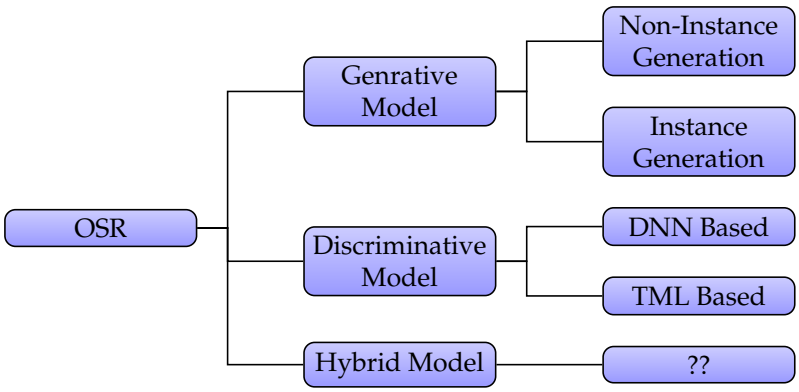


Figure 2.2: OSR METHODS. *OSR approaches, which are divided into different categories, depending on the methods used to minimize for open risk (Geng and Chen, 2020).*

model to an OSR setting. These DNN approaches are a central focus of study for this thesis and will be discussed in the following subsections in more detail.

2.2.1 Tresholding softmax scores

The softmax function is widely used in the field of machine learning and in the context of DNN multiclass classification, the function resides in the final layer as the activation function. The function can convert any vector into a probability distribution, which provides an evident service in determining the certainty of recognition tasks. Softmax was formally incorporated in the machine learning field by Bridle (1990) and is seen as a staple function in deep neural networks. The thresholding method enables every closed-set trained network to operate in the open-set scenario. Due to its simplicity and core features, the approach generally serves as a good benchmark for open-set classification purposes.

For this method, a neural network is commonly trained with only samples of known classes $\mathcal{C}$. Given a sample, the forward pass of the network provides a logit vector as the output. The softmax layer receives the logit vector as an input and converts it to positive numbers between 0 and 1. These numbers can be regarded as probability scores that have by design the sum of 1. Each score describes how probable it is for a sample to belong to a specific class. The softmax function is defined as follows:

$$S_c(x) = \frac{e^{l_c(x)}}{\sum_{c' \in \mathcal{C}} e^{l_{c'}(x)}} \tag{2.1}$$

where a sample x will receive a softmax value $S_c(x)$ for a specific class $c \in \mathcal{C} = \{1, ..., C\}$ and $l_c(x)$ represents the logit value of the specific class c. As such, the sum of all class probabilities equals to 1 for a given sample x:

$$\sum_{c=1}^{C} S_c(x) = 1 \tag{2.2}$$

In a closed-set scenario, generally, the maximum softmax score determines to what class a sample is classified to. Following this classification method, even unknown samples will be classified to a certain class, which by definition they certainly do not belong to. By thresholding to an arbitrary

softmax score, the closed-set recognition problem can be extended to an OSR scenario. Therefore, a sample will be classified as unknown if the highest softmax score is below the threshold. Given an arbitrary threshold value θ, a sample x is classified as unknown if it fulfills the condition:

$$\max_c S_c(x) < \theta \tag{2.3}$$

2.2.2 Softmax with a Background Class

Implementing a background or garbage class is another simplistic approach, which can be used in several ways (Zhang et al., 2017; Liu et al., 2016; Ren et al., 2015) to tackle open-set recognition tasks. This generally involves training the network with samples belonging to the classes of the sets $\mathcal{C}$ and $\mathcal{B}$. For the most simplistic case, a new class is created U, where all the background samples can be collected in. This unknown class U can be treated as one of the known classes, as their samples have their separate label. Let K be the number of classes to differentiate between and C number of known classes so that $C = K$ applies for the standard softmax approach. In the case of the background class approach, $C + 1 = K$ would apply. In a sense the conglomerate of the unknown classes U is contained as one class in the known classes $\mathcal{C} = \{1, ..., C, U\}$, while still not constraining these unknown samples in any way, besides them not belonging to the known classes. As such the softmax function for the probability scoring does not need any modification, besides the adjustment of the number of classes. The maximum softmax score of a sample still dictates which class it belongs to. By incorporating a background class for all the unknown samples, the thresholding scheme can be omitted and a previous closed-set neural network is operable in an open-set scenario. In an open-set recognition setting a sample x, with a target label $\hat{c}$ and all possible classes being $c \in \mathcal{C} = \{1, ..., C, U\}$, is correctly predicted by the neural network when the following condition applies:

$$\arg\max_c S_c(x) = \hat{c} \tag{2.4}$$

A natural drawback to this method is the needed addition of negative samples from $\mathcal{B}$ during training. This increases the training time and the selection of samples comes into question, which adds another layer of complexity. The wide variety of possible unknown samples and their usual larger quantity during training can cause a serious dataset bias (Tommasi et al., 2017), making the correct implementation of this approach much more complicated than the thresholding scheme. A way to counteract the bias while still including a wide variety of unknown samples is to apply a specific weight to the unknown class during the loss calculation. Let the standard categorical cross-entropy loss for a sample x be:

$$J_x = -\sum_{c=1}^{K} w_c t_{x,c} \log S_c(x) \tag{2.5}$$

K is the number of classes, w_c is the weight of a class c, $t_{x,c}$ is the target label value and $S_c(x)$ (2.1) is the softmax score of the sample x for the class c. For standard softmax loss, where the network is only trained with known classes, the classes are commonly weighted the same $\forall c : w_c = 1$. For the background approach, unknown samples can be weighted less depending on the size of the training set. In the case of multi-class classification, where the labels are one-hot encoded, $t_{x,c}$ is defined as:

$$\forall x, c \in \mathcal{C} : t_{x,c} = \begin{cases} 1 & \text{if } c = \hat{c}_x \\ 0 & \text{else} \end{cases} \tag{2.6}$$

where $\hat{c}_x$ represents the true class for a sample x. This means all softmax scores besides the one of the true class add 0 to the loss calculation for a given sample.

2.2.3 Entropic Open-Set Loss

The entropic open-set loss function was introduced by Dhamija et al. (2018) as an extension to the cross-entropy softmax loss. Same to the background class approach, neural networks operating on the EOS loss, require negative samples $\mathcal{B}$ during training to function properly in an OSR testing scenario. While the background approaches generally incorporate an additional class for the negatives, the entropic open-set loss approach does not add another class to the network. Instead, it uses the distribution of the softmax score to define what an unknown sample is. As stated by Dhamija et al. (2018), the entropic open-set loss maximizes the entropy of unknown samples, by using equivalent target values $t_{x,c}$ for every class, given a negative sample. For unknown samples during training i.e. negatives $x \in \mathcal{D}'_b$ the $t_{x,c}$ is defined as:

$$\forall x \in \mathcal{D}'_b, c \in \mathcal{C} : t_{x,c} = \frac{1}{|\mathcal{C}|} \tag{2.7}$$

while known samples during training i.e. positives $x \in \mathcal{D}'_c$ still have the one hot-encoded target values (2.6). For negative samples, the loss function, therefore, aims to evenly distribute softmax scores across the classes. This means that an unknown sample, in the best-case scenario, should receive the same probability score to belong to each of the known classes. In the same fashion as for standard softmax discussed in Section 2.2.1 a thresholding can be applied. The clear advantage here is that the neural network was explicitly trained for unknown classes, with the intent that the unknown samples will have evenly distributed softmax scores through the classes and thus fall below the threshold.

2.2.4 Open-Set Classification Rate

Additionally to the EOS loss Dhamija et al. (2018) introduced the Open-set Classification Rate (OSCR). Open-set recognition faces the problem of classifiers needing to accurately classify known classes while detecting and rejecting unknown samples. To perform the OSCR evaluation both samples from the known classes $\mathcal{D}_c$ and from the unknown classes $\mathcal{D}_u$ are needed. The *Correct Classification Rate* (CCR) is calculated by dividing the number of samples $\mathcal{D}_c$ which are correctly classified and have a probability score over θ by the total number of samples $|\mathcal{D}_c|$. The *False Positive Rate* (FPR) is calculated by dividing the number of unknown samples $\mathcal{D}_u$ which have a probability score for any known class $c \in \mathcal{C}$ over θ by their total number $|\mathcal{D}_u|$ (Dhamija et al., 2018).

$$\begin{aligned} FPR(\theta) &= \frac{|\{x \mid x \in \mathcal{D}_a \wedge \max_c P(c \mid x) \geq \theta\}|}{|\mathcal{D}_a|}, \\ CCR(\theta) &= \frac{|\{x \mid x \in \mathcal{D}_c \wedge \arg\max_c P(c \mid x) = \hat{c} \wedge P(c \mid x) \geq \theta\}|}{|\mathcal{D}_c|}. \end{aligned} \tag{2.8}$$

The CCR and FPR are plotted against each other, while the left side indicates a high probability threshold θ and the right side a low one. Values in the top left corner signify that a model has a high CCR and low FPR for a higher threshold θ. A balanced dataset is probably beneficial to capture a networks open-set perfomances. The OSCR metric is our general metric to evaluate the OSR capabilities of our models in the experiments.

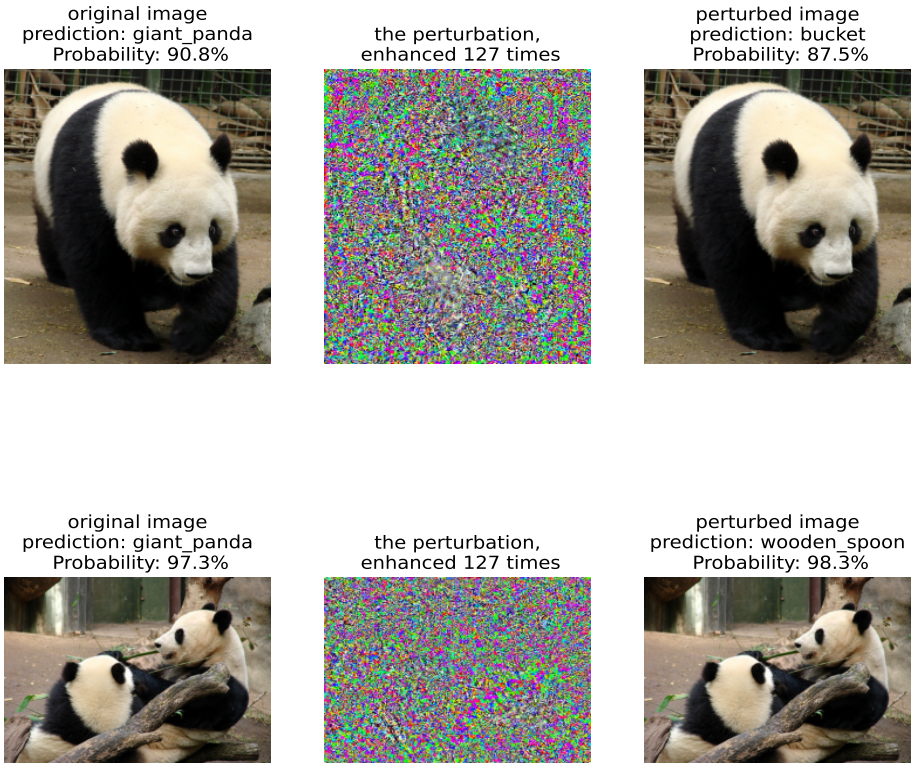


Figure 2.3: IMAGENET PERTURBATION. *Panda1 and Panda2 are perturbed and classified with help of Advertorch framework*[3]

2.3 Adversarial Attacks

Adversarial attacks are the second main topic of the thesis. The extensive survey by Akhtar and Mian (2018) shows that while deep neural networks have found great success on a variety of Computer Vision tasks, they are vulnerable to adversarial attacks. To have a more concise overview of the topic, multiple terms and definitions will be explained in the following sections.

2.3.1 Adversarial Samples

The term "adversarial samples" was first introduced by Szegedy et al. (2014), where the authors researched the stability of neural networks. They made the interesting observation that a state-of-the-art deep neural network can be fooled by slightly perturbed samples and coined such samples as "adversarial samples". Each of these samples has usually an original image counterpart, which was or would be correctly classified by the network. Contrary to rubbish (Goodfellow et al., 2015), negatives and unknown class samples, which humans would generally recognize as not belonging to the known classes, adversarial samples are ideally practically indistinguishable from their original counterparts by the human eye. This discovery raised major security concerns for deep neural networks and called for further research and discourse. In the adversarial attacks survey by Akhtar and Mian (2018), it is pointed out that there are varying explanations for the existence of adversarial samples. Goodfellow et al. (2015) hypothesizes their existence is the result of deep neural networks being too linear instead of nonlinear as previously thought. While this

linearity hypothesis explains the susceptibility of networks to certain adversarial attacks, other research (Tanay and Griffin, 2016; Luo et al., 2016) do not support it.

Taking Figure 2.3 for instance, both left images have been correctly classified by the network with a high probability. Through a minor but specific perturbation, the same network classifies the images to completely different classes, with a similar high probability. From a human perspective, the original images seem to be identical to the adversarial ones, so one must naturally conclude that the network has some form of defect as it fails to mimic the human recognition ability. Szegedy et al. (2014) formulates the creation of adversarial samples as an optimization problem:

$$\begin{aligned} min. \quad & ||x^{adv} - x||, \\ s.t. \quad & F(x^{adv}) = c \neq \hat{c}, \end{aligned} \tag{2.9}$$

where x is a clean sample, x^{adv} its adversarial counterpart, $||.||$ measures the difference of the samples, $F()$ is the classifier, c the predicted class and $\hat{c}$ the true class of x. The optimization problem can be read as finding the minimal perturbation needed so that the network misclassifies the perturbed sample. The image difference $||.||$ calculation is usually performed through l_p norms where $p = \{0, 1, 2, \infty\}$ (Goodfellow et al., 2015; Madry et al., 2019; Carlini and Wagner, 2017; Rozsa et al., 2017; Papernot et al., 2016):

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}. \tag{2.10}$$

2.3.2 Taxonomy of Adversarial Attacks and Defenses

There are numerous ways to perform adversarial attacks and likewise to defend against, Serban et al. (2020) underlines their main differences and how they can be categorized as:

- The attack *goal* can differ in a few ways. The most simplistic goal would be to produce a misclassification as formulated with (2.9). This is commonly referred to as an *untargeted* attack, as its purpose is the misclassification regardless to which class. A *targeted* attack, on the other hand, is very well interested in the misclassification to a specific class. What both attack goals have in common is the general *confidence reduction* on the true class.

- The available *knowledge* of the attacker plays a significant factor in the possible methods of attack. It is generally differentiated between a *white box* and *black box* attack. In the white box scenario, the attacker has unrestricted access to the model he wants to attack. He can use the information regarding the architecture, parameters, and most importantly the produced gradients of the neural network. In the black box scenario, the attacker is generally constricted to the knowledge he receives through input-output mechanisms. Some information about the network topology and the training set can be available in the black box scenario, but the significant difference is that he has no access to the gradients of the network. In the scenario black box, the attacker can make use of the transferability characteristic of adversarial samples (Zhang and Li, 2019), by having unrestricted access to a supposedly similar neural network.

- The *perturbation* based methods are divided into *noise perturbations* and *geometric transformations*. Noise perturbations change the original image by adding white noise as can be seen in Figure 2.3. They are the most common perturbation method and will be discussed later on. Geometric transformations include rotation, reflection, and translation of images.

- The defensive strategies against adversarial attacks can be divided into *Guards* and *Defense by Design*. Guards are defensive strategies that involve either detecting the adversarial samples beforehand or diminishing the attacks' effectiveness by removing the adversarial perturbation from the sample. Defense by design involves modifying the architecture design, the loss function, or the training data to achieve robustness against attacks. Adversarial training is when adversarial samples are used as additional training data during training.

In our experiments, we use a variety of attacks and defenses. For example, in terms of defensive strategy, not only adversarial training but also the entropic open-set loss function may provide some defensive mechanisms against adversarial attacks. An elaborate documentation of different attack types can be found in surveys such as Akhtar and Mian (2018); Chakraborty et al. (2018); Serban et al. (2020). For this thesis, we will mainly introduce the performed adversarial attacks and their variations in the following sections.

2.3.3 Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method (FGSM) was proposed by Goodfellow et al. (2015) as a simplistic method to solve the optimization problem (2.9). The method utilizes the gradients of the cost function produced by the neural network when inputting a clean sample. An adversarial sample can be described by the following equation:

$$x^{adv} = x + \epsilon \operatorname{sign}\left(\nabla_x J(x, \hat{c})\right). \tag{2.11}$$

where x is the inputted clean sample, $\hat{c}$ the true label of x and $\nabla_x J(x, \hat{c})$ the gradient of loss function with the respect to x and $\hat{c}$. The $\operatorname{sign}(.)$ depicts the sign function, which maximizes the gradients of the loss with respect to the l_∞ norm. ϵ is the specific hyperparameter, which restricts the magnitude of the perturbation so that $||x^{adv} - x||_\infty \leq \epsilon$ applies. This method can generally be classified as a white box attack, as it requires the access to the gradients of the neural network.

FGSM attacks are one of the most commonly used to test for adversarial stability, as they can exploit the linearity of deep neural networks (Akhtar and Mian, 2018). Different variants of this method have since then been proposed. Miyato et al. (2018) introduced a method, which does not use the sign function, but instead normalizes the gradient loss with its l_2 norm to create the adversarial sample:

$$x^{adv} = x + \epsilon \frac{\nabla_x J(x, \hat{c})}{||\nabla_x J(x, \hat{c})||_2}. \tag{2.12}$$

Kurakin et al. (2017) proposed two notable variants of the FGSM. The first one is the *targeted* version, where the attack now targets a deliberate class. This variant likewise utilizes the loss gradients and the sign function as the FGSM, but instead of using the calculated gradient of the loss for the true label $\hat{c}$ as a course of change, a specific class c^{target} can be chosen to dictate the direction of the perturbation change. With the same variables as abovementioned, the equation for the targeted variant is formulated as:

$$x^{adv} = x - \epsilon \operatorname{sign}(\nabla_x J(x, c^{target})). \tag{2.13}$$

The second variant Kurakin et al. (2017) is the *basic iterative method*. As the name implies the method iteratively uses the FGSM to create adversarial samples, while using a clipping function *Clip(.)* to perform per-pixel clipping, so the result will remain l_∞ bound within the ϵ distance. The clipping function is formulated as follows:

$$Clip_{x,\epsilon}\{x^{adv}\} = \min\{255, x + \epsilon, max\{0, x - \epsilon, x^{adv}\}\} \tag{2.14}$$

The pixel values are represented here as integers in the range of [0,255], thus the created adversarial samples stay within this range by not overstepping the minimum and maximum values. Such a clipping function is commonly used for image perturbations. Finally, the equation for the iterative method is formulated as:

$$x_0^{adv} = x, \quad x_{i+1}^{adv} = Clip_{x,\epsilon}\{x_i^{adv} + \alpha\, sign(\nabla_x J(x_i^{adv}, \hat{c}))\}. \tag{2.15}$$

where α is the step size and $Clip_{x,\epsilon}$ clips the values of the pixels of the image based on the argument ϵ and i is the number of the *ith* iteration. The step size is generally chosen as $\alpha = 1$, so the pixel value is changed only by 1 for each iteration. As for the number of iterations Kurakin et al. (2017) chose $\min(\epsilon + 4, 1.25\epsilon)$ as they found it to be a good balance in terms of computational cost and effectiveness of the adversarial sample. Naturally, this method's potential to create adversarial samples is higher than the FGSM ones, as it uses the created adversarial samples from the previous iterations and refines it with each step. For a given perturbation range ϵ the iterative method can incrementally choose the best path available to find the strongest adversarial sample, while the FGSM has only one step as an option. The clear drawback to the iterative method is the computation time.

2.3.4 Projected Gredient Descent

Madry et al. (2019) addressed the limitations of the FGSM method and explained the multistep descent variant is essentially a projected gradient descent (PGD) on the negative loss function. Furthermore, the authors clarified how all the abovementioned methods can be viewed as specific attempts to solve the saddle point problem describing the adversarial robustness of a network, which will be discussed in the next section. The slight difference between PGD and the basic iterative method is the starting point of the perturbation, as the PGD method usually chooses a random starting point inside $x + \mathcal{S}$, while the basic iterative method uses the original image as a starting point (Madry et al., 2019). As such the PGD method can be formulated as:

$$x_{i+i}^{adv} = \prod_{x+\mathcal{S}} (x_i^{adv} + \epsilon sign(\nabla_x J(x_i^{adv}, \hat{c}))) \tag{2.16}$$

where $\mathcal{S}$ is the set of allowed perturbations. In Madry et al. (2019) they chose the starting points randomly inside the l_∞-ball of the example, with 100 steps and used $2.5\epsilon/100$ as the step size to ensure that the adversarial sample may reach the boundary of the ϵ-ball.

2.4 Adversarial Training

To achieve some kind of robustness against these adversarial attacks, defensive mechanisms must naturally be explored. Many such defensive mechanisms have been proposed as mentioned in the survey Serban et al. (2020). Adversarial training belongs to the class of defense by design strategy. The concept of neural networks being trained against adversarial attacks was introduced by Goodfellow et al. (2015), demonstrating that neural networks trained with adversarial samples during training improve adversarial stability.

While training with adversarial samples does indeed improve the adversarial stability of networks, the degree of this robustness is hard to quantify. There are many different nuances to adversarial robustness, Madry et al. (2019) uses a natural saddle point formulation to cover prior proposals on defining adversarial stability. They view adversarial robustness through the lens of an optimization problem, which is composed of an "inner maximization" problem and an "outer minimization" problem. Finding an adversarial sample that achieves a high loss is the goal of inner maximization, while the goal of finding model parameters that minimize the loss caused by such a sample is the outer minimization. Taking the notation of Wong et al. (2020) the learning problem can be summarized as followed:

$$\min_{\phi} \sum_{i} \max_{\delta \in \Delta} J((x_i + \delta), \hat{c}_i) \tag{2.17}$$

where ϕ are the parameters of the network, Δ is the threat model, $J(.)$ the loss function, δ the perturbation, x_i and $\hat{c}_i$ the *ith* sample and its true label respectively. For example the FGSM (Goodfellow et al., 2015) operates on the assumptions that the threath model $\Delta = \{\delta : ||\delta||_\infty \leq \epsilon\}$ for an arbitrary $\epsilon > 0$ and uses $\delta = \epsilon \operatorname{sign}(\nabla_x J(x, \hat{c}))$ to approximate the inner maximization function. This approximation can be significantly improved by using the PGD method, due to its iterative procedure, in exchange for the computational cost. Including adversarial samples created by the FGSM has been generally proven to be effective as adversarial training, the findings of Wong et al. (2020) even suggest that FGSM adversarial training can be just effective as the more time-consuming PGD one, while Madry et al. (2019) concluded that adversarial samples created by the FGSM with a large perturbation magnitude do not increase adversarial robustness of networks, being still vulnerable to PGD adversarial attacks, and even having a poor performance for clean images.

2.4.1 Adversarial Training in OSR

Adversarial training is mainly done in a closed-set scenario, where the adversarial samples are being used as positives during the training process, meaning they have the same label as their non-perturbed counterpart. Schnyder (2021) incorporated adversarial training in an open-set scenario, where various perturbations were performed on samples from the known class $\mathcal{C}$ and declaring these newly produced adversarial samples to be part of the known unknown class $\mathcal{B}$, treating them essentially as negatives. The training was done using the entropic open-set loss function on the LeNet++ architecture, while the MNIST dataset was used as $\mathcal{D}'_c$ and their produced adversarial counterparts as $\mathcal{D}'_b$. It should be noted that a filtering method was performed prior to adversarial samples being included in the training set. The filter includes only samples which are either classified correctly or go above a certain threshold with their targeted softmax score. The open-set performance was tested through *open-set classification rate* (OSCR) (Dhamija et al., 2018) using letters from the EMNIST dataset as unknown samples $\mathcal{D}_a$.

Taking inspiration from the abovementioned approach, Palechor (2022) conducted research in a similar manner on the ImageNet dataset. The adversarial samples were produced through the FGSM and random noise method, choosing different perturbation magnitudes ϵ and using

filtering methods such as waiting for an appropriate number of epochs and $\mathcal{D}'_c$ samples having to be correctly classified before introducing their respective adversarial samples. As for the architecture, ResNet50 with either standard categorical cross-entropy loss, entropic open-set loss, or object sphere loss from Dhamija et al. (2018) was used and compared. The research concluded that adversarial samples were helpful in OSR scenarios, noting that training to reject adversarial samples enforces better knowledge of the $\mathcal{C}$ classes, which teaches the classifier to better differentiate features from each class. Interestingly in both studies batch normalization procedures on adversarial samples were attributed to cause problems for adversarial training, generating diverging issues.

Chapter 3

Approach

In this section, the methods being used to conduct the experiments are explained. First Framework, second dataset, third neural network architecture, fourth different loss functions, fifth adversarial training, sixth evaluation methods.

3.1 Pytorch Framework

Pytorch [1] (Paszke et al., 2019) is the core framework we used for implementing the machine learning experiments. It is one of the most widely used machine learning frameworks in the research world. Same as other frameworks, Pytorch has an extensive library containing many predefined functions and optimization algorithms, while enabling us to take advantage of GPU accelerations. The major functionality of the framework is the *torch.autograd* function that provides us with automatic computations of the gradients which are used for the backpropagation during model training. Particularly in our case the created loss gradients are used to create adversarial samples, which will be used during training and testing procedures.

3.2 Dataset

As for the dataset, we use the MNIST and EMNIST datasets. MNIST LeCun (1998) is a widely used dataset in machine learning due to its simplicity and small size. It serves well as an introductory dataset for training and evaluating models. The MNIST dataset consists of 70'000 images, which are generally divided into a training set of 60'000 and a testing or validating set of 10'000 images. Each image is a handwritten digit representing a number between 0 and 9 and are respectively labeled as such. The digits are all centered, grayscaled, and have the size of 28x28 pixels. These 10 numbers compose the classes of the MNIST dataset and represent our known classes $\mathcal{C}$ in all our experiments.

The EMNIST dataset (Cohen et al., 2017) is an extension of the MNIST dataset and includes handwritten letters. It has over 800'000 characters and various splitting options. One of those options being *letters*. This option provides us with a balanced set of 140'000 uppercase and lowercase letters split into 26 classes. We furthermore divide this set manually through their label index into two different sets. The first set will be utilized as the negative classes $\mathcal{B}$. These samples will be involved during training and testing for some cases, as will be elaborated upon in the following sections. The other set provides us with the unknown classes $\mathcal{A}$, which will be only used in the

[1] https://pytorch.org/

evaluation phase, as their purpose is to represent samples of classes that are truly unknown to all networks.

3.3 Architecture

For our training we use a variant of the LeNet architecture proposed by LeCun et al. (1998). The architecture is part of the VAST[2] library which contains various algorithms and software tools used in works from the members of the Vision And Security Technology (VAST) Lab. The architecture consists of 2 different convolutional layers, a pooling layer, an activation function, and 2 linear transformations.

The first convolution has 1 input channel, and 20 output channels, while the second one has 20 input channels and 50 output channels. Both convolutions apply the typical 2D 5x5 filtering operation and have a stride of 1 and padding of 2. The activation function is a standard ReLU function and is applied twice, each time after a convolutional layer. The pooling is done through a 2D max-pooling operation with the kernel size of 2x2, stride of 2, and padding of 0 and is applied both times after the activation function. The output is then rearranged, creating a flattened feature map with a total of 2450 features as input for the first linear transformation function. Lastly, the first linear transformation function outputs 500 features for the second linear transformation function, which then outputs 10 feature values per sample. These 10 values are the logit values, representing the known 10 classes. It is to be noted that the forward pass for the background approach is slightly modified to have 11 classes, as such its final output has 11 instead of 10 logit values.

Algorithm 1 Forward Pass Function

Require: $batch \quad x$
1: $x = Pool(ReLu(Conv1(x)))$
2: $x = Pool(ReLu(Conv2(x)))$
3: $x = x.view(-1, Conv2.outchannels * 7 * 7)$
4: $x = LinearTransformation1(x)$
5: $x = LinearTransformation2(x)$
6: $return \quad x$

Contrary to the prior work of Schnyder (2021) and Palechor (2022) which also involved adversarial training in an open-set scenario, this architecture features no batch normalization. Both parties attributed diverging issues during adversarial training due to batch normalization (Ioffe and Szegedy, 2015) being applied to their output of the convolutions. This aspect will be discussed during the adversarial training section further on.

[2] https://github.com/Vastlab/vast

3.4 Loss Functions

For the thesis, we are using three different approaches regarding loss functions. The first one is the standard categorical cross-entropy loss on top of softmax-activated logits, which is commonly referred to as the *Softmax loss* mentioned in Section 2.2.1. The second loss function is the same as the first one but incorporates an additional background class as discussed in Section 2.2.2, which shall be referred to as *BG*. The third loss function is the entropic open-set loss by Dhamija et al. (2018). For *EOS*, we use the implementations based on the VAST library[2]. The following paragraphs describe the most important implementation details regarding the loss functions:

- The Softmax loss is the *CrossEntropyLoss()* function provided by the Pytorch[1] framework and is only used in combinations with the MNIST dataset. The classes in use for this loss function are only the known classes $\mathcal{C}$. It serves as a benchmark for the other loss functions and in the thesis we use it as an approach where no negative/unknown classes $\mathcal{U}$ are incorporated during the training process.

- The BG loss uses the same *CrossEntropyLoss()* but has a slight adjustment to it. While it incorporates the same 10 MNIST digit classes as known classes $\mathcal{C}$, it is also able to handle negative/unknown classes $\mathcal{U}$. These negatives are either letters from the EMNIST set mentioned in Section 3.2 or adversarial samples which will be discussed in the following Section 3.5.2. The known samples have the same label as the number they represent, meaning an image of "1" has the label "1", while the negative samples all receive the same label of "10".

- The EOS loss is solely extracted from the VAST library[2]. Same as the BG loss, it treats the 0-9 MNIST digits as known classes $\mathcal{C}$ with their target label being the same number they represent, also it can handle negative/unknown classes $\mathcal{U}$. Contrary to the BG loss, the EOS loss does not incorporate an additional class, instead, all negative samples receive the label of "-1" (the number is arbitrary and could have been easily chosen to be $10 = |\mathcal{C}| + 1$ as the BG approach). The loss is calculated as described in Section 2.2.3.

3.5 Training

We establish two different approaches for training. The first one builds the baseline and the second one involves training with adversarial samples.

3.5.1 Part 1: Baseline Training

In this part, we train with the three different loss functions (Softmax *S*, Softmax with a background class *BG*, and entropic open-set *EOS*) without the use of any adversarial samples. The MNIST dataset is divided into the training and validation sets and is used as $\mathcal{D}_c$ and $\mathcal{D}'_c$. The EMNIST letters are also divided into the training and validation sets and serve as unknown samples $\mathcal{D}_u$ and $\mathcal{D}'_u$ respectively for the BG and EOS approaches. Furthermore, the first 13 letters [Aa-Mm] are used as negative samples $\mathcal{D}'_b/\mathcal{D}_b$, and the latter 13 letters [Nn-Zz] are used as unknown samples $\mathcal{D}_a$. In Table 3.1 the sizes are displayed for the training and validation sets.

Our training procedure is common practice as we utilize the forward pass, loss calculation, backward pass, and optimizer step for each epoch with training samples $\mathcal{D}'_c$ and $\mathcal{D}'_b$. For the validation procedure, we use the confidence scores of the validation samples $\mathcal{D}_c$ and $\mathcal{D}_b$. The confidence scores are a good indicator of how well a model performs for the given samples. For

Table 3.1: BASELINE TRAINING DATASET. *These are the samples sizes of the baseline training and validation datasets*

	Train		Validate	
Loss	$\mathcal{D}'_c$	$\mathcal{D}'_b$	$\mathcal{D}_c$	$\mathcal{D}_b$
Softmax	60'000	-	10'000	-
BG	60'000	57'600	10'000	9'600
EOS	60'000	57'600	10'000	9'600

the Softmax approach, the usual softmax score of the true label represents the confidence for a sample x:

$$conf_{S/BG}(x) = S_{\hat{c}}(x) \tag{3.1}$$

The same confidence calculation (3.1) applies for the BG approach, as in our case the unknown classes are treated as one separate class, essentially considered as one of the known classes $\mathcal{C}$. As for the EOS approach, the confidence calculation is the same for samples of known classes $\mathcal{D}_c$, but different for unknown samples $\mathcal{D}_b$ and can be described as follows:

$$conf_{EOS}(x) = \begin{cases} S_{\hat{c}}(x) & \text{if } x \in \mathcal{D}_c \\ 1 - \max S_c(x) + \frac{1}{|\mathcal{C}|}, & \text{if } x \in \mathcal{D}_b \end{cases} \tag{3.2}$$

where $S_{\hat{c}}(x)$ is the softmax score for the true class $\hat{c}$ and $\max S_c(x)$ the maximum softmax score of any class c. In our case where there are 10 known classes, it also follows that $\frac{1}{|\mathcal{C}|} = 0.1$. As such an unknown sample $\mathcal{D}_b$ receives a maximum confidence of 1.0 if the probability entropy is at its maximum giving each class c a softmax score $\max S_c(x) = 0.1$. The confidence calculations are provided by the VAST library and were already utilized in Schnyder (2021) and Palechor (2022). Finally, the mean value from the confidence scores of all validation samples is calculated, depending on the loss function:

$$CTL = \sum_{i=1}^{n} conf(x_i) \tag{3.3}$$

where n is the number of samples and $conf(x_i)$ the confidence score for the ith sample x depending on the loss function. In our experiments, we perform this validation calculation for each epoch with the validation samples $|\mathcal{D}_c + \mathcal{D}_b| = n$. This validation score dictates when a model shall be saved, as we want only the best model to be saved for evaluation purposes. We name this mean confidence calculation to avoid confusion: **Confidence of the True Label (CTL)**, as it will be used in alongside another confidence metric, discussed later on.

3.5.2 Part 2: Adversarial Training

For the adversarial training, we have different approaches depending on which loss function is being used. While the construction of adversarial samples for each approach remains essentially the same, how they are applied in training is altered depending on each loss function. Additionally to the different loss functions, we incorporated two different methods to create adversarial samples during training. The first one are random *noise* perturbations and the second one are *FGSM* perturbations mentioned in Section 2.3.3. The construction of adversarial samples through FGSM is done with the help of the Advertorch libary[3] (Ding et al., 2019), which is a toolbox con-

[3]https://github.com/BorealisAI/advertorch

taining various implementations of adversarial attacks and defenses. The noise perturbation were created by using the *torch.rand()* and *torch.rand()* functions of Pytorch[1].

A batch of adversarial samples is created immediately after a batch of clean samples went through the forward function. Thanks to the Autograd functionality the FGSM can take advantage of the calculated gradients and create adversarial samples (2.11), which are then passed through the forward function. Afterward, the clean and adversarial samples are concated together and their loss is calculated through one of the loss functions. Lastly, the backward pass is done on mean loss and the optimizer step is performed.

The different loss functions from Section 3.4 need different labeling for the adversarial samples. Let x be a sample with the true class label $\hat{c}$ and x^{adv} the adversarial sample constructed from x with the true class label c^{adv} with the *assign()* method:

1. *Softmax*: x^{adv} are positives and have the same class label as their clean counterpart $c = \hat{c}$
2. *BG*: x^{adv} are negatives and receive the class label $c = 10$
3. *EOS*: x^{adv} are negatives and receive the class label $c = -1$

Furthermore let ϕ be the current network parameters such that $J(\phi, x, \hat{c})$ and $J(\phi, x^{adv}, c)$ are the losses of the clean and adversarial samples respectively. As such we have a variant of the loss used in the adversarial training experiments of Goodfellow et al. (2015):

$$\tilde{J}(\phi, x, c) = J(\phi, x, \hat{c}) + J(\phi, x^{adv}, c) \tag{3.4}$$

Same as in the prior studies a *filter* method is used during the training process. The filter permits only certain clean samples to be perturbed into adversarial samples and reintroduced to the forward pass. The filter is simple and straightforward: Only samples x which have a softmax score $S_{\hat{c}}(x)$ for their target class $\hat{c}$ over a certain threshold point, will get perturbed to adversarial samples x^{adv} and reintroduced into the training process. This serves naturally to uphold the principle of adversarial samples, being that their respective clean sample counterpart first has to be classified correctly by the network for them to be able to fool them.

Performing this filter generally introduced almost all clean images to be perturbed as adversarial samples. In the case of Softmax, the size of known training samples $\mathcal{D}'_c$ doubled, whereas the open-set approaches received an almost equal sample size of adversarial samples for them to be included as negatives $\mathcal{D}'_b$ during training. The validation procedure remains the same as described in Section 3.5.1. This is since we want to uphold the purpose of the respective networks. While the closed-set network with the Softmax loss shall only be validated for closed-set purposes, the open-set networks with EOS and BG losses shall still be validated based on their open-set capabilities. The $\mathcal{D}_c$ and $\mathcal{D}_b$ are still the same validation digits and letters described in Section 3.5.1 and can be seen in Table 3.2. The following algorithm describes the steps for our adversarial training:

Table 3.2: ADVERSARIAL TRAINING SAMPLES. *These are the training and validation datasets used during adversarial trainig. Same letters are used during validation, otherwise only the MNIST dataset is used.*

	Train		Validate	
Loss	$\mathcal{D}'_c$	$\mathcal{D}'_b$	$\mathcal{D}_c$	$\mathcal{D}_b$
Softmax	$\approx$ 120'000	-	10'000	-
BG	60'000	$\approx$ 60'000	10'000	9'600
EOS	60'000	$\approx$ 60'000	10'000	9'600

Algorithm 2 Adversarial Training

Require: batch B, Dataset, training epochs N_{epochs}, model M_ϕ loss function J, $filter()$ function, $perturbation()$ function, optimizer, $assign()$ function.

for epoch = 1, N_{epochs} **do**
  for batch $B \subset$ Datatset **do**
    Get images and labels in $B : x_B, \hat{c}_B$
    Get prediction: $logits_B = M(x_B, \hat{c}_B)$
    if AdvTraining = True **then**
      Filter samples: $x_f, \hat{c}_f = filter(output, \hat{c}_B)$
      Assign new labels: $\hat{c}_f = assign(\hat{c}_f)$
      Calculate adversarial samples: $x_{adv} = perturburbation(x_f, \hat{c}_f)$
      Get prediction from adversarial samples: $logits_{adv} = M(x_{adv}, \hat{c}_f)$
      Concatenate logits: $logits_B = cat(logits_B, logits_{adv})$
      Concatenate true labels: $\hat{c}_B = cat(hatc_B, \hat{c}_f)$
    end if
    Calculate loss: $l = J(\phi, logits_B, \hat{c}_B)$
    Perform Backward pass on loss: $backward(l)$
    Do optimizer step: $optimizer.step()$
  end for
end for

The implementation of a waiting epoch is optional and was not used during the experiments. The model is always saved during the first 5 epochs, afterwards, it is only saved when the overall confidence tested through the validation set rose compared to the maximum of the prior ones. This is done due to the high oscillating probability during the first epochs and the assumption that the validation testing is indicative of the general classification of the model.

3.6 Evaluation

In this thesis, we evaluate different models based on their OSR prowess and their adversarial stability. Both fields need their respective evaluation methods. In the following subsections, the respective evaluation methods used during the experiments are documented.

3.6.1 Open-Set Capabilities

To evaluate the open-set capabilities of the networks we utilize the OSCR (Dhamija et al., 2018) described in Section 2.2.4. For the positive samples, we use the same samples as in the validation set $\mathcal{D}_c$. As for negative samples we have two different sets, the first one is the validation set of the first letters, which are the same negative validation samples for the open-set approaches. The second samples are the validation set of the latter 13 letters, which have not been seen by any of the networks prior $\mathcal{D}_a$. As such we calculate two OSCR curves per network to evaluate its OSR prowess given the samples in Table 3.3.

While the open-set letters trained models should have a favorable bias against the adversarial trained models for the "known unknown"/"negatives" OSCR evaluation, the bias should be less distinct in the "unknown unknown"/"unknowns" OSCR evaluation, as these specific letters have not been shown to any of the networks.

Table 3.3: OSCR SAMPLES. *These are the samples used for the OSCR evaluations*

	known unknown/negatives		unknown unknown/unknowns	
Loss	$\mathcal{D}_c$	$\mathcal{D}_b$	$\mathcal{D}_c$	$\mathcal{D}_a$
Softmax	10'000	9'600	10'000	10'400
BG	10'000	9'600	10'000	10'400
EOS	10'000	9'600	10'000	10'400

3.6.2 Adversarial Stability

Adversarial stability is tested through confidence scores. We use two different kinds of confidence scores, due to the diversity of the loss functions and to illustrate the probability distribution of an adversarial sample. The evaluation is performed with the 60'000 samples of the known classes $\mathcal{D}'_c$ which are used during training by every network. We take correctly classified samples $\mathcal{D}'_c$ and perturb them with different perturbation magnitudes ϵ into adversarial samples. As the samples are all positives, the correct classification is derived from the standard softmax score of a sample (2.4). We chose the maximimum perturbation magnitude ϵ in such a way that the pertubed images still remain recognizable for a human. The confidence scores are plotted against the perturbation magnitudes, giving us a visual representation of how the confidences change through perturbation.

The first confidence calculation we utilize is the same equation we used during the validation process. The mean confidence of the true class is calculated differently for each loss function as discussed in Section 3.5.1. These confidence scores are plotted against the perturbation magnitudes ϵ as to see what kind of impact a perturbation has on the confidence of the overall true labels. Through this plot, we receive the values representing how confident the model is that these adversarial samples still belong to the true class $\hat{c}$. Madry et al. (2019) uses the same plotting scheme but preferred to use accuracy instead of the confidence metric.

The second confidence scoring method focuses on the confidence of all the labels beside the true labels. This is interesting since while the score of the true label may be lowered through the perturbation we still do not have information of the other label scores. There may be cases where the other labels receive an equal bump in their confidence score, while in other cases only a specific label receives the majority of the confidence redistribution. The confidence is calculated as follows:

$$conf_{CNL}(x) = \max_{c \in \mathcal{C}} S_c(x), \quad \text{for } c \neq \hat{c} \tag{3.5}$$

naturally the mean score over all the samples x is taken as descrbied in (3.3) and likewise we term this mean confidence metric: **Confidence of the Non-True Label (CNL)**. With this metric, we get an indicator of the general probability distribution. The maximum score of the other classes illustrates how strong a model tends to misclassify, which can be interpreted differently depending on the approach as we will see in the experiments. It should be noted, that for the BG approach we do not include the tenth label which is the background label.

We perform a variety of attacks with different perturbation magnitudes, by utilizing Advertorch[3]. Due to cost of time we perform mainly FGSM perturbations. We use both the targeted and untargeted version of it as mentioned in Section 2.3.3. Instead of using the suggested method of creating targeted samples by Advertorch[3], we implemented our own design. The targeted labels are chosen by incrementing the true label of a sample and then performing the modulo opertion. In our case, where there are 10 classes the following describes the targeted FGSM:

$$\hat{c} \in \{0, ..., 9\} : c^{target} = (\hat{c} + 1) \mod 10. \tag{3.6}$$

where $\hat{c}$ is the true label of the correctly classified clean sample and c^{target} is the resulting target

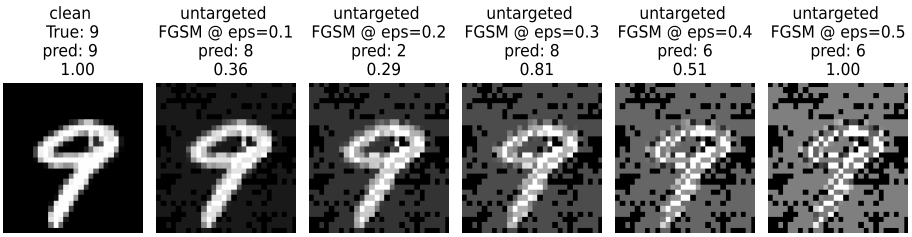


Figure 3.1: PERTURBATION SPECTRUM. *The number 9 is perturbed through the FGSM with different pertrubation magnitudes ϵ. Standard Softmax network classifies the label and maximium softmax score.*

label inputted to create an adversarial sample seen in (2.13). The same targeting scheme is applied for the PGD attacks which are used at the last experiments as the creation procedure is very time costing. Otherwise, we used the default implementations provided by the Adertorch framework[3]: *FGSM = GradientSignAttack(), PGD = LinfPGDAttack()*. Figure A.1 illustrates the different image perturbations at a given ϵ and the respective confidence values reached by the baseline Softmax trained model.

Chapter 4

Experiments

This chapter details the performed experiments and their respective results. The experiments can be divided into three parts. In the first part, we train our model without any adversarial samples. The negative samples for the open-set approaches are obtained from the letters as described in Section 3.5.1. These models serve as baselines and benchmarks for later experiments. The second part involves adversarial training with different ϵ, which are compared against each other. In this part, we search for the most suitable ϵ a specific network should be trained with. For the third part, we compare the best-performing adversarial trained model and their respective baseline counterpart in more detail. The following parameters are common throughout all three experiments:

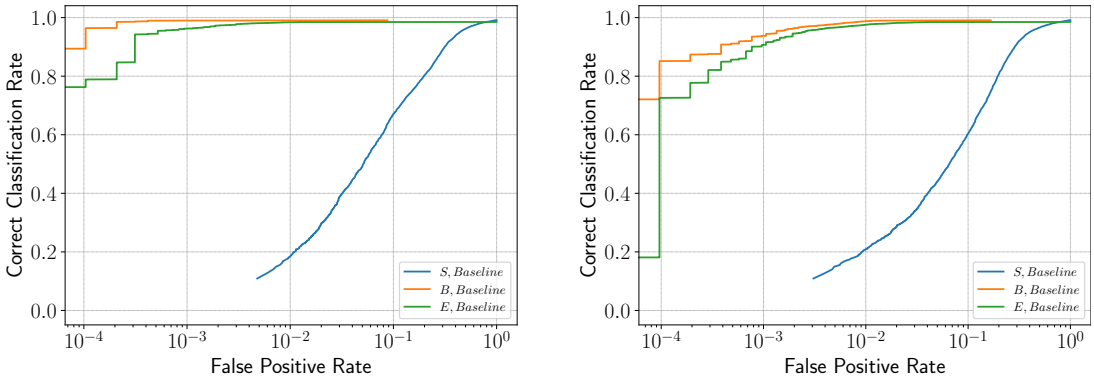
- Epochs: 100
- Optimizer: Stochastic Gradient Descent (SGD)
- Learning rate: 0.01
- Batch size: 128
- Seed: 0

4.1 Part 1: Baseline Experiments

For these experiments we use the tained the models using the methods as described in Section 3.5.1.

4.1.1 OSR Evaluation

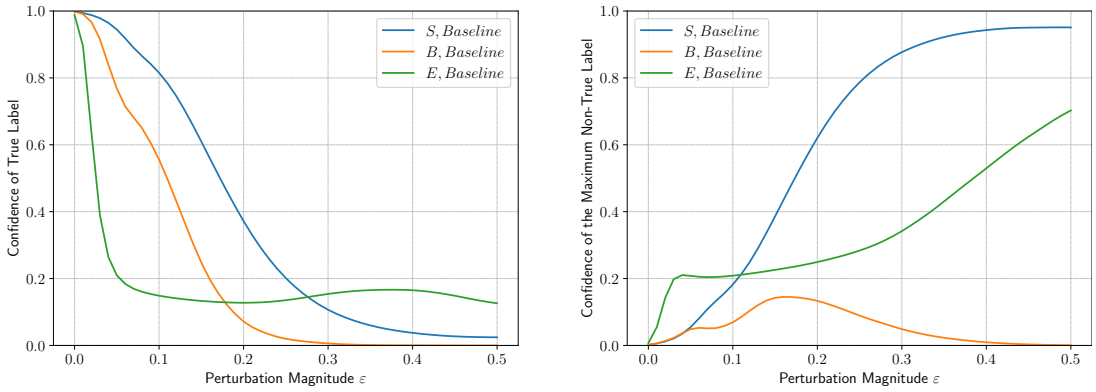
The open-set recognition capabilities are evaluated as described in Section 3.6.1. As can be seen by both subfigures in Figure 4.1, both the entropic open-set approach and the softmax with a background class approach (B) vastly outperform the standard Softmax (S) approach. This comes as no surprise and is consistent with the results and finding in Dhamija et al. (2018). A bit unexpected is the results of the BG approach, as it outperforms the EOS (E) approach in both scenarios and throughout the whole classification scope. Naturally, the performance gap between them is not as large as with the closed-set approach. Though these results stand in conflict with the findings of Dhamija et al. (2018), they are not entirely comparable. The difference may be caused due to the choice of the network (LeNet vs. LeNet++) or due to the relatively small data size and difference of the training and testing sets. Lastly, we see that in both open-set approaches,



(a) negatives (b) unknowns

Figure 4.1: OSCR BENCHMARK. *Figure (a) is the OSCR curve gained from using the first 13 letters as negatives. In figure (b) we used the latter 13 letters, which were never seen before by any network and truly unknown.*

the performance for known unknown/negative samples is higher than with the unknown unknown/unknown samples, as can be seen when comparing both top left corners, where the CCR is relatively smaller given the $FPR \approx 10^{-4}$. The performance drop is not substantially great and indicates that training with specific letters as negatives to be an effective strategy to separate distinct different letters from the known numbers.



(a) CTL (b) CNL

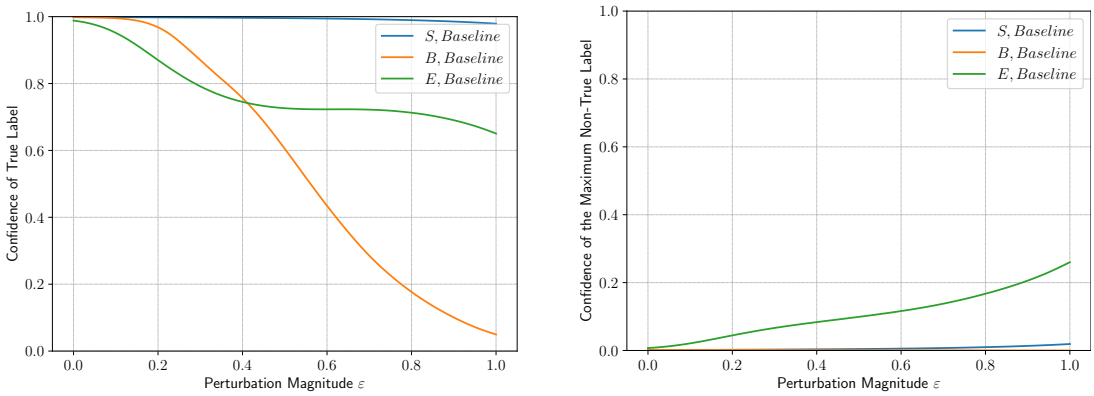
Figure 4.2: UNTARGETED FGSM ATTACK. *Figure (a) are the CTL scores and Figure (b) are the CNL scores for and untargeted FGSM attack with the magnitude ε*

4.1.2 Adversarial Stability Evaluation

The adversarial stability evaluation is done as described in Section 3.6.2. We use three different attacks to test the stability of the models, namely: untargeted FGSM, targeted FGSM, and Noise. The untargeted and targeted FGSM attacks are the same as described in Section 3.6.2. The Noise attack consists of applying a random noise perturbation to a clean image and clipping it. This is done through the *torch.clamp()* and *torch.rand()*[1]. The attack's strength correlates with the perturbation magnitude ε, meaning the bigger the ε, the farther apart the resulting pixel can be from the original one.

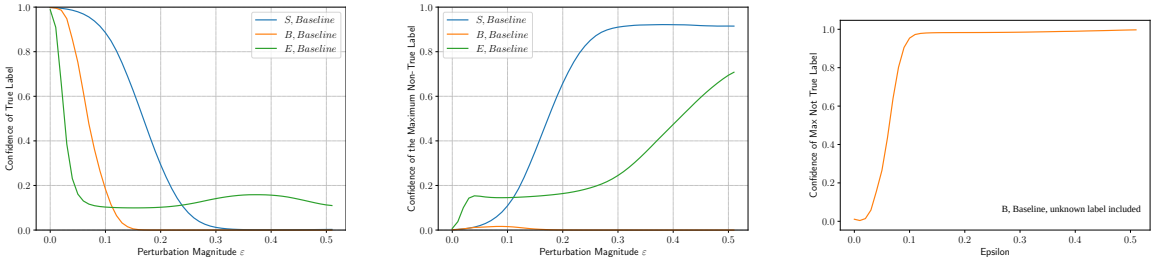
For untargeted FGSM attacks the EOS approach has the fastest true label confidence (CTL) drop compared to the other approaches as can be seen in Figure 4.2a. This means that with a relatively low perturbation magnitude ε the model's calculated confidence score of the true label dips drastically. On the other hand, the confidence scores of the other labels (CNL) do not seem to rise exceedingly as can be seen from Figure 4.2b. Until $\varepsilon = 0.3$ the EOS approach retains a relatively low maximum confidence score for the non-true labels. This indicates that the confidence scores are well distributed and as such the model expresses such adversarial samples as unknown.

The softmax model has the most gradual true label confidence drop and steepest confidence score rise of the other labels. This means that it is reasonably robust against adversarial attacks until $\varepsilon = 0.1$, but starts to misclassify samples with high confidence afterward. The BG approach trained model takes the middle ground in terms of the true label confidence drop but uniquely does not raise the confidence score of the other labels as can be seen by the flat curve in Figure 4.2b and rising CNL scoring with the unknown label included in Figure 4.4c. This means that the confidence redistribution mainly went to the background class, and as such the model classifies the samples as unknown likewise as the EOS approach. Both the EOS and BG models display partly this distinct defense mechanism, while their adversarial stability fair worse in the tradional sense, as their CTL scores drop faster compared to the standard Softmax approach.



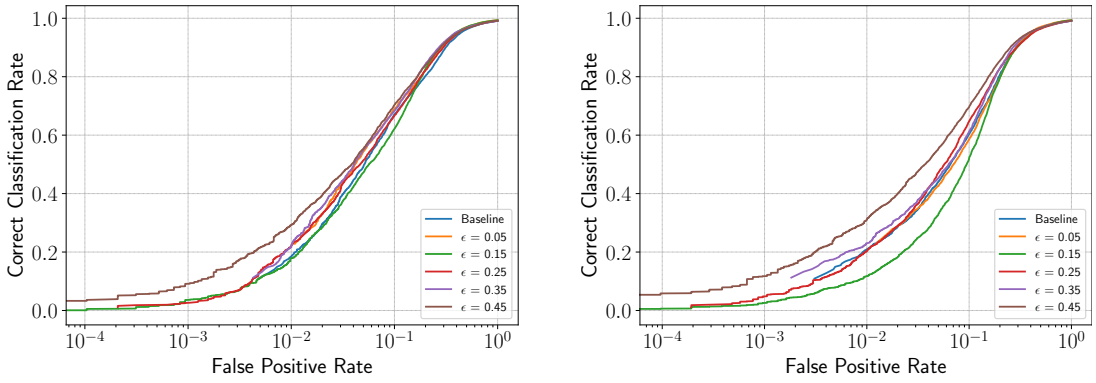
(a) CTL (b) CNL

Figure 4.3: NOISE ATTACK. *Adversarial stability evaluation against noise pertrubation attacks.*



(a) CTL (b) CNL (c) CNL with unknown label

Figure 4.4: TARGETED FGSM ATTACK. *Adversarial stability against targeted FGSM attacks*



(a) negatives (b) unknowns

Figure 4.5: OSCR OF FGSM SOFTMAX TRAINED NETWORKS. *Figure (a) is the OSCR curve gained from using the first 13 letters as negatives. In figure (b) we used the latter 13 letters, the "unknowns". The networks were trained with the FGSM method where ϵ denotes the perturbation magnitude of the adversarial samples during training.*

4.2 Part 2: Adversarial Training

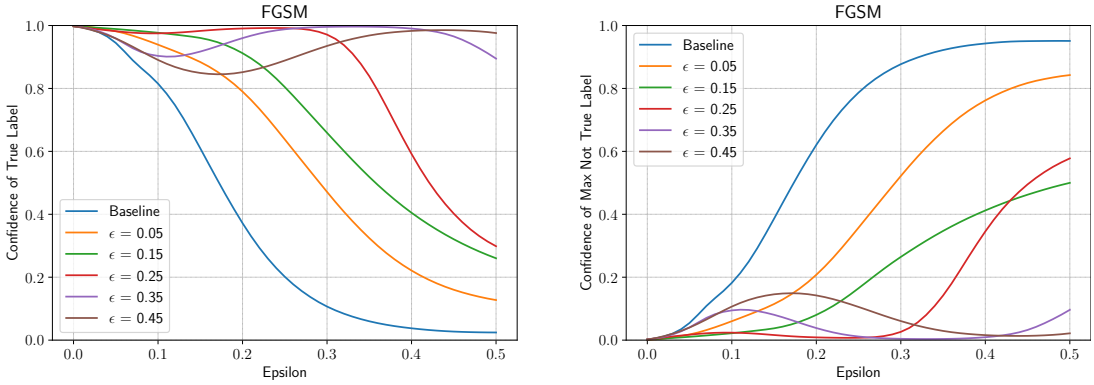
The adversarial training is performed as described in Section 3.5.2. We utilized noise perturbations and FGSM. Different perturbation magnitudes were used during training and these are plotted and compared against each other to find the most suitable magnitude for different scenarios. Noise trained models were mainly used as benchmarks to compare against FGSM models. The evaluation procedures are done as described in Section 3.6. The baseline models are added additionally to the plots to serve as general baselines and reference points.

4.2.1 Softmax

We use the traditional adversarial training for the Softmax loss approach as described in Section 3.5.2, as such the additional samples are used as positives.

In Figure 4.5 we see the open-set capabilities of the adversarial trained Softmax networks. Through the results of the different perturbation magnitudes ϵ we can recognize a few tendencies on how adversarial trained networks may behave compared to their baseline counterpart, here denoted as *Baseline* in the legends. Adversarial trained models trained with a very small perturbation magnitude, here $\epsilon = 0.05$, behave very similar to their baseline counterpart and achieve almost the same results as can be seen clearly in Figure 4.5b. Models trained with adversarial samples of a higher perturbation magnitude, here $\epsilon = 0.45$ seem to have better recognition performance in every regard, as it has a better CCR every FPR compared to the baseline model. While moderate perturbation magnitudes, here $\epsilon = 0.15$, generally fair worse in terms of open-set capabilities. The performance differences are not extreme but indicate that adversarial training may influence open-set performance.

As expected, adversarial training influences the adversarial robustness. The confidence scores of all adversarial trained models have a more gradual decline compared to their baseline counterpart seen in Figure 4.6a. Also noticeable is the confidence exchange between labels. The lost confidence of the true label in Figure 4.6a correlates well with the maximum confidence of a specific non-true label seen in Figure 4.6a. This indicates that confidence redistribution focuses



(a) CTL (b) CNL

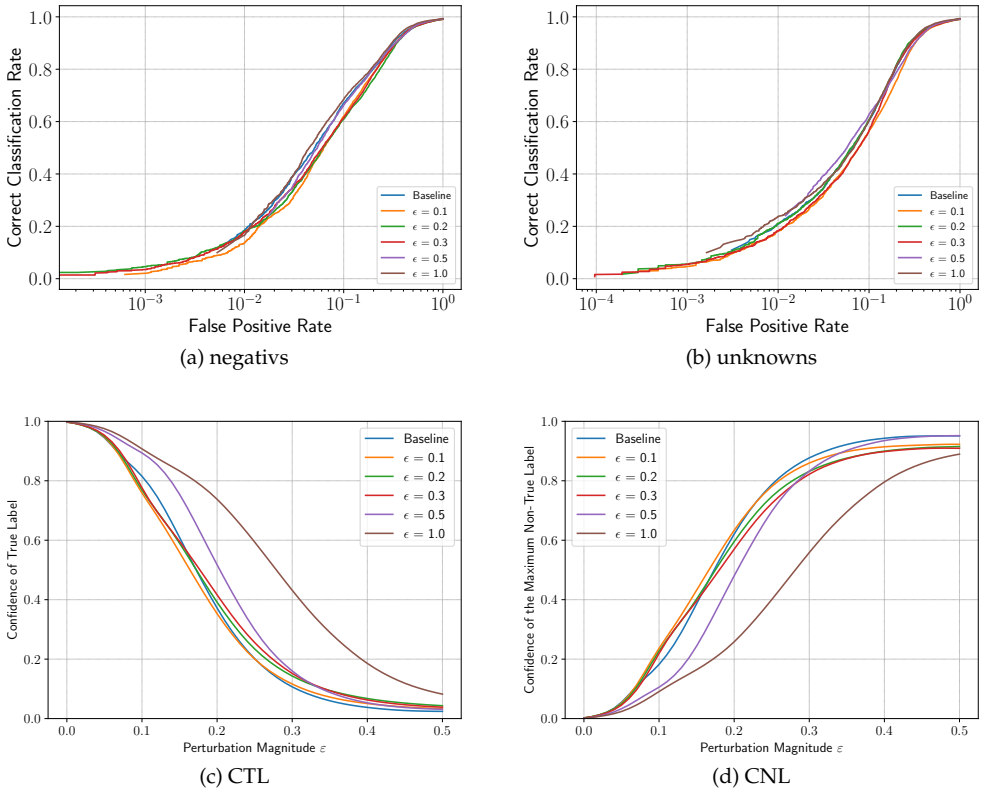
Figure 4.6: ADVERSARIAL STABILITY OF FGSM SOFTMAX TRAINED NETWORKS AGAINST UNTARGETED FGSM. *Figure (a) are the confidence scores of the true label (CTL) for adversarial samples created by the FGSM for a given perturbation magnitude. In figure (b) the maximum confidence score beside the true label (CNL) are depicted for the same samples.*

mainly on one label. Another interesting observation is the fact that adversarial trained models with samples of a specific magnitude perturbation ϵ are most resistant against the same perturbation magnitude ϵ of the adversarial evaluation samples. For example, the model with $\epsilon = 0.35$ has a gradual confidence decline, reaching a local minimum around $\varepsilon \approx 0.12$ and having a local maximum around $\varepsilon \approx 0.35$ for the adversarial evaluation samples. This reaffirms that adversarial training with specific adversarial perturbation magnitude samples is most robust against the same magnitude.

The results from the noise trained networks are less interesting as their performance difference are marginal and show fewer tendencies as seen in Figure 4.7. Subfigures (a) and (b) indicate that smaller perturbation magnitudes haave a negative impact on the open-set capabilities, while larger noise perturbation samples added during train seem to improve it. The differences in results are marginal for the size of the evaluation samples. Similar tendencies for adversarial stability can be observed in subfigures (c) and (d), where networks trained with $\epsilon < 0.5$ are even more prone to adversarial attacks with $\varepsilon < 1.5$, while networks trained on $\varepsilon \geq 0.5$ seem to improve adversarial stability. In conclusion, the network trained on FGSM with $\epsilon = 0.45$ seems to have the best performance in both open-set capabilities and adversarial stability for the Softmax approach.

4.2.2 BG

Prior research and our starting experiments implied using very small perturbation magnitudes for the adversarial samples as negatives deliver better results. As such we focused on the open-set approaches with relatively lesser perturbation magnitudes during testing compared to the Softmax approach. The open-set performance can be seen in Figure 4.8. The OSCR curves indicate that smaller perturbation magnitudes generally achieve better results. The outlier here is the network trained on the adversarial samples with perturbation magnitude $\epsilon = 0.3$, as it performs the worst for every FPR and CCR. For an unknown reason, this specific FGSM perturbation caused diverging issues as will be discussed later on. As expected, the baseline network outperforms even the best adversarial trained model. This may be attributed to the negative samples in



(a) negativs (b) unknowns

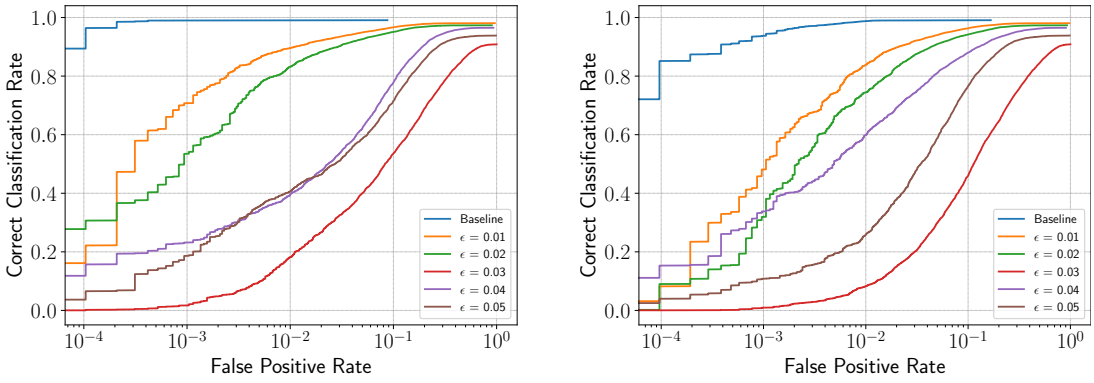
(c) CTL (d) CNL

Figure 4.7: EVALUATION OF NOISE TRAINED NETWORKS AGAINST UNTARGETED FGSM. *Figures (a) and (b) are the OSCR curve of noise trained networks. Figures (c) and (d) are the adversarial stability curves.*

the baseline model, as the letters represent a better choice for recognizing other different letters. Nevertheless, the performance of the models with the magnitudes $\epsilon = 0.01 and \epsilon = 0.01$ achieve remarkable results in terms of open-set capabilities.

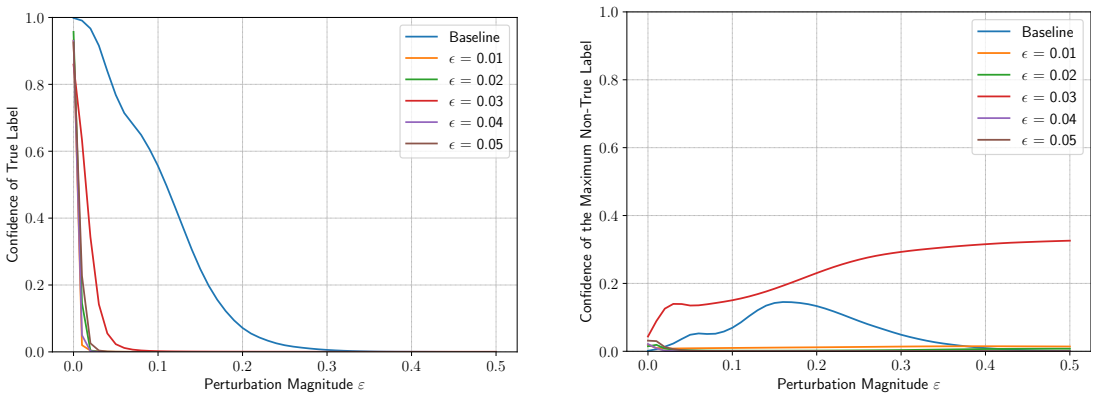
The results from the adversarial stability evaluations seen in Figure 4.9 show clear tendencies. While the confidences of the true label in (a) plummet almost instantly to 0.0, the confidences of non-true labels do not rise at all, as can be seen in (b). This means that all the confidence is re-distributed to the additional background label. At first glance, the adversarial training seemingly produced more susceptible networks to adversarial attack, as its traditional adversarial stability seen in the Softmax experiments Section 4.2.1 decreased tremendously. The networks are indeed more prone to adversarial attacks, but instead of misclassifying them to another known label, the networks classify adversarial samples as unknowns, making it an intriguing idea for a defensive mechanism.

The network trained with adversarial samples of the magnitude $\epsilon = 0.03$ is again the outlier in these evaluations. This may indicate that this form of adversarial training comes with the risk of diverging issues. Networks trained with random noise perturbations performed generally worse in terms of open-set capabilities compared to the FGSM trained ones but still achieved better OSCR scores than the regular Softmax approach variants. Interestingly, training very small noise perturbations have the same effect in terms of increasing adversarial susceptibility, achieving similar results to their FGSM trained counterpart.



(a) negatives (b) unknowns

Figure 4.8: OSCR OF FGSM BGSOFTMAX TRAINED NETWORKS. *Figure (a) is the OSCR curve gained from using the first 13 letters as negatives. In figure (b) we used the latter 13 letters, the truly unknowns. The networks were trained with the FGSM method where ϵ denotes the perturbation magnitude of the adversarial samples during training.*

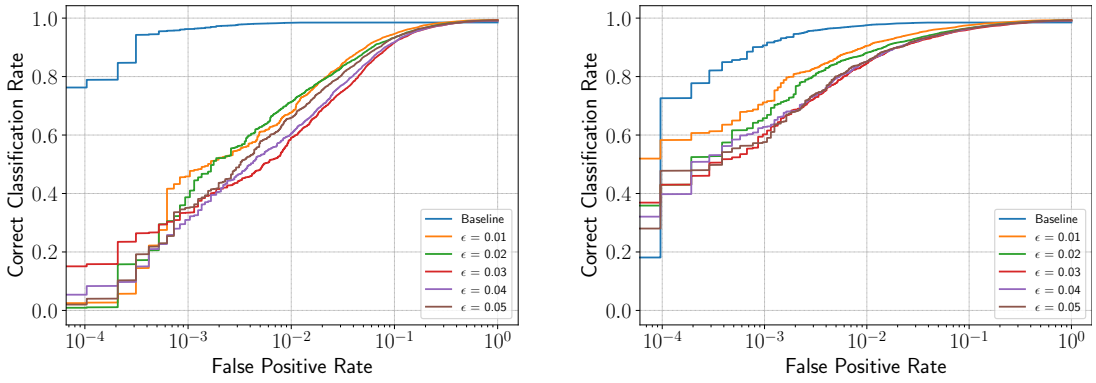


(a) CTL (b) CNL

Figure 4.9: ADVERSARIAL STABILITY OF FGSM BG TRAINED NETWORKS AGAINST UNTARGETED FGSM. *Figure (a) are the confidence scores of the true label for adversarial (CTL) samples created by the FGSM for a given perturbation magnitude ε. In figure (b) the maximum confidence score beside the true label (CNL) are depicted for the same samples.*

4.2.3 EOS

Finally, we performed the same experiments as in Section 4.2.2 for the EOS approach, which show similar tendencies depending on the usage of perturbation magnitude for adversarial samples during training. The OSCR curves in Figure 4.10a and Figure 4.10b illustrate that very small FGSM perturbed samples added during training achieve the best results. The network trained with $\epsilon = 0.01$ performs the best out of all the adversarial trained models and even outperforms the baseline model for a very small FPR $< 10^{-4}$ for the latter letters. The open-set capabilities seem to differ substantially depending on which letters are used for evaluation, even though the

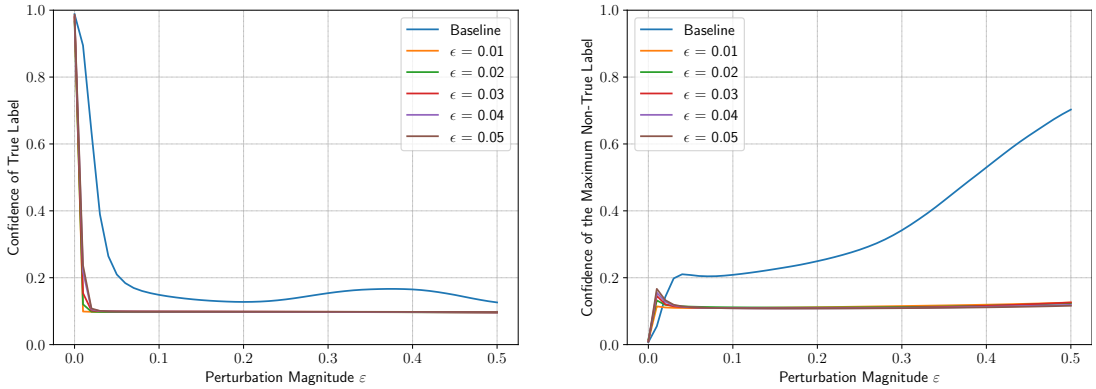


(a) negatives (b) unknowns

Figure 4.10: OSCR OF FGSM EOS TRAINED NETWORKS. *Figure (a) is the OSCR curve gained from using the first 13 letters as negatives. In figure (b) we used the latter 13 letters, which are truly unknowns for every network.*

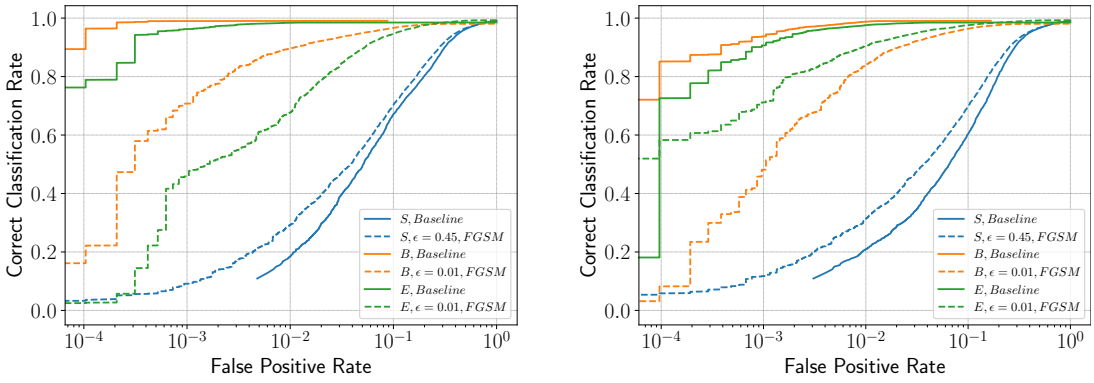
adversarial trained models have not been trained with the letters like the baseline model.

The adversarial stability tests in Figure 4.7 show the same adversarial susceptibility effect as mentioned in Section 4.2.2. The confidence scores both in (a) and (b) drop radically to the 0.1 mark, which represents the maximum entropy for the probability distribution of the labels. This indicates clearly that all adversarial trained models recognize the adversarial samples as unknown. The baseline model also displays this defensive mechanism, although it is with less effectiveness, as the confidence scores are just below the 0.2 mark in (a) and in (b) jump first off at the 0.2 mark, then gradually increasing, indicating a false classification by the model. The network trained with samples of the smallest perturbation magnitude $\epsilon = 0.01$ applies this defensive mechanism with the starkest degree, as it is susceptible to even the smallest adversarial attacks, while still enforcing the defensive mechanism for greater perturbation attacks.



(a) CTL (b) CNL

Figure 4.11: ADVERSARIAL STABILITY AGAINST UNTARGETED FGSM ATTACK OF FGSM EOS TRAINED NETWORKS. *Figure (a) the CTL scores and figure (b) the CNL scores against untargeted FGSM attacks with perturbation magnitudes ε*



(a) negatives (b) unknowns

Figure 4.12: OSCR BENCHMARK. *Figure (a) is the OSCR curve gained from using the first 13 letters as negatives. In figure (b) we used the latter 13 letters, which were never seen before by any network.*

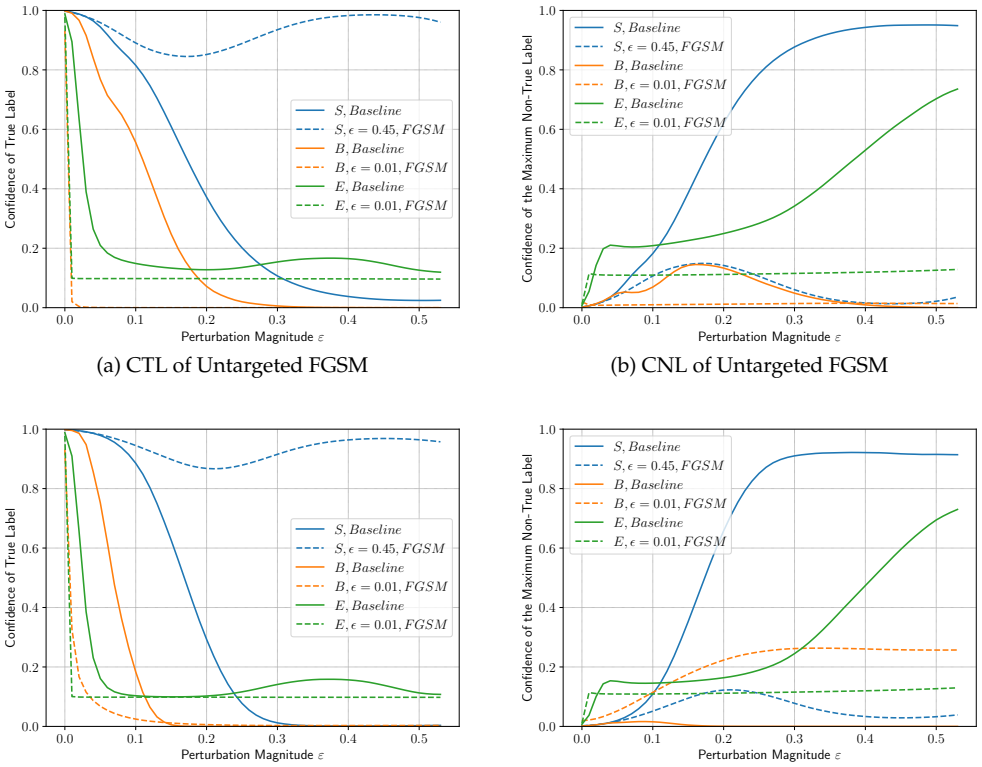
4.3 Part 3: Comparison

In this section, we use two different models from each loss approach respectively to illustrate and compare them against each other even further. The two models per approach are always the baseline model and the adversarial trained model, which performed the best during the evaluation tests, as shown in the previous sections. Specifically, alongside the baseline models, we use the FGSM trained models, with the sample adversarial perturbation magnitudes $\epsilon = 0.45$ for the Softmax approach, $\epsilon = 0.01$ for the BG approach and likewise $\epsilon = 0.01$ for the EOS approach.

The OSCR performance of the two open-set letters trained models is clearly well above their adversarial trained counterparts as seen in Figure 4.1. Nevertheless, both the open-set adversarial trained models outperform the models with the Softmax approach. While the adversarial trained BG model is superior for the first 13 letters (a), the adversarial trained EOS model outperforms it for the latter ones (b). The reasons for the performance difference are partly unclear.

The adversarial stability for simple attacks like the untargeted and targeted FGSM produced the expected results. In Figure 4.13 we see that the softmax baseline is resistant to adversarial attacks throughout all the perturbation magnitudes. The open-set adversarial trained models both have high susceptibility against the adversarial attacks, but employ their respective defensive mechanisms by not misclassifying these adversarial samples to another class but rather as unknown.

Stronger attacks such as untargeted and targeted projected gradient descent attack (PDG) leave us with different results. As seen in Figure 4.14 both the softmax baseline and its adversarial trained counterpart are susceptible to untargeted and targeted attacks. Surprisingly the Softmax FGSM trained model provides almost no stability increase against these iterative attacks. This finding was already concluded in Madry et al. (2019). The EOS baseline model is also very susceptible to these more sophisticated attacks as is shown by the drastic rise of the non-true label confidence rise both for the targeted and untargeted PGD attacks, as it does not display its distinct defense mechanism like it did for FGSM attacks. Interestingly, the BG baseline model still is able to display its defensive mechanism against the untargeted attack, by redistributing the confidence to the unknown label instead of the other known labels as seen in Figure 4.14(b), the same applies for its adversarial trained counterpart, as the non-true label confidence does not seem to rise even slightly. Surprisingly though, both BG approaches are susceptible to PGD targeted attacks, as their non-true label confidence scores do rise contrary to the results in the FGSM and untargeted



(a) CTL of Untargeted FGSM

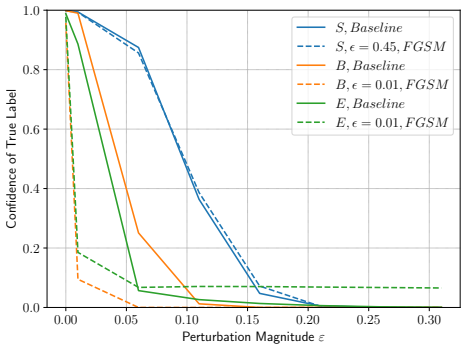
(b) CNL of Untargeted FGSM

(c) CTL of Targeted FGSM

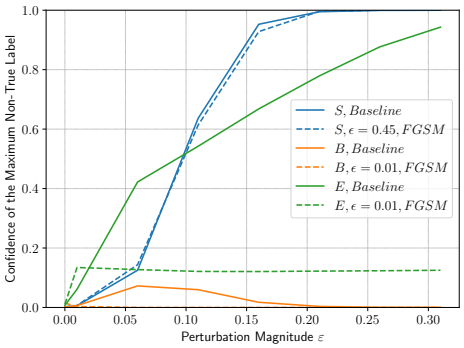
(d) CNL of Targeted FGSM

Figure 4.13: ADVERSARIAL STABILILTY AGAINST TARGETED AND UNTARGETED FGSM. *These figures show the adversarial stability against weaker attacks.*

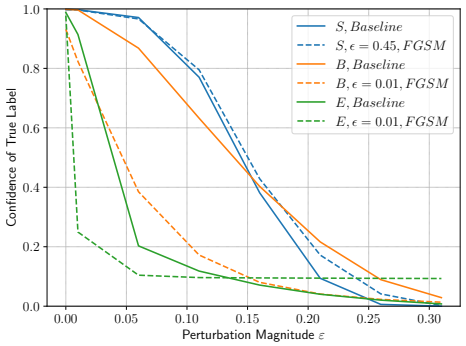
PGD evaluations. This may imply that the defensive mechanism of the BG approaches has a distinct vulnerability against strong targeted attacks. The only network still withstanding the PDG targeted attack is the EOS adversarial trained one, as its non-true label confidence score does not rise above the 0.2 mark. This means even the stronger targeted adversarial attacks are not able to provoke a misclassification to another label, the network would instead classify it as unknown depending on the classification scheme and threshold.



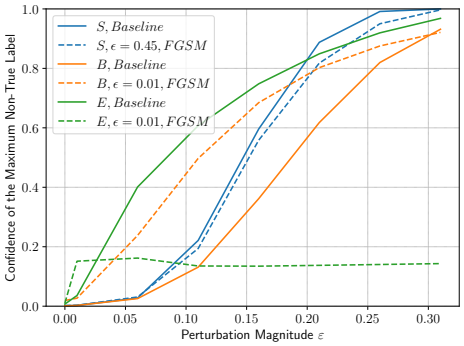
(a) CTL of Untargeted PDG



(b) CNL of Untargeted PDG



(c) CTL of Targeted PGD



(d) CNL of Targeted PGD

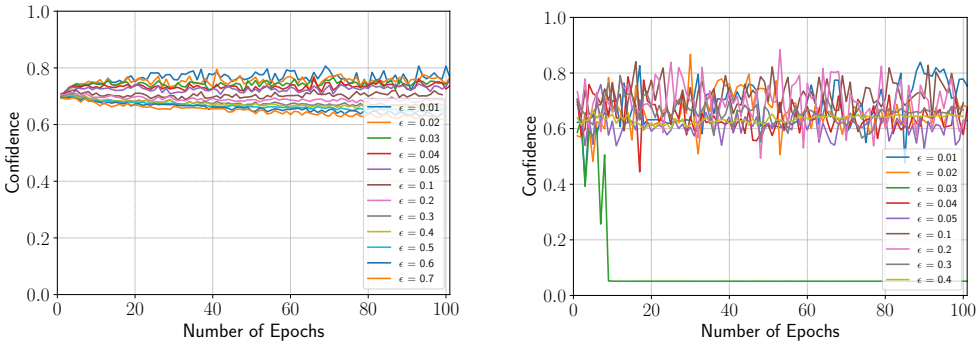
Figure 4.14: ADVERSARIAL STABILILTY AGAINST TARGETED AND UNTARGETED PGD ATTACKS. *These figures show the adversarial stability against stronger attacks.*

Chapter 5

Discussion

During the first part of our experiments we could reaffirm the suggested OSR capabilities of the networks. Naturally both the open-set networks performed better than the closed-set ones. Additionally, we addressed the innate defensive mechanism of open-set trained models against adversarial attacks. While these models are in fact more susceptible to adversarial attack, meaning their confidence scores of the true label are reduced faster than their closed-set trained counterpart, they do not necessariliy misclassify the adversarial samples as another known class.

In the second part, where we performed adversarial training, we received mixed results. We received the expected results for the Softmax approach in terms of adversarial stability and OSR performance. The utilization of adversarial samples as negatives both in the EOS and BG approaches is still a non-straightforward procedure, as the OSCR curves show that the open-set capabilities are inferior to the ones of the baseline models. Moreover, the search for a suitable ϵ is a difficult matter, as diverging issues also arise while there is no batch normalization in place. In Figure 5.1 we have the confidence scores of the validation sets as described in Section 3.5.1. For the EOS we see a clear tendency that the confidences rise for smaller perturbation magnitudes ϵ and this has been shown in the experiments as the model trained with the smallest perturbation resulted in having the best OSCR scores. For the BG approach the confidence scores do not indicate a clear tendency for a suitable ϵ and it just so happened to be the smallest one, as it had the strongest oscillations for the confidence scores. Furthermore, noise-trained models are generally outperformed by FGSM models in every evaluation we conducted.



(a) EOS (b) BG

Figure 5.1: OSCR BENCHMARK. *Figure (a) is the OSCR curve gained from using the first 13 letters as negatives. In figure (a) we used the latter 13 letters, which were never seen before by any network.*

The distinct defense mechanism seen during the experiments could be introduced against adversarial samples. Szegedy et al. (2014) noted that the set adversarial samples may be of extreme low probability and generally adversarial samples are thought as malicious attacks. So instead of trying to strive for the tradional adversarial stability by optimizing the CTL, classifying these malicious attacks as unknowns seems to be a viable strategy. The defense mechanism can be regarded as a mix of *Architectural Defense* and *Adversarial Training* (Serban et al., 2020) as only open-set networks displayed it, due to their unique designs against unknown classes. Open-set adversarial trained networks seem to display some viability, but further research needs to be conducted, preferably with larger datasets.

Appendix A

Attachments

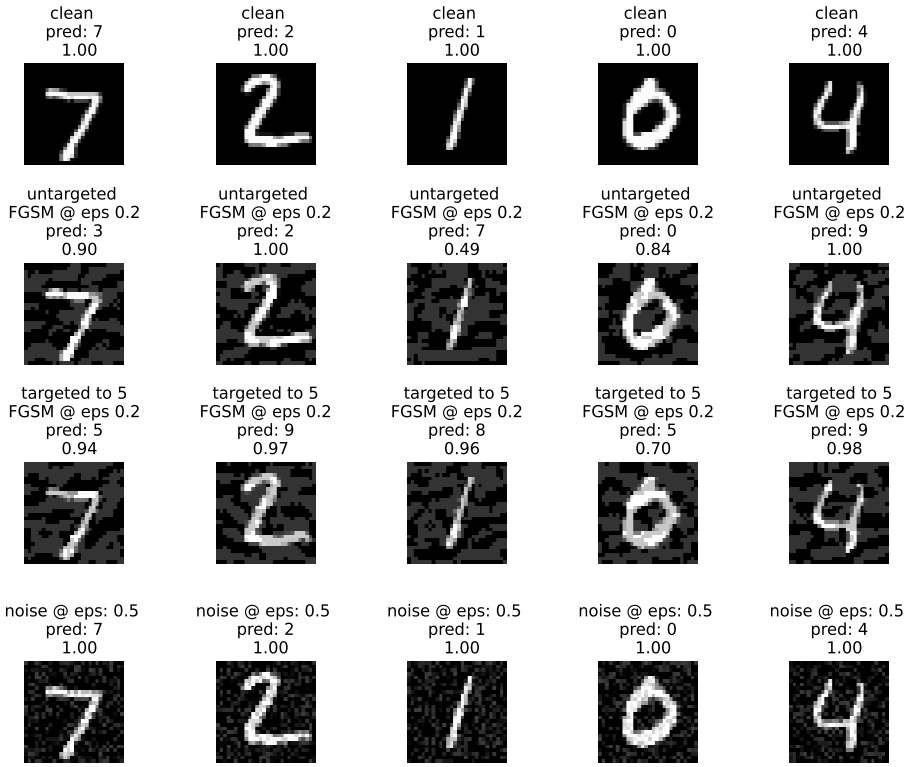
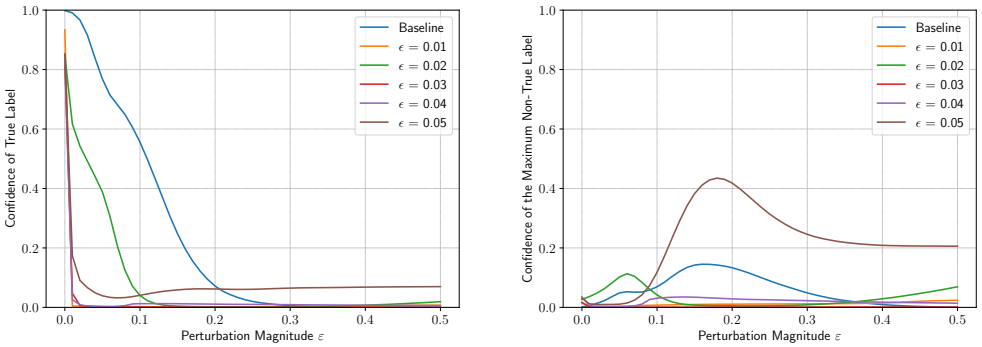
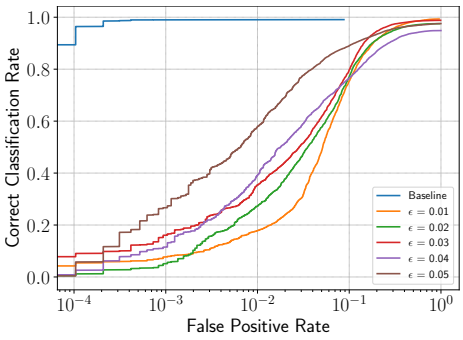


Figure A.1: EXAMPLE DIGITS. *Five different numbers are perturbed thorugh different methods and evaluated by the Standard Softmax network*

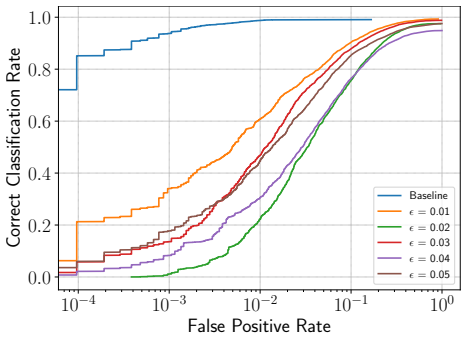


(a) CTL of Untargeted FGSM

(b) CNL of Untargeted FGSM

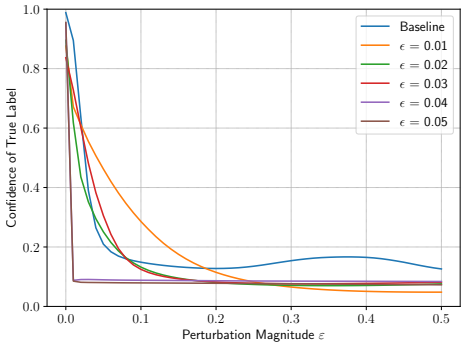


(c) negatives

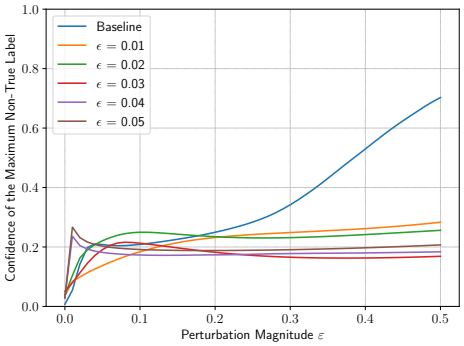


(d) unknown

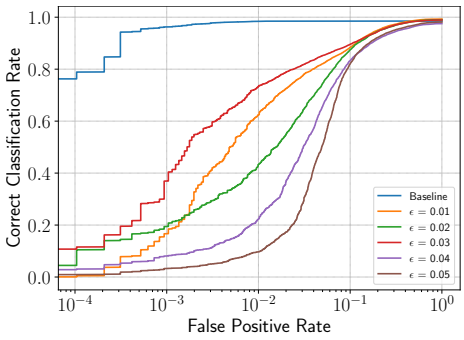
Figure A.2: STANDARD EVALUATION OF NOISE BG TRAINED MODELS. *These are the evaluations for the BG noise trained models. OSCR curves and adversarial stability against FGSM attacks.*



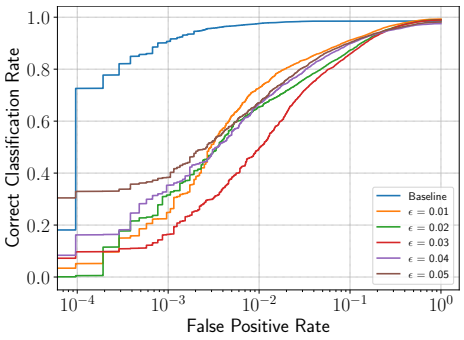
(a) CTL of Untargeted FGSM



(b) CNL of Untargeted FGSM



(c) negatives



(d) unknowns

Figure A.3: STANDARD EVALUATION OF NOISE EOS TRAINED MODELS. *These are the standard evaluations for the EOS noise trained models. OSCR curves and adversarial stability against FGSM attacks.*

List of Figures

List of Tables

List of Listings

Bibliography

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

Bendale, A. and Boult, T. E. (2016a). Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572.

Bendale, A. and Boult, T. E. (2016b). Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572.

Boult, T. E., Cruz, S., Dhamija, A. R., Gunther, M., Henrydoss, J., and Scheirer, W. J. (2019). Learning and the unknown: Surveying steps toward open world recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9801–9807.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pages 227–236. Springer.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE.

Dhamija, A. R., Günther, M., and Boult, T. (2018). Reducing network agnostophobia. *Advances in Neural Information Processing Systems*, 31.

Ding, G. W., Wang, L., and Jin, X. (2019). advertorch v0.1: An adversarial robustness toolbox based on pytorch.

Ge, Z., Demyanov, S., Chen, Z., and Garnavi, R. (2017). Generative openmax for multi-class open set classification.

Geng, C. and Chen, S. (2020). Collective decision for open set recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):192–204.

Geng, C., Huang, S.-j., and Chen, S. (2020). Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3614–3631.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial examples in the physical world.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer.

Luo, Y., Boix, X., Roig, G., Poggio, T., and Zhao, Q. (2016). Foveation-based mechanisms alleviate adversarial examples.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2019). Towards deep learning models resistant to adversarial attacks.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Palechor, A. (2022). Adversarial training for open-set classification. Master's thesis, University of Zurich.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Rozsa, A., Günther, M., and Boult, T. E. (2017). Lots about attacking deep features. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 168–176. IEEE.

Rudd, E. M., Jain, L. P., Scheirer, W. J., and Boult, T. E. (2018). The extreme value machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):762–768.

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boult, T. E. (2013). Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772.

Schnyder, J. (2021). Deep adversarial training for teaching networks to reject unknown inputs. Master's thesis, University of Zurich.

Serban, A., Poll, E., and Visser, J. (2020). Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 53(3):1–38.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks.

Tanay, T. and Griffin, L. (2016). A boundary tilting persepective on the phenomenon of adversarial examples.

Tommasi, T., Patricia, N., Caputo, B., and Tuytelaars, T. (2017). A deeper look at dataset bias. *Domain adaptation in computer vision applications*, pages 37–55.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. (2019). Improving adversarial robustness requires revisiting misclassified examples. In *International conference on learning representations*.

Wong, E., Rice, L., and Kolter, J. Z. (2020). Fast is better than free: Revisiting adversarial training.

Zhang, J. and Li, C. (2019). Adversarial examples: Opportunities and challenges. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–16.

Zhang, S., Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2017). Towards reaching human performance in pedestrian detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):973–986.