

Open-Set Classification with Ensembles of Binary Classifiers

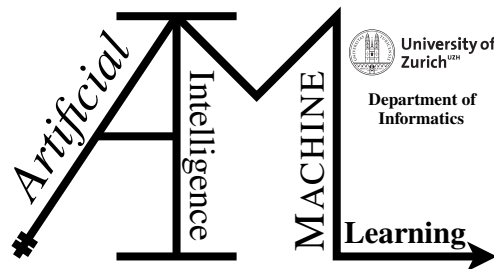
Master Thesis

Silvan Kübler

19-752-906

Submitted on
August 7 2024

Thesis Supervisor
Prof. Dr. Manuel Günther

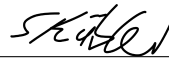


Declaration of Independence for Written Work

I hereby declare that I have **composed** this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT) – the use of generative AI to **improve** my composed work was permitted by the thesis supervisor. I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used. All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zurich, 07.08.2024

Place, Date



Silvan Kübler

Master Thesis

Author: Silvan Kübler, silvan.kuebler@uzh.ch

Project period: 07.02.2024 - 07.08.2024

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

Acknowledgements

I want to thank Prof. Dr. Manuel Günther for supervising this thesis. His constructive feedback, guidance, and expertise have been precious throughout all our discussions. Furthermore, I want to thank the AIML group for providing me with the computational infrastructure that was needed to conduct the experiments. Finally, I would like to thank my family and friends for their unconditional support.

Abstract

For many years, deep neural networks have achieved astonishing performance in image classification. These networks work very well when the classes during training and testing are the same. However, during deployment in the real world, often samples of previously unseen classes occur and should then be rejected. In contrast, a sample of a class seen before should still be classified correctly. This is what open-set classification is about, which artificially simulates such a real-world scenario. Traditional approaches to open-set classification are normally based on a single network that outputs probabilities over all classes. In this thesis, we take a different path and use ensembles of binary classifiers to tackle the open-set task. The multiclass classification problem is split into many binary classification problems, which are then learned separately and aggregated in the end to get a classification result. We cover the whole process, from the creation of a binary problem to training binary classifiers and evaluating them appropriately. The experiments are conducted on subsets of the ImageNet dataset, which try to resemble different levels of difficulty in their open-set task. We show that when using our binary ensemble approach, performance on the open-set task is on par with Softmax-based methods. When no negative samples for training are available, the binary ensemble does even perform better than a comparable Softmax-based approach. Achieving good performance is possible with a low number of binary classifiers.

List of Symbols

α	Weighting factor of focal loss
δ	Hamming distance
γ	Focusing parameter of focal loss
ϕ	Softmax function
ρ	Class similarity used with sigmoid probabilities
ρ^*	Class similarity used with logits
σ	Sigmoid function
τ_n	Label of sample n
E	Ensemble of binary classifier matrix
Y	Ensemble prediction matrix
θ	Probability threshold in OSCR
B	Total number of binary classifiers in E
B_{max}	Maximum possible number of binary classifiers for C
B_{min}	Minimum possible number of binary classifiers for C
C	Total number of classes
K	Total Number of known classes
k	Dimensionality of the separate fully connected layer of the combined+ approach
N	Number of samples in current batch
N_K	Total number of known test samples
N_U	Total number of negative/unknown test samples
$o_{m,n}$	Output of convolution at position m, n
t_n	Target label of the n th sample for the positive class
$t_{n,c}$	Target label of the n th sample for class c

$w_{i,j}$	Convolution kernel weight at position i, j
x	Input sample
y_n	Probability of the n th sample for the positive class
$y_{n,c}$	Probability of the n th sample for class c
z	Logits (model outputs before activation)

Contents

1	Introduction	1
2	Related Work	5
2.1	Open-Set Classification	5
2.2	Binary Classifiers	7
2.3	Ensemble Learning	7
2.4	Convolutional Neural Networks	8
3	Background	11
3.1	Activation and Loss Functions for Classification	11
3.1.1	Softmax Activation and Cross-Entropy Loss	11
3.1.2	Sigmoid Activation and Binary Cross-Entropy Loss	12
3.1.3	Focal Loss	12
3.2	Training with Negatives	13
3.3	Open-Set Classification Rate (OSCR)	13
4	Data	15
4.1	MNIST & EMNIST	15
4.2	ImageNet Open-Set Classification Protocols	16
5	Approach	19
5.1	Creating a Binary Classification Problem	19
5.1.1	Random Partitioning of Classes	19
5.1.2	Partitioning with maximizing Hamming Distance among Classes	21
5.2	Separate and Combined Binary Classifiers	22
5.3	Obtaining a Classification Score from the Ensemble Model Output	24
5.3.1	Training a Binary Ensemble with Negative Samples	25
6	Experiments and Results	29
6.1	Neural Networks	29
6.2	Preliminary Experiments on EMNIST	29
6.2.1	Optimal number of Classifiers and maximizing Distance between Classes	30
6.2.2	Separate vs. Combined Binary Classifiers	31
6.2.3	Evaluation Approaches	32
6.2.4	Training with Negatives	33
6.3	Experiments on ImageNet	34
6.3.1	Optimal Number of Classifiers	34
6.3.2	Combined+ vs. Combined Approach	35

6.3.3	Binary Ensemble on all three Protocols	36
7	Discussion	39
7.1	Creation of a Binary Classification Problem	39
7.2	Analysis of High-Confidence Misclassifications and Rare Errors	40
7.3	Analysis of Score Distributions	45
7.4	Limitations	46
8	Conclusion	49
8.1	Summary	49
8.2	Future Work	50
A	Attachments	51

Introduction

Image classification is ubiquitous in our modern world, be it scanning X-rays for tumors (Koh et al., 2022; Kaur and Garg, 2023), telling apart crops from weeds in agriculture for modern weed destruction (Rasti et al., 2019; Wang et al., 2019), or in video surveillance for real-time detection of suspicious behavior (Verma et al., 2022). This is made possible by the massive increase in performance of modern graphics processing units (GPUs), and the large availability of stored data linked with the invention of deep learning systems that can process these vast amounts of data. These advancements coupled with declining prices for hardware have accelerated AI development significantly.

The invention of convolutional neural networks (CNNs) which can learn feature maps independent of their position in the image brought huge improvements to image classification tasks (LeCun et al., 1995). CNNs learn feature maps, which are then followed by fully connected layers outputting a probability distribution over all possible classes. The maximum probability over all classes is the predicted class of the network. This class is then compared to the ground truth label during training. The resulting discrepancy between the probability distribution of the model's prediction and the ground truth label is quantified using a loss function, such as Categorical Cross-Entropy (CCE) loss. Depending on this loss the model's parameters are updated using backpropagation to achieve a smaller loss on the next run of training images that pass through the network. Up to this day, CNNs have proven to be very successful by achieving high accuracy in image classification tasks on many datasets ranging from small ones like MNIST¹, CIFAR-10², or CIFAR-100² to large ones such as ImageNet1K³. The current state-of-the-art approaches often use a transformer architecture to achieve even higher accuracies (Yuan et al., 2021).

These achievements in accuracy are remarkable, but the models are normally tested only on classes already seen during the model's training, which is not representative of a deployment scenario where there could also be a sample of a class not seen during training. Dhamija et al. (2018) show that when a letter is presented to a network trained on the digits of the MNIST dataset, the network predicts one of the learned digits. This can be very bad in reality, and depending on the use case, the system should be able to handle unseen samples correctly. In the field of Open-Set Classification (OSC), the model can classify samples from classes that have not been seen before as unknown and thus reject them (Palechor et al., 2023). Multiple approaches have evolved over the last decade to tackle OSC. They can be separated into post-processing and network-based approaches. The former are methods applied upon the deep features of trained networks, such as Maximum Softmax Scores, which thresholds the probability output from Softmax, and everything below that threshold is classified as unknown (Hendrycks and Gimpel, 2017). Another approach is OpenMax which introduces an alternative to the Softmax function that can model the probabil-

¹<https://yann.lecun.com/exdb/mnist/>

²<https://www.cs.toronto.edu/~kriz/cifar.html>

³<https://www.image-net.org>

ity for unknown (Bendale and Boult, 2016). A newer post-processing approach is PROSER, which fine-tunes an existing network to the open-set task by learning data and classifier placeholders for the unknown samples (Zhou et al., 2021).

On the other hand, network-based approaches modify the network in a way that it has the option to classify a sample as unknown (Bisgin et al., 2024). One of the oldest methods is the introduction of an additional garbage class as output of the network, which is then used as the target class for all unknown samples. This approach has been used many times by researchers (Zhou et al., 2021; Chen et al., 2021). Another approach is the Entropic Open-Set (EOS) loss, which avoids producing high probabilities for unknown samples whilst not explicitly outputting a probability for the unknown (Dhamija et al., 2018). All those methods have in common that they normally make use of a model that outputs some sort of Softmax probabilities. Moreover, they need to define training samples for the unknown class, which can be difficult to obtain. Alternative approaches are available but have not yet been explored in the domain of OSC.

In the domain of open-set face recognition, the model learns an incomplete set of faces and should thereafter also be able to handle unknown faces by dismissing them as unknown and hereby focus only on the relevant faces (Vareto et al., 2023). Vareto and Schwartz (2020) and Vareto et al. (2023) take a different approach to the open-set face recognition problem using an ensemble of binary classifiers in which each classifier makes a binary prediction for a subset, and the combination of these predictions yields the final scores for each face in the gallery. The face in the gallery with the highest score is the model’s prediction if it is above a certain threshold, otherwise, the sample is classified as unknown. The binary models can be trained separately as unique networks, or one can also train one model that consists of multiple binary classifiers. Rudd et al. (2016) show in the domain of facial attribute recognition that their mixed objective optimization network (MOON) even outperforms separate binary classifiers, while being a single network trained with multiple binary classifiers. These are promising results for open-set classification which should be explored.

The main motivation behind this thesis is applying the ensemble of binary classifier approach to open-set image classification problems, with the hope that it performs better than the classic Softmax-based approaches. However, using such a binary ensemble for open-set classification is not straightforward and does bring quite some challenges with it such as the decomposition of a multiclass problem into a binary one, the evaluation of the binary predictions, and rejecting before unseen samples. Four research questions guide this objective.

RQ 1: How can binary classifiers be used for open-set classification?

This research question investigates the applicability of established binary classification techniques to the domain of open-set image classification. In detail, two sub-questions are tackled, which further refine RQ 1:

RQ 1.1: Does maximizing the Hamming distance between classes improve the classification performance of the binary ensemble classifier compared to a random approach?

RQ 1.2: Do more binary classifiers lead to an improved classification performance?

RQ 1.1 addresses the decomposition of a multiclass problem into a binary one. In our experiments, we work with datasets that consist of many more than only two classes, and as such we need methods to obtain a binary classification problem while still being able to do multiclass classification in the end. For this, we use an ensemble of binary classifiers where each classifier learns a unique binary classification problem in which every class of the original dataset is either assigned to partition 0 or 1. To form those partitions we use a completely random approach and another one that maximizes the Hamming distance between the original classes, each defined through a binary vector which results from all the binary classifiers in the ensemble. RQ 1.2

explores if more binary classifiers give better results than fewer ones. Furthermore, the second research question explores how this impacts open-set performance.

RQ 2: How does using binary classifiers impact open-set performance?

Two sub-questions are explored, considering if one or multiple binary classifier models should be used, and which evaluation approach to obtain a score from the model is the best regarding performance. In the *combined* approach we do have a single neural network that contains multiple binary classifiers that rely on the same deep features, whereas in the *separate* approach, one complete network is used per binary classifier. In RQ 4.2 we introduce the *combined+* approach, which extends the combined approach through an individual fully connected outer layer per binary classifier. This allows the model to build upon shared deep features while still given the possibility to adapt highly to its unique binary classification task.

RQ 2.1: Does a combined or a separate binary classifier perform better?

RQ 2.2: Does the evaluation approach using probabilities outperform the simpler threshold and the unbounded logits approach?

RQ 3 investigates the impact of incorporating negative samples within the training process on the performance of an ensemble of binary classifiers.

RQ 3: How much do negative samples during training improve the binary ensemble?

This research question is divided into the following sub-questions:

RQ 3.1: Which approach results in better performance when training with negatives: training negatives as an extra class or with an equal approach that integrates them in both sets of the binary classifiers to learn a balanced output of 0.5?

The RQ 3.1 does examine if negatives should just be integrated like another class or if the network should be trained to output the prediction in a way that the value lies exactly between the two sets that should be learned. To address the problem of easy-vs-hard to classify samples, RQ 3.2 explores if using Focal Loss does bring improved performance compared over using Binary Cross Entropy loss.

RQ 3.2: Does using Focal Loss improve performance of the binary ensemble compared to just using BCE?

Research questions 1-3 are mostly examined in open-set experiments on the EMNIST dataset. The discoveries are then transferred to the ImageNet dataset to test if the findings also hold in such a large-scale case. This is investigated with research question 4.

RQ 4: How much is performance increased by using binary classifiers on open-set ImageNet protocols compared to Softmax-based approaches?

RQ 4.1: How does the optimal number of binary classifiers change concerning the Correct Classification Rate at a False Positive Rate (CCR@FPR) on Protocol 2?

RQ 4.2 Is the combined+ binary classifier approach superior to the combined one in terms of performance on Protocol 2?

RQ 4.3: Does using an ensemble of binary classifiers increase performance on Protocols 1, 2, and 3 compared to Softmax-based approaches?

Related Work

This chapter highlights the related work of open-set classification, binary classifiers, ensemble training, and gives an overview over Convolutional Neural Networks (CNNs).

2.1 Open-Set Classification

Image classification has a long history in machine learning. [LeCun et al. \(1995\)](#) presented convolutional neural networks (CNNs) as a novel approach to classify images. With CNNs, the accuracy for many image classification tasks such as classifying handwritten digits on the MNIST¹ dataset has been over 99% for more than 20 years. Even on larger datasets such as ImageNet, the classification accuracy over 1000 different classes is above 85% ([Liu et al., 2021](#)). These high accuracy scores are indeed an achievement in AI research. However, they only tell half of the story, as they only work with a clear number of finite classes. That does not reflect the real world and can be seen in the WordNet database, which currently contains over 100'000 classes ([Miller, 1998](#)). Moreover, most research is done on *closed-set* classification, where all classes are known and trained when creating the network ([Scheirer et al., 2012](#)). These networks operate under a closed-world assumption, presenting an oversimplified model of our world. In contrast, [Scheirer et al. \(2012\)](#) show the *open-set* classification scenario in which there are classes during testing that have not been seen previously by the network, resembling a scenario that could also occur in the real world.

If a closed-set network is tested in an open-set scenario it will have no capabilities to say: “I do not know” and will rather assign the sample to one of the trained classes, with a most likely high confidence, which leads to a poor performance of the network ([Wilber et al., 2013](#)). These unseen classes, which might occur during testing or deployment in the real world, are often referred to as *unknowns* ([Palechor et al., 2023](#)). In the case of such an unknown class, a network, being capable of handling this open-set task, should assign a low confidence score over all known classes, therefore allowing for the rejection of unknowns through thresholding.

A long-standing approach is the assignment of an unknown sample to an additional *garbage* class, often used as a catch-all category for unrecognizable or out-of-distribution data points ([Matan et al., 1990](#)). Adding a garbage class is a network-based approach, which requires adjustment of the underlying network ([Bisgin et al., 2024](#)). Although the garbage class approach is widely represented in the literature ([Zhang et al., 2018](#); [Liu et al., 2016a](#); [Ren et al., 2015](#); [Palechor et al., 2023](#)), it seems to be inferior to other approaches of incorporating negatives during training. Due to its separate garbage class, it produces deep features with high magnitudes for negative and unknown samples and forces the garbage class to lay in its own region in the latent

¹<https://yann.lecun.com/exdb/mnist/>

space (Dhamija et al., 2018). It has been shown that a better way to threshold confidence scores of negatives is to keep their deep feature magnitudes low. Dhamija et al. (2018) state that the garbage class approach is simple yet effective on many small datasets used in academia such as PASCAL and MS-COCO. However, on more “truly” open-set datasets such as the protocols 1-3 on ImageNet by Palechor et al. (2023) the garbage class approach shows its limitations in rejecting unknown samples.

Post-processing approaches also exist that use a closed-set network and add capabilities for open-set classification. One such approach is thresholding the Maximum Softmax Score (MSS) to deal with unknowns (Hendrycks and Gimpel, 2017). In MSS, if the highest probability from the Softmax function falls below a threshold, the output prediction is the unknown class. MSS assumes that for an unknown sample, the Softmax scores are distributed evenly with no class having a very high score and therefore all scores are below a certain threshold (Dhamija et al., 2018). However, the Softmax function is prone to producing high probability scores due to its use of the exponential function, which grows rapidly, and therefore it is uncommon to see a uniform distribution of the scores for an unknown sample (Hendrycks and Gimpel, 2017). In neural networks, logits are the raw output values before applying an activation function such as Softmax. They represent the unnormalized predictions of the model. Another related issue is that the Softmax function squashes the logit inputs to values between 0 and 1, which can result in information loss and prevents intra-class separability (Liu et al., 2016b; Hinton et al., 2015). This is why Hendrycks et al. (2022) introduced the MaxLogit approach, which uses the logits directly instead of putting them through Softmax first.

On top of that, many more approaches exist, which are not further discussed in this chapter. The network-based approaches do require training with *negative* data or also named *known unknown* data, which is not part of any known class (Bisgin et al., 2024). For sourcing negative data various approaches do exist: such as using genuine negative samples sourced from various datasets (Dhamija et al., 2018; Palechor et al., 2023), creating negative samples through the alteration of existing ones (Zhou et al., 2021), or the artificial generation of negative samples (Ge et al., 2017).

The open-set scenario is challenging for current networks, and there is still a lot of room for improvement (Bisgin et al., 2024). Palechor et al. (2023) show that even the evaluation of open-set classification networks poses quite a challenge and is often done wrong among researchers. Mainly, unsuitable metrics are used for measuring the performance of the networks, which do not take into account the specialities of open-set classification.

In many classification tasks, very high accuracy is often hardly possible due to hard-to-classify samples that result from class overlap or bad samples (Metzner et al., 2022). This is where open-set methods could benefit closed-set tasks by helping to identify when a sample is too ambiguous for a confident classification to one of the known classes (Panareda Busto and Gall, 2017).

Out-of-distribution detection is closely related to open-set classification. While both share some similarities they are still treated mostly as different areas of research. Open-set classification is normally a multi-class classification problem where unknowns can occur and should be rejected (Palechor et al., 2023). Out-of-distribution detection is usually a binary classification problem where each sample is classified as in- or out-of-distribution. The in-distribution samples are not further assigned a specific class (Hendrycks and Gimpel, 2017). Detecting whether the data is shifted a lot from the data distribution used during training is often a safety measure that ensures reliable predictions and prevents a model from operating outside its area of expertise (Ren et al., 2019).

Even with the differences between open-set classification and out-of-distribution detection, both have the common goal of making networks more robust and trustworthy with real-world data. Advances in one of these two areas can often be transferred to the other, thus advancing both (Vaze et al., 2022; Cao et al., 2021; Scheirer et al., 2014).

2.2 Binary Classifiers

Classification tasks find broad application in real-world scenarios, with a significant portion encompassing more than two classes, commonly referred to as multi-class problems (Galar et al., 2011). Such problems are present in many fields, naming only a few: sign-language detection (Aran and Akarun, 2010), cancer histology classification (Araújo et al., 2017), and quality control. Although multi-class problems are omnipresent, many times it is simpler to build a classifier that only has to distinguish between two classes, because the decision boundaries are often easier to learn (Lan et al., 2022). This *binary classification* can be applied when only two classes should be distinguished, such as dogs vs. cats. In most cases, however, there are more than two classes present. Therefore, decomposition strategies for creating a binary classification problem exist.

One such strategy is *one-vs-one* which divides the classification problem into as many binary problems as there are possible combinations between pairs of classes (Knerr et al., 1990). To get the classified output, all binary classifications are considered together and the class which is classified the most is the model's output (Galar et al., 2011).

Clark and Boswell (1991) describe the *one-vs-all* decomposition strategy, which is also well-known. In this strategy, one classifier is learned for each class, which learns to distinguish the class from all other classes. The single class that is detected in this *one-vs-all* approach is the result of the classification. It is important to notice that this strategy is very similar to Softmax, with the main difference being that Softmax learns all binary classifiers simultaneously, whereas in the described *one-vs-all* strategy, the classifiers are learned separately (Pawara et al., 2020).

2.3 Ensemble Learning

Ensemble learning techniques use multiple machine learning algorithms to generate predictive outcomes. These methods combine the outcomes using various voting mechanisms, resulting in superior performance compared to any individual algorithm used in isolation (Zhou, 2012).

The *one-vs-one* approach from Section 2.2 can also be extended to sets. One cannot only compare *one-vs-one* class but also a *set-vs-set*, whereby a set consists of multiple classes. However, not much research is conducted on set-vs-set classification. One of the few papers touching upon *set-vs-set* classification is from Vareto and Schwartz (2020) in the domain of open-set face identification. They employ *Affinity Propagation Clustering* (APC) to cluster the data (Frey and Dueck, 2007). In the paper of Vareto and Schwartz (2020) when a query sample is presented, the APC algorithm selects the most similar clusters and creates a training subset. During training, subjects from this subset are randomly divided into positive and negative subsets for training the ensemble models. During testing, the query sample is presented to the ensemble to obtain response values, which are then used to rank the samples with a simple voting procedure. Afterward, thresholding is applied to determine whether the probe sample belongs to the face training set or should be rejected as unknown.

Bagging (bootstrap aggregation) was first presented by Breiman (1996) and is one of the most commonly used ensemble learning approaches. The *bagging* algorithm constructs sample subsets by randomly selecting from the training dataset. These subsets are then employed to train individual base models of the same type for integration. Afterward, a simple majority vote is taken from the individual decisions which results in the most chosen class being the ensemble decision.

Gradient Boosting, introduced by Friedman (2002), uses an ensemble learning technique that sequentially adds weak learners to the model, each correcting the errors of its predecessor. At each iteration, the algorithm fits a new model to the residual errors of the ensemble, with a focus on minimizing a specified loss function. This iterative process continues until a predetermined

number of weak learners, often decision trees, are included or until the model's performance converges.

AdaBoost is yet another ensemble learning technique, which was introduced by [Freund and Schapire \(1997\)](#). It works by iteratively training a sequence of weak learners where each subsequent learner focuses more on the instances that the previous ones classified incorrectly. During training, AdaBoost assigns higher weights to misclassified instances. AdaBoost produces a strong classifier by combining the predictions of these weak learners. This approach effectively emphasizes the “hard” examples in the training data, improving the overall performance of the ensemble.

Mixture of Experts (MoE), presented by [Jacobs et al. \(1991\)](#), is an ensemble method that uses multiple separate networks trained simultaneously. In this approach, each network, known as an “expert”, specializes in a particular subset or region of the input space. The outputs of these individual networks are then combined, typically using a gating mechanism, to produce the final prediction. Unlike many ensemble methods that use fixed weights determined during training for their models, MoE dynamically assigns weights to its expert models based on each specific input, allowing the system to adapt its predictions to different types of data. This allows MoE to adaptively select the most relevant experts for a given input, improving prediction accuracy. Additionally, MoE can handle complex relationships within the data by effectively partitioning the input space and leveraging the strengths of different experts.

2.4 Convolutional Neural Networks

CNNs are inspired by the visual nervous system found in vertebrates, which can detect patterns based on geometrical similarity without being affected by their position in the visual field ([Fukushima, 1980](#)). [Hubel and Wiesel \(1959\)](#) discovered that in cats, when visual stimuli are present, the firing of neurons in the visual cortex is affected in a certain area known as the receptive field of these stimuli. According to [Lecun et al. \(1998\)](#), an early researcher to employ CNNs, local receptive fields are one of the main concepts on which CNNs are based. These receptive fields allow the network to extract elementary visual features such as edges, endpoints, or corners, which are aggregated in the later layers of the CNN to detect higher-order features. This operation, named convolution, passes a kernel over the input that does element-wise multiplication between the kernel and the input. A remarkable property of this feature detection operation is that it is independent of the location of the feature in the entire input. Formally, the convolution operation with an element of the input image x and an element of the convolutional kernel w with the dimensions I and J is defined as:

$$o_{m,n} = \sum_i^I \sum_j^J x_{m+i-1,n+j-1} \cdot w_{i,j} \quad (2.1)$$

$o_{m,n}$ is the element of the output matrix obtained by element-wise multiplication between w and x .

According to [Goodfellow et al. \(2016\)](#) convolution leverages three main ideas that make it so powerful. Sparse interaction is one of those ideas and describes the property achieved by using a kernel that is smaller in size than the input image, which allows the convolution to detect small and meaningful features in the input image, which might only span over a small number of pixels. The second idea is parameter sharing, which describes that each element of the kernel is used at almost every position in the input, except maybe the pixels on the edge. So instead of learning a different set of parameters for every input location, only one set has to be learned. This is quite different from a fully connected network, where each weight is only used once for computing the

output of a layer. The last idea is equivariance to translation, which in simple terms says that if the input changes the output changes in the same manner. This is a useful property because the convolution creates a 2-D feature map of where specific features appear in the input. If the object in the input is now moved, the feature in the feature map shifts the same amount. Goodfellow et al. (2016) mention that the typical convolutional layer can be separated into three stages. The first stage, also named the convolution stage, performs multiple convolutions in parallel to output a set of linear activations. In the second stage, named the detector stage, the previously produced linear activations are run through a nonlinear activation function. Lastly, in the third stage, named the pooling stage, the output is modified further. This is achieved by using a pooling function that replaces the output at a specific location with a summary statistic of the nearby outputs (Goodfellow et al., 2016). Pooling therefore enables the net to be invariant to small translations of the input, which means that when the input is translated by a small amount most of the pooled outputs are not subject to change. In a typical CNN, the three stages described are repeated multiple times depending on how many convolutional layers are used (Goodfellow et al., 2016). When doing classification, normally the convolutional layers are followed by some fully connected layers to do the final classification (Lecun et al., 1998).

Since the inception of LeNet-5, a popular CNN consisting of seven trainable layers developed by Lecun et al. (1998), there have been significant improvements. One of these is building deeper networks, like AlexNet presented by Krizhevsky et al. (2012), which won the ImageNet visual recognition challenge in 2012², which consists of 1000 different classes. AlexNet includes a total of eight trainable layers of which five are convolutional layers and three are fully connected ones. Krizhevsky et al. (2012) state that going deeper by adding more convolutional layers is not just straightforward, and they encountered severe overfitting early on in their development of AlexNet. To mitigate this problem, they used data augmentation techniques such as image translations and horizontal reflection; further, they applied dropout with a probability of 0.5 in the first two fully connected layers of AlexNet. Dropout is a technique that switches certain neurons off, with a certain probability, during the training of the neural network (Hinton et al., 2012). This forces the network to learn more robust features and prevents the network from relying too much on some neurons. Based on the success of AlexNet research interest in CNNs sparked (Goodfellow et al., 2016). Simonyan and Zisserman (2015) presented VGGNet which used a deeper architecture and smaller kernels than most of its predecessors like AlexNet. Whereas AlexNet used a kernel of size 11×11 in its first layer, the VGGNet only used a kernel of size 3×3 throughout all its layers. In terms of depth, they presented VGGNets between 11 and 19 trainable layers and showed that generally, the classification accuracy improved for deeper VGGNets. Concluding Simonyan and Zisserman (2015) showed that large kernels can be replaced by using smaller kernels with deeper networks.

While deep networks suffer from vanishing/exploding gradients there have been solutions to this problem, such as using normalization in the initialization layers (LeCun et al., 2002; Glorot and Bengio, 2010). However, there exists another problem that causes a degradation in performance when adding more layers and therefore making the network deeper (He et al., 2016). According to He et al. (2016), there exists a constructed solution that enables deeper networks without a decrease in performance when compared to shallower counterparts. This can be achieved by taking the trained layers from the shallow network and adding additional layers, which just do an identity mapping, ensuring the input is the same as the output. However He et al. (2016) state that the solvers are unable to find solutions that are on par or better than a constructed solution. With residual learning, they allow the layers to fit a desired underlying mapping directly, instead of hoping to achieve a mapping through several layers. In mathematical terms, the residual function can be expressed as $f(x) = h(x) - x$ (He et al., 2016). This allows the layers to fit the residual mapping, which is easier than fitting the original, unreferenced mapping (He et al., 2016).

²<https://image-net.org/challenges/LSVRC/2012/results.html>

He et al. (2016) used these residual building blocks in their Residual Network (ResNet) architecture and proposed several networks with 18, 34, 50, 101, and 152 layers, while the ResNet-18 and 34 networks used a residual connection after two layers and the remaining ones one after three. Generally speaking, more layers seemed to offer better performance, while the ResNet-50 offered a good tradeoff between performance and training time.

Since the introduction of ResNets, there has been more progress made in CNNs. This is only touched upon briefly since we do use a ResNet-50 in our main experiments later on. Huang et al. (2017) presented Dense Convolutional Networks (DenseNets), which connect all layers to every other layer in a feed-forward fashion. This brings several advantages with it, such as alleviating vanishing gradients, strengthened feature propagation and reuse. With CNNs becoming more complicated and deeper, they were unsuited for mobile and embedded applications due to memory and latency bottlenecks. That's why MobileNets were introduced by Howard et al. (2017). MobileNets use depthwise separable convolutions to reduce the computational complexity while being effective across a wide variation of tasks, such as object detection and facial attribute detection. In recent years Vision Transformer (ViT) based architectures have outperformed classic CNNs on many tasks (Liu et al., 2022). However, Liu et al. (2022) showed with their ConvNeXt architecture that the ViT-based architectures can be outperformed using a combination of only classical CNN building blocks, which have not been combined in this fashion before.

Background

This chapter provides the foundational concepts required for understanding the experiments conducted in this master thesis. It covers common activation and loss functions used for image classification tasks, open-set methods to incorporate negative samples into training, and introduces the Open-Set Classification Rate (OSCR) for evaluating open-set models.

3.1 Activation and Loss Functions for Classification

In the following section, we touch upon activation and loss functions, which are widely used in AI and especially image classification tasks. The goal is to foster a general understanding of the theoretical concepts, which is required as a basis for the experiments conducted later.

3.1.1 Softmax Activation and Cross-Entropy Loss

When following a standard classification approach, the model outputs a probability distribution over the C classes, where the probabilities sum to 1, representing the likelihood that the sample belongs to each class (Goodfellow et al., 2016). The network does not directly output a probability distribution but rather a vector of logits, which needs to be converted to a vector of probabilities. This is often done by using the *Softmax* function $\phi(z_n)$ below, adapted from Goodfellow et al. (2016):

$$\phi(z_n)_c = \frac{e^{z_{n,c}}}{\sum_{c'=1}^C e^{z_{n,c'}}} \quad (3.1)$$

The vector z represents logits of size C , the number of classes. The normalization of Softmax ensures that the resulting probabilities lie in a range between 0 and 1, and collectively sum to 1, forming a valid probability distribution.

Following the generation of the probability distribution typically involves computing a loss value that represents the dissimilarity between the predicted and actual probability distribution. One such metric, that has gained popularity and is often used in classification, is the *categorical cross-entropy loss* (Goodfellow et al., 2016). Extended from Zhang et al. (2021) it is defined as follows:

$$\mathcal{L}_{CCE} = -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C t_{n,c} \log(y_{n,c}) \quad (3.2)$$

With $y_{n,c}$ representing the predicted probability for sample n belonging to class c . N is the number of samples in the dataset and $t_{n,c}$ is the true probability that sample n belongs to class c . It is 1 if the sample is of class c and 0 otherwise. The log is taken because it increases monotonically and therefore penalizes wrong predictions drastically when a prediction is near 0 for a true class.

3.1.2 Sigmoid Activation and Binary Cross-Entropy Loss

Above we see the the Softmax activation function and cross-entropy loss, which are often used when a network should classify an input into one of many classes. However, for *binary classification* tasks, the *Sigmoid* activation function and *binary cross-entropy* loss are typically used. In the following, these are briefly explained. [Zhang et al. \(2021\)](#) mention that the sigmoid function is a good solution for binary classification when we want to obtain probabilities from the model output. The sigmoid function acts as a squashing function, which takes the logit output of the model and transforms it to a number between 0 and 1, which represents the probability. It is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (3.3)$$

where z is the logit output of the network, which can theoretically range from $-\infty$ to ∞ . The result is the probability of the positive class in our case 1.

To compute the loss of the binary classifier, often *binary cross-entropy* loss is used. In mathematical terms, it is defined as follows:

$$\mathcal{L}_{BCE} = -\frac{1}{N} \sum_{n=1}^N (t_n \log(y_n) + (1 - t_n) \log(1 - y_n)) \quad (3.4)$$

Where t_n is the true label either 0 or 1 for the n -th sample. On the contrary, y_n is the predicted probability by the model. With including $\frac{1}{N}$ we normalize the loss values by the number of samples, e.g., the current batch size. Batch size refers to the number of samples fed through the network at each iteration. Normally after passing through a batch the model's parameters are updated.

3.1.3 Focal Loss

[Lin et al. \(2017\)](#) proposed the *Focal Loss*, a novel loss function designed to address the issue of easy-to-classify examples overshadowing hard ones. The focal loss is an adaptation of the binary cross-entropy loss and incorporates two mechanisms to handle class imbalance and easy-to-classify samples. Adapted from [Lin et al. \(2017\)](#) it is defined as follows:

$$\mathcal{L}_{FL} = -\frac{1}{N} \sum_{n=1}^N (\alpha_n (1 - y_n)^\gamma t_n \log(y_n) + (1 - \alpha_n) y_n^\gamma (1 - t_n) \log(1 - y_n)) \quad (3.5)$$

The focal loss assigns a weighting factor α to balance the importance of positive and negative examples. Secondly, it introduces a factor $(1 - y)^\gamma$, where y is the predicted probability and $\gamma \geq 0$ is a tunable focusing parameter. This modulating factor assigns higher loss values to hard-to-classify examples, while down-weighting the easily classified ones. When $\gamma = 0$, the focal loss is equivalent to the binary cross-entropy loss with class-weighting. As γ increases, the contribution of easy examples to the overall loss is diminished, effectively extending the range in which an example receives a low loss. The focal loss mitigates the dominance of easily classified samples in the gradient computation, enabling the model to focus more on challenging examples and potentially allowing the model to learn better features.

3.2 Training with Negatives

As mentioned briefly in Section 2.1 training an open-set classification network with negative samples is an approach that often brings improved performance compared to networks trained without negatives. Negative samples can be obtained in various ways as mentioned in Section 2.1, such as using real negatives from a different dataset, modifying known samples, or even generating negative samples artificially (Bisgin et al., 2024). With the availability of negative samples, they can be incorporated into the training of the network.

The *Entropic Open-Set (EOS)* loss (Dhamija et al., 2018) keeps the deep feature magnitudes low compared to the garbage class approach, whereby just being a simple extension of the cross-entropy loss computed on Softmax scores in Equation 3.2. The EOS loss works on a network with C outputs and does assume one-hot-target encoded values for the known samples, adapted from Bisgin et al. (2024):

$$\forall n, c \in \{1, \dots, C\} : t_{n,c} = \begin{cases} 1 & c = \tau_n \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

with τ_n being the class label of sample n where $1 \leq \tau_n \leq K$ whilst K stands for the known classes. For the negative samples, identical target values are used:

$$\forall n, c \in \{1, \dots, C\} : t_{n,c} = \frac{1}{C} \quad (3.7)$$

3.3 Open-Set Classification Rate (OSCR)

As described by Dhamija et al. (2018), the previously used metrics to evaluate open-set performance were all flawed in one way or the other, urging the need for a new metric. Dhamija et al. (2018) proposed the *Open-Set Classification Rate (OSCR)*, which results from the *Correct Classification Rate (CCR)* and *False Positive Rate (FPR)*, defined by Bisgin et al. (2024) as follows:

$$\text{CCR}(\theta) = \frac{|\{n \mid \tau_n \leq K \wedge \arg \max_{1 \leq c \leq K} y_{n,c} = \tau_n \wedge y_{n,c} \geq \theta\}|}{|N_K|} \quad (3.8)$$

$$\text{FPR}(\theta) = \frac{|\{n \mid \tau_n > K \wedge \max_{1 \leq c \leq K} y_{n,c} \geq \theta\}|}{|N_U|} \quad (3.9)$$

Where θ is a probability threshold and n iterates all the test samples. N_K denotes the total number of known test samples, whereas N_U represents the total number of unknown or negative test samples. $\tau_n \leq K$ indicates a known test sample and $\tau_n > K$ an unknown/negative test sample, with τ_n representing the label of sample n . Furthermore, $y_{n,c}$ is the probability of n th sample for class c .

The OSCR curve is generated by plotting the FPR on the x-axis and the CCR on the y-axis, while varying the threshold from the lowest to the highest possible probability value. So for a given threshold θ , the corresponding CCR and FPR value is plotted. The curve is normally drawn with a logarithmic FPR axis because most applications require a low false positive rate. The optimal OSCR curve lies in the upper left corner, with a CCR being equal to 1 while the FPR is equal to 0. Bisgin et al. (2024) mention that when the probability score $y_{n,c}$ reaches the maximum value of 1 to any reasonable precision, there exists no threshold θ such that low FPR values can be computed, resulting in the OSCR curve not extending further to the left. Palechor et al. (2023) point out that the CCR, which is highly correlated with the accuracy of the known classes, might be biased when the number of known test samples per class is unbalanced. The reason for this

potential bias is that the CCR is computed as an average over all known test samples without considering the class imbalance. If one known class has significantly more test samples than others, its contribution to the overall CCR will be higher, even if the classifier performs poorly on other known classes with fewer samples. Therefore, in our experiments, we ensure that we always have a balanced test set.

Data

This chapter shows which of the many available image-based datasets were selected. In addition, the datasets are explained in detail. It is further explained why exactly these datasets for the experiments were selected.

4.1 MNIST & EMNIST

The MNIST dataset consists of handwritten digits (0-9) in the format of grayscale 28×28 pixel images (Lecun et al., 1998). This small data size per sample, together with the low number of only 10 classes, makes it perfect for fast experimenting and validation of ideas. The MNIST dataset has 60k training and 10k test samples. Since we also need unknown and negative samples in some experiments, MNIST is too small as a dataset alone. Luckily, there exists an extension of MNIST mentioned in the following.

The EMNIST dataset consists of handwritten digits, lower- and uppercase letters in grayscale 28×28 pixel images (Cohen et al., 2017). The EMNIST letter dataset includes only letters consisting of 387k training and 24k testing samples. The EMNIST letters are used as negatives and unknowns in the experiments. MNIST and EMNIST both rely on the same base data, the NIST database (Grother and Hanaoka, 1995). This database is hardly usable by itself with our modern computer systems due to its age and the compression format used for storing it (Cohen et al., 2017). To make EMNIST as comparable as possible to MNIST, from the data processing perspective, Cohen et al. (2017) have followed the MNIST conversion steps described by Lecun et al. (1998) to have a comparable end result. However, they used a different downsampling method to handle the variation in the size and shape of the characters in the NIST dataset. This results in the EMNIST dataset being more visually blurry when compared to MNIST (van den Bergh, 2023). This fact may render the distinction between MNIST and EMNIST letters easier for a neural network, and that is why we use the EMNIST digits dataset instead of the original MNIST dataset for the digits. This guarantees all samples underwent the same preprocessing steps, which ensures that there are no unintentional differences stemming from this aspect.

In Table 4.1 the EMNIST splits used in the preliminary experiments in Section 6.2 can be seen. We follow van den Bergh (2023) and do exclude the letters *g*, *i*, *l*, *o* from our dataset because they are visually often very similar to the digits *9*, *1*, *1*, *0* respectively and do therefore pose a separability problem between known and unknown classes, which would make it impossible for a neural network to learn distinct features. We justify this approach with our interest in the relative, rather than absolute performance, between different approaches in our preliminary experiments.

Table 4.1: EMNIST SPECIFICATION FOR PRELIMINARY EXPERIMENTS. *The table shows the EMNIST splits namely training, validation and test, which we use during our preliminary experiments.*

Dataset Split	Known Samples (Positives)	Unknown Samples (Negatives)
Training	EMNIST digits	First 14 EMNIST letters (except g, i and l)
Validation	EMNIST digits	First 14 EMNIST letters (except g, i and l)
Test	EMNIST digits test set	Last 12 EMNIST letters (except o)

4.2 ImageNet Open-Set Classification Protocols

ImageNet¹ is a large-scale dataset that is built upon the WordNet synsets (Deng et al., 2009). At its release, it spanned over 5247 categories with an average of 600 labeled images per category. Since then, up until the year 2024, the ImageNet dataset continued to grow to about 14 million images over 22k synsets. With this size, ImageNet is one of the largest labeled image datasets publicly available. However, the largest benefit of ImageNet is not just the sheer size of it, but rather the underlying semantic WordNet structure, which results in several relations between the synsets with “IS-A” being the most useful (Deng et al., 2009). These relationships together with the massive amount of data available, e.g., 147 dog categories, make ImageNet so popular among researchers. Based on the 1000 classes from the ILSVRC 2012 and the master thesis from Bhoumik (2021), Palechor et al. (2023) proposed three different protocols that create artificial open environments for training and evaluating open-set algorithms. The three protocols offer a progressive level of difficulty by increasing the level of visual similarity among inputs and the amount of overlap between features of known and unknown classes.

Protocol 1 (P_1), involves a scenario where the known and unknown classes are quite dissimilar, both in terms of semantic meaning and visual characteristics. The known classes comprise 116 dog breeds, which represents a challenging, fine-grained closed-set classification problem within a single broader category. In contrast, the unknown classes consist of 166 non-animal classes that are distinctly different from the known dog breeds. The negative classes include other four-legged animals. This setup creates a clear separation between the known and unknown categories, making it theoretically fairly easy for open-set classification algorithms, as the known and unknown samples do not exhibit significant overlap or similarity.

Protocol 2 (P_2), consists only of animal classes. The known classes are made up of 30 hunting dog classes posing more of a challenge than P_1 in terms of a closed-set classifier. The remaining 31 hunting dog classes are used as negative classes. Other four-legged animal classes are being used as unknown classes. Although the known and unknown classes are still semantically distinct from each other, they share certain visual characteristics, such as the presence of fur, making it theoretically harder for open-set classification algorithms to perform well. P_2 is the smallest of the three protocols, with about a fourth of the number of training samples compared to P_1 , which makes it an excellent protocol for hyperparameter optimization.

Protocol 3 (P_3), is made up of a mix of common classes including animals, plants, and objects. There are 151 known classes, 97 negative classes, and 164 unknown classes. This constellation makes it very hard for open-set classification algorithms to perform well since the known and unknown classes share many similarities. Of the three protocols, P_3 is the easiest to distinguish within the known classes, as they are only slightly similar.

The above ImageNet open-set classification protocols were chosen because of their clear definition and varying difficulty regarding open-set classification. Due to their recent publication, these protocols have not had the chance to be widely adopted in open-set classification papers

¹<https://www.image-net.org/>

and are currently only used in a paper in progress from [Bisgin et al. \(2024\)](#). Nonetheless, the structure of the protocols seems mature and we therefore use them in our work.

Approach

To tackle the open-set classification problem we propose a novel approach that uses an ensemble of binary classifiers. This approach involves a few steps such as modeling a multiclass classification as a binary classification, training a binary ensemble model, and finally aggregating the predictions from the model into a classification output. The approaches to these problems are described in the following.

5.1 Creating a Binary Classification Problem

To train an ensemble of binary classifiers we have to partition the dataset for each classifier separately, this allows each binary classifier to learn a unique discrimination task. For partitioning into a binary problem, we can use a completely random approach, which is similar to the strategy used by [Vareto et al. \(2023\)](#); [Vareto and Schwartz \(2020\)](#) for face recognition problems, or try to optimize the distance between classes.

5.1.1 Random Partitioning of Classes

We define the ensemble of binary classifiers as a matrix \mathbf{E} as:

$$\forall b \in \{1, \dots, B\}, c \in \{1, \dots, C\} : \mathbf{E}_{b,c} \in \{0, 1\} \quad (5.1)$$

with B being the total number of binary classifiers in the ensemble and C being the total number of classes. For training a single binary classifier $\mathbf{E}_{b,*}$, the set of classes C is randomly split into two disjoint partitions, which if possible are sized equally. Formally we define this as:

$$\forall b : (\mathbf{E}_{b,*}^0 \cup \mathbf{E}_{b,*}^1 = C) \wedge (\mathbf{E}_{b,*}^0 \cap \mathbf{E}_{b,*}^1 = \emptyset) \wedge \left(\sum_{c=1}^C \mathbf{E}_{b,*}^0 \approx \frac{C}{2} \right) \quad (5.2)$$

To prevent any two binary classifiers out of the ensemble from learning to distinguish among the same split, it is necessary to enforce the condition that no two binary classifiers share the same class partition. Mathematically, we express this constraint as follows:

$$\forall b_1, b_2 : b_1 \neq b_2 \implies \mathbf{E}_{b_1,*}^0 \neq \mathbf{E}_{b_2,*}^0 \quad (5.3)$$

Furthermore, to avoid symmetry, we want to ensure that no two partitions represent the same classes but with inverted labels (mirrored partitions). This condition can be formalized as:

$$\forall b_1, b_2 : \mathbf{E}_{b_1,*}^0 \neq \mathbf{E}_{b_2,*}^1 \quad (5.4)$$

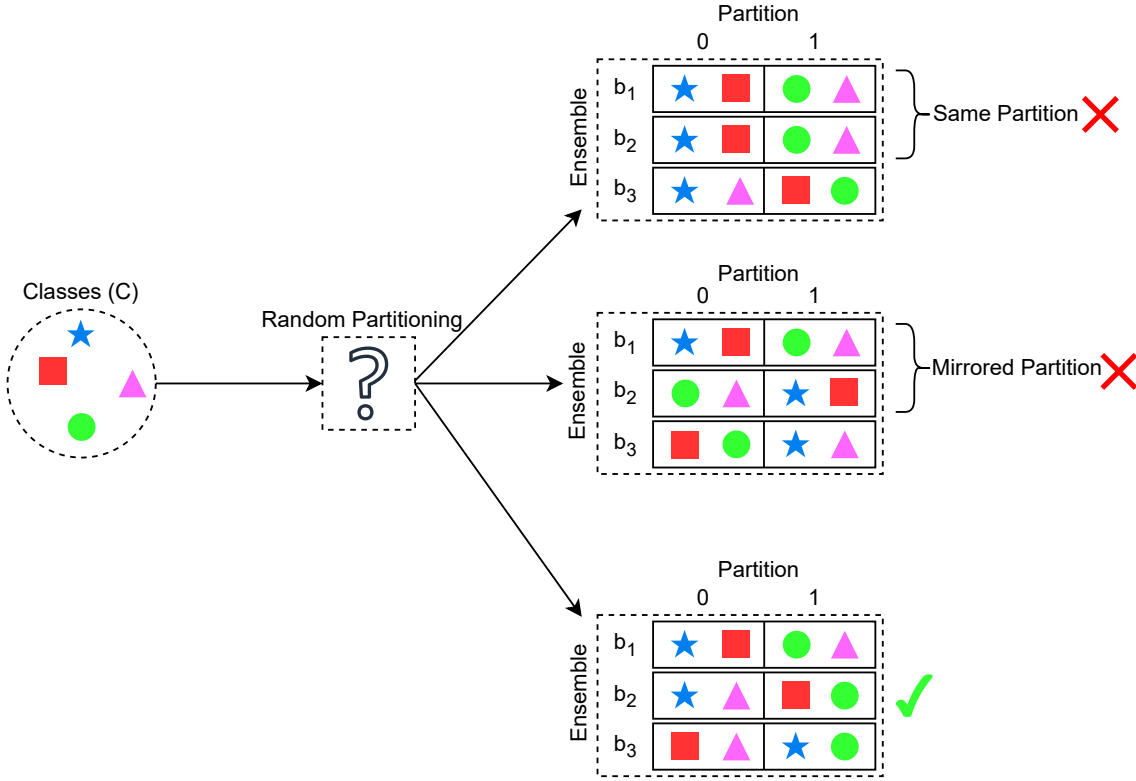


Figure 5.1: RANDOM PARTITIONING OF CLASSES. This figure shows a visual example of the random partitioning of the classes. Additionally, the constraints are shown in the upper two ensembles. The ensemble at the top is invalid due to the constraint in Equation 5.3, whereas the ensemble in the middle is invalid due to the constraint in 5.4. The ensemble at the bottom respects all the constraints outlined and is therefore valid.

A visual example of an ensemble consisting of four different classes with three binary classifiers can be seen in Figure 5.1, also the constraints defined above can be seen visually.

Furthermore, when defining an ensemble E for C classes there is a minimum and maximum number of binary classifiers that result from the constraints outlined above. The minimum number of binary classifiers for a given number of classes C is defined as follows:

$$B_{min} = \lceil \log_2 C \rceil \quad (5.5)$$

This equation results from the constraint that for each $c \in C$ we need to have a unique binary representation. This results in $2^B \geq C$, we reshape this equation by taking the logarithm on both sides and rounding up to the next bigger integer, since B must be an integer. On the other hand, we define the maximum number of possible binary classifiers as follows:

$$B_{max} = \begin{cases} \frac{\binom{C}{2}}{2} & \text{if } C \text{ is even} \\ \binom{C}{(C-1)/2} & \text{if } C \text{ is odd} \end{cases} \quad (5.6)$$

When C is even we choose $C/2$ out of C , which can be calculated using the binomial coefficient. This resulting number then needs to be divided by 2 to account for the mirrored partition constraint outlined in Equation 5.4. In the case when C is an odd number, the equation changes due

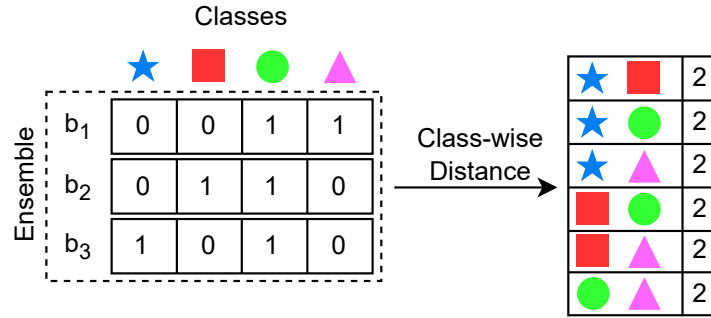


Figure 5.2: CLASS-WISE DISTANCE. This figure shows a visual example of the class-wise distance. The binary representation of the classes given through the different binary classifiers inside the ensemble can be used to calculate an element-wise difference between two classes. The resulting class-wise distance is shown in the table on the right for all the combinations of any two classes.

to the binary partition which cannot be balanced anymore (same number of classes in partition 0 and 1). We are choosing now $(C - 1)/2$, which represents the smaller partition, out of C to get the number of ways to choose the larger partition. Since this partition is never balanced, its mirror is always a different partition, so we do not need to divide by two to account for that.

5.1.2 Partitioning with maximizing Hamming Distance among Classes

Except for adhering to the constraints outlined, partitioning the original classes is done completely random in Section 5.1.1. However, this may not be optimal when considering the resulting class-wise distance, which results from their binary vector representation. A visual example of the class-wise distance can be seen in Figure 5.2 where we first see a binary representation of the classes that can then be used to calculate the class-wise distance between any two classes. Generally speaking, we want the class-wise difference between the binary vector representations of the classes to be as large as possible, as this hopefully makes the ensemble \mathbf{E} more resistant to classification errors of single classifiers $\mathbf{E}_{b,*}$. In simple terms, because the classes have larger class-wise distances among all classes, we minimize the risk of false classification from the whole ensemble when an individual classifier makes an error in its prediction.

We can measure this class-wise distance by computing the *Hamming* distance, which measures the difference between two points (Hamming, 1950). In Equation (5.7) the Hamming distance is adapted to our case, where we want to calculate the Hamming distance between any two binary vector class representations, as the number of positions where those two vectors differ. We define the Hamming distance as follows:

$$\forall c_1, c_2: \delta(\mathbf{E}_{b,c_1}, \mathbf{E}_{b,c_2}) = \sum_{b=1}^B |\mathbf{E}_{b,c_1} - \mathbf{E}_{b,c_2}| \quad (5.7)$$

Now instead of just randomly generating the binary partitions, we select the first binary subset for the classifier randomly b_1 , and afterward for all remaining binary classifiers we take the partitions that find the maximum minimum (infimum) Hamming distance between the classes among all other classifiers greedily, as can be seen in Algorithm 1.

Algorithm 1 MAXIMIZING HAMMING DISTANCE BETWEEN CLASSES. *This algorithm creates $n_classifiers$ number of binary partitions for $n_classes$, while maximizing the minimum (infimum) distance between any two classes (columns of the matrix) by greedily adding the binary classifiers.*

```

procedure MAXIMUM_HAMMING_DISTANCE( $n\_classifiers, n\_classes$ )
   $combinations \leftarrow$  nested vector list of the cartesian product of 0 and 1 with length  $n\_classes$ 
   $balanced \leftarrow$  empty list
  for  $vector$  in  $combinations$  do
    if  $vector$  starts with 1 then                                     ▷ Avoids inversion
      continue
    end if
    if  $n\_classes$  is even and  $\text{sum}(vector)$  equals  $n\_classes / 2$  then   ▷ Balanced partitions
      append  $vector$  to  $balanced$ 
    else if  $n\_classes$  is odd and  $\text{sum}(vector)$  equals  $(n\_classes - 1) / 2$  or
       $(n\_classes + 1) / 2$  then
      append  $vector$  to  $balanced$ 
    end if
  end for
   $binary\_partitions \leftarrow$  append random first vector from  $balanced$ 
  remove  $vector$  in  $binary\_partitions$  from  $balanced$ 
  while  $\text{len}(binary\_partitions) < n\_classifiers$  do
     $min\_column\_ham\_dist \leftarrow$  empty list to save the min Hamming dist per combination
    for  $combination$  in  $balanced$  do
       $matrix \leftarrow$  vertically stack ( $binary\_partitions, combination$ )
       $min\_column\_ham\_dist \leftarrow$  append  $min\_ham\_dist\_between\_all\_col(matrix)$ 
    end for
     $infimum\_col\_dist\_indices \leftarrow$  maximum value indices from  $min\_column\_ham\_dist$ 
     $vector\_index \leftarrow$  choose random index from  $infimum\_col\_dist\_indices$ 
     $binary\_partitions \leftarrow$  vertically stack
      ( $binary\_partitions, balanced\_combinations[vector\_index]$ )
  end while
  return  $binary\_partitions$                                        ▷ Returns matrix of shape  $n\_classifiers \times n\_classes$ 
end procedure

```

5.2 Separate and Combined Binary Classifiers

For the *separate* approach, we use a backbone network architecture, such as LeNet-5 for the EM-NIST dataset, as the base model. Each binary classifier $b \in B$ is then trained separately and learns its binary classification task. The binary partition of each binary classifier is defined as in Section 5.1.1 and does adhere to the constraints in the Equations 5.3 and 5.4. For each sample, the separate binary classifier model outputs a prediction $y_b \in [0, 1]$. This process is repeated for all binary classifiers in the ensemble, resulting in B independently trained models, of which the prediction can be combined to obtain the classification output of the ensemble for a given sample, as can be seen in Section 5.3.

Instead of training each binary classifier separately, we can also use a *combined* approach. In this combined approach, we also use the same backbone network, however, the backbone is shared among all binary classifiers and outputs multiple predictions simultaneously. The combined network has B binary outputs, one for each binary classifier $b \in B$, instead of having B separate models as in the separate approach. The combined network is trained on all splits si-

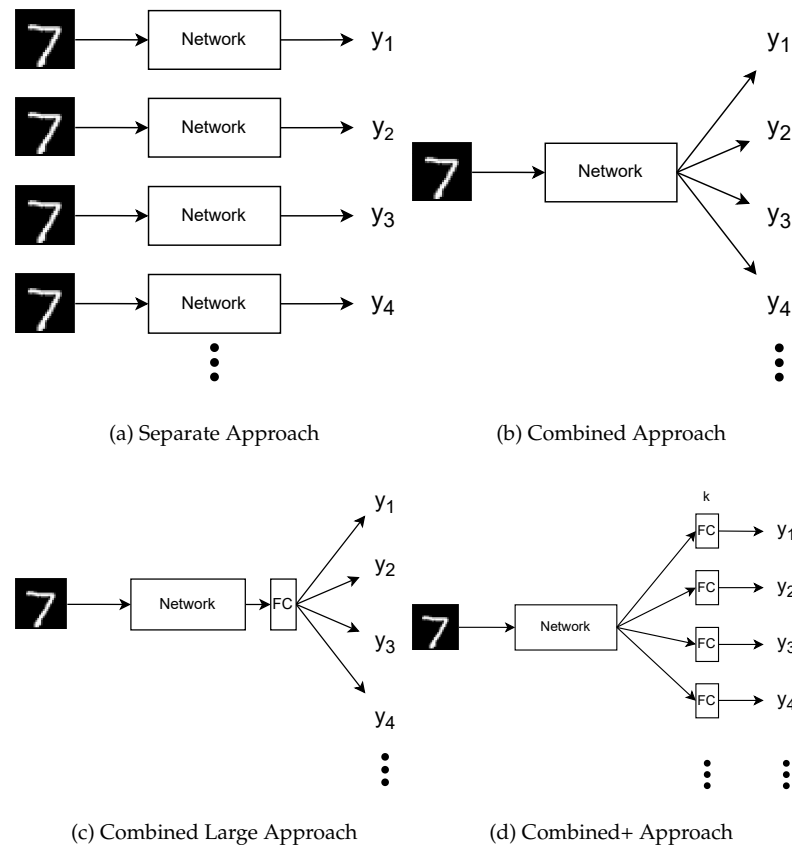


Figure 5.3: SEPARATE VS. COMBINED VS. COMBINED LARGE VS. COMBINED+ APPROACH. The separate approach (a) trains each network separately, whereas the combined approach (b) trains only one network with multiple classifiers. The combined large (c) and combined+ (d) networks extend the combined approach by training an additional fully connected layer or two fully connected layers of dimensionality k per network activated with ReLU respectively.

multaneously and optimizes an averaged loss function that incorporates the individual losses for each classifier, ensuring that it learns to discriminate between its corresponding binary partition. A depiction of both approaches can be seen in Figure 5.3.

Separate binary classifiers do need more computational resources to train when compared to the combined approach. This is because they normally have much more learnable parameters and each data sample has to be accessed multiple times. Rudd et al. (2016) have shown that their combined approach even outperforms their separate approach when measured on the classification error rate of recognizing facial attributes. This is promising in terms of the performance difference between our separate and combined approaches.

CNNs trained on images all learn similar first-layer features, such as Gabor filters or color blobs, regardless of the specific dataset that is used (Yosinski et al., 2014). This results in the CNN learning to detect edges and corners early on. With this in mind, the separate approach could be an unnecessary overhead of computation because the features learned in the first few layers are likely the same for each network. On the other side, the combined approach could give the network too little room to learn a good feature map for every binary classifier.

Table 5.1: EXAMPLE OF BINARY CLASS ENCODING. For each binary classifier in the ensemble E , the binary encoding for each class $c \in C$ is given. Every class has a unique binary encoding, which is shown in each column of a class. For c_1 the binary encoding would be $(0, 1, 0)$.

Binary Classifier	Classes			
	c_1	c_2	c_3	c_4
b_1	0	0	1	1
b_2	1	0	1	0
b_3	0	1	1	0

This is why we introduce additionally two slight variations of the *combined* approach. The simpler one is the *combined large* network, which just extends the *combined* approach by an additional fully connected layer in its final layer. This increases the capacity of the *combined large* network and allows it, theoretically, to learn more complex non-linear relationships between the features. Moreover, this would then in an optimal case result in better classification performance, when compared to the standard *combined* network. Building upon this idea of more fully connected layers, we present a second variation of the *combined* approach, namely the *combined+* approach, which can be seen in Figure 5.3d. In this *combined+* approach, the architecture is the same as in the normal *combined* approach, except for the last output layer. Whereas in the *combined* approach, the last layer consists of a single fully connected layer that produces all the binary outputs, in the *combined+* approach the output layer consists of two separate fully connected layers for each binary classifier in between which ReLU activation is used. The number of optimal parameters of these two layers is denoted with k in Figure 5.3d and is subject to fine-tuning. This gives the model much more capabilities to learn the binary classification task among its two sets, while preventing most of the computational overhead of the *separate* approach. In theory, this approach combines the best of both worlds.

5.3 Obtaining a Classification Score from the Ensemble Model Output

When using an ensemble model with either the *separate* or *combined/+* approach, as described in Section 5.2, one gets many binary outputs, each corresponding to the specific binary partition the model has been trained on. However, in the experiments, we are working with datasets that contain way more than only two initial classes. Resulting from this, a single binary prediction is insufficient to determine the class to which input sample x belongs. Luckily, for a given sample x , we have one binary prediction of each model b in the ensemble E . All those binary predictions can be combined, resulting in a single prediction. This is based on the unique binary encoding, defined for each class in C , which is defined in Section 5.1.1 and is exemplarily depicted in Table 5.1. To calculate the ensemble model's final prediction, we compare the binary classifier outputs from the ensemble, which we define as a matrix of predictions $\mathbf{Y}_{b,c}$, to the unique binary encodings of each class given in $\mathbf{E}_{b,c}$, selecting the class that most closely matches the ensemble's output. This is done by computing the Hamming distance defined in Equation 5.7 between the binary representations of the classes and the predictions of the ensemble:

$$\rho(\mathbf{E}, \mathbf{Y}) = \begin{cases} \delta(\mathbf{E}_{b,c}, \mathbf{Y}_{b,c}) & \text{if no threshold} \\ \delta(\mathbf{E}_{b,c}, \lfloor \mathbf{Y}_{b,c} \rfloor) & \text{if threshold} \end{cases} \quad (5.8)$$

Whereas \mathbf{E} corresponds to the matrix of target labels for each binary classifier. On the other

hand, \mathbf{Y} is the matrix composed of all individual binary classifier predictions after applying *sigmoid* to obtain probabilities between 0 and 1. These probabilities can be used to calculate the Hamming distance, also named similarity score ρ , or they can be further processed by applying a *threshold*, to round the probabilities to their nearest integer value, which is either 0 or 1 and is depicted as $\lfloor \mathbf{Y}_{b,c} \rfloor$. The resulting similarity score obtained for each possible class shows the most probable true class given the input sample. The lower the score the better and therefore the more similar the class is to the model's prediction. To obtain the predicted class from the similarity scores one can simply apply $\arg \min \rho$.

The model outputs can also be used directly without applying sigmoid activation. In this special case, where we use the *logits* directly, the Equation (5.8) is not applicable because the logits are scaled between $-\infty$ and $+\infty$. To solve this problem, we use the following formula:

$$\rho^*(\mathbf{E}, \mathbf{Y}) = \sum_{c=1}^C ((2\mathbf{E}_{*,c} - 1) \odot \mathbf{Y}_{*,c}) \quad (5.9)$$

First, we convert the target labels in \mathbf{E} to -1 and 1 respectively by applying the operation $2\mathbf{E} - 1$. Afterward, we calculate the Hadamard product of \mathbf{E} and \mathbf{Y} and sum over the columns. When the logits are correct, so negative or positive for the -1 and 1 target respectively, the resulting Hadamard product is also positive. In the case of an inverse prediction the resulting Hadamard product is negative and does therefore reduce the resulting class similarity. Through applying Equation 5.9 we obtain the class similarity for each class, where a higher score per class indicates a greater similarity. To get the predicted class from the computed similarity scores one can just select $\arg \max \rho^*$.

To obtain scores that can be interpreted as probabilities, and are therefore scaled between 0 and 1, from the resulting scores of Equations 5.8 and 5.9, we can apply the following formula:

$$y = \begin{cases} \frac{(B - \rho(\mathbf{E}, \mathbf{Y}))}{B} & \text{if } \rho \\ \frac{(\rho^*(\mathbf{E}, \mathbf{Y}))}{B} & \text{if } \rho^* \end{cases} \quad (5.10)$$

In conclusion, we show in Table 5.2(a) the three different evaluation approaches that we can use before calculating the similarity score.

5.3.1 Training a Binary Ensemble with Negative Samples

In Section 3.2 we have shown the EOS and in Section 2.1 the garbage class approach, to incorporate negatives into training, when using a Softmax-based training method. The garbage class approach is easy to implement as it just needs a reject option in case of an unknown sample, however, [Palechor et al. \(2023\)](#) have shown that it is inferior to other approaches of incorporating negatives into training. The EOS approach in contrast seems more promising but is infeasible in its initial form because we do not work with Softmax. That's why we need new, specific methods that work with an ensemble of binary classifiers.

The most straightforward approach is integrating the negative class just as any other class, resulting in $C + 1$ total classes. We call this the *integrated* approach from now on. The approach is somewhat similar to the *garbage* approach. As in the *garbage* approach, we have one more class that the network has to learn. The big difference, however, is that the network does not learn an additional output for this negative class, but the negative class is simply integrated into the binary partition and thus receives its binary encoding, as any other class does in Table 5.1. The benefit of the *integrated* approach is its easy integration into the training process, as it almost requires no change of the existing code. A potential downside could be that when there are many more negative samples than the average number of known samples for binary classification,

the network optimizes heavily for the negative samples because they are overrepresented in the training dataset.

Another approach to integrating negatives into the training adapts the EOS approach shown in Section 3.2 to our binary ensemble. We name this the *equal* approach. While the label for the known samples is still either 0 or 1 per binary classifier, for negative samples the label has an identical target value of 0.5. This can be computed similarly as in Equation 3.2, where now C just has a value of 2 because a single binary classifier in the ensemble does only do binary classification according to its unique partition. So in other words, we force the binary ensemble to learn a logit output of 0 which correlates to 0.5 probability after using sigmoid for negative samples. In theory, 0.5 as an output is the lowest value for each binary class 0 and 1, which can be predicted by a binary classifier because it has the same difference to both values of 0.5. The *equal* approach has the benefit of the network learning a different output for the negatives than the known samples, which helps to distinguish negatives from known samples. Like the *integrated* approach, the *equal* approach also suffers from an unbalanced number of negative and known samples. This imbalance may affect the network's ability to generalize effectively across all sample types.

To focus more on the hard-to-classify samples when training with negatives, we do an experiment using focal loss, described in Section 3.1.3. We use the provided PyTorch implementation of focal loss¹.

¹https://pytorch.org/vision/main/_modules/torchvision/ops/focal_loss.html

Table 5.2: EVALUATION APPROACHES AND METHODS FOR TRAINING WITH NEGATIVES. *This table shows the three evaluation approaches we use for evaluating the prediction of a binary ensemble model (a). Further, the two approaches are shown, which are used to train binary ensembles with negative samples (b). The approaches to incorporate negatives are independent of the evaluation approaches, opening up lots of possible combinations.*

(a) Evaluation Approaches			
Evaluation Method	Description	Advantages	Disadvantages
Logits (ρ^*)	Use the logits directly without non-linear activation function as in Equation 5.9	No information loss	May be prone to overreaching of logits and having excessive influence on final prediction
Sigmoid Probabilities (ρ)	Sigmoid applied to logits to get probabilities as in Equation 5.8	Interpretable probabilities and squashing of excessive logits	Potential information loss due to the application of non-linear transformation
Threshold (ρ with threshold)	Sigmoid applied to logits and then thresholding of probabilities as in Equation 5.8	Simple decision rule	Introduces severe information loss
(b) Training Binary Ensembles with negatives			
Method of incorporating Negatives	Description	Advantages	Disadvantages
Integrated	Integrate the negative samples like any other class	Needs almost no adaption of the code	With more negative samples than known samples the binary classifier may lose performance for the known samples
Equal	Force a 0.5 sigmoid probability output for negative samples	The ensemble is forced to keep the feature magnitude for negatives low	May suffer from unbalanced negative vs. known samples

Experiments and Results

In this chapter, experiments are performed based on the approaches described in the Chapter 5. Preliminary experiments are performed on the smaller EMNIST dataset and the final experiments are performed on ImageNet.

6.1 Neural Networks

Since our main goal is to evaluate different binary ensemble approaches, the networks are generally trained with the same hyperparameters to obtain comparable results. The neural networks are implemented in PyTorch¹ and are all trained on Nvidia RTX 2080 Ti graphics cards with 11 GB VRAM. Furthermore, the whole code is publicly available on GitHub² and is a fork from the code used by Palechor et al. (2023).

6.2 Preliminary Experiments on EMNIST

For our preliminary experiments, we use a small LeNet-5, which enables fast iteration, as the network trains very quickly due to the low computational needs. In the Table 6.1 the training parameters which are used can be seen. The preliminary experiments serve multiple purposes: due to the use of a small model, combined with the EMNIST dataset, they allow for fast iteration which is crucial when experimenting with a novel approach, in our case binary ensembles. Furthermore, the research questions RQ 1-3 can already be answered with our preliminary experiments. The networks are evaluated on their CCR@FPR at the FPR of 10^{-3} , 10^{-2} , 10^{-1} , 1, with the latter representing the closed-set accuracy. The findings of these preliminary experiments are then later applied to the experiments on the ImageNet protocols in Section 6.3.

To facilitate this, the setup and procedure in the preliminary experiments must be similar to the subsequent experiments on the ImageNet protocols, while still allowing for potential necessary adjustments. This is possible using the dataset described in Section 4.1. Furthermore, in the preliminary as well as in the following experiments, a CNN is used as the backbone of the classifier, which makes the experiments comparable, also in the dimension of network architecture.

¹<https://pytorch.org/>

²<https://github.com/Sirofjelly/openset-imagenet-comparison>

Table 6.1: TRAINING PARAMETERS OF PRELIMINARY EXPERIMENTS. *This table shows the training parameters used for the experiments on the EMNIST dataset. The separate and combined approaches are as shown in Figure 5.3 and the baseline does use a traditional Softmax-based approach. Number of Models is the number of models used simultaneously and Number of Binary Classifiers is the number of individual binary classifiers which compose the ensemble of binary classifiers. Values separated by a "/" indicate that this value is either one or the other depending on the experiment.*

Hyperparameter	Separate	Combined/Large	Baseline
Task	Binary Classification	Binary Classification	Classification
Number of Models	40	1	1
Number of Binary Classifiers (B)	40	126/40	No binary classifiers
Epochs	50	50	50
Batch Size	32	32	32
Loss	BCE	BCE/Focal	Softmax/EOS
Optimizer	SGD	SGD	SGD
Learning Rate	0.001	0.001	0.001

6.2.1 Optimal number of Classifiers and maximizing Distance between Classes

The first experiment aims to answer *RQ 1*. Concretely, we test whether maximizing the Hamming distance between classes leads to a better performance than a random approach (*RQ 1.1*). Furthermore, in the same experiment, we evaluate whether more binary classifiers lead to a better performance (*RQ 1.2*). To test this, a combined network with 126 binary classifiers, the maximum number for 10 classes given the constraints in Section 5.1.1 and Equation 5.6, is trained one time and then evaluated ten times on the validation set for all possible numbers of binary classifiers, using both approaches, namely the random and the one that maximizes the Hamming distance between the classifiers. We evaluate each approach ten times because this changes the binary classifiers that compose the ensemble each time, therefore possibly resulting in different CCR@FPR scores. The resulting summed CCR@FPR are shown in Figure 6.1.

The theoretical summed minimum and maximum of the CCR@FPR are zero and four respectively for a total of four FPR values. Considering this while looking at the results in Figure 6.1 we see that the summed CCR@FPR values always lay between 2.2 and 2.5, which is a rather small span. When looking at the differences for a given number of binary classifiers, we do see some small differences for the random vs. Hamming distance class partitioning approach. The random approach seems to reach a higher score of around 2.4 earlier between 15-20 binary classifiers, whereas in contrast, the Hamming distance approach does reach this score much later with 50 binary classifiers. We observe a smaller standard deviation for the Hamming distance approach when compared to the random approach. This is most likely due to the constraint of maximizing the Hamming distance, which results in less variation.

When we look at the number of binary classifiers and their impact on the CCR@FPR values, we observe a small trend towards more binary classifiers resulting in a better performance. The biggest leap in performance is made between the minimum number of four and 20 binary classifiers for the random approach, after that the curve is flattening out. For the Hamming distance approach, there is a more creeping improvement in performance which only starts flattening out at around 80 binary classifiers. This is interesting as it does not align with our hypothesis in *RQ 1.1*, where we assumed that a larger Hamming distance between the classes offers a better classification performance than a random approach.

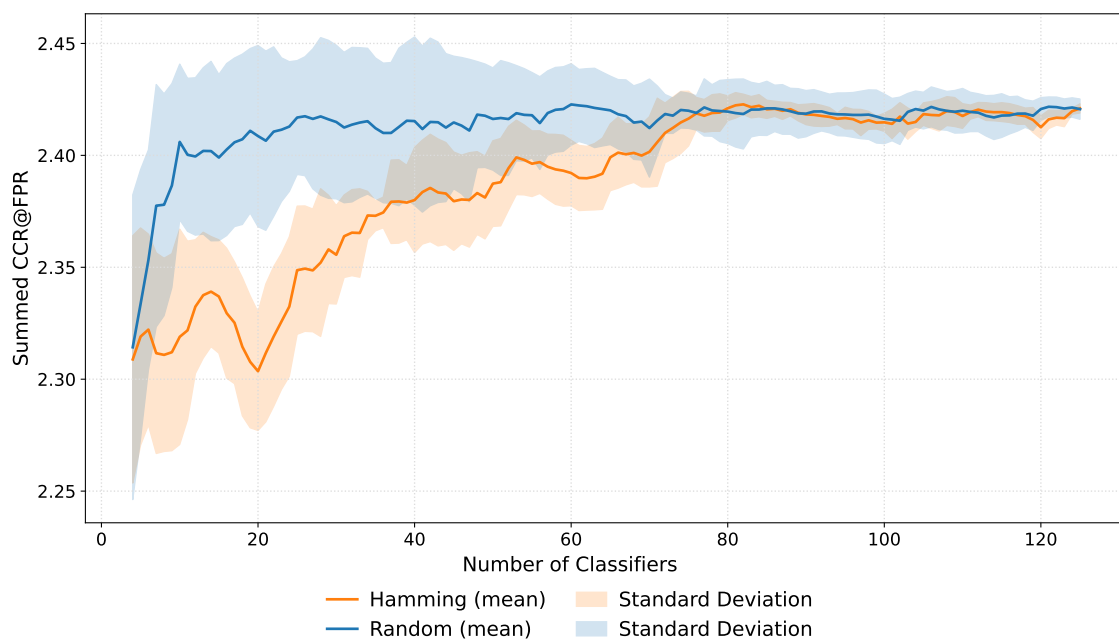


Figure 6.1: RANDOM VS. HAMMING DISTANCE APPROACH AND NUMBER OF BINARY CLASSIFIERS. This figure shows the number of binary classifiers on the x-axis and the summed CCR@FPR values on the y-axis for the FPR at $(10^{-3}, 10^{-2}, 10^{-1}, 1)$. The Hamming curve depicts models that maximized the minimum Hamming distance between classes as shown in Algorithm 1, compared to the random approach which just chose random models. Both approaches are evaluated ten times each and their mean and standard deviation is shown. These curves result from the evaluation on the validation set.

In summary, we can say that using the Hamming distance approach for maximizing the distance between classes does seem to be inferior compared to the random approach. Furthermore, for the number of binary classifiers, we did observe a slight trend which indicates that more classifiers can result in a marginally better classification performance. However, training with more binary classifiers also leads to slightly increased training time. The biggest gain in performance seems to be made when choosing more than the minimum number of binary classifiers, as defined in Equation 5.5. Going forward, we use 40 binary classifiers for the following preliminary experiments, due to the slightly faster training speed when compared to 126 classifiers, while still resulting in a minimum Hamming distance between the classes of 20 when computing it as seen in Equation 5.7 and visually in Figure 5.2. Furthermore, we do use the Hamming distance approach for generating the classes, as it only adds an insignificant amount of extra time at the beginning of the training. Although we have seen no performance increase when using it, it is theoretically still a good approach to keep the Hamming distance rather larger than smaller due to the increased error resistance.

6.2.2 Separate vs. Combined Binary Classifiers

By comparing *separate* and *combined* binary classifiers as described in Section 5.2 we want to answer RQ 2.1. The resulting CCR@FPR values are shown in Table 6.2. We do see that, except for an FPR of 10^{-3} , the separate classifier outperforms the combined approach by a small margin. For the FPR of 10^{-1} , the separate approach achieves a remarkably higher score. This is most likely due

Table 6.2: SEPARATE AND COMBINED BINARY CLASSIFIERS. This table shows the CCR@FPR values on the validation set for separate and combined binary classifiers and an additional combined large network that has been trained with an extra fully connected layer. B indicates the number of binary classifiers in the ensemble. An empty cell means this specific CCR has not been reached for a given FPR value. The best result per FPR value is marked **bold**.

Model	B	CCR@FPR			
		10^{-3}	10^{-2}	10^{-1}	1
Separate	40		0.1001	0.7059	0.9943
Combined	40	0.0059	0.0716	0.5342	0.9871
Combined Large	40	0.0004	0.0267	0.5630	0.9887

to it having much more trainable parameters in total when compared to the combined approach. An attempt to address this bottleneck and give the combined network more chance to learn the partitions is to add an extra fully connected layer at the end. This has been done in the *combined large* network shown in Figure 5.3(c). However, when we compare the CCR@FPR between the two combined approaches, it becomes clear that this does not seem to improve performance. This leads us to the conclusion that giving the network slightly more learnable parameters in its outer layers does not benefit the overall result and for an effect probably even more parameters would be needed in the outer layer.

In summary, we can say that the *separate* approach does seem to achieve a higher score. This was almost to be expected due to the many more parameters to be learned, even if it could theoretically be accompanied by a simpler latent space due to the learning task of only one binary classification per model. The big disadvantage of the separate approach is the training time, which is many times longer due to the many models. This is also the reason why in the next experiments the combined approach is used, even if it gives slightly worse results.

6.2.3 Evaluation Approaches

The binary ensemble model’s outputs can be used in various ways to obtain a final classification score, as discussed in Section 5.3. With this experiment, we want to answer RQ 2.2. By training one combined network with 40 binary classifiers and evaluating it using a threshold, probabilities, and logits directly, we get the results shown in Table 6.3. The closed-set accuracy is nearly the same for all three evaluation approaches and almost reaches 99%. When we look at the FPR for 10^{-1} and smaller, the *threshold* approach has not even reached this CCR@FPR. This indicates that it is performing poorly on the open-set task and has given the negative samples such a high confidence score that there is no threshold θ which allows computing low FPR values. Most likely this is due to the very high information loss which occurs when we just use a threshold that projects any negative logit to zero and any positive value to one. This is a large simplification that is not beneficial and this gets even more obvious when we look at the *probabilities* and *logits* approaches. Those two approaches reached every FPR value down to 10^{-3} . When comparing the CCR@FPR values directly, the *probability* approach always performs better than the *logits* approach except for the closed-set accuracy where they reach the same score. Likely this is the case because the sigmoid function used to obtain the probabilities squashes the logits between zero and one, which makes this approach less prone to very large or small logits. Although this brings some information loss due to the nature of the logit function, the squashing property of the sigmoid function seems more important than the complete information that is available when the logits are used directly.

The *probability* evaluation approach stands out on top and excels at every FPR in Table 6.3.

Table 6.3: EVALUATION METRICS. This table shows the CCR@FPR values on the validation set for the evaluation approaches threshold, probabilities, and logits. An empty cell means this specific CCR has not been reached for a given FPR value. The best result per FPR value is marked **bold**.

Evaluation Approach	CCR@FPR			
	10^{-3}	10^{-2}	10^{-1}	1
Threshold				0.9868
Probabilities	0.0059	0.0716	0.5342	0.9871
Logits	0.0018	0.0525	0.4479	0.9871

This is why it will also be used in all of the following experiments.

6.2.4 Training with Negatives

By integrating negative samples into the training, we hope to achieve better results when compared to training without negatives. With this experiment, we want to answer RQ 3.1 and RQ 3.2. We have trained the networks once with an *integrated* approach which integrates the negatives as an additional class and once with an *equal* approach where the network learns to predict 0.5 for the negative sample, as described in Section 5.3.1. The better one of the two binary ensemble options was then trained again using focal loss, with the idea in mind to focus the network more on hard-to-classify examples to reach better performance. Regarding the parameters α and γ of the focal loss, we set $\alpha = -1$ to ignore the class weighting and $\gamma = 2$ to focus more on the hard-to-classify examples. As a baseline, we have trained a network using plain Softmax without negatives and one with negatives using the Softmax-based EOS approach, as described in Section 3.2. With those baselines, we want to see if training with negatives boosts the performance on the open-set classification task.

When looking at the resulting CCR@FPR scores in Table 6.4, we can see that the *integrated* approach does perform marginally worse than the *equal* approach. This could be because an output of 0.5 introduces more differences in the latent space than just learning the negatives as an additional class and therefore fewer false classifications happen. When comparing the BCE against the focal loss trained on the same network architecture and using the *equal* approach to integrate negatives we see, that the accuracy of the focal loss approach is about .6% lower than the *equal* approach trained with BCE. Also, for the remaining FPR the *equal* approach trained with focal loss performs worse than its counterpart trained with just the BCE. This might be due to the nature of the focal loss, which in our setting focuses on the hard-to-classify samples at the expense of the easily-classified ones.

Taking our best approach, *equal* trained with BCE, and comparing it against the EOS baseline, we do observe, that the accuracy is virtually the same for both models. Moreover, it is worse than for the Softmax-based approach, which was not trained with negatives. This slightly lower accuracy on the closed-set seems like the sacrifice for increased open-set classification capabilities. This is further supported when we look at Table 6.4, the Softmax-based approach was the only one not reaching an FPR of 10^{-1} , which indicates that this baseline does produce such high probabilities that there is no threshold θ which would allow the computation of lower FPR values. At the FPR of 10^{-1} our *equal* approach does even outperform the EOS at about 2.5%. Although this is a very small difference, it might be an indication that training a binary ensemble with negatives to output 0.5 for negatives might be beneficial compared to the Softmax-based EOS approach. This positive result indicates that the *equal* approach is a promising way to integrate negative samples into the training.

Table 6.4: TRAINING WITH NEGATIVES. *This table shows the CCR@FPR values for networks that have been trained with negative samples. Additionally, the equal approach has been trained with focal loss to focus more on hard-to-classify samples. The EOS approach is the baseline to compare our negative approaches against. B indicates the number of binary classifiers in the ensemble. An empty cell means this specific CCR has not been reached for a given FPR value. The best result per FPR value is marked **bold**.*

Model	B	Loss	CCR@FPR			
			10^{-3}	10^{-2}	10^{-1}	1
Integrated (Extra Class)	40	BCE	0.0249	0.1009	0.5993	0.9864
Equal (0.5 Output)	40	BCE	0.0324	0.1149	0.6094	0.9852
Equal (0.5 Output)	40	Focal Loss ($\gamma = 2$)	0.0133	0.0722	0.5203	0.9791
EOS	0	Entropic			0.5487	0.9893
Softmax	0	CE				0.9918

Table 6.5: TRAINING PARAMETERS OF IMAGENET EXPERIMENTS. *This table shows the training parameters used for the experiments on the ImageNet protocols. The combined+ and combined approaches are as shown in Figure 5.3 and the baseline does use a traditional Softmax-based approach. Number of Models is the number of models used simultaneously and Number of Binary Classifiers is the number of individual binary classifiers which compose the ensemble of binary classifiers. Values separated by a “/” indicate that this value is either one or the other depending on the experiment.*

Hyperparameter	Combined+	Combined	Baseline
Task	Binary Classification	Binary Classification	Classification
Number of Models	1	1	1
Number of Binary Classifiers (B)	40	40/500	No binary classifiers
Epochs	120	120	120
Batch Size	32	32	32
Loss	BCE	BCE	Softmax/EOS
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001

6.3 Experiments on ImageNet

This section builds upon the results obtained from Section 6.2. The experiments are conducted on the ImageNet protocols described in Section 4.2. As backbone network we use a ResNet-50³, which offers a good trade-off between network size, training speed, and ability to learn the classification task on the ImageNet protocols. In Table 6.5 the training parameters used for the subsequent experiments on the ImageNet protocols can be seen.

6.3.1 Optimal Number of Classifiers

This experiment is very similar to the one in Section 6.2.1 and aims to answer RQ 4.1. To maintain comparability and to keep complexity low, we do not train with negative samples. We want to test again what the optimal number of binary classifiers is, this time on the ImageNet dataset, especially for protocol 2, since we do have a lot more classes and a different underlying model.

³Implementation adapted from here: <https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py#L736>

A combined model with 500 binary classifiers that have been selected randomly, as described in Section 5.1.1, is trained on protocol 2 and evaluated on the validation set. Thereafter, the same classifiers are also evaluated on the test set, by selecting a fraction of the 500 binary classifiers of the ensemble ranging from 5 to 500. In this experiment, only the random approach of generating the initial binary splits for the classifiers of the ensemble is used, because optimizing the Hamming distance gets very computationally expensive if one has as many as 30 classes, as in protocol 2. Two approaches are pursued on how to select the next binary classifier to add to the output of the ensemble, namely a greedy and a random approach. In the latter, we select the next classifier randomly out of all unused classifiers. This is done ten times and allows the computation of the mean and standard deviation of the random approach. The greedy approach chooses the next classifier out of the unused classifiers such that it maximizes the CCR@FPR on the validation set for a given number of binary classifiers.

When looking at the results in Figure 6.2 it can be seen that on the validation set all the results lie between 1.03 and 1.11, which is very close. Furthermore, it can be observed, that a very low number of binary classifiers performs worse than more. Interestingly, at around 40 binary classifiers the curve starts to flat out for the random approach and only increases CCR@FPR marginally. This is also true for the negative and unknown samples on the test set. This is very similar to what we have seen in Section 6.2.1 with the random sampling approach. On the other hand, the greedy approach does even reach its best score at around 40 binary classifiers on the validation set and does thereafter start to decline again. This might surprise at first, as it goes against our hypothesis, but we always maximize the CCR@FPR in the greedy approach, so it seems like some binary classifiers are not performing so well, and they are only added later on which then start to decrease the performance of the ensemble again. This is not true for the unknowns on the test set, where the greedy approach does not have this steep decline in performance and reaches its maximum score at around 120 binary classifiers. We do see that binary classifiers which maximize the score on the validation set perform well on the negatives on the test set but rather poorly on the unknowns. Therefore, we can assume that the best performing classifiers on the validation set do not translate to also the best performance on the unknowns, which is an indication that the greedy approach is overfitting on the validation set.

In conclusion, we can say that the optimum number of binary classifiers is similar to our results from Section 6.2.1. For the following experiments, we therefore use 40 binary classifiers.

6.3.2 Combined+ vs. Combined Approach

Due to the vast amount of classes on the ImageNet protocols, with some of them being quite challenging to learn for a binary classifier, it is a reasonable action to give each binary classifier more theoretical capability to learn its specific classification task based on the features provided by the backbone network. This experiment tests the combined+ approach against the standard combined approach introduced in Section 5.2 on protocol 2 and therefore tries to answer RQ 4.2. The combined+ approach comes with the parameter k which defines the dimensionality of the individual fully connected layer per binary classifier. In this experiment, k was set to arbitrary numbers of 5, 10, and 20.

When looking at the resulting CCR@FPR scores in Table 6.6 we can see that the combined+ approach had a significantly higher performance of 6-8% compared to the combined approach. For an FPR of 10^{-1} the combined+ approaches having k equal to 10 or 20 outperformed the combined approach at about 6%, whereas the combined+ approach with k equal to 5 performed worse than the normal combined approach. The combined+ approaches using k equal to 5 or 10 were the only ones to reach an FPR of 10^{-2} , which shows their superiority when compared to the combined approach.

Regarding the optimal number for parameter k , we see that 10 does result in the best CCR@FPR

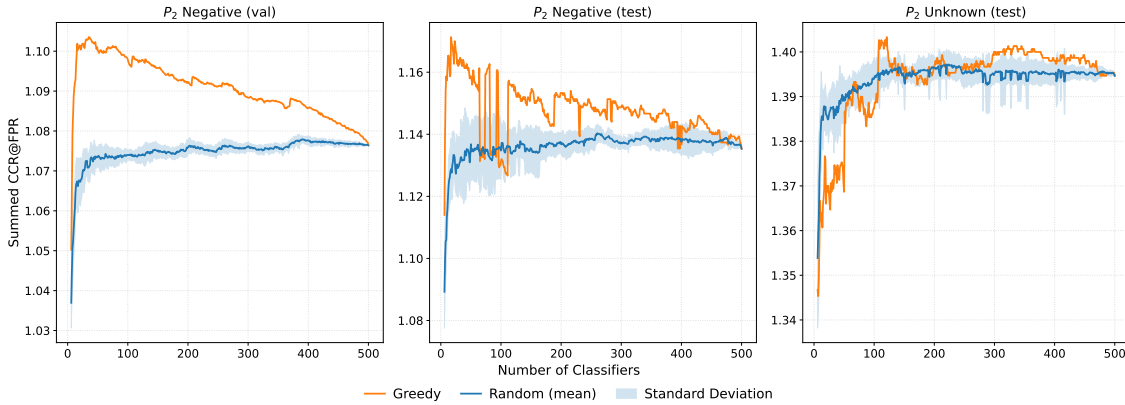


Figure 6.2: SUMMED UP CCR@FPR FOR AN INCREASING NUMBER OF CLASSIFIERS. This figure shows the number of binary classifiers on the x-axis and the summed CCR@FPR values on the y-axis for the FPR at $(10^{-3}, 10^{-2}, 10^{-1}, 1)$. The greedy approach chooses the classifier which maximizes the CCR@FPR for a given number of binary classifiers. The random approach is the mean over randomly choosing the classifiers ten times.

Table 6.6: COMBINED VS. COMBINED+ APPROACH. This table shows the CCR@FPR values for the combined and combined+ approach evaluated on the validation set of protocol 2. The latter uses two fully-connected output layers which are learned individually by each binary classifier. The parameter k is the dimensionality of the individual fully connected output layer. B indicates the number of binary classifiers in the ensemble. An empty cell means this specific CCR has not been reached for a given FPR value. The best result per FPR value is marked **bold**.

Model	B	k	CCR@FPR			
			10^{-3}	10^{-2}	10^{-1}	1
Combined	40				0.2728	0.5700
Combined+	40	5		0.0764	0.2019	0.6413
Combined+	40	10		0.0984	0.3373	0.6503
Combined+	40	20			0.3291	0.6324

in our experiment. When k is equal to 5 there most likely is too little room for the fully connected layers to adapt sufficiently to the data. On the other hand, with k being equal to 20 we do see also a worse performance than with k being equal to 10. The reason for this could be too large fully connected layers which are therefore unable to generalize enough and start memorizing the data, which results in overfitting.

In general, we see that the combined+ approach with k equal to 10 does outperform the combined approach by about 6-8% on the validation set and without using negatives for training. In this case, the result translates to the unknowns on the test set and is therefore well worth pursuing in the following experiments.

6.3.3 Binary Ensemble on all three Protocols

Figure 6.3 shows the OSCR curves for protocols 1, 2 and 3 evaluated on the negatives and unknowns of the test set. In the following, each protocol is discussed individually. The combined+ binary ensemble approach with $k = 10$ is used.

The protocol P_1 shows on the negatives that the binary ensemble does reach a slightly lower closed-set accuracy than its Softmax counterpart. However, for a marginally smaller FPR, it does outperform the Softmax-based approaches. On the unknowns it looks quite similar. The binary ensemble does outperform the Softmax-based approach by an even larger margin there. When looking at the binary ensemble which used negatives during training, it does perform worse by a small margin than the EOS-based approach until the FPR of 10^{-2} and afterward does outperform it. On the unknowns, the behavior is similar and the ensemble with negatives and EOS is outperformed by the binary ensemble without negatives at a CCR of 10^{-3} . It seems like the binary ensemble can learn the classification on the closed-set almost as well as the Softmax-based approach, while keeping the score for unknown samples lower in the case of the unknowns being very dissimilar from the known samples as in P_1 .

The protocol P_2 does show almost no differences on the negatives between the binary ensemble and the Softmax-based approach. Both perform rather poorly for low FPR values, which is mostly due to the protocol, as both, the known and negative classes are hunting dog species, which makes this a very hard task for a neural network. On the unknowns, the binary ensemble does perform slightly better than the Softmax-based approach. The binary ensemble trained with negatives does have about a 10% worse closed-set accuracy than the EOS approach. Also for smaller FPR values it is worse than its counterpart the EOS method. This is most likely because it must be very difficult for a binary classifier to distinguish among two random partitions of different hunting dog breeds, as they have very similar visual features.

The protocol P_3 also shows almost no difference between the binary ensemble and the Softmax-based approach on the negatives and unknowns. When comparing the approaches training with negative samples, the binary ensemble does have about a 5% lower closed-set accuracy than the EOS approach. On the unknown samples, these two approaches are quite similar regarding performance. The binary ensemble does have a slightly lower closed-set accuracy but a slightly better performance for smaller FPR values.

In general, we can say the binary ensemble approach seems to work better than its Softmax counterpart when no negatives for training are available and a higher score at a lower FPR is preferred over the best closed-set accuracy. Also, it works best when the unknown samples are semantically far from the known samples, such as in P_1 . This shows that the binary ensemble is well suited for out-of-distribution tasks and needs more improvement for true open-set scenarios.

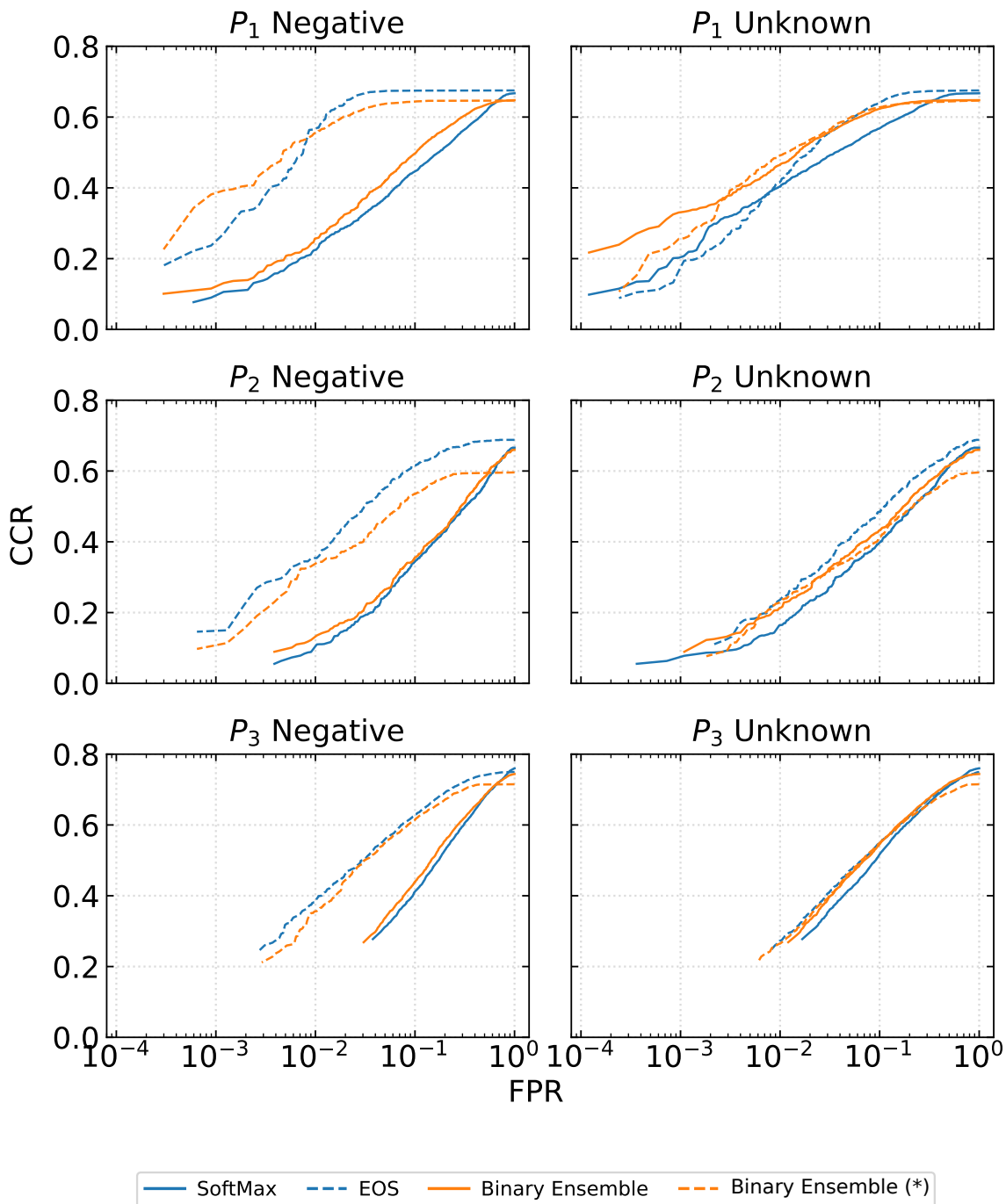


Figure 6.3: OSCR CURVES OF BINARY ENSEMBLE VS. SOFTMAX BASED APPROACHES. This figure shows the OSCR curves for negative and unknown samples on all three protocols. Color does separate binary ensemble (combined+ with $k = 10$) and Softmax-based approaches. Line style does separate incorporating negatives during training or not, whereas a solid line indicates no negatives are used for training and a dashed line shows negatives are included during training. The Binary Ensemble (*) and EOS networks have both been trained with negatives.

Discussion

This section discusses the results of the experiments carried out in Chapter 6. In the end, the limitations of the experiments are shown.

7.1 Creation of a Binary Classification Problem

We have explored two slightly different ways to create binary partitions used in the binary ensemble classifier, namely a random approach and one that maximizes the Hamming distance between the classes. The latter was expected to perform better than a random approach, however, as can be seen in Figure 6.1 the exact opposite happened. This is surprising and unexpected. To look further into it, the mean minimum Hamming distance for a given number of classifiers for the two approaches is plotted in Figure 7.1. The mean minimum Hamming distance is the minimum Hamming distance between any two classes $c \in C$, for a fixed number of binary classifiers $b \in B$. The mean is taken over ten iterations of this procedure. It can be seen that with an increasing number of binary classifiers, the minimum Hamming distance does steadily increase for both approaches. This aligns with the underlying theoretical property that more binary classifiers lead to a larger Hamming distance between the classes. Nonetheless, it is interesting to see, as the random approach does not consider the Hamming distance when creating sets. However, it might be the case that with the constraints formulated in Section 5.1.1, namely, that no classifier contained in the ensemble should learn the same binary split and also not the inverse, this results in an increasing Hamming distance already, as duplicates are avoided.

As can be further seen in Figure 7.1 the Hamming distance approach does in general only result in a slightly higher minimum Hamming distance between any two classes of the ensemble. However, in Figure 6.1 the approach maximizing the minimum Hamming distance between any two classes greatly underperformed the random approach on average, especially for lower numbers of binary classifiers. Therefore, it may be that the Hamming distance is not a desirable metric to optimize for, even though it might first seem so. Instead of maximizing the minimum Hamming distance between any two classes, maximizing it between the binary classifiers would be an alternative approach. Furthermore, it could be that the EMNIST dataset and the experiment implementation in the code are flawed and therefore misleading results are obtained. Another weak point could be that only one model has been trained for the experiment in Figure 6.1, which has thereafter been evaluated 10 times. The results may look different when multiple models are trained, as then the underlying partitions of the binary classifiers are also most likely vastly different.

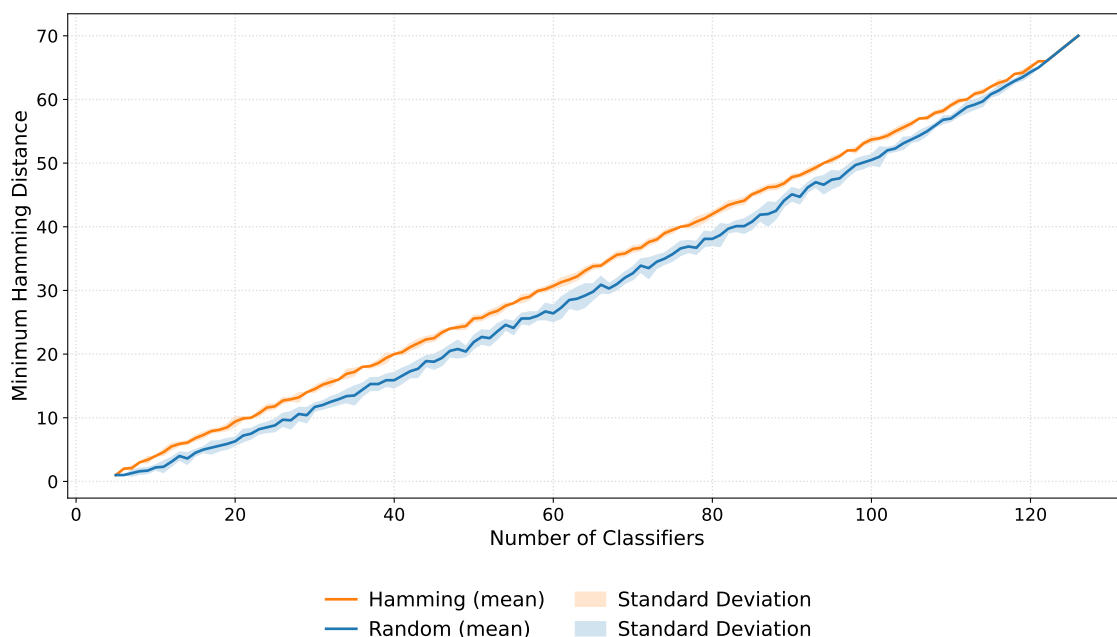


Figure 7.1: MEAN MINIMUM HAMMING DISTANCE FOR THE RANDOM AND HAMMING APPROACH. This figure shows the minimum Hamming distance between any two of 10 classes on the y-axis. The number of binary classifiers is shown on the x-axis. The mean and standard deviation are calculated over 10 iterations.

7.2 Analysis of High-Confidence Misclassifications and Rare Errors

Quantitative and qualitative evaluation of the model error allows us to grasp where the underlying binary classifier is struggling regarding its closed-set and open-set tasks. For this evaluation, the binary ensemble using the combined+ approach with $k = 10$, trained without negatives on protocols 1, 2, and 3 is evaluated. We look at the binary ensemble trained with no negatives, as this network was able to outperform its Softmax counterpart, in Figure 6.3, for lower FPR-values. Out of the 5800 known samples in P_1 , 2043 have been classified wrong; respectively, 510 out of 1500 for P_2 , and 1934 out of 7550 for P_3 . These numbers make it well worth exploring where the models make mistakes.

For the binary ensemble model on P_1 , the top-3 misclassified classes are misclassified more than 72% as can be seen in Table 7.1. Especially the known class “English Foxhound” gets misclassified 80% of the time. When it is misclassified, it is classified as a “Walker Hound” 40% of the time. This is an especially high rate. When looking at a misclassified sample in Figure 7.2 for P_1 in the left column and comparing it to a dog from the class “Walker Hound” this is, however, understandable to us humans. Without extensive knowledge, the dogs seem to have similar visual characteristics. This makes it difficult for the model to distinguish between those two dog breeds. A potential improvement could be made by including more training samples of those two dog breeds. Another interesting point to touch upon is the mean score of 0.80 or above, which results over all three protocols after applying Equation 5.10 to the scores of the binary ensemble. This shows us that the binary classifiers are often completely wrong with their prediction, which

Table 7.1: MISCLASSIFICATION ANALYSIS FOR TOP 3 CLASSES PER PROTOCOL. The table shows the ground truth class, overall misclassification rate (Misc.), score statistics (Score) including mean (μ), median (M), and standard deviation (σ), where 1 is the best score and 0 is the worst. The most common incorrect class prediction (Top Misc.), and the percentage of misclassifications attributed to this top misclassified class (TMR).

Protocol	Ground Truth	Misc.	Score ($\mu/M/\sigma$)	Top Misc.	TMR
P_1	English Foxhound	80%	0.88/0.91/0.11	Walker Hound	40%
	Appenzeller	74%	0.91/0.97/0.10	Entlebucher	54%
	Bloodhound	72%	0.80/0.79/0.11	Basset	19%
P_2	English Foxhound	82%	0.90/0.94/0.10	Walker Hound	32%
	Bloodhound	60%	0.87/0.91/0.11	Rhodesian Ridgeback	27%
	American Staffordshire Terrier	60%	0.87/0.91/0.11	Staffordshire Bullterrier	24%
P_3	Moving Van	68%	0.88/0.88/0.10	Trailer Truck	32%
	Ear	64%	0.89/0.99/0.13	Corn	72%
	Bloodhound	62%	0.80/0.77/0.14	Rhodesian Ridgeback	26%

is reasonable to assume when considering their misclassification rate.

On the other hand, it is also well worth looking at samples from the classes that have been classified correctly most of the time in Figure 7.3. Especially interesting are the samples from P_3 as the known samples are composed of classes that are widely different and do therefore have diverse ancestors. The sample in Figure 7.3(h) is interesting, where the monkey of class “Proboscis Monkey” was classified as an “Eel”. If we use our imagination, we can see why the model might have made that mistake, but we do not know if the model relies on such visual similarities at all.

After looking at the misclassification for known samples it is worth looking into some high-confidence misclassification for unknown samples in Figure 7.4. In P_1 all the dog classes are known samples and the unknowns consist of non-animal classes. For samples in Figure 7.4(a) and Figure 7.4(b), the dog present in the pictures was correctly predicted; however, they should have been rejected because the images are labeled as “muzzle” and “minivan” respectively. The sample in Figure 7.4(c) was predicted as “Miniature Pinscher” with a very high score of 0.99, which is wrong. This example shows clearly that there is enough room for the model to improve on its open-set task, as a “lighter” is semantically very different from a dog. Considering P_2 , hunting dog classes are known samples, while other 4-legged animal classes are unknown. Both the samples in Figure 7.4(e) and Figure 7.4(f) were predicted as “Norwegian Elkhound” when they should have been rejected as unknown. Although both samples are wolves according to their ImageNet class. The prediction of the model is wrong and “Norwegian Elkhound” and “wolves” are not closely related in the WordNet hierarchy, but the “Norwegian Elkhound” descended at least partially from a wolf (Vilà et al., 1999). In P_3 the known and unknown classes are both a mix of classes of animals, plants, and objects. The sample in Figure 7.4(i) is again a very reasonable misclassification by the model. Samples in Figure 7.4(g) and Figure 7.4(h) are clearly misclassified. “Granny Smith” shares some visual similarities with “Acorn”, such as both being green. However, we do not know why the model made those highly confident mistakes, but it seems like the network would benefit from more robust features, which hopefully lead to better classifications.

In conclusion, we can say that on all three protocols, the pattern for common misclassifications is very similar. The misclassifications are often visually close to the ground truth class. Some samples are also not attributable to one class, as can be seen in Figure 7.2(e), where the model predicted “Trailer Truck” but the ground truth is “Moving Van”. When looking at a sample from the class “Trailer truck” in Figure 7.2(f) it is clear that the prediction is theoretically correct, and the underlying dataset is ambiguous. More samples of unknown classes that received high confi-

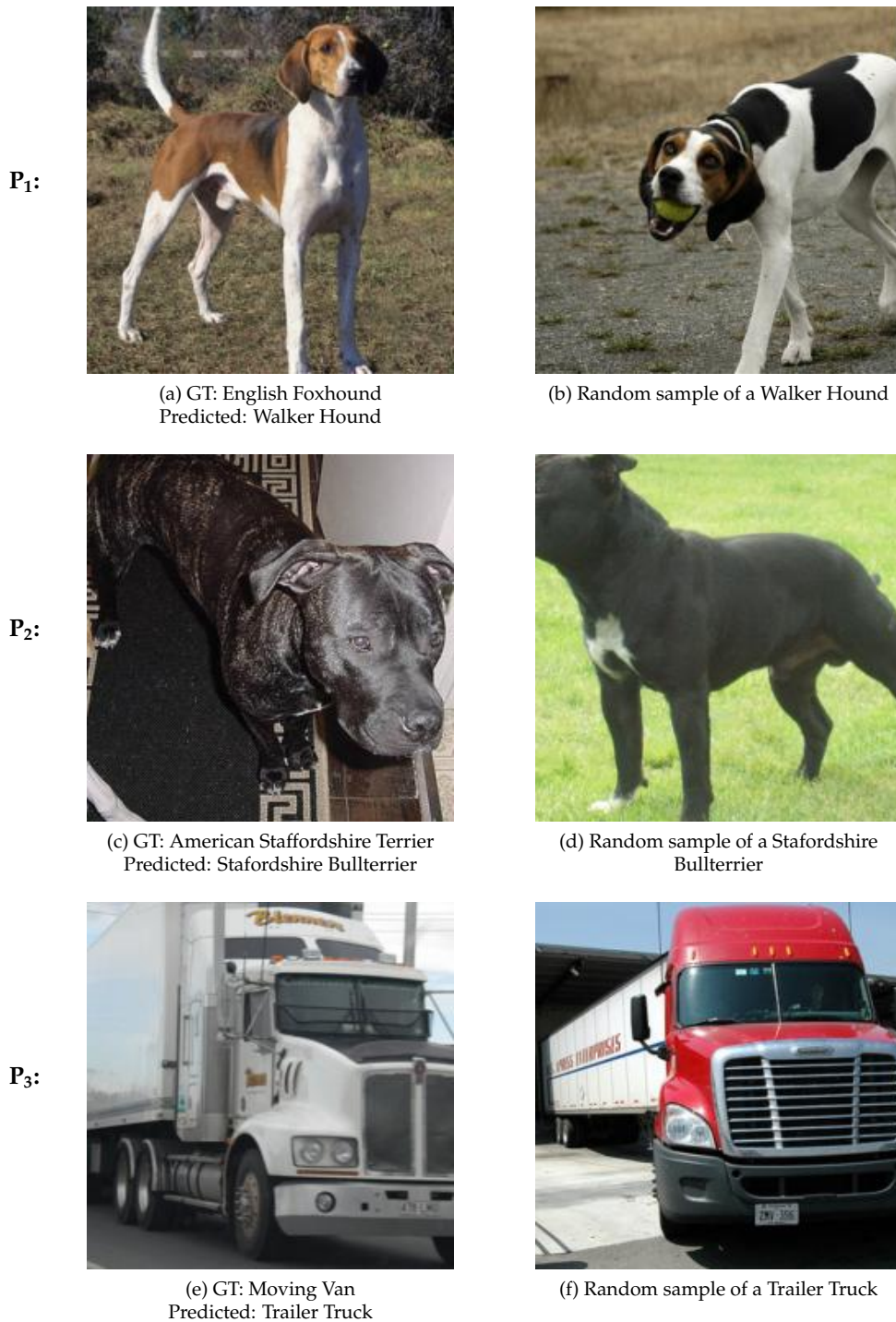


Figure 7.2: FREQUENT MISCLASSIFICATIONS. *Misclassification examples for different protocols (P_1 , P_2 , P_3). For each protocol, the left image shows a misclassified example with its ground truth and predicted label and the right image shows a sample of the class it was misclassified as.*

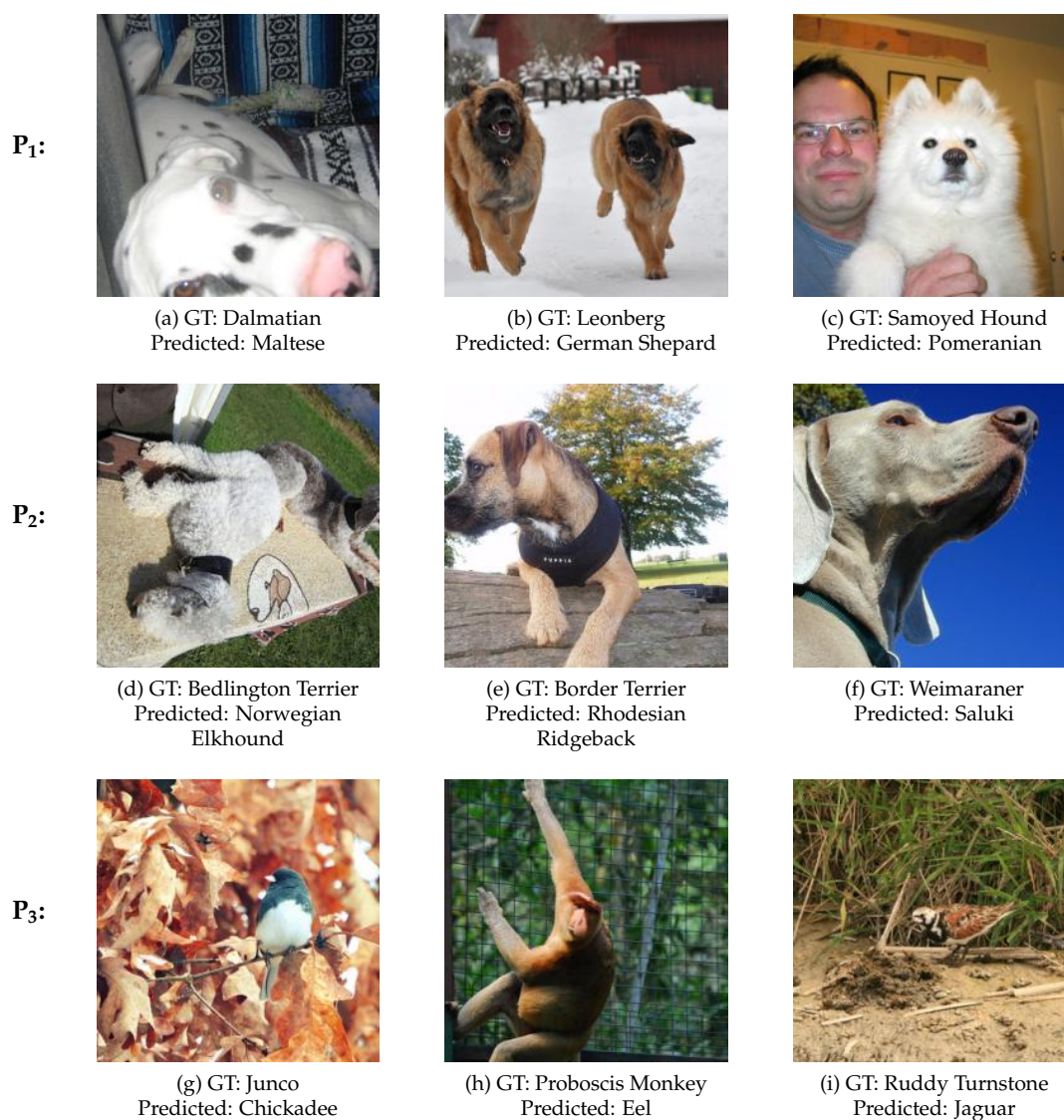


Figure 7.3: CLASSIFICATION OUTLIERS FOR DIFFERENT PROTOCOLS. *These outliers represent rare misclassifications where most samples of the same class are correctly classified.*

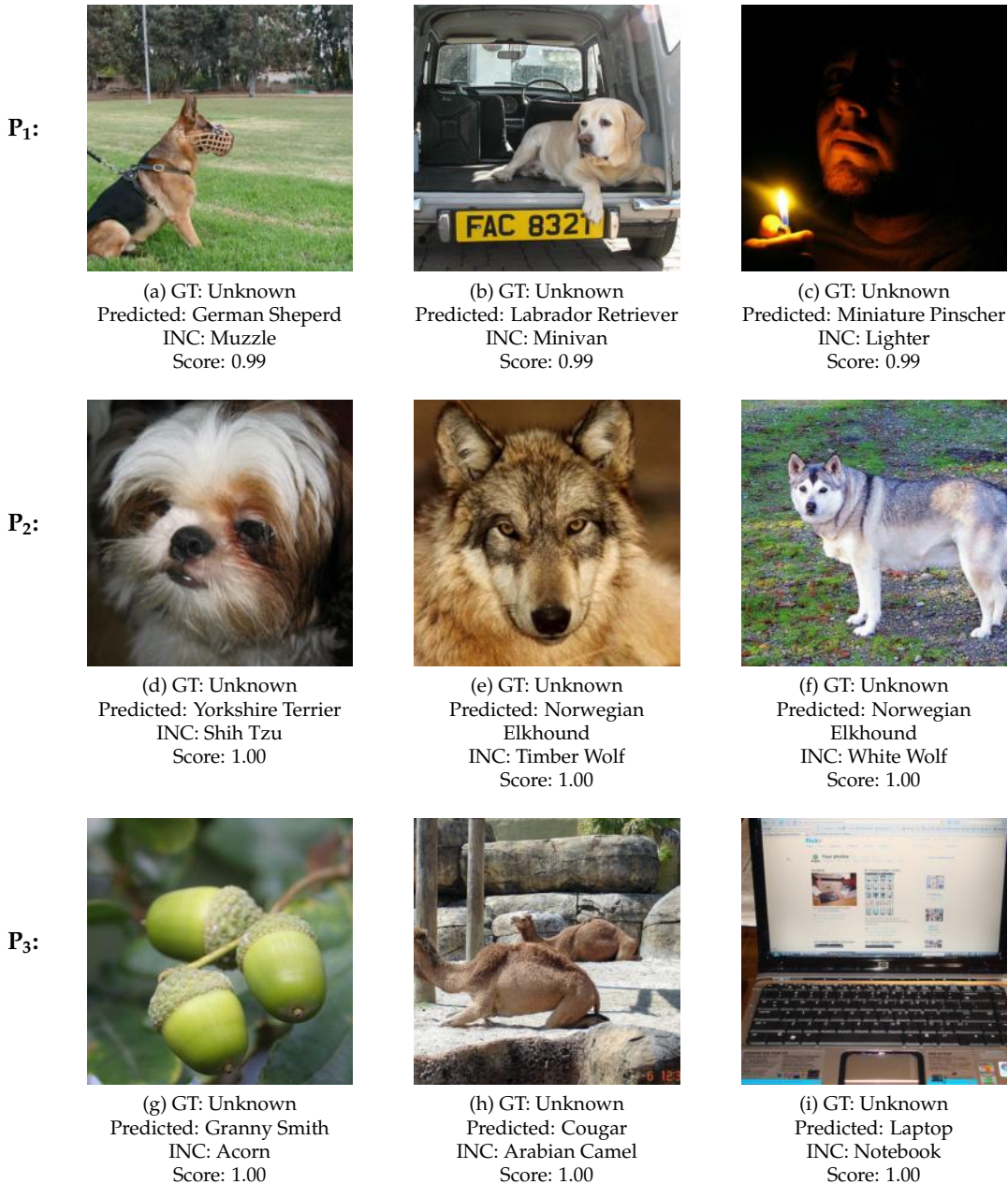


Figure 7.4: HIGH CONFIDENT PREDICTIONS FOR UNKNOWNNS. *The samples shown in the figure do all have a ground truth (“GT”) class of unknown. “Predicted” shows the predicted class with the probability score (“Score”). The ImageNet class (“INC”) shows the corresponding class from the ImageNet dataset, where the sample is not labeled as an unknown class.*

dence for a known class can be found in Appendix A.

7.3 Analysis of Score Distributions

We apply Equation 5.10 to the scores of the binary ensemble to receive probabilities that are comparable with the scores of the Softmax-based approaches. For the known samples, we use the probability assigned to the correct class. For the negative/unknown class, we select the maximum probability over all known classes. When looking at these scores across the protocols P_1 , P_2 , and P_3 between Softmax and binary ensemble-based approaches in Figure 7.5, we can see interesting differences. When inspecting the binary ensemble approaches, we see that many known samples almost receive the maximum possible score and many others get a score of around 0.5. The score of 0.5 is also what we would expect from a randomly guessing model. Interestingly, the binary ensemble trained with negative samples does give more known samples a low score than the binary ensemble without negatives. This is similar to what we see when training with Softmax vs. EOS, where the Softmax-based method also tends to give more known samples a higher score. When looking at the score distribution for the known samples between the binary ensembles and the Softmax approaches, it is recognizable that the scores are distributed more uniformly and are less concentrated around the extreme scores.

When looking at the scores for the unknown samples, many of them receive a score of around 0.5 for P_1 when using binary ensembles. This is especially great to see for the binary ensemble trained without negatives and indicates that it can successfully reject unknowns to a certain degree. This score distribution also explains why the binary ensemble trained without negatives performed similarly well to the EOS approach on the unknowns in Figure 6.3, in the top-right graph. In P_2 and P_3 the binary ensemble does give many unknowns a very high score of 1. However, when compared to its counterpart, the Softmax approach, we do see that for the binary ensemble still fewer samples are given such a high score. With the incorporation of negatives, the binary ensemble can give a big part of the unknown samples a low score of around 0.5, similar to what EOS does, when compared to the Softmax approach. With negatives, more unknown samples receive a lower score from the binary ensemble. However, this is counteracted by more known samples getting a lower score too, which again worsens the performance. This is very interesting to point out as this effect can also be seen for Softmax and EOS but there, when incorporating negatives, the effect of known samples receiving a very low score afterward is smaller. In P_2 and P_3 , this effect is very strong for the known samples, many of them get a low score when training the ensemble with negatives. This may be due that in both protocols there are many negative samples in total during training. In comparison there are not so many known samples per class, so likely, the model is not able to learn a good representation of the known samples because the negatives are overrepresented. An approach to tackle this problem could be to focus more on hard-to-classify known samples by using e.g., a specialized loss function.

The binary ensemble trained with negatives does a good job of classifying negative samples as negative by giving them a score of 0.5 on all three protocols. It does this even better than the EOS approach, which has more negatives with a higher score. Interestingly, the binary ensemble trained without negatives in P_1 can give most of the negative samples a rather low score when compared to the Softmax approach, which does give many negative samples a high score. For P_2 and P_3 , this is not the case anymore and many negative samples do receive a higher score by the binary ensemble.

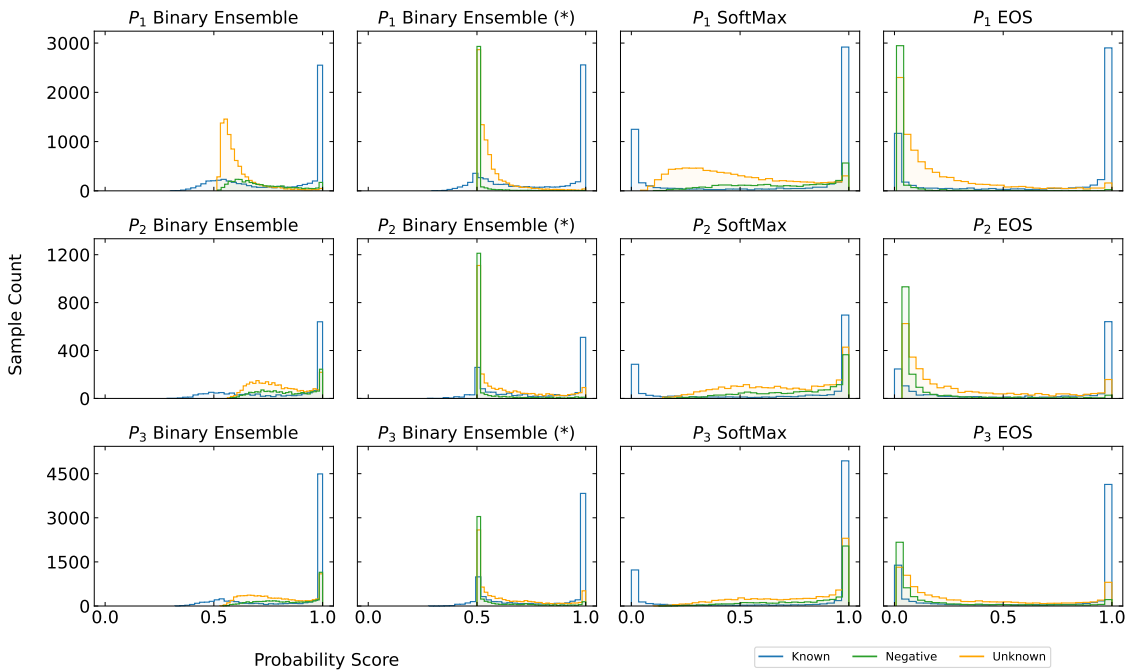


Figure 7.5: SCORE DISTRIBUTIONS. Score distributions for known, unknown, and negative samples across different protocols. Each row represents a different protocol, while columns show various algorithms. The x-axis represents the probability score, and the y-axis shows the sample count.

7.4 Limitations

Despite the high effort that has been carried out to design this master thesis so general and comprehensive, there are some limitations.

The hyperparameters were not all tuned to the same extension. Especially the learning rate, the optimizer used, and batch size have not been tuned, although they might bring a performance increase. Rather, a set of hyperparameters was taken that performs adequately, and then other hyperparameters were tuned, that are more specific to the binary ensemble such as the number of binary classifiers or hidden dimension size of the fully connected layer in the combined+ network. Following this approach, the number of binary classifiers in the ensemble was once chosen on protocol 2 and set as 40. This number may differ for the protocol 1 and 3, as they are quite different in their composition and number of total classes. Although the protocols are different, the backbone of all binary ensembles and also the Softmax-based approaches was always the ResNet-50 on the ImageNet protocols. This may not be optimal and was mainly done to keep the complexity at a reasonable level. It may be that for Protocol 3, which is very difficult considering its open-set task, a larger backbone network, such as ResNet-101, would boost the performance, as it could allow the network to learn more extensive feature representations

Considering the dataset, we have seen in the discussion that some labels of samples are ambiguous, like depicted in Figure 7.2(e), which could have a bad influence on the network's performance. Although filtering out ambiguous labeled data is very labor intensive, it might increase the model's closed- and open-set performance substantially. Furthermore, for the preliminary experiments conducted on the EMNIST dataset, it is unclear how much they transfer to the ImageNet protocols. The EMNIST dataset may not be optimal for preliminary experiments, and

another dataset that is similar to ImageNet could be used, such as CIFAR-10.

Finally, we need to point out that the binary ensemble approach, which uses completely separate networks as binary classifiers, performed very well on the EMNIST dataset, as we have seen in Table 6.2. Hence, the separate approach could potentially also yield such good performance on the ImageNet protocols. However, due to the immense training time it takes to train with this separate approach, this was not done in the scope of this thesis.

Conclusion

8.1 Summary

The aim of this master's thesis is the use of binary ensemble classifiers for open-set classification, which is a novel approach when compared to the mostly Softmax-based approaches that are used otherwise. The thesis covers the procedure that is needed to be able to use an ensemble of binary classifiers for open-set classification. We started by experimenting with different methods to create binary partitions to obtain a binary classification task out of a multi-class problem. We used a random approach with only a few constraints to prevent any two sets containing the same partition. The second approach for creating binary partitions used the same constraints but on top chose the sets so that the resulting Hamming distance between the classes is maximized among all classes. We found that maximizing this class-wise distance did not result in better performance of the whole binary ensemble. Furthermore, we conducted experiments on three different methods on how to obtain a single prediction from the binary ensemble out of the underlying binary classifiers. For this, we compared an approach that uses logits directly against two methods that use probabilities obtained through sigmoid activation. Resulting we found that the sigmoid approach is superior to using the logits directly.

For the networks, used to train the binary ensemble classifiers, we have tried different architectures based on CNNs, such as fully separate and combined approaches. The prior trained a completely separate binary classifier for each binary classification problem, whereas the latter uses a shared backbone, which is used for different binary classifiers simultaneously. The separate approach performed substantially better than the combined one in the preliminary experiments on the EMNIST dataset. However, due to its large computational needs and long training time, we introduced the combined+ approach, which is a mixture of these two approaches. With the extra fully connected output layer per classifier in the combined+ approach, we showed that it is possible to improve the performance when compared to the combined approach substantially.

We incorporated negative samples during training with the goal of improving the classifiers. We pursued two different approaches: the *integrated* approach included the negatives as just an extra class, whereas the *equal* approach trained the network to predict 0.5 for negatives, for a more balanced model output. We have shown that training the negatives with the *equal* approach, for a binary classifier, is superior to just training them as an extra class, but still inferior to the Softmax-based EOS approach.

Concluding, we can say that it is possible to train an ensemble of binary classifiers for an open-set problem. We have seen that on the ImageNet protocols with increasing difficulty for the open-set task, the binary ensemble trained without negative samples can outperform Softmax-based approaches for false positive rates smaller than one. For the latter, which resembles the closed-set accuracy, the binary ensemble approach is often worse than using Softmax-based methods. It is

therefore worth considering an ensemble of binary classifiers if there are no negative samples to train with and the performance for a low false positive rate on the unknowns is to be maximized.

8.2 Future Work

Future research directions that can be considered to improve binary ensembles for open-set classification are outlined in the following.

Although in theory, it makes sense that a larger Hamming distance between classes should result in superior models, this was not true in our case. Another direction would be to maximize the Hamming distance between the binary classifiers instead of the classes, which could potentially come with an increase in performance. We can say that further research is needed on the influence of different methods used to create binary partitions, such as the random or Hamming approach. Additionally, completely different methods like grouping semantically similar or dissimilar classes can be explored.

In our experiments, the binary classifier's outputs are all weighted the same. This could be a problem since, in theory, there most likely exist classifiers that work better than others for certain samples. Using a weighting mechanism for the outputs could be interesting, as well as using some sort of attention between the different classifiers in hopes of better classifications, all while being aware of potential overfitting on the training set. We have seen that especially classes that are very semantically and visually similar tend to get misclassified on the closed-set. This could potentially be improved by using a mixture-of-experts model, trained on these hard-to-classify examples, which could improve closed-set accuracy for ensembles of binary classifiers.

Attachments

Figures [A.1](#), [A.2](#), and [A.3](#) show more highly confident misclassifications of unknown samples over all three ImageNet protocols and are an extension of Section [7.2](#). Especially interesting are the samples in Figure [A.1](#), because the network should only be able to classify dogs and should reject everything else as unknown. Further in Figure [A.3](#) it is very interesting to see that there exist some very ambiguous classes, one of which is known and the other one belongs to the unknowns. An example is “Laptop” vs. “Notebook”. Moreover, there exist classes which are very similar but still different such as “Leopard” vs. “Snow Leopard”.

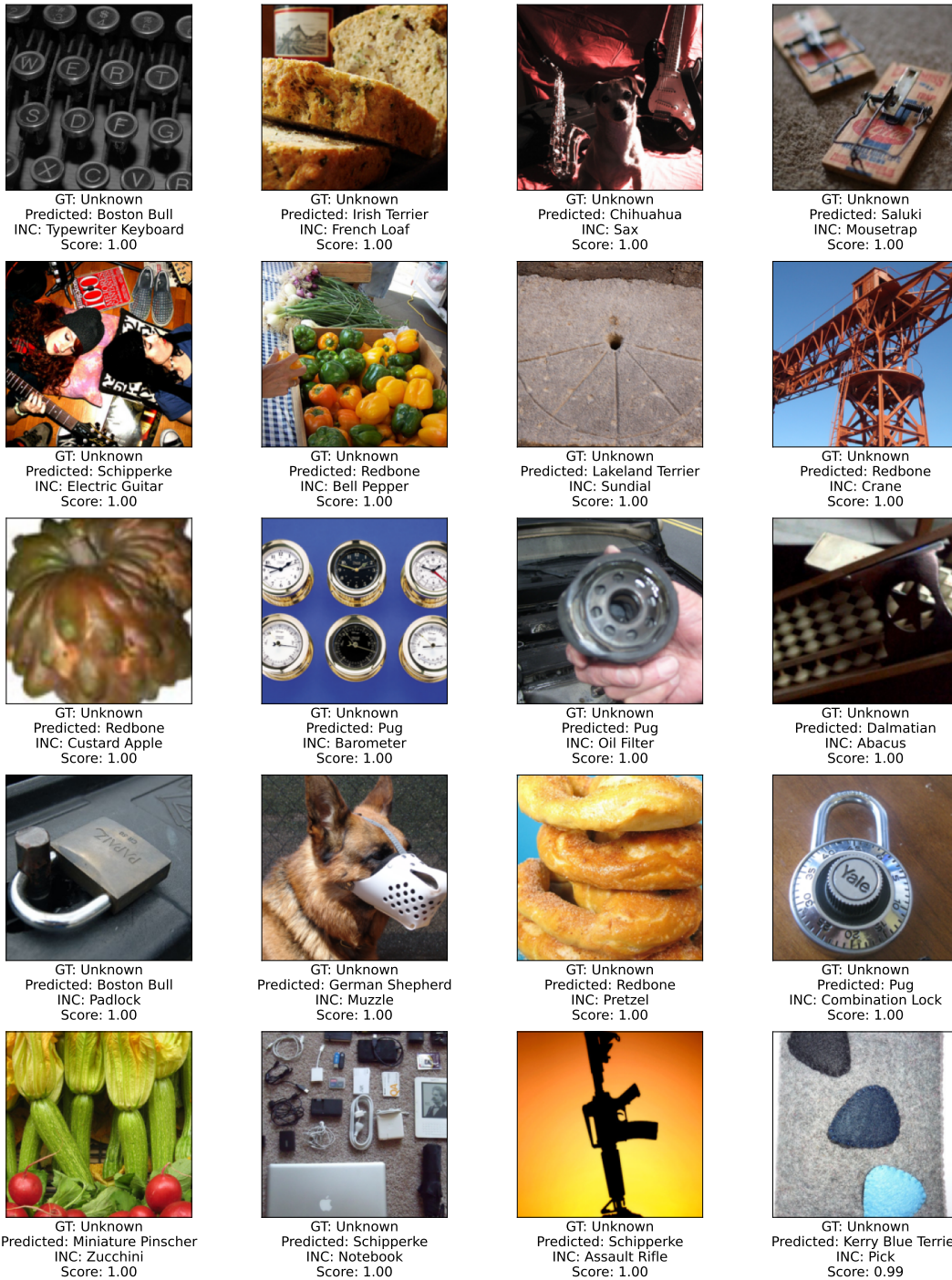


Figure A.1: HIGH CONFIDENT PREDICTIONS FOR UNKNOWNNS ON PROTOCOL 1. The samples shown in the figure do all have a ground truth (“GT”) class of unknown. “Predicted” shows the predicted class with the probability score (“Score”). The ImageNet class (“INC”) shows the corresponding class from the ImageNet dataset, where the sample is not labeled as an unknown class.



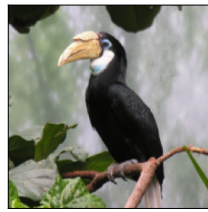
Figure A.2: HIGH CONFIDENT PREDICTIONS FOR UNKNOWN ON PROTOCOL 2. The samples shown in the figure do all have a ground truth (“GT”) class of unknown. “Predicted” shows the predicted class with the probability score (“Score”). The ImageNet class (“INC”) shows the corresponding class from the ImageNet dataset, where the sample is not labeled as an unknown class.



GT: Unknown
 Predicted: Limpkin
 INC: Alligator Lizard
 Score: 1.00



GT: Unknown
 Predicted: Laptop
 INC: Notebook
 Score: 1.00



GT: Unknown
 Predicted: Hornbill
 INC: Toucan
 Score: 1.00



GT: Unknown
 Predicted: Lionfish
 INC: Cardoon
 Score: 1.00



GT: Unknown
 Predicted: Lionfish
 INC: Cardoon
 Score: 1.00



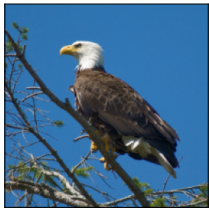
GT: Unknown
 Predicted: Granny Smith
 INC: Acorn
 Score: 1.00



GT: Unknown
 Predicted: Laptop
 INC: Notebook
 Score: 1.00



GT: Unknown
 Predicted: Laptop
 INC: Notebook
 Score: 1.00



GT: Unknown
 Predicted: Kite
 INC: Bald Eagle
 Score: 1.00



GT: Unknown
 Predicted: Cougar
 INC: Ram
 Score: 1.00



GT: Unknown
 Predicted: File
 INC: Chiffonier
 Score: 1.00



GT: Unknown
 Predicted: Leopard
 INC: Snow Leopard
 Score: 1.00



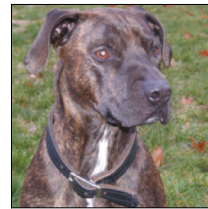
GT: Unknown
 Predicted: Laptop
 INC: Notebook
 Score: 1.00



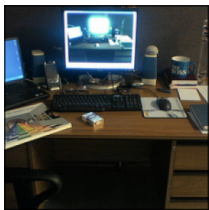
GT: Unknown
 Predicted: File
 INC: Chiffonier
 Score: 1.00



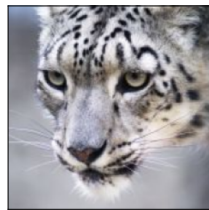
GT: Unknown
 Predicted: Corn
 INC: Bell Pepper
 Score: 1.00



GT: Unknown
 Predicted: Staffordshire Bullterrier
 INC: American Staffordshire Terrier
 Score: 1.00



GT: Unknown
 Predicted: Desktop Computer
 INC: Notebook
 Score: 1.00



GT: Unknown
 Predicted: Leopard
 INC: Snow Leopard
 Score: 1.00



GT: Unknown
 Predicted: Bolete
 INC: Mushroom
 Score: 1.00



GT: Unknown
 Predicted: Granny Smith
 INC: Bell Pepper
 Score: 1.00

Figure A.3: HIGH CONFIDENT PREDICTIONS FOR UNKNOWN ON PROTOCOL 3. *The samples shown in the figure do all have a ground truth ("GT") class of unknown. "Predicted" shows the predicted class with the probability score ("Score"). The ImageNet class ("INC") shows the corresponding class from the ImageNet dataset, where the sample is not labeled as an unknown class.*

List of Figures

5.1	Random Partitioning of Classes	20
5.2	Class-wise Distance	21
5.3	Separate vs. Combined vs. Combined Large vs. Combined+ Approach	23
6.1	Random vs. Hamming Distance Approach and Number of Binary Classifiers	31
6.2	Summed up CCR@FPR for an Increasing Number of Classifiers	36
6.3	OSCR Curves of Binary Ensemble vs. Softmax Based Approaches	38
7.1	Mean Minimum Hamming Distance for the Random and Hamming Approach	40
7.2	Frequent Misclassifications	42
7.3	Classification Outliers for Different Protocols	43
7.4	High Confident Predictions for Unknowns	44
7.5	Score Distributions	46
A.1	High Confident Predictions for Unknowns on Protocol 1	52
A.2	High Confident Predictions for Unknowns on Protocol 2	53
A.3	High Confident Predictions for Unknowns on Protocol 3	54

List of Tables

4.1	EMNIST Specification for Preliminary Experiments	16
5.1	Example of Binary Class Encoding	24
5.2	Evaluation Approaches and Methods for Training with Negatives	27
6.1	Training Parameters of Preliminary Experiments	30
6.2	Separate and Combined Binary Classifiers	32
6.3	Evaluation Metrics	33
6.4	Training with Negatives	34
6.5	Training Parameters of ImageNet Experiments	34
6.6	Combined vs. Combined+ Approach	36
7.1	Misclassification Analysis for Top 3 Classes per Protocol	41

Bibliography

- Aran, O. and Akarun, L. (2010). A multi-class classification strategy for Fisher scores: Application to signer independent sign language recognition. *Pattern Recognition*, 43(5):1776–1788.
- Araújo, T., Aresta, G., Castro, E., Rouco, J., Aguiar, P., Eloy, C., Polónia, A., and Campilho, A. (2017). Classification of breast cancer histology images using convolutional neural networks. *PloS one*, 12(6):1–14.
- Bendale, A. and Boulton, T. E. (2016). Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- Bhoumik, A. (2021). Open-set classification on ImageNet. Master’s thesis, University of Zurich.
- Bisgin, H., Palechor, A., Suter, M., and Günther, M. (2024). Large-scale evaluation of open-set image classification techniques. *arXiv preprint arXiv:2406.09112*.
- Breiman, L. (1996). Bagging predictors. *Machine learning*, 24:123–140.
- Cao, A., Luo, Y., and Klabjan, D. (2021). Open-set recognition with gaussian mixture variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6877–6884.
- Chen, G., Peng, P., Wang, X., and Tian, Y. (2021). Adversarial reciprocal points learning for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8065–8081.
- Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Machine Learning—EWSL-91: European Working Session on Learning Porto, Portugal, March 6–8, 1991 Proceedings 5*, pages 151–163. Springer.
- Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks*, pages 2921–2926.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.
- Dhamija, A. R., Günther, M., and Boulton, T. (2018). Reducing network agnostophobia. *Advances in Neural Information Processing Systems*, 31.
- Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.
- Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976.

- Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378.
- Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.
- Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44(8):1761–1776.
- Ge, Z., Demyanov, S., Chen, Z., and Garnavi, R. (2017). Generative OpenMax for multi-class open set classification. *British Machine Vision Conference*.
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Grother, P. J. and Hanaoka, K. (1995). NIST special database 19. *Handprinted forms and characters database, National Institute of Standards and Technology*, 10:69.
- Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Hendrycks, D., Basart, S., Mazeika, M., Zou, A., Kwon, J., Mostajabi, M., Steinhardt, J., and Song, D. (2022). Scaling out-of-distribution detection for real-world settings. *International Conference on Machine Learning*.
- Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations*.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- 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*.
- Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat’s striate cortex. *The Journal of physiology*, 148(3):574.
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

- Kaur, C. and Garg, U. (2023). Artificial intelligence techniques for cancer detection in medical image processing: A review. *Materials Today: Proceedings*, 81:806–809.
- Knerr, S., Personnaz, L., and Dreyfus, G. (1990). Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing: algorithms, architectures and applications*, pages 41–50. Springer.
- Koh, D.-M., Papanikolaou, N., Bick, U., Illing, R., Kahn Jr, C. E., Kalpathi-Cramer, J., Matos, C., Martí-Bonmatí, L., Miles, A., Mun, S. K., et al. (2022). Artificial intelligence and machine learning in cancer imaging. *Communications Medicine*, 2(1):133.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.
- Lan, G., Gao, Z., Tong, L., and Liu, T. (2022). Class binarization to neuroevolution for multiclass classification. *Neural Computing and Applications*, 34(22):19845–19862.
- 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.
- LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (2002). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.
- LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276):2.
- Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016a). SSD: Single shot multibox detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer.
- Liu, W., Wen, Y., Yu, Z., and Yang, M. (2016b). Large-margin softmax loss for convolutional neural networks. In *International Conference on Machine Learning*, pages 507–516. PMLR.
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.
- Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.
- Matan, O., Kiang, R., Stenard, C., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., and Le Cun, Y. (1990). Handwritten character recognition using neural network architectures. In *4th USPS advanced technology conference*, volume 2, pages 1003–1011.
- Metzner, C., Schilling, A., Traxdorf, M., Tziridis, K., Maier, A., Schulze, H., and Krauss, P. (2022). Classification at the accuracy limit: facing the problem of data ambiguity. *Scientific reports*, 12(1):22121.
- Miller, G. A. (1998). Nouns in wordnet. *WordNet: An electronic lexical database*, pages 23–46.

- Palechor, A., Bhoumik, A., and Günther, M. (2023). Large-scale open-set classification protocols for ImageNet. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 42–51.
- Panareda Busto, P. and Gall, J. (2017). Open set domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 754–763.
- Pawara, P., Okafor, E., Groefsema, M., He, S., Schomaker, L. R., and Wiering, M. A. (2020). One-vs-one classification for deep neural networks. *Pattern Recognition*, 108:107528.
- Rasti, P., Ahmad, A., Samiei, S., Belin, E., and Rousseau, D. (2019). Supervised image classification by scattering transform with application to weed detection in culture crops of high density. *Remote Sensing*, 11(3):249.
- Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Deprieto, M., Dillon, J., and Lakshminarayanan, B. (2019). Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32.
- Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.
- Rudd, E. M., Günther, M., and Boulton, T. E. (2016). MOON: A mixed objective optimization network for the recognition of facial attributes. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part V 14*, pages 19–35. Springer.
- Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boulton, T. E. (2012). Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772.
- Scheirer, W. J., Jain, L. P., and Boulton, T. E. (2014). Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324.
- Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*.
- van den Bergh, L. (2023). Improved losses for open-set classification. Master’s thesis, University of Zurich.
- Vareto, R. H., Günther, M., and Schwartz, W. R. (2023). Open-set face recognition with neural ensemble, maximal entropy loss and feature augmentation. In *2023 36th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 55–60. IEEE.
- Vareto, R. H. and Schwartz, W. R. (2020). Unconstrained face identification using ensembles trained on clustered data. In *2020 IEEE International Joint Conference on Biometrics*, pages 1–8.
- Vaze, S., Han, K., Vedaldi, A., and Zisserman, A. (2022). Open-set recognition: A good closed-set classifier is all you need? In *International Conference on Learning Representations*.
- Verma, K. K., Singh, B. M., and Dixit, A. (2022). A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system. *International Journal of Information Technology*, 14(1):397–410.
- Vilà, C., Maldonado, J. E., and Wayne, R. K. (1999). Phylogenetic relationships, evolution, and genetic diversity of the domestic dog. *Journal of Heredity*, 90(1):71–77.

- Wang, A., Zhang, W., and Wei, X. (2019). A review on weed detection using ground-based machine vision and image processing techniques. *Computers and electronics in agriculture*, 158:226–240.
- Wilber, M. J., Scheirer, W. J., Leitner, P., Heflin, B., Zott, J., Reinke, D., Delaney, D. K., and Boulton, T. E. (2013). Animal recognition in the Mojave Desert: Vision tools for field biologists. In *2013 IEEE Workshop on Applications of Computer Vision*, pages 206–213. IEEE.
- Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.
- Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., Tay, F. E., Feng, J., and Yan, S. (2021). Tokens-to-token ViT: Training vision transformers from scratch on ImageNet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567.
- Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2021). Dive into deep learning. *arXiv preprint arXiv:2106.11342*.
- Zhang, S., Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2018). Towards reaching human performance in pedestrian detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):973–986.
- Zhou, D.-W., Ye, H.-J., and Zhan, D.-C. (2021). Learning placeholders for open-set recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.
- Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.