Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Chemistry, Biotechnology and Food Science

Outlier Detection in Bayesian Neural Networks

Herman Ellingsen
MSc Data Science

Abstract

Exploring different ways of describing uncertainty in neural networks is of great interest. Artificial intelligence models can be used with greater confidence by having solid methods for identifying and quantifying uncertainty. This is especially important in high-risk areas such as medical applications, autonomous vehicles, and financial systems. This thesis explores how to detect classification outliers in Bayesian Neural Networks. A few methods exist for quantifying uncertainty in Bayesian Neural Networks, such as computing the Entropy of the prediction vector. Is there a more accurate and broad way of detecting classification outliers in Bayesian Neural Networks? If a sample is detected as an outlier, is there a way of separating between different types of outliers?

We try to answer these questions by using the pre-activation neuron values of a Bayesian Neural Network. We compare, in total, three different methods using simulated data, the Breast Cancer Wisconsin dataset and the MNIST dataset. The first method uses the well-researched Predictive Entropy, which will act as a baseline method. The second method uses the pre-activation neuron values in the output layer of a Bayesian Neural Network; this is done by comparing the pre-activation neuron value from a given data sample with the pre-activation neuron values from the training data. Lastly, the third method is a combination of the first two methods.

The results show that the performance might depend on the dataset type. The proposed method outperforms the baseline method on simulated data. When using the Breast Cancer Wisconsin dataset, we see that the proposed method is significantly better than the baseline. Interestingly, we observe that with the MNIST dataset, the baseline model outperforms the proposed method in most scenarios. Common for all three datasets is that the combination of the two methods performs approximately as well as the best of the two.

Preface

This master thesis is written in an article format. According to the guidelines for writing an article-based master thesis, I have also written a mantel, an extended introduction to the article. This mantel will be presented first, introducing the topic and giving a more in-depth review of the relevant theory and methodological choices. It will also elaborate on the results and the discussion of the results found when working on the article.

The article "Outlier Detection in Bayesian Neural Networks" will be presented in its entirety after the mantel. For now, the article is not written for a specific journal, so it is not made with regard to any journal-specific guidelines.

Contents

List of Figures

Abbreviations

AI	Artificial Intelligence
OOD	Out-Of-Distribution
IB	In-Between
ANN	Artificial Neural Network
SNN	Stochastic Neural Network
BNN	Bayesian Neural Network
MCMC	Markov Chain Monte Carlo
VI	Variational Inference
BBB	Bayes By Backprop
FDR	False Discovery Rate
PCA	Principal Component Analysis

Chapter 1

Introduction

Over the last decade, deep learning models have been adopted in a wide range of fields due to their ability to solve complicated problems. However, due to the complex nature of deep learning models, they are often treated as "black boxes", and their uncertainty related to predictions can be hard to quantify. This makes it problematic to adopt deep learning models in areas where making the right decision is critical. This has increased the interest in measuring uncertainty in deep learning models. Several approaches have been suggested, such as Probabilistic Modelling [1], Ensemble Methods [2], and Dropout [2]. This thesis will focus on Probabilistic Modelling, specifically Bayesian Inference, as the proposed method uses a Bayesian Neural Network.

This mantel will, in Section **1** introduce the topic and background of this thesis. Secondly, Section **2** will present relevant theory and concepts used in writing this thesis. The proposed method of this master thesis will be introduced in depth in Section **3**. In Section **4**, results are presented. These results will be discussed in Section **5**.

1.1 Background and Motivation

Providing a measure of uncertainty related to predictions in deep learning models is crucial in many fields of application, such as medical applications [3], autonomous vehicles [4], and financial systems [5]. This section will provide examples of application areas where uncertainty measures are necessary for avoiding undesirable results.

1.1.1 Medical Applications

The use of Artificial Intelligence (AI) in medicine is rapidly growing as the healthcare industry evolves into a more data-driven field. Use cases of AI application in medicine include disease detection and diagnosis [6], personalized disease treatment [7], medical imaging [8], accelerated drug development [9], and clinical trial efficiency [10]. For instance, a comparison between a stand-alone artificial intelligence system and 101 radiologists for detecting breast cancer in mammography showed that the AI system was statistically non-inferior to the average radiologist [8].

The use of artificial intelligence in medicine can however become problematic in cases where the model is uncertain and should ideally be able to refrain from making a decision. For human physicians, the natural approach when met with unusual and difficult clinical cases is to communicate uncertainty and seek second opinions from colleagues. However, AI models do not necessarily communicate uncertainty and seek second opinions when met with clinical cases of high uncertainty, although this is critically important in most medical applications. This is made clear when three of the most cited articles about AI models in medicine [11] [12] [13] since 2016 do not have mechanisms for abstaining when met with high uncertainty.

1.1.2 Autonomous Systems

Autonomous systems range from simple robotic vacuum cleaners in private homes to self-driving cars out in public. One can generally divide autonomous systems into rule-based systems, which use low-level machine learning algorithms, and those that can learn from their environment, i.e. reinforcement learning. Autonomous vehicles have, over the last decade, been an area of focus for many of the major car companies [14], where even technology companies like Apple and Google have their own autonomous vehicle projects [15].

While autonomous vehicles are becoming increasingly better, they still face major challenges concerning safety. Can AI-based systems that control self-driving cars be trusted to make the right decision? The self-driving cars use low-level feature extraction, such as image segmentation and localization, to process raw sensory input [16]. This is then fed into high-level decision-making systems, which have algorithms based on a set of rules, such as "if a cyclist is to your left, do not steer left". This could be problematic if mistakes made in the lower-level systems are fed into the high-level systems. This was made clear in a fatal car crash involving a Tesla Model S, and a tractor-trailer [17]. Here, the crash was caused by low-level systems in the Tesla not being able to distinguish between a bright sky and the white side of the trailer. Ideally, the low-level and high-level systems should be taking the uncertainty into account and be able to alert and hand over the control of the vehicle to the driver.

1.1.3 Financial Systems

The recent boom in FinTech, i.e. financial technology, has shown the great potential of artificial intelligence in the finance sector, much due to the recent growth in the volume of digital data and computational capacities. Examples include the use of chatbots in front office [18], anti-fraud detection systems [19], and high-frequency trading systems [20].

In the case of high-frequency trading systems, AI models are given control over systems that could potentially destabilize entire financial markets. This makes it difficult to justify using uncertainty-unaware AI models in trading. One approach would be verifying each trade manually, but this is practically impossible in high-frequency trading. If the model would provide uncertainty measures for each trade, one could "flag" the trades with high uncertainty such that a human can handle or verify these.

Chapter 2

Theory

2.1 Uncertainty in Deep Learning

Uncertainty in deep learning is commonly separated into two types of uncertainty: aleatoric and epistemic uncertainty.

Epistemic uncertainty [21], sometimes referred to as systematic uncertainty, is the uncertainty related to the model, which is often due to a lack of training data, poor model specifications and/or having a too simple model. Epistemic uncertainty is reducible, so increasing the amount of data and/or improving model specifications would decrease the epismtic uncertainty. One would expect a given model to have high epismetic uncertainty when met with Out-Of-Distribution (OOD) data samples. OOD data samples are samples drawn from a different distribution than the training data. Likewise, we would expect to see low epistemic uncertainty from samples drawn from the training data distribution.

Aleatoric uncertainty [21] represents the inherent randomness in the data. Data samples will never give perfect representation of the real world, because it will allways to some degree be contaminated by randomness and/or noise. Aleatoric uncertainty is not reduicble, because increasing the amount of data would also introduce more randomness and/or noise. We therefore say that aleatoric uncertainty is not a result of the model, because it is a property of the data.

Prediction uncertainty encapsulates both the epistemic and the aleatoric uncertainty.

2.1.1 Uncertainty Quantification

Quantifying uncertainty is the process of estimating uncertainty in neural network models. There exist several approaches to uncertainty quantification in deep learning, such as:

1. **Probabilistic Modelling**: Probabilistic Modelling, such as Bayesian Neural Networks [22, 1] (described in more detail in Section 2.4), can incorporate probabilistic representations that can give probabilistic predictions and uncertainty quantifications. This method can capture both aleatoric and epistemic uncertainty, which leads to a more complete picture of the uncertainty associated with the predictions.

2. **Deep Ensembles**: Deep ensemble methods in deep learning refers to combining N models to increase the predictive performance [23]. A prediction is the combination of these N models. These methods can not only improve the model accuracy but can also be used for uncertainty estimation. This is done by looking at the variance of the predictions from the N models. In the case of using deep ensembles for uncertainty estimation, one use the same type of models but use randomized initial weights. One should note that deep ensembles will only provide epistemic uncertainty.

3. **Dropout**: Dropout in deep learning is a regularization technique for reducing overfitting and improving model generalization. It has been widely adopted since it was first introduced in 2014 [24]. The term "dropout" refers to that certain nodes, in both input and hidden layers, are randomly dropped during training. Randomly dropping nodes forces the model to be less reliant on specific nodes. Dropout can be used for uncertainty estimation by applying dropout at test time and averaging the prediction over multiple dropout samples. As with deep ensembles, dropout also only estimates the epistemic uncertainty.

In this thesis, we will focus on the probabilistic approach using Bayesian Neuron Networks.

2.2 Artifical Neural Networks

An **Artifical Neural Network (ANN)** is a computational network inspired by the biological neural networks that make up the structure of the human brain. As with the human brain, an ANN has interconnected neurons, also referred to as nodes, that are linked up together in various layers of the network, as seen in Figure 2.2.1.

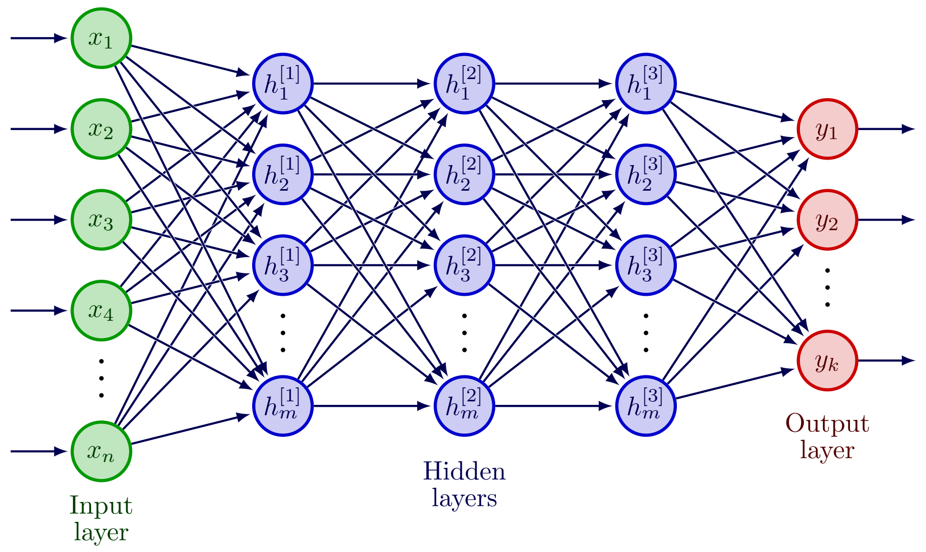

Figure 2.2.1: A fully connected neural network, with n input features, three hidden layers with m neurons each, and an output layer consisting of k neurons

A standard ANN with L layers can be structured as follows: an input layer l^0, $L-1$ hidden layers l^l, and an output layer l^L. The structure of a basic feedforward neural network is then, where each layer, except for the input layer, contains a linear transformation $(W^{[l]} \times l^{[l-1]} + b^{[l]})$ followed by a non-linear activation function $g^{[l]}$ [1]:

$$\begin{aligned} &l^{[0]} = x \\ &l^{[l]} = g^{[l]}(W^{[l]} \times l^{[l-1]} + b^{[l]}), l = 1, ..., L \\ &l^{[L]} = y \end{aligned}$$

This network contains weight parameters W and bias parameters b: $\theta = (W, b)$. The training process of an ANN is then to learn these parameters θ from a given training data set $\mathcal{D}$. Here, $\mathcal{D}$ is a series of input samples x and their target labels y. The activation functions in neural networks vary, where Sigmoid [25] and ReLU [26] are amongst the most popular. The sigmoid function takes in a real value and converts it into a value between 0 and 1:

$$g_{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

One problem with using the sigmoid function as an activtion function is that this can cause the vanishing gradients problem [27], which makes deep neural networks unable to propogate useful gradient information from the output. A solution to this problem is to use the ReLU function, which is defined as:

$$g_{ReLU}(x) = \max(0, x)$$

which set all values below zero to zero. In the case of building a model for classification, one usually use the softmax activation function in the last layer:

$$g_{softmax}(x)_i = \frac{e^{x_i}}{\sum_i^K e^{x_i}} \tag{2.1}$$

where K corresponds to the number of classes. The softmax function takes in a vector of K real numbers x and normalizes it into a probability distribution of K probabilities that sum to 1. Prior to applying the softmax function, the vector components can be any real number, but after activation, they will be real numbers in the interval $[0, 1]$. From Equation (2.1), we see that a larger input corresponds to a larger probability.

As mentioned, training a neural network is the process of learning a set of parameters θ from the training data $\mathcal{D}$. This is usually done by using first order optimization that use the backpropagation algorithm. The backpropagation algorithm computes the gradients of a loss function J with respect to the network parameters θ. These graidents are then used to update the parameters θ in order to minimize the loss function. Here, a given data sample x is forward passed through the ANN, leading to a prediction $\hat{y}$. This prediction $\hat{y}$ is compared to the true output label from the training data y. The discrepency between $\hat{y}$ and y is quantified in the loss function $J(\theta)$. Then a backward pass is made, where the gradients of the loss function with respect to the parameters θ are computed. The next step is to use the computed gradients to update the model parameters:

$$\theta_{t+1} = \theta_t - \eta\hat{\nabla}J(\theta_t)$$

where η denotes the learning rate, which controls how much the parameters θ are changed in response to the estimated loss $J(\theta)$ in each parameter update, i.e. the rate for convergence. This is an important hyperparameter because setting this too low or too high will negatively affect the training process. Too low leads to a slow convergence time, while setting it too high it can converge too quickly to a suboptimal solution. One iteration of this process for the entire training data is referred to as an epoch.

The loss function also requires a so-called optimization strategy, which is an algorithm for optimizing the loss function. Adam [28], short for Adaptive Moment Estimation, is a popular choice for optimizing the parameters of neural networks due to its adaptive learning rate mechanism and momentum-based updates.

ANNs have a tendency to overfit, which means that the model fits exactly against the training data, but does not genarlize well with unseen data. It is therefore common to include some form of regularization in the training process. Regularization is a technique used to prevent the model from overfitting by lowering the complexity of the network. Common types of regularization include L1, L2, and dropout [29]. L2 regularization, which is also known as weight decay or Ridge Regression, is amongst the most common regularization techniques. The L2-term is defined as the sum of all squared weight values in a weight matrix, also known as the Euclidean Norm of a weight matrix (Equation (2.2)):

$$\Omega(\theta) = \|\theta_2^2\| = \sum_i \sum_j \theta_{ij}^2 \tag{2.2}$$

During the training of an ANN with L2 regularization one extends the loss function $J(\theta)$ with the so-called regularization term, as shown in Equation (2.3). From Equation (2.3) one can see that the regularization term is weighted by a scalar λ. This scaler λ is known as the regularization rate and becomes an additional hyperparameter in the network:

$$J(\theta) = \frac{\lambda}{2}\|\theta_2^2\| + J(\theta) = \frac{\lambda}{2}\sum_i \sum_j \theta_{ij}^2 + J(\theta) \tag{2.3}$$

2.3 Stochastic Neural Networks

A **Stochastic Neural Network (SNN)** [1] uses stochastic components in the model. This can be done in two ways: either by giving the network stochastic activations or by stochastic weight parameters. Figure 2.3.1 shows the difference between a regular ANN, with fixed weight parameters, and a stochastic neural network with stochastic weight parameters. SNNs will, in contrast to traditional ANNs, simulate multiple values of parameters θ with their associated probability function $p(\theta)$ when making a prediction. One can therefore consider SNNs to be a special type of ensemble learning. When using ensemble learning with traditional ANNs one combines predictions from T different independent average-performing models with the assumption that these will outperform a single well-trained model. However, this is not necessarily the main intention when ensembling with SNNs. The main objective is rather to use ensembling to describe the uncertainty related to predictions for a given network structure. As comparing the predictions from the T different stochastic forwards passes in the network gives an uncertainty estimate.

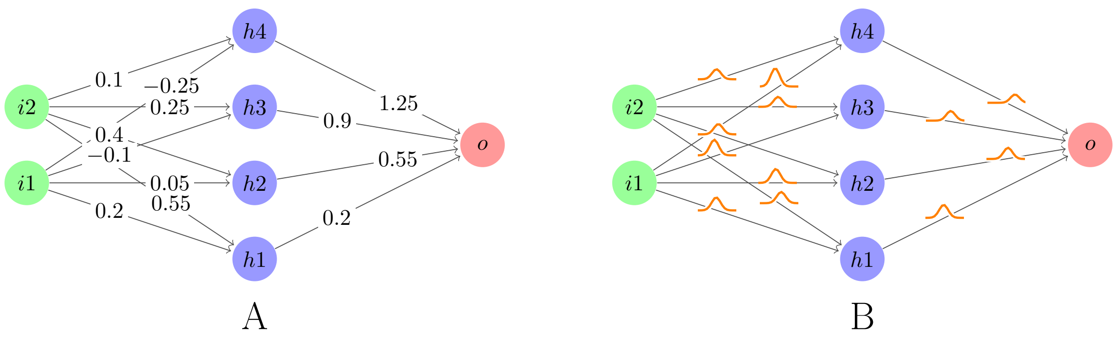

Figure 2.3.1: Both **Figure A** and **B** are neural networks with an input layer containing two features, a hidden layer with four neurons, and an output layer with a single neuron. **Figure A** illustrates a regular ANN, i.e. fixed parameter values. **Figure B** illustrates a neural network with stochastic weight parameters.

2.4 Bayesian Neural Networks

A **Bayesian Neural Network (BNN)** is a type of stochastic neural network which is trained using Bayesian Inference [30]. For a regular ANN we assume fixed weight parameters, whereas in a BNN we assume that the model parameters follow some probability distribution. We can define a prior distribution $p(\theta)$ over all model parameters θ, which includes weight parameters W and bias terms b. The Bayesian approach in statistics is different from the traditional frequentist approach in two main areas: in the way it defines probabilities and in the ability to incorporate prior beliefs. The classical frequentist approach is to interpret a probability as the relative frequency of occurances when the number of samples approach infinity, i.e. the long-run probability [31]. The Bayesian inpretation is on the contrary to treat the probability as a quantification of a personal belief [32]. Secondly, prior belief $p(\theta)$ can be updated in the light of new observations into a posterior belief, $p(\theta|\mathcal{D})$. This is described in Bayes' Theorem [33]:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

where $p(\mathcal{D}|\theta)$ denotes the likelihood of the data and $p(\mathcal{D})$ represents the marginal distribution of the data $\mathcal{D}$, which is calculated by:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta \tag{2.4}$$

In high dimensions, the integral in Equation (2.4) is intractable. Hence, sampling directly from the posterior will not be possible, and using posterior predictive distributions will not be straightforward. However, there are established methods for dealing with this problem, such as Markov Chain Monte Carlo [34] and variational inference [35]. These will be presented in more detail in Section 2.4.1 and 2.4.2.

In regular ANNs, one typically use the maximum likelihood estimates of θ when making a prediction $\hat{y}$ on new data. In BNNs however, the posterior predictive distribution $p(\hat{y}|\mathcal{D})$ is used when doing inference on new data:

$$p(\hat{y}|x, \mathcal{D}) = \int p(\hat{y}|x, \theta)p(\theta|\mathcal{D})d\theta$$

When making a prediction for a new data sample x, one will make multiple, say T, stochastic forward passes through the model. The average prediction from these T stochastic forward passes will give a relative probability of each class $p(\hat{y}|x, \mathcal{D})$. For classifications, the final prediction is the class with the highest probability:

$$\hat{y} = \arg\max_i p_i \in p(\hat{y}|x, \mathcal{D})$$

2.4.1 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods are a group of methods for systematic random sampling from probability distributions [34]. MCMC methods can be applied to solve problems related to integration and optimization in high-dimensional spaces. This is very relevant for applications in statistics, but also areas such as physics, decision analysis, and machine learning [36].

MCMC methods use Markov Chains to perform Monte Carlo estimates. The general idea is to start by using Markov Chains to construct a sequence of random samples S. From the properties of Markov Chains, each sample S_i will only depend on the previous samples S_{i-1}:

$$P(S_{n+1} = s_{n+1}|S_n = s_n, ..., S_0 = s_0) = P(S_{n+1} = s_{n+1}|S_n = s_n)$$

In contrast to other methods for sampling, for instance inversion- or rejection-sampling, MCMC algorithms usually require the Markov Chain to have an initial "burn-in" period. A "burn-in" is a method of discarding the first n steps of an MCMC run, so that the Markov Chain can reach its equilibrium distribution. Although "burn-in" is not a theoretical component of MCMC, the use of "burn-ins" has become the norm because it limits the number of posterior samples needed.

The sequence S of random variables might have high autocorrelaion. Autocorrelation, in the context of MCMC methods, is a measure of how independent the different samples from the distribution are. Here, a low autocorrelation is good, because this indicates more independent results. However, in cases with high autocorrelation, one would need to sample and store a large amount of samples, Θ, and then subsample from these samples to get independent samples [1]. This is computationally expensive because the collection of samples Θ needs to be stored after training.

Despite MCMC methods' challenges in terms of scaleability and expensive storage, they are considered to be very relavant for sampling from posterior distributions in Bayesian Statistics [37]. Amongst the most popular MCMC methods, we find **The Metropolis-Hastings algorithm** [38]. This algorithm begins with an initial guess, or starting point, θ^0. The next step is to use a proposed distribution $Q(\theta^*|\theta)$ to sample a new candidate θ^* close to the preceding sample θ. The candidate sample θ^* is then compared to the current sample with regard to the target distribution. Here, the candidate is accepted if it is considered more likely than the current sample, based on the target distribution. In the case of it being less likely, the candidate will be accepted with given a probability α, else rejected. This process will, after repeating it a sufficient amount of times, give a sequence of samples that converge to the target distribution [1].

2.4.2 Variational Inference

Variational Inference (VI) [39] is an alternative approach to MCMC for solving intractable integrals in Bayesian Inference. Here, we define a variational posterior distribution $q_\phi(\theta)$, which is parameterized by parameters ϕ. The objective is to optimize the parameters ϕ in order to move the variational posterior distribution $q_\phi(\theta)$ close to the true posterior distribution $p(\theta|\mathcal{D})$. We can then do posterior inference using the variational posterior distribution $q_\phi(\theta)$.

The "closeness" between the true posterior distribution $p(\theta|\mathcal{D})$ and the variational poster distribution $q_\phi(\theta)$ is commonly measured using the Kullback-Leibler (KL) Divergence [40], which is defined as:

$$\begin{aligned}
D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})] &= \int q_\phi(\theta) \log \frac{q_\phi(\theta)}{p(\theta|\mathcal{D})} d\theta \\
&= \mathbb{E}_{q_\phi(\theta)}\left[\log \frac{q_\phi(\theta)}{p(\theta|\mathcal{D})}\right] \\
&= \mathbb{E}_{q_\phi(\theta)}[\log q_\phi(\theta)] - \mathbb{E}_{q_\phi(\theta)}[\log p(\theta|\mathcal{D})] \qquad (2.5)\\
&= \mathbb{E}_{q_\phi(\theta)}[\log q_\phi(\theta)] - \mathbb{E}_{q_\phi(\theta)}[\log p(\theta,\mathcal{D})] + \mathbb{E}_{q_\phi(\theta)}[\log p(\mathcal{D})] \\
&= \underbrace{\mathbb{E}_{q_\phi(\theta)}[\log q_\phi(\theta)] - \mathbb{E}_{q_\phi(\theta)}[\log p(\theta,\mathcal{D})]}_{-ELBO} + \log p(\mathcal{D})
\end{aligned}$$

However, since we are unable to compute the evidence $p(\mathcal{D})$ from the last line of Equation 2.5, we need to define a different objective to optimize that is equivalent to the KL-divergence up to a constant. We can then use the Evidence Lower Bound (ELBO) [41], which is defined as:

$$ELBO(q) = \mathbb{E}_{q_\phi(\theta)}[\log p(\theta,\mathcal{D})] - \mathbb{E}_{q_\phi(\theta)}[\log q_\phi(\theta)]$$

which accounts for the first two terms in Equation 2.5. Equation 2.5 can be rewritten as:

$$\log p(\mathcal{D}) = ELBO(q) + D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})]$$

KL-divergence is non-negative, hence we know that:

$$\log p(\mathcal{D}) \geq ELBO(q)$$

Since the log evidence $\log p(\mathcal{D})$ cannot be less than $ELBO$, maximizing the $ELBO$ will minimize the KL-divergence $D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})]$.

2.4.2.1 Bayes-By-Backprop

From Section 2.4.2 we saw that VI is a good tool for approximating Bayesian Inference. However, stochasticity stops backpropagation from functioning as it would in a regular ANN [42]. Several solutions have been proposed to solve this problem, such as **Bayes-By-Backprop (BBB)** [43] and probabilistic backpropagation [44]. This thesis will focus on BBB.

Algorithm 1 Bayes-by-Backdrop [43]

```
1: $\phi = \phi_0$
2: for $i = 0, ..., N$ do
3:     Draw $\epsilon$ from $q(\epsilon)$
4:     $\theta = t(\epsilon, \phi)$
5:     $f(\theta, \phi) = log(q_\phi(\theta) - log(p(D_y|D_x, \theta)p(\theta))$
6:     $\nabla_\phi f = backprop_\phi(f)$
7:     $\phi = \phi - \alpha\nabla_\phi f$
8: end for
```

From Algorithm 1, we sample a random variable ϵ from $q(\epsilon)$ in each iteration. Here, ϵ acts as a non-variational source of noise. Although ϵ is sampled in each iteration, it will still be treated as a constant towards the other variables. Then, θ is not sampled directly but rather obtained from deterministic transformation $t(\epsilon, \phi)$, where $\theta = t(\epsilon, \phi)$ follows $q_\phi(\theta)$. From Algorithm 1 we see that the rest of the transformations are non-stochastic, which enables backpropagation to work as normal with regards to the variational parameters ϕ.

2.4.3 Usecases of MCMC and VI

From Section 2.4.1 and 2.4.2 we see that MCMC and VI have their advantages and disadvantages. VI is less computationally expensice than MCMC methods. However, VI cannot guarantee samples that are (asymtotically) exact with regards to the target distribution [45], but only finding a distribution close to the target. In addition, VI is known to underestimate predictive variance [46].

Despite this, VI will be better suited for applications with large data sets and when one wants to explore many different models quickly. MCMC methods will on the other side be better suited for cases with smaller data sets and when one would gladly pay the price of higher computational costs for exact samples, for instance, in cases with small and expensive datasets.

2.5 Dimension Reduction

Dimensionality reduction is a technique used to reduce the number of variables in a dataset while keeping the most important information. Principal Component Analysis (PCA) [47] is amongst the most popular dimensionality techniques. PCA identifies the directions in the dataset that captures the most information and projects the data in these directions. Each projection is called a principal component and is a linear combination of the original variables. The first principal component captures the most variation, and the second principal component, which will be orthogonal to the first component, captures the second most variance, and so on.

PCA can be used to project high-dimensional data into lower dimensions, e.g. two, making it possible to visualize and compare high-dimensional data. One can create scree plots from PCA, which tells us how much variance each principal component captures.

Chapter 3

Methods

This chapter will introduce the proposed method for the detection of classification outliers in BNNs. The first section will shortly present the properties of the simulated data used to explain and visualize the proposed method. Secondly, the baseline method is presented. Then, the two proposed alternative methods are explained in detail. Lastly, an additional idea for separating OOD and In-Between (IB) outliers is briefly introduced.

Data simulation is done to facilitate the exploration of alternative approaches. This approach enables control over the data characteristics and properties. For example, one can manipulate the data's balance or adjust the variables' covariances.

Data used in initial experiments are simulated according to Algorithm 2. Here, the algorithm's input is a set of expectations μ, a covariance matrix Σ, and numbers of samples n. Matrix U is a $(n \times d)$-sized matrix with values sampled from a standard normal distribution, i.e. $N(0, 1)$. Here, n and d correspond to the number of samples and dimensions. Secondly, a Cholesky Decomposition [48] is performed on the input covariance matrix Σ, creating a decomposed matrix $\Sigma^{\frac{1}{2}}$. Matrix U is then multiplied with $\Sigma^{\frac{1}{2}}$ to create matrix W. Lastly, the expectations μ are added to their corresponding dimension in W.

Algorithm 2 Simulating data from d-dimensional distribution

```
1: $\mu$ = set of expectations
2: $\Sigma$ = covariance matrix
3: U = $(n \times d)$-sized matrix with values drawn from $N(0, 1)$
4: $\Sigma^{\frac{1}{2}}$ = CholeskyDecomposition($\Sigma$)
5: $W = U \times \Sigma^{\frac{1}{2}}$
6: for $i = 1, ..., d$ do
7:     $data[:, i] = W[:, i] + \mu[i]$
8: end for
```

The initial experiments, which are used to go through the baseline method in Section 3.1 and the proposed methods later in sections 3.2, 3.3 and 3.4, are performed on a simulated dataset shown in Figure 3.0.1. This data set consists of four classes with 200 samples in each class, i.e. a total of 800 samples.

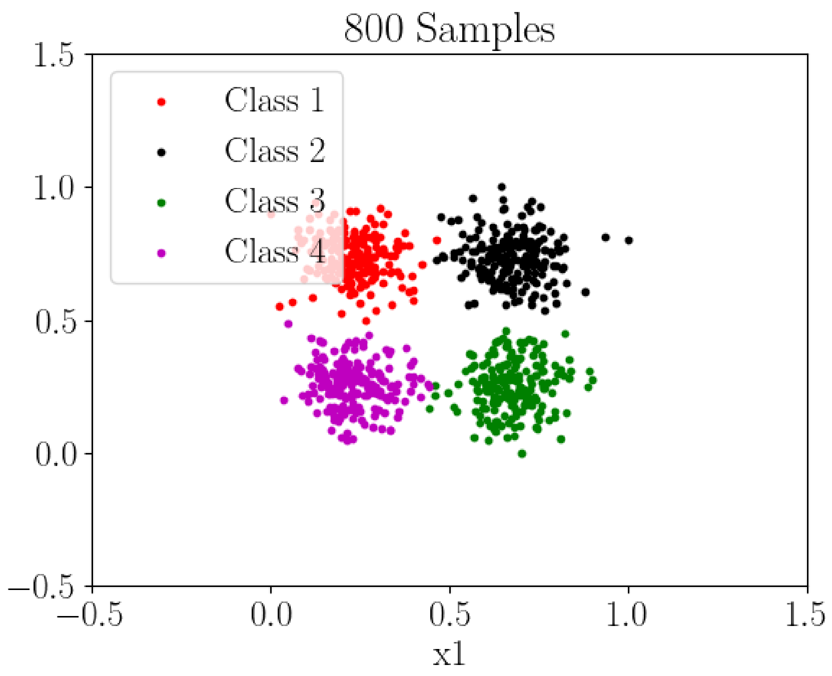

Figure 3.0.1: Simulated data consisting of four classes with 200 samples in each class, in total 800 samples.

3.1 Background

When measuring predictive uncertainty in a classification setting one can use different approaches, such as variation ratios [49], mutual information [50], and predictive entropy [50]. Due to the scope of this thesis, we will only focus on predictive entropy.

Predictive Entropy, denoted *ENT*, which has its foundations in information theory, captures the average amount of information in the predictive distribution:

$$\mathbb{H}[y|x,\mathcal{D}] = -\sum_{k=1}^{K} p(y_k|x,\mathcal{D}) \log p(y_k|x,\mathcal{D})$$

where K corresponds to all possible classes in the given model. The case of complete uncertainty, which corresponds to maximum entropy, is when each $\hat{y}_k = \frac{1}{K}$. The ideal case, i.e. minimum entropy, occurs when the output is 1 for one of the $\hat{y}_k$ and 0 for the rest. This indicates no uncertainty about the given prediction.

We approximate the predictive entropy by sampling the probability vectors, i.e. the softmax activations in the last layer, from T stochastic forward passes through the BNN. For each class in K we take the average from the T stochastic forward passes, $\frac{1}{T}\sum_t p(\hat{y} = k|x,\mathcal{D},\theta_t)$, where θ_t is the sampled network parameters in stochastic forward pass t, $\theta_t \sim q_\phi^*(\theta)$ [51].

In Figure 3.1.1, we see the predictive entropy in a grid of coordinates close to the training sample clusters. In this case, the predictive entropy is high in areas between class clusters. This is not that surprising because this is typically where a model would be uncertain. In a point in the middle of the four training class clusters, the prediction vector $\hat{y}$ would typically give high probabilities to all four classes, which would then yield a high entropy. However, we also see a possible flaw with using predictive entropy. In areas outside the training class clusters, in dark blue in Figure 3.1.1, we see no indication of uncertainty.

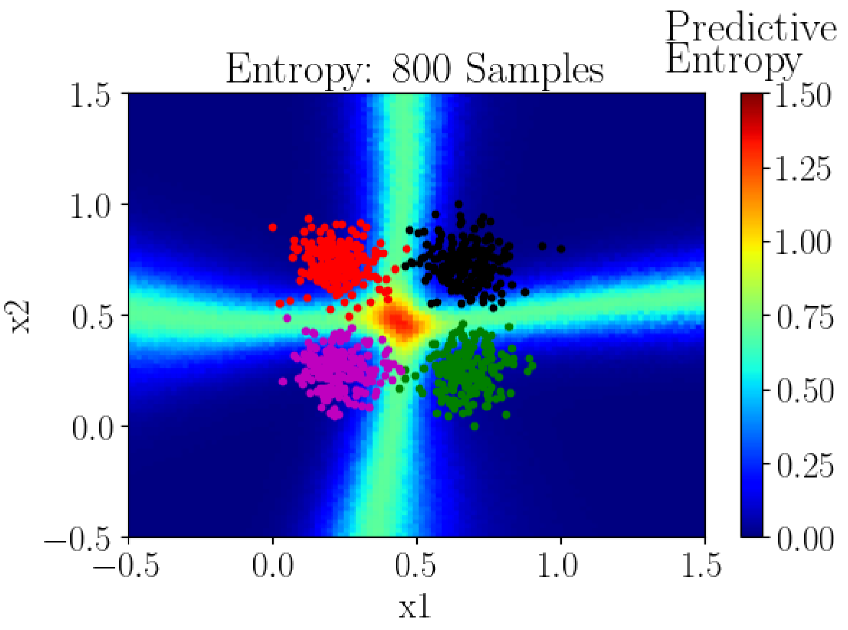

Figure 3.1.1: Predictive entropy in a grid of coordinates close to the training data clusters.

3.2 Detecting Classification Outliers by using the Pre-Activation Neuron Values in the Output Layer

The first proposed alternative method, denoted *PRE-ACT*, is based on using the pre-activation neuron values in the output layer of the model. Here, the idea, as briefly sugguested in [52], is that the output layer of the BNN has accumulated enough information to decide whether an input sample x is an outlier or not, by checking if the pre-activation neuron values is unusual. In a BNN with $n_l = L$ layers, the pre-activation neuron values in the last layer, u^L, are defined as:

$$u^L = W^L \times A^{L-1} + b^L$$

where W^L is the weight parameters in the last layer, A^{L-1} is the activations from the previous layer, and b^L is the bias term in the last layer. The number of units in u^L corresponds to the output layer size of the model, i.e. number of classes, n_y:

$$u_i^L : i = 1, ..., n_y$$

BNNs are probabilistic models, where a prediction is the result of T stochastic forward passes in the model, u^L will, therefore for a given data sample x be the average pre-activation neuron values of these T stochastic forward passes:

$$u^L = \frac{1}{T}\sum_{t=1}^{N} u_t^L$$

From the properties of the Softmax Activation Function, we can see that the indices of the highest value in u^L will correspond to the indices of the predicted class after the activation function in the output layer:

$$g(u_i^L) = \frac{e^{u_i^L}}{\sum_{k=1}^{n} e^{u_k^L}}$$

Firstly, we will therefore look at the highest pre-activation neuron value, i.e. $\max_1(u^L)$. Figure 3.2.1 shows how $\max_1(u^L)$ behaves in a grid of data points close to the training sample clusters. In the case of Figure 3.2.1 we see that the $\max_1(u^L)$ tends to be low in the middle of the class clusters, and higher as the data points move further away from the class clusters.

However, as we see from Figure 2.2.1, between classes, we get horizontal and vertical "spikes" where $\max_1(u^L)$ do not increase as much. This is to be expected because, in areas between two class clusters, one would expect the model to be uncertain about which class to choose, hence one would see the two highest values in u^L are close to being the same: $max_1(u^L) \approx max_2(u^L)$. It is, therefore, also interesting to look at the sum of the two highest values in u^L:

$$\sum_{k=1}^{2} \max_k u^L = \max_1(u^L) + \max_2(u^L)$$

Figure 3.2.2 shows that when using the sum of the two highest values in u^L, i.e. $\sum_{k=1}^{2} \max_k(u^L)$, we get smaller "spikes" and they shift 45 degrees. We can also see that $\sum_{k=1}^{2} \max_k(u^L)$ shifts to higher values closer to the class cluster than from using $\max_1(u^L)$.

As this is an illustrative example using a simple dataset with two variables, x_1 and x_2, and four classes, we will not present any other combinations of the pre-activation neuron values. It is, however, important to note that the idea is that in higher dimensional data with K classes, this method opens up for using not only $\sum_{k=1}^{2} \max_k(u^L)$, but also summing all the way up to $\sum_{k=1}^{K-1} \max_k(u^L)$. What is best will be situation dependent, where the dimension of the dataset d and the number of classes K are relevant factors.

Now that we know the values u^L for a given data sample x changes depending on how close it is to the training samples, we need to find a range of "approved" values for u^L. The intuitive approach is to look at the pre-activation neuron values from the training data X:

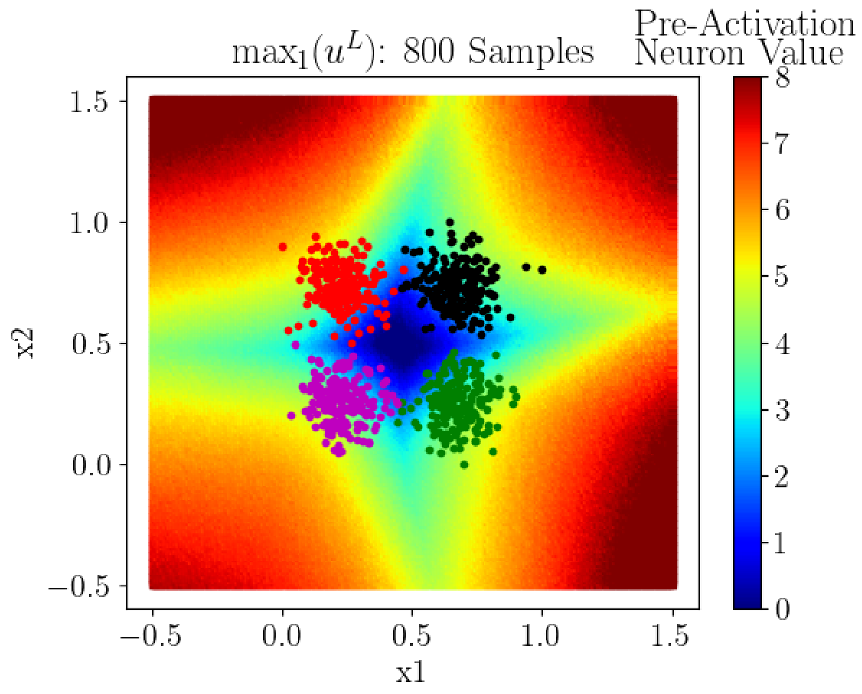

Figure 3.2.1: Highest Pre-Activation Neuron Value, $\max_1 u^L$, in a grid of coordinates close to the training data clusters.

$$\tilde{U}_j^L : j = 1, ..., n_X \tag{3.1}$$

where n_x corresponds to the number of samples in the training data X. $\tilde{U}^L$ serves as a distribution of "approved" values for u^L. The next step is to create two p-values, p_1 and p_2:

$$\begin{aligned} p_1 &= \frac{\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L > u^L)}{n_X} \\ p_2 &= \frac{\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L < u^L)}{n_X} \end{aligned} \tag{3.2}$$

where $\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L > u^L)$ is the number of samples in $\tilde{U}^L$ with a higher pre-activation neuron value than u^L and $\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L < u^L)$ is the number of samples in $\tilde{U}^L$ with a lower pre-activation neuron value than u^L. In the denominator, n_X is the total number of samples in $\tilde{U}^L$. We test both p_1 and p_2, because this enables to detect both too high and too low pre-activation neuron values, i.e. a two-tailed test. As p_1 and p_2 functions are p-values, we can set a significance level α. If p_1 or p_2 for a given data sample x is below the significance level α, it is labeled as an outlier. In Figure 3.2.3, a significance level α of 0.05 is used. Here, we see that areas marked in red are classified as outliers, and areas in dark blue are classified as inliers.

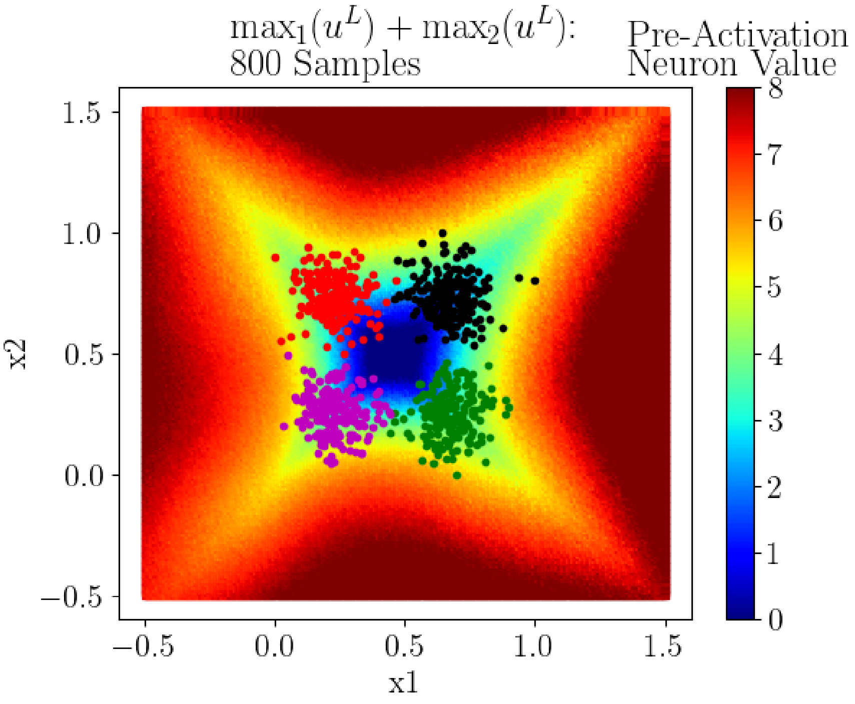

Figure 3.2.2: The sum of the two highest Pre-Actication Neuron Values, $\sum_{k=1}^{2} \max_k u^L$, in a grid of coordinates close to the training data clusters.

3.3 Combining Pre-Activation Neuron Values and Predictive Entropy

In addition to the baseline model in Section 3.1 and the proposed alternative method in Section 3.2, we will combine the two into one method. This method is denoted *PRE-ACT+ENT*. The motivation behind combining the two methods is that it might be able to extract the strengths of both methods. This method will then mark a data sample x as an outlier if either of these two methods labels it as an outlier. In Figure 3.3.1, we observe that combining the two methods also separates areas between two class clusters, which we did not see in Figure 3.2.3. Here, one would need to set a separate threshold for the predictive entropy and a significance level α for the method using pre-activation neuron values.

3.4 Using Pre-Activation Neuron Values to Separate Between Out-Of-Distribution and In-Between Outliers

In Section 3 we created two test statistics, p_1 and p_2. These p-values measure the probability of a pre-activation neuron value to be above and below the given pre-activation neuron value u^L, respectively. The idea is that these two p-values can indicate if a given outlier is either an IB outlier or an OOD outlier. The motivation behind this method is what we observed in Figure 3.2.1 and 3.2.2.

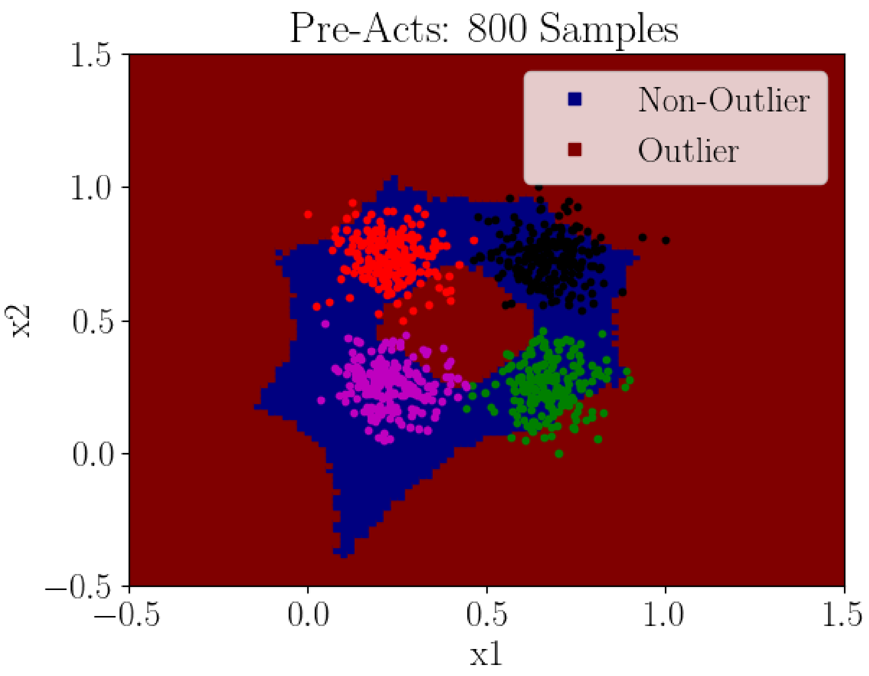

Figure 3.2.3: Calculating p-values on a grid of coordinates close to training data clusters and setting a significance level of 0.05. Coordinates with values below 0.05 are marked as outliers in red, and those above are marked as non-outliers in dark blue.

Here, we observed that the pre-activation neuron values between class clusters tend to be low and that pre-activation neuron values tend to be higher the further it is away from the training sample clusters. The assumption is then that if a data sample x is labeled as an outlier, we can classify it as an IB or OOD outlier by seeing which one of p_1 and p_2 triggered the data samples x to be labeled as an outlier. Here, p_1 would correspond to an OOD outlier, and p_2 to an IB outlier.

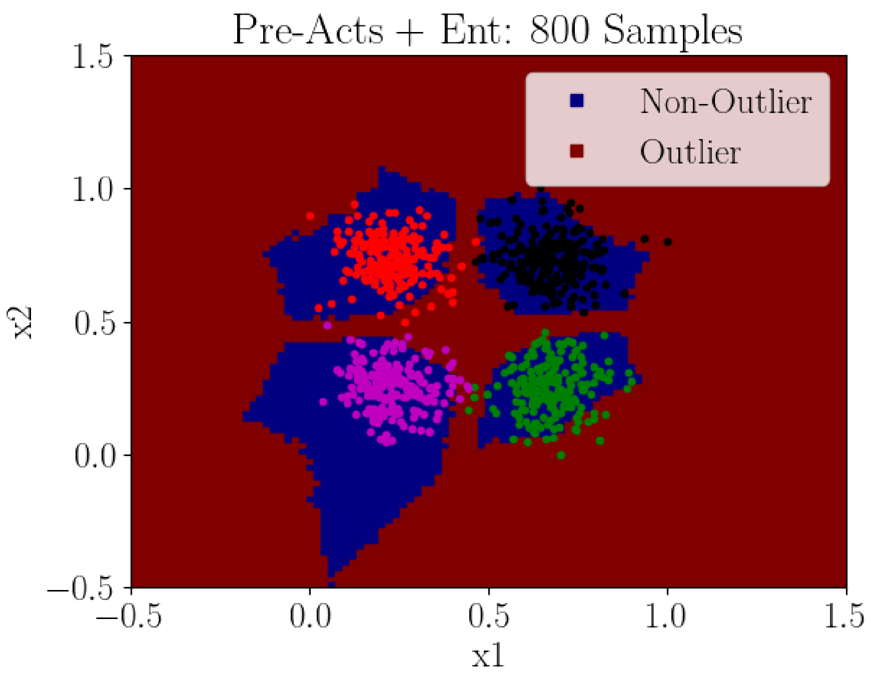

Figure 3.3.1: Combining classification outlier detection using both pre-activation neuron values and predictive entropy.

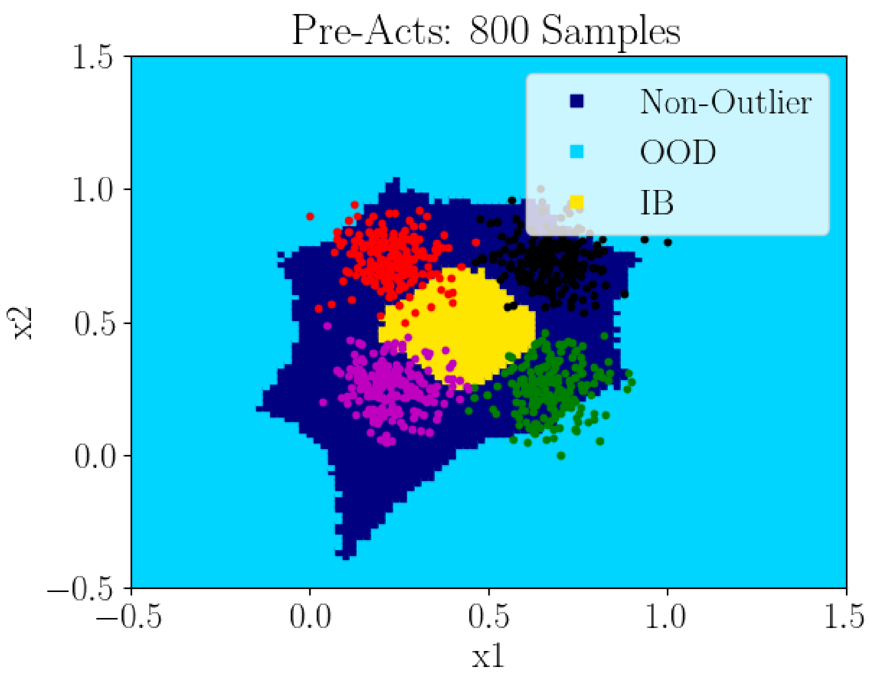

Figure 3.4.1: Using pre-activation neuron values to detect classification outliers and separate between In-Between outliers and Out-Of-Distribution outliers.

Chapter 4

Results

4.1 Background

This chapter will compare the three methods for outlier detection on three different datasets introduced in Section 3: *ENT*, *PRE-ACT*, and *PRE-ACT+ENT*. These three methods use the same BNN, but the method varies. The methods differ in the way they classify an outlier.

The three methods will be tested using a fully connected BNN: one input layer, one hidden layer, and one output layer. The hidden layer contains 100 neurons. For each data sample x, we do $T = 10$ stochastic forward passes. The model implementation follows the implementation from [53], based on [43].

The experiments will consider three performance metrics: False Discovery Rate (FDR), Power, and accuracy, as seen in Equation (4.1), where TP, TN, FP, FN are acronyms for True Positives, True Negatives, False Positives, and False Negatives, respectively:

$$\begin{aligned} Accuracy =& \frac{TP + TN}{TP + TN + FP + FN} \\ FDR =& \frac{FP}{TP + FP} \\ Power =& 1 - \beta = 1 - \frac{FN}{TP + FN} \end{aligned} \tag{4.1}$$

Accuracy is the number of correct predictions made by the given model in relation to the total number of predictions made. In this case, we will only measure the accuracy for those samples that are not labeled as outliers. FDR is a metric for measuring the type I errors, i.e. the number of false positives. Power measures the probability that the given test will reject the null hypothesis when the alternative hypothesis is true.

Code used to perform experiments, including the models, can be found on GitHub: https://github.com/hermanelling/master-thesis.git

4.2 Classification Outlier Detection on Simulated Data

Classification outlier detection experiments in this section are performed on the simulated data from Section 3. The model is trained on 800 training samples. To simulate outliers, additional samples are sampled outside and between the class clusters from the training data (Figure 4.2.1). The test data set consists of 160 non-outlier samples, i.e. drawn from the same distribution as the training data and 250 outlier samples.

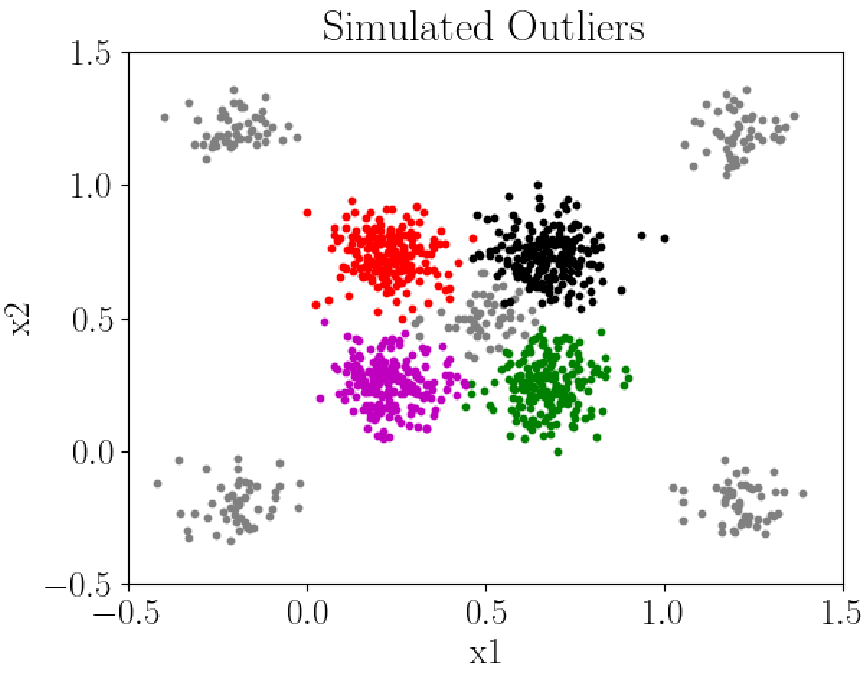

Figure 4.2.1: Figure shows where the outlier samples, in grey, are positioned compared to the class clusters from the training data, shown in colors. 50 samples in each cluster of outliers, in total 250 outlier samples.

Figure 4.2.2 shows the FDR, POWER, and accuracy scores for the three methods on the simulated test dataset from Figure 4.2.1. Here, we see that *PRE-ACT* and *PRE-ACT+ENT* give similar results, where *PRE-ACT+ENT* gives a slightly higher FDR and accuracy. Both give POWER scores above 0.9. The *ENT* method gives a lower FDR score than both *PRE-ACT* and *PRE-ACT+ENT*, but its POWER and accuracy score is significantly lower. This makes *ENT* the poorest method of the three when applied to the simulated data.

4.3 Classification Outlier Detection on Breast Cancer Data

The next classification outlier detection experiment uses the Breast Cancer Wisconsin (diagnostic) dataset [54] and transformed versions of this dataset. The

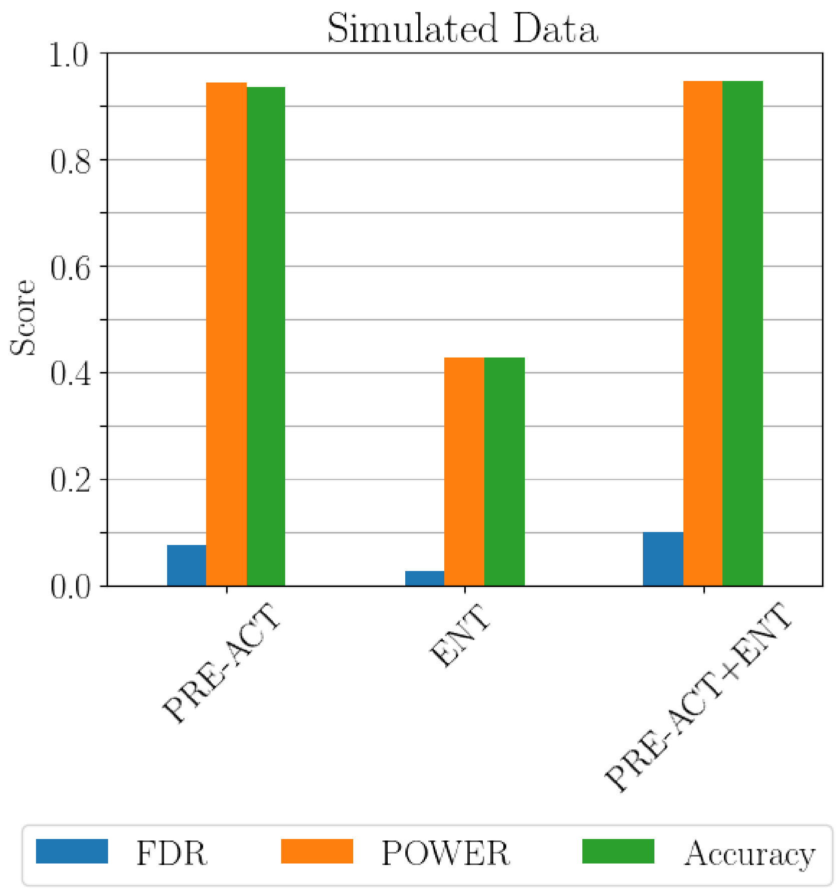

Figure 4.2.2: Barplot showing the FDR, POWER, and accuracy score for classification outlier detection for the simulated dataset. Accuracy is only calculated on samples that are not classified as an outlier.

Breast Cancer Wisconsin dataset is a binary classification dataset that contains 32 variables and 569 samples. In these experiments, 455 samples are used for training and 114 for testing. The dataset is scaled to values between 0 and 1 before training.

To simulate outliers, the test dataset is transformed in two ways. Firstly, by adding a constant, 1, to each variable. Secondly, by adding Gaussian Noise to the dataset. The Gaussian noise is created by adding a sample drawn from a standard normal distribution, $N(0, 1)$, to each variable. The test data consist of 50% samples from the original Breast Cancer Wisconsin data and 50% transformed data, in total 228 samples in each test datasets.

Figure 4.3.1 shows the FDR, POWER, and accuracy score on the test datasets. When noise is added to the data, we observe a low FDR on all three methods, with *PRE-ACT* having the lowest FDR score with 0.01, followed by *ENT* with 0.07 and *PRE-ACT+ENT* with 0.08. *PRE-ACT* also accounts for the highest POWER score with 0.60, which is quite low, but still higher than *ENT*'s 0.49, and *PRE-ACT+ENT*'s 0.59. We observe a similar ratio with the accuracy score, where *PRE-ACT* gives 0.57, *ENT* gives 0.48, and *PRE-ACT+ENT* gives 0.58. While non of the methods give especially good results, we see that both *PRE-ACT*

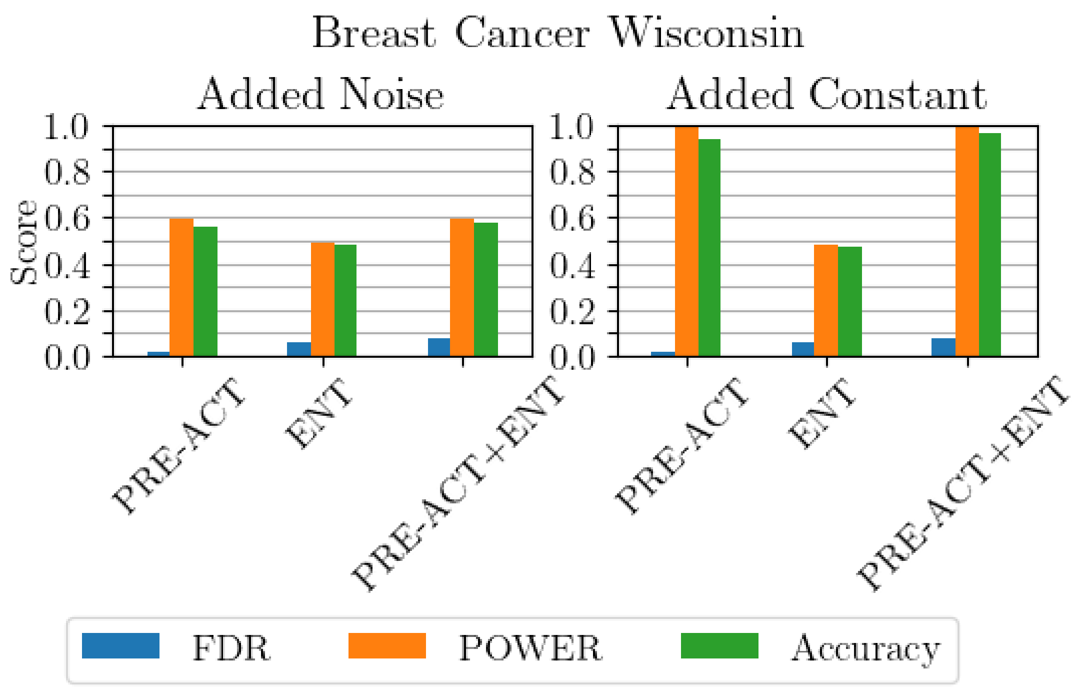

Figure 4.3.1: Barplot showing the FDR, POWER, and accuracy score for classification outlier detection for two transformed Breast Cancer Wisconsin datasets. Accuracy is only calculated on samples that are not classified as an outlier.

and *PRE-ACT+ENT* perform better than *ENT*

For the data where a constant is added, we observe a considerable difference between *PRE-ACT* and *ENT*. Here, *PRE-ACT* has the lowest FDR with 0.01 and the highest POWER with 1.0, while *ENT* has a FDR score of 0.07 and a POWER score of 0.49. Here, we observe that *PRE-ACT+ENT* performs similarly to *PRE-ACT*, having the same POWER score, and a slightly higher FDR and accuracy score.

Overall for both Breast Cancer Wisconsin test datasets, we see that both *PRE-ACT* and *PRE-ACT+ENT* outperform *ENT*, while the difference between *PRE-ACT* and *PRE-ACT+ENT* is small.

4.4 Classification Outlier Detection on MNIST and FMNIST

In this section, classification outlier detection experiments are done with MNIST [55], FMNIST [56], and transformed versions of MNIST. MNIST is a database of handwritten digits ranging from 0 to 9 (Figure 4.4.1). FMNIST, also referred to as Fashion-MNIST, is a database of fashion items, such as shoes, t-shirts, and coats (Figure 4.4.1). The MNIST and the FMNIST datasets contain $70,000$ images each, where $60,000$ is used for training and $10,000$ is used for validation. The format of both datasets is 28×28 grayscale images. These are well-known datasets often used for benchmarking in image classification tasks.

The model is trained on the MNIST dataset, so the FMNIST dataset will work as a dataset of outliers. In addition, the MNIST dataset will also be transformed in 5 ways: adding Gaussian blur, inverting images, adding constant (100) to all pixels, shuffling all pixels, and shuffling pixels in a section of the image (Figure 4.4.2). In the test datasets, 50% will be test data from the MNIST dataset, and the remaining 50% will be outlier samples. In total, $20,000$ samples are in each test dataset.

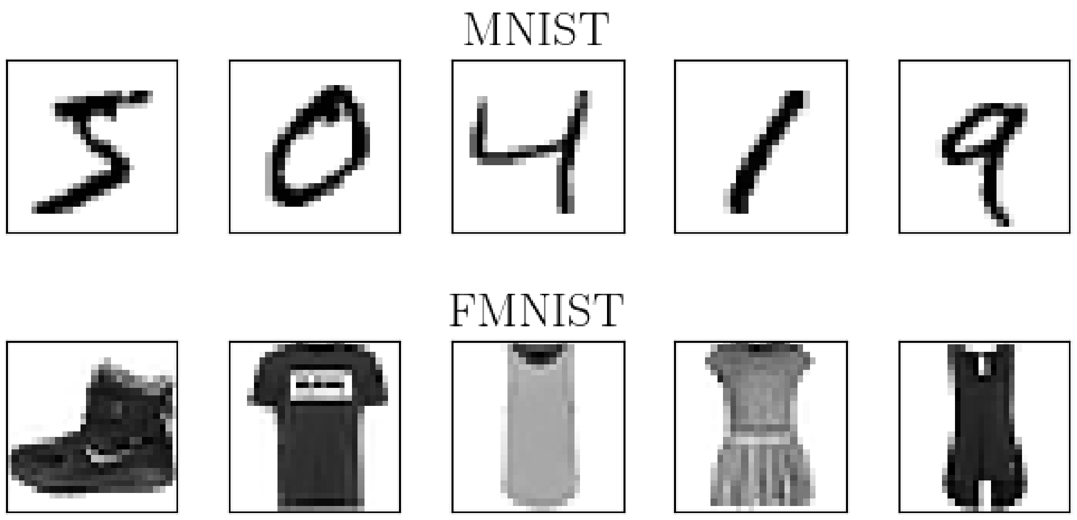

Figure 4.4.1: Examples from MNIST and FMNIST data set.

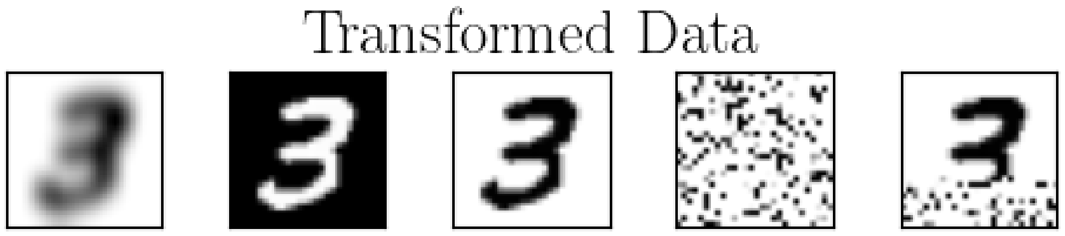

Figure 4.4.2: Examples from transformed MNIST dataset.

Figure 4.4.3 shows the FDR, POWER, and accuracy scores for the three methods on the six test datasets. With FMNIST, we see that *PRE-ACT* gives a relatively high FDR score and low POWER and accuracy score. *PRE-ACT* also gives similar results on blurred, inverted, shuffled, and partially shuffled images. *ENT* performs well on inverted and shuffled images, with POWER scores over 0.9, FDR scores below 0.08, and accuracy scores of 0.9. However, it does not perform as well on FMNIST, blurred images, and partially shuffled images. Overall, we see that *ENT* gives the lowest FDR score, highest POWER score, and highest accuracy score in five of the six datasets. The only exception is when a constant is added; here, we see that *PRE-ACT* gives a slightly higher POWER score but also a higher FDR score.

Common for all the six datasets is that the combination of *PRE-ACT* and *ENT*, i.e. *PRE-ACT+ENT*, gives a POWER and accuracy score similar to the best of the first two methods. We also observe that the FDR score tends to be higher than the worst of the two methods.

4.5 Dimension Reduction Results

Dimensionality reduction was performed on the test datasets used in Section 4.4 and 4.3. The dimension reduction was performed with Principal Component Analysis (PCA) [47], where two components were kept. This was done in order for us to be able to visualize and explore high-dimensional data. The results for the MNIST datasets and the Breast Cancer Wisconsin dataset can be found in Figures 4.5.1 and 4.5.2.

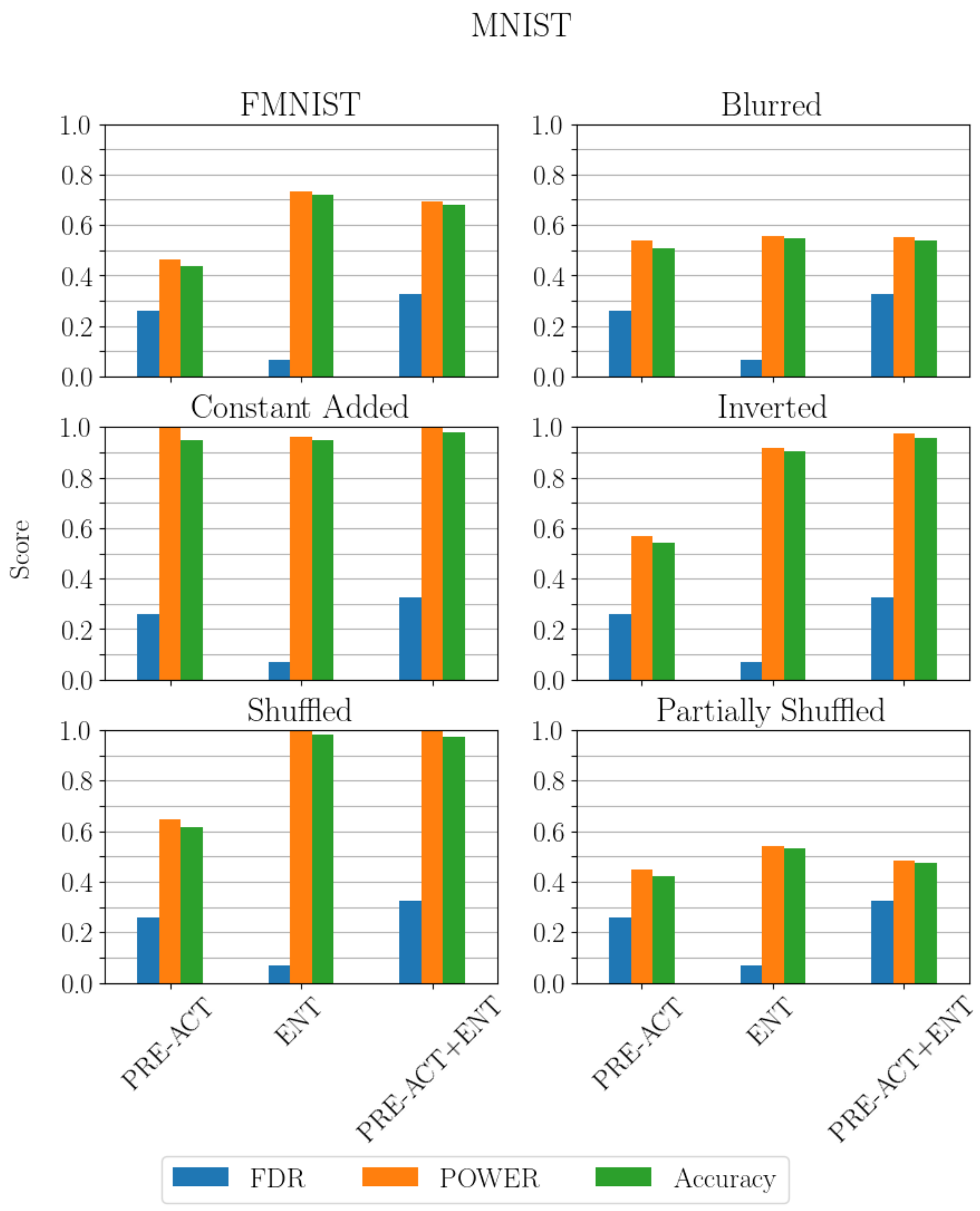

Figure 4.4.3: Barplot showing the FDR, POWER, and accuracy score for classification outlier detection for the MNIST, FMNIST, and transformed MNIST datasets. Accuracy is only calculated on samples that are not classified as an outlier.

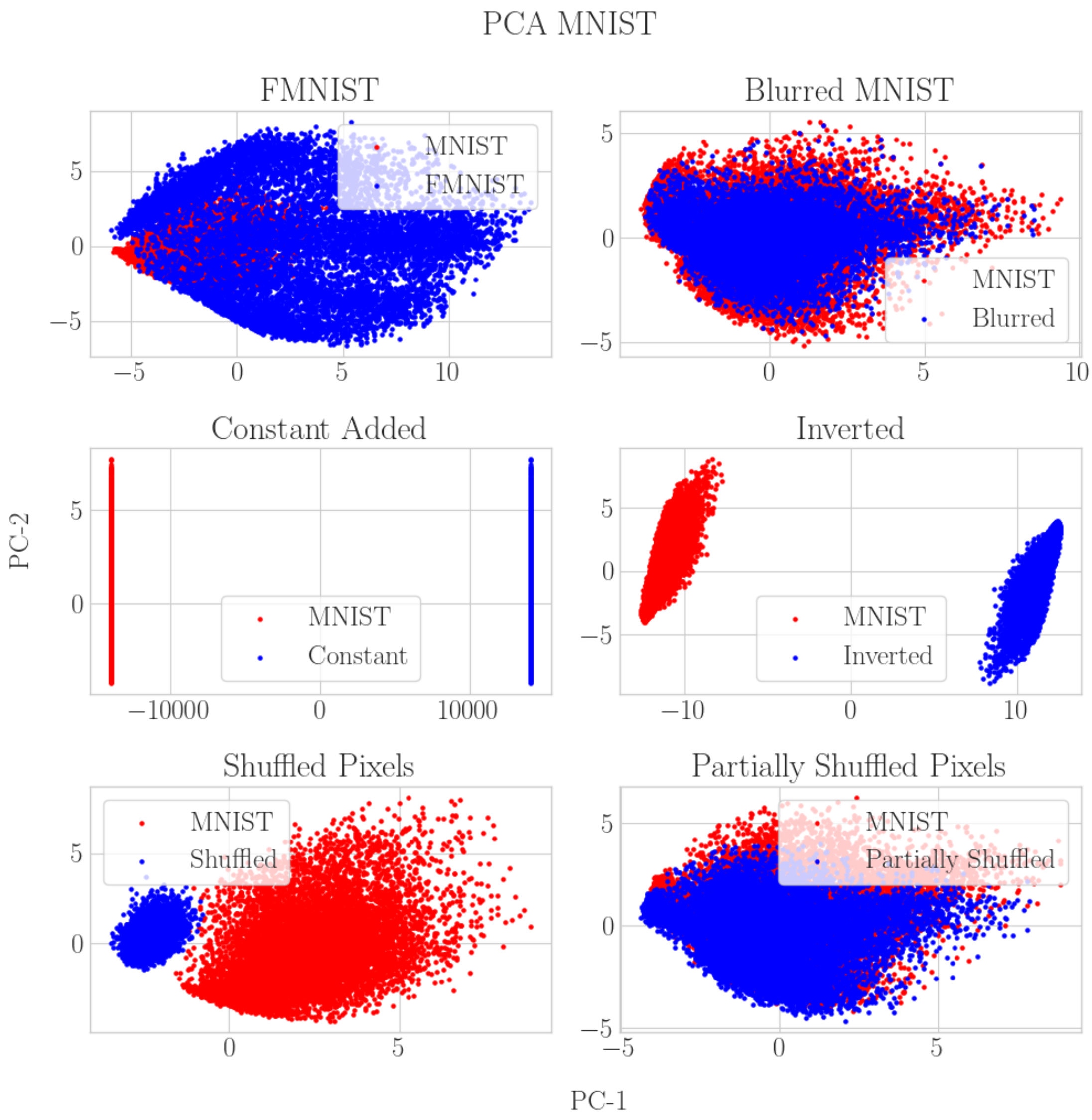

Figure 4.5.1: PCA for the MNIST test datasets.

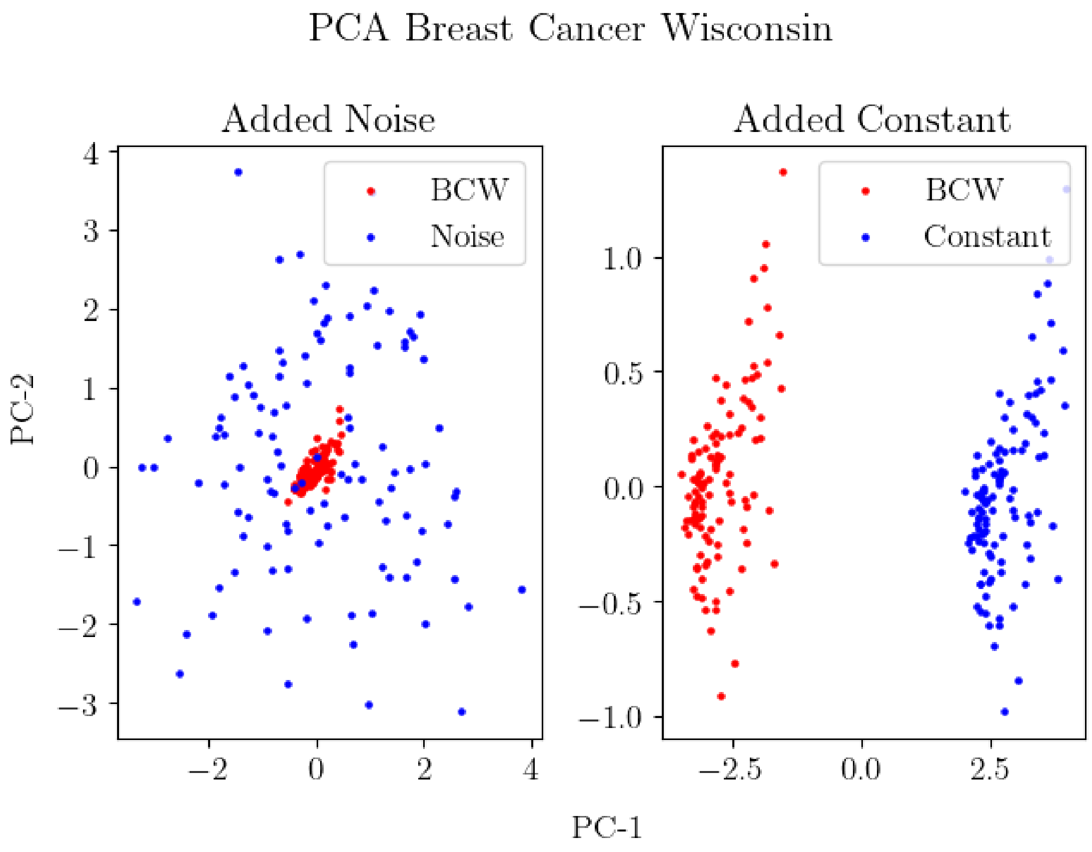

Figure 4.5.2: PCA for the Breast Cancer Wisconsin test datasets

Chapter 5

Discussion

This thesis has focused on detecting classification outliers in a Bayesian Neural Network by using the pre-activation neuron values in the last layer of the model, more specifically comparing the pre-activation neuron value in the last layer of a new sample x with the ones from the training data. The motivation behind this method was that it might outperform the established method, especially with Out-Of-Distribution data samples. The proposed methods were tested on three different datasets.

5.1 Key Findings

In Section 4.2, 4.4, and 4.3, we explored and empirically compared new ways of detecting classification outliers in Bayesian Neural Networks. We compared a total of three different methods, and two of these were new. The methods were tested on three different datasets: the simulated dataset, the MNIST dataset, and the Breast Cancer Wisconsin dataset.

Interestingly, we observed that *PRE-ACT* and *PRE-ACT+ENT* outperformed the baseline model, predictive entropy, with the simulated data and the Breast Cancer Wisconsin dataset. While with the MNIST dataset, we saw that the new methods did not outperform the baseline model, except for when a large constant is added to the pixels.

It is hard to point out a single reason for the poor result of the proposed methods on the MNIST dataset. From Figure 3.1.1 in Section 3.1, we saw that predictive entropy worked well In-Between the class clusters in the simulated data. However, we also saw that samples in areas outside the class clusters, i.e. Out-Of-Distribution, did not give high entropy scores. One idea is that with the images in MNIST, which is 784-dimensional, the outlier samples are in In-Between areas. In Figure 4.5.1 in Section 4.5 we see when the MNIST and FMNIST datasets are reduced to two dimensions and compared, the datasets have areas of overlap. Here, about 34% of the variance is explained (computations not shown). We expected to see a clearer separation between these two.

Another thing to consider is the number of classes K in the dataset. The simulated data have four classes and the BCW dataset have two classes, while MNIST have ten classes. The idea here is that with datasets with few classes it is easier to get Out-Of-Distribution samples, while with many classes we get more samples in In-Between areas. This can be the reason why using *ENT* works well with the MNIST dataset.

5.2 Limitations of our work

Due to time limitations, several things were not explored. This section will quickly mention them. Firstly, the method proposed in Section 3.4 for separating OOD and IB outliers have not been explored and validated further on test datasets. While it looked like it was working on the simulated data, this will need to be studied further before we can say anything about its performance.

Furthermore, the combination used in the proposed *PRE-ACT* methods, i.e. choosing which k to use for $\sum_{k=1}^{K-1} \max_k(u^L)$ and significance level α is based on empirical results in visualizations. Here, we considered the best combination as the one that gave the best trade-off FDR, POWER, and accuracy score. Ideally, we would have liked to use some optimization here. For instance, Neyman-Pearson optimization [57] could have been applied to minimize the FDR and maximize the POWER.

Lastly, the datasets used in this thesis. Section 4 indicates that the results might depend on the type of data used. This is especially relevant for the simulated data. Simulations are not precise, and results might depend on the simulation settings, such as the number of classes K, dimensions d, and samples n.

5.3 Further Research

Finding other ways to use the pre-activation neuron values in a Bayesian Neural Network might be interesting. For instance, instead of just looking at the last layer, one could see the result when looking at the pre-activation neuron values of the entire network. Another path is to not look at the highest or the sum of the K-highest pre-activation neuron values but rather the structure of the "set" of the pre-activation neuron values.

Chapter 6

Acknowledgements

Firstly, I would like to express my special gratitude to Prof. Solve Sæbø, Associate Prof. Aliaksandr Hubin, and Research Scientist Filippo Remonato for guidance and help during the work of this master's thesis. I would also like to thank Sintef Digital in Oslo for your input and helpful advice. Lastly, I would like to thank my fellow students, especially Olle Gilje Gunnarshaug and Thomas Yordy, in the Data Science program at NMBU for motivation and (more or less) productive discussions throughout the process of writing this thesis.

Bibliography

[1] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun, "Hands-on bayesian neural networks—a tutorial for deep learning users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[2] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021.

[3] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol, "Ai in health and medicine," *Nature medicine*, vol. 28, no. 1, pp. 31–38, 2022.

[4] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2020.

[5] Longbing Cao, "Ai in finance: A review," *Available at SSRN 3647625*, 2020.

[6] Nilkanth Mukund Deshpande, Shilpa Gite, and Rajanikanth Aluvalu, "A review of microscopic analysis of blood cells for disease detection with ai perspective," *PeerJ Computer Science*, vol. 7, pp. e460, 2021.

[7] Steven E Dilsizian and Eliot L Siegel, "Artificial intelligence in medicine and cardiac imaging: harnessing big data and advanced computing to provide personalized medical diagnosis and treatment," *Current cardiology reports*, vol. 16, pp. 1–8, 2014.

[8] Alejandro Rodriguez-Ruiz, Kristina Lång, Albert Gubern-Merida, Mireille Broeders, Gisella Gennaro, Paola Clauser, Thomas H Helbich, Margarita Chevalier, Tao Tan, Thomas Mertelmeier, et al., "Stand-alone artificial intelligence for breast cancer detection in mammography: comparison with 101 radiologists," *JNCI: Journal of the National Cancer Institute*, vol. 111, no. 9, pp. 916–922, 2019.

[9] Vijil Chenthamarakshan, Payel Das, Samuel Hoffman, Hendrik Strobelt, Inkit Padhi, Kar Wai Lim, Benjamin Hoover, Matteo Manica, Jannis Born, Teodoro Laino, et al., "Cogmol: Target-specific and selective drug design for covid-19 using deep generative models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4320–4332, 2020.

[10] Stefan Harrer, Pratik Shah, Bhavna Antony, and Jianying Hu, "Artificial intelligence for clinical trial design," *Trends in pharmacological sciences*, vol. 40, no. 8, pp. 577–591, 2019.

[11] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al., "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[12] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[13] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al., "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[14] Rasheed Hussain and Sherali Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2018.

[15] Russel Hotten, "Carmakers face challenge from google and apple," *BBC*.

[16] Shaoshan Liu, Jie Tang, Zhe Zhang, and Jean-Luc Gaudiot, "Caad: Computer architecture for autonomous driving," *arXiv preprint arXiv:1702.01894*, 2017.

[17] National Highway Traffic Safety Administration U.S. Department of Transportation, "Tesla crash preliminary evaluation report, nhtsa. pe 16-007. technical report," 2017.

[18] Takuma Okuda and Sanae Shoda, "Ai-based chatbot service for financial industry," *Fujitsu Scientific and Technical Journal*, vol. 54, no. 2, pp. 4–8, 2018.

[19] Najmeddine Dhieb, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud, "A secure ai-driven architecture for automated insurance systems: Fraud detection and risk measurement," *IEEE Access*, vol. 8, pp. 58546–58558, 2020.

[20] Kristian Bondo Hansen, "The virtue of simplicity: On machine learning models in algorithmic trading," *Big Data & Society*, vol. 7, no. 1, pp. 2053951720926558, 2020.

[21] Eyke Hüllermeier and Willem Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, pp. 457–506, 2021.

[22] D Michael Titterington, "Bayesian methods for neural networks and related models," *Statistical science*, pp. 128–139, 2004.

[23] Zhi-Hua Zhou, *Ensemble methods: foundations and algorithms*, CRC press, 2012.

[24] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[25] Jun Han and Claudio Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995 Proceedings 3*. Springer, 1995, pp. 195–201.

[26] Kunihiko Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969.

[27] Sepp Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[28] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] Sebastian Raschka, *Python machine learning*, Packt publishing ltd, 2015.

[30] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin, *Bayesian data analysis*, CRC press, 2013.

[31] David Kaplan, *Bayesian statistics for the social sciences*, Guilford Publications, 2014.

[32] Bruno De Finetti, *Theory of probability: A critical introductory treatment*, vol. 6, John Wiley & Sons, 2017.

[33] Thomas Bayes, "Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s," *Philosophical transactions of the Royal Society of London*, , no. 53, pp. 370–418, 1763.

[34] Charles J Geyer, "Practical markov chain monte carlo," *Statistical science*, pp. 473–483, 1992.

[35] David M Blei, Alp Kucukelbir, and Jon D McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[36] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan, "An introduction to mcmc for machine learning," *Machine learning*, vol. 50, pp. 5–43, 2003.

[37] Rémi Bardenet, Arnaud Doucet, and Chris Holmes, "On markov chain monte carlo methods for tall data," *Journal of Machine Learning Research*, vol. 18, no. 47, 2017.

[38] Siddhartha Chib and Edward Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[39] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, pp. 183–233, 1999.

[40] Solomon Kullback and Richard A Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[41] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[42] Wray L Buntine, "Operations for learning with graphical models," *Journal of artificial intelligence research*, vol. 2, pp. 159–225, 1994.

[43] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning.* PMLR, 2015, pp. 1613–1622.

[44] José Miguel Hernández-Lobato and Ryan Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International conference on machine learning.* PMLR, 2015, pp. 1861–1869.

[45] Christian P Robert, George Casella, and George Casella, *Monte Carlo statistical methods*, vol. 2, Springer, 1999.

[46] RE Turner and M Sahani, "Two problems with variational expectation maximisation for 3614 time-series models," *Bayesian Time series*, vol. 3615, pp. 109–130.

[47] Michael E Tipping and Christopher M Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[48] Commandant Benoit, "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrésa un systeme d'équations linéaires en nombre inférieura celui des inconnues. application de la méthodea la résolution d'un systeme défini d'équations linéaires (procédé du commandant cholesky)," *Bulletin géodésique*, vol. 2, no. 1, pp. 67–77, 1924.

[49] Elmer H Johnson, "Freeman: elementary applied statistics: for students in behavioral science (book review)," *Social Forces*, vol. 44, no. 3, pp. 455, 1966.

[50] Claude E Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[51] Yarin Gal et al., "Uncertainty in deep learning," 2016.

[52] Lars Skaaret-Lund, Geir Storvik, and Aliaksandr Hubin, "Sparsifying bayesian neural networks with latent binary variables and normalizing flows," 2023.

[53] Shishir Sharma, "bayes-by-backprop," `https://github.com/nitarshan/bayes-by-backprop`, 2020.

[54] Dheeru Dua and Casey Graff, "UCI machine learning repository," 2017.

[55] Li Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

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

[57] Jerzy Neyman and Egon Sharpe Pearson, "Ix. on the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, no. 694-706, pp. 289–337, 1933.

Article

OUTLIER DETECTION IN BAYESIAN NEURAL NETWORKS

Herman Ellingsen

OUTLIER DETECTION IN BAYESIAN NEURAL NETWORKS

Herman Ellingsen

BIAS, NMBU Norwegian University of Life Sciences, Norway
Email: herman.ellingsen@nmbu.no

ABSTRACT

Describing uncertainty is one of the major issues in modern deep learning. Artificial Intelligence models could be used with greater confidence by having solid methods for identifying and quantifying uncertainty. This article proposes two alternative methods for classification outlier detection in Bayesian Neural Networks. This is done by looking for unusual pre-activation neuron values in the last layer of a Bayesian Neural Network.

The proposed methods are compared to a baseline method for outlier detection, Predictive Entropy, on three datasets: a simulated dataset, the MNIST dataset, and the Breast Cancer Wisconsin dataset. In addition, we introduce an idea for separating In-Between and Out-Of-Distribution outliers.

The results indicate that the proposed methods' performance depends on the dataset type. In the case of a simple simulated dataset, we see that the proposed methods outperform the baseline method. The proposed method also outperforms the baseline method on the Breast Cancer Wisconsin dataset. However, when tested on the MNIST image dataset, we observed that the baseline is better than the proposed method. Common for all three datasets is that a combination of the two methods gives approximately as well POWER score as the best of the two, but also a higher FDR score than the best of the two.

Index Terms— Classification Outlier Detection, Uncertainty, Bayesian Inference, Bayesian Neural Networks,

1. INTRODUCTION

Due to their ability to solve complicated problems, Deep Learning Models have, over the last decade, expanded into an increasingly large amount of fields, such as medicine [1], autonomous vehicles [2], and finance [3]. However, neural networks are generally prone to overfitting, which affects their generalizability [4]. Furthermore, deep learning models' inability to say "I don't know" in cases where it should be uncertain is problematic. Generally, one will, therefore, not know when a deep learning model makes a sensible choice or not. There have been proposed several approaches to solve this problem, such as Probabilistic Modelling [5], Ensemble Methods [6] and Dropout [6]. Amongst the most promising is the type of probabilistic modeling using the Bayesian approach. This is because Bayesian Neural Networks offer a natural way to describe and quantify a model's uncertainty.

Outlier detection with Bayesian Neural Networks has already been explored quite a bit. Here, the approach has been to use the predictive entropy of the predictive distributions to measure uncertainty, which has worked well with datasets like MNIST and FMNIST [7]. However, as indicated in [8], this approach might not work well with samples drawn from Out-Of-Distribution samples. This is illustrated in Figure 1, where this method demonstrates high uncertainty in In-Between areas but indicates no uncertainty in dark blue corner areas.

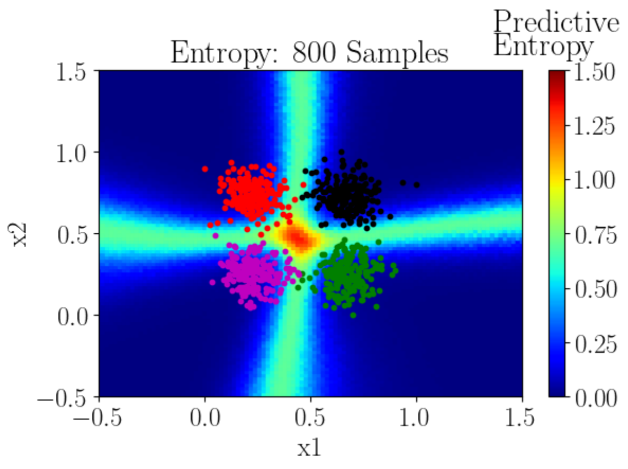

Fig. 1: Predictive entropy in a grid of coordinates with simulated binormal data consisting of four classes with 200 samples in each class.

In this paper, we propose alternative methods for outlier detection in Bayesian Neural Networks. Unlike predictive entropy methods, these methods aim to detect both In-Between and Out-Of-Distribution outliers. This paper is organized as follows: Section **2** will present the most relevant theoretical aspects, including an introduction to Bayesian Neural Networks, which include Artificial Neural Networks, Bayesian Inference, and Variational Inference. In Section **3**, we detail the proposed methods used in this paper. The empirical results are presented in Section **4**. Finally, Section **5** will discuss the key findings, limitations, and suggestions for further work.

2. THEORY

2.1. Uncertainty in Deep Learning

Uncertainty in deep learning is generally divided into two types: aleatoric and epistemic uncertainty. Epistemic uncertainty is the type of uncertainty related to the model. This can be caused by several things, such as a lack of data, poor model specifications, and/or having a too simple model. Epistemic uncertainty is reducible, which means that, in the case of lacking data, one would decrease the epistemic uncertainty by adding more data. Aleatoric uncertainty is on the other hand caused by inherent randomness in the training data. Data samples will never perfectly represent the real world because they will always contain some randomness and/or noise. Adding more data will not reduce the aleatoric uncertainty because more data will also introduce more randomness and/or noise. We, therefore, say that aleatoric uncertainty is irreducible. Predictive uncertainty, the confidence we have in a prediction, results from both epistemic and aleatoric uncertainty.

2.2. Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by how the human brain's biological neurons work. ANNs are traditionally structured with an input layer l^0, $L-1$ hidden layers l^l, and an output layer l^L [5],

$$\begin{aligned} l^0 &= x \\ l^l &= g^l(W^l \times l^{l-1} + b^l), l = 1, ..., L-1 \\ l^L &= y \end{aligned}$$

where each layer l, except the input layer, represents a linear transformation, which is followed by a non-linear transformation $g(x)$, i.e. an *activation function*. The objective is to learn the model parameters $\theta = (W, b)$ from the training data $\mathcal{D}$. Here, $\mathcal{D}$ is a set of input samples x and class labels y. This is usually done with first-order optimization [9], using the back-propagation algorithm for approximating a minimal cost point estimate for the model parameters $\hat{\theta}$. In regular ANNs, each parameter gets a fixed value (Figure 2A).

ANNs offer a good tool for solving complicated problems. Still, they have been shown to be overconfident and unreliable when met with test samples drawn from a different distribution than from its training samples [10].

2.3. Bayesian Neural Networks

Bayesian Neural Networks (BNNs) [5, 11] introduce uncertainty by treating its model parameters θ as random variables. These random variables have their own distribution, which is marginalized to form a predictive distribution (Figure 2B). We can incorporate prior information by defining a prior distribution over the model parameters, $p(\theta)$. When observing new data, one can use Bayes' rule to update the prior belief and get the posterior distribution of the parameters [12],

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta} \tag{1}$$

where $p(\mathcal{D}|\theta)$ is referred to as the likelihood of the data and $p(\mathcal{D})$ is the marginal distribution of the data.

In high dimensions, the integral in Equation (1) is intractable. It is therefore not possible to compute exact or sample directly from the posterior distribution $p(\theta|\mathcal{D})$. However, there are established methods for dealing with this problem, such as Markov Chain Monte Carlo methods [13] and Variational Inference [14].

There are several benefits with BNNs, such as their natural ability to distinguish between epistemic and aleatoric uncertainty with respectively $p(\theta|\mathcal{D})$ and $p(y|x, \theta)$ [15]. This gives BNNs the ability to learn from small datasets without overfitting [16]. Secondly, BNNs provide better calibration than regular ANNs [17, 18], which mean that they give more consistent uncertainty and are less likely to be overconfident.

2.3.1. Variational Inference

Variational Inference (VI) [14] is an alternative approach to MCMC methods for solving intractable integrals in Bayesian Inference. The main idea is to use approximation rather than sampling. In contrast to MCMC methods, VI methods scale well with more network parameters. This makes VI suitable for applications in deep learning.

The idea behind VI is to introduce a variational distribution, $q_\phi(\theta)$, parameterized by parameters ϕ. The objective is to optimize the parameters ϕ to move the variational distribution $q_\phi(\theta)$ close to the true posterior distribution $p(\theta|\mathcal{D})$. The "closeness" between the true posterior distribution and the approximate distribution $q_\phi(\theta)$ is measured using **Kullback-Leibler divergence (KL-divergence)** [19]:

$$D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})] = \int q_\phi(\theta) \log \frac{q_\phi(\theta)}{p(\theta|\mathcal{D})} d\theta \tag{2}$$

From Equation 2 we see that in order to solve $D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})]$, one will need to compute $p(\theta|\mathcal{D})$, which was the problem from the start. A way to tackle this problem is to use the *evidence lower bound* (ELBO) [20],

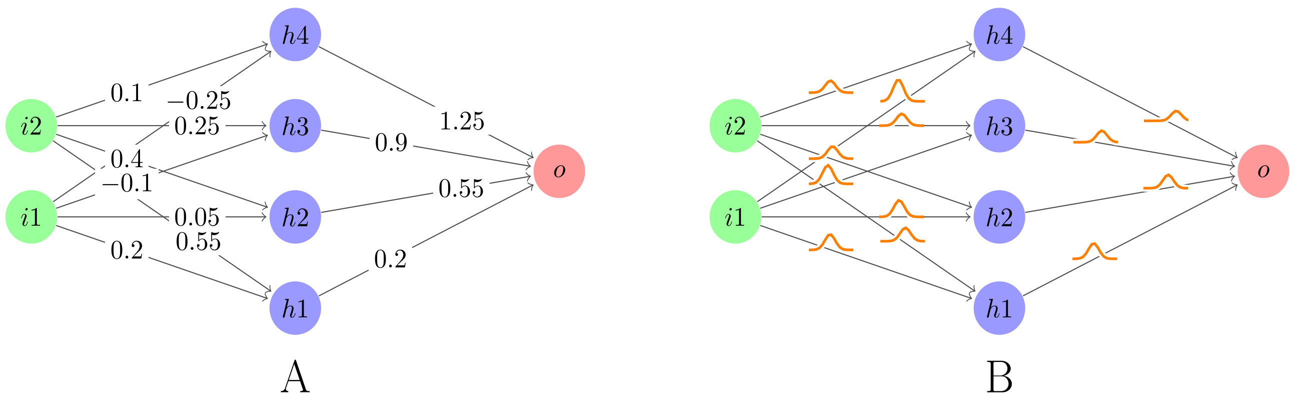

Fig. 2: Both **Figure A** and **B** are neural networks with an input layer containing two features, a hidden layer with four neurons, and an output layer with a single neuron. **Figure A** illustrates a regular ANN, i.e. fixed parameter values. **Figure B** illustrates a neural network with stochastic weight parameters.

$$ELBO(q_\phi) = \mathbb{E}_{q_\phi(\theta)}[\log p(\theta, \mathcal{D})] - \mathbb{E}_{q_\phi(\theta)}[\log q_\phi(\theta)]$$

which will serve as a loss. This can be rewritten into:

$$\log p(\mathcal{D}) = ELBO(q_\phi) + D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})]$$

And since the KL-divergence is non-negative, we can say that:

$$\log p(\mathcal{D}) \geq ELBO(q_\phi)$$

Furthermore, we know that the log evidence $\log p(\mathcal{D})$ cannot be less than $ELBO$, maximizing the $ELBO$ will therefore, minimize the KL-divergence $D_{KL}[q_\phi(\theta) \parallel p(\theta|\mathcal{D})]$.

2.3.2. Bayes By Backprop

While VI offers a good theoretical tool for approximating intractable integrals, it still needs a practical implementation for usage in deep learning models. The problem is stochasticity prevents backpropagation from working as in regular ANNs [21]. Bayes By Backprop (BBB) [22], which is a practical implementation of stochastic variational inference [23] in combination with a reparametrization trick [24], offers a backpropagation-compatible algorithm.

As seen in Algorithm 1, this method samples a random variable ϵ from $q(\epsilon)$ in each iteration. Here, ϵ acts as a non-variational source of noise. Even though ϵ is sampled, it is treated as a constant towards the rest of the variables. We then see that θ is obtained from a deterministic transformation $t(\epsilon, \phi)$, where $\theta = t(\epsilon, \phi)$ follows $q_\phi(\theta)$. As for the rest of the transformations in Algorithm 1 we see that they are non-stochastic, which enables backpropagation to function as normal towards the variational parameters ϕ.

Algorithm 1 Bayes-by-Backdrop [22]

```
1: $\phi = \phi_0$
2: for $i = 0, ..., N$ do
3:     Draw $\epsilon$ from $q(\epsilon)$
4:     $\theta = t(\epsilon, \phi)$
5:     $f(\theta, \phi) = log(q_\phi(\theta) - log(p(D_y|D_x, \theta)p(\theta))$
6:     $\nabla_\phi f = backprop_\phi(f)$
7:     $\phi = \phi - \alpha \nabla_\phi f$
8: end for
```

3. PROPOSED METHOD

This section presents the methods used in this paper, which includes the baseline method and two proposed alternative methods. Lastly, we introduce an idea for separating In-Between and Out-Of-Distribution outliers. All examples in this section use the simulated data in Figure 3, which consist of four classes of 200 samples each, in total 800 samples, drawn from a bivariate normal distribution.

3.1. Background

The standard approach, denoted *ENT*, for quantifying predictive uncertainty in BNNs is using the predictive entropy of the prediction vector $\hat{y}$ [25]:

$$\mathbb{H}[y|x, \mathcal{D}] = -\sum_{k=1}^{K} p(y_k|x, \mathcal{D}) \log p(y_k|x, \mathcal{D})$$

From the properties of the Softmax activation function [26], we know that each of the K elements in y is a real number in the range $[0, 1]$, which sums to 1. Maximum uncertainty, i.e. maximum entropy, is achieved when each $\hat{y}_k$ in $\hat{y}$ is uniformly

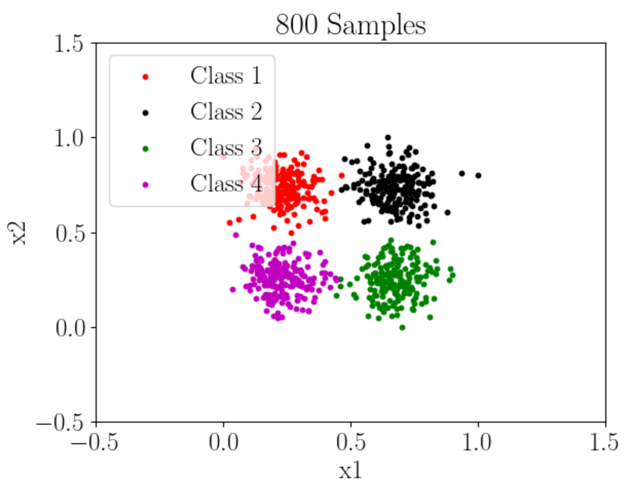

Fig. 3: Simulated data consisting of four classes with 200 samples in each class, in total 800 samples.

distributed. Minimum uncertainty is achieved when one of the given $\hat{y}_k = 1$ and the rest is 0.

As BNNs are probabilistic models where a prediction is the result of T stochastic forward passes, the predictive entropy is calculated from the average of these T stochastic forward passes:

$$p(y_k|x,\mathcal{D}) = \frac{1}{T}\sum_{t=1}^{T} p(y=k|x,\mathcal{D},\theta_t)$$

for each of the K classes, where θ_t are the sampled network parameters in stochastic forward pass t from the variational posterior distribution, $\theta_t \sim q^*_\phi(\theta)$.

In Figure 1 we visualize the predictive entropy in a grid of coordinates in and around the training samples. Here, we observe a high predictive entropy between the training samples in In-Between areas, shown in red, yellow, and turquoise colors. However, in Out-Of-Distribution areas outside the class clusters, we see low predictive entropy, shown in dark blue.

3.2. Outlier Detection with Pre-Activation Neuron Values

This section presents the first proposed alternative method, denoted *PRE-ACT*, for classification outlier detection in BNNs. This method uses the pre-activation neuron values in the last layer to indicate whether a data sample x is an outlier. The pre-activation neuron values in the last layer of a model with L layers are defined as:

$$u^L = W^L \times A^{L-1} + b^L$$

where W^L are the weight parameters of the last layer, A^{L-1} are the activations from the previous layer and b^L is the bias term of the last layer. Here, u^L is a vector of n_y values, where n_y corresponds to the output layer size, i.e. the number of classes in the model:

$$u^L_i : i = 1, ..., n_y$$

Since BNNs are probabilistic models, u^L will here be defined as the average from T stochastic forward passes in the model:

$$u^L = \frac{1}{T}\sum_{t=1}^{T} u^L_t$$

From the properties of the Softmax function [26], we know that the highest value in the pre-activation neuron values u^L, $\max_1 u^L$, will correspond to the highest probability after activation:

$$g(u^L_i) = \frac{e^{u^L_i}}{\sum_{i=1}^{n_y} e^{u^L_k}}$$

where g is the Softmax function. As the highest probability after Softmax activation corresponds to the predicted class, we will therefore initially look at the highest pre-activation neuron values in the last layer, $\max_1(u^L)$. Figure 4 shows how $\max_1(u^L)$ changes depending on its distance from the training samples. In this case, we can see that $\max_1(u^L)$ is lower in the middle of the four class clusters and it increases as it moves further away from the training samples.

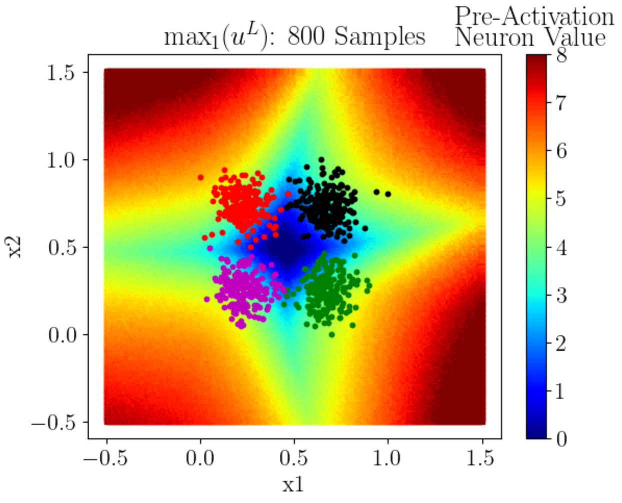

Fig. 4: Highest Pre-Activation Neuron Value, $\max_1 u^L$, in a grid of coordinates close to the training data clusters.

However, as we observe from Figure 4, we get ”spikes” of low pre-activation values between the class clusters, shown in green and yellow. This is not that surprising, as one would expect the model to be uncertain about which class to choose between two class clusters. Therefore, for the data samples in the "spikes" with low pre-activaion values we observe that the two highest values are close to being the same: $max_1(u^L) \approx max_2(u^L)$. This makes it interesting to also look at the sum of the two highest values in u^L:

$$\sum_{k=1}^{2} \max_k u^L = \max_1(u^L) + \max_2(u^L)$$

Figure 5 shows how the sum of the two highest pre-activation neuron values in the last layer, i.e. $\sum_{k=1}^{2} \max_k u^L$, changes depending on its distance to the training samples. Here, we observe that we get smaller "spikes" of low pre-activation neuron values and that they shift 45 degrees.

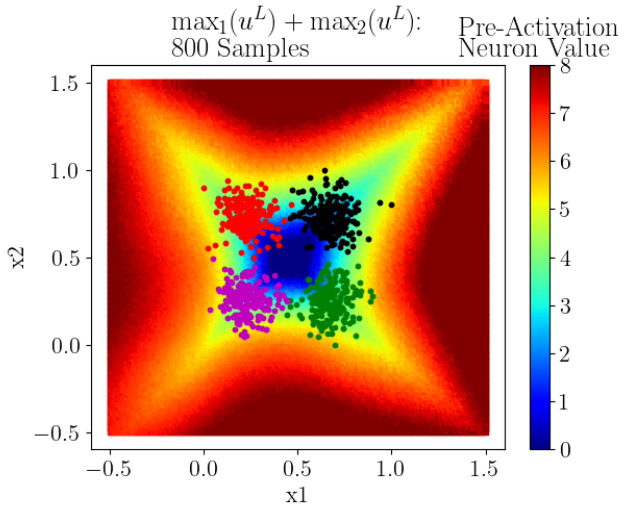

Fig. 5: The sum of the two highest Pre-Actication Neuron Values, $\sum_{k=1}^{2} \max_k u^L$, in a grid of coordinates close to the training data clusters.

Examples shown in Figure 4 and 5 are illustrative examples from a simple simulated dataset. It is, however, important to note that the idea is that in data with K classes, this method does not necessarily only use $\sum_{k=1}^{2} \max_k u^L$, but also sums up to $K-1$: $\sum_{k=1}^{K-1} \max_k(u^L)$. What is best in the given situation will depend on the data, where dimensions d and the number of classes K can be relevant factors.

From Figures 4 and 5 we saw that u^L changes depending on how close it is to the training samples. The next step is to find a range of approved values for u^L. Here, it makes sense to use the pre-activation neuron values from the training data because they represent the range of values one would expect from data samples drawn from the same distribution as the training data:

$$\tilde{U}_j^L : j = 1, ..., n_X$$

here, n_X corresponds to the number of samples in the training data X. We then create two p-values, p_1 and p_2:

$$p_1 = \frac{\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L > u^L)}{n_X}$$
$$p_2 = \frac{\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L < u^L)}{n_X}$$

where $\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L > u^L)$ is the number of samples in $\tilde{U}^L$ with a higher pre-activation neuron value than u^L and $\sum_{j=1,...,n_X} \mathrm{I}(\tilde{U}_j^L < u^L)$ is the number of samples in $\tilde{U}^L$ with a lower pre-activation neuron value than u^L. In the denominator, n_X is the total number of samples in $\tilde{U}^L$. The idea is then to test new data samples based on these p-values, p_1 and p_2, where if the pre-activation neuron values for a given data sample x give a p-value, either p_1 or p_2, below a given significance level α, it is labeled as an outlier. This is illustrated in Figure 6 with a significance $\alpha = 0.05$. Here, the areas marked in red are labeled as outliers, and those in dark blue are labeled as non-outliers.

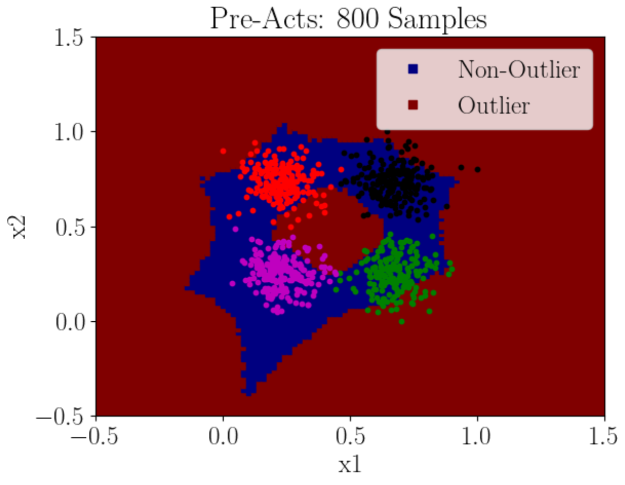

Fig. 6: Calculating p-values, p_1 and p_2, setting a significance level $\alpha = 0.05$. Samples, where p_1 or p_2 are below 0.05, are marked as outliers in red, and those above are marked as non-outliers in dark blue.

3.3. Combinding Predictive Entropy and Pre-Activation Neuron Values for Outlier Detection

The second proposed method, denoted *PRE-ACT+ENT*, combines the baseline model in Section 3.1 and the proposed alternative method in Section 3.2. The motivation is that this might extract the strengths of both methods into one. Here, one would mark data sample x as an outlier if the baseline model or the proposed alternative model labels x as an outlier. This method requires a separate threshold for the predictive entropy and a significance level α for the proposed alternative method. Figure 7 shows how this method combines both methods. Different from Figure 6 in Section 3.2 is that this method also marks areas between two class clusters as outliers, not only areas between all four class clusters.

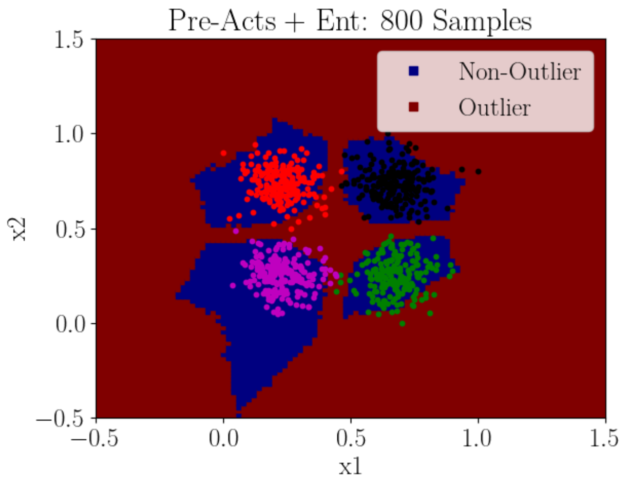

Fig. 7: Combining classification outlier detection using predictive entropy and pre-activation neuron values. The threshold for predictive entropy is 0.6, and the significance level α used for the pre-activation neuron values is 0.05.

3.4. Separating between In-Between and Out-Of-Distribution Outliers

From Section 3.2 we have two p-values: p_1 and p_2, which measure the probability of a pre-activation neuron value to be above and below the given pre-activation neuron value u^L, respectively. The idea is that p_1 and p_2 can indicate if a sample, already labeled as an outlier, is an In-Between or Out-Of-Distribution outlier. The motivation for this idea comes from what we observed in figures 4 and 5, where u^L tends to be low between the class clusters and high outside the class clusters. The assumption is then that we can classify the outlier as an In-Between outlier or Out-Of-Distribution outlier by seeing which one of p_1 and p_2 triggered the data sample x to be labeled as an outlier.

In Figure 8 we applied this method to the simulated data. Here, the method labels areas between the four class clusters as In-Between outliers and areas outside the class clusters as Out-Of-Distribution outliers.

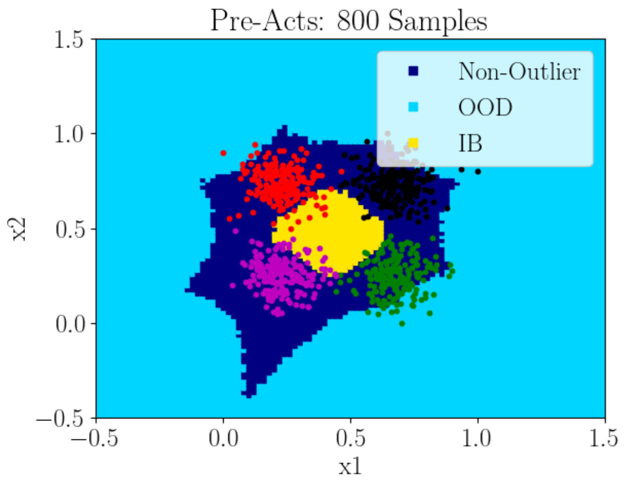

Fig. 8: Using pre-activation neuron values to detect classification outliers and separate between In-Between outliers and Out-Of-Distribution outliers.

4. EXPERIMENTAL RESULTS

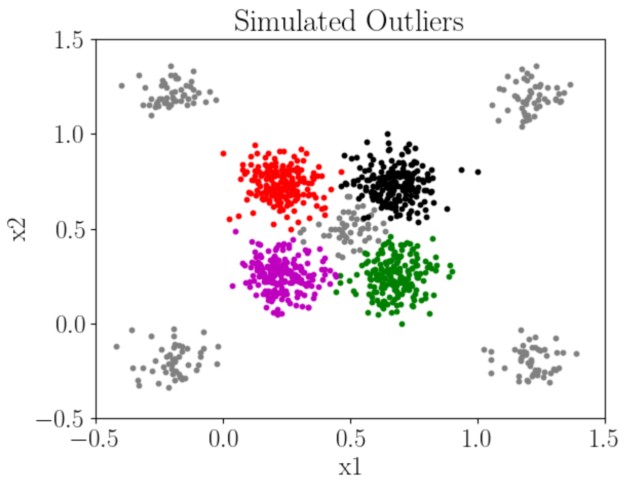

Fig. 9: Figure shows where the outlier samples, in grey, are positioned compared to the class clusters from the training data, shown in colors. 50 samples in each cluster of outliers, in total 250 outlier samples.

This section will compare three different methods for classification outlier detection on three different datasets. All three methods use the same BNN but vary in how they detect classification outliers. The model used in the experiments uses the BBB method [27] and follows the implementation by [28]. The model is a shallow BNN with one input layer, one hidden layer, and one output layer. The hidden layer consists of 100 neurons.

The experiments will be evaluated using three performance metrics: False Discovery Rate (FDR), POWER, and accuracy. These are defined in Equation (3), where TP, TN, FP, FN are acronyms for True Positives, True Negatives, False Positives, and False Negatives, respectively:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ FDR &= \frac{FP}{TP + FP} \\ Power &= 1 - \beta = 1 - \frac{FN}{TP + FN} \end{aligned} \tag{3}$$

The code implementation can be found on GitHub: `https://github.com/hermanelling/master-thesis.git`

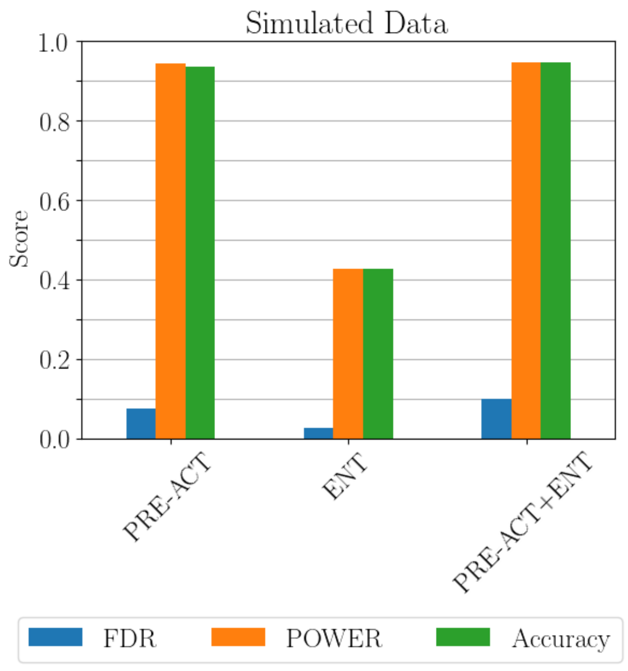

Fig. 10: Barplot showing the FDR, POWER, and accuracy score for classification outlier detection for the simulated dataset. Accuracy is only calculated on samples that are not classified as an outlier.

4.1. Classification Outlier Detection on Simulated Data

In this section, classification outlier detection experiments are performed on the simulated data presented in Figure 3 in Section 3. Here, the model is trained on 800 samples. In order to simulate outliers, we have sampled outlier samples in areas between all four class clusters and outside the class clusters, as seen in Figure 9. The test data consists of 160 samples drawn from the same distribution as the training data, i.e. non-outliers, and 250 outlier samples.

In Figure 10 we see the FDR, POWER, and accuracy score for the three methods on the simulated test dataset. We observe that *PRE-ACT* and *PRE-ACT+ENT* give similar results, where *PRE-ACT+ENT* gives slightly higher FDR, but also a higher accuracy score. Both *PRE-ACT* and *PRE-ACT+ENT* give POWER scores above 0.9. *ENT* gives a considerably lower POWER and accuracy score with 0.42 on both metrics, but a lower FDR than both *PRE-ACT* and *PRE-ACT+ENT*.

4.2. Classification Outlier Detection on Breast Cancer Wisconsin Dataset

In the next classification outlier experiment, we test the Breast Cancer (diagnostic) Wisconsin (BCW) dataset [29]. The BCW dataset is a binary classification dataset with 32 variables and 569 samples. In this experiment, the model is trained on 455 samples, and 114 samples are set aside for testing. This dataset was scaled to values between 0 and 1 before training. To simulate outliers, we have transformed the test dataset in two ways. Firstly by adding Gaussian Noise from a standard normal distribution to each variable. Secondly, a constant, 1, is added to each variable in the test

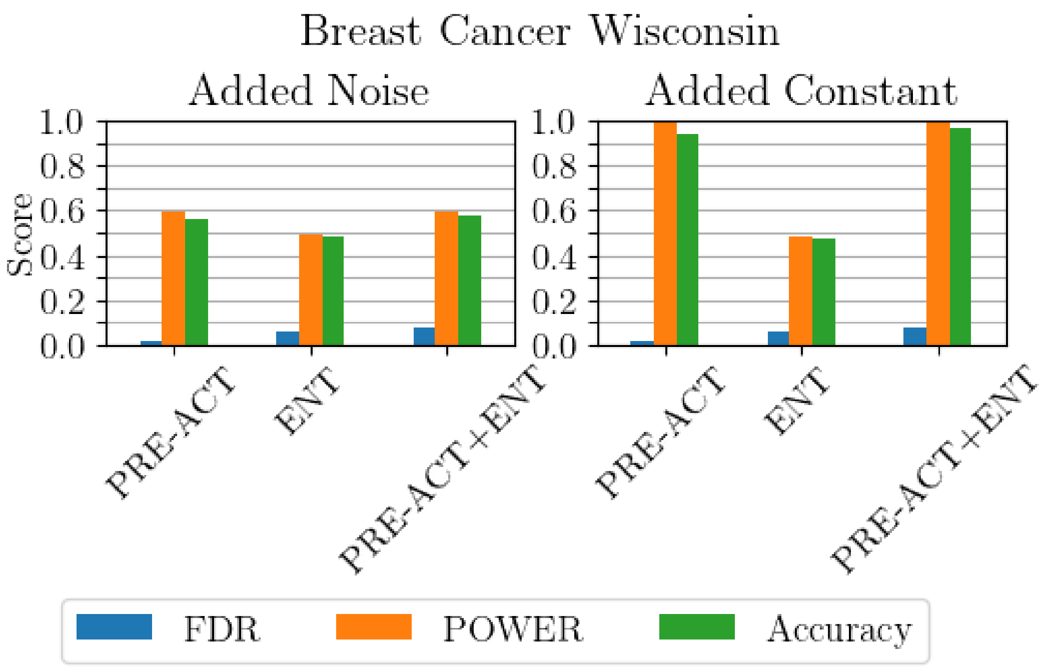

Fig. 11: Barplot showing the FDR, POWER, and accuracy score for classification outlier detection for two transformed Breast Cancer Wisconsin datasets. Accuracy is only calculated on samples that are not classified as an outlier.

dataset. Our two datasets will then consist of 50% samples from the original BCW test data and 50% transformed data.

Figure 11 shows the FDR, POWER, and accuracy for the two Breast Cancer Wisconsin test datasets. When noise is added we observe a low FDR on all three methods, where *PRE-ACT* has the lowest with an FDR of 0.01, followed by *ENT* with 0.07 and *PRE-ACT+ENT* with 0.09. *PRE-ACT* gives the highest POWER with 0.60, which is quite low, followed by *PRE-ACT+ENT* with 0.50. Common for all three methods is that the accuracy score is slightly below the POWER score.

For the test dataset where a constant is added, we observe a considerable difference between the baseline method *ENT* and the two proposed alternative methods *PRE-ACT* and *PRE-ACT+ENT*. Here, *PRE-ACT* gives an FDR, POWER, and accuracy of 0.01, 1.0, and 0.94, respectively. Followed by *PRE-ACT+ENT* with 0.09, 1.0, and 0.98. Similar to *PRE-ACT+ENT*, *ENT* gives an FDR score of 0.07 but a considerably lower POWER and accuracy score of 0.49 and 0.48.

4.3. Classification Outlier Detection on MNIST

In the next experiment, the model is trained on the MNIST dataset [30], which is a dataset of handwritten digits in the range $0-9$ of size 28×28 (Figure 12). It consists of 60,000 training samples and 10,000 test samples. For testing, we will also use FMNIST and transformed versions of MNIST. FMNIST is an image dataset of fashion items from the Zalando fashion catalog [31] (Figure 12). Similar to MNIST, it consists of 60,000 training samples and 10,000 test samples in size 28×28. The transformed MNIST datasets are transformed in the following ways: blurred, inverted, constant added to pixels, shuffled pixels, and partially shuffled pixels, as seen in Figure 13. For each outlier dataset, i.e. FMNIST and the transformed versions of MNIST, we create a test dataset consisting of 50% samples from MNIST and 50% outliers samples, in a total of six datasets with 20,000 samples in each.

Figure 14 shows the FDR, POWER, and accuracy scores for the three methods on the six different test datasets. With FMNIST, we see that *PRE-ACT* gives a relatively high FDR score, low POWER score, and lower accuracy compared to *ENT*. *PRE-ACT* also gives similar results on blurred, inverted, shuffled, and partially shuffled images. *ENT* performs well on inverted and shuffled images, with POWER scores over 0.90, FDR scores below 0.08, and accuracy scores over 0.90. *ENT* does not perform equally well on FMNIST, blurred images, and partially shuffled images. Still, we see that *ENT* performs best overall, with the lowest FDR, highest POWER score, and highest accuracy score in five out of six test datasets. The only exception is when a constant is added, here we see that *PRE-ACT* gives a slightly higher POWER score but also a higher FDR score.

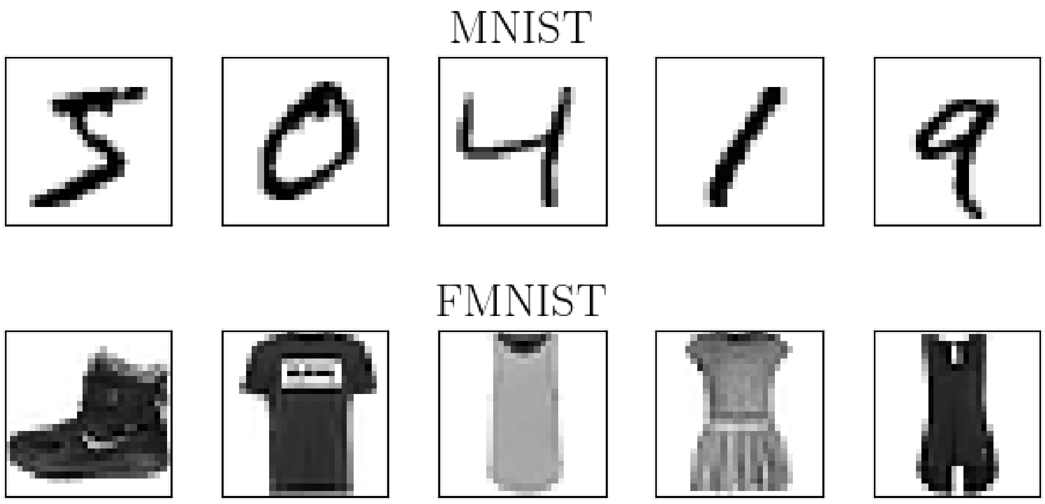

Fig. 12: Examples from MNIST and FMNIST data set.

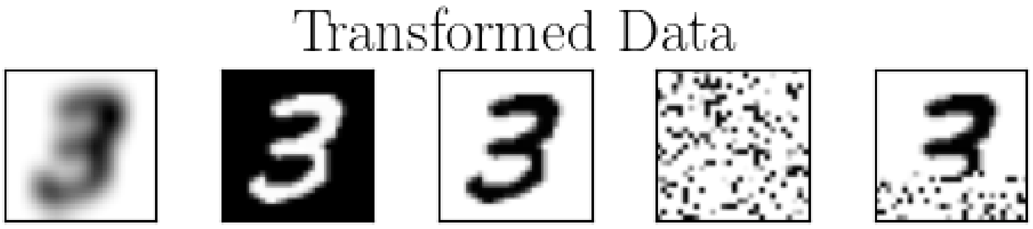

Fig. 13: Examples from transformed MNIST dataset.

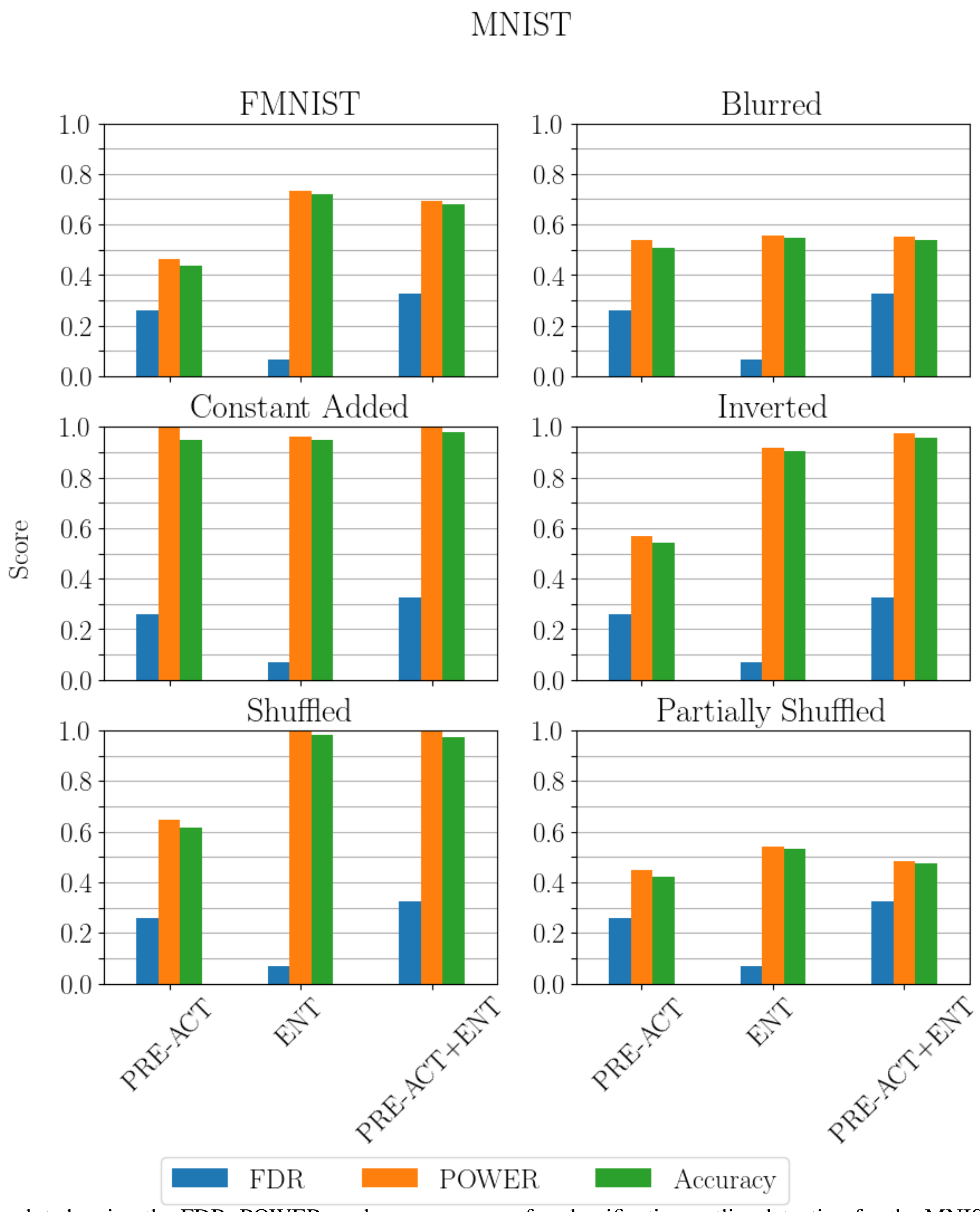

Fig. 14: Barplot showing the FDR, POWER, and accuracy score for classification outlier detection for the MNIST, FMNIST, and transformed MNIST datasets. Accuracy is only calculated on samples that are not classified as an outlier.

Common for all six test datasets is the *PRE-ACT+ENT* gives POWER and accuracy scores similar to the best of *PRE-ACT* and *ENT*, but also gives the highest FDR out of the two.

4.4. Dimension Reduction Results

Principal Component Analysis (PCA) [32] was performed on the MNIST test datasets and the BCW test datasets. This was done because PCA can transform high-dimensional data down to low dimensions, which enables us to visualize and explore high-dimensional data.

Results are found in figures 15 and 16.

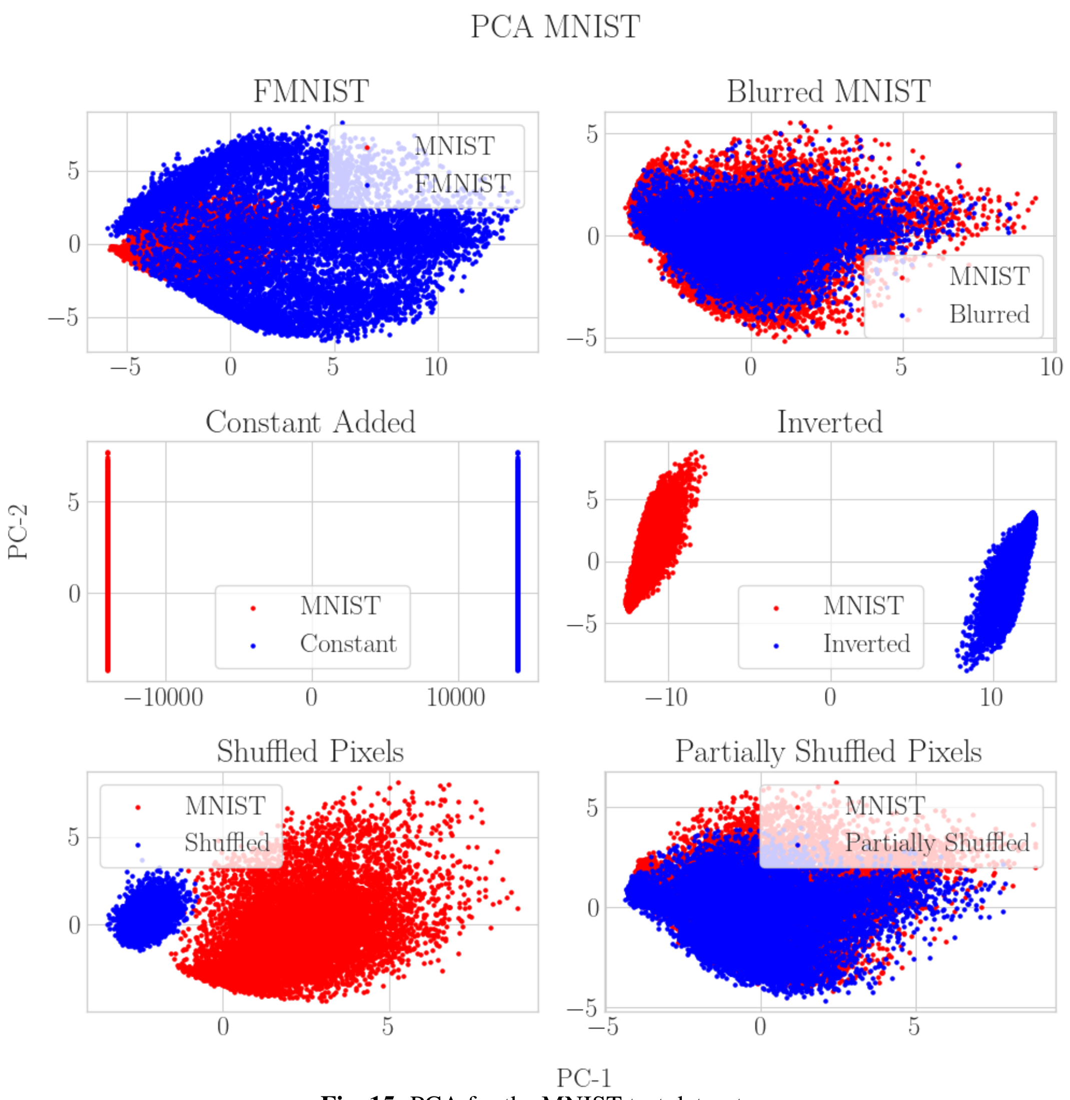

Fig. 15: PCA for the MNIST test datasets.

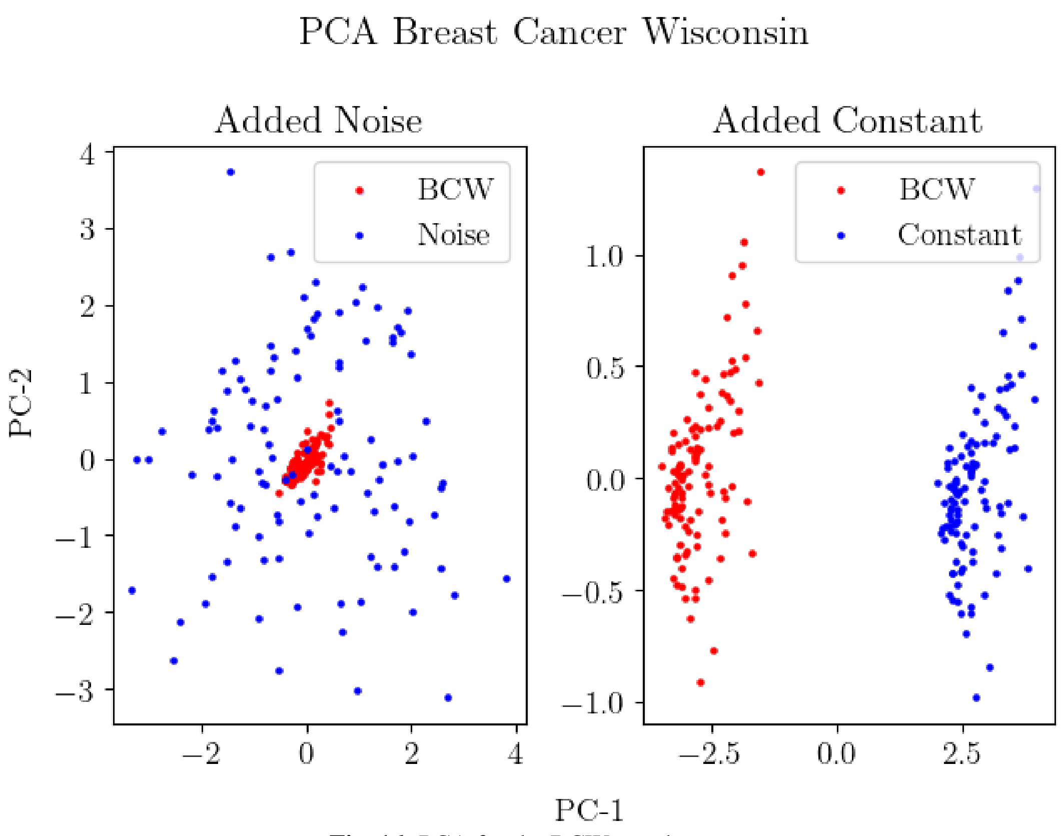

Fig. 16: PCA for the BCW test datasets

5. DISCUSSION

In this paper we introduced *PRE-ACT* and *PRE-ACT+ENT* - two alternative methods for classification outlier detection in Bayesian Neuron Networks, which is based on looking for unusual pre-activation neuron values in the last layer of the model.

5.1. Key findings

In Section 4.1, 4.2, and 4.3, we explored and compared the baseline method for classification outlier detection and the proposed alternative methods. Here, we compared in total three different methods, two of which were new, on three different datasets: simulated data, the Breast Cancer Wisconsin dataset, and the MNIST dataset.

The results gave indications that the performance of the methods are dependent on the type of dataset. The simulated dataset is a simple bivariate simulated data, the Breast Cancer Wisconsin dataset is tabular data, and MNIST is an image dataset. The proposed methods performed better than the baseline on both the simulated dataset and the Breast Cancer Wisconsin dataset but performed worse than the baseline *ENT* in five out of six MNIST test datasets.

The reason for the results depending on the type of dataset is unclear. We see a potential reason in the PCA of the test dataset with MNIST and FMNIST (Figure 15). Here, 784 dimensions are reduced to two dimensions, where about 34% of the variance is explained (computations not shown). This figure shows that there is an overlap between MNIST and FMNIST, in contrast to the BCW test datasets in Figure 16, where we can see which samples are outliers. In the case of the PCA of MNIST and FMNIST, we would have liked to see a clear separation between the two, as we can with the MNIST data and where a constant is added to the MNIST data Figure 15. However, in the PCA figure, we also see a clear separation between MNIST and the inverted MNIST, *PRE-ACT* still performs poorly here.

Another explanation could be the number of classes K. The simulated dataset has four classes, the BCW has two classes, and the MNIST dataset has ten classes. Getting OOD outliers in datasets with few classes K might be easier. In contrast, in the datasets with many classes K we might get a situation where the outliers somehow end up being in In-Between areas, which enables *ENT* to work well.

5.2. Limitations

We briefly mention limitations to this study, which should be subject to further study. Firstly, the combination used in the proposed method *PRE-ACT* and *PRE-ACT+ENT*, i.e. which k to use for $\sum_{k=1}^{K-1} \max_k(u^L)$ and significance level α, were based on empirical results from visualizations. The best method was evaluated by what gave the best trade-off between FDR, POWER, and accuracy. We would ideally use optimization, for instance, the Neyman-Pearson optimization [33], to minimize the FDR and maximize the POWER.

Secondly, as indicated in Section 5.1, the results depend on the dataset used in experiments. This is especially relevant for the simulated data. Results might depend on the simulation settings, such as the number of classes K, dimensions d, and samples n. The balance of the dataset is also something to consider.

Lastly, the proposed method for separating In-Between and Out-Of-Distribution outliers in Section 3.4 was not validated on any test datasets beyond what is seen in Figure 8. Before concluding if this is a viable method, one would need to explore this further.

5.3. Furthers works

As the proposed methods only focus on the last layer of the network, it might be interesting to explore the possibilities of using every layer of the model, as this might improve the results. The idea is that information might be lost during a forward pass in the network.

Another interesting path is not to look at the sum of the $K - 1$ highest pre-activation neuron values in the last layer but rather at each element in the set of pre-activations. Here, one might gain information by looking at the structure of the pre-activation neuron values.

Acknowledgement

I would like to thank Prof. Solve Sæbø, Associate Prof. Aliaksandr Hubin, and Research Scientist Filippo Remonato for guidance and help during the work of this paper.

6. REFERENCES

[1] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol, "Ai in health and medicine," *Nature medicine*, vol. 28, no. 1, pp. 31–38, 2022.

[2] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2020.

[3] Longbing Cao, "Ai in finance: A review," *Available at SSRN 3647625*, 2020.

[4] Mohammad Mahdi Bejani and Mehdi Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artificial Intelligence Review*, pp. 1–48, 2021.

[5] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun, "Hands-on bayesian neural networks—a tutorial for deep learning users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[6] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021.

[7] Christos Louizos and Max Welling, "Multiplicative normalizing flows for variational bayesian neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2218–2227.

[8] Lars Skaaret-Lund, Geir Storvik, and Aliaksandr Hubin, "Sparsifying bayesian neural networks with latent binary variables and normalizing flows," 2023.

[9] Léon Bottou, Frank E Curtis, and Jorge Nocedal, "Optimization methods for large-scale machine learning," *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 1321–1330.

[11] D Michael Titterington, "Bayesian methods for neural networks and related models," *Statistical science*, pp. 128–139, 2004.

[12] Thomas Bayes, "Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s," *Philosophical transactions of the Royal Society of London*, , no. 53, pp. 370–418, 1763.

[13] Charles J Geyer, "Practical markov chain monte carlo," *Statistical science*, pp. 473–483, 1992.

[14] David M Blei, Alp Kucukelbir, and Jon D McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[15] Armen Der Kiureghian and Ove Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[16] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft, "Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1184–1193.

[17] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *Advances in neural information processing systems*, vol. 32, 2019.

[18] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig, "Being bayesian, even just a bit, fixes overconfidence in relu networks," in *International conference on machine learning*. PMLR, 2020, pp. 5436–5446.

[19] Solomon Kullback and Richard A Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[20] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] Wray L Buntine, "Operations for learning with graphical models," *Journal of artificial intelligence research*, vol. 2, pp. 159–225, 1994.

[22] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[23] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, 2013.

[24] Diederik P Kingma, Max Welling, et al., "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[25] Yarin Gal et al., "Uncertainty in deep learning," 2016.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, "6.2. 2.3 softmax units for multinoulli output distributions," *Deep learning*, vol. 180, 2016.

[27] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight uncertainty in neural network," in *Proceedings of the 32nd International Conference on Machine Learning*, Francis Bach and David Blei, Eds., Lille, France, 07–09 Jul 2015, vol. 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, PMLR.

[28] Shishir Sharma, "bayes-by-backprop," `https://github.com/nitarshan/bayes-by-backprop`, 2020.

[29] Dheeru Dua and Casey Graff, "UCI machine learning repository," 2017.

[30] Li Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

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

[32] Michael E Tipping and Christopher M Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[33] Jerzy Neyman and Egon Sharpe Pearson, "Ix. on the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, no. 694-706, pp. 289–337, 1933.