UNIVERSITY
OF OSLO

Master's thesis

Explainable Artificial Intelligence for Out-of-Distribution Detection

Using irregularities in machine learning explanations to detect when a model is faced with unusual data

Jonatan Hoffmann Hanssen

Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2025

Jonatan Hoffmann Hanssen

Explainable Artificial Intelligence for Out-of-Distribution Detection

Using irregularities in machine learning explanations to detect when a model is faced with unusual data

Supervisors:
Hugo Lewi Hammer
Kyrre Harald Glette

Abstract

As Artificial Intelligence becomes a larger and larger part of society, the need for robust and understandable models becomes paramount. When neural networks are used in high-impact settings such as cancer detection or autonomous driving, we must require that they

Sammendrag

Here comes the abstract in a different language.

Contents

List of Figures

List of Tables

OOD Out-of-Distribution

DL Deep Learning

ID In-Distribution

CAM Class Activation Mapping

GradCAM Gradient Class Activation Mapping

GAP Global Average Pooling

MSP Maximum Softmax Probability

LIME Local Interpretable Model-Agnostic Explanations

ReLU Rectified Linear Unit

MLS Maximum Logit Score

MSP Maximum Softmax Probability

RMD Relative Mean Absolute Difference

QCD Quartile Coefficient of Determination

VIM Virtual Logit Matching

SoTA State-of-the-Art

CNN Convolutional Neural Network

PCA Principal Component Analysis

SLIC Simple Linear Iterative Clustering

FFNN Feed Forward Neural Network

CV Coefficient of Variance

GBP Guided Backpropagation

ML Machine Learning

LRP Layer Relevance Propagation

XAI Explainable Artificial Intelligence

AI Artificial Intelligence

AUROC Area Under Receiver Operating Characteristic

AUPR Area Under Precision Recall Curve

ROC Receiver Operating Characteristic

FPR False Positive Rate

TPR True Positive Rate

FPR95 False Positive Rate at 95% Recall

Preface

Generative Artificial Intelligence (AI) has **not** been used to generate or enhance any written text contained in this thesis. However, generative AI has been used for some programmatic tasks. Services such as GPT UiO have been used for code debugging, for generating TikZ code used for creating diagrams, and for generating some boilerplate Python code. All data and personal information have been processed in accordance with the University of Oslo's regulations, and I, as the author of the document, take full responsibility for the validity of any generated code used as part of this work.

Chapter 1

Introduction

1.1 Motivation

Machine Learning (ML) generally, and Deep Learning (DL) specifically, have seen a tremendous increase in performance in recent years, performing comparable to humans in tasks such as image classification, speech and handwriting recognition, as well as many others [1]. Consequently, DL methods have been deployed in a multitude of fields and have become a part of our daily lives through their role in web search, text translation, computer vision, and in many other technologies which are taken for granted. In medicine, deep learning has the potential to provide faster and more accurate detection of diseases by being trained on cases from thousands of previous patients [2]. Despite this, the adoption DL in high impact fields, such as the medical field, has been slow, with [3] stating that: "surprisingly little in health care is driven by machine learning".

To explain this discrepancy, we should consider that despite their impressive performance, the application of deep learning methods is not without drawbacks. Firstly, deep neural networks are inherently unexplainable due to the large number of parameters that any non-trivial network has. State-of-the-Art (SoTA) models will perform millions of operations to evaluate a single data point, and it is therefore impossible for humans to comprehend and explain the entire process which lead the model to make a particular decision. In medicine, this is a major limitation of deep learning methods, as both doctors and patients expect to be able to understand why a decision was made [4]. In other high-impact fields, such as autonomous driving, this lack of transparency also has serious practical and legal ramifications.

Secondly, although neural networks may attain high accuracy on test data and appear to have learned great insights about the tasks they are employed in, they often lack robustness and can suffer large drops in performance on data points which are slightly different from the training data. As [5] has shown, it is possible to create data points which are imperceptibly different from normal data points, yet still fool otherwise high performing models. More problematically, unlike humans, who recognize when they are faced with a novel situation where their expertise might be lacking, DL methods will predict equally confidently on data points which are far outside the data they have been trained on [4].

These two problems lead to the fields of XAI, and OOD detection. XAI attempts to explain the reasons why a model came to a decision, which helps to remedy the black-box nature of complicated DL models. In a healthcare setting, such explanations can be inspected by medical practitioners to confirm the diagnosis, and can be used to give patients information about why decisions regarding their health were made. In

autonomous driving or other automated high impact fields, XAI can be used to detect failure modes or to understand and improve deployed models. OOD detection attempts to uncover when a data point is too different from the training data to be classified reliably. These methods could alert medical practitioners when such data points occur, thus avoiding potentially fatal misclassifications. In autonomous driving, the system could detect novel situations and cede control back to the user, avoiding accidents.

Both of these fields have seen increased interest in recent years, and are vital parts of any integration of DL in high impact settings. As two vibrant fields of study, there is great potential to combining insights from one field to improve performance in the other, an area which is underexplored. This thesis will focus on OOD detection, but will attempt to use XAI methods to improve detection performance. The overarching intuition is that by inspecting the explanation of a model on a specific data point, we may be able to uncover flaws or irregularities in the explanation which could help us determine whether the data point is OOD.

1.2 Problem Statement

As explained in the previous segment, OOD detection is a developing field, which has become more important in recent years as machine learning is being used for higher impact tasks, such as disease detection, autonomous driving or infrastructure inspection. Finding novel methods which improve a model's ability to detect when input is OOD is important to increase the robustness of machine learning models as they are used in these real-world scenarios. The field of XAI is concerned with understanding the inner workings of a model, and could thus offer insights which can help us detect unusual behaviour in the model as a result of OOD data points. The problem statement is thus as follows:

In what ways can methods from the field of Explainable Artificial Intelligence be used to improve Out-of-Distribution Detection?

To answer this question, I introduce 2 objectives:

1. Develop OOD detection methods which rely solely on XAI explanations, and compare these methods to traditional OOD detection methods.

2. Develop OOD detection methods that combine traditional and XAI based methods in different ways, and compare these methods to traditional OOD detection methods.

1.3 Scope

As we will see in chapter 2, both the fields of XAI and OOD detection are very large, which makes it impossible to explore all the possible ways one might combine XAI and OOD detection. Thus, it is necessary to restrict the scope of both the XAI and OOD detection methods used. In this section, I will describe the choices I've made and give a short explanation. In later chapters, the choices will be justified more thoroughly.

The field of OOD detection is primarily concerned with image classification tasks. Thus, my project will also deal exclusively with image classification datasets. Given this type of data, the choice of XAI algorithms naturally gravitates towards post-hoc, saliency based methods. The choice of OOD detection methods is not significantly

restricted by the choice to deal exclusively with image data, but I will exclude methods which use outlier exposure to improve performance, or which require retraining of the model. These choices are informed by [6], which have found that outlier data is not necessary to achieve SoTA performance, and that "post-hoc methods [...] are generally no worse than methods that require training".

1.4 Research Methods

Use ACM.

1.5 Ethical Considerations

What are the ethical considerations?
Computing power, environmental impact
Using ImageNet, CIFAR
Ethics of using ai for high impact tasks

1.6 Main Contributions

In this thesis, the goal has been to investigate whether XAI methods can be used for OOD detection, and to introduce proof-of-concept OOD detection models which use XAI explanations as part of their functioning. As part of this work, I have introduced three forms of XAI inspired OOD detection models, which use the saliency maps (heatmaps) generated by XAI methods applied to images: *Saliency Aggregation*, *Saliency Aggregation+Logits* and *SaliencyVIM*.

Through the testing of these three methods, I show that XAI methods can indeed be used for OOD detection, contrary to the results found by previous research [7]. The key takeaways of this thesis are as follows:

- XAI saliency mapping methods generate values which can be used to predict OOD samples comparable to baseline methods. However, most XAI methods output normalized saliencies, which removes valuable information necessary to perform OOD detection. By removing this normalization and using raw saliency values, I show substantial improvements over previous methods [7].
- A simple aggregation of raw saliency values performs slightly below the baseline methods of Maximum Logit Score (MLS) and Maximum Softmax Probability (MSP) on Near-OOD datasets, and sometimes outperforms the baselines on Far-OOD datasets. This shows that XAI saliency maps, on their own, capture enough information about a network's response to a data sample to effectively discriminate between ID and OOD data points.
- Depending on the choice of XAI method, XAI saliency maps output values which are relatively uncorrelated with traditional baseline OOD methods. This means that their outputs can be combined with other OOD detection methods to increase OOD detection performance. By combining XAI saliency aggregates with MLS by a simple addition of metrics, I find that the performance is increased by several percentage points over the baselines in several cases, and in one case performs quite close to the SoTA amongst OOD detection models which do not retrain the underlying classifier.

- Saliency values can also be appended directly to the model logits in OOD detection methods such as Virtual Logit Matching (VIM), leading to statistically significant improvements over using the method without XAI saliency values in some cases.

The three methods are highly general and function as frameworks for further research into the integration of XAI and OOD detection. In addition, I have had code merged into the codebase for OpenOOD, the definitive OOD detection framework and benchmarking tool. By improving the code used when benchmarking new OOD detection methods, I have contributed to the broader field of OOD detection.

1.7 Thesis Outline

Chapter 2 gives a short introduction to machine learning, followed by a deeper look at the fields of XAI and OOD detection. Chapter 3 introduces my methodology; the methods I will use to compare and contrast XAI explanations on ID and OOD data, as well as a three general OOD detection methods which integrate explanations into their functioning. Furthermore, I introduce the different datasets which will be used to test each method, the statistical methods used to ensure that the results are statistically significant and the software and hardware architecture used. 4 will first go into the results of the comparison between ID and OOD explanations, detailing the differences between different XAI methods, these investigations will be done on a validation set. Based on these results, I will make a selection of parameters for the three OOD detectors introduced in chapter 3. These methods will be tested on a bootstrapped test set and compared against baseline methods using statistical analysis. After the experiments follow a discussion (chapter 5), where I reflect on the benefits and drawbacks of using XAI methods for OOD detection, and summarize the results from 4. Finally, the conclusion (chapter 6) gives a short conclusion of my findings, and envisions a way forward for future work.

Chapter 2

Background

In this chapter, I give a short introduction to important concepts in the field of machine learning generally, followed by a more in-depth look at the fields of OOD detection and XAI. Finally, I give an overview of related works; papers which have attempted to use XAI for OOD detection.

2.1 Machine Learning

Machine Learning is the field of algorithms that are able to learn from data, as opposed to being explicitly programmed. Such algorithms use statistical methods to learn relationships in data, and use these relationships to generalize to unseen data. More formally, [8] gives the following definition of machine learning algorithms:

Definition. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

Thus, machine learning is a different paradigm from traditional problem solving, where programs are made to solve problems by following explicit rules. For example, a traditional image classification system attempting to differentiate between malignant and benign tumors might use hand-crafted rules which consider the texture, color and size of a tumor, developed by medical professional with years of experience. As one might imagine, such rules will quickly become very complicated when we consider all the possible factors which might influence the appearance of a tumor. Using a machine learning approach, we would instead feed an algorithm with thousands of images of both benign and malignant tumors, and the rules could then be automatically updated until the algorithm predicted the correct category with a high enough accuracy.

Machine Learning is commonly divided into the three subcategories of supervised, unsupervised and reinforcement learning

2.1.1 Supervised Learning

Supervised Learning is a subcategory of machine learning where we have a dataset containing both inputs and desired outputs. In the example above, we could use supervised learning by creating a dataset of images of tumors (the input) and corresponding labels which indicate whether each tumor is malignant or benign (the desired output). The learning goal of the algorithm is then to associate images with the correct label. Because we know the correct answer, we are able to fine-tune the

algorithm automatically whenever it makes a mistake. However, supervised learning requires labeled data, which can be very costly, especially in the medical domain, where deciding whether a tumor is malignant or benign requires expert knowledge.

2.1.2 Unsupervised Learning

In unsupervised learning, we do not have any labels. In these cases, we might not know whether data points belong to different classes or not. Instead, we can use machine learning to uncover patterns in the data, for example by attempting to cluster the data into different groups and seeing if these groups are sufficiently separated. An example use case could be for fraud detection in a bank. By feeding financial transaction from many different users into an unsupervised learning model and asking it to perform clustering of the data, it might be possible to find a group of users whose transactions differ substantially from the rest, which might indicate that their transactions are fraudulent.

2.1.3 Reinforcement Learning

Reinforcement Learning deals with problems where we do not know exactly what the correct solution is, but we are able to assess whether a given solution is good or not. For example, when controlling a robot arm, it is difficult to say exactly what angles each joint should be for every millisecond when picking up an object, but if the arm does not pick up the object, we know the algorithm has failed. In these problems, the algorithm is trained through reinforcement, where good attempts are rewarded and bad attempts punished.

2.2 Neural Networks

Neural Networks are a class of machine learning algorithms, which have become the clear SoTA in almost all fields where machine learning is applied. Notable examples are computer vision, image classification, speech recognition, text and image generation and machine translation. Neural networks are loosely inspired by our own brains, where neurons are connected together and send information between each other. By connecting thousands of neurons together, neural networks are able to learn complicated relationships between the input and output.

2.2.1 Feed Forward Neural Networks

The Feed Forward Neural Network (FFNN), also known as a Multilayer Perceptron, traces its roots to the very beginning of machine learning, through the work of Frank Rosenblatt [9]. It forms the basic structure for neural networks which has been adapted and modified over the years to form more complex architectures such as convolutional, recurrent or residual neural networks. The basic structure of an FFNN is that the input values are passed through an affine transformation (a matrix multiplication followed by the addition of a bias), and then passed through an activation function, which produces outputs. These outputs can then go through the same process again, which constitutes a single "layer". By stacking several of these layers, with non-linear activation functions, an FFNN is able to learn arbitrarily complex mappings between inputs and outputs[1].

[1] In fact, by the Universal Approximation Theorem [10], only a single hidden layer between the input and output is necessary, although this theorem does not give a way to construct such a network for any given function

Figure 2.1[2] shows a simple FFNN architecture with three hidden layers. In this case, the bias has been omitted for brevity. Here, we can see how all nodes of a layer are connected to the following layer. By using an activation function on the nodes of the hidden layer, before their values are sent to the next layer, we achieve the non-linearity required to learn complex patterns.

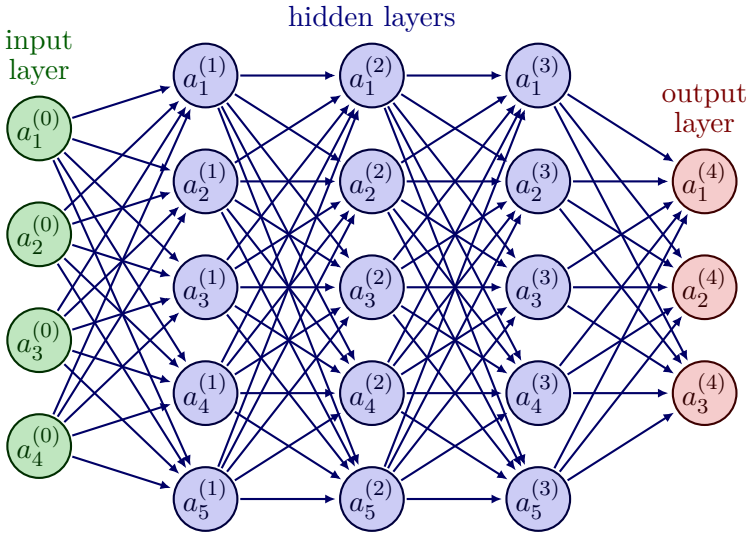

Figure 2.1: Figure showing a simple Feed Forward Neural Network, with nodes labeled. The number in parentheses indicates the layer number while the subscript indicates the node number within the layer.

Mathematically, a single layer can then be described as follows:

$$\boldsymbol{x_{i+1}} = \sigma_i(A_i\boldsymbol{x_i} + \boldsymbol{b_i}) \tag{2.1}$$

Here, the input $\boldsymbol{x_i}$ is linearly transformed by the weights of the matrix A_i from the input space to the output space, then each value of the new vector in the output space is adjusted by an addition of a bias term, and finally an activation function (σ_i) is applied to each value. The size of the input and output layers is determined by the number of input and output features, respectively. Furthermore, the activation function of the output layer is also determined by the application, most commonly *sigmoid* for binary classification, *softmax* for multi-class classification and simply identity for regression.

2.2.2 Convolutional Neural Networks

FFNNs have some inherent flaws which make them unsuitable for working with high dimensional, spatially connected data, such as the pixels which make up an image. Firstly, each input of a FFNN is connected to every output of the following layer. If we want to connect the input pixels of a 224 by 224 image to a layer of 100 nodes, our first layer will have over 5 million weights, which is already quite a lot for a relatively small image. Furthermore, these weights will have to encode redundant information, because each pixel is considered separately. Consider a network attempting to detect the presence of a cat in an image. We would want the network to detect the cat regardless of whether

[2]Figure by Izaak Neutelings, "Neural Network with coefficients, arrows", TikZ.net, licensed under CC BY-SA 4.0, [https://tikz.net/neural_networks/].

it is in the middle, the right corner, or any other position in the image. In an FFNN, the weights connected to any of these positions in the image would then have to encode a cat detector separately from all the others.

Convolutional Neural Networks (CNNs) solve both these issues by using small kernels of weights which are "slid" across the entire input. By using the same weights across all positions of the image, we do not need to train separate detectors for different positions, giving us translation invariance. Figure 2.2 shows the functioning of a convolutional kernel on a 3-channel image. Each value in the output is a weighted sum of a neighbourhood of values in the input image, where the weights are defined by the kernel. As we can see, the same weights are used on all positions, drastically reducing the number of parameters that need to be tuned. In a 2d-convolution, the kernel has the same number of channels as the input, and is only slid across the height and width dimension. The kernel in this figure is a *Sobel Operator*, and detects vertical edges. In a CNN, the weights of each kernel are not specified manually, but rather learned through backpropagation.

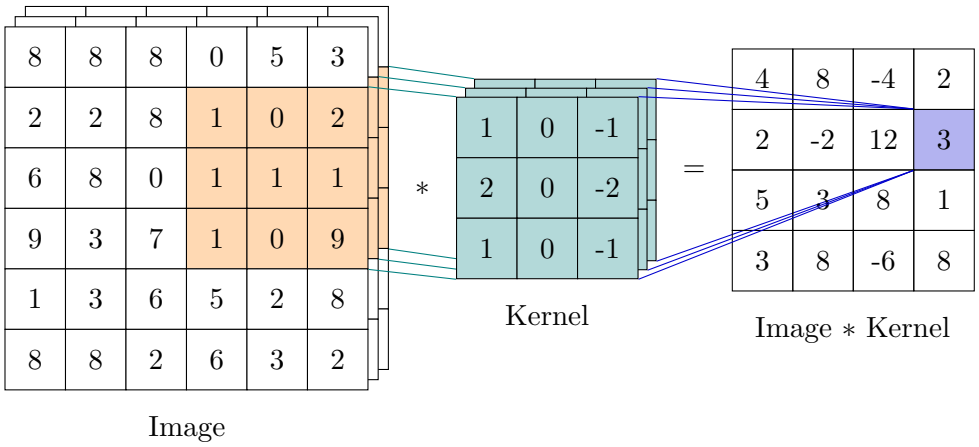

Figure 2.2: Figure showing a convolutional kernel applied to a 3-channel image.

By using several different kernels, we can detect many different patterns despite each kernel only detecting a single type. By using the outputs of all the kernels as inputs to a new set of kernels, we can use the same type layer structure as in an FFNN, allowing us to extract information in a hierarchical manner. It is common to see that trained CNNs have early layers that detect edges and texture, later layers that use these edge and pattern detections to detect larger shapes, while the final layers combine the shapes to detect entire objects [11].

By not evaluating every possible position in the input image, CNNs downsample the image, and are able to reduce the number of operations considerably. Simultaneously, this downsampling enables each subsequent layer to consider a larger area of the input image than the previous (a larger field-of-view), which allows larger patterns to be discovered. Simultaneously, it is common to use a larger and larger amounts of kernels on the new input, thus increasing the channel depth while the spatial dimensions are reduced. Between each layer, we use non-linear activation functions, similarly to how the are used in FFNNs.

After several such convolutions, we can flatten the output, either by aggregating each channel using Global Average Pooling (GAP) or a similar method, or we can simply

flatten all dimensions and consider the three dimensional feature map as a long vector of shape $C \times H \times W$. By doing this, we can pass the output to one or more linear layers, which can perform classification or regression on the extracted features and give us a final prediction. Figure 2.3 shows a high level overview of this process. Here, the input, which has only 3 channels, has its spatial dimensions reduced while its channel depth increases through consecutive convolutions. Finally, we have a certain number of channels in our final feature map, which are flattened (in this case with GAP) and processed through a linear layer to give a final prediction.

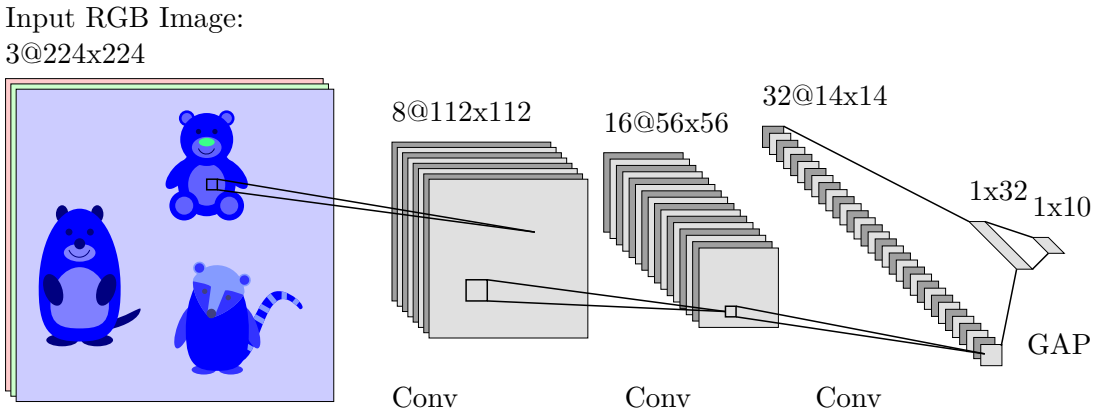

Figure 2.3: Figure showing a high level overview of how a CNN functions

2.3 Model evaluation

2.3.1 Metrics

OOD detection is essentially simply a binary classification problem. Thus, the metrics I will use in this thesis are those used for such problems. In the field of OOD detection, AUROC and False Positive Rate (FPR) are most commonly used. As such I shall focus on these metrics, as opposed to for example the Area Under Precision Recall Curve (AUPR).

Accuracy

Accuracy is the simplest metric used in binary classification. It is simply the ratio of correct predictions over all instances in the data set.

Accuracy has the advantage of being simple to understand and calculate, but it is very often insufficient. A simple (and quite common) scenario where accuracy fails to capture the performance of a model is any situation where there are large class imbalances. For example, imagine we have trained a CNN to predict whether a person has lung cancer or not, based on CT-scans of their lungs. As most people do not have lung cancer, we can imagine that such a dataset is highly imbalanced, for example that only 1 in 100 people actually have cancer. If a model simply predicts that no one ever has lung cancer, it will be correct in 99% of cases, and will thus have an accuracy of 99%, although it has missed every instance of cancer and is completely unusable in any real context.

For the purposes of OOD detection, accuracy is thus insufficient, as we have no guarantees that ID and OOD will be balanced. In fact, we expect OOD data to be relatively rare, given that the goal of developing an AI model is to ensure high

performance during deployment, which necessitates having training data which covers as much as possible of the data seen during inference.

Metrics utilizing the binary classification confusion matrix

Instead of simply considering whether a prediction was correct or not, we should take into account the different combinations of prediction and ground truth, considering positive and negative classes separately. Figure 2.4 shows the possible four possible combinations given a ground truth class and a predicted class.

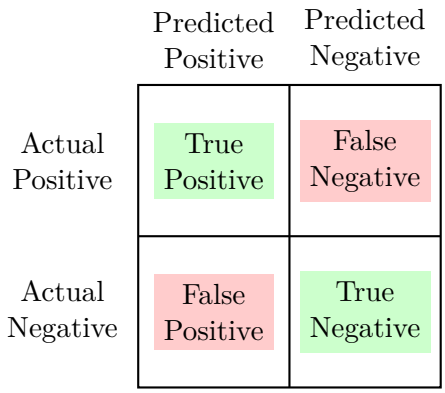

Figure 2.4: Figure showing the binary classification confusion matrix, denoting the four possible combinations created by the ground truth and predicted class. Green cells denote correct predictions, while red cells denote wrong predictions.

With these possibilities defined, we can begin to gain a clearer picture of the performance of a model.

Precision and Recall

Precision is the share of positive predictions that were actually positive. With a high false positive rate, we have a low precision, which means that the model is erroneously flagging many negative classes as positive. In an OOD detection setting, a model which flags many ID samples as OOD would have a low precision, if we treat OOD samples as the positive class.

Recall is the share of actual positive samples that were predicted positive. Recall tells us how many of positive samples we missed. In and OOD detection context, a model which lets many OOD samples slip by undetected will have a low recall score.

Precision and recall are often used together because evaluate the model in different ways that complement each other. If a model has both high precision and high recall, it does not erroneously flag many negative classes as positive, nor does it miss many positive classes.

Sensitivity and Specificity

Sensitivity and specificity is another pair of metrics that is commonly used for evaluating binary classification. Sensitivity is equivalent to recall; the share of positive samples that were correctly predicted as positive. Specificity is the share of the negative samples that were correctly predicted as negative. Sensitivity and specificity are also known as True Positive Rate (TPR) and *True Negative Rate*.

Threshold Independent Metrics

The previous metrics are a clear improvement over simply using accuracy. However, they still have the problem that they are all dependent on what threshold one sets when predicting something to be a negative or positive class. Thus, it becomes harder to compare different models by using these metrics. Indeed, by simply increasing the threshold of any classifier, we can increase the true negative rate. Similarly, by decreasing the threshold, we can increase the true positive rate.

Area under Receiver Operating Characteristic:
AUROC remedies this problem by looking at all possible thresholds, and calculating the TPR (equivalent to sensitivity, recall), and the FPR (equivalent to 1− specificity) for each possible threshold. With these values calculated, we can plot each point on a graph, giving us an Receiver Operating Characteristic (ROC) plot. Figure 2.5 shows this plot, for three different models.

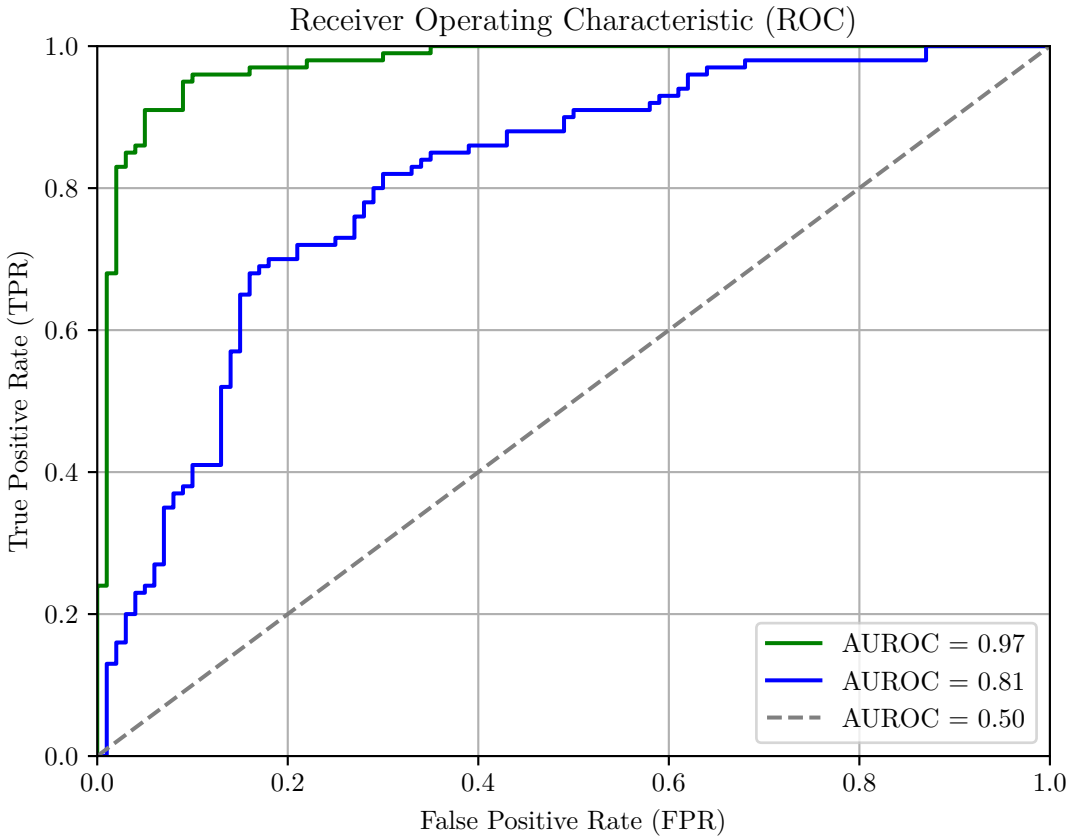

Figure 2.5: Graph showing the distribution of hypothetical ID, Near-OOD and Far-OOD data for an unspecified metric. The shaded region shows the overlap between the ID and OOD samples.

Once we have done this, we can calculate the integral under this curve, giving us the AUROC. If a binary classification model can perfectly separate the two classes, then all possible thresholds will either have 100% TPR or 0% FPR, giving an area under the curve of 1. If instead a model has no discriminative power, then the predicted values

of positive and negative classes are entirely random, and all changes to the threshold will increase one of the metrics at the expense of the other. Such a model would have TPRs and FPRs making a straight line of points, and an AUROC of 0.5. In between these extremes, we can evaluate different models, without having to consider different thresholds. 2.5 shows what the ROC-curve of a model with either 0.97 or 0.80 AUROC looks like, as well as that of a random classifier with AUROC = 0.50.

One important thing to note about the AUROC is that values lower than 0.50 do not mean that a model is worse than random guessing. This is because if a model is consistently wrong, we can simply choose the opposite category of what the model outputs, and gain a new model which is better than random guessing. For example, if a cat versus dog detector gave an actual image dog a higher chance of being a cat than an actual image of a cat in 95% of cases, it would have an AUROC of only 0.05. However, if we simply multiplied all outputs of the model by -1, we would suddenly have a model which correctly gives a cat a higher chance of being a cat in 95% of the cases, and an AUROC of 0.95. Thus, we really only care about getting AUROC scores far away from 0.50.

False Positive Rate at 95% Recall
False Positive Rate at 95% Recall (FPR95) is another way of comparing models without having to consider specific thresholds. Instead, we simply select the threshold which gives a recall (equivalent to TPR) of 0.95, and calculate the FPR at this threshold. The drawback to this metric as opposed to AUROC is that we do not get a general view of how the model performs. However, if we have a requirement that the model has a very high true positive rate, we may not care about how the model performs at any other threshold, and thus this metric is suitable. It is of course also possible to calculate this metric at any other recall value, depending on the application. However, in the field of OOD detection, FPR95 is the metric that is used in the vast majority of cases [6, 12–15].

2.3.2 Statistics: Bootstrapping and T-tests

Bootstrapping [16] is a way to get a better estimation of the true generalization error of a model, by performing the same experiments several times on resampled versions of the original dataset.

2.4 Explainable Artificial Intelligence

Below follows a thorough introduction to XAI, as well as detailed look at some important methods for explainability for neural networks applied to images. Specifically, saliency methods will be explained in detail, as they constitute a core part of my thesis.

2.4.1 The motivation for Explainable Artificial Intelligence

Given the impressive performance of DL methods, one might be convinced that these models do not need to be explainable or interpretable, and that we instead should just place our faith in the model without knowing exactly how it came to a decision. However, as [17] points out, "a single metric, such as classification accuracy, is an incomplete description of most real-world tasks". Small differences between the data distribution when the test data was collected and when the model is deployed may have a large impact on the model's performance, or the model may have learned artifacts or specificities in the training dataset which were also present in the test dataset, leading to a false

belief that the model has gained generalizable knowledge when it has not. By using explainable methods, we may reveal these shortcomings. In my case, this may also have the secondary effect of separating ID and OOD data points.

XAI is also especially important whenever the model is used in settings where its decisions have a high impact. If a model is used by a hospital for disease detection, both the patient and doctor will probably want to be able to understand why the model has found that a disease is present. For them, high performance on a test set of different cases may not be enough. As [2] states, "for the regulated healthcare domain, it is utmost important to comprehend, justify, and explain the AI model predictions for a wider adoption of automated diagnosis". In other high impact areas, such as autonomous driving, the impact of wrong decisions by the network can have fatal consequences, and customers and regulators will want to be absolutely sure that the models used are robust and base their decisions on relevant factors as opposed to quirks in the training data. Furthermore, the right to an explanation of an automated decision affecting a person is included in the EU's General Data Protection Regulation, which states that "In any case, such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right [...] to obtain an explanation of the decision reached after such assessment and to challenge the decision." [18].

2.4.2 Taxonomy of Explainable Artificial Intelligence

This section goes through three axes which define an XAI method:

- Intrinsically explainable models versus post hoc methods
- Model dependent versus model agnostic methods
- Global versus local explanations

Intrinsically explainable models versus post hoc methods

Intrinsically explainable models are models which have sufficiently low complexity, such that it is feasible for a human to understand them without further modifications. Examples of such methods are linear regression, logistic regression and decision trees [19].

Post hoc methods are methods which are applied to the model after training. These methods do not aim to constrain the model to be interpretable, but inspect the model after training. For example, after using a convolutional neural network to classify a CT-scan of a tumour (which gave a prediction of malignant), we could run post hoc algorithms on the network which are able to extract which part of the image contributed the most to the prediction. Thus, post hoc methods remove the need for the model to be simple enough for a human to understand by extracting the relevant information for us.

Model dependent versus model agnostic methods

Model dependence/agnosticity denotes whether an XAI method uses specifics of a particular type of model to generate the explanation, or whether the method can generate an explanation without using specifics of the model at all. Explanations based intrinsically explainable models are clearly model dependent, while methods that only use the input and output of the model instead of looking at the internal operations

are model agnostic. An example of a model dependent method (which is not simply an intrinsically explainable model) is Class Activation Mapping, which requires a CNN with a specific architecture to function, while an example of a model agnostic method are Shapley values, which use the inputs and outputs to calculate the marginal effect of a single feature on the output value.

Global versus local explanations

Global explanations provide general relationships between the input features and outputs learned by the model over the entire dataset [20]. In this way, they can show how a specific feature affects the output in general, instead of just how it affects the output of a single point. These methods are ideal for finding trends in the data, but may not be suitable for a patient wanting an explanation for their specific case.

Local explanations do not describe general trends, but focus only on a single data point. These methods give insight into how the features influenced the prediction of a single data point, but these relationships may not hold for other data points, and as such these methods do not give the same insight into the general behaviour of the model.

2.4.3 XAI methods adapted to images: Saliency maps and segmentation

As explained in section 1.3, the field of OOD detection is primarily focused on image data, and as such this is the focus of this thesis as well. Thus, before delving into specific XAI methods, it is beneficial to elaborate on how XAI methods are adapted to images. When explaining tabular data made up of categorical and numerical values, it is often common to explain each feature by associating it with some change in the output prediction. For example, one might say that increasing the number of rooms in a house by one increases the predicted sale price by 30 000 NOK, or that the absence of a balcony decreases it by 25 000 NOK. But, given that images are made up of tens or hundreds of thousands of features (pixels), such an approach may not be suitable.

Firstly, given that we have so many features, it may no longer be interesting to know exactly how much each feature contributes to a prediction, given that we don't expect any single pixel to have a very large impact. Furthermore, inspecting the contribution of each individual pixel, like one might do for tabular data, is not even realistically feasible, due the overwhelming number of features. Given that our input is in the form of an image, which is best understood visually as opposed to numerically, it is then natural to instead present the explanation in the same way.

These visual explanations take the form of saliency maps, as shown in figure 2.6. Here, we display the saliency values of all pixels in a heatmap. Instead of considering the absolute values, we instead display the saliencies in relation to each other, such that the most important and least important pixel has colors on opposite sides of the colormap. In this way, it is easy and intuitive to see where the important regions of the image are, and we do not need to consider the absolute values of each pixel.

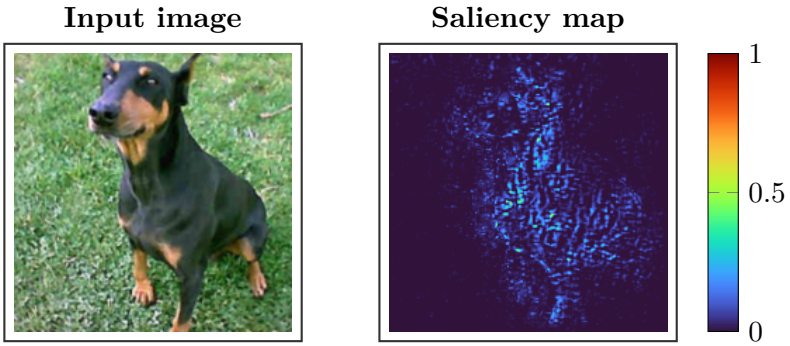

Figure 2.6: Figure showing an input image and the corresponding saliency map, generated by an XAI algorithm which shows the important pixels for the predicted class (Rhodesian Ridgeback). The colorbar and heatmap show the relative saliency for all pixels, with 0 being the least important pixel and 1 being the most important pixel

Secondly, many methods which are designed for data tabular data (which have may have 10-20 features) are far too slow when the number of features may be over a hundred thousand. Methods such as LIME, Occlusion, or Shapley fall into this category, as they permute the input at a rate which scales with the number of inputs.

A solution to this is to segment the image into larger regions, which are treated as one. As opposed to asking "how does the prediction change if we change this pixel", we instead ask "how does the prediction change if we change this *region*". All pixels in a region are then awarded the same amount of importance. This drastically reduces the dimensionality of the problem, and allows us to use a much larger range of methods. The output of such methods is thus also a saliency map, but one where each pixel is not given an individual saliency score, but rather a score derived from the larger region in which it is contained.

Regions of pixels can be created in different ways. The simplest option is consider a window of a specific height and width, and slide this window over the image with a specific stride. By adjusting the stride and size, the number of dimensions can be adjusted. The benefit of such a simple approach is that the segmentation itself introduces no extra computation. However, a substantial downside is that each region can contain completely unrelated objects, because the regions are created without considering the underlying image. For example, if a 60 by 60 pixel region contains in its lower right corner a part of a dog, occluding this region could lead to a large change in confidence and thus high importance. However, it is not just the lower right corner that is awarded high importance, but the entire region, which may contain other objects or parts of the background which are not actually important.

By using more sophisticated segmentation methods, we can avoid this problem. Methods such as Simple Linear Iterative Clustering (SLIC) [21] create regions which can be considered more intuitive than simply using a rectangular sliding window. SLIC performs an iterative clustering of pixels, grouping similar pixels into larger regions called superpixels. The downside to this method is that the iterative, CPU-bound process introduces a considerable computational overhead. Figure 2.7 shows a comparison between the saliency map of the LIME algorithm applied to an image segmented using rectangular regions and to an image segmented using SLIC. As we can see, the rectangular segmentation means that the some of the grass in the background is given a high saliency, because it happens to be in the same region as the dogs head. Using

SLIC, there are few regions which contain both grass and the dog, and thus the high saliency of the dog's face does not affect other parts of the image.

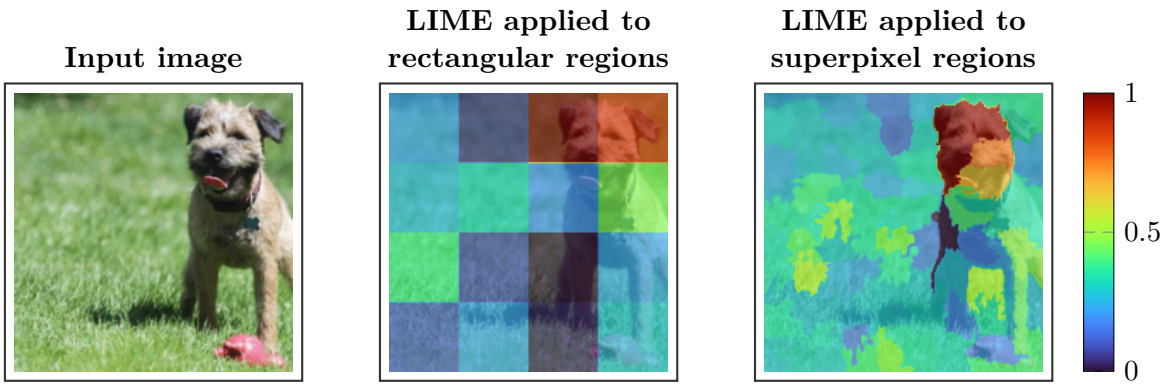

Figure 2.7: Figure showing a comparison between generating saliency maps using the LIME algorithm on rectangular and superpixel regions

2.4.4 Specific methods

The following section goes through several specific XAI methods. This serves two purposes: Firstly, by explaining how commonly used XAI methods function in more detail, it is easier to understand how XAI methods work in practice. Secondly, this section also covers the specific methods that will be analyzed and evaluated as part of the rest of the thesis. By thoroughly detailing the functions of these methods in this chapter, it becomes easier to analyse and explain the proposed methods in chapter 3 and the results in chapter 4.

Local Interpretable Model-Agnostic Explanations (LIME)

LIME [22] is a post-hoc, model independent XAI method. The method is built on the idea that while the decision function of a large neural network (or any other large model) might be far too complex to easily interpret, it can most likely be approximated quite well by a simpler function, as long as we only look at the feature space around a single data point. For example, we could approximate a large feed forward neural network with a simple linear regression model, which can be intrinsically explained due to its low complexity.

To create a locally interpretable model, we need a neighbourhood of data points around our point of interest. To do this, we can sample a number of points from our dataset and weigh them by their distance to our original point. This sampling can be done in many ways, for example by calculating a mean and variance for each feature and sampling from a normal distribution. For image data, we can create new points similar to the image by masking out different regions of the image [19]. The distance measure depends on the type of data we are dealing with. Regardless, the distance values are passed through a smoothing kernel which can be tuned to adjust the size of the "neighbourhood".

With these new data points, we can generate new predictions using the original, complex model. Thus, we now have a series of points, each with a weighting based on their distance to our original point, and each with a predicted score from our original

model. With such a dataset, we can train a simpler model, which will then approximate the complex model around the point of interest. By inspecting this simple model (for example the learned weights of a linear regression model, or the structure of a decision tree), we can learn approximately how the complex model functions in a region around this single data point.

Occlusion methods

Occlusion methods are a family of post-hoc model independent XAI methods. They function by masking different parts of the image and inspecting the change in output score. If an area leads to a large drop in softmax score for the predicted class when masked, this area must have been important for the network when making the prediction. The mask can be as simple as replacing all masked pixels with a single color, such as gray [23], or they could use more advanced inpainting methods using generative models, for example by replacing a masked tumor with generated healthy tissue.

Regardless of the mask, one can easily calculate the importance of any pixel for a prediction by calculating the average change in the output score for all masks which contain the specific pixel [24]. Occlusion methods have the advantage of being completely model independent, since they do not consider the internals of the model. However, the computation can be expensive, because we need to run a forward pass for each position of the mask on the image. Figure **??** shows the process visually.

Class Activation Mapping (CAM)

Class Activation Mapping (CAM) [25] is a model dependent, post hoc XAI method, which is used on Convolutional Neural Nets (CNNs). For a specific output node of a model (for example, the one denoting the presence of a specific class, such as "cat"), CAM outputs a heat map showing which areas of the input image contributed to this node. In this way, CAM gives a visual explanation to which parts of an image the model focused on when making a decision to classify an image to a specific class. This method is model dependent, because it requires a specific architecture in the final layers of the network to work.

CAM is a relatively simple method to understand. It exploits the fact that various convolutional layers of CNNs actually behave as object detectors, even when the training objective is classification [25]. As [11] explains, the earlier layers "extract elementary visual features such as oriented edges, end-points [or] corners", which can be used by subsequent layers to detect higher-order features. In this manner, the final convolutional layer will detect very high level visual features, combining the extracted information from all the previous layers. This layer is composed of several feature maps, where each map can be thought of as denoting the presence of some specific feature across the original image. The authors perform global average pooling (GAP) on these feature maps, giving a single value for each map, which is followed by a single dense layer and the Softmax activation function. In this way, each output node in the final layer is a weighted sum of all the global average pooled feature maps from the final convolutional layer. This means that we can represent the areas of the image which were used to perform the classification by performing the same weighted sum on the actual feature maps instead, which gives us a heat map which we can overlay on the original image (after upsampling the feature maps).

Figure 2.8 shows the process visually. From this we can see that the resulting Class Activation Map (bottom right) gives an intuitive explanation for why the image in the

top left gives a high score for the presence of the class "Australian Terrier".

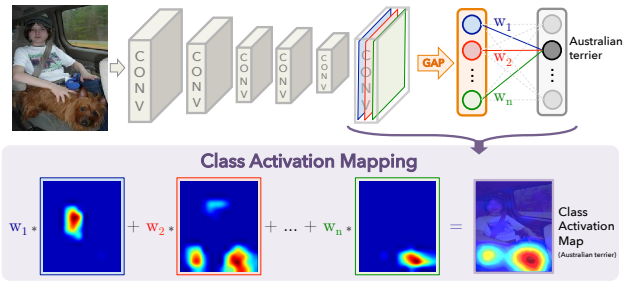

Figure 2.8: Figure taken from [25], showing the steps required to create a Class Activation Map

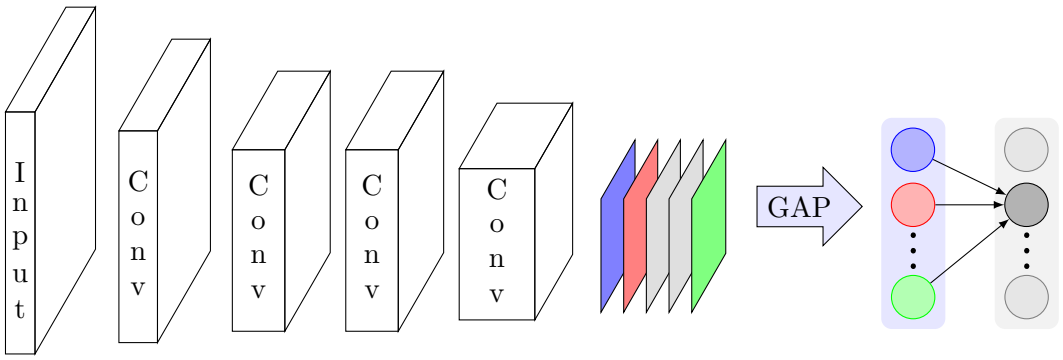

Figure 2.9: Figure showing a high level overview of how a CNN functions

Although CAM is an intuitive and effective method of visualizing the inner workings of a CNN, it has some downsides. Firstly, it is highly model dependent, requiring that the model only have a single dense layer after the convolutions. Although there are some SoTA models which only use a single dense layer, this still places a limit on what models can be used, or requires the simplification of models that use more than a single dense layer. [25, p. 4] notes a 1-2% drop in classification performance when performing this simplification. Secondly, the output of CAM is simply a weighted sum of all the feature maps after the final convolutional layer. As we move deeper in a CNN, we reduce the spatial resolution by downsampling, while increasing the number of channels (increasing the depth of the output while reducing the height and width). Because of this, the CAM will have a drastically lower resolution than the original image, often less than $10\,x\,10$, while the input image may be hundreds of pixels in both dimensions. Because of this, CAM can only show general areas, as opposed to pixel wise explanations.

Gradient Class Activation Mapping (GradCAM)

GradCAM [26] is an improvement on CAM, which generalizes the method to function with any CNN architecture, thus making the method much less model dependent and avoiding the performance drop incurred when simplifying the model with CAM. Instead of using the weights of a final layer to calculate a weighted sum of feature maps in the last convolutional layer, GradCAM uses gradients flowing from the relevant output node to the activation maps to calculate the weights for each feature map. Furthermore, the

authors prove that this method is a strict generalization of CAM [26, p. 5], so that no information is lost by using gradients instead of weights.

Like the simplicity of the CAM method, the calculation of the weights using the gradients is also quite simple, as seen in Equation 2.2.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\delta y^c}{\delta A_{ij}^k} \tag{2.2}$$

Here, c represents the index of the class we are interested in, k the index of the feature map, and i and j the width and height of the image. y^c is the element of the output vector y which corresponds to the class c, while A^k is the k'th feature map. Z is equal to $i * j$, and simply normalizes the sum. Thus, we are actually just performing global average pooling of the gradients of A^k with respect to y^c, which gives us a single value we can use as the weight for this feature map. Doing this for all feature maps for a specific class gives us all the weights we need to calculate a weighted sum, which we can upsample and visualize to get an explanation for the decision of the CNN.

Thus, GradCAM improves upon CAM by making the method less model dependent. However, the explanations are still the same low resolution, which may not be ideal in all cases.

Guided Backpropagation

GBP [27] is another XAI method which utilizes the gradients of the network to calculate saliencies. In this case, we do not stop at the final feature map, but backpropagate through the entire network to the input image. Simply backpropagating in this way produces a saliency map for the entire input image. However, [27] finds that by only backpropagating positive gradients through ReLU functions, they are able to produce "sharper visualizations of descriptive image regions than the previously known methods". Although the choice to simply neglect negative gradients when backpropagating lacks theoretical justification, GBP has been shown to be accurate and trustworthy compared to other methods [28, 29]. Compared to the previous methods, GBP differs in the sense that it produces a saliency value for every single input feature (every pixel, for example) as opposed to regions.

Integrated Gradients

Integrated Gradients [30] is another gradient based XAI method. The method was developed as part of an effort to create an XAI method which satisfies two axioms; sensitivity, and implementation invariance. Sensitivity states that if an input and a baseline differ in only one input feature, and have different outputs, then this input feature should have non-zero saliency. [30] shows that gradients violate this property because "the prediction function may flatten at the input and thus have zero gradient despite the function value at the input being different from that at the baseline". Implementation invariance states that if two models are functionally identical (their outputs are equal for all inputs), then their attributions should be identical as well. Gradients satisfy this property, but methods which modify the gradients, such as GBP and Layer Relevance Propagation (LRP) [31], break this axiom.

Both these axioms represent desirable qualities for an XAI method; we don't want our attribution method to miss any features which lead to a change in the model prediction, and we don't want the internal structure of the model to affect the attribution if it

does not affect the output score in any way. Thus, it is problematic that there are few methods which satisfy both criteria. [30] finds that by integrating the gradients of the network in the straight line path between a baseline and the input, both these axioms are satisfied. Mathematically, the saliency for a specific input feature $\mathbf{x}_i$ is defined as follows, given an input $\mathbf{x}$, a baseline $\mathbf{x}'$ and a deep learning model f:

$$\text{IntegratedGradients}_i(\mathbf{x}) := (\mathbf{x}_i - \mathbf{x}'_i) \times \int_{\alpha=0}^{1} \frac{\delta f(\mathbf{x}' + \alpha \times (\mathbf{x} - \mathbf{x}'))}{\delta \mathbf{x}_i} d\alpha \tag{2.3}$$

By cumulating the gradients of the network at all points between the baseline and the input, [30] manages to combine the implementation invariance of gradients with the sensitivity of techniques like LRP. In addition, they prove that if the model f is differentiable "almost everywhere", then the sum of all attributions is equal to the difference in output between the baseline and the input:

$$\sum_{i=1}^{n} \text{IntegratedGradients}_i(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}') \tag{2.4}$$

If we then choose a baseline which has $f(\mathbf{x}') \approx 0$, we see that integrated gradients is equivalent to distributing the output prediction amongst the input features, a very desirable interpretation for an XAI attribution method.

2.5 Out-of-Distribution Detection

This section discusses OOD detection, the field which attempts to tackle the second problem discussed in the introduction; that ML models have significantly worse performance on OOD data points and will often "fail silently", making completely wrong predictions with apparent high confidence [32]. OOD detection is a developing field, and still in an initial stage [33]. In 2017, [12] proposed a baseline OOD detection method. This section will discuss this method and the methods which follow it.

2.5.1 Motivation for Out-of-Distribution Detection

When training a model using supervised learning, we implicitly use the "closed-world assumption", which means that we assume that test data will be drawn from the same distribution as the training data [14]. However, when a model is deployed, the data we see may not obey this assumption. Without OOD detection, the model will behave in the exact same way when encountering OOD samples or in distribution (ID) samples, and may even claim to be highly confident in its prediction although the sample is far away from the distribution of the training data [34, p. 1]. In any system where models make high impact decisions, this is a huge problem. We do not want a model to claim high confidence when predicting if a woman has lung cancer if the model has only been trained on men, nor do we want a model to attempt to classify a rare disease that was not part of the training data. Thus, OOD detection methods are necessary, so that OOD samples can be caught before the model makes a prediction and dealt with correctly.

Intuitively, one might assume that distinguishing ID and OOD samples from each other can be solved by simple binary classification using a dataset of ID samples and one of OOD samples. Indeed, if one has sufficient amount of high quality OOD samples, this can be done. However, this can be difficult to obtain in practice [14, p. 15], thus requiring more sophisticated methods of OOD detection.

2.5.2 Semantic versus covariate shift

The first distinction to make in OOD detection tasks is whether an OOD sample is OOD because of *semantic* or *covariate* shift. Semantic shift refers to samples with different classes than the ones the model is trained on. A picture of a giraffe would represent a semantic shift for a model trained to differentiate between different breeds of dogs, as a giraffe does not belong to any breed of dog. Covariate shift refers to samples which come from a different distribution while still belonging to one of the classes of the original data set. An image of a Beagle puppy could represent covariate shift for a dog breed classifier despite Beagle being one of the classes, if all ID images were of adult dogs. Likewise, an image of a dog in a dark room could represent covariate shift, if all the ID images were of dogs outside, in well lit conditions. Figure 2.10 shows this distinction visually. Here, we can easily see the difference between covariate and semantic shift; covariate shifted images come from the same classes as ID samples, while semantically shifted images come from completely unknown classes.

Figure 2.10: Figure showing in-distribution, covariate shifted and semantically shifted images for an imagined dog breed classifier. Images are taken from the ImageNet dataset

The detection of semantic shift, as opposed to covariate shift, is the main focus of most OOD detection tasks [14]. In many applications, it is expected that the model should be able to generalize its prediction to covariate-shifted data, and therefore the focus is on detecting semantic shift. However, the field of medical image classification is one where detecting covariate shift is also important, as the model should only make predictions on data points which are very similar to its training data [14].

Given that the detection of semantic shift has been the main focus of most OOD literature, my work will primarily deal with semantic shift as well. Thus, unless otherwise specified, when I refer to OOD data points, I mean data points which are semantically shifted, i.e that come from another class than those the model has been trained on.

2.5.3 Benchmarking

The performance of an XAI is hard to quantify, because the quality of an explanation is not easily reduced to a number. For OOD detection, performance is much easier to measure, as the problem can be described as a binary classification problem, with OOD and ID samples as the positive and negative class, respectively. Thus, we can calculate many different metrics and compare methods against each other. For OOD methods, the two most common metrics to report are FPR95 and the AUROC (see section 2.3.1). It is common to use ImageNet or CIFAR as the ID dataset, and calculate FPR95 and AUROC on other datasets which contain no overlapping class labels. When selecting OOD datasets, it is common to differentiate between **Near-OOD** and **Far-OOD**. Far-OOD samples are samples which are drastically different from the ID samples, while Near-OOD samples only differ slightly. For a cat-versus-dog classifier, a tiger and a wolf would represent Near-OOD semantic shift, while a plane and a car would represent Far-OOD semantic shift. As one might expect, detecting Near-OOD samples is much harder.

In 2021, [14] defined a generalized OOD detection framework, and in 2023 [6] introduced a comprehensive benchmark called OpenOOD, which evaluates all relevant OOD methods under this framework. Prior to this work, different methods were tested on different datasets, with different image preprocessing procedures, and with other externalities which inhibited effective comparison between methods [6].

OpenOOD includes 11 different benchmarks across Anomaly Detection, Open Set Recognition and OOD detection, three fields which are very closely related. Of these, 6 benchmarks are used to test methods for OOD detection, which are the ones I will be concerned with. Each benchmark is defined by an ID dataset, with 6 or more corresponding OOD datasets, separated into Near-OOD and Far-OOD. For each benchmark, AUROC, AUPR and FPR95 is reported over all OOD datasets, and Near-OOD AUROC is used to rank methods against each other.

2.5.4 Methods

This section will follow the same outline as section 2.4; firstly, the overarching categories of methods will be discussed, followed by a more detailed look at a selection of specific methods within the field.

The field of OOD is separated into four categories of methods [14]:

- Classification-based methods
- Density-based methods
- Distance-based methods
- Reconstruction-based methods

All methods can also be categorized by whether they are post-hoc or training based. Post-hoc methods take an already trained network and attempt to extract information which separates ID and OOD samples out of the network during inference. These methods have the obvious advantage that they can work out of the box with large pre-trained network without requiring expensive training from scratch. Training based methods train the network in ways which maximize the difference between ID and OOD samples. These methods do not necessarily require OOD samples, but can train using

auxiliary loss functions which amplify the differences in network behaviour when faced with OOD data as opposed to ID. Regardless, these methods come with a much higher computational requirement than post-hoc methods, as they require training from scratch or at least retraining using the new loss criterion.

Given the fact that post-hoc methods can be applied to trained networks out of the box, it is quite common to combine both post-hoc and training strategies to achieve the best performance.

Below follows a short explanation of each the four categories mentioned above.

Classification-based methods

Classification-based methods usually use the softmax score or logits of a model to attempt to distinguish OOD and ID samples. [12] made the observation that while the softmax score may be a poor indication of the actual confidence of the model on a single data point, it is still higher on average for ID samples as opposed to OOD samples. By using this simple distinction, they created a baseline model which separated OOD and ID samples. Using input perturbations and temperature scaling, [13] further improved on this method, by amplifying the difference in softmax score of ID and OOD data.

More generally, classification-based methods do not need to use the softmax score, but may attempt to find any metric which separates the distribution of ID samples from OOD samples. Figure **??** shows the probability density for an unspecified metric for both OOD and ID samples. The goal of classification-based OOD detection is to find metrics or training methods which make these probability densities have as little overlap as possible, such that they are easily separated by a threshold. The threshold, often denoted as δ, is a parameter that needs to be set at a specific value based on the values of a validation ID and validation OOD data set. Often, one sets the δ such that a certain percentage of OOD samples in the validation set are correctly detected, for example 95%.

There are several SoTA methods which utilize a classification-based approach, and these make up a large part of the representative methodologies for OOD detection today [14, p. 8]. As such, I shall devote the majority of section 2.5.5 to classification-based methods.

Density-based methods

Density-based methods explicitly try to model the in-distribution [14], which is then used to detect outliers in low likelihood regions. Although the idea is intuitive, learning the distribution of the data set can often be prohibitively expensive, and thus these methods often lag behind classification-based methods [14].

Distance-based methods

Distance-based methods attempt to detect OOD samples by calculating their distance to ID samples. Many different distance measures are used, such as Mahalanobis distance to estimated Gaussian distributions, cosine distance to the first singular vector of the data set or Euclidean distance in an embedding space.

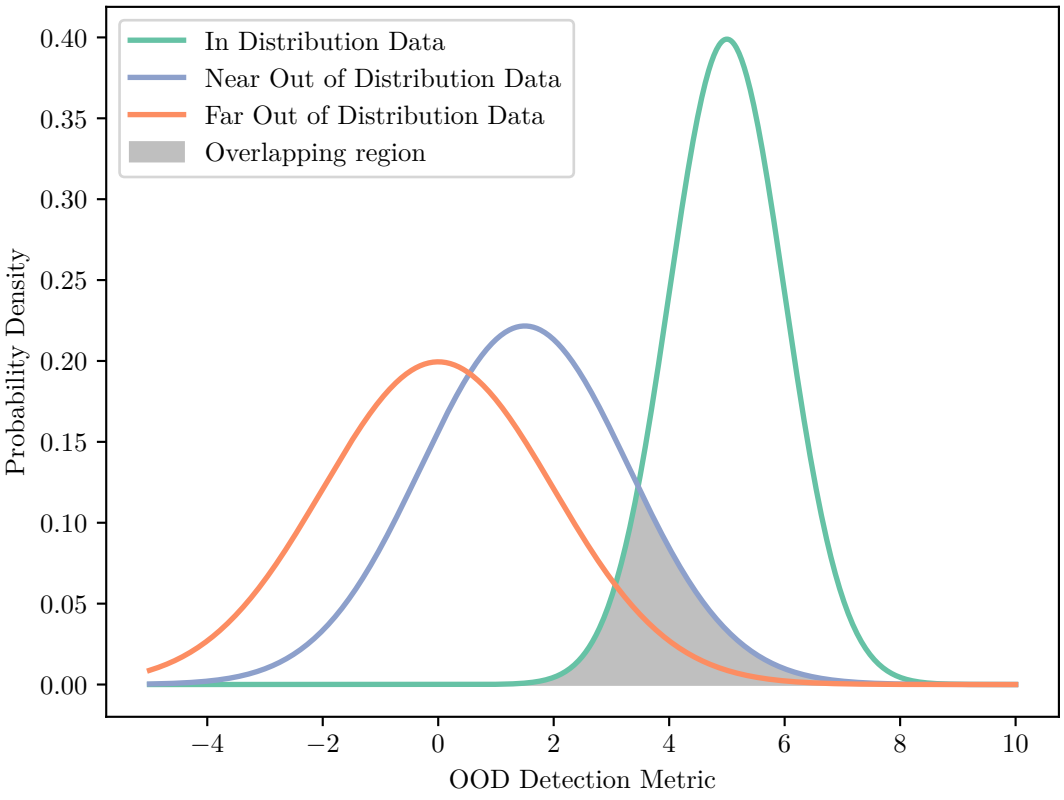

Figure 2.11: Graph showing the distribution of hypothetical ID, Near-OOD and Far-OOD data for an unspecified metric. The shaded region shows the overlap between the ID and OOD samples.

Reconstruction-based methods

Reconstruction-based methods are based on encoder-decoder frameworks, where the core idea is that the model will be much worse at reconstructing OOD data than ID. By measuring the reconstruction loss, we can detect OOD samples.

2.5.5 Specific methods

Below follows a more detailed look a selection of specific OOD detection methods.

Baseline models

The baseline model created by [12] is extremely simple, yet effective. It simply compares the softmax score the predicted class to a threshold, and labels it as OOD if it falls below this threshold. Let us assume we have a model $f : \boldsymbol{x} \to \mathbb{R}^C$ that takes an input $\boldsymbol{x}$ (which may be an image, a vector of values, or something else) and returns a vector of logits with length equal to the number of classes C. If we define a softmax score function

$$S_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)}, \tag{2.5}$$

then the OOD detector has the following simple form, given a threshold δ:

$$g(\boldsymbol{x}; \delta) = \begin{cases} \text{in} & \max_i S(\boldsymbol{x}) \geq \delta \\ \text{out} & \max_i S(\boldsymbol{x}) < \delta \end{cases}, \tag{2.6}$$

As explained previously, the δ must be set by the user of the system based on results from a validation set of ID and OOD data.

This method reasonably well, because the softmax scores for ID data generally is higher than for OOD data. Two years later, [35] showed that this baseline could be improved in settings with larger datasets by forgoing the softmax normalization and instead only looking at the maximum logit, with the even simpler form

$$g(\boldsymbol{x}; \delta) = \begin{cases} \text{in} & \max_i f(\boldsymbol{x}) \geq \delta \\ \text{out} & \max_i f(\boldsymbol{x}) < \delta \end{cases}, \tag{2.7}$$

Perhaps surprisingly, both these methods are quite good at detecting OOD samples. However, there are still many ways to improve these methods, as will be shown in the following sections.

GradNorm

As opposed to using the feature or output space, GradNorm [36] attempts to use the gradient space of a network to calculate OOD-ness. They find that the gradients of the weights actually contain valuable information that allows for effective separation of ID and OOD samples, and perform ablation studies which show that this methods outperforms many other methods, including the previously mentioned ODIN and Energy methods.

The gradients are calculated with regards to the Kullback-Leibler divergence between the softmax values and a uniform distribution. An important distinction from other methods is that all the softmax values are used, as opposed to the *softmax score* which would be only the score of the predicted class. Thus, this method captures information about the uncertainty across all categories, as opposed to just the most likely class [36, p. 3]. Once the gradients have been calculated, the threshold is simply done on the L_p-norm of these gradients, giving us the following thresholding function [36]:

$$S(x) = \| \frac{\partial D_{\text{KL}}(u \parallel \text{softmax}(f(x))}{\partial w} \|_p \tag{2.8}$$

As shown in figure 2.12, we see that the gradient norms are consistently lower for OOD data (gray) than ID data (blue).

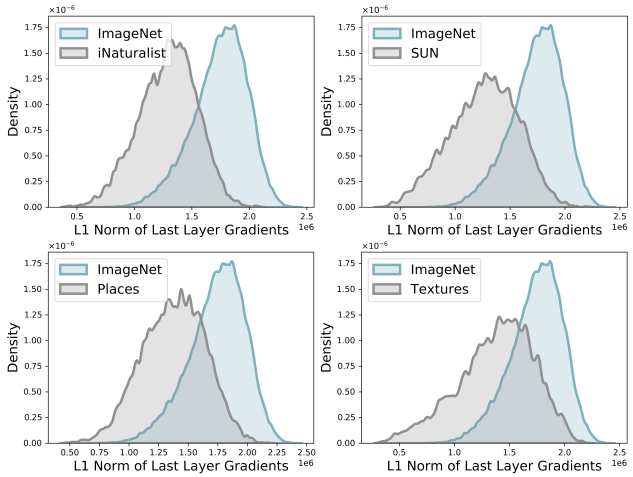

Figure 2.12: Figure taken from [36], showing the difference in gradient norms between ID and OOD data

[36] find that it is sufficient to only calculate the gradients for the last layer of the network, and that the L_1-norm performs the best, as it weights all gradients equally, as opposed to higher norms which place more importance on larger values.

In their mathematical analysis, they show that GradNorm captures joint information from both the feature and output space. By decomposing the L_1-norm of gradients of weights of the last layer with regards to the Kullback-Leibler divergence, they reach the following equality:

$$S(\boldsymbol{x}) = \frac{1}{CT} \left(\sum_{i=1}^{m} |x_i| \right) \left(\sum_{j=1}^{C} \left| 1 - C \cdot \frac{e^{f_j/T}}{\sum_{j=1}^{C} e^{f_j/T}} \right| \right) \tag{2.9}$$

From this, we see that $S(\boldsymbol{x})$ is a product of a factor which is simply the L_1-norm of the feature vector $\boldsymbol{x}$, and another term which captures information about the softmax values in the output space.

Virtual Logit Matching (ViM)

[15] attempts to improve OOD detection by calculating a score based on the feature, the logit and the softmax probability at once, as opposed to just one of them. By looking

at all three elements in conjunction, they see an increase in performance over models which only rely on a single input source (such as the previously mentioned ODIN).

The reasoning behind not just looking at the logits or softmax probability is that there is a lot of information that is lost when going from features to logits [15]. Once we project the features down to logits, we have only class dependent information, and have lost the class agnostic information which is contained within the features. To show how this information is lost, the authors give an example based on null space analysis [37]:

Let us assume that we have a simplified network with only a single layer. Then, we have $\hat{\boldsymbol{y}} = W\boldsymbol{x}$, where $\hat{\boldsymbol{y}}$ is the vector containing the logits, $\boldsymbol{x}$ is the feature vector of the input (with an additional 1 for the bias term) and W is the matrix containing the weights and biases transforming the feature vector into logits. A null space $\text{Null}(W)$ of a matrix W is the set of all vectors that map to the zero vector, such that $W\boldsymbol{a} = \boldsymbol{0} \iff \boldsymbol{a} \in \text{Null}(W)$. The null space of a matrix may be trivial (empty), but a matrix which projects vectors to a lower dimension have non-trivial null spaces. Given that the final layer of a neural network projects down to logits, which are the same dimension as the number of classes, this will almost always be the case. Because of the distributivity of matrix multiplication, we have the following:

$$W(\boldsymbol{x} + \boldsymbol{a}) = W\boldsymbol{x} + W\boldsymbol{a} = W\boldsymbol{x} + \boldsymbol{0} = W\boldsymbol{x} \tag{2.10}$$

The vector $\boldsymbol{x}$ can be decomposed into $\boldsymbol{x}^W + \boldsymbol{x}^{\text{Null}(W)}$, where $\boldsymbol{x}^W$ is the projection of $\boldsymbol{x}$ onto the column space of W and $\boldsymbol{x}^{\text{Null}(W)}$ is the projection of $\boldsymbol{x}$ onto the null space of W. It follows from this and equation 2.10 that when going from features to logits using the projection $W\boldsymbol{x}$, we lose all information contained in $\boldsymbol{x}^{\text{Null}(W)}$. [37] shows how this can be exploited by adversarial methods, by creating images with added noise derived from the null space of a matrix within the network, which are classified as if the noise was not present, despite having no resemblance to the original image. See figure 2.13.

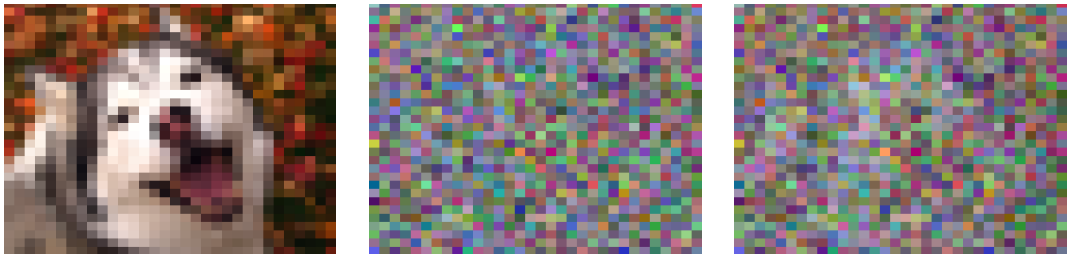

Figure 2.13: Image taken from [37]. Left: Original image. Center: Additive null space noise. Right: Final image, indistinguishable from original image according to the network the noise in the center column is sampled from.

From this, we can see that potentially large amounts of information can be lost when going from features to logits. Using this information, it is also possible to perform OOD detection, as shown by [37]. Another method which uses the features performs Principal Component Analysis (PCA) and looks at the residual information lost when using the first N principal components [38]. However, the information in the features is still class agnostic, and [15] aims to go beyond using just one input source and combine several elements of the network.

To do this, they propose using a *Virtual Logit*. The Virtual Logit is calculated as follows: First, they center the feature space, so that "it is bias free in the computation of logits" [15]. They then perform PCA as in [38], and calculate the residual of $\boldsymbol{x}$ with regards to the principal components, which is the projection $\boldsymbol{x}$ onto the null space of

the principal subspace P. The residual represents the information lost when using the projection P.

$$\text{Residual}(x) = ||\boldsymbol{x}^{\text{Null}(P)}|| \tag{2.11}$$

This value is scaled based on the average values of the maximum logit across the dataset, and is appended to the rest of the logits as a Virtual Logit:

$$l_0 := \alpha||\boldsymbol{x}^{\text{Null}(P)}|| \tag{2.12}$$

This now takes part in the computation of the softmax values, and thus is affected by the size of the rest of the logits. They call the softmax value of the Virtual Logit the *ViM score*. In this way, the ViM score represents the size of the residual in comparison with the predictions of the model. If the model is very confident, then the norm of the residual will be small in comparison, and the ViM score will be low. If the residual is very large, the ViM score will be higher, and more indicative of an OOD sample. In this way, [15] have combined information from the feature, the logit and the softmax probability level to perform OOD detection.

COMBOOD

`COMBOOD` [39] is another OOD detection method which combines information from different sources to increase performance. Unlike VIM, which combines different internal signals from the network, `COMBOOD` combines information from two different metrics calculated from the feature space. The two metrics are Mahalanobis distance and nearest neighbour distance, which have both seen decent performance on their own [40, 41]. [39] builds on these works by showing that their combination into one single score can increase performance far above either one. Indeed, the performance of `COMBOOD` is State-of-the-Art, being the highest performing OOD detector on the ImageNet200 and ImageNet1K benchmarks in the OpenOOD framework.[3]

To understand how `COMBOOD` works, we must first understand how Mahalanobis and nearest neighbour distance work as OOD detectors. Mahalanobis distance assumes that the features generated by the network are distributed as a multivariate Gaussian, and calculates the distance based on the mean and standard deviations of this Gaussian. This method is a generalization of calculating the Z-score for a univariate Gaussian, and has the following form, given a mean $\boldsymbol{\mu}$ and covariance matrix Σ, calculated over the ID training set:

$$\text{Mahalanobis}(\boldsymbol{x}) = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2.13}$$

Nearest neighbour distance has the advantage that it does not impose any assumptions about the distribution of the feature space, and is thus non-parametric. The k-nearest neighbour distance, is simply the distance from a given datapoint to the k'th nearest neighbour in feature space. [40], which used this distance metric for OOD detection, normalized the features and used Euclidean distance as the distance metric, giving us the following equation:

$$\text{NN}(\boldsymbol{x}) = \|\boldsymbol{x}^* - \boldsymbol{z}_k\|_2, \tag{2.14}$$

where $\boldsymbol{x}^*$ is the normalized feature $\boldsymbol{x}$ and $\boldsymbol{z}_k$ is the k'th nearest neighbour from the ID training set.

[3] https://zjysteven.github.io/OpenOOD/

COMBOOD combines these metrics by computing "confidence scores" for each distance, which are then added together to produce a final score. In addition, they find that by using different feature extracting methods for each of the different methods, their combined performance can be enhanced considerably, concluding that "COMBOOD performs best when the nonparametric and the parametric components use different feature extraction strategies, penultimate layer embeddings for the former and global extrema of the features for the latter" [39].

2.6 Related work

In a broad sense, this work is about OOD detection. In this way, the field of OOD detection, and methods described in the preceding sections, can be thought of as constituting the related work. However, more specifically, I attempt to use XAI methods to enhance OOD detection performance, and thus we may look towards previous work which has attempted a similar combination.

While the combination of XAI and OOD detection has been explored in many previous works, the majority of them focus on explaining why a data point was marked as OOD, as opposed to using XAI to aid the detection itself. Works like [42], [43] and [44] are papers which combine XAI and OOD for this purpose. Within network security, XAI has been as part of anomaly detection systems to detect malicious or faulty network traffic. Here, it has been used to explain detections ([45], [46]), but also to aid in detection itself by inspecting the explanations of the detection system ([47], [48]). These methods thus use XAI to aid OOD detection in a similar manner to my work, however they are strictly focused on sequential network traffic data as opposed to images, and are mostly concerned with detection "unnatural" data samples such as intentionally malicious traffic or that generated by faulty equipment, as opposed to natural OOD data caused by semantic or covariate shift occurring when a model is deployed.

[7] is the most relevant previous work. Here, the authors explicitly aim to use XAI to improve OOD detection on images. They do this by looking at saliency maps produced by a GradCAM-based XAI method (section 2.4.4) during inference, i.e the heatmaps that explain which parts of the image was most influential to classify the image as a specific class. Using these heatmaps, they perform distance-based OOD detection (section 2.5.4): By collecting all explanations for each image in the ID dataset, they are able to construct archetypical explanations, and can make clusters of explanations. To perform OOD detection, they simply compare the explanation of a new data point to the clusters of archetypical explanations, and mark it as OOD if it has a distance which is over a certain threshold.

This method performs decently on toy datasets, achieving scores similar to SoTA methods when using *Fashin MNIST* as ID and *MNIST* as OOD. However, this method fails catastrophically in more complicated scenarios, achieving an AUROC score of only 52% on *CIFAR10* vs *SVHN*, which is only marginally better than pure guessing and far below any other method. The paper thus ends with the authors concluding that "OoD detection approaches that are specifically designed for the purpose achieve in general better detection scores at the cost of an additional computational burden in the model's construction" [7].

For more potential related work, we can look to OpenOOD ([6]), which aims to provide a comprehensive benchmark of all relevant methods in the field of OOD detection. Out of all 41 OOD detection methods included in this benchmark, there are

no methods which use XAI. However, as many XAI methods utilize the gradients of the network to generate saliency values, we could also consider OOD detection models which utilize gradients in some form as tangentially related to this thesis. In this regard [36] is related, as they utilize the norm of the gradients of the network with respect to the Kullback-Leibler distance between the outputs and a uniform distribution.

From the absence of any relevant method utilizing XAI in OpenOOD and from the poor results of [7], we can see that the potential for a truly effective OOD detection system using XAI has not been fully realized in any previous work.

2.7 Summary

In this chapter, I have given a thorough introduction.

Chapter 3

Methodology

As shown in the preceding chapter, there exists a plethora of XAI methods, which exploit gradient information, differences in output scores when altering model inputs, marginal contributions of input features, as well as many other intricacies of deep learning models. The core idea of this master thesis is that these methods, in their attempt to explain a model decision, may also inadvertently extract information which is valuable for OOD detection. Thus, there may be an unexplored potential in these methods to function not just as explanations, but also as classifiers which allow us to separate ID and OOD data. Intuitively, we might expect the explanations to be more spread out on OOD images, given that there are (by definition) no objects of interest in the image that the model can definitely be said to focus on. In contrast, we might expect an explanation on an ID image to be more focused on a specific area, which contains an object of interest. Furthermore, given that saliency methods give a numerical value to each region of the image, we might be able to extract information about the "OOD-ness" of an image by inspecting the magnitudes of these values. Intuitively, it may be the case that such values should be lower for OOD than ID, reflecting the higher uncertainty present in OOD data.

3.1 Proposed XAI methods for OOD detection

With the preceding intuitions in mind, I present three proposed methods for OOD detection which utilize XAI methods as part of their detection pipeline.

3.1.1 Stand-alone saliency method: Aggregate of Saliency

As we have seen from section 2.6, there has been little research into using explanations for OOD detection, aside from the work of [7]. Thus, I begin by presenting a simple baseline method which uses saliency values generated by XAI methods to calculate an OOD score.

As mentioned previously, the field of OOD detection started with the simple baseline introduced by [12], which simply uses the MSP (i.e the confidence score of the predicted class) as a way to measure OOD. Later, it was shown that using the MLS could also serve as an effective baseline. As such, I propose a similarly simple baseline when using explanations for OOD detection. The analogue of the maximum logit in an XAI context can reasonably be said to be the explanation generated of the predicted class (the class corresponding to the maximum logit). As explained previously, this explanation will take the form of an $n \times m$ saliency map. This saliency map is not a single scalar value,

and does thus not make a suitable OOD score by itself. Instead, we may perform some form of aggregation on the saliency map (such as taking the mean, the vector norm, the variance or some other form), and use this as the OOD score.

The intuition for this method is informed by the fact that there are many forms of aggregation over saliencies which one might reasonably expect to be different for ID and OOD data. As an example, let us consider the implications of aggregating in some way which captures the magnitude of the saliencies, such as the *mean*, the *vector norm*, or the *max value*. When we generate a saliency map using the predicted class, the XAI saliency method attempts to calculate a measure of importance for each region of the input image, with regard to this class. For ID data, as long as the model predicted correctly, we know that there really are regions of the input image which contain the predicted class. If we instead are looking at semantically shifted OOD data, we know that no input image contains any of the ID classes. Thus, when a neural network makes a prediction on such a data point, it will always be wrong, because it will always predict one of the ID classes. By generating a saliency map of this prediction, we are asking a method to decide how each region contributed to a false decision. Given that there are no objects of the predicted class, in any region, we may reasonably assume that the saliency values are very different than in an ID case, where such objects actually are present.

To further justify this intuition, I will present a simple example scenario (figure 3.1) where one might expect the ID saliencies to be higher than the OOD saliencies. Here, I imagine a model which has been trained to differentiate between different breeds of dogs. In the first case, it is given an image of a dog, and a prediction of "English Foxhound" is made, which happens to be correct. Generating an explanation for this prediction, each region of the image is given a measure of importance, calculated using gradient information, differences in prediction score on counterfactual examples or by other means. As there actually is an English Foxhound in the image, we may expect that these methods generate saliencies which have a magnitude which is higher than if there was no dog present.

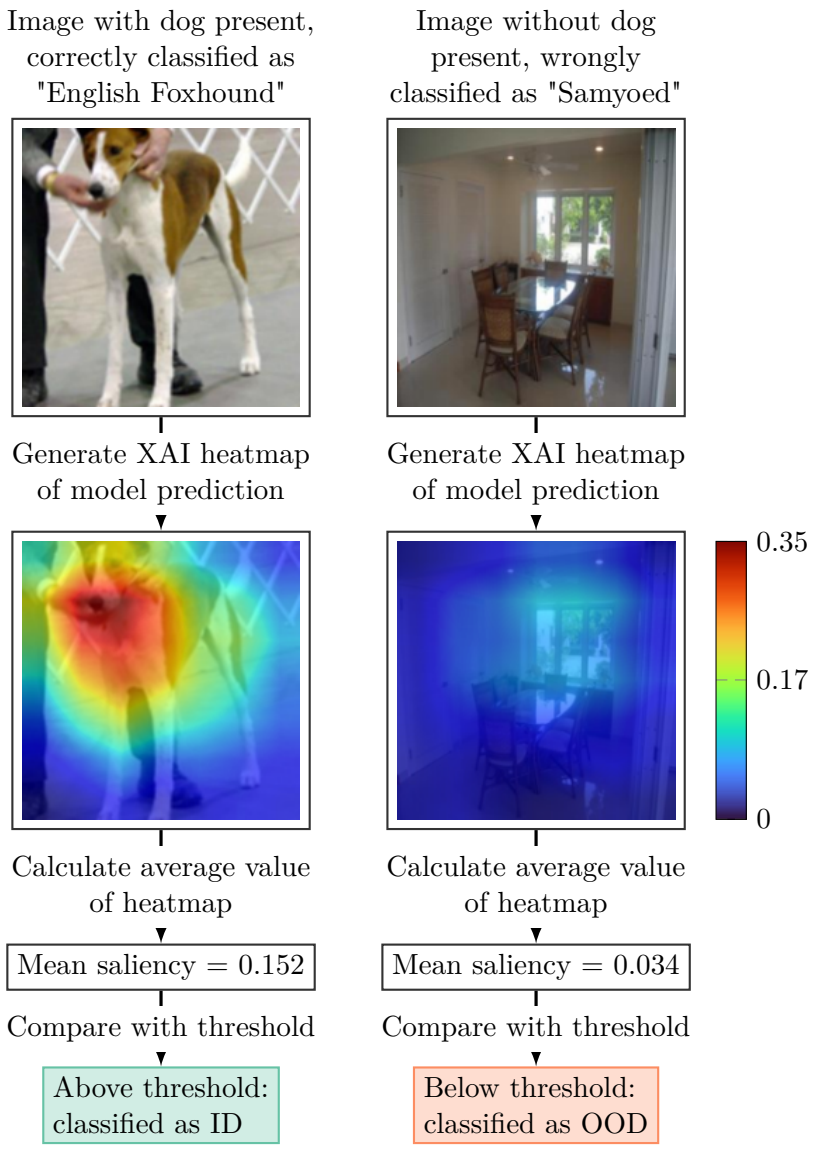

Figure 3.1: Figure showing the functioning of the Aggregate of Saliency OOD detector, using mean as the aggregation, in a hypothetical scenario where a model trained on images of dogs is shown an image with no dogs present. The heatmaps here show the maximum value of both saliency maps as dark red, reflecting the lack of normalization

In the second case, the model is given an image without any dogs present. Given that there is no class for images without dogs in them, the model will classify the image as one of the possible dog breeds. In this case, the model predicted the class of "Samoyed", a decision which can be considered essentially arbitrary. When a explanation is generated for this decision, the methods for calculating importance scores will most likely assign saliencies to most regions, given that no region contains a dog. As such, if we calculate mean saliencies for both the image with the dog and the one without, we expect the image with the dog to have a higher mean saliency. As long as we work with semantically shifted OOD data, it will always be the case that the prediction on OOD data is wrong, and thus we may also expect that the generated explanations in general output smaller saliencies.

On the contrary, we could also expect that OOD saliency maps have *higher*

magnitudes than ID saliency maps. As has been well documented, neural networks behave unpredictably when exposed to examples far from their training distribution [49–51]. Thus, it is not unreasonable to expect that explanations based on this unpredictable behaviour may also be unstable and unpredictable. This instability could lead to large outliers in saliency maps, which would give larger magnitudes when compared to ID data points. Figure 3.2 shows an imagined example scenario where this could be the case. Here, we imagine that the same dog breed classifier is shown two images: the first an image of a Border Terrier in a park, an scenario which we could assume is quite common in the training dataset. The second is a night time picture of a illuminated archway, an image which is very far away from the training distribution of the model, and contains many sharp changes in pixel intensity. In such a scenario, it is possible that a network will behave unpredictably, which may be reflected in the saliency map of the model. As explained in section 2.4.4, methods such as GradCAM, GBP and integrated gradients use gradient information from the network, which may be affected by sharp activations in the internals of the model.

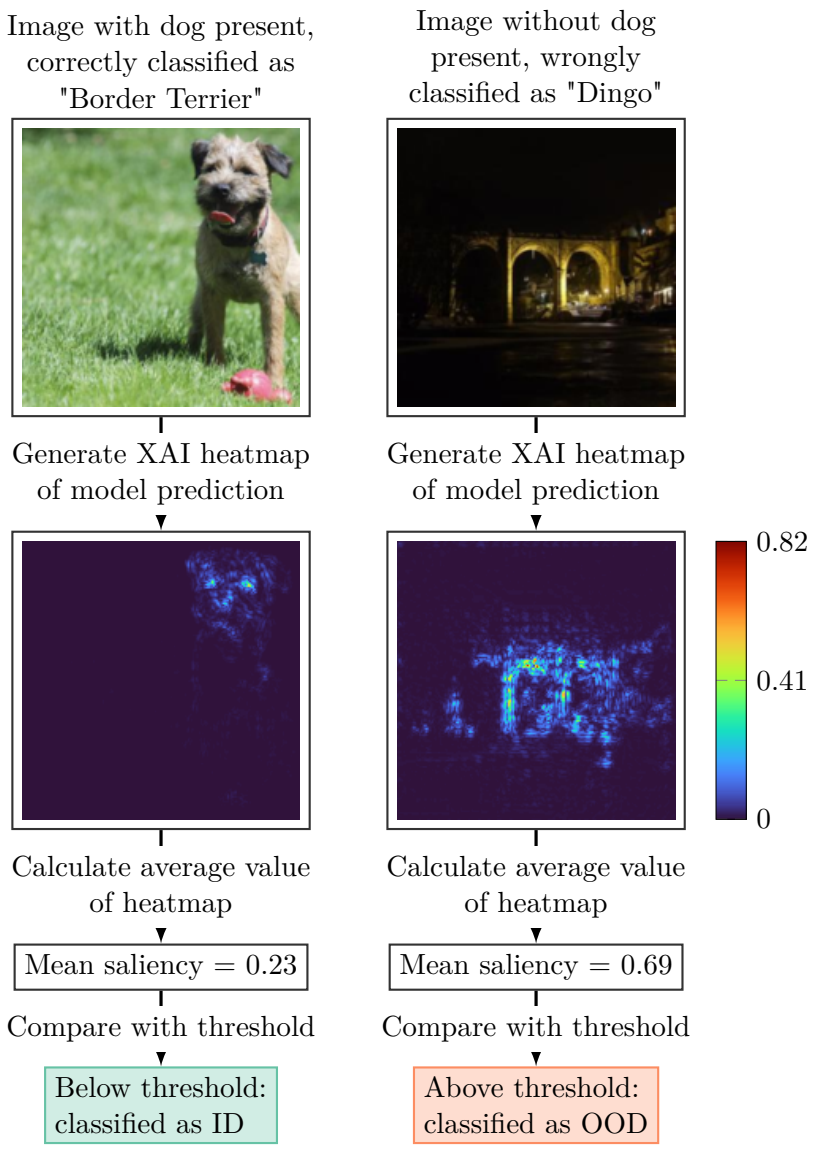

Figure 3.2: Figure showing the functioning of the Aggregate of Saliency OOD detector, using mean as the aggregation, in a hypothetical scenario where a model trained on images of dogs is shown an image with no dogs present. As opposed to the previous figure, here we assume that ID samples have lower mean values on average, and as such the thresholding is inverted.

Apart from aggregations which convey information about the magnitude of the saliencies, we may also expect the variation or dispersion of the saliencies to be different between ID and OOD data. We might expect the heatmap on OOD data to be less concentrated and more evenly spread out, given that there are no actual objects of interest present. This would give OOD data points a low variance, and ID data points a high variance. Alternatively, we might expect OOD saliency maps to have large outliers, which could give OOD saliency maps high variance.

Figure 3.3 shows a hypothetical scenario in which calculating the spread could be beneficial in OOD detection. Like in the previous case, we imagine a model trained on dogs, which is fed two images, one which contains a dog and one which does not. In this case, we do not care about the magnitudes, but instead only the spread of the values in relation to each other, and thus the heatmap is normalized. As we can see, in this

case the heatmap is more spread out in the explanation where there is no dog present than for the image with a dog. By calculating a suitable metric, such as the Relative Mean Absolute Difference (RMD), Coefficient of Variance (CV) or another measure of statistical dispersion, we can get a single number which represents how spread out the heatmap is, regardless of its magnitude. Using these values, we can define a threshold which is below most ID data and above most OOD data, allowing us to separate these distributions.

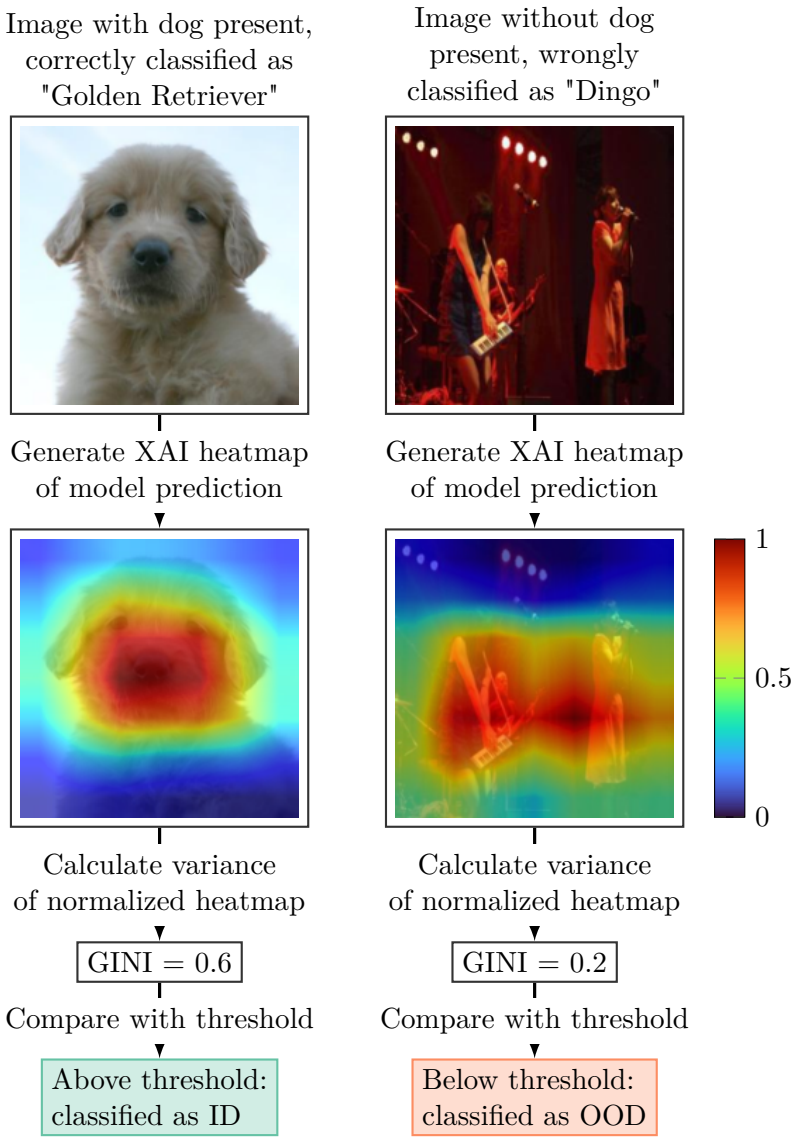

Figure 3.3: Figure showing the functioning of the Spread of Saliency OOD detector in a hypothetical scenario where a model trained on images of dogs is shown an image with no dogs present. The heatmaps are here based on the normalized values of each saliency map, meaning that magnitude information is ignored

With the intuition explained, we may now formalize the method. To define this detector mathematically, let us first define the necessary components. As in chapter 2, we assume we have a model $f : \boldsymbol{x} \to \mathbb{R}^C$. In this case, $\boldsymbol{x} \in \mathbb{R}^{D \times H \times W}$, i.e a D channel image of height H and width W. In addition, we define a XAI saliency mapping method

$s : (f, \boldsymbol{x}) \to \mathbb{R}^{H_s \times W_s}$. This function takes the model f and an input $\boldsymbol{x}$ and returns a H_s by W_s saliency map for the predicted class, i.e. the class corresponding to the highest logit. We also define an aggregation function $A : \boldsymbol{x} \to \mathbb{R}$, where $\boldsymbol{x}$ can be of any shape. Given the fact that we do not know whether the saliency aggregation of a specific XAI method s and a specific model f will be larger or smaller for ID data, we must also decide whether larger or smaller values should be considered ID. To do this, we can calculate the mean value of the OOD detector over a validation ID and validation OOD dataset, and compare their values. Requiring the presence of a validation set does not impose any actual limitations on the method, as a validation set is required for all OOD detection methods, to be able to set the threshold δ. If we denote the mean value of the aggregation over the ID validation dataset as μ_{id}, and over the OOD dataset as μ_{ood}, we can use $\text{sign}(\mu_{id} - \mu_{ood})$ to multiply the OOD detection score by 1 or -1, respectively. This ensures that the OOD detector follows the convention of the OOD detection field, which is that ID samples should have higher scores than OOD samples on a given OOD detection metric.

Thus, the OOD detector has the following form, given a threshold δ:

$$g(\boldsymbol{x}; s, A, \delta) = \begin{cases} \text{in} & \text{sign}(\mu_{id} - \mu_{ood}) \cdot A(s(\boldsymbol{x}, f)) \geq \delta \\ \text{out} & \text{sign}(\mu_{id} - \mu_{ood}) \cdot A(s(\boldsymbol{x}, f)) < \delta \end{cases} \tag{3.1}$$

An astute reader may note that aggregation functions are permutation invariant, meaning that all positional information from the two-dimensional saliency maps is lost when aggregating. This may seem strange, as it is primarily the positions of the different values that is important when using XAI methods for explaining model predictions on images. However, there is good reason to believe that for many image classification tasks, the positions of the points of interest in an XAI saliency map does not carry much discriminative potential. For datasets such as CIFAR10 and ImageNet200, there is a huge variety in the positions of the ground truth class (a dog may appear in the middle, the top right corner, or any other position and still be of the class 'dog'). As such, it is not a given that the removal of positional information will be massively detrimental. In fact, when [7] reflects upon the poor results of their saliency heatmap clustering for detecting OOD samples on CIFAR10, it is exactly this variability in position they highlight: "Indeed, the CIFAR10-C dataset does not afford the positional bias and low intra-class variability observed in the previous case studies: informative objects for the classes to be predicted appear in arbitrary parts of the image and have a high degree of compositional variability".

Given the exploratory nature of this thesis, it is reasonable to try many different forms of aggregation. Even when just considering the magnitude, it would be insufficient to just use the mean or maximum value of each saliency map, as each form of aggregation captures different qualitites about the underlying data. The maximum value, for example, is sensitive to outliers, which may be detrimental. The median value is far less sensitive to outliers, but given that one might expect ID data to have regions which are very important while most regions are relatively unimportant, this metric may not capture these high values. The vector norm and range are invariant to the sign of the saliencies, which means that if there are both large positive and negative values, these aggregates will return large scores. This is in contrast to the mean, median and third quartile, which will be lower if there are many negative values. In summary, we do not have the prerequisite knowledge about the distribution of saliency maps on ID and

OOD data to make an informed selection, and as such we should cast a wide net when choosing which forms of aggregation to use.

In the experiments, the following aggregations have been used.

Magnitude:

- Mean
- Median
- Vector norm
- Range
- Maximum
- Third Quartile

Statistical dispersion:

- Coefficient of Variance (CV)
- Quartile Coefficient of Determination (QCD)
- Relative Mean Absolute Difference (RMD)

These two categories follow the intuition explained above, and will test whether either the magnitude or statistical spread of ID saliency maps differ substantially from those of OOD saliency maps.

3.1.2 Saliency integrated into existing OOD detection algorithms

Given the poor results of [7], one might expect that saliency maps on their own are insufficient to differentiate ID and OOD data. [39] has shown that by combining the OOD detection scores of two different methods, the total performance can be improved considerably. Furthermore, [15] has shown improvements by considering the softmax, logit and feature space in tandem. Thus, there is reason to believe that if XAI saliency maps have some discriminatory capabilities, these could be combined with traditional method OOD, resulting in a performance gain. As such, I further present two methods which use both saliencies and traditional OOD detection metrics, to investigate if the addition of saliency values can enhance existing methods.

Maximum Logit + Saliency Aggregate

In 2024, [39] introduced `COMBOOD`. This OOD detector combines the OOD detection scores of two different distance metrics; Mahalanobis distance and nearest neighbour distance. Each distance metric uses a different feature extraction method, which allows the two methods to collect different forms of information and complement each other. The combination is done by a simple unweighted addition of the confidence scores computed from the log distributions of the two metrics. `COMBOOD` achieves SoTA performance, and is (at the time of writing) by far the best performing method on ImageNet200 in the OpenOOD benchmark. Based on these results, I propose a similar method for combining XAI-based and traditional OOD detection metrics. If it is the case that XAI methods extract OOD information from the model in ways which are substantially different from traditional OOD detection strategies, we may see improvements similar to those observed by [39].

For this method, the OOD score is a sum of the maximum logit and a saliency aggregate. However, due to the fact that both the logit and saliency values can be of any arbitrary magnitude, we must normalize them before summing if we want each part to contribute equally to the final score. Thus, we can sum the Z-scores of each metric instead. This ensures that the values of the maximum logit and the saliency aggregate are distributed in about the same way. To calculate the Z-scores, we can simply subtract the mean and divide by the standard deviation over an entire ID validation dataset, for each metric. Thus, we calculate the mean and standard deviations of the max logit over an ID validation set μ^{id}_{MLS} and σ^{id}_{MLS}, as well as the mean and standard deviation of the aggregate of saliencies μ^{id}_{Agg} and σ^{id}_{Agg}. In addition, we must calculate the mean value of the aggregation metric over a validation OOD dataset, as we do not know whether a given aggregation is higher or lower for ID data. We denote this value as μ^{ood}_{Agg}. We now have the necessary values required to define this method mathematically:

As in the previous section, we assume we have a model $f : \boldsymbol{x} \to \mathbb{R}^C$, an XAI saliency mapping method $s : (f, \boldsymbol{x}) \to \mathbb{R}^{H_s \times W_s}$, and an aggregation function $A : \boldsymbol{x} \to \mathbb{R}$. Let us denote $\text{sign}(\mu^{id}_{\text{Agg}} - \mu^{ood}_{\text{Agg}})$ as S, for readability. The OOD detector then has the following form, given a threshold δ:

$$g(\boldsymbol{x}; s, A, \delta) = \begin{cases} \text{in} & S \cdot \frac{A(s(\boldsymbol{x},f)) - \mu^{id}_{\text{Agg}}}{\sigma^{id}_{\text{Agg}}} + \frac{\max_i S(\boldsymbol{x}) - \mu^{id}_{\text{MLS}}}{\sigma^{id}_{\text{MLS}}} \geq \delta \\ \text{out} & S \cdot \frac{A(s(\boldsymbol{x},f)) - \mu^{id}_{\text{Agg}}}{\sigma^{id}_{\text{Agg}}} + \frac{\max_i S(\boldsymbol{x}) - \mu^{id}_{\text{MLS}}}{\sigma^{id}_{\text{MLS}}} < \delta \end{cases} \tag{3.2}$$

In fact, this detector can be simplified somewhat. Consider the following:

$$S \cdot \frac{A(s(\boldsymbol{x}, f)) - \mu^{id}_{\text{Agg}}}{\sigma^{id}_{\text{Agg}}} + \frac{\max_i S(\boldsymbol{x}) - \mu^{id}_{\text{MLS}}}{\sigma^{id}_{\text{MLS}}} = \tag{3.3}$$

$$S \left(\cdot \frac{A(s(\boldsymbol{x}, f))}{\sigma^{id}_{\text{Agg}}} - \frac{\mu^{id}_{\text{Agg}}}{\sigma^{id}_{\text{Agg}}} \right) + \frac{\max_i S(\boldsymbol{x})}{\sigma^{id}_{\text{MLS}}} - \frac{\mu^{id}_{\text{MLS}}}{\sigma^{id}_{\text{MLS}}} = \tag{3.4}$$

$$S \cdot \frac{A(s(\boldsymbol{x}, f))}{\sigma^{id}_{\text{Agg}}} + \frac{\max_i S(\boldsymbol{x})}{\sigma^{id}_{\text{MLS}}} - \left(S \cdot \frac{\mu^{id}_{\text{Agg}}}{\sigma^{id}_{\text{Agg}}} + \frac{\mu^{id}_{\text{MLS}}}{\sigma^{id}_{\text{MLS}}} \right) \tag{3.5}$$

Notice how all the values in the third term of the above summation are constants; they do not depend on $\boldsymbol{x}$. Thus, we can disregard these terms, as all they do is shift all outputs by a constant value. The final OOD detector thus has the following form:

$$g(\boldsymbol{x}; s, A, \delta) = \begin{cases} \text{in} & S \cdot \frac{A(s(\boldsymbol{x},f))}{\sigma^{id}_{\text{Agg}}} + \frac{\max_i S(\boldsymbol{x})}{\sigma^{id}_{\text{MLS}}} \geq \delta \\ \text{out} & S \cdot \frac{A(s(\boldsymbol{x},f))}{\sigma^{id}_{\text{Agg}}} + \frac{\max_i S(\boldsymbol{x})}{\sigma^{id}_{\text{MLS}}} < \delta \end{cases} \tag{3.6}$$

SaliencyVIM: Virtual Logit Matching with Saliencies

As a final proof-of-concept method, I propose adding saliency values directly into preexisting OOD detection models. For this, VIM [15] is a fitting choice, as VIM attempts to combine information from different sources, namely from the feature, logit and softmax probability space. As we can recall from section 2.5.5, VIM uses a "virtual

logit" which is calculated from the output features on the penultimate layer and appended to the original logits. As a proof-of-concept integration of saliencies into this method, we may append XAI saliencies to the features prior to the generation of this virtual logit. The virtual logit in VIM is calculated as the information lost when performing a PCA feature reduction on the features. By appending the saliencies, the PCA takes into account not just how the penultimate features vary together, but also the interplay between the penultimate features and the XAI saliencies, which may be substantially different between ID and OOD datasets. This increase in variance could increase the reconstruction loss when projecting OOD samples using the principal components calculated from the ID dataset, increasing the separability of ID and OOD data points. Because there is no equivalent to centering the feature space as described in [15] for saliencies, I keep the saliencies as they are while the features are centered.

To define this OOD detector mathematically, we follow the definition of [15] very closely. We assume we have a model $f : \boldsymbol{x} \to \mathbb{R}^C$ and a XAI saliency mapping method $s : (f, \boldsymbol{x}) \to \mathbb{R}^{H_s \times W_s}$. We further assume we can extract the penultimate features $h \in \mathbb{R}^K$, with a function $g : (f, \boldsymbol{x}) \to \mathbb{R}^K$. We denote these extracted features as $\boldsymbol{z}$, and the entire set of feature vectors over the ID dataset as $Z \in \mathbb{R}^{N \times K}$. As in [15], we calculate an offset $\boldsymbol{o} = -(W^T)^+ \boldsymbol{b}$, where W and $\boldsymbol{b}$ are the weights and biases which transform the features into logits. The feature matrix Z is transformed by this offset: $Z^\dagger = Z + \boldsymbol{o}$. In addition to this feature matrix, we also calculate saliencies over the ID dataset, and flatten them to produce a saliency matrix $S \in \mathbb{R}^{N \times (H_s \cdot W_s)}$. These two matrices are concatenated, giving us the matrix $Y = \left[Z^\dagger, S\right]$. We then perform and eigendecomposition on $Y^T Y$, as part of the PCA and in accordance with the method described by [15]:

$$Y^T Y = Q \Lambda Q^{-1} \tag{3.7}$$

The first D columns of Q make up the D-dimensional principal subspace P. D is a hyperparameter that must be tuned, and is usually some fraction of K, the amount of logits in the penultimate layer (for example, if the number of logits is 1024, D may be 512 or 256). For this thesis, D has been fixed at 256, to avoid the extra computational overhead required for hyperparameter tuning. Regardless of the value of D, the remaining $K - D$ columns make up the null-space of P. We denote matrix making up these remaining columns as R. With this matrix, we can calculate the residual of $\boldsymbol{x}$ onto P as $\boldsymbol{x}^{P^T} = RR^T \boldsymbol{x}$. The virtual logit then has the following form:

$$l_0 := \alpha ||\boldsymbol{x}^{\text{Null}(P)}|| = \alpha \sqrt{\boldsymbol{x}^T R R^T \boldsymbol{x}} \tag{3.8}$$

This is exactly the same as the virtual logit for VIM, with the only difference being that R is calculated from the concatenated matrix Y, which includes both saliencies and logits, as opposed to only logits. To make it clear that R no longer depends solely on the network f and the input data, but also on the choice of saliency generator s, we can denote R as R_s. As with VIM, α is calculated from the average virtual and maximum logit over the ID dataset before the OOD detector is put into use.

The OOD detection score is the softmax score of the virtual logit when appended to the rest of the logits:

$$\text{SaliencyVIM}(\boldsymbol{x}) = \frac{e^{\alpha \sqrt{\boldsymbol{x}^T R_s R_s^T \boldsymbol{x}}}}{\sum_{i=1}^{C} e^{l_i} + e^{\alpha \sqrt{\boldsymbol{x}^T R_s R_s^T \boldsymbol{x}}}} \tag{3.9}$$

Thus, the formal definition of the SaliencyVIM OOD detector is as follows, given a threshold δ:

$$g(\boldsymbol{x};\delta) = \begin{cases} \text{in} & \text{SaliencyVIM}(\boldsymbol{x}) \geq \delta \\ \text{out} & \text{SaliencyVIM}(\boldsymbol{x}) < \delta \end{cases} \tag{3.10}$$

Because we are no longer aggregating the saliencies, the choice of saliency method is no longer as free as with the previous two methods. When we wish to use the saliencies directly, we cannot use methods which output one value for every single pixel in the input image, as this would lead tens of thousands of dimensions over which to compute a Principal Component Analysis (PCA). This is computationally intractable. Instead, we must use methods which calculate saliencies over larger windows, such as occlusion, LIME or GradCAM.

3.2 Datasets

After having introduced the three novel OOD detection methods that I plan to test, I will now introduce the datasets that will be used. To best align this thesis with the broader OOD detection field, I will use the OpenOOD framework, and their collection of ID and OOD datasets. Each OOD detection benchmark is defined by its ID dataset, with corresponding OOD datasets. As described in [52], the four standard OOD detection benchmarks included in OpenOOD are CIFAR10, CIFAR100, ImageNet200, and ImageNet1K. Testing on all these benchmarks is not feasible in the time frame available for this thesis, as such I chose one CIFAR benchmark and one ImageNet benchmark. For this thesis, the datasets that have been used are CIFAR10 and ImageNet200.

3.2.1 CIFAR10

CIFAR10 [53] is a 10-class dataset for general object recognition [6]. This dataset is commonly used in AI research [54]. Each RGB image has dimensions 32×32, and there are 50 000 training images and 10 000 test images. The ten classes are as follows: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. OpenOOD presents a series of Near-OOD and Far-OOD datasets which are used in conjunction with this dataset for evaluation. Table 3.1 presents a short description of each dataset and the number of samples.

Dataset	Size	Description
CIFAR10	10 000	General objects
CIFAR100	10 000	General objects from same collection as CIFAR10
TinyImageNet	10 000	General objects
MNIST	10 000	Handwritten digits from 0 to 9
SVHN	10 000	Street view house numbers
Texture	10 000	Street view house numbers
Places365	10 000	Places, scenes, locations

Table 3.1: Hello

In addition, a sample of images from CIFAR10 and the corresponding OOD datasets are shown in figure 3.4.

Figure 3.4: Figure showing three example images from CIFAR10 and from the corresponding OOD datasets

Dataset	Size	Description
ImageNet200	10 000	General objects
SSB-Hard	10 000	General objects
NINCO	10 000	General objects
iNaturalist	10 000	Handwritten digits from 0 to 9
Texture	10 000	Street view house numbers
OpenImage-O	10 000	General objects

Table 3.2: Hello

3.2.2 ImageNet200

ImageNet is another ubiquitous image classification dataset, which is used in large majority of computer vision works. In contrast to CIFAR, ImageNet images are substantially larger, at 256×256[1]. This increase in dimensionality could be used to argue that ImageNet tasks are more realistic, as it is rare that one would use 32×32 images in practical applications. The ImageNet dataset contains 1000 classes of general objects, however there exists smaller versions such as ImageNet200, which I have chosen to use. As with the other ID datasets, OpenOOD contains a selection of OOD datasets which are chosen to test a model's ability to differentiate between different forms of OOD data. Table 3.2 lists these datasets, and gives a short description for each one.

In addition, figure 3.5 shows three example images for each dataset.

[1]It is common to center crop as part of preprocessing on ImageNet, and as such you will more often see the dimensions 224×224 mentioned in this thesis

Figure 3.5: Figure showing three example images from ImageNet200 and from the corresponding OOD datasets

3.2.3 Overview of testing environment

After having introduced the specific datasets, I will describe the testing environment that has been used, so that it is clear for the reader how the OOD detection metrics are calculated.

As we have seen, for a given ID dataset, we have a series of OOD datasets, which

are all semantically shifted in relation to the ID dataset. These OOD datasets are categorized into Near- and Far-OOD, depending on the degree of their semantic shift. The ID dataset is split into a train, validation and test set, while the OOD datasets are split into validation and test sets. A network is trained on the train ID set. First, one singular OOD validation set is chosen, and any hyperparameters that the OOD detection algorithm may have are tuned on this validation set and the ID validation dataset. After having trained the network and tuned the OOD detector, we now have every part needed to perform OOD detection and calculate performance metrics.

First, we calculate OOD detection scores over the entire ID test set. By convention, these are expected to be higher than for OOD samples. For example, if we use the MSP OOD detector as described in section 2.5.5, we would calculate the softmax score of the predicted class for all samples in the ID test set, and store them. Then, for each OOD test dataset, we perform the same calculation of OOD detection scores. To calculate AUROC and FPR95, each set of OOD test set scores is compared against the ID test set scores we calculated previously by denoting all ID samples as class 0 and all OOD samples as class 1. After having done this for each OOD test set individually, we have a series of Near- and Far-OOD AUROC and FPR95 scores. The average performance of our classifier on Near- and Far-OOD can then be calculated by averaging over these individual scores. Figure **??** shows a diagram of this process.

This process describes how the performance of an OOD detector can be calculated for a given set of ID and OOD datasets. However, when developing a new method, we cannot repeatedly perform this process on the same ID and OOD datasets, as we will then be biasing our results. Because of this, I have separated all datasets into meta validation and meta test sets. Specifically, all ID and OOD validation sets, and all ID and OOD test sets, have been split in two equally sized subsets, of which one half has been used during development and the other half has been used during the final testing. This ensures that the results reported in chapter 4 have not been biased by repeated testing on the same data.

3.3 Networks

While it may be interesting to see how these new methods function on different network architectures, the combination of several different novel OOD detection algorithms and XAI saliency methods, as well as two different datasets already presents a considerable amount of evaluation. Thus, to focus the thesis on comparing different OOD detection methods against each other, I believe it is best to fix other parameters such as network architecture. With this in mind, I choose to limit myself to the ResNet [55] family of neural networks. In particular, I use ResNet18 for all tests, either one trained on 32×32 images (for Cifar10), or one trained on 224×224 images (for ImageNet200).

Aside from CNNs, Vision Transformers also perform exceptionally on computer vision tasks, and achieve SoTA results in many settings [56]. On ImageNet, they are even dominant, and the top 10 models when considering (top-1) accuracy are all based on vision transformers, as opposed to CNNs.[2] As such one might question the choice of a CNN model, when newer and better models have been developed.

However, given that a large part of the XAI methods have been developed in the CNN paradigm, they are not easily adapted to vision transformers. Methods such as GradCAM and LRP exploit specific parts of CNN architectures when generating

[2]https://paperswithcode.com/sota/image-classification-on-imagenet

explanations [57], and are thus difficult to use with different architectures. To be able to use a broad section of the representative XAI methods in use today, it is thus preferable to use CNNs as opposed to vision transformers.

3.4 XAI Saliency Methods

The theory of each XAI saliency method used in this thesis has been described in section 2.4.4 in chapter 2. In the following section, I will describe the specifics of how I have applied each method in my efforts to better align them with the goal of separating ID and OOD data points.

For all saliency methods, one significant difference between my application and those commonly used for model explanation is that I modify all methods to return unnormalized and unrectified saliencies. For most XAI applications, it is common to rectify the saliencies such that negative saliencies are set to zero. In addition, some methods output normalized saliencies, as XAI saliency maps are intended to be inspected individually. In these cases, the absolute magnitude can be considered less relevant, and looking at saliencies relative to each other more informative. For my purposes, both negative values and the magnitude may convey important information which should not be discarded.

3.4.1 LIME

As is common when applying LIME to images, I have used segmentations of the input image to reduce the dimensionality of the input that is fed into the LIME algorithm. As explained in section 2.4.3, superpixel segmentation algorithms such as SLIC may increase the accuracy of the explanations, but are CPU-bound and thus introduce a considerable computational overhead (which is problematic when tens of thousands of samples are to be evaluated). Thus, I have chosen to stick to a simple rectangular segmentation instead. Each image is split into $N \times N$ equally sized rectangular regions, with N being set to 4.

3.4.2 Occlusion

Like with LIME, I have used a rectangular segmentation approach, with each image being split into $N \times N$ regions. As with LIME, N is set to 4, which ensures that accurate comparisons between LIME and occlusion can be made.

3.4.3 GradCAM

When utilizing GradCAM, a choice must be made about which convolutional layer one should calculate gradients from. As described in section 2.2.2, earlier layers have higher resolution (more precise spatial information) but lower field of view (less comprehensive semantic information) than later layers. In most cases, the final convolutional layer is seen as most informative, with [26] stating that "convolutional layers naturally retain spatial information which is lost in fully-connected layers, so we can expect the last convolutional layers to have the best compromise between high-level semantics and detailed spatial information". For this reason, I also use the final convolutional layer.

As mentioned previously, I do not apply any Rectified Linear Unit (ReLU) activation function after calculating the saliencies, despite this being an explicit part of the

methodology of [26]. This is because information about what the model regards as detrimental to the prediction may also be informative for OOD detection.

3.4.4 Guided Backpropagation

GBP is a very simple method which is completely non-parametric, and as such no specific choices have had to be made. Like with all other methods, no normalization or rectification has been done to the output of GBP.

3.4.5 Integrated Gradients

For integrated gradients, the only choice that must be made is the choice of baseline. An ideal baseline should convey no information, and should ideally lead to an output of zero from the network. As [58] shows, there are no perfect choices for such a baseline, as all methods carry some assumption about our dataset and what it means for an input to "convey no information". However, the common choice for image classification tasks is the zero vector (a completely black image), as recommended by [30]. As such, this is the baseline I have used.

3.5 Evaluation

3.5.1 Metrics

AUROC and FPR95 are the most common metrics used for OOD detection [6, 12–15]. In OpenOOD ([6]), AUROC is chosen as the primary metric used to rank methods against each other, as mentioned previously. As [6] by far presents the most comprehensive complete benchmark of all OOD detection methods to date, I have followed their methodology and used AUROC when evaluating my methods, while also calculating FPR95 as a secondary metric.

AUROC is agnostic to which class is the positive and which is the negative. Whether we consider OOD samples as 0 or 1, the result will be the same. However, the same is not true for FPR95, necessitating a choice of positive and negative classes. There is no correct answer, but [6] chooses to consider OOD samples as the positive class, to "align with ML convention": It is common to consider abnormalities, anomalies, or the unexpected as the positive class (for example, a cancer detection system would consider the presence of cancer to be part of the positive class). This aligns with the goal of OOD detection, which is to detect abnormal inputs with regard to the data the model is trained on. Thus, I have also followed this convention and considered OOD samples as positive, and ID samples as negative. In this case, FPR95 has the following interpretation: "If a specific threshold leads to 95% of OOD samples being correctly classified as OOD, what percentage of ID samples are then incorrectly classified as OOD"? In other words, FPR95 measures the rate of "false alarms" when 95% of actual anomalies are caught.

For both of these metrics, the calculations are done on each OOD dataset individually, as opposed to comparing all OOD samples to the ID dataset. Afterwards, the metrics of all Near-OOD and Far-OOD are averaged, giving us two general performance metrics which tell us how a method functions on either Near-OOD or Far-OOD. This aligns with the methodology of [6]. Similarly, when plotting density plots for a given OOD detection metric, the densities of all Near- and all Far-OOD data points are combined, giving a more comprehensible overview than if all OOD data sets are individually. Figure 3.6 shows density plots before and after combining Near- and Far-OOD, on the maximum

logit score of images from the Cifar10 dataset. As we can see, there is not that much internal variance between the OOD datasets, allowing us to combine them without losing too much information.

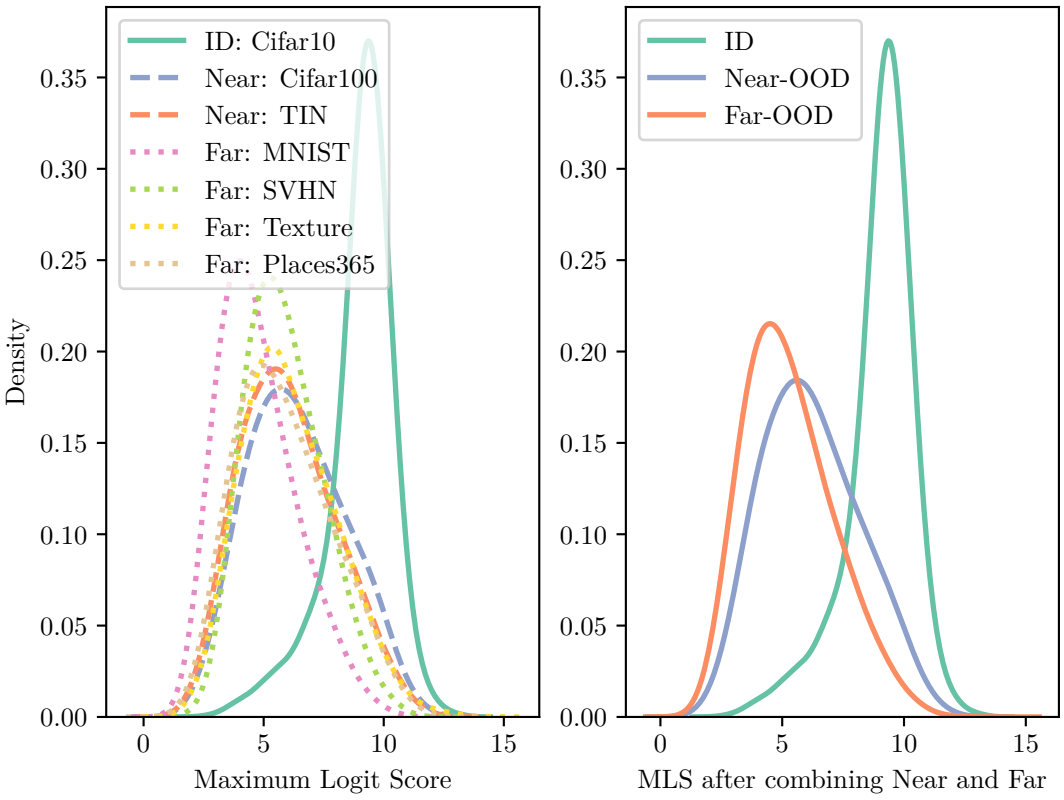

Figure 3.6: Density plots of MLS on CIFAR10 for all datasets individually and after combining Near- and Far-OOD

3.5.2 Development and Test Sets

Given the scope of my thesis, which focuses on post-hoc OOD methods which do not use outlier exposure or require any form of training, there is no need for a training dataset. Despite this, there is still a chance of overfitting, due to the exploratory nature of developing an entirely new class of methods. By trying out different methods and by tuning different parameters to find out how one might best separate ID and OOD data using XAI, I am also biasing the results. Therefore, it may be possible that potential improvements in performance are not due to the increased quality of the method, but instead because of peculiarities in the specific datasets that the method is used on.

Thus, I have split all the datasets mentioned in 3.2 into development and test sets, and have performed all development of the methods exclusively on the development set. Only when calculating the final metrics have I used the test set, to remove any bias associated with performing multiple experiments on the same data. Due to the large amount of data available in the CIFAR10 and ImageNet200 datasets, and their associated OOD datasets, I have simply split all datasets into two equally large parts.

3.5.3 Statistical Analysis of Results

When evaluating a new method, it is not enough to simply report the results from a single experiment. Instead one should run the same experiment multiple times and perform statistical analysis to ensure that the results are robust. Therefore, I have bootstrapped the test set ten times during the calculation of the final test results. This makes it possible to report means and standard deviations of the results as opposed to a single point estimate. With these repeated experiments, it is also possible to perform t-tests comparing the new methods against baseline OOD detection methods. Given that I have designed my methods to be general frameworks for integrating XAI saliencies, it is possible to test the same method with many different XAI algorithms. Doing this, however, will introduce the *multiple comparison problem*, where a p-value of 0.05 will no longer be sufficient for each test, as the chance of randomly receiving results that seem statistically significant is increased by the presence of multiple tests. Thus, I use Bonferroni corrected p-values whenever multiple tests are conducted on the same method. I set the threshold for statistical significance at 0.05, leading to a Bonferroni corrected p-value of $0.05/n$ for a given method, where n is the number of experiments conducted for a single method.

3.6 Implementation

This section goes over the implementation details of my thesis.

3.6.1 Basic hardware and software

Python is the most popular programming language for data science and ML research [59], and as such it is my language of choice as well. For this thesis, `Python 3.9` has been used. Below is a table of the key libraries that have been utilized.

Library	Version number	Short description
OpenOOD	1.5	Comprehensive OOD detection framework [6]
PyTorch	2.4.1	GPU-accelerated ML library
Captum	0.7.1	XAI methods integrated with PyTorch
Scikit-Learn	1.5.2	Various ML methods
NumPy	1.26.4	Efficient matrix multiplication and scientific computing
Matplotlib	3.9.2	Visualization library
Seaborn	0.13.2	Visualization library built on top of Matplotlib
Pandas	2.2.3	Data visualization and manipulation library
SciPy	1.13.1	Scientific computing library

All development and computation was done on a single computer with an `Intel i7-8700K` CPU and an `Nvidia GeForce RTX 3090` GPU.

3.6.2 Method Evaluation: OpenOOD

As explained previously, OpenOOD [6] represents the most comprehensive benchmark for OOD detection methods. It is also a framework which easily allows for development and benchmarking of new methods, and is thus the ideal framework for the purposes of this thesis.

In particular, OpenOOD includes a "unified, easy to use evaluator" [52] that makes evaluating new methods very simple. All that is required is that new methods inherit from a base class (`BasePostprocessor`[3]), and override the calculation of OOD scores. Code listing 3.1 shows all the code required to create the Aggregate of Saliency OOD detector.

```
class SaliencyAggregatorPostprocessor(BasePostprocessor):
    def __init__(self, config, saliency_generator, aggregator):
        super().__init__(config)

        self.saliency_generator = saliency_generator
        self.aggregator = aggregator

    def postprocess(self, net: nn.Module, data: Any):
        predictions = torch.argmax(net(data), dim=-1)

        saliencies = self.saliency_generator(net, data)
        score_ood = self.aggregator(saliencies, dim=-1)

        return predictions, score_ood
```

Listing 3.1: Source code listing for the Aggregate of Saliency OOD detector

With this `postprocessor` defined, evaluating it on a specific dataset is similarly simple (listing 3.2):

```
resnet18_pretrained = get_network('cifar10')

ood_detector = SaliencyAggregatorPostprocessor(None, GradCAM, torch.mean)

evaluator = Evaluator(
    net=resnet18_pretrained,
    id_name='cifar10',
    postprocessor=ood_detector,
)

metrics = evaluator.eval_ood()

print(metrics)
```

Listing 3.2: Source code listing for evaluating methods within the OpenOOD framework

This code will calculate the OOD scores for all data samples in both the ID and OOD datasets, and subsequently calculate the AUROC and FPR95 for all the OOD datasets when comparing their OOD values to the values of the ID datasets. Code listing 3.3 shows this output when using the baseline MSP method.

```
           FPR@95  AUROC  AUPR_IN  AUPR_OUT   ACC
cifar100    61.36  86.51    84.20     85.05 95.56
tin         42.02  88.88    88.81     85.57 95.56
nearood     51.69  87.69    86.51     85.31 95.56 # average of two above
mnist       19.38  93.86    79.72     98.89 95.56
svhn        24.78  91.38    84.26     95.49 95.56
texture     43.31  88.68    91.01     80.97 95.56
places365   41.62  89.21    68.49     96.28 95.56
farood      32.27  90.78    80.87     92.91 95.56 # average of four above
```

Listing 3.3: Output of calling `evaluator.eval_ood` with CIFAR10 as the dataset and MSP as the detector

[3]In OpenOOD, `Postprocessors` are the OOD detection algorithms that generate an OOD score during inference

As we can see, OpenOOD allows for very easy evaluation of new methods. Furthermore, it allows for easy comparisons between methods, one of the stated goals of the framework [6]. This makes it an ideal framework for this thesis.

Modifications to OpenOOD

As explained previously, continually testing new methods on the same datasets will bias the final results. As of the writing of this thesis, there are no functionalities in OpenOOD which allow for creating development or test sets, all evaluations are done with entire datasets. For my purposes, which involve continuous exploration of different methods and careful inspection of the datasets, this is inadequate. Thus, I have made modifications to the `Evaluator` class such that it takes a `data_split` parameter during initialization, and have also modified the function `get_id_ood_dataloader` to accept this parameter and return the correct split accordingly.

Furthermore, there is no functionality for sampling from the datasets as opposed to using them as they are, which is necessary to perform bootstrapping and calculate the statistical significance of the results. This has been done by passing a seed to the `Evaluator` class, which, if defined, will be used to seed a random sampling operation on the datasets. By instantiating several `Evaluator` classes with different seeds, we can bootstrap the evaluation and perform statistical analysis on the results.

3.6.3 Implementation of Saliency Methods

There are many libraries which implement XAI methods [22, 60, 61]. When these implementations were suitable for my purposes, I used them. However, given I am not using these explanations for their original purpose (elucidating why a model came to a specific decision), there are many cases where the current implementations are inadequate. The two main problems are lack of access to the raw saliency values and slow speeds.

GradCAM

GradCAM has been implemented from scratch in `PyTorch`. There are several libraries which implement GradCAM [60, 61]. However, given that these libraries are concerned with simply producing heatmaps that users can inspect, they do not output the raw saliency values, but upscale the saliencies to match the input image dimensions. As explained in 2.4.4, GradCAM uses the final feature map to generate an explanation, which usually has a spatial dimension which is far smaller than the input image (e.g. 7×7 versus 224×224). When overlaying these values on the input image, it is common to use bilinear interpolation [61], which interpolates all 224×224 positions based on the original 7×7 saliency map. For visualizations, this is reasonable. When attempting to use this data to separate ID and OOD data however, this is undesirable. Bilinear interpolation introduces new values, which changes many statistical qualities of the saliency values. This may reduce the separability of different samples. Furthermore, given that these new values do not add any new information, it is inefficient to involve these upscaled saliency maps in any computational operation.

Although it is relatively simple to modify the source code of these libraries to remove the interpolation, GradCAM is not very difficult to implement, and as such I have simply used my own implementation.

LIME

LIME has an implementation for Python, written by the original authors [22]. However, this implementation is not suitable for my purposes. The main issue is that it is far too slow, being implemented with `NumPy` which restricts the computation to the CPU. Furthermore, [22] also returns full size heatmaps, although the actual number of saliency values used to create this heatmap is far smaller than the number of pixels in the image.

Thus I have implemented LIME myself, using PyTorch whenever possible.

Occlusion

Captum [60] is a library of XAI methods implemented in PyTorch, and this library contains a suitable implementation of occlusion. As explained previously, occlusion occludes parts of the image and compares the prediction scores before and after the occlusion. By occluding all parts of the image, we can get a saliency value for all positions. Occlusion is usually done using a sliding window, similar to a convolutional kernel, which is slid over all parts of the image. Such a window is rarely a single pixel, because it is often not interesting to see how a single pixel contributes to a prediction, but rather a larger region. What this means is that the final heatmap, although it is the same size as the input image, actually contains far fewer unique values. To avoid performing computations on thousands of repeated values, I reduce the size of the heatmap by sampling one pixel from each of the positions the sliding window has been applied, which can be efficiently done using a 1×1 MaxPooling kernel with a stride equal to the stride used during the occlusion.

Guided Backpropagation and Integrated Gradients

Captum also contains a suitable implementation for guided backpropagation and integrated gradients, which I have utilized in essentially unmodified forms. By definition (see section 2.4.4 and 2.4.4), these methods return saliency maps over the entire input image dimensions. This separates them from the other methods, which return a far lower number of distinct saliencies, which are upscaled or transformed to the entire image (LIME and Occlusion output the same values for all pixels within a segment, GradCAM outputs an amount of saliencies corresponding to the final feature map, which is then upscaled). Because of this, the saliencies returned are exactly equivalent to the raw values I require, and no modifications are necessary.

However, this lack of dimensionality reduction also poses a technical challenge when we wish to store these saliency maps for later data analysis. For example, storing all the saliencies, without dimensionality reduction, for ImageNet200 and its associated OOD datasets would require 28 GB. Thus, instead of storing the saliencies themselves, I calculate the aggregate values during saliency generation and store them instead. This has the downside that if a new form of aggregation is to be tested, the whole dataset of saliencies has to be generated again. However, the decrease in storage requirements is immense, being between a 1000 and 10 000 times decrease depending on the number of aggregates stored.

3.7 Summary

Chapter 4

Experiments and Results

This chapter contains the results of experiments as described in chapter 3. In section 4.1, I show how different forms of aggregation over saliency maps separate ID and OOD data points. This step is important, because it informs the choice of aggregation used in the Saliency Aggregation and Saliency Aggregation+Logits OOD detection methods. I report AUROC for different forms of aggregation over CIFAR10 and ImageNet200. This is done on the validation sets, to avoid biasing the final results.

In section 4.2, OOD detection on the test sets is performed using the three methods I have introduced in section 3.1. The choice of aggregation and XAI method is informed by the findings from the preceding section. The AUROC for each method is calculated over ten bootstraps for each dataset, enabling statistical analyses which investigate whether XAI OOD detection methods outperform baseline methods.

4.1 Data Analysis of Saliency Maps

This section will detail how various XAI methods generate explanations which differ between ID and OOD. The section considers each dataset individually. For each dataset, I first present the level of separation achieved by the two baseline methods MSP and MLS, to give a general intuition about how easily the ID and OOD datasets are to separate. Following this, I go through the different XAI methods and their different statistical qualities.

4.1.1 ImageNet200

ImageNet200 is a suitable dataset to begin with, given ImageNet's ubiquity in AI research. Figure 4.1 shows the distribution of the maximum softmax score and the maximum logit. Here, we see that there is a decent amount of separation between ID and OOD data points, even with the simple baseline methods introduced by [12] and [35]. Separating the distributions using MSP, we get an AUROC score of 0.834 for Near-OOD and 0.915 for Far-OOD. Using MLS, we get an AUROC score of 0.833 for Near-OOD and 0.903 for Far-OOD.

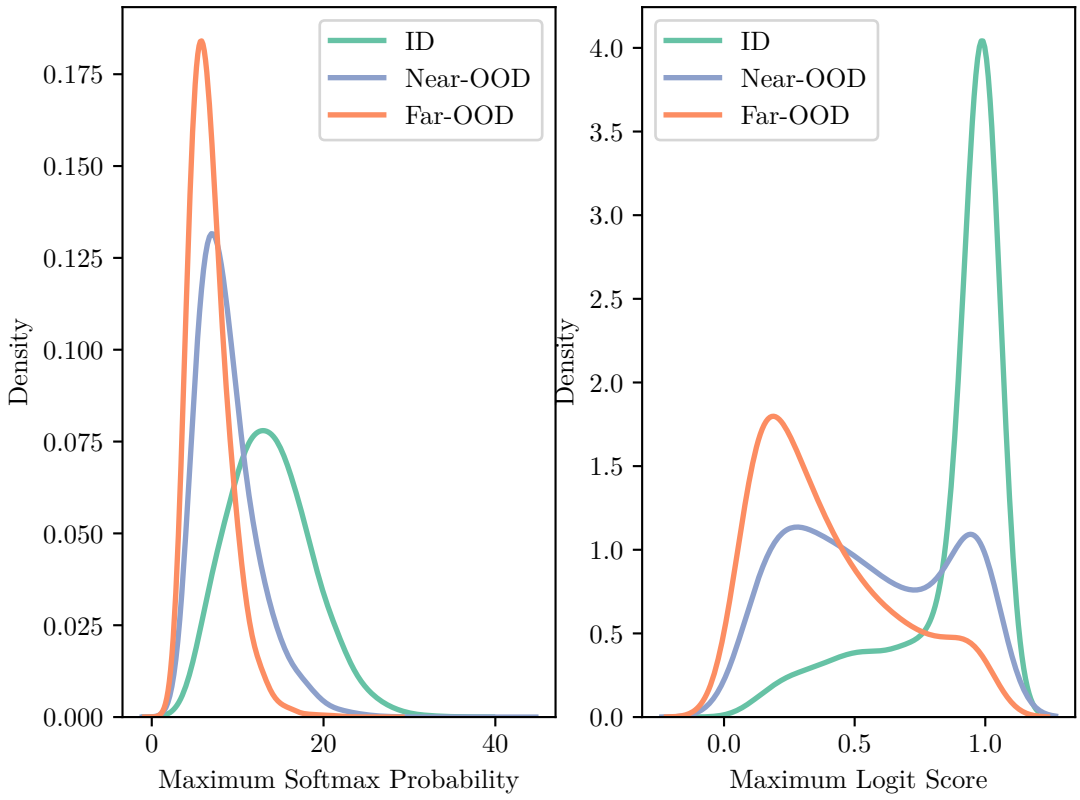

Figure 4.1: Density plot of the maximum softmax probability and maximum logit score on ImageNet200

LIME

With the baselines reported, we turn our attention to the first XAI saliency method, LIME. Figure 4.2 presents graphs representing the two main forms of aggregation that has been applied; those which consider the magnitude of the saliencies and those which consider the statistical spread. These two forms are here represented by the vector norm and the RMD.

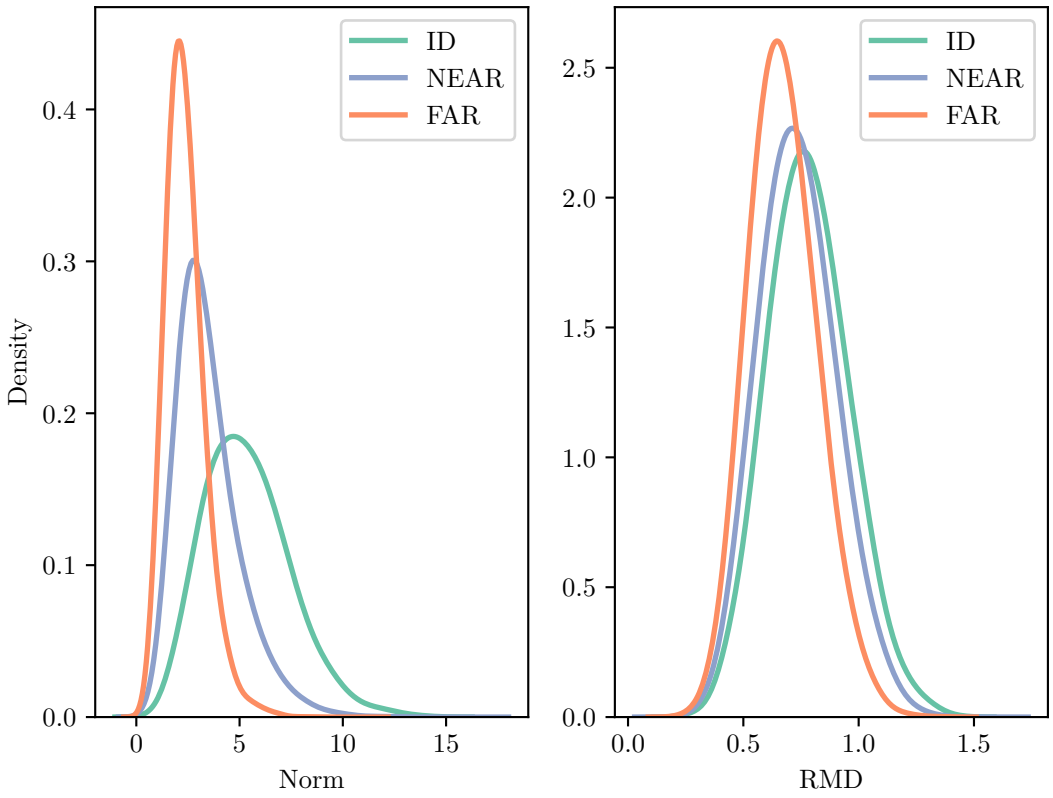

Figure 4.2: Vector norm and RMD density plots for LIME on ImageNet200

From the figure we can see that for LIME on ImageNet200, it is indeed the case that saliencies are higher for ID data points than for OOD data points. Using the vector norm to distinguish ID and OOD, we get an AUROC of 0.814 on Near-OOD, which is slightly lower than the baseline methods, and 0.925 on Far-OOD, which is actually higher than the baselines. These results are far better than those attained by [7], the only other related work which has attempted to use saliency maps to separate ID and OOD data. This work achieved an AUROC score of only 0.52 when used on datasets which were not just small toy datasets, which is for all practical purposes equivalent to guessing. This large improvement suggests that the usage of raw saliency values, rather than normalized heatmaps (as was used by [7]) is highly consequential for the performance of XAI OOD detection algorithms.

Measuring the statistical spread using RMD, we find that there is indeed also a higher spread in ID data when compared to OOD data. However, in this case the overlap is substantial, which is reflected in the AUROC scores attained when discriminating using RMD: The Near-OOD AUROC score was 0.594, and the Far-OOD score was 0.695. In both cases, the scores attained are far lower than the both of the baselines.

Table 4.1 shows the AUROC scores for all forms of aggregation on LIME, as well as the previously mentioned AUROC of the baseline methods. The AUROC scores are multiplied by 100, to avoid having a redundant leading zero in all cells. In addition to the AUROC, the correlation between the aggregates and the maximum logit and maximum

softmax score is reported, which gives insight into how XAI saliency maps are related to the prediction confidence of the model.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV↓	RMD	QCD↓
Near-OOD AUROC	83.3	83.4	78.8	69.3	**81.4**	77.1	78.1	78.0	53.7	59.4	53.2
Far-OOD AUROC	91.5	90.3	88.1	75.2	**92.5**	90.2	91.4	87.4	49.3	69.5	49.8
Correlation with MLS	-	-	0.75	0.60	0.71	0.45	0.54	0.72	-0.01	0.08	-0.01
Correlation with MSP	-	-	0.61	0.48	0.61	0.41	0.48	0.60	-0.01	0.09	-0.01

Table 4.1: AUROC scores for LIME on ImageNet200. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

From this table, we can note several interesting observations. Firstly, we see that the magnitude based aggregations in general perform quite well, with vector norm in this case being the best. In contrast, the methods based on statistical dispersion perform poorly, putting into question the hypothesis that the saliency maps of ID data is more concentrated and less spread out than those on OOD data. Indeed, we see that while the RMD is higher on average for ID data, the QCD and CV is lower on average when generating saliencies using LIME. This further puts doubt on the idea that the spread of saliency maps is a reliable indicator of OOD-ness.

Secondly, the aggregates which capture information about the magnitude of the saliencies are highly correlated with both the maximum logit and the maximum softmax score of the predicted class. This is not unexpected, as LIME generates saliencies using differences in prediction values as different parts of the image is masked. If the predicted value is higher on average for ID data, then it is likely that the drop in predicted value when masking parts of the image is larger as well, leading to higher saliencies. Regardless, it seems clear that it is not only the correlation to the model output which explains the discriminative power of these aggregates, as we see that vector norm aggregation has lower correlation with MLS and equal correlation with MSP when compared to the mean, but has higher scores.

In general, the results from these aggregations are promising, and show that there is definite potential for OOD detection algorithms based on XAI outputs. However, the reader should note that these results are done on the validation set, and that no statistical tests have been done at this point. The statistical significance of using XAI saliency maps for OOD detection will be revealed in section 4.2, when the final testing is done on the test set and bootstrapping is performed.

Occlusion

Now, we turn our attention to occlusion saliency mapping. Looking at figure 4.3, we see that there is far more overlap between the ID and OOD densities than with LIME.

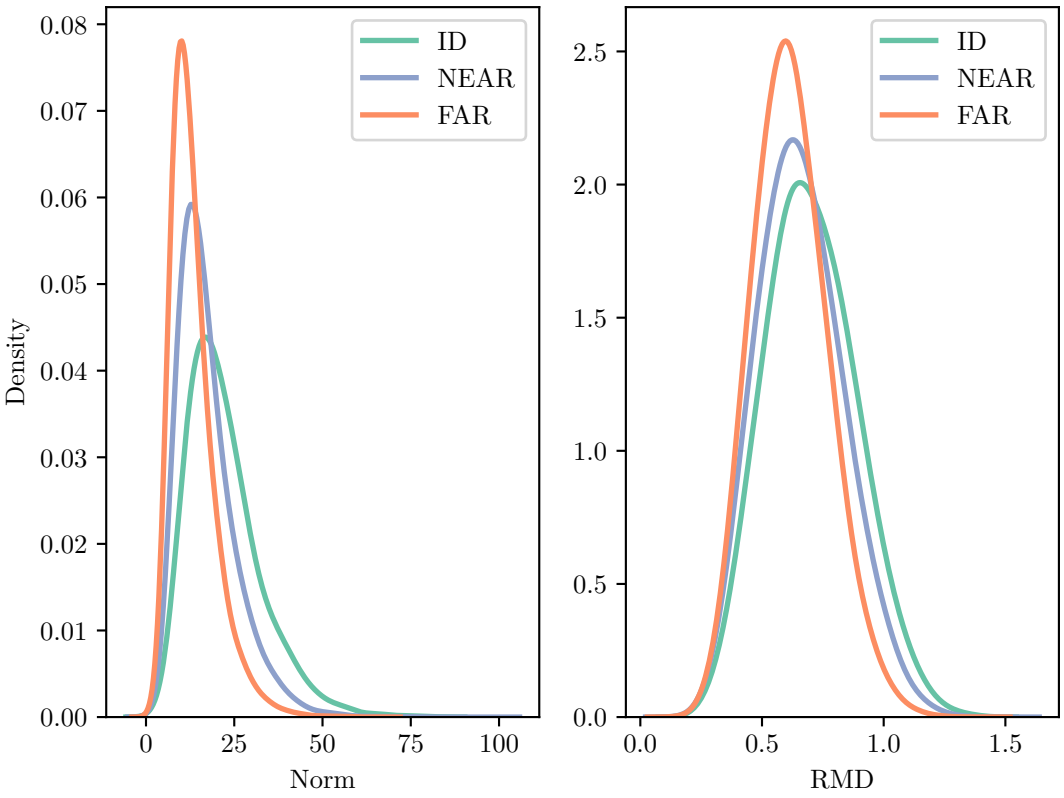

Figure 4.3: Density plots of Norm and RMD for occlusion on ImageNet200

From table 4.8 we see that this trend is apparent over all of the different forms of aggregation, not just the vector norm and RMD. Regardless, there is still a clear trend ID saliency magnitudes being higher than OOD magnitudes, as all AUROC scores are above 0.50 without negation. With this saliency method, aggregating using range performs the best, with max relatively close behind. Interestingly, we see that for occlusion, all the statistical dispersion aggregates are higher for ID data, as was the original hypothesis. Regardless, the AUROC scores are very poor, and thus these metrics do not seem suitable to discriminate between ID and OOD.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV	RMD	QCD
Near-OOD AUROC	83.3	83.4	58.6	54.2	65.7	**69.1**	66.8	60.4	55.0	57.7	55.4
Far-OOD AUROC	91.5	90.3	69.7	62.9	77.8	**84.6**	83.5	72.3	63.8	64.7	62.7
Correlation with MLS	-	-	0.37	0.31	0.42	0.44	0.47	0.41	0.01	0.14	0.01
Correlation with MSP	-	-	0.29	0.24	0.37	0.41	0.41	0.34	0.01	0.13	0.01

Table 4.2: AUROC scores for Occlusion on ImageNet200. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

GradCAM

After having inspected two completely model independent XAI methods, we turn our attention to the first of the three gradient based methods, GradCAM. As we have seen density plots for both LIME and occlusion, I will omit these here, as the table contains the necessary information. From table 4.9, we see that vector norm aggregation of GradCAM saliencies actually beats the baselines on both Near- and Far-OOD detection. The increase on Far-OOD is particularly impressive, with an increase of 1.4 percentage points. However, we should keep in mind that these results are done on the validation set and that the final results and their statistical significance will only be explored in section 4.2.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV	RMD	QCD
Near-OOD AUROC	83.3	83.4	83.3	80.3	**83.8**	80.5	81.9	83.5	50.8	51.8	51.7
Far-OOD AUROC	91.5	90.3	91.5	87.6	**92.9**	92.2	92.8	92.7	63.6	64.9	64.8
Correlation with MLS	-	-	1.00	0.94	0.97	0.69	0.81	0.96	-0.16	-0.12	-0.12
Correlation with MSP	-	-	0.83	0.78	0.82	0.62	0.70	0.81	-0.11	-0.07	-0.07

Table 4.3: AUROC scores for GradCAM on ImageNet200. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

A second interesting observation is that the mean value is completely correlated with the maximum logit. The vector norm is also almost completely correlated, but slightly less, and has a higher score than the mean.

Integrated Gradients

Looking at table 4.4, we see that the trend of larger saliency magnitudes on ID data continues to hold for integrated gradients as well. With integrated gradients the mean and the vector norm seem to be the most discriminative, with the mean saliency having scores which are around 1 percentage point below the baselines.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV↓	RMD	QCD
Near-OOD AUROC	83.3	83.4	**82.1**	55.3	67.3	63.9	63.5	64.1	66.1	51.3	50.5
Far-OOD AUROC	91.5	90.3	**90.5**	56.0	87.8	86.7	85.9	79.6	49.8	39.1	53.4
Correlation with MLS	-	-	0.94	0.14	0.40	0.31	0.30	0.35	-0.16	0.01	0.01
Correlation with MSP	-	-	0.79	0.10	0.36	0.29	0.28	0.31	-0.15	0.00	0.00

Table 4.4: AUROC scores for IntegratedGradients on ImageNet200. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

GBP

We now turn our attention to the final XAI saliency method, GBP. Here we see something interesting; as the only method on CIFAR10, GBP actually has two magnitude metrics where OOD data points have higher values than ID data points. Looking at table 4.5, we see that both the mean and median saliency is lower on ID data. Moreover, the mean is considerably lower, with an AUROC of 0.737 on Near-OOD and 0.817 on Far-OOD when choosing lower values as ID as opposed to higher. The vector norm, range and maximum aggregations are still higher for ID data. These aggregations are the ones which are not affected by large negative values. The conclusion to draw from these results is as follows: For GBP, ID XAI saliency maps exhibit higher magnitudes, but they are not restricted to positive attributions.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean↓	Median↓	Norm	Range	Max	Q3	CV	RMD↓	QCD↓
Near-OOD AUROC	83.3	83.4	73.7	55.7	**77.4**	71.3	71.8	71.0	50.2	52.8	51.5
Far-OOD AUROC	91.5	90.3	81.7	51.1	**92.3**	90.7	90.6	68.7	43.5	72.3	50.7
Correlation with MLS	-	-	-0.33	-0.08	0.45	0.25	0.25	0.36	0.00	-0.01	0.01
Correlation with MSP	-	-	-0.30	-0.08	0.42	0.25	0.24	0.35	0.00	-0.00	0.01

Table 4.5: AUROC scores for GBP on ImageNet200. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

Looking at figure 4.4, we see that the mean saliency value is indeed lower for ID data when compared to all the OOD data sets.

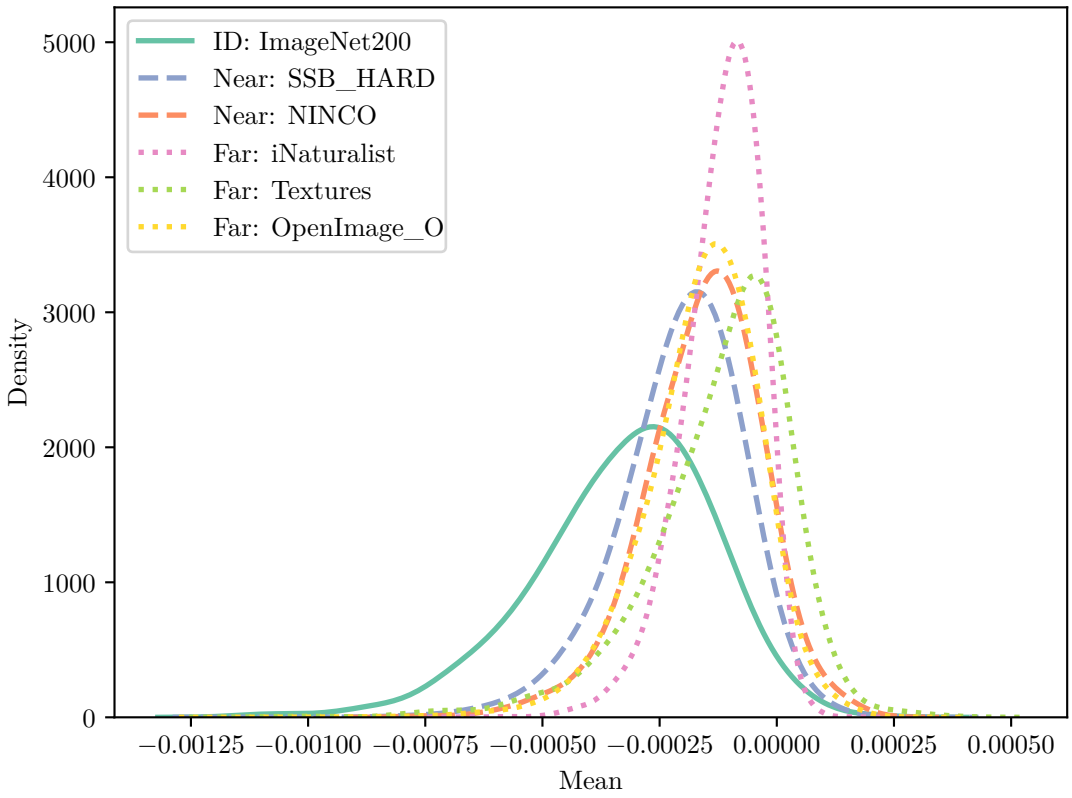

Figure 4.4: Density plots of Norm and RMD for occlusion on ImageNet200

Overall results on ImageNet200

Next, we consider the overall performance of the different aggregations and XAI saliency methods on ImageNet200. Figure 4.5 shows the average Near- and Far-AUROC for magnitude aggregations. From this, we see that the vector norm, range and maximum aggregations performed the best over all XAI saliency methods. All statistical dispersion methods performed poorly, and as such I do not plot them. In general, saliency aggregation methods seem to perform far above the heatmap clustering performed by [7], at least on these validation datasets.

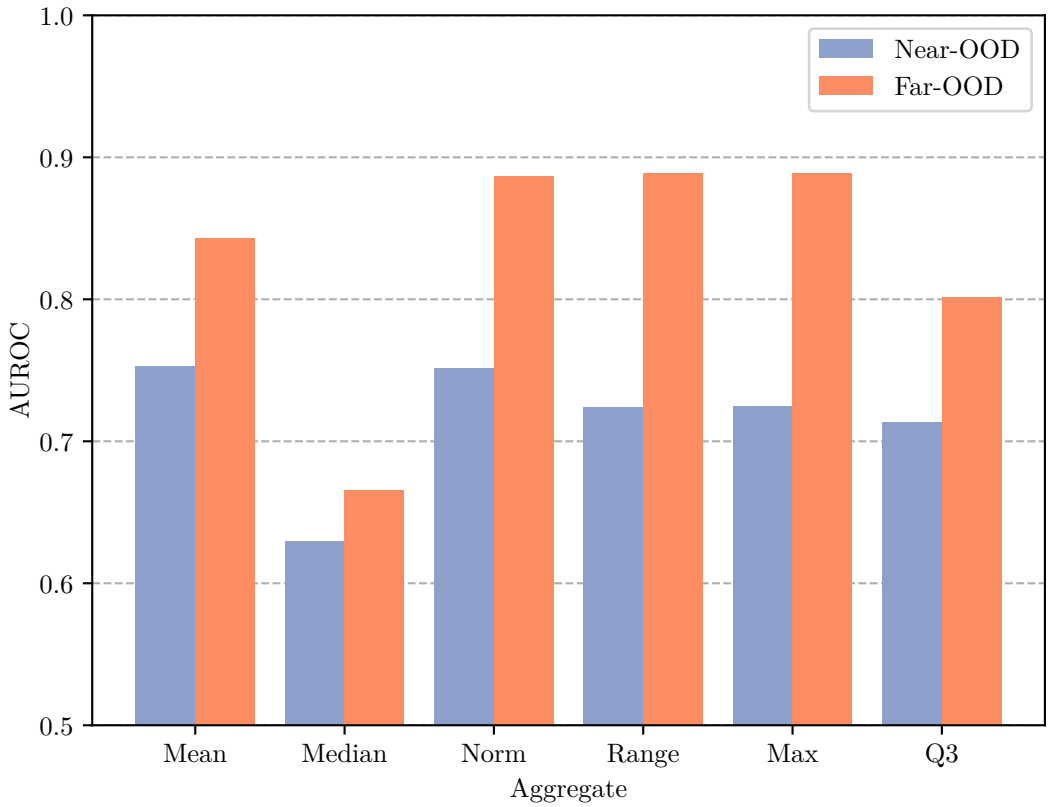

Figure 4.5: Barplot of average AUROC scores on ImageNet200. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Table 4.6 shows the underlying data for the figure above, as well as the mean AUROC for both Near and Far. Here, we can see that the mean performed the best on Near-OOD while the range performed the best on Far-OOD. However, as we know, the mean was lower for GBP and lower scores had to be considered ID, which makes it less desirable than the other methods, which were consistently larger for ID data. The vector norm was only slightly worse than the mean on Near-OOD and only slightly worse than the range on Far-OOD, making it the best aggregation overall. As we can see, all the statistical dispersion methods performed poorly.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV	RMD	QCD
Near-OOD AUROC	83.3	83.4	**75.3**	63.0	75.1	72.4	72.4	71.4	55.2	54.6	52.5
Far-OOD AUROC	91.5	90.3	84.3	66.6	88.6	**88.9**	88.8	80.1	54.0	62.1	56.3
Mean AUROC	87.4	86.9	79.8	64.8	**81.9**	80.6	80.6	75.7	54.6	58.4	54.4

Table 4.6: Average AUROC scores for all XAI saliency methods on ImageNet200. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

If we instead look at the best aggregation for each XAI saliency method (figure 4.6), and compare these to the baselines, we find that LIME, GradCAM and GBP beat the baselines on Far-OOD, while GradCAM is the only method who beats the baselines on Near-OOD. Except for occlusion, all method performed reasonably well, given a correct choice of aggregation.

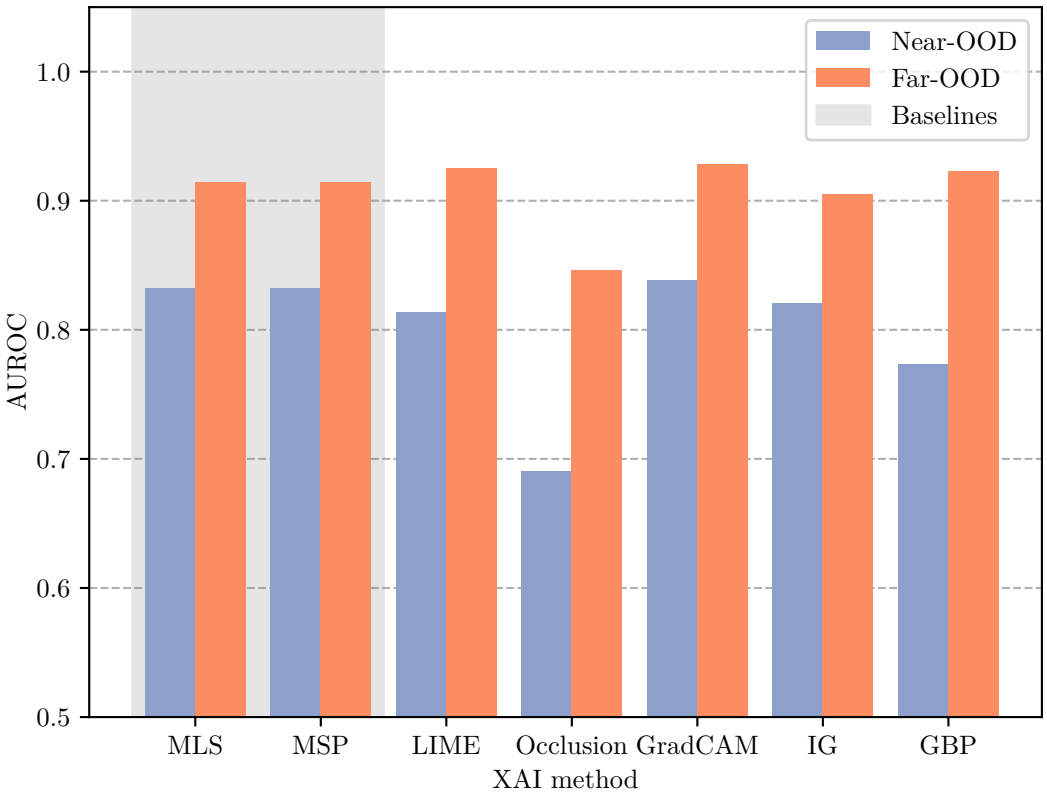

Figure 4.6: Barplot of the highest AUROC scores per XAI saliency method on ImageNet200. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Finally, figure 4.7 shows a heatmap of the average performance over both Near-OOD and Far-OOD for all combinations of aggregation and XAI saliency method. From this we can get a visual overview over what combinations perform well.

Figure 4.7: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

From this, we can see that GradCAM performs well overall (likely due to its high correlation with MLS) while occlusion performs poorly, as we saw previously. Apart from this, we see that some promising combinations are LIMENorm, GBPNorm and IGMean.

The results above demonstrate that saliency maps themselves, without any other information from the network, seem to be able to adequately separate ID from OOD data points, contrary to what was found by [7]. In the next section, I will explore whether these results hold on CIFAR10 as well. After these tests, we will have an informed opinion about what XAI saliency methods and aggregations best separate the data on the validation set, and we can choose a selection of combinations and gather final results on the test set.

4.1.2 CIFAR10

CIFAR10 differs from ImageNet200 in some important respects, making it an ideal second dataset to investigate. While ImageNet200 has 200 classes, CIFAR10 has only 10. This means that ImageNet200 is a much broader classification task, with networks that are trained to recognize a wide variety of different objects. With a more narrow ID dataset of only 10 classes, we may see different behaviour when a network is exposed to OOD data. Another important change is the size of the images. ImageNet200 images are

224×224 pixels, while CIFAR10 images are only 32×32. This reduction in resolution may also affect how saliency maps are generated, as each pixel now covers a much larger area of the total image and is thus more important compared to each pixel in ImageNet200 images. First, let us inspect the baseline performances.

From figure 4.8, we can see that there is a decent degree of separation when using the maximum logit and the maximum softmax score. The maximum softmax score is highly concentrated around 0.95-1.00 for ID data, to a much higher degree than with ImageNet200. This is not surprising, given the much smaller number of classes in CIFAR10. With fewer and more easily separated classes, it is much easier for the network to saturate the maximum softmax score.

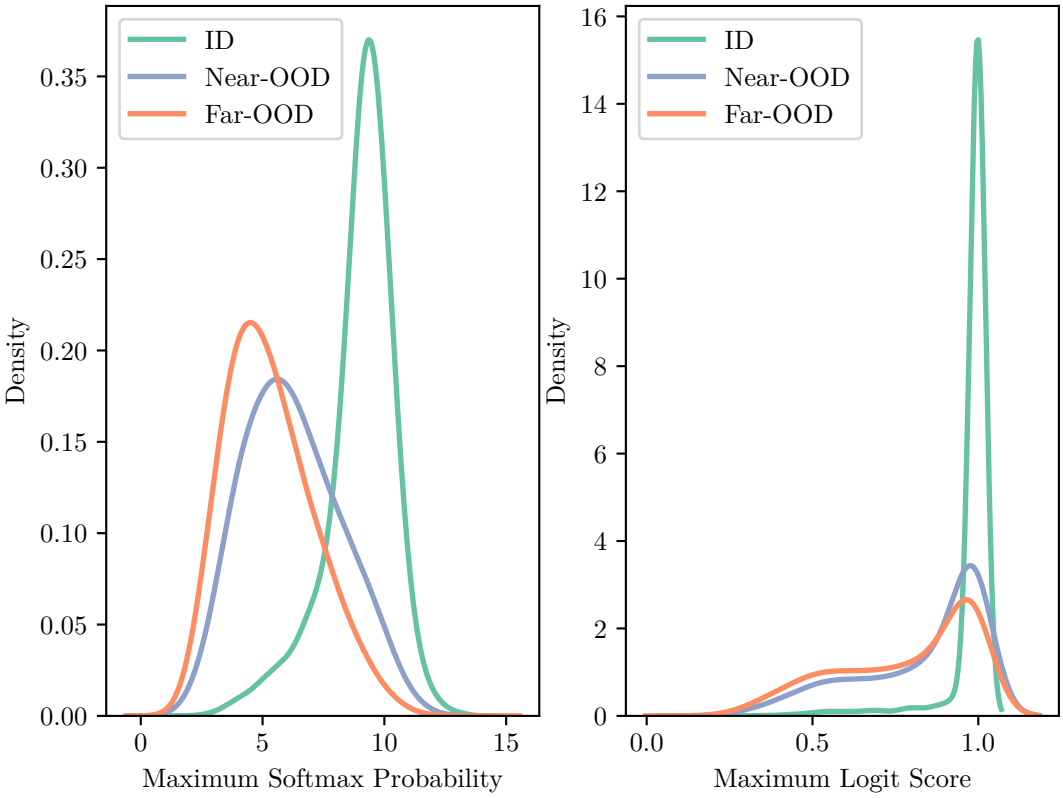

Figure 4.8: Barplot of average AUROC scores on ImageNet200. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Separating the distributions using MSP, we get an AUROC score of 0.877 for Near-OOD and 0.908 for Far-OOD. Using MLS, we get an AUROC score of 0.867 for Near-OOD and 0.914 for Far-OOD. With the baselines reported, we turn our attention to the first XAI saliency method, LIME.

LIME

Table 4.7 shows the results for the aggregations on the saliency maps generated by LIME. From this table, we see a similar trend as when we applied LIME to ImageNet200: the

magnitude of saliency methods all show a clear trend of higher values on ID data, while the statistical dispersion methods are poor and uninformative. However, the performance compared to the baselines is worse on CIFAR10, with over 5 percentage points lower scores on both Near- and Far-OOD. This is far worse when considering that LIME actually achieved a better AUROC than the baselines on ImageNet200 when using vector norm aggregation. It may be that because of the lower resolution of the images, each occluded region used to generate LIME explanations carries less information, which introduces some instability in when generating explanations.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV↓	RMD	QCD↓
Near-OOD AUROC	86.7	87.7	**81.2**	73.0	77.7	67.1	76.1	76.6	63.6	61.0	59.0
Far-OOD AUROC	91.4	90.8	**86.1**	77.9	84.0	74.5	81.9	83.9	59.0	61.2	53.2
Correlation with MLS	-	-	0.40	0.28	0.34	0.22	0.31	0.37	-0.00	0.11	-0.00
Correlation with MSP	-	-	0.29	0.20	0.23	0.15	0.22	0.26	0.01	0.08	-0.00

Table 4.7: AUROC scores for LIME on CIFAR10. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

Regardless of the lower scores when compared to ImageNet200, the scores still show that the magnitudes of XAI saliency maps differ between ID and OOD data when using LIME to generate them.

Occlusion

Table 4.8 shows the results of using occlusion to generate saliency maps. Here we see something quite surprising: all magnitude metrics are lower for ID data. This result forces us to reconsider the theory that the magnitude of saliencies is higher on average for ID data, as we see a complete inversion of the results we got when using occlusion on ImageNet200.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean↓	Median↓	Norm↓	Range↓	Max↓	Q3↓	CV	RMD	QCD↓
Near-OOD AUROC	86.7	87.7	63.4	63.1	**76.6**	74.2	67.3	75.5	52.4	54.7	50.3
Far-OOD AUROC	91.4	90.8	61.4	63.1	**74.6**	71.6	63.8	74.6	51.8	57.2	51.4
Correlation with MLS	-	-	0.22	0.11	0.13	0.05	0.27	0.13	0.00	0.34	0.00
Correlation with MSP	-	-	0.22	0.12	0.14	0.07	0.25	0.15	0.00	0.26	0.00

Table 4.8: AUROC scores for Occlusion on CIFAR10. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

Looking at the distribution for the vector norm (figure 4.9), we find a consistently lower ID value when compared to all OOD datasets. Essentially the same plot can be seen with the other magnitude metrics as well.

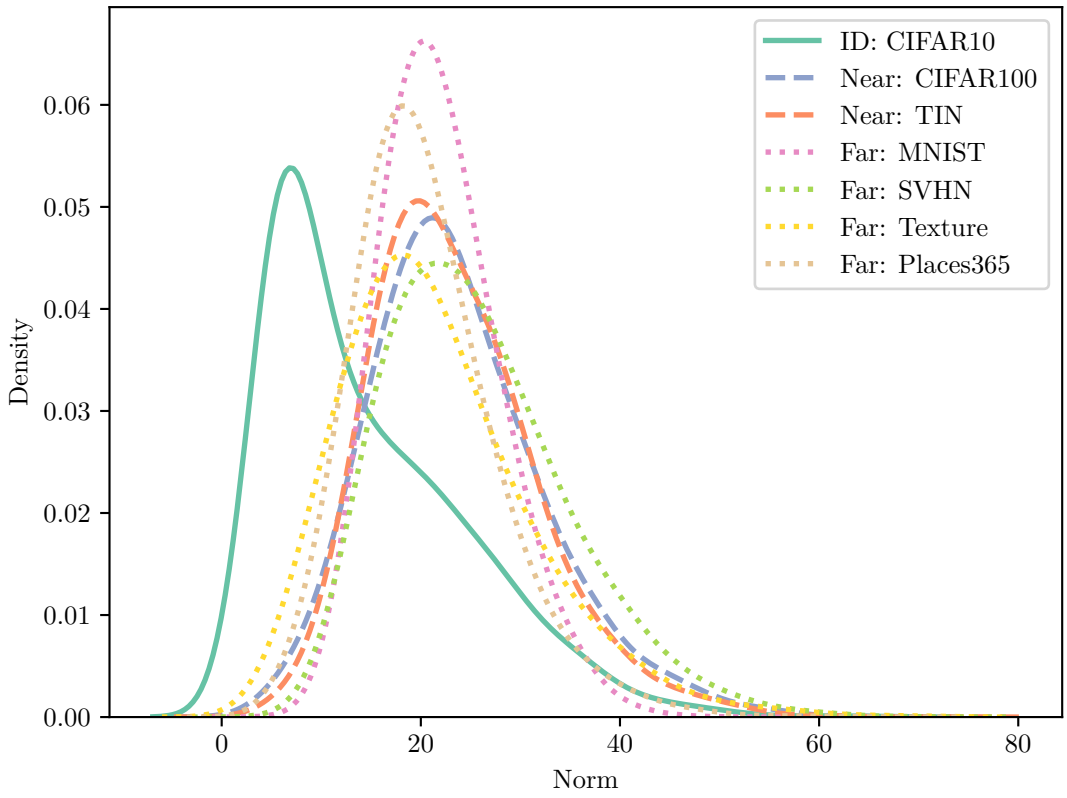

Figure 4.9: Barplot of average AUROC scores on ImageNet200. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

These results are very interesting, as they show that there is no guarantee that XAI saliency maps will be of higher magnitude on ID data. Even more interesting is the fact that when they are lower, they are lower across all OOD datasets and still allow for separation between ID and OOD by considering low saliency magnitudes as ID as opposed to high saliency magnitudes. These results imply that even if the main hypothesis of ID saliency maps having higher magnitudes may not be correct, this does not mean that we cannot separate ID and OOD based on magnitude.

GradCAM

Saliency maps made from GradCAM, with their mathematical equivalence to the maximum logit when applying the mean, unsurprisingly do not suffer from the same problems as occlusion. As we can see from table 4.9, the results for mean are equivalent to the maximum logit.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV↓	RMD↓	QCD↓
Near-OOD AUROC	86.7	87.7	**86.8**	85.9	86.7	71.8	86.5	85.2	75.3	76.7	74.2
Far-OOD AUROC	91.4	90.8	**91.4**	91.2	91.4	80.7	91.2	90.6	78.8	80.1	74.5
Correlation with MLS	-	-	1.00	0.99	1.00	0.68	0.96	0.98	-0.47	-0.48	-0.45
Correlation with MSP	-	-	0.79	0.78	0.79	0.57	0.77	0.78	-0.42	-0.42	-0.41

Table 4.9: AUROC scores for GradCAM on CIFAR10. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

Integrated Gradients

With integrated gradients we see another surprising result. Here, the mean and median are higher for ID data, while the vector norm, range, maximum and third quartile are lower.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm↓	Range↓	Max↓	Q3↓	CV↓	RMD↓	QCD↓
Near-OOD AUROC	86.7	87.7	83.1	59.2	78.6	76.0	74.8	78.4	**84.6**	51.0	52.7
Far-OOD AUROC	91.4	90.8	**88.4**	57.9	68.6	65.1	64.0	70.7	80.0	57.4	51.0
Correlation with MLS	-	-	0.65	0.08	-0.10	-0.07	-0.06	-0.11	-0.03	-0.01	0.00
Correlation with MSP	-	-	0.50	0.06	-0.07	-0.05	-0.05	-0.06	-0.02	0.01	-0.00

Table 4.10: AUROC scores for IntegratedGradients on CIFAR10. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

This implies that ID data has higher positive saliencies on average than OOD data, but that the magnitude of both positive and negative saliencies is higher on OOD data. Looking at figure 4.10, we see this difference clearly.

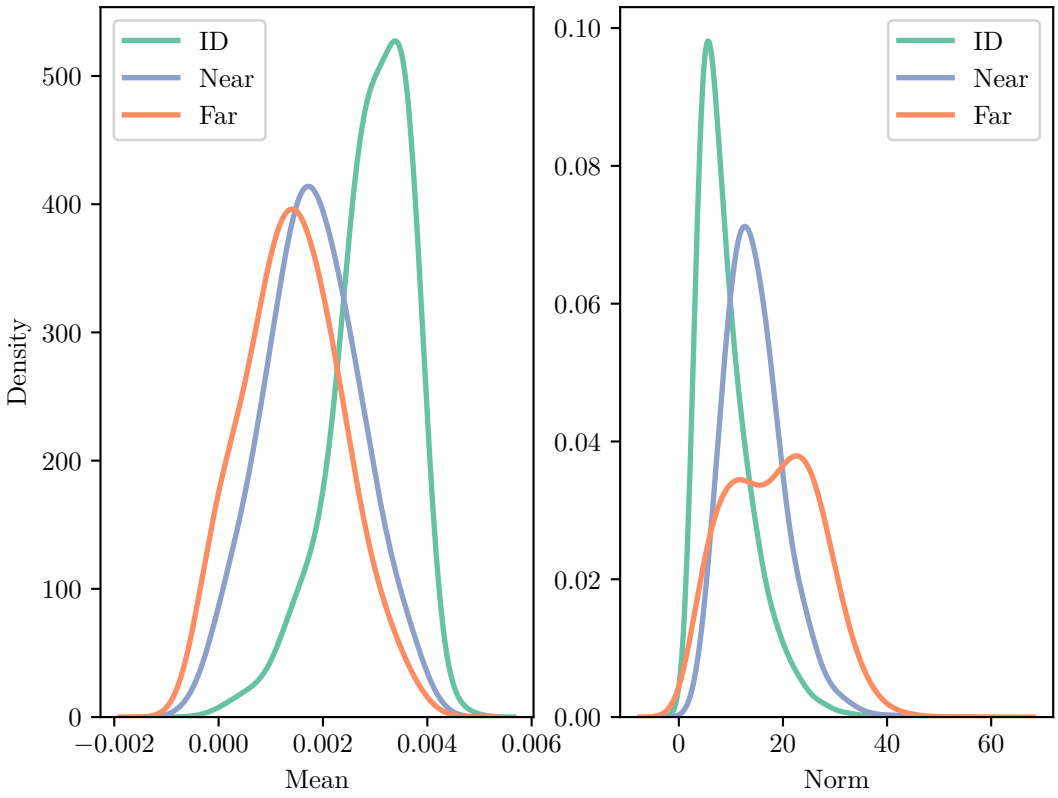

Figure 4.10: Barplot of average AUROC scores on ImageNet200. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

GBP

GBP performs decently, with the best saliency aggregation method (vector norm) achieving scores which are a few percentage points below the baselines. The only surprise here is that the mean is lower on average for ID data, which it also was when applying GBP to ImageNet200.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean↓	Median	Norm	Range	Max	Q3	CV↓	RMD↓	QCD
Near-OOD AUROC	86.7	87.7	56.0	58.6	**83.4**	81.2	80.6	65.5	53.1	64.6	51.8
Far-OOD AUROC	91.4	90.8	50.0	70.6	**89.6**	88.7	87.5	79.1	61.6	69.2	60.9
Correlation with MLS	-	-	-0.11	0.03	0.30	0.22	0.22	0.18	-0.04	-0.03	-0.02
Correlation with MSP	-	-	-0.08	0.01	0.21	0.15	0.15	0.11	-0.02	-0.02	-0.01

Table 4.11: AUROC scores for GBP on CIFAR10. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

Overall results on CIFAR10

Figure 4.11 shows the average scores for each aggregation over the 5 different XAI saliency methods. From this we see that again, vector norm performs the best overall, and occlusion the worst.

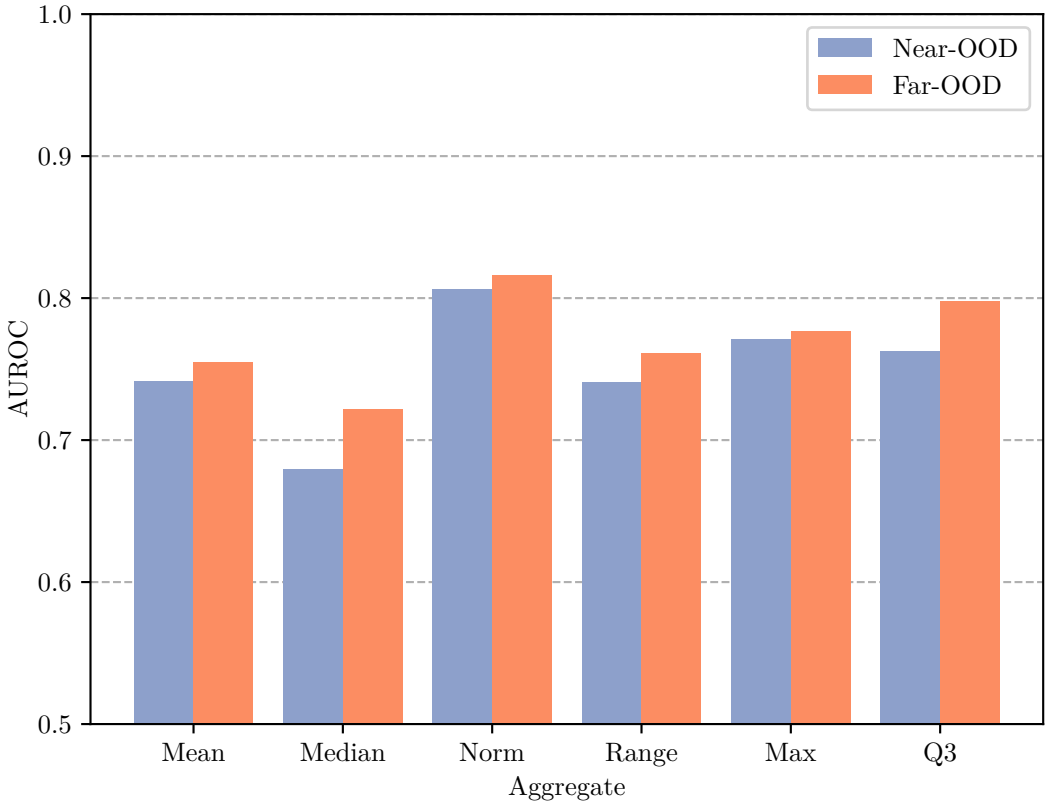

Figure 4.11: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Table 4.12 shows the results in more detail. Here, we see that not only is the vector norm the best when averaging across Near and Far, but also individually on each of the categories as well.

Aggregation type	Baselines		Magnitude of saliencies						Statistical dispersion		
Aggregate	MLS	MSP	Mean	Median	Norm	Range	Max	Q3	CV	RMD	QCD
Near-OOD AUROC	86.7	87.7	74.1	68.0	**80.6**	74.1	77.1	76.2	65.8	61.6	57.6
Far-OOD AUROC	91.4	90.8	75.4	72.1	**81.6**	76.1	77.7	79.8	66.2	65.0	58.2
Mean AUROC	89.1	89.2	74.8	70.1	**81.1**	75.1	77.4	78.0	66.0	63.3	57.9

Table 4.12: Average AUROC scores for all XAI saliency methods on CIFAR10. The highest non-baseline value for Near- and Far-OOD is highlighted in bold. ↓ denotes that ID data points more often have a lower score with this aggregation, and thus the output values have been negated (as described in section 2.3.1)

Figure 4.12 shows the results when we instead look at the best aggregation for each XAI saliency method. Here we see that, as with ImageNet200, GradCAM, integrated gradients and GBP perform reasonably well, while occlusion performs poorly. Unlike ImageNet200, LIME does not perform particularly well on CIFAR10, and lags behind the gradient based methods.

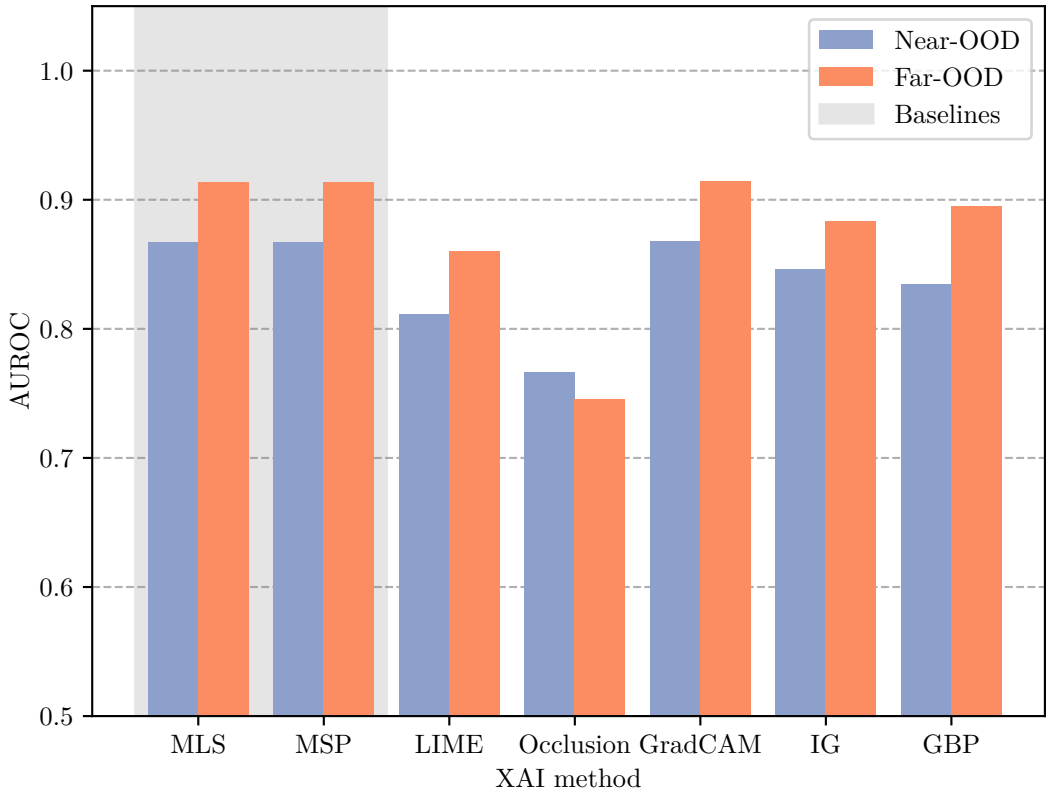

Figure 4.12: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Finally, we can look at all combinations of aggregation and XAI method when we combine both Near- and Far-OOD performance. From figure 4.13, we see that many of the combinations which performed well on ImageNet200 also perform well on CIFAR10: the vector norm with GBP and GradCAM and the mean with integrated gradients all perform very well. Contrasting with ImageNet200, we find that on CIFAR10, using mean aggregation is actually better than the vector norm on LIME saliencies. In addition, we also see that the mean leads to an every slightly higher score for GradCAM.

Figure 4.13: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

4.1.3 Overall results on both datasets

Finally, let us consider both datasets together. Figure 4.14 shows the average performance over both Near-OOD and Far-OOD for both ImageNet200 and CIFAR10. Using this heatmap, we can select the most promising combinations to use for the final OOD detectors on the test set.

Figure 4.14: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

As we can see, occlusion has consistently underperformed on both datasets, and as such I will not use this XAI method in the following sections. For the remaining four XAI methods, I select the highest performing aggregation on both of the validation sets for use on the test set. From the heatmap, we can see that this results in using vector norm for LIME, GradCAM and GBP, and mean for integrated gradients. In the next section, I will denote these combinations as LIMENorm, GradCAMNorm, IGMean and GBPNorm.

4.2 Evaluation of XAI OOD detectors

This section contains the final tests conducted on the test set, using the XAI methods that performed the best on the validation set, as described in the preceding section. This section will be divided into three parts, corresponding to the three proposed OOD detection methods introduced in the methodology: Saliency Aggregation, Saliency Aggregation plus Logits and SaliencyVIM. Each of these sections will detail the performance of the corresponding method on the test set, and statistically compare the results to the baseline methods. The statistical analysis is based on ten bootstraps of the test set, which has been performed on each method as well as the baselines. As mentioned in section 3.5.3, the Bonferroni-corrected threshold for statistical significance

is set at $0.05/n$, where n is the number of experiments conducted on each method.

When reporting the p-values, I follow the `R`-standard for appending "significance stars", which give a quick visual indication of the statistical significance of a result. In `R`, any p-value lower than 0.05 has an asterisk appended, any p-value lower than 0.01 has two asterisks appended, and any p-value lower than 0.001 has three asterisks appended. I also append these asterisks, but use Bonferroni corrected p-values when determining if a value should have an asterisk appended. This means that whenever a p-value has at least one asterisk appended, the reader knows that the corresponding t-test showed statistical significance according to a Bonferroni corrected level of significance of $0.05/n$.

4.2.1 Results for Saliency Aggregation

Saliency Aggregation, as described in 3.1.1, is the first OOD detection method that will be tested. As we have seen from the validation set, saliency aggregation on its own can sometimes perform on par with the baselines on Near-OOD and can sometimes surpass the baselines on Far-OOD. In this section, the generation of saliency maps and aggregation will be done again on ten bootstrapped test sets, and AUROC scores will be calculated for each bootstrap.

ImageNet200

As in the preceding section, we first investigate the results on ImageNet200. Figure 4.15 shows the mean AUROC for the baselines and the three combinations of XAI methods and aggregation functions mentioned above. In addition, the confidence intervals for each mean value is plotted as whiskers.

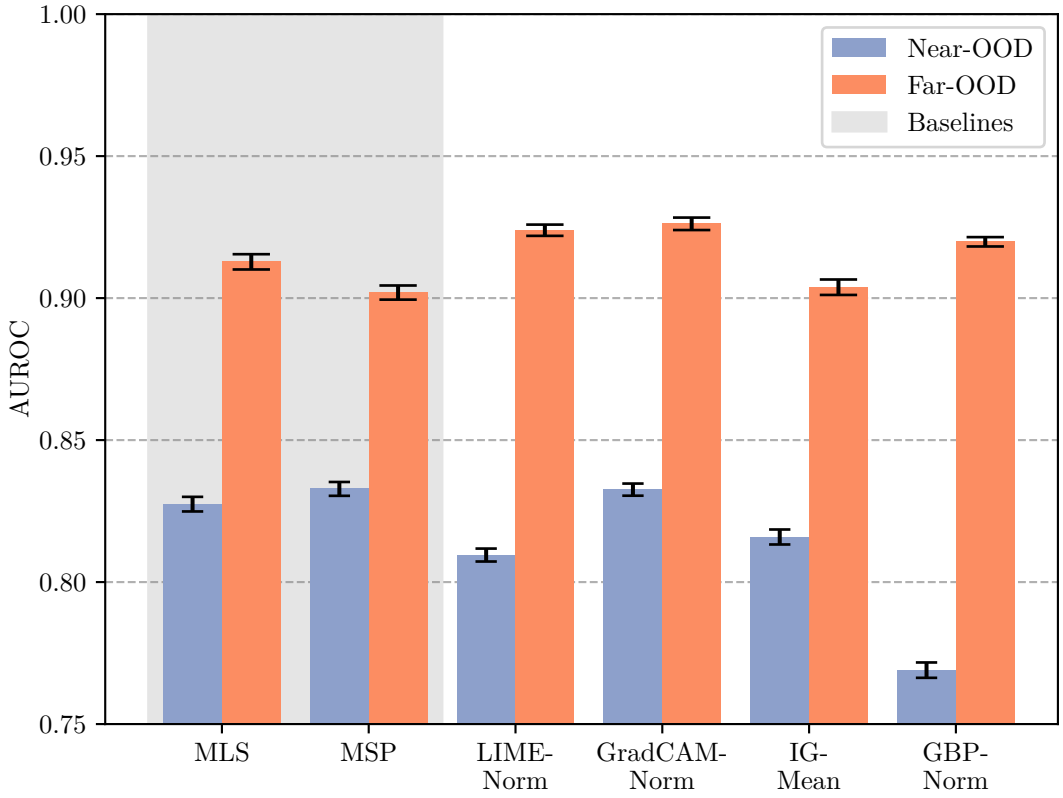

Figure 4.15: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

As we can see, the vector norm of LIME, GradCAM and GBP actually seems to outperform the baselines on Far-OOD, while the mean of the integrated gradients is about on par with the Far-OOD performance of MSP. When it comes to Near-OOD, only the norm of GradCAM seems to compete with the baselines.

To get a more precise understanding of the performance of the different models, we turn to table 4.13, which shows the results of the t-tests done against the baseline methods. From this table we see that the Far-OOD results for LIMENorm, GradCAMNorm and GBPNorm were indeed statistically significantly higher than both baselines. These results show that XAI explanations can be used to perform OOD detection, and may actually beat baseline methods in certain settings.

Dataset	AUROC	ΔAUROC MLS	P-value MLS	ΔAUROC MSP	P-value MSP
LIMENorm					
Near-OOD	80.95	-1.792	1.000	-2.329	1.000
Far-OOD	92.39	+1.113	5.3e-07 ***	+2.195	7.1e-12 ***
GradCAMNorm					
Near-OOD	83.26	+0.511	0.002 **	-0.026	0.567
Far-OOD	92.62	+1.339	6.7e-08 ***	+2.421	2.8e-12 ***
IGMean					
Near-OOD	81.59	-1.156	1.000	-1.694	1.000
Far-OOD	90.38	-0.894	1.000	+0.188	0.142
GBPNorm					
Near-OOD	76.90	-5.845	1.000	-6.382	1.000
Far-OOD	91.98	+0.706	6.6e-05 ***	+1.788	7.3e-11 ***

Table 4.13: Results of performing a t-test on the AUROC means of against MLS and MSP, showing the mean AUROC over 10 runs on ImageNet200, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

However, we see that on Near-OOD, the performance of Saliency Aggregation falls short, with no method seeing statistically significant improvements over both baselines.

In general, these results show that the simple baseline method of saliency aggregation, as introduced in section 3.1.1, is able to discriminate between ID and OOD samples. The performance on ImageNet200 is lower on Near-OOD for most combinations, but the Far-OOD performance is actually higher for all combinations except for IGMean.

CIFAR10

Next, we turn our attention to CIFAR10. Figure 4.16 shows the bootstrapped means and the confidence intervals. Here, we can see that the results are in general worse than the baselines, mirroring the initial findings from the validation dataset (section 4.1.2). In this case, only GradCAMNorm can compete with the baseline methods.

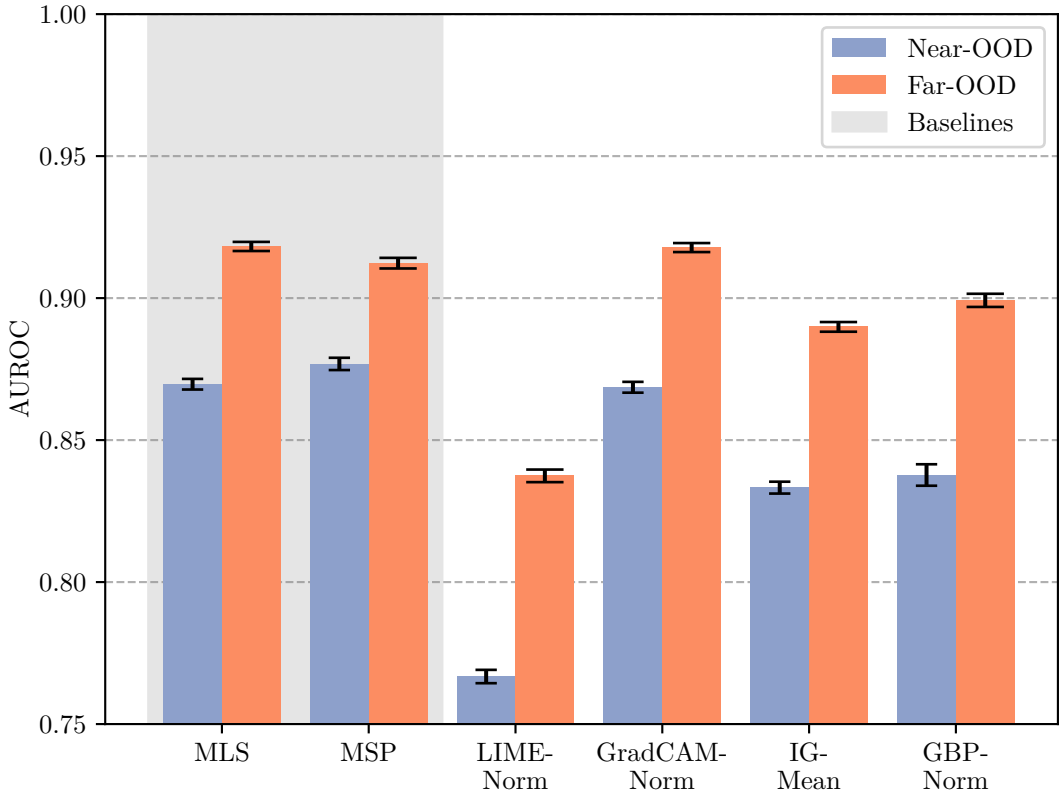

Figure 4.16: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

From this plot, we do not expect to see many statistically significant improvements. Indeed, table 4.14 shows that no method outperforms both baselines on either Near- or Far-OOD, and in general all methods have a mean AUROC which is far lower than the baseline methods.

Dataset	AUROC	ΔAUROC MLS	P-value MLS	ΔAUROC MSP	P-value MSP
LIMENorm					
Near-OOD	76.67	-10.294	1.000	-11.008	1.000
Far-OOD	83.74	-8.078	1.000	-7.490	1.000
GradCAMNorm					
Near-OOD	86.86	-0.108	0.803	-0.822	1.000
Far-OOD	91.78	-0.039	0.644	+0.549	6.5e-05 ***
IGMean					
Near-OOD	83.33	-3.642	1.000	-4.357	1.000
Far-OOD	88.99	-2.832	1.000	-2.244	1.000
GBPNorm					
Near-OOD	83.77	-3.197	1.000	-3.912	1.000
Far-OOD	89.92	-1.898	1.000	-1.310	1.000

Table 4.14: Results of performing a t-test on the AUROC means of against MLS and MSP, showing the mean AUROC over 10 runs on CIFAR10, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

As we can see, the performance of saliency aggregation on CIFAR10 is far less impressive than that on ImageNet200. However, the corresponding results on CIFAR10 from [7] was an AUROC of 0.52 when using SVHN as the OOD dataset, meaning that all these methods vastly outperform this work, even if they do not beat the baseline methods.

CIFAR100

Next, we turn to CIFAR100.

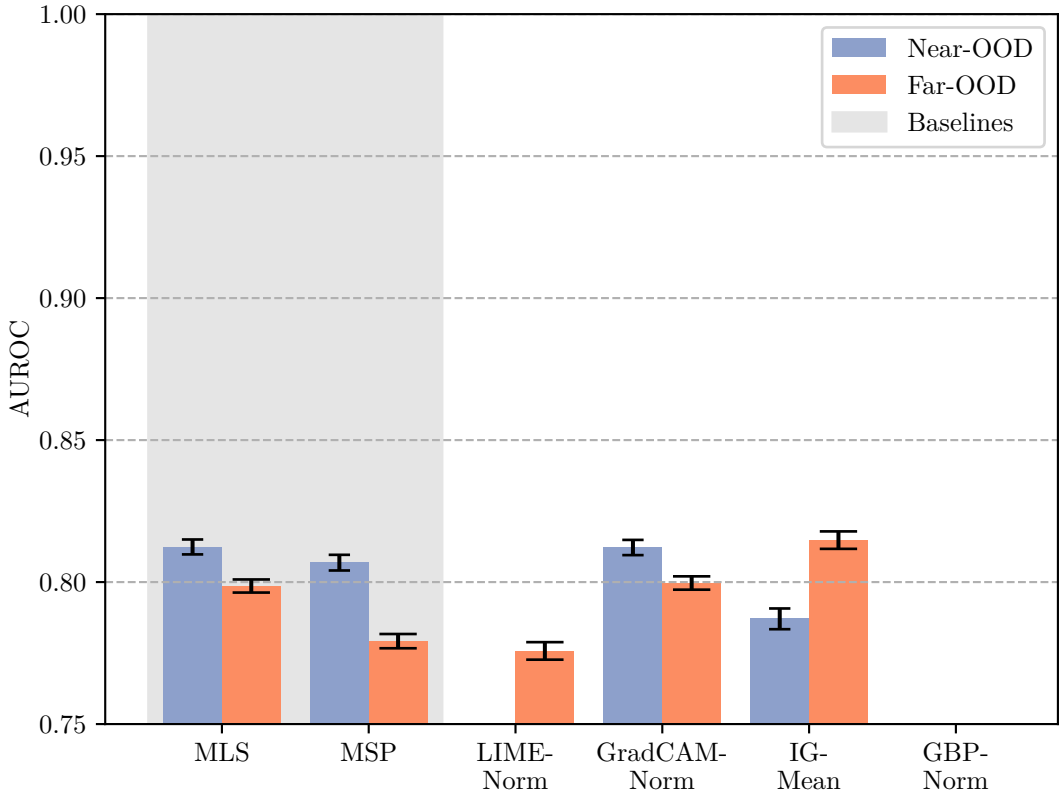

Figure 4.17: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Dataset	AUROC	ΔAUROC MLS	P-value MLS	ΔAUROC MSP	P-value MSP
LIMENorm					
Near-OOD	70.51	-10.728	1.000	-10.175	1.000
Far-OOD	77.58	-2.285	1.000	-0.344	0.961
GradCAMNorm					
Near-OOD	81.22	-0.020	0.546	+0.533	0.004 *
Far-OOD	79.97	+0.106	0.248	+2.047	8.1e-11 ***
IGMean					
Near-OOD	78.71	-2.532	1.000	-1.979	1.000
Far-OOD	81.48	+1.616	1.9e-08 ***	+3.557	8.3e-14 ***
GBPNorm					
Near-OOD	64.20	-17.039	1.000	-16.486	1.000
Far-OOD	71.35	-8.509	1.000	-6.568	1.000

Table 4.15: Results of performing a t-test on the AUROC means of against MLS and MSP, showing the mean AUROC over 10 runs on CIFAR100, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

Overall results of Saliency Aggregation

Overall, the results from simply aggregating the saliency maps of different XAI methods are very promising, especially considering that the only other work which has attempted to use XAI saliency maps for OOD detection barely achieved AUROC scores above 0.50. These results show that XAI methods extract valuable information from the network, which can be used to effectively discriminate between ID and OOD data samples. The results on ImageNet200 are particularly interesting, as LIMENorm, GradCAMNorm and GBPNorm all outperform the baselines on Far-OOD. The results on CIFAR10 are less impressive, and show that these methods may not achieve consistently good results on all datasets. However, it should be noted that this inconsistency is not unusual amongst OOD detectors, as one of the takeaways from from [52] was that there is "no single winner that always outperforms others across multiple data sets".

4.2.2 Results for Saliency Aggregation Plus Logits

Given the fact that Saliency Aggregation has shown itself to be capable of differentiating ID and OOD samples in many cases, especially for Far-OOD, we might expect good results from Saliency Aggregate Plus Logits. As mentioned in section 3.1.2, [39] achieved SoTA results on ImageNet200 when combining two complementary distance metrics. As we have seen from section 4.2.1, most XAI methods (except for GradCAM) have relatively low correlation with either MSP or MLS, which could give similar benefits as those found by `COMBOOD`.

ImageNet200

Figure 4.18 shows the results of the 10 bootstraps. Right away we can see that the poor performances on Near-OOD have mostly disappeared when combining saliency aggregation with MLS. These results concur with those found by [39], which also

found that the subpar Near-OOD performance of their Mahalanobis OOD detector was improved when combined with nearest neighbour OOD detection.

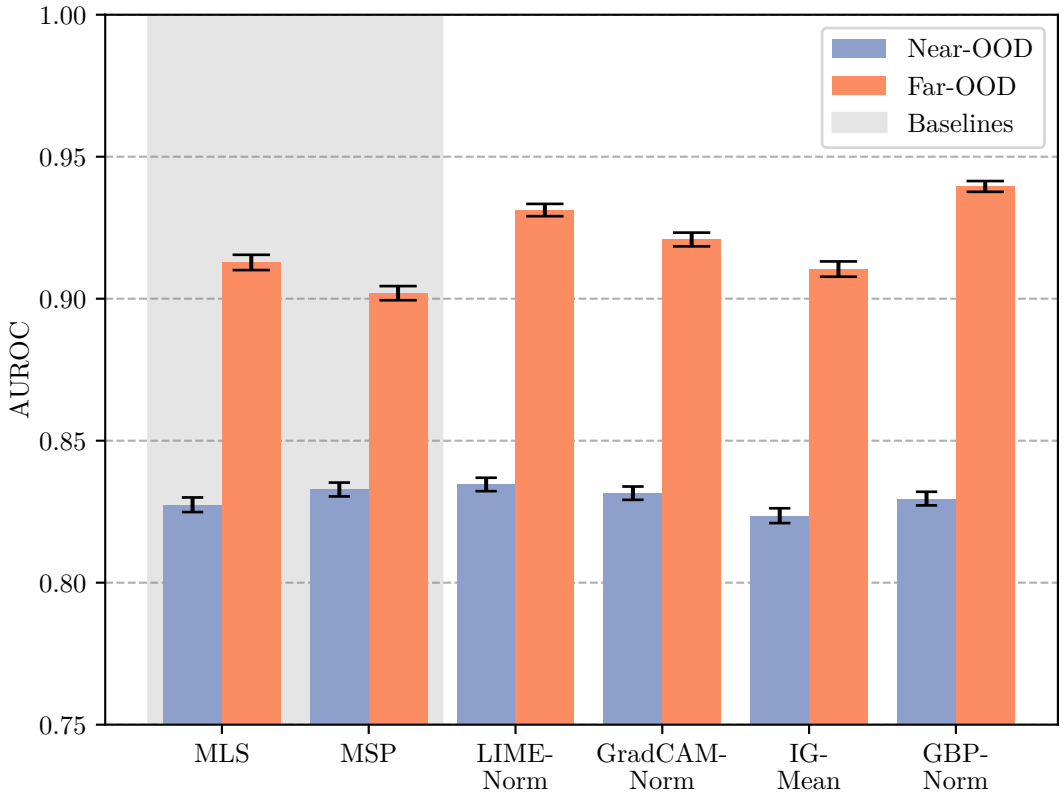

Figure 4.18: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

In addition to the Near-OOD improvements, GBPNorm+Logits achieves a very impressive Far-OOD performance on ImageNet200. LIMENorm+Logits substantially improves on LIMENorm, and also beats the baselines on Far-OOD. GradCAMNorm+Logits also outperforms the baselines, but we should remember that GradCAMNorm (without the addition of logits) also performed well, so these results may not be better than simply using GradCAMNorm alone. Let us look closer at the performance of each method, considering the results of the t-tests performed against the baseline methods.

As we see from table 4.16, we indeed see far more improvements when combining our saliency aggregation method with the MLS. Where we previously saw multiple percentage points lower Near-OOD on several methods with Saliency Aggregation, we now have no method which performs worse than a single percentage point than either baseline. The addition of a second metric seems to have had a stabilizing effect on the detection performance in addition to increasing the performance in some cases.

Dataset	AUROC	ΔAUROC MLS	P-value MLS	ΔAUROC MSP	P-value MSP
LIMENorm					
Near-OOD	83.46	+0.715	1.6e-04 ***	+0.178	0.136
Far-OOD	93.12	+1.845	4.6e-10 ***	+2.927	9.8e-14 ***
GradCAMNorm					
Near-OOD	83.15	+0.407	0.010 *	-0.130	0.791
Far-OOD	92.09	+0.808	6.7e-05 ***	+1.890	3.4e-10 ***
IGMean					
Near-OOD	82.36	-0.384	0.982	-0.922	1.000
Far-OOD	91.04	-0.233	0.899	+0.848	4.6e-05 ***
GBPNorm					
Near-OOD	82.96	+0.217	0.099	-0.320	0.971
Far-OOD	93.95	+2.677	4.3e-13 ***	+3.759	5.0e-16 ***

Table 4.16: Results of performing a t-test on the AUROC means of against MLS and MSP, showing the mean AUROC over 10 runs on ImageNet200, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

For LIMENorm+Logits, we see that the poor Near-OOD performance is improved substantially, leading to statistically significant improvements over MLS (but not over MSP). In addition, the Far-OOD performance is improved further, giving us an OOD detector which performs well on both Near- and Far-OOD.

The performance of GradCAMNorm+Logits is good, but not any better than simply using GradCAMNorm saliency aggregation. This is not unexpected, given the extremely high correlation between GradCAM saliency aggregations and the MLS. There is little to be gained from combining two metrics which are almost entirely the same, as we do not gain any supplementary information over just using one of them. [39] found best results when using different feature extraction strategies for the two distance metrics, which allows for complementary information to be extracted by each metric. With such a high correlation, it is not surprising that no performance was gained when combining MLS and GradCAM vector norms.

For IGMean+Logits, the performance is increased over IGMean on both Near- and Far-OOD. This is most likely due to the extra information which is being added by including the logit. However, the performance is not better than simply using MLS. This is surprising, given that the performance of LIMENorm was far worse than IGMean, yet its performance when combined with MLS was far higher.

Finally, for GBPNorm+Logits, the improvement is considerable, especially on Far-OOD, where the performance is actually quite close to the SoTA models, which is around 0.95. However, the Near-OOD performance is not statistically significantly better than either MLS or MSP.

CIFAR10

Next, we turn to CIFAR10, where GBPNorm+Logit again demonstrates a very impressive performance gain. As we can see from figure 4.19, GBPNorm+Logit not only beats the baselines in the Far-OOD category, but also on Near-OOD, where the performance is considerably higher. In addition, LIMENorm+Logits and IGMean+Logit outperform the baselines on Far-OOD.

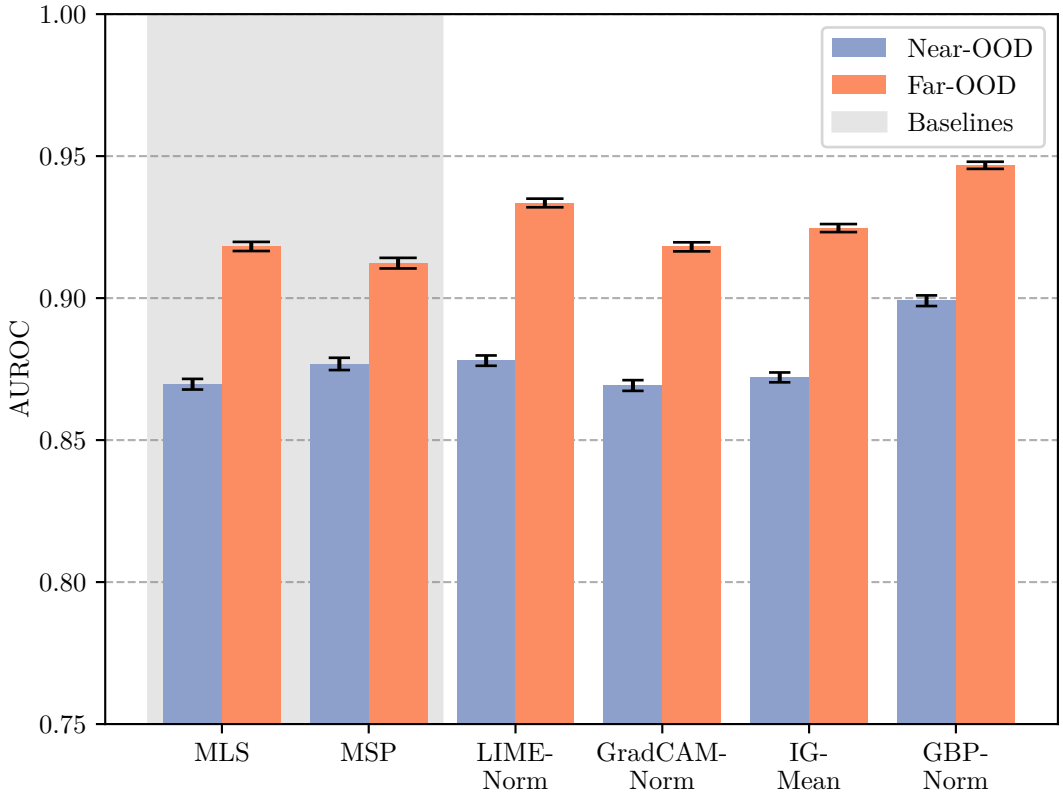

Figure 4.19: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Like in the previous sections, let look at the results of the t-tests to gain a more robust understanding of the performance of the different methods. As table 4.17 shows, we now see statistically significant results on Far-OOD for all methods except for GradCAMNorm+Logits. In addition, GBPNorm+Logits sees a large improvement on both Near- and Far-OOD datasets, leading to a classifier which is close to the SoTA amongst post-hoc OOD methods.

Dataset	AUROC	ΔAUROC MLS	P-value MLS	ΔAUROC MSP	P-value MSP
LIMENorm					
Near-OOD	87.80	+0.831	9.7e-07 ***	+0.116	0.193
Far-OOD	93.35	+1.532	6.6e-12 ***	+2.120	1.1e-13 ***
GradCAMNorm					
Near-OOD	86.92	-0.046	0.643	-0.760	1.000
Far-OOD	91.81	-0.013	0.550	+0.575	4.1e-05 ***
IGMean					
Near-OOD	87.21	+0.240	0.028	-0.475	0.999
Far-OOD	92.47	+0.647	1.9e-06 ***	+1.236	5.7e-10 ***
GBPNorm					
Near-OOD	89.91	+2.940	2.0e-15 ***	+2.225	9.4e-13 ***
Far-OOD	94.68	+2.858	3.3e-17 ***	+3.446	7.0e-18 ***

Table 4.17: Results of performing a t-test on the AUROC means of against MLS and MSP, showing the mean AUROC over 10 runs on CIFAR10, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

Next, we inspect the performance of GradCAMNorm+Logit. As in the previous section, the high correlation between the MLS and GradCAM saliencies means that this method does not perform any better than either MLS or GradCAMNorm.

Next; IGMean+Logit. Here, we see significant improvements over IGMean, as with ImageNet200. In addition, the method is statistically significantly better than both baselines on Far-OOD, and better than MLS on Near-OOD.

Finally, we inspect the results for GBPNorm+Logits, which are very impressive. Here, we see substantial improvements over both baselines of several percentage points on all datasets except for MNIST. Furthermore, these results are pretty close to the SoTA amongst methods which do not require retraining of the network, with `COMBOOD` being the leader at 91.13/94.65 Near-OOD/Far-OOD performance.

CIFAR100

Next, we turn to CIFAR100, where GBPNorm+Logit again demonstrates a very impressive performance gain. As we can see from figure 4.19, GBPNorm+Logit not only beats the baselines in the Far-OOD category, but also on Near-OOD, where the performance is considerably higher. In addition, LIMENorm+Logits and IGMean+Logit outperform the baselines on Far-OOD.

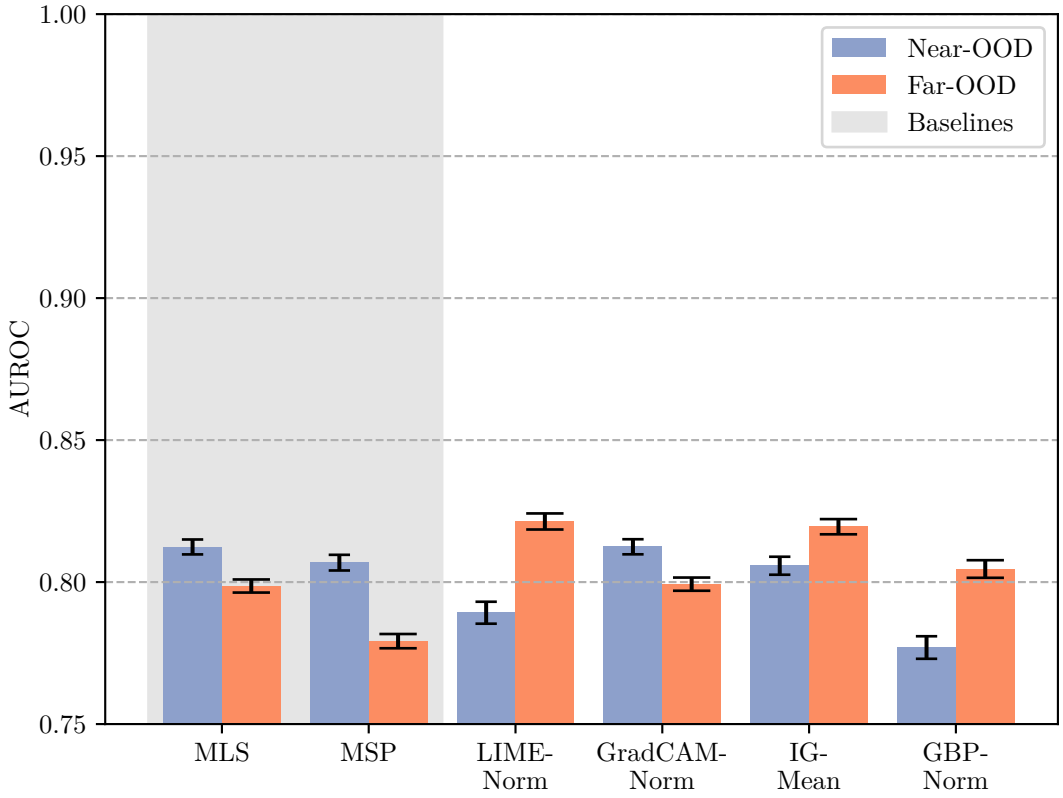

Figure 4.20: Barplot of bootstrapped AUROC scores on CIFAR100.

Dataset	AUROC	ΔAUROC MLS	P-value MLS	ΔAUROC MSP	P-value MSP
LIMENorm					
Near-OOD	78.92	-2.316	1.000	-1.763	1.000
Far-OOD	82.13	+2.273	3.9e-11 ***	+4.214	1.9e-15 ***
GradCAMNorm					
Near-OOD	81.24	+0.006	0.486	+0.559	0.003 *
Far-OOD	79.93	+0.066	0.335	+2.007	1.0e-10 ***
IGMean					
Near-OOD	80.58	-0.662	0.999	-0.108	0.708
Far-OOD	81.95	+2.089	9.0e-11 ***	+4.030	2.4e-15 ***
GBPNorm					
Near-OOD	77.70	-3.542	1.000	-2.989	1.000
Far-OOD	80.46	+0.599	0.002 **	+2.540	2.7e-11 ***

Table 4.18: Results of performing a t-test on the AUROC means of against MLS and MSP, showing the mean AUROC over 10 runs on CIFAR100, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

4.2.3 Results for SaliencyVIM

For the final tests, we consider SaliencyVIM, the method described in section 3.1.2. As described in section 3.1.2, SaliencyVIM requires saliency methods which output lower dimensional saliency maps, as opposed to methods such as GBP and integrated gradients, which output a value for each pixel. Thus, the applicable XAI methods included in this thesis are occlusion, LIME and GradCAM. Although occlusion performed very poorly on the validation set when aggregating saliencies, SaliencyVIM uses every saliency value and could thus capture information from this XAI method which has not been captured by aggregation methods. Thus, I include occlusion in the testing.

In this case, it does not make sense to use MLS or MSP as the baselines to compare against, since the method is based on VIM. Instead, we compare our results with VIM, to see whether the addition of saliencies can improve this method.

ImageNet200

We start with ImageNet200. From figure 4.21, we see that the differences are extremely small between the baseline and the XAI methods. However, the confidence intervals are also very small, which might mean that there are statistically significant performance increases. Thus, we look to the results of the t-tests.

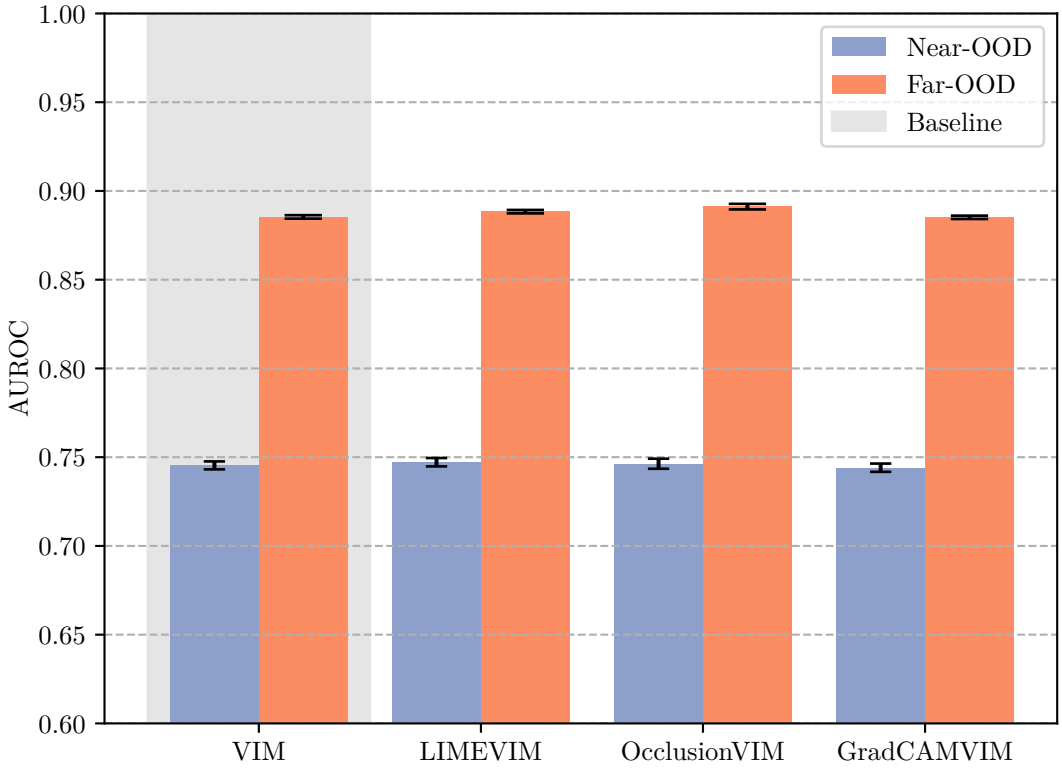

Figure 4.21: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

From table 4.19, we see that both OcclusionVIM and LIMEVIM outperform VIM on Far-OOD. These results are interesting, because they show that even though aggregation of occlusion saliencies had very little discriminative power, their inclusion directly into the PCA based VIM method leads to increased performance.

Dataset	AUROC	ΔAUROC VIM	P-value VIM
LIMEVIM			
Near-OOD	74.72	+0.178	0.126
Far-OOD	88.83	+0.298	9.6e-05 ***
OcclusionVIM			
Near-OOD	74.63	+0.094	0.291
Far-OOD	89.12	+0.580	1.2e-06 ***
GradCAMVIM			
Near-OOD	74.41	-0.130	0.803
Far-OOD	88.51	-0.022	0.634

Table 4.19: Results of performing a t-test on the AUROC means of against VIM, showing the mean AUROC over 10 runs on ImageNet200, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

We see that for ImageNet200, GradCAMVIM sees no increases in performance over the baseline, and no method shows any statistically significant improvement over the baseline on Near-OOD. At this point, we can note a clear pattern emerging over the three methods, where it seems that XAI methods perform far better on Far-OOD than Near-OOD.

CIFAR10

For the final tests of this chapter, we perform OOD detection using VIM and SaliencyVIM on CIFAR10. From figure 4.22, SaliencyVIM with GradCAM as the saliency generator seems to be the best performer here, above the baseline. Occlusion and LIME on the other hand, seems to perform worse. These results are surprising, as they are reversed when compared to ImageNet200. Clearly, XAI saliency methods are not entirely consistent over different datasets, as was noted with the previous methods as well. We turn our attention to the results of the t-tests.

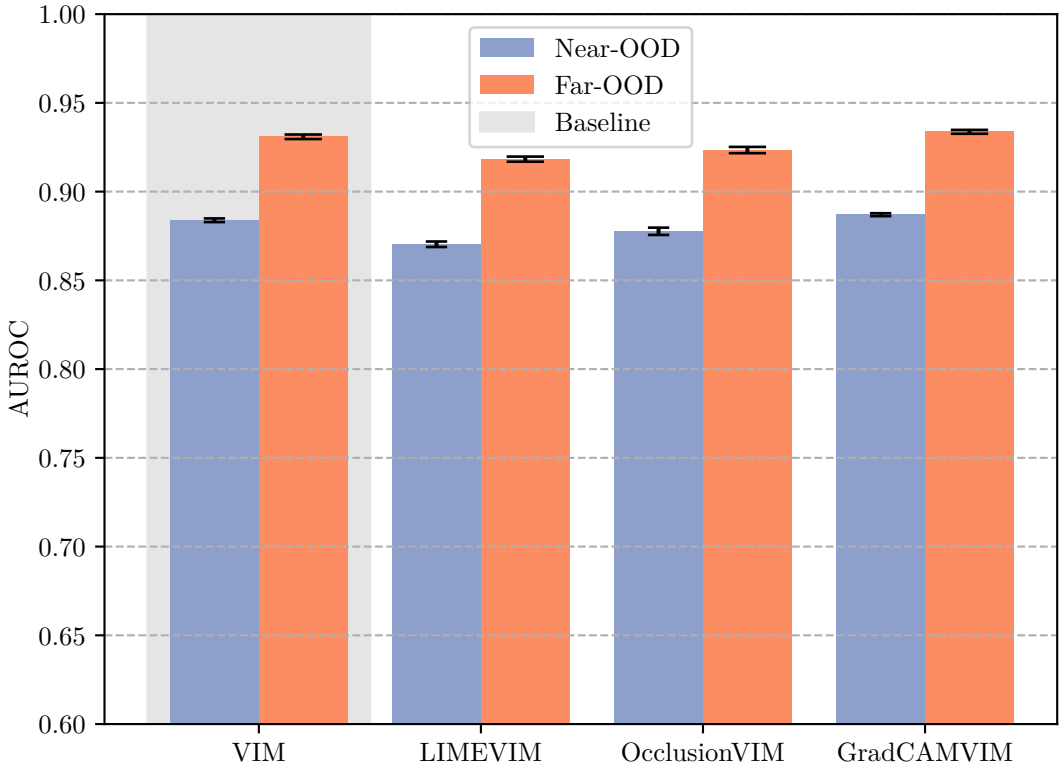

Figure 4.22: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

As we see from table 4.20, GradCAMVIM outperforms the baseline on both Near- and Far-OOD. Both occlusion and LIME see drops in performance across the board, contrary to the results on ImageNet200.

Dataset	AUROC	ΔAUROC VIM	P-value VIM
LIMEVIM			
Near-OOD	87.03	-1.352	1.000
Far-OOD	91.83	-1.260	1.000
OcclusionVIM			
Near-OOD	87.76	-0.621	1.000
Far-OOD	92.34	-0.747	1.000
GradCAMVIM			
Near-OOD	88.70	+0.315	3.1e-05 ***
Far-OOD	93.37	+0.280	0.001 **

Table 4.20: Results of performing a t-test on the AUROC means of against VIM, showing the mean AUROC over 10 runs on CIFAR10, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

CIFAR100

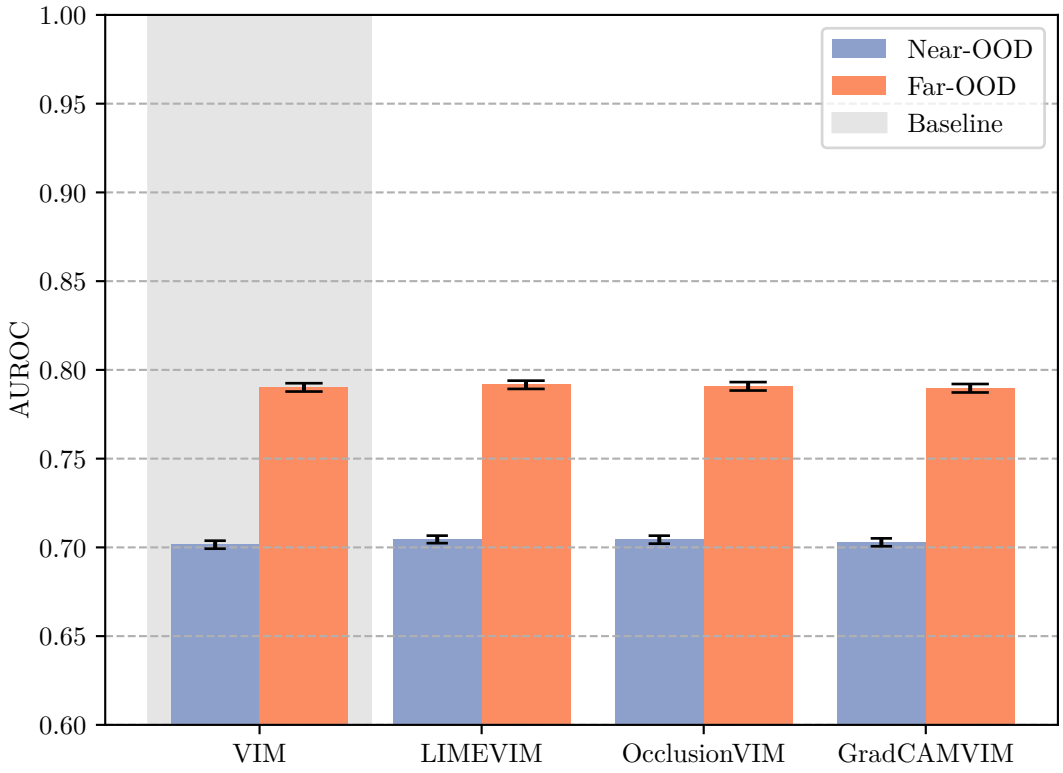

Figure 4.23: Barplot of average AUROC scores on CIFAR10. Note that the y-axis starts at 0.50, the practical floor for an AUROC score.

Dataset	AUROC	ΔAUROC VIM	P-value VIM
LIMEVIM			
Near-OOD	70.45	+0.298	0.026
Far-OOD	79.16	+0.147	0.175
OcclusionVIM			
Near-OOD	70.44	+0.283	0.035
Far-OOD	79.08	+0.060	0.350
GradCAMVIM			
Near-OOD	70.29	+0.134	0.187
Far-OOD	78.97	-0.048	0.620

Table 4.21: Results of performing a t-test on the AUROC means of against VIM, showing the mean AUROC over 10 runs on CIFAR100, the difference in means compared to the baselines, and the corresponding p-values. Each p-value is appended a significance code which follows the R-standard.

Overall results of SaliencyVIM

Overall, the results of integrating XAI saliencies directly into the already developed VIM OOD detector shows potential, but is not consistently better across both datasets.

4.3 Summary

In this chapter, I have presented the results from inspecting saliency aggregation over the validation dataset, as well results from the three different XAI based OOD detection algorithms I introduced in chapter 3. As we have seen, the results are very promising, and show that there is definite potential for using XAI methods for OOD detection. In general, XAI methods show better performance on Far-OOD than Near-OOD, but impressive performance is attained in both scenarios on several occasions. GBPNorm+Logits is one such example, performing far above the baseline on ImageNet200 and almost at the level of SoTA methods on CIFAR10 (amongst methods which do not require retraining).

Chapter 5

Discussion

This chapter will discuss the findings of chapter 4 in more depth. As we have now seen, XAI methods can indeed perform OOD detection at a level competitive with OOD detection baselines, and can in some cases even perform close to SoTA models amongst methods which do not retrain the underlying classifying network. The computational overhead of such methods is minimal, and good performance can be achieved without retraining.

Chapter 6

Conclusion

To conclude, it really does work.

6.1 Future work

Do it on more realistic datasets.

Bibliography

[1] Shaveta Dargan et al. 'A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning'. In: *Archives of Computational Methods in Engineering* 27.4 (Sept. 2020), pp. 1071–1092. ISSN: 1886-1784. DOI: 10.1007/s11831-019-09344-w. URL: https://doi.org/10.1007/s11831-019-09344-w.

[2] Sajid Nazir, Diane M. Dickson and Muhammad Usman Akram. 'Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks'. In: *Computers in biology and medicine* 156 (2023), p. 106668. URL: https://api.semanticscholar.org/CorpusID:257067347.

[3] Alvin Rajkomar, Jeffrey Dean and Isaac Kohane. 'Machine Learning in Medicine'. In: *New England Journal of Medicine* 380.14 (2019), pp. 1347–1358. DOI: 10.1056/NEJMra1814259. eprint: https://www.nejm.org/doi/pdf/10.1056/NEJMra1814259. URL: https://www.nejm.org/doi/full/10.1056/NEJMra1814259.

[4] Jordan Zheng Ting Sim et al. 'Machine learning in medicine: what clinicians should know'. en. In: *Singapore Med J* 64.2 (May 2021), pp. 91–97.

[5] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[6] Jingyang Zhang et al. 'OpenOOD v1.5: Enhanced Benchmark for Out-of-Distribution Detection'. In: *arXiv preprint arXiv:2306.09301* (2023).

[7] Aitor Martinez-Seras, Javier Del Ser and Pablo Garcia-Bringas. 'Can Post-hoc Explanations Effectively Detect Out-of-Distribution Samples?' In: *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2022, pp. 1–9. DOI: 10.1109/FUZZ-IEEE55066.2022.9882726.

[8] Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[9] F. Rosenblatt. 'The perceptron: A probabilistic model for information storage and organization in the brain.' In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519.

[10] George V. Cybenko. 'Approximation by superpositions of a sigmoidal function'. In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314. URL: https://api.semanticscholar.org/CorpusID:3958369.

[11] Yann Lecun et al. 'Gradient-Based Learning Applied to Document Recognition'. In: *Proceedings of the IEEE* 86 (Dec. 1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[12] Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. 2018. arXiv: 1610.02136 [cs.NE].

[13] Shiyu Liang, Yixuan Li and R. Srikant. *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks*. 2020. arXiv: 1706.02690 [cs.LG].

[14] Jingkang Yang et al. *Generalized Out-of-Distribution Detection: A Survey*. 2024. arXiv: 2110.11334 [cs.CV].

[15] Haoqi Wang et al. *ViM: Out-Of-Distribution with Virtual-logit Matching*. 2022. arXiv: 2203.10807 [cs.CV].

[16] B. Efron. 'Bootstrap Methods: Another Look at the Jackknife'. In: *The Annals of Statistics* 7.1 (1979), pp. 1–26. DOI: 10.1214/aos/1176344552. URL: https://doi.org/10.1214/aos/1176344552.

[17] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. arXiv: 1702.08608 [stat.ML].

[18] European Union. *Article 71: European Data Protection Board*. Accessed: February 13, 2024. 2016. URL: https://www.privacy-regulation.eu/en/r71.htm.

[19] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. Independently published, 2022. URL: https://christophm.github.io/interpretable-ml-book.

[20] Bas H.M. van der Velden et al. 'Explainable artificial intelligence (XAI) in deep learning-based medical image analysis'. In: *Medical Image Analysis* 79 (2022), p. 102470. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2022.102470. URL: https://www.sciencedirect.com/science/article/pii/S1361841522001177.

[21] Radhakrishna Achanta et al. 'SLIC Superpixels Compared to State-of-the-Art Superpixel Methods'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120.

[22] Marco Tulio Ribeiro, Sameer Singh and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG]. URL: https://arxiv.org/abs/1602.04938.

[23] Matthew D. Zeiler and Rob Fergus. 'Visualizing and Understanding Convolutional Networks'. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10590-1.

[24] Håvard Horgen Thunold et al. 'A Deep Diagnostic Framework Using Explainable Artificial Intelligence and Clustering'. In: *Diagnostics* 13.22 (2023). ISSN: 2075-4418. DOI: 10.3390/diagnostics13223413. URL: https://www.mdpi.com/2075-4418/13/22/3413.

[25] Bolei Zhou et al. *Learning Deep Features for Discriminative Localization*. 2015. arXiv: 1512.04150 [cs.CV].

[26] Ramprasaath R. Selvaraju et al. 'Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization'. In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[27] Jost Tobias Springenberg et al. *Striving for Simplicity: The All Convolutional Net*. 2015. arXiv: 1412.6806 [cs.LG]. URL: https://arxiv.org/abs/1412.6806.

[28] Leila Arras, Ahmed Osman and Wojciech Samek. 'CLEVR-XAI: A benchmark dataset for the ground truth evaluation of neural network explanations'. In: *Information Fusion* 81 (2022), pp. 14–40.

[29] Theerasarn Pianpanit et al. 'Parkinson's disease recognition using SPECT image and interpretable AI: A tutorial'. In: *IEEE Sensors Journal* 21.20 (2021), pp. 22304–22316.

[30] Mukund Sundararajan, Ankur Taly and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703.01365 [cs.LG]. URL: https://arxiv.org/abs/1703.01365.

[31] Sebastian Bach et al. 'On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation'. In: *PLOS ONE* 10.7 (July 2015), pp. 1–46. DOI: 10.1371/journal.pone.0130140. URL: https://doi.org/10.1371/journal.pone.0130140.

[32] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[33] Peng Cui and Jinjia Wang. 'Out-of-Distribution (OOD) Detection Based on Deep Learning: A Review'. In: *Electronics* 11.21 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11213500. URL: https://www.mdpi.com/2079-9292/11/21/3500.

[34] Weitang Liu et al. *Energy-based Out-of-distribution Detection*. 2021. arXiv: 2010.03759 [cs.LG].

[35] Dan Hendrycks et al. *Scaling Out-of-Distribution Detection for Real-World Settings*. 2022. arXiv: 1911.11132 [cs.CV]. URL: https://arxiv.org/abs/1911.11132.

[36] Rui Huang, Andrew Geng and Yixuan Li. *On the Importance of Gradients for Detecting Distributional Shifts in the Wild*. 2021. arXiv: 2110.00218 [cs.LG].

[37] Matthew Cook, Alina Zare and Paul Gader. *Outlier Detection through Null Space Analysis of Neural Networks*. 2020. arXiv: 2007.01263 [cs.LG].

[38] Ibrahima Ndiour, Nilesh Ahuja and Omesh Tickoo. *Out-Of-Distribution Detection With Subspace Techniques And Probabilistic Modeling Of Features*. 2020. arXiv: 2012.04250 [cs.LG].

[39] Shashi Shekhar et al. *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. Ed. by Shashi Shekhar et al. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2024. DOI: 10.1137/1.9781611978032. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611978032. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611978032.

[40] Yiyou Sun et al. *Out-of-Distribution Detection with Deep Nearest Neighbors*. 2022. arXiv: 2204.06507 [cs.LG]. URL: https://arxiv.org/abs/2204.06507.

[41] Kimin Lee et al. *A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks*. 2018. arXiv: 1807.03888 [stat.ML]. URL: https://arxiv.org/abs/1807.03888.

[42] Eoin Delaney, Derek Greene and Mark T. Keane. *Uncertainty Estimation and Out-of-Distribution Detection for Counterfactual Explanations: Pitfalls and Solutions*. 2021. arXiv: 2107.09734 [cs.LG]. URL: https://arxiv.org/abs/2107.09734.

[43] John Sipple and Abdou Youssef. 'A General-Purpose Method for Applying Explainable AI for Anomaly Detection'. In: *Foundations of Intelligent Systems*. Ed. by Michelangelo Ceci et al. Cham: Springer International Publishing, 2022, pp. 162–174. ISBN: 978-3-031-16564-1.

[44] AJ Tallón-Ballesteros and C Chen. 'Explainable AI: Using Shapley Value to Explain Complex Anomaly Detection ML-Based Systems'. In: *Machine Learning and Artificial Intelligence: Proceedings of MLIS 2020* 332 (2020), p. 152.

[45] Osvaldo Arreche et al. 'E-XAI: Evaluating Black-Box Explainable AI Frameworks for Network Intrusion Detection'. In: *IEEE Access* 12 (2024), pp. 23954–23988. DOI: 10.1109/ACCESS.2024.3365140.

[46] Basim Mahbooba et al. 'Explainable Artificial Intelligence (XAI) to Enhance Trust Management in Intrusion Detection Systems Using Decision Tree Model'. In: *Complexity* 2021.1 (2021), p. 6634811. DOI: https://doi.org/10.1155/2021/6634811. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2021/6634811. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/6634811.

[47] Erzhena Tcydenova et al. 'Detection of adversarial attacks in AI-based intrusion detection systems using explainable AI'. In: *Human-Centric Comput Inform Sci* 11 (2021).

[48] Tahmina Zebin, Shahadate Rezvy and Yuan Luo. 'An Explainable AI-Based Intrusion Detection System for DNS Over HTTPS (DoH) Attacks'. In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 2339–2349. DOI: 10.1109/TIFS.2022.3183390.

[49] Anh Nguyen, Jason Yosinski and Jeff Clune. *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*. 2015. arXiv: 1412.1897 [cs.CV]. URL: https://arxiv.org/abs/1412.1897.

[50] Robert Geirhos et al. 'Shortcut learning in deep neural networks'. In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.

[51] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML]. URL: https://arxiv.org/abs/1412.6572.

[52] Jingyang Zhang et al. 'OpenOOD v1.5: Enhanced Benchmark for Out-of-Distribution Detection'. In: *arXiv preprint arXiv:2306.09301* (2023).

[53] Alex Krizhevsky. 'Learning Multiple Layers of Features from Tiny Images'. In: (2009), pp. 32–33. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[54] Samira Pouyanfar et al. 'A survey on deep learning: Algorithms, techniques, and applications'. In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–36.

[55] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[56] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[57] Walid Bousselham et al. *LeGrad: An Explainability Method for Vision Transformers via Feature Formation Sensitivity*. 2025. arXiv: 2404.03214 [cs.CV]. URL: https://arxiv.org/abs/2404.03214.

[58] Pascal Sturmfels, Scott Lundberg and Su-In Lee. 'Visualizing the Impact of Feature Attribution Baselines'. In: *Distill* (2020). https://distill.pub/2020/attribution-baselines. DOI: 10.23915/distill.00022.

[59] Giang Nguyen et al. 'Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey'. In: *Artificial Intelligence Review* 52 (2019), pp. 77–124.

[60] Narine Kokhlikyan et al. *Captum: A unified and generic model interpretability library for PyTorch*. 2020. arXiv: 2009.07896 [cs.LG].

[61] Jacob Gildenblat and contributors. *PyTorch library for CAM methods*. https://github.com/jacobgil/pytorch-grad-cam. 2021.