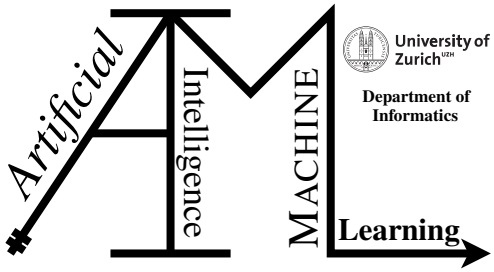
Adversarial Training for Open-Set Classification

Master Thesis

Andres Palechor
18-795-583

Submitted on
20 March 2022

Thesis Supervisor
Prof. Dr. Manuel Günther



Master Thesis

Author: Andres Palechor, jesusandres.palechoranacona@uzh.ch

Project period: 20 September 2021 - 20 March 2022

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

Acknowledgements

I want to thank Prof. Manuel Güenther for his guidance, patience and in-depth discussions during this project. Also, many thanks to my family and friends for their unconditional support.

Abstract

Open Set Recognition (OSR) is a promising field that aims to adapt deep neural models to a real-world scenario, i.e., the presence of unknown data distributions at inference time. Recent approaches address OSR by including into the training set negative classes that a classifier learns to reject, expecting that these data increase the robustness of the classifier on unknown classes. Furthermore, a critical question on OSR is how to select representative negative classes. In this work, we study the performance an OSR approach, the Entropic open-set and Objectosphere, two recently proposed loss functions to deal with unknowns. We show that these losses can be used in complex custom scenarios that use natural images and several levels of similarity between the negative classes. Additionally, we investigate a method to adapt adversarial training into OSR scenarios. Our experimental results show that it is possible to use synthetic adversarial examples to increase the robustness of a classifier in OSR.

Zusammenfassung

Open Set Recognition (OSR) ist ein vielversprechender Bereich, der darauf abzielt, tiefe neuronale Modelle an ein reales Szenario anzupassen, d. h. an das Vorhandensein unbekannter Datenverteilungen zum Zeitpunkt der Inferenz. Neuere Ansätze befassen sich mit OSR, indem sie in die Trainingsmenge negative Klassen aufnehmen, die ein Klassifikator abzulehnen lernt, in der Erwartung, dass diese Daten die Robustheit des Klassifikators bei unbekannten Klassen erhöhen. Eine wichtige Frage im Zusammenhang mit OSR ist, wie repräsentative negative Klassen ausgewählt werden können. In dieser Arbeit untersuchen wir die Leistung eines OSR-Ansatzes, der Entropic open-set und Objectosphere, zwei kürzlich vorgeschlagene Verlustfunktionen für den Umgang mit unbekannten Klassen. Wir zeigen, dass diese Verlustfunktionen in komplexen benutzerdefinierten Szenarien verwendet werden können, die natürliche Bilder und verschiedene Ähnlichkeitsgrade zwischen den Negativklassen verwenden. Darüber hinaus untersuchen wir eine Methode, mit der sich das adversarische Training in OSR-Szenarien anpassen lässt. Unsere experimentellen Ergebnisse zeigen, dass es möglich ist, synthetische Negativbeispiele zu verwenden, um die Robustheit eines Klassifikators in OSR zu erhöhen.

Contents

1 Introduction **1**
- 1.1 Motivation and Description of Work . . . 1
- 1.2 Thesis Outline . . . 2

2 Background and Related Work **3**
- 2.1 The open set scenario . . . 3
- 2.2 Open Set Recognition Approaches . . . 4
  - 2.2.1 OpenMax . . . 5
  - 2.2.2 Generative OpenMax . . . 6
- 2.3 Loss Functions . . . 8
  - 2.3.1 Softmax Loss . . . 8
  - 2.3.2 Softmax with Garbage Class . . . 8
  - 2.3.3 Entropic Open-Set Loss . . . 8
  - 2.3.4 Objectosphere Loss . . . 9
- 2.4 Imagenet Open Set Protocols . . . 10
  - 2.4.1 Imagenet Dataset . . . 11
  - 2.4.2 Imagenet Open-Set Protocols . . . 12
- 2.5 Adversarial Attacks . . . 13
  - 2.5.1 Taxonomy of Adversarial Attacks . . . 15
  - 2.5.2 Fast Gradient Sign Attacks . . . 16
  - 2.5.3 Iterative Attacks . . . 17
- 2.6 Adversarial Training . . . 17
  - 2.6.1 Adversarial Training in OSR . . . 17

3 Methods **19**
- 3.1 DNN Classifier . . . 19
  - 3.1.1 Batch Normalisation . . . 20
- 3.2 Adversarial Training for OSR . . . 22
  - 3.2.1 Sample Filtering . . . 22
  - 3.2.2 Waiting Epochs . . . 22
  - 3.2.3 Loss function . . . 23

4 Experiments and Performance **25**
- 4.1 Evaluation Metrics . . . 25
  - 4.1.1 Open-Set Classification Rate . . . 25
  - 4.1.2 Confidence Score . . . 25
- 4.2 Data preprocessing . . . 26

4.3 Experiments 27
4.3.1 Phase 1: Training with clean images 27
4.3.2 Phase 2: Adversarial Training 31
4.3.3 Protocol 2 33
4.3.4 Protocol 1 35
4.3.5 Protocol 3 36

5 Discussion **37**

6 Conclusions **41**

A Attachments **43**

Chapter 1

Introduction

1.1 Motivation and Description of Work

It seems truly impressive the progress achieved by deep learning algorithms. Since its early stages (arguably) in the 1950s with the invention of Rosenblatt's perceptron (Rosenblatt, 1958), in only a few decades, deep learning methods enabled significant advances in several fields such as robotics, health care and computer vision (Lenz et al., 2015; Punjani and Abbeel, 2015; Mariolis et al., 2015; Miotto et al., 2018; Qayyum et al., 2020; Voulodimos et al., 2018). The outcomes of these algorithms started to exceed human performance in some supervised classification tasks. For example He et al. (2015) reports an ImageNet classification error rate of 3.57%, while the human error rate is 5.1% (Russakovsky et al., 2015).

Most Deep Neural Networks (DNNs) work under the closed-set assumption, i.e., classifying a finite set of known-classes assumes that the classes seen in training and inference have equivalent distributions. However, this assumption is rarely fulfilled in real-world scenarios; A deployed model could face data from unknown classes at inference time. The deployment of DNN-based systems to critical applications like autonomous driving requires robust methods to unseen data distributions, yet many systems do not have the strategies to handle unknown classes. In fact, Dhamija et al. (2020) showed that DNNs might predict samples from unseen classes as knowns with high confidence.

The presence of unknown classes changes the task's nature from classification to recognition, as the systems should be able to classify samples from known classes (positives) while rejecting samples from unknown classes (negatives). This task is known as *Open-Set Recognition* (OSR). In real-world applications, DNNs could face unknown classes from many sources and distributions. Furthermore, the features of positive and negative samples might be mixed in unexpected ways. For example, in object recognition, a system should reject a possibly infinite set of background objects, combined in unpredictable ways inside an image. There exist approaches that attempt to increase the capacity of DNNs to recognize positive classes and discard negatives, but handling the unknown is still an open problem. Recent approaches extend the training data with background classes (known-unknowns) (Yu et al., 2017; Neal et al., 2018; Dhamija et al., 2018), so the DNN learns to discard them, expecting the model generalizes this behaviour at inference time. However, it is challenging to select representative known-unknown classes to improve the robustness to completely unknown samples. For example, Dhamija et al. (2018) proposed two novel unknown-aware losses, the *entropic open-set loss* and *objectosphere loss*, and showed that training with known-unknowns similar to known classes improves the DNN performance, yet this implies that the task is extended to find optimal known-unknowns.

Recent approaches also consider finding good known-unknowns. For instance, Bhoumik (2021) proposed three Imagenet-based protocols to simulate scenarios with varying similarities

between known and unknown classes, providing the option to explore open-set approaches in diverse and challenging scenarios. Other methods search for 'good' known-unknowns producing synthetic data, typically with Generative Adversarial Networks (GANs) Ge et al. (2017); Perera et al. (2020). Alternatively, Schnyder (2021) included adversarial samples to improve the model's performance. Although this method is promising, it was tested only on the MNIST dataset, and it requires further analysis (e.g. different DNNs and natural images).

In this work, we address two related topics. First, we elaborate on the work of Dhamija et al. (2018) by evaluating the behaviour of their approach when facing the diverse and similarity-varying scenarios proposed by (Bhoumik, 2021). Second, we extend the method proposed by Schnyder (2021), studying the effect of adversarial training in the context of open set recognition in complex scenarios. The major contributions of this work can be summarized as follows:

- We evaluate the entropic open-set and objectosphere approaches under diverse and similarity-varying scenarios.
- We propose a training method to include adversarial samples into OSR scenarios.
- We show that training with adversarial samples under specific conditions helps to improve the performance in OSR.

1.2 Thesis Outline

This document is organized as follows: Chapter 2 reviews the description of open-set scenarios and the most relevant related approaches to this work. Chapter 3, describes our proposed methodology to face open-set scenarios in detail. The findings of this work are presented in chapter 4 and discussed in chapter 5. Finally, Chapter 6 presents the conclusions and future work.

Chapter 2

Background and Related Work

In this chapter, we briefly summarise relevant strategies to OSR and adversarial training. Section 2.1 introduces the open set scenario definition. Section 2.2 presents approaches that address OSR. Section 2.3 describes the main methods related to this work, the entropic open-set and objectosphere losses. Section 2.4 describes the ImageNet protocols, an approach to create OSR scenarios. The final two sections are dedicated to adversarial training. Section 2.5 describes methods to generate adversarial samples, and section 2.6 describes adversarial training approaches.

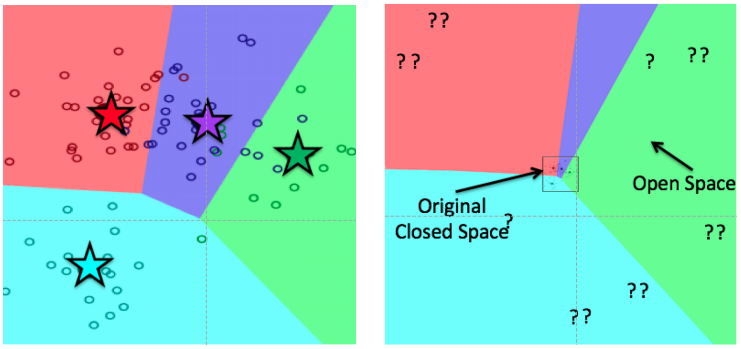
2.1 The open set scenario

The excellent performance reported by DNNs-based systems is typically achieved under a static environment or closed-set assumption, i.e., the classes are known and the samples drawn from the same distributions in training and test time; no new unknown classes are present at inference time. However, real data is dynamic and sometimes unpredictable, and DNNs need to cope with unknown samples. The OSR approach aims to correctly classify samples from the known classes while discarding samples from unknown classes. Dhamija et al. (2018) propose the following notation:

- Set $\mathcal{C} = \{c_1, \ldots, c_C\}$ of Known classes: Classes of interest, positive examples that the classifier shall identify. The training samples of known classes $\mathcal{C}$ are denoted $\mathcal{D}_c{}'$ and the test samples $\mathcal{D}_c$.
- Set $\mathcal{U}$ of unknown classes: All classes the classifier needs to reject. This set is partitioned in:
  - Set $\mathcal{B}$ of known-unknown classes: Negative examples or garbage classes present in training. The training samples from known unknown classes $\mathcal{B}$ are denoted $\mathcal{D}_b{}'$ and the test samples $\mathcal{D}_b$.
  - Set $\mathcal{A}$ of unknown-unknown classes: The infinite set of classes not available during training, only present at testing time. The samples from unknown unknown classes $\mathcal{A}$ are represented by $\mathcal{D}_a$.

Figure 2.1 shows the feature space of a four-class classification example using Nearest Class Mean. In figure 2.1a, the data is reasonably classified. Still, suppose more regions of feature space are considered, like in figure 2.1b. In that case, the class boundaries will include faraway unknown examples, possibly the "?" samples do not belong to any of the known classes. The space that is far from $\mathcal{C}$ classes is named *open space*. Recent approaches model the open space with $\mathcal{B}$, where the classifier learns to reject them by thresholding a score. However, it is challenging to find appropriate $\mathcal{B}$ classes to model the open space since $\mathcal{U}$ is infinite. Note that a trade-off

can be created between strong dejection of unknown samples at the cost of accuracy in the $\mathcal{C}$ classification. For example, in principle, a standard cross-entropy loss achieves high accuracy rates by admitting all unknown samples.



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

Figure 2.1: OPEN SPACE. (a) shows a closed-set approach using Nearest Class Mean. Region boundaries and mean classes (stars) are innacurate when zooming out to the open space like in (b). The unknown samples "?" will get a known label. Source Boult et al. (2019).

2.2 Open Set Recognition Approaches

The open set recognition task aims to correctly classify samples from the known classes while discarding samples from unknown classes. However, the study of unknown instances is not new in the literature. For example, Novelty Detection (ND) focus on identifying test instances that do not belong to training classes. ND can be seen as a binary classification problem that determines if an instance belongs to any of the training classes or not, but without exactly deciding which class (Bodesheim et al., 2015). This method is also known as Out of Distribution Detection (OOD). OOD includes approaches in supervised, semi-supervised and unsupervised learning (Jiang et al., 2018; Ren et al., 2019; Golan and El-Yaniv, 2018). Additionally, the formulation of multiclass-OOD is similar to OSR.

Since Scheirer et al. (2012) formalized OSR as a constrained risk optimization problem; researchers have proposed several approaches for robust systems to unknown classes. The problem consists on finding balance between known space and open space. Geng et al. (2021) conveniently classifies the OSR approaches based on two perspectives: discriminative and generative models, as depicted in figure 2.2. The discriminative models are divided into Traditional Tachine Learning (TML) and DNNs. TML-based approaches include adaptations to the OSR scenario of methods like SVM, sparse representation and nearest neighbour. Similarly, DNN-Based models use their powerful learning representation and adapt it to the OSR scenario. Representative examples are OpenMax (Bendale and Boult, 2016), Classification reconstruction for OSR (Yoshihashi et al., 2019) and both the Entropic Open Set loss and Objectosphere loss proposed by Dhamija et al. (2018).

Generative models can be divided into two groups: instance and non-instance generation. The first group uses adversarial learning techniques to account for $\mathcal{B}$ classes with samples generated by a Generative Adversarial Network (GAN) or an autoencoder. A few representative examples are Generative OpenMax (Ge et al., 2017) and Open-Category Classification by Adversarial Sample Generation (Yu et al., 2017). On the other hand, non-instance generation models are not

widely studied, but they rely on Dirichlet processes to deal with the OSR scenario, for instance, Collective Decision for Open Set Recognition (Geng and Chen, 2022).

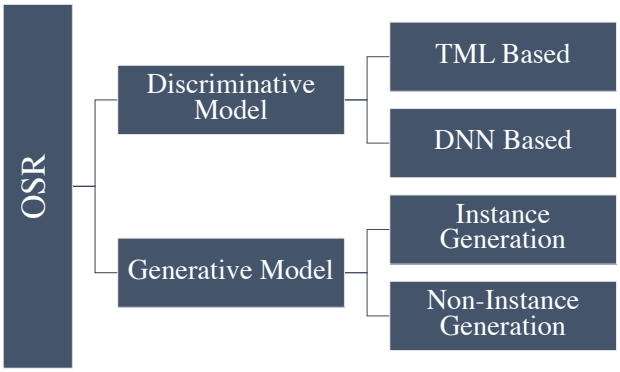


Figure 2.2: OSCR APPROACHES CLASSIFICATION. The approaches are classified by model type. Proposed by Geng et al. (2021).

2.2.1 OpenMax

Bendale and Boult (2016) proposed OpenMax, a meta-recognition approach that is one of the first attempts to adapt DNNs to OSR without $\mathcal{B}$ classes. This approach extends a pretrained softmax-based DNN classifier with a score calibration module. Instead of using the softmax scores, it captures the sample's information from the activations that will be normalised by the softmax: *logits*. The intuition is that the logits (here named Activation Vectors AV) are not class independent, but instead, they reflect the similarity or the distance between classes. The author model each class by the Mean of the Activation Vectors (MAV) of the correctly classified samples of such class, expecting that samples from the same class will have a similar AV. Note that the distribution of each class is represented by a single point in the AV space. Furthermore, the author showed that samples from similar classes (like shark and shark wale) do indeed have similar activation vectors.

The distance from a sample to its corresponding MAV is calculated during training. The distance function is a weighted sum of Euclidean and cosine distances, but several others can be implemented. Additionally, the distances between samples and the MAV can be modelled as random variables. In particular, using Extreme Value Theory (EVT), the maximum distance between a correctly classified sample and its MAV is fitted using a Weibull distribution for each class.

Each resulting distribution uses a parameter p_i that estimates the probability of a sample being an outlier for the corresponding class i. Every p_i is used to calibrate the final class score. Instead of using the maximum activation of an AV, the top$-j$ activations are employed, assuming the other activations do not carry meaningful information. The top$-j$ activations are used to recalibrate the distributions weights and class activations. Additionally, a pseudo-activation is calculated and inserted in the first position of the calibrated AV. The pseudo-activation represents the unknown class. Finally, softmax normalisation is applied over all the calibrated activations, obtaining comparable scores over all classes.

OpenMax effectively handles OSR extending the softmax cross-entropy loss. The algorithm was implemented using a pre-trained AlexNet network (Krizhevsky et al., 2012). OpenMx was tested on ImageNet data, including sets of adversarial images (disturbed images that cause mis-

classification), open set images (real images from unknown classes), fooling images (artificial images created to make a particular class have high activation). Figure 2.3 shows the AV representation of the model. Note that there is a strong correlation in natural image response patterns (model, real image).

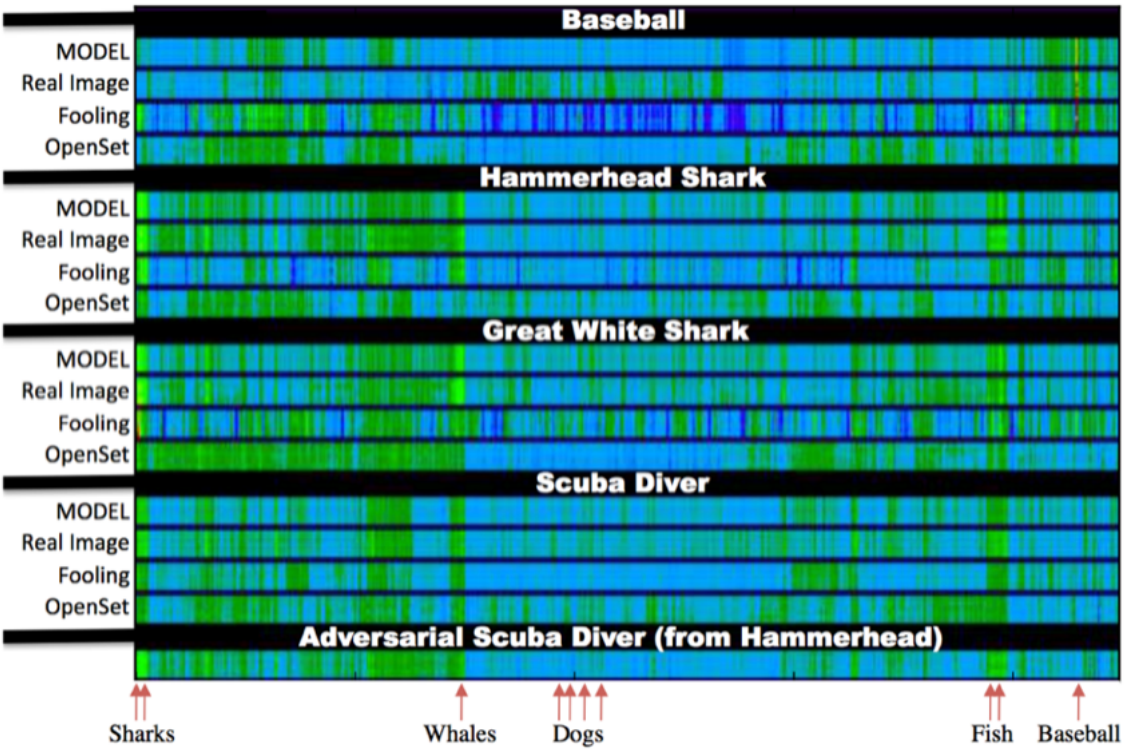


Figure 2.3: OSCR APPROACHES CLASSIFICATION. Activation vectors for 5 images, each one separated for a black line. Each AV is depicted as a 10x450 color pixels, green means high activation values. For related natural images AV have strong correlation patterns like in Hammerhead Shark and Great White Shark. Range of 5 categories are identified at the bottom of the image. Source (Bendale and Boult, 2016).

OpenMax showed that the AV (logits) contain information about the class distribution and can be used to estimate a rejection score. In our study we evaluate state-of-the-art approaches that use the logits and the activations before the logits (deep features) to increase score separation to discard unknown samples.

2.2.2 Generative OpenMax

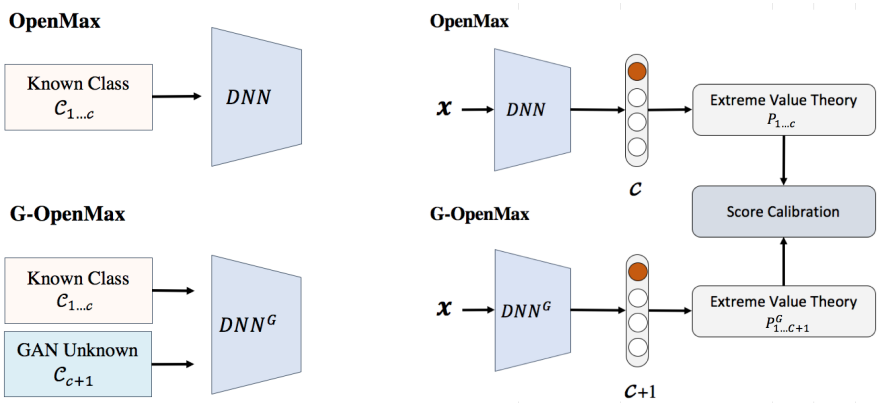
Ge et al. (2017) proposed Generative OpenMax (G-OpenMax) as a natural extension of OpenMax. Instead of calculating a pseudo activation for the unknown classes, G-OpenMax creates synthetic images to directly calculate the probability of the unknown class. This approach creates synthetic samples that are distinct from known samples and possibly good representatives of the open space. These synthetic samples are processed as an extra training label.

Apart for creating unknown examples, this approach is very similar to OpenMax, as it keeps the use of MAVs and Weibull distributions for score calibration. G-OpenMax provides explicit

probability estimation of the synthetic class, thus the score calibration is done over $C + 1$ classes, where the additional class represents the unknowns. The additional class enables the model to find a threshold to separate $\mathcal{C}$ from synthetic $\mathcal{B}$ samples.

G-OpenMax assumes that the open space classes belong to a subspace of the universal space. In this assumption, unknown classes share some patterns with the known classes. For example, if a system is trained for gray-scale English character recognition, an open set class might be related to characters of other alphabets. For instance, an RBG image is not expected in the character's trained system. The assumption allows treating the open space classification problem as closed space (a very strong assumption). Consequently, to represent the open space, samples from the known subspace are drawn. This artificial samples are generated from a mixture of the $\mathcal{C}$ distributions in latent space. A generator is trained following the standard minimax optimization in conditional GAN (Odena et al., 2017).

Figure 2.4 shows the diagram of the preprocessing and score calibration in OpenMax and G-OpenMAX. The pretraining of G-OpenMax utilizes the synthetic sample as class $C + 1$ and the score calibration follows the same process as OpenMax. G-OpenMax works well under small datasets like MNIST (LeCun et al., 2010) and HASY (Thoma, 2017) but does not improve performance with natural images (ImageNet).



(a) Network pretraining (b) Score Calibration

Figure 2.4: OpenMax and G-OpenMax Training. (a) OpenMax trains a DNN using the set of known classes. In contrast, G-OpenMax uses a generative adversarial network DNN^G to train the network including a synthetic sample as the unknown class. (b) Both use EVT to fit Weibull activations; no pseudo activation is required in G-OpenMax. Source Ge et al. (2017).

This is one of the first approaches that used GANs to fabricate $\mathcal{B}$ class samples, there are more robust systems like Generative-Discriminative Feature Representation for OSR (GDFR) (Perera et al., 2020), Open Set Learning with Counterfactual images (Neal et al., 2018) or Classification by Adversarial Sample Generation (Yu et al., 2017).

In our study instead of using GANs, we fabricate adversarial samples of natural images to represent samples from $\mathcal{B}$.

2.3 Loss Functions

This section presents the description of the classical softmax cross-entropy loss, and a recent approach that extends the softmax loss to include unknowns.

2.3.1 Softmax Loss

DNNs typically use the softmax cross-entropy loss to solve a classification task. The softmax is a normalisation function applied to the logits activations. The raw non-normalized logits vector is transformed into positive scores that sum to 1 across all input values (Bridle, 1990). Consider a set of known classes $\mathcal{C} = \{c_1, \ldots, c_C\}$, the softmax value of a class c of an input sample x is defined by:

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

Where $l_c(x)$ is the logit value for class c. Moreover, the cross-entropy loss for an input sample is:

$$J_S(x) = -\log S_c(x) \tag{2.2}$$

The softmax cross-entropy loss has dominated the classification task approaches because its outputs are considered class probabilities and have good differentiation properties. The softmax outputs are also referred to as confidences or softmax scores. During training, the objective of the softmax loss is to make the predicted probability of the target class larger than other classes. Despite its great success, softmax has two main drawbacks. First, it tends to bias the probabilities towards certain classes even when the differences between logit values are minimal (Matan et al., 1990); consequently, DNN could predict with high confidence an incorrect class. Second, the normalisation is inherent to the closed-set assumption, limited to $\mathcal{C}$ classes, making it unsuitable for OSR scenarios.

2.3.2 Softmax with Garbage Class

Detection approaches use a modified version of cross-entropy loss to discard background objects (Liu et al., 2016; Ren et al., 2016). All samples from uninteresting classes are grouped under a common garbage or background class. In training, DNNs use samples from $\mathcal{B}$ to find boundaries between $\mathcal{C}$ and $\mathcal{B}$ classes, hoping that the boundaries generalize well when facing $\mathcal{A}$ classes. Nevertheless, this approach is limited in some real-world applications since it is a probable source of dataset bias (Tommasi et al., 2017), and $\mathcal{U}$ can contain significantly diverse examples. Figure 2.5 shows the feature space of a bottleneck network for MNIST classification using a softmax with garbage class, note that $\mathcal{A}$ samples overlap with regions of $\mathcal{C}$ classes.

2.3.3 Entropic Open-Set Loss

Dhamija et al. (2018) proposed a simple but elegant approach to OSR. Rather than directly creating mechanisms to reject unknown samples, they proposed the Entropic Open-Set loss, a function that aims to produce robust deep features against unknown classes and increase separability between them. The entropic open-set requires known unknown samples during training ($\mathcal{D}_b{}'$) to learn how to reject them. It contemplates two cases. First, it uses the traditional softmax loss in training samples from $\mathcal{C}$ classes ($\mathcal{D}_c{}'$), penalizing predictions of low scores for a target class. Second,

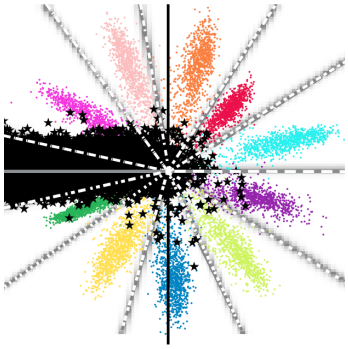


Figure 2.5: FEATURE SPACE VISUALIZATION. Feature space of LeNet++ for MNIST classification with garbage class. The unknown $\mathcal{A}$ samples (black) are pushed to the left side of the space. Source Dhamija et al. (2018).

for unknown samples, it attempts to maximize the entropy of the softmax responses by making them uniform over all $\mathcal{C}$ classes. As a result, the Entropic Open-Set loss increases the separability between knowns and unknowns and is defined by:

$$J_E(x) = \begin{cases} -\log S_c(x), & \text{if } x \in \mathcal{D}'_c \\ -\frac{1}{C}\sum\limits_{c\in\mathcal{C}} \log S_c(x), & \text{if } x \in \mathcal{D}'_b \end{cases} \tag{2.3}$$

This loss achieves its minimum value for $\mathcal{D}_b$' samples when the softmax scores are identical. Intuitively, since the softmax scores are interpreted as class probabilities, the maximum entropy is achieved when all classes have the same predicted score. Note that equal softmax scores require equal logit values for each class. In practice, one can use weights to balance the loss contribution of $\mathcal{D}_b$'. For instance, if $x \in \mathcal{D}'_b$ the loss can be calculated as:

$$J_E(x) = -\frac{\omega}{C}\sum_{c\in\mathcal{C}} \log S_c(x) \tag{2.4}$$

Where w influences the model's behaviour between improving the accuracy of known classes and the separability between the knowns and unknowns, $\omega = 0$ is equivalent to the softmax loss. This approach brings alternatives to the problem of predicting wrong classes with high confidence. Ideally during inference, $\mathcal{D}_b$ samples have low scores on all classes while target classes of $\mathcal{D}_c$ have high scores. Therefore, thresholding the score would be a suitable option to discard unknowns.

2.3.4 Objectosphere Loss

Dhamija et al. (2018) observed that on networks trained with Entropic open-set loss, the magnitude of the deep features of unknown samples is, in general, smaller than the magnitude of the known samples. Furthermore, in networks whose logit layer does not have a bias term, the Entropic Open-Set loss is minimized if the deep features vector is $\vec{0}$.

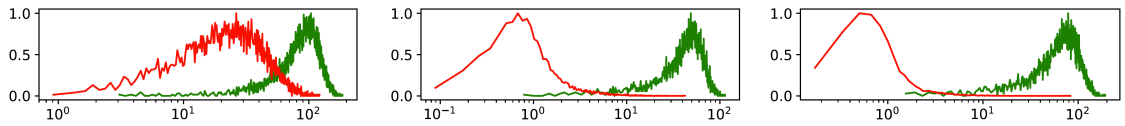
Let $F(x) \in \mathbb{R}^D$ be a deep feature vector, $L(x) \in \mathbb{R}^C$ the corresponding logit vector and $W \in \mathbb{R}^{C\times D}$ the weight matrix that connects $F(x)$ to $L(x)$. For networks with no bias in the logit layer $L(x) = W \cdot F(x)$. If $F(x) = \vec{0}$, then $L(x) = \vec{0}$, and the softmax entropy is maximized with probability of $\frac{1}{C}$ for all classes. The solution is not unique since there might be more vectors in the nullspace of W.

The Objectosphere loss attempts to push F of unknown samples to the center of the feature space by penalizing its magnitude, ideally $\|F\| = 0$ if $x \in \mathcal{D}'_b$. In contrast, the loss penalizes small $\|F(X)\|$ of known classes by setting a minimum threshold ξ. The trained network should produce robust features to unknown classes, responding only to samples from $\mathcal{C}$ classes. The objectosphere loss is formulated as follows:

$$J_O(x) = J_E(x) + \alpha * \begin{cases} \max(\xi - \|F(x)\|, 0)^2, & \text{if } x \in \mathcal{D}'_c \\ nomenclature\|F(x)\|^2 & \text{if } x \in \mathcal{D}'_b \end{cases} \tag{2.5}$$

ξ sets a norm threshold but also scales the loss value. In practice, α and ξ must be chosen, so the second term in 2.5 is comparable with J_E.

The histograms in figure 2.6 show the effect over the deep feature's magnitudes of the softmax, entropic open-set, and objectosphere losses. Although the objectoshpere loss best separates known and unknown samples, it does not guarantee the best classification accuracy between known classes. Section 4.1 will deepen on this topic and discuss appropriate evaluation methods in the OSR scenario.



(a) Softmax (b) Entropic Open-Set (c) Objectosphere

Figure 2.6: DEEP FEATURES HISTOGRAMS. Normalized histogram of $\|F\|$. Known samples (green) correspond to the 10 classes in MNIST, unknown classes (red) come from the Devanagari dataset. In (a) the unknown samples have in average smaller deep feature magnitudes. The separation is improved when including unknown samples in the Entropic Open-Set loss (b) and Objectosphere loss (c). Source Dhamija et al. (2018).

The previous approaches require training a DNN model with samples of $\mathcal{B}$ classes. Naturally, it is challenging to choose representative known-unknown samples because the open space is infinite. Dhamija et al. (2018) observed that samples from $\mathcal{B}$ classes distinctly different from $\mathcal{C}$ classes do not provide robustness to unknown unknowns $\mathcal{A}$, however, training with $\mathcal{D}_b$' with some similarity to $\mathcal{D}_c$' can increase the robustness of the model. In their experiments, the known classes are restricted to CIFAR-10 (Krizhevsky and Hinton, 2009) or MNIST digits (LeCun et al., 2010), and unknown-unknowns are sampled from small datasets such as SVHN (Netzer et al., 2011), Devanagri (Acharya et al., 2015) or subsets of CIFAR100.

One of the objectives of this thesis is to evaluate the performance of the entropic open-set and objectosphere approaches in more challenging OSR scenarios, including natural images of diverse known and unknown classes.

The following section describes the Imagenet dataset and the Imagenet open-set protocols, an approach to construct OSR scenarios with different levels of diversity and similarity between known and unknown classes.

2.4 Imagenet Open Set Protocols

In this section, the ImageNet Open-Set protocols are described. Section 2.4.1 introduces the Imagenet dataset and section 2.4.2 describes the protocols.

2.4.1 Imagenet Dataset

As the image processing research advances, more diverse and challenging datasets are needed to feed state-of-the-art algorithms. Small datasets like MNIST, CIFAR or NIST have enabled significant progress in the ML community serving as training and evaluation benchmarks. However, nowadays, new algorithms require more extensive and diverse datasets such as Imagenet (Deng et al., 2009).

Imagenet is an extensive hierarchical image database that provides more than 14 million human-annotated images. The critical property of Imagenet is its WordNet-based hierarchical structure (Miller, 1998). WordNet is a lexical database of English nouns, verbs, adjectives and adverbs grouped into sets of cognitive synonyms *synsets* [1]. Words that denote the same concept are grouped into the same synset. The synsets are linked using relations, which most relevant is the "ISA" relation. For example, it links a synset of general meaning "vehicle" to increasingly specific synsets like "craft" and "watercraft", indicating that the synset "vehicle" contains "craft", which in turn contains "watercraft". Figure 2.7 shows two examples from paths starting in "mammal" and "vehicle" respectively. In this tree-like structure, we call *superclass* to any synset that contains one or more synsets, a *subclass* to any synset contained by a superclass, and *leaves* to synsets with no subclasses. To date, Imagenet provides between 500-1000 images per each of the 21'831 indexed synsets.

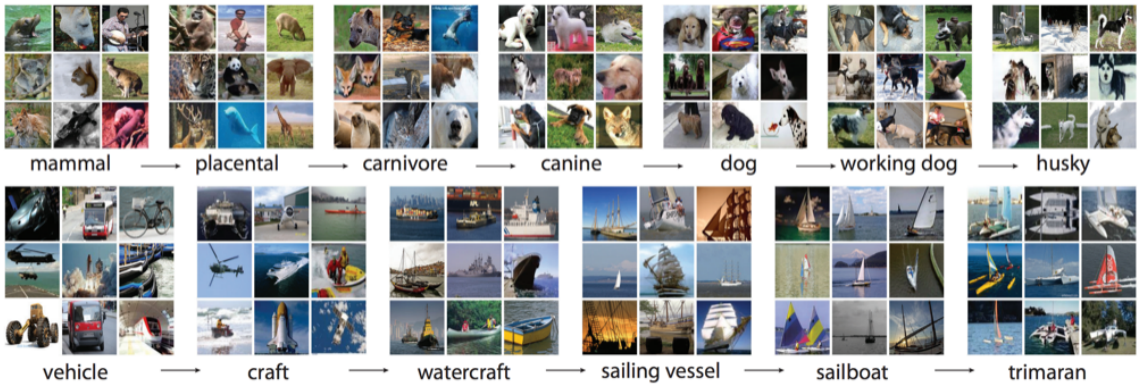


Figure 2.7: IMAGENET SNAPSHOT. Path examples from "mammal" and "vehicle" superclasses. In every step the superclass is more specific until reaching leaves synsets. Soruce Deng et al. (2009).

Russakovsky et al. (2015) proposed the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), which included image classification, object localization, and object detection tasks. ILSVRC provides training, validation and test sets which are subsets of Imagenet. The training and validation labels are publicly available, however, the test labels are kept private. Furthermore, The data contains 1000 classes, including subclasses and leaf classes that do not overlap, the training data contains 1.2 million images, the validation 50'000 and the testing 100'000.

ILSVRC was held between 2010 and 2017, becoming the standard benchmark for image classification, although the latest dataset update was released in 2012 (ILSVRC2012). The winners of the challenge include well-known DNN architectures such as AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015), VGG (Simonyan and Zisserman, 2015), ResNet (He et al., 2016) and SeNet (Hu et al., 2018).

[1] https://wordnet.princeton.edu/

2.4.2 Imagenet Open-Set Protocols

In order to create more general and challenging OSR scenarios, Bhoumik (2021) proposes three Open-set protocols. Each protocol is a subset of ILSVRC2012, which exploits the WordNet hierarchical structure to sample classes to construct $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{A}$. Each protocol creates an artificial open space with varying similarities between $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{A}$. Additionally, the protocols vary the similarity inside $\mathcal{C}$ classes, increasing the difficulty to classify them while discarding $\mathcal{B}$ correctly. The notions of similarity and difficulty are qualitative and determined mainly by visual inspection.

We denote $\mathcal{C}_i$, $\mathcal{B}_i$, and $\mathcal{A}_i$ as the knowns, known-unknowns and unknown-unknowns of protocol i respectively. Table 2.1 shows the criteria of the selected synsets for each protocol.

	Class Type		
Protocol	$\mathcal{C}$	$\mathcal{B}$	$\mathcal{A}$
1	All dog classes: Dog (116 synsets)	4-legged animals excluding dogs: Fox, Wild dog, Wolf, Feline, Bear, Musteline, mammal, Ungulate, Primate (67 synsets)	Non-animals: Food, Motor vehicle, Device (166 synsets)
2	Subclasses of Hunting dog (31 synsets)	Subclasses of Hunting dog mutually exclusive with $\mathcal{C}_2$ (30 synsets)	Subclasses of dog, 4-legged animals, mutually exclusive with $\mathcal{C}_2$ and $\mathcal{B}_2$: Toy dog, Fox, Wild dog, Wolf, Feline, Bear, Musteline mammal, Ungulate (55 synsets)
3	Subclasses of living-beings and objects: Subclasses of Dog, Bird, Insect, Furniture, Fish, Monkey,Car, Cat,Truck, Fruit, Fungus, Boat, Computer (159 synsets)	Subclasses of living-beings and object classes mutually exclusive with $\mathcal{C}_3$: Subclasses of Dog, Bird, Insect, Furniture, Fish, Monkey, Car, Cat, Truck, Fruit, Fungus, Boat, Computer (137 Synsets)	Subclasses of Superclasses of $\mathcal{C}_3$ mutually exclusive with $\mathcal{C}_3$: Reptile, Clothing, Ungulate, Vegetable,Aircraft (116 synsets)

Table 2.1: IMAGENET PROTOCOLS CLASSES OVERVIEW. The general description and count of the synsets included in each protocol. Source Bhoumik (2021).

Protocol 1 (P_1) has significantly different $\mathcal{C}_1$ and $\mathcal{A}_1$ classes, while $\mathcal{C}_1$ are similar to $\mathcal{B}_1$ classes. In principle, it should be easy to separate knowns from unknowns. However, achieving high accuracy in known classes can be challenging.

In Protocol 2 (P_2), $\mathcal{C}_2$ and $\mathcal{B}_2$ samples should look very alike, as both are types of Hunting dog. Furthermore $\mathcal{A}_2$ should also be similar as includes 4-legged animals and more subclasses of dogs. Since all sets are similar, it can be difficult to classify $\mathcal{C}_2$ and separate $\mathcal{A}_2$. This is the most specialized protocol.

Protocol 3 (P_3) uses a broad spectrum of synsets. Subclasses and superclasses of selected

synsets combine living beings and objects in all $\mathcal{C}_3$, $\mathcal{B}_3$, and $\mathcal{A}_3$. The $\mathcal{A}_3$ can be similar to both $\mathcal{C}_3$ and $\mathcal{B}_3$. In principle, it should be difficult to classify knowns and discard unknowns correctly. This protocol should be the most challenging and it could indicate how well the models generalize in broad open space and intermixed classes.

The protocols ensure that the data for known classes is balanced. Figure 2.8 shows randomly sampled images from the three protocols. $\mathcal{A}_1$ examples (figure 2.8c) are clearly different from $\mathcal{C}_1$ (figure 2.8a) and $\mathcal{B}_1$ (figure 2.8b), but $\mathcal{C}_1$ and $\mathcal{B}$ 1 share some similarities, such as the fur and the face features of four-legged animals. Moreover, in $\mathcal{B}_1$, some bears and foxes can be very similar to dogs. The examples of $\mathcal{C}_2$ and $\mathcal{B}_2$ are very similar (figures 2.8d and 2.8e), although there are differences between the dogs' colors and shapes. Finally, given its design, P_3 is complex and diverse, so it is difficult to see a pattern in the small snapshot (figures 2.8g to 2.8i). We must be cautious with these observations and with the intuition behind the protocols, as normally DNN classifiers try to minimize a defined loss function, and tend to use any available pattern to do so, including patterns that look incomprehensible to humans (Ilyas et al., 2019). So the presence of a "tail" or "4-legged animals" is not more natural to a classifier than any other predictive feature in the data.

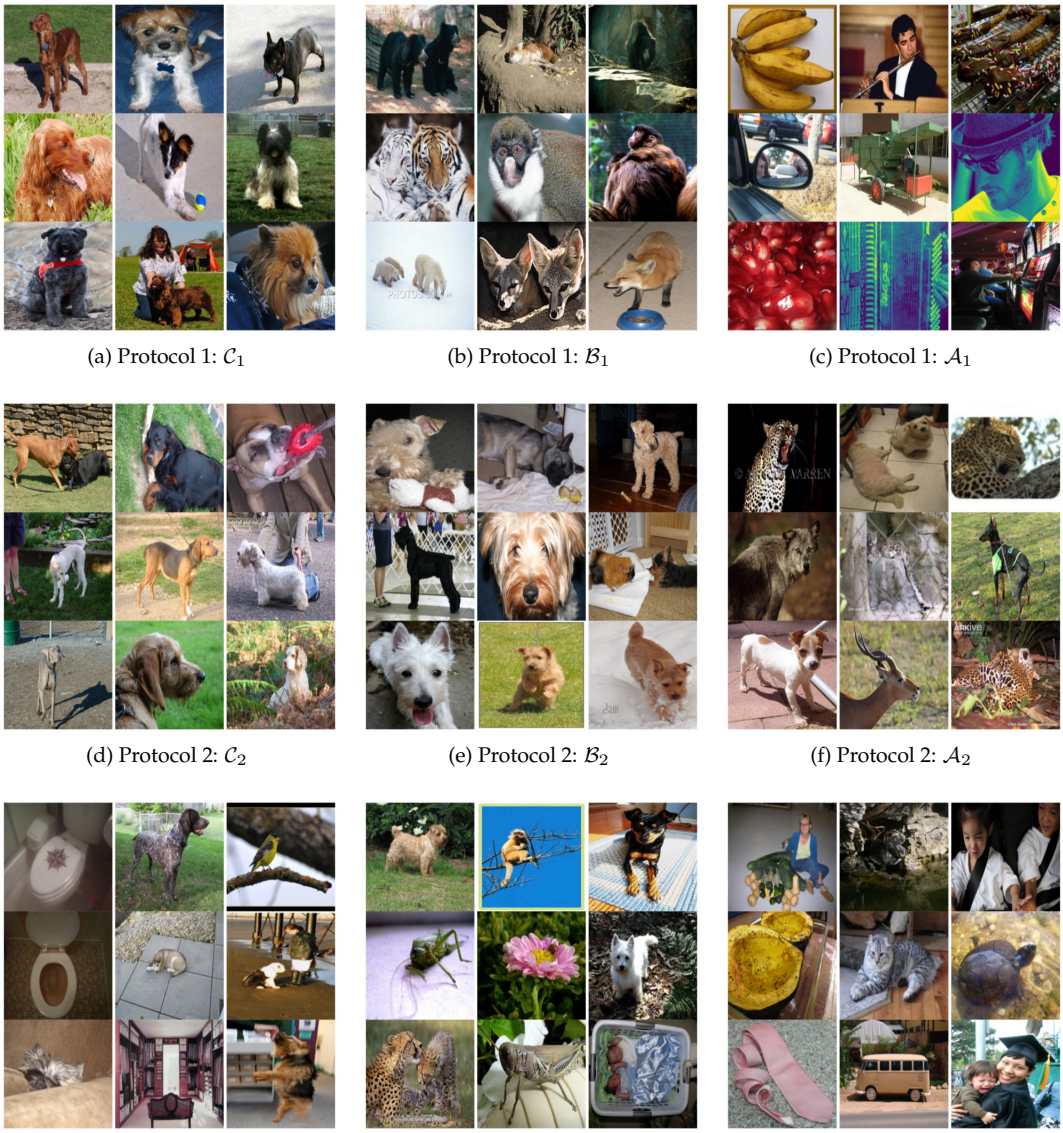
In this work, we will evaluate the performance of a DNN model in the three protocols using softmax, entropic open-set and objectosphere losses. Additionally, we will explore a method to improve the classifier's performance by including adversarial samples in training time. In section 2.5 we describe the methods to generate adversarial samples. Finally, section 2.6 describes the basic notion of adversarial training.

2.5 Adversarial Attacks

DNNs have supported significant progress in image classification and object detection tasks, becoming essential components in autonomous driving, face recognition, or biomedical image systems. Given the broad applications, concerns about the security and robustness of the DNN algorithms have been raised. In particular, the study of robustness and performance of DNN algorithms in the presence of adversarial samples is a topic of great interest.

An adversarial attack is a constructed sample designed to induce misclassification of a DNN classifier. Szegedy et al. (2014) discovered that many models are vulnerable to adversarial samples, pointing out that it is possible to take a correctly classified sample and apply a small perturbation (sometimes imperceptive to the human eye) to get a wrong classification. Figure 2.9 shows examples of adversarial samples for three DNN architectures. Attacks with adversarial samples have been studied extensively because DNN algorithms should perform well under varying levels of uncertainty in the input space (Ben-Tal et al., 2009). For example, when faced with data coming from slightly different distributions like adversarial samples or possibly very distinct distributions, such as an OSR scenario. The phenomenon of adversarial samples is not fully understood. Ilyas et al. (2019) recently approached adversarial vulnerability as a consequence of the supervised learning paradigm, claiming that adversarial vulnerability is a result of the DNN sensitivity to well-generalizing features in the data. These features often look incomprehensible to humans.

Szegedy et al. (2014) formally define an adversarial sample as follows: Given a classifier H_θ where θ are the classifier parameters, an unperturbed (clean) image sample x with ground truth label $\hat{c}$, an adversarial sample x' is constructed by applying the minimal perturbation δ such that x' is classified with a different label $\tilde{c}$: $\arg\min_{\delta} H_\theta(x+\delta) = \tilde{c}$. Finding the perturbation is an



(a) Protocol 1: $\mathcal{C}_1$ (b) Protocol 1: $\mathcal{B}_1$ (c) Protocol 1: $\mathcal{A}_1$

(d) Protocol 2: $\mathcal{C}_2$ (e) Protocol 2: $\mathcal{B}_2$ (f) Protocol 2: $\mathcal{A}_2$

(g) Protocol 3: $\mathcal{C}_3$ (h) Protocol 3: $\mathcal{B}_3$ (i) Protocol 3: $\mathcal{A}_3$

Figure 2.8: IMAGENET OPEN-SET PROTOCOLS SNAPSHOT. Rows correspond to each protocol.

optimization problem:

$$\begin{aligned} &\min_{x'} \|x' - x\|_p \\ &s.t.\ H_\theta(x') = \tilde{c} \end{aligned} \tag{2.6}$$

Where $\|\cdot\|_p$ is a distance function between the image's pixels:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{2.7}$$

Typical choices are $p = \{0, 1, 2, \infty\}$, denoted as L_p. In particular, L_∞ measures the maximum change in any element of x without restricting the number of pixels.

(a) CaffeNet (b) VGG-F (c) GoogLeNet

Figure 2.9: ADVERSARIAL SAMPLES FOR DNN MODELS. Examples for (a) CafeNet, (b) VGG-F and (c) GoogLeNet architectures. Although the perturbation is visible, it does not change the content of the image. However, all images are misclassified with confidence >80% Source Akhtar and Mian (2018).

There are many types of adversarial attacks, we will describe two methods that approximate a solution to equation (2.6). The following section 2.5.1 describe types of adversarial attacks. Section 2.5.2 presents the Fast Gradient Sign method and section 2.5.3 describes iterative methods.

2.5.1 Taxonomy of Adversarial Attacks

Serban et al. (2020) classifies the adversarial attacks based on three criteria: attack goal, attack knowledge and attack strategy.

- Attack goal:
  - Untargeted attack: The objective is to produce a sample classified to any incorrect class.
  - Targeted attack: The sample is misclassified to a specific incorrect class.
- Attack knowledge:
  - White box: the attack has complete knowledge of the model, i.e. an attacker has the necessary information to replicate the model or learn the data-generation distribution.
  - Black box: The attacker can only observe the labels assigned by the DNN for chosen inputs.
- Attack strategy: An attack can be classified using the perturbation type and the algorithm type.
  - Perturbation type:

    - Noise-based: Adding white noise in specific image regions.
    - Geometric-based: Use common geometric transformations like rotation or translation to induce classification error.
  - Algorithm type:
    - Optimization: Use optimization algorithms to find solutions for equation 2.6.
    - Use sensitivity analysis algorithms to find sensitive features and perturb them.
    - Generative: Learn distribution of perturbations using generative models.

We focus on two white-box untargeted attacks that are proven simple yet effective.

2.5.2 Fast Gradient Sign Attacks

Goodfellow et al. (2015) proposes the *Fast Gradient Sign Method* (FGSM), assuming DNN classifiers have a linear behaviour in higher-dimensional space. The perturbation δ is computed as follows:

$$\delta_{FGSM} = \epsilon \cdot sign\left(\nabla_x J(\theta, x, \hat{c})\right) \tag{2.8}$$

Where ϵ is a small scalar that limits pixel perturbation under an L_∞ constrain, $J(\theta, x, \hat{c})$ is the DNN cost function given the parameters θ, image x and target class $\hat{c}$. FGSM perturbs the image to increase the loss of the classifier, the $sign$ function ensures the magnitude of the loss is maximized.

Miyato et al. (2018) proposed a similar method, normalizing the variation of FGSM using L_2 distance:

$$\delta_{FGL_2} = \epsilon \cdot \frac{\nabla_x J(\theta, x, \hat{c})}{\|\nabla_x J(\theta, x, \hat{c})\|_2} \tag{2.9}$$

Rozsa et al. (2016) proposed the *Fast Gradient Value* (FGV), where instead of using the gradient sign, the perturbation uses a scaled raw loss gradient. The objective is to find more diverse adversarial samples by applying larger perturbations to pixels with bigger gradients, in contrast with FGSM, that uses the same perturbation magnitude for all pixels. Compared to FGSM, this slight modification leads to different directions where different adversarial examples exist. Figure 2.10 shows that samples produced by FGSM cover the entire image while FGV generates more focused perturbations.

$$\delta_{FGV} = \epsilon \cdot \nabla_x J(\theta, x, \hat{c}) \tag{2.10}$$

(a) FGSM (b) FGV

Figure 2.10: FGSM EXAMPLE. In (a) the perturbation produced by FGSM. In (b) the perturbation by FGV. Note that FGSM affects more regions of the image.

FGSM methods apply a one-step gradient ascent on the classifier's loss, focusing on the efficiency of perturbation computation rather than achieving high success rates.

2.5.3 Iterative Attacks

Many adversarial attacks find the perturbation through iterative methods. For example, Projected Gradient Descent (Madry et al., 2019), Carlini-Wagner Attacks (Carlini and Wagner, 2017) and DeepFool (Moosavi-Dezfooli et al., 2016). Instead of using ϵ for a single-step gradient ascent, iterative methods utilize small steps while adjusting the direction after each step. Kurakin et al. (2017) propose the *Basic Iterative Method* (BIM) to compute the adversarial sample as:

$$\begin{aligned} x'_0 &= x, \\ x'_{i+1} &= clip_\epsilon\Big\{x'_i + \beta \cdot sign\big(\nabla_x J(\theta, x'_i, \hat{c})\big)\Big\} \end{aligned} \tag{2.11}$$

Where x'_i denotes the adversarial sample at iteration i, β is a small step size, $clip_\epsilon$ clips pixel values of intermediate results after each iteration to ensure they are in an ϵ-neighborhood of the initial image x, this method finds stronger adversaries at a higher computational cost. A natural extension replaces $\hat{c}$ for a target class c^t in equation (2.11). For instance, the *iterative least-likely class method* chooses c^t to be the least probable class predicted by the classifier.

2.6 Adversarial Training

Given the facility of adversarial samples to fool a DNN classifier, many defence mechanisms have been proposed to increase robustness, for example, Network distillation (Papernot et al., 2016), Input transformation (Guo et al., 2017), Adversarial Detecting (Metzen et al., 2017) or Dynamic Adversarial Training (Wang et al., 2021). Goodfellow et al. (2015) proposed the basic notion of adversarial training, which consists in including adversarial samples in training time, keeping the original class label $\hat{c}$ of the clean image x, and modifying the loss function as follows:

$$\begin{aligned} x' &= x + \epsilon \cdot sign(\nabla_x J(\theta, x, \hat{c})) \\ \tilde{J}(\theta, x, \hat{c}) &= \gamma J(\theta, x, \hat{c}) + (1-\gamma) J(\theta, x', \hat{c})) \end{aligned} \tag{2.12}$$

Where γ is a hyperparameter that weights the contribution of each term. $\tilde{J}$ intends to make the classifier robust against a modified x'. Note that the adversarial samples are continually updated to face the current version of the model.

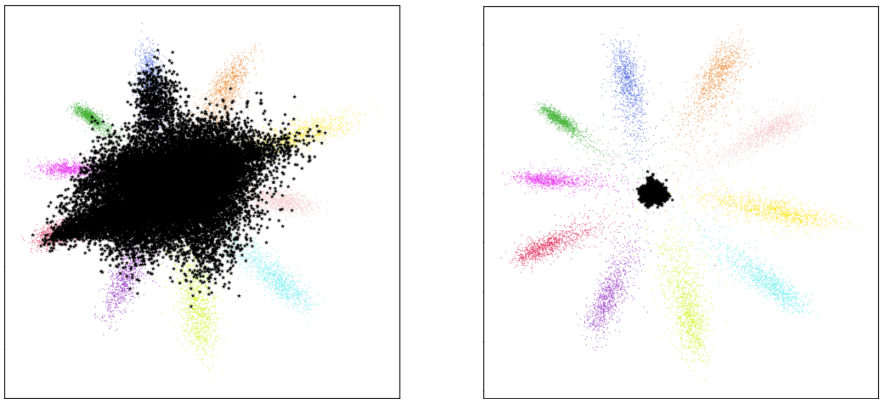
2.6.1 Adversarial Training in OSR

Recently Schnyder (2021) investigated a method to include adversarial training into OSR scenarios. The approach consists of training a classifier using known $\mathcal{C}$ classes and including adversarial samples as $\mathcal{B}$. Hoping that the adversarial samples are good open-space representatives. The model is trained using the entropic open set loss on data from the MNIST dataset. Furthermore, several adversarial attacks were tested, including FGSM, Projected Gradient Descent, Carlini-Wagner and LOTS. The adversarial samples are included after the network correctly learns to classify the original clean sample.

The method is tested on LeNet++ (Wen et al., 2016). This convolutional network allows to visualise the feature space, since the last layers reduce the dimensionality of the features to 2 dimensions. Additionally, training LeNet++ on MNIST data allowed to try several adversarial attacks. Figure 2.11 shows an example of the feature space of 10 MINST known classes and EMNIST unknown classes. The approach showed promising results since the reported performance of the classifier improved when compared with the same classifier trained on only $\mathcal{D}_c$ data. Especially,

training with projected gradient descent (PGD-attack) adversaries improved the classification accuracy.

However, the tested datasets are small and do not include natural images. Inspired by this approach, we implement adversarial training in a similar way. However, we extend the work to more complex OSR scenarios, using natural images and the size of the dataset requires an efficient training method, our approach will be discussed in section 3.2.



(a) Training without adversarial samples (b) Training with adversarial samples

Figure 2.11: LENET++ FEATURE SPACE. Feature space of the LeNet++ classifier when trained on MNIST. In (a) the classifier was trained only with $\mathcal{D}_c$ samples, the colored points distinguish the class and the black points are EMNIST samples used as $\mathcal{D}_a$. In (b) the classifier was trained with adversarial samples, the unknown samples are pushed to the origin of the feature space. Source Schnyder (2021).

Chapter 3

Methods

This work has two main objectives:

- Investigate the performance of state-of-the-art approaches in complexity-varying OSR scenarios. We evaluate the performance of DNNs trained on the ImageNet Open-set protocols using entropic open-set and objectosphere losses.
- Investigate the impact of training with adversarial samples as hard negatives applied to the OSR scenario.

The significant difference between our approach and previous methods is that we evaluate the performance on the ImageNet protocols instead of related work approaches that use smaller and more uniform datasets such as MNIST and CIFAR. Similarly, we use more $\mathcal{C}$ and $\mathcal{B}$ classes. The ImageNet protocols provide several similarity levels between known and unknown classes, making this evaluation closer to real-world scenarios.

The intuition behind the second objective is to create synthetic data to represent $\mathcal{B}$ classes from the open space. Our approach focuses to utilize strong negatives to improve a classifier's robustness to unseen classes $\mathcal{A}$. Since the set $\mathcal{A}$ is infinite, the choice of hard $\mathcal{B}$ classes is crucial. Instead of modelling the open space with predefined $\mathcal{B}$ classes, we will use adversarial images. Ideally, the classifier should learn to recognise classes of interest while discarding adversarial samples, which are very similar in pixel space; therefore, if the model can discard similar synthetic samples, it could be more robust to further samples from the open space. Finally, we also investigate the effect of training a classifier with $\mathcal{B}$ classes and adversarial samples.

The training method is fundamental, as using adversarial examples is computationally expensive and can lead to stability issues. Hence, we aim for efficient training methods, reusing gradient calculations for sample generation and filtering the generation of adversarial samples. All the implementation is done using the PyTorch framework [1].

This chapter is organised as follows: In section 3.1 we describe the DNN architecture and section 3.2 describes in detail the training methods used in our approach.

3.1 DNN Classifier

In this work we use ResNet50, a DNN architecture that addresses a *degradation problem* shown by previous models; when DNNs started converging, the accuracy was saturated and then rapidly decayed, this condition was worsened in deeper networks (He et al., 2016). Moreover, ResNet architectures allow deeper models to avoid degradation, using residual learning. If $m(x)$ is the

[1] https://pytorch.org/docs/stable/index.html

desired mapping that a DNN is supposed to learn, residual learning fits the residual mapping $f(x) := m(x) - x$ instead, and the original mapping is recast to $f(x) + x$. Optimising the residual map is more manageable than optimising the original map since the residual map avoids vanishing gradient issues.

The residual mapping is implemented by adding shortcut connections between stacked layers termed residual blocks. Figure 3.1 shows a ResNet50 residual block that comprises a shortcut connection, convolutional layers, batch normalisation layers and activation layers. The shortcut connection could also include convolutional layers to keep dimensions consistent.

ResNet architectures perform very well on object recognition and classification tasks, even winning the 2015-ILSVRC. We will use ResNet50 because it achieves high classification accuracy with less complexity (about 0.85 billion parameters) than other state-of-the-art architectures. Larger ResNet architectures like ResNet101 (1.7 billion) and ResNet152 (19.5 billion) only reduce the top-1 classification error by 1.4% at a high parameters complexity cost.

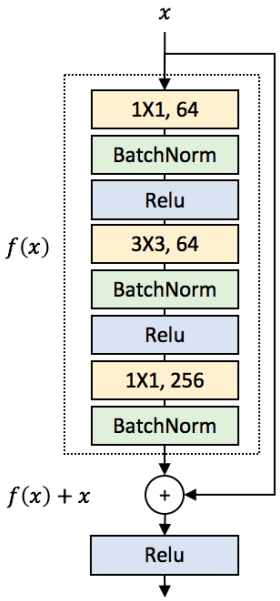


Figure 3.1: RESIDUAL BLOCK. The first convolutional layer (yellow) consists of kernel size 1x1 and 64 channels, batch normalization (green) is applied before the activation layer (blue). Source He et al. (2016).

Figure 3.2 shows the architecture of ResNet models. Note that the last layer is a 1000-d fully connected layer. For simplicity, we will reduce this layer dimension to fit the number of known classes C, but other values (for example, to create a bottleneck) could impact the classifier performance. We will extract the deep features from this layer's activations. Lastly, we append a fully connected layer to obtain the logits activations.

3.1.1 Batch Normalisation

The batch normalisation layers (BN) play an essential role in our experiments. BN layers proposed by Ioffe and Szegedy (2015) make the training process converge faster and avoid exploding gradient issues. The objective is to reduce changes in the distribution of the network activations due to changes in the network parameters during training. Moreover, BN normalises the activations

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3.2: RESNET ARCHITECTURES. The architectures use only two types of stacked blocks, of two and three convolutional layers. Although ResNet 152-layer has 11.3×10^9 FLOPs, still is a competitive value For example VGG19 has 19.6×10^9. Source He et al. (2016).

by estimating the mean and variance of each activation in the current batch. Consider a batch $\mathcal{S} = \{x_1, \ldots, x_M\}$ of size M, the mean $\mu_\mathcal{S}$ and variance $\sigma_\mathcal{S}$ are calculated as follows:

$$\mu_\mathcal{S} = \frac{1}{M}\sum_{i\in\mathcal{S}} x_i \qquad ; \qquad \sigma_\mathcal{S} = \sqrt{\frac{1}{M}\sum_{i\in\mathcal{S}}(x_i - \mu_\mathcal{S})^2 + r} \tag{3.1}$$

Where r is a small constant for numeric stability. Then a normalised activation is defined as:

$$\hat{x}_i = a \cdot \frac{x_i - \mu_\mathcal{S}}{\sigma_\mathcal{S}} + b \tag{3.2}$$

Where a and b are two learnable parameters. The activation values use a moving average and variance that depend on both, the current sample and the other samples in the batch. Additionally, the statistics are accumulated during the training process to get estimations over the complete dataset. However, moving statistics are not desirable at inference time since the model's output should be deterministic. Hence, the BN-layer's parameters and estimators are frozen during inference, and the accumulated mean and variance during training are used at testing.

Note that BN assumes that the inference and training data follow the same distribution, as it utilises the accumulated statistics. However, in an OSR scenario, the collected mean and variance are incorrect, as data from new distributions are present at inference time. Furthermore, the issue worsens when training with adversarial samples since $\mu_\mathcal{S}$ and $\sigma_\mathcal{S}$ would include values from purposely perturbed samples to change the data distribution. Hence, the shifts in $\mu_\mathcal{S}$ and $\sigma_\mathcal{S}$ generate performance degradation in both validation and testing. Figure A.3 shows an example of the performance degradation caused by the accumulated statistics in BN layers.

We address this issue by stopping accumulating statistics from batches containing adversarial images; if an adversarial image is present in the batch, BN layers use the accumulated statistics up to the current iteration like in inference time. However, in mixed batches (containing clean and adversarial samples), activations from some clean samples will be left out of the estimation. Note that we do not freeze the learnable parameters. This can be efficiently done using the `mode.eval()` and `mode.train()`[2] functions in PyTorch.

[2]https://pytorch.org/docs/stable/generated/torch.nn.Module.html

3.2 Adversarial Training for OSR

To include the adversarial samples, we adapt the adversarial loss proposed by Goodfellow et al. (2015) (eq. 2.12). Let x be a clean image, $\hat{c}$ the corresponding target label, θ the current network parameters, x' an adversarial sample fabricated from x, $J(\theta, x, \hat{c})$ the loss function, for simplicity we refer to the loss as $J(x, \hat{c})$ since the dependency on θ is clear. The loss for adversarial training is calculated as follows:

$$\tilde{J}(x, c) = J(x, \hat{c}) + J(x', c = -1) \tag{3.3}$$

Note that for x', we reserve $c = -1$ to define: $x' \in \mathcal{D}'_b$. Objectosphere and entropic open-set are not designed to produce an unknown class label. Instead, they procure to increase the separation between $\mathcal{C}$ and $\mathcal{U}$, so that it is possible to define threshold values for the softmax scores. Naturally, the total gradient is the sum of the clean and adversarial loss gradients:

$$\nabla_\theta \tilde{J}(x, c) = \nabla_\theta J(x, \hat{c}) + \nabla_\theta J(x', c = -1) \tag{3.4}$$

Recall that for FGSM in equation (2.8), x' depends on the loss gradient with respect to the input $\nabla_x J(x, \hat{c})$ and the current network parameters θ. From an implementation perspective, the adversarial sample x' is obtained only after performing the forward pass and the backward pass of x. Moreover, x' should be used in the current iteration before the parameters θ are updated.

This approach increases the training time by calculating two losses in the same iteration with sequential data. However, our implementation calculates ∇_θ and ∇_x in one backward pass using the Pytorch AutoGrad package [3], since the package allows to access multiple gradients on the same backward pass. The adversarial samples should be produced on the fly in the current iteration, and an additional forward pass is needed to compute $J(x', c = -1)$. AutoGrad automatically accumulates the gradients. Finally, recall that we will stop collecting batch normalisation statistics before using adversarial samples.

The approach can be applied to losses that consider unknowns and reserve unknown class labels like entropic open set, objectosphere or softmax with garbage class, even mixtures between different losses. Furthermore, this approach can also consider training cases with both $\mathcal{B}$ and adversarial images. However, note that we are introducing more unknown samples to the training data. For example, in a primary case, we can create one adversarial sample for every clean sample, which duplicates the actual size of the dataset, surely the loss hyperparameters will require revision.

3.2.1 Sample Filtering

A classifier learns more from using an adversarial of a correctly classified sample than from a misclassified one. In fact, an adversarial sample of a class that is not learned deviates from the original adversarial purpose. Consequently, we add sample filtering, perturbing only correctly classified clean samples. Sample filtering avoids duplicating the actual size of the dataset. From equation (3.3), our intuition is that $J(x, \hat{c}) \geq J(x', c = -1)$ when x is correctly classified; in this case, the DNN could effectively learn to reject x'.

3.2.2 Waiting Epochs

Using a similar intuition as sample filtering, we consider including perturbed samples at different training moments. The objective is to wait for certain epochs N_{wait} before including the adversaries, hoping that the model has already learned to classify some $\mathcal{C}$ classes. An edge case of this

[3] https://pytorch.org/docs/stable/autograd.html

approach is to start using perturbations from the beginning of the training. However, we also evaluate if letting the classifier train only with clean images for a certain epochs improves the performance. Another edge case is to take a classifier fully trained and then include adversarial samples in a fine-tuning fashion. For example, we take a fully trained DNN with softmax loss and fine-tune it using an open-set loss with adversarial images.

We avoid using iterative methods like BIM (eq. 2.11) due to the intrinsic overhead of doing iterations. However, doing an additional forward and backward pass is computationally expensive. Algorithm 1 presents the pseudocode of our approach. It assumes $filter()$ gets the indices of correctly classified samples, and $perturb()$ creates adversarial examples.

Algorithm 1 Adversarial Training for OSR using FGSM attacks

Require: Dataset, training epochs N_{epochs}, model H_θ, loss function J, $filter()$ function, $perturb()$ function, optimiser, learning rate η.

for epoch = $1, N_{epochs}$ **do**
  for batch $B \subset$ Dataset **do**
    Enable BN layers statistics collection
    Get images and labels in B: x_B, c_B
    Get model's prediction: $c_B = H_\theta(x_B)$
    Calculate loss: $l = J(\hat{c}_B, c_B)$
    Do Backward pass on l to get: ∇_θ, ∇_x
    if epoch $>= N_{wait}$ **then**
      Filter samples: $idx = filter(\hat{c}_B, c_B)$
      if $idx \neq \emptyset$ **then**
        $x_{B\_corr} \leftarrow x_B[idx],\ \nabla_{x\_corr} \leftarrow \nabla_x[idx]$
        Disable BN layers statistics collection
        Calculate adversarial samples: $x'_B = perturb(x_{B\_corr}, \nabla_{x\_corr})$
        Get model's prediction: $c'_B = H_\theta(x'_B)$
        Calculate loss: $l' = J(c'_B, \vec{-1})$
        Do Backward pass on l' to get: ∇'_θ
        Accumulate gradients: $\nabla_\theta \leftarrow \nabla_\theta + \nabla'_\theta$
      end if
    end if
    Do optimiser step: optimiser$(\theta, \nabla_\theta, \eta)$
  end for
end for

Additionally, on top of algorithm 1, an early stopping method is implemented on the validation data. We can prioritise to obtain the maximum separation between $\mathcal{C}$ and $\mathcal{B}$ samples. For example as a binary classifier. All known samples are treated as positives, and the unknown as negatives.This way, we can find the maximum separation at the cost of classification accuracy.

3.2.3 Loss function

We use entropic open set, objectosphere and softmax cross-entropy losses. The softmax loss is used as a benchmark, providing the closed-set reference point to all experiments. From the implementation perspective, we use the default softmax cross-entropy loss from Pytorch [4]. Given that the known classes of the protocols are balanced, we do not add class weights. Entropic open

[4]https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

set loss and objectosphere loss implementations are based on the VAST library [5]. These losses are efficiently calculated in VAST, yet slight modifications are needed to fix issues in edge cases. For example, when a batch contains only samples from $\mathcal{B}$ classes.

In case of an unbalanced dataset with respect to the $\mathcal{B}$ classes, the parameter w of the entropic loss (eq. 2.4) needs to be tuned. Grid search can be implemented to find the best parameter to obtain balance between classification accuracy and rejection of unknowns.

For objectosphere (eq. 2.5) ξ and α affect the loss value. Note that ξ could be seen as a threshold that scales the deep feature's magnitude. Still, we believe α is more important to increase separability between $\mathcal{C}$ and $\mathcal{U}$ because it regularises the contribution of the deep feature's magnitudes to the loss. For simplicity, we variate α and keep a fix ξ. Moreover, in objectosphere, the J_E term enforces correct classifications intra $\mathcal{C}$ classes and separation from $\mathcal{B}$, but the deep features term only enforces unknowns separability. Hence, large α values could overshadow J_E and decrease the classifier performance.

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

Chapter 4

Experiments and Performance

This chapter describes the experiments and performance. Section 4.1 present the evaluation metrics used across all experiments. Section 4.2 describes the datasets and transformations used in the images. Finally, section 4.3 presents a description of the experiments and the performance achieved in each one.

4.1 Evaluation Metrics

4.1.1 Open-Set Classification Rate

Good evaluation metrics in OSR scenarios should address two requirements: the classifier's capacity to reject samples from unknown classes and the correct classification of known classes. Typical classification metrics only attend the second requirement. Recently Dhamija et al. (2018) proposed the *Open-Set Classification Rate Curve* (OSCR), which addresses the two fronts. OSCR utilises an operating threshold τ to calculate the Correct Classification Rate (CCR) and the False Positive Rate (FPR). CCR is the ratio of $\mathcal{D}_c$ samples in which the target class $\hat{c}$ has the maximum probability, and it is greater than τ. FPR calculates the ratio of $\mathcal{D}_u$ samples that are classified as any of the known classes, with a probability greater than τ.

$$\begin{aligned} FPR(\tau) &= \frac{\left|\{x \mid x \in \mathcal{D}_a \wedge max_c P(c|x) \geq \tau\}\right|}{|\mathcal{D}_a|} \\ CCR(\tau) &= \frac{\left|\{x \mid x \in \mathcal{D}_c \wedge arg\,max_c P(c|x) = \hat{c} \wedge P(c|x) \geq \tau\}\right|}{|\mathcal{D}_c|} \end{aligned} \tag{4.1}$$

Note that when τ is the smallest $P(c|x)$ on $\mathcal{D}_c{}'$, CCR is equivalent to the close-set accuracy. Furthermore, CCR is calculated only on samples from $\mathcal{D}_c$, avoiding dataset bias issues.

4.1.2 Confidence Score

We are interested in evaluating the capacity of the classifier to separate between samples from $\mathcal{C}$ and $\mathcal{U}$. For instance, the softmax score of a correctly classified sample should be as close to 1 as possible, and unknown samples should have maximum entropy, i.e., the softmax score of every class is $\frac{1}{|\mathcal{C}|}$. The confidence score resumes these observations as follows:

$$conf(x) = \begin{cases} S_{\hat{c}}(x), & \text{if } x \in \mathcal{D}_c \\ 1 - \max\limits_{c \in \mathcal{C}} S_c(x) + \frac{1}{|\mathcal{C}|}, & \text{if } x \in \mathcal{D}_a \end{cases} \tag{4.2}$$

Where $S_{\hat{c}}(x)$ is the softmax score of the correct class. An unknown sample has high confidence score if the maximum softmax score among the class predictions is low, ideally $\frac{1}{|\mathcal{C}|}$.

4.2 Data preprocessing

Table 4.1 shows the dataset size of each protocol. The biggest training dataset is from P_3 containing 202'138 images, the smallest one is P_2 with 60'684. Table 4.2 shows the samples per class distribution. All training sets are balanced as they contain approximately 1000 samples per known class. P_2 has only 31 classes, all corresponding to subclasses of dogs. Given its size, P_2 is used as a test scenario during algorithm development and parameter tuning.

	Train			Validation			Test		
Protocol	$\mathcal{D}'_c$	$\mathcal{D}'_b$	$\mathcal{D}'_a$	$\mathcal{D}'_c$	$\mathcal{D}'_b$	$\mathcal{D}'_a$	$\mathcal{D}_c$	$\mathcal{D}_b$	$\mathcal{D}_a$
1	116'212	69'680	-	29'061	17'420	-	5'800	3'350	8'300
2	30'629	30'055	-	7'661	7'517	-	1'550	1'500	2'750
3	161'661	140'477	-	40'425	35'122	-	7'950	6'850	5'800

Table 4.1: IMAGENET PROTOCOLS SIZE. The data size of each protocol. P_1 and P_3 have in total more known samples than unknowns. However, P_2 has a similar total number of known and unknown samples. Source Bhoumik (2021)

All images from the ImageNet ILSVRC2012 dataset are stored as JPEG files, distributed on several directories that define the class label, i.e., all images on the same directory are from the same synset. A picture is in a particular category c_k if it contains a c_k entity, but an image might contain different entities in the background. Each ImageNet protocol provides the class labels and the corresponding paths to the images of such class. The class label -2 is reserved for all $\mathcal{D}_a$ samples, class label -1 is assigned to all samples from $\mathcal{B}$ classes, and the labels of samples of $\mathcal{C}$ classes are integer values $\{0, \ldots, |\mathcal{C}_i|\}$, where $|\mathcal{C}_i|$ represents the number of known classes in protocol i.

		Train		Validation		Test	
Protocol	$\lvert\mathcal{C}\rvert$	Mean	Std	Mean	Std	Mean	Std
1	116	1'001.8	108.1	250.5	26.9	50.0	0.0
2	31	1'001.7	120.2	250.5	29.9	50.0	0.0
3	152	1'027.9	56.2	257.0	14.0	50.0	0.0

Table 4.2: KNOWN CLASSES DISTRIBUTION. Distribution of the known classes over the dataset splits. All protocols are constructed to have aprox. 1000 samples per known class in the training set.

The images have different sizes and resolutions. To standardise the input to the classifier, we apply standard transformations like Krizhevsky et al. (2012). A raw JPEG image is transformed to RGB format; then, the image is scaled, so the smaller dimension is resized to 256 pixels. We

use bilinear interpolation in this transformation. Next, a 224x224 random crop is extracted. Additionally, we use basic data augmentation, i.e. a random horizontal flip with probability $p = 0.5$. However, we do not modify the colour intensities suggested by Krizhevsky et al. (2012). Finally, the image is transformed into a normalised tensor with pixel values between [0,1] and shape (3, 224, 224). It is a common practice to normalise the images with pre-calculated mean and standard deviation of the ImageNet dataset (improves training stability and in some casses accuracy) Nevertheless, we will keep a [0,1] range since we add L_∞-bounded perturbations and clip values to [0,1] range when using adversarial samples. This process is handled by a custom class that inherits the Pytorch Dataset class [1].

Finally, the data is shuffled and segmented into batches using a standard Pytorch data loader before sending it to the classifier. Typically, data pre-processing is a bottleneck, so we use several workers to prepare the data, usually 4 per GPU. Moreover, we explored Nvidia Dali [2], a library that accelerates preprocessing by using CPU cores and GPUs to prepare the data. The data is handled using pipelines, where several transformations are available such as reading files, scaling, cropping and colour transformations. The main difference is that a pipeline can execute transformations in a GPU, internally DALI perform transformations using CUDA tensors. Additionally, DALI uses an iterator similar to the Pytorch data loader. Equivalent transformations were created with Dali; although there is an improvement in the training speed, it is not significant for the scale of this data. Nevertheless, the library can be convenient for larger-scale datasets.

4.3 Experiments

We divided our experiments into two phases; that are closely related to our main objectives. The first phase evaluates the performance of entropic open-set and objectosphere losses on each protocol. The findings of this phase are used as benchmarks for the second phase. Finally, the second phase tests our adversarial training for OSR approach. This phase includes generating adversarial samples, assessing performance for fixed and varying perturbation magnitudes, and varying the training epoch at which the adversarial images are formed.

P_2 is the smallest protocol and due to the computation cost, we reduced the number of experiments for P_1 and P_3. Initially, an experiment is tested in P_2, considering evaluation metrics, stability and training time. In particular, training with adversarial samples might cause divergence problems. Finally, if the P_2 experiment is successfully finished and the evaluation metrics indicate a promising approach, we replicate it to other protocols.

4.3.1 Phase 1: Training with clean images

This phase presents the experiments for training with softmax, entropic open-set and objectosphere losses. The validation dataset is used to evaluate the training progress, and it requires a defined threshold τ. We use $\tau = 0.5$ on all experiments. However, the OSCR curve evaluates the τ on the range [0,1]. Furtermore, we denote S_i, as the classifier trained with softmax loss in protocol i, similarly E_i for entropic open-set and O_i for objectosphere losses trained in protocol i. Common hyperparameters over all experiments are:

- Epochs: 100
- Optimiser ADAM: (default parameters)
- Learning rate: 1e-3

[1] https://pytorch.org/docs/stable/data.html
[2] https://docs.nvidia.com/deeplearning/dali/user-guide/docs/pipeline.html

- Batch size: 64
- Seed: 343443 (Same in all sources of randomness)
- Validation τ: 0.5

Table 4.3 shows the loss hyperparameters for every protocol. These parameters are found using grid search, an example is presented in the supplementary figures A.1 and A.2.

Protocol	Entropic	Objectosphere	
	w	α	ξ
P_1	1	1	10
P_2	1	0.01	10
P_3	0.1	0.1	10

Table 4.3: LOSS HYPERPARAMETERS. In the objectosphere loss ξ as fixed to 10.

Table 4.4 shows the results of phase 1. The confidence of every entropic open-set and objectosphere classifier is greater than the corresponding softmax. However, for $FPR = 1$, the CCR of softmax classifiers is higher in every protocol. Figure 4.1 shows the OSCR curves of the trained classifiers. These models are used as benchmarks in the second phase. To determine the generalisation capacity of the models, in this phase, we depict the curves of validation and test sets. In the validation set, we use samples from $\mathcal{B}$ to calculate FPR. While in the test set, only $\mathcal{A}$ samples are employed, excluding possible samples from $\mathcal{B}$ classes. This compares the model's performance on unseen samples from the open space.

Figure 4.2 shows the normalised histograms of the score and deep features magnitude of the $\mathcal{C}$ samples and the $\mathcal{A}$. In the histograms is possible to determine the separation between the samples.

		CCR at FPR of				
Experiment	Conf	10^{-3}	10^{-2}	10^{-1}	0.5	1
Protocol 1						
S_1	0.5283	**0.2848**	**0.4798**	0.6378	**0.7284**	**0.7329**
E_1	0.7867	0.1040	0.1040	**0.6659**	0.7067	0.7086
O_1	**0.8088**	0.1064	0.4488	0.6212	0.6250	0.6260
Protocol 2						
S_2	0.3732	-	-	0.4477	0.6645	**0.7458**
E_2	**0.7195**	-	0.2265	0.4335	0.6065	0.6497
O_2	0.6507	-	**0.2684**	**0.5135**	**0.6994**	0.7361
Protocol 3						
S_3	0.4151	-	-	0.5507	0.7422	**0.7989**
E_3	**0.7063**	-	0.1729	**0.5925**	**0.7536**	0.7884
O_3	0.6947	-	**0.2514**	0.5829	0.7309	0.7676

Table 4.4: RESULTS OF PHASE 1 IN TEST SET. The last column $FPR = 1$ is equivalent to closed set classification. All values are calculated on the test set. Moreover, the maximum CCR at a FPR of each protocol is highlighted in bold. P1 is the only protocol that reaches $FPR = 10^{-3}$.

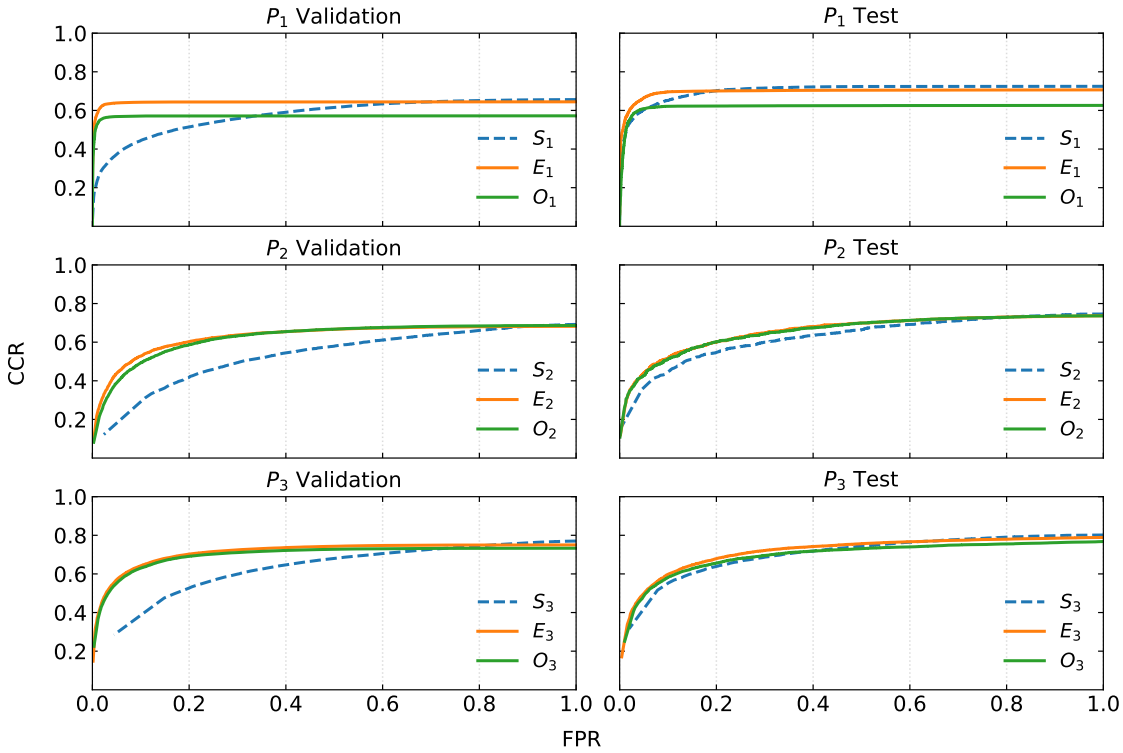
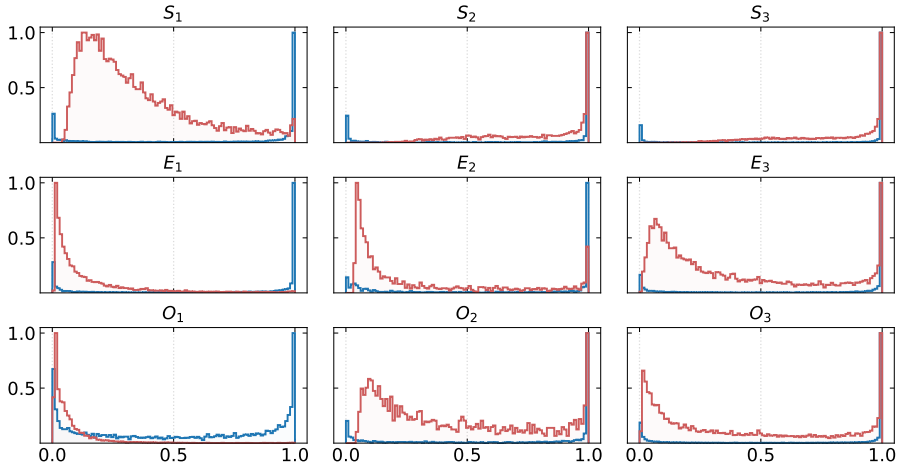
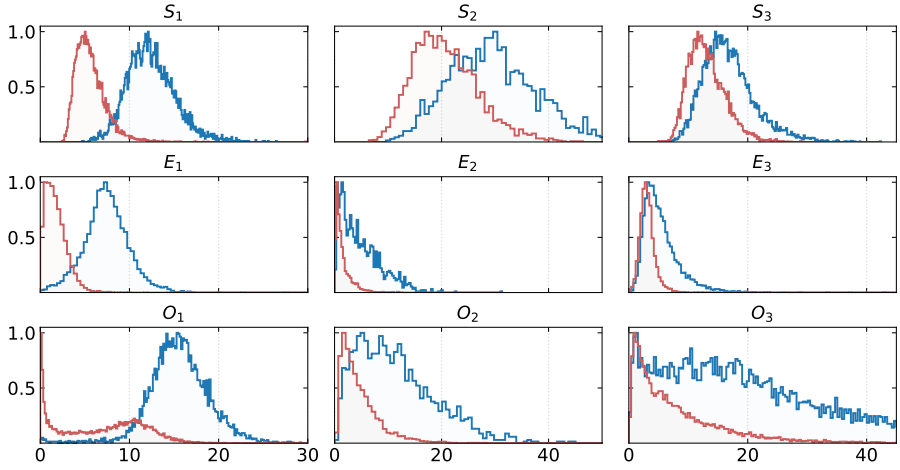


Figure 4.1: OSCR BENCHMARK. OSCR curve for every protocol (row) for validation and test sets (columns). **Validation Sets:** E_1 and O_1 get quickly saturated, and the classifiers achieve high CCR values at low FPRs. E_2, O_2, E_3 and O_3 have similar behaviours, showing better performance than the traditional softmax in every protocol. At $FPR = 1$ all entropic and objectosphere losses achieve close values to the softmax loss (Excepting O_1). **Test Sets:** The performance of all entropic and objectosphere losses are similar to the softmax. This implies that for $\mathcal{A}$ samples, entropic open-set and objectosphere drop rejection capacity. However, note E_1 performs better than S_1 at $FPR \leq 0.2$. Similarly, E_2 and O_2 perform better at $FPR \leq 0.6$. Note that in P_1 all losses are saturated at $FPR \geq 0.2$, this means in P_1 it is easy to reject samples from $\mathcal{A}$ classes. Finally, note that at $FPR = 1$ the losses have slightly bigger CCR values. This might be caused by the sampling process of the protocols.



(a) Softmax Score



(b) Features Magnitude

Figure 4.2: NORMALISED HISTOGRAMS. **(a) Score Histogram:** In blue is the score of the predicted class of the $\mathcal{D}_c$ samples. In red is the maximum class score of the $\mathcal{D}_a$. In P_1, the S_1 classifier assigns lower scores to the unknowns, E_1 and O_1 improve this behaviour. P_2 is more challenging, S_2 gives high scores to unknown values, E_2 and O_2 can reduce the scores of $\mathcal{D}_a$ samples. However, there still are overlaps for scores close to 1. The classifiers in P_3 have similar behaviour. **(b) Features magnitudes histogram:** The classifiers for P_1 show separation between the samples. Note that E_1 pushes the norms of the unknowns to values close to 0. O_1 increases the separation, but there is a slight overlap for magnitudes around the xi value. The distributions for P_2 show more overlap between the samples. For example, E_2 reduces the magnitudes of both $\mathcal{D}_c$ and $\mathcal{D}_a$, O_2 corrects this behaviour by increasing the magnitudes of $\mathcal{D}_c$. In P_3, the magnitudes overlap in S_3 and E_3. However, O_3 can increase the magnitudes of $\mathcal{D}_c$.

4.3.2 Phase 2: Adversarial Training

In this phase we include adversarial samples using the adversarial training in algorithm 1. We investigated three main cases:

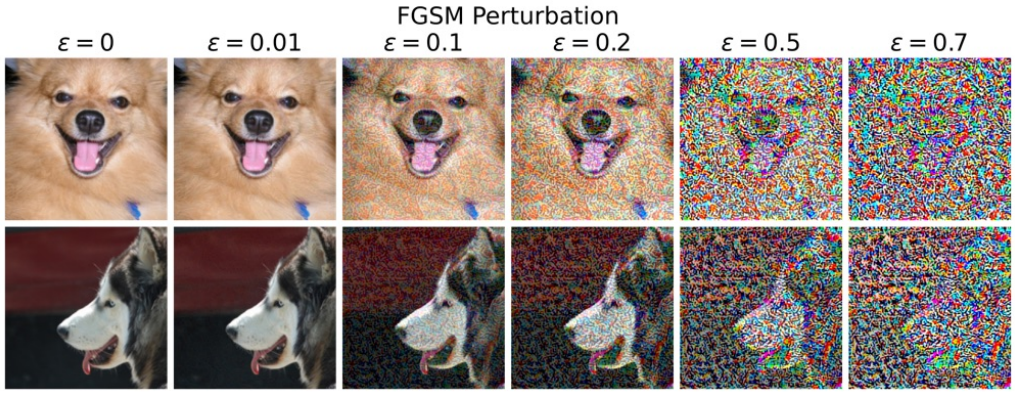
- Case 1: Determine if adversarial samples can increase the robustness of a classifier to unknown samples. We train an entropic open-set or objectosphere classifier with only adversarial samples as unknowns. No other samples from $\mathcal{B}$ are used. Since the training dataset includes only samples from $\mathcal{C}$ classes, the results are compared with a softmax benchmark. We call these experiments *basic adversarial*.
- Case 2: Determine if adversarial samples can increase the performance of a previously trained model. We use the softmax classifier of phase 1 and fine-tune it using the perturbed images. We use either entropic open-set or objectosphere losses with only adversarial samples as unknowns to fine-tune the network. We refer to these experiments as *fine-tune closed set*. In this case, the performance is compared with the original pre-trained classifier.
- Case 3: Determine if adversarial samples plus $\mathcal{B}$ samples increase the robustness of an open-set classifier. In this case, we take an open-set trained classifier of phase 1 as the pre-trained network and add only adversarial samples as unknowns for fine-tunning. We refer to this experiment as *fine-tune open-set*.

In the experiments, we test different adversarial perturbation magnitudes. Additionally, we use random noise perturbations as reference. Figure 4.3a shows examples of perturbed images for distinct ϵ values. Similarly, we explore the progressive increase of the difficulty of the attack; for this, we use an epsilon decaying strategy. The value of epsilon depends on μ, the decay factor, and the current epoch. Additionally, as mentioned in the description of algorithm 1, we modify the start epoch ep on which the perturbed images are generated. The list of hyperparameters is extended as follows:

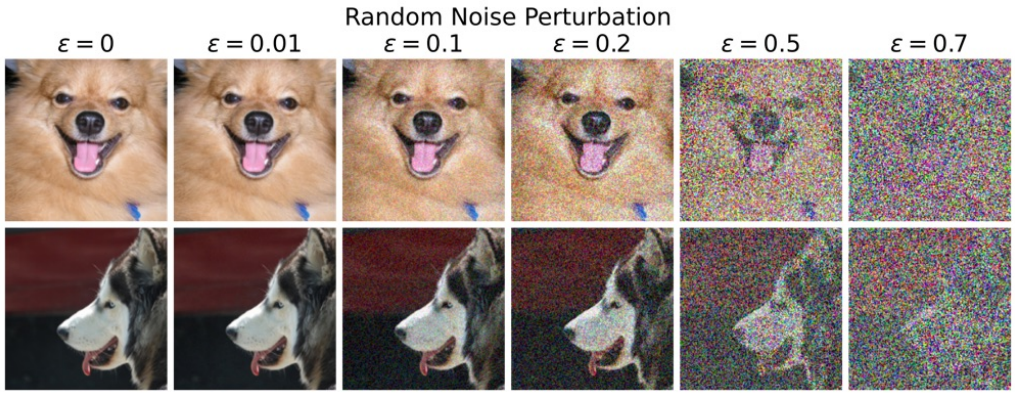
- Attacker:
  - $FGSM$: Fast Gradient Sign.
  - RN: Random Noise.
- Filtering type:
  - $filter$: Perturb corrected classified images.
  - $full$: Perturb all images.
- ϵ: Magnitude of the perturbation, applies for $FGSM$ and RN
- μ: Decaying factor
- ep: Starting epoch of adversarial samples generation.

We denote the experiments as follows: $E_i, \epsilon = e, \mu = u, ep = n, attacker_{(type)}$. For example, $E_1, \epsilon = 0.5, ep = 33, FGSM_{(full)}$ is a classifier trained in P_1 with entropic loss, using $\epsilon = 0.5$, the image perturbation starts from epoch 33, the samples are disturbed by FGSM without filtering the images $(full)$. Additionally, to reduce the nomenclature, assume that $\mu = 1$, attacker=$FGSM$, $ep = 0$ and type=$filter$ unless another value is explicitly written.

Section 4.3.3 shows the results of P_2. We present this protocol first as our approaches were tested in P_2 before replicating them in P_1 or P_3. Next, section 4.3.4 present the results of P_1 and section 4.3.5 the results of P_3.



(a) FGSM Perturbations



(b) Random Noise Perturbations

Figure 4.3: IMAGE PERTURBATIONS. In (a), the FGSM perturbations have a pattern. At $\epsilon = 0.01$ is challenging to distinguish the perturbations. In $\epsilon = 0.1$, the perturbations are observable. In (b), the random perturbations do not follow a pattern; from $\epsilon = 0.1$, the image is blurred.

4.3.3 Protocol 2

The results of P_2 are presented in table 4.5. We report our results for each experimental case. Additionally, the results are compare with the explicitly indicated benchmark, the best performing classifier is highlighted in bold.

		CCR at FPR				
Experiment	Conf	10^{-3}	10^{-2}	10^{-1}	0.5	1
Basic Adversarial						
Benchmark: S_2	0.3485	-	-	0.4477	0.6645	0.7458
$E_2, \epsilon = 0.1, ep = 0$	**0.3712**	0.1071	**0.2303**	0.4355	0.6142	0.6671
$E_2, \epsilon = 0.5, ep = 0$	0.3555	-	-	0.4903	0.6761	0.7206
$E_2, \epsilon = 0.5, ep = 33$	0.3560	-	-	0.4910	0.6716	0.7271
$O_2, \epsilon = 0.5, ep = 0$	0.3473	-	-	0.4716	0.6542	0.7019
$O_2, \epsilon = 0.5, ep = 33$	0.3482	-	0.1826	**0.5045**	**0.6923**	**0.7516**
Fine-tune Closed-Set						
Benchmark: S_2	0.3485	-	-	0.4477	0.6645	0.7458
$E_2, \epsilon = 0.5, FGSM_{full}$	0.3230	-	-	0.5471	0.7323	0.7832
$E_2, \epsilon = 0.5, \mu = 0.96, FGSM_{filter}$	**0.7411**	-	-	0.0026	0.0252	0.0329
$E_2, \epsilon = 0.5, RN_{filter}$	0.3368	-	0.2581	0.5116	0.7110	0.7665
$O_2, \epsilon = 0.5, FGSM_{full}$	0.3228	-	-	0.5445	0.7310	0.7794
$O_2, \epsilon = 0.5, \mu = 0.96, FGSM_{filter}$	**0.7395**	-	0.0013	0.0077	0.0245	0.0368
$O_2, \epsilon = 0.5, RN_{filter}$	0.3295	-	-	**0.5561**	**0.7439**	**0.7910**
Finte-tune Open-Set						
Benchmark: E_2	**0.6424**	-	0.2587	0.5187	0.6994	0.7368
$E_2, \epsilon = 0.2, FGSM_{filter}$	0.3784	-	**0.3523**	0.5587	0.7323	0.7755
$E_2, \epsilon = 0.5, FGSM_{filter}$	0.3818	-	-	**0.5639**	0.7323	0.7794
$E_2, \epsilon = 0.5, RN_{filter}$	0.3368	-	0.2581	0.5116	0.7110	0.7665
Benchmark: O_2	0.5690	0.1084	0.2684	0.5135	0.6994	0.7361
$O_2, \epsilon = 0.2, FGSM_{filter}$	0.3651	-	0.3174	0.5594	**0.7394**	**0.7852**
$O_2, \epsilon = 0.5, FGSM_{filter}$	0.3689	-	0.3026	0.5503	0.7303	0.7748
$O_2, \epsilon = 0.5, RN_{full}$	0.3792	-	-	0.4839	0.6723	0.7213

Table 4.5: P_2 RESULTS. Confidence and CCR for each experiment. Every set of experiments has a benchmark on top of the block. The highest value per experiment type is marked in bold.

The basic adversarial experiment in the first row of figure 4.4 shows that including the adversarial samples helps to improve the performance of the classifier. The improvement using entropic loss is clear when $FPR \leq 0.2$, for bigger FPRs the difference is smaller, even S_2 overperforms at $FPR \geq 0.6$. There is no significant difference between including the samples at epochs 0 and 33. However, the difference is notorious for the objectosphere since the classifier ($O_2, \epsilon = 0.5, ep = 33$) performs better than the softmax in all FPR values.

The second experiment shows that fine-tunning the softmax pre-trained classifier with adversarial samples increases the performance in the OSR scenario. Note that in both cases, entropic and objectosphere, the random noise has a similar effect as the FGSM perturbations. This similar behaviour can be due to the considerable epsilon value since the distortion at $\epsilon = 0.5$ hides the original image. Note that the classifiers with decaying epsilon reduce the CCR values to almost 0.

However, table 4.5 showed the confidence of the classifier is higher than the others in the experiment. This increased confidence is explained by small epsilon values, as training with $\epsilon <= 0.1$ reduces the network's classification power. The adversarial images are too similar to the original, and the model only learns to assign low scores to every sample, since the P_2 test dataset has almost three times more unknowns than knowns, the confidence is high.

Similarly, the third experiment shows a marginal increase in the performance of entropic and objectosphere classifiers with respect to the E_2 and O_2 references. Note that the benchmarks have been trained with known and unknown samples, and they can reject certain classes. In this experiment, the FGSM perturbation is superior to the random noise (RN_{filter} and RN_{full}).

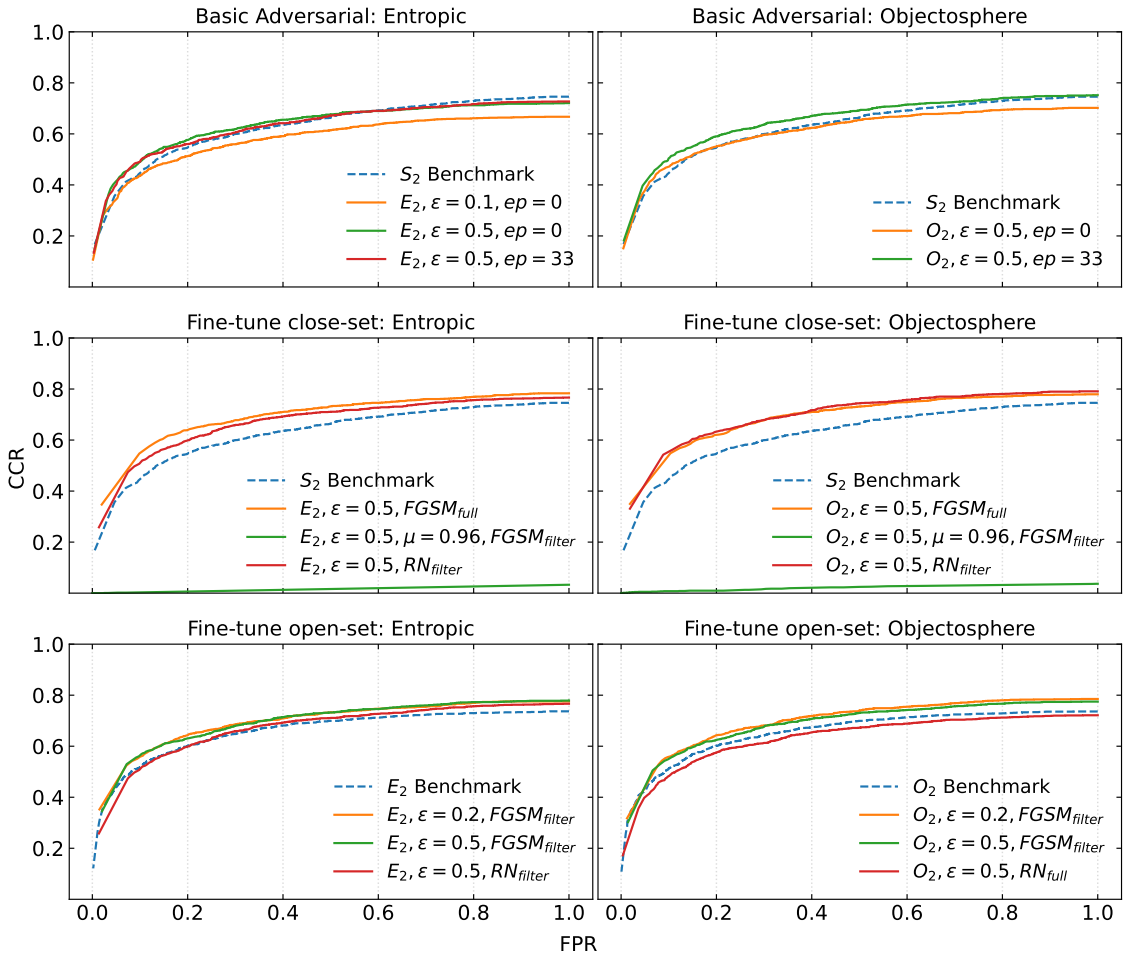


Figure 4.4: P_2 RESULTS. Results are depicted for entropic (left) and objectosphere (right) classifiers and for all three experimental cases (rows).

4.3.4 Protocol 1

The results of P_1 experiments are presented in table 4.6 and figure 4.5. In this protocol, entropic classifiers are mainly evaluated.

		CCR at FPR				
Experiment	Conf	10^{-3}	10^{-2}	10^{-1}	0.5	1
Basic Adversarial						
Benchmark S_1	0.5283	-	**0.4802**	0.6378	**0.7284**	**0.7329**
$E_1, \epsilon = 0.5, pe = 20$	**0.6950**	0.1278	0.4386	0.6740	0.6936	0.6936
$O_1, \epsilon = 0.5, pe = 20$	0.6914	**0.2057**	0.4714	**0.6803**	0.7000	0.7000
Fine-tune Closed-Set						
Benchmark S_1	**0.5283**	-	**0.4802**	0.6378	**0.7284**	**0.7329**
$E_1, \epsilon = 0.2, FGSM_{filter}$	0.5210	-	0.4597	0.6234	0.7095	0.7155
$E_1, \epsilon = 0.5, FGSM_{filter}$	0.5182	-	0.4724	**0.6434**	0.7224	0.7274
Finte-tune Open-Set						
Benchmark E_1	**0.7867**	-	0.1040	**0.6659**	0.7067	0.7086
$E_1, \epsilon = 0.5, FGSM_{full}$	0.6196	0.2055	**0.4916**	0.6500	**0.7255**	**0.7283**

Table 4.6: P_1 RESULTS. Results of selected experiments

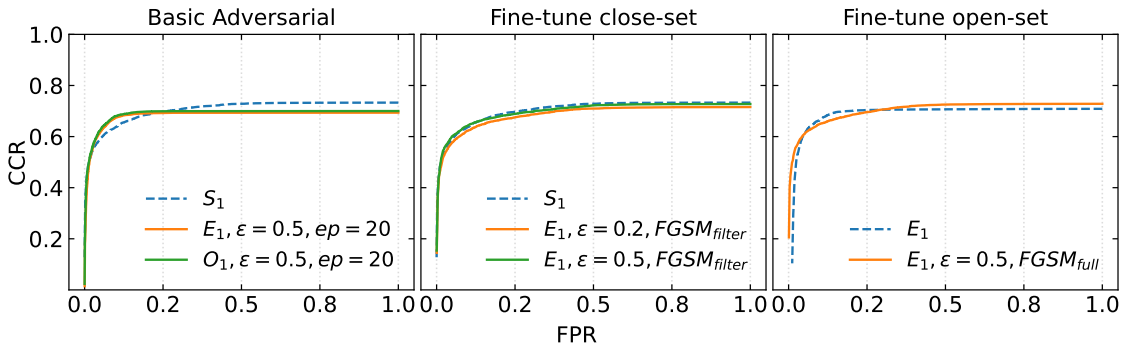


Figure 4.5: P_1 RESULTS. **Left**: Basic adversarial case, both classifiers E1 and O_1 included. **Middle**:Fine-tune close-set tested with entropic and large epsilons ($\epsilon = 0.2, 0.5$). **Right**: fine-tune open-set tested with $\epsilon = 0.5$.

4.3.5 Protocol 3

The results of the selected experiments are presented in table 4.7 and figure 4.6.

		CCR at FPR				
Experiment	Conf	10^{-3}	10^{-2}	10^{-1}	0.5	1
Basic Adversarial						
S_3	0.4151	-	-	**0.5517**	**0.7422**	**0.7989**
$O_3, \epsilon = 0.5, pe = 50$	**0.4280**	-	0.2216	0.5142	0.7093	0.7662
Fine-tune closed-set						
S_3	0.4151	-	-	0.5517	0.7422	**0.7989**
$E_3, \epsilon = 0.5, FGSM_{full}$	0.4405	-	-	0.5524	0.7349	0.7855
$E_3, \epsilon = 0.5, RN_{(full)}$	**0.5347**	-	-	**0.5820**	**0.7582**	0.7933
Fine-tune open-set						
E_3	**0.7063**	-	0.1729	**0.5925**	**0.7536**	**0.7884**
$O_3, \epsilon = 0.2, FGSM_{full}$	0.5679	-	-	0.5717	0.7428	0.7778
$E_3, \epsilon = 0.5, FGSM_{full}$	0.4405	-	-	0.5524	0.7349	0.7855

Table 4.7: P_3 RESULTS. Note that in fine-tune closed-set and fine-tune open-set, the tested classifiers do not reach FPR= 10^{-2}. It is challenging to discard unknowns in this protocol.

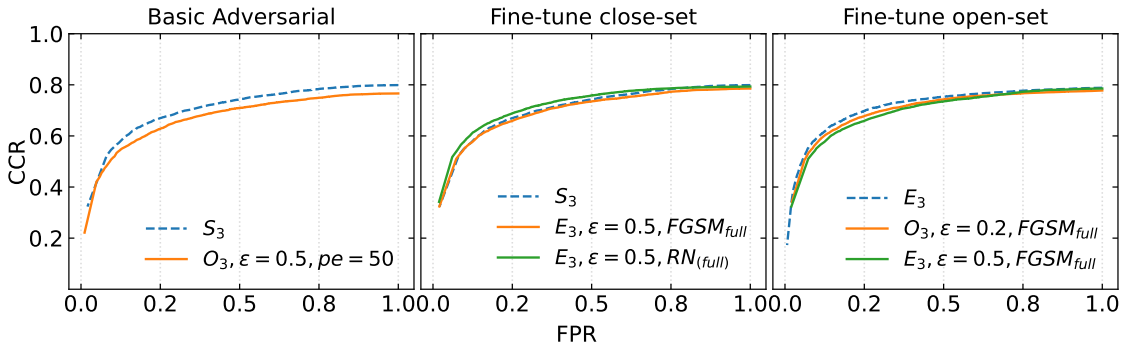


Figure 4.6: P_3 RESULTS. **Left:** Basic adversarial tested with objectosphere classifier. **Middle:** Fine-tune close-set tested with entropic classifier. There is no significant improvement to the benchmark. However, surprisingly, the random noise is making minor improvements. **Right:** Fine-tune open-set tested with objectosphere and entropic classifiers.

Chapter 5

Discussion

In this chapter, we discuss the results of our experiments.

In the first phase, we can check that it is not challenging for a classifier trained on entropic and objectosphere losses to reject images that are dissimilar to the known classes. For example, figure 4.1 shows the case where E_1 and O_1 quickly saturate the CCR value in the validation set. Similarly, in P_2 and P_3, the classifiers have similar behaviour, but the saturation of the CCR is not as fast as in P_1. That is expected, as the unknown values are more similar to the target images, these classifiers would need a lower threshold τ to operate at their top CCR. Note that in the test set, the entropic and objectosphere classifiers perform very similar to the softmax classifier in all protocols, indicating that the classifiers learned to reject $\mathcal{B}$ but do not generalise to $\mathcal{A}$.

Note that the CCRs of S_1 and E_1 are higher in the test set than in the validation set. Especially the performance of S_1 is improved in the test dataset. This behaviour can be due to a bias in the test dataset or weigth parameters specific to the S_1 classifier. Naturally, a way to corroborate the S_1 performance is to cross-validate. However, cross-validation is not recommended since it is computationally expensive.

The histograms in figure 4.2a suggest that in P_1 it is easy even for the softmax to separate the unknown samples. Note that the score and features magnitude histograms showed clear separation in S_1. This behaviour is exaggerated in E_1 and O_1. Additionally, in figure 4.2b, the O_1 feature's magnitudes are pushed close to 0 to improve separation. However, note that O1 shows a high number of unknown samples around magnitudes=10 (our chosen ξ) . This might explain why E_1 reaches higher CCR values than O_1.

One reason for this might be that the hyperparameters are not optimal. The hyperparameters were selected using grid search as depicted in figure A.1. I hypothesise that forcing the magnitudes restriction works very well in small datasets and a small number of classes, but it might not generalise well to natural images. Another reason for the objectosphere classifier's behaviour is that it does not limit the magnitude of the features of the known samples so that the features can be infinitely large. Nevertheless, the feature's norms do not improve the correct classification rate, it is meant to better separate knowns from unknowns.

In the objectosphere loss, J_E learns to correctly classify $\mathcal{C}$ and discard $\mathcal{B}$, while J_O enforces that behaviour by penalising the features magnitudes. This behaviour might occur when optimal hyperparameters are chosen. It is challenging to find optimal parameters in practice, as both α and ξ scale the loss. In the total loss, J_E and J_O should have comparable values. Yet, the two represent two different conceptual quantities as J_E accounts for the cross-entropy while J_O for vectors magnitudes. As Dhamija et al. (2018) suggested, the objectosphere is more challenging to optimise.

In experiments with epsilon decay, model training with $\epsilon \leq 0.12$ caused stability issues. Figure 5.1 shows an example of the training progress for 150 epochs. The validation AUC and confidence have high variability. In general, when $\epsilon \leq 0.12$, the classifier performance dramatically

reduces. This might be because the adversarial sample is still very similar to the clean image, as experiments with bigger epsilons are more stable. The figure also shows that the optimiser balances the entropic and adversarial losses.

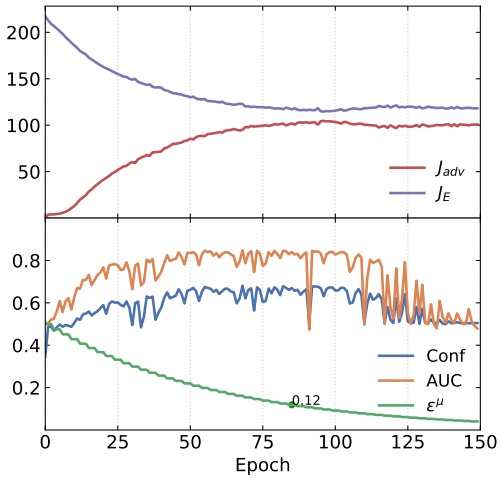


Figure 5.1: ADVERSARIAL TRAINING IN P_2. Adversarial training with decaying epsilon. In this example, the classifier was trained for 150 epochs with a decaying epsilon, adding adversarial samples from the start of the training. The training reaches a loss balance between J_E and J_{adv}. This might be because, after certain epochs, the model learned to classify specific classes and the number of correctly classified samples is the same for a small number of epochs. Furthermore, the validation metrics are, on average, increasing until reaching values equal or below 0.12

The experiments of phase 2 show that adversarial samples are helpful in OSR scenarios. For example, in P_2, the fine-tuning close-set with entropic loss improves up to 22% the CCR at FPR=0.1 and 10% at FPR=0.5. The remaining experiments reach similar improvements in P_2. Additionally, random noise perturbations also increase the performance for some cases in P_2. This is interesting since this perturbation does not require any gradient and can be efficiently calculated, it might be a cost-effective approach to improve a classifier.

The fine-tune experiments in P_2 show that it is possible to increase the CCR even when FPR=1. For example, in fine-tune open-set, the objectosphere classifier increases the CCR 6% from the benchmark. The reason can be that training to reject adversarial enforces better knowledge of the $\mathcal{C}$ classes, the classifier might learn better the differentiating features of each class.

Finally, the limitations of this work include:

- As seen in phase 2, it is difficult to replicate the results from P_2 to P_1 and P_3. We cannot guarantee improvements on other datasets. However, P_2 has very specialised and similar known and unknown classes, this might be a good starting point to design more scenarios to test this approach.

- During training, stability issues were generated. For some hyperparameters combinations, the loss diverges after a few epochs. The quick fix is to reduce the learning rate, which helps to mitigate the problem. However, using small learning rates have drawbacks, for example, staying in local minimum. One may explore simple strategies to reduce the loss when adversarial samples are present, like randomly sampling candidates to be perturbed on the batch.

- The batch normalisation layers are a problem for adversarial training. Diverging issues were generated when the model froze accumulating statistics. Moreover, freezing the statistics collection depending on the presence of adversarial samples can raise reproducibility problems.
- Training first a model in P2 and then trying to replicate it in P1 and P3 helped in the development of the pipeline and early tests. Nevertheless, this method can bias the approach to P1 and P3 by assuming similar parameters. Given computational restrictions, we decided to work on this way, but the classifiers for P_1 and P_3 might be suboptimal.

Chapter 6

Conclusions

In this work, we evaluated the performance of entropic open-set and objectosphere losses in complexity varying open set scenarios. Using the ImageNet protocols designed by Bhoumik (2021), we showed that the approach of Dhamija et al. (2018) outperforms the typical softmax cross-entropy loss by large margins in the validation set. However, our analysis also indicates the margins are reduced when the metric is obtained for only unknown unknown classes. Our experiments confirm that it is easy for the classifiers to reject very different samples from the known examples; even the softmax loss has acceptable behaviour. However generalising the models and making them robust to unknown distributions is challenging, mostly because the set of unknowns is infinite.

We attempted to improve the classifier's performance using adversarial samples as known unknowns, motivated by the work of Schnyder (2021). We defined a training method and designed three types of experiments. The results suggest that FGSM adversarial samples help to increase the robustness of the classifiers, as well as modified samples with random noise. The results indicate that the magnitude of the perturbation should not be small. In our case, a value of 0.1 started degrading the performance in the classifiers. Moreover, we improved the classifier's performance in protocol 2, where the known and unknown samples are very similar, showing our approach's potential.

It would be interesting to continue this work with the other FGSM attacks since they are fast to calculate. Moreover, other attacks like PGD should be practical to evaluate, but the computation time should be assessed.

Random noise perturbations improved the results in P2, suggesting that more attacks can be implemented. In particular, one can use data augmentations techniques to perturb the images and train the models with this augmented data as unknowns. On the other hand, to reduce the computation time, one could try approaches like re-utilizing the adversarial samples from epoch to epoch. We use the adversarial samples for the current parameters of the network. Nevertheless, reusing a batch of precalculated samples might save training time.

Including weights to the adversarial loss can be a tool to regularize the adversarial loss. Finally, the creation of more protocols can be helpful to understand the response of the losses to different data similarities.

Appendix A

Attachments

Hyperparameter Tunning

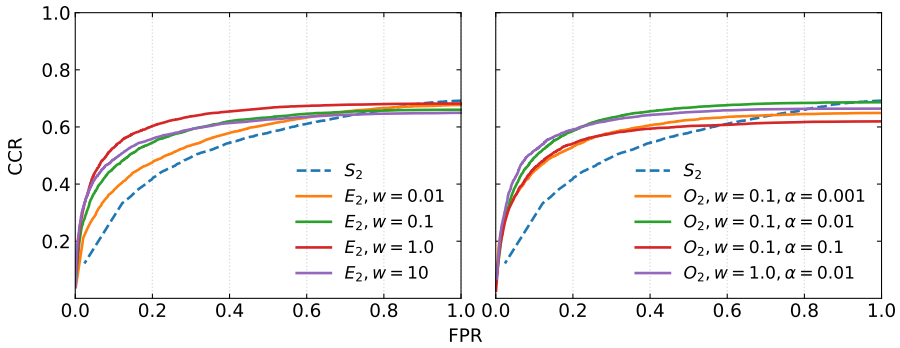


Figure A.1: P_2 HYPERPARAMETERS TUNNING. The effect of modifying the hyperparameters in validation set for entropic open-set loss (left) and objectosphere (right).

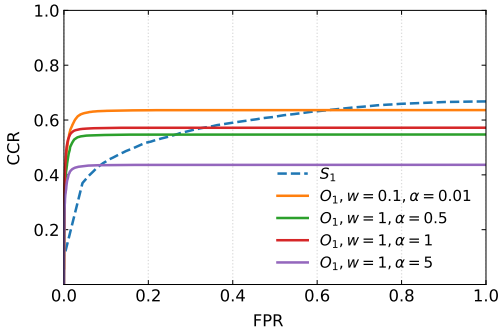


Figure A.2: P_1 HYPERPARAMETERS TUNNING. Effect of w and α in the objectosphere loss.The CCR is saturated at low FPRs, meaning it is easy for the classifier to discard unknowns. However the CCR values are reduced by sub-optimal parameters

Batch Normalisation Effect

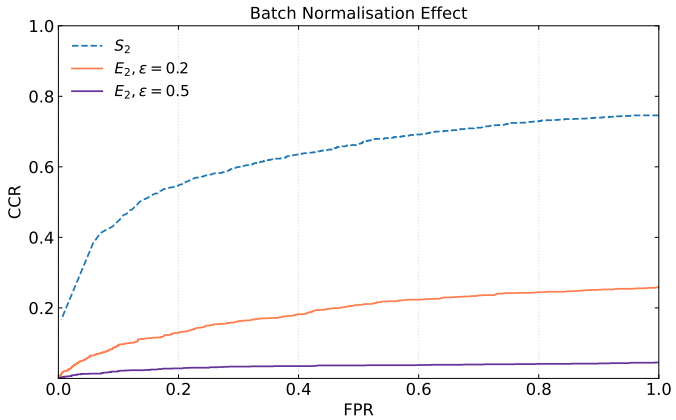


Figure A.3: BATCH NORMALISATION EFFECT. The undesired effect of the BN layer shifting statistics when training with adversarial samples.

Additional Basic Adversarial Training

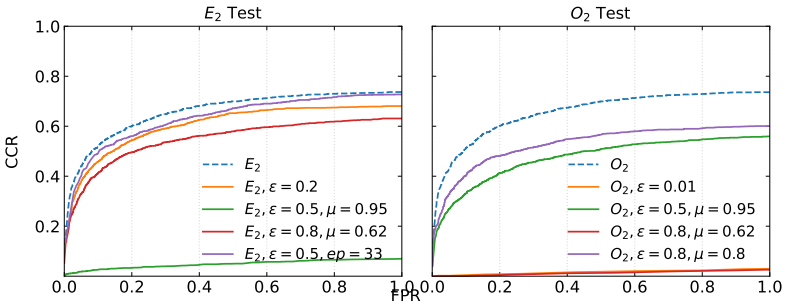


Figure A.4: ADDITIONAL CONFIGURATIONS IN P_2. Different configurations for adversarial generation and decay.

List of Figures

2.1 Open Space . . . 4
2.2 OSCR Approaches Classification . . . 5
2.3 OSCR Approaches Classification . . . 6
2.4 OpenMax and G-OpenMax Training . . . 7
2.5 Feature space visualization . . . 9
2.6 Deep Features Histograms . . . 10
2.7 Imagenet snapshot . . . 11
2.8 Imagenet open-set protocols snapshot . . . 14
2.9 Adversarial samples for DNN models . . . 15
2.10 FGSM Example . . . 16
2.11 LeNet++ Feature Space . . . 18

3.1 Residual Block . . . 20
3.2 ResNet architectures . . . 21

4.1 OSCR Benchmark . . . 29
4.2 Normalised Histograms . . . 30
4.3 Image Perturbations . . . 32
4.4 P_2 Results . . . 34
4.5 P_1 Results . . . 35
4.6 P_3 Results . . . 36

5.1 Adversarial Training in P_2 . . . 38

A.1 P_2 Hyperparameters tunning . . . 43
A.2 P_1 Hyperparameters tunning . . . 43
A.3 Batch normalisation effect . . . 44
A.4 Additional Configurations in P_2 . . . 44

List of Tables

2.1 Imagenet Protocols classes Overview 12

4.1 Imagenet Protocols Size 26
4.2 Known Classes distribution 26
4.3 Loss Hyperparameters 28
4.4 Results of phase 1 in test set 28
4.5 P_2 Results 33
4.6 P_1 Results 35
4.7 P_3 Results 36

List of Algorithms

1 Adversarial Training for OSR using FGSM attacks 23

Bibliography

Acharya, S., Pant, A. K., and Gyawali, P. K. (2015). Deep learning based large scale handwritten Devanagari character recognition. In *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pages 1–6.

Akhtar, N. and Mian, A. (2018). Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access*, 6:14410–14430.

Ben-Tal, A., Ghaoui, L. E., and Nemirovski, A. (2009). *Robust Optimization*. Princeton University Press.

Bendale, A. and Boult, T. E. (2016). Towards Open Set Deep Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, Las Vegas, NV, USA. IEEE.

Bhoumik, A. (2021). Open-set Classification on ImageNet. Master's thesis, University of Zurich.

Bodesheim, P., Freytag, A., Rodner, E., and Denzler, J. (2015). Local Novelty Detection in Multi-class Recognition Problems. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 813–820.

Boult, T. E., Cruz, S., Dhamija, A., Gunther, M., Henrydoss, J., and Scheirer, W. (2019). Learning and the Unknown: Surveying Steps toward Open World Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9801–9807.

Bridle, J. S. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In Soulié, F. F. and Hérault, J., editors, *Neurocomputing*, pages 227–236, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Miami, FL. IEEE.

Dhamija, A. R., Gunther, M., Ventura, J., and Boult, T. E. (2020). The Overlooked Elephant of Object Detection: Open Set. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1010–1019, Snowmass Village, CO, USA. IEEE.

Dhamija, A. R., Günther, M., and Boult, T. E. (2018). Reducing Network Agnostophobia. *arXiv:1811.04110 [cs]*. arXiv: 1811.04110.

Ge, Z., Demyanov, S., Chen, Z., and Garnavi, R. (2017). Generative OpenMax for Multi-Class Open Set Classification. *arXiv:1707.07418 [cs]*. arXiv: 1707.07418.

Geng, C. and Chen, S. (2022). Collective decision for open set recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):192–204. arXiv: 1806.11258.

Geng, C., Huang, S.-J., and Chen, S. (2021). Recent Advances in Open Set Recognition: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3614–3631.

Golan, I. and El-Yaniv, R. (2018). Deep anomaly detection using geometric transformations. *Advances in neural information processing systems*, 31.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*. arXiv: 1412.6572.

Guo, C., Rana, M., Cisse, M., and Van Der Maaten, L. (2017). Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*. arXiv: 1502.01852.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA. IEEE.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-Excitation Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial Examples Are Not Bugs, They Are Features. *arXiv:1905.02175 [cs, stat]*. arXiv: 1905.02175.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*. arXiv: 1502.03167.

Jiang, H., Kim, B., Guan, M., and Gupta, M. (2018). To trust or not to trust a classifier. *Advances in neural information processing systems*, 31.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*. Publisher: Citeseer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*. arXiv: 1607.02533.

LeCun, Y., Cortes, C., and Burges, C. (2010). MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lenz, I., Lee, H., and Saxena, A. (2015). Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724. Publisher: SAGE Publications Sage UK: London, England.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *arXiv:1512.02325 [cs]*, 9905:21–37. arXiv: 1512.02325.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2019). Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083 [cs, stat]*. arXiv: 1706.06083.

Mariolis, I., Peleka, G., Kargakos, A., and Malassiotis, S. (2015). Pose and category recognition of highly deformable objects using deep learning. In *2015 International conference on advanced robotics (ICAR)*, pages 655–662. IEEE.

Matan, O., Kiang, R. K., Stenard, C. E., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., and Lecun, Y. (1990). Handwritten character recognition using neural network architectures. In *Proceedings of the 4th US Postal Service Advanced Technology Conference, Washington D.C., November 1990*.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017). On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*.

Miller, G. A. (1998). *WordNet: An electronic lexical database*. MIT press.

Miotto, R., Wang, F., Wang, S., Jiang, X., and Dudley, J. T. (2018). Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246. Publisher: Oxford University Press.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *arXiv:1704.03976 [cs, stat]*. arXiv: 1704.03976.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582.

Neal, L., Olson, M., Fern, X., Wong, W.-K., and Li, F. (2018). Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading Digits in Natural Images with Unsupervised Feature Learning.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE.

Perera, P., Morariu, V. I., Jain, R., Manjunatha, V., Wigington, C., Ordonez, V., and Patel, V. M. (2020). Generative-discriminative feature representations for open-set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11814–11823.

Punjani, A. and Abbeel, P. (2015). Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. IEEE.

Qayyum, A., Qadir, J., Bilal, M., and Al-Fuqaha, A. (2020). Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180. Publisher: IEEE.

Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., and Lakshminarayanan, B. (2019). Likelihood ratios for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 32.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*. arXiv: 1506.01497.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. Publisher: American Psychological Association.

Rozsa, A., Rudd, E. M., and Boult, T. E. (2016). Adversarial Diversity and Hard Positive Generation. *arXiv:1605.01775 [cs]*. arXiv: 1605.01775.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Scheirer, W. J., Rocha, A., Sapkota, A., and Boult, T. E. (2012). Towards Open Set Recognition. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, page 14.

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. *arXiv:2008.04094 [cs]*. arXiv: 2008.04094.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. _eprint: 1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going Deeper With Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*. arXiv: 1312.6199.

Thoma, M. (2017). The hasyv2 dataset. *arXiv preprint arXiv:1701.08380*.

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

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018. Publisher: Hindawi.

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. (2021). On the convergence and robustness of adversarial training. *arXiv preprint arXiv:2112.08304*.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A Discriminative Feature Learning Approach for Deep Face Recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 499–515, Cham. Springer International Publishing.

Yoshihashi, R., Shao, W., Kawakami, R., You, S., Iida, M., and Naemura, T. (2019). Classification-Reconstruction Learning for Open-Set Recognition. *arXiv:1812.04246 [cs]*. arXiv: 1812.04246.

Yu, Y., Qu, W.-Y., Li, N., and Guo, Z. (2017). Open-Category Classification by Adversarial Sample Generation. *arXiv:1705.08722 [cs]*. arXiv: 1705.08722.