Departmental Coversheet

MSc in Computer Science 2020-21

Project Dissertation

Project Dissertation title: On Representation Learning for Deterministic Uncertainty Estimation

Term and year of submission: Trinity Term 2021

Candidate Number: 1047481

On Representation Learning for Deterministic Uncertainty Estimation

Candidate Number: 1359080

University of Oxford

A thesis submitted for the degree of

Master of Science in Computer Science

Trinity Term 2021

Abstract

Deep neural networks (DNNs) have made tremendous progress and become the dominant approaches in many applications. However, their deployment in safety-critical areas remains limited since they can make over-confidently wrong predictions and cannot inform humans about their uncertainties. Therefore, uncertainty estimation is crucial for DNNs to be further deployed in real-world applications. Despite the differences among current uncertainty estimators, all these methods rely on the rich representations learned by DNNs. In this dissertation, we will investigate two problems in understanding the representations for uncertainty estimation in DNNs. In Chapter 3, we will investigate a line of models that aim to solve the so-called *feature collapse* problem. This problem describes that data far apart in the input space can be mapped to near-identical points by the feature extractor of DNNs, making it impossible to produce reasonable uncertainty estimates. We show that the considered models can preserve the distances in the low frequency domain under mild conditions, thus preventing feature collapse in this domain. In Chapter 4, we will investigate the interaction of representations and uncertainty estimation. Specifically, we will show that even state-of-the-art uncertainty estimation methods cannot consistently perform well on different out-of-distribution detection benchmarks. Using a decomposition of the representations, we will demonstrate that the desired representations for uncertainty estimation in different tasks are different. With our decomposition, we can achieve consistently high performance across different benchmarks. Finally, we conclude our dissertation with a discussion of our limitations and offer outlooks on extensions and promising future directions.

Contents

1 Introduction

Contents

1.1 The Importance of Uncertainty in Deep Neural Networks

In the last decade, deep neural networks (DNNs) have made tremendous progress and have become the dominant approach to a variety of applications such as computer vision [1, 2], natural language processing [3, 4], and healthcare [5, 6]. Despite the impressive scalability and supervised learning performance, the deployment of these models remains limited in real-world applications where intelligent systems are involved in making decisions that can affect humans. There are many reasons for this limitation. In particular, DNNs are very sensitive to domain shifts and frequently give overconfident predictions on out-of-distribution data [7, 8]. For example, Figure 1.1 shows a famous example of how easily DNNs are fooled to give

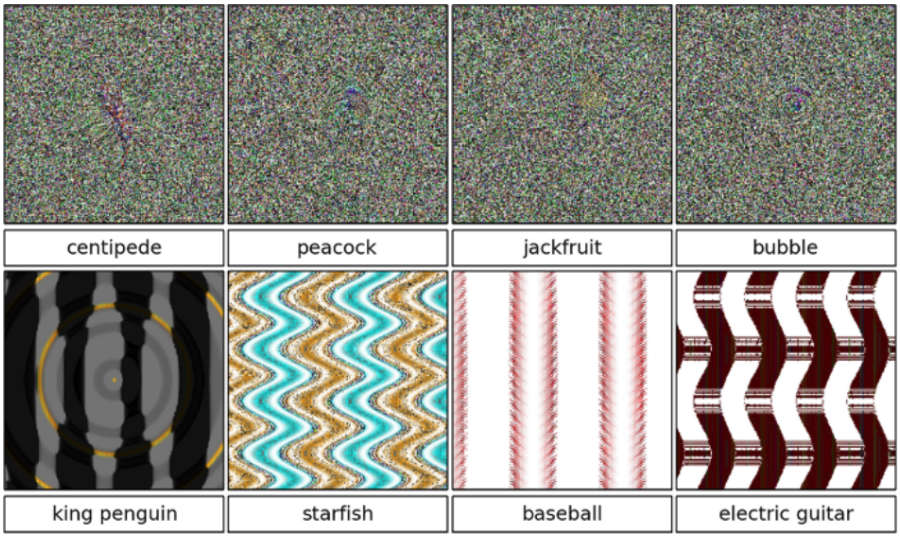


Figure 1.1: Example images that fool state-of-the-art DNNs trained on ImageNet[1] to classify with $\geq 99.6\%$ certainty to be a familiar object [7].

high confident predictions [7]. In real-world deployments, depending on applications, an intelligent system may frequently encounter data that are unlike the training data. The overconfident predictions on these data may then pose devastating results. To overcome these problems, it is essential to provide uncertainty estimates, so that the system would withhold uncertain predictions and defer to human experts.

Consider a concrete example: in many autonomous driving vehicles, DNNs are responsible for feature extraction tasks such as image segmentation to process raw sensory input [10]. The outputs of DNNs are then fed into a decision-making system which may rely on a fixed set of rules. In this case, failures in the feature extraction process can lead to decisions that may endanger human lives. In fact, the first fatality of autonomous driving was just caused by the failure of the DNNs to distinguish the tractor trailer and the background, leaving the Automatic Emergency Brake untriggered [11].

Therefore, uncertainty is crucial for DNNs to be further deployed in a broader range of real-world applications. However, as we will discuss in the next section, it is not straightforward to obtain uncertainty estimates from DNNs.

1.2 Challenges of Uncertainty Estimation in Deep Neural Networks

Deep learning

Before discussing the challenges of uncertainty estimation, we first formally define what a deep neural network is. A deep neural network (DNN) is a hierarchy of functions where each intermediate function is called a *layer* and the outputs of such functions are called intermediate *feature maps* or *representations*. In its simplest form, a DNN can be constructed by repeating matrix multiplications and element-wise non-linearities. For example, a 2-layer neural network can be implemented as $f(x) = W_2\sigma(W_1x)$, where W_1, W_2 are matrices and σ is an element-wise non-linearity function (e.g., tanh or ReLU [12]). There are also other function choices that suit specific data types. In this work, we will mainly be working with image datasets, so we will only introduce the convolutional neural networks (CNNs), specifically designed for data with some spatial topology (e.g., images, videos). The core component of a CNN is the convolutional layer, which accepts a filter w and a high-dimensional input a (i.e., a tensor) as inputs and convolves this filter by sliding it across all spatial positions of the input tensor and computing a dot product at each position. To be concrete, consider a convolutional layer between layer l and $l+1$, suppose the layer input a is of shape $m_l \times n_l \times F_l$, and we apply F_{l+1} filters each of shape $W_f \times H_f \times F_l$, the the output z of the convolutional layer can be expressed as:

$$z^{l+1}_{i',j',f'} = b^{l+1,f'} + \sum_{i=1}^{W_{f'}} \sum_{j=1}^{H_{f'}} \sum_{f=1}^{F_l} a^l_{i'+i-1,j'+j-1,f} w^{l+1,f'}_{i,j,f}.$$

The filters w and biases b are then the weights of convolutional layers. In classification tasks, the output of the penultimate convolutional layer is fed into a logistic regressor whose weights are trained together with the convolutional layers. Therefore, we also call the function from raw input to the output of penultimate convolutional layer the *feature extractor*.

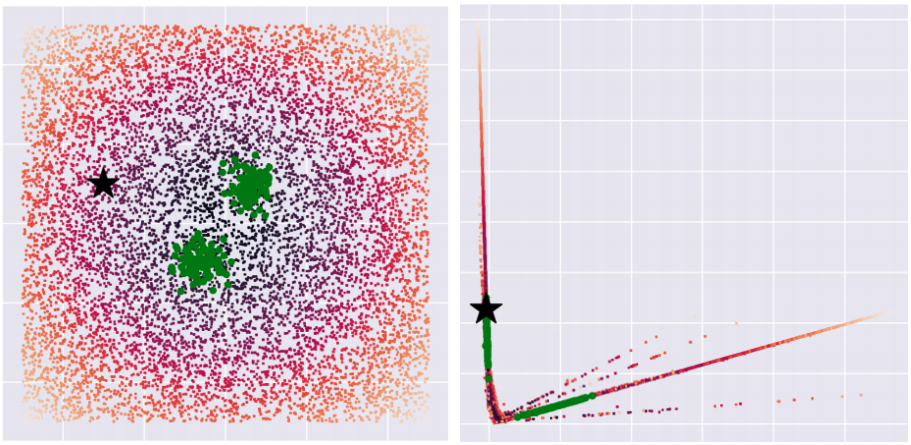
To train such models, the typical approach is to use (stochastic) gradient descent [13] to update the weights of DNNs, i.e., $W' = W - \alpha\nabla_W\mathcal{L}$, where W, W'

are the old and new weights, α is the step size, and $\nabla_W \mathcal{L}$ is the gradient of the objective function $\mathcal{L}$ on W. Specifically, given an objective function which computes the errors $\mathcal{L}(f(x), y)$ between predictions $f(x)$ and the targets y, the gradients of the objective function are calculated using the backpropagation algorithm [14] and summed over the entire dataset or a minibatch. The choice of the objective function is task-dependent. In classification, we usually use the cross-entropy function [13], which encourages the model to distinguish different classes. Thus DNNs trained using such methods are discriminative models.

One property that makes DNNs highly efficient is that it only requires a single forward pass to produce the predictions. This is because their weights are deterministic. That is, we only have point estimates of the weights instead of a distribution over all possible weights. Therefore, we cannot use Bayesian inference to compute a distribution over the outputs that describe how likely a specific value is. In classification tasks, the predictive probability of the final logistic regressor, i.e., the softmax output, is sometimes mistakenly treated as model confidence [7, 15]. However, as shown by [16, 17], data points that are far from the training distribution will always be assigned high softmax probability due to the extrapolation property of DNNs that use ReLU activations. Figure 1.1 just shows many examples of out-of-distribution data that are assigned ≥ 99.6 softmax probability.

The deterministic uncertainty estimation problem

Since uncertainty is essential yet not a built-in property of DNNs, we would like to equip a DNN with uncertainty estimation ability. This is the so-called *deterministic uncertainty estimation* problem, which is different from uncertainty estimation under Bayesian modeling where we have (approximate) probabilistic distributions over weights or posteriors [18]. In recent years, this field has attracted a lot of attention [19–26]. Approaches include combining a deterministic DNN feature extractor with a kernel-based last-layer and then approximating the variance over the posterior distribution $p(y|x)$ as the uncertainty estimate [22, 23]; or using a density estimate of the intermediate features $p(z|x)$ to capture the uncertainty [24,



(a) (b)

Figure 1.2: On the left, we visualize a 2D binary classification task where the training data are from two Gaussian distributions (green). We also colored other points according to their log probability under the distribution of the training data. Additionally, a specific point is marked with a star. On the right, we see the features computed by a standard DNN. As we can see, the inputs are collapsed into a single line, and the marked star moves from an unrelated area in input space to the class data in feature space, making it impossible to produce reasonable uncertainty estimates on top of these features. Figure is taken from [23].

26]. There are also other approaches that make use of the activity patterns of the intermediate features [25, 27]. We will review these methods in detail in Chapter 2.

Despite the differences in the uncertainty estimators, all these methods involve using a DNN-based feature extractor. The outputs of the feature extractor, also known as representations, will thus have a significant influence over the uncertainty estimates. For example, in Figure 1.2 we show a phenomenon called "feature collapse" which describes that the DNNs can map two points that are far apart in input space to a near identical point in the feature space, that is, given a feature extractor f, we can find points x_1, x_2 such that $0 \approx ||f(x_1) - f(x_2)|| \ll ||x_1 - x_2||$. This will make any uncertainty estimator on top of such representations unable to give a reasonable estimate. This problem motivates the introduction of bi-Lipschitz constraint in DNNs [21, 23], that is, given a feature extractor f,

$$\exists\, L_1, L_2, s.t.\ L_1||x - y|| \leq ||f(x) - f(y)|| \leq L_2||x - y||, \forall x, y.$$

The lower bound of the bi-Lipschitz constraint can thus effectively prevent the aforementioned feature collapse problem. Besides feature collapse, there may be

many other properties of representations from a discriminatively trained neural network that are undesirable for uncertainty estimation, e.g., consisting of many high-frequency signals of inputs that are spuriously correlated with the labels [8]. Also, we know very little about the interaction between representations and uncertainty estimators, which may explain the performance differences of uncertainty estimators in tasks like out-of-distribution detection (see Chapter 2). In short, our understanding of the representations for deterministic uncertainty estimation is far from complete. In this dissertation, we aim to fill this gap by targeting this problem:

> *What representations should we use for deterministic uncertainty estimation?*

In particular, we investigate two problems under this theme. In Chapter 3, we will investigate the feature collapse problem and the previously proposed bi-Lipschitz models that aimed to solve this problem [22, 23, 26]. We demonstrate that despite being well-motivated in theory, current implementations of bi-Lipschitz models are not bi-Lipschitz in practice. However, with a frequency analysis, we show that such models can still mitigate the feature collapse in the low-frequency domain. The conditions of our theorem provide us with new insights into the desirable properties of representations that can prevent feature collapse.

The second problem is how the representations interact with the uncertainty estimators. Specifically, we will show that even state-of-the-art uncertainty estimation methods cannot consistently perform well on different out-of-distribution (OoD) detection benchmarks (Table 4.1). Such performance instability is undesirable and casts doubts on how these methods would behave in various real-world applications. In Chapter 4, we show that this is because the desired representations for uncertainty estimation in different tasks are different. Moreover, even when the full representations contain the desired representations, the redundant parts will also influence the uncertainty estimates, resulting in degraded performance. Therefore, we propose to solve this problem by decomposing the representations and performing uncertainty estimation on them separately. We show that we can achieve consistently high performance across the different benchmarks with a proper

decomposition and a simple integration of the uncertainty estimates. Furthermore, our decomposition can also help us interpret the uncertainty estimates by comparing the contributions from different types of representations. This work provides us with insights on how to build uncertainty estimation methods that can adapt to different tasks by taking the types of representations into consideration.

1.3 Structure of the Dissertation

In Chapter 2, we will first review the necessary background to understand our work. Specifically, we will introduce uncertainty estimation and one of its applications – out-of-distribution detection. In Chapter 3 and 4, we will investigate two problems in representation learning for deterministic uncertainty estimation. These two chapters aim to understand how current implementations of bi-Lipschitz models influence the representations and how the interaction between representations and uncertainty estimation can be taken advantage of, respectively. We hope they can provide insights for future research in this field. Finally, in Chapter 5, we conclude our thesis by pointing out our limitations and offering outlook on interesting directions for future work.

2
Background

Contents

2.1 Uncertainty Estimation

As we discussed in Chapter 1, uncertainty is vital for the real-world deployment of DNNs. In this section, we will first introduce the sources and types of uncertainty and then review representative methods for uncertainty estimation. Finally, we will discuss the representation learning for uncertainty estimation.

2.1.1 Sources and Types of Uncertainty

From data acquisition to inference by our model, every step in a machine learning pipeline can incur uncertainty and errors, which influence the uncertainty of the

final prediction of a DNN. In [28], the authors summarized that five factors are most vital for the cause of uncertainty in the predictions:

1. **The variability in real-world situations.** The real-world situations are ever-changing, and during the test, many variables may have been different from the training data. For example, COVID data that were collected in the second wave may be different from those in the first wave because the genome type of the virus has changed [29]. This is also known as a distribution shift. Studies show that neural networks are sensitive to distribution shifts, and this can cause significant changes in the prediction [30, 31].

2. **The errors inherent to the measurement systems.** These include random noise in the measurement of data (x, y), e.g., measurement noise and labeling errors. Besides, limited information in the measurements can also cause such errors, e.g., images with low resolution. Such errors can only be reduced by improving the measurement systems.

3. **The errors in the architecture specification of the DNN.** The architecture of a neural network can directly influence the prediction and its uncertainty. For example, the number of layers a DNN can influence its number of learnable parameters and thus the model capacity. If the model has too many layers, it can over-fit the training data, leading to over-confident predictions [32].

4. **The errors in the training procedure of the DNN.** The training of a neural network is stochastic due to the randomness in decisions like the order of dataset and initialization of weights. Since the loss landscape of a DNN is shown to be highly non-linear [33, 34], such stochasticity will result in differently trained DNNs to have very different final weights, and thus lead to changes in their predictions.

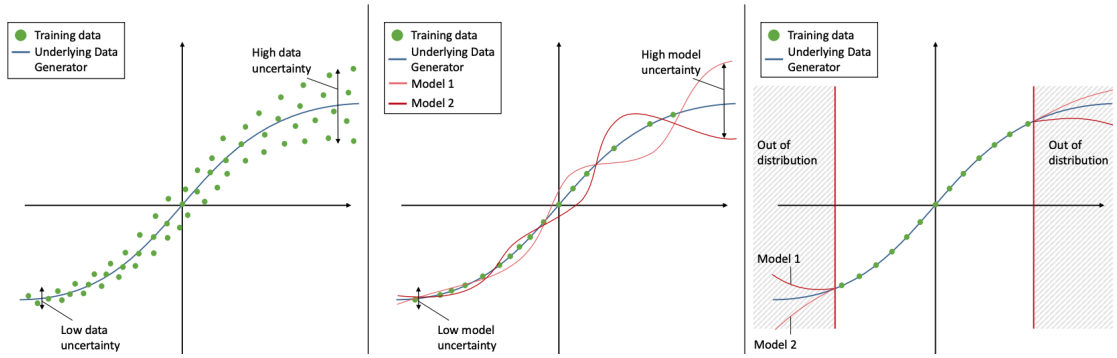


Figure 2.1: Types of uncertainty [28]

5. **The errors caused by unknown data.** Such errors occur during inference (test) time. At this stage, the neural network can encounter data that are entirely out of the domain of its training data, resulting in high uncertainty. This is similar but different from the first factor, where distribution shifts cause uncertainty. Unknown data can be described as data shifted so significantly that they should be considered inputs for a completely different task (while data under distribution shifts are still for the same task).

We can then categorize these sources of uncertainty into two categories (1) *data (aleatoric) uncertainty* (the second factor), which is caused by the inherent measurement noise or ambiguity in the data; and (2) *model (epistemic) uncertainty* (all the factors except the second factor), which is caused by shortcomings of our model and describes our ignorance about the models that are most suitable to explain the data. In Figure 2.1, we illustrate some examples of model and data uncertainty. Note that data uncertainty is not reducible by enlarging the dataset size and can only describe the uncertainty of the in-domain data, i.e., the data in the same domain as the training data. While model uncertainty can be reduced with more data or knowledge by, for example, collecting more training data to cover more variability in the environment.

2.1.2 Methods for Uncertainty Estimation

Uncertainty estimation is a challenging task since the different uncertainties as described above can, in general, not be modeled accurately. For the *data (aleatoric)*

uncertainty, since it is caused by the inherent noise in the data, we can often use the ambiguity of the prediction (the output) of the DNNs, e.g., the entropy of the softmax probability output of a DNN trained for classification can be used to capture the data uncertainty [26, 35]. Note that, as we mentioned, data uncertainty has to be estimated only on the in-domain data. Therefore, it is essential to make sure that out-of-distribution data are filtered before estimating data uncertainty. For the *model (epistemic) uncertainty*, there are many different approaches. Roughly, they can be categorized as probabilistic and deterministic methods.

Probabilistic methods use Bayesian modeling, and estimates the model (epistemic) uncertainty as the variance of a posterior prediction by marginalizing over the distributions of network weights [36, 37]. To be specific, the posterior distribution over the prediction is

$$p(y \mid x, D) = \int p(y \mid x, \theta) p(\theta \mid D) d\theta, \tag{2.1}$$

where D is the training dataset, $p(\theta \mid D)$ is the posterior distribution over the weights of the DNN. Apparently, these methods also require modeling distributions over the weights, significantly increasing the computation burden during training, limiting their scalability. Besides modeling distribution over the weights, recent works have also investigated structure uncertainty such as uncertainty over depths and activation functions [38, 39]. Alternatively, when an explicit posterior is not available, we can still sample from an approximate posterior by using dropout samples [19], i.e., performing multiple forward passes with the dropout [40] to get different outputs, or using several individual models as in [20].

Deterministic methods for uncertainty estimation try to use representations of the models directly. Therefore, they can keep the scalability and performance of DNNs. For example, we can combine a deterministic neural network with a kernel-based final layer(s), such as Gaussian Processes [41]. To estimate the uncertainty, we can use an approximate variance over the posterior predictive distribution $p(y|x)$ of the kernel-based layers [21, 22, 42]. Methods of this type allow us to combine the scalability and representation learning power of neural networks and the uncertainty

estimation of Bayesian methods. However, they need to modify the architecture and train the new models, which are not ideal for many real-world applications. Alternatively, some works use a density estimation of the intermediate features of a DNN [21–24, 26, 43]. Besides density estimation, we can also utilize the activity patterns of intermediate layers [25] and the correlations of these patterns [27] to capture epistemic uncertainty. These methods can perform on a pre-trained model with a single forward pass and are thus more cost-effective and scalable than the other two approaches. When done carefully, these models can achieve competitive performance without high computational costs.

Representation learning for deterministic uncertainty estimation

In this dissertation, we will study the deterministic methods for uncertainty estimation due to their scalability. Despite the differences in the deterministic uncertainty estimation methods, they all use the representations learned by a DNN. However, as we mentioned in Chapter 1, understanding the desirable properties of representations for deterministic uncertainty estimation is still an open research question. Note that the desirable properties of representations can be task-dependent. For example, in an object classification task, we may want our representations to focus more on high-level semantic features such as shapes of objects. In this case, the uncertainty estimates will give low uncertainty to data that have shapes of a particular object class but different low-level features such as textures and high uncertainty to data of new shapes. This task dependence is sometimes overlooked in many previous works.

Current works in representations for uncertainty estimation can thus be categorized based on their aimed properties of representations. The most commonly desired property is to include more diverse representations, specifically, both classification-relevant features encouraged by the cross-entropy loss and features that can describe the in-distribution data but may not be essential for classification (e.g., textures). For example, to make the representations more aware of semantically meaningful features, [44] trains DNNs using rotation prediction (i.e., training DNNs

to recognize the 2D rotation applied to the image that it gets as input), [45] uses contrastive learning (i.e., encourages the features of transformed versions of the same image to cluster together and to pull all other images away). However, although these methods aim to learn representations as inclusive as possible, in practice, they are focusing more on high-level features suitable for specific tasks. This may limit their scope of applications. Another property of representations is to contain features that can distinguish between the in distribution data and (general) out-of-distribution data. This can be done by explicitly using an auxiliary OoD dataset and a modified objective function [17, 46, 47], or by adversarial training, which may help representations aware of the "boundaries" of the in-distribution representations [48]. The problem with such methods is two-fold: first, they need extra training time and memory; second, it is hard to show that their representations can learn representations that are aware of *all* the OoD data – since just as representations, OoD is also a task-dependent concept, as we will introduce in the next section. Finally, motivated by the feature collapse problem introduced in Chapter 1 (Figure 1.2), some works propose to enforce bi-Lipschitzness on the feature extractor, i.e., given a feature extractor f, $\exists L_1, L_2$, s.t. $L_1||x-y|| \leq ||f(x)-f(y)|| \leq L_2||x-y||$. The feature collapse can then be prevented since the distance between features are lower bounded by the distance between inputs. To implement bi-Lipschitz models, current implementations [22, 23, 26] used convolutional residual networks [2] with spectral normalization [49]. These models have been empirically successful. However, as we will show in Chapter 3, these models are not bi-Lipschitz. Using a frequency analysis, we will explain how they may prevent the feature collapse under the low-frequency domain.

In summary, we still have limited knowledge of the desirable properties of representations for uncertainty estimation and how to implement them. In this dissertation, we will also look into this problem and hope to provide new insights.

2.2 Out-of-distribution Detection

2.2.1 What is Out-of-distribution Detection?

One important and challenging application of uncertainty estimation is out-of-distribution (OoD) detection. OoD detection has been attracted much attention in recent years [15, 24, 50, 51] since it is closely related to the safety of machine learning models. Without effectively detecting OoD data, the predictions of DNNs on them can often be over-confidently wrong. In this dissertation, we will use OoD detection as the main application to evaluate the effectiveness of uncertainty estimation.

As an informal definition, OoD detection requires an OoD detector to assign a scalar score $s(x)$ to the test-time input x and determine whether it is OoD or in-distribution based on a certain threshold γ. Possible scalar scoring functions include uncertainty estimate [19, 20], density estimate [52, 53], and some distance metric [25]. We note here that the definition of OoD is also task-dependent. For example, given a face recognition dataset, depending on the task, the OoD data may be faces captured using new types of cameras (in which case we need to rely on low-level statistics to detect such OoD), or can be non-human faces like dog faces (in which case we need to rely on high-level semantic features). In other words, when we decide the OoD detection method (and the representations), we also determine the OoD data we can detect.

2.2.2 Methods for Out-of-distribution Detection

As mentioned, there are two common choices of scoring functions for OoD detection: uncertainty or likelihood. When we use (epistemic) uncertainty to detect OoD data, we are essentially defining OoD as data on which our model is unable to make a trustworthy *prediction* $p(y|x)$. For example, this could be data very unlike the training data (an image of a bicycle, while our training data are human faces), but also data that could have been part of the training data but with too few examples for our model to learn to generalize. When we use likelihood to detect OoD data, we define OoD as data with low density $p(x;\theta)$ under an estimated

distribution of the training data. These data may have some overlap with data that have high uncertainty. However, in terms of representations, uncertainty relies on features from a discriminatively trained model since it is directly related to prediction, while likelihood prefers features that are most prevalent in the training data and does not limit to any specific types of features.

The problem with likelihood-based methods is that the representations extracted by common deep generative models consist of a high proportion of low-level features, which OoD features may also share and thus not ideal for OoD detection [52, 54]. Many works aim to understand and solve this problem [54–56]. For example, [54] proposes to use likelihood ratio to mitigate the influence of background information, [56] uses a multi-scale invertible model to extracts high-level features for likelihood calculation. In general, the empirical performances of these likelihood-based methods are still inferior to state-of-the-art uncertainty estimation methods. We believe it would be an interesting future direction to investigate how to improve these methods further. Nevertheless, in this dissertation, we will focus on uncertainty-based methods.

2.2.3 Evaluating Out-of-distribution Detection Methods

When evaluating OoD detection performances, since the output of the OoD detector is a scalar that shows how likely/unlikely a data point is OoD, it is generally challenging to determine a threshold because of the trade-off between false negatives (FN) and false positives (FP). Faced with this issue, previous works typically chose to use threshold-independent metrics such as the Area Under the Receiver Operating Characteristic curve (AUROC) [57] and the Area Under the Precision-Recall (AUPR) [58]. The ROC curve plots true positive rate (TPR=TP/(TP+FN)) against false positive rate (FPR=FP/(FP+TN)), and the PR curve plots the precision (TP/(TP+FP)) and the recall (TP/(TP+FN)) against each other. The AUPR is sometimes considered more informative when there is a high class (positive and negative) imbalance [57].

One challenge of evaluating OoD detection is to choose the proper benchmarks. Although most OoD methods were proposed to detect "general OoD data", i.e., any data that are unlike the training data, in practice, we can hardly evaluate the methods in such a way since it is impractical to exhaust all the OoD data. Instead, almost all the OoD detection methods were evaluated using several OoD datasets, hoping that different OoD datasets reflect different types of OoD data. However, there are relatively few works that investigate whether these benchmarks are good enough for evaluation. We have so little knowledge of these benchmarks that usually, we can only show empirically that some OoD benchmarks are more "difficult" than others.

Recently, [45] evaluated their OoD detection method using *a spectrum of OoD datasets* with different similarities to the training data. The similarity of datasets were calculated by first training a classifier that simultaneously classify the inliers and outliers and then averaging the softmax probability in the in-distribution classes of OoD data:

$$D_{\mathcal{C}_{\text{in}}}\left(\mathcal{D}_{\text{test}}\right) = \log\left(\frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{\mathbf{x}\in\mathcal{D}_{\text{test}}}\sum_{k\in\mathcal{C}_{\text{in}}}\hat{p}(\hat{y}=k\mid\mathbf{x})\right).$$

Evaluating a spectrum of OoD datasets is conceptually more meaningful and exhaustive than using only a few OoD datasets. However, we point out that the dataset distance should not only be limited to the classification ambiguity. Depending on the application, we may care about other types of distances. For example, the background information may not be helpful for classification. However, we may want to detect OoD data with new backgrounds during test time (since these data may result in over-confidently wrong predictions due to spurious correlation [8]). Therefore, besides the spectrum of OoD datasets with different classification ambiguity (or similarity in discriminative features), we should also evaluate OoD detection methods along other spectrums, e.g., datasets with different similarities in non-discriminative features, to have a full understanding of our OoD detection methods and the representations we use.

3

A Frequency Analysis of the Bi-Lipschitz Regularization for Convolutional ResNets

Contents

3.1 Introduction

Modern out-of-distribution detection methods use the representations extracted by deep neural networks to perform uncertainty or density estimation. However, it is shown that standard neural networks can learn a function that maps the out-of-distribution data close to the training data in the feature space since the learned function is unconstrained [21, 26]. See Figure 1.2 for an example. This phenomenon is called “feature collapse”.

A recent line of work proposed to resolve this problem by introducing distance-awareness into residual neural networks by enforcing the learned function to be bi-Lipschitz [22, 26]. The idea of the bi-Lipschitz constraint is to bound the distortion of the learned representations from the input data and thus encouraging the learned function to map the OoD data away from the training data. Specifically, if our function mapping has the form $f(x) = x + g(x)$ and spectral normalization [49] is applied to g such that its bi-Lipschitz constant is smaller than 1, then f is bi-Lipschitz [59], that is,

$$\exists\ L_1, L_2, s.t.\ L_1||x-y|| \leq ||f(x)-f(y)|| \leq L_2||x-y||, \forall x, y. \tag{3.1}$$

By definition, a bi-Lipschitz function has a lower bound on the feature distance between two distinct data points. Therefore, points sufficiently far apart in input space are guaranteed to be mapped far from each other, preventing feature collapse entirely.

However, two departures from the theories are made in current implementations of the residual networks with spectral normalization. First, despite having a bounded Lipschitz constant on some of the operations in the residual skipping, the overall learned function is still not bi-Lipschitz due to the use of dimension-reduction (downsampling) operations. Specifically, the input space changes that lie in the null space of the dimension-reduction operations will result in no changes in the features[1], violating the lower bound of bi-Lipschitz constraint. Moreover, in the current implementations, the spectral coefficient is always set to be larger than 1 for training stability [22, 26]. However, to prove the lower bound of the bi-Lipschitz of the whole residual block, it is required that the bi-Lipschitz constant of the skip connection, i.e., the spectral coefficient, is smaller than 1. These violations leave the effectiveness of current models unexplained: if the learned function is not bi-Lipschitz, what are alternative reasons for residual networks' empirical success with spectral normalization?

[1]In fact, as we will discuss in Section 3.3.1, the dimension-reduction operations can be thought of as a dimension-preserving operation followed by an image decimation step, where we keep every n^{th} sample from the input. So the input changes not in every n^{th} sample will not influence the outputs.

In Section 3.3, we answer the above question using a frequency analysis of the model. The main theorem states that residual blocks must preserve the distance between low pass filtered versions of their inputs under some relatively mild conditions. Hence, even though the learned function is not bi-Lipschitz for the full image, it can still be distance-aware, thus preventing feature collapse on the low-frequency domain of the image.

Finally, we verify these theoretical claims empirically (Section 3.4). We find that even if the mathematical conditions for our main theorems are violated, the conclusion of the theorem still holds in most cases. This indicates that our mathematical conditions can be further relaxed. We then discuss the limitations of our results and implications for future work.

Individual Contribution: This work is part of a paper submitted to NeurIPS 2021. My contribution includes a lemma in the proof of the main theorem and all the experiments presented in this chapter.

3.2 Problems of Current Implementation for Bi-Lipschitz Regularization

Current implementations of bi-Lipschitz constraints mainly use the spectral normalization scheme with convolutional residual network architecture. Spectral normalization enforces the weight matrices to have a spectral norm, i.e., the largest singular value, less than 1, so that for the corresponding nonlinear residual block $g(x) = \sigma(Wx + b)$, we have $Lip(g) < 1$. We follow the method in [59] to perform spectral normalization: at every training step, we estimate the spectral norm $\sigma \approx ||W||_2$ using the power iteration method [60], and then normalize the weights as:

$$\mathbf{W}_l = \begin{cases} c * \mathbf{W}_l / \hat{\sigma} & \text{if } c < \hat{\sigma} \\ \mathbf{W}_l & \text{otherwise} \end{cases},$$

where $c > 0$ is the spectral coefficient used to adjust the spectral norm upper bound on $||W||_2$ so that $||W||_2 \leq c$ (note if $||W||_2 < 1$, $Lip(g) < 1$).

As we point out in the introduction, there are two problems with current implementations that make the learned function not bi-Lipschitz anymore. The

first is the use of dimension-reduction operations, which theoretically cannot be bi-Lipschitz. As a simple example, when f is a linear function, the inputs in the null space of f will be mapped to the same output, which breaks the lower bound bi-Lipschitzness. In fact, bi-Lipschitz mappings have to be dimension-preserving:

Theorem 1. *Let f be a function from $\mathbb{R}^n$ to $\mathbb{R}^m$, with $m < n$. Then f is not bi-Lipschitz.*

In addition, current models also set the spectral coefficient to be larger than 1 for training speed and stability. This also breaks the requirement for proving the lower bound of the bi-Lipschitzness. This can be seen directly from the following inequality:

$$\begin{aligned} \|\mathbf{x}-\mathbf{x}'\| &\leq \|\mathbf{x}-\mathbf{x}'-(f(\mathbf{x})-f(\mathbf{x}'))+(f(\mathbf{x})-f(\mathbf{x}'))\| \\ &\leq \|(f(\mathbf{x}')-\mathbf{x}')-(f(\mathbf{x})-\mathbf{x})\|+\|f(\mathbf{x})-f(\mathbf{x}')\| \\ &\leq \|g(\mathbf{x}')-g(\mathbf{x})\|+\|f(\mathbf{x})-f(\mathbf{x}')\| \\ &\leq \alpha\|\mathbf{x}'-\mathbf{x}\|+\|f(\mathbf{x})-f(\mathbf{x}')\|, \end{aligned} \tag{3.2}$$

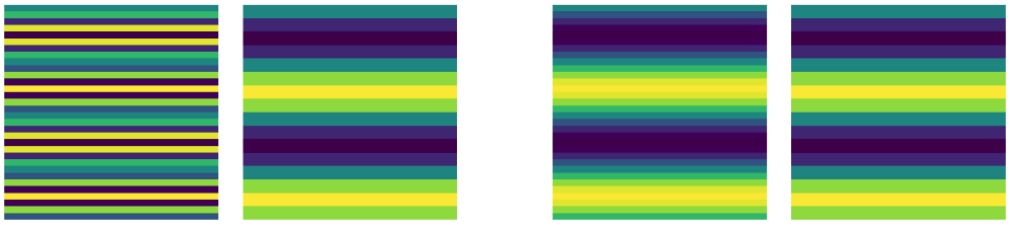
where $f_l(\mathbf{x})) = \mathbf{x} + g_l(\mathbf{x}))$ and $\mathrm{Lip}(g_l) \leq \alpha$. The lower bound of bi-Lipschitz constraint of f_l, $(1-\alpha)\|\mathbf{x}-\mathbf{x}'\| \leq \|f(\mathbf{x})-f(\mathbf{x}')\|$, only has effect when $\alpha < 1$. However, we note here that this requirement is only a sufficient condition for the lower bound of bi-Lipschitz to hold. In practice, it is still possible that without this constraint, the lower bound still holds.

We will first investigate the downsampling operations from a frequency perspective in Section 3.3 and empirically investigate the effect of relaxing the spectral coefficient condition in Section 3.4.

3.3 A Frequency Analysis of Residual Networks with Bi-Lipschitz Regularization

3.3.1 Background: Downsampling in Convolutional Networks from a Frequency Perspective

In this section, we introduce the background for understanding the downsampling operations in convolutional networks. In signal processing, we have two well-known



(a) (b)

Figure 3.1: An example of aliasing between two Fourier modes. The left images in (a) and (b) are two different Fourier basis functions: (a) is a Fourier basis function above the Nyquist rate, (b) is a basis function below the Nyquist rate. The right images in (a) and (b) are the decimation (by a factor 2) results of the corresponding left images, and we find them to be indistinguishable. Figure is taken from [61].

facts about the sampling frequency. The first is the Nyquist-Shannon sampling theorem which states that if a signal sampled at frequency u_s contains no frequency contents above the Nyquist frequency $u_s/2$, then it can be reconstructed exactly [62]. On the other hand, if a signal sampled at frequency u_s contains frequency contents above the Nyquist frequency $u_s/2$, these contents will be indistinguishable from a lower-frequency component. This is called "aliasing". For example, the Stroboscopic effect, that is, wheels in videos sometimes appear to be rotating differently from its true rotation, is due to the frame rate not meeting the Nyquist frequency [63]. As we will see, aliasing closely connects with the feature collapse: it can help us describe feature collapse in the high-frequency contents. In Figure 3.1, we provide another example.

Aliasing is undesirable because we do not want low-frequency contents to be contaminated by aliased signals. The classical way to resolve this is to anti-alias by low-pass filtering the signal [64], i.e., remove the high-frequency contents before downsampling. In [65], the authors extended this idea to convolutional networks by combining an anti-aliasing filter with the downsampling operation. This can make convolutional networks achieve shift-invariance and much better performance. We will keep the same practice to perform downsampling with anti-aliasing filtering in this chapter.

As noted by [65], the downsampling operations in convolutional networks - including average pooling, strided convolutions, and max pooling - can be thought

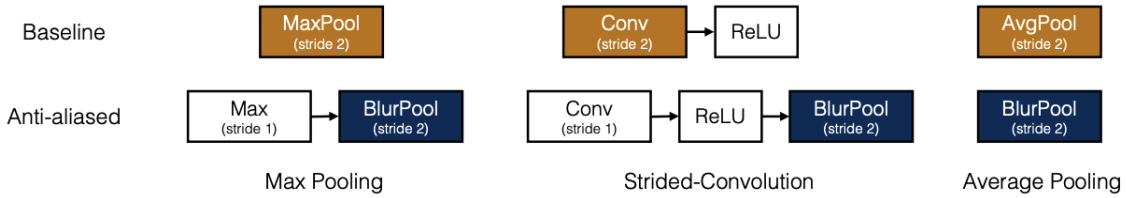


Figure 3.2: Common downsampling operations can be better anti-aliased using a combination of a dimension-preserving operation and a image decimation operation with anti-aliasing filter (i.e., BlurPool). See [65] for more details.

of as a dimension preserving operation followed by an image decimation step, where we keep every n^{th} sample from the input. See Figure 3.2. For example, strided convolution is equivalent to a standard convolution followed by an image decimation step. Using the previous theories, we know that the sufficient condition for a downsampling operation to produce no feature collapse (aliasing). If we have two signals that differ in their contents below the Nyquist frequency, they will still be distinguishable after the operation since their low-frequency contents will be preserved. In other words, the feature collapse induced by the downsampling operations can only exist in high-frequency contents since they are removed before decimation by the anti-aliasing filtering.

3.3.2 Residual Networks and the Frequency Content of Images

As discussed above, all downsampling operations in ResNets can be understood as dimension-preserving operations followed by decimation. When combined with anti-aliasing, the frequencies above the Nyquist frequency will be removed after downsampling, and those below will be unaffected. Therefore, downsampling operations will preserve the distances in the low-frequency domain. In this section, we extend this distance preserving property to the convolutional ResNets. To this end, we need to understand how other components of a convolutional ResNet influence the frequency contents of an image.

To start with, a convolutional ResNet consists of identity mappings, additions, convolutions, nonlinearities (ReLUs), and batch normalizations. The identity

mapping does not affect the frequency contents of an image. The pointwise addition adds the frequency content of two images. The batch normalization does not change the relative scale of the frequency since it only multiplies each channel by a scalar and adds a bias. From the convolution theorem [64], we know that convolutions act in the frequency domain as pointwise multiplication of the frequency components of the input signal with the corresponding frequency components of the convolution kernel. Hence, it will only change the scale of each frequency component, and when we apply spectral normalization, we will see a contraction on each frequency band separately. Finally, for the ReLU nonlinearity, we can show that it also acts as a convolution in the frequency domain. To see this, let $m = I[x > 0]$ be the binary mask on the locations in the image x when ReLU is activated. Then $\text{ReLU}(x) = x \cdot m(x)$. By the convolution theorem, the action of ReLU on X (the Fourier transform of x) will be a convolution of X with the Fourier transform $M(x)$ of the binary mask $m(x)$, i.e., $\text{ReLU}(X) = M(X) * X$ ($*$ denotes convolution). Using this understanding, we can show that ReLU acts as a contraction on the low-frequency component X_u of its input (see proof in [61]).

Lemma 1. *Let H_u be a low-pass filter that removes all frequencies in a signal above the cutoff frequency u, let x, y be images, and let $v = x - y$. Assume that the difference image, v, is dominated by frequencies below a cutoff frequency u[2]. Then* $||H_u(\text{ReLU}(x) - \text{ReLU}(y))|| < ||H_u(x) - H_u(y)||$.

Now that we have understood all the components of a convolutional ResNet, we can reach our main theorem, which states that a residual block will preserve the low-frequency distances:

Theorem 2. *Let $f = x + g(x)$ be a convolutional residual block (i.e g is a series of convolutions, batch normalisation and ReLUs), and assume that g is regularised to be contraction (*$\text{Lip}(g) < 1$*), and that the conditions of Lemma 1 hold on the input to the ReLU (i.e the difference image $x - y$ is low-frequency dominant). Then,*

[2]The domination is defined as the following: Let f and g be (normalized) measures. An interval of length L starting at x is a 'dominant' interval in f when if $\int_x^{x+L} f(t)dt = C$, then $\int_y^{y+L} f(t)dt \leq C \; \forall y$.

the low-passed distances between the output are lower-bounded by the low-passed distances on the input, that is $L||H_u(x) - H_u(y)|| \leq ||H_u(f(x)) - H_u(f(y))||$ *for some constant* $L > 0$.

We can easily extend this result to the whole network – if $\text{Lip}(f_1) = L_1$ and $\text{Lip}(f_2) = L_2$, then $\text{Lip}(f_1 \circ f_2) = L_1 L_2$. Since the theorem provides a lower bound for the low-frequency distances, it guarantees that we will not see feature collapse in the low-frequency contents when the conditions of the theorem are satisfied [3]. This provides a theoretical explanation of the strong empirical OoD detection performance of the convolutional ResNets with downsampling operations.

3.4 Experiments

In this section, we conduct empirical experiments to investigate further the reasons for the success of current spectral normalized ResNets. There are three key questions we want to answer: 1) Does the condition of Lemma 1 hold? That is, are the images and intermediate feature maps low-frequency dominant? 2) Does the conclusion of Theorem 2 hold? From the proof in [61], we know that theoretically the conclusion holds because the residual block is a contraction on the low-frequency contents, i.e., $||f(H_u(x_i)) - f(H_u(x_j))|| < ||H_u(x) - H_u(y)||$. So we can check the contraction as a verification of the lower bound conclusion in our experiments. 3) Whether we can find violations of the theorem when the Lipschitz condition is relaxed? That is, when the spectral coefficient is larger than 1, will we still see a contraction in the low-frequency domain? This is motivated by how previous works performed the spectral normalization [22, 26]. While in the proof, we assume the convolutions are strict contractions ($Lip(g) < 1$) since it is only a sufficient condition, it is interesting to find out whether breaking such condition would result in violations of the theorem empirically.

In our experiments, we train Wide ResNets [66] on MNIST [67], FashionMNIST [68] and CIFAR10 [69]. We will answer the previous three questions by checking

[3] Note the low-frequency dominance condition is not unreasonable for image inputs. As we can see from Table 3.1 and Table 3.2, the natural images are indeed low-frequency dominant.

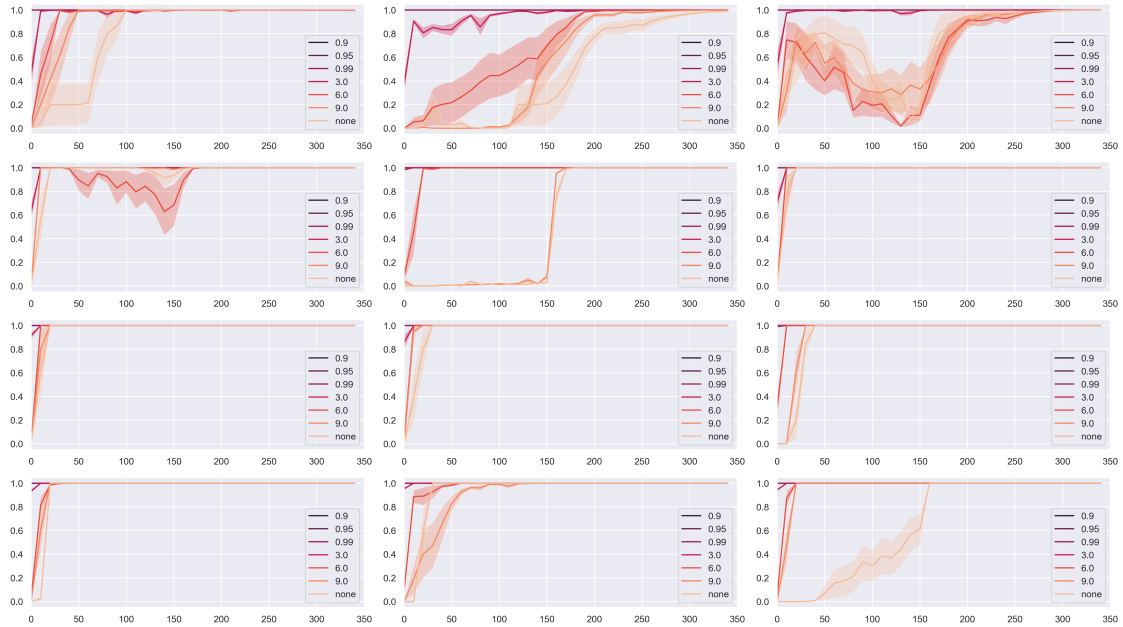


Figure 3.3: Evolution of the value of the theorem check (i.e., proportion of the data that satisfy the theorem) during training for all the blocks of a Wide ResNet trained on CIFAR10. Lines are colored according to the values of the spectral normalization coefficient (Lip(g)). As can be seen, while there are violations of the theorem during training, all blocks satisfy the conclusion of Theorem 2 at convergence. This indicates that while a larger spectral coefficient may influence the training trajectory, it may not degrade a spectral normalized ResNet's low-frequency contraction property. The blocks are shown in order from left to right and top to bottom. We plot the mean and standard error across five seeds.

on data from these datasets and the intermediate feature maps of the residual network. For the domination condition of an image, we can numerically check on its Fourier transform. For the contraction on the low-frequency contents, we consider a mini-batch of data $\{x_i \mid i \in 1, ..., m\}$, compare the pair-wise distances before and after the residual block, and calculate the proportion of the batch for which $||g(H_u(x_i)) - g(H_u(x_j))|| < ||H_u(x_i) - H_u(x_j)||$. Note that since this check is only performed on a particular dataset, it does not necessarily prove that the theorem holds for *all* possible inputs. However, if we cannot find violations on natural image datasets, this does suggest that low-frequency distance is preserved *between natural images*. The results are shown in Table 3.1 and Table 3.2.

As we can see, the condition of Lemma 1 (low-frequency domination) does not hold in quite a few intermediate feature maps. Despite this violation of condition,

Table 3.1: Results of empirically checking the conditions of Lemma 1 (low-frequency domination) and the conclusion of Theorem 2 (low-frequency contraction) on MNIST and FashionMNIST. Models with different spectral coefficients are compared. Numbers are the proportion of the dataset that the condition or the theorem holds true (1 means no violations were found, 0.5 means only half of the inputs meeted the condition/theorem, etc.) We report means and standard error over 25 seeds for these datasets. We use a WideResNet with depth 10 and widen factor of 1 for these datasets.

	Dataset	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
Lemma 1	MNIST	0.9	0.80 ± 0.00	0.18 ± 0.04	0.31 ± 0.03	0.63 ± 0.02	0.44 ± 0.02
		0.95	0.80 ± 0.00	0.18 ± 0.04	0.30 ± 0.05	0.62 ± 0.04	0.45 ± 0.02
		0.99	0.80 ± 0.00	0.20 ± 0.03	0.31 ± 0.03	0.63 ± 0.02	0.45 ± 0.01
		3.0	0.80 ± 0.00	0.23 ± 0.02	0.43 ± 0.04	0.71 ± 0.01	0.47 ± 0.02
		6.0	0.80 ± 0.00	0.22 ± 0.03	0.42 ± 0.03	0.71 ± 0.01	0.48 ± 0.02
		9.0	0.80 ± 0.00	0.22 ± 0.04	0.41 ± 0.03	0.71 ± 0.01	0.48 ± 0.01
		no	0.80 ± 0.00	0.24 ± 0.02	0.39 ± 0.05	0.70 ± 0.01	0.46 ± 0.02
	FashionMNIST	0.9	0.86 ± 0.00	0.18 ± 0.04	0.24 ± 0.04	0.48 ± 0.04	0.47 ± 0.03
		0.95	0.86 ± 0.00	0.18 ± 0.04	0.25 ± 0.02	0.48 ± 0.03	0.46 ± 0.01
		0.99	0.86 ± 0.00	0.18 ± 0.05	0.25 ± 0.04	0.51 ± 0.02	0.48 ± 0.01
		3.0	0.86 ± 0.00	0.24 ± 0.03	0.28 ± 0.04	0.53 ± 0.04	0.47 ± 0.02
		6.0	0.86 ± 0.00	0.24 ± 0.03	0.28 ± 0.02	0.53 ± 0.02	0.47 ± 0.02
		9.0	0.86 ± 0.00	0.25 ± 0.01	0.28 ± 0.04	0.54 ± 0.04	0.47 ± 0.01
		no	0.86 ± 0.00	0.24 ± 0.03	0.25 ± 0.03	0.53 ± 0.03	0.47 ± 0.01
Theorem 2	MNIST	0.9	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.95	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.99	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		3.0	n/a	0.69 ± 0.31	0.96 ± 0.11	1.00 ± 0.00	1.00 ± 0.00
		6.0	n/a	0.55 ± 0.26	0.98 ± 0.05	1.00 ± 0.00	1.00 ± 0.00
		9.0	n/a	0.61 ± 0.28	0.99 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
		no	n/a	0.60 ± 0.30	0.99 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
	FashionMNIST	0.9	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.95	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		0.99	n/a	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		3.0	n/a	0.98 ± 0.04	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		6.0	n/a	0.95 ± 0.11	0.92 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
		9.0	n/a	1.00 ± 0.01	0.85 ± 0.12	1.00 ± 0.00	1.00 ± 0.00
		no	n/a	1.00 ± 0.00	0.86 ± 0.07	1.00 ± 0.00	1.00 ± 0.00

Theorem 2 (low-frequency contraction) holds for all the models whose Lipschitz constant of the residual skipping is constrained to be smaller than 1. This suggests that the conditions may be relaxed, and the theorem may hold in more cases.

As for the influence of the spectral coefficient, we note that on MNIST and FashionMNIST, we can observe some violations of the theorem. This is expected since the Lipschitz condition plays a crucial role in proving the low-frequency contraction theorem. However, on CIFAR10 (and deeper layers on MNIST and FashionMNIST), we also observe that the low-frequency contraction holds even when the spectral coefficient exceeds one by a large margin. In Figure 3.3, we

Table 3.2: Results of empirically checking the conditions of Lemma 1 (low-frequency domination) and the conclusion of Theorem 2 (low-frequency contraction) on CIFAR10 - see the caption of Table 3.1 for a more detailed description. We report means and standard errors over 5 seeds for this dataset. We use a WideResNet with a depth of 28 and widen factor of 10.

	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
Lemma 1	0.9	0.66 ± 0.00	0.53 ± 0.11	0.56 ± 0.20	0.41 ± 0.13	0.41 ± 0.13	0.41 ± 0.12	0.67 ± 0.01
	0.95	0.66 ± 0.00	0.53 ± 0.03	0.41 ± 0.19	0.35 ± 0.07	0.37 ± 0.06	0.39 ± 0.04	0.67 ± 0.06
	0.99	0.66 ± 0.00	0.52 ± 0.03	0.35 ± 0.17	0.30 ± 0.05	0.30 ± 0.04	0.33 ± 0.04	0.65 ± 0.01
	3.0	0.66 ± 0.00	0.54 ± 0.03	0.26 ± 0.02	0.19 ± 0.03	0.20 ± 0.02	0.26 ± 0.02	0.64 ± 0.01
	6.0	0.66 ± 0.00	0.94 ± 0.00	0.36 ± 0.03	0.28 ± 0.09	0.21 ± 0.06	0.25 ± 0.03	0.62 ± 0.02
	9.0	0.66 ± 0.00	0.88 ± 0.13	0.43 ± 0.04	0.27 ± 0.05	0.20 ± 0.04	0.18 ± 0.02	0.60 ± 0.01
	no	0.66 ± 0.00	0.92 ± 0.03	0.41 ± 0.04	0.24 ± 0.07	0.17 ± 0.02	0.18 ± 0.02	0.60 ± 0.01
	Lip(g)	$f_7(x)$	$f_8(x)$	$f_9(x)$	$f_{10}(x)$	$f_{11}(x)$	$f_{12}(x)$	$f_{13}(x)$
	0.9	0.69 ± 0.03	0.71 ± 0.03	0.78 ± 0.04	0.85 ± 0.08	0.77 ± 0.01	0.78 ± 0.03	0.94 ± 0.01
	0.95	0.67 ± 0.03	0.69 ± 0.02	0.77 ± 0.05	0.81 ± 0.09	0.76 ± 0.02	0.76 ± 0.03	0.95 ± 0.02
	0.99	0.67 ± 0.01	0.69 ± 0.01	0.75 ± 0.04	0.81 ± 0.08	0.77 ± 0.01	0.78 ± 0.02	0.93 ± 0.02
	3.0	0.63 ± 0.02	0.66 ± 0.02	0.72 ± 0.01	0.76 ± 0.02	0.84 ± 0.02	0.81 ± 0.04	0.83 ± 0.03
	6.0	0.63 ± 0.01	0.65 ± 0.01	0.73 ± 0.02	0.85 ± 0.07	0.77 ± 0.01	0.76 ± 0.05	0.82 ± 0.04
	9.0	0.63 ± 0.01	0.66 ± 0.02	0.69 ± 0.01	0.69 ± 0.06	0.76 ± 0.01	0.71 ± 0.05	0.85 ± 0.02
	no	0.63 ± 0.01	0.68 ± 0.02	0.70 ± 0.01	0.75 ± 0.10	0.74 ± 0.03	0.81 ± 0.01	0.74 ± 0.02
	Lip(g)	x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
Theorem 1	0.9	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.95	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.99	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	3.0	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	6.0	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	9.0	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	no	\	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	Lip(g)	$f_7(x)$	$f_8(x)$	$f_9(x)$	$f_{10}(x)$	$f_{11}(x)$	$f_{12}(x)$	$f_{13}(x)$
	0.9	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.95	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	0.99	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	3.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	6.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	9.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	no	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00

plot the evolution of the theorem check quantities during training. It is intriguing to see that although the theorem can be heavily violated during training, after convergence, it holds for all choices of spectral coefficients across all the blocks. We do not have a theoretical explanation for why this should happen. However, this result is in agreement with the previous empirical studies, which showed that distance-aware learning with spectral normalization coefficients above one could perform well [21–23, 26].

3.5 Discussion

In this work, we investigate an intriguing phenomenon in current literature – despite an apparent violation of the theoretical motivation to use ResNets with spectral normalization, these models perform exceptionally well for uncertainty estimation and out-of-distribution detection. We have demonstrated that, under mild assumptions, residual networks will be approximately distance preserving on the low-frequency contents of their input. Therefore, feature collapse on the low-frequency domain can still be prevented. In our experiments, we show that this theorem holds on natural image datasets even when the assumptions of our mathematical proof are relaxed, for instance, by training with a spectral normalization coefficient above 1. Our work provides a new perspective to understand the effect of existing residual architecture and normalization schemes in the context of representation learning for deterministic uncertainty estimation. We also hope that our work could provide insights on developing new models that can prevent feature collapse, e.g., by enforcing the low-frequency dominance more explicitly.

All models are wrong, but some models are useful.

— George Box [70]

4

Decomposing Representations for Out-of-distribution Detection

Contents

4.1 Introduction

Deep neural networks (DNNs) have achieved remarkable progress during the last decade. However, they assume that test data come from the same distribution as the training data. However, in real applications, it is inevitable for a model to make predictions on *Out-of-Distribution (OoD)* data instead of in-distribution data

on which the model is trained, leading to failures of DNNs [47, 71]. When DNNs are part of a decision-making system in safety-critical tasks, the failures of DNNs can propagate through the system and result in fatal errors [11]. Therefore, it is crucial to detect such OoD data and defer them to human experts instead of making predictions. In recent years, OoD detection, or anomaly detection, has become a vital component of the machine learning system in many high-risk applications like autonomous driving and medical diagnosis [72, 73].

As introduced in Chapter 2, OoD detection requires an OoD detector to assign a scalar score $s(x)$ to the test-time input x and determine whether it is OoD or in-distribution based on a certain threshold γ. To improve the quality of the scoring function s, modern OoD detection methods rely on the rich representations of deep neural networks. However, even if the representations are rich enough, we can still obtain degraded performance because the redundant representations may influence or dominate the scoring function. For example, deep generative models are known to be able to learn a rich representation of the inputs [59, 74]. However, the work by [52] shows that these models give higher likelihoods to OoD data than in-distribution data. In many following works [54, 56], it is shown that this is because the representations of deep generative models are dominated by low-level features (background features) that are also shared by many images in the chosen OoD datasets. In this case, high-level features are more desirable for OoD detection (among images). In practice, depending on the application, the ideal OoD detection may mainly depend on another type of feature, so we need more understanding of the representations and the target OoD detection tasks. We include more concrete examples of this issue in Section 4.3. In this work, we propose decomposing the representations and using the decomposed features to explain and solve the problem mentioned above.

While there are many choices of decomposition, in this work, we specifically choose to investigate the *discriminative* and *non-discriminative features*, that is, features that are related and unrelated to classification (prediction). For example, in an object classification task, the shapes of objects may be discriminative features,

while the background of the images may be non-discriminative features[1]. Based on the decomposition, we propose a simple algorithm to integrate the OoD detection scores calculated on each kind of feature and a dataset distance metric to measure the difference of in-distribution and OoD data. We show that our decomposition can improve the OoD detection performances and the interpretability of the uncertainty estimates through experiments.

In summary, in this chapter, we make the following contributions:

- Make the observation that state-of-the-art density estimation methods which are successful on some OoD detection benchmarks can fail (or be suboptimal) on others (Section 4.3);
- Propose the decomposition of features to solve the problem mentioned above, and design an algorithm and a dataset distance metric based on the decomposition (Section 4.4);
- Improve both OoD detection performances and interpretability of uncertainty estimates using the decomposition (Section 4.5).

4.2 Background

4.2.1 Representation-based OoD Detection

As introduced above, the task of OoD detection depends on a scoring function which can be either uncertainty or density estimation. In recent years, most modern approaches make use of the rich representations learned by deep neural networks (DNN) and perform the uncertainty or density estimation on the activations in an intermediate feature space of a DNN classifier [21, 22, 24, 25]. There are mainly two ways to improve the OoD detection performances: improve the density (or uncertainty) estimator, or adapt the training strategy to learn better representations. To improve representation learning for OoD detection, the key is to include more

[1]In practice, we acknowledge that the learned discriminative features may not be consistent with human perception. For example, in [75], it is shown that high-frequency features that are imperceptible to human can be highly predictive, and thus discriminative.

features that can distinguish in-distribution and OoD datasets. Recent efforts include using extra OoD training datasets [46] and using self-supervised learning or contrastive learning to learn richer representations [44, 45]. For the approximated density estimator, since high-dimensional density estimation is notoriously difficult, most approaches propose a distance metric as an alternative. One of the most successful methods is the Mahalanobis distance [24] which models the in-distribution as a class-conditional multivariate Gaussian distribution using training samples and for each test sample, calculate the maximum of log likelihoods across classes as the final score for OoD detection. Other approaches include Feature Space Singularity Distance (FSSD) which calculates a Euclidean distance from the test sample to an approximate OoD center [25, 76] and using Gram matrices from multiple feature maps to capture abnormal activity patterns [27].

The evaluation of OoD detection methods requires the use of OoD datasets, which is a task-dependent design choice. To explore the difference of OoD datasets, recent works propose the concept of "near OoD" and "far OoD" [43, 45]. "Near OoD"refers to data that are semantically similar to the in-distribution (e.g., shape, pose), while "far OoD" are more unrelated (e.g., brightness). Our work is an addition to this line of work. We show that the differences in OoD datasets can greatly influence the performance of OoD detection methods (see Section 4.3), and we can achieve optimal performance by integrating the discriminative and non-discriminative features which are responsible for detecting "near" and "far" OoD data respectively.

4.2.2 Mahalanobis Distance for OoD Detection

Mahalanobis distance [24] has been very successful and a common component in many representation-based OoD detection methods. Since our work is based on the Mahalanobis distance method and its variants to explain the ideas, it is beneficial for us to have a more detailed introduction to the method.

We first distinguish the general Mahalanobis distance and the Mahalanobis distance method for OoD detection. In its general form, Mahalanobis distance

calculates the distance between a point x and a distribution D. It is a multi-dimensional generalization of the idea of measuring how many standard deviations away x is from the mean of D. Given the mean μ and the covariance matrix Σ of D, the Mahalanobis distance of x is:

$$M(x) = \sqrt{(x-\mu)\Sigma^{-1}(x-\mu)}. \tag{4.1}$$

The Mahalanobis distance method for OoD detection first models the distribution of the intermediate features as a class-conditional multivariate Gaussian distribution with a tied covariance matrix. Then it calculates class-wise Mahalanobis distance of a new feature $z = f(x)$ using this distribution and uses the maximum of the negative squared Mahalanobis distance as its score:

$$M(x) = \max_c -(f(x) - \hat{\boldsymbol{\mu}}_c)^T \hat{\Sigma}^{-1} (f(x) - \hat{\boldsymbol{\mu}}_c), \tag{4.2}$$

where

$$\hat{\Sigma} = \frac{1}{N} \sum_{c=1}^{C} \sum_{i:y_i=c} \left(f(\boldsymbol{x}_i) - \hat{\boldsymbol{\mu}}_c\right) \left(f(\boldsymbol{x}_i) - \hat{\boldsymbol{\mu}}_c\right)^T, \ \hat{\boldsymbol{\mu}}_c = \sum_{i:y_i=c} f(\boldsymbol{x}_i).$$

Equivalently, we can also calculate the maximum of the log likelihood[2] of the class-conditional Gaussian distribution $p(z|y=c) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_c, \Sigma)$. As a natural extension, we can also model the distribution using different covariance matrices for each class. The maximum of the log likelihood across classes[3] then has the form:

$$M(\mathbf{z}) = \max_c \left[-\left(\mathbf{z} - \hat{\boldsymbol{\mu}}_c\right)^T \hat{\boldsymbol{\Sigma}}_c^{-1} \left(\mathbf{z} - \hat{\boldsymbol{\mu}}_c\right) - \log\left((2\pi)^d \det \hat{\boldsymbol{\Sigma}}_c\right) \right], \tag{4.3}$$

where

$$\hat{\Sigma}_c = \frac{1}{|\{i : y_i = c\}|} \sum_{i:y_i=c} \left(f(\boldsymbol{x}_i) - \hat{\boldsymbol{\mu}}_c\right) \left(f(\boldsymbol{x}_i) - \hat{\boldsymbol{\mu}}_c\right)^T, \ \hat{\boldsymbol{\mu}}_c = \sum_{i:y_i=c} f(\boldsymbol{x}_i).$$

Since this new modeling has higher flexibility than the original model and may have better uncertainty estimation when there are enough data, it is also widely used in recent works [26, 45]. These two modeling approaches correspond to the

[2]Alternatively, we can also calculate the sum of the log likelihoods across classes, which is the likelihood under the modeled distribution $p(z) = \sum_c -(f(x) - \hat{\boldsymbol{\mu}}_c)^T \hat{\Sigma}^{-1} (f(x) - \hat{\boldsymbol{\mu}}_c)$. Empirically, there are no big difference between these two approaches.

[3]Similarly, we can also calculate the sum of the log likelihoods as in the previous case.

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) of Gaussian discriminant analysis (GDA), where we fit a generative model for classification using the class-conditional Gaussian distribution on the data [77]. Therefore, we will call the original Mahalanobis distance method as *Maha* and the above extension as *QDA-Maha*.

Other variants of Mahalanobis distance were proposed to improve on *some specific tasks*. Notably, marginal Mahalanobis distance (*Marginal Maha*) [78] was proposed to show that principal components with small explained variance contribute a lot to the performance of *Maha*. These principal components were shown to be weakly related to classification. Their method thus calculates the mean and covariance without using class-relevant information. Specifically, in Equation (4.2),

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i)-\boldsymbol{\mu}\right)\left(f(\boldsymbol{x}_i)-\boldsymbol{\mu}\right)^T,\ \boldsymbol{\mu}_c = \boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i).$$

Empirically, it was reported that the *Marginal Maha* could already achieve comparable perfomrance to the original *Maha* method on many "far OOD" datasets. Another variant is called Relative Mahalanobis distance (*Relative Maha*) [43]. Just the opposite of the *Marginal Maha* method, *Relative Maha* aims to mitigate the influence of the class-irrelevant information by calculating the distance as $M_{rel}(x) = M(x) - M_{marg}(x)$ where M_{rel} and M_{marg} refers to *Relative Maha* and *Marginal Maha*, respectively. Empirically, the authors showed the effectiveness of *Relative Maha* on many *near OOD* benchmarks like CIFAR10 vs. CIFAR100.

4.3 Motivating Observations

In this section, we compare the variants of Mahalanobis distances on two OoD benchmarks: CIFAR10 vs. SVHN and CIFAR10 vs. CIFAR100. These are two widely used benchmarks in OoD detection literature [15, 24, 50] and can reflect two types of different OoD: SVHN consist of street numbers and are highly different from the CIFAR10, which contains objects from 10 classes; while CIFAR100 are collected using the same pipeline as CIFAR10 but contains objects from 100 other classes [69]. We train a Wide ResNet 28-10 [66] as [43] and calculate distances on

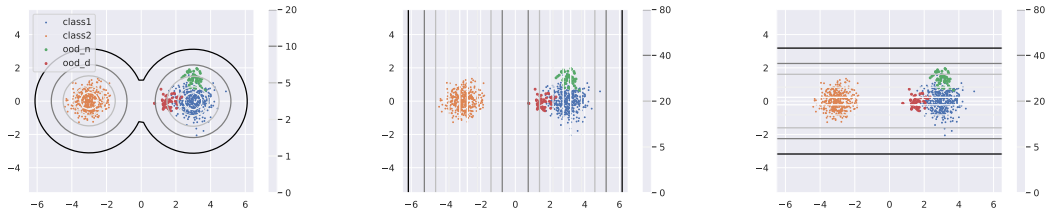
the penultimate layer. The methods to be compared are: Mahalanobis distance (*Maha*), Marginal Mahalanobis distance (*Marginal Maha*), and Relative Mahalanobis distance (*Relative Maha*). As we can see, the key difference between the three variants is how the class-relevant information is (not) used. *Maha* is a density estimation that uses class-relevant information, *Marginal Maha* is class-independent, and *Relative Maha* is equivalent to computing a likelihood ratio where the an approximate background likelihood $\log p_{marg}$ is subtracted from the class-dependent likelihood $\log p_{maha}$. So in terms of the degree of class-relevance, we have: *Relative Maha* $>$ *Maha* $>$ *Marginal Maha*.

The OoD detection results can be found in Table 4.1. We can see that on CIFAR10 vs. SVHN, all the methods can perform relatively well with *Maha* being the top. While on CIFAR10 vs. CIFAR100, there is a sharp difference among the methods: *Relative Maha* shows significantly better performance than others while *Marginal Maha* almost completely mixes the two datasets (AUROC=50% means random guess). Note that although the compared methods are well-known, none of them has been tested simultaneously on both benchmarks. Our experiments demonstrate that none of these methods can beat every other method on both benchmarks.

Considering the difference in the degree of class relevance among the three methods, we hypothesize that the differences in OoD detection performances can be attributed to the different desired degrees of class relevance on different OoD detection benchmarks. To further test this hypothesis, we decompose the features into discriminative (class-relevant) ones and non-discriminative (class-irrelevant) ones. In Figure 4.1, we show an analytic 2D example where the discriminative direction is along the x-axis, and the non-discriminative direction is along the y-axis. We then consider OoD data along both directions. Since the covariance matrix on the full feature treats variations in both directions equally, it will have a circular decision boundary, which is not perfect for detecting either OoD type. Alternatively, if we use discriminative or non-discriminative features, we can get the best performance in detecting the corresponding type of OoD data.

Table 4.1: AUROC (%) comparison of variants of Mahalanobis distance calculated on features at penultimate layer (pre-logit).

Method	CIFAR10 vs SVHN	CIFAR10 vs CIFAR100
Maha	97.95	75.75
Marginal Maha	96.06	59.90
Relative Maha	96.28	91.28



(a) Mahalanobis distance on full features.
AUROC$_{red}$ = 0.94,
AUROC$_{green}$=0.94.

(b) Mahalanodis distance on discriminative features.
AUROC$_{red}$= 0.98,
AUROC$_{green}$ = 0.36.

(c) Mahalanodis distance on non-discriminative features.
AUROC$_{green}$ = 0.98,
AUROC$_{red}$ = 0.34.

Figure 4.1: A 2D example showing the effect of decomposition. The *orange* and *blue* points are two in-distribution classes, *green* points are OoD from non-discriminative directions and *red* points are OoD from discriminative directions. Contour lines are colored according to the Mahalanobis distance to the in-distribution points (darker means higher). We can see that *Maha* on the full feature weighs on both discriminative and non-discriminative directions simultaneously (and in our case, equally). This makes *Maha* on full features (**(a)**) suboptimal in detecting OoD data compared to *Maha* calculated on either direction only (**(b), (c)**).

We also conduct a high-dimensional test where features are 128D but only discriminative in the first ten dimensions. This can be seen as a simulation of the CIFAR10 representations. Again, we compare the Mahalanobis distance calculated on the full, discriminative, and non-discriminative features. Since the covariance matrix of Mahalanobis distance treats each dimension equally, the Mahalanobis distance calculated on the full feature would be heavily influenced by the variations of the 118D non-discriminative features. We find *Maha* achieves 83.95% AUROC in detecting OoD data along the discriminative directions. However, when using discriminative features alone, we could detect discriminative features almost perfectly (near 100% AUROC). See Section 4.5 for details of this experiment.

4.4 Decomposing Representations

Inspired by our previous observations, we propose decomposing the discriminative and non-discriminative information from the learned representations. This allows us to combine and choose "optimal" features based on our understanding of the OoD detection benchmark and a new perspective to interpret the uncertainty estimation.

Given a learned feature extractor, a straightforward way to perform the decomposition of representations is to perform Principal Component Analysis (PCA) on the features. Specifically, given the feature matrix of a dataset[4] $X \in \mathbb{R}^{n \times p}$ where n is the number of inputs, p is the length of a feature (representation), we first compute its SVD decomposition [79] $X = U\Sigma V^T$, where $U \in \mathbb{R}^{n \times n}, \Sigma \in \mathbb{R}^{n \times p}, V \in \mathbb{R}^{p \times p}$. We can then perform an orthogonal linear transformation on the feature matrix: $T = XV$. The columns (coordinates) of T are ordered such that they are uncorrelated, and the variance over the dataset is decreasing. We call first d columns of T *top principal components (PCs)* and the other columns *bottom principal components (PCs)* and we can then use them separately as representations for uncertainty estimation. In this work, d is chosen to be the number of classes in the dataset. Despite being linear and not using the label information, PCA can usually do a good job in decomposition since the representations of a discriminatively trained neural network are usually clustered by classes. Specifically, using the features from a Wide ResNet 28-10 trained on CIFAR10, when we train logistic regressors on the top and bottom PCs separately for classification, we can get 95.7% and 24.6% accuracy on the test set, respectively, suggesting that the top PCs have captured most of the discriminative features.

To fully utilize the label information, we propose a more advanced method to perform the decomposition. Given a feature extractor $f : \mathbb{R}^M \to \mathbb{R}^D$ which maps from inputs to representations, we first perform a transformation of the features $z = f(x)$ using an invertible function $F : \mathbb{R}^D \to \mathbb{R}^D$. Our aim is then to extract discriminative features z_s into the first d dimensions of $F(z)$ and non-discriminative

[4]Assume we have already subtract the mean of each column of the feature matrix before we perform the PCA.

features z_n into the other dimensions. As a convenient choice, we set $d = C$, i.e., the number of classes of the training dataset. z_s then serves as logits for the classification. We then uses another linear map $D : \mathbb{R}^D \to \mathbb{R}^d$ to map z_n to a $d-$dimensional logits. In this way, we can now use the so-called independence cross-entropy (iCE) loss [80] to encourage the decomposition of discriminative and non-discriminative features. The iCE loss is defined as:

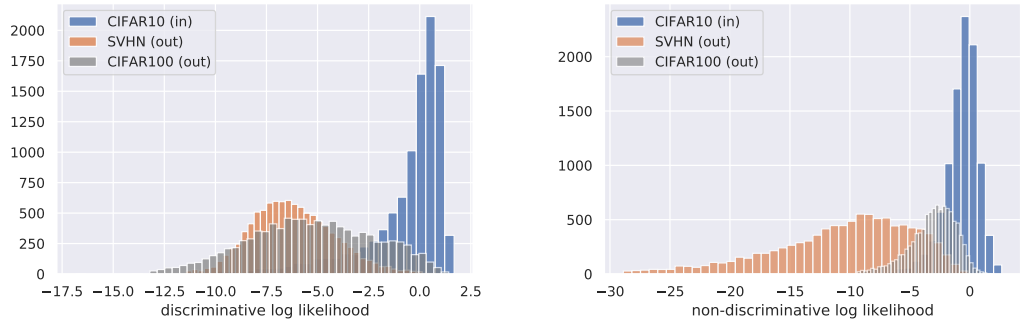
$$\min_{\theta} \max_{\phi} \mathcal{L}_{iCE}(\theta, \phi) = \underbrace{\sum_{i=1}^{C} -y_i \log \operatorname{softmax} F_\theta(z_s)_i}_{=:\mathcal{L}_s CE(\theta)} + \underbrace{\sum_{i=1}^{C} y_i \log \operatorname{softmax} D_\phi\left(F_\theta(z_n)\right)_i}_{=:\mathcal{L}_{nCE}(\theta,\phi)}. \tag{4.4}$$

To further understand this loss function, it is worth pointing out that the loss $\mathcal{L}_{nCE}$ can be understood using a variational lower bound on mutual information. Moreover, it can be proved that the minimization is concerning a lower bound on $I(y; z_n)$, while the maximization aims to tighten the bound [80]. Therefore, the iCE loss aims to maximize $I(y; z_s)$ (first term) and minimize $I(y; z_n)$ (second term) at the same time. Therefore, we call z_s the discriminative features and z_n the non-discriminative features[5].

4.4.1 New Scoring Function Based on the Decomposition

We can design a simple algorithm based on the decomposition: $M'(z) := \lambda M(z_s) + (1-\lambda)M(z_n)$, where $M(z)$ is the original scoring function. The coefficient λ controls the proportion of the desired mixing proportion of two features. When some exemplar OoD data is available, we can use them to tune λ. When there are no available OoD data, we can manually choose λ based on the prior knowledge of OoD data or set it as $1/2$. In our later experiments, if not specified, we set $\lambda = 1/2$ by default.

[5]Note the "discriminative" and "non-discriminative" terms are just names for the features. It is possible that the representations of a discriminatively trained neural netowrk only contain discriminative features. In this case, when we perform the decomposition, the extracted "non-discriminative" features would be "less-discriminative" features, i.e., features that contribute less to the classification (prediction) than the "discriminative" features.



(a) **(b)**

Figure 4.2: OoD detection using different features. CIFAR10 is the in-distribution dataset, and SVHN and CIFAR100 are two different OoD datasets. We can see that when using non-discriminative features, CIFAR100 is much closer to the in-distribution while SVHN is much further from the in-distribution.

4.4.2 Dataset Distance Metric Based on Decomposition

We can also design a new dataset distance metric by measuring the difference in the distributions of OoD scores. The decomposition gives such divergence a clear meaning: dataset distances in either discriminative or non-discriminative directions.

To make the scores of the two features comparable, we first normalize the discriminative (dis) and non-discriminative (non-dis) scores of the test data using the mean and standard deviation of the dis and non-dis scores calculated on the training data. To reflect the difference between the two distributions of scores $p_{in}(M(z))$ and $p_{out}(M(z))$, the standard practice is to use Kullback–Leibler (KL) divergence $D_{KL}(p_{in}\|p_{out}) = \sum_{k=1}^{K} p_{in,k} \log \frac{p_{in,k}}{p_{out,k}}$. In our case, since we only have the samples drawn from the two distributions p_{in}, p_{out}, we use an estimator of the KL-divergence based on k-nearst-neighbor distances [81]. We show the dataset distance in the extracted discriminative and non-discriminative directions in Table 4.2.

4.5 Experiments

4.5.1 OoD Detection on Simulated Data

We consider both low-dimensional (2D) simulation and high-dimensional (128D) simulation. For the low dimensional simulation, we generate two Gaussian distri-

butions with mean (3,0) and (-3,0) and tied covariance matrix $\Sigma = 0.5I_2$. The two OoD distributions are also Gaussian distributions with mean (3,1.4) and (1.6, 0) and the same covariance matrix $\Sigma = 0.3I_2$. We then use Equation (4.2) to calculate the Mahalanobis distance on the chosen features (full, discriminative, non-discriminative). The results in Figure 4.1 show that *Maha* calculated on the full features can only reach sub-optimal performance (94% AUROC) compared to on the decomposed features for the corresponding type of OoD data (98% AUROC).

For the high-dimensional simulation, we generate ten 128D Gaussian distributions $\{\boldsymbol{z}_c \sim \mathcal{N}(\boldsymbol{\mu}_c, \Sigma)\}_{c=1}^{10}$ with mean $\boldsymbol{\mu}_c = 10\boldsymbol{e}_c$, diagonal covariance matrix $\Sigma = I_{128}$. Here $\boldsymbol{e}_c$ is the standard basis. We also consider two types of OoD: (1) OoD along discriminative directions: $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$ where $\boldsymbol{\mu}_k = 5\boldsymbol{e}_k,\ k \in \{1, 2, ..., 10\}$.; (2) OoD along non-discriminative directions: $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{k'}, \Sigma)$ where $\boldsymbol{\mu}_{k'} = 10\boldsymbol{e}_c + 5\boldsymbol{e}_l$, $c \in \{1, 2, ..., 10\}, 11 \leq l \leq 128$. The remaining process of calculating Mahalanobis distance is the same as the low-dimensional simulation. When using Mahalanobis distance on the full feature, the OoD detection performance is 83.95% AUROC. While we we use Mahalanobis distance on the decomposed features, we can achieve 99.13% AUROC on OoD data along discriminative directions and 84.41% along non-discriminative directions. This shows that the OoD detection performance when using the full feature is heavily influenced by the non-discriminative features. This is similar to the CIFAR10 vs CIFAR100 benchmark (see Table 4.2).

4.5.2 OoD Detection on Image Datasets

Setup

For the experiments on image datasets, we use CIFAR10 [69] as the in-distribution datasets and SVHN [82], CIFAR100 [69], and the union of the two datasets as three different benchmarks. This is to demonstrate better how OoD datasets' differences influence the OoD detection performances using different methods.

We use WRN-28-10 [66] trained on CIFAR10 as our pretrained model to extract the penultimate features for different methods. For the decomposition using iCE loss, the invertible transformation is implemented as a 4-layer invertible

Table 4.2: AUROC (%) comparison of different methods calculated on features at penultimate layer (pre-logit).

Method	**Datasets**		
	D_{in} = CIFAR10 D_{out} = SVHN $d_{dis} = 3.75$ $d_{nondis} = 4.09$	D_{in} =CIFAR10 D_{out} =CIFAR100 $d_{dis} = 4.55$ $d_{nondis} = 1.15$	D_{in} =CIFAR10 D_{out} = SVHN ∪ CIFAR100 $d_{dis} = 2.08$ $d_{nondis} = 1.73$
Baseline (full feature)	98.31	86.15	92.23
PCA top score	94.59	90.20	92.39
PCA bottom score	98.23	83.77	91.00
PCA top score + bottom score	97.95	89.90	93.92
Discriminative score	94.76	**90.37**	92.56
Non-discriminative score	**98.59**	85.63	92.11
Dis score + Non-dis score	98.06	90.12	**94.09**

residual networks [59] with linear residual function and 0.9 spectral coefficient. The decomposition takes 3000 iterations using SGD optimizer, and we get 96.12% test accuracy for discriminative logits and 11.2% test accuracy for non-discriminative logits. For each method, we repeat the experiments using five differently randomly initialized models and report the mean.

Main Result

In Table 4.2, we show the comparison of different methods on the three OoD benchmarks. The scoring function we use is QDA-Maha. However, for other choices of scoring functions, we also see similar patterns. From Table 4.2, we can see that:

1. Our dataset distance metric is a good indicator of the OoD types and can explain the AUROC differences. For example, SVHN is far from CIFAR10 in both features and is further in non-discriminative features. So while using either type of feature can achieve high AUROC for detecting SVHN, non-discriminative features are slightly better. For CIFAR100, the distance in discriminative features is much larger than that in non-discriminative features. So when using discriminative scores to detect CIFAR100, we can see a considerable performance boost.

2. Discriminative score performs best on discriminative features, and non-discriminative score performs best on non-discriminative features. Moreover,

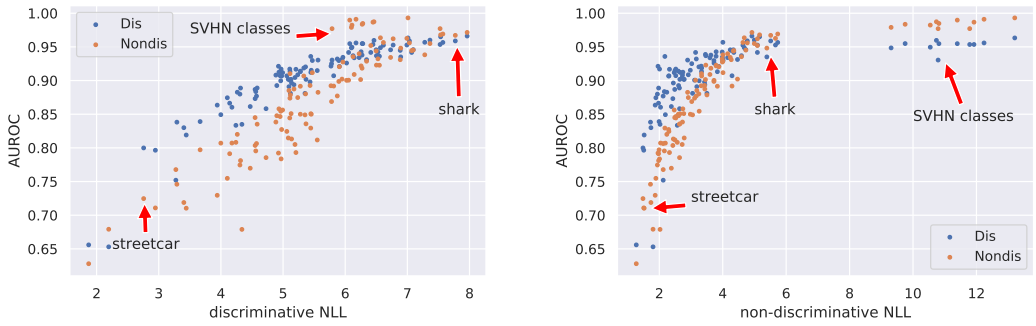


Figure 4.3: AUROCs of OoD detection using discriminative and non-discriminative features (y-axis) against the two dataset distances (x-axis). Every two dots with the same x-axis coordinate belong to the same class in either SVHN or CIFAR100.

the sum of the two scores combines the best of both features, yielding close to best performances on both benchmarks and best performance on the union benchmark.

3. PCA is already a reasonably good way for the decomposition. Note that this conclusion can change if we add the decomposition into the training process instead of using a discriminatively trained neural network.

To further understand the dataset distances under two different features, in Figure 4.2, we show the histograms of the two scores (log likelihoods) calculated on the three datasets. From the histogram, we can see that SVHN is far from CIFAR10 in both histograms, but the absolute values of non-discriminative log likelihoods are larger, so they can be easier to detect. For CIFAR100, there is a sharp difference between two features: the discriminative log likelihoods are more flat and far from CIFAR100, while the non-discriminative log likelihoods are very close to CIFAR10. So discriminative scores are a better choice to detect CIFAR100. These observations are aligned with the AUROC results of OoD detection and the dataset distances in Table 4.2.

4.5.3 Interpreting Uncertainty

The dataset distance under different features also provides us with a tool to interpret the uncertainty estimates. In Figure 4.3, we conduct a class-wise analysis for a fine-grained understanding of the OoD classes. Specifically, we calculate the AUROC using discriminative and non-discriminative scores to detect the 110 classes of SVHN ∪ CIFAR100 from CIFAR10 (one class at a time). This gives us two AUROC scores for each class, *dis AUROC* and *nondis AUROC*. We then plot them against the dataset distance between CIFAR10 and each class. In this way, we can interpret the OoD detection performances (y-axis) by looking at dataset distances (x-axis).

We highlight three types of OoD: (1) Classes with much higher non-discriminative distances than discriminative ones, e.g., *SVHN classes*. For these classes, non-discriminative features are more suitable for OoD detection. (2) Classes with small discriminative distances, e.g., *streetcar*. The small discriminative distances are usually due to similar categories in CIFAR10, e.g., automobile and truck for the streetcar. These classes typically also have small non-discriminative distances. Usually, discriminative distances are still larger, so more suitable for OoD detection. (3) Classes with large discriminative distances, e.g., *shark*. These classes typically also have large non-discriminative distances. The OoD performances using the two kinds of distances are usually similar.

4.6 Discussion

In this work, we showed that current state-of-the-art uncertainty estimation methods could not consistently outperform other methods across different benchmarks. We solved this problem by decomposing the features. Precisely, we summed the OoD scores calculated separately on the discriminative and non-discriminative features and achieved consistently high performance across different types of benchmarks. Our decomposition can also help us interpret the uncertainty of OoD data by looking at the uncertainty estimates on the discriminative/non-discriminative features separately.

We also note that the variants of Mahalanobis distance (*Maha*) all compute distances on the full features and change the extracted information by changing the class-dependence of the covariance matrix and the mean vectors. Although *Relative Maha* and *Marginal Maha* can be thought of as the counterparts of our scores on discriminative and non-discriminative features, our proposal of decomposition is still different in that we now allow a flexible combination of discriminative and non-discriminative scores (informed by our prior of OoD dataset or some exposure to the OoD data), and offer an easy way to interpret the uncertainty estimates (Figure 4.2, 4.3). Moreover, when we decompose, compute the OoD score, and integrate by summing them up, we can achieve better OoD detection performance compared to simply using the full features (Table 4.1, 4.2).

We hope future work can build on our analysis to develop new methods for interpreting the uncertainty estimates and integrating different information in features. We also hope that our dataset distance metric can serve as a reference for developing more comprehensive evaluation methods of OoD detection.

5 Discussion

Contents

5.1 Conclusions and Limitations

Representation learning for deterministic uncertainty estimation is a broad and challenging topic. In this dissertation, we have investigated two particular problems in this field.

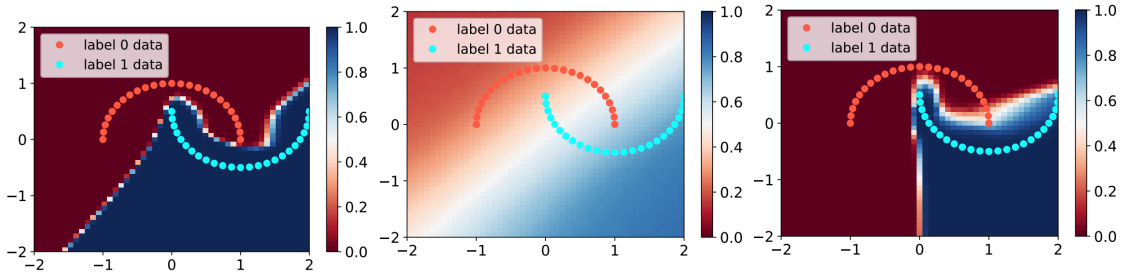
In Chapter 3, we investigated the representations learned by current implementations of bi-Lipschitz models. We demonstrated that the ResNets with spectral normalization can preserve low-frequency distances and thus prevent feature collapse in low-frequency contents under some mild conditions. This may explain the empirical success of the previous works [22, 23, 26]. Through experiments, we showed that the sufficient (theoretical) conditions of the bi-Lipschitzness, i.e., low-frequency dominance and $Lip(g) < 1$ requirement, are often not necessary for the feature extractor to be bi-Lipschitz in practice, suggesting that we may be able

to further relax the conditions. To address these theoretical limitations, further extensions of this work could explore how we can relax these conditions theoretically and enforce these conditions in our model explicitly. Besides, it is also beneficial to revisit the validity of the bi-Lipschitz constraint on the feature extractor. We will elaborate on this direction in the next section.

In Chapter 4, we studied the interaction of representations and uncertainty estimation. Specifically, by decomposing the representations into discriminative and non-discriminative ones, we can explain how the differences in various OoD detection benchmarks can influence the uncertainty estimation results. For instance, some OoD data are closer to the in-distribution data in non-discriminative directions. Thus it is harder to get reasonable uncertainty estimates when mainly using non-discriminative information/representations. Based on our insights, we proposed a simple method to integrate the uncertainty estimates calculated separately on discriminative and non-discriminative features and achieved consistently high performance across different types of benchmarks. Due to time constraints, we limit the experiments to only two image OoD detection benchmarks in this project. We also lack enough ablation studies of our decomposition method. In the future, we hope to extend this work by experimenting with more benchmarks and conducting ablation studies to understand further the properties of the decomposed representations (e.g., within-class variance and between-class variance) and the influence of different components of the decomposition. Besides, our work also sheds light on how to build interpretable uncertainty estimation methods that can perform well across different tasks. We will discuss this more in the next section.

5.2 Outlook

In this final section, we go further from our two projects in this dissertation and discuss two more interesting future directions in representation learning for deterministic uncertainty estimation.



(a) 4-layer MLP

(b) 4-layer MLP; spectral coefficient = 1.

(c) 4-layer MLP; spectral coefficient = 10.

Figure 5.1: Results of softmax prediction probability of three models with different regularization scheme on the two moon dataset. We can see that spectral normalization has a great impact on the model capacity. Figure is taken from [83].

5.2.1 Rethinking the Bi-Lipschitz Constraint

As discussed in Chapter 3, the bi-Lipschitz constraint essentially controls how the output of a function varies with changes in its input with a lower and upper bound. However, it is questionable whether such constraint is desirable for representations.

In previous works, the motivation to use the bi-Lipschitz constraint is to fix the feature collapse problem [21, 23, 26]. However, in the description of the feature collapse, we only look at one side of the coin: the collapse of features from OoD data with in-distribution data. While such collapse is undesirable, there are also desirable collapse (invariance) in representations. For example, we may want the representations to be invariant to the changes of backgrounds in object classification tasks. Since current implementations of bi-Lipschitz models (ResNets with spectral normalization) will enforce bi-Lipschitzness as a global property, the desirable invariances will also be reduced. This suggests that instead of enforcing a global bi-Lipschitz constraint, we may need to consider the task dependence of representations and provide different (heterogeneous) regularization in different data regions. Besides, the bi-Lipschitz constraint may also severely limit the capacity of the model since the l_2 distances in the input space are notorious for not being an informative distance metric for natural images. In Figure 5.1, we show an example of how spectral normalization may fail when input space distances are not very informative about the classification.

Therefore, although it may be beneficial to add regularization on the smoothness (bi-Lipschitzness) of the learned function, we have to rethink how to perform it. Conceptually, we may need to find a transformation h such that the bi-Lipschitz constraint on the learned function with $h(x)$ as input can consider the heterogeneity of desirable invariances and the task-dependence. As a preliminary experiment, during the MSc project, we have tried to combine equivariant models [84, 85] with spectral normalization. That is, we first encode a few desired invariances using methods like equivariant CNNs [84] and then add bi-Lipschitz regularization globally. In this way, we may keep some desirable invariances while mitigating the undesirable ones that cause feature collapse. Unfortunately, such models cannot provide much empirical improvement, possibly because we have a very limited set of invariances to encode (i.e., shift-invariance, rotation-invariance, and reflection-invariance). This may suggest that future works should focus on learnable invariances [86] instead of manually defined invariances when working with representation learning for uncertainty estimation. Overall, I believe this would be a promising and fruitful direction to explore.

5.2.2 Interpreting the Uncertainty Estimates

Uncertainty and interpretability of DNNs are normally considered as two separate problems in trustworthy machine learning [87, 88]. The intersection of them, i.e., interpretability of uncertainty estimation, however, is relatively less studied. By interpreting the uncertainty estimates, we can understand why the model is uncertain about the inputs (which is different from why the model reaches specific predictions as investigated in current literature [89, 90]). Also, this would provide insights for researchers to improve the current uncertainty estimation methods, e.g., developing uncertainty estimates that can perform consistently well in different tasks. Therefore, I think this is also an important direction in this field.

Many methods in the interpretable AI literature may be transferred to help explain the uncertainty. For example, the disentanglement (decomposition) of features [91, 92] may help us understand the contributions of different types of

features to the uncertainty. Our work in Chapter 4 is an initial attempt in this direction. In the future, other ways to disentangle the features can be used, e.g., into shape, background, and texture [93]. In addition, counterfactual explanations can also be used to explain the uncertainty estimation [94]. These methods aim to explain the uncertainty by finding the smallest change that could be made to an input to lower its estimated uncertainty while keeping it in distribution. It would be interesting to see how such methods could be extended to explain the uncertainty estimation on OoD data. Besides, the gradient visualization techniques in explainable AI [95–97] can also help us understand the contributions of different features to the uncertainty.

In the future, with progress in the interpretability of uncertainty estimation, we will be able to develop more advanced methods for uncertainty estimation and trustworthy machine learning in general.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778. URL: http://ieeexplore.ieee.org/document/7780459/ (visited on 05/31/2020).

[3] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[4] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] Maria Joao Cardoso et al. "Artificial Intelligence (AI) in Breast Cancer Care - Leveraging multidisciplinary skills to improve care". In: *Artificial Intelligence in Medicine* (2020), p. 102000. URL: https://www.sciencedirect.com/science/article/pii/S0933365720312653.

[6] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare". In: *Artificial Intelligence in Medicine* 104 (2020), p. 101822. URL: https://www.sciencedirect.com/science/article/pii/S0933365719310437.

[7] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 427–436.

[8] Robert Geirhos et al. "Shortcut learning in deep neural networks". en. In: *Nature Machine Intelligence* 2.11 (Nov. 2020). Bandiera_abtest: a Cg_type: Nature Research Journals Number: 11 Primary_atype: Reviews Publisher: Nature Publishing Group Subject_term: Computational science;Human behaviour;Information technology;Network models Subject_term_id: computational-science;human-behaviour;information-technology;network-models, pp. 665–673. URL: https://www.nature.com/articles/s42256-020-00257-z (visited on 08/23/2021).

[9] Christiane Fellbaum, ed. *WordNet: An Electronic Lexical Database*. en. Language, Speech, and Communication. Cambridge, MA, USA: A Bradford Book, May 1998.

[10] Mariusz Bojarski et al. “End to End Learning for Self-Driving Cars”. In: *CoRR* abs/1604.07316 (2016). arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316.

[11] NHTSA. PE 16-007. “Tesla Crash Preliminary Evaluation Report”. In: *Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration* (Jan. 2017).

[12] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *CoRR* abs/1502.01852 (2015). arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[14] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. en. In: *Nature* 323.6088 (Oct. 1986). Bandiera_abtest: a Cg_type: Nature Research Journals Number: 6088 Primary_atype: Research Publisher: Nature Publishing Group, pp. 533–536. URL: https://www.nature.com/articles/323533a0 (visited on 08/27/2021).

[15] Dan Hendrycks and Kevin Gimpel. “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks”. In: *Proceedings of International Conference on Learning Representations* (2017).

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

[17] Alexander Meinke and Matthias Hein. “Towards neural networks that provably know when they don’t know”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=ByxGkySKwH.

[18] Yarin Gal. “Uncertainty in Deep Learning”. In: 2016.

[19] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. URL: http://proceedings.mlr.press/v48/gal16.html.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 6402–6413. URL: http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf.

[21] Joost van Amersfoort et al. “Uncertainty Estimation Using a Single Deep Deterministic Neural Network”. In: *International Conference on Machine Learning (ICML*. 2020.

[22] Balaji Lakshminarayanan et al. “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: *Advances in Neural Information Processing Systems 33*. 2020.

[23] Joost van Amersfoort et al. "On Feature Collapse and Deep Kernel Learning for Single Forward Pass Uncertainty". In: *arXiv preprint arXiv:2102.11409* (2021).

[24] Kimin Lee et al. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 7167–7177.

[25] Haiwen Huang et al. "Feature Space Singularity for Out-of-Distribution Detection". In: *Proceedings of the Workshop on Artificial Intelligence Safety 2021 (SafeAI 2021)*. 2021.

[26] Jishnu Mukhoti et al. *Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty*. 2021. arXiv: 2102.11582 [cs.LG].

[27] Chandramouli Shama Sastry and Sageev Oore. "Detecting Out-of-Distribution Examples with In-distribution Examples and Gram Matrices". In: *CoRR* abs/1912.12510 (2019). arXiv: 1912.12510. URL: http://arxiv.org/abs/1912.12510.

[28] Jakob Gawlikowski et al. *A Survey of Uncertainty in Deep Neural Networks*. 2021. arXiv: 2107.03342 [cs.LG].

[29] Swapnil Mishra et al. "Changing composition of SARS-CoV-2 lineages and rise of Delta variant in England". In: *EClinicalMedicine* 39 (2021), p. 101064. URL: https://www.sciencedirect.com/science/article/pii/S2589537021003448.

[30] Benjamin Recht et al. "Do ImageNet Classifiers Generalize to ImageNet?" In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 5389–5400. URL: https://proceedings.mlr.press/v97/recht19a.html.

[31] Dan Hendrycks and Thomas Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HJz6tiCqYm.

[32] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1321–1330. URL: https://proceedings.mlr.press/v70/guo17a.html.

[33] Hao Li et al. "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf.

[34] Haiwen Huang, Chang Wang, and Bin Dong. "Nostalgic adam: Weighting more of the past gradients when designing the adaptive learning rate". In: *arXiv preprint arXiv:1805.07557* (2018).

[35] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5574–5584. URL: http://papers.nips.cc/paper/7141-what-uncertainties-do-we-need-in-bayesian-deep-learning-for-computer-vision.pdf.

[36] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

[37] Stefan Depeweg et al. "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1184–1193. URL: https://proceedings.mlr.press/v80/depeweg18a.html.

[38] Javier Antoran, James Allingham, and José Miguel Hernández-Lobato. "Depth Uncertainty in Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 10620–10634. URL: https://proceedings.neurips.cc/paper/2020/file/781877bda0783aac5f1cf765c128b437-Paper.pdf.

[39] Pablo Morales-Alvarez et al. "Activation-level uncertainty in deep neural networks". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=UvBPbpvHRj-.

[40] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[41] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.

[42] Joost van Amersfoort et al. *Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression*. 2021. arXiv: 2102.11409 [cs.LG].

[43] Jie Ren et al. "A Simple Fix to Mahalanobis Distance for Improving Near-OOD Detection". In: *arXiv:2106.09022 [cs]* (June 2021). arXiv: 2106.09022. URL: http://arxiv.org/abs/2106.09022 (visited on 08/03/2021).

[44] Dan Hendrycks et al. "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[45] Jim Winkens et al. "Contrastive Training for Improved Out-of-Distribution Detection". In: *CoRR* abs/2007.05566 (2020). arXiv: 2007.05566. URL: https://arxiv.org/abs/2007.05566.

[46] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. "Deep Anomaly Detection with Outlier Exposure". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyxCxhRcY7.

[47] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. "Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 41–50.

[48] Jiefeng Chen et al. "Robust Out-of-distribution Detection via Informative Outlier Mining". In: *CoRR* abs/2006.15207 (2020). arXiv: 2006.15207. URL: https://arxiv.org/abs/2006.15207.

[49] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=B1QRgziT-.

[50] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=H1VGkIxRZ.

[51] Ryo Kamoi and Kei Kobayashi. "Why is the Mahalanobis Distance Effective for Anomaly Detection?" In: *arXiv:2003.00402 [cs, stat]* (Apr. 2020). arXiv: 2003.00402. URL: http://arxiv.org/abs/2003.00402 (visited on 05/31/2020).

[52] Eric Nalisnick et al. "Do Deep Generative Models Know What They Don't Know?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=H1xwNhCcYm.

[53] Robin Schirrmeister et al. "Understanding Anomaly Detection with Deep Invertible Networks through Hierarchies of Distributions and Features". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21038–21049. URL: https://proceedings.neurips.cc/paper/2020/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf.

[54] Jie Ren et al. "Likelihood Ratios for Out-of-Distribution Detection". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 14707–14718. URL: http://papers.nips.cc/paper/9611-likelihood-ratios-for-out-of-distribution-detection.pdf (visited on 05/31/2020).

[55] Eric Nalisnick et al. *Detecting Out-of-Distribution Inputs to Deep Generative Models Using Typicality*. 2019. arXiv: 1906.02994 [stat.ML].

[56] Robin Schirrmeister et al. "Understanding Anomaly Detection with Deep Invertible Networks through Hierarchies of Distributions and Features". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21038–21049. URL: https://proceedings.neurips.cc/paper/2020/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf.

[57] Jesse Davis and Mark Goadrich. "The Relationship between Precision-Recall and ROC Curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240. URL: https://doi.org/10.1145/1143844.1143874.

[58] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 1999. URL: http://nlp.stanford.edu/fsnlp/.

[59] Jens Behrmann et al. "Invertible Residual Networks". en. In: *International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, May 2019, pp. 573–582. URL: http://proceedings.mlr.press/v97/behrmann19a.html (visited on 12/16/2020).

[60] Henry Gouk et al. *Regularisation of Neural Networks by Enforcing Lipschitz Continuity*. 2020. arXiv: 1804.04368 [stat.ML].

[61] Lewis Smith et al. *Can convolutional ResNets approximately preserve input distances? A frequency analysis perspective*. 2021. arXiv: 2106.02469 [cs.LG].

[62] C.E. Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[63] David Finlay, Peter Dodwell, and Terry Caelli. "The Waggon-Wheel Effect". In: *Perception* 13.3 (1984). PMID: 6514509, pp. 237–237. eprint: https://doi.org/10.1068/p130237. URL: https://doi.org/10.1068/p130237.

[64] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[65] Richard Zhang. "Making Convolutional Networks Shift-Invariant Again". In: *ICML*. 2019.

[66] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *BMVC*. 2016.

[67] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[68] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

[69] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: *Tech Report* (2009).

[70] George E. P. Box. "Science and Statistics". In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799. URL: http://www.jstor.org/stable/2286841.

[71] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 1396–1408. URL: http://papers.nips.cc/paper/8420-failing-loudly-an-empirical-study-of-methods-for-detecting-dataset-shift.pdf (visited on 05/31/2020).

[72] Viraj Bhise et al. "Defining and measuring diagnostic uncertainty in medicine: a systematic review". In: *Journal of general internal medicine* 33.1 (2018), pp. 103–115.

[73] Angelos Filos et al. "Can Autonomous Vehicles Identify, Recover From, and Adapt to Distribution Shifts?" In: *International Conference on Machine Learning (ICML)*. 2020.

[74] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.

[75] Dong Yin et al. "A Fourier Perspective on Model Robustness in Computer Vision". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/b05b57f6add810d3b7490866d74c0053-Paper.pdf.

[76] Siqi Deng et al. "Harnessing Unrecognizable Faces for Face Recognition". In: *CoRR* abs/2106.04112 (2021). arXiv: 2106.04112. URL: https://arxiv.org/abs/2106.04112.

[77] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: probml.ai.

[78] Ryo Kamoi and Kei Kobayashi. "Why is the Mahalanobis Distance Effective for Anomaly Detection?" In: *arXiv:2003.00402 [cs, stat]* (Apr. 2020). arXiv: 2003.00402. URL: http://arxiv.org/abs/2003.00402 (visited on 08/03/2021).

[79] G. H. Golub and C. Reinsch. "Singular Value Decomposition and Least Squares Solutions". In: *Numer. Math.* 14.5 (Apr. 1970), pp. 403–420. URL: https://doi.org/10.1007/BF02163027.

[80] Joern-Henrik Jacobsen et al. "Excessive Invariance Causes Adversarial Vulnerability". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=BkfbpsAcF7.

[81] Qing Wang, Sanjeev R Kulkarni, and Sergio Verdú. "Divergence estimation for multidimensional densities via k-nearest-neighbor distances". In: *IEEE Transactions on Information Theory* 55.5 (2009), pp. 2392–2405.

[82] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2011. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[83] Mihaela Rosca et al. *A case for new neural network smoothness constraints*. 2021. arXiv: 2012.07969 [stat.ML].

[84] Maurice Weiler and Gabriele Cesa. "General E(2)-Equivariant Steerable CNNs". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2019.

[85] Taco Cohen and Max Welling. "Group Equivariant Convolutional Networks". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2990–2999. URL: https://proceedings.mlr.press/v48/cohenc16.html.

[86] Gregory Benton et al. "Learning Invariances in Neural Networks from Training Data". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 17605–17616. URL: https://proceedings.neurips.cc/paper/2020/file/cc8090c4d2791cdd9cd2cb3c24296190-Paper.pdf.

[87] K. Siau and Weiyu Wang. "Building Trust in Artificial Intelligence, Machine Learning, and Robotics". In: 2018.

[88] Philipp Schmidt and Felix Bießmann. "Quantifying Interpretability and Trust in Machine Learning Systems". In: *CoRR* abs/1901.08558 (2019). arXiv: 1901.08558. URL: http://arxiv.org/abs/1901.08558.

[89] Umang Bhatt et al. "Explainable Machine Learning in Deployment". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* '20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 648–657. URL: https://doi.org/10.1145/3351095.3375624.

[90] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15. URL: https://www.sciencedirect.com/science/article/pii/S1051200417302385.

[91] Irina Higgins et al. "Towards a Definition of Disentangled Representations". In: *CoRR* abs/1812.02230 (2018). arXiv: 1812.02230. URL: http://arxiv.org/abs/1812.02230.

[92] Babak Esmaeili et al. "Structured Disentangled Representations". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 16–18 Apr 2019, pp. 2525–2534. URL: https://proceedings.mlr.press/v89/esmaeili19a.html.

[93] Axel Sauer and Andreas Geiger. "Counterfactual Generative Networks". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=BXewfAYMmJw.

[94] Javier Antoran et al. "Getting a {CLUE}: A Method for Explaining Uncertainty Estimates". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=XSLF1XFq5h.

[95] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *Workshop at International Conference on Learning Representations*. 2014.

[96] Daniel Smilkov et al. "SmoothGrad: removing noise by adding noise". In: *CoRR* abs/1706.03825 (2017). arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825.

[97] Mingwei Li, Zhenge Zhao, and Carlos Scheidegger. "Visualizing Neural Networks with the Grand Tour". In: *Distill* (2020). https://distill.pub/2020/grand-tour.