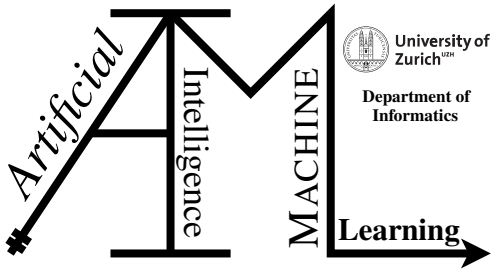
Suppressing the Extraction of Features from Unknown Samples

Bachelor Thesis

Lars Bösch
18-921-981

Submitted on
December 1 2023

Thesis Supervisor
Prof. Dr. Manuel Günther



Bachelor Thesis

Author: Lars Bösch, lars.boesch@uzh.ch

Project period: June 1 2023 - December 1 2023

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

Acknowledgements

I would like to extend my sincere gratitude to Prof. Dr. Manuel Günther for granting me the opportunity to write my thesis under his guidance, sharing his expertise on the subject matter, and providing valuable advice on the direction of my work. I am also thankful for his support in facilitating access to the required computing resources. Furthermore, I wish to extend my thanks to my friends, with a particular note of appreciation for Chris and Gilles.

Abstract

This thesis addresses the limitation of closed-set classification algorithms, designed for predefined classes, in real-world scenarios where unknown samples may be encountered. Building on prior work by Dhamija et al. (2018), who introduced the entropic open set loss and objectosphere loss as strategies to mitigate the extraction of features from unknown samples with the underlying idea that if a network refrains from extracting these features, it becomes more manageable and effective to apply a threshold to the outputted probabilities, our objective is to enhance the robustness of convolutional neural networks (CNNs) in identifying unknown samples while maintaining closed-set classification performance. In pursuit of this objective, we aim to investigate whether we can further hinder the network's learning process of extracting features from unknown samples. We attempt this by removing the deep-feature layer, extracting logits directly from the convolutional features and further impose non-negativity on the convolutional features and enforce the use of positive weights in the subsequent layer with two distinct approaches. We hope that this modification allows us to exert better control over these features. Additionally, we investigate the integration of these networks with an out-of-distribution classification network. Our experiments reveal that the altered networks exhibit comparable performance to the original topology but may not inherently improve open-set classification capabilities. While the suggested combination strategies yield promising outcomes upon evaluation using open-set classification rate curves, it is noteworthy that, despite some of the explored approaches effectively suppressing feature extraction for unknown samples, we do not achieve more favorable outcomes with networks extracting lower magnitudes for unknown samples compared to the magnitudes of known samples.

Contents

Chapter 1

Introduction

In the past, numerous algorithms have been designed and refined for closed set classification, assuming a finite and predefined set of classes. These networks have demonstrated highly impressive and promising results in the domain of image recognition (Huang et al., 2017) and, for instance, even in the field of object detection (Girshick, 2015). However, as Scheirer et al. (2013) acknowledged, this closed world assumption does not align well with many real-world use cases and should be addressed. The reason is that the world is incredibly diverse, potentially encompassing an almost infinite number of classes. Attempting to train a machine learning network to recognize and categorize all these possible classes is not only impractical but almost infeasible. This limitation poses a significant challenge for traditional closed set classification methods when confronted with scenarios where the class distribution is open, and new classes can emerge. As a result, open set classification has gained prominence as an alternative approach to address this issue by allowing models to handle uncertainty and acknowledge the existence of unknown or unseen classes.

1.1 Problem formulation

In this thesis, our primary objective is to enhance the robustness of convolutional neural networks in distinguishing unknown samples, all the while ensuring that their performance in closed-set classification remains unaffected. Our research builds upon the foundation laid by Dhamija et al. (2018), who introduced several novel contributions, including two distinct loss functions: the entropic open set loss and the objectosphere loss. These innovative loss functions are specifically designed to mitigate the extraction of features from images of unknown samples. At that time, an aspect which has not been investigated further was the characteristics of the fully connected layer responsible for extracting deep features from the convolutional features. As the rationale behind these loss functions centers on inhibiting the extraction of features from unknown images, a potential concern arises because the convolutional layers maintain the ability to learn to extract features from unknown samples, but the fully connected layer can aggregate these values in a way that the deep feature layer might still produce a zero vector. This deep feature zero vector would incorrectly imply that no features have been extracted. But given that the objective is to prevent the convolutional layer from learning features for unknown samples, this outcome is not aligned with our desired goal.

The intuition behind the underlying problem is that the network might learn to associate the presence of, for example, two specific learned features, suggesting that the input is an unknown sample. Nonetheless, as the network needs to subsequently identify various new unknowns, it cannot be assumed that these particular features will consistently co-occur. Hence, the preferable strategy is to prevent the network from learning these features in the initial stages.

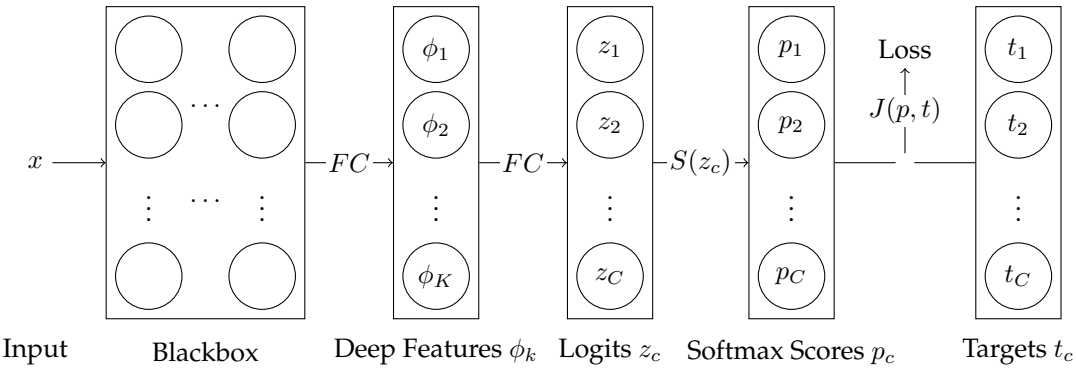


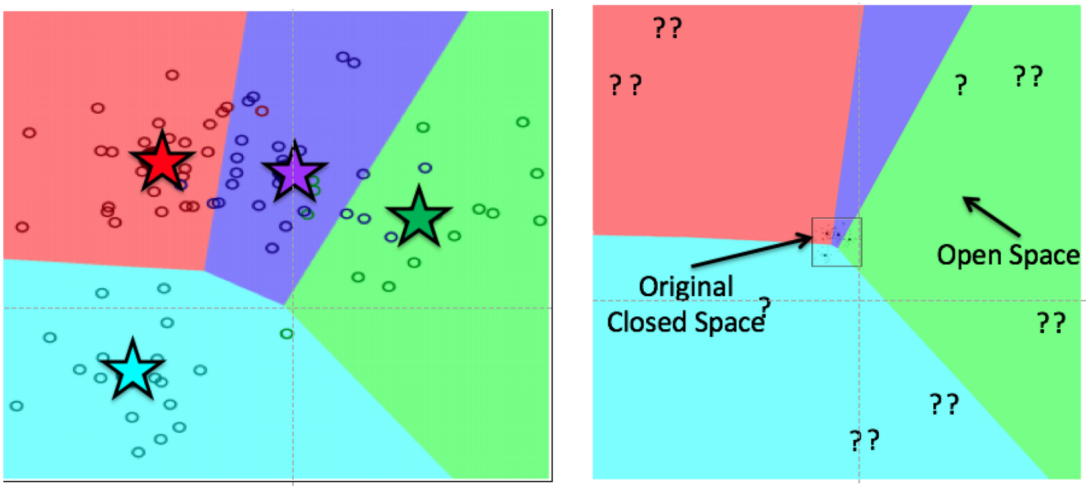
Figure 1.1: CONVOLUTIONAL NEURAL NETWORK OVERVIEW. *In this figure, a schematic representation of a convolutional neural network is depicted. Inside the black box layer, there exist N convolutional layers typically integrated with MaxPooling layers. Subsequently, a flattening operation is applied, ultimately producing the convolutional features as the concluding output. These values are then transformed into deep features ϕ_k through a fully connected layer. Another fully connected layer subsequently generates the logits z_c from the deep features. The SoftMax activation, $S(z)$, is then used to compute the SoftMax scores p_c. Together with the targets, the scores $\boldsymbol{p}$ are utilized to compute the loss that we use for optimization during the training.*

1.2 Our contribution

To tackle this issue, we consider eliminating the additional layer and directly extracting the logits from the convolutional features. Additionally, to eliminate the possibility of cancellation effects, we take the approach of not only ensuring non-negativity within the convolutional features but also enforcing the subsequent layer to exclusively employ positive weights. To address this, we explore two distinct approaches. In the first approach, we ensure that negative values on the weights are prevented by mapping any negative values to zero. In the second approach, we test the feasibility of using a penalty function for negative weights, with the aim of encouraging the network to exclusively train positive values.

We also attempt to apply this method to both a categorical open-set classification network, tasked with categorizing inputs into distinct known classes, and to an out-of-distribution binary classification network, designed to discern whether an input belongs to the known classes or is unknown. Finally, we explore the potential for combining these ideas. This leads us to the following research questions of this thesis:

- **RQ 1:** Is it possible for a network to achieve effective performance when it has been trained exclusively with positive deep-features and positive weights in the last layer?
  - **RQ 1a:** How does the performance of a categorical classification network with these adjustments compare to that of the original topology in a closed-set context?
  - **RQ 1b:** How does the performance of a categorical classification network with these adjustments compare to that of the original topology in a open-set context?
  - **RQ 1c:** How does the performance of a binary classification network with these adjustments compare to that of the original topology?
- **RQ 2:** Is it possible for a network to achieve effective performance when the deep-feature layer has been removed and it has been trained exclusively with positive features in the convolutional features and positive weights in the last layer?



(a) Closed Set (b) Open Set

Figure 1.2: CLOSED VS. OPEN SET. *(a) depicts the domain that the network identifies and anticipates the input to be assigned to, encompassing all inputs the network can allocate to a learned class. In contrast, (b) portrays the real-world scenario confronted by the network, where the learned closed set in the middle represents only a portion of the diverse inputs it may encounter. source: Boult et al. (2019)*

  - **RQ 2a:** In a closed-set context, how do these modifications affect the performance of a categorical classification network in comparison to the original topology?
  - **RQ 2b:** In an open-set context, how do these modifications affect the performance of a categorical classification network in comparison to the original topology?
  - **RQ 2c:** How do these modifications affect the performance of a binary classification network in comparison to the original topology?
- **RQ 3:** How can the out-of-distribution detection capability of binary classifiers be integrated into a categorical classification network?
- **RQ 4:** Did the methodology described in research questions 2 or 3 successfully prevent the network from extracting convolutional features from unknown samples?

Thesis Outline The thesis is structured as follows:
Chapter 2 provides an overview of open set classification methods, showing how different approaches have been used in the past and highlights the promising networks that have emerged from them. Chapter 3 goes into depth about the research that this thesis is based on and introduces the notation that is utilized. In Chapter 4, we elaborate on all the modifications made to the network and provide a comprehensive explanation of the procedures involved in running and evaluating our experiments. Additionally, Chapter 5 offers an overview of the dataset we utilized and illustrates how we are employing it in our research and provides all the information to ensure the reproducibility of the experiments. Chapter 6 unveils the experimental results, which are then thoroughly discussed in Chapter 7. To sum things up, Chapter 8 closes the thesis with a conclusion.

Chapter 2

Related Work

In this chapter, we emphasize the differences between traditional closed-set and open-set classification, and provide a comprehensive review of the relevant research efforts aimed at addressing this issue.

2.1 Open Set Classification

The open set classification problem, as outlined by Boult et al. (2019), is conceptualized in the following manner. By zooming out from the training data space, wherein all classes are known, and each input is assignable to a specific class, the concept of open space becomes evident. The open set denotes the territory situated directly around and up to far away from the training samples (see Figure 1.2). A traditional closed set classifier, when faced with these unknown inputs, has no other choice but to assign labels to them within the trained classes, even for points that are extremely distant.

Mahdavi and Carvalho (2021) identify two fundamental approaches to tackle open-set classification in the context of predicting probability distributions over known classes: the background/-garbage class approach and the thresholding SoftMax scores approach. We further elaborate on these methods in Section 2.1.1 and Section 2.1.2. In addition to these, various other strategies exist, including the OpenMax technique proposed by Bendale and Boult (2016), or the Uncertainty Estimation method introduced by Lakshminarayanan et al. (2017). Considering that this thesis is primarily built upon the framework that tries to predict probabilities, the main focus lies on the first two approaches, namely, the incorporation of a background/garbage class and the implementation of SoftMax score thresholds.

2.1.1 Background/Garbage Class

The utilization of the background or garbage class approach can be perceived as a basic extension of conventional closed-set networks, which offers a simple mechanism for these networks to classify unknown inputs accordingly. In contrast to the conventional SoftMax approach, which will further be elaborated on in Section 3.2, we incorporate an extra class, resulting in a total of C + 1 output classes. Furthermore, acquiring additional training data becomes imperative as we must provide the network with examples, called the negative classes, enabling it to discern their distinctions from the known ones. Although these negatives might not share direct similarities, they are assigned the same target label to ultimately be categorized under the background class. According to Dhamija et al. (2018), this strategy can prove highly effective, particularly when the negative classes adequately represent the entirety of the unknowns. While this holds true

for certain academic datasets like PASCAL (Everingham et al., 2010), its applicability in a real-world context may be less certain. Boult et al. (2019) noted that the problem with Background-class-based methodologies is that while they do capture certain aspects of the negatives, these approaches do not restrict the space to contain a finite number of positively labeled open spaces and are therefore not formally Open-Set recognition approaches.

2.1.2 Thresholding Softmax Scores

Another widely adopted technique to enable the network to label inputs as unknown involves assigning the probabilities generated by the SoftMax layer to one of the known classes only when the probability surpasses a predetermined threshold θ. Nevertheless, as Mahdavi and Carvalho (2021) discovered, establishing a fixed global threshold without any prior information about the potential unknown classes the network might encounter during testing poses significant challenges. As highlighted by Matan et al. (1990), a notable issue with this method is that the traditional SoftMax relies on the disparities in the logit values to determine the probabilities. This means, a class could potentially receive a high probability even with a low logit value, solely because the other classes have even lower values.

In addressing this challenge, Dhamija et al. (2018) suggested additional improvements to the existing method. The objective was to align the resultant probabilities more closely with a uniform distribution by incorporating negative samples during training. Subsequently, efforts were made to ensure that negative samples exhibited similar logit values across all classes and, in a subsequent stage, even induce small feature magnitudes for unknown inputs. These approaches facilitate improved thresholding capabilities and are known as the EOS and Objectosphere loss functions, elaborated upon extensively in Section 3.4.1 and Section 3.4.2.

2.2 Out of Distribution Detection

Given the absence of a consistent definition for out-of-distribution classification and its distinction from the open set classification task, we adopt the following definition: Open set recognition entails the multi-class classifier accurately classifying test samples from known classes while detecting test samples from unknown classes simultaneously. In contrast, out-of-distribution detection is primarily focused on recognizing or rejecting invalid inputs (Yang et al., 2022) (Gillert and von Lukas, 2021).

According to the findings of Cui and Wang (2022), various out-of-distribution approaches can be classified based on the underlying machine learning paradigm. These approaches fall into three main categories: supervised, semi-supervised, or unsupervised. Additionally, they can be further categorized based on technical means, such as model-based, distance-based, or density-based methods. Therefore, in our study, our primary emphasis will be on supervised model-based deep learning methods. Methods belonging to a similar category include anomaly detection (Chandola et al., 2009), novelty detection (Pimentel et al., 2014), and outlier detection (Hodge and Austin, 2004).

Out-of-distribution detection has proven successful in a range of applications, notably in medical image processing (Mårtensson et al., 2020) (Gao and Wu, 2020), intrusion detection, fraud detection, and system health monitoring, showing that this type of classification is also viable in security and risk-critical fields. But despite the positive outcomes, there are certain challenges associated with out-of-distribution detection using deep learning. While this approach performs effectively in low-dimensional space, out-of-distribution features are not easily discernible in high-dimensional space, and many existing methods primarily rely on point features. Addressing the

detection of conditional features and group features represents unresolved challenges in this context (Cui and Wang, 2022).

Chapter 3

Background

In this chapter, we familiarize ourselves with significant network topologies, loss functions, and the central concepts that form the basis of our proposed network, along with the introduction of relevant notation and terminology. We will begin by exploring the architecture of convolutional neural networks. Following this introduction, we will provide a succinct overview of the structure of the LeNet++ and the SmallScale-CNN topologies, before we will delve into the definition of the SoftMax and the Binary-Cross-Entropy loss. Finally, we will present two techniques designed to enhance its effectiveness in open-set classification: the Entropic Open-Set loss and the Objectosphere loss.

3.1 Convolutional Neural Networks

A convolutional neural network is a specialized deep learning model primarily designed to process and analyze visual data, such as images. In addition to the fully connected layers, they integrate various image-specific operation layers, including convolutional layers and pooling layers, to effectively capture features at different levels of abstraction. By convolving filters across the input image and progressively reducing the spatial dimensions, CNNs can detect patterns, shapes, and complex structures within the data, making them highly effective in tasks such as image classification. However, in this thesis, our primary emphasis is not on the intricacies of the convolutional layers, referred to later as the blackblox, but rather on the three following layers responsible for computing the deep features, the logits, and the SoftMax scores. As depicted in Figure 1.1 the initial CNN that served as the starting point employs a fully connected layer that takes the convolutional features, the final layer within the blackbox, and combines these values through a fully connected layer to yield the deep features. These represent the high-level abstract representations of the input data learned through the network's layers. These features capture complex patterns and distinctive characteristics within the data, enabling the network to differentiate between various classes. The last fully connected layer - connecting the deep features to the logits - is responsible for detecting the presence or absence of specific deep features and endeavors to identify patterns, thereby determining plausible combinations relevant to various classes.

In a fully connected layer, values are computed by combining inputs through a set of weights and biases. Each neuron in such a layer is linked to every neuron in the preceding layer. To determine the value of an output node, each input from the previous layer is multiplied by its associated weight w, signifying the connection strength between neurons. The products of these multiplications are aggregated by adding them up, yielding the output for that particular node. This process is then reiterated for each output node in the layer, incorporating distinct weights

Algorithm 1: LeNet++ Forward Function

Input: data x

$x \leftarrow \text{PReLU}(\text{MaxPool2d}(\text{BatchNorm2d}(\text{Conv2d2}(\text{Conv2d1}(x)))))$;

$x \leftarrow \text{PReLU}(\text{MaxPool2d}(\text{BatchNorm2d}(\text{Conv2d4}(\text{Conv2d3}(x)))))$;

$x \leftarrow \text{PReLU}(\text{MaxPool2d}(\text{BatchNorm2d}(\text{Conv2d6}(\text{Conv2d5}(x)))))$;

$x \leftarrow x.\text{view}(-1, \text{self.Conv2d6.out\_channels} \cdot 3 \cdot 3)$;

$df \leftarrow \text{Linear1}(x)$;

$logits \leftarrow \text{Linear2}(df)$;

return $logits, df$;

and biases for each node. Mathematically, this can be articulated as follows:

$$y_i = \sum_{j=1}^{J}(x_j \cdot w_{ij}) + b_i) \tag{3.1}$$

where y_i is an output node and $x_j, \forall j \in \{1, \ldots, J\}$ is the value of an input node. Alternatively, for simplicity and clarity, we can represent it in vector notation as:

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} \tag{3.2}$$

here $\mathbf{W}$ denotes the weight matrix, where each row encapsulates the connection strengths for an output node.

By employing a SoftMax activation, we can calculate probabilities from the logits, which, in conjunction with the ground truth labels, are instrumental in computing the loss for the provided input data (see Section 3.2). Utilizing the principles of backpropagation, we iteratively tweak the network's parameters to minimize the disparity between the predicted and actual target outputs for various inputs. This process aims to optimize the network's weights and biases, enhancing its capacity to precisely classify the input data (Skansi, 2018) (Krizhevsky et al., 2012).

3.1.1 LeNet++

In our pursuit to enhance the capabilities of the network originally presented by Dhamija et al. (2018), we have adopted the same foundational architecture as they did, namely the extended version of the original LeNet 5 architecture (Lecun et al., 1998) known as LeNet++.

This network comprises a total of six convolutional layers, which are subsequently followed by two fully connected layers, each equipped with learnable parameters. The convolutional layers have been enhanced through a combination of max pooling, batch normalization and ReLU activation techniques. This network is equipped with 2 deep-feature nodes. In our classical case, when tasked with categorizing the 10 MNIST classes, it uses 10 logit nodes and we denote the network as the categorical classification network. However, this configuration needs adjustment when the network is assigned the task of classifying a different dataset with a varying number of classes.

To conclude the architecture, we employ a SoftMax classifier, responsible for categorizing the input images into their respective classes.

Algorithm 2: SmallScale-CNN Forward Function

Input: data x

$x \leftarrow \text{MaxPool2d}(\text{ReLU}(\text{Conv2d1}(x)))$;
$x \leftarrow \text{MaxPool2d}(\text{ReLU}(\text{Conv2d2}(x)))$;
$x \leftarrow x.\text{view}(-1, \text{self.Conv2d2.out\_channels} \cdot 7 \cdot 7)$;
$df \leftarrow \text{Linear1}(x)$;
$logits \leftarrow \text{Linear2}(df)$;
return $logits, df$;

3.1.2 SmallScale Convolutional Neural Network

In contrast, the SmallScale-CNN architecture, a simpler variant, consists of two convolutional layers. These convolutional layers are enhanced with ReLU activation and max-pooling techniques. The two fully connected layers and the SoftMax activation function have remained unchanged from the previous LeNet++ configuration, except for the expansion of the deep features to 500.

3.2 SoftMax

In order to transform the raw output values generated by the logit layer into normalized probabilities suitable for the output layer, we employ the SoftMax function. Mathematically, the SoftMax function can be expressed as:

$$p_c = S_c(\mathbf{z}) = \frac{e^{z_c}}{\sum_{c' \in C} e^{z_{c'}}} \in (0, 1) \tag{3.3}$$

where z_c is the logit value of the class of interest and $c' \in \mathcal{C}$ are all the known classes. Through element-wise application to all classes in the set $\mathcal{C}$ we obtain the probability distribution $\mathbf{p} = (p_1, ..., p_C)$ (Goodfellow et al., 2016). This function ensures that each resulting probability falls within the range $[0, 1]$ and collectively, the probabilities sum to 1 *i.e.* $\sum_{c \in \mathcal{C}} p_c = 1$. It is noteworthy that in the special case where all logits are equal, the SoftMax function yields a probability distribution in which each class is assigned an equal probability of $\frac{1}{|\mathcal{C}|}$ *i.e.* $z_c = k, \forall c \in \mathcal{C} \Rightarrow p_c = \frac{1}{|\mathcal{C}|}$, where $|\mathcal{C}|$ denotes the total number of classes.

Categorical Cross-entropy Loss The CCE loss function takes probabilities (often denoted as $\mathbf{p}$) and their corresponding target values $\mathbf{t}$, and it quantifies the error that the network incurs for a given input. This information serves as crucial feedback for the network, enabling it to iteratively enhance its performance. The closed-set loss function serves as the foundational building block upon which we introduce additional open-set loss functions in Section 3.4. The CCE loss is defined as:

$$J_{\text{CCE}}(\mathbf{p}) = -\sum_{c=1}^{C} t_c \log p_c = -\log p_c \tag{3.4}$$

as the target values $\mathbf{t}$ in (3.4) are all 0 for except for the ground-truth class, the equation can be simplified.

3.3 Binary Cross Entropy

In a binary decision case, where a network contains only a single logit node instead of possessing N nodes, the use of the SoftMax activation function is not feasible for obtaining the desired probabilities. In such instances, we use the logistic activation function, which accepts logit values within the range of $(-\infty, \infty)$ and produces a probability output ranging from 0 to 1. The logistic activation function is defined as:

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} \in (0, 1) \tag{3.5}$$

Binary Cross Entropy Loss Training a network of this nature requires a loss function designed specifically for single output nodes. In contrast to the Categorical Cross-Entropy (CCE) used previously, we can employ the Binary Cross-Entropy loss function in this scenario, defined as follows:

$$J_{\text{BCE}}(p) = -(t \cdot \log p + (1 - t) \cdot \log(1 - p)) \tag{3.6}$$

Here, $t \in \{0, 1\}$ represents the target class of the current input, and p denotes the corresponding outputted probability ($= \sigma(z)$). This implies that when t is 1, only the first part becomes active, and if t is 0, the second part becomes active. Multiplying it by the logarithm of the corresponding probability provides a metric for the distance between the probability and the ground truth.
This could also be directly computed from the logits:

$$J_{\text{BCE}}(z) = \begin{cases} z - zt + \log(1 + e^{-z}), & \text{if } z \geq 0 \\ -zt + \log(1 + e^{z}), & \text{if } z < 0 \end{cases} \tag{3.7}$$

which can be simplified to:

$$J_{\text{BCE}}(z) = \max(z, 0) - zt + \log(1 + e^{|z|}) \tag{3.8}$$

During training, it might be more convenient to compute the loss directly from the logit value. However, during testing, it is necessary to activate that value using the logistic activation function (3.5) to obtain the desired probability.

3.4 Extensions for Open-Set Classification

Dhamija et al. (2018) introduced two distinct enhancements built upon the SoftMax loss function. Their aim was to train neural networks such that, for unknown samples, the network should produce a uniform distribution, making each class equally likely. This uniform distribution has practical implications, particularly in score thresholding, where it enhances decision-making processes for open-set classification.

As open-set classification lacks a consistent definition, we will adopt the notation proposed by Dhamija et al. (2018), which operates as follows:

- $\mathcal{C} = \{1, ..., C\} \subset \mathcal{Y}$: The known classes of interest that the network shall identify.
- $\mathcal{U} = \mathcal{Y} \setminus \mathcal{C}$: The unknown classes containing all types of classes the network needs to reject. The set $\mathcal{U}$ can further be divided:
  - $\mathcal{B} \subset U$: The negative classes.

– $\mathcal{A} = \mathcal{U} \setminus \mathcal{B} = \mathcal{Y} \setminus (\mathcal{C} \cup \mathcal{B})$: The unknown classes, which represent the remainder.

The samples observed during training are denoted as D'_b when they belong to $\mathcal{B}$ or D'_c when they belong to $\mathcal{C}$. Conversely, the samples observed during testing are denoted as D_b or D_c respectively.

3.4.1 Entropic Open-Set Loss

The Entropic Open-Set loss can be considered an extension of the SoftMax loss, as it broadens the loss function's applicability to encompass unknown samples. At its core, EOS aims to train the network in a way that, for any negative sample $x_i \in D'_b$, the resulting probability distribution $\mathbf{p}$ exhibits maximum entropy across C classes. The EOS loss can be defined as:

$$J_{\text{EOS}}(\mathbf{p}) = \begin{cases} -\log p_c, & \text{if } x \in D'_c \\ -\frac{1}{C}\sum_{c=1}^{C} \log p_c, & \text{if } x \in D'_b \end{cases} \tag{3.9}$$

If we define the target values t_c as $t_c = \frac{1}{C}, \forall c \in \mathcal{C}$ for all negative samples D'_b it can be rewritten very similar to the J_{CCE} loss function:

$$J_{\text{EOS}}(\mathbf{p}) = -\sum_{c=1}^{C} t_c \log p_c \tag{3.10}$$

It has been shown that J_{EOS} is minimized when all SoftMax values in $\mathbf{p}$ from (3.3) are equal (Shannon, 1948). However, given that our thesis primarily focuses on restraining feature extraction from unknown samples, it is important to note that the logit values $\mathbf{z}$ do not have to be zero. Rather, any set of logits with equal values leads to uniformly distributed SoftMax scores. Consequently, this loss function does not inherently encourage a reduction in feature magnitudes.

3.4.2 Objectosphere Loss

While EOS encourages the generation of logit values with equal magnitudes for negative and unknown samples, it does not impose the requirement to make them small. Therefore, it does not compel the network to suppress the extraction of features from unknown samples, which can then lead to poor results on unknown samples that have not been seen during training. The objectosphere loss addresses this issue by promoting large feature magnitudes and low entropy in known samples through a penalty for small feature magnitudes. Conversely, it penalizes large feature magnitudes in negative samples to minimize feature length and maximize entropy (Dhamija et al., 2018). The loss function is defined as:

$$J_R = J_{\text{EOS}} + \lambda \begin{cases} \max(\xi - \|\phi\|, 0)^2 & \text{if } x \in \mathcal{D}'_c \\ \|\phi\|^2 & \text{if } x \in \mathcal{D}'_b \end{cases} \tag{3.11}$$

where $\|\phi\|$ is the absolute magnitude of the deep feature vector, ξ is the minimal feature magnitude that is accepted for known samples and $\lambda \in \mathbb{R}^+$ can be viewed as the weight of the regularization term.

Chapter 4

Approach

In this chapter, we elaborate on our ideas regarding the adjustment of the network architecture to suppress feature extraction for unknown samples. Initially, we introduce all the adjustments to a standard open-set classification network as the one Dhamija et al. (2018) used. Subsequently, we present the out-of-distribution detection network architecture. Finally, we demonstrate how we plan to combine these techniques first independently and then into a novel unified single network.

4.1 Categorical Network

To address Research Question 1, we largely retain the network architecture as depicted in Figure 1.1 without significant alterations. The primary adjustment is the restriction of the deep feature layer from producing negative values. To achieve this, we still permit the network to train deep features $\in \mathbb{R}$, but any negative values are then mapped to 0. For the second part of our adjustments, we aim to ensure that all the weights in the last fully connected layer, which connects the deep features layer to the logits layer, are positive. To achieve this, we employ two distinct approaches that should yield the desired result. The first idea is to utilize a different type of layer. Here, before each multiplication of the features with the corresponding weights (see Section 3.1), the weight is mapped to zero if it happens to be negative, similar to the function applied to the deep features. But this essentially means, that once a weight is negative, it implies that it is not going to be updated further. To address this issue, we ensure that all weights in this layer are initially set to positive values. The second idea involves incentivizing the network to train positive values on its own. We introduce a penalty for negative values in that layer, effectively increasing the loss when negativity is detected. The penalty we choose to implement is defined as:

$$\text{penalty} = \sum_{w \in \mathbb{W}^-} w \cdot \left(\frac{|\mathbb{W}^-|}{|\mathbb{W}|} \right) \tag{4.1}$$

where $|\mathbb{W}|$ is the total number of weights in the layer and $\mathbb{W}^-$ is the set of all negative weights.

Depending on the weighting assigned to this penalty, the network should primarily train values greater than 0. To ensure the network functions as intended and is unable to produce any negative outputs, we do not allow these layers to have a bias term. By implementing these measures, we can guarantee that the only way to produce a logits vector with all zeros is for the deep features to be zero as well.

For answering Research Question 2, we employ a very similar approach to what was described in the paragraph above, with one additional change. Instead of including an extra fully connected layer immediately after the convolutional features, *i.e.* the last layer in the blackbox, where we aim

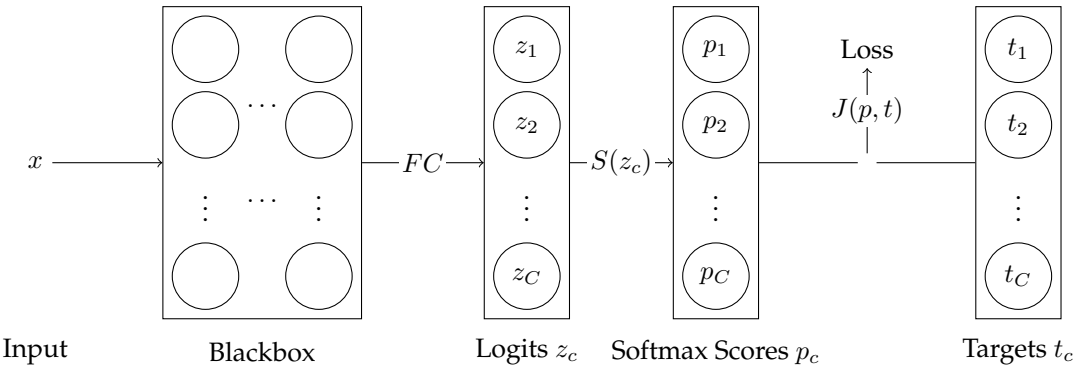


Figure 4.1: MODIFIED NETWORK OVERVIEW. *This figure presents the newly proposed architecture, illustrating all the adaptations made to the network introduced in Figure 1.1. The blackbox layers remain unchanged. However, the deep feature layer has been eliminated, and the convolutional features are directly linked to the logits layer through a fully connected layer. The computation of the SoftMax scores and the subsequent calculation of the resulting loss remain unchanged.*

Table 4.1: OVERVIEW OVER THE DIFFERENT NETWORK TYPES. *This table presents an overview of the six distinct network types we implement. The column labeled "Deep-Features" signifies the retention status of the deep feature layer. Meanwhile, "pos. Features" indicates whether the features from the last feature layer before the logits undergo ReLu activation. The "Restriction" column specifies the type of positive weight enforcement applied.*

Network	Deep-Features	pos. Features	Restriction
regular	Yes	No	-
mod. 1A	Yes	Yes	ReLu
mod. 1b	Yes	Yes	Penalty
mod. 2	No	No	-
mod. 2A	No	Yes	ReLu
mod. 2B	No	Yes	Penalty

to extract the deep features, we bypass that step and directly utilize the convolutional features to derive the corresponding logits (see Figure 4.1). The additional modifications, wherein we restrict the network from having negative weight values and a bias term in the last layer, and enforce the use of only positive values for the convolutional feature values (analogously to the strictly non negative deep features described above), are also applied in this context.

To make things more clear, we will refer to the different adaptations in the upcoming sections using the following names, which are additionally summarized in Table 4.1. The networks that serve as baseline values, remaining unaltered from their original state, will be referred to as **regular**. We assign the label **modification 1A** to the modification involving the use of solely positive values and weights for the deep feature layer, accomplished through ReLU implementation. Similarly, we term the approach involving weight training using the penalty function as **modification 1B**. **Modification 2** entails removing the deep feature layer from the network without imposing additional constraints on either the features or the weights. Similarly to before, we term the modifications in which the deep feature layer was eliminated **modification 2A** for the weights activated with ReLU and **modification 2B** for the weights trained with the penalty function.

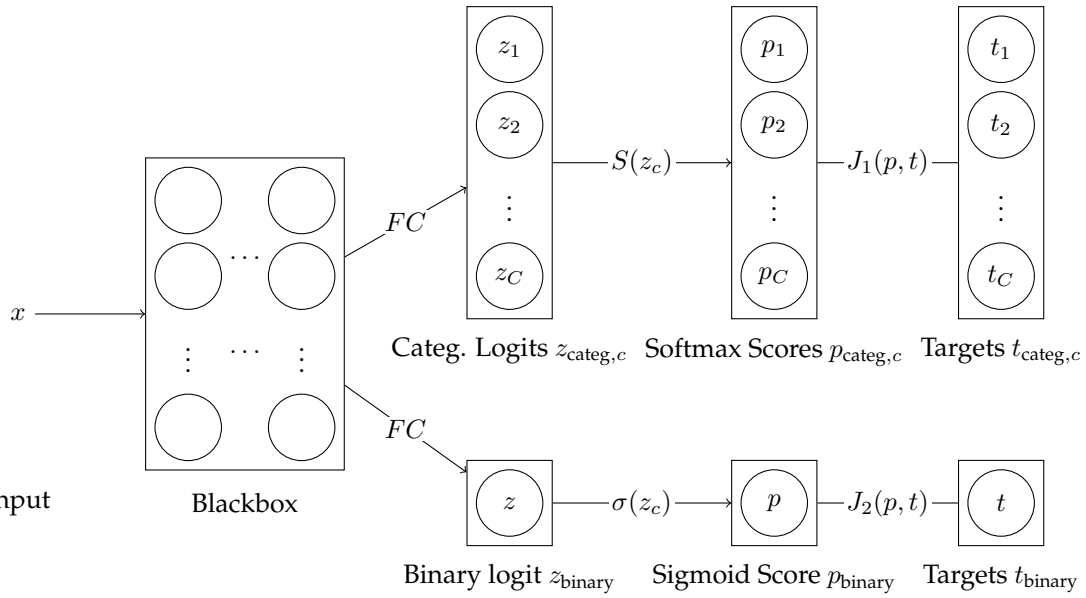


Figure 4.2: COMBINED NETWORK OVERVIEW. *This figure presents the proposed architecture for the combined network (without the deep feature layer). In contrast to Figure 4.1, the current setup includes two distinct logits and probability layers, both linked to the convolutional features layer through a fully connected layer. Subsequently, we compute the SoftMax scores from the categorical logits in a manner consistent with our previous approach. Additionally, in parallel, we calculate a sigmoid score from the binary logit. Subsequently, the loss is computed for both score layers and then aggregated by adding* $J_1(p,t) + J_2(p,t)$.

4.2 Binary Network

For both research question 1 and research question 2, we want to provide answers not only for the conventional categorical classification network but also for a second type of networks termed the out-of-distribution binary classification network that we would like to introduce. This network is dedicated to the specific task of determining whether an input belongs to the classes of interest in $\mathcal{C}$. Consequently, there is no requirement for the logit layer to possess $|\mathcal{C}|$ output nodes but a single node suffices. Hence, the sole adjustment to the topology we need to make is to reduce the number of output classes to 1 while leaving all the other architectural aspects mentioned above unchanged. In addition, we change the activation function to the logistic activation and employ the binary cross-entropy loss function for training. To ensure the proper functionality of this network, we again incorporate a bias term, the specifics of which are explained in detail in Section 5.2. Furthermore, to enable the network to process the input data, the necessity for distinct target values for the known classes $\mathcal{C}$ is eliminated, and they are now designated as 1, while the negative/unknown classes $\mathcal{U}$ are designated as 0. Once more, our curiosity lies in understanding whether our new adaptations can exert control over feature extraction and if the behavior of feature extraction differs from that observed in the above described classification networks.

4.3 Combined Network

With research question 3 we are particularly interested to explore whether we can enhance the performance of open-set classification by combining a categorical network as described above with a secondary component that focuses exclusively on classifying inputs into either known or unknown. Furthermore, we intend to implement all the previously described network topologies for both categorical networks and out-of-distribution binary networks and evaluate all possible combinations of successful approaches. By doing this, we want to assess the feasibility of this idea and determine if there exists a combination of two networks that outperforms the others or if they all exhibit similar performance levels.

The simplest and most direct approach to combine both of these networks is to train them independently. Then, during the evaluation phase, we can combine them by multiplying the output probabilities:

$$\mathbf{p}_{\text{comb}} = \mathbf{p}_{\text{categ}} \cdot p_{\text{binary}} \tag{4.2}$$

where $\mathbf{p}_{\text{categ}}$ are the SoftMax scores from the categorical network for each class $c \in C$ and p_{binary} is the probability of the input belonging to the known classes C from the binary network.

Another approach we intend to explore is training a single network with two parallel output logit layers. We want to integrate both components into a unified network with the expectation that it will yield a more user-friendly system. This approach offers the benefits of more efficient training processes, as only one network needs to be trained. Additionally, we anticipate the potential to enhance the convolutional layer's focus on positive features, thereby creating a more meaningful and effective network. Here, in the first output, we maintain the typical number of nodes corresponding to the classes of interest, enabling us to assign inputs to one of these classes. However, the other output will consist of just one node, which will assist us in determining the probability of the input belonging to either the known or the unknown classes. We then attempt to compute a loss by utilizing the binary cross-entropy loss function for the binary output logits and either of the categorical loss functions described in Section 3.2 and Section 3.4 for the categorical output logits (see Figure 4.2). Our goal during training is to minimize this combined loss. In doing so, our aspiration is that a single network can effectively handle both aspects of the two-part network described previously.

4.4 Magnitudes

To facilitate the analysis of the extracted magnitudes from the distinct networks, we have to establish a clear definition beforehand. These magnitudes are characterized as follows:

$$\text{mag} = \frac{\sum_{f \in \mathbb{F}} f^2}{|\mathbb{F}|} \tag{4.3}$$

where $\mathbb{F}$ describes the features within the convolutional features. This implies that we derive the mean squared magnitude across all nodes within the convolutional features.

Chapter 5

Experiments

5.1 Data

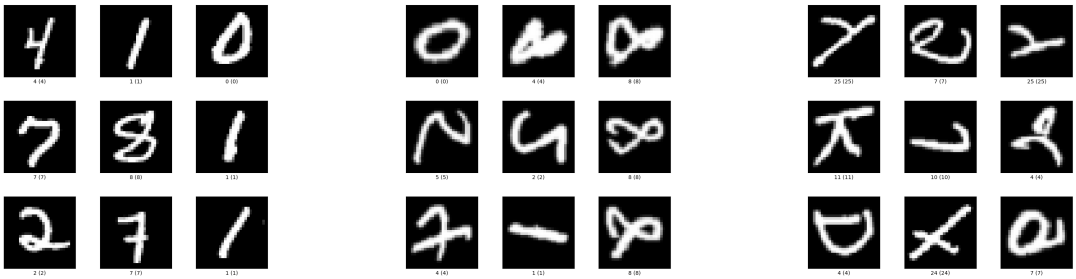
Given that this thesis primarily relies on the research of Dhamija et al. (2018), we intended to adopt a similar dataset and known and unknown split as their study. For training they employed the MNIST digits (Figure 5.1(a)) Lecun et al. (1998) as the known classes of interest and the NIST letters (Grother and Hanaoka, 2016) as the negative classes. For testing they then used the datasets Devanagri (Pant et al., 2012), NotMNIST, and CIFAR10 (Krizhevsky, 2012) as the unknown classes.

For simplicity, we planned to conduct our experiments using the MNIST digits (Figure 5.1(a)) as the known classes and the first half of the EMNIST letters (Figure 5.1(c)) Cohen et al. (2017) as the negatives *i.e.* $D_b \cup D'_b$ and the second half as the unknowns *i.e.* $D_a \cup D'_a$. However, as demonstrated by Van den Bergh (2023), there are disparities in the conversion process applied to the EMNIST dataset, such as distinct downsampling methods. Consequently, the resulting images tend to be slightly more blurry in comparison to MNIST. This blurriness provides neural networks with the ability to discern whether a sample is known or unknown. To address this, he introduced a new dataset split by utilizing the EMNIST MNIST dataset (Figure 5.1(b)), which includes all the digits from 0 to 9 but is generated using the same conversion process as the unknown set.

Furthermore, due to the existence of character and digit pairs with limited distinguishability, such as "9" and "g" or "1" and "l," the author advocated for a revised dataset split that takes this into consideration. His rationale behind this adjustment is to facilitate the creation of more visually clear and readily interpretable visualizations. Consequently, for all subsequent experiments, we adopt the following dataset configuration: the known dataset comprises all the digits from the EMNIST MNIST dataset. For the unknown dataset, we utilize the initial 11 letters from the alphabet, excluding 'o,' 'i,' and 'l,' as negatives, and the final 11 letters, excluding 'g,' as the unknown letters.

5.2 Experimental Setup

To conduct our experiments, we use two different types of network designs. The first one is the two-dimensional LeNet++, which we chose based on the configuration used by Dhamija et al. (2018) in their experiments. The second one is the SmallScale-CNN architecture with 500 dimensions. Both of these designs are thoroughly explained in Section 3.1. We then implement the exact same modifications, which are outlined below, for both of these network variants.



(a) MNIST Digits (b) EMNIST Digits (c) EMNIST Letters

Figure 5.1: MNIST DIGITS, EMNIST DIGITS AND EMNIST LETTERS. *The MNIST dataset (a) and the EMNIST MNIST dataset (b) encompass hand-written digits spanning from 0 to 9, while the EMNIST Letters dataset (c) comprises lower and upper case letters from 'a' to 'z,' all depicted in grayscale. Each image within these datasets is a 28x28 pixel square, resulting in a total of 784 pixels per image.*

To examine research question one, we implement the proposed modification outlined in Section 4.1. For the first part, we want to prohibit negative values for the deep features, which we try to achieve by applying the ReLU function from the `torch.nn.functional` package to the deep features. For the second adjustment, where we seek to permit only positive weights on the associated layer, we have two distinct implementations. In the first approach, we introduce a novel layer type where we conventionally initialize the weights and, if present, the bias. However, during the forward pass, we incorporate a ReLU activation, using the same method from the `torch.nn.functional` package, on each weight before multiplying them with the feature values. In the second approach, the penalty term for negative weights, as described in (4.1), is calculated, multiplied by a factor of two, and then added to the loss.

To address research question two, where the omission of the deep feature layer is intended, we straightforwardly eliminate the second-to-last layer. Subsequently, we adjust the incoming number of nodes from the preceding layer to match the incoming nodes from the original second-to-last layer. All the modifications described above concerning the non-negativity of the features and weights will be applied here in a one-to-one fashion.

Because we want to investigate the previously mentioned adjustments within both the closed set and open set contexts, we train the suggested configurations using the three distinct loss functions, namely SoftMax (`CrossEntropyLoss` from the `torch.nn` package), EOS, and Objectosphere (`entropic_openset_loss` and `objectoSphere_loss` from the `vast` package), described in Section 3.4.

In line with our research questions, we will investigate RQ 1 and RQ 2 using a binary classifier. To arrive at such a network, we will utilize the same architectures mentioned earlier, modifying only the number of output nodes to one. To train the model, we will use the Binary Cross Entropy function (`BCEWithLogitsLoss`) from the `torch.nn` package. It is important to note that after making the necessary adjustments to allow only positive features and weights, the network will be limited to computing logits $z \geq 0$. Consequently, the Binary Cross Entropy function applies the sigmoid activation function to map these logits to probabilities. This function assigns a probability $p = \text{sig}(z) \geq 0.5, \forall z \geq 0$. To accommodate probabilities below 0.5, we will incorporate a bias to adjust the threshold. The rest remains unaltered and is implemented in the same manner as the categorical classification network.

Regarding RQ 3, which examines the integration of a categorical classification network with

an out-of-distribution binary classification network, we employ the two approaches outlined in Section 4.3. For the first approach, where we train the networks separately, we employ the previously trained networks used to address research questions 1 and 2. We then evaluate the performance of each resultant combination for the closed-set and open-set context using the evaluation metrics described in Section 5.4. In the second approach, we aim to construct a unified network that handles both the categorical classification task and the binary classification task. We intend to reintegrate all the successful architectural modifications from the previous approach. The binary output logit layer is consequently optimized using the `BCEWithLogitsLoss`, while the categorical output logit layer is trained with the three established loss functions: CCE, EOS, and Objectosphere. It is important to note that both output logit layers are subject to identical architectural modifications. In practical terms, this signifies that if, for instance, we train a network without an additional deep feature layer and solely employ positive weights in the final layer, this adjustment will consistently impact both the binary and categorical logit layers.

For the final research question, RQ4, where we aim to explore whether the adjustments made in the preceding inquiries effectively suppresses feature extraction for unknown samples, we undertake the following process: For every network, we extract the values of the convolutional features, consisting of 2450 dimensions for the SmallScale-CNN architecture and 1152 dimensions for the LeNet++ architecture. Additionally, for Modification 1A and 2A, we perform this extraction process only after the values have undergone ReLU activation. We then compute the magnitude for each sample according to the definition in (4.3). Furthermore, we construct a histogram that showcases the relationship between the magnitudes of all samples in the set and their corresponding frequencies, differentiating between samples from the known set D_c and the unknown set D_u.

Data All training procedures are conducted utilizing the data described in chapter 5.1

Details The implementation details are publicly available.[1]

5.3 Experiments

For our experiments, we follow a method where each suggested network from RQ 1 is implemented, trained, and evaluated. Subsequently, we only proceed with the promising results for RQ 2, while discarding the other networks. These selected networks are then adapted for RQ 2 and trained and evaluated again. We will then eliminate the underperforming networks once more, ensuring we are left with only the well-performing ones, which we subsequently consider for RQ 3, aiming to explore potential combinations between them.

Training All networks are trained using the stochastic gradient descent optimization technique, a learning rate of 0.01, and a batch size of 128. When we train a network with the Objectosphere loss function, we utilize the parameters $\xi = 50$ and $\lambda = 0.0001$. During the training, we evaluate the network's performance using the confidence metric outlined in (5.4) or (5.3), respectively. In each epoch, we compare this value with the performance of the previously best-performing network and save the current network only if it surpasses the previous one. If, over 10 epochs, the network fails to exhibit improvement, we stop the training. Furthermore, we impose a maximum limit of 70 epochs during the training of the network. The training and experiments are conducted using Nvidia GeForce RTX 2080 Ti GPUs.

[1] https://github.com/larsuzh/osc_thesis/

5.4 Evaluation

To assess the performance of our trained networks on the provided data, we employ three different metrics: the accuracy metric (5.1) and (5.2), the confidence metric (5.3) and (5.4), and the OSCR Curve, comprising equations (5.5) and (5.6).

We must emphasize that throughout the training process, to assess the combined network's performance, we consistently rely on the metrics of the categorical part of the network, concentrating solely on the classification output of the known inputs.

5.4.1 Accuracy

We define accuracy for the out-of-distribution network as:

$$\text{acc}_{\text{ood}} = \frac{|\{x|\lfloor P(c|x) + \frac{1}{2} \rfloor = c'_{\text{binary}}\}|}{|\mathcal{D}_c \cup \mathcal{D}_u|}, \forall x \in \mathcal{D}_c \cup \mathcal{D}_u \tag{5.1}$$

where $P(c|x)$ represents the probability output for a given input x. Adding $\frac{1}{2}$ and flooring the result converts the probability to either 0 or 1. In this context, the target class c'_{binary} is 0 for unknown samples and 1 for known samples.

Additionally, we define accuracy for the categorical network as:

$$\text{acc}_{\text{cn}} = \frac{|\{x \,|\, \arg\max_c P(c|x) = c'_{\text{categ}}\}|}{|\mathcal{D}_c|}, \forall x \in \mathcal{D}_c \tag{5.2}$$

where c'_{categ} denotes the class of the data point, ranging from 0 to 9. In this context, only the known classes are considered, and inputs belonging to the unknown class are excluded. This metric provides insight into the network's performance in accurately classifying inputs into their respective classes.

5.4.2 Confidence

Furthermore, we define confidence for the out-of-distribution network as:

$$\text{conf}_{\text{ood}} = \frac{\sum_{x \in \mathcal{D}_c} P(c|x) + \sum_{x \in \mathcal{D}_u} (1 - P(c|x))}{|\mathcal{D}_c \cup \mathcal{D}_u|} \tag{5.3}$$

and define confidence for the categorical network as:

$$\text{conf}_{\text{cn}} = \frac{\sum_{x \in \mathcal{D}_c} P(c'_{\text{categ}}|x) + \sum_{x \in \mathcal{D}_u} [1 + \frac{1}{|\mathcal{C}|} - \max_c P(c|x)]}{|\mathcal{D}_c \cup \mathcal{D}_u|} \tag{5.4}$$

where c' represents the corresponding target label for the current input x and therefore $\sum_{x \in \mathcal{D}_c} P(c'|x)$ represents the aggregate of the returned probabilities for the ground truth class. The sum $\sum_{x \in \mathcal{D}_u} [1 + \frac{1}{|\mathcal{C}|} - \max P(c|x)]$ offers an indication of how effectively the network achieves a uniform distribution for unknown inputs. The inclusion of the term $\frac{1}{|\mathcal{C}|}$ is crucial, as in the optimal scenario, we anticipate all probabilities to be of that magnitude. Consequently, we aim to penalize any deviation from this ideal state. Summing these terms and dividing by the size of the training dataset yields a metric that not only characterizes the network's classification accuracy but also the confidence with which it does so (Palechor et al., 2023). This implies that a confidence of 1 can be attained if, for all known inputs, the network consistently outputs a probability of 1 for the correct class and 0 for the rest. For unknown inputs, it should return a probability of $\frac{1}{|\mathcal{C}|}$ for all classes, effectively maximizing entropy.

5.4.3 OSCR Curve

For a more comprehensive analysis of performance on the open set, we employ the OSCR curve, initially introduced by Dhamija et al. (2018). This curve later underwent subsequent enhancements by Palechor et al. (2023) and Bisgin et al. (2023).This curve plots the values of two distinct metrics against each other. These metrics include the correct classification rate (CCR), which addresses the known samples, and the false positive rate (FPR), which addresses the unknown samples and both of which are defined as a function of θ:

$$\text{FPR}(\theta) = \frac{|\{x \,|\, x \in \mathcal{D}_a \wedge \max_c P(c|x) \geq \theta\}|}{|D_c|} \tag{5.5}$$

$$\text{CCR}(\theta) = \frac{|\{x \,|\, x \in \mathcal{D}_c \wedge \arg\max_c P(c|x) = c' \wedge P(c'|x) \geq \theta\}|}{|D_c|} \tag{5.6}$$

where, FPR(θ) signifies the proportion of misclassified inputs, where the probability for any class is expected to be low, but is actually higher than the specified threshold. CCR(θ) on the other hand represents the fraction of accurately classified inputs, where the highest probability not only corresponds to the correct class but is also greater than the specified threshold. The OSCR curves are constructed by plotting the resulting CCR values against the FPR values while gradually increasing the threshold θ from 0 to 1.

When interpreting the curve, it is crucial to consider that the plot is influenced by the selected θ, although this variable is not directly depicted in the plot. In essence, our goal is to maximize the CCR as close to 1 as possible across all FPR values, particularly focusing on lower values. This signifies that the network effectively classifies inputs correctly while minimizing misclassifications of unknown samples as known ones to obtain a straight line at the top of the CCR-axis for all FPR values.

Chapter 6

Results

This chapter presents the results of all the conducted experiments. We reference the various network types using the same naming convention introduced in Section 4.1 and summarized in Table 4.1.

6.1 Evaluation of Network Topology Adjustments

Due to unsatisfactory results obtained with the LeNet++ architecture, we have chosen to exclusively showcase the outcomes of the SmallScale-CNN in this presentation. Section 7 will elaborate on the shortcomings of the LeNet++ architecture. Furthermore, all subsequent experiments following modification 1 are conducted exclusively using the SmallScale-CNN architecture.

6.1.1 Research Question 1

Table 6.1 shows the performance metric results for the SmallScale-CNN architecture with the regular, modification 1A, and modification 1B topologies. With this network architecture, it is evident that all three topologies demonstrate high performance after beeing trained with the SoftMax, EOS, or Objectosphere loss. In all of these cases, the accuracy consistently exceeds 99% for the training dataset and remains above 98% for both the validation and test sets. Furthermore, the confidence remains above 93% for all cases on the training and validation sets. On the test set, the networks attain a confidence level around 60% to 80%, with no clear deviation of modification 1A and 1B from the regular network.

In Figure 6.1, we provide the results for the different SmallScale-CNN network topologies within the OSCR context. Additionally, we offer a comprehensive breakdown of the test-set results in Table A.4. Here, it is evident that on the validation set, all three topologies exhibit improved performance when trained with the EOS or Objectosphere loss compared to the SoftMax loss. However, on the test set, this advantage diminishes, with all three topologies demonstrating roughly similar performance regardless of the loss function used during training. It gets clear from this, that the networks achieve similar performance even with the additional modifications 1A and 1B.

In Table 6.2, we showcase the performance metrics of the SmallScale-CNN architecture in an out-of-distribution context. The table demonstrates that the original network, when trained with an output node and a BCEWithLogitsLoss, can indeed yield good results in terms of accuracy and confidence during the training and validation phase. It attains nearly 100% accuracy and confidence on the training set and approximately 98% on the validation set. The same holds true for modification 1A and 1B, where we also achieve nearly 100% on the training set and slightly

Table 6.1: SMALLSCALE-CNN PERFORMANCE OF MODIFICATION 1A AND 1B. *This table displays the accuracy and confidence metric results for the regular, 1A, and 1B topologies across the training, validation, and test sets.*

Arch	Loss	Training		Validation		Test	
		Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
regular	SoftMax	1	0.99988	0.9929	0.99251	0.9933	0.60162
	EOS	0.99575	0.94703	0.9898	0.93967	0.9898	0.8012
	Objectosphere	0.9938	0.93925	0.9895	0.93365	0.9895	0.79968
mod. 1A	SoftMax	1	0.99980	0.99360	0.99301	0.99360	0.60677
	EOS	1	0.99844	0.99220	0.98090	0.99220	0.80132
	Objectosphere	0.99997	0.99568	0.99140	0.97921	0.99140	0.79485
mod. 1B	SoftMax	1	0.99981	0.9943	0.99322	0.9943	0.60825
	EOS	1	0.99846	0.9931	0.9803	0.9931	0.798
	Objectosphere	0.99985	0.99565	0.9924	0.98029	0.9924	0.79722

Table 6.2: SMALLSCALE-CNN OUT-OF-DISTRIBUTION PERFORMANCE OF MODIFICATION 1A AND 1B. *This table displays the accuracy and confidence metric results for the regular, 1A, and 1B topologies across the training, validation, and test sets in an out-of-distribution context.*

Arch	Training		Validation		Test	
	Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
regular	0.99832	0.99709	0.97835	0.97746	0.53191	0.75626
mod. 1A	0.99999	0.99955	0.98553	0.98422	0.53191	0.64921
mod. 1B	1	0.99984	0.98532	0.98468	0.53191	0.75913

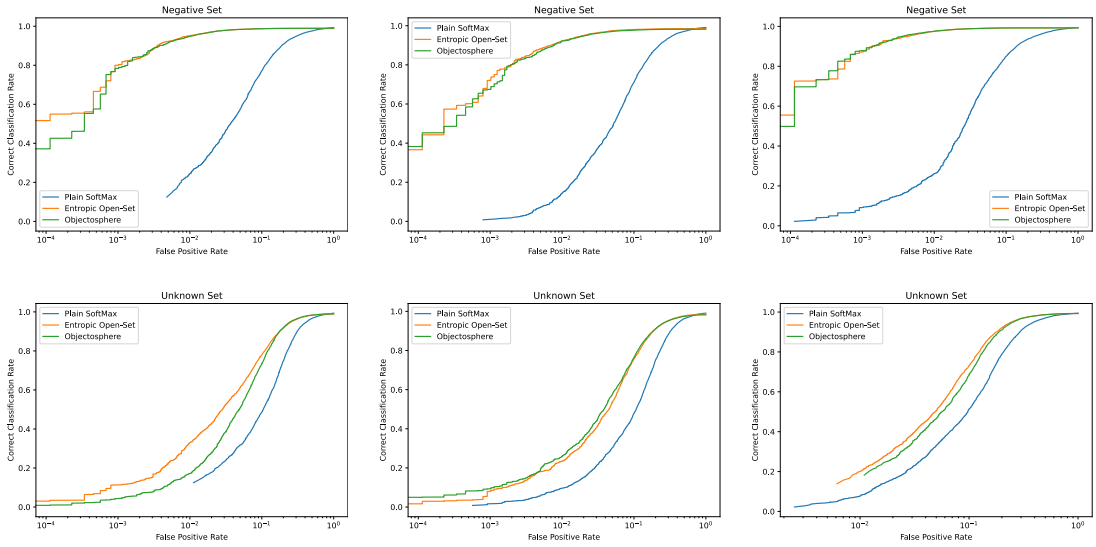
over 98% on the validation set. On the test set, all three topologies exhibit unfavorable outcomes, reaching a maximum of 53% accuracy and 75% confidence for the regular and modification 1B, and an even lower 65% for modification 1A.

6.1.2 Research Question 2

In Table 6.3, we are presenting the performance metric results of the 2, 2A and 2B modifications. Here you can see that all three approaches deliver very good results, regardless of the type of loss function utilized. In these instances, no clear differences can be observed between the three modifications. It is also noteworthy that the accuracy on the test set is approximately 99% for all cases and even the decline in confidence compared to other datasets is fairly consistent for the three networks.

The OSCR curve in Figure 6.2 (the results for the test set are once more comprehensively illustrated in Table A.5) presents the open set performance results for Modification 2, 2A and 2B. In the validation set, networks trained with EOS and Objectosphere loss functions exhibit commendable outcomes, achieving CCRs ranging between 60% to 80% for FPRs as low as 10^{-3}. Conversely, the network trained with the SoftMax loss lags behind, producing acceptable CCR only at approximately 10^{-1} FPR. Nonetheless, on the test set, all of the networks encounter a decline in performance relative to the validation set, with the curves closely resembling those illustrated in Figure 6.1.

In Table A.6, we present the performance metrics of the adjusted SmallScale-CNN architecture without the deep feature layer in an out-of-distribution context. The table highlights that the networks with Modification 2, 2A, and 2B, when trained with one output node and a BCEWith-



(a) regular (b) mod. 1A (c) mod. 1B

Figure 6.1: SMALLSCALE-CNN OSCR CURVES. *This figure presents the OSCR (Open-Set Classification Rate) curve for the regular architecture (a), modification 1A (b), and modification 1B (c). The first row represents the validation set, while the second row depicts the test set.*

LogitsLoss, are capable of delivering high performance outcomes. They achieve just over 98% accuracy and 97% confidence on the training set and approximately 97% on the validation set. Once again, there is a noticeable performance decline when assessing all three networks on the test set, with all of them achieving only 53% accuracy and 75% confidence for modification 2A, 76% for modification 2, and 77% for modification 2B.

6.2 Combination Strategies

This section showcases the outcomes of the various approaches we employed to integrate a categorical network with a binary network. Initially, we present the results for the technique where the networks were trained independently and then merged for evaluation. Throughout these results, we label the networks as follows: name of the categorical network - name of the binary network.

Combined Networks Figure 6.3 depicts the OSCR curves for all the carried out combinations. Additionally, a more comprehensive view of the test set results from these experiments is provided in Table A.7. On the validation set, the combinations demonstrate similar outcomes for the EOS and Objectosphere loss functions compared to the previous results, with the SoftMax loss function showing even better performance. For all three loss functions, around an FPR of approximately 10^{-3}, a CCR of 60% to 80% is noticeable, which escalates to over 80% at 10^{-2} and gradually converges to 1 with increasing FPRs. What is interesting to note here, in contrast to the preceding findings, is the remarkable similarity between the performance on the test set and that on the validation set, with instances where the test set performance is even better. Among the tested combinations, only the regular - regular pairing demonstrates a minor decline in performance on the test set, particularly in the objectosphere case. In contrast, the mod. 2B - mod. 2B combination exhibits the most impressive performance, where there are clear advantages of

Table 6.3: SMALLSCALE-CNN PERFORMANCE OF MODIFICATION 2, 2A AND 2B. *This table displays the accuracy and confidence metric results for the 2A and 2B topologies across the training, validation, and test sets.*

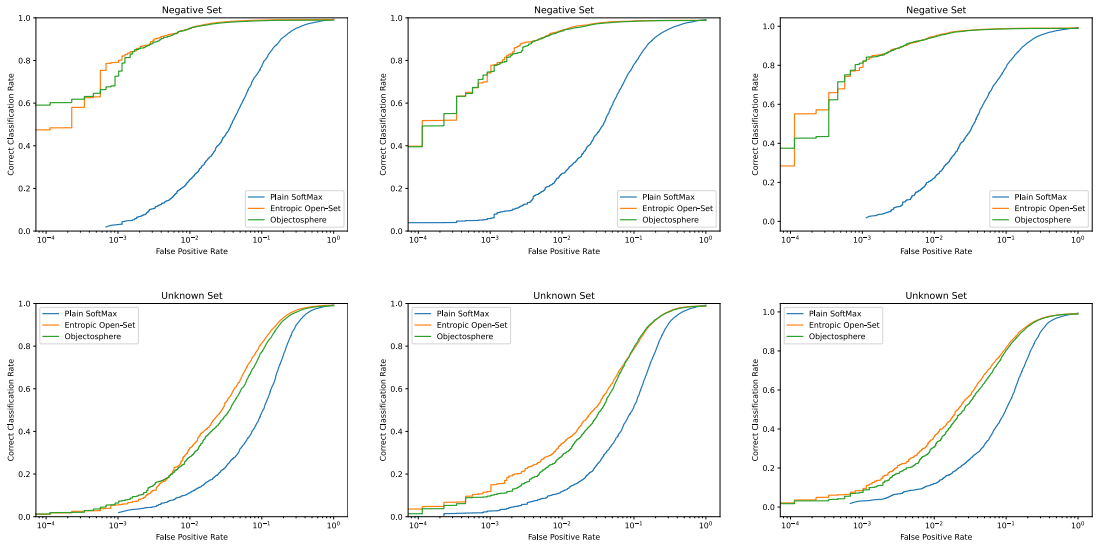
Arch	Loss	Training		Validation		Test	
		Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
mod. 2	SoftMax	1	0.99949	0.9928	0.99174	0.9928	0.60842
	EOS	0.99555	0.94414	0.9918	0.93776	0.9918	0.8064
	Objectosphere	0.99447	0.94389	0.9892	0.93819	0.9892	0.80183
mod. 2A	SoftMax	0.99950	0.99821	0.99220	0.99039	0.99220	0.61264
	EOS	0.99370	0.94168	0.98830	0.93518	0.98830	0.80748
	Objectosphere	0.99298	0.94293	0.98760	0.93661	0.98760	0.81352
mod. 2B	SoftMax	1	0.9997	0.9934	0.99188	0.9934	0.60798
	EOS	0.99567	0.94464	0.9904	0.93772	0.9904	0.80724
	Objectosphere	0.99498	0.94334	0.989	0.93654	0.989	0.80547

the EOS and Objectosphere loss functions over the SoftMax loss function. For the remaining networks, we do not observe a definitive trend indicating that one of the loss functions is clearly superior in distinguishing known inputs from unknown inputs.

Mixed Networks For the mixed network approaches depicted in Figure 6.4 and Table A.9, we note slightly inferior results on the validation set compared to the results of the combination approach. However, the results appear to be consistently comparable to those achieved with the corresponding modifications from Figure 6.1 and Figure 6.2 with the notable exception of the SoftMax results, which exhibit significantly better performance in this case. Again, we can see that the results on the test set remain consistent with those on the validation set. Moreover, the EOS and Objectosphere results for the mod. 2B network continue to demonstrate highly impressive outcomes, boasting above-average CCR values, particularly for very low FPR values. With $\Sigma = 3.848$, this particular network, when being trained with the Objectosphere loss, attains the highest cumulative sum over the FPRs among all the conducted experiments. For the sake of completeness, we furnish the accuracy and confidence performance for the categorical part of these networks in Table A.8.

6.3 Feature Extraction Suppression

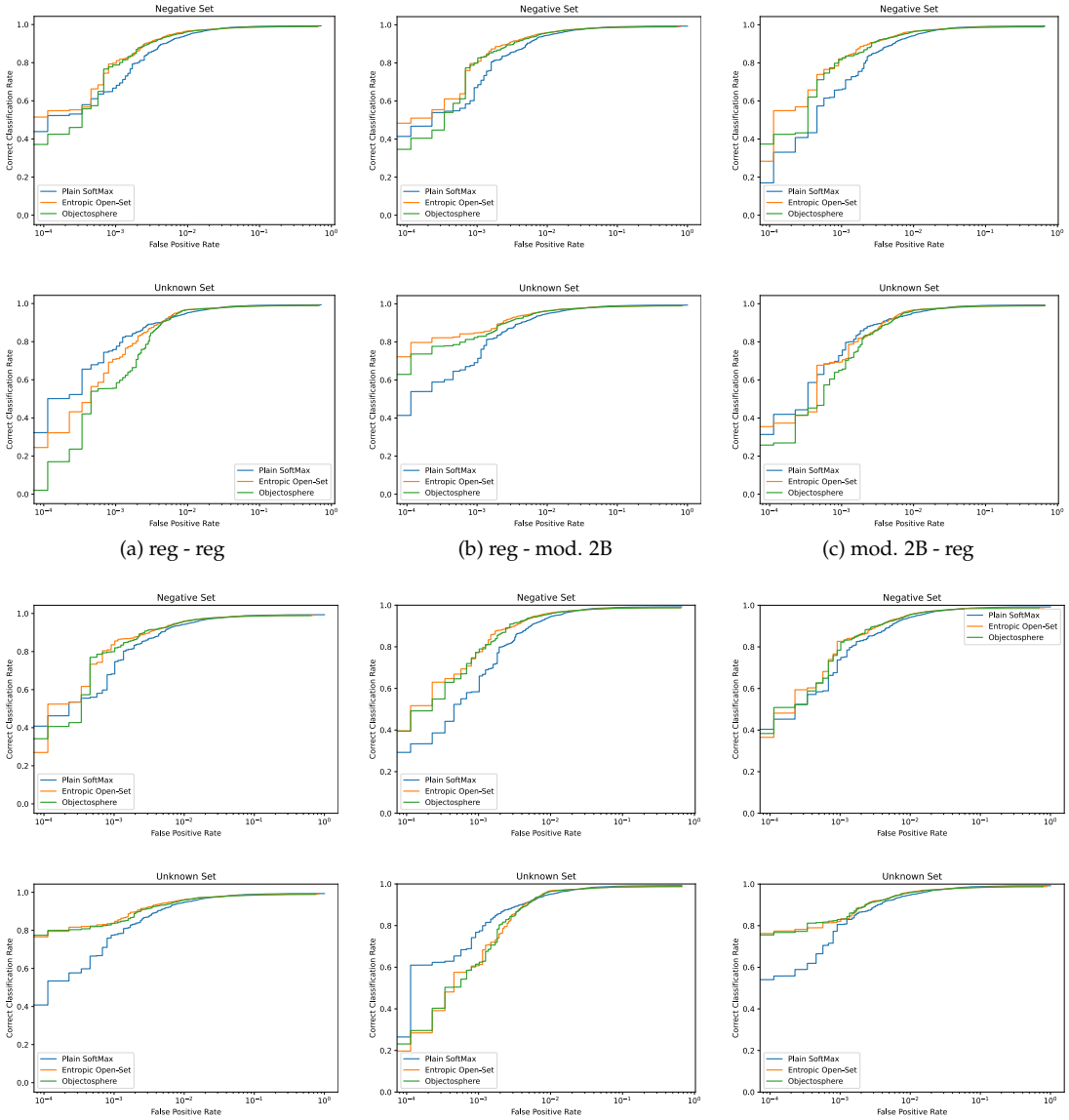
Table A.10, Table A.11, and Table A.12 present the extracted magnitudes from distinct networks at the convolutional features layer. The provided data includes the mean and standard deviation for each network and dataset split. The data suggests that there is a considerable range in magnitudes between different modifications. For instance, mod. 1A, trained with Objectosphere loss, displays significantly larger magnitudes, reaching up to 33. In contrast, mod. 2A maintains much lower magnitudes, as low as 0.653. Additionally, the resulting magnitudes for each input set exhibit a relatively minor standard deviation. For instance, for the regular network trained with the SoftMax loss function, the mean ranges from 2.197 to 2.239, with the standard deviation consistently below 0.641. In the case of categorical networks, all architectures successfully achieved lower magnitudes for the D_a and D_b split compared to the D_c split after being trained with either the EOS or Objectosphere loss function. It is noteworthy that this trend does not hold true for SoftMax, as it occasionally results in higher values for negative or unknown data. In the out-of-distribution network, all suggested architectures yielded lower values, except for the regular one, which recorded magnitudes of 0.154 for known values and 0.157 for negative values. With modi-



(a) mod. 2 (b) mod. 2A (c) mod. 2B

Figure 6.2: MODIFIED SMALLSCALE-CNN OSCR CURVES. *This figure presents the OSCR (Open-Set Classification Rate) curve for the modification 2 (a) 2A (b) and modification 2B (c). The first row represents the validation set, while the second row depicts the test set.*

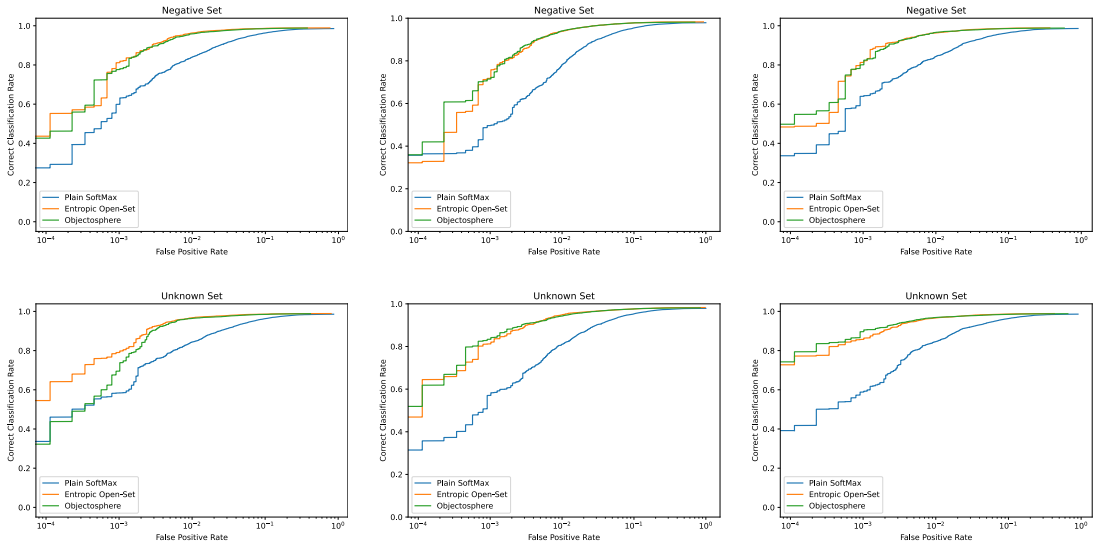
fication 2A, the network succeeded in suppressing only half of the magnitudes, specifically 0.006 for the unknown samples compared to the 0.012 for the positive values. In the mixed networks, all experiments, including those incorporating the SoftMax loss, consistently produced lower values for the negative and unknown cases. In both Figure 6.5 and Figure A.4, we attempted to visualize the results graphically. The plots showcase the extracted magnitudes plotted against their respective frequencies for the various networks. The red line corresponds to the magnitudes of the known inputs, while the blue line represents the magnitudes of the unknown inputs. Note that the top figure corresponds to the validation set, whereas the bottom figure corresponds to the test set. Once more, one can observe that there is a variance in magnitudes between the two datasets, as the blue lines are consistently to the left of the red lines.



(a) reg - reg (b) reg - mod. 2B (c) mod. 2B - reg

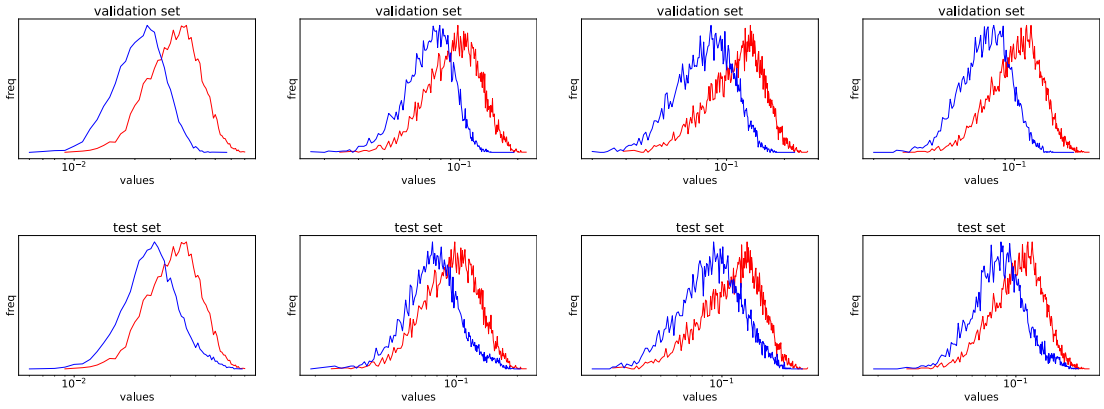
(d) mod. 2B - mod. 2B (e) mod. 2A - reg (f) mod. 2A - mod. 2B

Figure 6.3: COMBINED SMALLSCALE-CNN OSCR CURVES. *This figure presents the OSCR (Open-Set Classification Rate) curve for all the implemented combinations of categorical and binary networks. Each combination is associated with two figures, with the upper row displaying the validation set results and the lower row showcasing the results for the test set.*



(a) regular (b) mod. 2A (c) mod. 2B

Figure 6.4: MIXED SMALLSCALE-CNN OSCR CURVES. *This figure presents the OSCR (Open-Set Classification Rate) curve for all the mixed networks. The first row displays the validation set results and the second row displays the results for the test set.*



(a) regular (b) mod. 2 (c) mod. 2A (d) mod. 2B

Figure 6.5: MAGNITUDE PLOTS FOR THE SMALLSCALE-CNN WITH EOS LOSS FUNCTION. *This figure presents the extracted magnitudes for the regular architecture (a), modification 2 (b), modification 2A (c), and modification 2B (d) after being trained with the EOS loss function. We graph the resulting magnitudes extracted from the unknown input in blue, and from the known inputs in red, against the frequencies at which they occurred across all inputs in the set. The first row represents the resulting magnitudes of the validation set, while the second row depicts the resulting magnitudes from the test set.*

Chapter 7

Discussion

The LeNet++ architecture with the regular topology demonstrates high performance across all data-set splits, irrespective of the three loss functions used during training. However, modification 1A does not surpass 70% accuracy for any dataset split, regardless of the loss function utilized. Furthermore, modification 1B performs even lower, with a maximum accuracy of 10%, failing to surpass the accuracy attainable by chance alone.

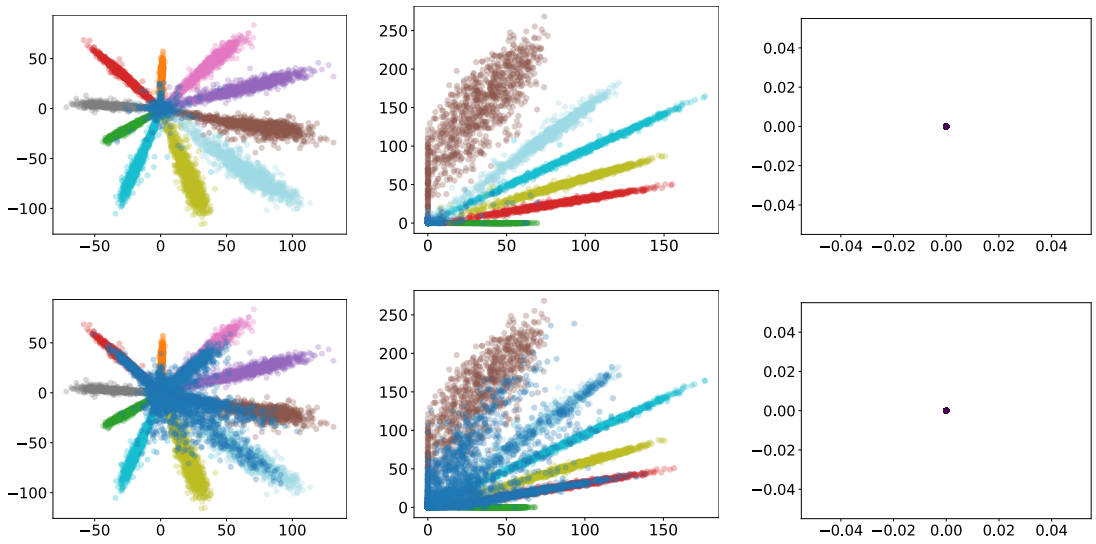
Given that, in an open-set context, modification 1A consistently is not exceeding 70%, and modification 1B remains at 10% across all false positive rates (FPRs), it is clear that both modifications demonstrate performance deficiencies in comparison to the regular network regarding the open-set classification task (The complete set of results can be found in the appendix in Table A.1, Figure A.1, and Table A.2).

Following a similar approach to Wen et al. (2016), in Figure 7.1, we present the resulting 2-D deep features displayed on a two-dimensional plane, illustrating the distribution of the LeNet++ network trained using the EOS loss. Here it is very exciting to see that the regular network (Figure 7.1(a)) is able to separate the 10 classes very clearly from each other. Especially on the validation set the unknown classes (shown in dark blue) are near the middle. However, Figure 7.1(b) depicts the deep features of the network with modification 1A, and it becomes evident that restricting the values exclusively to positive values significantly restricts the feature space, allowing only six classes to be distinctly separated from each other and the remaining features seem to be mapped to the $0/0$ point. Additionally, as Figure 7.1(c) shows, modification 1B yields an uninformative network, as no discernible features are extracted, causing all classes to converge precisely at the center.

In the out-of-distribution context, as illustrated in Table A.3, both the regular topology and mod. 1A demonstrate strong performance on the training and validation sets, but neither network exceeds random chance performance on the test set. Once again, the feature plot illustrated in Figure A.2 emphasizes that these adjustments fail to produce meaningful results, especially on the test set.

In summary, we can conclude that constraining the LeNet++ architecture to utilize only positive features significantly narrows the network's feature space. This limitation hinders the ability to effectively distinguishing the classes in both the categorical and the binary case, as the available feature space becomes insufficient for proper class differentiation.

On the other hand, as the results outlined in Section 6.1.1 suggest, the same is not applicable to the SmallScale-CNN. In the categorical context, both modifications 1A and 1B consistently show very similar performance to the regular baseline, regardless of the loss function used during training, in both closed and open set evaluations. In the binary context, all three networks exhibit very similar performance with no clearly noticeable deviation, achieving high accuracy and confidence on the training and validation sets. However, when it comes to the test set, results cannot surpass those expected by chance alone.



(a) Regular (b) mod. 1A (c) mod. 1B

Figure 7.1: LeNet++ Categorical Classification Feature-Plot. *This figure illustrates the extracted features for the regular topology (a), modification 1A (b), and modification 1B (c) of the two-dimensional LeNet++ trained with the EOS loss function. Feature pairs attributed to different classes are represented in distinct colors. The first row represents the validation set, while the second row depicts the test set.*

To summarize the findings for *Research Question 1*, it can be concluded that the modified SmallScale-CNN performs very similar to the regular topology in both closed and open set contexts for the categorical case, and it also demonstrates it's competence in the out-of-distribution context. Given that the tested networks primarily differ in the number of features in the last layer, these results suggest that, for a CNN, employing such alterations is feasible when sufficient feature space is provided.

For *Research Question 2*, the numbers presented in Section 6.1.2 indicate that the performance of the modifcation 2, 2A and 2B remains consistent with the baseline results across all cases — whether categorical or binary, and in both closed and open set scenarios. Hence, we can conclude that a network with the deep feature layer removed can still perform effectively, even when trained with additional weight restrictions. Nevertheless, while these networks maintained similar performance to the baseline, it does not imply an improvement with these approaches. The confidence on the test set experiences a decrease compared to the training and validation sets, and the OSCR curves illustrate that we are still unable to attain satisfactory results.

In response to *Research Question 3*, our findings indicate that both of our proposed implementation ideas are effective. This implies that it is viable to train a categorical network alongside a binary out-of-distribution network. Subsequently, combining the output probabilities in the way we described in Section 4.3 enables the acquisition of open-set classification capabilities. In this scenario, there is no discernible difference between the proposed topologies, and none of the three applied loss functions demonstrates a clear advantage over the others, but all tests yield promising results. Moreover, the mixed network implementation proves its effectiveness in achieving strong open-set classification performance. Interestingly, the objectosphere loss function outperforms both SoftMax and EOS loss functions for all network topologies but there seems to be no evident advantage of one topology over the other among the four investigated in this context.

To asses the results for *Research Question 4* for the categorical networks, we perform a one-

Table 7.1: EFFECT SIZE OF THE DIFFERNCES IN EXTRACTED MAGNITUDES OF THE CATEGORICAL AND MIXED NETWORKS. *This table presents the avarage Cohen's D value and its corresponding standard deviation of the differences in extracted magnitudes between the known and unknown samples.*

Arch	**Loss**	**Cohen's D**	
		classical	mixed
regular	EOS	0.804 ± 0.016	0.559 ± 0.015
	Objectosphere	0.612 ± 0.015	0.569 ± 0.014
mod. 2	EOS	0.670 ± 0.015	0.449 ± 0.015
	Objectosphere	0.833 ± 0.015	0.940 ± 0.015
mod. 2A	EOS	0.696 ± 0.015	0.987 ± 0.015
	Objectosphere	0.926 ± 0.015	1.047 ± 0.016
mod. 2B	EOS	0.717 ± 0.016	0.579 ± 0.016
	Objectosphere	0.792 ± 0.016	0.945 ± 0.015

sided t-test to check if the magnitudes extracted for negative or unknown samples are smaller than those for known samples. Here, we state the null hypothesis $H0 : \mu_2 \geq \mu_1$ and the alternative hypothesis $H1 : \mu_2 < \mu_1$, with μ_2 being the mean of the extracted magnitudes from D_a and μ_1 being the mean of the extracted magnitudes from D_c. The conducted tests show that all the topologies, except for the regular and modification 2 networks after being trained with the SoftMax loss, showed a p-value much smaller than 0.05 and thus indicate a significant difference. Considering Cohen's D statistics (see Table 7.1), the effect sizes for the classical networks range from the upper end of medium to high (Brydges, 2019). Here we obtained the mean Cohen's D and its corresponding standard deviation values through the application of bootstrapping. The Objectosphere loss in mod. 2A yielded the highest effect size at 0.926 ± 0.015. Nevertheless, this architecture, despite utilizing the same approach, only attained a value of 0.696 ± 0.015, failing to surpass the analogous result of the standard network. Hence, the inference is that the newly proposed architectures may exhibit notably reduced feature magnitudes for unknown samples, indicating considerable effect sizes. However, attributing this solely to the implemented modifications is unclear, given the inconsistency in effect sizes and the lack of substantial differences compared to the baseline. Examining the results for the mixed networks reveals notable achievements, particularly with mod. 2A displaying very high effect sizes. Additionally, the Objectosphere results for mod. 2 and 2B are high as well. In contrast, the regular topology only attains 0.559 ± 0.015 or 0.569 ± 0.014, respectively. However, the high effect sizes do not consistently align with the outcomes in open set classification tests. For instance, mod. 2B with EOS performs exceptionally well in OSCR curve analysis but does not exhibit a correspondingly high effect size when we evaluate its magnitudes. For the out-of-distribution networks, the differences are much bigger. The regular network attains a notably low effect size of 0.039 ± 0.014, while modification 2 reaches $0.112 \pm 0.0.16$ and 2B achieves 0.421 ± 0.016. In stark contrast, mod. 2A demonstrates a substantial effect size of 0.896 ± 0.014. However, it is important to note that these topologies include a bias term, introducing a complexity that makes the results less interpretable, since the presence of these bias values may hinder the networks from strictly combining positive values in the last layer.

Chapter 8

Conclusion

In this thesis, our primary aim was to assess the impact of removing deep features, introducing a positive feature, and imposing weight restrictions on the suppression of convolutional feature extraction from unknown samples. We wanted to investigate the consequences of these modifications on the performance of out-of-distribution classification networks in the open set context and categorical networks in both open and closed set contexts. Our investigation encompassed training on EMNIST Digits as known classes and EMNIST Letters as negative and unknown classes. Additionally, we explored the integration of categorical and out-of-distribution network types into a unified classification network and evaluated its performance.

Our findings indicated that introducing these alterations to a convolutional neural network did not lead to a significant variation in performance, provided the network possessed a high-dimensional feature space. However, we also observed no improvement in performance for the proposed topologies over the regular one. Despite this, our combination efforts yielded promising results. Both the combination of distinct networks and the proposed mixed network demonstrated increased correct classification rates, particularly at low false positive rates, compared to the initial networks. Nevertheless, despite the implemented modifications successfully achieving the suppression of feature extraction for unknown samples, we were unable to demonstrate a consistent superiority over what can be achieved with EOS or Objectosphere loss independently. Moreover, classical and mixed networks that effectively achieved suppression did not consistently exhibit improved performance in open-set classification tasks, leaving us uncertain as to whether the suppression effect was insufficiently large or if the introduced alterations had not brought about meaningful improvements.

Looking forward, numerous avenues for future research emerge. Most importantly it should be investigated whether the observed high correct classification rate values for the combined networks are reproducible when applied to a larger dataset with more complex inputs, resembling real-life scenarios. Furthermore, delving into feature visualization to comprehend the extracted features and their differences from those of the original networks would probably offer valuable insights into the underlying mechanisms of our proposed modifications. Finally, additional research efforts aimed at understanding whether the good performance of these networks comes from their ability to refrain from extracting features from unknown samples or what other factors contribute to distinguishing the unknown from the known would be desirable.

Appendix A

Attachments

Table A.1: LeNet++ Performance of Modification 1A and 1B. *This table displays the accuracy and confidence metric results for the regular, 1A, and 1B topologies across the training, validation, and test sets.*

Arch	**Loss**	Training		Validation		Test	
		Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
regular	SoftMax	1	0.99991	0.991	0.98989	0.9906	0.63553
	EOS	0.99982	0.99823	0.9872	0.98262	0.9848	0.7982
	Objectosphere	0.99932	0.99472	0.9886	0.98192	0.986	0.80183
mod. 1A	SoftMax	0.19982	0.23058	0.19960	0.23179	0.19960	0.19960
	EOS	0.69355	0.81958	0.68840	0.81567	0.68140	0.70859
	Objectosphere	0.49638	0.74324	0.49400	0.74389	0.49240	0.68763
mod. 1B	SoftMax	0.0998	0.10004	0.1	0.1	0.1	0.52128
	EOS	0.09992	0.52125	0.1	0.52128	0.1	0.52128
	Objectosphere	0.09992	0.52124	0.1	0.52128	0.1	0.52128

Table A.2: LeNet++ Open-Set Performance of Modification 1A and 1B. *The table exhibits the achieved test-set CCR (Correct Classification Rate) results at various FPR (False Positive Rate) levels of* 10^{-3}, 10^{-2}, 10^{-1}, *and* 10^{0}, *along with the cumulative sum across these values, for the regular, 1A, and 1B topologies.*

Arch	**Loss**	1e-3	1e-2	1e-1	1e-0	Σ
regular	SoftMax	0.0369	0.2514	0.8431	0.9907	2.1221
	EOS	0.0796	0.2605	0.7752	0.9849	2.1002
	Objectosphere	0.1233	0.3135	0.7892	0.986	2.212
mod. 1A	SoftMax	0.03	0.0723	0.0984	0.1996	0.4003
	EOS	0.0097	0.0836	0.5048	0.6813	1.2794
	Objectosphere	0.0039	0.1057	0.3904	0.4923	0.9923
mod. 1B	SoftMax	0.1	0.1	0.1	0.1	0.4
	EOS	0.1	0.1	0.1	0.1	0.4
	Objectosphere	0.1	0.1	0.1	0.1	0.4

Table A.3: LeNet++ Out-Of-Distribution Performance of Modification 1A and 1B. *This table displays the accuracy and confidence metric results for the regular, 1A, and 1B topologies across the training, validation, and test sets in an out-of-distribution context.*

Arch	Training		Validation		Test	
	Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
regular	1	1	0.98947	0.98919	0.53191	0.7559
mod. 1A	0.99984	0.99957	0.98968	0.98920	0.53191	0.75162
mod. 1B	0.53191	0.50198	0.53191	0.5026	0.53191	0.5026

Table A.4: SMALLSCALE-CNN OPEN-SET PERFORMANCE OF MODIFICATION 1A AND 1B. *The table exhibits the achieved test-set CCR (Correct Classification Rate) results at various FPR (False Positive Rate) levels of* 10^{-3}, 10^{-2}, 10^{-1}, *and* 10^{0}, *along with the cumulative sum across these values, for the regular, 1A, and 1B topologies.*

Arch	**Loss**	1e-3	1e-2	1e-1	1e-0	Σ
regular	SoftMax	0.1246	0.1246	0.4864	0.9933	1.7289
	EOS	0.1137	0.3304	0.78	0.9898	2.2139
	Objectosphere	0.0444	0.1729	0.7334	0.9895	1.9402
mod. 1A	SoftMax	0.0349	0.0572	0.5161	0.9936	1.6018
	EOS	0.2059	0.2059	0.6954	0.9922	2.0994
	Objectosphere	0.2292	0.2292	0.6353	0.9914	2.0851
mod. 1B	SoftMax	0.0222	0.0785	0.5139	0.9943	1.6089
	EOS	0.1386	0.2009	0.7306	0.9931	2.0632
	Objectosphere	0.1821	0.1821	0.6846	0.9924	2.0412

Table A.5: SMALLSCALE-CNN OPEN-SET PERFORMANCE OF MODIFICATION 2, 2A AND 2B. *The table exhibits the achieved test-set CCR (Correct Classification Rate) results at various FPR (False Positive Rate) levels of* 10^{-3}, 10^{-2}, 10^{-1}, *and* 10^{0}, *along with the cumulative sum across these values, for the 2A and 2B topologies.*

Arch	**Loss**	1e-3	1e-2	1e-1	1e-0	Σ
mod. 2	SoftMax	0.0194	0.1124	0.4876	0.9928	1.6122
	EOS	0.0566	0.3232	0.8135	0.9918	2.1851
	Objectosphere	0.0735	0.2808	0.7729	0.9892	2.1164
mod. 2A	SoftMax	0.0274	0.1194	0.5179	0.9922	1.6569
	EOS	0.1498	0.3438	0.7871	0.9883	2.269
	Objectosphere	0.1012	0.2865	0.7937	0.9876	2.169
mod. 2B	SoftMax	0.0304	0.1207	0.5019	0.9934	1.6464
	EOS	0.0956	0.3623	0.8172	0.9904	2.2655
	Objectosphere	0.0877	0.3114	0.8028	0.989	2.1909

Table A.6: SMALLSCALE-CNN OUT-OF-DISTRIBUTION PERFORMANCE OF MODIFICATION 2, 2A AND 2B. *This table displays the accuracy and confidence metric results for the 2A and 2B topologies across the training, validation, and test sets in an out-of-distribution context.*

Arch	Training		Validation		Test	
	Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
mod. 2	0.99199	0.98592	0.97883	0.97439	0.53191	0.75818
mod. 2A	0.98454	0.97145	0.97362	0.96276	0.53191	0.75690
mod. 2B	0.98309	0.97134	0.97479	0.96505	0.53191	0.7789

Table A.7: OPEN-SET PERFORMANCE OF IMPLEMENTED COMBINATIONS. *The table exhibits the achieved test-set CCR (Correct Classification Rate) results at various FPR (False Positive Rate) levels of 10^{-3}, 10^{-2}, 10^{-1}, and 10^{0}, along with the cumulative sum across these values, for all implemented combinations of the categorical and binary networks.*

Arch		**Loss**	1e-3	1e-2	1e-1	1e-0	Σ
categ.	binary						
regular	regular	SoftMax	0.7764	0.9517	0.9907	0.9933	3.7121
		EOS	0.7109	0.9677	0.9872	0.9898	3.6556
		Objectosphere	0.585	0.968	0.9871	0.9895	3.5296
regular	mod. 2B	SoftMax	0.7139	0.9486	0.9895	0.9933	3.6453
		EOS	0.8487	0.9625	0.9857	0.9898	3.7867
		Objectosphere	0.829	0.9623	0.9855	0.9895	3.7663
mod. 2B	regular	SoftMax	0.7571	0.9518	0.9909	0.9934	3.6932
		EOS	0.706	0.9672	0.9878	0.9904	3.6514
		Objectosphere	0.6572	0.9652	0.9866	0.989	3.598
mod. 2B	mod. 2B	SoftMax	0.7781	0.9478	0.9896	0.9934	3.7089
		EOS	0.8479	0.9619	0.9863	0.9904	3.7865
		Objectosphere	0.8389	0.9592	0.9849	0.989	3.772
mod. 2A	regular	SoftMax	0.7751	0.9517	0.9896	0.9922	3.7086
		EOS	0.6131	0.9674	0.9862	0.9883	3.555
		Objectosphere	0.6248	0.9641	0.9855	0.9876	3.562
mod. 2A	mod. 2B	SoftMax	0.8063	0.9481	0.9881	0.9922	3.7347
		EOS	0.8333	0.9621	0.9847	0.9883	3.7684
		Objectosphere	0.8313	0.9594	0.9839	0.9876	3.7622

Table A.8: PERFORMANCE OF THE MIXED SMALLSCALE-CNN NETWORK TOPOLOGIES. *This table displays the accuracy and confidence metric results for mixed topologies across the training, validation, and test sets.*

Arch	**Loss**	Training		Validation		Test	
		Acc	**Conf**	**Acc**	**Conf**	**Acc**	**Conf**
regular	SoftMax	0.98458	0.66629	0.9857	0.68007	0.9857	0.65244
	EOS	0.99323	0.93345	0.9891	0.92902	0.9891	0.79308
	Objectosphere	0.99233	0.93401	0.9883	0.92945	0.9883	0.79033
mod. 2	SoftMax	0.98483	0.67052	0.9858	0.68363	0.9858	0.65448
	EOS	0.9943	0.93777	0.9894	0.9314	0.9894	0.79544
	Objectosphere	0.99395	0.93731	0.9892	0.93033	0.9892	0.7965
mod. 2A	SoftMax	0.97825	0.67313	0.97880	0.68239	0.97880	0.65753
	EOS	0.98652	0.92074	0.98280	0.91657	0.98280	0.79507
	Objectosphere	0.98650	0.92415	0.98120	0.91961	0.98120	0.79020
mod. 2B	SoftMax	0.98482	0.67095	0.986	0.68502	0.986	0.6551
	EOS	0.99423	0.94017	0.989	0.93268	0.989	0.79449
	Objectosphere	0.99387	0.93879	0.9881	0.93272	0.9881	0.79708

Table A.9: OPEN-SET PERFORMANCE OF THE MIXED SMALLSCALE-CNN NETWORK TOPOLOGIES. *The table exhibits the achieved test-set CCR (Correct Classification Rate) results at various FPR (False Positive Rate) levels of 10^{-3}, 10^{-2}, 10^{-1}, and 10^{0}, along with the cumulative sum across these values, for all implemented mixed Network Topologies.*

Arch	**Loss**	1e-3	1e-2	1e-1	1e-0	Σ
regular	SoftMax	0.5845	0.8439	0.963	0.9857	3.3771
	EOS	0.7984	0.9675	0.986	0.9891	3.741
	Objectosphere	0.7388	0.9651	0.9848	0.9883	3.677
mod. 2	SoftMax	0.591	0.8485	0.9632	0.9858	3.3885
	EOS	0.8416	0.9667	0.9864	0.9894	3.7841
	Objectosphere	0.8703	0.9684	0.9862	0.9892	3.8141
mod. 2A	SoftMax	0.5833	0.8102	0.9535	0.9788	3.3258
	EOS	0.8209	0.9484	0.9761	0.9828	3.7282
	Objectosphere	0.8423	0.9432	0.9762	0.9812	3.7429
mod. 2B	SoftMax	0.593	0.8459	0.9634	0.986	3.3883
	EOS	0.8642	0.9668	0.9866	0.989	3.8066
	Objectosphere	0.9058	0.969	0.9851	0.9881	3.848

Table A.10: EXTRACTED MAGNITUDES OF THE CATEGORICAL NETWORKS. *This table presents the resulting extracted magnitudes of the various categorical networks. In the different columns, we have provided the mean and corresponding standard deviation for both the known and unknown categories on the validation and test sets, respectively.*

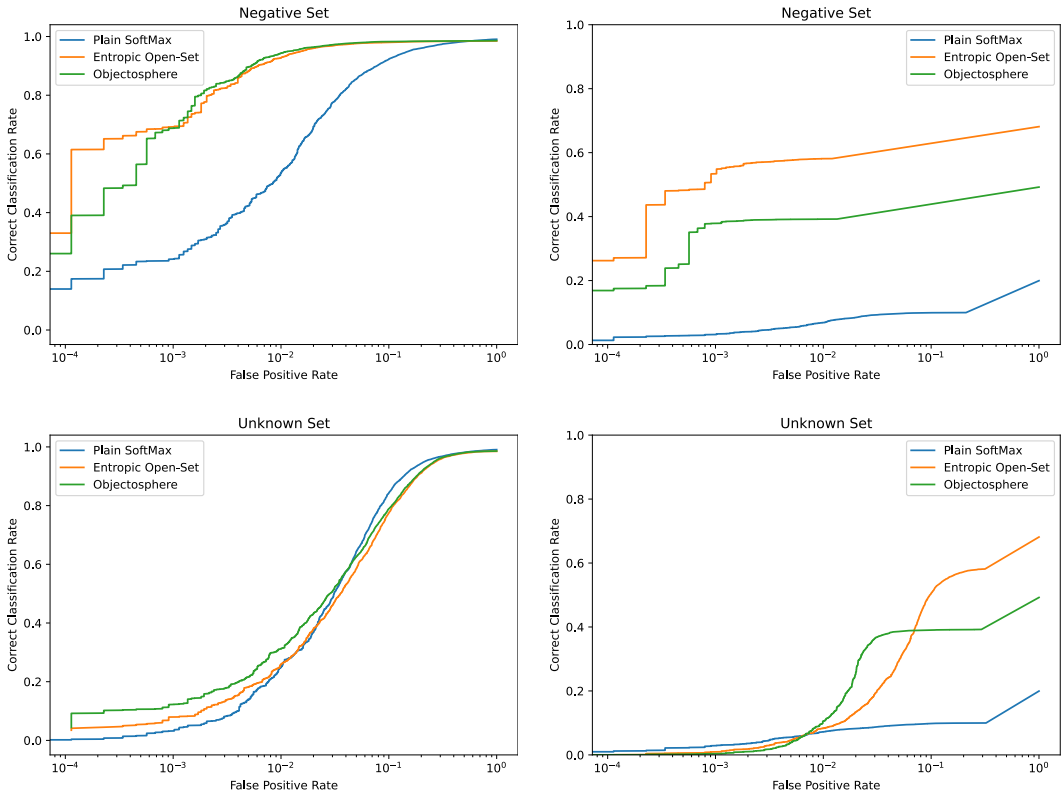
Arch	**Loss**	D_c		D_b		D_a	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
regular	SoftMax	2.197	0.641	2.239	0.604	2.217	0.596
	EOS	0.035	0.010	0.023	0.006	0.027	0.009
	Objectosphere	0.456	0.122	0.326	0.082	0.385	0.110
mod. 1A	SoftMax	5.629	1.731	5.573	1.625	5.534	1.603
	EOS	14.038	3.827	13.771	3.443	13.836	3.508
	Objectosphere	34.680	9.184	33.989	8.504	34.193	8.565
mod. 1B	SoftMax	3.545	1.046	3.500	0.988	3.480	0.974
	EOS	10.840	2.909	10.644	2.656	10.661	2.707
	Objectosphere	22.001	5.626	21.308	5.233	21.562	5.367
mod. 2	SoftMax	4.106	1.143	4.138	1.085	4.101	1.077
	EOS	0.105	0.030	0.077	0.020	0.086	0.026
	Objectosphere	0.595	0.336	0.142	0.127	0.343	0.260
mod. 2A	SoftMax	4.978	1.305	4.953	1.230	4.917	1.239
	EOS	0.127	0.037	0.089	0.027	0.103	0.033
	Objectosphere	0.653	0.336	0.162	0.146	0.368	0.275
mod. 2B	SoftMax	4.792	1.218	4.806	1.158	4.796	1.139
	EOS	0.113	0.030	0.080	0.020	0.092	0.028
	Objectosphere	0.559	0.295	0.154	0.128	0.340	0.255

Table A.11: EXTRACTED MAGNITUDES OF THE OUT-OF-DISTRIBUTION NETWORKS. *This table presents the resulting extracted magnitudes of the various binary networks. In the different columns, we have provided the mean and corresponding standard deviation for both the known and unknown categories on the validation and test sets, respectively.*

Arch	D_c		D_b		D_a	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
regular	0.154	0.031	0.157	0.035	0.153	0.035
mod. 1A	6.254	1.319	6.137	1.324	6.161	1.323
mod. 1B	4.032	0.866	3.982	0.837	3.974	0.854
mod. 2	0.849	0.145	0.818	0.177	0.832	0.167
mod. 2A	0.012	0.007	0.005	0.003	0.006	0.005
mod. 2B	0.404	0.072	0.391	0.096	0.372	0.083

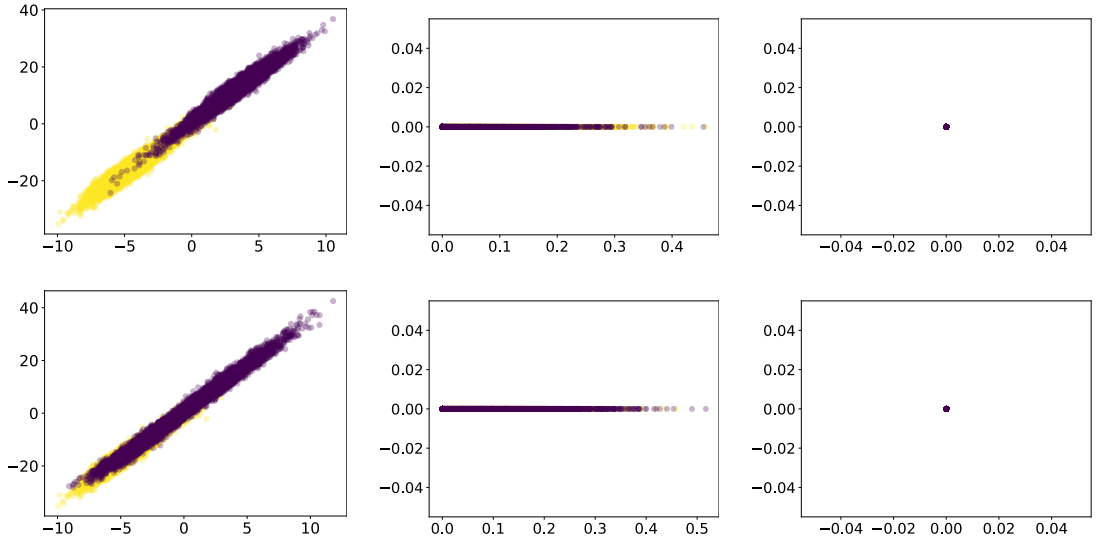
Table A.12: EXTRACTED MAGNITUDES OF THE MIXED NETWORKS.. *This table presents the resulting extracted magnitudes of the various mixed networks. In the different columns, we have provided the mean and corresponding standard deviation for both the known and unknown categories on the validation and test sets, respectively.*

Arch	**Loss**	D_c		D_b		D_a	
		Mean	**Std**	**Mean**	Std	**Mean**	**Std**
regular	SoftMax	0.315	0.074	0.306	0.073	0.310	0.072
	EOS	0.094	0.022	0.076	0.017	0.082	0.020
	Objectosphere	0.374	0.100	0.285	0.065	0.320	0.090
mod. 2	SoftMax	0.505	0.123	0.497	0.122	0.500	0.120
	EOS	0.309	0.067	0.266	0.055	0.280	0.064
	Objectosphere	0.558	0.255	0.204	0.116	0.343	0.194
mod. 2A	SoftMax	0.273	0.097	0.256	0.075	0.254	0.081
	EOS	0.085	0.029	0.051	0.016	0.06	0.022
	Objectosphere	0.46	0.203	0.203	0.111	0.273	0.145
mod. 2B	SoftMax	0.498	0.122	0.493	0.122	0.495	0.120
	EOS	0.308	0.085	0.245	0.056	0.261	0.074
	Objectosphere	0.541	0.229	0.211	0.114	0.344	0.183



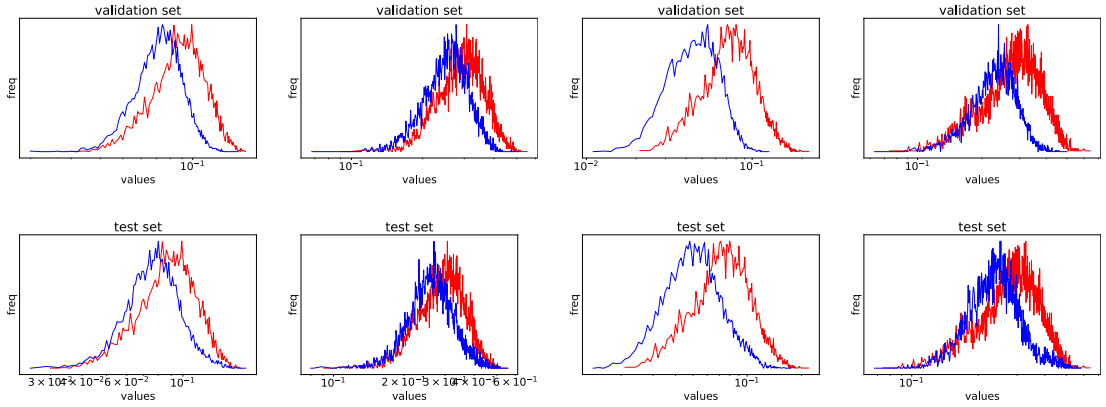
(a) regular (b) mod. 1A

Figure A.1: LENET++ OSCR CURVES. *This figure presents the OSCR (Open-Set Classification Rate) curve for the regular architecture (a) and modification 1A (b). The first row represents the validation set, while the second row depicts the test set.*



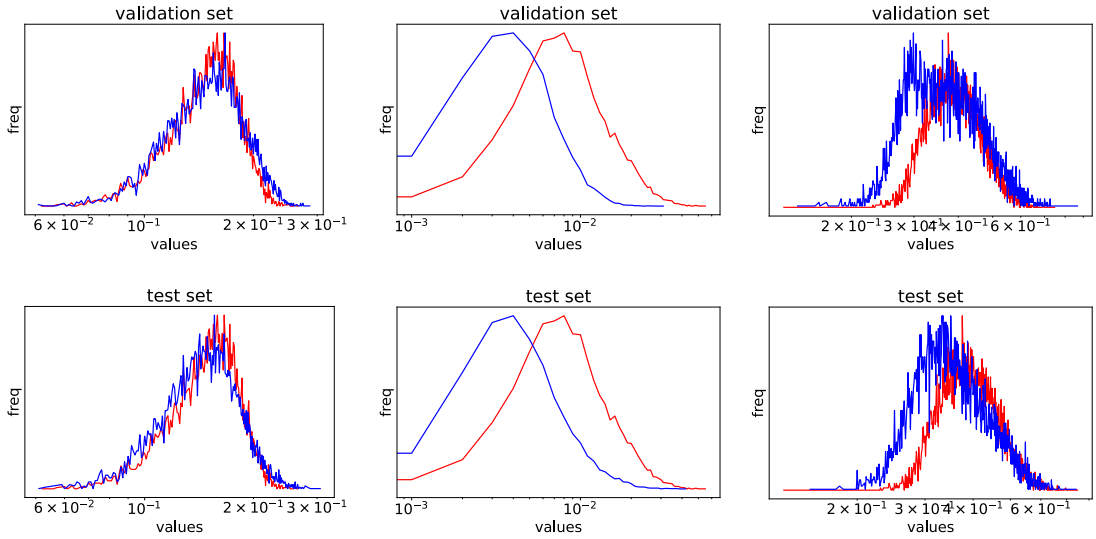
(a) regular (b) mod. 1A (c) mod. 1B

Figure A.2: LeNet++ Binary Classification Feature-Plot. *This figure illustrates the extracted features for the regular topology (a), modification 1A (b), and modification 1B (c) of the two-dimensional LeNet++ with one output node trained with the BCEWithLogitsLoss function. Feature pairs attributed to the known or unknown classes are represented in distinct colors. The first row represents the validation set, while the second row depicts the test set.*



(a) regular (b) mod. 2 (c) mod. 2A (d) mod. 2B

Figure A.3: Magnitude Plots for the mixed SmallScale-CNN with EOS Loss Function. *This figure presents the extracted magnitudes for the regular architecture (a), modification 2 (b), modification 2A (c), and modification 2B (d) after being trained with the EOS loss function. We graph the resulting magnitudes extracted from the unknown input in blue, and from the known inputs in red, against the frequencies at which they occurred across all inputs in the set. The first row represents the resulting magnitudes of the validation set, while the second row depicts the resulting magnitudes from the test set.*



(a) regular (b) mod. 2A (c) mod. 2B

Figure A.4: MAGNITUDE PLOTS FOR THE OUT-OF-DISTRIBUTION CONTEXT. *This figure presents the extracted magnitudes for the regular architecture (a), modification 2A (b), and modification 2B (c) for the Out-of-Distribution binary Network. We graph the resulting magnitudes extracted from the unknown input in blue, and from the known inputs in red, against the frequencies at which they occurred across all inputs in the set. The first row represents the resulting magnitudes of the validation set, while the second row depicts the resulting magnitudes from the test set.*

List of Figures

List of Tables

List of Algorithms

Bibliography

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.

Bisgin, H., Palechor, A., Suter, M., and Günther, M. (2023). Large-scale evaluation of open-set image classification techniques. In *Journal of Machine Learning Research (JMLR)*. Under Submission.

Boult, T. E., Cruz, S., Dhamija, A., Gunther, M., Henrydoss, J., and Scheirer, W. (2019). Learning and the unknown: Surveying steps toward open world recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9801–9807.

Brydges, C. R. (2019). Effect Size Guidelines, Sample Size Calculations, and Statistical Power in Gerontology. *Innovation in Aging*, 3(4).

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3).

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

Cui, P. and Wang, J. (2022). Out-of-distribution (OOD) detection based on deep learning: A review. *Electronics*, 11(21).

Dhamija, A. R., Günther, M., and Boult, T. (2018). Reducing Network Agnostophobia. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Everingham, M., van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2010). The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Gao, L. and Wu, S. (2020). Response score of deep learning for out-of-distribution sample detection of medical images. *Journal of Biomedical Informatics*, 107:103442.

Gillert, A. and von Lukas, U. F. (2021). Towards combined open set recognition and out-of-distribution detection for fine-grained classification. In *VISIGRAPP (5: VISAPP)*, pages 225–233.

Girshick, R. (2015). Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grother, P. and Hanaoka, K. (2016). NIST special database 19 handprinted forms and characters 2nd edition.

Hodge, V. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. *University of Toronto*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Mahdavi, A. and Carvalho, M. (2021). A survey on open set recognition. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE.

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

Mårtensson, G., Ferreira, D., Granberg, T., Cavallin, L., Oppedal, K., Padovani, A., Rektorova, I., Bonanni, L., Pardini, M., Kramberger, M. G., Taylor, J.-P., Hort, J., Snædal, J., Kulisevsky, J., Blanc, F., Antonini, A., Mecocci, P., Vellas, B., Tsolaki, M., Kłoszewska, I., Soininen, H., Lovestone, S., Simmons, A., Aarsland, D., and Westman, E. (2020). The reliability of a deep learning model in clinical out-of-distribution MRI data: A multicohort study. *Medical Image Analysis*, 66:101714.

Palechor, A., Bhoumik, A., and Gunther, M. (2023). Large-scale open-set classification protocols for imagenet. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 42–51, Los Alamitos, CA, USA. IEEE Computer Society.

Pant, A. K., Panday, S. P., and Joshi, S. R. (2012). Off-line nepali handwritten character recognition using multilayer perceptron and radial basis function neural networks. In *2012 Third Asian Himalayas International Conference on Internet*, pages 1–5.

Pimentel, M., Clifton, D., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99:215–249.

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.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Skansi, S. (2018). *Introduction to Deep Learning - From Logical Calculus to Artificial Intelligence*. Springer, Berlin, Heidelberg.

Van den Bergh, L. (2023). Improved losses for open-set classification. Master's thesis, University of Zurich.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 499–515, Cham. Springer International Publishing.

Yang, J., Zhou, K., Li, Y., and Liu, Z. (2022). Generalized out-of-distribution detection: A survey.