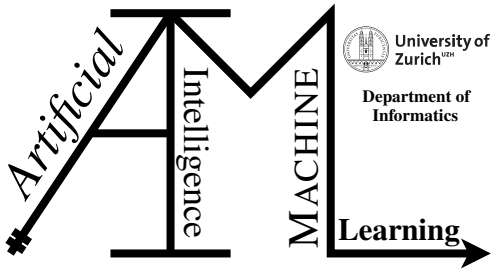
Open-set Classification on ImageNet

Master Thesis

Annesha Bhoumik
18-744-953

Submitted on
September 14 2021

Thesis Supervisor
Prof. Dr. Manuel Günther



Master Thesis

Author: Annesha Bhoumik, annesha.bhoumik@uzh.ch

Project period: March 15 2021 - September 14 2021

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

Acknowledgements

I completed this thesis under the supervision of Prof. Dr. Manuel Günther, Assistant Professor for Artificial Intelligence and Machine Learning (AIML) at the Department of Informatics of the University of Zurich. So, firstly, I would like to extend my immense gratitude to him for giving me the opportunity to do my thesis under him and for continuously guiding and supporting me at every step of the thesis. Despite the challenges of COVID-19, he ensured that the thesis took place smoothly by making use of online communication channels. Further, the "Deep Learning" lecture taught by him at the University, made my understanding of underlying concepts stronger than before. Lecture slides and exercises from this lecture served as a quick reference point for me throughout this thesis.

I would also like to thank Ms. Isabel Margolis and Mr. Vincent Rüegge, who recently completed their thesis under Prof. Günther, for guiding me whenever reached out for advice. Additionally, I would like to thank them for helping me with the German translation of the English abstract of the thesis. My family and friends have also played a major role when it came to reaching out for quick doses of advice and motivation, not only during the course of this thesis, but also throughout the entire length of my Master's studies. Lastly, I would like to thank all my fellow batchmates, Professors and members of the administration at the University of Zurich for their kind support.

Abstract

A real-world test situation not only comprises of classes that a supervised model is trained with, but also consists of unknown classes that the model is unaware of. Such a test situation forms part of an open space, and an open-set classifier, trained using open-set algorithms is used to handle it. These algorithms are developed using artificially generated open space, created using available datasets and their classes. Open-set algorithms that are currently in place are either developed using small datasets or, address only those unknown samples that are distinctly different from known classes. In this thesis, I propose three ImageNet based open-set protocols that closely align with the real-world open space and also, address unknown samples that are similar to known classes. I use these protocols to compare the performance of two easy-to-implement open-set algorithms, SoftMax with Garbage Class and Entropic Open-set Loss, that were developed using small datasets, and compare their performance to that of a baseline Traditional SoftMax.

Zusammenfassung

In realen Testsituationen bestehen die Klassen nicht nur aus den bekannten Klassen, die zum Trainieren des Modells benutzt werden, sondern auch aus unbekannten Klassen, die das Modell nicht kennt. Solche Testsituationen sind Teil eines Open Spaces und ein mit open-set Algorithmen trainiertes Open-Set-Bildklassifikationsmodell wird eingesetzt, um es zu behandeln. Anhand der verfügbaren Datensätze und Klassen werden künstliche Open Sets generiert. Diese Open Sets werden benutzt, um die Open-Set Algorithmen zu entwickeln. Open-Set-Algorithmen, die heutzutage benutzt werden, wurden entweder nur mit kleinen Datensätzen entwickelt, oder befassen sich nur mit Datensätzen, die sich deutlich von den bekannten Klassen unterscheiden. In dieser Thesis, schlage ich drei ImageNet-basierte Open-Set-Protokolle vor, die dem realen Open Set sehr ähnlich sind und Datensätze berücksichtigen, deren Klassen unbekannt, jedoch mit den bekannten Klassen verwandt sind. Mit Hilfe dieser Protokolle, vergleiche ich zwei Open-Set-Algorithmen miteinander. Beide Open-Set-Algorithmen, «SoftMax with Garbage Class» und «Entropic Open-Set Loss», sind einfach zu implementieren und wurden mit kleinen Datensätzen entwickelt. Zudem vergleiche ich die Performance beider Algorithmen mit einer Baseline, dem traditionellen SoftMax.

Contents

Chapter 1

Introduction

On February 12, 2002, the United States Secretary of Defense, Donald Rumsfeld, answered a question posed to him during a news briefing regarding the lack of evidence that proves that the Government of Iraq is responsible for supporting terrorist groups with weapons of mass destruction, and the exact words of his answer were as follows:

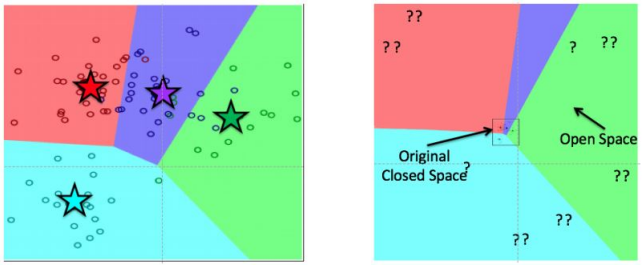
"Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns - the ones we don't know we don't know. And if one looks throughout the history of our country and other free countries, it is the latter category that tends to be the difficult ones."[1]

This response is relevant not only in case of humans but also in case of machines. Supervised image classification algorithms have achieved tremendous success when it comes to detecting classes from a finite number of *known known* classes, what is commonly known as evaluation under the closed-set assumption (Boult et al., 2019). However, gauging their performance is tricky when they are fed with *unknown unknown* samples, existence of which is an absolute mystery to the model. In fact, current classification models sometimes confuse unknown unknown samples as known classes with high confidence, when ideally, they are expected to detect them as unknown and reject them (Sünderhauf et al., 2018). Figure 1.1(a) shows how under closed-set classification, samples are correctly classified as one of the finite number of known classes. The model develops decision regions for each of the known classes represented by the different colors. But, it is upon zooming out of this closed space that one can visualize the real-world test space, termed as open space by Boult et al. (2019), as seen in Figure 1.1(b), samples from which are likely to be encountered during testing. When decision regions learnt using closed-set classification are extended into the open space, unknown samples, represented by "?" get comfortably misclassified as one of the known classes depending on the extended decision region they fall in. However, models are expected to identify them as unknown (that is none of the known classes), and reject them, as seen in Figure 1.2. To achieve this objective, researchers must shift from designing perfect closed-set classifiers to developing near-perfect open-set classifiers. For this, it is important to understand the composition of open space. Dhamija et al. (2018) divide the open space into three parts and this division also aligns with Mr. Donald Rumsfeld's answer.

1. **Known Classes**: A finite known space occupied by known classes, that also form the closed-space in closed-set classification. The model is trained using these classes and has complete information about them. Closed-set as well as open-set classifiers are expected to correctly classify all these classes.

2. **Known Unknown Classes**: A finite known unknown space occupied by known unknown classes. The model only has partial information about these classes, and sometimes, they

[1] https://en.wikipedia.org/wiki/There_are_known_knowns



(a) Classification under closed-set assumption

(b) Extension into Open-space

Figure 1.1: CLOSED SPACE VERSUS OPEN SPACE. This figure shows how under closed-set classification, all samples are correctly classified as seen in (a), until the closed-space is extended to open-space as seen in (b). *Source: (Boult et al., 2019)*

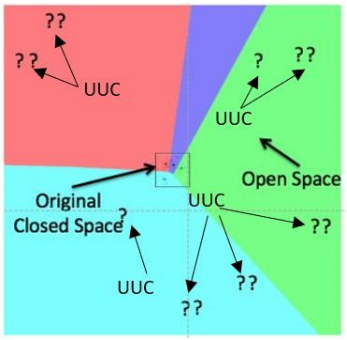


Figure 1.2: OPEN-SET CLASSIFICATION. Under open-set classification, a classifier is expected to correctly classify all known classes and correctly detect all unknown samples in open space. *Source: (Geng et al., 2020)*

are used for training to provide the model with a hint of possible unknown unknown samples likely to be encountered during testing, assuming the unknown unknown samples will behave similarly as samples from these classes.

3. **Unknown Unknown Classes**: An infinite unknown unknown space occupied by samples that the model does not see at all during training and hence has no knowledge about. An ideal open-set algorithm is expected to detect all samples from these classes.

I call these classes as open-set partitions of the open space and sometimes, jointly refer to known unknown and unknown unknown samples as just unknown samples. While it is impossible to train a classification model with all possible unknown examples, it is possible to distinguish known samples from unknown samples by using a known decision boundary threshold beyond which an input becomes unknown or/and by manipulating their feature representations (Roady et al., 2020). It is using these concepts that models can detect unknown samples and reject them, while maintaining their performance in correctly classifying known samples.

Till now, several approaches have been taken by researchers to satisfy the open-set goal, and each approach addresses the open-set problem in a different way and has been developed using different datasets and different theories (Dhamija et al., 2018; Bendale and Boult, 2016; Rudd et al., 2018; Zhou et al., 2021; Matan et al., 2001).[2] In this thesis, I use two of these approaches,

[2] https://github.com/iCGY96/awesome_OpenSetRecognition_list

namely SoftMax with Garbage Class and Entropic Open-set Loss, which attempt to reduce overlap between known and unknown classes in open space (Dhamija et al., 2018), and compare the performance of these algorithms on a large-scale hierarchical dataset called ImageNet. SoftMax with Garbage Class uses a garbage class created using *known unknown* classes, to train classification models such that they separate samples from garbage class and unknown unknown classes in a space different from that of all the known classes. Entropic Open-Set Loss uses a novel loss function to shorten the Euclidean length of the feature vector of known unknown and unknown unknown test samples in comparison to that of known samples. Both the approaches prove to be effective on open-set partitions created using small datasets, where known and unknown classes are also distinctly different from each other in appearance.

To test their performance on a large hierarchical dataset, I design three ImageNet protocols, with different open-set partitions, for open-set evaluation. This is done since ImageNet is a closed-set classification dataset and never before have there been open-set partitions created using the same ImageNet dataset, that address varying levels of difficulty faced in the real-world open space test situation. I explore different levels of hierarchy and similarity between ImageNet classes to design these protocols to provide a stronger sense of open-set evaluation for the two approaches. It is also a first time that such an intensive evaluation of the two open-set algorithms is carried out using open-set ImageNet protocols.

The thesis is outlined as follows:

- **Dataset** - In this chapter, I will discuss about ImageNet and the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) dataset that I use to perform my experiments.
- **Related Work** - In this chapter, I will first introduce traditional SoftMax and then dive into different open-set algorithms, two of which are experimented with in this thesis. I will also provide a comparison between open-set classification and another application that is closely related to it called Out-of-Distribution Detection.
- **Open-set ImageNet Protocols** - Here, I will highlight how I design the three open-set ImageNet protocols to address varying levels of difficulty in terms of correctly classifying known classes and correctly detecting unknown unknown classes that are similar to known classes.
- **Experimental Set-up** - Here, I will explain how I set-up my experiments that is how ImageNet is accessed at different hierarchical levels to create the open-set protocols, how custom datasets and dataloaders are created, the network selected for the experiments and the hyperparameters used.
- **Experiments** - In this chapter, I will first introduce two novel evaluation metrics used for open-set evaluation. I will then detail how I execute SoftMax with Garbage Class, Entropic Open-Set Loss and the baseline traditional SoftMax and train the network using these algorithms. I will also share the results of my experiments and draw observations from them in this chapter.
- **Discussion** - In this chapter, I will derive insights using my observations from the results and discuss potential limitations.
- **Conclusion and Future Work** - In this chapter, I will draw conclusions from my thesis by summarizing the results and insights from my open-set experiments. I will also discuss gaps that can be filled through future work.

Chapter 2

Dataset

While small datasets provide a quick way to experiment and develop advanced computer vision algorithms, it is only by developing an algorithm using a rich, hierarchical dataset that a high quality benchmark can be achieved to compare against. ImageNet is one such large-scale hierarchical dataset, that serves both as a benchmark dataset in developing advanced computer vision algorithms and a benchmark to compare performance of state-of-the-art algorithms. It is due to this reason that I select ImageNet to perform my experiments. Specifically, I use a subset of ImageNet that was created for ImageNet based 2012 challenge, details of which I discuss in section 2.2.

2.1 ImageNet - An Image Database

ImageNet is a huge collection of human annotated and quality controlled images that are structured as per the WordNet hierarchy.[1] WordNet is a large lexical database of English words, where adjectives, verbs, nouns and adverbs, that are linked together by a semantic relationship to represent the same distinct concept, are clubbed together to form "synsets" or synonym sets. This semantic hierarchy can be visualized as a tree and the complete hierarchy of a word can be easily explored using the WordNet online tool.[2] Two hierarchical examples are shown in Figure 2.1. Here, *husky* and *trimaran* represent the most distinct noun, which get grouped under semantically similar nouns, eventually leading to the root nodes of *mammal* and *vehicle* respectively. To address these nodes more clearly, I use the following terms:

- **Superclass** - A class that has one or more classes under it is called a Superclass. For example, in Figure 2.1, all classes preceding *husky* are superclasses.

- **Subclass** - A class under a superclass is called a subclass and classes with with no further subclasses are called leaves. The terms subclass and descendant are used interchangeably. In Figure 2.1, *dog* as well as *husky* are subclasses of *mammal*, where *husky* is also a leaf.

Since the publication of (Deng et al., 2009), ImageNet has grown from 5,247 synsets with 3.2 million images to 21,481 synsets with $\approx$ 14.2 million images as per its latest update.[3] It is an ongoing research effort which aims to address around 80,000 noun synsets of WordNet, with an average of at least 1000 clean and full resolution images per synset.

[1] https://wordnet.princeton.edu/
[2] http://wordnetweb.princeton.edu/perl/webwn
[3] http://www.image-net.org/

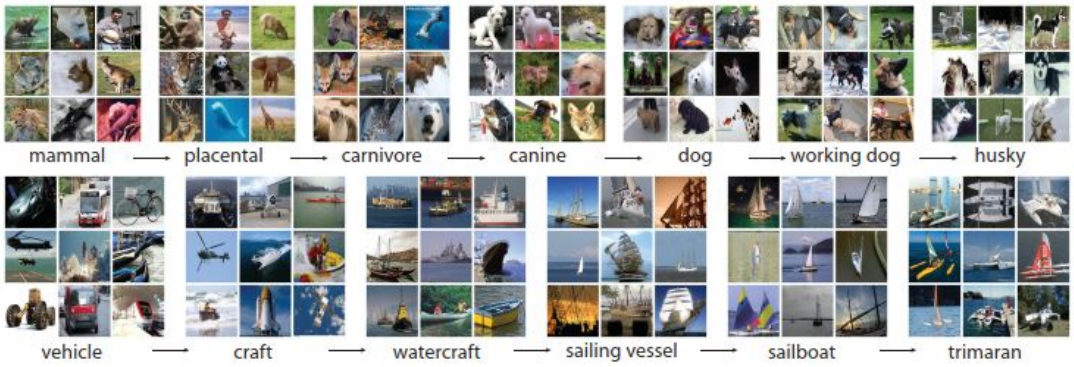


Figure 2.1: SNAPSHOT OF IMAGENET. This figure shows an example of two root-to-leaf branches of ImageNet. The class on the right side of an arrow is a subclass of the preceding superclass and the class on the extreme right end is a leaf. *Source: (Deng et al., 2009)*

Ever since its creation, ImageNet has been used rigorously by the computer vision community for several applications like image classification, tree based image classification, object localization, and object detection. Image classification is fairly simple, where given an image, an algorithm needs to identify the object class present in the image. Tree based image classification takes advantage of the semantic hierarchy within ImageNet to not only classify an image by its superclass, but to also provide information about its subclass. Object localization identifies the location of objects within an image and places a bounding box around it, and object detection uses these bounding boxes to identify the class of the bounded object. These applications of ImageNet were further explored through a competitive challenge, details of which are described in the following section.

2.2 ImageNet Large Scale Visual Recognition Challenge

ImageNet Large Scale Visual Recognition Challenge or ILSVRC, was an annual competition held between 2010 and 2017. Subsets of ImageNet were created for these challenges, with the aim of comparing advances in image classification, object detection and localization algorithms, as well as to track the progress of large-scale computer vision algorithms (Russakovsky et al., 2014). Each subset consisted of a training, validation and test set. The training set was created using human annotated ImageNet images, while the test set was created using images from outside ImageNet, whose annotations were held back from the public.

Performance on ILSVRC classification task, that is the task of producing a list of five most probable object classes to classify an image, was evaluated under the closed-set assumption.[4] Improvement in closed-set classification performance by winning teams between 2010 and 2014, in terms of top-5 classification error, that is the fraction of test samples for which the correct object class is not present in the five most probable classes predicted by the model, can be seen in Figure 2.2. This performance improved further between 2015 - 2017 with the top-5 classification error being as low as 2.251% for the winning entry in ILSVRC2017.[5] With such tremendous improvement in classification of ImageNet images over the years, the question that lingers is how does the

[4] https://image-net.org/challenges/LSVRC/
[5] https://www.kaggle.com/getting-started/149448

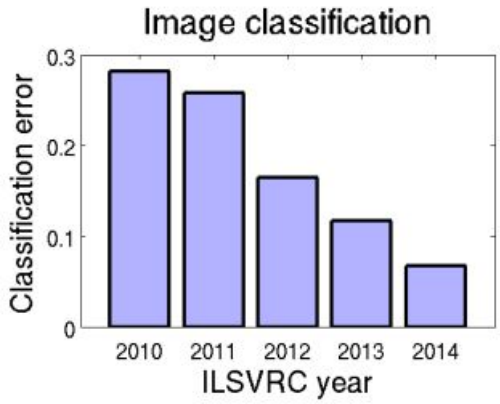


Figure 2.2: CLOSED-SET CLASSIFICATION PERFORMANCE ON IMAGENET. This figure shows the improvement in top-5 classification error of winning entries in closed-set classification tasks from ILSVRC 2010 - 2014. *Source: (Russakovsky et al., 2014)*

performance look like when the classification task is changed from being closed-set to open-set.

ILSVRC2012: Other than the task of producing a list of five object classes in descending order of confidence and specifying their location in the image using bounding boxes, the 2012 challenge included a fine-grained image classification task for 120 dog classes, with one prediction per image. The training set of ILSVRC2012 consists of 1000 classes with $\approx$ 1.2 million human annotated ImageNet images spread almost equally across all the classes or synsets (732-1300 per synset). The validation and test set consist of 150,000 images, out of which only 50,000 annotated images (50 images per synset) were available as validation set to the public, while the remaining 100,000 images (100 images per synset) were made available without annotations only during testing. The 1000 classes are available as 1000 synset IDs.

Chapter 3

Related Work

While many researches have been published in the last decade in the field of open-set image classification, I believe this thesis draws inspiration from five of them and I focus on discussing those in this chapter. But before discussing the related literature and diving into open-set algorithms, it is essential to understand how closed-set classification with deep networks work using a traditional SoftMax and why it is not effective in open-set classification. I also provide a comparison of open-set image classification with another application called Out-of-Distribution Detection, which is closely related to it.

3.1 Traditional SoftMax

A SoftMax function is used to represent the probability distribution in a multi-class classification problem, where the predicted class can be one among O different classes. In other words, when the output of a network is not binary, but has O possible options, a SoftMax function is used. A SoftMax converts real-valued logit values, calculated using network weights and activation functions, to probabilities (Goodfellow et al., 2016). Under closed set classification, the probabilities thus generated sum to 1. The class with the largest probability is then predicted as the class of the input sample by the network. The SoftMax function can be calculated as follows:

$$y_o = \frac{e^{z_o}}{\sum_{o'=1}^{O} e^{z_{o'}}} \tag{3.1}$$

Here, y_o is the predicted probability for class o, calculated by dividing the exponential of logit values for class o (z_o) by the sum of the exponential of logit values for all O classes. The O output classes are sometimes called SoftMax targets and the predicted probabilities are called confidences. For the task of image classification, a traditional network looks like that shown in Figure 3.1, where convolutional layers are used for effective and efficient classification.[1] As can be seen, SoftMax function is used in the final fully-connected layer, as the output of the network.

3.2 Open-set Algorithms

In open-set classification, a classifier is expected to correctly classify known test samples and correctly detect unknown test samples and reject them. SoftMax may be effective in predicting

[1] https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

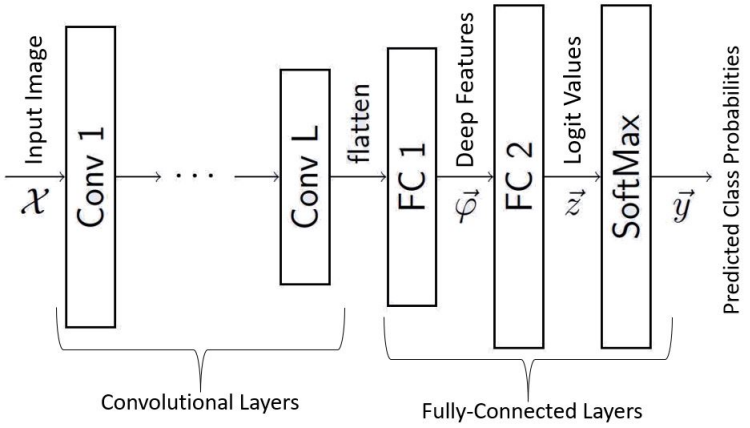


Figure 3.1: TRADITIONAL IMAGE CLASSIFICATION NETWORK. This figure shows a traditional convolutional classification network, with L convolutional layers (Conv) and three fully-connected layers - FC1, FC2 and SoftMax.

classes that the network has seen before (known classes), but what happens when the input to the network is an unknown class? An easy approach would be to threshold on the maximum class probability, using what is known as the confidence threshold, assuming that for an unknown input, the probability would be distributed across all the classes and hence, would be low (Dhamija et al., 2018; Matan et al., 2001). But such an approach is not effective when unknown inputs overlap significantly with known decision regions and tend to get misclassified as a known class with high confidence. This can be represented visually using 2D bottleneck plots or flower plots, created using features extracted from fully connected layer FC1 in Figure 3.1, that is selected to have two neurons to produce 2D plots (Dhamija et al., 2018). Training on MNIST[2] using a traditional SoftMax would produce a bottleneck plot as shown in Figure 3.2, where colored dots represent test samples from 10 known MNIST classes and black dots represent test samples from unknown unknown Devanagari classes (Pant et al., 2012). Gray lines denote class boundaries and if a sample falls on any class boundary, it has equal chances of being detected as belonging to either of the two neighbouring known classes. It can be seen from Figure 3.2 that there is a large amount of overlap between known and unknown unknown test samples, which leads to the misclassification of many unknown unknown samples as known classes with high confidence.

It is therefore essential to devise techniques that are more effective than simply thresholding SoftMax probabilities in detecting unknown inputs. Some initial approaches include extension of 1-class and binary Support Vector Machine (SVM) as implemented by Scheirer et al. (2013) and devising recognition systems to continuously learn new classes as implemented by Bendale and Boult (2015) and Rudd et al. (2018). In this chapter, I will discuss the work of Rudd et al. (2018) in detail and then discuss other recent approaches that attempt to solve the open-set problem.

3.2.1 Extreme Value Machine

Extreme Value Machine or EVM incorporates a compact online incremental update mechanism which incrementally updates known classes by making use of their decision boundaries (Rudd et al., 2018). Under EVM, known classes are characterized by a set of specific data points and distributions or extreme vectors (EV) that best summarize the class. These EVs are then used to

[2] http://yann.lecun.com/exdb/mnist/

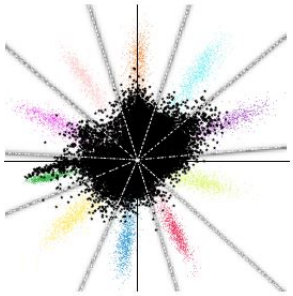


Figure 3.2: BOTTLENECK PLOT USING TRADITIONAL SOFTMAX. This figure shows bottleneck or flower plot for 10 known MNIST classes (colored dots) and unknown unknown Devanagari classes (black dots), when Traditional SoftMax is used. Overlap between these classes leads to misclassification of unknown samples as known classes. *Source: (Dhamija et al., 2018)*

generate radial boundaries for each known class that determine the probability of an input belonging to that class or the probability of sample inclusion, as termed in the paper. This avoids considering redundant data points, thereby reducing computing cost and storage requirements, without compromising on classification accuracy. For their experiments, Rudd et al. (2018) considered a data point of a class as redundant if it had greater than 50% probability of getting covered by other data points of that class.

During the initial training phase, first set of EVs are obtained, post which the model is retrained with a new batch of data to update its EVs using old EVs and new training data points. If EVM spots an unknown data point among new training data points, it saves its distribution to produce EVs for new classes. This leads to producing updated EVs for each known class and identification of EVs for new classes. This process is repeated for multiple new training batches to update the model's EVs using previous EVs, EVs of new classes and new training points.

Dataset of ILSVRC2014 was used for training and testing EVM. The initial training phase consists of samples from 50 classes of ILSVRC2014 training set followed by classification of 0, 50, 100, 150 and 200 unknown classes from the test set. After the initial training step, the model is incrementally retrained with samples of 50 additional classes at each increment from the training set, followed by classification of samples of known classes and samples from 0, 50, 100, 150 and 200 additional unknown classes from the test set. The Nearest Non-Outlier (NNO) algorithm, which generates a decision boundary using a thresholded distance from the nearest class mean (Mensink et al., 2013), was used to compare the performance of EVM on ILSVRC2014. EVM performed better than NNO both in terms of F1 score and classification accuracy.[3]

Despite its advantages, the computational load and storage requirements of EVM is highly dependent on how much redundancy is avoided/allowed in the model, while maintaining a good accuracy. Moreover, it fails to elaborate on the impact of encountering an outlier of a known class during training. Its performance is also dependent on the selection of certain distributional hyperparameters for which there is no known definite selection window. Also, authors used a pretrained AlexNet (Krizhevsky et al., 2012), pretrained on ImageNet to extract features of input images, which ensures that features of known and unknown classes are different and thus separable. However, it is not known how EVM would perform when unknown classes are not seen at all during training, that is when a non pretrained model is used to extract features. Therefore, while EVM might be an interesting open-set algorithm, it needs further work to address its disadvantages.

[3] https://www.kaggle.com/getting-started/221303

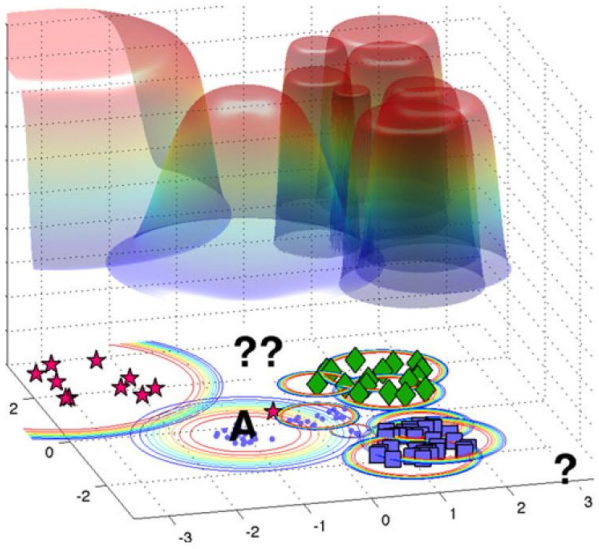


Figure 3.3: EXTREME VALUE MACHINE. This figure shows EVM algorithm trained on four classes where "?" represents unknown data points. "A" is the extreme vector (EV) for the class represented by blue dots. The isocontour rings represent the radial probability of sample inclusion boundaries for each of the four known classes. *Source: (Rudd et al., 2018)*

3.2.2 OpenMax

Bendale and Boult (2016) use two main concepts to develop a robust open-set classification algorithm for deep networks - Multi-class Meta-Recognition and OpenMax. A Meta Recognition system is known to guide the decisions of a recognition system by analyzing its recognition performance. In case of open-set classification, this Multi-class Meta-Recognition dwells on the idea of how far an input is from the known training space, wherein the known training space is defined for each known class. The function of OpenMax is to modify the SoftMax function by removing the constraint of all known class probabilities adding to 1, to accommodate prediction of an unknown class. These concepts together form the foundation that open-set deep networks use for classifying known samples with sufficient accuracy while rejecting any unknown samples during testing.

Using network scores/logit values from the penultimate layer (FC2 in Figure 3.1), also termed as Activation Vector (AV), a Mean Activation Vector (MAV) is computed for each class by taking the mean of all correctly classified training sample AVs. Each known class is thus represented as a point denoted by its MAV. The MAV for a given class is also assumed to be representative of related classes, since AVs for related classes are similar. Figure 3.4 shows AVs for four different ImageNet classes, denoted by the line corresponding to their real image. Related classes like hammerhead shark and great white shark respond similarly and hence, are seen to have similar levels of activation for correlated classes like shark, whales and fish.

During training, AVs are used to generate MAVs for each known class. Based on prior work on meta recognition that suggest that the final system scores follow Weibull distribution,[4] Weibull fitting is used to threshold on the largest of the distances between MAV of a known class and all correctly classified training samples of that class to demarcate a known boundary for that class (Scheirer et al., 2011). This gives rise to a probability parameter that estimates the probability

[4] https://www.weibull.com/hotwire/issue14/relbasics14.htm

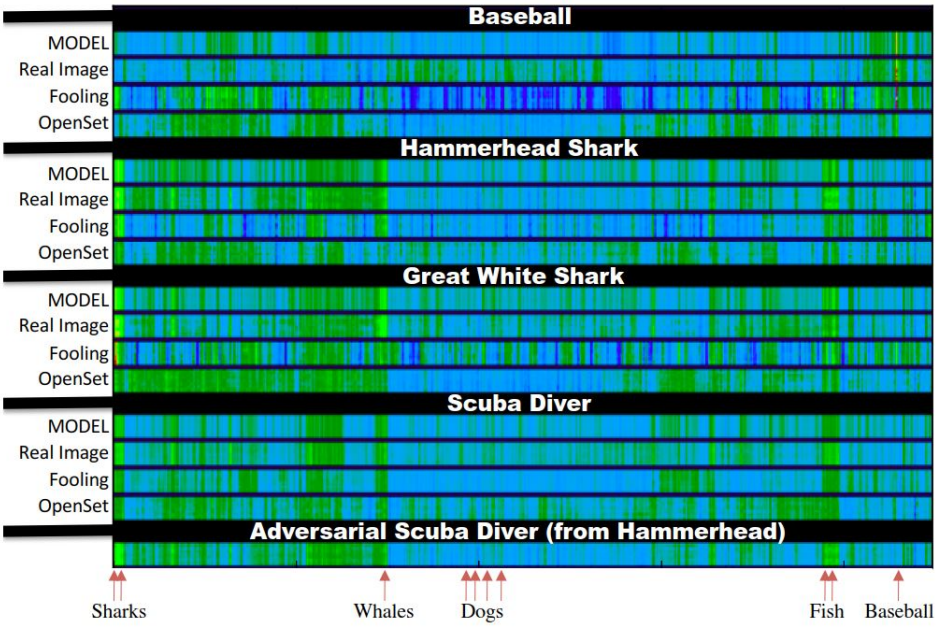


Figure 3.4: ACTIVATION VECTORS. This figure shows activation vectors or AVs for four different ImageNet classes and their corresponding fooling and open set AVs. While all 1000 classes are used for training, each AV in this Figure is shortened to have responses from the first 450 ImageNet classes, with responses for some categories shown at the bottom. Green lines represent high activation while blue lines represent low activation. *Source: (Bendale and Boult, 2016)*

of an input being outside the known boundary. This probability also provides meaningful insights about if an input belongs to the top few classes, instead of just the top-1 class for which it produced maximum activation during testing. Therefore, activation (logit values) for the top few classes are weighted and also, a pseudo activation is computed for unknown unknown class while maintaining the total activation level. An uncertainty threshold is further used on these OpenMax probabilities to determine the class of the input during testing.

The Meta-Recognition and OpenMax system thus allows to develop a robust open-set classification algorithm by training only on known samples. The training set of ILSVRC2012 was used to train OpenMax model. Since the test set of ILSVRC2012 is not publicly available, a test set was created using 50,000 validation images from ILSVRC2012, 15,000 fooling images (provided by Nguyen et al. (2015)) and 15,000 open set images drawn from 360 categories of ILSVRC2010 that were not present in ILSVRC2012. For evaluation, F-score is used, wherein true positives are determined using correct classification of validation set images, false positives are determined using incorrect classification of validation set images and false negatives are determined by the number of fooling and open-set images that are incorrectly classified as known classes by OpenMax.

OpenMax is shown to work well with open set images (unknown images that are known to exist in real world) and abstract fooling images as seen in Figure 3.5. However, OpenMax is shown to fail when presented with open-set images that are similar to known classes. For example, Bendale and Boult (2016) show that OpenMax incorrectly classifies an image of a Police van as an Ambulance. But, they do not experiment extensively with more such unknown samples and hence, it is difficult to gauge the effectiveness of OpenMax when it comes to unknown samples that are similar to known classes. Further, the disadvantage of using a pretrained network, pretrained on ImageNet, as mentioned in subsection 3.2.1, exists in case of OpenMax as well.

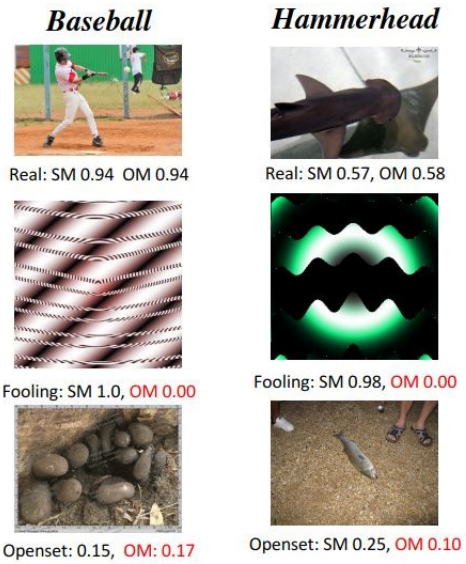


Figure 3.5: SOFTMAX VERSUS OPENMAX. This figure shows SoftMax scores (SM) and OpenMax scores (OM) for real, fooling, and open-set images of two ImageNet classes. OpenMax correctly identifies fooling and open-set images as long as they are not similar to known classes. For example, the image in the bottom left is not rejected as unknown, since it shares similar features with the known class of baseball. *Source:(Bendale and Boult, 2016)*

3.2.3 SoftMax with Garbage Class

Some recent approaches use a garbage class during training to enhance the network's capability of detecting unknown unknown classes during testing. Dhamija et al. (2018) use a separate set of known unknown classes to create a garbage class while Zhou et al. (2021) mix known classes to create the same. In this subsection, I provide details about these two approaches.

Garbage Class created using Known Unknown Classes

For this SoftMax with Garbage Class approach, a garbage class is created using a separate set of known unknown classes. The network is trained with samples from these classes, hoping that they are sufficiently representative of unknown unknown test samples. Dhamija et al. (2018) used NIST letters (Cohen et al., 2017) as known unknowns along with all MNIST classes as known classes. All known unknown classes are collected into a single garbage class by assigning them a common class label. If there are O known classes, then under this algorithm, there will be $O + 1$ total classes with O known classes plus 1 garbage class. The garbage class can be labeled as $t = 0$ and known classes can be labeled as $t = 1, 2, 3..., O$. Alternatively, known classes can be labeled as $t = 0, 1, 2, ..., O - 1$ and the garbage class can be labeled as $t = O$. The garbage class is labeled in such a way so that the model identifies it as one among the known classes. All known and known unknown classes are then weighted according to the number of training samples per class according to the following equation:

$$\lambda_t = \frac{N}{O \cdot N_t} \tag{3.2}$$

where N is the total number of training samples, O is the number of known classes and N_t is the number of training samples in the class for which the weight is being calculated. These weights

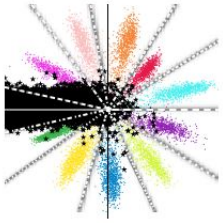


Figure 3.6: BOTTLENECK PLOT USING SOFTMAX WITH GARBAGE CLASS. This figure shows bottleneck plot for 10 known MNIST classes (colored dots) and unknown unknown Devanagari classes (black dots) when trained using SoftMax with Garbage Class, created using known unknown NIST letters. *Source:(Dhamija et al., 2018)*

follow the logic that classes with a lower number of samples are weighted higher than those with a higher number of samples. When garbage class is added as one of the classes, it usually contains a higher number of samples as compared to the known classes, given that it is a collection of a number of known unknown classes. Known classes not only contain fewer samples than the garbage class but are also almost balanced in a typical dataset. So, all the known classes tend to have similar weights while the garbage class gets a much smaller weight. All weights sum to the total number of samples. The loss that is then calculated is weighted by these weights. As a result, garbage class samples are pushed into a separate part of the deep feature space than that occupied by the known classes as can be seen in Figure 3.6.

Garbage Class created by mixing Known Classes

Zhou et al. (2021) propose a novel open-set algorithm called the PlaceholdeRs for Open-SEt Recognition or PROSER, where a classifier learns adaptive thresholds for different known class combinations to effectively separate known space from unknown space. PROSPER also makes use of a garbage class to achieve this open-set objective, but instead of using a separate set of known unknown classes, it mixes known classes to create a garbage class.

PROSPER learns two kinds of placeholders to achieve the open-set goal - classifier placeholders and data placeholders. The function of learning classifier placeholders is to create dummy classifiers which output the second best known class label for an input image and thus get placed somewhere between the target known class and non-target known class. For example, as can be seen in Figure 3.7, a dummy classifier is created using a sample of Known Class 3 and gets placed between Class 3 (target known class) and Class 2 (non-target class). Thus, when closed-set classifiers are augmented with these dummy classifiers, they serve as input sample based adaptive thresholds, learnt by only using known class samples.

The function of learning data placeholders is to generate samples that have a distribution different from that of known samples and are generated by mixing pairs of known sample distributions. These new samples then push the decision boundaries of the known classes using which they are generated, resulting in tighter boundaries for those known classes and thus reducing overlap between known and unknown space as illustrated in Figure 3.8.

It is by adding these placeholders to pretrained closed-set classifiers that the training is then carried out, without the use of separate known unknown classes, to perform open-set classification. Upon evaluating PROSPER using different multi-class datasets and comparing it with other state-of-the-art open-set algorithms like OpenMax, it is shown to achieve better performance in both correctly classifying known classes and correctly detecting variety of unknown samples.

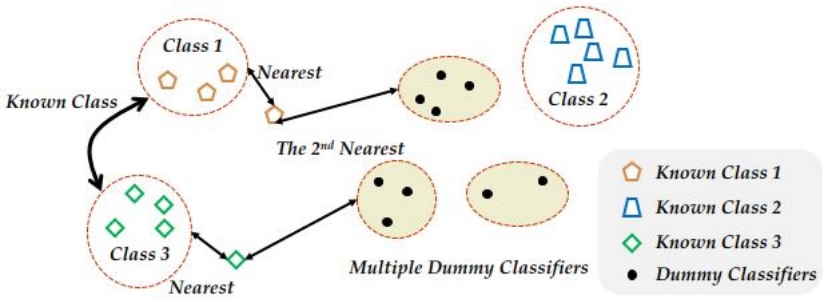


Figure 3.7: CLASSIFIER PLACEHOLDERS IN PROSPER. Multiple dummy classifier placeholders are created using samples of known class 1 and known class 3, which when augmented to a closed-set classifier, serve as input sample based adaptive thresholds for these known classes. *Source: (Zhou et al., 2021)*

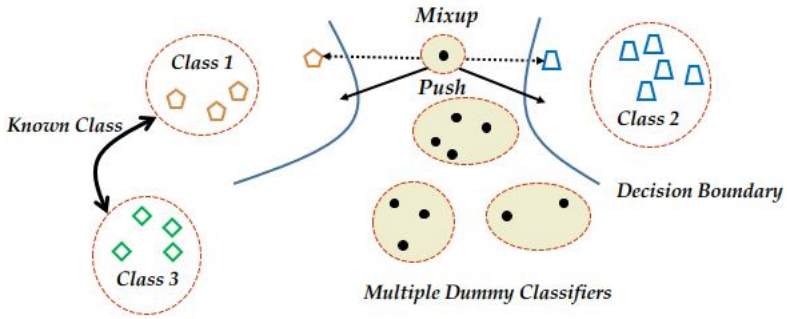


Figure 3.8: DATA PLACEHOLDERS IN PROSPER. A Data Placeholder is created by mixing known distributions of Class 1 and Class 2. The presence of this data placeholder pushes for tighter decision boundaries for Classes 1 and 2. *Source:(Zhou et al., 2021)*

3.2.4 Entropic Open-set Loss

In this approach, a novel Entropic Open-set Loss is used to satisfy the goal of reducing overlap between known and unknown samples. During training, logit values for all samples from known unknown classes are made to be equal, so that they produce equal SoftMax probabilities for all known classes when presented with an unknown sample (Dhamija et al., 2018). Unknown unknown test samples are assumed to behave similarly as known unknown training samples. Known class labels (SoftMax targets) are converted into one-hot vectors and known unknown class labels are converted into $1/O$ vectors of length O, where O is the number of known classes. This is done assuming that unknown samples share features with known samples in the open space. For example, if there are five known classes $t = 1, 2, 3, 4, 5$, then the one-hot vector for $t = 1$ will be $[1, 0, 0, 0, 0]$ while the $1/O$ vector for a known unknown class like $t = 0$ will be $[1/5, 1/5, 1/5, 1/5, 1/5]$. Samples are then trained using these modified targets and a modified softmax loss. The Entropic Open-set Loss can be calculated as follows:

$$J_E(x) = -\sum_{o=1}^{O} t_o \ln y_o \tag{3.3}$$

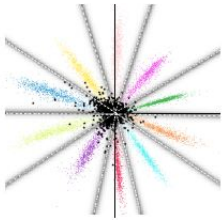


Figure 3.9: BOTTLENECK PLOT USING ENTROPIC OPEN-SET LOSS. This figure shows bottleneck plot for 10 known MNIST (colored dots) and unknown unknown Devanagari classes (black dots) when trained using Entropic Open-set Loss on MNIST classes and known unknown NIST letters. *Source: (Dhamija et al., 2018)*

where t_o represents element of the target vector for class o and y_o represents the predicted SoftMax output for it.

During training, if a sample belongs to a known class, it is trained as it is, i.e., using a standard Categorical Cross Entropy loss using one-hot targets, which can be calculated in the same way as Equation (3.3), the only difference being that all targets are one-hot vectors and there are no $1/O$ vectors. When a sample belongs to a known unknown class, its logit values are modified to be equal for all the known classes, and this value is very close to 0. This is done to make the network behave equally towards all known classes, without any bias for their features. Equal logit values means equal SoftMax probabilities, and when this is the case, deep features are seen to gather around the origin of the feature space, as seen in Figure 3.9.

I select SoftMax with Garbage Class, created using a separate set of known unknown classes and Entropic Open-set Loss to experiment with in this thesis due to their simplicity in terms of implementation and ease of understanding the concepts behind these algorithms. It would indeed be interesting to see if these easy-to-implement open-set algorithms are equally effective on open-set partitions created using large and rich datasets, as they are on open-set partitions created using small datasets. It would also be interesting to see how these algorithms handle unknown unknown classes that are similar to known classes in appearance.

3.3 Comparison with Out of Distribution Detection

Out-of-Distribution Detection or OoD detection is an application that is closely related to open-set classification. The main difference between the two is that the former considers only a binary problem at hand, where the task is to classify a sample as either known or unknown, with no further need to correctly classify known classes. In other words, OoD detection only serves the goal of determining whether an input belongs to the training distribution or not, irrespective of whether the input is correctly classified when it does belong to the training distribution.

Roady et al. (2020) compare OoD detection and open-set classification algorithms using ImageNet, and show that Out-of-Distribution detector for Neural networks or ODIN (Hsu et al., 2020), an OoD detection algorithm, performs better than OpenMax (Bendale and Boult, 2016), an open-set classification algorithm. But the catch is that it compares only the OoD performance of these algorithms. So, ODIN may outperform OpenMax in terms of classifying an input as known or unknown, but how accurately do the identified known classes get classified correctly remains a question to be answered. When it comes to comparing the computational and memory cost of these algorithms, Roady et al. (2020) show that OpenMax has a much lower cost as compared to ODIN or Mahalanobis (Lee et al., 2018), another OoD detection algorithm.

Despite the difference between the two applications, they behave similarly when it comes to

system design and performance. As in the case of designing an effective open-set classifier, designing an effective OoD detector sometimes requires OoD data to tune hyperparameters (Lee et al., 2018; Liang et al., 2018). But given the infinite space of OoD data, it is not possible to tune hyperparameters using every possible OoD data, thereby leading to tuning them only for selective OoD data. While there are some algorithms that do not need OoD data for training like ODIN and OpenMax, both open-set classification and OoD detection algorithms face a common challenge of correctly detecting unknown test samples that are closely related to the known classes/distribution.

An ideal open-set classifier must not only detect unknown test samples, but it must also be able to correctly classify known samples. To satisfy this two-fold goal, a two-stage classifier needs to be designed, where at the first stage OoD samples are detected and in the second stage, closed-set classification of known samples is done. Even though this is a feasible solution, the computational and memory cost disadvantage of effective OoD classifiers like ODIN would still persist. Further, it won't be possible to learn separate features for unknown samples using OoD detection, to reduce overlap between known and unknown samples. This is why I skip evaluating a two-stage classifier in this thesis.

Chapter 4

Open-set ImageNet Protocols

An ideal open-set algorithm must correctly classify known samples and correctly detect unknown samples. This includes detecting unknown samples that are not similar in appearance to known classes as well as detecting those that are similar or are related to the known classes in some way. For example, if known classes are birds, dogs and fruits, an ideal open-set classifier must correctly classify an image of a flying bird as well as an image of a bird perched on a tree as belonging to the class of birds. Similarly, it must classify an image of a banana and an image of an apple as belonging to the class of fruits. This is given that the classifier has been trained with different types of images of birds, dogs and fruits. Now, when this open-set classifier is shown an image of a computer, it must detect it as an unknown, since it never saw anything like that during training. Similarly, it must also detect an image of a flying plane as an unknown during testing and not confuse it with an image of a flying bird. It must also detect an image of a beetroot as an unknown and not incorrectly classify it as a fruit, given that it might look similar to the image of an apple that the classifier was trained with.

To achieve this open-set objective, I design three open-set ImageNet protocols using the dataset of ILSVRC2012. Each protocol has three sets of classes, corresponding to known, known unknown and unknown unknown components of the open space, as described in chapter 1. All 1000 classes of ILSVRC2012 are not used to create the open-set partitions. Instead, a small number of classes are used, which reduces the amount of time taken to perform the experiments, while maintaining hierarchical richness of the dataset and closeness to real-world open space. Till now, there have been no research work that use the same ImageNet dataset to create these components of open space. For example, Bendale and Boult (2016) used 1000 classes of ILSVRC2012 as known classes and, fooling images and images from 360 categories of ILSVRC2010 that were not present in ILSVRC2012, as unknown classes, while Roady et al. (2020) used 500 random ImageNet classes as known classes and the remaining 500 classes as unknown classes.

I design three different protocols to create three different artificial open spaces, with each having a varying level of similarity in appearance between known and unknown unknown classes. The protocols also incorporate varying level of difficulty in classifying known classes that share similar features. Each protocol is also created using different sets of known unknown classes, to train the network to better differentiate between known and unknown classes. Known unknown classes are selected such that they have increasing similarity to known classes, which increases the network's ability to detect unknown unknown samples during testing (Dhamija et al., 2018). It is on these protocols that I then compare the performance of the algorithms, that is SoftMax with Garbage Class and Entropic Open-set Loss, thus setting a high quality benchmark for these algorithms. It is also a first time there has been such a comparison between state-of-the-art open-set algorithms using open-set ImageNet protocols that address different real-world test situations and are designed using the same ImageNet challenge dataset. The protocols and their open-set partitions are presented in Table 4.1. The superclasses that make these protocols are presented in

Table 4.2. The superclasses, along with their class IDs are presented in Table A.1, in Appendix A. I study the ImageNet-1k tree structure[1] and use the `robustness` library to create these protocols, details of which I explain later in section 5.1.

As can be seen in Table 4.1, Protocol 1 uses all descendants of the dog superclass as known classes and addresses unknown unknown classes that are not related to dogs in any way, thereby making it easier to identify them. For example, an image of a car does not share any features with an image of a Siberian husky and thus is easier to reject as unknown. But the known classes are difficult to classify in this protocol, as the classifier needs to correctly identify the exact descendant of the dog superclass. Known unknown classes like zebra, monkey and fox are similar to the known dog classes in that they are all four-legged animals.

Protocol 2 considers some descendants of a subclass of the dog superclass as known classes (I choose the subclass of hunting dogs as this subclass has the highest number of descendants) and addresses unknown unknown classes that share similar features as known classes. For example, wolves and toy dogs share features like four legs and ears on top of head with hunting dogs and hence, there is the possibility of the classifier confusing these images with that of a hunting dog. Known classes are difficult to classify in this protocol as well, given that the classifier needs to identify the exact descendant of hunting dog. Known unknown classes are similar to the known classes in that they are both descendants of hunting dog superclass, but different descendants of hunting dogs are used as known unknown classes than those used as known classes.

Protocol 3 is created using `robustness` library's pre-packaged ImageNet dataset of "mixed_13" which has a mixture of living things/classes and non-living things/classes. It uses some descendants of `mixed_13` as known classes, and descendants of some classes, other than the `mixed_13` classes as unknown unknown classes, in a way that the unknown unknown classes are also a mixture of living things/classes and non-living things/classes. This leads to the possibility of the unknown unknown classes being related to one or more known classes. For example, an image of a flying aircraft may be confused for an image of a flying bird or since it is a means of transportation, the classifier might confuse it with other means of transportation like car, truck or boat. Similarly, an image of an ungulate like goat or horse or hippopotamus share features like being 4-legged with an image of a dog or a monkey or a cat, thus making it difficult for the open-set algorithm to avoid confusing these unknown unknown classes as one of the known classes. But unknown unknown classes in this protocol may not share as many features with known classes as in case of Protocol 2, and hence might be easier to detect in comparison to those in Protocol 2. Classification of known classes should be easy for this protocol as there is clear distinction between superclasses, but identifying the exact descendant of a superclass might still be difficult. Known unknown classes are made similar to the known classes by using different descendants of `mixed_13` superclasses than those used as known classes.

The open-set algorithms experimented with in this thesis use known unknown classes for training. Therefore, the training and validation sets are made up of known and known unknown classes. The test set is made up of known, known unknown and unknown unknown classes. Known unknown classes are also included in the test set to check whether the open-set algorithm behaves differently towards samples of these classes that were seen during training and those that were never seen. The number of training, validation and test images for known, known unknown and unknown unknown classes for each Protocol is shown in Table 4.3.

[1] https://observablehq.com/@mbostock/imagenet-hierarchy

Open-set Partition	Protocol 1	Protocol 2	Protocol 3
Known classes	All dog classes (116 descendants)	Some classes of a particular dog subclass (31 descendants)	Some classes of some ancestors (159 descendants)
Known Unknown classes	Some other 4-legged animal classes like zebra, monkey, fox, etc (67 descendants)	Some other classes of the same dog subclass as the known classes (30 descendants)	Other classes of the same ancestors as the known classes (137 descendants)
Unknown Unknown classes	Some non-animal classes like car, food, sunglasses, etc (166 descendants)	Some classes of another dog subclass and some other 4-legged animal classes (55 descendants)	Some classes of other ancestors (116 descendants)

Table 4.1: OVERVIEW OF OPEN-SET IMAGENET PROTOCOLS. This table gives an overview of the three open-set ILSVRC2012 based protocols and how they are split into known, known unknown and unknown unknown classes, along with the number of descendants for each partition (highlighted in red).

Open-set Partition	Protocol 1	Protocol 2	Protocol 3
Known classes	Dog	Some descendants of Hunting dog	Some descendants of Dog, Bird, Insect, Furniture, Fish, Monkey,Car, Cat,Truck, Fruit, Fungus, Boat, Computer
Known Unknown classes	Fox, Wild dog, Wolf, Feline, Bear, Musteline mammal, Ungulate, Primate	Other descendants of Hunting dog	Other descendants of Dog, Bird, Insect, Furniture, Fish, Monkey, Car, Cat, Truck, Fruit, Fungus, Boat, Computer
Unknown Unknown classes	Food, Motor vehicle, Device	Toy dog, Fox, Wild dog, Wolf, Feline, Bear, Musteline mammal, Ungulate	Reptile, Clothing, Ungulate, Vegetable, Aircraft

Table 4.2: IMAGENET SUPERCLASSES THAT MAKE THE PROTOCOLS. This table shows the ImageNet superclasses that were used to create the protocols. Known and known unknown classes are used for training the open-set algorithms experimented with in this thesis, and unseen samples from all the three partitions are used for testing.

Protocols	Open-set Partition	Training Size	Validation Size	Test Size
Protocol 1	Known	116,212	29,061	5,800
	Known Unknown	69,680	17,420	3,350
	Unknown Unknown	-	-	8,300
Protocol 2	Known	30,629	7,661	1,550
	Known Unknown	30,055	7,517	1,500
	Unknown Unknown	-	-	2,750
Protocol 3	Known	161,661	40,425	7,950
	Known Unknown	140,477	35,122	6,850
	Unknown Unknown	-	-	5,800

Table 4.3: PROTOCOL SPLITS. Number of training, validation and test images for open-set partitions of each open-set ImageNet protocol.

Chapter 5

Experimental Set-up

To perform experiments using Entropic Open-set Loss and SoftMax with Garbage Class (created using known unknown classes) on open-set ImageNet protocols, I need to first set up the experiment base. That is create dataset for each protocol by accessing specific hierarchies of ImageNet, wrap the dataset into an iterable of batches, initialize the network and set hyperparameters. In this chapter, I provide details about each of these steps and thus lay the foundation for the experiments ahead. All experiments follow the same set up. I use Python programming language, the PyTorch deep learning framework and GPUs (CUDA 10.1) provided by the Artificial Intelligence and Machine Learning Group (AIML) at the University of Zurich, Switzerland to create this set-up and for performing my experiments.

5.1 Accessing ImageNet and its hierarchy

ImageNet can be accessed in two ways to conduct experiments - accessing the entire dataset or accessing hierarchical parts of the dataset. Given the fast-paced developments in computer vision, both ways of accessing ImageNet have been packed into easy-to-use packages and libraries, to expedite the process of experimental set-up.

5.1.1 Accessing entire dataset

In PyTorch, one can simply make use of `torchvision.datasets.ImageNet`[1] to access the entire dataset, which can then be loaded into the memory using PyTorch `DataLoader`[2], in an iterative manner. In other words, the dataset is wrapped into an iterable of batches, with certain fixed number of samples per batch called the `batch_size`.

Once batches of data are loaded into the memory, the entire dataset is ready to be experimented with. But given the hierarchical complexity of ImageNet, the larger number of classes and the higher resolution of images as compared to other existing image datasets like MNIST and CIFAR-10,[3] it takes significant amount of time to run experiments using the entire dataset. It would be easier to run experiments using a smaller number of classes while still maintaining the hierarchical richness of ImageNet.

[1] https://pytorch.org/vision/stable/datasets.html#imagenet
[2] https://pytorch.org/docs/stable/data.html
[3] https://www.cs.toronto.edu/~kriz/cifar.html

```
WordNet ID: n03093574, Name: consumer goods, #ImageNet descendants: 61
WordNet ID: n01905661, Name: invertebrate, #ImageNet descendants: 61
WordNet ID: n02087122, Name: hunting dog, #ImageNet descendants: 61
WordNet ID: n04341686, Name: structure, construction, #ImageNet descendants: 55
WordNet ID: n01503061, Name: bird, #ImageNet descendants: 52
```

(a) Browsing superclasses and the number of descendants under each

```
Superclass | WordNet ID: n04230808, Name: skirt
ImageNet subclass | WordNet ID: n04136333, Name: sarong
ImageNet subclass | WordNet ID: n03866082, Name: overskirt
ImageNet subclass | WordNet ID: n03770439, Name: miniskirt, mini
ImageNet subclass | WordNet ID: n03534580, Name: hoopskirt, crinoline
Superclass | WordNet ID: n04105068, Name: roof
ImageNet subclass | WordNet ID: n04417672, Name: thatch, thatched roof
ImageNet subclass | WordNet ID: n04523525, Name: vault
ImageNet subclass | WordNet ID: n03220513, Name: dome
ImageNet subclass | WordNet ID: n04435653, Name: tile roof
```

(b) Browsing superclasses and their subclasses

Figure 5.1: BROWSE IMAGENET HIERARCHY USING `ROBUSTNESS` LIBRARY. This figure shows how the ImageNet hierarchy can be accessed using `robustness` library. (a) shows an example of the first way of accessing this hierarchy, where one can view the Superclass ID, its name and the number of ImageNet descendants under it. (b) shows an example of another way of accessing the hierarchy where one can view the Superclass ID and name, along with the class IDs and names of each of its underlying subclasses.

5.1.2 Accessing hierarchical parts of the dataset

To access the hierarchy of ImageNet at different levels, I use the `robustness` library[4] developed by students of Massachusetts Institute of Technology's (MIT) Mądry Lab, created using the original WordNet hierarchy. This library provides a three-fold functionality when it comes to ImageNet.

Firstly, it provides pre-packaged ImageNet-based smaller datasets. "living_9", for example, is a pre-packaged dataset provided by this library which includes the superclasses of Dog, Bird, Arthropod, Reptile, Primate, Fish, Feline, Bovid, and Amphibian. More of these pre-packaged ImageNet based smaller datasets are defined on their website.[5] Creating these smaller datasets also comes with an option of selecting balanced number of images for all the classes. I use one such pre-packaged dataset provided by this library, called `mixed_13`, to design one of the three open-set ImageNet protocols for this thesis, as mentioned in chapter 4.

Secondly, this package provides an easy way to browse the hierarchy within ImageNet, either in terms of superclasses and the number of descendants under each as shown in Figure 5.1(a) or in terms of superclasses and their corresponding subclasses as shown in Figure 5.1(b). Given a superclass id, the `robustness` library uses a `tree` object, to enumerate through all the descendants of the superclass.

Lastly, this library provides the functionality of creating custom datasets with desired number of subclasses using a `get_superclasses()` function. An option of selecting a superclass for the subclasses is also provided, along with the option of having balanced number of images for each subclass. For example, one can create a dataset of three subclasses of the *dog* superclass or create a dataset with four random ImageNet subclasses with the choice of balanced or unbalanced number of images per class.

The `robustness` library provides these functionalities through the `CustomImageNet` and `ImageNetHierarchy` classes. Open-set ImageNet protocols, described in chapter 4 are created using these classes and functionalities.

[4] https://robustness.readthedocs.io/en/latest/index.html
[5] https://robustness.readthedocs.io/en/latest/example_usage/custom_imagenet.html

5.2 Custom Dataset and Dataloader

To prepare the dataset, I leverage PyTorch's functionality of creating custom datasets, which can be directly fed into a PyTorch `DataLoader` for training a network, just like when using in-built PyTorch datasets. Custom datasets can be created using PyTorch's `Dataset`[6] class by simply overriding two of its subclass functions, that is `__len__()`, which returns the size of the dataset and `__getitem__()`, which returns a sample of the dataset using indexing. For this thesis, I create custom datasets for each open-set ImageNet protocol, after accessing the desired level of hierarchy using `robustness` library, as per Tables 4.1 and 4.2. While creating these custom datasets, I also modify the labels for each class to match an integer numbering system.

I have access to only the training and validation sets of ILSVRC2012 and therefore, I use 80% of the training set for training the network, remaining 20% of the training set to validate and the entire validation set to test the network. The network is trained using known and known unknown classes and tested using known, known unknown and unknown unknown classes.

While creating the dataset, certain transformations[7] need to be applied to images to have a uniform image dataset. Since ImageNet images come in different shapes and sizes, all of them first need to be resized to a common size using `Resize`. Since I considered square images for my experiments, I resized all images to a size of 300×300, where the first dimension (H) corresponds to the height of the image and the second dimension (W) corresponds to the width of the image. I then crop all the images at the center to a size of 224×224 using `CenterCrop` and use `ToTensor` to convert these images to PyTorch tensors. I also normalize the images across each channel (C) using `Normalize`. Since ImageNet images are color images, there are three channels ($C = 3$) and images need to be normalized across all the channels, with a mean and standard deviation for each channel. The mean and standard deviation that I use to normalize the images are $[0.485, 0.456, 0.406]$ and $[0.229, 0.224, 0.225]$ respectively.[8] Apart from these transformations, I also augment the training data using two additional image transformations - `RandomHorizontalFlip` and `RandomRotation`. Data augmentation is shown to reduce overfitting to the training data and improve validation accuracy (Margolis, 2021). Hence I augment the training data by horizontally flipping images with the default probability of 0.5 and randomly rotating images by an angle of ± 10 degrees. The way these transformations are applied to the images is specific to this thesis and is not necessarily the recommended way (Krizhevsky et al., 2012).

Training, validation and test custom datasets created for each protocol, can be directly wrapped into an iterable of batches using PyTorch `DataLoader`. Since I use images of size 224×224, which is quite large, it is possible to accommodate a maximum `batch_size` of 64 in the memory. Increasing the `batch_size` beyond 64 throws a memory error. I also shuffle the training images to avoid overfitting (`shuffle = True`) and use 10 loader worker processes to allow multi-process data loading (`num_workers = 10`).[9]

5.3 Network and Hyperparameters

Once the data is loaded, the next step is to train a network with this data. For my experiments, I use the deep network architecture of ResNet-50 (He et al., 2016).

Deeper networks are not only shown to achieve better accuracy on object classification and object localization tasks using ImageNet (Simonyan and Zisserman, 2015), but they are also shown

[6] https://pytorch.org/tutorials/beginner/data_loading_tutorial.html#dataset-class
[7] https://pytorch.org/vision/stable/transforms.html
[8] https://towardsdatascience.com/how-to-train-cnns-on-imagenet-ab8dd48202a9
[9] https://pytorch.org/docs/stable/data.html#single-and-multi-process-data-loading

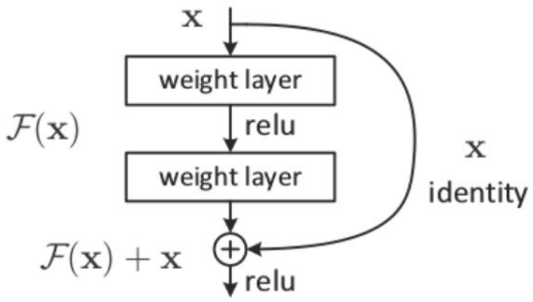


Figure 5.2: RESIDUAL BLOCK. This figure shows a residual unit in a deep residual learning framework. The curved arc represents a skip connection that directly feeds the input x to the end of the unit using identity mapping, to give the output of $\mathcal{F}$(x) + x, thus avoiding a pass through the activation layers. *Source: (He et al., 2016)*

to achieve better open-set performance (Roady et al., 2020). But plain deep networks created by simply stacking layers face the issue of vanishing/exploding gradients. Gradients are calculated during back-propagation using the chain-rule,[10] where gradients for the initial layers would consist of a long series of multiplications of derivatives in case of a very deep network. Amongst these derivatives, the derivative of the sigmoid and tanh activation functions are very small, which when repeatedly multiplied with other derivatives, lead to smaller and smaller numbers, eventually causing a vanishing gradient problem.[11] This leads to the network failing to learn the weights and biases (trainable parameters) of the initial layers, which play a crucial role in identifying the most basic features of an input image, thereby causing the network accuracy to degrade.

He et al. (2016) present a solution to this problem by introducing the concept of deep residual learning framework. A residual network makes use of residual or skip connections as shown in Figure 5.2, to directly add the value of the input x at the end of the residual block to give an output of $\mathcal{F}$(x) + x. The skip connection, which has an identity mapping, thus skips the pass through the activation layers (*relu* activation as shown in Figure 5.2) that suppresses the gradient, resulting in a larger overall derivative of the block. In case the dimensions of x and $\mathcal{F}$(x) do not match (which should match as it is a necessary condition to perform the addition operation), the residual block uses convolutional layers and batch normalization (Ioffe and Szegedy, 2015) to reduce dimensions.

ResNet architectures use two kinds of residual blocks - residual blocks that skip two convolutional layers, as used in ResNet-18 and ResNet-34, and residual blocks that skip three convolutional layers, as used in ResNet-50 onwards. Figure 5.3 shows ResNet architectures of ResNet-18, 34, 50, 101 and 152, with a square bracket representing a residual block and the number of such blocks used shown by the side of these brackets. The number of floating point operations per second (FLOPs) for each ResNet model is shown at the bottom. As mentioned earlier, I use ResNet-50 for my experiments, highlighted in red in Figure 5.3, which has 48 convolutional layers along with one MaxPool and one AveragePool layer, and can perform as many as 3.8 billion FLOPs. I use ResNet-50 because it is a sufficiently deep network to learn complex features, with a considerable number of trainable parameters and considerably good top-1 and top-5 accuracy[12] on ImageNet. Going deeper greatly increases the number of trainable parameters but does not lead to a significant increase in accuracy (He et al., 2016; Bianco et al., 2018; Leong et al., 2020).

[10] https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
[11] https://www.kaggle.com/shrutikunapuli/activation-functions-for-neural-networks
[12] https://pytorch.org/vision/stable/models.html

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 5.3: RESNET ARCHITECTURES. This figure shows deep residual learning architectures with 18, 34, 50, 101 and 152 layers. Square brackets represent a residual block. I use ResNet-50 for my experiments, highlighted in red. *Source: (He et al., 2016)*

I load the network using PyTorch's `torchvision.models.resnet50`.[13] One can use both pretrained and non pretrained version of the model, where the former is pretrained on ImageNet images. Since the pretrained ResNet-50 model has already seen ImageNet images, it is not justified that I use that version for open-set classification, where a classifier/network faces unknown unknown images only during testing and has no idea about their existence during training. Hence, I turn the `pretrained` parameter of `torchvision.models.resnet50` to `False` for all my experiments.

Additionally, the network output or the last SoftMax layer is no longer needed to have a 1000-way classification, since I do not use all the classes for my experiments. The SoftMax layer is modified to have a reduced number of output classes depending on the protocol and open-set algorithm experimented with.

After loading the model, I use the Stochastic Gradient Descent with momentum (SGD) optimizer with a learning rate of 0.01 and a momentum of 0.9, to store and update the model parameters during training, and I use `torch.optim.SGD`[14] for this purpose (Liu et al., 2020). I maintain the same optimizer, learning rate and momentum for all my experiments. The reason I choose SGD optimizer (that is mini-batch SGD with a batch size of 64) is that it is relatively easy to understand and simple to implement. A learning rate of 0.01 is used to avoid both slow learning, which happens if I use a learning rate of 0.001 and, to avoid divergence from the point of minimum loss, which happens if I use a learning rate of 0.1, and thus, maintain a *just right* learning rate.[15] Using momentum decreases the time taken by SGD to converge to the point of minimum loss by shifting gradient vectors in the right direction by a certain factor. By using a momentum factor of 0.9, I strike a balance between minimally shifting the gradient vectors (momentum factor of 0.5) and shifting them too much (momentum factor of 0.99) in the direction of minimum loss (Goodfellow et al., 2016).[16]

[13] https://pytorch.org/vision/stable/models.html#torchvision.models.resnet50
[14] https://pytorch.org/docs/stable/generated/torch.optim.SGD.html
[15] https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/
[16] https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d

Chapter 6

Experiments

In this thesis, I experiment with two state-of-the-art open-set algorithms namely Entropic Open-set Loss and SoftMax with Garbage Class. The main idea behind these algorithms is to reduce the overlap between known and unknown classes, given that under closed-set circumstances, unknown test samples tend to overlap with known decision regions, resulting in misclassification of unknown samples as known classes, sometimes with high confidence. The selected algorithms use known unknown classes for training, assuming that unknown unknown test samples will behave similarly as the known unknown training samples. Their open-set performance is compared using ImageNet based ILSVRC2012 protocols to that of a baseline traditional SoftMax, using two novel evaluation metrics. In this chapter, I will first discuss about these novel evaluation metrics and then dive into how I execute the selected open-set algorithms, train the network and use these metrics to evaluate their open-set performance.

6.1 Evaluation Metrics

It is difficult to gauge the performance of open-set algorithms, both in terms of capturing the ability to correctly classify known classes and correctly reject unknown classes, using commonly used evaluation metrics. For example, F1-score, the evaluation metric used by Bendale and Boult (2016), is mainly used to evaluate a binary problem of how successful a classifier is in understanding if a sample belongs to a particular class or not. This is a problem for multi-class classification as the F1-score is calculated for each class in a one-vs-rest manner, with different confidence thresholds for each class, rather than calculating an overall F1-score which judges the overall performance of the multi-class classifier.[1] Additional challenges are faced when it comes to comparing the performance of different open-set algorithms due to the unduly large number of unknown samples used to develop the algorithms as compared to known samples, different number of unknown samples used for developing different algorithms and different design and performance of algorithms (Dhamija et al., 2018). To overcome these difficulties, I use two novel evaluation metrics, details of which I discuss in this section.

6.1.1 Open Set Classification Rate Curve

Dhamija et al. (2018) developed the Open-Set Classification Rate Curve (OSCR curve) to evaluate the performance of open-set algorithms. This metric accurately satisfies the two-fold open-set goal of correctly classifying known samples as well as correctly rejecting unknown samples. The

[1] https://www.baeldung.com/cs/multi-class-f1-score

OSCR Curve is composed of two parts, as given below:

$$\begin{aligned} FPR(\tau) &= \frac{|\{x|t \in \text{unknown classes} \wedge \max_{1\leq o\leq O} y_o \geq \tau\}|}{\text{Number of unknown samples}}, \\ CCR(\tau) &= \frac{|\{x|t \in \text{known classes} \wedge \operatorname{argmax}_{1\leq o\leq O} y_o = t \wedge y_t \geq \tau\}|}{\text{Number of known samples}} \end{aligned} \tag{6.1}$$

where y_o represents the predicted SoftMax probability (confidence) for a sample x and y_t represents the predicted confidence for the correct class or ground truth t. FPR or False Positive Rate is the fraction of unknown samples that got incorrectly classified as a known class with a maximum confidence greater than confidence threshold τ. CCR or Correct Classification Rate is the fraction of known samples that got correctly classified (that is the correct class has the maximum confidence) with a confidence greater than τ. These two parts are then plotted against each other for varying values of τ between $[0, 1]$, with $\tau = 0$ and $FPR = 1$ corresponding to closed-set classification. Different confidence thresholds are computed for different algorithms at a given value of FPR. The larger the region covered by the OSCR curve, the better the performance of the open-set algorithm. OSCR curves can be plotted for both known unknowns and unknown unknowns, to measure the performance of an algorithm on both these unknown classes.

6.1.2 Average Confidence

Another novel evaluation metric that I use to evaluate open-set performance of the experimented with algorithms is Average Confidence. It measures how confident the network is in correctly classifying known classes and correctly rejecting unknown classes. I share this evaluation metric with the bachelor thesis of Schnyder (2021). It is computed differently for SoftMax with Garbage class and Entropic Open-set Loss, and differently for known and unknown classes. Confidence for computing Average Confidence for SoftMax with Garbage class can be computed as follows:

$$Conf = \begin{cases} y_t & \text{for t being the correct known class} \\ 1 - \max\limits_{o} y_o & \text{for unknown in SoftMax with Garbage Class} \end{cases} \tag{6.2}$$

Confidence for computing Average Confidence for Entropic Open-set approach can be computed as follows:

$$Conf = \begin{cases} y_t & \text{for t being the correct known class} \\ 1 - \max\limits_{o} y_o + \frac{1}{O} & \text{for unknown in Entropic Open-set approach} \end{cases} \tag{6.3}$$

The confidence formula for unknown classes applies to both known unknown and unknown unknown samples. Once the confidence is calculated, it can then be added for all the samples and divided by the total number of samples in the dataset, to compute the Average Confidence.

If a sample belongs to a known class, its confidence is equal to the SoftMax probability for that class. If a sample belongs to an unknown class, the goal is to minimize the maximum softmax probability. If the confidence for an unknown sample is high, upon getting subtracted from 1, this score drops drastically. In case of Entropic Open-set approach, this score can drop to a minimum limit of $1/O$. Average Confidence serves as a good stopping criterion to evaluate if a model is sufficiently capable of rejecting unknown samples as well as correctly classifying known samples during validation. Its value ranges from 0 to 1, with values closer to 1 showing better open-set performance than those close to 0.

6.2 Execution of Algorithms

Execution of the selected algorithms differs mainly in terms of the integer label for known unknown classes, dimension of the SoftMax output, the 1-Dimensional (1D) targets used for training (one-hot vectors for known classes and, one-hot vector or $1/O$ vector for known unknown classes, for O known classes), the loss function and the calculation of confidences to compute average confidence. The dimension of SoftMax output is modified while initializing the network depending on the algorithm used. To compute the loss, either a custom loss function is defined, or a PyTorch implemented loss is used. To convert integer class labels into 1D targets and calculate confidences, I use a Python Class.[2] `__init__()` and `__call__()` methods are used to initialize 1D vector objects and return a batch of images and their 1D targets. It must be kept in mind that 1D targets are created only for known and known unknown classes, to be used only during training. During validation and testing, original integer class labels are used. A batch of images returned have dimension of batch_size$\times$number of channels in the input image C$\times$its height H$\times$its width W, and their 1D targets have a dimension of batch_size $\times$ number of output classes. Two others methods are created to predict the class of an input image and calculate confidences based on if the input sample is from a known class or from an unknown class. Two differently calculated confidences are returned using these methods, one to compute FPR and CCR for OSCR curve (Equation (6.1)) and another to compute average confidence (Equation (6.2) or (6.3)). It is expected that when the input is a known sample, the predicted class is also the correct class, with a confidence greater than the confidence threshold, and when the input is an unknown sample, the maximum confidence is lower than that threshold. Three different classes are used for the three algorithms - Entropic Open-set Loss, SoftMax with Garbage Class and Traditional SoftMax. In this section, I provide details about the execution of the three algorithms.

6.2.1 Entropic Open-set Loss

The Entropic Open-set approach uses one-hot targets for O known classes and $1/O$ targets for known unknown classes, and a novel Entropic Open-set Loss for training, as described in subsection 3.2.4. The SoftMax layer is an O-dimensional vector. If an input sample belongs to a known class, it is expected that the network produces maximum confidence for that class. If an input is unknown, it is expected that the network produces equal confidence for all the O known classes, and this value is further expected to be as low as $1/O$, such that it is easily rejected as an unknown at confidence threshold values greater than $1/O$.

Integer labels used for the different open-set partitions for this algorithm are 0 to $O-1$ for O known classes, -1 for known unknown classes and -2 for unknown unknown classes. This is done so that, during validation and testing, the network only has partial information about the known unknown classes, in that the network only knows that these classes are different from the known classes as all the known classes have a zero or positive label while the known unknown classes have a common negative label. I select a different negative label for unknown unknown classes for differentiating between known unknown and unknown unknown samples during testing, though this is not necessary. Upon receiving these labels as arguments, `__init__()` initializes one-hot and $1/O$ vector objects, using which `__call__()` substitutes known class labels with their one-hot elements and known unknown class labels with their $1/O$ targets in the batches. Algorithm 1 can be used to have a detailed look of the Python Class created for this open-set algorithm.

[2] https://docs.python.org/3/tutorial/classes.html

6.2.2 SoftMax with Garbage Class

SoftMax with Garbage Class uses one-hot targets for $O + 1$ classes and a weighted Categorical Cross Entropy loss to push unknown samples into a separate part of the deep feature space, different from that of the known classes, as described in subsection 3.2.3. The garbage class, created using known unknown classes, has a higher number of samples than each of the known classes, and thus when weighted according to the number of training samples per target class, the garbage class gets a lower weight than the known classes. For this method, the SoftMax layer is a $(O + 1)$-dimensional vector, for O known classes and 1 garbage class. If an input sample belongs to a known class, it is expected that the network produces maximum confidence for that class. If an input is unknown, it is expected that the network produces very low confidences for all the O known classes, but a comparatively higher confidence for the garbage class.

Integer labels used for the three open-set partitions for this algorithm are 0 to $O - 1$ for O known classes, O for known unknown classes and -2 for unknown unknown classes. This is done so that the network treats the garbage class similarly as the known classes during validation and testing. Both known class and garbage class integer labels in the batches are then converted to one-hot vectors. Algorithm 2 provides details about the Python Class created for this open-set algorithm. The weighted Categorical Cross Entropy Loss can be calculated as follows:

$$J_E(x) = \lambda_t * (-\sum_{o=1}^{O} t_o \ln y_o) \tag{6.4}$$

where λ_t is the weight calculated for target class t as per Equation (3.2), t_o is an element of the one-hot vector and y_o is predicted confidence. PyTorch's `torch.nn.CrossEntropyLoss`[3] can be used to implement this loss, with the calculated weights assigned to its `weight` parameter.

6.2.3 Traditional SoftMax

The network is trained and validated with only known classes using traditional SoftMax and tested with known, known unknown and unknown unknown classes. It may guarantee a great performance in terms of correctly classifying known samples, but it cannot always guarantee correct rejection of unknown samples. There might be many unknown samples that get incorrectly classified as a known class with a high confidence because of large amount of overlap with known decision regions. These samples will therefore not show up under the rejection condition of having maximum confidence below a threshold. I use Traditional SoftMax as a baseline for comparing open-set performance of Entropic Open-set Loss and SoftMax with Garbage Class, which attempt to reduce the overlap between known and unknown samples, thereby preventing high known class confidence values for unknown inputs. Training the network with traditional softmax is almost same as using the Entropic Open-set method, except that I use a plain Categorical Cross Entropy loss and do not use known unknown classes for training. Table 6.1 summarizes how the network is trained using each of Entropic Open-set Loss, SoftMax with Garbage Class and Traditional SoftMax.

6.3 Network Training, Validation and Testing

A common function is defined for training, validating and testing the network. SoftMax with Garbage Class and traditional SoftMax are trained for 100 epochs, while Entropic Open-set Loss is trained for 200 epochs as it takes more epochs to train the network using Entropic Open-set Loss to have a better performance on known samples in the validation set. Average confidences

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

on validation and test sets are calculated after every epoch of training, but only the average validation confidence is used to monitor the performance of the open-set algorithm. An optional confidence threshold of 0.3 is also set, to calculate the percentage of correctly classified known samples and correctly rejected unknown samples, using this threshold. These values do not say anything about the overall performance of the open-set algorithm, but just provide an example of how the algorithm performs at the selected threshold. Setting a confidence threshold and calculating these values is completely optional and can be skipped. The overall performance is measured only using average confidence and the OSCR plot. During the training phase, for each batch of images and their 1D targets, the network calculates logit values and loss, performs backpropagation to calculate gradients and, updates the weights and biases (trainable parameters) using these gradients. For one epoch on the training set, the network's performance is validated on the validation set and then tested on the test set. During the validation phase, the trained network is used to predict the class for each input sample in the validation set and calculate average confidence using Equation (6.2) or (6.3). Average validation confidence can also serve as a good stopping criterion, but for my experiments, I keep training the network for the entire range of epochs without setting any condition for early stopping. Optionally, if a sample belongs to a known class, it is checked if its predicted class is equal to the correct class and if its confidence is greater than or equal to the confidence threshold of 0.3. If a sample belongs to a known unknown class, it is checked if its confidence is less than 0.3. During the testing phase, steps similar to the validation phase are carried out on known, known unknown and unknown unknown samples in the test set. Also, confidences for each test sample are saved to calculate CCR and FPR as per Equation (6.1) for plotting the OSCR curves. In terms of the optional step, it is further checked if the predicted confidence for an unknown unknown sample is also below 0.3.

	Open-set Partition	**Entropic Open-set Loss**	**SoftMax with Garbage Class**	**Traditional SoftMax**
Integer Labels	Known	0 to $O-1$		
	Known unknown	-1	O	-1
	Unknown Unknown	-2		
1D Labels	Known	one-hot vector		
	Known unknown	$1/O$ vector of length O	one-hot vector	Not used for training
Network Output	-	O	$O+1$	O
Loss Function	-	Entropic Open-set Loss	Weighted Categorical Cross Entropy Loss	Categorical Cross Entropy Loss
Avg. Confidence	Known	Predicted SoftMax output for ground truth y_t		
	Known unknown Unknown Unknown	$1-\max_o y_o+\frac{1}{O}$	$1-\max_o y_o$	$1-\max_o y_o+\frac{1}{O}$

Table 6.1: TRAINING USING ALGORITHMS. This table summarizes the way each algorithm is trained for O known classes (y_o is the SoftMax probability for class o).

Input: Batch of Images and their integer targets
Output: Batch of Images and their 1D targets
Data: Training Data

def `__init__`(*range of known targets* $= 0$ *to* $O - 1$, *known unknown target* $= -1$)**:**
    initialize known and known unknown targets
    initialize one-hot vector for known targets
    initialize a dictionary with integer known targets as keys and their one-hot elements as values
    initialize $1/O$ vector of length O

def `__call__`(*batch of images*, *batch of targets*)**:**
    input_vec = zero tensor of same dimensions as *batch of images*
    target_vec = zero tensor of dimension *batch_size* $\times$ O
    known_indices = indices where *target* $\neq$ *known unknown target*
    unknown_indices = $\neg$ known_indices
    fill input_vec with *batch of images* at known_indices and unknown_indices
    fill target_vec at known_indices with respective one-hot elements
    fill target_vec at unknown_indices with $1/O$ vector
    return input_vec (*batch of images*), target_vec (*batch of 1D targets*)

def `predict`(*batch of logit values*, *batch of targets*)**:**
    confidences = SoftMax probabilities calculated using *logit values*
    indexes = index with largest *logit value*
    max_confidence = confidence of class with largest *logit value*
    return [**for** *every sample i in batch* **do**
        (predicted known class for i^{th} sample,
        predicted confidence for i^{th} sample,
        if i^{th} *sample is from known class* **then**
            confidence of class of i^{th} sample
        else
            maximum confidence for i^{th} sample)
    ]

def `confidences`(*batch of logit values*, *batch of targets*)**:**
    confidences = SoftMax probabilities calculated using *logit values*
    return [**for** *every sample i in batch* **do**
        if i^{th} *sample is from known class* **then**
            confidence of class of i^{th} sample
        else
            1 - maximum confidence for i^{th} sample + $1/O$
    ]

Algorithm 1: PYTHON CLASS CREATED FOR ENTROPIC OPEN-SET APPROACH. This algorithm explains the Class created for Entropic Open-set algorithm with O known classes.

Input: Batch of Images and their integer targets
Output: Batch of Images and their 1D targets
Data: Training Data

```
def __init__(range of known targets = 0 to O − 1, known unknown target = O):
    initialize known and known unknown targets
    initialize one-hot vector for known targets
    initialize a dictionary with integer known class labels as keys and their one-hot
     elements as values
    initialize one-hot vector for known and known unknown targets
    initialize a dictionary with known and known unknown targets as keys and their
     one-hot elements as values

def __call__(batch of images, batch of targets):
    input_vec = zero tensor of same dimensions as batch of images
    target_vec = zero tensor of dimension batch_size × O
    known_indices = indices where target ≠ known unknown target
    unknown_indices = ¬ known_indices
    fill input_vec with batch of images at known_indices and unknown_indices
    fill target_vec at known_indices and unknown_indices with respective one-hot
     elements
    return input_vec (batch of images), target_vec (batch of 1D targets)

def predict(batch of logit values, batch of targets):
    logit = pick logit values of only known classes
    confidences = SoftMax probabilities calculated using logit
    indexes = index with largest logit
    max_confidence = confidence of class with largest logit
    return [for every sample i in batch do
        (predicted known class for i^th sample,
        predicted confidence for i^th sample,
        if i^th sample is from known class then
            confidence of class of i^th sample
        else
            maximum confidence for i^th sample)
        ]

def confidences(batch of logit values, batch of targets):
    logit = pick logit values of only known classes
    confidences = SoftMax probabilities calculated using logit
    return [for every sample i in batch do
        if i^th sample is from a known class then
            confidence of the class of i^th sample
        else
            1 - maximum confidence for i^th sample
        ]
```

Algorithm 2: PYTHON CLASS CREATED FOR SOFTMAX WITH GARBAGE CLASS. This algorithm explains the Class created for SoftMax with Garbage Class algorithm with O known classes. The steps highlighted in red are the ones that are different from Algorithm 1.

6.4 Results

In this section, I provide the results from my open-set experiments and compare the performance of the algorithms on the three open-set ImageNet protocols using two evaluation metrics - Average Confidence and OSCR curve. Average validation and test confidences are shown in Table 6.2 and OSCR curves for Protocols 1, 2 and 3 are plotted in Figures 6.1, 6.2 and 6.3 respectively. Average confidences are shown after every 25^{th} epoch of training on the training set, and OSCR curves are plotted for both known unknown and unknown unknown samples. As mentioned earlier, Entropic open-set Loss is trained for 200 epochs while the others are trained for 100 epochs, and hence, average confidence is calculated only till the end of training for 100 epochs for the other algorithms. Performance of the algorithms at the optional confidence threshold of 0.3 for Protocols 1, 2 and 3, in terms of percentage of correctly classified known samples with predicted confidence above the threshold and percentage of correctly rejected unknown samples with predicted confidence below the threshold, is shown in Tables A.2, A.3 and A.4 in Appendix A.

I summarize my observations based on average test confidence as follows:

- Average confidence improves as the network is trained for more number of epochs using Entropic Open-set Loss on Protocols 1 and 3. On Protocol 2, it keeps fluctuating between a small range. Highest average confidence is obtained on Protocol 1, followed by similar average confidences at the end of training for Protocols 2 and 3.
- Average confidence is seen to degrade as the network is trained for more number of epochs using SoftMax with Garbage Class, on all the three protocols. This degradation is more pronounced for Protocols 1 and 2, than for Protocol 3. At the end of training, this value is seen to be the highest for Protocol 1, followed by Protocol 3 and then Protocol 2.
- Training the network using plain SoftMax also results in decrease in average confidence for all the protocols, across epochs. At the end of training, highest average confidence is seen to be obtained on Protocol 1, followed by Protocol 3 and then Protocol 2.

In terms of OSCR curve, following are my observations for Protocol 1:

- At False Positive Rate (FPR) of 1, training the network using SoftMax with Garbage Class and Traditional SoftMax show almost equally better Correct Classification Rate (CCR) than when trained using Entropic Open-set Loss, for both known unknown and unknown unknown samples.
- At FPR of 0.1, training using all the three algorithms leads to almost equal CCR for known unknowns. For unknown unknowns, however, SoftMax with Garbage Class is seen to achieve better CCR than the others.
- For FPR values lower than 0.1, Entropic Open-set Loss is seen to achieve higher CCR than the other methods for known unknowns. For unknown unknowns, however, SoftMax with Garbage Class achieves higher CCR than the others, and almost similar performance is shown by Entropic Open-set Loss and Traditional SoftMax up till FPR values of around ≈ 0.001.
- Further, at low FPR values, curves are not seen to extend completely, for both known unknowns and unknown unknowns, except for the OSCR curve for Entropic Open-set Loss for known unknowns. The curve stops extending earlier for Traditional SoftMax as compared to the others, for both known unknowns and unknown unknowns.

Following are my observations regarding OSCR plots for Protocol 2:

- At FPR of 1, same observation can be made for all the three algorithms as that made on Protocol 1.
- Between FPR of 1 and 0.01, SoftMax with Garbage Class obtains higher CCR than the others, except for some FPR values close to and greater than 0.01, for known unknowns, where SoftMax with Garbage Class and Entropic Open-set Loss perform similarly.
- OSCR curves for SoftMax with Garbage Class and Traditional SoftMax stop extending beyond FPR of ≈ 0.01 for both known unknowns and unknown unknowns. OSCR curve for Entropic Open-set Loss extends more than OSCR curves for the others, and even extends completely to the y-axis for known unknowns.

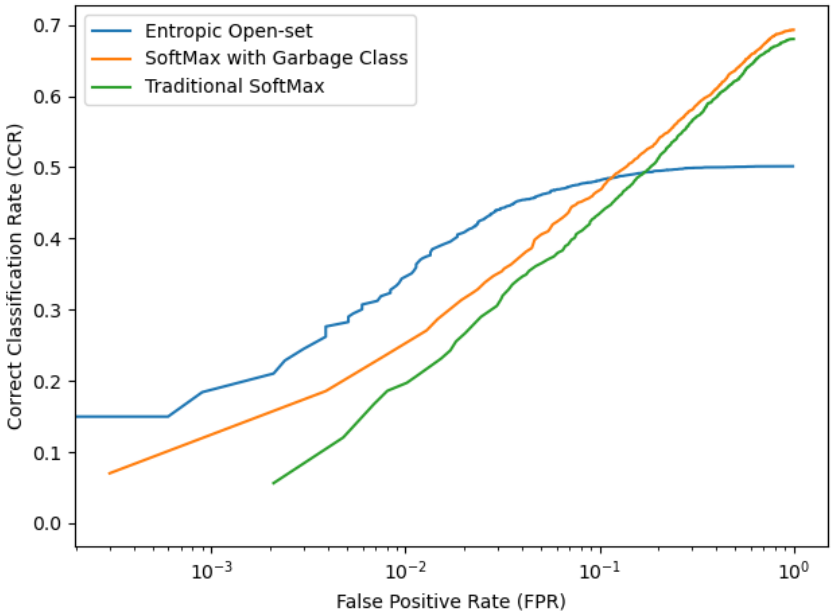
On OSCR curves for Protocol 3, following are my observations:

- As the FPR value goes from 1 to 0.01, the performance of SoftMax with Garbage Class and Traditional SoftMax is more diverged for this protocol than in case of Protocol 2 for known unknowns.
- For unknown unknowns, SoftMax with Garbage Class achieves higher CCR than the others with significantly higher CCR than Entropic Open-set Loss, until its curve stop extending (except at FPR values close to 1, where SoftMax with Garbage Class and Traditional SoftMax have almost same CCR).
- Observation about extension of OSCR curves for Protocol 2 can be made in case of Protocol 3 as well, except that it extends slightly beyond FPR of 0.01 for SoftMax with Garbage Class and Traditional SoftMax for unknown unknowns.

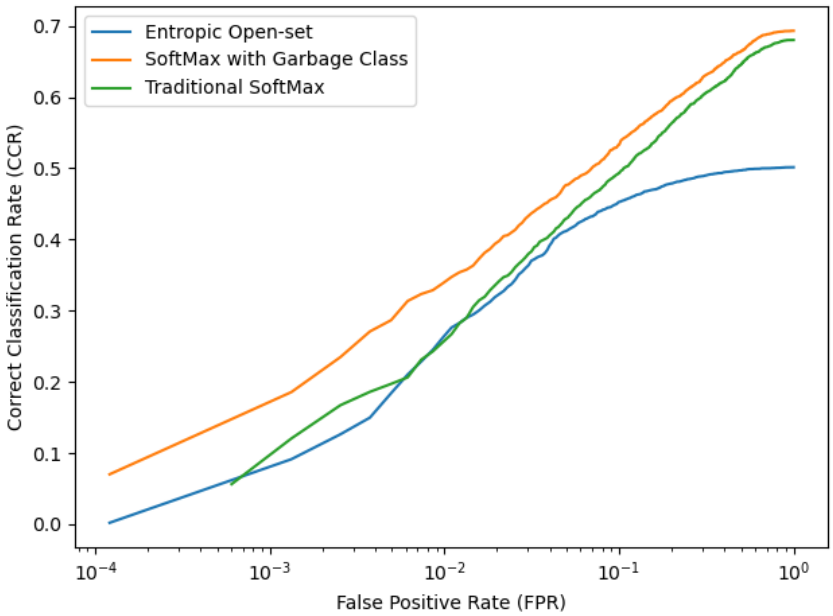
All in all, at FPR of 1, SoftMax with Garbage Class and Traditional SoftMax have higher CCR than Entropic Open-set Loss on all the protocols. Between FPR of 1 and the last plotted point of the respective curves of the algorithms, SoftMax with Garbage Class has higher CCR than the others, except for FPR values less than 0.1 for known unknowns of Protocol 1, where Entropic Open-set Loss performs better and some FPR values close to and greater than 0.01 for Protocol 2, where Entropic Open-set Loss and SoftMax with Garbage class perform similarly. OSCR curve for Entropic Open-set Loss extends more than the others for FPR values below ≈ 0.01, and this observation can be made more prominently for Protocols 2 and 3. OSCR curves for Entropic Open-set Loss are seen to extend completely to the y-axis for known unknowns for all the three protocols. A comparison of OSCR curves across all the protocols when the network is trained using Entropic Open-set Loss and SoftMax with Garbage Class is provided in Figures A.1 and A.2 in Appendix A.

Protocol	Epochs	Entropic Open-set Loss		SoftMax with Garbage Class		Traditional SoftMax	
		Avg. Val. Conf.	Avg. Test Conf.	Avg. Val. Conf.	Avg. Test Conf.	Avg. Val. Conf.	Avg. Test Conf.
Protocol 1	25	0.38	0.66	0.52	0.61	0.51	0.64
	50	0.40	0.66	0.49	0.49	0.56	0.51
	75	0.42	0.67	0.49	0.49	0.58	0.47
	100	0.45	0.69	0.49	0.47	0.59	0.46
	125	0.48	0.71	-	-	-	-
	150	0.52	0.72	-	-	-	-
	175	0.53	0.73	-	-	-	-
	200	0.56	0.73	-	-	-	-
Protocol 2	25	0.49	0.70	0.45	0.45	0.45	0.44
	50	0.50	0.68	0.41	0.35	0.56	0.36
	75	0.51	0.67	0.40	0.33	0.58	0.34
	100	0.52	0.65	0.39	0.32	0.61	0.33
	125	0.53	0.65	-	-	-	-
	150	0.52	0.65	-	-	-	-
	175	0.52	0.64	-	-	-	-
	200	0.52	0.66	-	-	-	-
Protocol 3	25	0.46	0.60	0.51	0.49	0.65	0.48
	50	0.48	0.60	0.50	0.45	0.70	0.44
	75	0.49	0.61	0.50	0.45	0.71	0.43
	100	0.51	0.62	0.50	0.44	0.72	0.42
	125	0.53	0.63	-	-	-	-
	150	0.55	0.64	-	-	-	-
	175	0.56	0.65	-	-	-	-
	200	0.58	0.66	-	-	-	-

Table 6.2: AVERAGE CONFIDENCES. This table shows Average Validation Confidence (Avg. Val. Conf.) and Average Test Confidences (Avg. Test Conf.) obtained after every 25^{th} epoch upon training the network using different algorithms for all three ImageNet protocols. Entropic Open-set Loss is trained for 200 epochs while the others are trained for 100 epochs.

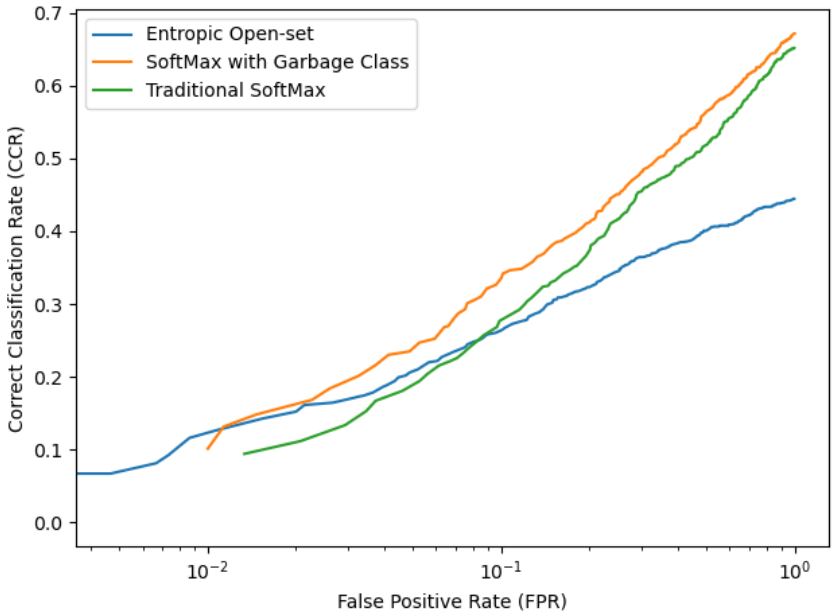


(a) OSCR curves for Protocol 1 - Known Unknown

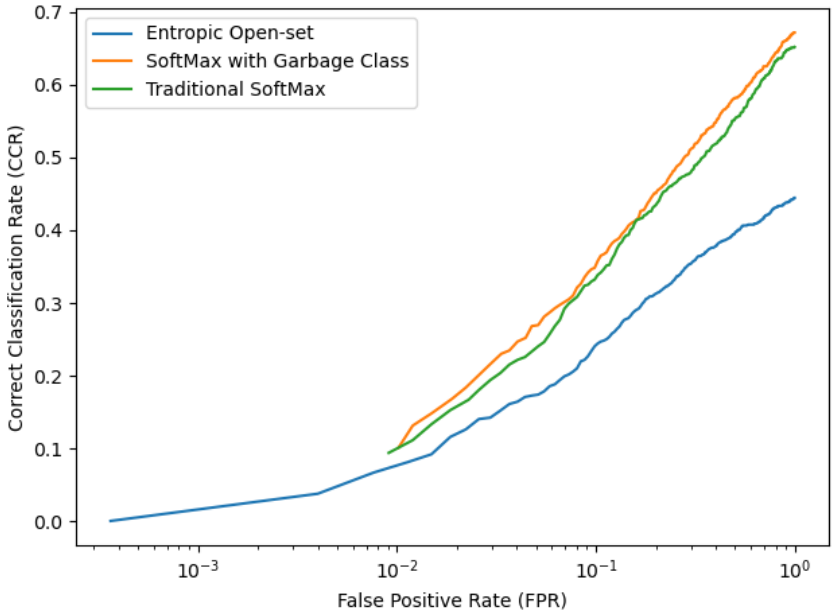


(b) OSCR curves for Protocol 1 - Unknown Unknown

Figure 6.1: OSCR CURVES FOR PROTOCOL 1. This figure shows Open-Set Classification Rate curves when the network is trained using Entropic Open-set Loss, SoftMax with Garbage Class and Traditional SoftMax on Protocol 1.

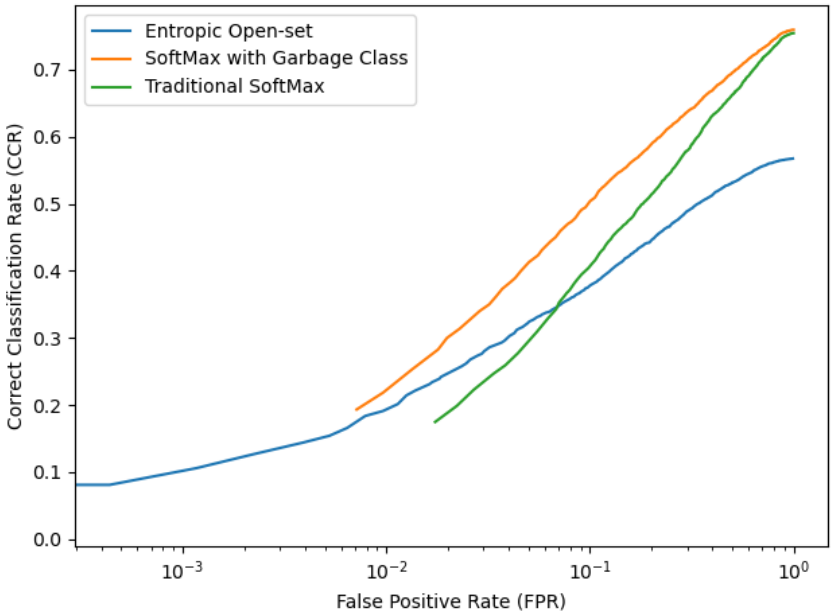


(a) OSCR curves for Protocol 2 - Known Unknown

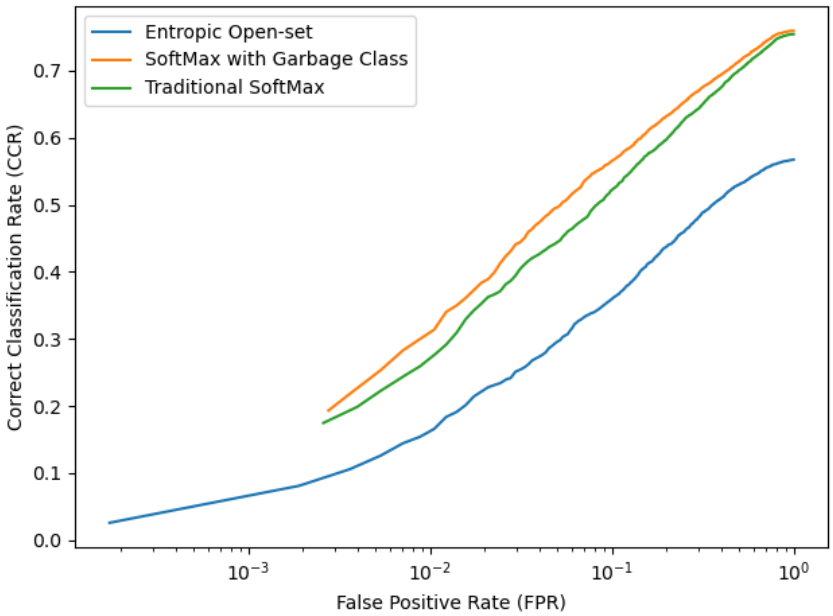


(b) OSCR curves for Protocol 2 - Unknown Unknown

Figure 6.2: OSCR CURVES FOR PROTOCOL 2. This figure shows Open-Set Classification Rate curves when the network is trained using Entropic Open-set Loss, SoftMax with Garbage Class and Traditional SoftMax on Protocol 2.



(a) OSCR curves for Protocol 3 - Known Unknown



(b) OSCR curves for Protocol 3 - Unknown Unknown

Figure 6.3: OSCR CURVES FOR PROTOCOL 3. This figure shows Open-Set Classification Rate curves when the network is trained using Entropic Open-set Loss, SoftMax with Garbage Class and Traditional SoftMax on Protocol 3.

Chapter 7

Discussion

In this thesis, I compare the performance of Entropic Open-set Loss and SoftMax with Garbage class with a baseline Traditional SoftMax on three open-set ImageNet-based ILSVRC2012 protocols, designed by me, to understand how effectively the selected open-set algorithms satisfy the two-fold open-set goal of correctly classifying known samples and correctly rejecting unknown samples. After training a ResNet-50 network using these algorithms, I compare their performance using Average Confidence and Open-Set Classification Rate (OSCR) curves, for all the three protocols, the results of which I show in section 6.4. In this chapter, I draw insights using my observations from these results and discuss potential limitations.

In general, both Entropic Open-set Loss and SoftMax with Garbage Class did not perform at par on open-set partitions of ImageNet as compared to open-set partitions created using MNIST, NIST letters and Devanagari classes in terms of OSCR curves as seen from Figure 7.1 (Dhamija et al., 2018) and OSCR curves shown in section 6.4. Significantly higher Correct Classification Rate (CCR) is obtained by these algorithms at different values of False Positive Rate (FPR) for the latter. Further, OSCR curves are seen to extend completely, till the y-axis, for the two algorithms and this is observed even for Traditional SoftMax, in case of the latter. The poor performance on open-set ImageNet protocols can be attributed to the relatively large number of classes in ImageNet, hierarchical nature of the dataset and high resolution of the images, when compared to the small datasets Dhamija et al. (2018) experimented with. Further, varying levels of difficulty in terms of correctly classifying similar known classes and correctly detecting unknown unknown classes that are similar to known classes makes open-set classification on the ImageNet-based protocols even more difficult. Even though the performance is not at par, just seeing how the selected open-set algorithms, developed initially using small datasets, pan out on a large, hierarchical dataset, provides a first step in the direction of improving these algorithms to effectively satisfy the two-fold open-set goal on real-world-like open-set protocols.

Among the three open-set ImageNet protocols, open-set algorithms are seen to perform the best on Protocol 1, followed by Protocol 3 and then Protocol 2, in terms of both Average Confidence and OSCR curves. I attribute this order of performance to the way the three protocols are designed. Due to lesser similarity between known and unknown unknown classes in Protocols 1 and 3, as compared to that in Protocol 2 (as mentioned in chapter 4), it might be easier for the algorithms to identify unknown unknown classes of those protocols, thus making Protocol 2 to be the most difficult protocol. An improvement that can be made while designing Protocol 3 is adding some subclasses of the known ancestors to the unknown unknown partition, in addition to classes of other ancestors. This would lead to having both unknown unknown classes that are very similar to known classes as well as classes that are not so similar to known classes, thus making it more difficult than Protocol 2.

Among the two open-set algorithms, as per Average Confidence value, Entropic Open-set Loss performs better than SoftMax with Garbage Class, once the network is completely trained. How-

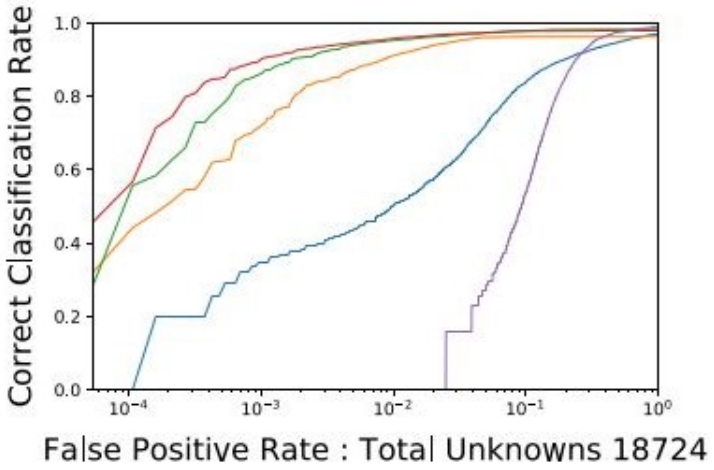


Figure 7.1: PERFORMANCE ON MNIST, NIST LETTERS AND DEVANAGARI CLASSES. This figure shows OSCR curves when a network is trained using Entropic Open-set Loss (green curve), SoftMax with Garbage Class (orange curve) and Traditional SoftMax (blue curve) on 10 known MNIST classes, known unknown NIST letters and unknown unknown Devanagari classes. *Source: (Dhamija et al., 2018)*

ever, in terms of OSCR curves, for unknown unknowns, SoftMax with Garbage Class performs better than Entropic Open-set Loss for all False Positive Rate (FPR) values for Protocol 1 and at high FPR values for Protocols 2 and 3. This performance can be attributed to the ability of SoftMax with Garbage Class to better classify known samples and not confuse them with unknown unknown samples. At low FPR values, the OSCR curve for SoftMax with Garbage Class does not extend as much as it does for Entropic Open-set Loss, as seen for Protocols 2 and 3. I suspect that this might be due to the fact that, for SoftMax with Garbage Class, the calculated confidence thresholds for low FPR values (different confidence thresholds are calculated for each algorithm for a given value of FPR) is 1, due to which CCR cannot be computed for those values (since there might be practically zero known samples that would get correctly classified with a predicted confidence ≥ 1). Entropic Open-set Loss may not be as effective as SoftMax with Garbage Class in correctly classifying known samples, but it keeps effectively rejecting unknown unknown samples at all FPR values. This can be attributed to its ability of reducing the confidence to very low value for unknown unknown samples.

One way of improving the performance of the two open-set algorithms is to train the network using these algorithms for more number of epochs. Hence, I train the network using Entropic Open-set Loss for 100, 200 and 500 epochs (as it takes more epochs to better classify known samples using this algorithm) and train it using SoftMax with Garbage Class for 100 and 200 epochs, for Protocol 2 (as training on this protocol takes the least amount of time) to check if there is any significant improvement in performance. OSCR curves after training the network for the respective number of epochs can be seen in Figures 7.2 and 7.3. As can be seen, the performance of both the algorithms improves only slightly, especially on unknown unknowns. Using Entropic Open-set Loss, training for more epochs slightly improves its performance at low FPR values, while for SoftMax with Garbage Class, it improves slightly at high FPR values.

Another way of improving the performance of Entropic Open-set Loss would be to add class weights, similar to that done in case of SoftMax with Garbage Class, or to modify the loss to an Objectosphere loss as shown by Dhamija et al. (2018). An Objectosphere loss creates an objectosphere and penalizes known samples if their feature magnitude is inside this objectosphere and penalizes unknown samples if their feature magnitude is greater than zero. OSCR curve

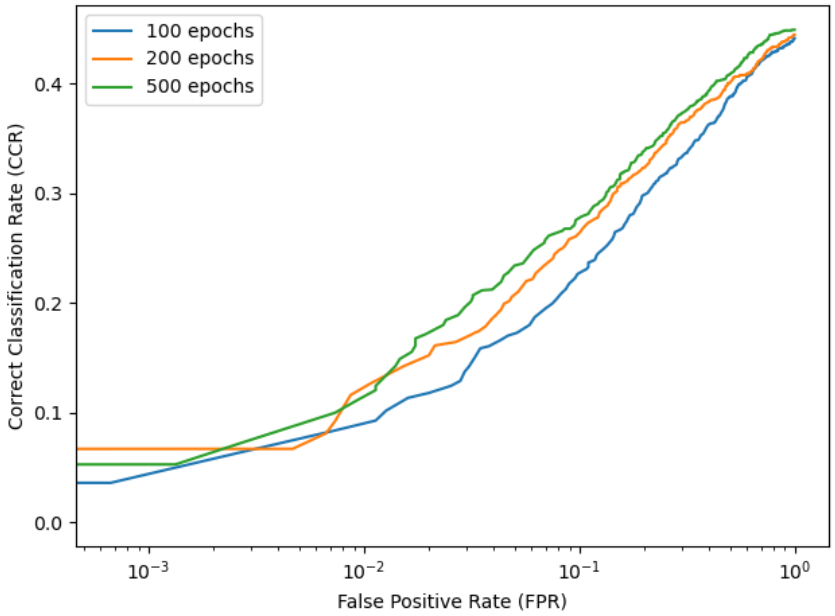
produced when the network is trained with Objectosphere loss on 10 known MNIST classes and known unknown NIST letters, can be seen in Figure 7.1, as the red curve. This improvement can be experimented with in future work.

When compared with OSCR curve for Traditional SoftMax, SoftMax with Garbage Class performs slightly better than it in terms of obtaining higher CCR. Entropic Open-set Loss performs poorer than Traditional SoftMax at high FPR values, but its curve is seen to significantly extend more than that of Traditional SoftMax at low FPR values.

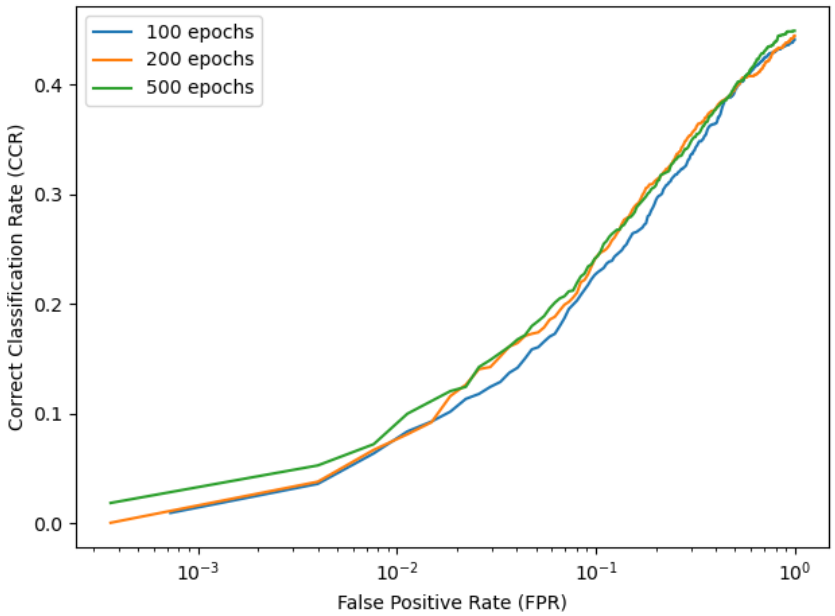
Limitations: Here, I highlight the potential limitations of my work.

- The experiments are conducted using a basic set-up, that is a fixed learning rate of 0.01, a momentum of 0.9 and SGD optimizer are used to train the network. It is possible that different experiments might require different hyperparameter setting to effectively train the network.
- It is possible that different network architectures are differently capable in detecting unknown samples (Roady et al., 2020). For example, since Protocol 1 is comparatively easy in terms of detecting unknown samples, a ResNet-18 architecture might be sufficient for it, while for Protocol 2, a ResNet-101 might be more effective. For my experiments, I use a common ResNet-50 architecture, which might not produce the best results for all the protocols.
- Krizhevsky et al. (2012) suggest that for a rectangular image, first the shorter side is scaled down to a certain length (they use 256), and then a center crop is extracted from the scaled down image (they use a center crop of 256×256). I scale down the images to a size of 300×300 and then take a center crop of 224×224. This might impact the results of my experiments due to change in aspect ratio of the image and the objects contained.
- I use `CenterCrop` to extract a fixed center patch from the training images, where it is recommended to use `RandomCrop` to crop the images at a random location, for effective data augmentation. This might be causing some overfitting to the training data. Further, use of `RandomRotation` for augmenting the training data for my experiments might be causing an increase in training time.
- The protocols designed in this thesis provide only one way of evaluating the open-set algorithms, and thus the results might be limited to this design. It might be possible to design the protocols in a better way to produce better results, especially the design of known unknown classes used for training the network to learn to differentiate between known and unknown samples.
- While comparing two open-set algorithms, developed using small datasets, on ImageNet protocols makes sense, it does not help in understanding how they compare against an algorithm that is developed using ImageNet, like OpenMax (Bendale and Boult, 2016), on these protocols. I could not implement OpenMax due to time constraints.
- For OSCR curves, different confidence thresholds are computed for each algorithm at a given FPR value. Therefore, using a common confidence threshold of 0.3 might not be a good example for estimating the performance of the algorithms.

While my thesis might not be all-inclusive, I believe it provides a good starting point on how ImageNet can be used to design open-set protocols, and provides a good reference point for future researchers who are interested in diving deeper on improving the performance of Entropic Open-set Loss and SoftMax with Garbage Class. Concepts behind three other open-set algorithms, namely OpenMax, Extreme Value Machine and Placeholders for open-set recognition, are also discussed in details to ease the task of applying these algorithms on the open-set ImageNet protocols.

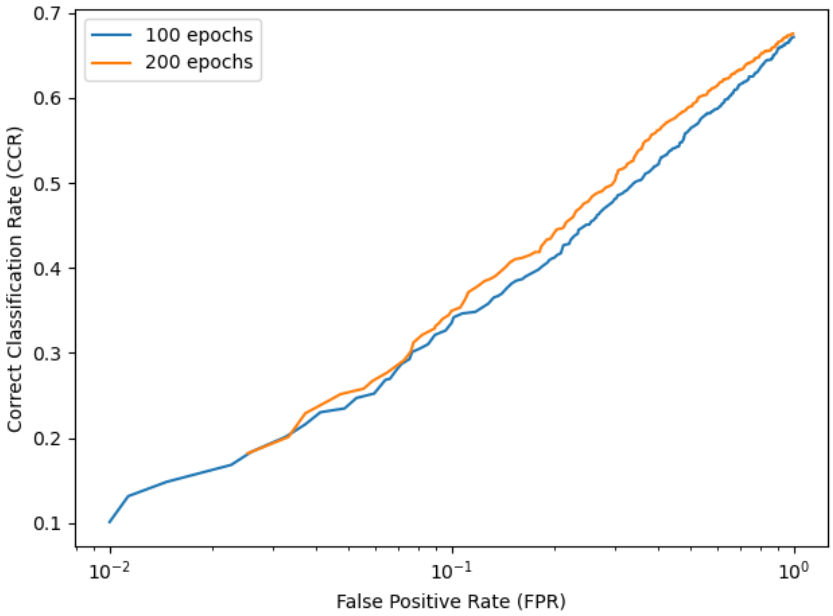


(a) Performance on Known Unknown classes when trained for 100, 200 and 500 epochs

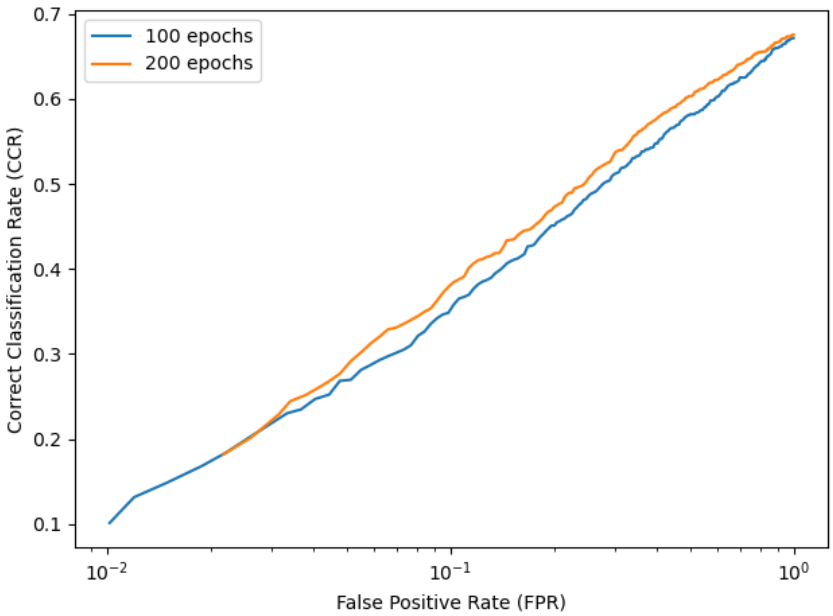


(b) Performance on Unknown Unknown classes when trained for 100, 200 and 500 epochs

Figure 7.2: PERFORMANCE OF ENTROPIC OPEN-SET LOSS ACROSS EPOCHS. In this figure, I compare the open-set performance of Entropic Open-set Loss when the network is trained for 100, 200 and 500 epochs on Protocol 2.



(a) Performance on Known Unknown classes when trained for 100 and 200 epochs



(b) Performance on Unknown Unknown classes when trained for 100 and 200 epochs

Figure 7.3: PERFORMANCE OF SOFTMAX WITH GARBAGE CLASS ACROSS EPOCHS. In this figure, I compare the open-set performance of SoftMax with Garbage Class when the network is trained for 100 and 200 epochs on Protocol 2.

Chapter 8

Conclusion and Future Work

Open-set classification focuses on a two-fold goal of correctly classifying known samples and correctly detecting unknown samples. In this thesis, I study the performance of two open-set algorithms - Entropic Open-set Loss and SoftMax with Garbage class, on three open-set ImageNet protocols, designed by me and compare their performance to a baseline Traditional SoftMax. I design the protocols for open-set partitions of known, known unknown and unknown unknown classes using the dataset from ILSVRC2012, and address varying levels of difficulty in correctly classifying known classes and correctly detecting unknown unknown samples that share some similarity with known classes. Never before have there been open-set partitions created using the same ImageNet dataset, and these protocols provide a first idea on how ImageNet hierarchy can be explored to closely replicate a real-world open test space in different ways. The two open-set algorithms, developed using small datasets, are selected due to their ease of implementation, and to see how effectively they pan out in terms of performance on the designed ImageNet protocols. Two novel evaluation metrics are used to evaluate the performance of these algorithms, namely Average Confidence and Open-Set Classification Rate (OSCR) curve. Average Confidence is mainly used to monitor the performance of the algorithms across epochs, while OSCR curve is used to evaluate the overall performance once the training is complete. Both the open-set algorithms perform almost equally bad on all the three ImageNet protocols, which suggests that the known unknown classes used for training the network might not be effective in making the network learn to differentiate between known and unknown samples. SoftMax with Garbage Class is seen to perform slightly better than Entropic Open-set Loss and Traditional SoftMax at high values of False Positive Rate (FPR), but OSCR curves for SoftMax with Garbage Class and Traditional SoftMax do not extend as much to the left of the OSCR plots as it does for Entropic Open-set Loss, that is for lower FPR values. Among the protocols, the algorithms perform the best on Protocol 1, as compared to Protocols 2 and 3, suggesting that they find it easier to detect unknown unknown classes that do not share similarity with known classes. All the algorithms find it almost equally difficult to correctly classify known classes above a confidence threshold for all FPR values, for all the protocols. I also show that training the network for more number of epochs using the open-set algorithms does not significantly improve their performance. These conclusions are based on the experimental set-up and protocol design used for my experiments and might be prone to certain limitations, which I also highlight in chapter 7.

Future Work: While the groundwork for open-set classification on ImageNet protocols is laid, there are certain gaps that can be filled through future work. These recommendations are provided using insights drawn in chapter 7.

- It would be helpful if the known unknown classes could be better designed to improve the performance on the protocols. Additionally, modifying unknown unknown classes of Protocol 3, to have varying levels of similarity between these classes and known classes,

would intensify the difficulty level of this protocol, and would be interesting to evaluate using the open-set algorithms.

- It would be beneficial to compare the performance of the selected open-set algorithms, developed using small datasets, with an open-set algorithm developed using ImageNet, on the designed open-set ImageNet protocols.
- It would be interesting to know how the training improves if the aspect ratio of images is maintained and better data augmentation techniques are used.
- Training using Entropic Open-set loss can be specifically improved by introducing class weights or modifying the loss to better differentiate known and unknown samples.
- The training time for algorithms could be reduced through different hyperparameter setting for different algorithms, use of different network architectures for different protocols and through efficient code optimization techniques. It can be further reduced by removing inefficient data augmentation techniques.
- Finally, while creating the protocols, I sometimes manually study the hierarchy between classes (that is the tree structure) before selecting them using the `robustness` library. It would be interesting to know if it is possible to make this process less manual.

Appendix A

Attachments

Table A.1 shows the Superclass names and their respective Class IDs (in brackets) used to design the open-set ImageNet protocols for this thesis.

Open-set Partition	Protocol 1	Protocol 2	Protocol 3
Known classes	Dog(n02084071)	Some descendants of Hunting dog(n02087122)	Some descendants of Dog(n02084071), Bird(n01503061), Insect(n02159955), Furniture(n03405725), Fish(n02512053), Monkey(n02484322), Car(n02958343), Cat(n02120997), Truck(n04490091), Fruit(n13134947), Fungus(n12992868), Boat(n02858304), Computer(n03082979)
Known Unknown classes	Fox(n02118333), Wild dog(n02115335), Wolf(n02114100), Feline(n02120997), Bear(n02131653), Musteline mammal(n02441326), Ungulate(n02370806), Primate(n02469914)	Other descendants of Hunting dog(n02087122)	Other descendants of Dog(n02084071), Bird(n01503061), Insect(n02159955), Furniture(n03405725), Fish(n02512053), Monkey(n02484322), Car(n02958343), Cat(n02120997), Truck(n04490091), Fruit(n13134947), Fungus(n12992868), Boat(n02858304), Computer(n03082979)
Unknown Unknown classes	Food(n07555863), Motor vehicle(n03791235), Device(n03183080)	Toy dog(n02085374), Fox(n02118333), Wild dog(n02115335), Wolf(n02114100), Feline(n02120997), Bear(n02131653), Musteline mammal(n02441326), Ungulate(n02370806)	Reptile(n01661091), Clothing(n03051540), Ungulate(n02370806), Vegetable(n07707451), Aircraft(n02686568)

Table A.1: IMAGENET SUPERCLASS IDS USED FOR CREATING PROTOCOLS. This table shows the ImageNet superclasses, along with their class IDs that were used to create the protocols.

Tables A.2, A.3 and A.4 show performance of Entropic Open-set Loss, SoftMax with Garbage Class and Traditional SoftMax in terms of percentage of correctly classified known samples and correctly rejected unknown samples (known unknown and unknown unknown) from Protocols 1, 2 and 3 respectively, at a confidence threshold of 0.3, during validation and testing, after every 25 epochs of training on the training set.

Algorithm	**Epochs**	**% of Correctly Classified Known Validation samples**	**% of Correctly Rejected Known Unknown Validation samples**	**% of Correctly Classified Known Test samples**	**% of Correctly Rejected Known Unknown Test samples**	**% of Correctly Rejected Unknown Unknown Test samples**
Entropic Open-set Loss	25	0.56	99.85	0.55	99.85	99.59
	50	5.99	98.07	6.47	97.85	98.18
	75	9.33	97.27	11.24	97.01	97.22
	100	16.47	98.05	20.86	98.36	97.63
	125	22.76	98.14	29.26	97.91	97.02
	150	30.14	96.83	38.02	96.66	94.96
	175	33.19	96.77	40.88	96.66	94.13
	200	37.07	96.37	44.57	96.63	91.05
SoftMax with Garbage class	25	58.66	19.80	64.90	17.37	56.98
	50	60.93	6.07	67.43	5.37	17.53
	75	62.16	4.49	68.66	4.63	15.66
	100	62.67	3.81	69.29	3.31	10.86
Traditional SoftMax	25	55.27	-	61.64	28.99	69.41
	50	58.55	-	64.55	7.04	22.18
	75	59.975	-	66.84	5.49	13.28
	100	60.44	-	67.97	4.90	10.17

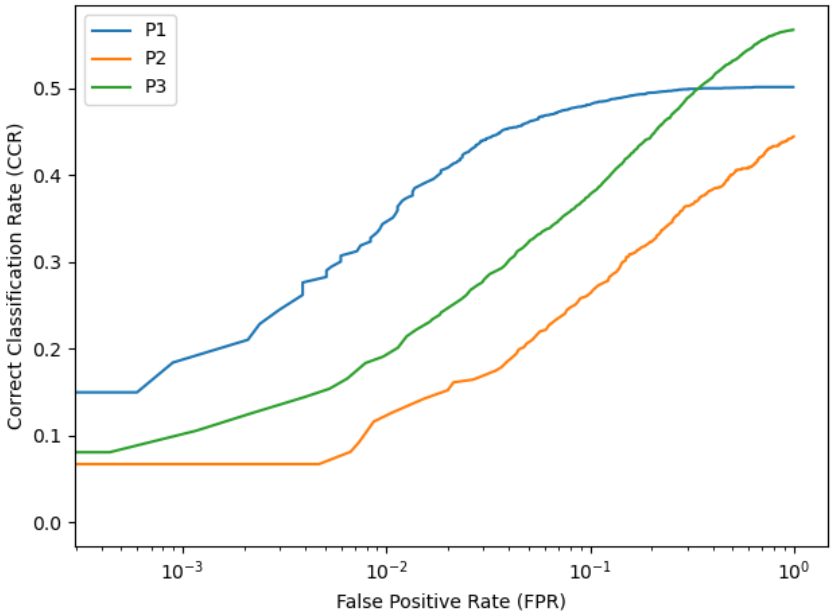
Table A.2: PERFORMANCE ON PROTOCOL 1 AT 0.3 THRESHOLD. This table shows the performance of different algorithms on Protocol 1 at a confidence threshold of 0.3 after every 25^{th} epoch of training on the training set.

Algorithm	Epochs	% of Correctly Classified Known Validation samples	% of Correctly Rejected Known Unknown Validation samples	% of Correctly Classified Known Test samples	% of Correctly Rejected Known Unknown Test samples	% of Correctly Rejected Unknown Unknown Test samples
Entropic Open-set Loss	25	3.71	96.54	4.65	97.87	97.45
	50	13.78	88.83	14.52	90.27	91.96
	75	20.31	82.28	22.26	85.53	85.38
	100	27.75	73.73	29.10	75.87	77.85
	125	32.49	71.73	33.94	71.93	70.73
	150	29.54	72.89	32.45	72.00	72.51
	175	32.96	68.42	36.39	68.60	68.04
	200	28.55	74.74	33.94	76.73	73.42
SoftMax with Garbage class	25	55.31	9.55	61.55	8.33	16.11
	50	61.11	1.32	64.77	0.67	2.04
	75	62.37	0.97	65.61	1.07	1.24
	100	61.40	0.77	67.16	0.40	1.05
Traditional SoftMax	25	49.43	-	53.29	6.73	12.73
	50	58.23	-	61.81	1.07	1.93
	75	59.74	-	64.06	0.60	1.31
	100	61.68	-	65.10	0.67	0.95

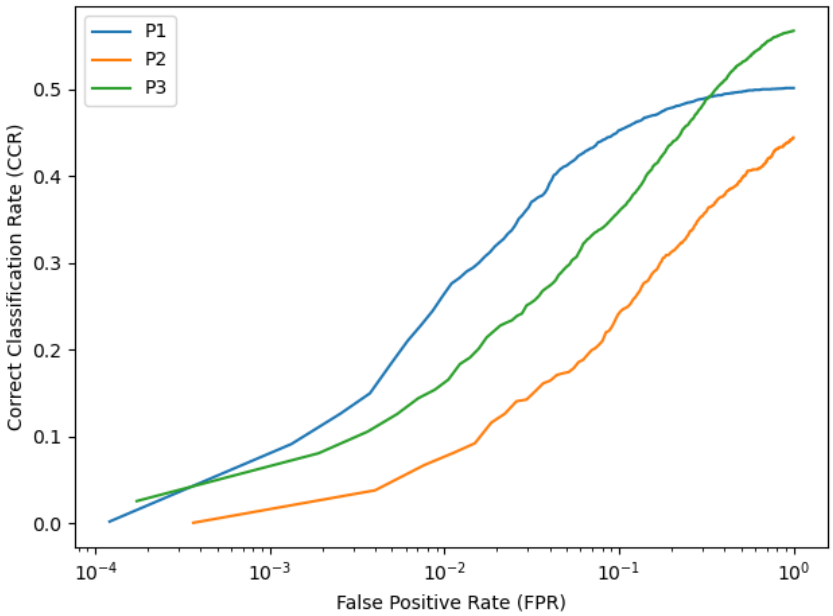
Table A.3: PERFORMANCE ON PROTOCOL 2 AT 0.3 THRESHOLD. This table shows the performance of different algorithms on Protocol 2 at a confidence threshold of 0.3 after every 25^{th} epoch of training on the training set.

Algorithm	Epochs	% of Correctly Classified Known Validation samples	% of Correctly Rejected Known Unknown Validation samples	% of Correctly Classified Known Test samples	% of Correctly Rejected Known Unknown Test samples	% of Correctly Rejected Unknown Unknown Test samples
Entropic Open-set Loss	25	3.06	98.12	2.97	98.10	97.12
	50	8.89	95.80	8.67	95.94	94.95
	75	15.17	93.54	15.86	93.43	93.40
	100	20.78	91.66	22.25	91.68	91.72
	125	26.35	89.79	28.35	89.28	89.48
	150	31.01	88.27	33.13	87.97	87.26
	175	35.23	87.32	37.92	86.53	85.03
	200	38.47	86.47	41.25	85.77	84.14
SoftMax with Garbage class	25	70.56	6.58	73.28	6.01	14.55
	50	72.66	2.93	75.12	2.58	6.47
	75	73.69	2.57	75.89	2.34	4.78
	100	74.10	2.19	75.95	1.88	4.03
Traditional SoftMax	25	68.12	-	70.57	6.23	14.05
	50	71.08	-	73.07	1.82	4.93
	75	72.26	-	74.26	1.30	3.47
	100	72.80	-	75.43	1.24	2.76

Table A.4: PERFORMANCE ON PROTOCOL 3 AT 0.3 THRESHOLD. This table shows the performance of different algorithms on Protocol 3 at a confidence threshold of 0.3 after every 25^{th} epoch of training on the training set.

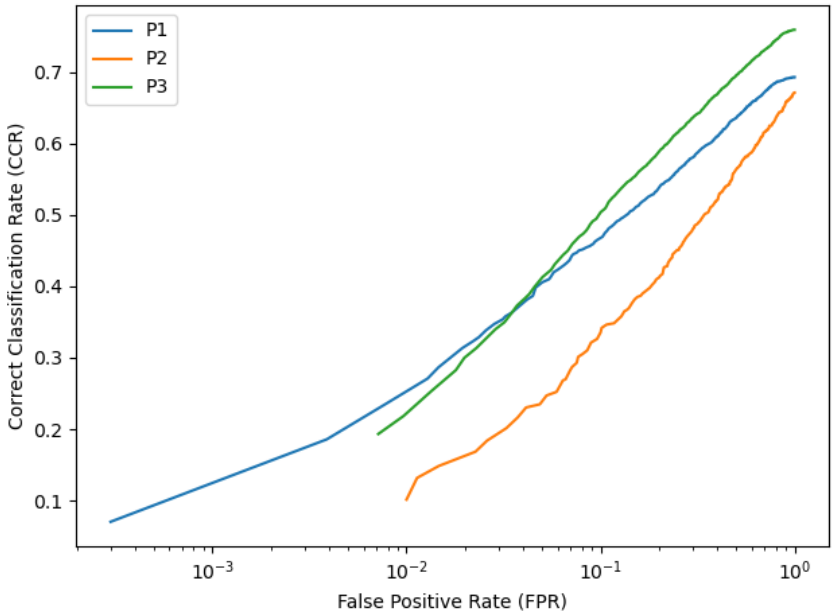


(a) Known Unknown

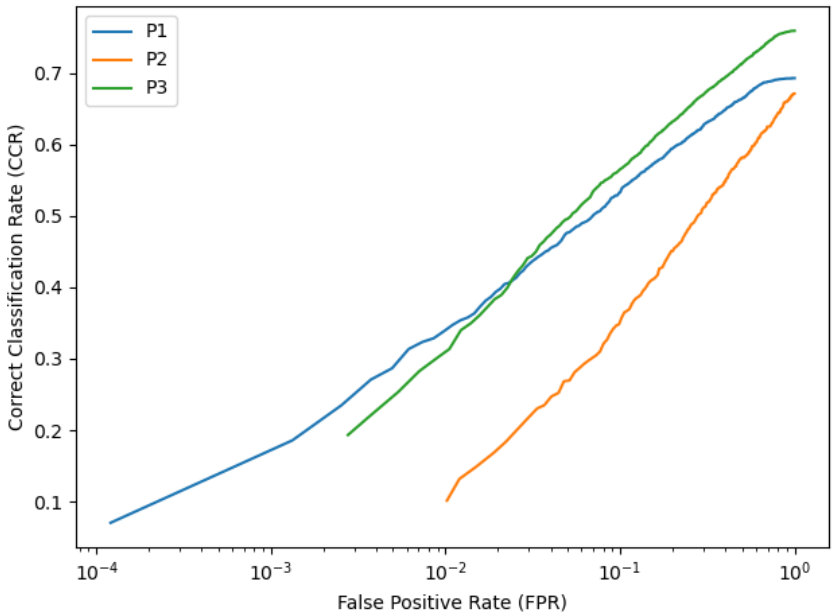


(b) Unknown Unknown

Figure A.1: OSCR CURVES FOR ENTROPIC OPEN-SET LOSS ON ALL PROTOCOLS. This figure shows Open-Set Classification Rate curves across the three protocols when the network is trained using Entropic Open-set Loss.



(a) Known Unknown



(b) Unknown Unknown

Figure A.2: OSCR CURVES FOR SOFTMAX WITH GARBAGE CLASS ON ALL PROTOCOLS. This figure shows Open-Set Classification Rate curves across the three protocols when the network is trained using SoftMax with Garbage Class.

List of Figures

List of Tables

Bibliography

Bendale, A. and Boult, T. (2015). Towards Open World Recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1893–1902.

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.

Bianco, S., Cadene, R., Celona, L., and Napoletano, P. (2018). Benchmark Analysis of Representative Deep Neural Network Architectures. *IEEE Access*, 6:64270–64277.

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

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926.

Deng, J., Dong, W., Socher, R., Li, L., 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.

Dhamija, A. R., Günther, M., and Boult, T. (2018). Reducing Network Agnostophobia. In *Advances in Neural Information Processing Systems*, volume 31.

Geng, C., Huang, S. J., and Chen, S. (2020). Recent Advances in Open Set Recognition: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*, volume 1. MIT Press, Cambridge, MA, USA.

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

Hsu, Y.-C., Shen, Y., Jin, H., and Kira, Z. (2020). Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern (CVPR)*, pages 10948–10957.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *32nd International Conference on Machine Learning (PMLR)*, volume 37, pages 448–456.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25.

Lee, K., Lee, K., Lee, H., and Shin, J. (2018). A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems*, volume 31.

Leong, M., Prasad, D., Lee, Y. T., and Lin, F. (2020). Semi-CNN Architecture for Effective Spatio-Temporal Learning in Action . In *Applied Sciences*, volume 10, page 557.

Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the Reliability of Out-of-distribution Image Detection in Neural Networks. In *International Conference on Learning Representations*.

Liu, Y., Gao, Y., and Yin, W. (2020). An improved analysis of stochastic gradient descent with momentum. In *Advances in Neural Information Processing Systems*, volume 33, pages 18261–18271.

Margolis, I. (2021). Data Augmentation for Improved Classification in Neural Networks. Master's thesis, University of Zurich.

Matan, O., Kiang, R., Stenard, C., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., and Cur, Y. (2001). Handwritten Character Recognition Using Neural Network Architectures. In *1990 USPS Advanced Technology Conference*.

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2013). Distance-Based Image Classification: Generalizing to New Classes at Near-Zero Cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern (CVPR)*, pages 427–436.

Pant, A. K., Panday, S. P., and Joshi, S. R. (2012). Off-line Nepali handwritten character using Multilayer Perceptron and Radial Basis Function neural networks. In *2012 Third Asian Himalayas International Conference on Internet*, pages 1–5.

Roady, R., Hayes, T. L., Kemker, R., Gonzales, A., and Kanan, C. (2020). Are open set classification methods effective on large-scale datasets? *PLOS ONE*, 15(9).

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

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Fei-Fei, L. (2014). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115.

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

Scheirer, W. J., Rocha, A., Micheals, R. J., and Boult, T. E. (2011). Meta-: The Theory and Practice of Score Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1689–1695.

Schnyder, J. (2021). Deep Adversarial Training for Teaching Networks to Reject Unknown Inputs. Bachelor's thesis, University of Zurich.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image . In *3rd International Conference on Learning Representations*.

Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. (2018). The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420.

Zhou, D.-W., Ye, H.-J., and Zhan, D.-C. (2021). Learning Placeholders for Open-Set . In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern (CVPR)*, pages 4401–4410.