IMPROVING OUT-OF-DISTRIBUTION DETECTION IN DEEP MODELS BY LATENT SPACE ANALYSIS

by

Ozan Özgür

B.S., Electrical Electronics Engineering, Boğaziçi University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to thank my supervisor, İnci Meliha Baytaş, for providing invaluable guidance, feedback, and encouragement throughout my journey in this research and in my master's program at Boğaziçi University.

I am also thankful to my jury members, Lale Akarun and Ayşe Tosun, for their time and constructive feedback.

I also wish to thank my family and friends for their support and encouragement.

Thank you all for making this accomplishment possible.

ABSTRACT

IMPROVING OUT-OF-DISTRIBUTION DETECTION IN DEEP MODELS BY LATENT SPACE ANALYSIS

Despite their widely known success in various applications, deep models have vulnerabilities due to their inner mechanics. Sensitivities against adversarial and out-of-distribution (OOD) samples are some of the issues causing state-of-the-art networks to make wrong predictions with high confidence. These problems are critical for high-stakes applications such as self-driving vehicles and medical imaging. The underlying reasons for this sensitivity could be explained by overfitting and the trade-off between generalization and robustness. Some studies investigate the latent spaces of the networks to gain a more in-depth understanding of why networks capture some details that lead to poor generalization. This thesis focuses on understanding the inner dynamics of deep neural networks (DNNs) by investigating their intermediate layer weights and activations. A latent space exploration method is designed to analyze the intermediate layer activations of DNNs. The adversarial robustness of DNNs is studied based on the observations from this analysis. Finally, an out-of-distribution (OOD) sample detection approach based on the intermediate layer activations and OOD samples is proposed. The proposed OOD detection method trains a set of prototype vectors on the in-distribution (ID) dataset. The prototype vectors are used to extract features from ID and OOD sample activations to train a multi-layer perceptron for discriminating ID and OOD instances. Experiments on several benchmark datasets show that the proposed OOD detection approach could perform state-of-the-art.

ÖZET

GİZLİ UZAY ANALİZİ İLE DERİN MODELLERDE DAĞILIM DIŞI ALGILAMAYI GELİŞTİRMEK

Çeşitli uygulamalarda geniş çapta bilinen başarılarına rağmen, derin modellerin bazı zayıflıkları bulunmaktadır. Hasmane ve dağılım dışı örneklere karşı duyarlılık, yüksek performanslı derin ağların yüksek bir güvenle yanlış tahminler yapmasına yol açan sorunlardan bazılarıdır. Bu problemler, otonom araçlar ve tıbbi görüntüleme gibi yüksek riskli uygulamalar için kritik öneme sahiptir. Bu duyarlılığın altında yatan sebeplerden bazıları aşırı uyumlama ve genelleme ile dayanıklılık arasındaki ikilem olarak açıklanabilir. Bazı çalışmalar ise derin sinir ağlarının kötü genellemeye yol açacak bazı ayrıntılara neden önem verdiğini daha iyi anlayabilmek için sinir ağlarının gizli uzaylarını incelemektedir. Bu tez, derin sinir ağlarının içsel dinamiklerini anlamaya odaklanarak, ağların ara katman ağırlıklarını ve aktivasyonlarını incelemektedir. derin sinir ağlarının ara katman aktivasyonlarını analiz etmek için bir gizli uzay keşif yöntemi tasarlanmıştır. Derin sinir ağlarının hasmane ataklara karşı dayanıklılığı, bu analizden elde edilen gözlemler temelinde incelenmiştir. Son olarak, ara katman aktivasyonları ve dağılım dışı örnekleri kullanarak bir dağılım dışı örnek tespiti yaklaşımı önerilmiştir. Önerilen tespit yöntemi, dağılım içi veri kümesi üzerinde bir dizi prototip vektörünü eğitir. Bu prototip vektörleri, dağılım içi ve dışı örnek aktivasyonlarından özellikler çıkarmak ve bu örnekleri ayırt etmek için çok katmanlı bir algılayıcı eğitmek için kullanılır. Birçok benchmark veri kümesi üzerinde yapılan deneyler, önerilen dağılım dışı örnek tespit yönteminin istenilen seviyede performans sergileyebildiğini göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
1. INTRODUCTION . . . 1
1.1. Thesis Contributions . . . 4
2. RELATED WORK . . . 6
2.1. Adversarial Robustness . . . 7
2.1.1. Adversarial Training . . . 8
2.2. Out-of-Distribution Detection . . . 8
2.3. Out-of-Distribution Detection Methods . . . 9
2.3.1. Internal Methods . . . 10
2.3.2. External Methods . . . 12
3. METHODOLOGY . . . 14
3.1. Latent Space Analysis . . . 15
3.1.1. Latent Space Analysis by Likelihood Values . . . 15
3.1.2. Likelihood Value Calculation . . . 15
3.1.3. Latent Space Subsets . . . 16
3.1.4. Likelihood Value . . . 17
3.1.5. Density Estimation . . . 18
3.2. Adversarial Robustness . . . 18
3.2.1. Cluster Assignment . . . 19
3.2.2. Activation Histogram Regularization . . . 21
3.3. Out-of-Distribution Detection . . . 23
3.3.1. Prototype Learning . . . 24
3.3.2. OOD Classifier . . . 27
4. EXPERIMENTS . . . 29

4.1. Datasets . . . 29
4.1.1. OpenOOD Benchmark . . . 29
4.1.2. OOD Detection ID Datasets . . . 30
4.1.3. OOD Detection Training Datasets . . . 30
4.1.4. OOD Detection Test Datasets . . . 31
4.1.5. OOD Detection Validation Set . . . 31
4.1.6. Benchmark Setup . . . 32
4.2. Models . . . 32
4.3. Latent Space Analysis . . . 33
4.4. Adversarial Robustness . . . 34
4.4.1. Experiment Settings . . . 34
4.4.2. Metrics . . . 34
4.4.3. Experiment Results . . . 35
4.4.3.1. Investigation of Perturbation Effects on Intermediate Layers . . . 35
4.4.3.2. Investigation of Correlations Between Weights . . . 35
4.5. Out-of-Distribution Detection . . . 36
4.5.1. Baseline Studies . . . 36
4.5.2. Experiment Settings . . . 38
4.5.2.1. Benchmark Process . . . 38
4.5.2.2. Prototype Feature Generation . . . 39
4.5.2.3. Selected Layers . . . 39
4.5.2.4. OOD Classifier . . . 40
4.5.2.5. Hyperparameters . . . 40
4.5.2.6. Prototype Count . . . 40
4.5.2.7. Validation . . . 40
4.5.2.8. Augmentations . . . 42
4.5.2.9. Computational Cost . . . 42
4.5.2.10. Prototype Contributions . . . 42
5. CONCLUSION . . . 43
REFERENCES . . . 45

LIST OF FIGURES

Figure 3.1. Comparison of cluster-class alignment between a standard (left) and robust model (right). During this analysis, a cluster from a class is compared with the overall distributions of other classes. Dashed lines indicate the range of the cluster. The standard model has arbitrary intersections. Also, activations of classes have aligned extreme values in the robust model. 19

Figure 3.2. The target histogram in the second experiment. Activation distribution on each coordinate of the activation map is optimized to have this distribution. 21

Figure 3.3. The proposed OOD detection method consists of two main components: prototype learning modules and an OOD classifier. The OOD classifier is trained using ID and OOD data, while the prototypes are trained only with the ID data. Several layers are selected for feature calculation. A separate prototype learning module is created for each selected layer. 23

Figure 4.1. OOD detection performance comparison of various layers. It is shown that the layers closer to the penultimate layer have the most significant contribution. Solid lines indicate the performance of individual layers. Our method uses convolutional layers from block 1,2,3 as well as the penultimate and logit layers. The dashed lines indicate the results with the selected layers. 38

Figure 4.2. Relationship of OOD detection performance with the OOD dataset size. Left figure is the AUROC metric and the right figure is the FPR95. We show that the number of required examples before a plateau depends on the dataset's complexity. For example, CIFAR100 requires a larger OOD dataset than CIFAR10. 41

LIST OF TABLES

Table 4.1. Comparison of our proposed method with selected methods. We present the results that are averaged over three separate runs. Near-OOD and Far-OOD are the average metrics for their respective datasets. 37

Table 4.2. Ablation Study Metrics 41

1. INTRODUCTION

In recent years, deep neural networks (DNN) have significantly improved the state-of-the-art performance in many tasks, such as computer vision and natural language processing. Although DNNs have been shown to be powerful, our comprehension of their inner dynamics still needs to be explored. For instance, it is still not clear how the effects of the adversarial perturbations and out-of-distribution (OOD) samples are propagated through intermediate layers. DNNs are known to be susceptible to OOD and adversarial samples since they are highly parametrized and prone to overfitting [1]. Such samples can be misclassified with high confidence by the state-of-the-art networks [2, 3]. Therefore, it is essential to investigate the dynamics of intermediate layer activations that might enable us to improve the reliability of the DNNs by detecting the problematic samples or introducing robustness.

In one of the earlier studies on adversarial examples, Szegedy *et al.* [4] explained that adversarial examples exist due to the decision boundaries being very close to the examples in the input space. This property of DNNs is closely related to their inner mechanics. For example, Montúfar *et al.* [5] discussed the advanced learning capacity of DNNs through paper folds. In their analogy, each nonlinear function is similar to a paper fold, and a fold in deeper layers is applied to all earlier layers, which could be the reason behind adversarial examples. Although adversarial samples look perceptually similar to the natural samples in the input space, their effects in the intermediate layers are similar to out-of-distribution (OOD) samples. DNNs are trained with the assumption that there will not be a significant difference between the training and test data distribution. However, data obtained from a real-world setting might have OOD instances unfamiliar to the trained DNN. One coping mechanism for the existence of OOD samples is to recognize them before feeding them into the network.

OOD detection tackles this problem by introducing additional OOD detection modules in the network. The current state-of-the-art OOD techniques mainly utilize

only the penultimate layer. Besides, the research focusing on the intermediate layers often relies on the statistical properties of the feature maps. For instance, Park *et al.* [6] computed the feature map norms and detected OOD samples based on the deviation from these norms. Also, Wan *et al.* [7] employed a density estimation technique based on the training set and assessed the deviations from the training distribution for OOD detection. In addition, characteristics of the intermediate layer activations of OOD examples are yet to be clearly understood. For these reasons, this thesis focuses on first understanding the intermediate layer characteristics of DNNs and then, from the insights, developing new methods to enhance the adversarial robustness and OOD detection capabilities of DNNs.

Investigating the intermediate layer activations is challenging for three primary reasons. Firstly, DNNs eliminate the information that is not directly relevant to the training task as the information travels throughout the layers, concentrating the semantic information closely related to the training task. Thus, the earlier layers contain information that is not directly related to the task, posing a challenge in filtering out the noise. Secondly, the intermediate layer activations in convolutional models are 3-dimensional tensors containing information from different input image regions. As a result, the information structure in feature maps is highly dependent on the input image orientation, which poses a significant challenge to exploration. Lastly, the intermediate layers are generally not directly optimized to make each class separable. Therefore, the activations from different classes are challenging to cluster or compare. For these reasons, making observations on the global structure of the activation maps may not be possible. Instead, the literature mainly focuses on two methods. The first is creating visualizations in the input space to explore the most related input to a feature [8]. The second method of investigating the intermediate layer activations is through the statistical properties of the feature maps [6,7,9–11]. Some studies create manifolds of the activations in the forms of curves and hyperspheres. However, these studies focus on the penultimate layer activations.

This thesis focuses on analyzing the latent spaces of intermediate layers of a convolutional neural network (CNN) trained for object recognition tasks and leveraging the insights gained from the analyses to design an OOD detection framework. For this purpose, we investigate the distributions of the latent representation of the samples obtained from a network trained with natural and adversarial samples. This analysis aims to explain the structure of the decision boundary by approximating where each example lies with respect to the empirical decision boundary. Thus, we could gain an understanding of how various design choices affect the structure of the decision boundary in all layers of a neural network and how adversarial examples and adversarial training work. Observations from this analysis show that a more structured intermediate feature space is the primary source of robustness. By leveraging the observations of the analysis, we aim to improve the out-of-distribution detection capabilities of DNNs by utilizing the intermediate layer activations. By definition, OOD samples have a distribution that deviates from the training set. For this reason, they can prevent a deep-learning model from making accurate predictions.

The OOD detection task aims to classify examples based on their proximity to the training dataset distribution, called the in-distribution (ID) dataset. The OOD detection is closely related to the anomaly, novelty detection, and adversarial example problems [1, 12, 13], all of which are caused by the vulnerabilities of DNNs to various distribution shifts, such as covariate and semantic shifts. These problems investigate the different aspects of being outside the training distribution. However, they all commonly focus on the image classification task with CNNs. Therefore, this thesis also considers designing an OOD detection framework for CNN architectures. OOD detection approaches for DNNs can be categorized into internal and external approaches [1]. Internal approaches aim to enhance the OOD detection performance by fine-tuning the backbone model with various techniques, such as augmentations, regularizations, or using OOD data in training [13–18]. On the other hand, the external approaches are concerned with creating additional external modules that classify the output features from the backbone model. Several external methods also aim to fine-tune the backbone model.

The main goal of these methods is to improve the separability of the OOD samples from the ID counterparts in logit, penultimate, or intermediate layers [12, 19–23]. Internal approaches focus on enhancing the backbone model to produce features that make the ID distribution more separable from others, while external approaches investigate how the ID and OOD data can be separated. However, several methods do not fit into this categorization. For example, they can fine-tune the backbone while also making OOD detection. Current OOD detection methods commonly utilize only the penultimate layer, and an approach using intermediate layer activations has yet to yield promising results. Furthermore, some of the current OOD detection studies do not utilize any OOD data to generalize unseen cases. However, incorporating OOD data into the detection process can also have benefits. Essentially, using OOD data allows the detector to be tailored to a desired definition of being OOD through modifications to the training dataset.

1.1. Thesis Contributions

This thesis investigates how the insights from intermediate layer activations can be leveraged to enhance certain capabilities of DNNs, such as adversarial robustness and out-of-distribution detection. An OOD detection framework that incorporates OOD samples is proposed. To prevent overfitting to the OOD data, a set of prototypes is trained on the ID data to learn its characteristics that allow distance-based feature extraction. Then, a binary classifier model is trained on prototype distance features extracted from ID and OOD datasets. Our study [24], accepted for publication, is the basis for the OOD detection part of this thesis. The figures produced during this thesis work have been used in the thesis book in accordance with the publisher's "publication policy on the reuse of writings and graphics created by the author," as stated on the publisher's website. Contributions of this thesis are outlined below:

- Investigating the distribution of the intermediate layer activations in the feature space. Analysis in this stage involves the exploration of statistical properties and designing various approximation methods to extract generalizations.

- Exploring how differently an adversarially robust model processes information throughout its layers compared to a natural model. The effect of adversarial and OOD examples on the intermediate activations was investigated.
- A two-stage OOD detection framework was proposed. In the first stage, the prototype learning module is trained using only the ID data. Prototypes are trained on several intermediate and penultimate layers in an unsupervised setting. In the second stage, prototype distance features are calculated from both ID and OOD datasets, and a binary classification model is trained.
- The proposed OOD detection method allows using any backbone model without a significant overhead. Using prototypes trained on ID data reduces the chance of overfitting OOD data.
- Evaluation of the proposed method is based on the commonly used benchmarks and datasets in the OOD detection problem. The qualitative and quantitative results show that the proposed OOD detection method is on par with the state-of-the-art OOD detectors on CIFAR10 and CIFAR100 datasets on the OpenOOD benchmark.

2. RELATED WORK

The latent space of a DNN contains the encoded information from the input in the form of feature maps and embeddings. The output of the last fully-connected (FC) layer, or the penultimate layer, is commonly called the latent space. Earlier layer activations are called the intermediate layer feature maps, which contain lower-level information, while the penultimate layer activations have the information that is most related to the training task.

Improving our understanding of the latent space structure can allow us to enhance DNNs in various aspects. Examples of these are adversarial robustness, which aims to improve the robustness against adversarial examples; NN explainability, which focuses on attributing the model's decisions to a set of rules; and out-of-distribution detection, which aims to detect examples that don't belong in the training dataset distribution.

One of the earlier studies explaining how DNNs process the input data is introduced by Montúfar *et al.* [5], who analyze the model capacity through the number of linear piecewise functions required to approximate the decision boundary in the input space. They also illustrate how non-linear activations functions transform the input space with an analogy of folding a 2D paper. In addition, they explain that the paper folds in deeper layers are applied to multiple partitionings in the input space. Pascanu *et al.* [25] explain why deeper models can have more capacity compared to shallow networks. Briefly, they attribute the higher capacity in deeper models to earlier layers forming primitive functions and deeper layers reusing them. To support their claim, they show that a deeper model can form a much higher number of linear regions in the input space.

Many adversarial robustness methods are inspired by studies that investigate the relationship between training examples and decision boundaries. However, most of the observations are made on the input space. In one of the earliest studies on

adversarial robustness, Madry *et al.* [26] explain why a higher network capacity can enhance adversarial robustness. In their illustration of the input space, they claim that adding a margin around examples requires a more complex decision boundary. Out-of-distribution detection is also a task that benefits from latent space analysis. For example, Sun*et al.* [27] analyzed the structure of the penultimate layer activations and observed that the ID and OOD data did not have smooth boundaries. Inspired by this, the authors used a K-Nearest Neighbors-based approach to improve OOD detection. Studies investigating the geometry of feature maps are less common compared to the penultimate layer. An example is Meegahapola*et al.* [28], who explain the decision boundaries of classes through Prior Activation Distributions (PADs). In their method, the authors use the statistical properties of the hidden layer activations by calculating KL-Divergence scores to ID data distributions.

2.1. Adversarial Robustness

Deep learning models have proven themselves to have state-of-the-art capabilities in many fields, including computer vision, natural language processing, and reinforcement learning. Deep learning models have also been observed to have vulnerabilities to small perturbations in the input space, which can flip the output of the model. Besides, it has been discovered that we can use perturbations with magnitudes smaller than humans can notice and still successfully flip the outcome.

One of the earliest research on adversarial examples was carried out by Szegedy *et al.* [4] in 2013. After that, researchers discovered many other methods to generate adversarial examples. Various methods were also developed to enhance the robustness of neural networks towards these attacks. The current most successful defense methods generally involve adversarial training, which consists of generating adversarial examples and then using these perturbed inputs with the same labels. Other methods build upon adversarial training with various regularization techniques, improving robustness against adversarial examples.

Generating an adversarial example is a constrained optimization task, with constraint being the maximum size of the perturbations. The objective is to maximize the model output corresponding to a misclassification. The size of the perturbations is generally measured in the L-inf norm or the L2 norm. While the L2 norm is the Euclidean norm of a perturbation, the L-inf norm measures the largest difference in any pixel. Projected Gradient Descent (PGD) [26] is one of the well-known adversarial attacks. PGD relies on calculating a gradient of the input space around the input example, generating a perturbation, and then projecting the resulting perturbation onto the feasible region. A feasible region in this method is the space around the input example that satisfies the perturbation limit. The L2 norm limit will be a hypersphere; with the L_∞ norm limit, it will be a hypercube.

2.1.1. Adversarial Training

Adversarial training is training a robust model by providing a perturbed version of the original dataset, where the labels are unchanged. In the theory of adversarial training, this process is depicted as an outer minimization problem, where the inner optimization is a maximization of an output other than the actual class [26]. An adversarial attack carries out this first optimization. Then, the second part, outer minimization, is carried out by training the model with this perturbed example but using the original label. We first run an adversarial attack to maximize the loss and generate a perturbed example. After that, we train our model on the perturbed example with the original label, minimizing our loss. The main goal of this process is to guarantee that an L2 or L_∞ norm space around all training examples will be correctly classified.

2.2. Out-of-Distribution Detection

One shortcoming of deep learning models that limits their use in high-stakes applications is the possibility of critical decision mistakes that can have catastrophic consequences. Such applications often have strict reliability requirements limiting deviation from the ideal predictions. However, with deep learning models, it is often

impossible to guarantee that the model will not make a critical mistake. For example, it is unacceptable for a self-driving car to run a red light. With the current state of deep learning models, researchers can gather a large dataset for testing. However, even with enormous datasets, covering every possible scenario is often impossible, resulting in untested out-of-distribution examples. A possible solution to this issue is recognizing when the network encounters an out-of-distribution case. The predictions on such unseen cases can then be adequately managed to minimize risks.

OOD detection methods focus on a wide variety of applications. For example, OOD detection systems are incorporated into autonomous driving systems to detect unknown objects and take appropriate action [29]. In addition, OOD detection is also used in diagnosis from medical images and manufacturing to find defects in materials [30].

2.3. Out-of-Distribution Detection Methods

OOD detection is comprised of two main aspects. The first one is the detection input, which is generated by the model to be used in OOD detection. In most studies, the sources of the detection input are the activations from the logit layer and the penultimate layer. Other less commonly used sources are the gradients or differences caused by various perturbations. Studies using earlier layers also exist. A primary distinction of our method is the incorporation of earlier layers.

Out-of-distribution detection techniques can be grouped into two categories. The first one is the internal methods, which modify the target network to improve the separability of the OOD examples from the ID ones. The second one is the external methods, which analyze the activations and other patterns in the network to classify the examples without affecting the target network. A significant number of internal and external methods improve independent aspects of OOD detection. Therefore, an external method can be used in conjunction with an internal method if the external method does not involve fine-tuning the target model.

Every OOD detection method requires calculating an OOD score. The quality of this score depends on two main aspects. The first is the capability of the OOD detection module to calculate the score. The second one is the inherent separability and the noise detection module input. The quality of this input data depends on the examples' inherent characteristics and the target model's generalization capabilities. Generally, external methods aim to create a better OOD detection module, improving the first aspect.

In contrast, internal methods aim to improve the target model to produce a better input to the detection module. However, a significant number of hybrid approaches also exist. Our method is an external approach; therefore, it can be used with an internal method to improve detection quality.

2.3.1. Internal Methods

The main distinction of the internal methods is fine-tuning the backbone to improve the separability of OOD and ID examples in various layers. One of the first instances of this approach is Outlier Exposure. In this study, Hendrycks *et al.* [13] fine-tuned the backbone model while mixing OOD examples into the training set and optimizing the backbone to produce low softmax predictions on OOD examples. Some internal methods fine-tune the backbone while not introducing a detector module. To compare them, a baseline detection method proposed by Hendrycks and Gimpel [14] is used. This method uses the maximum softmax score as the OOD prediction score. When the backbone is trained, conserving the performance on the ID examples is crucial. However, several internal approaches degrade the performance on ID data. Such methods often incorporate synthetic data into the training dataset [31–33]. For example, Du *et al.* [31] fine-tuned the backbone with virtual outliers, causing a decline in the metrics on the ID dataset.

Approaches that avoid generating synthetic OOD data involve self-supervised training, data augmentation techniques, and regularization. For instance, DeVries

and Taylor [15] employed the CutOut augmentation method for regularization. Also, Hendrycks *et al.* introduced the RotPred method as an augmentation-based approach [34] to improve the model robustness and in turn, enhance the adversarial robustness and OOD detection. In another study, Tack *et al.* [35] created the distributionally shifted versions of the training examples and compared them to the originals in a contrastive learning setting. Winkens *et al.* [36] also followed a contrastive learning approach with a class-based density estimation method.

Various regularization approaches are proposed for the OOD detection task in the literature. For instance, Ruff *et al.* [37] posed a one-class classification strategy in conjunction with regularizing the output data distribution to form a small hypersphere. Meanwhile, Wei et al. [38] regularized the logit norms since their vector norms are observed to enlarge as the network gets overconfident. Similarly, Sun *et al.* [2] rectified the activations to reduce the overconfidence of the networks for OOD examples. Song *et al.* [39, 40] modified the activation maps by removing the rank-1 feature associated with the singular vector corresponding to the largest singular value of the feature map. There are also studies focusing directly on the activation maps and weights. Sun and Li [16] pruned the weights that undermine the OOD detection while Djurisic *et al.* [17] sparsified the activations. Ming *et al.* [18] projected the penultimate layer activations on a hypersphere and regularized the activations to be concentrated around the mean values of classes.

Examples of studies using generated examples include Du et al. [31], and Tao et al. [32], who generate synthetic OOD samples using density estimation and learn OOD detection utilizing the synthetic data; Wang et al. [33], who boost the OOD detection capabilities of the target model by generating informative OOD examples in the hidden layers through perturbations in the model weights.

2.3.2. External Methods

External methods keep the backbone model intact. This is achieved by using only a detector module. External methods incorporate the softmax scores, activations from various layers, and the gradients into their decision process. While the internal methods also require a detection module, the difference from the external approaches is that they also fine-tune the backbone.

A commonly used method is proposed by Hendrycks and Gimpel [14]. In this method, the maximum softmax score is used as the prediction score for OOD detection. In another study that utilizes softmax scores, Liang *et al.* [22] use the score change in the softmax scores caused by perturbations on the input.

Gradient-based approaches analyze various characteristics of the gradients during the inference and training stage. For example, Lee and AlRegib [20] measured the L2 norms of the gradients for OOD detection. The main idea behind their method is that perturbations to the OOD examples would cause a minor change in the softmax activations compared to ID examples. Huang *et al.* [41] calculated the gradients of the KL divergence of the softmax scores from the uniform distribution. Then, they compared the gradients produced by the ID and OOD examples and observed that they were more significant for ID examples. As an extension to this method, Hsu *et al.* [19] also computed OOD confidence scores from the logits by adding another NN branch to the penultimate layer. Also, they employed various intermediate layers, similar to other studies in literature [39,40]. Another study focusing on the logit difference is from Liang *et al.* [22]. In their method, the authors applied perturbations to the inputs and measured the logit difference for OOD classification.

Instead of averaging the feature map similarly to global pooling, the proposed approach calculates distance features for each spatial dimension. The distances are the cosine and Euclidean distances to the prototypes. The prototype approach is similar to the memory bank method adopted by several external methods. For example, Roth

et al. [42] proposed locally aware features to calculate local OOD scores, which are used for anomaly detection. Also, Zhang *et al.* [43] introduced a method first to memorize patterns using Hopfield networks, then calculate Hopfield energy scores from the patterns for OOD classification. This thesis's methodology involves training a set of prototypes to represent the ID data characteristics in the intermediate layers. Then, an OOD classifier model is trained on the distance features calculated from the ID and OOD datasets. The main difference of our proposed method is calculating separate distance features for each spatial coordinate. However, most related studies calculate the OOD scores from the global average of the feature maps.

In a following study, Lee *et al.* [12] use density estimation on the penultimate layer activations of the ID examples. Their method assumes that the ID data follows a class-conditional Gaussian distribution, while the OOD data does not. Motivated by this assumption, they calculate a Mahalanobis distance to each ID class and use the minimum distance for OOD detection. In order to calculate a Mahalanobis distance, firstly, the means and the covariances of each class are calculated on ID data. During the inference phase, a Mahalanobis distance is calculated for each class. Then, a threshold is applied to the minimum distance. Contrary to Hendrycks *et al.* [13], they calculate the distances on the penultimate layer.

We decided to exclude several methods focusing on intermediate layers from our comparison, such as [44, 45], because they were not included in the OpenOOD benchmark [46,47]. The main reason is the significant difference between the reported metrics and the OpenOOD benchmark metrics from several previous studies. This shows us that creating a reliable test setting in OOD research is challenging.

3. METHODOLOGY

This chapter presents the methodology used in this thesis, which consists of three stages. The first stage is the latent analysis. In this stage, the structure of the intermediate layer activations is explored. This exploration aims to observe how examples are positioned with respect to the decision boundary and then introduce generalizations to facilitate designing better adversarial robustness and OOD detection methods.

The second stage is adversarial robustness. In this stage, after an initial exploration phase focused on adversarial robustness, several experiments were carried out to improve the adversarial robustness of DNNs. The third stage is out-of-distribution(OOD) detection. In this stage, various experiments were conducted to enhance the OOD detection performance of DNNs after an initial exploration phase. Also, a novel OOD detection method that is on par with the state-of-the-art methods is proposed.

Although Adversarial robustness and OOD detection focus on different properties of DNNs, these topics have similar aspects in this thesis. Firstly, both stages focus on the intermediate layer activations. Also, each stage investigates how the characteristics of the activations differ from the training dataset.

Analysis for adversarial robustness investigates how adversarial examples can manipulate intermediate activations and how a robust model can filter out the perturbations. Afterward, adversarial robustness experiments regularize the weights or activations during a fine-tuning stage.

In the OOD detection stage, we first present our analysis outcomes, and then we propose a novel OOD detection method. The proposed OOD detection method is inspired by our findings in the previous two stages. In the analysis stage, we developed a latent space analysis tool. Then, using this tool, in the adversarial robustness stage, we observed that separate regions of the intermediate layer activations can have different

contributions to the prediction. This observation inspired our proposed OOD detection method, which uses prototypes to represent a part of the 3D feature maps and calculate separate distances for each spatial dimension.

3.1. Latent Space Analysis

3.1.1. Latent Space Analysis by Likelihood Values

Explaining high-dimensional data usually requires a processing step that generates the target information. The reason for that is that, as human beings, we cannot view the whole dataset with all dimensions at once and make a deduction about the properties of the data. Unless there is a simple relation between the dimensions, we either need to reduce the dimensionality, analyze different dimensions separately, or build an overall understanding of the data. In other words, preserving the global structure of a high-dimensional data during an analysis is challenging. As an alternative this thesis focuses on analyzing the local region around the samples.

3.1.2. Likelihood Value Calculation

The main goal of this stage is to explore how robustness is related to the local region around the examples and develop new methods of adversarial robustness and OOD detection. To explore the latent spaces, this thesis considers the local region around the examples. This is acheived by a customized likelihood calculation method based on density estimation. However, to deal with the high dimensionality issue, the proposed likelihood function calculates separate likelihoods for each dimension. Also, this likelihood is calculated from the examples close to the target sample.

The likelihood value indicates the closeness of a particular example to the other classes in all dimensions of the latent space. The simplest method to calculate this is to keep other examples around our particular example and estimate the density of different classes along each dimension. However, since latent spaces often have

high dimensionality, this simple approach is not possible because, in most cases, this hypercube contains examples from only a single class or only our particular example.

For this reason, our approach in this work is to slice the latent space around an example only along a specific dimension. This dimension maximizes the separability of our example. However, since we are calculating a likelihood value for each dimension, this process must be repeated for all dimensions. For instance, consider an example in a 3-D latent space. We would first calculate the likelihood for dimension A. For this calculation, we will slice the latent space around dimensions B and C, then accept B and C based on how much they increase the likelihood value along A. Then, this maximum value is the likelihood value of A for this example. This process is repeated for dimensions B and C.

3.1.3. Latent Space Subsets

A subset of the dataset $\mathbf{X} = \{\mathbf{x}_i\}$ is represented by $\overline{\mathbf{X}_{e,d}}$. This subset is created based on the dimension d of the example e, where the examples in the subset have percentile values along dimension d within a range of the percentile value of the slicing example e. Dataset subsets can be represented as

$$\overline{\mathbf{X}_{e,d}} = \{x_i : p_{i,d} \in [p_{e,d} - r, p_{e,d} + r]\} \tag{3.1}$$

$$\overline{\mathbf{l}_{e,d}} = \{l_i : p_{i,d} \in [p_{e,d} - r, p_{e,d} + r]\} \tag{3.2}$$

where $p_{i,d}$ is the percentile value of the example i along the dimension d and r is a specified percentile range. $\overline{\mathbf{l}_{e,d}}$ is the corresponding subset of the targets. The dimension along which we will calculate the likelihood is

$$\overline{\mathbf{x}_{e,d_t}} = \{\overline{\mathbf{X}_{e,d_i}} : i = t\} \tag{3.3}$$

which is the 1-D vector from the dimension t of the dataset subset.

3.1.4. Likelihood Value

Then, the likelihood value for an example along dimension t is calculated by

$$L_{e,t} = \max_{d} \lambda(\overline{\mathbf{x}_{e,d_t}}, \overline{\mathbf{l}_{e,d}}, x_{e,t}, l_e) \tag{3.4}$$

where $L_{e,t}$ is the likelihood ratio for the example e along dimension t. The first parameter is the data on which we create the KDE representation, and $X_{e,t}$ is the value of the example e along the dimension t. Each KDE is calculated on the dimension t. However, the dataset is sliced along another dimension d in each step. In other words, slicing along each different d produces a different subset along t, and then a likelihood value for the example is found on these subsets. Finally, the resulting likelihood is the largest of these values. λ is a function that calculates a likelihood ratio for a given dataset and an example, along with their targets. It uses a one-versus-rest approach where we estimate density for the example's class l_e, then make another density estimation for the rest of the data, accepting them as the 'other' class. The result of the function is the ratio of the samples from these two estimations based on the example's value. Note that since we use this function on each dimension separately, the value of the example is a scalar. Calculation in the λ function can be represented as

$$\lambda(\overline{\mathbf{x}_{e,d_t}}, \overline{\mathbf{l}_{e,d}}, x_{e,t}, l_e) = \sigma\left(\frac{\gamma(\mathbf{c}_{e,d_t}, x_{e,t})}{\gamma(\mathbf{c\prime}_{e,d_t}, x_{e,t}) + \epsilon} - 1.0\right) \tag{3.5}$$

where $\mathbf{c}_{e,d}$ is the subset of $\overline{\mathbf{x}_{e,d_t}}$ with examples that belong to the target class of the slicing example, l_e. Moreover, $\mathbf{c\prime}_{e,d}$ are the examples from other classes. ϵ is a small value added for numerical stability. σ is the sigmoid function, where $\sigma(x) = \frac{1}{1+e^{-x}}$. Two subsets for the example's class and other classes are created by

$$\mathbf{c}_{e,d_t} = \{x_{i,t} : x_{i,t} \in \overline{\mathbf{x}_{e,d_t}} \wedge l_i = l_e\} \qquad \text{Example's Class} \tag{3.6}$$

$$\mathbf{c\prime}_{e,d_t} = \{x_{i,t} : x_{i,t} \in \overline{\mathbf{x}_{e,d_t}} \wedge l_i \neq l_e\}. \qquad \text{Other Classes} \tag{3.7}$$

3.1.5. Density Estimation

Function γ calculates a kernel density estimation value with a Gaussian kernel, which can be represented by

$$\gamma(\mathbf{x}, x_e) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h\sqrt{2\pi}} \exp^{-0.5(\frac{x_n - x_i}{h})^2} \tag{3.8}$$

where $\mathbf{x} = \{x_1, x_2, x_3, ..., x_N\}$.

3.2. Adversarial Robustness

This section focuses on improving adversarial robustness with methods inspired by analysis of how adversarially robust models differ from the standard ones. Two experiments are presented, designed from the exploration stage outcomes that compare robust models with standard ones.

In one of earlier studies on adversarial examples, Szegedy *et al.* [4] describe adversarial examples as an input image with a specifically crafted noise added to it. This noise is referred to as a perturbation, crafted by an adversarial example method to manipulate the model's output prediction. When the attacker has access to the model weights, adversarial examples can be generated to manipulate the model to produce a highly confident prediction for any class. Also, the authors attribute this weakness of DNNs to the local linearity of DNNs; they support their claim by stating that other linear models also have this weakness.

In this section, a model subjected to adversarial training is called a robust model. Before each experiment, a robust model is compared to a standard one to explore a robustness-related property. In the first experiment, the alignment of activation clusters with other classes is compared, while in the second one, the overall activation distribution is compared. A promising property of the robust model is selected in this exploration process. Then, in the experimentation stage, a regularization-based

method is designed to fine-tune the standard model to induce the previously observed property. Both experiments focus on fine-tuning the pre-trained standard model with regularization loss function to enhance its robustness. In addition, both experiments focus on intermediate-layer activations.

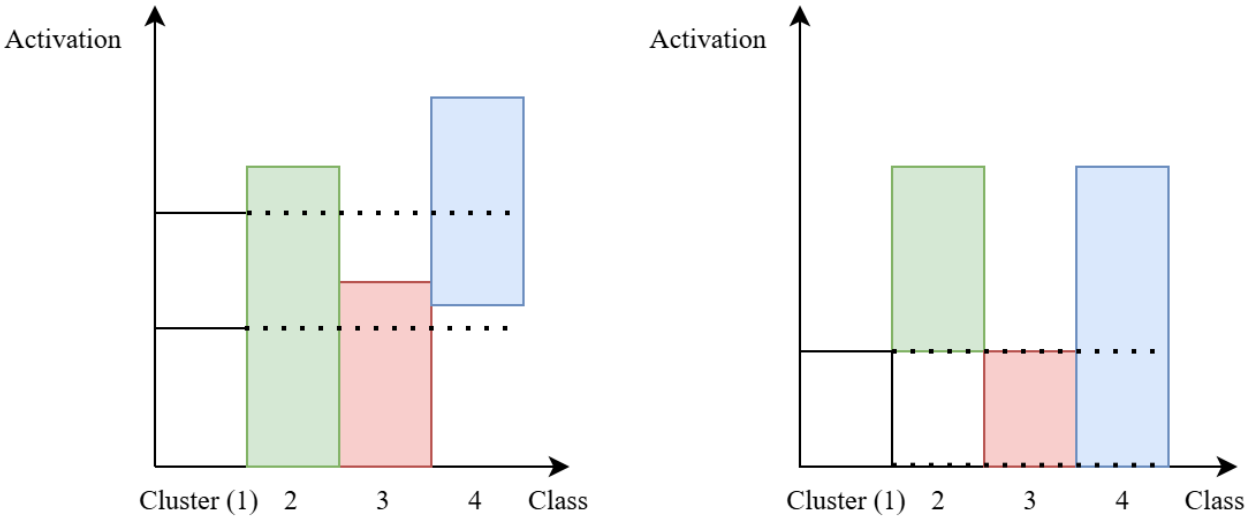

Figure 3.1. Comparison of cluster-class alignment between a standard (left) and robust model (right). During this analysis, a cluster from a class is compared with the overall distributions of other classes. Dashed lines indicate the range of the cluster. The standard model has arbitrary intersections. Also, activations of classes have aligned extreme values in the robust model.

3.2.1. Cluster Assignment

In our investigation focusing on adversarial robustness, we created clusters from the activations using the analysis tool developed in the previous stage. In addition, the clusters are created from the examples in the same class. Then, the range of each cluster is compared with other classes. As a result of this comparison, we observed that the clusters from the robust model had very high intersections with other classes or did not intersect. However, low intersection levels were typical for the standard model. This observation is illustrated in Figure 3.1.

Inspired by this, we hypothesize that a robust model avoids low levels of intersections in most dimensions. In other words, a robust model makes some dimensions completely separable while making others completely non-separable. For an adversarial attack to be successful, the example must move towards the distribution of another class in many dimensions. The standard model clusters are already close to the other classes in many dimensions. The robust model clusters are distant from other classes in some dimensions while fully intersecting in others. This can enhance robustness for two reasons. Firstly, when a cluster is already fully intersecting with other classes, the adversarial attack cannot gain from an attack targeted to this activation because it is already inside the distribution of other classes. Second, the non-intersecting clusters are more challenging to manipulate because they are more distant from other classes.

Inspired by this investigation, we designed a loss function to induce the same characteristics during the fine-tuning of the standard model. The loss function is based on calculating a desired shift for a cluster and then regularizing the activations for that class to shift them. The proposed loss function can be represented as

$$\mathcal{L} = \frac{1}{N} \sum_{i,j,k} \left(f_{i,j,k} - t_{i,j,k} \right)^2, \quad \text{where } t_{i,j,k} = f_{i,j,k} + s_{i,j,k}. \tag{3.9}$$

where $f_{i,j,k}$ is the activation in the specific spatial coordinates of the 3D activation map, $s_{i,j,k}$ is the calculated shift for this example and $t_{i,j,k}$ represents a target value for the activation. In each optimization iteration, we calculate a target value from the calculated shift instead of optimizing based on the shift itself because that would make the backpropagation much more complex.

To calculate the shift values, we consider the intersection of a cluster with other classes. Shift value $s_{i,j,k}$ is a cluster's total shift calculated from the intersections with all other classes. During its calculation, the intersection with all other classes is checked independently, and then a shift value is calculated. Finally, all shift values are added together to find a total shift. If the intersection of a cluster is more than 20% of the cluster range, a shift towards the class is added to the total shift. If the intersection is

less than 20%, a shift away from the class is added to the total shift. This optimization aims to induce the alignment property in the robust model through regularization.

According to our experiments, we managed to enhance the adversarial robustness of the MNIST model. However, the improvement in the CIFAR10 model was minimal.

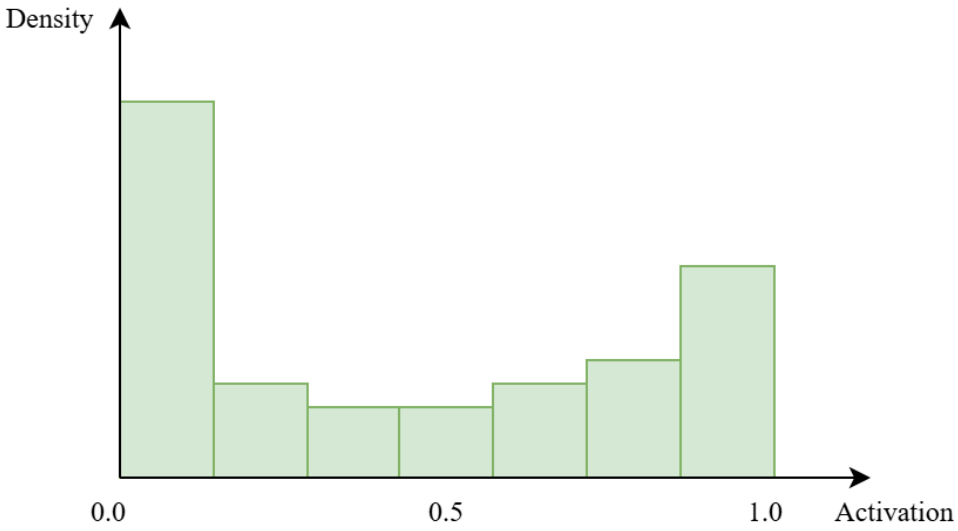

Figure 3.2. The target histogram in the second experiment. Activation distribution on each coordinate of the activation map is optimized to have this distribution.

3.2.2. Activation Histogram Regularization

Secondly, the overall activation distribution in each coordinate of the 3D activation map is investigated. For each coordinate on the feature map, a histogram is calculated from the set of activations on the dataset. The set of activations used for each histogram can be represented as

$$\{f_{i,j,k}^{(n)} \mid n = 1, 2, \ldots, N\} \tag{3.10}$$

where n is the example index and $f_{i,j,k}^{(n)}$ is the activation for a single example on a specific coordinate on the 3D activation map. A separate histogram is calculated for each coordinate.

In this investigation, activations of the batch normalization layers were considered. After comparing the robust model with the standard, we observed that activations of the robust model were concentrated on the extreme values of the activation range. Histograms of the robust model showed two peaks around the extreme values. Also, the peak on the lower end was significantly below zero, indicating that a slight change in the activation would not affect the ReLU output. In contrast, the standard model had its activations mostly around the mean value, which was mostly above zero. In addition, both MNIST and CIFAR10 models exhibit this property. However, the CIFAR10 model has several layers that do not adhere to this generalization.

Inspired by this exploration, we hypothesize that activations around the mean value are pushed toward the extremes when a model is subjected to adversarial training. The main goal can be creating a large margin from the decision boundary.

To induce this property by fine-tuning, we first defined a target histogram representing the robust model characteristics, shown in Figure 3.2. Then, we calculated separate histograms for each coordinate on each batch. Then, we calculated a KL-Divergence loss and minimized it. This loss function is applied to the activations of the convolutional layers before the batch norm layers. As a result of this experiment, we significantly increased the adversarial robustness of the MNIST model when the perturbation value was 0.1, without any impact on the natural accuracy. However, the improvement in the CIFAR10 model was minimal compared to adversarial training.

After the development of two adversarial robustness methods, we decided to apply our insigts in a more general way to OOD detection problem. This change in our approach is due to that our methods were challenging to optimize and they reduced the natural accuracy.

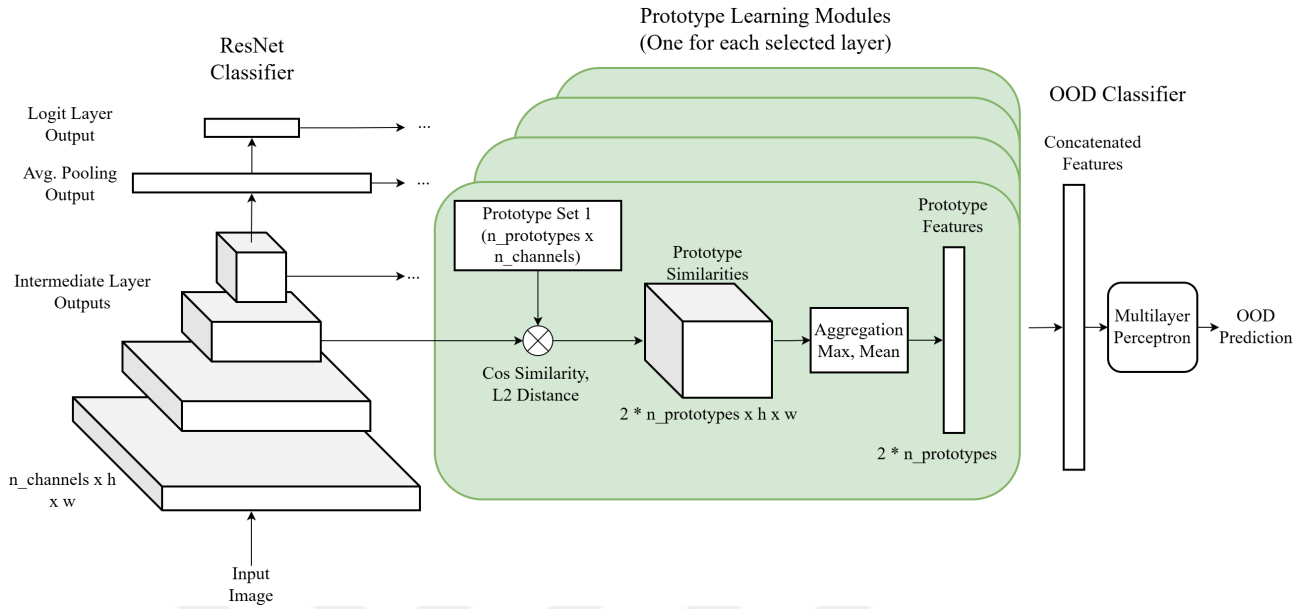

Figure 3.3. The proposed OOD detection method consists of two main components: prototype learning modules and an OOD classifier. The OOD classifier is trained using ID and OOD data, while the prototypes are trained only with the ID data. Several layers are selected for feature calculation. A separate prototype learning module is created for each selected layer.

3.3. Out-of-Distribution Detection

An out-of-distribution (OOD) example is a sample that does not belong to any of the training classes. For example, an image of an airplane will be an OOD example for a model trained on a dataset of land vehicles. In contrast, the data from the training classes is referred to as in-distribution (ID). However, according to Yang *et al.* [48], deviations from training data have two aspects: the covariate shift and the semantic shift. Covariate shift refers to the changes in pose, color, or lighting. Such changes do not change the meaning of an example. For this reason, examples from a training class with a covariate shift are still ID samples. However, semantic shifts refer to deviations from the meaning or object class training set. OOD detection aims to detect the samples with a semantic shift.

All OOD detection methods require a backbone and an OOD score calculation module. In some studies, this method is a calculation based on the activations of

softmax scores [14]. In other studies, a machine learning model uses activations or gradients as the input. Also, in some studies, the detection module is trained only on ID data, while others incorporate OOD samples into the dataset and follow a supervised learning approach. The method proposed in this thesis uses OOD data during the training of the OOD detector module. Also, several methods fine-tune the backbone while others keep it intact. For example, Hendrycks *et al.* [13] add the OOD data to the training dataset and optimize the backbone model to produce low softmax scores. The proposed method in this thesis does not fine-tune the backbone.

Our study accepted for publication [24] is the basis for the OOD detection method in this thesis. Our proposed method consists of two stages: prototype learning and OOD classifier. The first stage trains a set of prototype vectors on the ID dataset to represent the characteristics of the ID dataset. In the second stage, a set of distance features are calculated on the ID and OOD datasets. A binary classifier model consisting of fully-connected layers is trained in a supervised learning approach.

In the first stage, only ID data is used during prototype training. This aims to prevent overfitting to the OOD data in the OOD classifier training stage. Also, each selected intermediate layer, besides the penultimate layer, has separate prototype vectors, producing separate distance features. However, these features are concatenated and fed to a single binary classifier model. This allows the method to be used in any architecture. Not fine-tuning the backbone also has benefits. Firstly, it allows the proposed method to be used without access to the entire training dataset. Second, it significantly reduces the computation cost required to train the OOD classifier. However, our method can be used with models fine-tuned by other methods, further improving the detection performance.

3.3.1. Prototype Learning

Using intermediate layer activations from multiple layers for binary classification is a challenge due to the characteristics of the intermediate layer activations. Firstly,

intermediate layer activations have much higher dimensionalities than the penultimate layer. For this reason, using all activations as the input for a binary classifier will cause an overfit. Also, the location of the information in the intermediate layer activations depends on the input image orientation. In addition, before using a binary classifier model, 3D activation maps must be converted to a vector. The most straightforward approach for this would be using global pooling. However, this method incorporates unrelated background information into the feature vector, drastically reducing OOD detection performance. In order to overcome these issues, we employed a prototype approach. This approach converts the 3D activation maps to distance features through prototype vectors. Training of these prototype vectors is presented in this section.

Prototype learning aims to capture the ID data characteristics in prototype vectors, $\mathbf{p} \in \mathbb{R}^{n_{\text{channels}}}$, where n_{channels} represents the channels of the activation map. The prototype vectors are learned in a way that the activation maps can be reconstructed when some of the n_{channels}-dimensional vectors on the $n_{\text{channels}} \times h \times w$*theprototypesreplace* activation map. Thus, prototypes are trained unsupervised using only the ID samples. Therefore, the prototypes are intended to learn how to represent the ID sample distribution at the activation level. During the training process, n_{channels}-dimensional vectors at each spatial coordinate of the activation map are replaced with the most similar prototype regarding cosine similarity. The replaced vector in the activation map can be represented as

$$\mathbf{f}'_{i,j} = \arg\max_{\mathbf{p} \in \mathcal{P}} S_C\left(\mathbf{f}_{i,j}, \mathbf{p}\right) \tag{3.11}$$

where $\mathbf{f}_{i,j} \in \mathcal{R}^{n_{\text{channel}}}$ is the ith and jth vector on the activation map with $i = 1, \cdots, h$ and $j = 1, \cdots, w$, $\mathcal{P} = \{\mathbf{p}_1, \cdots, \mathbf{p}_{N_p}\}$ is the set of prototypes of size N_p, and $S_C(\cdot, \cdot)$ is the cosine similarity.

The prototypes are optimized to minimize the reconstruction loss,

$$\mathcal{L}_{\text{MSE}}\left(\mathbf{F}, \mathbf{F}'\right) = \frac{1}{N_{ID}} \sum_{n=1}^{N_{ID}} \|\mathbf{F}_n - \mathbf{F}'_n\|_2^2 \tag{3.12}$$

where $\mathbf{F}_n \in \mathcal{R}^{n_{\text{channels}} \times h \times w}$ and $\mathbf{F}'_n \in \mathcal{R}^{n_{\text{channels}} \times h \times w}$ represent the original activation map and the activation map reconstructed by the most similar prototypes of the layer, respectively. The loss function shows that prototypes are trained only on the ID data. This aims to prevent the prototype from overfitting to the OOD samples. After the prototype training stage, they are used to extract similarity features for OOD classification. During the training stage for OOD classification, prototypes are not modified.

The primary motivation behind using prototypes is that the activations produced from the OOD examples would be outliers in the feature spaces of the intermediate layers. Another intuition is that during the initial training, the backbone model learns to ignore the unrelated parts of input images. Ignoring unrelated parts is driven by blocking the information from the unrelated parts from being transferred to the next layer. In our initial analysis of the activations generated from CIFAR10, we observed that the features on the spatial dimensions that correspond to the background are more similar to a fixed background pattern. In the middle sections of the spatial dimensions, activation vectors along the channels are more unpredictable. In addition, activations produced from the OOD examples are commonly more similar to the background pattern, which is a pattern frequently generated on the regions of the activation maps that correspond to the background on the input image.

To verify the prototypes' diversity, we explore each prototype's contributions in Chapter 4. Our observations show that the trained prototypes learn to represent separate regions of feature maps. On the CIFAR10 dataset, a significant portion of the prototypes were more similar to the regions close to the edges regarding cosine similarity. However, another group of prototypes showed higher cosine similarities around the center of the layer output. Also, the similarities to the center occurred only in specific examples. Based on these observations, we hypothesize that the prototype

approach can successfully learn the characteristics of the training dataset. The second and final step of the method is OOD classifier training. This step is presented in the next section.

3.3.2. OOD Classifier

The second stage of the method consists of feature calculation with prototypes and OOD classification. In feature extraction, the distance between a prototype and all vectors across the spatial dimensions of the feature map is calculated. Then, the minimum cosine similarity and the maximum Euclidean distance are the features of a prototype. Distances are calculated as

$$d_{kij}^{E} = \|\mathbf{p}_k - \mathbf{f}_{ij}\|_2 \tag{3.13}$$

$$S_C\left(\mathbf{p}_k, \mathbf{f}_{ij}\right) = \left(\frac{\mathbf{p}_k \cdot \mathbf{f}_{ij}}{\|\mathbf{p}_k\|_2 \|\mathbf{f}_{ij}\|_2}\right) \tag{3.14}$$

where d_{kij}^{E} is the Euclidean distance and $S_C\left(\mathbf{p}_k, \mathbf{f}_{ij}\right)$ is the cosine similarity, which are calculated between the prototype vectors and the activations where $i = 1, \cdots, h$, $j = 1, \cdots, w$, and $k = 1, \cdots, N_p$. Each activation map produces multiple distances across the spatial dimensions. Then, the minimum of the cosine similarities and the maximum of the Euclidean distances from the $h \cdot w$ spatial coordinates of N_p are used as the two features for a prototype. This results in an $N_p \cdot h \cdot w \cdot 2$ dimensional feature vector for each layer, which is used for classification. Features are generated independently from each used layer and finally concatenated.

The next step is training the OOD classifier module, a fully connected 3-layer neural network trained on a binary classification task. However, since the binary classification is a supervised training, this step also requires OOD data. ID and OOD samples are fed to the backbone to create the binary classification dataset, and the distance features are calculated from the feature maps. Then the loss function is

calculated as

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{n=1}^{N} t_n \log \hat{t}_n + (1 - t_n) \log \left(1 - \hat{t}_n\right) \tag{3.15}$$

which is the cross-entropy loss used for optimization, where N is the number of training examples, t_n is the ground truth, which is a binary label, and $\hat{t}_n$ is the output of the OOD classifier module for nth sample.

During the inference stage, the feature extraction operates in the same fashion. However, the OOD classifier module's OOD score prediction is compared to a threshold. In practice, this threshold is selected based on the desired rate of false positives. In the testing stage, scores from the ID and OOD datasets are compared using AUROC, which measures the separation between ID and OOD scores. Compared to the backbone, the proposed method is lightweight, and the computation cost depends on the number of selected layers and used prototypes.

4. EXPERIMENTS

4.1. Datasets

MNIST [49] and CIFAR10 [50] are among the most commonly used datasets in adversarial robustness research. Therefore, this thesis also focused on these datasets in the analysis and adversarial robustness stages. The MNIST dataset contains 60,000 training samples and 10,000 test samples, while the CIFAR10 dataset has 50,000 for training and 10,000 for testing. The MNIST dataset contains digits from 10 classes, while the CIFAR10 has 10 classes that are commonly known objects such as airplane, bird, and cat.

An OOD detection system requires an ID dataset to train the backbone and the OOD classifier. Besides our proposed method, some studies also use an OOD dataset for training, such as Hendrycks *et al.* [13]. The required datasets for training mainly consist of an ID training dataset and an external OOD. For the test, the test part of the ID dataset, a set of ID datasets, and a set of OOD datasets are required. These ID and OOD datasets are presented in the following sections.

4.1.1. OpenOOD Benchmark

In this thesis, all pre-trained models and datasets for OOD detection were obtained from the OpenOOD benchmark. This benchmark was created by Yang *et al.* [46] in 2022 to standardize the OOD detection evaluation. This benchmark was then updated by Zhang *et al.* [47] to include several datasets that focused on OOD detection. New datasets considered examples with a covariate shift as ID and the samples with semantic shift as OOD. Yang *et al.* [48] have defined these two deviations from the training set. By their definition, a covariate shift is a change in color, pose, or lightning, while a semantic shift is a change in meaning or class. In this fashion, Yang *et al.* [3] introduced three new benchmarks using the ImageNet-1k as the ID dataset. The

main difference between these benchmarks is that various augmentations are added to test ID samples, enhancing the covariate shift on the ID samples. Also, they cleaned the dataset so that the same class did not exist in both ID and OOD datasets.

4.1.2. OOD Detection ID Datasets

The ID datasets are CIFAR10 and CIFAR100 [50], which consist of 50,000 training examples from 10 and 100 classes, respectively. CIFAR10 and CIFAR100 have mutually exclusive classes. While they have classes in different granularities, examples of no class in CIFAR10 exist in the CIFAR100. For example, on of the CIFAR10 classes is horse, while CIFAR100 has many large herbivore classes similar to a horse, but the horse is not in CIFAR100. This is an original feature of CIFAR10 and CIFAR100 datasets. ImageNet-1k [51] is the third ID dataset. Also, a subset version of ImageNet-1k is created by Zhang *et al.* [47] to make quicker OOD experiments. The number of classes was reduced to 200 from 1000 in this version named ImageNet-200. On the OpenOOD benchmark, ImageNet-1k and ImageNet-200 have separate benchmarks: the ID dataset. Another version of ImageNet is ImageNet-200-FS, which incorporates several versions of ImageNet as the ID data. These datasets are Imagenet-V2 [52], ImageNet-C [53] and ImageNet-R [54]. These OOD datasets were used to train the backbone and the OOD classifier. However, we sampled 100,000 examples from the ImageNet-200 to reduce the computational cost to train the OOD classifier.

4.1.3. OOD Detection Training Datasets

Our proposed method uses OOD datasets during the training of the OOD classifier module. In the OpenOOD benchmark, each benchmark has a set of ID datasets. Also, an OOD dataset is selected to train the OOD classifier for each benchmark. This dataset is not in the test set and is used only as a training OOD dataset, referred to as an external OOD dataset. During our experiments, we used two external OOD datasets created by Deng *et al.* [51], Tiny-ImageNet(resize) and ImageNet-1k.

In the benchmarks that had CIFAR10 and CIFAR100 as the ID dataset, Tiny-ImageNet(resize) was used as the external OOD dataset. However, the OpenOOD benchmark defines a subset of the Tiny-ImageNet(resize) as the external dataset, which consists of 29,850 samples from 596 categories.

Benchmark using ImageNet-200 as the ID dataset has a modified version of the ImageNet-1k as the external OOD dataset. This version has 800 classes because it excludes the classes from ImageNet-200. The modified ImageNet-1k was introduced by Zhang *et al.* [47]. For training the OOD classifier in this benchmark, we sampled 100,000 examples from ImageNet-800 and used this as the OOD class. The ID class consisted of 100,000 samples from ImageNet-200.

4.1.4. OOD Detection Test Datasets

On the OpenOOD benchmark, each ID dataset has a defined set of OOD datasets, separated into two categories: Near-OOD and Far-OOD. Near-OOD datasets are similar to the ID dataset and are more difficult to separate from the ID class. These datasets exhibit small semantic shifts. For example, a Near-OOD dataset for CIFAR10 is the remaining classes of CIFAR100, which have different classes but similar characteristics. However, Far-OOD datasets show significant semantic and covariate shifts. For example, a Far-OOD dataset for CIFAR10 is Textures [55], which does not consist of objects like CIFAR10 and is easier to separate than CIFAR100.

4.1.5. OOD Detection Validation Set

The validation datasets were provided by the OpenOOD benchmark [46,47]. The CIFAR10 and CIFAR100 datasets have a validation set of 972 and 1,000 examples from the TinyImageNet dataset [51]. For the ImageNet-200 benchmark, 1,000 examples from the same dataset are defined as the validation set.

4.1.6. Benchmark Setup

We used the pre-trained models in each benchmark, which had been trained on the ID datasets. The ID dataset and an external OOD dataset were used to train the OOD detection module in each benchmark, which was designated in the OpenOOD benchmark. All other OOD datasets were used for testing. Also, during the training of the prototype learning modules, only the ID dataset was used. However, the ID and external OOD datasets were used to train the binary classification module.

In the CIFAR10 [50] benchmark, the ID dataset is the CIFAR10, while the external OOD dataset is a subset of the TinyImageNet [51] dataset. The OOD datasets for this benchmark are CIFAR100 [50], other classes of the TinyImageNet, MNIST [49], SVHN [56], Textures [55] and Places365 [57].

The CIFAR100 benchmark datasets are the same as the CIFAR10 benchmark, except that the CIFAR100 is the ID and CIFAR10 is the OOD dataset.

The ID dataset in the ImageNet-200 benchmark is the ImageNet-200 [51], while the external OOD dataset is the other 800 classes from the ImageNet-1k dataset. The OOD datasets are SSB-hard [58], NINCO [59], iNaturalist [60], OpenImage-O [61]

The ID dataset in the ImageNet-200 Full Spectrum (FS) benchmark are Imagenet-V2 [52], ImageNet-C [53] and ImageNet-R [54], which are the covariate shifted versions of the ImageNet-200. The OOD and external OOD datasets are the same as the ImageNet-200 benchmark.

4.2. Models

In the Latent Space Analysis stage, we used two kinds of architectures. The model for the MNIST dataset was a simple model with two conv layers and two fully-connected layers. For the CIFAR10 dataset, we analyzed five different models from

the adversarial robustness benchmark, RobustBench [62]. We picked five models with varying robust and natural accuracies. However, all models had the WideResNet-28-10 architecture.

The adversarial Robustness stage used the same 4-layer model for the MNIST dataset. The model used in CIFAR10 experiments was a ResNet-18 model [63]. All experiments investigate the intermediate layer activations in the analysis stage and regularize in the training stage. Standard versions of the models are trained on CIFAR10 and MNIST using a cross-entropy loss. The robust versions are subjected to adversarial training on the same datasets using a TRADES loss [64].

In the OOD detection experiments, the ResNet18 [63] model was used as the backbone, and its pre-trained weights were obtained from OpenOOD benchmark [46, 47]. Each benchmark has its separate model, trained on CIFAR10, CIFAR100, and ImageNet-200.

4.3. Latent Space Analysis

Our experiments analyzed the latent spaces of various models trained on several datasets. In each analysis, we first sampled 100 examples from each class and kept their latent activations. After that, we calculated the likelihood values of each example on all layers and dimensions. In our analysis of the MNIST dataset, we also used the robustness properties of examples. We divided the examples into clusters in the likelihood space and observed that clusters closer to the decision boundary on more dimensions showed a loss of robust accuracy. We observed that the likelihood values mainly increased with robustness on both datasets. However, the results are sensitive to the KDE value of the density estimation function. For this reason, we can also implement a KDE bandwidth optimization algorithm in future work.

4.4. Adversarial Robustness

This section presents the datasets, metrics, and experiment settings used in the adversarial robustness stage of the thesis.

4.4.1. Experiment Settings

During each experiment, a standard model is first compared with its robust version, which was subjected to adversarial training. The first experiment focuses on the activation ranges, while the second investigates the relationship between the convolutional layer weights. In the analysis stage, a property of the robust model is selected as a possible cause behind the robustness. Then, the standard model is fine-tuned with a loss function inspired by the property.

Afterward, the fine-tuned model is tested using adversarial attacks to measure its adversarial robustness. During the testing phase, robust accuracy is measured by calculating the accuracy when the input examples have perturbations. For adversarial attacks, the TRADES [64] method is used. Also, the perturbations of adversarial attacks are generally limited by various norms. The norm used in this thesis is the L-inf norm. The maximum perturbation size is denoted by ϵ. Our experiments used an epsilon of 0.1 for MNIST and 3/255 for CIFAR10. However, the regular values used for this problem are 0.3 for MNIST and 8/255 for CIFAR10.

4.4.2. Metrics

The two metrics measured in the adversarial robustness study are the natural accuracy and the robust accuracy. Natural accuracy measures the regular accuracy of the test set without any perturbation. Robust accuracy measures the accuracy at which an adversarial attack modified each test sample. A robust accuracy of 0.5 indicates a 50% accuracy on examples with perturbations.

4.4.3. Experiment Results

In both experiments, we enhanced the adversarial robustness in the MNIST dataset. However, the improvements were measured only when the ϵ values were lower than the common benchmark standards. Also, improvements were insignificant in the CIFAR10 dataset compared to the adversarial training.

Our adversarial robustness experiments achieved robustness improvements only on lower epsilon settings. However, the second experiment produced a small robustness enhancement on both datasets without any significant trade-off with the natural accuracy. Still, adversarial training obtained far better results than our experiments.

4.4.3.1. Investigation of Perturbation Effects on Intermediate Layers. We also investigated how an adversarial example can cross the decision boundary despite slightly moving towards the other classes in each dimension. We observed that in a standard model, an example can be misclassified when it is shifted by a small amount across many dimensions. In other words, the effect of each dimension on the final prediction is additive. In contrast, a robust model's weights are structured to prevent this additive effect. According to our observations, this additive effect is still present when the intermediate layer activations are directly perturbed. However, the weights before that layer prevent such perturbations. This is an interesting observation, but we could not manage to induce this property through regularization without a decline in the natural accuracy.

4.4.3.2. Investigation of Correlations Between Weights. Another property of the robust model was that two channels in two successive layers produced an activation function that limited the feature space to a specific region, similar to a two-sided ReLU. These two channels also interacted with other channels, producing similar effects that limited the output regions in other channels. However, when we tried to reproduce this effect and directly introduce robustness using two-sided ReLU func-

tions, robustness did not increase, and the standard accuracy declined. Therefore, the characteristics of proportional weights and channels working together restrict the output space while preserving the standard accuracy.

In both methods, we successfully improved the adversarial robustness. However, the improvements were only in a setting with a lower epsilon value than the values commonly used in benchmarks. Also, with each incremental enhancement, the versatility of the adversarial robustness method decreased due to the increase in computational cost and difficulty in optimizing. As a result of our experiments, we deduced that a single low-level rule that explains the adversarial attacks did not exist. Rather than a single rule, the flaws in the optimization process and the dataset produced many low-level imperfections in different layers.

4.5. Out-of-Distribution Detection

This section presents the details of implementation, baseline methods, ablation studies, results, and the datasets.

4.5.1. Baseline Studies

We selected three related studies from the OpenOOD benchmark for comparison. The first is Outlier Exposure (OE) [13]. This method incorporates OOD samples into the training set and fine-tunes the backbone to produce low softmax scores for OOD samples. Since every OOD detector requires a detection module, they employ MSP [14] as the post-processor, which uses the maximum softmax score as the OOD score. However, this method can be paired with any other detection module that does not fine-tune the backbone. This method was selected because of its use of OOD data in training.

Table 4.1. Comparison of our proposed method with selected methods. We present the results that are averaged over three separate runs. Near-OOD and Far-OOD are the average metrics for their respective datasets.

ID Dataset	OOD Dataset	Methods									
		OE [13] + MSP [14]		k-NN [23]		SHE [43]		Prototype(Ours)		1D-Conv(Ours)	
		FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
CIFAR10	CIFAR100	36.71	90.54	37.64	89.73	81.00	80.31	**33.46**	**91.64**	35.43	90.83
	TinyImageNet	**2.97**	**99.11**	30.37	91.56	78.30	82.76	14.93	96.61	17.23	96.01
	Near-OOD	**19.84**	**94.82**	34.01	90.64	79.65	81.54	24.19	94.12	26.33	93.42
	MNIST	24.67	90.22	20.05	94.26	42.22	90.43	**2.81**	**99.37**	3.68	99.01
	SVHN	**1.25**	**99.60**	22.60	92.67	62.74	86.38	4.31	98.98	5.75	98.52
	Textures	**12.07**	**97.58**	24.06	93.16	84.60	81.57	12.14	97.52	13.15	97.41
	Places365	**14.53**	**96.58**	30.38	91.77	76.36	82.89	15.45	96.58	16.39	96.43
	Far-OOD	13.13	96.00	24.27	92.96	66.48	85.32	**8.68**	**98.11**	9.75	97.85
CIFAR100	CIFAR10	61.26	76.70	72.80	77.02	**60.41**	**78.15**	73.84	71.93	91.07	57.43
	TinyImageNet	**0.21**	**99.89**	49.65	83.34	57.74	79.74	41.92	88.53	54.97	85.24
	Near-OOD	**30.73**	**88.30**	61.22	80.18	59.07	78.95	57.88	80.23	73.02	71.34
	MNIST	53.31	80.68	48.58	82.36	58.78	76.76	18.89	95.07	**5.04**	**98.86**
	SVHN	51.84	84.37	51.75	84.15	59.15	80.97	34.60	89.31	**20.46**	**93.08**
	Textures	55.83	82.18	**53.56**	83.66	73.29	73.64	63.02	**83.97**	85.65	81.52
	Places365	58.30	78.39	60.70	79.43	65.24	76.30	**51.77**	84.79	67.90	**88.29**
	Far-OOD	54.82	81.41	53.65	82.40	64.12	76.92	**42.07**	**88.29**	44.76	88.29
ImageNet200	SSB-Hard	**64.67**	**82.34**	73.71	77.03	72.64	78.30	75.44	71.45	85.65	64.28
	NINCO	**39.93**	**87.35**	46.64	86.10	60.96	82.07	64.31	76.74	77.89	70.84
	Near-OOD	**52.30**	**84.84**	60.18	81.57	66.80	80.18	69.88	74.09	81.77	67.56
	iNaturalist	27.03	90.30	**24.46**	**93.99**	34.38	91.43	50.04	82.31	57.11	80.15
	Textures	41.92	87.76	**24.45**	**95.29**	45.58	90.51	40.41	92.29	30.84	93.79
	OpenImage-O	33.56	89.01	**32.90**	**90.19**	46.54	87.49	63.45	79.12	73.91	74.34
	Far-OOD	34.17	89.02	**27.27**	**93.16**	42.17	89.81	51.30	84.57	53.95	82.76
ImageNet200-FS	SSB-Hard	**88.71**	**54.31**	91.73	44.05	91.16	52.82	92.10	45.05	94.12	45.48
	NINCO	**77.80**	**58.85**	81.23	54.51	86.49	56.64	87.47	50.86	90.20	52.15
	Near-OOD	**83.26**	**56.58**	86.48	49.28	88.82	54.73	89.78	47.95	92.16	48.82
	iNaturalist	69.18	61.08	**69.10**	71.53	71.48	**72.20**	80.17	57.26	77.56	61.86
	Textures	78.88	60.75	69.06	81.88	78.98	74.27	74.02	80.55	**55.63**	**85.60**
	OpenImage-O	**73.85**	60.57	74.43	62.12	79.50	**64.95**	87.17	55.90	88.07	57.23
	Far-OOD	73.97	60.80	**70.86**	**71.84**	76.65	70.47	80.46	64.57	73.75	68.17

The second selected method is k-NN [23]. Using the penultimate layer activations, k-NN calculates the distances to the k nearest ID samples and decides based on these distances. We selected this method because of its distance-based approach and the high metrics obtained on OpenOOD.

The third selected method is SHE, introduced by Zhang *et al.* [43]. Their method trains a pattern for each class from the penultimate layer activations. This method was selected because their pattern approach is analogous to our proposed prototype approach.

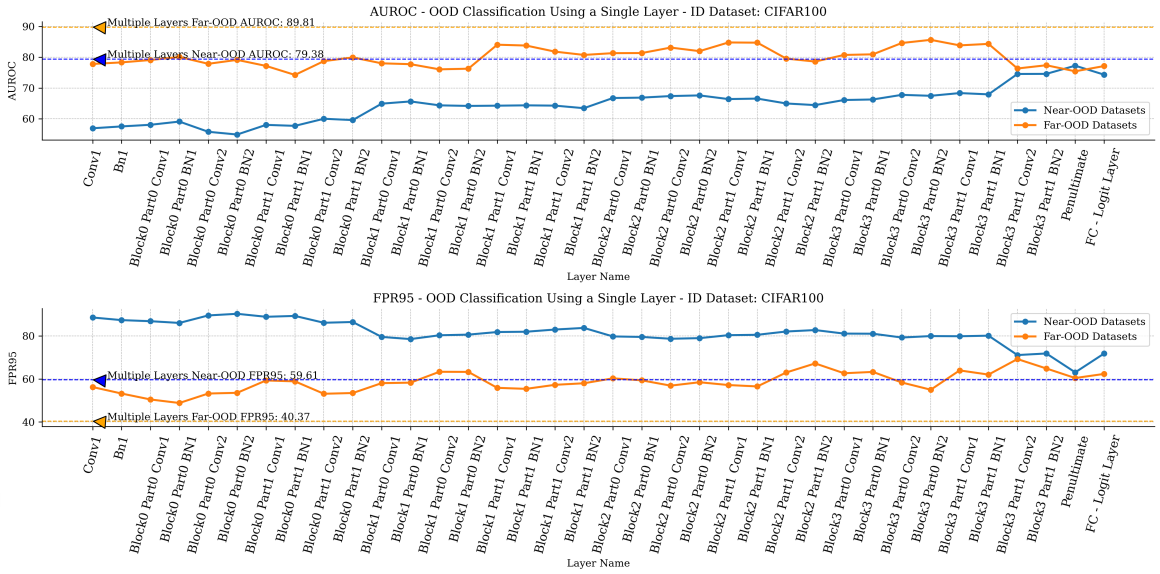

Figure 4.1. OOD detection performance comparison of various layers. It is shown that the layers closer to the penultimate layer have the most significant contribution. Solid lines indicate the performance of individual layers. Our method uses convolutional layers from block 1,2,3 as well as the penultimate and logit layers. The dashed lines indicate the results with the selected layers.

In addition, we proposed a simpler alternative to the prototype approach for comparison. This approach uses a 1D convolutional layer for distance calculation instead of prototypes. The weights of the 1D conv layer are trained on the ID and OOD samples, similar to the OOD classifier. In this approach, the output of the 1D convolutional layer is fed to the OOD classifier instead of the prototype features. Our results show that the prototype approach outperformed the convolutional layer approach, justifying the complexity introduced by the prototype approach.

4.5.2. Experiment Settings

4.5.2.1. Benchmark Process. The benchmark process consists of training the prototypes on the ID dataset, training the OOD classifier on the ID and external OOD datasets, and testing the test part of the ID and OOD datasets. OpenOOD benchmark defines an ID dataset, an external OOD dataset, and OOD datasets.

4.5.2.2. Prototype Feature Generation. Prototype features are the minimum cosine distances and the minimum euclidean distances from the prototype to the vectors in the 3D feature maps. Each selected layer has a separate set of prototype vectors, and the features of each layer are calculated separately. Then, the features from each layer are concatenated before the OOD classification.

The individual contribution of each layer is measured by training an OOD classifier using only that specific layer. The results of this study are shown in Fig. 4.1. Based on these results, deeper layers contribute more to the OOD detection performance. However, two conv layers, that is, two layers before the penultimate, had the most significant contribution.

4.5.2.3. Selected Layers. The ResNet architecture is used throughout our experiments. This model has four main blocks that consist of two parts. Each part has two convolutional layers. The ResNet model also includes a penultimate layer and another fully connected (FC) layer. In our experiments, we used only the first convolutional layer of the second part from various blocks. The logit layer and the AvgPool were also used, which adds up to a maximum of six selected layers for each experiment. The selected layer setup is also a hyperparameter and must be optimized on the validation set. The selected layers for each experiment setup are:

- CIFAR10 is ID: Part 2 ReLU 2, Part 3 ReLU 2, Part 4 ReLU 2, Logit Layer
- CIFAR100 is ID: Part 2 ReLU 2, Part 3 ReLU 2, Part 4 ReLU 2, Avgpool, Logit Layer
- ImageNet-200, ImageNet-200 FS is ID: Part 4 ReLU 2, Avgpool, Logit Layer

Also, training the prototypes on the earlier layers introduced a significant memory cost. For this reason, we created a downsampling mechanism for the layers with large spatial dimensions for resource optimization.

4.5.2.4. OOD Classifier. The OOD classifier is a multi-layer perceptron (MLP) trained on a binary classification task using binary cross-entropy loss. Before this training, distance features are calculated from the ID and OOD datasets using the prototypes. The two classes in this task are the ID and OOD, while by definition, the positive class is the OOD samples. In the prototype approach, only the MLP is optimized during backpropagation. However, in the 1D-Conv setting, the convolutional layer is also trained.

4.5.2.5. Hyperparameters. Hyperparameters of the prototype learning module are the number of learning rates, prototype count, training epochs, batch size, and the selected layers. The OOD classifier module uses the number of hidden units, learning rate, batch size, and epoch count. Also, the level of augmentations is a hyperparameter for both modules, which significantly impacts the results. All hyperparameters are optimized on the validation set. However, some extreme settings resulted in severe overfitting to the validation set, such as adding a center cropping to all samples. In such cases, we did not proceed with the optimal hyperparameters on the validation set.

4.5.2.6. Prototype Count. Additionally, each layer can have a different number of prototypes. We previously discussed that the number of prototypes should be higher than the number of categories. However, using many prototypes also causes overfitting, according to our observations. Generally, the number of prototypes largely depends on the dataset and must be optimized for each use case using the validation set.

4.5.2.7. Validation. Validation was conducted using the validation set for evaluation instead of the test set. However, we did not use a hyperparameter optimization technique for systematic optimization. The final hyperparameters were selected based on validations using manually selected values for hyperparameters.

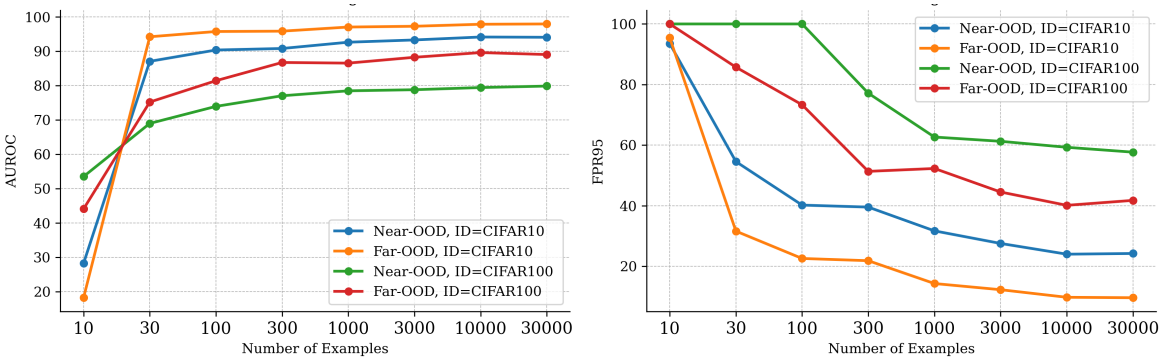

Figure 4.2. Relationship of OOD detection performance with the OOD dataset size. Left figure is the AUROC metric and the right figure is the FPR95. We show that the number of required examples before a plateau depends on the dataset's complexity. For example, CIFAR100 requires a larger OOD dataset than CIFAR10.

Our results showed that the hyperparameters with the most significant impact were the augmentation level, the number of hidden units of the OOD classifier, and the batch size. Training the prototypes for many epochs, such as 100, reduced OOD detection performance, indicating a need for early stopping. Also, a high learning rate in OOD classifier training caused overfitting. We used early stopping during the training of the OOD classifier, which was generally activated between epochs 5-10. Also, we observed significant overfitting during the training of the OOD classifier for the ImageNet-200 dataset, which could be why our method is underperforming on this dataset.

Table 4.2. Ablation Study Metrics

Method	ID Dataset	Near-OOD		Far-OOD	
		FPR95	AUROC	FPR95	AUROC
Regular (Augmentation, Prototype Training, Separate Features for Spatial Dimensions)	CIFAR10	23.90	94.16	**8.23**	**98.15**
	CIFAR100	58.25	80.59	**41.81**	**88.52**
No Augmentations	CIFAR10	**21.65**	**94.94**	10.04	97.73
	CIFAR100	**55.72**	**81.20**	50.54	84.66
No Prototype Training	CIFAR10	23.82	94.20	9.94	97.86
	CIFAR100	57.73	80.10	43.29	88.52
Spatial Averaging (Avg. and Max Pooled Similarities)	CIFAR10	24.81	93.80	9.94	97.81
	CIFAR100	60.11	79.66	48.07	86.12

4.5.2.8. Augmentations. Our experiments demonstrate that two augmentations can effectively enhance the metrics on the CIFAR100 test set: scaling and center crop. Interestingly, applying the augmentations to all datasets reduced the overall performance. We observed that using these augmentations on the OOD samples during the training phase improved the outcome. Augmentations were not used during the testing stage because they would introduce a bias to our results and invalidate them. This would also make our results incompatible with the OpenOOD benchmark.

We also tested other augmentations such as rotation, skew, color, and perspective distortion. However, these augmentations did not provide any improvements.

4.5.2.9. Computational Cost. The proposed method has a computational cost largely depends on the number of selected layers and the prototype count. However, the method is lightweight during inference, adding close to 1 ms for each layer to the response time. However, the training stage can have large memory requirements if very early layers are selected. In order to alleviate this issue, we introduced downscaling to each layer so that each activation layer is downsampled to make their spatial dimensions less than a defined value.

4.5.2.10. Prototype Contributions. In the analysis stage of the OOD detection experiments, we investigated the similarity characteristics of the prototypes to the training dataset to examine the information that they had learned. The most significant observation was that some prototypes exhibited higher cosine similarities to the feature vectors around the edges of the feature maps. These prototypes also showed higher similarities to the OOD samples. Therefore, we hypothesize that these prototypes learned features related to the background. Another set of prototypes showed higher similarities to the center but towards a smaller set of examples. This behavior indicates that this set of prototypes learned to represent class-related features. For this reason, the required number of prototypes can be less than the number of classes because background-learning prototypes can be used to recognize easier OOD samples.

5. CONCLUSION

This thesis explored how intermediate layer activations of DNNs could provide insight to enhance the adversarial robustness and OOD detection performance of DNNs, then proposed a novel OOD detection method that is on par with the state-of-the-art methods on some datasets.

This thesis consisted of three stages. In the first stage, we explored the latent space of DNNs and proposed a latent space exploration framework.

In the next stage, we further explored the intermediate layer activations from an adversarial robustness perspective and conducted several experiments to improve the adversarial robustness of DNNs. Also, this thesis presents various properties that adversarial training induces, such as the alignment between data clusters, activation distributions, perturbation filtering, and correlations between weights.

In the OOD detection stage, we first investigated how OOD data differ from ID in the latent space. Inspired by that, we proposed our novel method, prototype similarity. The detection performance of our proposed method is on par with the state-of-the-art methods on several datasets. Also, in the current state of the OOD detection research, no high-ranking method on the OpenOOD benchmark uses intermediate layer activations. This thesis highlights that the intermediate layer activations can enhance the OOD detection performance.

Enhancing the detection performance in datasets with larger image sizes is essential in further research, such as ImageNet. Also, in our adversarial robustness experiments, we adopted a bottom-up approach that directly aims to induce the observed property. For example, the cluster alignment experiment directly shifted clusters. This approach made the optimization process considerably more challenging. In future work, we could design experiments that interact with the model through more general objec-

tives. Also, the exploration tool proposed in the first stage can be used in other tasks, such as data visualization.

REFERENCES

1. Gawlikowski, J., C. R. T. Njieutcheu, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A Survey of Uncertainty in Deep Neural Networks", *Artificial Intelligence Review*, Vol. 56, No. 1, pp. 1513–1589, 2023.

2. Sun, Y., C. Guo and Y. Li, "ReAct: Out-of-Distribution Detection with Rectified Activations", *35th Conference on Neural Information Processing Systems*, Vol. 34, pp. 144–157, Virtual Conference, 2021.

3. Yang, J., K. Zhou and Z. Liu, "Full-Spectrum Out-of-Distribution Detection", *International Journal of Computer Vision*, Vol. 131, No. 10, pp. 2607–2622, 2023.

4. Goodfellow, I. J., J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples", ArXiv:1412.6572, 2014.

5. Montúfar, G., R. Pascanu, K. Cho and Y. Bengio, "On the Number of Linear Regions of Deep Neural Networks", ArXiv:1402.1869, 2014.

6. Park, J., J. C. L. Chai, J. Yoon and A. B. J. Teoh, "Understanding the Feature Norm for Out-of-Distribution Detection", *IEEE/Computer Vision Foundation International Conference on Computer Vision*, pp. 1557–1567, Los Alamitos, CA, USA, 2023.

7. Wan, W., W. Zhang, Q. Zhou, F. Yi and C. Jin, "Out-of-Distribution Detection Using Neural Activation Prior", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 1234–1243, Los Alamitos, CA, USA, 2024.

8. Zeiler, M. D. and R. Fergus, "Visualizing and Understanding Convolutional Networks", D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (Editors), *European Conference on Computer Vision*, pp. 818–833, Cham, Switzerland, 2014.

9. Huang, Q., I. Katsman, Z. Gu, H. He, S. Belongie and S.-N. Lim, "Enhancing Adversarial Example Transferability With an Intermediate Level Attack", *2019 IEEE/Computer Vision Foundation International Conference on Computer Vision*, pp. 4732–4741, Los Alamitos, CA, USA, 2019.

10. Pócoš, , I. Bečková and I. Farkaš, "Examining the Proximity of Adversarial Examples to Class Manifolds in Deep Networks", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 1234–1243, New Orleans, LA, USA, 2022.

11. Renkhoff, J., W. Tan, A. Velasquez, W. Y. Wang, Y. Liu, J. Wang, S. Niu, L. B. Fazlic, G. Dartmann and H. Song, "Exploring Adversarial Attacks on Neural Networks: An Explainable Approach", *IEEE International Performance, Computing, and Communications Conference*, pp. 41–42, Los Alamitos, CA, USA, 2022.

12. Lee, K., K. Lee, H. Lee and J. Shin, "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks", *Advances in Neural Information Processing Systems*, Vol. 31, pp. 7167–7177, 2018.

13. Hendrycks, D., M. Mazeika and T. Dietterich, "Deep Anomaly Detection with Outlier Exposure", *International Conference on Learning Representations*, New Orleans, Louisiana, USA, 2019.

14. Hendrycks, D. and K. Gimpel, "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks", ArXiv:1610.02136, 2018.

15. DeVries, T. and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout", ArXiv:1708.04552, 2017.

16. Sun, Y. and Y. Li, "DICE: Leveraging Sparsification for Out-of-Distribution Detection", *European Conference on Computer Vision*, pp. 691–708, Tel Aviv, Israel, 2022.

17. Djurisic, A., N. Bozanic, A. Ashok and R. Liu, "Extremely Simple Activation Shaping for Out-of-Distribution Detection", ArXiv:2209.09858, 2022.

18. Ming, Y., Y. Sun, O. Dia and Y. Li, "How to Exploit Hyperspherical Embeddings for Out-of-Distribution Detection?", ArXiv:2203.04450, 2022.

19. Hsu, Y.-C., Y. Shen, H. Jin and Z. Kira, "Generalized ODIN: Detecting Out-of-Distribution Image without Learning from Out-of-Distribution Data", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 10951–10960, Seattle, Washington, USA, 2020.

20. Lee, J. and G. AlRegib, "Gradients as a Measure of Uncertainty in Neural Networks", *2020 IEEE International Conference on Image Processing*, pp. 2416–2420, Abu Dhabi, United Arab Emirates, 2020.

21. Liu, W., X. Wang, J. Owens and Y. Li, "Energy-Based Out-of-Distribution Detection", *34th Conference on Neural Information Processing Systems*, pp. 21464–21475, Virtual Conference, 2020.

22. Liang, S., Y. Li and R. Srikant, "Enhancing the Reliability of Out-of-Distribution Image Detection in Neural Networks", *6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

23. Sun, Y., Y. Ming, X. Zhu and Y. Li, "Out-of-Distribution Detection with Deep Nearest Neighbors", *39th International Conference on Machine Learning*, pp. 20827–20840, Baltimore, Maryland, USA, 2022.

24. Özgür, O. and İnci Meliha Baytaş, "Improving Out-of-Distribution Detection in Deep Models by Latent Space Analysis", *IEEE International Conference on Knowledge Graph*, pp. 1–9, Abu Dhabi, United Arab Emirates, 2024.

25. Pascanu, R., G. Montufar and Y. Bengio, "On the Number of Response Regions of Deep Feed Forward Networks with Piecewise Linear Activations",

ArXiv:1312.6098, 2014.

26. Madry, A., A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks", ArXiv:1706.06083, 2017.

27. Sun, Y., Y. Ming, X. Zhu and Y. Li, "Out-of-Distribution Detection with Deep Nearest Neighbors", *39th International Conference on Machine Learning*, Vol. 162, pp. 20827–20840, Baltimore, Maryland, USA, 2022.

28. Meegahapola, L., V. Subramaniam, L. Kaplan and A. Misra, "Prior Activation Distribution: A Versatile Representation to Utilize Deep Neural Network Hidden Units", ArXiv:1907.02711, 2019.

29. Sinhamahapatra, P., F. Schwaiger, S. Bose, H. Wang, K. Roscher and S. Guennemann, "Finding Dino: A Plug-and-Play Framework for Unsupervised Detection of Out-of-Distribution Objects Using Prototypes", ArXiv:2404.07664, 2024.

30. Rabbi, M. S. E., A. H. M. Rubaiyat, Y. Zhuang and G. K. Rohde, "A Sliced-Wasserstein Distance-Based Approach for Out-of-Class-Distribution Detection", ArXiv:2302.01459, 2023.

31. Du, X., Z. Wang, M. Cai and Y. Li, "Virtual Outlier Synthesis: Learning What You Don't Know by Virtual Outlier Synthesis", *International Conference on Learning Representations*, Vienna, Austria, 2022.

32. Tao, L., X. Du, X. Zhu and Y. Li, "Non-Parametric Outlier Synthesis", ArXiv:2303.02966, 2023.

33. Wang, Q., J. Ye, F. Liu, Q. Dai, M. Kalander, T. Liu, J. Hao and B. Han, "Out-of-Distribution Detection with Implicit Outlier Transformation", ArXiv:2303.05033, 2023.

34. Hendrycks, D., M. Mazeika, S. Kadavath and D. Song, "Using Self-Supervised

Learning Can Improve Model Robustness and Uncertainty", *Advances in Neural Information Processing Systems*, Vol. 32, 2019.

35. Tack, J., S. Mo, J. Jeong and J. Shin, "Contasting Shifted Instances: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 11839–11852, Vancouver, Canada, 2020.

36. Winkens, J., R. Bunel, A. G. Roy, R. Stanforth, V. Natarajan, J. R. Ledsam, P. MacWilliams, P. Kohli, A. Karthikesalingam, S. Kohl *et al.*, "Contrastive Training for Improved Out-of-Distribution Detection", ArXiv:2007.05566, 2020.

37. Ruff, L., R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller and M. Kloft, "Deep One-Class Classification", *35th International Conference on Machine Learning*, pp. 4393–4402, Stockholm, Sweden, 2018.

38. Wei, H., R. Xie, H. Cheng, L. Feng, B. An and Y. Li, "Mitigating Neural Network Overconfidence with Logit Normalization", *39th International Conference on Machine Learning*, Vol. 162, pp. 23631–23644, Baltimore, Maryland, USA, 2022.

39. Song, Y., N. Sebe and W. Wang, "RankFeat: Rank-1 Feature Removal for Out-of-Distribution Detection", *36th Conference on Neural Information Processing Systems*, Vol. 35, pp. 17885–17898, New Orleans, Louisiana, USA, 2022.

40. Song, Y., N. Sebe and W. Wang, "RankFeat&RankWeight: Rank-1 Feature/Weight Removal for Out-of-Distribution Detection", ArXiv:2311.13959, 2023.

41. Huang, R., A. Geng and Y. Li, "On the Importance of Gradients for Detecting Distributional Shifts in the Wild", *Advances in Neural Information Processing Systems*, Vol. 34, No. 1, pp. 677–689, 2021.

42. Roth, K., L. Pemula, J. Zepeda, B. Schölkopf, T. Brox and P. Gehler, "Towards Total Recall in Industrial Anomaly Detection", *IEEE/Computer Vision Founda-*

tion Conference on Computer Vision and Pattern Recognition, pp. 14318–14328, New Orleans, Louisiana, USA, 2022.

43. Zhang, J., Q. Fu, X. Chen, L. Du, Z. Li, G. Wang, X. Liu, S. Han and D. Zhang, "Out-of-Distribution Detection Based on In-Distribution Data Patterns Memorization with Modern Hopfield Energy", *Eleventh International Conference on Learning Representations*, Kigali, Rwanda, 2023.

44. Lin, Z., S. D. Roy and Y. Li, "MOOD: Multi-level Out-of-distribution Detection", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 15308–15318, Virtual Conference, 2021.

45. Park, J., J. C. Long Chai, J. Yoon and A. B. Jin Teoh, "Understanding the Feature Norm for Out-of-Distribution Detection", *IEEE/Computer Vision Foundation International Conference on Computer Vision*, pp. 1557–1567, Los Alamitos, CA, USA, 2023.

46. Yang, J., P. Wang, D. Zou, Z. Zhou, K. Ding, W. Peng, H. Wang, G. Chen, B. Li, Y. Sun, X. Du, K. Zhou, W. Zhang, D. Hendrycks, Y. Li and Z. Liu, "Open Out-of-Distribution Detection: Benchmarking Generalized Out-of-Distribution Detection", *Thirty-Sixth Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, pp. 32598–32611, New Orleans, Louisiana, USA, 2022.

47. Zhang, J., J. Yang, P. Wang, H. Wang, Y. Lin, H. Zhang, Y. Sun, X. Du, K. Zhou, W. Zhang *et al.*, "OpenOOD v1.5: Enhanced Benchmark for Out-of-Distribution Detection", ArXiv:2306.09301, 2023.

48. Yang, J., H. Wang, L. Feng, X. Yan, H. Zheng, W. Zhang and Z. Liu, "Semantically Coherent Out-of-Distribution Detection", *IEEE/Computer Vision Foundation International Conference on Computer Vision*, pp. 8301–8309, Montreal, Canada, 2021.

49. Deng, L., "The Modified National Institute of Standards and Technology Database of Handwritten Digit Images for Machine Learning Research", *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp. 141–142, 2012.

50. Krizhevsky, A. and G. Hinton, "Learning Multiple Layers of Features from Tiny Images", Technical report, University of Toronto, 2009.

51. Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database", *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, Miami, Florida, USA, 2009.

52. Recht, B., R. Roelofs, L. Schmidt and V. Shankar, "Do ImageNet Classifiers Generalize to ImageNet?", ArXiv:1902.10811, 2019.

53. Hendrycks, D. and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations", ArXiv:1903.12261, 2019.

54. Hendrycks, D., S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt and J. Gilmer, "The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization", *IEEE/Computer Vision Foundation International Conference on Computer Vision*, pp. 8320–8329, Montreal, QC, Canada, 2021.

55. Cimpoi, M., S. Maji, I. Kokkinos, S. Mohamed and A. Vedaldi, "Describing Textures in the Wild", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, Columbus, Ohio, USA, 2014.

56. Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng *et al.*, "Reading Digits in Natural Images with Unsupervised Feature Learning", *Neural Information Processing Systems Workshop on Deep Learning and Unsupervised Feature Learning*, Vol. 2011, p. 7, Granada, Spain, 2011.

57. Zhou, B., A. Lapedriza, A. Khosla, A. Oliva and A. Torralba, "Places: A 10 Million

Image Database for Scene Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 6, pp. 1452–1464, 2018.

58. Vaze, S., K. Han, A. Vedaldi and A. Zisserman, "Open-Set Recognition: A Good Closed-Set Classifier is All You Need", ArXiv:2110.06207, 2021.

59. Bitterwolf, J., M. Müller and M. Hein, "In or Out? Fixing ImageNet Out-of-Distribution Detection Evaluation", ArXiv:2306.00826, 2023.

60. Van Horn, G., O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona and S. Belongie, "The iNaturalist Species Classification and Detection Dataset", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 8769–8778, Salt Lake City, Utah, USA, 2018.

61. Wang, H., Z. Li, L. Feng and W. Zhang, "ViM: Out-Of-Distribution with Virtual-logit Matching", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 4921–4930, New Orleans, Louisiana, USA, 2022.

62. Croce, F., M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal and M. Hein, "RobustBench: A Standardized Adversarial Robustness Benchmark", ArXiv:2010.09670, 2021.

63. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, Nevada, USA, 2016.

64. Zhang, H., Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui and M. I. Jordan, "Theoretically Principled Trade-off Between Robustness and Accuracy", *33rd Conference on Neural Information Processing Systems*, pp. 7428–7437, Vancouver, Canada, 2019.

65. van der Maaten, L. and G. Hinton, "Visualizing Data Using t-Distributed Stochastic Neighbor Embedding", *Journal of Machine Learning Research*, Vol. 9, No. 86, pp. 2579–2605, 2008.

66. McInnes, L., J. Healy and J. Melville, "Uniform Manifold Approximation and Projection for Dimension Reduction", ArXiv:1802.03426, 2018.

67. Gowal, S., S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian and T. Mann, "Improving Robustness Using Generated Data", ArXiv:2110.09468, 2021.

68. Hendrycks, D., M. Mazeika and T. Dietterich, "Deep Anomaly Detection with Outlier Exposure", ArXiv:1812.04606, 2018.

69. Yang, J., P. Wang, D. Zou, Z. Zhou, K. Ding, W. Peng, H. Wang, G. Chen, B. Li, Y. Sun *et al.*, "OpenOOD: Benchmarking Generalized Out-of-Distribution Detection", *Advances in Neural Information Processing Systems*, Vol. 35, pp. 32598–32611, New Orleans, Louisiana, USA, 2022.

70. Zhang, J., J. Yang, P. Wang, H. Wang, Y. Lin, H. Zhang, Y. Sun, X. Du, K. Zhou, W. Zhang *et al.*, "Openood v1.5: Enhanced Benchmark for Out-of-Distribution Detection", ArXiv:2306.09301, 2023.

71. Kansizoglou, I., L. Bampis and A. Gasteratos, "Deep Feature Space: A Geometrical Perspective", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 44, No. 10, pp. 6823–6838, 2022.

72. Hinton, G. E. and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", *Science*, Vol. 313, No. 5786, pp. 504–507, 2006.

73. Fisher, R. A., "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, Vol. 7, No. 2, pp. 179–188, 1936.

74. Liu, X., Y. Lochman and C. Zach, "GEN: Pushing the Limits of Softmax-Based Out-of-Distribution Detection", *IEEE/Computer Vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 23946–23955, Vancouver, Canada, 2023.

75. Regmi, S., B. Panthi, S. Dotel, P. K. Gyawali, D. Stoyanov and B. Bhattarai, "T2FNorm: Extremely Simple Scaled Train-Time Feature Normalization for OOD Detection", ArXiv:2305.17797, 2023.

76. Zhang, J., N. Inkawhich, R. Linderman, Y. Chen and H. Li, "Mixture Outlier Exposure: Towards Out-of-Distribution Detection in Fine-Grained Environments", *IEEE/Computer Vision Foundation Winter Conference on Applications of Computer Vision*, pp. 5531–5540, Waikoloa, Hawaii, USA, 2023.

77. Sastry, C. S. and S. Oore, "Detecting Out-of-Distribution Examples with Gram Matrices", *37th International Conference on Machine Learning*, pp. 8491–8501, Vienna, Austria, 2020.

78. Hendrycks, D., A. Zou, M. Mazeika, L. Tang, B. Li, D. Song and J. Steinhardt, "PixMix: Dreamlike Pictures Comprehensively Improve Safety Measures", *2022 IEEE/Computer vision Foundation Conference on Computer Vision and Pattern Recognition*, pp. 16762–16771, New Orleans, Louisiana, USA, 2022.

79. Yu, Q. and K. Aizawa, "Unsupervised Out-of-Distribution Detection by Maximum Classifier Discrepancy", ArXiv:1908.04951, 2019.

80. Erhan, D., Y. Bengio, A. Courville and P. Vincent, "Visualizing Higher-Layer Features of a Deep Network", , 2009, technical Report 1341, Université de Montréal.

81. Ngiam, J., Z. Chen, D. Chia, P. Koh, Q. Le and A. Ng, "Tiled Convolutional Neural Networks", J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel and A. Culotta (Editors), *Advances in Neural Information Processing Systems*, Vol. 23, pp. 1046–1054, Vancouver, Canada, 2010.

82. ElAraby, M., S. Sahoo, Y. Pequignot, P. Novello and L. Paull, "GROOD: GRadient-Aware Out-Of-Distribution Detection in Interpolated Manifolds", ArXiv:2312.14427, 2023.