UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Mixup and Metric Learning in Out-of-Distribution Detection

Author:
Oleksandra HUTOR

Supervisor:
Dr. Giorgos TOLIAS

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences and Information Technologies
Faculty of Applied Sciences

Lviv 2023

Declaration of Authorship

I, Oleksandra HUTOR, declare that this thesis titled, "Mixup and Metric Learning in Out-of-Distribution Detection" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Mixup and Metric Learning in Out-of-Distribution Detection

by Oleksandra HUTOR

Abstract

In this work, we analyse the problem of out-of-distribution detection, which includes inlier classification and outlier detection, and the use of two methods for it, i.e., Mixup and metric learning, as well as their combination. Inspired by the use of Mixup between inliers and outliers in MixOE (Zhang et al., 2023), our first objective is to identify the key ingredients of their method and to combine it with seven other Mixup techniques. We also change the size and diversity of the auxiliary outlier dataset, which is used for training. We find that in the fine-grained OoD setting, where outliers come from the same domain as inliers, mixing only the label with the one that comes from the uniform probability distribution without mixing the inlier and outlier training samples has a similar performance as mixing both pairs of training samples and labels. At the same time, in the coarse-grained settings, where outliers come from a completely different domain than inliers, the more outliers are used for mixing, the better the detection performance is. Our second objective is the investigation of metric learning in the form of triplet loss for the same setup with different types of triplet combinations, some of which are created using Mixup. In the coarse-grained settings, the triplet combination with only inlier classes performs better than such combinations with outliers, mixed or not. Finally, we combine MixOE and our metric learning approach to show that, for some datasets, the detection performance in the coarse-grained settings is comparably larger than the previous best result, MixOE.

Acknowledgements

I want to express extreme gratitude to my supervisor, Dr. Giorgos Tolias, for guiding me through this thesis and my beginnings in academic research. This work would not have been accomplished without his valuable advice and supervision. I also thank the Center for Machine Perception at the Czech Technical University in Prague for providing the computational resources.

I am grateful to the Faculty of Applied Sciences at the Ukrainian Catholic University, especially Oles Dobosevych, M.Sc. and Dr. Taras Firman, for encouraging me to grow academically, personally and professionally at different times during these four amazing but nonetheless challenging years. During the last year, none of that would be possible without the Armed Forces of Ukraine defending our country and giving me the ability to study, work and stay alive.

I feel deeply thankful that I could share this unforgettable four-year journey with my friends offline and online.

Last but not least, I want to thank my parents and my sister for their indefinite support of all of my endeavours.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1

2 Related work 3
2.1 Mixup . . . 3
2.1.1 Input Mixup and manifold intrusion . . . 3
2.1.2 Manifold Mixup . . . 4
2.1.3 AlignMixup . . . 5
2.1.4 Local Mixup . . . 5
2.1.5 Mixup for metric learning . . . 6
Metrix . . . 6
Mixup for triplet loss . . . 6
2.2 Out-of-distribution detection . . . 7
2.2.1 Overview . . . 7
OoD, anomaly and novelty detection . . . 7
Detection methods . . . 7
Outlier Exposure . . . 8
2.2.2 Out-of-distribution detection with metric learning . . . 8
2.2.3 Out-of-distribution detection with Mixup . . . 9

3 Problem formulation and Contribution 11
3.1 Formulation . . . 11
3.2 Contribution . . . 12

4 Method 14
4.1 MixOE analysis . . . 14
4.1.1 Auxiliary outlier dataset . . . 14
4.1.2 Mixup variants . . . 14
Mixup with K nearest neighbours . . . 15
Mixup with Outlier Exposure . . . 15
Mixup with labels . . . 16
Mixup with noise . . . 16
4.2 Metric learning for out-of-distribution detection . . . 16
4.2.1 Triplet loss . . . 16
4.2.2 Meaningful choice of the margin . . . 17
4.2.3 Mixup in metric learning for out-of-distribution detection . . . 17
4.3 Evaluation metrics . . . 18
4.3.1 Confidence-based True Negative Rate . . . 18

5 Experiments **19**
5.1 Implementation and experimental details . . . 19
5.1.1 Implementation and training details . . . 19
5.1.2 Datasets . . . 19
5.1.3 Experimental details . . . 20
Mixup variants . . . 20
Changing outlier distribution . . . 22
Metric learning with and without Mixup for OoD detection . . . 23
5.2 Results . . . 24
5.2.1 Impact of the size of auxiliary outlier dataset and Mixup with labels . . . 24
5.2.2 Impact of the diversity of auxiliary outlier dataset . . . 26
5.2.3 Impact of Mixup variants . . . 26
Input Mixup (Mixup with inliers) and Mixup with noise . . . 26
Manifold MixOE and Align MixOE . . . 26
MixOE with KNN . . . 26
MixOE with and without OE loss . . . 26
5.2.4 Impact of metric learning . . . 26
Impact of (I1, I1, I2) triplet . . . 26
Impact of hyperparameter β for the triplet loss . . . 28
Impact of an outlier in the triplet . . . 28
Impact of MixOE with triplet loss . . . 29
Choosing best-performing hyperparameters . . . 29

6 Conclusion and Future Work **36**
6.1 Conclusion . . . 36
6.2 Future Work . . . 37

Bibliography **38**

List of Figures

2.1 Illustration of Mixup. Orange and green colours indicate different classes; blue colour indicates $p(y = 1|x)$, where y is the class, and x is a sample from one of the classes when standard Empirical Risk Minimisation is used (left) and Mixup (right). Image is taken from (Zhang et al., 2018). 4
2.2 Illustration of Manifold Mixup. The left image shows the embedding space without using Manifold Mixup. The image on the right shows a prominent decision boundary near two classes as a result of applying Manifold Mixup. The wider the white boundary is, the less certain the prediction is near it. Image is taken from (Verma et al., 2019). 4

3.1 Illustration of fine- and coarse-grained out-of-distribution examples used for measuring detection performance. Image is taken from (Zhang et al., 2023). 11

5.1 Classification comparison between MixOE (green), baseline (blue) and Mixup with labels (red). Mean values are reported. Standard deviation is reported in the form of error bars. 20
5.2 Detection comparison between MixOE (green), baseline (blue) and Mixup with labels (red) using confidence-based metric. Fine-grained on the right and coarse-grained on the left. 20
5.3 Coarse-grained comparison between MixOE, baseline, Mixup with labels (label_mix) and Mixup with a different number of outliers (mixoe_outl, green) or classes (mixoe_cls, blue) using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. Every experiment for this and further plots is reported as a mean value and standard deviation in the form of error bars. The darker the green (blue) colour is, the more outliers (classes) are used during training. 22
5.4 Fine-grained comparison between MixOE, baseline, Mixup with labels (label_mix) and Mixup with a different number of outliers (mixoe_outl, green) or classes (mixoe_cls, blue) using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the green (blue) colour is, the more outliers (classes) are used during training. 23
5.5 Coarse-grained comparison between MixOE, baseline and four different combinations of triplets according to Table 4.1 with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. Every experiment for this and further plots is reported as a mean value and standard deviation in the form of error bars. 24

5.6 Fine-grained comparison between MixOE, baseline and four different combinations of triplets according to Table 4.1 with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR based on best-performing hyperparameters for coarse-grained detection on the Car, Bird, Butterfly and Aircraft datasets. 25
5.7 Coarse-grained comparison between MixOE, baseline and (I1; I1; I2) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. Every experiment for this and further plots is reported as a mean value and standard deviation in the form of error bars. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 27
5.8 Fine-grained comparison between MixOE, baseline and (I1; I1; I2) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 28
5.9 Coarse-grained comparison between MixOE, baseline and (I1; I1; O) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 30
5.10 Fine-grained comparison between MixOE, baseline and (I1; I1; O) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 31
5.11 Coarse-grained comparison between MixOE, baseline and random selection between (I1,O; I1; O) and (I1; I1,O; O) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 32
5.12 Fine-grained comparison between MixOE, baseline and random selection between (I1,O; I1; O) and (I1; I1,O; O) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 33
5.13 Coarse-grained comparison between MixOE, baseline and random selection between (I1,O; I1; I2) and (I1; I1,O; I2) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 34
5.14 Fine-grained comparison between MixOE, baseline and random selection between (I1,O; I1; I2) and (I1; I1,O; I2) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$ 35

List of Tables

4.1 Table of possible triplet combinations. 17

5.1 Detection comparison of Mixup variants in a coarse-grained setting using TNR@TPR95. 21
5.2 Detection comparison of Mixup variants in a fine-grained setting using TNR@TPR95. 21
5.3 Classification comparison of Mixup variants using accuracy. 21

List of Abbreviations

OoD	**O**ut-**o**f-**D**istribution
ID	**I**n **D**istribution
OE	**O**utlier **E**xposure
ODM	**O**ut-of-**D**istribution **M**ining
TNR@95TPR	**T**rue **N**egative **R**ate **A**t **95**% of **T**rue **P**ositive **R**ate

List of Symbols

m	Margin for metric learning losses
x_i, x_j	A random example from a training distribution
x_{in}	A random in-distribution example
x_{out}	A random out-of-distribution example
d	A distance function
a, p, n	Anchor, a positive example and a negative example for the triplet loss
n_g	Gaussian noise image
$\hat{x}, \hat{a}, \hat{p}$	Mixed examples
L_{ce}	Cross-Entropy loss
L_{oe}	Outlier Exposure loss
L_{trip}	Triplet loss
α	Parameter of the shape of Beta distribution
λ	An interpolation factor
β	A weighting factor for the losses

Dedicated to the Armed Forces of Ukraine

Chapter 1

Introduction

During classification, neural networks provide a class label and a class probability (confidence). When an example belongs to the domain of classes the network can recognise, the confidence should be high for the targeted class. When such an input example is not from the training distribution of examples, that is, an out-of-distribution example, its confidence should be low for every class (Hendrycks and Gimpel, 2016).

Out-of-distribution (OoD, outlier) detection is a vast and ongoing research domain that is relevant across multiple disciplines, including medical diagnosis (Li, Desrosiers, and Liu, 2022), segmentation (Cen et al., 2021), and network security (Aliakbarisani, Ghasemi, and Wu, 2019), to name a few. Out-of-distribution detection encompasses the detection of OoD examples and the classification of ID examples. This thesis focuses on neural network performance when given out-of-distribution images during inference. The goal is to prevent overconfident predictions of OoD examples. We analyse the impact of Mixup, a data augmentation technique, during training as it is used in the MixOE approach (Zhang et al., 2023). We also investigate the use of metric learning for out-of-distribution detection.[1]

Since the performance of a neural network in its majority depends on the data it is trained on, it is known to output an overconfident class when making predictions on examples utterly different from the ones it is trained on (Hein, Andriushchenko, and Bitterwolf, 2019). Thus, such behaviour results in performance degradation. By making models more robust to outliers, confidence-based or other detection methods more effectively discard the examples that cannot be classified within the domain of available classes instead of falsely predicting the wrong class with large confidence.

Mixup (Zhang et al., 2018) is known to boost the regularisation of neural networks (Carratino et al., 2020). It is applied to increase performance while being implemented with a few lines of code. Mixup is also used in metric learning and out-of-distribution detection domains (Venkataramanan et al., 2022b; Zhang et al., 2023), which benefit from one of the advantages of Mixup to decrease the confidence around the decision boundary. We first analyse how Mixup is used between inliers and outliers in MixOE (Zhang et al., 2023), whose model and dataset setup we use in our experiments.[2] We explore how the size and diversity of an auxiliary outlier dataset affect training. We also apply other Mixup variants to understand which components of Mixup help to improve the performance.

The hypothesis proposed in previous work (Koner et al., 2021; Masana et al., 2018; Venkataramanan et al., 2022b) states that a success of an out-of-distribution detection and classification generally lies in the network's ability to learn an embedding space in a specific way. Embeddings of in-distribution (ID, inlier) examples should be

[1] The code for this thesis can be found via Github: https://github.com/Oleksandra2020/metric_mix_oe.
[2] The code for MixOE can be found via GitHub: https://github.com/zjysteven/MixOE.

compact within the cluster of embeddings from the same class. At the same time, embeddings from different classes should be far from each other. Ruled by this statement, we analyse the literature on **metric learning** applied to out-of-distribution detection. Most contingent work to ours use contrastive learning between in-distribution samples for detecting outlier examples. However, to our knowledge, metric learning with OoD examples for out-of-distribution detection is an underexplored area of research. Out-of-Distribution Mining (ODM) (Masana et al., 2018) uses contrastive loss between ID and OoD examples, but the authors do not tune the hyperparameters for contrastive loss and do not investigate the use of triplet loss. The authors of (Yang et al., 2020) use Triplet Network, but do not use an auxiliary outlier dataset. In our work, we provide possible combinations of triplets (with and without outliers) for triplet loss for the problem of OoD detection and analyse meaningful values of the margin. Inspired by the work on Mixup in metric learning (Venkataramanan et al., 2022b; Lee and Kim, 2022), we use Mixup to create anchors and positive examples resulting in new triplet combinations. We aim to show an approach and results that will hopefully provide more understanding about the topics of Mixup and metric learning in out-of-distribution detection.

Chapter 2 reviews related work on Mixup and out-of-distribution detection, including the impact of Mixup and metric learning for OoD detection. Chapter 3 formulates the problem this thesis considers. Chapter 4 suggests the methodology for out-of-distribution detection, specifically Mixup and metric learning. Chapter 5 presents executed experiments and current results, and Chapter 6 concludes the thesis.

Chapter 2

Related work

2.1 Mixup

Throughout the years, there has been massive research done on different variants of Mixup, including Manifold Mixup (Verma et al., 2019), CutMix (Yun et al., 2019), Local Mixup (Baena, Drumetz, and Gripon, 2022), AlignMixup (Venkataramanan et al., 2022b), and many more. Mixup is observed to smooth the confidence around the decision boundary (Zhang et al., 2018) and is applied to input samples (Zhang et al., 2018) as well as to embeddings (Verma et al., 2019). When applied to the latter, (Venkataramanan et al., 2022a) observed that such models need longer training to show better performance.

2.1.1 Input Mixup and manifold intrusion

Mixup (Zhang et al., 2018) is a data augmentation technique used to create a convex combination of two samples and their labels from a training batch resulting in a new example with a new label. It can be implemented in a few lines of code and is usually used between random pairs of samples inside a single batch:

$$\hat{x} = \lambda x_i + (1 - \lambda)x_j, \hat{y} = \lambda y_i + (1 - \lambda)y_j, \tag{2.1}$$

where x_i, x_j are training examples, y_i, y_j are their one-hot encoded labels respectively, and $\lambda \in [0, 1]$, which is a mixing factor, usually drawn from a Beta distribution $\text{Beta}(\alpha, \alpha)$ with a set hyperparameter α.

Thus, the authors suggest using Vicinal Risk Minimization (VRM) principle, which defines a vicinity distribution v in the area around the given training examples (x_i, y_i), creating a virtual pair $(\hat{x}, \hat{y})$ (Chapelle et al., 2000):

$$P_v(\hat{x}, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} v(\hat{x}, \hat{y} | x_i, y_i) \tag{2.2}$$

Sampling from a new dataset $D = \{(x_i, y_i)\}_{i=1}^{m}$, we can minimize empirical vicinal risk:

$$R_v = \frac{1}{m} \sum_{i=1}^{m} l(f(\hat{x}_i), \hat{y}_i), \tag{2.3}$$

where l is the loss function, a standard cross-entropy, in our case. The formulations defined above are taken from (Zhang et al., 2018).

Figure 2.1 shows how Mixup can create samples between the classes and help reduce the certainty near the decision boundary. This is an essential advantage of Mixup that some out-of-distribution work benefit from.

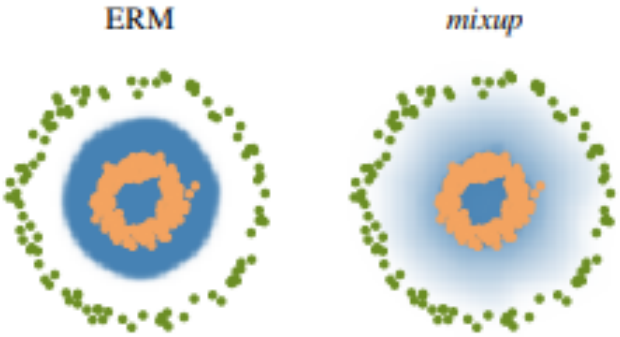

FIGURE 2.1: Illustration of Mixup. Orange and green colours indicate different classes; blue colour indicates $p(y = 1|x)$, where y is the class, and x is a sample from one of the classes when standard Empirical Risk Minimisation is used (left) and Mixup (right). Image is taken from (Zhang et al., 2018).

(Guo, Mao, and Zhang, 2019) discovered an inherent problem that Mixup produces, called manifold intrusion. Manifold intrusion happens when a mixed example resembles an existing example in the dataset but is assigned a label that differs from the true one. According to the authors, such a condition produces under-fitting and decreases the model's performance.

2.1.2 Manifold Mixup

Manifold Mixup (Verma et al., 2019) interpolates hidden representation instead of input examples. The authors state that this provides uncertainty around a decision boundary on multiple representation layers:

$$\hat{g}_k = \lambda g_k(x_i) + (1 - \lambda) g_k(x_j), \tag{2.4}$$

where mixing factor λ and label $\hat{y}$ are defined as in equation 2.1 and g_k is a layer k in a neural network before which Manifold Mixup is applied (Verma et al., 2019).

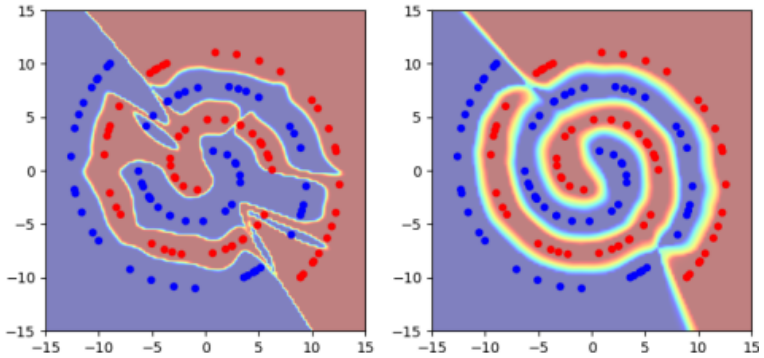

FIGURE 2.2: Illustration of Manifold Mixup. The left image shows the embedding space without using Manifold Mixup. The image on the right shows a prominent decision boundary near two classes as a result of applying Manifold Mixup. The wider the white boundary is, the less certain the prediction is near it. Image is taken from (Verma et al., 2019).

Figure 2.2 illustrates how Manifold Mixup affects the decision boundary in the embedding space.

It is shown that Manifold Mixup learns class representations more compactly, that is, with fewer directions of variance, which improve the network's ability to perform on slightly perturbed adversarial or novel examples (Verma et al., 2019). This is because Manifold Mixup flattens representations per class by reducing the number of principal components (Verma et al., 2019).

Interestingly, in the implementation provided along with the paper, Manifold Mixup is never executed after the same layer of a neural network during each iteration. The layer is randomly chosen between the first few layers per iteration.

2.1.3 AlignMixup

AlignMixup (Venkataramanan et al., 2022a) combines two images in a more meaningful way than just interpolation. The authors take two images and align them such that the pose of one image and the texture of the other are retained in a resulting mixed image. The alignment is done between feature vectors and never on the input samples.

The features of training samples x_i and x_j are obtained using an encoder network F. These feature vectors are flattened to form matrices A_i and A_j, respectively, of size $r \times r$. The cost matrix is calculated to contain pairwise distances between the columns of the flattened matrices.

Then, an optimal transport plan P^* containing joint probabilities is derived using Sinkhorn distance (Cuturi, 2013). Using P^*, a stochastic matrix $R = rP^*$ is obtained and used for aligning feature matrices for the final mix:

$$\hat{A}_1 = A_2 R^T, \hat{A}_2 = A_1 R \tag{2.5}$$

The aligned feature matrices $\hat{A}_1$ and $\hat{A}_2$ are transformed back to their original vector shape, and Mixup is performed between $(A_1, \hat{A}_1)$ and $(A_2, \hat{A}_2)$.

The advantages of such an approach are that the feature vector resolution is low and the features closer to the classifier are small (Venkataramanan et al., 2022a). The authors also state that their approach has a similar computational overhead to existing Mixup variants when trained for the same number of epochs. They find that AlignMixup, and Manifold Mixup, in general, benefit from longer training compared to the input Mixup variants.

2.1.4 Local Mixup

The authors of Local Mixup (Baena, Drumetz, and Gripon, 2022) create weights based on the distances between mixed examples when computing loss. They also mention that they experiment with Mixup for K nearest neighbours (KNN), but it does not give the best performance for their setup, which is standard classification. In our experiments, we use Mixup with K nearest neighbours, but between different distributions.

2.1.5 Mixup for metric learning

Metrix

Mixup has also been investigated for metric learning in Metrix (Venkataramanan et al., 2022b). Since metric learning operates on a few examples at a time, just like Mixup, authors propose contrastive loss for mixed examples.

In Metrix (Venkataramanan et al., 2022b), input, feature and embedding Mixup variants are randomly picked during each iteration to mix anchor-negative or positive-negative examples. For input Mixup, three hardest negative examples are used, and for other types of Mixup, all pairs are used. Here, Manifold Mixup is applied randomly on one of the last layers per iteration. Input Mixup and Manifold Mixup are defined in Equation 2.1 and Equation 2.4, respectively. Feature Mixup for training samples (x_i, x_j) is defined as:

$$\hat{x} = f_k(\lambda g_m(x_i) + (1 - \lambda) g_m(x_j)), \tag{2.6}$$

where g_m and f_k are mappings from the input layer to an intermediate layer and from an intermediate layer to an embedding, respectively (Venkataramanan et al., 2022b).

The authors only use contrastive loss along with its variants and do not use Mixup for triplet loss. In the loss, there is either a mix between an anchor and a negative example or between a positive and a negative example. In this case, positives are examples from the same class, and negatives are examples from a different class closest to the anchors. During training, the authors combine contrastive loss with clean examples and contrastive loss with mixed examples, the latter weighted with a factor w.

The authors state that the advantage of such an approach is the inclusion of samples during training not present in the training dataset. Thus, it has a higher chance of learning representations similar to the ones from the testing classes (Venkataramanan et al., 2022b). The authors discover that the best setting is the random combination of all Mixup variants, but feature Mixup works best out of all of them regarding their individual performances. The downside is that mixing features or embeddings takes longer than an original input Mixup.

Mixup for triplet loss

(Lee and Kim, 2022) use Mixup for triplet loss for the task of human activity recognition. They create mixed positive and negative examples by mixing a positive example with an anchor and a negative example with an anchor, respectively. They also define an additional hyperparameter for the probability of mixing these pairs of examples:

$$\hat{p} = \mathbb{1}_{p_{mixup}} \lambda a + (1 - \mathbb{1}_{p_{mixup}} \lambda) p, \hat{n} = \mathbb{1}_{p_{mixup}} \lambda a + (1 - \mathbb{1}_{p_{mixup}} \lambda) n, \tag{2.7}$$

where $(a; p; n)$ are anchors, positive and negative examples, respectively. $\hat{p}$ and $\hat{n}$ are mixed examples. $\mathbb{1}_{p_{mixup}}$ is the indicator function that applies Mixup with probability p_{mixup}. The formulas are adapted from the formulations in the original paper. In our approach, we either mix anchors with negative examples or positive with negative examples.

2.2 Out-of-distribution detection

2.2.1 Overview

OoD, anomaly and novelty detection

In the literature, the terms out-of-distribution (OoD) detection, anomaly detection, and novelty detection have been used to define different but related problems. The terminology described here has been prevalent across papers (Zhang et al., 2023; Bodesheim et al., 2015; Jézéquel et al., 2022; Masana et al., 2018; Winkens et al., 2020; Sun et al., 2022).

Anomaly and novelty detection deal only with the classification of the samples as being from the same distribution as the training data or not, while out-of-distribution detection also deals with the classification between in-distribution classes. We can describe a novelty or anomaly detector $G(x_i)$:

$$G(x_i) = \begin{cases} 0 & \text{if } x_i \text{ is OoD} \\ 1 & \text{if } x_i \text{ is ID} \end{cases} \tag{2.8}$$

depending on whether sample x_i is an inlier or outlier. On the other hand, OoD classifier $C(x_i)$ first identifies the class of the sample. Certain scoring- or distance-based detection methods are used to identify an outlier during inference.

While the definitions of these three detection areas are sometimes used interchangeably in publications, out-of-distribution detection is considered the most general term, involving novelty and anomaly detection. Anomalies are the farthest examples from the in-distribution ones; there is no resemblance between ID examples and anomalies. For instance, if a classifier is trained to distinguish different classes of bugs, the data with bugs are inside the distribution. Say, the data that comes for testing are images of tigers. Such data are from a different distribution and are regarded as an anomaly. Novelty detection is the hardest since such images may have similar features to the ones from the training distribution. However, a slight difference in the image makes it an out-of-distribution sample. Since the boundary between these examples is so thin, some approaches, after detecting a novelty, incorporate it into training as one of the classes or as a separate one (Bodesheim et al., 2015). In our work, detected novelty is not included in the training but is considered out-of-distribution. An example of a novelty is when the training distribution consists of different kinds of butterflies but, during testing, is presented with a dragonfly, which is also a bug and may be similar to a butterfly but is not one and, thus, must be regarded as an out-of-distribution example.

The difference between anomalies and novelties is also accentuated in (Zhang et al., 2023), where the authors divide OoD detection into fine- and coarse-grained.

Detection methods

There are typically score-based (Hendrycks and Gimpel, 2016; Liang, Li, and Srikant, 2017; Hsu et al., 2020; Techapanurak, Suganuma, and Okatani, 2019), distance-based (Sun et al., 2022; Koner et al., 2021) and distribution-based (Winkens et al., 2020; Lee et al., 2017) detection and training techniques.

In (Hendrycks and Gimpel, 2016), the authors propose to derive the statistics from softmax scores, namely the maximum softmax probability. These statistics are taken from the predicted class on a test set, and then an outlier is identified. The authors of (Liang, Li, and Srikant, 2017) use temperature scaling to compute softmax and input processing that adds small perturbations to the input image. The

detector preprocesses an image, calculates its temperature-scaled softmax score and determines if it is an outlier or not using a defined threshold. The work of (Hsu et al., 2020) is based on the previously discussed one, but the authors change the input preprocessing and propose the approach of decomposed confidence. (Techapanurak, Suganuma, and Okatani, 2019) propose changing the layer to use softmax of scaled cosine similarity while still minimizing cross-entropy loss for the multi-class ID classification task.

The authors of (Sun et al., 2022) are the first to use K nearest neighbours to estimate whether an example is an outlier during inference. They do not require any OoD samples during training or the change in activation functions. Instead, first, they train a multi-class classification model on in-distribution data. Then, they compute the distance between the embedding of a test example and the embeddings of every training sample. They define a threshold, which satisfies a 95% True Positive Rate for inliers, and, based on this threshold and the distance value to K-th nearest neighbour, they determine if the test sample is OoD or not. In (Koner et al., 2021), the authors use a transformer for training ID samples and compute the mean embedding for each class. They use these values to calculate the distance to the test embedding. Then they use a threshold for distance and confidence. The outlier is identified if either of these thresholds is violated.

(Winkens et al., 2020) use contrastive learning for ID examples. During inference, they fit a Gaussian distribution to the activations of the training data. It is used to estimate an OoD score for each class using the highest class-conditional density (Winkens et al., 2020). During training, (Lee et al., 2017) add a loss term to minimize the Kullback-Leibler divergence to shift the distribution of OoD examples to the Uniform one, ensuring less confident predictions for such examples.

Outlier Exposure

Outlier Exposure, or the inclusion of outliers during training (Hendrycks, Mazeika, and Dietterich, 2019), while effective on its own, also enhances performance in tandem with other methods (Zhang et al., 2023; Papadopoulos et al., 2021). Such a training method requires the use of an auxiliary outlier dataset whose distribution is different from an OoD testing set. The loss term is then:

$$L_{ce}(f(x_{in}), y_{in}) + \beta L_{oe}(f(x_{out}), f(x_{in})), \tag{2.9}$$

x_{in} are the ID examples, x_{out} are the OoD examples. L_{ce} is a standard cross-entropy loss, and L_{oe} is the cross-entropy from $f(x_{out})$ to the Uniform distribution, defined as in (Hendrycks, Mazeika, and Dietterich, 2019).

2.2.2 Out-of-distribution detection with metric learning

The fundamental idea metric learning operates on is that embeddings from a penultimate layer of a neural network and higher bear more information about the representation of the input sample (Mao et al., 2019). It aims to learn a distance metric such that different classes are far from each other and same-class samples are close.

Specifically, the concepts of anchors and positive and negative examples are introduced. An anchor is a sample regarding which the distance to other samples is calculated. A positive example is usually a randomly chosen example from the same class as an anchor, and a negative example is an example from a different class. A hard negative example is the closest example to the anchor from a class different from the anchor's class. Contrastive loss (Hadsell, Chopra, and LeCun, 2006) and triplet

loss (Wang et al., 2014) are classic examples of metric losses used during training. In this thesis, we use the latter.

Extensive research has been done on contrastive learning for out-of-distribution detection (Winkens et al., 2020; Koner et al., 2021; Tack et al., 2020; Cho, Seol, and Lee, 2021). Contrastive learning takes in-distribution samples and their augmented versions and, using contrastive loss or its variant, learns how to keep these samples close in the embedding space. Regarding metric learning in out-of-distribution detection, the most similar work to ours is (Masana et al., 2018; Yang et al., 2020). In this case, embeddings are learned so that samples from different classes are pushed apart and vice versa. Authors of ODM (Masana et al., 2018) use an auxiliary outlier dataset during training, from which they sample negative examples for a modified contrastive loss that only includes pairs that have either both in-distribution samples or one of the samples is out-of-distribution:

$$l(x_i, x_j, y; W) = \frac{1}{2}(1-y)zD_W^2 + \frac{1}{2}yz(max(0, m - D_W))^2, \tag{2.10}$$

where x_i, x_j are two examples from the training set, $y \in \{0, 1\}$ indicates if the samples are from the same class or not, and $z \in \{0, 1\}$ is 0 when both samples are OoD and 1 otherwise, m is the margin and D_W is the distance between the embeddings of x_i and x_j. The authors provide anomaly and novelty detection results, where novelties are classes from an in-distribution dataset excluded from the training, and anomalies are samples from a different dataset. They discover that their approach works as well as other state-of-the-art methods, improving performance on novelty detection but harming classification accuracy. During training, they use a quarter of all pairs consisting of one ID and one OoD sample every two batches. Their code suggests that they test their approach based on a distance-based function to detect outliers. In (Yang et al., 2020), the authors use Siamese and Triplet Networks, only in-distribution data during training, and a confidence-based score function to detect outliers. Both papers use a True Negative Rate at 95% True Positive Rate (TNR@95TPR), which we also use to evaluate and compare our experiments.

(Mao et al., 2019) use metric learning to solve a similar problem to out-of-distribution detection, which is adversarial robustness. They generate an adversarial example and use triplet loss with cosine distance to bring the example closer to its original class. They also include a loss term with the L2 norms of anchors, positive and negative samples.

2.2.3 Out-of-distribution detection with Mixup

Models trained with Mixup show lower confidence and better calibration when tested on out-of-distribution samples (Thulasidasan et al., 2019). (Ravikumar et al., 2020; Chun et al., 2020) apply Mixup only for ID examples, (Ravikumar et al., 2020) also considers Mixup between OoD examples and Gaussian noise; finally, (Zhang et al., 2023) uses Mixup between ID and OoD examples.

In this thesis, we focus on the MixOE approach (Zhang et al., 2023) since, so far, it has shown the most promising results in OoD detection. This approach uses an auxiliary outlier dataset during training that is never used during inference in combination with input Mixup between inliers and outliers. That is, in equation 2.1, instead of x_j, a random outlier x_{out} is used, and instead of y_j the label is drawn from a uniform probability distribution U. This way, the label ensures the same confidence for all outliers:

$$\hat{x} = \lambda x_{in} + (1 - \lambda) x_{out}, \hat{y} = \lambda y_{in} + (1 - \lambda) U \tag{2.11}$$

The method of MixOE uses outliers during training. These outliers are taken from WebVision 1.0 (Li et al., 2017) and are not used during testing. The loss term consists of cross-entropy on ID examples and cross-entropy on mixed examples with a weighting factor β:

$$L_{ce}(f(x_{in}), y_{in}) + \beta L_{ce}(f(\hat{x}), \hat{y}), \tag{2.12}$$

where (x_{in}, y_{in}) is a training ID example, $(\hat{x}, \hat{y})$ is a mixed example, f is a mapping of input examples to embeddings, and L_{ce} is a standard cross-entropy loss. The formulas are taken from the original paper (Zhang et al., 2023). It is important to note that the authors do not mix inliers with each other and do not use outliers without mixing during training. The authors state that applying both Mixup between ID and OoD examples and Outlier Exposure (Hendrycks, Mazeika, and Dietterich, 2019) results in manifold intrusion (Guo, Mao, and Zhang, 2019).

In MixOE, both fine- and coarse-grained OoD detection metrics are improved. They correspond to whether the examples are from the same domain as inliers or not. However, the detection performance depends largely on the datasets and their splits, based on the results from (Zhang et al., 2023).

Chapter 3

Problem formulation and Contribution

3.1 Formulation

In this thesis, we tackle the problem of out-of-distribution detection for neural networks. We follow MixOE (Zhang et al., 2023) OoD detection setup, including datasets, a model and metrics.

Out-of-distribution detection is a problem that is relevant in classification and regression, various tasks like segmentation, medical diagnosis, etc. The goal is to make a robust model that generalises well to the training distribution by outputting the correct class with high class probability but which also provides small confidence for examples from a different distribution. The problem of out-of-distribution detection is especially difficult when it deals with examples very close to the in-distribution ones (fine-grained examples) (Tack et al., 2020; Zhang et al., 2023).

OoD detection encompasses both the **classification** of ID examples and the **detection** of OoD examples (Zhang et al., 2023). Classification is measured by accuracy on in-distribution data only. Detection deals with recognising an example from a different distribution than the network is trained on. In this work, detection performance is estimated via confidence that the neural network outputs along with the class label.

Detection is divided between **fine-grained** and **coarse-grained** OoD testing **settings** (Zhang et al., 2023). The fine-grained dataset consists of classes from the same dataset that the model is trained on but which were excluded from the training set. Coarse-grained data contain other datasets that the model is not trained on. The outlier data used during training comes from a different distribution, an auxiliary outlier dataset, which is WebVision 1.0 (Li et al., 2017) in this case (Zhang et al., 2023).

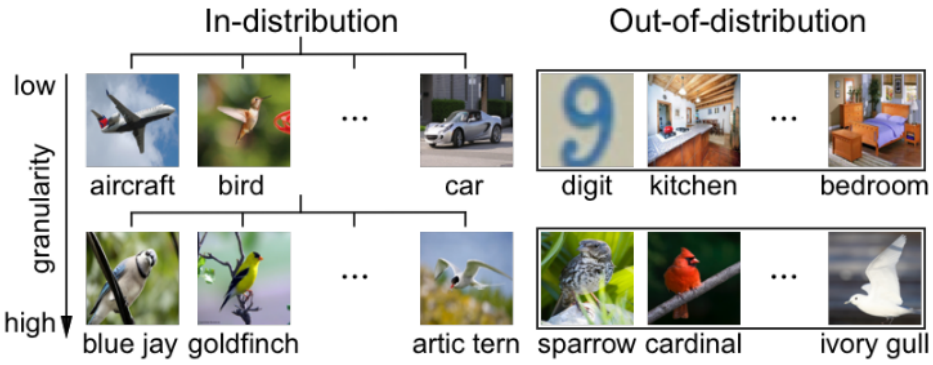

FIGURE 3.1: Illustration of fine- and coarse-grained out-of-distribution examples used for measuring detection performance. Image is taken from (Zhang et al., 2023).

3.2 Contribution

There are numerous approaches to out-of-distribution detection, but we focus on the use of two methods, Mixup and metric learning. We decompose Mixup between inliers and outliers as proposed in MixOE (Zhang et al., 2023) by changing the size and diversity of the auxiliary outlier dataset and adding other Mixup variants. We use metric learning, specifically triplet loss and investigate its impact on the problem of OoD detection using different combinations of triplets, some of which are created using Mixup. Finally, we combine MixOE and metric learning to investigate if together they provide an improvement to the problem of OoD detection as they do separately.

The contributions of this work are:

1. Mixup for OoD detection, based on MixOE (Zhang et al., 2023).
   (a) Modeling auxiliary outlier dataset for Mixup between ID and OoD examples. Changing the size and diversity of the dataset. Findings:
      - The large size of the auxiliary outlier dataset is more important for the coarse-grained detection settings than for the fine-grained settings.
      - Diversity of the auxiliary outlier dataset. When as few as ten classes are used, the same performance is achieved as when all 1000 classes are used for the coarse-grained settings.
   (b) Comparison and implementation of Mixup variants applied between ID and OoD examples to investigate the most impactful components of Mixup. Findings:
      - Mixing labels alone without mixing ID and OoD training samples is almost as effective as a full Mixup for the fine-grained settings.
      - Most Mixup variants between inliers and outliers we used are as effective as the input Mixup.
2. Metric learning for OoD detection.
   (a) Development and implementation of triplet loss with different triplet combinations. Findings:
      - Triplet combination with only inliers performs better than or similar to the triplets with outliers for both settings.
      - For the coarse-grained settings, a triplet combination with only inliers performs significantly better than the baseline. In contrast, triplet combinations with outliers as negative examples have marginally the same or significantly lower performance than the baseline.
      - Combination of MixOE and metric learning having inconclusive results. It either has marginal improvement over MixOE or significantly improves the detection performance for the coarse-grained setting. For the fine-grained settings, triplet combinations involving outliers perform similarly to the baseline or worse.
   (b) Analysis of the triplet loss margin and its effect on coarse- and fine-grained detection performance. Findings:
      - A trade-off between fine- and coarse-grained detection depending on the size of the margin for triplet combination involving only inliers.
3. Mixup with metric learning for OoD detection

(a) Creating combinations of triplets with Mixup between an anchor and negative example or between positive and negative examples. Findings:
   - When combined with MixOE, the coarse-grained detection performance improvement over MixOE compromises classification accuracy for all datasets.
   - Marginal or no improvement over the baseline for the fine-grained detection.

Chapter 4

Method

4.1 MixOE analysis

As was mentioned before, we first analyse MixOE (Zhang et al., 2023) and follow their model and datasets setup for further experiments. We propose ways to disassemble its components to see which ones are contributing the most to the performance boost.

MixOE uses outliers in the form of the auxiliary outlier dataset discussed in Section 2.2.3. It enables us to mix outliers and inliers during training (Equation 2.11).

We focus on different Mixup variants (the layer where Mixup is applied, which samples are mixed (e.g., mixing neighbouring samples), the loss term and the label of an outlier) and the configuration of the auxiliary outlier dataset.

In Chapter 5, we provide experiments on how the number of outliers and outlier classes during training affects the results (Figures 5.4 and 5.3). The results for the Mixup variants are in Tables 5.1-5.3 and Figures 5.1 and 5.2.

4.1.1 Auxiliary outlier dataset

Auxiliary outlier dataset is used only during training and is not used in a testing stage, according to the original MixOE work. By removing examples from the auxiliary outlier dataset, we also control the outlier distribution that the model is exposed to. Our goal is to determine if both coarse- and fine-grained detection are affected by the number of outliers that the model is presented with or if other components are more significant.

Specifically, we target:

- the size of the outlier dataset (random choice of N outliers from the auxiliary outlier dataset, WebVision)
- the diversity of the outlier dataset (random choice of K outlier classes from the auxiliary outlier dataset, WebVision)

4.1.2 Mixup variants

We use these Mixup variants between ID and OoD samples when applicable:

1. Input Mixup on inliers as in (Zhang et al., 2018). Referred in Chapter 5 as **Mixup with inliers**.
2. **Manifold MixOE**, based on Manifold Mixup (Verma et al., 2019) and MixOE (Zhang et al., 2023).
3. **Align MixOE**, based on Align Mixup (Venkataramanan et al., 2022a) and MixOE (Zhang et al., 2023).

4. **MixOE with Outlier Exposure** on minimum confidence outliers. We add Outlier Exposure (Hendrycks, Mazeika, and Dietterich, 2019) loss term in our training and include only minimum confidence outliers for mixing.
5. MixOE with K nearest neighbours, **MixOE with KNN**, inspired by (Baena, Drumetz, and Gripon, 2022).
6. **Mixup with labels**.
7. **Mixup with noise**.

Mixup with K nearest neighbours

Inspired by Local Mixup (Baena, Drumetz, and Gripon, 2022), who experiment with Mixup with KNN for classification, we use a different setting and mix between samples from different distributions. For every sample in the ID training batch, we draw K nearest neighbours from the auxiliary outlier dataset. In our experiments, we use K=10 and draw one random example from these K examples for each inlier per iteration before performing Mixup. We create pairs of examples that are near each other in the embedding space but come from different distributions. Since these examples are from different distributions and are close to each other in the embedding space, it means that they are close to the decision boundary. This is why we investigate how the KNN approach works in this situation since Mixup induces less confident predictions near the decision boundary (Zhang et al., 2018).

We do this by first extracting the embeddings of all samples in WebVision 1.0 (Li et al., 2017). We normalise the embeddings, and then we use Euclidean distance to find the nearest neighbours for every inlier. We save these examples and later use them for input Mixup during training.

Mixup with Outlier Exposure

Mixup between inliers and outliers in combination with Outlier Exposure in the form of OE loss does not show promising results in (Zhang et al., 2023), that is, adding outliers separately in the loss, and not only for mixing. However, we investigate a situation when we include outliers, but only those that have the smallest confidence already. (Chen et al., 2021), an OoD detection work for mining the outliers, uses Outlier Exposure to choose between perturbations of the same outlier example to maximise cross-entropy. Inspired by their approach, our hypothesis is that outlier examples with minimum confidence result in high entropy and can help the model learn the outlier distribution better. Following Equations 2.9 and 2.12, the loss term is then:

$$L_{ce}(f(x_{in}), y_{in}) + \beta L_{oe}(f(x_{out}), f(x_{in})) + \beta_1 L_{ce}(f(\hat{x}), \hat{y}), \tag{4.1}$$

x_{in} are the ID examples, x_{out} are five outlier examples with the lowest confidence from the current batch, $\hat{x}, \hat{y}$ are mixed examples and labels, respectively. L_{ce} is a standard cross-entropy loss and L_{oe} is defined as in Equation 2.9. Every ID example from the batch is randomly mixed with one of these five outliers.

In Chapter 5, we show two experiments:

- Mixup with minimum confidence outliers (referred to as MixOE (min)).
- Mixup with minimum confidence outliers with Outlier Exposure (referred to as MixOE (min) with OE loss, which is a second term in Equation 4.1).

Mixup with labels

The outlier sample always has a label drawn from a uniform distribution, so we investigate how mixing only labels between ID and OoD examples affects the training, namely:

$$\hat{x} = x_{in}, \hat{y} = \lambda y_{in} + (1 - \lambda)U, \tag{4.2}$$

where (x_i, y_i) is a training example. It was shown that Mixup improves regularisation not only due to the mixing of examples but also one-hot encoded labels (Zhang et al., 2018; Carratino et al., 2020) and calibrates the confidence (Thulasidasan et al., 2019). Thus, by mixing one hot encoded label with a label that comes from a uniform distribution (uniform label), we induce less confidence in the labels of the training ID examples, leaving the example itself untouched.

Mixup with noise

Since Gaussian noise is often used for implementing adversarial attacks (Liu et al., 2021) and is also used to mix with the outlier examples during training for OoD detection (Ravikumar et al., 2020), we experiment with Gaussian noise by mixing an input image with a Gaussian noise image which has a uniform label. That is,

$$\hat{x} = \lambda x_{in} + (1 - \lambda)n_g, \hat{y} = \lambda y_{in} + (1 - \lambda)U, \tag{4.3}$$

where every pixel of image n_g is a Gaussian noise image.

4.2 Metric learning for out-of-distribution detection

As was discussed earlier, the most common way to apply metric learning for OoD detection is to use contrastive learning between ID examples and their augmented or transformed versions. The work that explicitly targets metric learning using ID and OoD classes for the loss do not tune the margin or use hard negatives. This is why we propose to explore the use of triplet loss with different triplet setups and a possible way to choose the margin.

4.2.1 Triplet loss

Unlike contrastive loss (Hadsell, Chopra, and LeCun, 2006), which separately draws the same-class examples closer, and examples from different classes further apart, triplet loss (Wang et al., 2014) takes into account the distance between an anchor and a positive example and between the anchor and a negative example:

$$L_{trip}(a, p, n; d, m) = max\{d(a_i, p_i) - d(a_i, n_i) + m, 0\}, \tag{4.4}$$

where a, p, n are anchor, positive, and negative examples, respectively, d is a distance function, typically Euclidean or cosine distance are used, and m is a margin. In our case, we use cosine distance. Anchors are every inlier example from ID batch. If the distance between the anchor and the negative example is smaller than the distance between the anchor and the positive example, then there is a penalty applied to ensure that negative examples are further from the anchor than positive ones.

The final loss includes standard cross-entropy and triplet loss:

$$L_{ce}(f(x_{in}), y_{in}) + \beta L_{trip}(a, p, n), \tag{4.5}$$

where (x_{in}, y_{in}) are ID training examples, L_{ce} is a standard cross-entropy, f is a mapping used for extracting a predicted class, $(a; p; n)$ are anchors, positive and negative examples, respectively, and L_{trip} is a triplet loss defined in Equation 4.4. $(a; p; n)$ are embeddings extracted from the penultimate layer of the neural network.

anchor	positive	negative
I1	I1	I2
I1	I1	O
I1	I1,O	O
I1,O	I1	O
I1	I1,O	I2
I1,O	I1	I2

TABLE 4.1: Table of possible triplet combinations.

In the out-of-distribution setup with an auxiliary outlier dataset, there are different ways to define triplets, which are written in Table 4.1. $I1$ is a random example drawn from an in-distribution class 1, $I2$ is a random example drawn from an in-distribution class 2, O is the closest example from the auxiliary outlier dataset to the anchor, and $(I1, O)$ is a mixed example, derived using Equation 2.1.

In this work, we investigate these triplets: $(I1; I1; I2)$, $(I1; I1; O)$ along with triplets derived with Mixup. Following the work of (Yang et al., 2020), we use the triplet $(I1; I1; I2)$. Inspired by ODM (Masana et al., 2018), who use outliers in contrastive loss, we create $(I1; I1; O)$ triplet combination. Inspired by Metrix (Venkataramanan et al., 2022b), we target Mixup for (a, n) and (p, n) by randomly selecting between $(I1; I1, O; I2)$ and $(I1, O; I1, I2)$ at each training iteration. We do analogously for $(I1; I1, O; O)$ and $(I1, O; I1; O)$. Initial experiments show that random selection works better than always mixing the same triplet type, so we show the results in Chapter 5 only for random selection. When we further write $(I1; I1, O; I2)$ or $(I1; I1, O; O)$, **we refer to the random choice between mixed anchor or positive examples.**

We also combine metric learning with MixOE. In this case, the loss is:

$$L_{ce}(f(x_{in}), y_{in}) + \beta L_{trip}(a, p, n) + \beta_1 L_{ce}(f(\hat{x}), \hat{y}), \tag{4.6}$$

where (x, y) is a training example, $(\hat{x}, \hat{y})$ is a mixed example, f is a mapping of input examples to embeddings, and L_{ce} is a standard cross-entropy loss.

4.2.2 Meaningful choice of the margin

In this work, embeddings for the triplet loss are in the range of $[0, 1]$ since they are extracted from the layer just before the fully connected layer, which is ReLU. The embeddings are then normalised. Thus, to penalize those negative examples that are closer to the anchor than the positive examples with a margin m, the margin m is chosen in the range of $[0, 1]$.

If we choose $m > 1$, then the loss will always penalize examples, even where $d(a, p_i) < d(a, n_i)$, and the margin becomes meaningless, which we do not aim to do. In our work, we experiment with a few different margins and see if there is a pattern in performance depending on the coarse- or fine-grained detection.

4.2.3 Mixup in metric learning for out-of-distribution detection

Mixup in metric learning has already shown prominent results for deep metric learning benchmarks (Venkataramanan et al., 2022b). As was mentioned earlier,

Mixup produces virtual examples that can help explore the vicinal distribution. Thus, it is important to investigate the impact of this combination on the OoD detection setup since, during inference, the model is presented with examples that may lie in the vicinity of training distribution.

We experiment with different mixing options for our triplet loss setup. In Metrix (Venkataramanan et al., 2022b), contrastive loss and its variants are used, and either anchors or positives are mixed with negatives. Inspired by their work, we pass both mixed examples and the clean ones in the loss. The mixed anchors and positive examples can be written as:

$$\hat{a} = \lambda a + (1 - \lambda)n, \hat{p} = \lambda p + (1 - \lambda)n \tag{4.7}$$

We choose to mix embeddings just before feeding them to the triplet loss. We create triplets by sampling the hardest negatives to the anchor and all positive examples from the inlier batch with the same class as the anchor.

Thus, for the pair $(I1, O; I1; O)$ or $(I1; I1, O; I2)$, the loss will be:

$$L_{ce}(f(x_{in}), y_{in}) + \beta(L_{trip}(a, p, n) + L_{trip}(\hat{a}, p, n)) + \beta_1 L_{ce}(f(\hat{x}), \hat{y}) \tag{4.8}$$

For the pair $(I1, O; I1; O)$ or $(I1; I1, O; I2)$, the loss will be:

$$L_{ce}(f(x_{in}), y_{in}) + \beta(L_{trip}(a, p, n) + L_{trip}(a, \hat{p}, n)) + \beta_1 L_{ce}(f(\hat{x}), \hat{y}) \tag{4.9}$$

For triplets $(I1; I1; I2), (I1; I1; O)$, Equation 4.5 is used for the loss.

4.3 Evaluation metrics

We use standard accuracy to evaluate classification for in-distribution classes. We use the True Negative Rate at 95% of the True Positive Rate (TNR@95TPR, or TNR95 for short) as our primary evaluation metric for detecting outliers. This metric is prevalent across OoD work (Lee et al., 2017; Zhang et al., 2023; Masana et al., 2018).

Thus, TNR@95TPR is defined as:

$$TNR = \frac{TN}{FP + TN} \tag{4.10}$$

when TPR = 95%:

$$TPR = \frac{TP}{FN + TP}, \tag{4.11}$$

TN = True Negative, FN = False Negative, TP = True Positive, FP = False Positive. With this metric, we assume that the model predicts 95% inliers correctly, and then we measure the detection rate when outliers are correctly identified.

4.3.1 Confidence-based True Negative Rate

To calculate TNR@95TPR, we need to use the confidence score derived from the maximum softmax score. True Positive Rate is defined such that 95% of inlier scores are correctly identified. For that, a confidence-based threshold is estimated by sorting these scores in increasing order. The window is used to find the threshold that reaches a 95% TPR rate. Once this threshold is set, True Negative Rate is calculated as in Equation 4.10. TNR@95TPR is defined and implemented as in (Lee et al., 2018) and (Zhang et al., 2023).

Chapter 5

Experiments

5.1 Implementation and experimental details

5.1.1 Implementation and training details

Two NVIDIA Tesla V100 GPU cards are used to execute the experiments. The batch size is 32, SGD optimiser (Ruder, 2016) and a step scheduler with cosine annealing for learning rate are used.

The implementation is based on the MixOE (Zhang et al., 2023), which includes the datasets, model as well as codes for setting them up. Their codes for confidence-based TNR95@TPR metric calculation are also used since they are the same across multiple research work (Lee et al., 2017). The model used is ResNet50 (He et al., 2016). The implementation is based on PyTorch (Paszke et al., 2019) and NumPy (Harris et al., 2020) for model training and data processing. Matplotlib (Hunter, 2007) is used for visualisations.

All experiments are trained on top of the baseline for ten epochs, following MixOE conventions. The model, which is ResNet50, is trained for 90 epochs separately for each ID split of each dataset using standard cross-entropy loss with no outliers in training and is considered a baseline. The model is not pre-trained. In the original paper, the results are reported separately for every split. In our work, we report the averaged result across all three splits. For each split, the experiments are executed 5 times. **We report the mean value and standard deviation for every dataset across 15 runs, five runs per split.** For every run, a random set of outliers is used out of the auxiliary outlier dataset, that is, WebVision 1.0 (Li et al., 2017).

We compare detection (TNR@95TPR) and classification (accuracy) performance metrics of the MixOE approach and the baseline for each dataset with our experiments.

5.1.2 Datasets

The dataset setup is taken from MixOE (Zhang et al., 2023). The setup includes three different splits of four datasets, namely, Car (Krause et al., 2013), Bird (Van Horn et al., 2015), Butterfly (Chen et al., 2018) and Aircraft (Maji et al., 2013).

Each split of each dataset is divided into ID classes for training and OoD classes for fine-grained OoD detection testing. These classes are never seen during training and are only used to evaluate the efficiency of the methods. For coarse-grained testing settings, other datasets different from the one the model is trained on are used. Each ID split is divided into training, validation and testing sets. Classification accuracy is calculated using this ID testing set. The detection metric TNR@TPR95 for testing is derived from either OoD set from the split for the fine-grained setting or the other three datasets for the coarse-grained setting.

The auxiliary outlier dataset used only for training is WebVision 1.0 (Li et al., 2017). The classes of cars, birds, butterflies and aircraft vehicles are removed from WebVision 1.0, resulting in 1948K images in total (Zhang et al., 2023).

5.1.3 Experimental details

Mixup variants

In the first part of our experiments, we consider seven Mixup variants that mix inliers and outliers in a way different from the original MixOE technique mentioned in Section 2.2.3. The codes for AlignMixup (Venkataramanan et al., 2022a), Manifold Mixup (Verma et al., 2019) and Outlier Exposure (Hendrycks, Mazeika, and Dietterich, 2019) were taken from corresponding implementations included with their manuscripts to run these experiments.

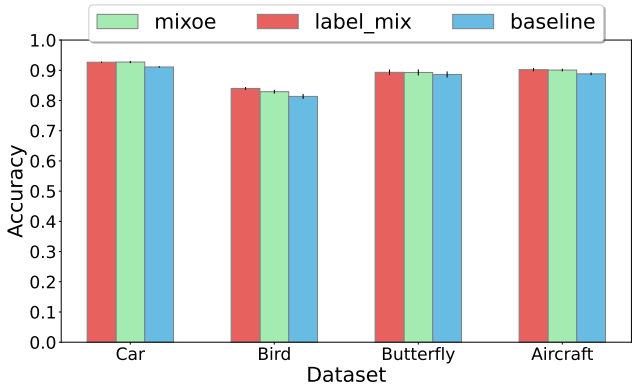

FIGURE 5.1: Classification comparison between MixOE (green), baseline (blue) and Mixup with labels (red). Mean values are reported. Standard deviation is reported in the form of error bars.

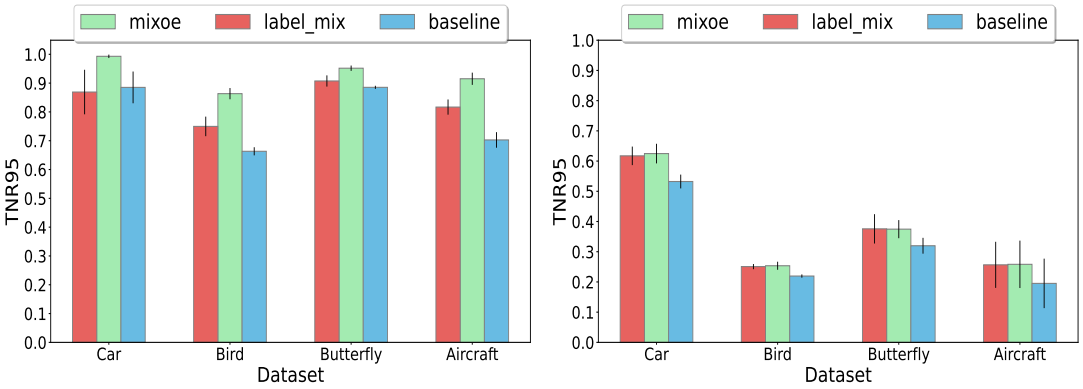

FIGURE 5.2: Detection comparison between MixOE (green), baseline (blue) and Mixup with labels (red) using confidence-based metric. Fine-grained on the right and coarse-grained on the left.

We now show the results for different Mixup variants, as discussed in Chapter 4 in Tables 5.1-5.3. The baseline is the model trained for 90 epochs on ID data, and MixOE with its variants are trained for ten epochs on top of the baseline. Manifold MixOE and Align MixOE are implemented according to their original implementations, but outliers are added for the mixing. Mixup with inliers is the original Input Mixup, as discussed in 2.1.1. Mixup with a noise image and MixOE with KNN are implemented,

as discussed in 4.1.2. MixOE (min) involves sampling five outliers with the lowest confidence per batch and mixing every inlier in the ID batch with a random outlier out of these five. MixOE (min) with OE loss includes these five lowest confidence outliers in the loss as in Equation 4.1. For all Mixup experiments, we use $\lambda = 1.0$, which is an interpolation factor for Mixup, and $\beta = 5.0$, which is a weighting factor for the cross-entropy loss used for MixOE.

Mixup variant	Car	Bird	Butterfly	Aircraft
Baseline	88.51 ± 5.51	66.38 ± 1.46	88.53 ± 0.54	70.27 ± 2.70
MixOE	99.30 ± 0.59	86.34 ± 1.94	**95.17 ± 0.92**	91.51 ± 2.13
Mixup with inliers	84.85 ± 8.34	68.46 ± 3.58	87.77 ± 2.62	80.50 ± 3.79
Mixup with a noise image	87.97 ± 7.89	74.48 ± 4.79	90.96 ± 1.48	76.92 ± 3.85
MixOE (min)	97.98 ± 1.10	80.41 ± 2.58	93.97 ± 0.68	88.39 ± 2.52
MixOE (min) with OE loss	87.48 ± 15.11	65.11 ± 12.69	78.26 ± 6.96	85.70 ± 10.05
MixOE with KNN	98.85 ± 0.70	84.76 ± 3.00	94.28 ± 0.86	43.62 ± 38.73
Manifold MixOE	**99.51 ± 0.38**	**89.62 ± 2.52**	94.93 ± 0.97	88.17 ± 3.73
Align MixOE	99.48 ± 0.24	87.39 ± 3.50	94.24 ± 2.37	**92.95 ± 2.36**

TABLE 5.1: Detection comparison of Mixup variants in a coarse-grained setting using TNR@TPR95.

Mixup variant	Car	Bird	Butterfly	Aircraft
Baseline	53.24 ± 2.28	21.95 ± 0.54	31.98 ± 2.62	19.52 ± 8.18
MixOE	**62.48 ± 3.24**	25.36 ± 1.33	37.47 ± 2.99	25.83 ± 7.84
Mixup with inliers	61.09 ± 2.70	**26.47 ± 1.32**	34.25 ± 3.53	25.68 ± 6.29
Mixup with a noise image	55.69 ± 4.58	23.03 ± 2.06	34.58 ± 4.98	21.93 ± 6.96
MixOE (min)	56.94 ± 3.54	24.21 ± 1.32	36.99 ± 4.99	24.15 ± 7.78
MixOE (min) with OE loss	24.88 ± 8.09	13.79 ± 2.55	22.57 ± 3.73	8.36 ± 5.23
MixOE with KNN	61.56 ± 3.04	24.98 ± 1.09	35.47 ± 3.7	5.57 ± 2.96
Manifold MixOE	60.90 ± 3.73	24.92 ± 1.40	**37.53 ± 4.52**	**26.26 ± 7.85**
Align MixOE	55.66 ± 2.88	22.8 ± 1.44	35.83 ± 3.4	22.56 ± 7.11

TABLE 5.2: Detection comparison of Mixup variants in a fine-grained setting using TNR@TPR95.

Mixup variant	Car	Bird	Butterfly	Aircraft
Baseline	91.09 ± 0.27	81.35 ± 0.81	88.66 ± 1.02	88.82 ± 0.27
MixOE	**92.72 ± 0.36**	82.91 ± 0.62	**89.27 ± 1.01**	**90.07 ± 0.44**
Mixup with inliers	92.62 ± 0.31	**83.14 ± 0.74**	**89.27 ± 1.12**	90.01 ± 0.32
Mixup with a noise image	91.36 ± 0.87	82.00 ± 1.49	88.61 ± 0.76	86.54 ± 2.88
MixOE (min)	91.76 ± 0.47	81.49 ± 1.18	88.58 ± 0.84	89.42 ± 0.45
MixOE (min) with OE loss	78.20 ± 3.99	69.07 ± 1.44	80.40 ± 1.48	71.44 ± 5.23
MixOE with KNN	92.41 ± 0.49	82.62 ± 0.65	88.77 ± 0.82	22.36 ± 19.51
Manifold MixOE	92.44 ± 0.48	82.95 ± 0.91	89.05 ± 1.09	89.63 ± 0.49
Align MixOE	91.77 ± 0.52	82.48 ± 0.73	88.45 ± 1.10	89.79 ± 0.47

TABLE 5.3: Classification comparison of Mixup variants using accuracy.

Changing outlier distribution

We conduct experiments for varying the number of classes and samples in the auxiliary outlier dataset. There are a total of 1000 classes or concepts.

Varying classes. Since we run each experiment per split five times, each time, different random $N \in \{1, 5, 10, 15, 50, 100, 500\}$ classes from WebVision 1.0 are sampled.

Varying the number of outliers. We experiment with $N \in \{1, 5, 10, 15, 20, 50, 100, 250, 500, 1000, 10000, 20000\}$ different outliers, which are randomly sampled for each run.

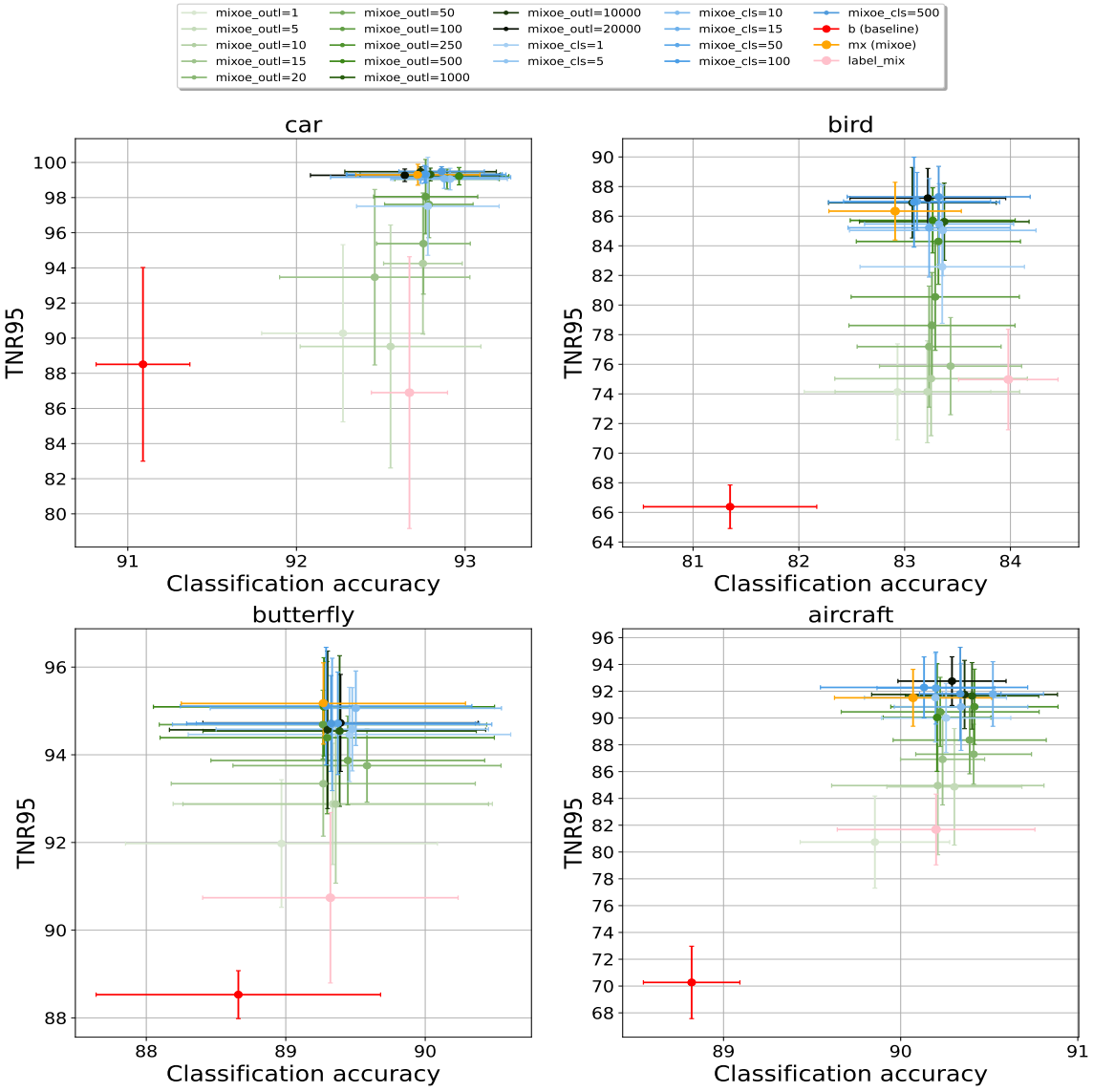

FIGURE 5.3: Coarse-grained comparison between MixOE, baseline, Mixup with labels (label_mix) and Mixup with a different number of outliers (mixoe_outl, green) or classes (mixoe_cls, blue) using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. Every experiment for this and further plots is reported as a mean value and standard deviation in the form of error bars. The darker the green (blue) colour is, the more outliers (classes) are used during training.

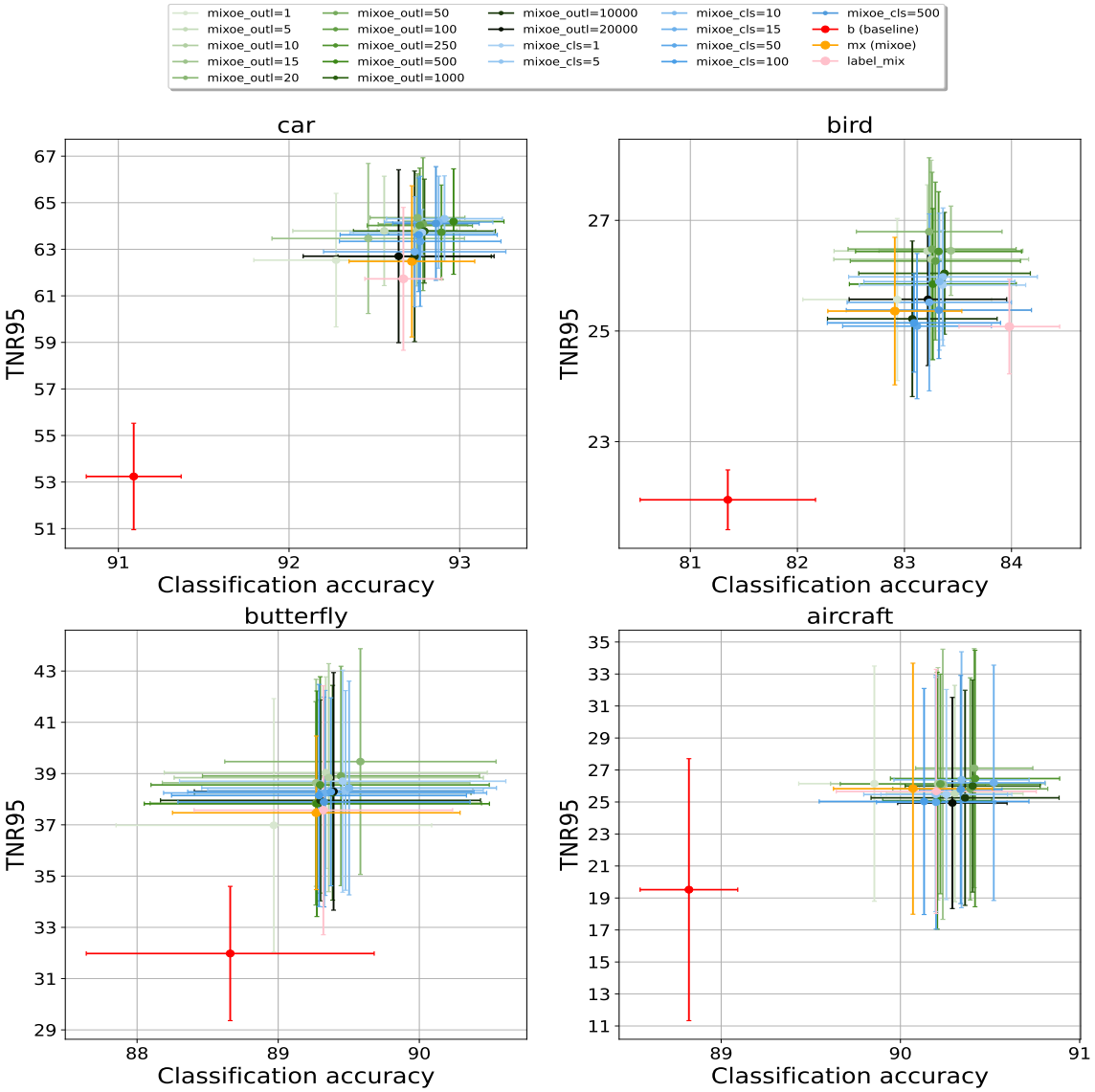

FIGURE 5.4: Fine-grained comparison between MixOE, baseline, Mixup with labels (label_mix) and Mixup with a different number of outliers (mixoe_outl, green) or classes (mixoe_cls, blue) using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the green (blue) colour is, the more outliers (classes) are used during training.

Metric learning with and without Mixup for OoD detection

We show the results of using the triplets mentioned in Table 4.1. We vary the weighting factor $\beta \in \{0.1, 0.3, 0.5\}$ and margin $m \in \{0.05, 0.1, 0.5, 1.0\}$, introduced in Equation 4.5. Anchors are ID examples from the inlier batch, positive examples have the same class as the anchor in the batch, and negative examples, depending on the triplet, are the farthest examples from the anchor from either the other ID class or an outlier.

We provide Figures 5.5 and 5.6 for the coarse- and fine-grained settings based on the hyperparameters that result in the best coarse-grained detection performance. We describe results using the full plots with different values of these hyperparameters in Figures 5.7-5.14.

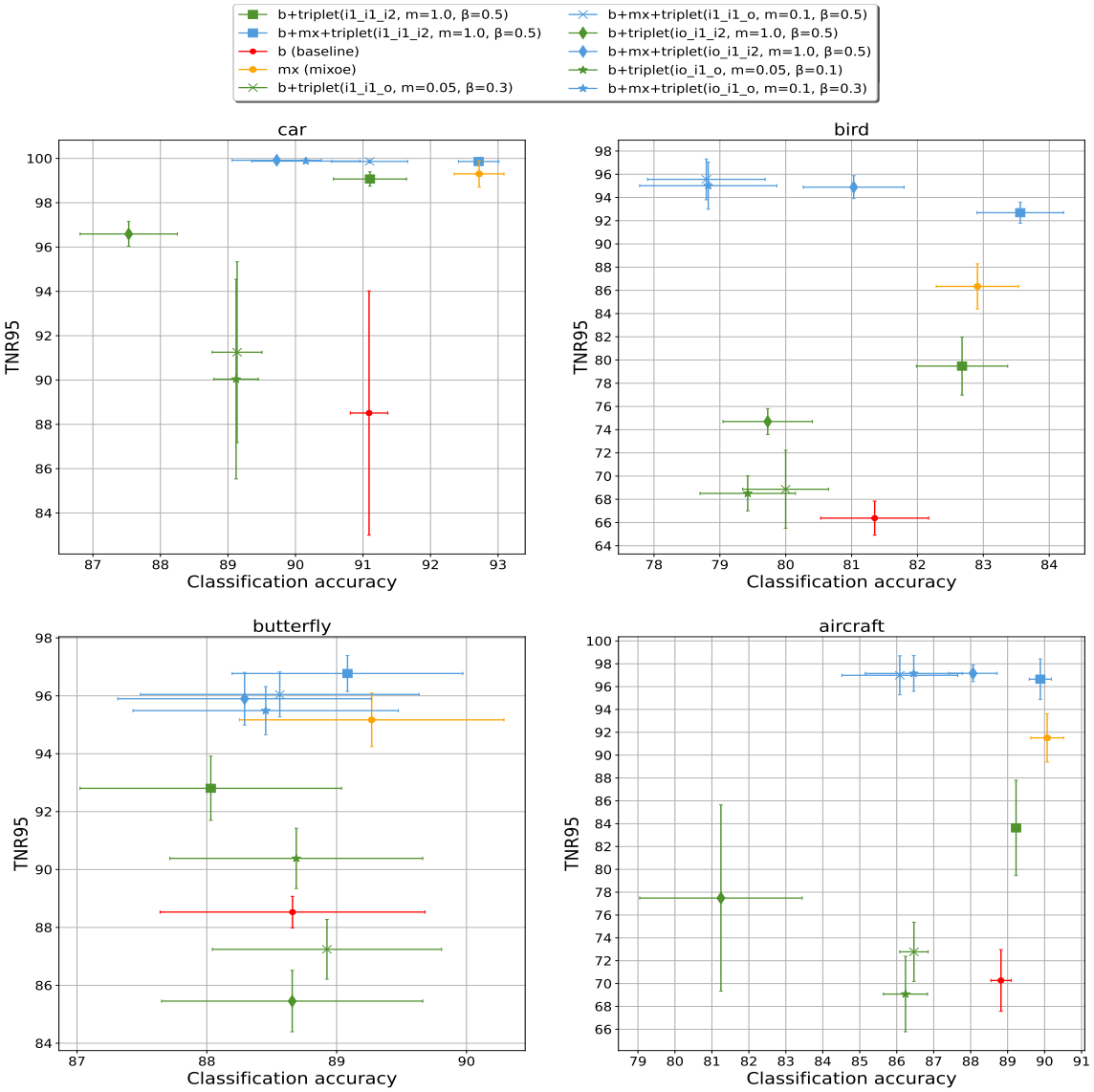

FIGURE 5.5: Coarse-grained comparison between MixOE, baseline and four different combinations of triplets according to Table 4.1 with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. Every experiment for this and further plots is reported as a mean value and standard deviation in the form of error bars.

5.2 Results

5.2.1 Impact of the size of auxiliary outlier dataset and Mixup with labels

Given Figure 5.2 and Figures 5.3, 5.4, we can see three patterns:

- The number of outliers affects the TNR@95TPR for the coarse-grained setting in Figure 5.3. The fewer inliers there are, the lower the detection performance and vice versa.
- Mixing only labels without mixing inlier and outlier images performs almost as well as full MixOE for the fine-grained settings in Figure 5.2.

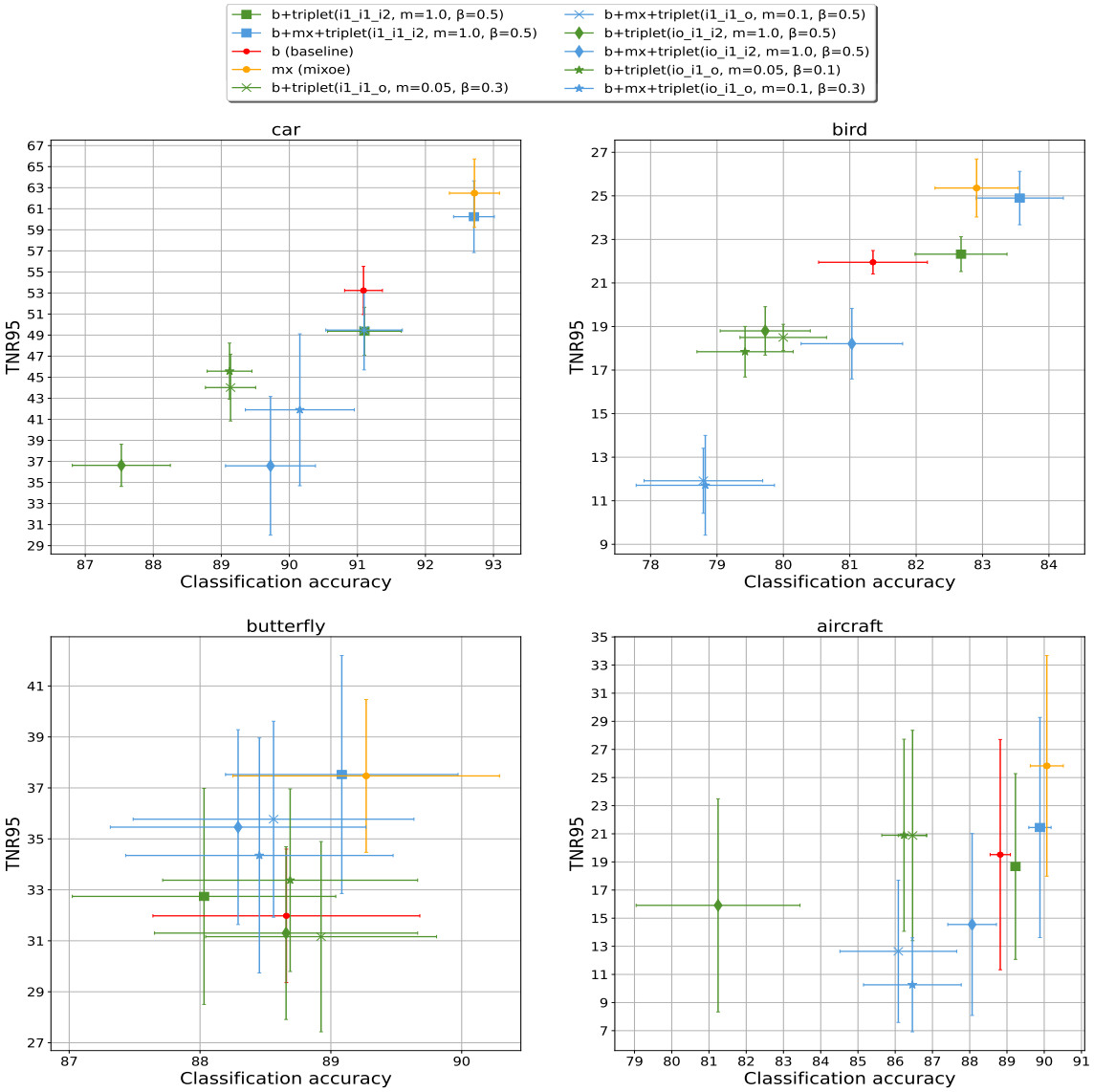

FIGURE 5.6: Fine-grained comparison between MixOE, baseline and four different combinations of triplets according to Table 4.1 with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR based on best-performing hyperparameters for coarse-grained detection on the Car, Bird, Butterfly and Aircraft datasets.

- The number of outliers has no relationship with accuracy or TNR@95TPR for the fine-grained settings in Figure 5.4.

These findings suggest that:

1. If the problem is focused on detecting outliers that are similar to the training distribution, then making the labels of training samples less confident without using an additional auxiliary outlier dataset, which may be expensive to gather, gives a close performance as to when the auxiliary outlier dataset was used.

2. When the problem is about the coarse-grained outlier detection, the inclusion of outliers during training in the form of the auxiliary outlier dataset becomes more beneficial as the size of such a dataset grows.

5.2.2 Impact of the diversity of auxiliary outlier dataset

After experimenting with the number of classes, Figures 5.3 and 5.4 suggest that classification and detection metrics are already similar when as few as ten classes are used compared to all 1000.

5.2.3 Impact of Mixup variants

The results are given in Tables 5.1-5.3.

Input Mixup (Mixup with inliers) and Mixup with noise

Classification and fine-grained detection performance are comparable with MixOE, but Mixup with inliers generally works better than Mixup with noise. However, MixOE significantly improves the coarse-grained detection setting.

Manifold MixOE and Align MixOE

Manifold MixOE and Align MixOE target Mixup in between the layers of the neural network. Both variants perform similarly on classification and coarse-grained detection, except for Align MixOE for the fine-grained detection, which performs worse than MixOE and Manifold MixOE. Manifold MixOE is as good as MixOE, or for some datasets, gives a comparably better performance (Table 5.1).

MixOE with KNN

For three out of four datasets, the performance of MixOE with KNN is similar to MixOE. For Aircraft, we find that the performance is much worse than the baseline, and the standard deviation is high, meaning that for some runs in the splits, the neighbours are detrimental to the training. As to why this happens, further experimentation is needed, and no argumentative explanation can be provided.

MixOE with and without OE loss

In spite of our assumption that the five outliers with minimum confidence would help the learning, both MixOE with and without the OE loss term perform worse than MixOE. When the OE loss term is added only for these five outliers, we notice even further detection and classification performance degradation, as is also shown in the original MixOE work.

5.2.4 Impact of metric learning

The results are given in Figures 5.5-5.14. The mean value is reported. Standard deviation is reported in the form of error bars.

Impact of (I1, I1, I2) triplet

When using triplet loss without outliers during training in Figures 5.7 and 5.8 (green measurements), we see the division in the performance between fine- and coarse-grained settings. The **trade-off** between the detection of coarse- and fine-grained performance emerges. Specifically, as we increase the size of the margin, the detection and classification performance of coarse-grained settings improves, while in the fine-grained settings, they decrease. In this scenario, the performance improves over the

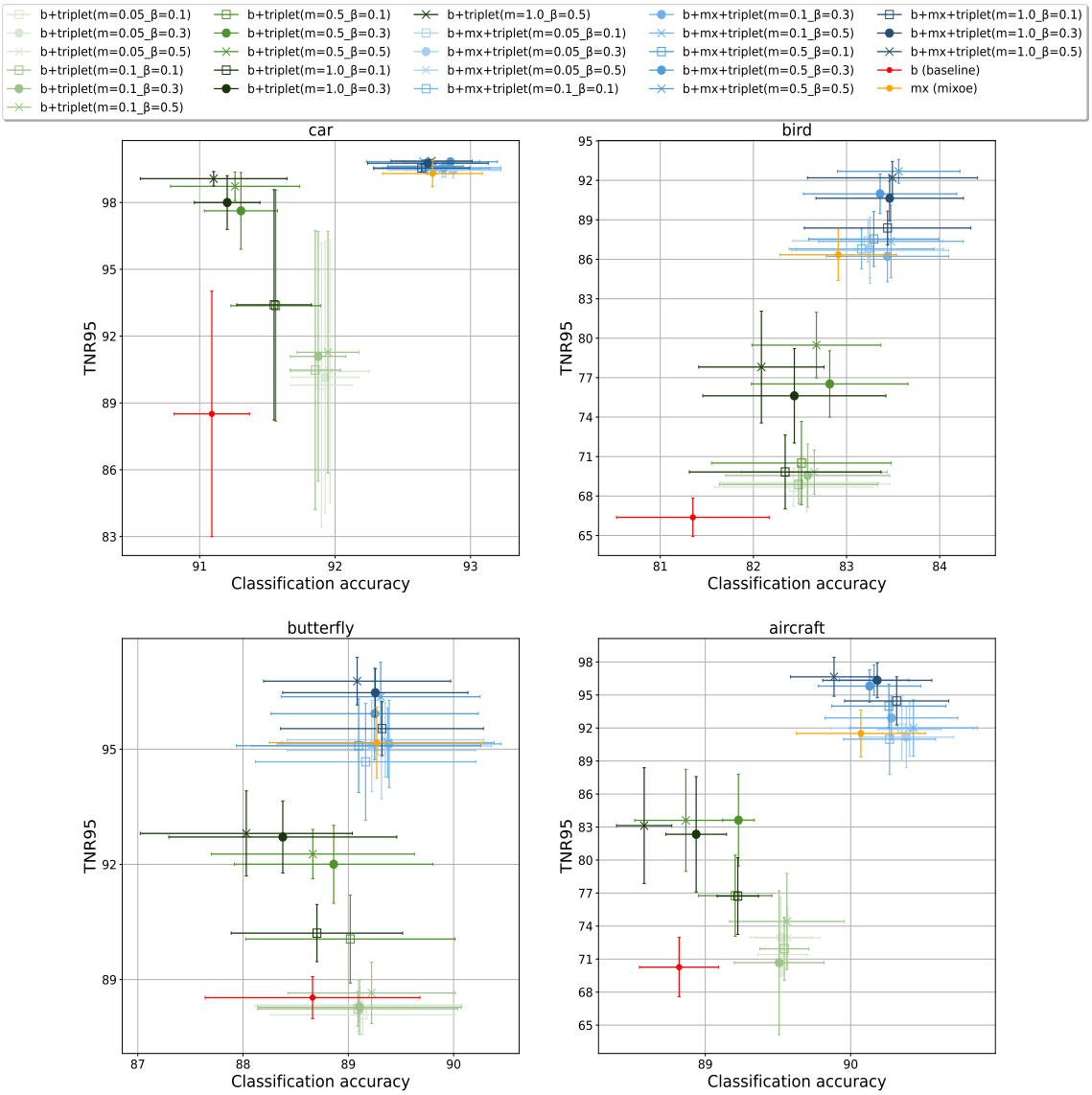

FIGURE 5.7: Coarse-grained comparison between MixOE, baseline and (I1; I1; I2) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. Every experiment for this and further plots is reported as a mean value and standard deviation in the form of error bars. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

baseline, but the performance of MixOE is still noticeably better. When we train with MixOE as in Equation 4.6 in Figures 5.7 (blue measurements), we see improvement in three out of four datasets in coarse-grained settings, but there is no trade-off between the margins. For the Car dataset, the detection metric is already around 99% for MixOE, so there is little room for improvement.

This triplet is the only one out of those that (with the right set of hyperparameters) gives improvement for the fine-grained settings for two out of four datasets (Car and Bird) in Figure 5.8 (green measurements). Such a triplet combination is a standard triplet that is used in metric learning and contrastive learning, and **is considerably the best-performing triplet combination out of all four**.

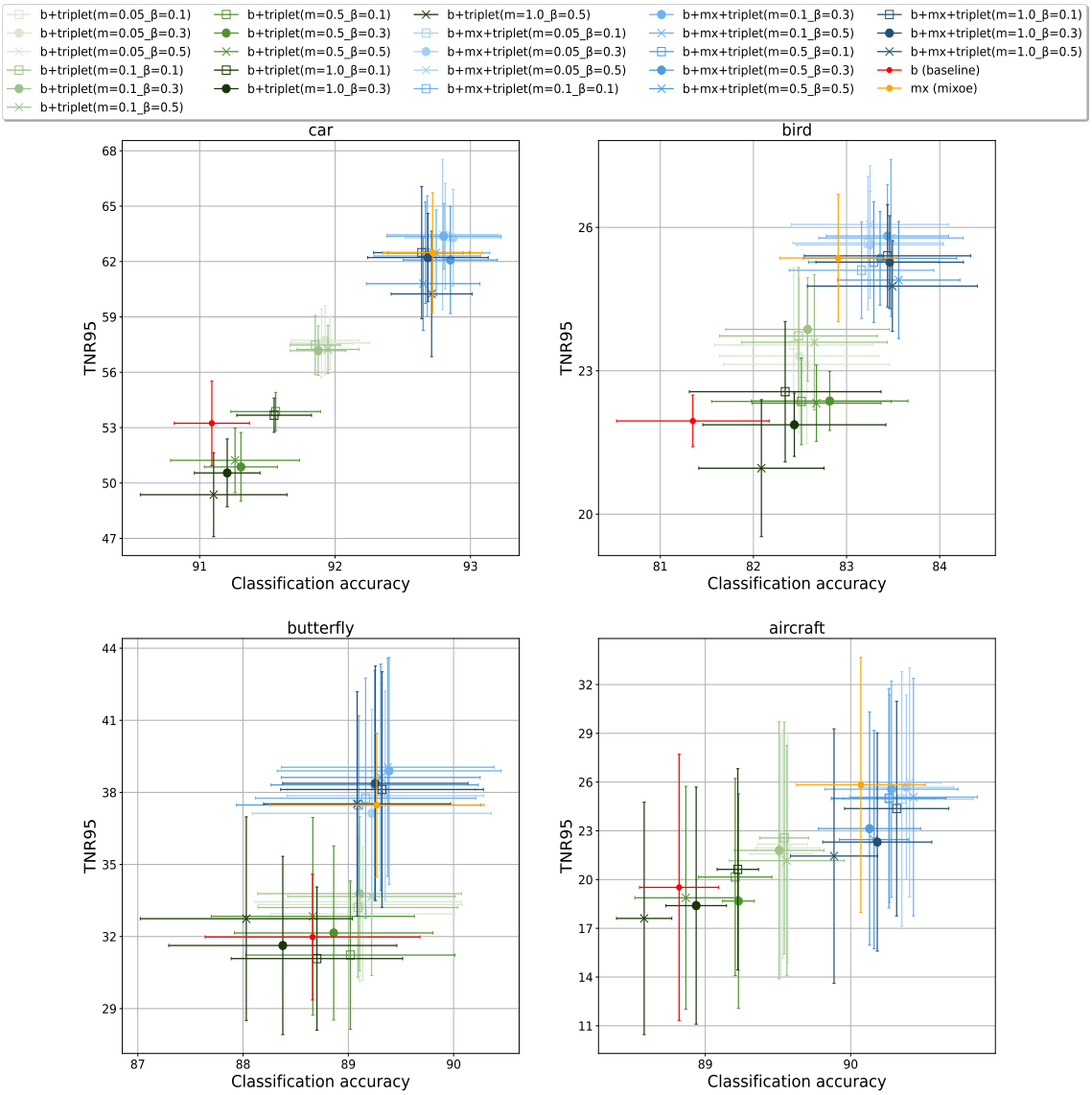

FIGURE 5.8: Fine-grained comparison between MixOE, baseline and (I1; I1; I2) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

Impact of hyperparameter β for the triplet loss

As was mentioned earlier, we experiment with three different values for β hyperparameter for the triplet loss as in Equations 4.5, 4.8 and 4.9. As a result of our experiments, we see that this hyperparameter does not produce trade-offs like the margin.

Impact of an outlier in the triplet

Triplet combinations, where a negative example is an outlier, do not significantly improve detection or classification performance compared to the baseline for all datasets and setups in Figures 5.9, 5.10 and Figures 5.11, 5.12 (green measurements).

In the case of a $(I1, O; I1; I2)$ triplet, that is, a random choice between $(I1, O; I1; I2)$ and $(I1; I1, O; I2)$, we see coarse-grained detection improvement in two out of four datasets (Car and Bird) with the right set of hyperparameters, accompanied by slight degradation in accuracy in Figure 5.13 (green measurements). The improvement is less significant than when $(I1; I1; I2)$ triplet is used in Figure 5.7 (green measurements). **Triplet combinations with outliers either worsen classification and fine-grained detection results or perform closely to the baseline**.

When adding MixOE, there appears a similar but less prominent trade-off with the margin for triplets with mixed anchors or positive examples in Figures 5.13, 5.14 for $(I, O; I1; I2)$ and Figures 5.11, 5.12 for $(I1, O; I1; O)$ (blue measurements).

Impact of MixOE with triplet loss

We notice that MixOE with metric learning improves the detection performance for the **coarse-grained settings**. It is seen in Figure 5.5, where the best detection results are displayed out of the set of hyperparameters:

- For $(I1; I1; I2)$, the classification accuracy stays the same while detection performance grows for the coarse-grained settings.
- For the pairs involving outliers, namely $(I1; I1; O)$, $(I1; I1, O; I2)$, and $(I1; I1, O; O)$, there is an improvement in the coarse-grained detection performance compared to MixOE, but classification accuracy is compromised. There is no improvement in the fine-grained setup compared to the baseline.

When triplet $(I1; I1; I2)$ is combined with MixOE, the classification and detection performance stays comparably the same as in MixOE for the **fine-grained settings** in Figures 5.6. For other triplets, the performance is generally lower or the same as the baseline.

Since Mixup is used for decreasing the confidence between the decision boundary and metric learning induces more strictness between different classes of the training data, it is surprising for us to see that the combination of these two seemingly opposite methods improves the coarse-grained detection performance.

Choosing best-performing hyperparameters

In Figures 5.5 and 5.6, we show the detection and classification performance based on hyperparameters that give the best detection performance for the coarse-grained settings. We see that for the fine-grained settings in Figure 5.6, the performance of triplet $(I1, I1, I2)$ with such hyperparameters is still dominant, especially when combined with MixOE. When such a triplet is used without MixOE, we see that, for some datasets, its performance is worse than or similar to the baseline.

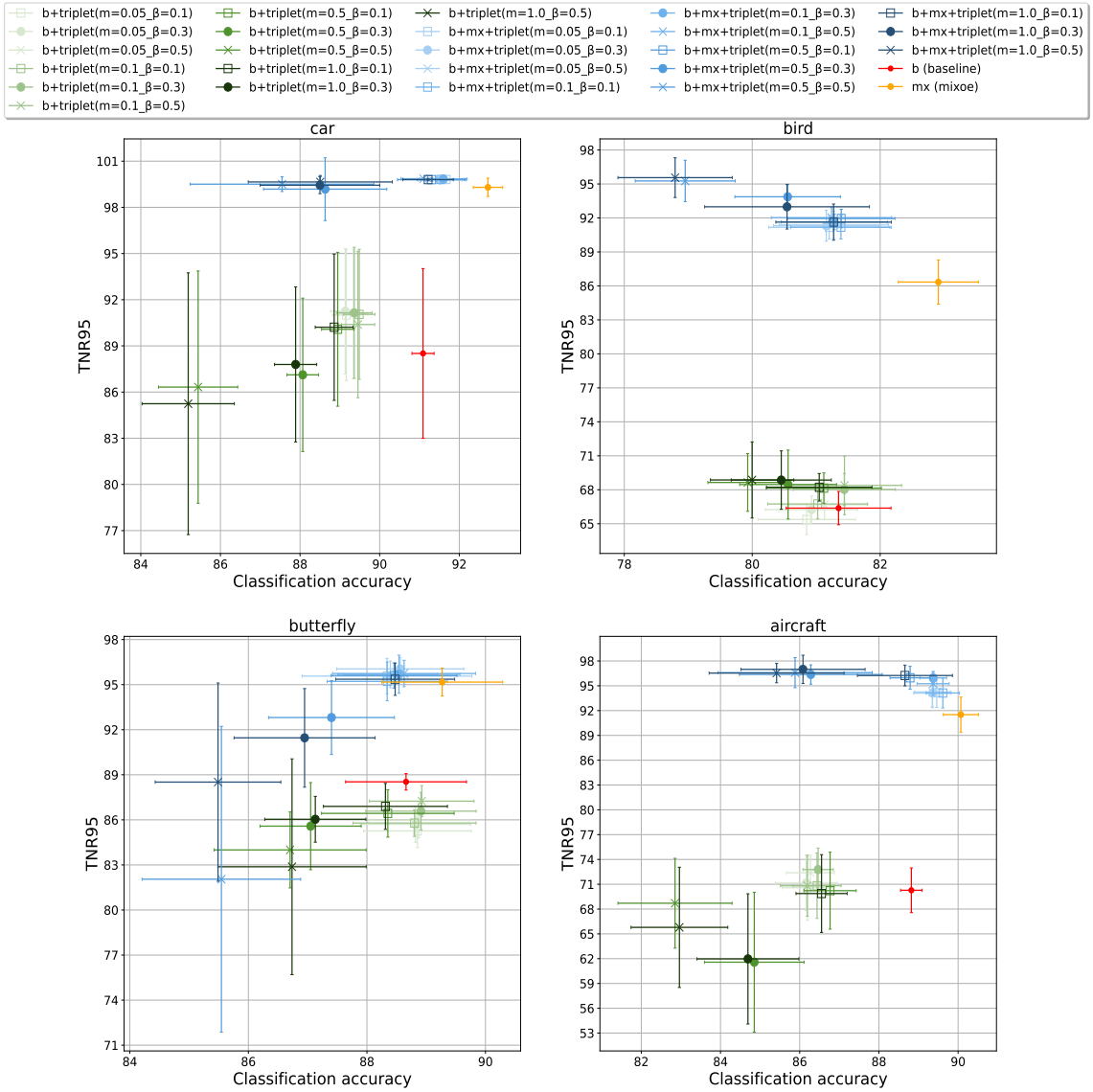

FIGURE 5.9: Coarse-grained comparison between MixOE, baseline and (I1; I1; O) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

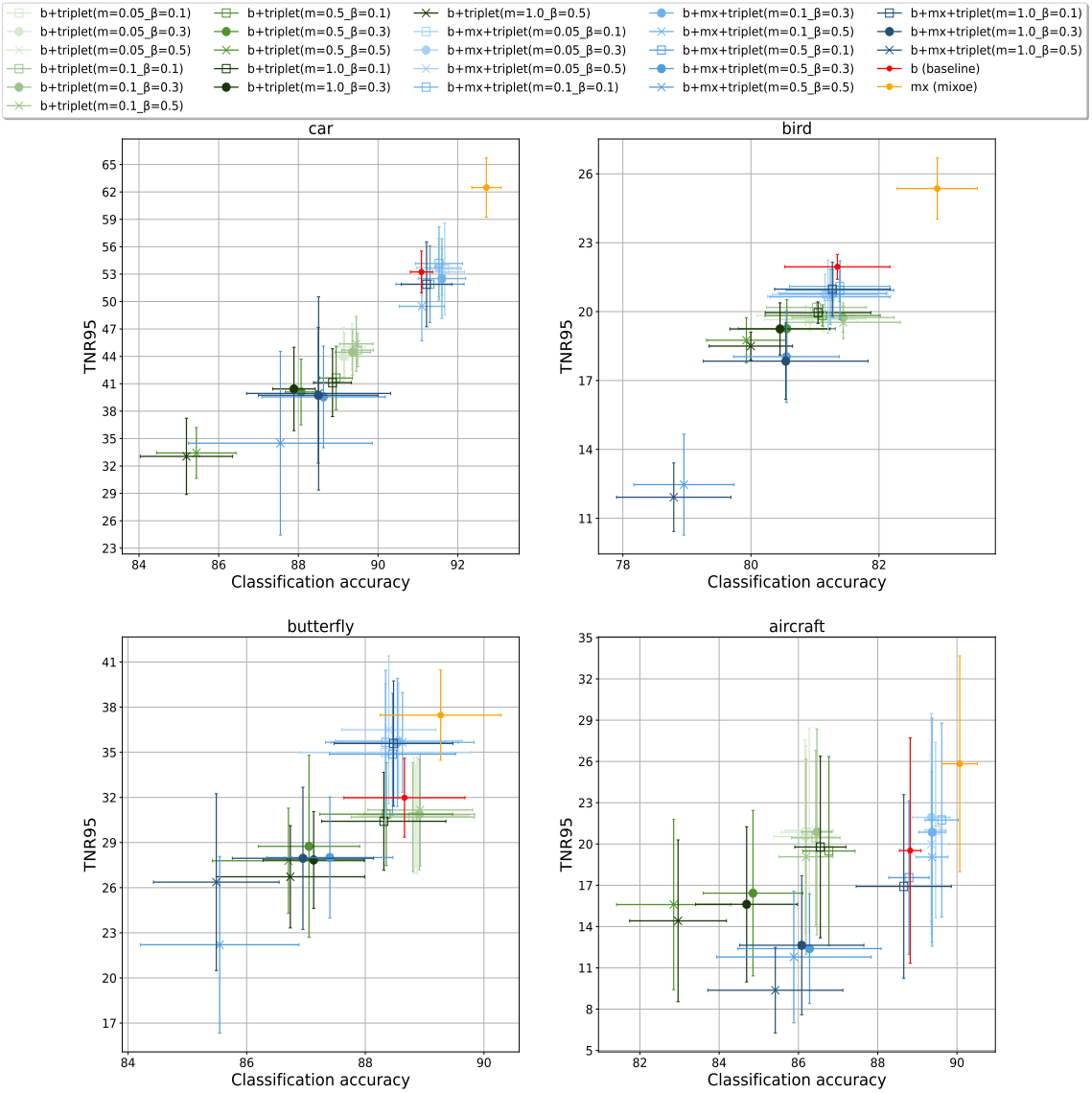

FIGURE 5.10: Fine-grained comparison between MixOE, baseline and (I1; I1; O) triplets with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

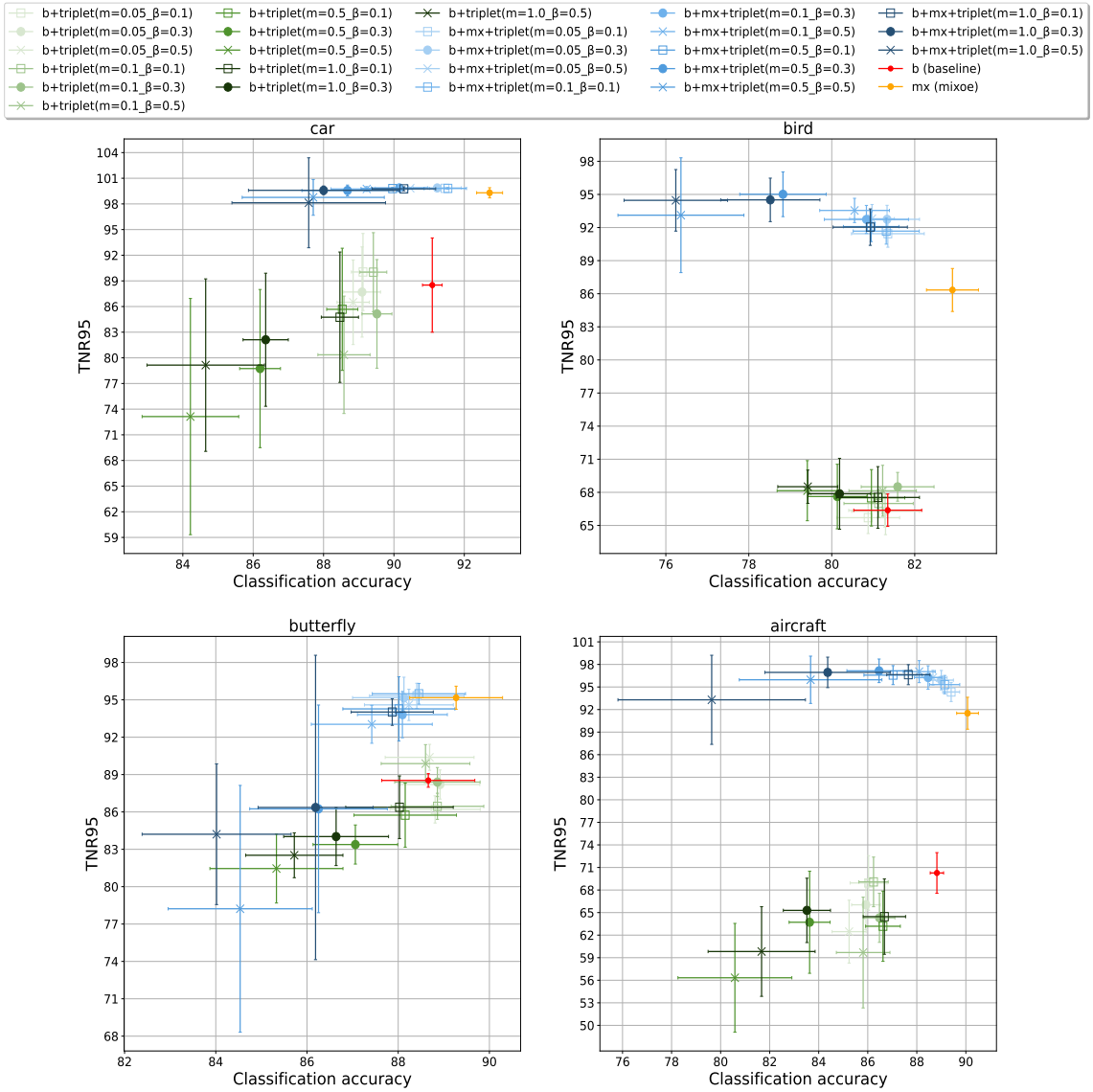

FIGURE 5.11: Coarse-grained comparison between MixOE, baseline and random selection between (I1,O; I1; O) and (I1; I1,O; O) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

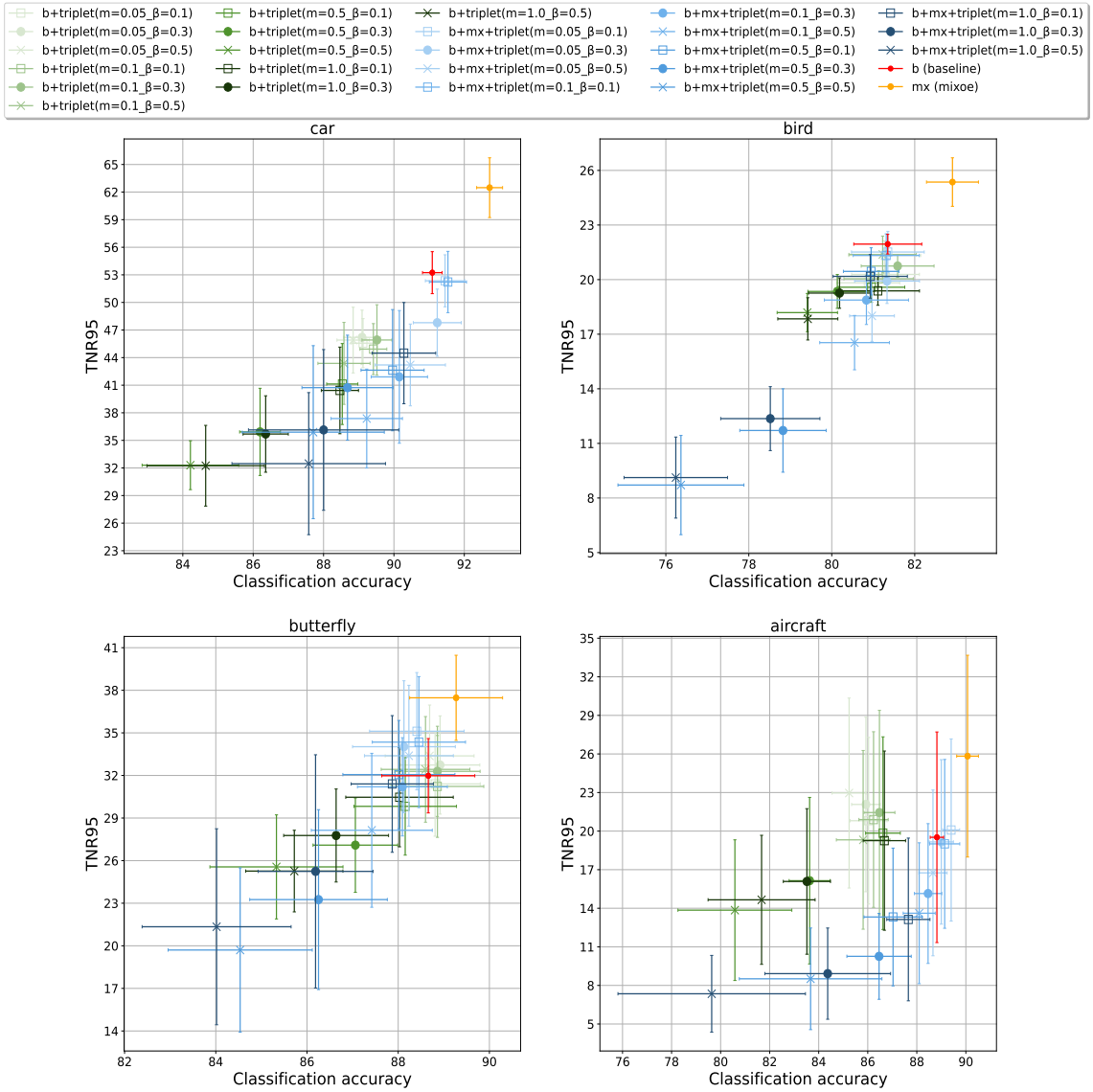

FIGURE 5.12: Fine-grained comparison between MixOE, baseline and random selection between (I1,O; I1; O) and (I1; I1,O; O) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

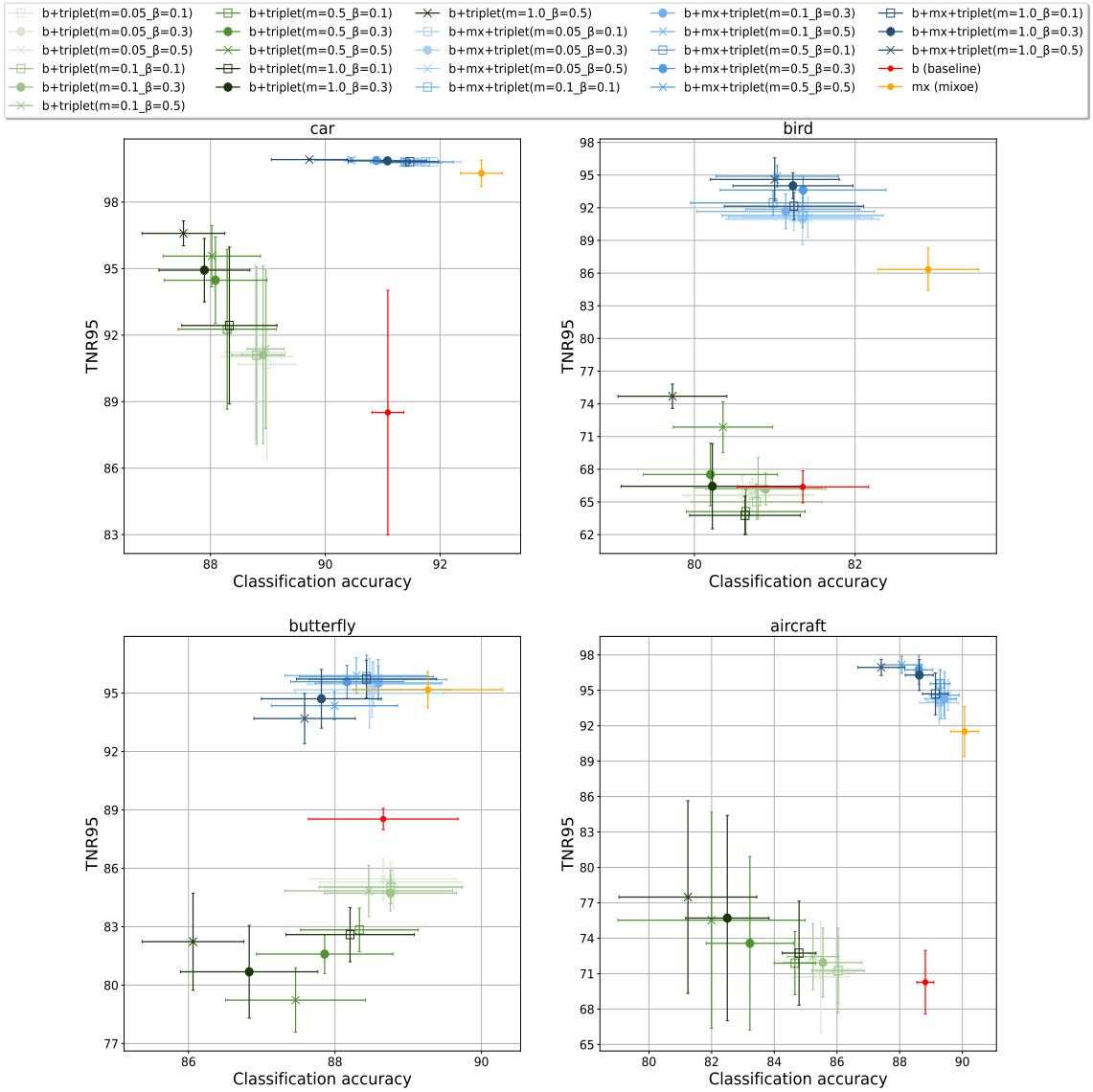

FIGURE 5.13: Coarse-grained comparison between MixOE, baseline and random selection between (I1,O; I1; I2) and (I1; I1,O; I2) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

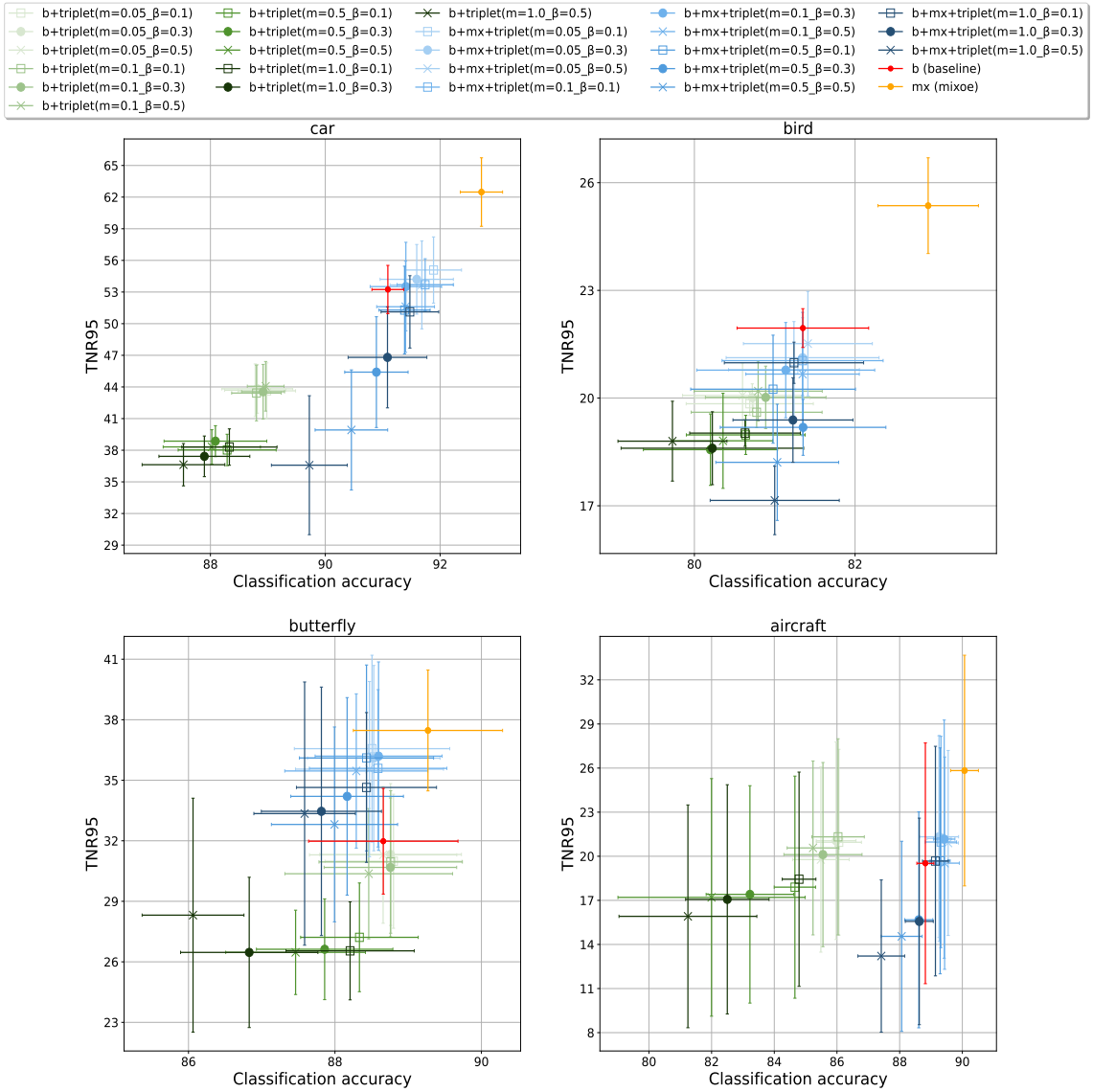

FIGURE 5.14: Fine-grained comparison between MixOE, baseline and random selection between (I1,O; I1; I2) and (I1; I1,O; I2) during each iteration with (blue) and without (green) MixOE using accuracy and confidence-based TNR@95TPR on the Car, Bird, Butterfly and Aircraft datasets. The darker the colour, the larger the margin. Different markers correspond to $\beta \in \{0.1; 0.3; 0.5\}$

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this work, we looked at the broad problem of OoD detection, which involves ID sample classification and OoD sample detection. We investigated the method of Mixup when it is used between outliers and inliers by changing the Mixup components and an auxiliary outlier dataset:

- We find that mixing only the label with the uniform label provides almost as good of a performance as the standard MixOE, dispensing with the collection of the auxiliary outlier dataset for fine-grained outlier detection.
- The large size of such a dataset for the coarse-grained settings significantly improves detection.

We also provide our findings on metric learning in OoD detection by using triplet loss with and without outliers via creating new triplet combinations, with and without Mixup between ID and OoD examples:

1. We find that triplets comprised of inliers of the same class as positive examples and inliers of different classes as negative examples provide the best results both for coarse- and fine-grained detection. This is a standard approach that is used in metric learning, including contrastive learning. A trade-off occurs between coarse- and fine-grained detection performance depending on the margin size.
2. We find that the triplet combinations with outliers, mixed or not, mostly worsen fine-grained detection results or perform similarly to the baseline. Detection performance improvement is observed on coarse-grained settings for such triplets, compromising classification accuracy.
3. Metric learning **without MixOE**:
    - Using metric learning without MixOE improves (with the right set of hyperparameters) the **coarse-grained** detection performance compared to the baseline for the $(I1; I1; I2)$ and $(I1, O; I1; I2)$ triplets for four and two datasets out of four, respectively (Figure 5.7 and 5.13, green measurements), that is, when negative examples are inliers from a different class from the anchor.
    - No triplet combination improves over the **fine-grained** detection MixOE performance. Only $(I1; I1; I2)$ improves (with the right set of hyperparameters) fine-grained detection performance compared to the baseline for two

datasets (Car and Bird) in Figure 5.8 (green measurements). Such a combination does not use the auxiliary outlier dataset, which is a considerable advantage. When the hyperparameters are used so that the coarse-grained detection performance is the best, as in Figure 5.6 (green measurements), mostly, the detection performance of such a triplet is close to the baseline.

4. Metric learning **with MixOE**:

   - For the **coarse-grained** detection, depending on the triplet combination, detection performance is either significantly improved over MixOE, or performs similarly to MixOE (Figure 5.5, blue measurements).
   - For the coarse-grained detection, triplet combination with only inliers, that is, $(I1, I1, I2)$, improves detection without decreasing classification accuracy.
   - For the coarse-grained detection, triplet combinations with outliers trade an increase in detection performance for the classification performance.
   - For the **fine-grained** settings, only $(I1; I1; I2)$ triplet combined with MixOE could produce comparably similar results to MixOE. Other triplets, which are $(I1; I1; O)$, $(I1, O; I1; O)$, $(I1, O; I1; I2)$ perform similarly to the baseline or worse (Figure 5.6, blue measurements).

6.2 Future Work

Mixup is a never-ending plethora of research, so future work could focus on the incorporation of Mixup with labels with other Mixup variants that do not use a uniform label during training.

As for metric learning, the combination of different types of triplets is worth investigating due to their diverse nature. For example, random choice between $(I1, I1, I2)$ and $(I1, I1, O)$ for each anchor among many other combinations.

Given the found benefit from the combination of MixOE and $(I1, I1, I2)$ triplet, future work includes finding the reasoning behind these results.

Bibliography

Aliakbarisani, Roya, Abdorasoul Ghasemi, and Shyhtsun Felix Wu (2019). "A data-driven metric learning-based scheme for unsupervised network anomaly detection". In: *Computers & Electrical Engineering* 73, pp. 71–83.

Baena, Raphael, Lucas Drumetz, and Vincent Gripon (2022). "Preventing manifold intrusion with locality: Local mixup". In: *arXiv preprint arXiv:2201.04368*.

Bodesheim, Paul et al. (2015). "Local Novelty Detection in Multi-class Recognition Problems". In: *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 813–820. DOI: 10.1109/WACV.2015.113.

Carratino, Luigi et al. (2020). "On mixup regularization". In: *arXiv preprint arXiv:2006.06049*.

Cen, Jun et al. (Oct. 2021). "Deep Metric Learning for Open World Semantic Segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15333–15342.

Chapelle, Olivier et al. (2000). "Vicinal risk minimization". In: *Advances in neural information processing systems* 13.

Chen, Jiefeng et al. (2021). "Atom: Robustifying out-of-distribution detection using outlier mining". In: *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21*. Springer, pp. 430–445.

Chen, Tianshui et al. (2018). "Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding". In: *Proceedings of the 26th ACM international conference on Multimedia*, pp. 2023–2031.

Cho, Hyunsoo, Jinseok Seol, and Sang-goo Lee (2021). "Masked Contrastive Learning for Anomaly Detection". In: *CoRR* abs/2105.08793. arXiv: 2105.08793. URL: https://arxiv.org/abs/2105.08793.

Chun, Sanghyuk et al. (2020). "An empirical evaluation on robustness and uncertainty of regularization methods". In: *arXiv preprint arXiv:2003.03879*.

Cuturi, Marco (2013). "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in neural information processing systems* 26.

Guo, Hongyu, Yongyi Mao, and Richong Zhang (2019). "Mixup as locally linear out-of-manifold regularization". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01, pp. 3714–3722.

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE, pp. 1735–1742.

Harris, Charles et al. (Sept. 2020). "Array programming with NumPy". In: *Nature* 585, pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

He, Kaiming et al. (June 2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hein, Matthias, Maksym Andriushchenko, and Julian Bitterwolf (2019). "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem". In.

Hendrycks, Dan and Kevin Gimpel (2016). "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *arXiv preprint arXiv:1610.02136*.

Hendrycks, Dan, Mantas Mazeika, and Thomas Dietterich (2019). "Deep Anomaly Detection with Outlier Exposure". In: *Proceedings of the International Conference on Learning Representations*.

Hsu, Yen-Chang et al. (2020). "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10951–10960.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Jézéquel, Loïc et al. (2022). "Semi-supervised anomaly detection with contrastive regularization". In: *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 2664–2671.

Koner, Rajat et al. (2021). "OODformer: Out-Of-Distribution Detection Transformer". In: *arXiv preprint arXiv:2107.08976*.

Krause, Jonathan et al. (2013). "3D Object Representations for Fine-Grained Categorization". In: *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561. DOI: 10.1109/ICCVW.2013.77.

Lee, Kimin et al. (2017). "Training confidence-calibrated classifiers for detecting out-of-distribution samples". In: *arXiv preprint arXiv:1711.09325*.

Lee, Kimin et al. (2018). "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in neural information processing systems* 31.

Lee, Minjung and Seoung Bum Kim (2022). "Sensor-Based Open-Set Human Activity Recognition Using Representation Learning With Mixup Triplets". In: *IEEE Access* 10, pp. 119333–119344.

Li, Wen et al. (2017). "Webvision database: Visual learning and understanding from web data". In: *arXiv preprint arXiv:1708.02862*.

Li, Xuan, Christian Desrosiers, and Xue Liu (2022). "Symmetric Contrastive Loss for Out-of-Distribution Skin Lesion Detection". In: *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5. DOI: 10.1109/ISBI52829.2022.9761434.

Liang, Shiyu, Yixuan Li, and Rayadurgam Srikant (2017). "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *arXiv preprint arXiv:1706.02690*.

Liu, Aishan et al. (2021). "Training robust deep neural networks via adversarial noise propagation". In: *IEEE Transactions on Image Processing* 30, pp. 5769–5781.

Maji, Subhransu et al. (2013). "Fine-grained visual classification of aircraft". In: *arXiv preprint arXiv:1306.5151*.

Mao, Chengzhi et al. (2019). "Metric learning for adversarial robustness". In: *Advances in Neural Information Processing Systems* 32.

Masana, Marc et al. (2018). "Metric Learning for Novelty and Anomaly Detection". In: *British Machine Vision Conference (BMVC)*.

Papadopoulos, Aristotelis-Angelos et al. (2021). "Outlier exposure with confidence control for out-of-distribution detection". In: *Neurocomputing* 441, pp. 138–150. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.02.007. URL: https://www.sciencedirect.com/science/article/pii/S0925231221002393.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc. URL: https://proceedings.

neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Ravikumar, Deepak et al. (2020). "Exploring Vicinal Risk Minimization for Lightweight Out-of-Distribution Detection". In: *arXiv preprint arXiv:2012.08398*.

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747*.

Sun, Yiyou et al. (2022). "Out-of-distribution Detection with Deep Nearest Neighbors". In: *ICML*.

Tack, Jihoon et al. (2020). "CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances". In: *Advances in Neural Information Processing Systems*.

Techapanurak, Engkarat, Masanori Suganuma, and Takayuki Okatani (2019). "Hyperparameter-free out-of-distribution detection using softmax of scaled cosine similarity". In: *arXiv preprint arXiv:1905.10628*.

Thulasidasan, Sunil et al. (2019). "On mixup training: Improved calibration and predictive uncertainty for deep neural networks". In: *Advances in Neural Information Processing Systems* 32.

Van Horn, Grant et al. (2015). "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.

Venkataramanan, Shashanka et al. (2022a). "AlignMixup: Improving Representations By Interpolating Aligned Features". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Venkataramanan, Shashanka et al. (2022b). "It Takes Two to Tango: Mixup for Deep Metric Learning". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ZKy2X3dgPA.

Verma, Vikas et al. (June 2019). "Manifold Mixup: Better Representations by Interpolating Hidden States". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 6438–6447. URL: http://proceedings.mlr.press/v97/verma19a.html.

Wang, Jiang et al. (2014). "Learning fine-grained image similarity with deep ranking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1386–1393.

Winkens, Jim et al. (2020). "Contrastive training for improved out-of-distribution detection". In: *arXiv preprint arXiv:2007.05566*.

Yang, Donghun et al. (2020). "Out-of-distribution detection based on distance metric learning". In: *The 9th International Conference on Smart Media and Applications*, pp. 214–218.

Yun, Sangdoo et al. (2019). "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *International Conference on Computer Vision (ICCV)*. published.

Zhang, Hongyi et al. (2018). "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

Zhang, Jingyang et al. (Jan. 2023). "Mixture Outlier Exposure: Towards Out-of-Distribution Detection in Fine-Grained Environments". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5531–5540.