A survey of Explainable AI and Out-of-Distribution detection methods

Jonatan Hoffmann Hanssen
Master Student, Department of Informatics
The Faculty of Mathematics and Natural Sciences
Oslo, Norway
jonatahh@ifi.uio.no

Abstract

The field of Deep Learning (DL) has seen an impressive rise in performance in the last decade, leading to widespread adoption for a variety of different tasks and fields. However, the use of DL methods is not without problems, especially when used in safety-critical applications. Two of these problems are the inherent inexplainability of DL models, and their inability to detect when a data point is too different from their training data and thus unlikely to be predicted correctly by the model. These shortcomings give rise to the fields of Explainable Artificial Intelligence (XAI) and Out-of-Distribution (OOD) Detection. In this essay, I will give a short introduction to these fields and look in detail at a selection of methods from both fields.

I. Introduction

Machine Learning generally, and Deep Learning specifically, have seen a tremendous increase in performance in recent years, performing comparable to humans in many tasks, for example image classification, speech recognition and many others [Dargan et al., 2020]. In medicine, deep learning has the potential to provide faster and more accurate detection of diseases by being trained on thousands of previous patients [Nazir et al., 2023].

However, deep learning methods are not without their flaws. Firstly, deep neural networks are inherently unexplainable due to the large number of parameters that any non-trivial network has. State of the art 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. 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 [Szegedy et al., 2014] has shown, it is possible to create data points which are imperceptibly different from normal data points, yet still fool otherwise high performing models.

These two problems lead to the two fields of machine learning which I shall discuss in this essay. These are Explainable Artificial Intelligence (XAI), and Out-of-Distribution (OOD) detection.

A. Terminology

Before proceeding, this small section will go through the choice of terminology used in this paper.

Explainability and *interpretability* are sometimes used to mean slightly different things, and some may place a distinction between *interpretable* and *explainable* machine learning. In this paper, I shall use the term *Explainable AI* broadly, and will not make this distinction.

I assume the reader is familiar with both machine learning in general and deep learning specifically. Whenever I use the term *model*, I typically mean a neural network, or other ML models such as Decision Trees or Support Vector Machines. When I use the term *network*, I exclusively mean a neural network, and not any other ML model. This essay will primarily deal with methods applied to neural networks.

II. Explainable AI

Below follows a thourough introduction to XAI, as well as detailed look at some important methods for explainability for neural networks applied to images.

A. The motivation for XAI

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 [Doshi-Velez and Kim, 2017] 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.

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 [Nazir et al., 2023] 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." [European Union, 2016].

B. Taxonomy of XAI

This sections 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

1) 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 [Molnar, 2022].

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.

2) 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.

3) Global versus local explanations: Global explanations provide general relationships between the input features and outputs learned by the model over the entire dataset [van der Velden et al., 2022]. 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.

C. Specific methods

The following section goes through several specific XAI methods, specifically ones related to images.

1) Class Activation Mapping (CAM): CAM [Zhou et al., 2015] 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 [Zhou et al., 2015]. As [Lecun et al., 1998] 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 1 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”.

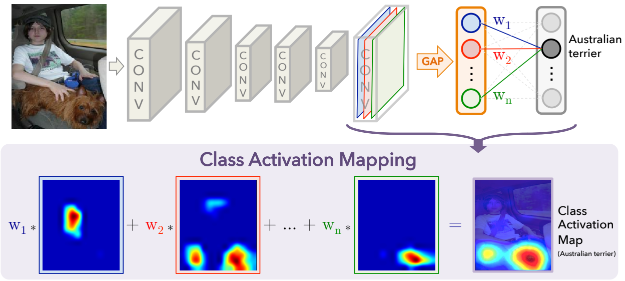

Fig. 1. Figure taken from [Zhou et al., 2015], showing the steps required to create a Class Activation Map

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 state of the art 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. [Zhou et al., 2015, 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.

2) Gradient Class Activation Mapping (Grad-CAM): Grad-CAM [Selvaraju et al., 2019] 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, Grad-CAM 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 [Selvaraju et al., 2019, 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 1.

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

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, Grad-CAM 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.

3) Layer-Wise Relevance Propagation (LRP): LRP is another XAI method which generates a visual explanation of the areas of an image which lead to a classification decision. Unlike CAM and Grad-CAM, LRP outputs a map which describes the relevance of every single pixel in the input image, and is thus produces a much more fine grained explanation than these other methods. LRP also differs in that [Bach et al., 2015] does not define it as a specific method, but rather as a concept defined by a certain set of constraints which can be satisfied by different implementations depending on the type of model.

LRP assumes that we can model the relevance R_i for any node i in a neural network, and aims to find the relevance for all the input nodes (the pixels in an image). Relevance is the contribution of any node to the final prediction $f(x)$ of a network, and the idea is to take the relevance of the output layer (simply defined as the output $f(x)$), and iteratively propagate this backwards through the network. Relevance scores are subject to a conservation property, which means that the sum of relevances must be equal for all layers (Equation 2). Furthermore, nodes must also conserve relevance, such that the sum of relevances a node receives from the previous layer is equal to the amount it distributes to the next layer. Once the relevance scores have been propagated from the output to the input layer in accordance with these constraints, we have a measure for how each input pixel contributed to the final output.

$$f(x) = ... = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l+1} R_d^{(l+1)} = ... = \sum_{d} R_d^{(1)} \tag{2}$$

To distribute relevance between the nodes in a way which obeys the constraints defined in Equation 2, a rule for propagation of relevances must be defined. As [Bach et al., 2015] shows, simply satisfying the constraints is not guaranteed to lead to meaningful explanations, nor is the decomposition of relevance unique. However, they show that by using a suitable propagation rule, we gain a visual explanation which shows which areas contribute to the final decision and which areas make the final decision less likely [Bach et al., 2015, 28].

4) 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 [Zeiler and Fergus, 2014], 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 [Thunold et al., 2023]. 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.

III. Out-of-Distribution (OOD) 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 [Goodfellow et al., 2015]. OOD detection is a developing field, and still in an initial stage [Cui and Wang, 2022]. In 2017, [Hendrycks and Gimpel, 2018] proposed a baseline OOD detection method. This section will discuss this method and the methods which follow it.

A. Motivation for OOD 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 [Yang et al., 2024]. 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 [Liu et al., 2021, 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. 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 [Yang et al., 2024, 15], thus requiring more sophisticated methods of OOD detection.

B. 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 cats and dogs, as a giraffe does not belong to either the ”dog” or ”cat” class. Covariate shift refers to samples which come from a different distribution while still belonging to one of the classes of the original data set. A picture of a chihuahua could represent a covariate shift for the same cat-versus-dog model if the training data contained only other races of dogs. The detection of semantic shift, as opposed to covariate shift, is the main focus of most OOD detection tasks [Yang et al., 2024]. 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 [Yang et al., 2024].

C. 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. Thus, we can calculate many different metrics and compare methods against each other. For OOD methods, the two most common metrics to report is the False Positive Rate at 95% recall (FPR95) and the Area Under Receiver Operating Curve (AUROC). It is common to use ImageNet as the ID dataset, and calculate FPR95 and AUROC on other datasets which contain no overlapping class labels. Common OOD datasets are iNaturalist, Texture, SUN or Places.

D. Methods

This section will follow the same outline as section II; 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 [Yang et al., 2024]:

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

Below follows a short explanation of each of the methods.

1) Classification-based methods: Classification-based methods usually use the softmax score or logits of a model to attempt to distinguish OOD and ID samples. [Hendrycks and Gimpel, 2018] 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, [Liang et al., 2020] further improved on this method, by amplifying the difference in softmax score of ID and OOD data. There are several other state of the art methods which utilize a classification-based approach, and these make up a large part of the representative methodologies for OOD detection today [Yang et al., 2024, 8]. As such, I shall devote the majority of section III-E to classification-based methods.

2) Density-based methods: Density-based methods explicitly try to model the in-distribution [Yang et al., 2024], 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 [Yang et al., 2024].

3) 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.

4) 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.

E. Specific methods

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

1) Baseline model: The baseline model created by [Hendrycks and Gimpel, 2018] 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. This works reasonably well, because the softmax scores for ID data generally is higher than for OOD data. However, such as simple method has its shortcomings, and there are many ways to improve the method, as will be shown in the following sections.

2) Out-of-Distribution Detector for Neural Networks (ODIN): [Liang et al., 2020] improves on the work of [Hendrycks and Gimpel, 2018] by introducing two simple modifications to the method which amplify the difference between the softmax score of ID and OOD samples. Firstly, they alter the input image slightly by adding small perturbations based on the gradients of the cross-entropy loss, as shown in equation 3

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \epsilon\,\mathrm{sign}(-\nabla_{\boldsymbol{x}}\mathrm{log}_{\hat{y}}(\boldsymbol{x};T)) \tag{3}$$

Secondly, they add temperature scaling to the softmax calculation, as shown in equation 4:

$$S_i(\tilde{\boldsymbol{x}};T) = \frac{\exp\left(f_i(\tilde{\boldsymbol{x}})/T\right)}{\sum_{j=1}^{N}\exp\left(f_j(\tilde{\boldsymbol{x}})/T\right)} \tag{4}$$

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

$$g(\boldsymbol{x};\delta,T,\varepsilon) = \begin{cases} 1 & \text{if}\max_i S(\tilde{\boldsymbol{x}};T) \le \delta, \\ 0 & \text{if}\max_i S(\tilde{\boldsymbol{x}};T) > \delta. \end{cases} \tag{5}$$

With these modifications, they report large improvements over the baseline [Liang et al., 2020, 4]. To explain this increase, we should look at the mathematical justification for these modifications.

The idea behind perturbing the input image based on the gradient of the cross entropy loss is that ID data points have a higher gradient than OOD data in general. By moving our data point slightly in the direction of the negative gradient, we should expect to see a higher softmax score than if we did not move, regardless of whether the data point is ID or OOD. However, because the gradients are larger for ID data, we expect that the difference between the new softmax scores will be larger than they were before the perturbations, because the ID data point has moved further towards higher softmax values, as shown in figure 2, taken from [Liang et al., 2020, 8]. Here we see two data points, one ID (red) and one ODD (blue), which are both perturbed. As we can see, the resulting softmax scores after the perturbations differ more than before the perturbation.

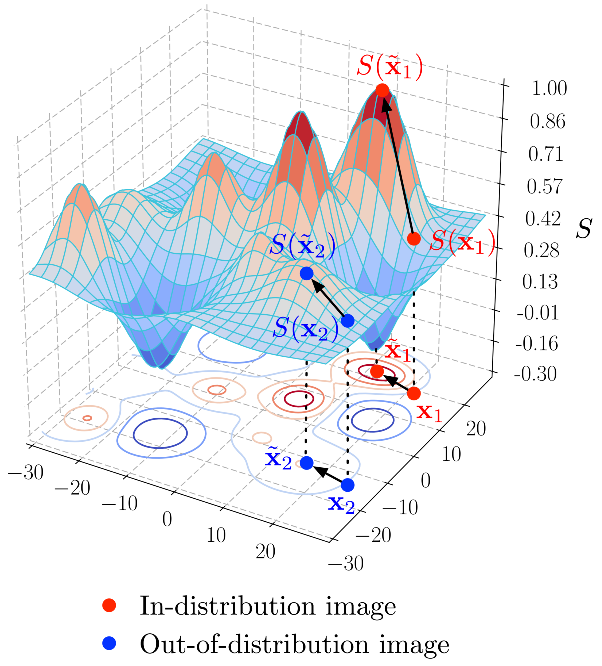

Fig. 2. Figure taken from [Liang et al., 2020], showing the difference in gradients between ID and OOD data points

The interpretation of the temperature scaling is slightly more complex. By performing a Taylor expansion and omitting third and higher orders, we can rewrite the softmax score (i.e the value of the predicted class, the highest value) as $S \propto (U_1 - U_2/2T)/T$, with U_1 and U_2 defined as follows [Liang et al., 2020, 4]:

$$U_1(\boldsymbol{x}) = \frac{1}{N-1}\sum_{i\neq\hat{y}}[f_{\hat{y}}(\boldsymbol{x}) - f_i(\boldsymbol{x})] \tag{6}$$

$$U_2(\boldsymbol{x}) = \frac{1}{N-1}\sum_{i\neq\hat{y}}[f_{\hat{y}}(\boldsymbol{x}) - f_i(\boldsymbol{x})]^2 \tag{7}$$

$\hat{y}$ is the predicted class, and is thus also the index for the highest value in $f(\boldsymbol{x})$. Thus, U_1 represents "the extent to which the largest unnormalized output deviates from the remaining outputs", while U_2 measures how the remaining outputs deviate from each other [Liang et al., 2020, 6].

[Liang et al., 2020] makes the two following observations with regards to these two values: Firstly, they find that the largest unnormalized output tends to deviate more for ID samples, making U_1 larger than for OOD samples, because the model is more confident in its prediction. Secondly, they find that $E[U_2|U_1]$ is larger for ID data samples than for OOD samples, which shows that ID samples have more separation in the remaining unnormalized inputs than OOD samples.

Returning to the Taylor approximated softmax score $S \propto (U_1 - U_2/2T)/T$, we see that U_1 contributes to making the softmax score higher, while U_2 reduces the softmax score. Given that both these values are higher for ID data, we will want to reduce the impact of U_2 and increase the impact of U_1. As U_1 is divided by T, while U_2 is divided by $2T^2$, increasing the temperature achieves this, as U_2 will decrease much faster than U_1. Thus, we can see how an increased temperature increases the softmax scores for ID data, and thus increases the gap between softmax scores for ID and OOD samples, making them easier to differentiate.

With these two modifications to the simple baseline proposed by [Hendrycks and Gimpel, 2018], [Liang et al., 2020] manages to increase the gap between the softmax scores of ID and OOD data and thus facilitates much more effective OOD detection.

3) Energy Based OOD Detection: [Liu et al., 2021] proposes using an *energy score* as opposed to the softmax score. They show mathematically that "the softmax confidence score is a biased scoring function that is not aligned with the density of the inputs" [Liu et al., 2021], and thus seek to use a different measurement which is better aligned with the probability density.

An energy function is a function $E(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ which maps any data point into a non-probabilistic scalar called energy. Energy values can be converted to probabilities using the Gibbs distribution defined below (equation 8):

$$p(y \mid x) = \frac{e^{-E(x,y)/T}}{\int_{y'} e^{-E(x,y')/T}} = \frac{e^{-E(x,y)/T}}{e^{-E(x)/T}}, \tag{8}$$

This equation is quite similar to the softmax function, and we can see that by defining $E(\boldsymbol{x}, y) = -f_y(x)$. We can write the Gibbs distribution as the normal softmax output of a neural network:

$$p(y \mid x) = \frac{e^{f_y(x)/T}}{\sum_i^K e^{f_i(x)/T}}, \tag{9}$$

By using the *Helmholtz free energy* measurement, we can get an energy score for each data point given to the model, which can be used to detect OOD data points. Given that we define $E(\boldsymbol{x}, y) = -f_y(x)$, we can write the Helmholtz free energy $E(\boldsymbol{x})$ as:

$$E(x; f) = -T \cdot \log \sum_i^K e^{f_i(x)/T}. \tag{10}$$

The authors show that when training with negative log likelihood loss, the optimization will reduce the free energy of ID data points, and that the difference between ID and OOD energy scores is higher than the difference in softmax scores [Liu et al., 2021]. Thus, thresholding the free energy function is an effective way to separate ID and OOD data points. Furthermore, they also present a method for fine tuning a pre trained model using a loss function that is based on the energy score. By doing this, the gap between ID and OOD energy scores can be increased even further.

4) ReAct: ReAct [Sun et al., 2021] is a very simple method, which also aims to increase the difference in confidence scores between ID and OOD data. It does this by rectifying high activations in the penultimate layer, which surprisingly achieves this very effectively. Figure 3 gives an intuition for why this is the case:

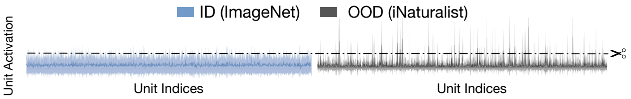

Fig. 3. Figure taken from [Sun et al., 2021], showing the activations for the nodes in the penultimate layer for ID and OOD data

From this, we see that OOD samples have much more irregular activations, with a higher variance and many high value outliers. This gives an explanation for why OOD samples produce highly confident softmax scores: sharp positive outliers manifest in the model output, producing high logits in the output layer [Sun et al., 2021]. By using a positive upper limit to rectify these outliers, we can remove their impact and reduce the confidence for OOD data.

This gives rise to a very simple OOD detector: Let us denote the feature vector of the penultimate layer as $h(\boldsymbol{x})$, where $\boldsymbol{x}$ is the input feature vector. The logits of the network would be calculated by the function

$$f(\boldsymbol{x}) = W\,h(\boldsymbol{x}) + \boldsymbol{b}, \tag{11}$$

where W is a matrix which projects $h(\boldsymbol{x})$ down to the output space. $h(\boldsymbol{x})$ is the vector which contains the high activations for OOD data, so by rectifying this vector with ReAct$(\boldsymbol{x}; c) = min(\boldsymbol{x}, c)$ for a $c > 0$, we can remove these outlier activations. We then get

$$\bar{h}(\boldsymbol{x}) = \text{ReAct}(h(\boldsymbol{x}; c), \tag{12}$$

which gives us the new output logits

$$f^{\text{ReAct}}(x; \theta) = W^{\top}\bar{h}(x) + \mathbf{b}. \tag{13}$$

These logits can be used by any other OOD method which uses the output values to separate ID and OOD samples [Sun et al., 2021]:

$$G_{\lambda}(x; f^{\text{ReAct}}) = \begin{cases} \text{in} & S(x; f^{\text{ReAct}}) \geq \lambda \\ \text{out} & S(x; f^{\text{ReAct}}) < \lambda \end{cases}, \tag{14}$$

This simple methods performs well on many benchmarks, with the added benefit that it can be combined with many other methods. For example, we can use ODIN or Energy with output scores calculated using ReAct instead of unrectified outputs, which leads to improvements over the methods used by themselves.

5) Virtual Outlier Synthesis (VOS): Generating outliers to expose to the model during training is another way to reduce the model's confidence on OOD data. However, creating realistic OOD data points can be difficult, especially if the input space is of a high dimension, such as in image classification. [Du et al., 2022] presents a more tractable method, which synthesizes outliers not in the input space, but in the feature space, which can be of a much lower dimensionality.

In this lower dimension space, previously intractable methods are now less computationally expensive. To synthesize outliers, [Du et al., 2022] simply estimate class conditional Gaussian distributions by computing empirical class means and covariances, and sample outliers from the class boundaries between these Gaussians.

Using these outliers, they present a ”unknown-aware training objective”, which can be used during training to maximize the separability between ID and OOD data during inference.

6) GradNorm: As opposed to using the feature or output space, GradNorm [Huang et al., 2021] 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 [Huang et al., 2021, 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 [Huang et al., 2021]:

$$S(x) = \| \frac{\partial D_{\text{KL}}(u \parallel \text{softmax}(f(x))}{\partial w} \|_p \tag{15}$$

As shown in figure 4, we see that the gradient norms are consistently lower for OOD data (gray) than ID data (blue).

[Huang et al., 2021] 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.

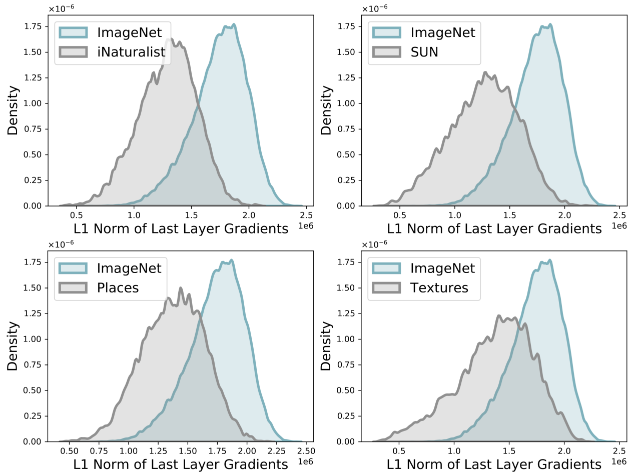

Fig. 4. Figure taken from [Huang et al., 2021], showing the difference in gradient norms between ID and OOD data

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:

$$S(\boldsymbol{x}) = \frac{1}{CT}\left(\sum_{i=1}^{m}|x_i|\right)\left(\sum_{j=1}^{C}\left|1 - C \cdot \frac{e^{f_j/T}}{\sum_{j=1}^{C} e^{f_j/T}}\right|\right) \tag{16}$$

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.

7) Virtual Logit Matching (ViM): [Wang et al., 2022] 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 [Wang et al., 2022]. Once the inputs have passed through the network and become 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 [Cook et al., 2020]:

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:

$$W(\boldsymbol{x} + \boldsymbol{a}) = W\boldsymbol{x} + W\boldsymbol{a} = W\boldsymbol{x} + \boldsymbol{0} = W\boldsymbol{x} \tag{17}$$

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 17 that when going from features to logits using the projection $W\boldsymbol{x}$, we lose all information contained in $\boldsymbol{x}^{\text{Null}(W)}$. [Cook et al., 2020] 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 5.

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 [Cook et al., 2020]. Another method which uses the features performs Principal Component Analysis (PCA) and looks at the residual information lost when using the first N

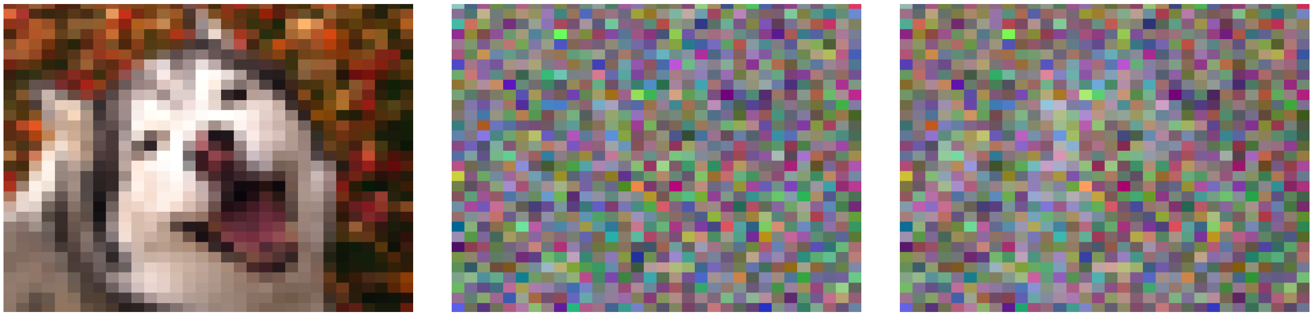

Fig. 5. Image taken from [Cook et al., 2020]. Left: Original image. Center: Additive null space noise. Right: Final image, indistinguishable from original image according to the network the noise in the center column is sampled from.

principal components [Ndiour et al., 2020]. However, the information in the features is still class agnostic, and [Wang et al., 2022] 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” [Wang et al., 2022]. They then perform PCA as in [Ndiour et al., 2020], 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.

$$\text{Residual}(x) = ||\boldsymbol{x}^{\text{Null}(P)}|| \tag{18}$$

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:

$$l_0 := \alpha||\boldsymbol{x}^{\text{Null}(P)}|| \tag{19}$$

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, [Wang et al., 2022] have combined information from the feature, the logit and the softmax probability level to perform OOD detection.

IV. Conclusion

In this essay I have given an introduction to the fields of Explainable Artificial Intelligence and Out-of-Distribution detection. I have looked at a selection of specific methods from each field, which gives a good overview of the current state of each of the fields.

References

[Bach et al., 2015] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

[Cook et al., 2020] Cook, M., Zare, A., and Gader, P. (2020). Outlier detection through null space analysis of neural networks.

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

[Dargan et al., 2020] Dargan, S., Kumar, M., Ayyagari, M. R., and Kumar, G. (2020). A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092.

[Doshi-Velez and Kim, 2017] Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning.

[Du et al., 2022] Du, X., Wang, Z., Cai, M., and Li, Y. (2022). Vos: Learning what you don’t know by virtual outlier synthesis.

[European Union, 2016] European Union (2016). Article 71: European data protection board. Accessed: February 13, 2024.

[Goodfellow et al., 2015] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples.

[Hendrycks and Gimpel, 2018] Hendrycks, D. and Gimpel, K. (2018). A baseline for detecting misclassified and out-of-distribution examples in neural networks.

[Huang et al., 2021] Huang, R., Geng, A., and Li, Y. (2021). On the importance of gradients for detecting distributional shifts in the wild.

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

[Liang et al., 2020] Liang, S., Li, Y., and Srikant, R. (2020). Enhancing the reliability of out-of-distribution image detection in neural networks.

[Liu et al., 2021] Liu, W., Wang, X., Owens, J. D., and Li, Y. (2021). Energy-based out-of-distribution detection.

[Molnar, 2022] Molnar, C. (2022). *Interpretable Machine Learning*. Independently published, 2 edition.

[Nazir et al., 2023] Nazir, S., Dickson, D. M., and Akram, M. U. (2023). Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks. *Computers in biology and medicine*, 156:106668.

[Ndiour et al., 2020] Ndiour, I., Ahuja, N., and Tickoo, O. (2020). Out-of-distribution detection with subspace techniques and probabilistic modeling of features.

[Selvaraju et al., 2019] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2019). Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359.

[Sun et al., 2021] Sun, Y., Guo, C., and Li, Y. (2021). React: Out-of-distribution detection with rectified activations.

[Szegedy et al., 2014] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks.

[Thunold et al., 2023] Thunold, H. H., Riegler, M. A., Yazidi, A., and Hammer, H. L. (2023). A deep diagnostic framework using explainable artificial intelligence and clustering. *Diagnostics*, 13(22).

[van der Velden et al., 2022] van der Velden, B. H., Kuijf, H. J., Gilhuijs, K. G., and Viergever, M. A. (2022). Explainable artificial intelligence (xai) in deep learning-based medical image analysis. *Medical Image Analysis*, 79:102470.

[Wang et al., 2022] Wang, H., Li, Z., Feng, L., and Zhang, W. (2022). Vim: Out-of-distribution with virtual-logit matching.

[Yang et al., 2024] Yang, J., Zhou, K., Li, Y., and Liu, Z. (2024). Generalized out-of-distribution detection: A survey.

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.

[Zhou et al., 2015] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2015). Learning deep features for discriminative localization.