MASTER'S THESIS

Bayesian Deep Learning for Morphological Classification of Radio Galaxies

---

Author:
Devina Mohan

Supervisor:
Prof. Anna Scaife

A thesis submitted to the University of Manchester
for the degree of Master of Science by Research
in the

Department of Physics and Astronomy in the School of Natural Sciences
Faculty of Science and Engineering

2022

Contents

List of Figures

List of Tables

List of Abbreviations

AGN	Active Galactic Nucleus
BBB	Bayes by Backprop
CE	Cross-Entropy
CNN	Convolutional Neural Network
ELBO	Evidence Lower Bound
FIM	Fisher Information Matrix
FR	Fanaroff-Riley
GMM	Gaussian Mixture Model
KL	Kullback–Leibler
MAP	Maximum *a Posteriori*
MC	Monte Carlo
ML	Machine Learning
MLE	Maximum Likelihood Estimate
MLP	Multi-Layer Perceptron
NLL	Negative Log Likelihood
NN	Neural Network
PAC	Probably Approximately Correct
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
VI	Variational Inference

Abstract

The next-generation of radio facilities will produce huge volumes of data and the use of deep learning methods seems inevitable. However, in general these models produce overconfident predictions and provide no uncertainty estimates. In this work we use variational inference (VI) as an approximation to Bayesian inference and present the first application of a VI-based approach to morphological classification of radio galaxies, using a binary FRI/FRII classification. We show how posterior uncertainties on model predictions can be recovered and find that on average model uncertainty is correlated with the degree of belief of the human classifiers who curated the data set. Additionally, to reduce the computational and storage cost of these models at deployment, we test model pruning strategies and find that a Fisher information based metric allows for a higher proportion of the fully-connected layer weights of the network to be pruned, by up to 60%, compared to a SNR-based metric without compromising on model performance. We find that model uncertainty is reduced for both pruning methods. Finally, we show that our model experiences a cold posterior effect and examine whether this effect is due to model misspecification. We observe no difference in model performance on testing a hypothesis for mitigating this effect and conclude that model misspecification is not the major contributing factor to the cold posterior effect observed in our work.

Declaration of Authorship

I declare that no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

Acknowledgements

I would like to thank my supervisor Prof. Anna Scaife for her incredible mentorship and guidance throughout the project, for always being kind and supportive and for all the discussions on new and exciting topics.

I would like to thank my family for their love and support throughout the years.

I would also like to thank my fellow MScs, Alex and Nathan, for staying connected in a disconnected time from a thousand miles away.

Chapter 1

Introduction

1.1 Radio Galaxies

Radio galaxies are a sub-class of AGN, which along with radio-loud quasars are classified as radio-loud AGN. These galaxies are characterised by large scale jets and lobes which can extend up to mega-parsec distances from the central black hole and are observed in the radio spectrum. These jets emit synchrotron radiation due to the presence of highly relativistic electrons that interact with magnetic fields. Synchrotron emission has a steeply falling spectrum, with a spectral index of $\alpha \sim -0.7$, thus these galaxies have a higher flux density at low frequencies.

1.2 Radio Galaxy Classification

1.2.1 The Original Dichotomy

Fanaroff & Riley (1974) studied 57 sources in the 3CR catalogue and proposed a classification of extended radio sources based on the ratio of the distance between the highest surface brightness regions on either side of the galaxy to the total extent of the radio source, $R_{\rm FR}$. Based on a threshold ratio of 0.5, the galaxies were classified into two classes as follows: if $R_{\rm FR} < 0.5$, the source was classified into Class I (FRI; edge-darkened), and if $R_{\rm FR} > 0.5$ it was classified into Class II (FRII; edge-brightened), see Figure 1.1. Additionally, the two classes were also distinguishable on the basis of a threshold radio luminosity, $L_{178\ \rm MHz} = 2 \times 10^{25}\ \rm W\,Hz^{-1}\,sr^{-1}$, such that most of the FRIs were found to exist below this threshold and FRIIs above it. Thus it was concluded that there exists

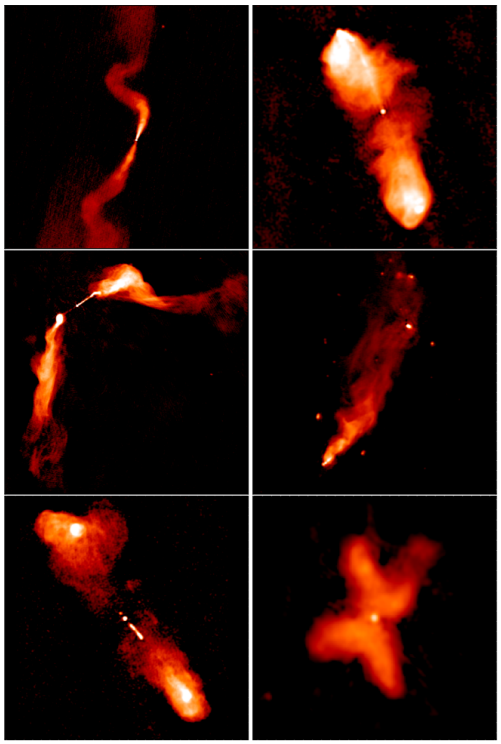


FIGURE 1.1: Different morphologies of radio galaxies, as presented in Hardcastle & Croston (2020). Starting clock-wise from the top-left corner: FRI source 3C 31, FRII source 3C 98, Wide-Angle Tail (WAT) source 3C 465, Narrow-Angle Tail (NAT) source NGC 6109, Double-Double source 3C219, core-restarting radio galaxy 3C 315.

a correlation between the positions of regions of high/low brightness emission and luminosity. However, even in the original samples, these luminosity boundaries were not very sharp.

Based on our current understanding of jet dynamics, the differences in FRI/FRII morphology can be explained on the basis of their relativistic jets: FRIs have jets that are initially relativistic, but get disrupted at a short distance from the central BH and decelerate on kpc scales, hence they are also known as *edge-darkened* or *centrally brightened*, whereas FRIIs have relativistic jets that extend out to the lobes and terminate in hotspots due to internal shock, hence they are also known as *edge-brightened* (Bicknell, 1995; Laing & Bridle, 2014). However, there is still a continuing debate about the exact interplay between extrinsic effects, such as the interaction between the jet and the environment, and intrinsic effects, such as differences in central engines and accretion modes, that give rise to the different morphologies.

1.2.2 Extensions to the original classification

Over the years, several other morphologies have been observed, some of which are described in this section, see Figure 1.1 for examples of these galaxies.

Bent-Tail sources: Bent-tail sources are thought to be produced due to the movement of galaxies through the intra-cluster medium (ICM) in clusters of galaxies, with cluster winds bending the tails at an angle (Rudnick & Owen, 1976). Depending on the degree at which the tails bend, bent-tail sources are further classified into Wide-Angle Tails (WATs) and Narrow-Angle Tails (NATs).

Using VLA observations, O'Dea & Owen (1985) studied 57 sources in galaxy structures and observed 41 NATs, 9 WATs and 7 sources with complex morphologies that could not be classified as NAT/WAT. Out of the 41 NATs, some were single-tailed sources whereas others were twin-jet structures, classified on the basis of how the jet transitioned. While some jets transitioned into broad diffuse tails, others were observed to expand gradually and were predicted to merge into a single tail. It was also found that some jets were brighter near the core and others had high luminosity jets with gaps in emission near the core. Out of the 41 NAT sources, 11 were observed to have large scale curvature in their tails. Among the 9 WAT sources, some structures were associated with cD galaxies (galaxies in the centers of clusters). Complex morphologies included single and twin-jet sources, with extremely distorted jets and diffuse lobes, and physically interacting twin-jet sources. Thus we can see how even within a classification, there may be significant variation.

Hybrid Sources: Gopal-Krishna & Wiita (2000) introduced a new class of radio sources, 'HYbrid MOrphology Radio Sources' (HYMORS), which exhibited FRI-like morphology in one lobe, and FRII-like in the other. This led them to hypothesise that perhaps the FR dichotomy arises because of the way in which jets interact with the environment of the host galaxy, rather than a difference in central jet engines. More recently, Harwood et al. (2020) performed detailed spectral study of five hybrid sources and showed that these objects have spectral features of FRII objects and that hybrid morphology could be explained based on orientation alone.

Double-Double Sources: Schoenmakers et al. (2000) presented 4 double-double sources

and defined them as two radio sources with edge-brightened morphology with a common center. Kaiser et al. (2000) proposed that the inner jets may be due to a re-started AGN. Double-Double sources are now thought to be re-starting FR II galaxies, and thus important for understanding the life-cycles of radio-loud AGN (Mahatma et al., 2020).

FR Type 0: These are sources with no extented emission (Baldi et al., 2015). Although Hardcastle & Croston (2020) discourage the inclusion of FR0s as a 'morphological' class since they are compact extensions of the FRI class of sources, we include them here for completeness. Cheng & An (2018) studied 14 FR Type 0 radio galaxies using VLBI higher resolution images. It was found that these sources have a compact structure and could be a mixed population comprising of GHz-peaked spectrum (GPS), evolved compact steep-spectrum (CSS) and a mix of CSOs (compact symmetric objects) and MSOs (medium-sized symmetric objects). In addition to that, Garofalo & Singh (2019) also studied the FR0 class of active galaxies, which had similar properties to that of FRIs, except that the ratio of the core to total emission was 30 times higher in FR0s. It was predicted that if sufficient fuel exists, FR0s could evolve into FRIs.

1.2.3 Current state of observational data

More recently, Mingo et al. (2019) conducted the largest morphological investigation of extended radio loud AGN using the LOFAR Two-Meter Sky Survey (LoTSS) and studied 5805 sources. Notably they found that a significant number of low-luminosity FRII sources, which are referred to as 'FRII-lows', with luminosities that were 3 orders of magnitude below the traditional FRI/FRII threshold luminosity $L \sim 10^{25}$ W Hz^{-1}, see Figure 1.2. Thus, it was concluded that the luminosity break discovered by Fanaroff & Riley (1974) does not predict the class of the radio galaxy. This was attributed to the difference in the sensitivity of the two surveys and also its relationship to redshift, since FRII-lows are rare in the local universe.

FRII-lows: Two theories were proposed and tested to explain the existence of FRII-low galaxies : (i) Some FRII-lows may be an older population of FRIIs, which have started fading (Shabala et al., 2008; Hardcastle, 2018). Through the LoTSS observations, it was found that some FRII-lows have higher spectral indices compared to FRIIs at a higher luminosity, but that many FRII-lows also exist within the same range as FRII-highs. Thus,

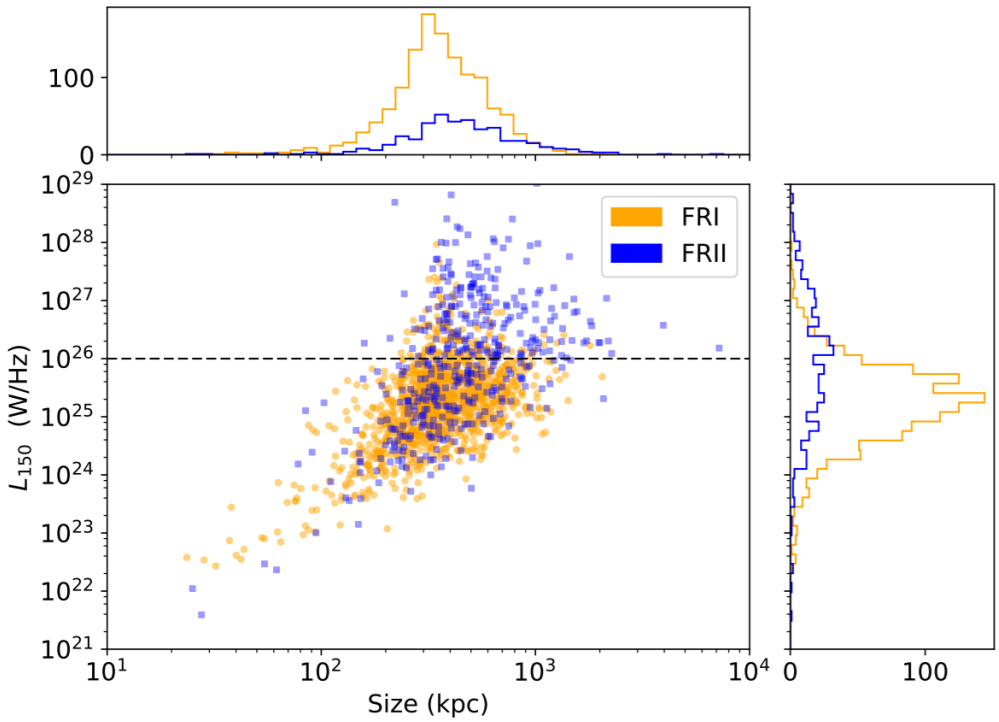


FIGURE 1.2: Luminosity at 150 MHz vs Physical Size of FRI and FRII galaxies from Mingo et al. (2019). Many FRIs and FRIIs can be found beyond their traditional luminosity threshold predicted by Fanaroff & Riley (1974).

this could not be the only explanation. (ii) FRIIs exist mostly in low density environments, which allows the jets to remain undisrupted. FRII-lows in the LoTSS sample were found to exist in host galaxies with lower luminosities, and were also found to have fainter hosts than FRIs at similar luminosities.

It was also found that populations of both FRIs and FRIIs are heterogeneous, including NATs, WATs and double-double sources. FRII-lows were also found to have a heterogeneous population, including bent-tail sources.

Correlation between radio and optical luminosities: Ledlow & Owen (1996) studied the relationship between host galaxy luminosity in the optical and radio luminosity using the Owen-Ledlow diagram for the FRI/FRII division as it was originally defined by Fanaroff & Riley (1974) and found that they are approximately related as $L_{\text{radio}} \propto L^2_{\text{optical}}$ However, later works showed that this does not entirely hold true (Gendre et al., 2013; Mingo et al., 2019). For instance, Mingo et al. (2019) found FRI and FRIIs sources across a range of optical luminosities.

1.2.4 Theories to explain different radio loud AGN populations

While so far we have discussed a morphological classification of radio galaxies, in this section we also consider a more fundamental dichotomy of radio galaxies based on excitation modes defined using optical spectral lines. Based on the relative intensity of high and low excitation lines, radio galaxies are also classified as: High-Excitation Radio Galaxies (HERGs) and Low-Excitation Radio Galaxies (LERGs). These populations are thought to be associate with two different feedback pathways in AGN and are influenced by factors such as fuelling mechanisms, host galaxy properties and environmental influences, among others. They are thus important for understanding the role of radio-loud AGN in galaxy evolution.

Although there does not exist a one-to-one mapping between morphological classes and excitation modes, it has been suggested that most FRIs are LERGs, whereas FRIIs will comprise of both HERGs and LERGs (Mingo et al., 2019; Hardcastle & Croston, 2020).

On accretion modes and fuelling

The two excitation modes have distinct accretion rates, as shown by Best & Heckman (2012): $L_{\rm HERG} \sim 0.1\ L_{\rm edd}$ and $L_{\rm LERG} \sim 0.01\ L_{\rm edd}$, where $L_{\rm edd}$ is the Eddington luminosity, which is the luminosity at which the gravitational force and radiation force balance out. Thus, HERGs are also known as *radiatively efficient*, whereas LERGs are known as *radiatively inefficient*.

LERGs are also known as *hot mode*, since they are believed to be fuelled by accretion of hot gas from the intergalatic medium (IGM) , whereas HERGs are known as *cold mode* and believed to be fuelled by cold gas supplied by recent mergers or interactions. However, at a given radio luminosity, FR morphology is thought to be independent of accretion modes (Gendre et al., 2013)

On host galaxy properties and environment

Best & Heckman (2012) found that LERGs are found in more massive galaxies that have large BH mass; whereas HERGs reside in host galaxies that have lower stellar mass and lower BH mass at a similar radio power. Since most FRIs are LERGs, we expect to find them in higher mass host galaxies.

Using the Owen-Ledlow diagram to study the distribution of radio galaxies and including the dust properties of the two FR classes, Saripalli (2012) found a connection between host galaxy properties and the conditions that give rise to the FR dichotomy. FRIs were found to be hosted by higher mass elliptical galaxies; whereas interactions and mergers in low mass ellipticals were proposed to cause FRII morphology. Since most HERGs are FRIIs, we also find HERGs in environments where there are more mergers and interactions. This is broadly consistent with the finding that the host galaxies of HERGs are bluer than that of LERGs, which means that they have a higher star formation rate than LERGs (Janssen et al., 2012; Best & Heckman, 2012). However Miraghaei & Best (2017) also compared FRI LERGs and FRII LERGs and found that FRIs were hosted by smaller galaxies with high stellar mass to BH ratios compared to FRIIs.

Links between environment and excitation modes have also been established: HERGs are found in low density/poorer environments, whereas LERGs are found in a range of cluster environments (Gendre et al., 2013; Ineson et al., 2015). This is consistent with the finding that FRIs are predominantly located in high density environments (Gendre et al., 2013) where it is thought that the high density environment disrupts the jets, as described in Section 1.2.1, whereas FRII LERGs are found in richer environments than FRI LERGs (Hardcastle, 2004).

On evolution, particle content

HERGs and LERGs also have different redshift evolution at fixed radio luminosities (Best & Heckman, 2012). While HERGs evolve strongly at all radio luminosities, LERGs evolve weakly or do not evolve at all.

It has also been shown that HERGs and LERGs cannot be distinguished on the basis of the particles in their lobes and hence the contents of lobes are not related to excitation modes; whereas it was also shown that FRIs and FRIIs have different plasma conditions, which was attributed to the presence of more proton content in FRIs than FRIIs (Croston et al., 2018).

From the preceding discussion it should be evident that although the original FR classification has been used for more than four decades, it is still unclear exactly how it maps to the physical and/or environmental properties of the galaxies themselves. Intrinsic

and environmental effects are often difficult to disentangle using radio luminosity alone as systematic differences in particle content, environmental effects and radiative losses make radio luminosity an unreliable proxy for jet power (Croston et al., 2018). Hence the use of morphology for inferring the environmental impact on radio galaxy populations is therefore important for gaining a better physical understanding of the FR dichotomy, and of the full morphological diversity of the population. It is hoped that the new generation of radio surveys, with improved resolution and sensitivity, will play a key part in finally answering this question.

1.3 Machine Learning Approaches to Radio Galaxy Classification

The upcoming radio telescopes such as the Square Kilometre Array and its precursors which are already operational are expected to detect millions of radio sources, making classification via visual inspection impossible within a reasonable time frame. The Rapid ASKAP continuum survey (McConnell et al., 2020) has already detected three million extended sources and the upcoming Evolutionary Map of the Universe (EMU) survey is expected to detect 70 million radio sources. The large data volumes have led to the increased use of automated detection and classification tools. In the past decade, the field of Deep Learning (DL) has also revived due to the availability of large labelled data sets and advances in GPU-accelerated computing. This has led to the adoption of deep learning techniques for radio galaxy classification. In this section we review previous works that have been done with radio galaxy data using machine learning and discuss the challenges associated with the application of these algorithms to catalogues of radio galaxies. In the next chapter we describe deep learning models and how they learn from data in detail.

Aniyan & Thorat (2017) were the first to examine deep learning for morphological classification of radio galaxies. They used Convolutional Neural Networks (CNNs), a type of deep learning model used with grid-like data such as images, for classifying sources into FRI, FRII, and bent-tail sources. Following that, Wu et al. (2018) developed ClaRAN, an end-to-end pipeline for source identification and classification using Faster Region-based CNN. A combination of localisation and recognition networks were used

to find regions of interest in a given field of view and morphology was determined based on the number of components and peaks in that region. It was also found that a technique known as *transfer learning* could significantly improve model performance (Pratt et al., 1993).

Transfer learning involves using weights trained on a different data set to initialise (and possibly freeze) another model's weights. This method works especially well when only used for the initial layers of a network because these layers learn similar features such as lines and edges, irrespective of the data set being used. While Wu et al. (2018) transferred weights of the first five layers from a VGG-16 model, a deep neural network with 16 layers and 138 million parameters, trained on the ImageNet data set and found that it improved their model performance, Tang et al. (2019) made use of cross-survey transfer learning to test whether weights inherited from models trained on one radio survey could be used as a starting point for training the same model on another survey. It was found that using a higher resolution survey data such as FIRST can improve the performance of a model being trained on lower resolution survey data such as NVSS but transfer learning in the opposite direction did not perform as well.

Several machine learning algorithms including logistic regression, random forests and CNNs were examined by Alger et al. (2018) for cross-matching galaxy hosts in infrared to their radio counterpart. It was found that training models for cross-identification using citizen science data from Radio Galaxy Zoo was equivalent to training models on cross-identifications made by experts. However, it was noted that their methods did not outperform classifications made by the nearest neighbour algorithm. Since the data set mostly contained unresolved sources, it was suggest that it may be more useful to use the nearest neighbour algorithm if most of the sources are unresolved. A similar observation was made by Wu et al. (2018), who note that classifying compact sources does not benefit from using CNNs. Therefore, one must be careful before applying complicated neural network architectures to problems that can more easily be solved by other, more simpler learning algorithms.

Lukic et al. (2019) also compared several variations of Capsule Networks (Sabour

et al., 2017) to CNNs. A capsule network consists of multiple capsules (groups of neurons) that represent the instantiation parameters (such location, position, scale and orientation) of the feature in an image. These instantiation parameters are learned and capsules in multiple layers of the network are connected through a routing protocol to help preserve the relative positions of each feature in the image. A CNN on the other hand learns the features of an image, but not their relative positions. It was found that CNNs outperformed capsule networks on radio galaxy data. Thus, it was concluded that preserving the relative location of features is not necessary and may be detrimental for classification. This is again an example where a simpler CNN architecture was found to work better than a more complicated architecture. It was also found that FRIIs were less accurately classified by their model than FRI or unresolved sources. However, in this case their sample also had fewer FRIIs than other sources.

In a step toward interpretability of deep learning models for radio galaxy classification, Bowles et al. (2021) used attention-gated CNNs to generate attention maps. These maps showed what regions of the images were being used by the model to make classifications. The results showed that attention-gated CNNs focus on similar areas in the image that a human would use to distinguish between FRI/FRII: for FRIs the model focused at the central regions of the images, whereas for FRIIs the model focused on the outer regions of the images which contained lobes. Notably, the attention-gated model could achieve comparable accuracy to standard CNNs while using 50% fewer parameters.

Standard CNNs are equivariant to translation only, which means that any translation of the input features will result in an equal translation in the output feature map. This property is achieved by weight sharing and it ensures that features such as lines, edges, and motifs can be detected anywhere in the image. Scaife & Porter (2021) showed how group equivariant CNNs (G-CNNs) which make the model equivariant to translation, rotation and refection could be applied to radio galaxy data. G-CNNs are a more generalised form of CNNs that preserve group equivariance through the convolutional layers that contain groups of symmetry transforms. In particular, they used Steerable G-CNNs that describe E(2)-equivariant convolutions. The Euclidean group E(2) is the group of isometries of the plane $\mathbb{R}^2$ that contains translations, rotations and reflections. Cyclical

and dihedral subgroups of the E(2) group were considered. It was also shown that classification performance for the best performing D_{16} model, which contains a set of discrete rotations in multiples of $2\pi/16$ and reflections around $x = 0$, could be improved by using Monte Carlo (MC) Dropout. MC dropout, which is an approximate Bayesian technique, was also used to quantify epistemic uncertainty in the classifications.

Challenges: Applying deep learning to radio astronomy comes with unique challenges. Unlike the application of DL to large data sets such as MNIST, which contains $\sim$ 70,000 images, and ImageNet, which contains 14 million images, there is a dearth of labelled data in radio astronomy. For example, the MiraBest data set used in this work has $\sim$ 1200 images only. This creates the need to augment pre-existing data sets. A possible solution for the lack of labelled data was given by Bastien et al. (2021), who used structured variational inference (VI) to simulate populations of radio galaxies. Structured VI consists of a variational autoencoder (VAE) which learns representations of input data in a low dimensional latent space. These representations are the parameters of a probability distribution. The distribution is then sampled by a decoder to generate a new data sample. The VAE is trained to minimise the reconstruction loss which measures the difference between the input sample and the generated image. Any biases associated with small data sets are also propagated to the larger augmented data sets using which DL models are trained. We also note that, while all other works using the data set we have used apply some form of augmentation, in this work we do not use any data augmentation and study how a variational inference based Bayesian neural network (BNN) performs with small data sets.

Another challenge is that of artefacts, misclassifed objects and ambiguity arising from how the morphologies in these data sets are defined. If we compare images in Figure 1.1 to the images in our data set in Figure 4.1, we can see how different the ML data set looks from the canonical radio galaxy images shown in Figure 1.1. While BNNs cannot remove any of these effects, they can give an estimation of how uncertain the model is in its prediction. When properly calibrated, these uncertainty estimates can serve as a diagnostic tool to mitigate these effects.

With the exception of Scaife & Porter (2021), no published work in radio galaxy classification has currently looked at model uncertainty. As we shall see in the following chapters, using a VI-based BNN allows us to identify galaxies that have artefacts, or may

be misclassified. However, there are some examples for which we found that the model confidently makes predictions that are wrong. In this case, we found the input galaxies to be ambiguously defined.

The thesis is organised as follows: in Chapter 2 we describe neural networks and how they are trained using data. We discuss challenges associated with training neural networks, such as regularisation, and discuss the limitations of these models. In Chapter 3, we present the *Bayes by Backprop* algorithm, which is a variational inference-based Bayesian neural network (Blundell et al., 2015). This algorithm forms the basis of the models used in this work. We also show some preliminary experiments with MNIST and present a weight pruning method to see how these work when there is plenty of data. In Chapter 4 we apply the techniques described to a data set of radio galaxies and analyse the results. Finally in Chapter 5, we present some considerations for improving variational inference for radio galaxy classification.

Chapter 2

Neural Networks

2.1 Introduction

While a single layer neural network was first proposed by Rosenblatt in 1958, it took several decades for these models to achieve state-of-the-art accuracy. The recent success of neural networks can be attributed to *big data* and leveraging GPUs for parallel computation. In a computer vision competition, Krizhevsky et al. (2012) showed that deep convolutional neural networks could outperform the next best entry by $\sim$ 10%. Neural network models now find application in a wide range of tasks such as object detection, natural language processing, image generation and reconstruction. In astronomy, deep learning has been used for tasks such as classifying pulsar candidates (Wang et al., 2019), diagnosing radio telescopes (Mesarcik et al., 2020) and classifying glitches in LIGO data (George et al., 2018) among many others. In this chapter we will discuss the how neural networks models are trained using data in the context of supervised learning where we have a set of input and output data pairs.

2.2 The Artificial Neuron

The artificial neuron is a mathematical model inspired by biological neurons. Each neuron calculates a weighted sum of its real-valued inputs and applies an activation function to produce an output. The weight values determine the importance of a particular input for producing the output. The activation function assigns an activation value to the neuron, which determines the threshold that will switch it on. This function may be linear or

non-linear. The activation value is analogous to the action potential that causes biological neurons to fire in the brain. An additional bias term may be added to allow the threshold values of individual neurons to be shifted. An example of such a neuron is shown in Figure 2.1. Networks constructed using multiple neurons arranged in various architectures are known as Artificial Neural Networks (ANNs) or Neural Networks (NNs).

Using a linear threshold activation function with a single neuron results in a simple binary classifier known as the perceptron algorithm (Rosenblatt, 1958). It is also referred to as a single-layer neural network because the inputs are directly connected to the output layer neuron through a set of weights. However, these single-layer networks are only able to learn outputs that are linearly separable. For instance, while the perceptron algorithm can learn Boolean functions such as AND, OR and NOT, it cannot learn the XOR function.

The single-layer model can be extended by introducing multiple layers in the network, but in order to take advantage of the multi-layered architecture to learn complex functions, we use non-linear activation functions (Rumelhart et al., 1986).

The neurons across multiple layers may be connected such that information only flows forward through the neurons, resulting in a Feed-Forward Neural Network. Alternatively, the neurons may be connected in a cyclic fashion, resulting in a class of networks known as Recurrent Neural Networks (RNNs) which can be used with time-domain data (Naul et al., 2018). In this work we will focus on feed-forward neural networks and look closely at the two types of feedforward NNs widely used for classification: Multi-layer Perceptrons (Section 2.3) and Convolutional Neural Networks (Section 2.10).

2.3 Multi-Layer Perceptrons

A multi-layer perceptron consists of several neurons arranged in a layered architecture. All the neurons in adjacent layers are connected to each other, which is why these networks are also known as fully-connected neural networks (FCNN). Connections between neurons of adjacent layers are parameterised by point-estimates of weights, which quantify the importance of the connection for the final output of the network. A MLP architecture consists of three types of layers:

1. **Input Layer:** The first layer of the network, which is known as the input layer, takes in the pre-processed data and passes it to the next layer. For example, if the input

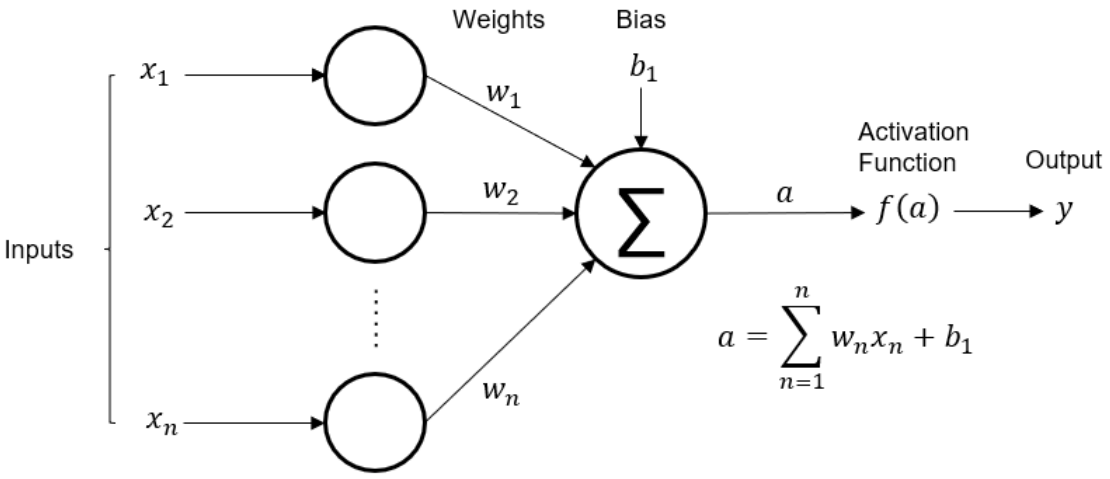


FIGURE 2.1: The artificial neuron: A single unit of a neural network

data is a set of greyscale images, for a fully-connected NN the input layer will be a flattened array of pixels. For a CNN, it will be a 2D array of pixels.

2. **Hidden Layer:** One or more hidden layers are used to model complex relationships between the input and the output. The number of hidden layers determines the complexity of the solution that can be modelled. We define the depth of a neural network by the number of hidden layers. One of the reasons for the recent success of neural networks is due to the strides in computational capabilities that allow *deep* neural networks to be trained with large data sets.

3. **Output Layer:** The output layer makes classifications by passing its outputs through a Softmax function. This normalises the outputs to give a prediction score between $(0, 1)$. Each neuron in the output layer corresponds to one of the predefined classes that we want to classify our data into.

Hornik et al. (1989) showed that multi-layer feed forward neural networks can be considered as a type of universal function approximator. The layered architecture allows these models to approximate the underlying data distribution by learning non-linear feature representations. These representations become more abstract as the depth of the network increases, which helps the model distinguish between features that are essential for classification such regions of emission in the image of a galaxy and ignore irrelevant information such as background noise.

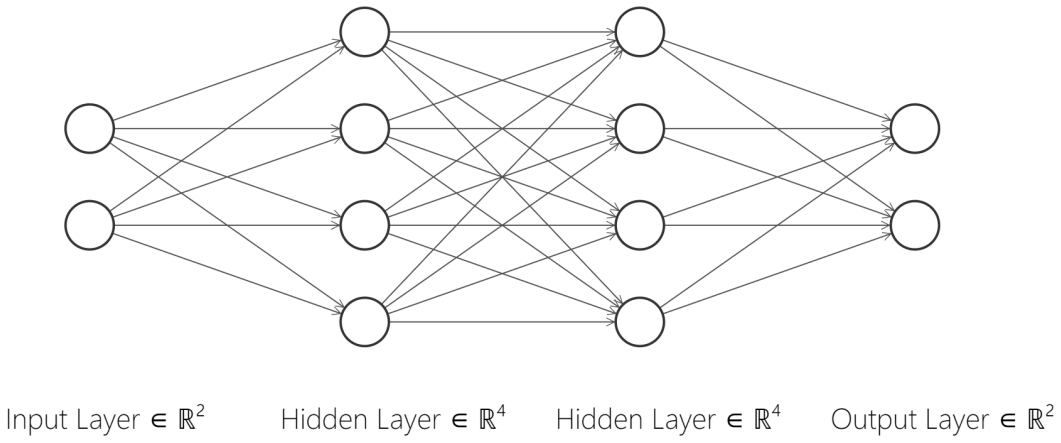


FIGURE 2.2: A fully-connected neural network

Figure 2.2 shows a multi-layer perceptron with two inputs, 2 hidden layers with 4 neurons each and an output layer with 2 neurons. We use the notation w^l_{jk} to describe the weight from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the $(l)^{th}$ layer. For example, the connection between the last neuron of the third layer and the last neuron in the output layer in the figure will be denoted as w_{24}. Each neuron in a layer has a bias value b^l_j. This simple neural network has a total of 44 learnable parameters: 32 weights and 12 biases. The optimal values of these weights and biases are learned from data.

2.4 Learning from Data

The data set for a supervised learning problem consists of a set of example input-output pairs (x_i, y_i), where each input has a corresponding label or class. For instance, for a radio galaxy classification problem, the image of the galaxy will be the input and its classification into FRI or FRII will be the label. We divide the data set into three parts to facilitate learning and measure the performance of our trained model:

1. **Training Set**: During training, the learning algorithm uses a subset of the data set called the training set to fit the model to the data. This subset should ideally be representative of the different types of classes in the entire data set. It should also be class-balanced i.e., each class should have equal representation in the data

set. However, this is not always possible, for instance there are very few hybrid radio sources compared to FRI/FRII sources and we must use suitable performance metrics to evaluate models trained using imbalanced data sets.

2. **Validation Set**: A portion of the training set, known as the validation set, is used for choosing the optimal hyper-parameters of the network, such as the learning rate, ϵ, and weight decay coefficient, α, which are not learnable parameters of the network.

3. **Test Set**: The remaining part of the data set is reserved for testing the generalisation performance of the network. This is known as the test set and the network does not see this data at any point during training. After a model has been trained, we calculate the percentage of incorrectly classified samples by comparing the output of the network to the labels in the test set. This is known as the *test error*.

We assume that the data set, which is described by the empirical distribution $\hat{p}_{\text{data}}(\mathbf{x}, \mathbf{y})$, is constructed by taking samples from an underlying data generating process defined by the joint distribution $p_{\text{data}}(\mathbf{x}, \mathbf{y})$, which is unknown to us. We assume that the samples are independently and identically distributed (i.i.d.), which allows us to use the training set to learn about the test set (Goodfellow et al., 2016).

In supervised learning, the goal is to learn a mapping from the inputs to a set of predefined output classes. The hidden layers of the neural network perform a set of non-linear transformations on the input via activation functions (Section 2.5), which help the model learn complex functions. In what is known as a forward pass, the network produces an output $\hat{\mathbf{y}}$ by passing the inputs through the hidden layers. The output is compared with the ground truth labels $\mathbf{y}$. The model parameters are then updated using gradient descent based optimisers (Section 2.8) such that the difference between $\hat{\mathbf{y}}$ and $\mathbf{y}$, as measured by a loss function (Section 2.6), is minimised. For example, if a model predicts that a galaxy is an FRI during the intial stages of training, but its label is FRII, the calculated loss will be high and the network parameters will be updated to minimise this loss.

During training, the algorithm iterates over the entire training set multiple times which we can set by defining the number of epochs. An epoch involves a forward pass, backpropagation of errors (Section 2.7) and optimisation of network parameters. The network will see the entire training data set once during an epoch. The batch size refers

to the number of data samples used by the optimisation algorithm to compute the gradients and update the model parameters. Within each epoch, we train using mini-batches of data with N_{batch} samples.

A mini-batch is constructed by dividing the data set into M partitions which are randomly sampled from the training set. Typical values of batch size include $N_{\text{batch}} = \{16, 32, 64, 128\}$, though recently it has been shown that smaller batch sizes between $N_{\text{batch}} = 2$ and $N_{\text{batch}} = 32$ give better performance (Masters & Luschi, 2018). Mini-batches are used because: (i) it requires a lot of memory to train entire data sets at once, and the GPUs used to train large networks with big data sets have insufficient memory; (ii) it has been found that using mini-batches allows the optimiser to perform better than batch gradient descent (batch size = entire data set) and pure stochastic gradient descent (batch size = 1). If gradients are calculated for the whole data set, the algorithm could get stuck in a saddle point. Conversely, if we compute the gradients after every single data point, the gradients would become very stochastic and the algorithm could keep oscillating in the loss landscape without finding a local minima; (iii) smaller batch sizes give better generalization performance.

In the following sections we will go through all the elements of the training process described above in detail.

2.5 Activation Functions

In order to learn complex relationships between the inputs and output classes, we need to introduce non-linearities in our network.

In each layer we perform an affine transformation of the input vector to produce an output vector, $\mathbf{a}$, using the learnable parameters of the network (weights and biases). We then apply an element-wise activation function, f, which introduces non-linearities into the network to produce the output of the hidden layer, $\mathbf{h}$. For the first layer of the network with input $\mathbf{x}$, the output is given by:

$$\mathbf{h}^{(1)} = f(\mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}) = f(\mathbf{a}^{(1)}), \tag{2.1}$$

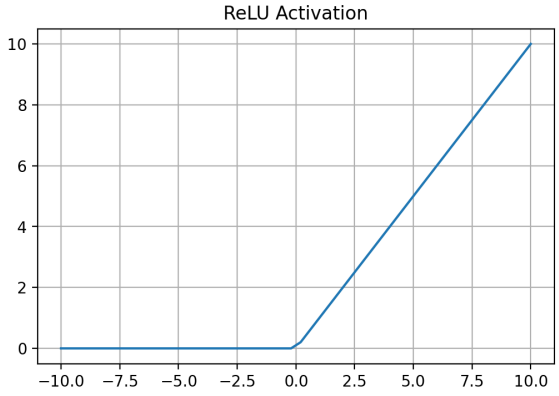


FIGURE 2.3: ReLU activation function visualised for a set of arbitrary input values between [-10,10]

where $\mathbf{W}^{(1)}$ is the matrix of weights, $\mathbf{b}^{(1)}$ is the bias vector, and $\mathbf{h}^{(1)}$ is the output of the layer, which is also called a feature vector.

A deep neural network has several such hidden layers, which take the output of the previous layer as input:

$$\mathbf{h}^{(n)} = f(\mathbf{W}^{(n)T}\mathbf{h}^{(n-1)} + \mathbf{b}^{(n)}) = f(\mathbf{a}^{(n)}). \tag{2.2}$$

In recent years, the Rectified Linear Unit (ReLU) has emerged as the most robust activation function for training MLPs and CNNs. ReLU is a piece-wise linear function which is zero for negative-valued input and equal to the input for positive-valued input. It can be expressed as:

$$f(z) = \max\{0, z\}, \tag{2.3}$$

see Figure 2.3. It is robust to the pitfalls of previously used activation functions such as the sigmoid:

$$f(z) = \frac{1}{(1 + \exp(-z))} \tag{2.4}$$

and tanh:

$$f(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \tag{2.5}$$

functions, which are prone to the problem of vanishing-gradients caused by saturated units (Hochreiter, 1991; Hochreiter et al., 2001).

For a classification problem, the output of the final layer gives a pseudo-probability for each output class by applying the Softmax function. This function acts as a squashing

function that normalises any real valued input from $(-\infty, \infty)$ to a value between $(0, 1)$ such that the sum of outputs for all the neurons in the last layer add up to 1:

$$\text{softmax}(\mathbf{x_i}) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}, \tag{2.6}$$

where n is the number of units in the last layer. The outputs of the Softmax function should be interpreted as prediction scores rather than true probabilities that represent uncertainty in the output. The argmax function is applied to the Softmax outputs in order to get the model's predictions, $\hat{\mathbf{y}}$. This function returns a 1 for the class with the highest prediction score and returns 0 for all the others.

2.6 Loss Functions

The network learns the optimal values of the model parameters by minimising a loss function which quantifies the difference between the model's predictions, $\hat{\mathbf{y}}$, and the ground truth labels, $\mathbf{y}$. Most of the loss functions used with neural networks are based on the principle of maximum likelihood, which is described below, following Goodfellow et al. (2016).

According to the principle of maximum likelihood, we can find the optimal parameters of the network by finding the parameters, θ^*, that maximise the conditional probability, $p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i}; \theta)$. The maximum likelihood estimator of θ is given by:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{n} p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i}; \theta). \tag{2.7}$$

To make the numerical computation stable, we solve an equivalent optimization problem by taking the logarithm:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i}; \theta). \tag{2.8}$$

Equation 2.8 is equivalent to the minimising the Kullback–Leibler (KL) divergence between the empirical data distribution, $\hat{p}_{\text{data}}(\mathbf{x}, \mathbf{y})$, and the model, $p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i}; \theta)$:

$$D_{\text{KL}}(\hat{p}_{\text{data}}, p_{\text{model}}) = \mathbb{E}_{x,y\sim\hat{p}_{\text{data}}}[\log \hat{p}_{\text{data}}(\mathbf{x}, \mathbf{y}) - \log p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i}; \theta)]. \tag{2.9}$$

While training the model, we only need to minimise the terms that are a function of the model. Thus, we can minimise the KL divergence by minimising the negative log-likelihood:

$$J(\theta) = -\mathbb{E}_{x,y\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i};\theta), \tag{2.10}$$

which we denote as the loss function, $J(\theta)$. This is often also referred to as the cost function.

We can calculate this expectation by taking an average over all the training samples:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} -\log p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i};\theta). \tag{2.11}$$

Equation 2.8 is also equivalent to minimisation of the cross entropy between the predicted output, $\hat{y}$, and the actual output, y. We can understand cross-entropy from an information theoretic perspective as follows:

An event with a very high probability contains less information,

$$I = -\log p(x). \tag{2.12}$$

We define the entropy of an event as the average amount of information inherent in it:

$$H(p) = \mathbb{E}_{x\sim p}[I]. \tag{2.13}$$

We can then define the cross entropy as a measure of the dissimilarity between two probability distributions p and q over the same set of events x:

$$H(p,q) = \mathbb{E}_{x\sim p}[-\log q(x)], \tag{2.14}$$

which is similar to Equation 3.14.

In supervised ML, if y is the ground truth and $\hat{y}$ is the predicted label, the cross entropy loss for a binary classfication problem is given by:

$$L(y,\hat{y}) = -y\log\hat{y} - (1-y)\log(1-\hat{y}). \tag{2.15}$$

For a multi-class classification problem, we can calculate the cross entropy using the Softmax outputs, see Equation 2.6, for each class as follows:

$$\text{loss} = \sum_{i=1}^{n} -\log(\text{softmax})_i. \tag{2.16}$$

Note on terminology: In practical implementation, deep learning libraries such as PyTorch allow us to calculate this loss in two ways:

1. Negative log likelihood (NLL) loss function: NLL loss takes log likelihoods as input and outputs a sum. It is used in conjunction with the *log softmax* activation function to calculate Equation 2.16.

2. Cross Entropy (CE) loss function: The CE loss calculates the negative log of its inputs and then calculates a sum. This combines the functionality of the log softmax and negative log likelihood functions and is used in conjunction with the *softmax* activation function.

Using the operations described in the previous sections, we summarise the operations in a forward pass in Algorithm 1.

Algorithm 1 Forward Pass (Goodfellow et al., 2016)

Require: Network depth, l
Require: $\mathbf{W}^{(i)}$, $i \in 1, ..., l$, weight matrices
Require: $\mathbf{b}^{(i)}$, $i \in 1, ..., l$, bias vectors
Require: $(\mathbf{x}, \mathbf{y})$, input and target output pairs
$\mathbf{h}^{(0)} = \mathbf{x}$
for k = 1,....l **do**
$\mathbf{a}^{(k)} = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{k}$ ▷ Calculate weighted sum of input and bias
$\mathbf{h}^{(k)} = f(\mathbf{a}^{(k)})$ ▷ Calculate output of layer k by applying non-linear activation function
end for
$\hat{\mathbf{y}} = \mathbf{h}^{(\mathbf{l})}$ ▷ Output of last layer gives the predicted output
$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_i L(\hat{y}^{(i)}, y^{(i)})$ ▷ Calculate the loss function

2.7 Backpropagation of Errors

In order to decide how to update the network parameters to minimise the loss function, we need a measure of the rate of change of the loss function with respect to each parameter in the network. For a single parameter of the network, say weight w_{jk}, this can

be calculated using a partial derivative: $\frac{\partial L(\hat{y},y)}{\partial w_{jk}}$. However, for calculating the gradient of millions of parameters in a neural network we need an efficient and scalable algorithm.

The back-propagation algorithm (Rumelhart et al., 1986), which is also referred to as *backprop*, is used to compute the gradient of the loss function with respect to the network parameters, $\nabla_\theta J(\theta)$, efficiently. Backprop makes use of the chain rule of calculus to compute the layer-wise gradients of the weighted input for a fixed input-output pair, (x_i, y_i), starting from the last layer to propagate the error in the predicted output backwards through the previous layers.

The following steps are performed in the backprop algorithm (Goodfellow et al., 2016):

1. Calculate the gradient of the loss function with respect to the output of the final layer: $\mathbf{g} = \nabla_{\hat{\mathbf{y}}} J$.

2. Convert this gradient to a function of the input $\mathbf{a^{(k)}}$ (weighted sum + bias) of the previous layer by doing an element wise multiplication of the calculated gradient with the derivative of the activations of the previous layer $f(\mathbf{a^{(k)}})$:

$$\nabla_{\mathbf{a}^{(k)}} J = \mathbf{g} \odot f'(\mathbf{a}^{(k)})$$

3. Calculate the gradient with respect to the weights and biases: $\nabla_{\mathbf{b}^{(k)}} J = \mathbf{g}$ and $\nabla_{\mathbf{w}^{(k)}} J = \mathbf{g}\mathbf{h}^{(k-1)T}$

4. Propagate the gradient with respect to the previous layer's activations.

2.8 Optimisation Algorithms

Optimisation is the process of finding the arguments of a function that minimise or maximise it. An optimisation scheme called gradient descent is used to update the network parameters using the calculated gradients to train neural networks. It involves iterative evaluation of the cost function for different values of network parameters to find the minimum of the cost function. The network parameters are updated by taking a step towards the negative gradient. A good optimiser should be able to train the model to generalise well to unseen data and it should be able to converge in a reasonable amount of time.

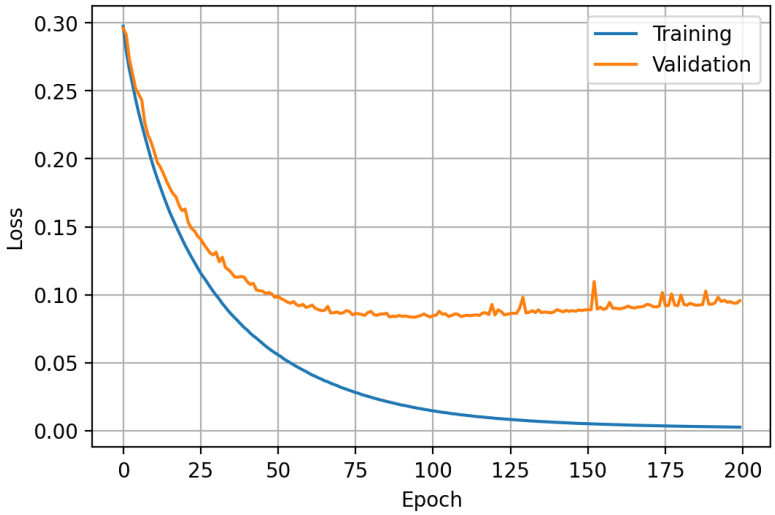


FIGURE 2.4: Learning curves for a MLP model trained using SGD with momentum: Training and validation loss curves for a MLP trained with SGD ($\epsilon = 10^{-3}, \alpha = 0.9$) using the cross-entropy loss function. The training loss converges to a small value and the validation loss reaches a minimum at epoch = 94, after which it starts increasing. This is a sign that the model is overfitting to the data. The gap between the two curves suggests that there is some scope for improvement in the learning process.

In this section we discuss the two most widely used optimisation algorithms for training neural networks: (1) Stochastic Gradient Descent (SGD), and its variant with momentum (2) Adam.

2.8.1 SGD

In practical implementation of neural networks, instead of calculating gradients of the loss function for the entire training set, we calculate the gradient, $\mathbf{g}$, of Equation 2.11 for each minibatch with N_{batch} samples:

$$\mathbf{g} = \nabla J(\theta) = \frac{1}{N_{\text{batch}}} \sum_i L(\hat{y}^{(i)}, y^{(i)}). \tag{2.17}$$

This is also known as Mini-batch SGD. We then update the parameters, θ, of the network as follows:

$$\theta \leftarrow \theta - \epsilon \mathbf{g}. \tag{2.18}$$

The learning rate, ϵ, defines the amount by which the parameters will change based on the calculated gradient i.e., the step size in the loss landscape. It is a hyper-parameter of the network and needs to be tuned using the validation set.

SGD as described above is often slow to converge. In order to accelerate optimisation, the concept of momentum is often used (Polyak, 1964). The SGD with momentum algorithm is shown in Algorithm 2. In this case, the parameter update depends not only on the gradients calculated for the current mini-batch, but it also takes into account previous gradients which are accumulated in a velocity vector and weighted in an exponentially decaying fashion. We calculate the velocity, $\mathbf{v}$,

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \epsilon\mathbf{g}, \tag{2.19}$$

and then perform parameter updates as follows:

$$\theta \leftarrow \theta + \mathbf{v}. \tag{2.20}$$

The decay coefficient, α, is a hyperparameter of the network that determines the rate of decay of previous gradients. The momentum term allows the optimiser to navigate through a steep descent quickly by adding up contributions of the previous gradients along the direction of steepest descent. It also controls the oscillation near a minima. Qian (1999) showed that the momentum parameter can be interpreted as a particle moving through viscous drag under a conservative force field in Newtonian dynamics. Instead of moving with a constant step size, the accumulated gradients increase the velocity of the particle where the loss surface is steep and the drag dampens the oscillation of the particle near a minima. Figure 2.4 shows learning curves for a model trained using SGD with momentum.

Algorithm 2 Minibatch Stochastic Gradient Descent with momentum Goodfellow et al. (2016)

Require: Learning rate ϵ, momentum parameter α
Require: Initial parameter vector $\boldsymbol{\theta}$
  initial velocity vector $\mathbf{v}$
  while θ not converged or stopping criterion not met **do**
    Sample minibatch of size m $\{\mathbf{x^{(1)}}, ..., \mathbf{x^{(m)}}\}$ with targets $\mathbf{y^{(i)}}$
    $\mathbf{g} \leftarrow \frac{1}{m}\sum_i \nabla L(\hat{y}^{(i)}, y^{(i)})$ ▷ compute gradient estimate
    $\mathbf{v} \leftarrow \alpha\mathbf{v} - \epsilon\mathbf{g}$ ▷ compute velocity estimate
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}$ ▷ Update parameters
  end while

2.8.2 Adam

The Adam algorithm (Kingma & Ba, 2014) makes use of an adaptive learning rate. It uses the first and second moments of the gradient to update the parameters. These moments are estimated using exponentially moving averages (EMAs) of the gradient and the square of the gradient, respectively. The decay rates of these EMAs are controlled by hyperparameters β_1 and β_2. Adam also corrects for the bias introduced in these estimates at the start of training due to initialisation of the moments with zeros. The steps of optimisation are shown in Algorithm 3.

Algorithm 3 Adam, (Kingma & Ba, 2014)

Require: Learning rate ϵ
Require: Exponential decay rates for moment estimates $\beta_1, \beta_2 \in [0, 1)$
Require: initial parameter vector $\boldsymbol{\theta}$
  initialise first moment vector $\mathbf{s_0} \leftarrow 0$
  initialise second moment vector $\mathbf{r_0} \leftarrow 0$
  initialise time step $t = 0$
  small constant for numerical stability δ
  while θ not converged or stopping criterion not met **do**
    $t \leftarrow t + 1$ ▷ update time step
    Sample minibatch of size m $\{\mathbf{x^{(1)}}, ..., \mathbf{x^{(m)}}\}$ with targets $\mathbf{y^{(i)}}$
    $\mathbf{g} \leftarrow \frac{1}{m} \sum_i \nabla L(\hat{y}^{(i)}, y^{(i)})$ ▷ compute gradient estimate
    $\mathbf{s} \leftarrow \beta_1 \mathbf{s} + (1 - \beta_1)\mathbf{g}$ ▷ compute biased first moment estimate
    $\mathbf{r} \leftarrow \beta_2 \mathbf{r} + (1 - \beta_2)\mathbf{g}^2$ ▷ compute biased second raw moment estimate
    $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \beta_1^t}$ ▷ compute bias-corrected first moment estimate
    $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \beta_2^t}$ ▷ compute bias-corrected second raw moment estimate
    $\boldsymbol{\Delta\theta} = -\epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}$ ▷ Compute update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta\theta}$ ▷ Update parameters
  end while

2.9 Regularisation

The goal of supervised learning is to learn from the training data in such a way that the model is able to generalise well to unseen data, to gain some insight about the true data generating distribution, $p_{\text{data}}(\mathbf{x}, \mathbf{y})$, from the trained model, $p_{\text{model}}(\mathbf{x}, \mathbf{y})$. If the model does not have enough capacity (depth and hidden units) to learn complex functions, it will not learn anything from the training data, i.e. it will underfit to the training data (high bias and low variance). The problem of underfitting can be easily solved by increasing the model capacity. On the other hand, if the model is very deep and has a

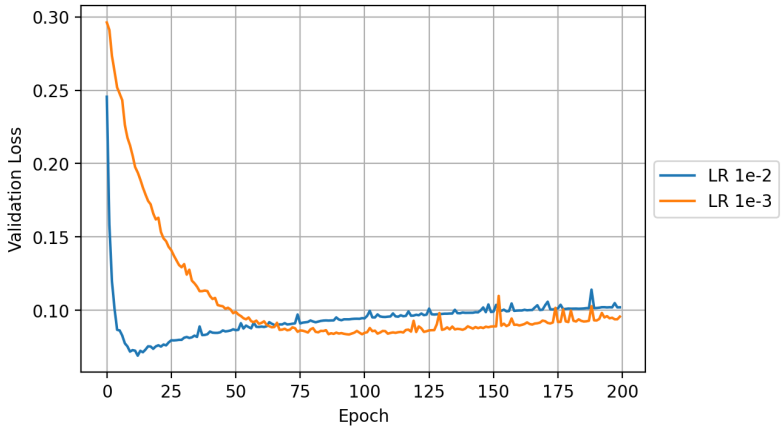


FIGURE 2.5: Validation loss curves showing overfitting: Validation loss curves for a MLP trained with SGD using two different values of the learning rate ($\epsilon = 10^{-2}, 10^{-3}$) and $\alpha = 0.9$ using the cross-entropy loss function. In both cases, the model overfits to the data. The effect of the learning rate can also be seen.

lot of hidden units, it is very easy for a neural network to start memorising the training data rather than learning patterns that will generalise to unseen data because the number of parameters in the network is far greater than the data samples (low bias and high variance). To prevent overfitting, there is a need to develop regularisation strategies that penalise the model and put constraints on its complexity. Overfitting can also be reduced by increasing the number of training samples, however this not always possible due to limited availability of data.

To check if the model is overfitting, we can monitor the validation loss curve. If at any point during training the validation loss has reached a minimum and starts increasing again, it is a sign that the model has started overfitting to the training data. An example is shown in Figure 2.5.

One way to reduce the complexity of the model is to encourage the model to learn smaller values of weights by penalising large values of weights. We describe two such methods which are known as L2 and L1 regularisation, in addition to some other commonly used regularisation methods.

1. **L2 /Weight Decay Regularisation:** We add a term proportional to the sum of the squared weights to the loss function:

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}) + \lambda \sum \mathbf{w}^2. \tag{2.21}$$

The hyperparameter λ is used to control the weight of the penalty term. (Krogh & Hertz, 1991)

2. **L1 Regularisation**: We add a term proportional to the sum of the absolute value of the weights to the loss function.

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}) + \lambda \sum |\mathbf{w}|. \tag{2.22}$$

L1 regularisation introduces sparsity in the network (weight values become zero).

3. **Early Stopping**: Early stopping is a cheap and efficient way to prevent the model from overfitting. It involves defining a criterion to monitor overfitting, using which the training algorithm is stopped automatically. One simple way to achieve this is to store the model parameters that correspond to the lowest validation error during training, rather than saving the model at the end of training. More complicated early stopping criteria may also be used (Prechelt, 1998).

4. **Dropout**: Dropout reduces the dependence of the model on particular neurons and connections. During training, we randomly set input and hidden units to zero according to a predefined probability p. This probability also needs to be tuned as a hyperparameter and different values of p can be chosen for different layers. It can be shown that dropout is equivalent to training an ensemble of models (Hinton et al., 2012; Srivastava et al., 2014).

In practice, a combination of these regularisation schemes is used.

2.10 Convolutional Neural Networks

In traditional Computer Vision, predefined filters or kernels are used to detect specific features in an image, followed by a classification algorithm for object detection. For example, the Canny and Sobel filters may be used for edge detection. LeCun & Bengio (1998) proposed Convolutional Neural Networks (CNNs) to automate the feature detection process by creating feature extractors which *learn* the values of the kernels.

CNNs are able to exploit the hierarchical structure of data with grid-like topology, such as 1D signals, 2D images and 3D volumetric data such as multi-channel images and

have now become the predominant neural network architecture used with images. Since our aim is to classify images of radio galaxies, we focus our discussion on images.

One of the key advantages of using CNNs over MLPs for image classification is that weight sharing allows the extracted features to be equivariant to translation. This significantly reduces the number of learnable parameters of the network and allows these networks to scale well to high resolution images. In case of an MLP, each pixel of the image is considered an input value, whereas for a CNN a few kernels with shared weights can extract multiple features from the whole image. For example, a 28x28 greyscale image with 784 input values fed into a fully-connected layer with a 100 units contains 78,400 weights. If we use a similar input with a CNN with convolutional kernel of size 5x5, the number of weights will be (28*28)*(5*5) = 17,500. Thus, we can see that for m inputs and n outputs, for a MLP the number of operations will be $m \times n$, while for a CNN it will be $m \times k$. See Table 2.2 and Table 2.3 for a comparison of the number of learnable parameters for a practical classification task. Shared weights also allow for some variability in the image preprocessing related to normalisation and centering.

In addition to the layers described in Section 2.3, we introduce two more layers to describe CNNs:

1. **Convolutional Layers:** The foundation of a CNN architecture is the convolutional layer, which performs a convolution between an input, I, and a kernel, K, to produce a feature map. The kernel consists of a set of learnable weights arranged in a 2D array. The kernel size is kept much smaller than the input size so that it can be applied to multiple overlapping regions of the image. This allows a particular feature to be learned from all regions in the image that contain it. Figure 2.6 illustrates a 2D convolution operation with a fixed kernel. In practice, convolutional layers perform the convolution operation on their inputs with multiple kernels to extract multiple features maps at the same location.

   We can express a 2D discrete convolution operation between the input, I, and filter, K, as follows:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{2.23}$$

In most deep learning libraries, the convolution operation is calculated using a cross-correlation, though it retains the same name:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n). \tag{2.24}$$

The convolution operation performs a linear transformation of the input. The values in a feature map are then passed through a non-linear activation function such as ReLU (Section 2.5).

2. **Pooling Layers:** The pooling layers are used perform local aggregation of features in a neighbourhood. Pooling effectively downsamples the feature maps to a lower resolution and makes the network invariant to small local distortions in the input. A max-pooling scheme which outputs the maximum value in the pooling kernel's receptive field is commonly used in CNNs. An example of max-pooling for a 2D feature map is shown in Figure 2.7 .

Two other parameters are used to control the shape of the output:

- Stride, S: The stride defines the amount by which a kernel will shift before performing the next convolution operation.
- Padding, P: The input volume may be padded with zeros to control the output dimension, as shown in Figure 2.6.

For an input volume with dimensions $W_1 \times H_1 \times D_1$, the convolution operation requires the following hyperparameters: number of kernels, K; kernel size, F; stride, S; and zero-padding, P, to produce an output volume of dimensions $W_2 \times H_2 \times D_2$ where:

- $W_2 = (W_1 - F + 2P)/S - 1$
- $H_2 = (H_1 - F + 2P)/S - 1$
- $D_2 = K$

The architecture of a typical CNN consists of several successive convolution and pooling layers to extract features from data. The initial layers extract low-level features such as lines and edges, which are then combined by the later layers to detect complex features

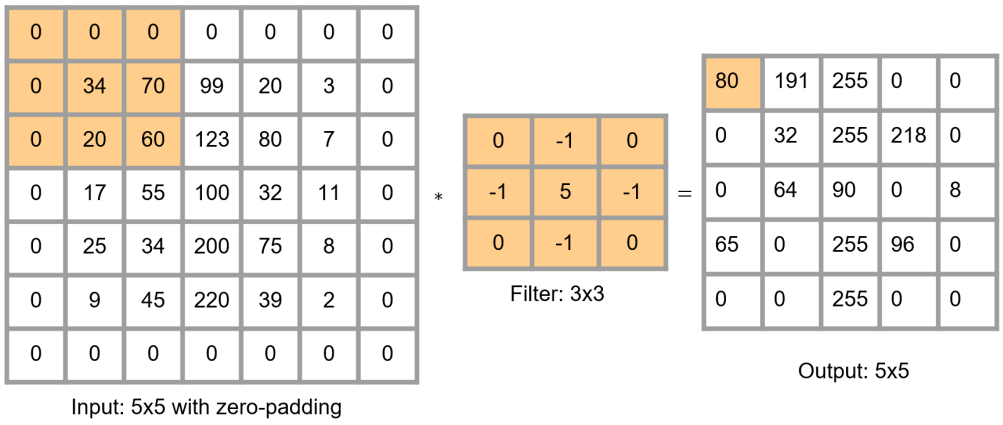


FIGURE 2.6: 2D convolution operation with 3x3 kernel

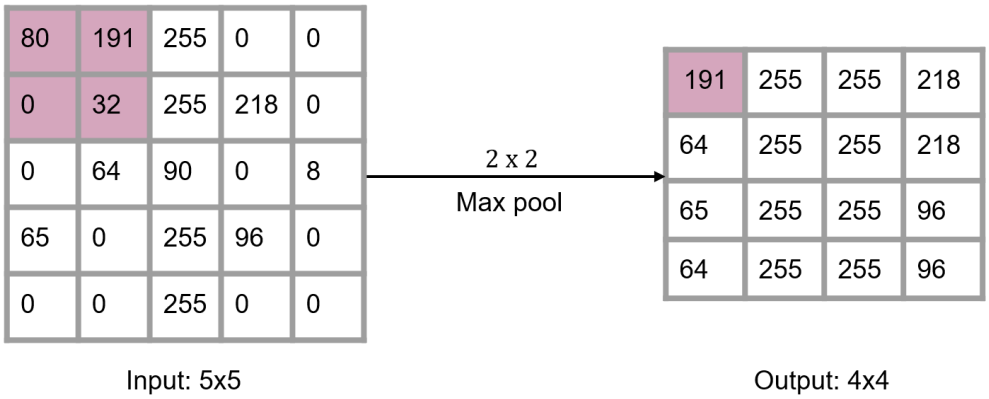


FIGURE 2.7: A 2x2 maxpooling operation

such as motifs and objects. Fully-connected layers are then used to make classifications using the extracted features.

An example of a CNN architecture known as LeNet-5 is shown in Figure 2.8. LeNet-5 was designed for handwritten digits and recognition of machine-printed characters (Lecun et al., 1998).

2.11 Classification experiments with MNIST

In this section, we use the MNIST dataset (LeCun & Cortes, 2010) to show how the concepts discussed in this chapter are implemented in practice and establish the baseline performance metrics for non-Bayesian models for MNIST. We use the specifications of the MLP architecture described in Blundell et al. (2015) and additionally implement the LeNet-5 CNN architecture to establish a baseline for Bayesian-CNN experiments.

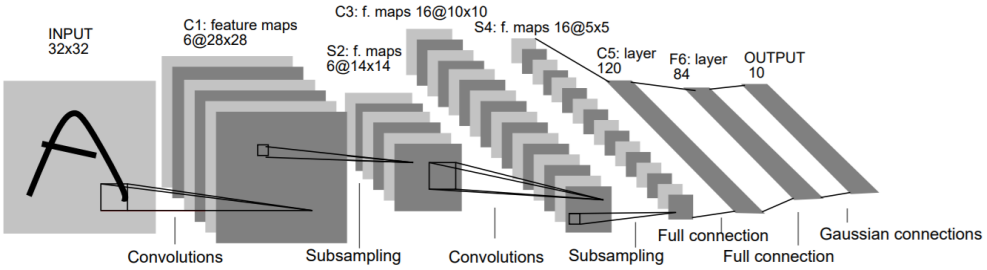


FIGURE 2.8: LeNet-5 Architecture (Lecun et al., 1998)

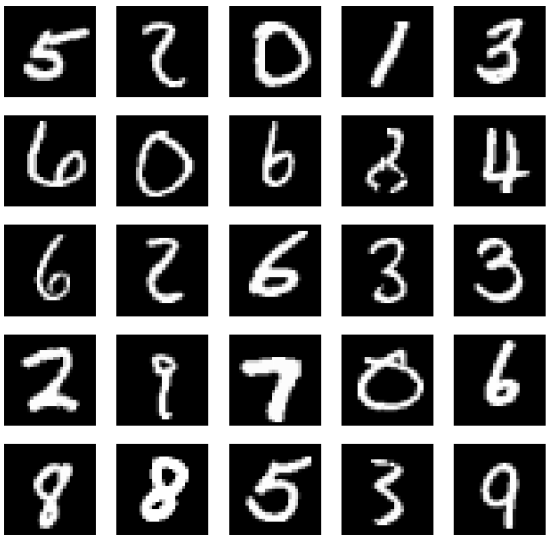


FIGURE 2.9: 25 randomly selected samples from the MNIST data set

Although we use the same architecture as Blundell et al. (2015), there are some differences in our implementation: (1) We use the Adam optimiser, whereas they use SGD. (2) They only use dropout for regularisation. We additionally use early stopping and implement weight decay with a very small decay factor ($\alpha = 10^{-5}$).

2.11.1 Data set and Image Preprocessing

The MNIST data set consists of 28x28 greyscale images of handwritten digits from 0-9. Of the 70, 000 images in the data set, we use 50, 000 for the training set, 10, 000 for the validation set and 10, 000 images are reserved for the test set. The images are normalised using the mean and standard deviation values computed for all the images in the MNIST data set ($\mu = 0.1307, \sigma = 0.3081$) to accelerate convergence (LeCun et al., 2012). Some examples from the dataset are shown in Figure 2.9.

TABLE 2.1: CNN Architecture: LeNet-5. Stride = 1 is used for all the convolutional and max pooling layers.

Operation	Kernel	Channels	Padding
Convolution	5 x 5	6	1
ReLU			
Max Pooling	2 x 2		
Convolution	5 x 5	16	1
ReLU			
Max Pooling	2 x 2		
Fully-Connected		120	
ReLU			
Fully-Connected		84	
ReLU			
Dropout, p = 0.5			
Fully-Connected		2	
Log Softmax			

2.11.2 Network Summary

1. MLP: We use a MLP with 2 fully-connected hidden layers with 800 units each and an output layer with 10 units, one for each of the digits. The weights are initialised using random values drawn from a normal distribution with $\mu = 0$ and $\sigma = 0.01$. The bias values are initialised with a constant value of 0. The hidden units use ReLU activations and log softmax activation is used for the output layer. Equation 2.16.

   We additionally use dropout with $p = 0.2$ before the input layer and $p = 0.5$ before both the hidden layers

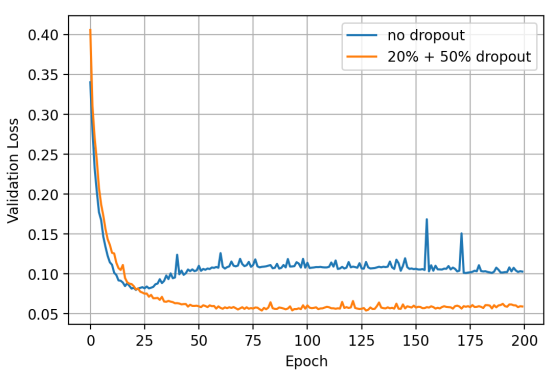
2. LeNet: We use a LeNet-5 style architecture as shown in Table 2.1, with a dropout layer with $p = 0.5$ before the last hidden layer.

2.11.3 Training

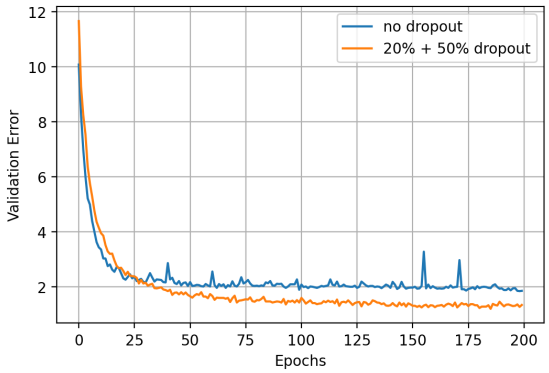
We use mini-batches of size 128 and train the models for 200 epochs. The negative log likelihood function is used to compute the loss. For both the architectures we use the Adam optimiser with learning rate $\epsilon = 5e-5$ and weight decay with $\alpha = 1e-5$ to optimise the parameters of the networks. These values were found by manual hyperparameter tuning. We also use an early stopping criterion based on validation error.

TABLE 2.2: Classification error on MNIST using MLP

Method	#Parameters	Test Error
no dropout	1,276,810	1.80%
20% + 50% dropout	1,276,810	1.30%



(A) Validation loss for MLP trained on MNIST



(B) Validation error for MLP trained on MNIST

FIGURE 2.10: Validation curves for MLP trained on MNIST

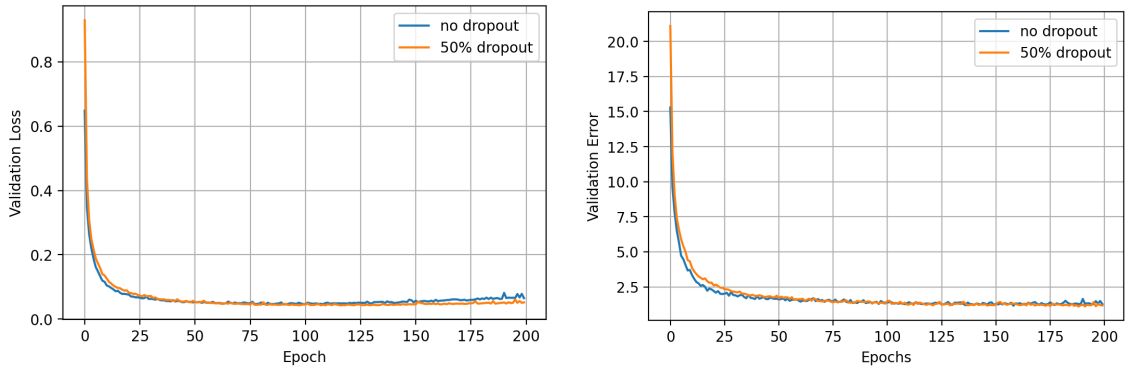
2.11.4 Results

The results of our experiments with MLP are shown in Table 2.2. We find that there is a significant improvement in the performance of the network when dropout is implemented. The validation loss curves shown in Figure 2.10a give us some insight into the difference in training the model with and without dropout. For the model without dropout, we observe that around the 25^{th} epoch the model reaches minimum validation loss and begins increasing after that. This is a classic sign of overfitting. We note that this overfitting happens after an extensive hyperparameter search and with two regularisation schemes (weight decay and early stopping) already implemented. From the validation error curves in Figure 2.10b we observe that even though learning with dropout happens at a slightly lower rate than without dropout, the validation error is smaller after the initial few epochs.

For the LeNet-5 architecture, we find that the models trained with and without dropout perform comparably as shown in Table 2.3. However, we must keep in mind that the training set for this experiment is large with 50,000 samples, and the similarity in performance might not translate to models trained with smaller datasets such as our catalogue of radio galaxies. Validation loss curves are shown in Figure 2.11a. A very slight overfitting effect can be seen for the model trained without dropout, though the validation error

TABLE 2.3: Classification error on MNIST using LeNet

Method	#Parameters	Test Error
no dropout	61,706	1.09%
50% dropout	61,706	1.11%



(A) Validation loss for MLP trained on LeNet **(B)** Validation error for LeNet trained on MNIST

FIGURE 2.11: Validation curves for LeNet-5 trained on MNIST

curves in Figure 2.11b are mostly overlapping. We also note that the learning curves of LeNet are smoother than that of the MLP.

2.12 Limitations of standard neural networks

Guo et al. (2017a) showed that even though modern deep learning models achieve state-of-the-art accuracy, these models are not well-calibrated i.e., there is an increasing gap between their accuracy and confidence, and these models are often over-confident in their predictions.

In addition to that, we have seen how neural networks produce deterministic outputs using maximum likelihood estimate of the parameters, which do not tell us anything about how uncertain the model is in its prediction. For scientific applications, and for radio galaxy classification specifically, if we want use the catalogues generated by deep learning models for research, we need to quantify the uncertainties in their outputs, which is difficult to do with deterministic parameters.

The results in the previous section also show that even with a lot of regularisation, the neural network models can overfit. These limitations of standard neural networks form the basis of our motivation for exploring Bayesian Neural Networks (Chapter 3) for classifying radio galaxies.

Chapter 3

Bayesian Deep Learning

Quantifying the confidence with which each object is assigned to a particular classification is crucial for understanding the propagation of uncertainties within astrophysical analysis of data classified using deep learning approaches. To provide uncertainties on model outputs, we need probabilistic methods such as Bayesian (and approximately Bayesian) neural networks (MacKay, 1992a,b). When properly calibrated, the uncertainty estimates from probabilistic methods can serve as a diagnostic tool to mitigate the effect of increasingly distant data points and out-of-distribution examples. This is particularly important for detecting new objects in future radio astronomy surveys. However, with the exception of Scaife & Porter (2021), to date there has been little work done on understanding the degree of confidence with which CNN models predict the class of individual radio galaxies. In this chapter we discuss how variational inference can be used to approximate Bayesian inference in neural networks using the *Bayes By Backprop* algorithm.

3.1 Bayesian Inference

To set up the problem of Bayesian inference, we consider a set of observations, $\mathbf{x}$, and a set of hypotheses, $\mathbf{z}$. For instance, $\mathbf{z}$ could be the parameters of a neural network model.

Bayesian modelling involves making *a priori* assumptions about the hypotheses of a model and computing a posterior distribution based on the prior distribution and the likelihood of the observed data. The prior distribution represents our beliefs about the hypotheses before any data has been observed. As the model sees more data, the posterior is updated using the Bayes rule. The posterior probability distribution, $p(z|x)$, can

then be used to make predictions about new data. Bayes rule gives us:

$$p(z|x) = \frac{p(z,x)}{p(x)} = \frac{p(x|z)\,p(z)}{p(x)}, \tag{3.1}$$

where $p(x|z)$ is the *likelihood* of the data given the hypothesis and quantifies how well the hypothesis fits to the data; $p(z)$ is the prior distribution; and the denominator, which is called the *evidence*, is the marginalised probability of the observations.

To compute the evidence, we need a solution to the following integral:

$$p(x) = \int p(z,x)\,\mathrm{d}z = \int p(x|z)\,p(z)\,\mathrm{d}z, \tag{3.2}$$

which is obtained by integrating over all possible values of z. This integral is often intractable either because there is no closed form solution available or the computation is exponential in time because it requires evaluating the integral for all possible values of z, which is very high dimensional. This in turn makes the posterior, $p(z|x)$, intractable.

Several techniques have been developed to perform approximate inference, including Markov Chain Monte Carlo (MCMC) methods and Variational Inference (VI), which are those most commonly used. MCMC algorithms allow us to sample from the exact posterior, but they do not scale well to large models such as deep neural networks (Blei et al., 2016). In this work we focus on VI and show how it is applied to neural network models.

3.2 Variational Inference

Variational inference was developed as a method to approximate densities that are difficult to compute directly (Peterson, 1987; Jordan et al., 1999; Wainwright & Jordan, 2008). In fact, the first application of VI to a specific class of models was done for neural networks (Peterson, 1987; Hinton & van Camp, 1993).

In variational inference we define a parameterised probability distribution, $q(z)$, as a variational approximation to the true posterior. The family of probability distributions, $\mathbb{D}$, defines the complexity of the solution that can be modelled. For instance, a family of Gaussians,

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{3.3}$$

parameterised by mean μ and variance σ^2 may be defined. The goal of VI is to find the selection of parameters which most closely approximates the exact posterior. The difference between the variational posterior, $q(z)$, and the exact posterior, $p(z|x)$, is typically measured by the KL divergence (Kullback & Leibler, 1951). Therefore, to find the optimal variational density from the family of densities, we solve the following optimisation problem:

$$q^*(z) = \arg\min_{q \in \mathbb{D}} \mathrm{KL}(q(z)||p(z|x)). \tag{3.4}$$

If we expand above equation using the formula for KL divergence:

$$\mathrm{KL}[a(x)||b(x)] = \int a(x) \log \frac{a(x)}{b(x)} \, \mathrm{d}x, \tag{3.5}$$

we get:

$$\begin{aligned} q^*(z) &= \arg\min_{q \in \mathbb{D}} \int q(z) \log \frac{q(z)}{p(z|x)} \, \mathrm{d}z \\ &= \arg\min_{q \in \mathbb{D}} \int q(z) \log q(z) \, \mathrm{d}z - \int q(z) \log p(z|x) \, \mathrm{d}z \\ &= \arg\min_{q \in \mathbb{D}} \mathbb{E}_{q(z)}[\log q(z)] - \mathbb{E}_{q(z)}[\log p(z|x)] \\ &= \arg\min_{q \in \mathbb{D}} \mathbb{E}_{q(z)}[\log q(z)] - \mathbb{E}_{q(z)}[\log p(z,x)] + \log p(x), \end{aligned} \tag{3.6}$$

where the expectation value is calculated according to:

$$\mathbb{E}_{b(x)}[a(x)] = \int a(x) b(x) \, \mathrm{dx}. \tag{3.7}$$

We see the dependence of minimising the KL divergence on the intractable integral $p(x)$ in Equation 3.6. Therefore, to find the optimal variational density $q^*(z)$, we only minimise the above equation up to an additive constant:

$$q^*(z) = \arg\min_{q \in \mathbb{D}} \mathbb{E}_{q(z)}[\log q(z)] - \mathbb{E}_{q(z)}[\log p(z,x)]. \tag{3.8}$$

Minimising the above function is equivalent to maximising the following objective function, which is formulated in the literature as the Evidence Lower Bound (ELBO) or

variational free energy (Saul et al., 1996; Neal & Hinton, 1998):

$$\begin{aligned} \text{ELBO}(q) &= \mathbb{E}_{q(z)} \log p(z, x) - \mathbb{E}_{q(z)}[\log q(z)] && (3.9) \\ &= \mathbb{E}_{q(z)} \log p(x|z) + \mathbb{E}_{q(z)} \log p(z) - \mathbb{E}_{q(z)}[\log q(z)] && (3.10) \\ &= \mathbb{E}_{q(z)} \log p(x|z) - \text{KL}(q(z)||p(z)). && (3.11) \end{aligned}$$

The name ELBO stems from the fact that the log evidence is bounded by this function such that: $\log p(x) \geq \text{ELBO}$. Equation 3.11 will be further scrutinised in the context of neural networks in the following sections.

Thus, we see how variational inference reduces Bayesian inference to an optimisation problem, which can then be solved by optimisation algorithms such as SGD and Adam (Section 2.8).

3.3 Variational Inference for Neural Networks

The notion of "noisy weights" that can adapt during training was first proposed by Hinton & van Camp (1993) to reduce the amount of information in network weights and prevent overfitting in neural networks. Graves (2011) developed a stochastic variational inference (SVI) method by applying stochastic gradient descent to VI using biased estimates of gradients. SVI allows VI to scale to large data sets by taking advantage of mini-batching (Section 2.4). Various choices of standard prior and posterior distributions such as the Delta function, Gaussian and Laplace distributions were also considered. Blundell et al. (2015) built on this work and proposed the *Bayes by Backprop* algorithm, which combines stochastic VI with the reparameterization trick (Kingma et al., 2015) to overcome the problems encountered while using backpropagation with SVI. Using this algorithm, we can calculate unbiased estimates of the gradients and use any tractable probability distribution to represent uncertainties in the weights.

In the equations described in the following sections, we only refer to the weights, w, of the network for simplicity, though the equations are applicable to the biases as well.

3.3.1 Deriving the Cost Function

We posit a family of distributions with parameters, θ, over the network parameters to define a variational approximation to the posterior, $q(w|\theta)$. Following the derivation described in the previous section, we find a member of the family that is closest to the true Bayesian posterior, $P(w|D)$, by minimising the following objective function:

$$\theta^* = \arg\min_\theta \mathrm{KL}[q(w|\theta)||P(w|D)], \tag{3.12}$$

where D denotes the training data. Using the formula for KL divergence given in Equation 3.5,

$$\begin{aligned}
\theta^* &= \arg\min_\theta \int q(w|\theta) \log \frac{q(w|\theta)}{P(w|D)} \,\mathrm{d}w \\
&= \arg\min_\theta \int q(w|\theta) \log \frac{q(w|\theta)}{P(w)P(D|w)} \,\mathrm{d}w \\
&= \arg\min_\theta \int q(w|\theta) [\log q(w|\theta) - \log P(w) - \log P(D|w)] \,\mathrm{d}w \qquad (3.13) \\
&= \arg\min_\theta \int q(w|\theta) [\log \frac{q(w|\theta)}{P(w)} \mathrm{d}w - \int q(w|\theta) \log P(D|w)] \,\mathrm{d}w \\
&= \arg\min_\theta \mathrm{KL}[q(w|\theta)|P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(D|w)]. \qquad (3.14)
\end{aligned}$$

Thus, we arrive at the cost function in Equation 3.14. This is composed of two components: the first term is a *complexity cost* which depends on the prior over the weights, $P(w)$, and the second is a *likelihood cost* which depends on the data and describes how well the model fits to the data. The cost function also has a minimum description length interpretation according to which the best model is the one that minimises the cost of describing the model and the misfit between the model and the data to a receiver (Hinton & van Camp, 1993; Graves, 2011).

More practically, the cost function used by Blundell et al. (2015) is given by Equation 3.13, which can be simplified as follows:

$$\begin{aligned}
\mathcal{F}(D,\theta) &= \int q(w|\theta) [\log q(w|\theta) - \log P(w) - \log P(D|w)] \,\mathrm{d}w \qquad (3.15) \\
&= \int q(w|\theta) f(w,\theta) \,\mathrm{d}w \qquad (3.16) \\
&= \mathbb{E}_{q(w|\theta)}[f(w,\theta)]. \qquad (3.17)
\end{aligned}$$

The cost, $\mathcal{F}$, is an expectation of the function $f(w, \theta)$ with respect to the variational posterior, $q(w|\theta)$. Minimising this cost is also computationally intractable. Consequently, the authors employ the reparameterization trick to obtain unbiased estimates of the cost function and its gradients.

3.3.2 Differentiating the Cost Function

In order to optimise the cost function, we need to calculate its gradient with respect to the variational parameters, θ. To make $\mathcal{F}(D, \theta)$ differentiable, the authors first employ the reparameterization trick to calculate samples from the variational posterior, $q(w|\theta)$, that are differentiable and then use Monte Carlo estimates of the gradients to approximate the cost function.

Reparameterization trick

A variation of this method was developed to be used with stochastic hidden units in variational autoencoders and was adopted by Blundell for the weights of a network (Kingma & Welling, 2013; Kingma et al., 2015) .

The reparameterization trick makes use of the change of variables technique to map between probability densities of random variables. We sample a random deviate from a known probability distribution and map it to a sample from the variational posterior through a differentiable deterministic function. This allows us to reparameterize samples from the variational posterior:

$$w \sim q(w|\theta), \tag{3.18}$$

through a differentiable deterministic function:

$$w = t(\epsilon, \theta), \tag{3.19}$$

where ϵ is a random deviate sampled from the distribution $\epsilon \sim q(\epsilon)$, such that:

$$q(w|\theta)\, \mathrm{d}w = q(\epsilon)\mathrm{d}\epsilon. \tag{3.20}$$

We then estimate the cost function by taking Monte Carlo samples from the variational posterior:

$$w^{(i)} \sim q(w^{(i)}|\theta) \tag{3.21}$$

such that Equation 3.15 becomes:

$$F(D,\theta) \approx \sum_{i=1}^{n} \log q(w^{(i)}|\theta) - \log P(w^{(i)}) - \log P(D|w^{(i)}). \tag{3.22}$$

In the following sections, each term of the cost function is discussed in detail: (i) variational approximation to the posterior, $\log q(w^{(i)}|\theta)$; (ii) the prior, $\log p(w^{(i)})$; and (iii) the likelihood, $\log P(D|w^{(i)})$.

Monte Carlo sampling is used to approximate sums and integrals that cannot be computed exactly by applying the following steps:

(i) Write the sum (integral), S, as an expectation over a probability distribution, $p(x)$:

$$S = \int p(x)f(x)dx = \mathbb{E}_{p(x)}[f(x)], \tag{3.23}$$

(ii) Approximate S by drawing n samples from $p(x)$ and calculate an empirical average:

$$\hat{S} = \frac{1}{n}\sum_{i=1}^{n} f(x). \tag{3.24}$$

Mini-Batching

To take advantage of mini-batch optimisation for Bayes by Backprop, we need to use a weighted complexity cost (Graves, 2011). This is because the likelihood is calculated for each mini-batch and then summed up for each epoch, whereas the complexity cost which involves calculating the prior and posterior over the entire network should be calculated only once per epoch.

The simplest weighting factor we could use is $M = N_{\text{batches}}$ and the cost function for the i^{th} mini-batch in Equation 3.22 becomes:

$$F_i(D_i,\theta) = \frac{1}{M}\text{KL}[q(w|\theta)||P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(D_i|w)]. \tag{3.25}$$

Several published results have reported a cold posterior effect which involves further down-weighting of the complexity cost. This effect is discussed in detail in Chapter 5.

Gradients

Blundell showed by proof that reparameterizing samples from the variational posterior makes the cost function, $\mathcal{F}(D, \theta)$, differentiable:

$$\begin{aligned}
\frac{\partial}{\partial \theta}\mathbb{E}_{q(w|\theta)}[f(w,\theta)] &= \frac{\partial}{\partial \theta}\int f(w,\theta)q(w|\theta)\,\mathrm{d}w && (3.26)\\
&= \frac{\partial}{\partial \theta}\int f(w,\theta)q(\epsilon)\,\mathrm{d}\epsilon && (3.27)\\
&= \int \frac{\partial f(w,\theta)}{\partial \theta}q(\epsilon)\,\mathrm{d}\epsilon && (3.28)\\
&= \mathbb{E}_{q(\epsilon)}\frac{\partial f(w,\theta)}{\partial \theta} && (3.29)\\
&= \mathbb{E}_{q(\epsilon)}[\frac{\partial f(w,\theta)}{\partial w}\frac{\partial w}{\partial \theta} + \frac{\partial f(w,\theta)}{\partial \theta}]. && (3.30)
\end{aligned}$$

Thus, the derivative of an expectation can be written as the expectation of a derivative, which makes the cost function differentiable using the standard backpropagation algorithms implemented for neural networks.

3.3.3 Variational Posteriors

The reparameterization trick allows us to use a variety of family densities for the variational distribution. Kingma & Welling (2013) give some examples of $q(w|\theta)$ for which the reparameterization trick can be applied:

1. Any tractable family of densities such as the Exponential, Logistic, Cauchy distributions.

2. Any location-scale family such as Gaussian, Laplace, Uniform densities can be used with the function: $t(.) = \text{location} + \text{scale} \cdot \epsilon$.

To illustrate this we can consider the case where the variational posterior $q(w|\theta)$ is parameterised by a family of Gaussians. In this case, the variational parameters will be $\theta = (\mu, \rho)$, where the standard deviation σ is parameterised as $\log(1 + \exp(\rho))$ so that it remains positive. To get a posterior sample, w, of the weight from the variational posterior:

1. Sample ϵ from a unitary Gaussian: $\epsilon \sim \mathcal{N}(0,1)$

2. Map ϵ to a Gaussian distribution $\mathcal{N}(\mu,\sigma)$ through the function:

$$t(\epsilon,\mu,\rho) = w = \mu + \epsilon.\log(1+\exp(\rho)), \tag{3.31}$$

   i.e., scale the random deviate by the standard deviation and shift it by the mean of the variational posterior.

We can then calculate the gradient of the cost function with respect to the variational parameters $\theta = (\mu,\rho)$:

$$\Delta_\mu = \frac{\partial f(w,\theta)}{\partial w}\frac{\partial w}{\partial \mu} + \frac{\partial f(w,\theta)}{\partial \mu} \tag{3.32}$$

$$= \frac{\partial f(w,\theta)}{\partial w} + \frac{\partial f(w,\theta)}{\partial \mu}, \tag{3.33}$$

and

$$\Delta_\rho = \frac{\partial f(w,\theta)}{\partial w}\frac{\partial w}{\partial \rho} + \frac{\partial f(w,\theta)}{\partial \rho} \tag{3.34}$$

$$= \frac{\partial f(w,\theta)}{\partial w}\frac{\epsilon}{1+\exp(-\epsilon)} + \frac{\partial f(w,\theta)}{\partial \rho}, \tag{3.35}$$

where $\frac{\partial w}{\partial \mu}$ and $\frac{\partial w}{\partial \rho}$ are calculated for the function $t(\epsilon,\mu,\rho)$. The variational parameters can then be updated using the standard optimisation algorithms that are used with neural networks:

$$\mu \leftarrow \mu - \alpha\Delta_\mu \tag{3.36}$$

$$\rho \leftarrow \rho - \alpha\Delta_\rho. \tag{3.37}$$

3.3.4 Priors

Priors reflect our beliefs about the distribution of weights before the model has seen any data. The simplest prior we could use is a Gaussian prior:

$$P(w) = \prod_j \mathcal{N}(w_j|0,\sigma), \tag{3.38}$$

which is often the default prior used with Bayesian NNs. We discuss the implications of choosing a simple prior such as this in Chapter 5.

Blundell suggest the use of a spike-and-slab-like Gaussian Mixture Model (GMM) prior defined over all the j weights in the network, to allow for a wide range of weight values to be learned:

$$P(w) = \prod_j \pi \mathcal{N}(w_j|0,\sigma_1) + (1-\pi)\mathcal{N}(w_j|0,\sigma_2) \tag{3.39}$$

$$\log P(w) = \sum_j \log(\pi \mathcal{N}(w_j|0,\sigma_1) + (1-\pi)\mathcal{N}(w_j|0,\sigma_2), \tag{3.40}$$

where $\sigma_1 > \sigma_2$ and $\sigma_2 << 1$. The weight of each component in the mixture is defined by π. These parameters are chosen by comparing the model performance on the validation set, like the hyperparameters of the network.

Some regularisation techniques used with point-estimate neural networks have theoretical justifications using Bayesian inference. For instance, it can be shown that *Maximum a Posteriori* (MAP) estimation of NNs with some priors is equivalent to regularisation (Jospin et al., 2020). For example, using a Gaussian prior over the weights is equivalent to weight decay regularisation, whereas using a Laplace prior induces L1 regularisation (Section 2.9):

$$w^{\text{MAP}} = \arg\max_w \log P(D|w) + \log P(w). \tag{3.41}$$

Gal & Ghahramani (2015) showed that dropout can also be considered an approximation to variational inference, where the variational family is a Bernoulli distribution.

3.3.5 Likelihood

The likelihood can be calculated using the negative log likelihood loss descibed in Equation 2.16.

3.3.6 Bayesian Convolutional Neural Networks

We extend the Bayes by Backprop algorithm for CNNs by sampling the weights from a variational distribution defined over the shared weights of the convolutional kernels. This is followed by fully-connected layers that have weights with a variational distribution defined over them. Our implementation differs from that for Bayesian CNNs

proposed by Shridhar et al. (2019), where the activations of each convolutional layer are sampled instead of the weights to accelerate convergence.

3.3.7 Posterior Predictive Distribution

After the variational posterior distribution has been learned by optimising the ELBO function, we can use it to predict the labels of new observations, D^*, using the posterior predictive distribution, $q(D^*|D)$. The variational posterior distribution conditioned on training data D, $q(w|D)$, can be used to calculate this distribution by integrating out the variational parameters:

$$\begin{aligned} q(D^*|D) &= \int q(D^*, w|D)\,\mathrm{d}w && (3.42)\\ &= \int q(D^*|w, D)q(w|D)\,\mathrm{d}w && (3.43)\\ &= \int q(D^*|w)q(w|D)\,\mathrm{d}w, && (3.44) \end{aligned}$$

where $q(D^*|w, D) = p(D^*|w)$ because for a given w, all data is conditionally independent (i.i.d. assumptions). From Equation 3.44, we can see how the posterior predictive distribution is an average of all possible variational parameters weighted by their posterior probability.

The posterior predictive distribution can be estimated using MC samples as follows:

1. Sample variational parameters from the variational posterior distribution: $w^{(i)} \sim q(w|D)$.
2. Sample prediction $D^{*(i)}$ from $q(D^*|w^{(i)})$.
3. Repeat steps 1 and 2 to construct an approximation to $q(D^*|D)$ using N samples:

$$\begin{aligned} q(D^*|D) &= \mathbb{E}_{q(w|D)} q(D^*|w) && (3.45)\\ &= \mathbb{E}_{w^{(i)}\sim q(w|D)} q(D^*|w^{(i)}) && (3.46)\\ &\approx \frac{1}{N}\sum_{i=1}^{N} q(D^*|w^{(i)}) && (3.47) \end{aligned}$$

Thus using BBB allows us to construct an approximate posterior predictive distribution, which can then be used to estimate uncertainties (Section 4.2).

3.4 Network Pruning

Variational inference based neural networks have several advantages over non-Bayesian NNs; however, for a typical variational posterior such as the Gaussian distribution, the number of parameters in the network double compared to a non-Bayesian model with the same architecture because both the mean and standard deviation values need to be learned. This increases the computational and memory overhead at test time and during deployment. Thus, there is a need to develop network pruning approaches, using which we can remove the parameters that contain less information. Several authors have also considered pruning to improve the generalisation performance of the network (LeCun et al., 1990). Many of the pruning methods that have been developed can also be applied to non-Bayesian NNs, but in this section we discuss a Signal-to-Noise (SNR) based pruning criterion which can be applied naturally to models trained with Gaussian variational densities (Graves, 2011; Blundell et al., 2015).

We calculate the SNR of a model weight as follows:

$$\text{SNR} = |\mu|/\sigma, \tag{3.48}$$

where μ and σ are the values of the variational parameters after a model has been trained. In practice, we use the SNR values in dB:

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \text{SNR}. \tag{3.49}$$

Based on a threshold value between (0, 1), the weights of the network with the lowest SNR values are removed. The effect of removing these parameters is measured by re-calculating the test error using the pruned model.

While the SNR-based method may seem very simple, it allows for a large proportion of weights to be removed for some models. We also consider an alternative pruning approach based on Fisher information in Chapter 5.

3.5 Experiments with MNIST

We use the MNIST data set and apply the training: validation: test split and image preprocessing as described in Section 2.11.1. We use the same architecture for the BBB model as Blundell, but our implementation differs in the following ways: (i) We reparameterize $\sigma = \exp(\rho)$ instead of $\sigma = \log(1 + \exp \rho)$ as they have done. (ii) We use a simple weighting scheme for the complexity cost where $M = N_{\text{batches}}$ (Equation 3.25), whereas they have used a more complicated weighting scheme. No posterior initialisation values were specified by Blundell; however, we note that we found the posterior initialisations to have a significant effect on model convergence and performance. The values are chosen like a hyperparameter using the validation set.

3.5.1 Network Summary

The MLP architecture as described in the previous chapter is used (Section 2.11), without any dropout. For the Bayesian MLP, we consider two different priors:

1. **Gaussian prior:** The following values were considered:
   - $\sigma = \{1, \exp(-1), \exp(-2)\}$
2. **Gaussian Mixture Model (GMM) prior:** The following values were considered, as suggested by Blundell:
   - $\sigma_1 = \{\exp(0), \exp(-1), \exp(-2)\}$
   - $\sigma_2 = \{\exp(-6), \exp(-7), \exp(-8)\}$
   - $\pi = \{1/4, 1/2, 3/4\}$

The results discussed are for the best model found through the validation set, specifications of which are given below.

1. **Prior Specification:**
   - Gaussian Prior, $\mu = 0, \sigma = \exp(-1)$
   - Gaussian Mixture Model Prior, $\{3/4, \exp(-1), \exp(-7)\}$
2. **Posterior Initialisation**

TABLE 3.1: Classification error on MNIST using BBB

Method	#Parameters	Test Error
Gaussian Prior	1,276,810 x 2	$1.95 \pm 0.08\%$
GMM Prior	1,276,810 x 2	$1.91 \pm 0.06\%$

- w_{mu} and b_{mu} are initialised from the uniform distribution $U(-0.1, 0.1)$
- w_{rho} and b_{rho} are initialised from the uniform distribution $U(-5, -4)$, which corresponds to σ values being initialised from $U(0.0067, 0.0183)$

3. **Likelihood:** The Negative Log Likelihood loss, see Equation 2.16, is used to calculate the log likelihood.

3.5.2 Training

The SGD with momentum optimiser is used with an initial learning rate $\epsilon = 10^{-4}$ and momentum $\alpha = 0.9$. In addition, a learning rate scheduler is used that reduces the learning rate by 95% if the negative log likelihood does not improve for four consecutive epochs. Here we have not considered the total loss as a criterion for reducing the learning rate because we found it necessary to prevent the model from overfitting. An early stopping criterion based on validation accuracy is used.

Minibatches of size 128 are used. We train the Gaussian prior model for 300 epochs and the GMM prior model for 600 epochs.

The validation metrics are evaluated like test metrics. The model with the best validation accuracy up to that point in training is used to calculate the validation loss and error. This is contrast to how we used the validation set for the non-Bayesian NN, where the model corresponding to the active epoch is used to evaluate the validation metrics. We found this approach more useful for finding the optimal hyperparameters and debugging as this makes the metrics less stochastic.

3.5.3 Results

Classification

The results of our classification experiments with the Bayes by Backprop algorithm on MNIST are shown in Table 3.1. The test error is calculated by making 20 forward passes through the trained network to obtain the mean and standard deviation values.

We find that the models trained with Gaussian and GMM priors perform comparably. However, in the following section on weight pruning we will see how the GMM priors allow for a higher proportion of weights to be removed.

For the model trained with a Gaussian prior, the test error is comparable to the value achieved by Blundell (1.99%). However, for the model trained with a GMM prior, we report a higher value of the test error than Blundell (1.34%). This discrepancy could be due to the differences in our implementation. We also note that other implementations of BBB have also found it difficult to achieve the same performance as reported by Blundell.

BBB is able to classify the test set within 0.2% of the Non-Bayesian MLP model trained without dropout (Table 2.2); while providing uncertainties in its prediction.

Figure 3.1 shows the training loss curves for the model trained with a GMM prior. We see how the total loss is composed of two components: the likelihood cost and the complexity cost (Equation 3.25). The model optimises for the likelihood cost quickly, which can be been by the convergence in the likelihood cost plot. After that, the model continues to optimise for the complexity cost, which converges around the 500^{th} epoch. The corresponding validation loss curves are shown in Figure 3.2. The steps in the graph are due to the difference in how the validation loop is evaluated, as explained in the previous sub-section. We note that even though the training likelihood has reached a minimum very early during training, the optimisation of the complexity cost has an effect on the validation likelihood at a later stage during training as well. The total validation loss values converge after around 350 epochs, which suggests that the incremental improvement in the training after that epoch does not have a significant effect on the validation performance.

In Figure 3.3, the evolution of the density of variational parameter μ, i.e., the mean of the network weights, is shown for the model trained with a Gaussian prior. At epoch $= 0$, we can see the density of has a uniform shape, as specified by the posterior initialisation. As training progresses, the network learns smaller weight values.

SNR based Weight Pruning

For a given threshold, k, our implementation of SNR-based pruning involves the following steps:

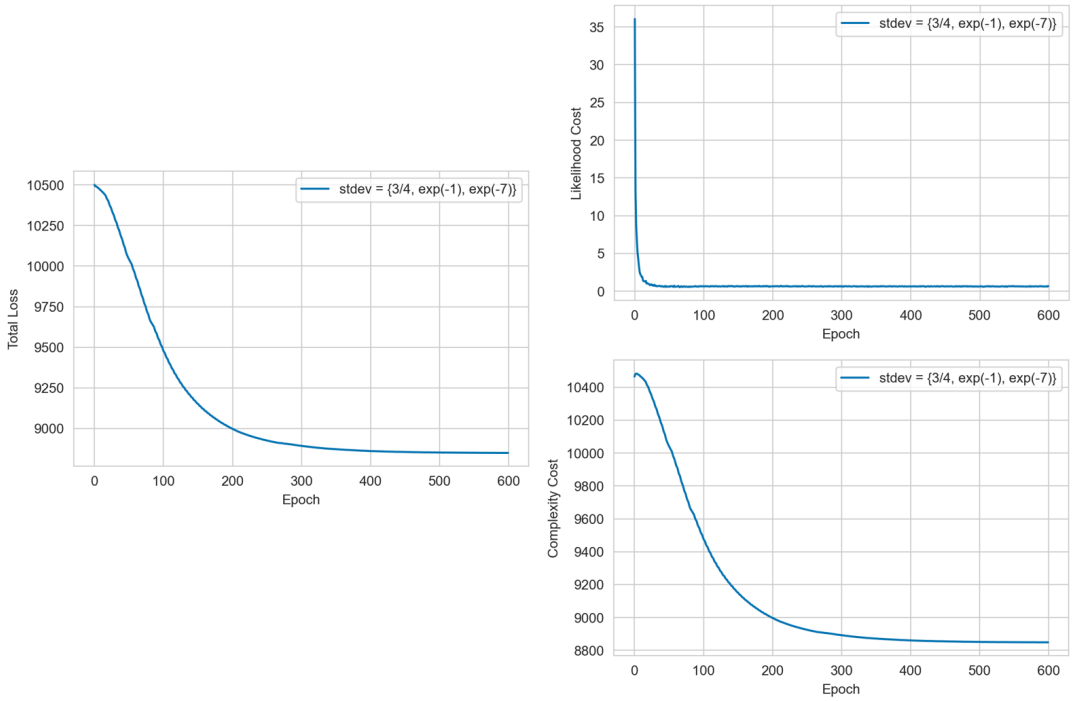


FIGURE 3.1: Training Loss Curves for BBB on MNIST using a GMM Prior: The plot on the left shows the total loss which is composed of two components: the likelihood cost and the complexity cost, which are shown on the right.

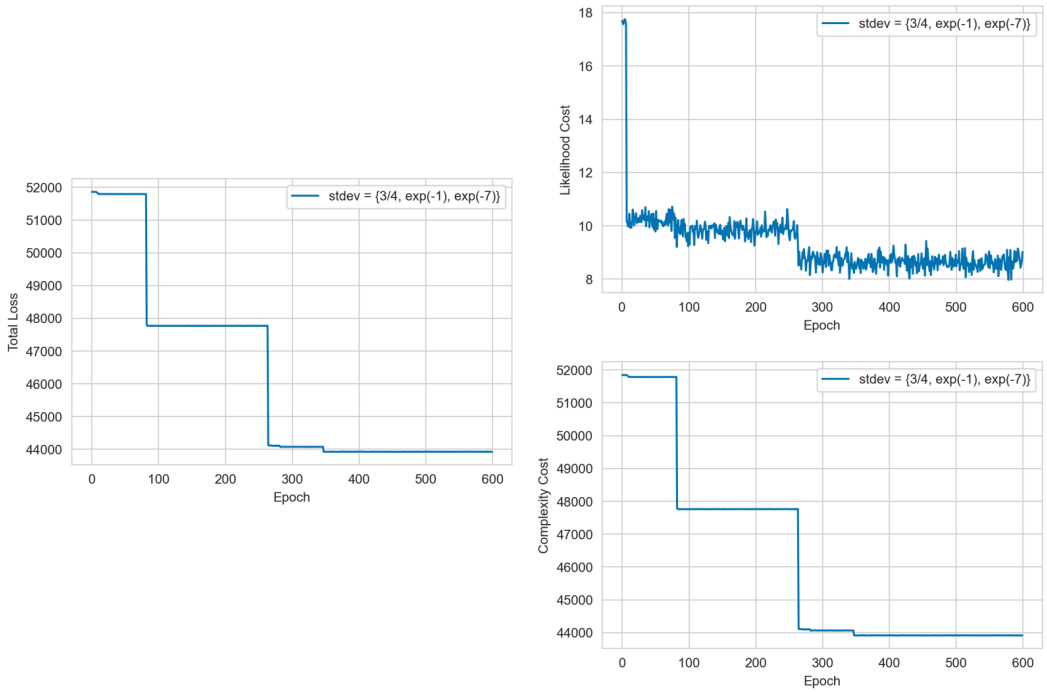


FIGURE 3.2: Validation Loss Curves for BBB on MNIST using a GMM Prior. The steps in the plots are due to the difference in how we have evaluated the validation metrics for Bayesian and non-Bayesian models.

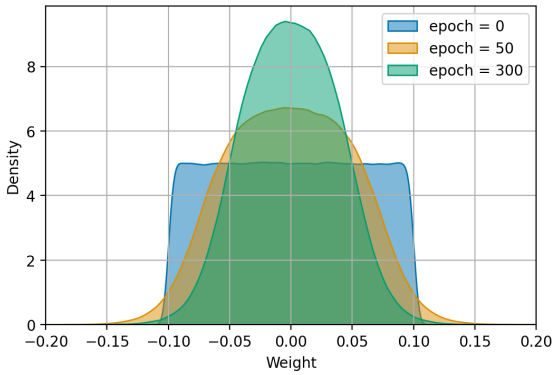
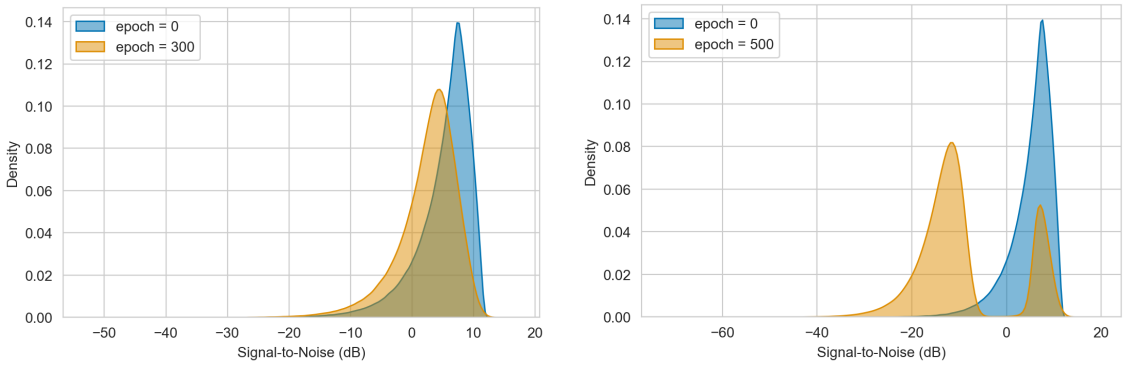


FIGURE 3.3: Evolution of the density of variational parameters μ i.e., the mean of the weights, for a model trained with a Gaussian prior.



(A) Evolution of SNR density for the model trained with a Gaussian prior.

(B) Evolution of SNR density for the model trained with a GMM prior.

FIGURE 3.4: Evolution of SNR density

1. Calculate the SNR in dB according to Equation 3.49 and sort the SNR values in ascending order.

2. Calculate SNR threshold by extracting the index of the SNR value corresponding to: $j * t$, where j is the number of weights in the network.

3. Remove all weights with SNR values below the SNR value calculated for the given threshold

4. Calculate test error by making 20 forward passes through the network.

In order to build intuition about SNR-based pruning, we plot the density of SNR values and show how the density evolves during training in Figure 3.4a for a model trained with a Gaussian prior and in Figure 3.4b for a model trained with a GMM prior. We note that at epoch = 0, density for both the models looks similar. The proportion of weights

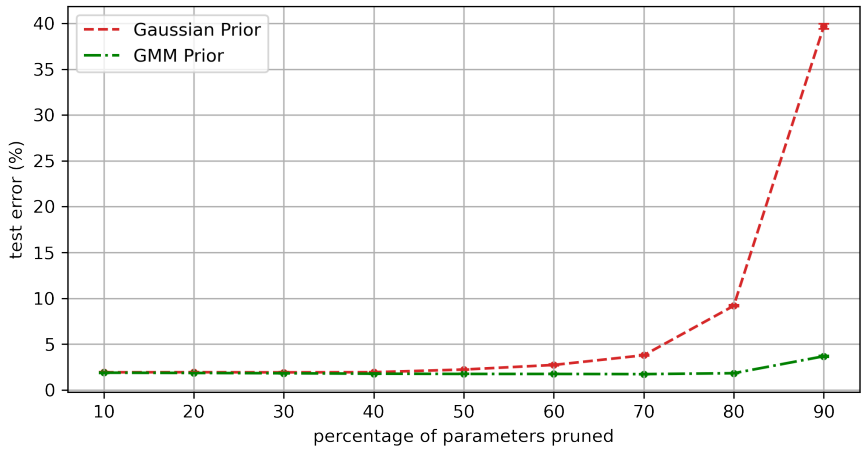
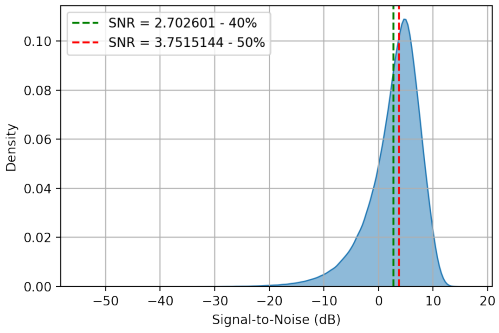


FIGURE 3.5: Test error for different pruning thresholds for models trained with Gaussian and GMM priors. Using a GMM prior allows for a higher proportion of weights to be removed (up to 70%) without affecting the model performance. The standard deviation values corresponding to the test errors are very small, which is why the markers have collapsed into a blob on the plot.

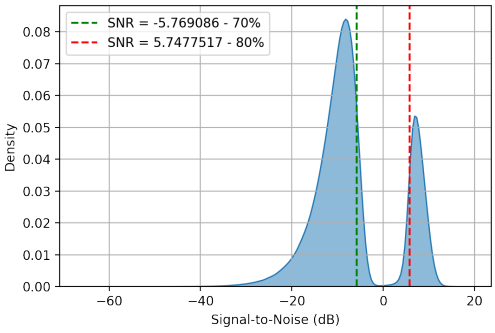
with higher SNR reduces as a function of epoch. This could mean that: (i) smaller values of the mean, μ, of the weights are being learned, which is good for reducing the complexity of the model (ii) values of ρ corresponding to higher standard deviation values are being learned, which could be interpreted as the model learning which weights are less important.

The choice of prior directly influences the shape of the SNR density and the proportion of weights that can be pruned without affecting model performance. The test error for different pruning thresholds for both the models is shown in Figure 3.5. For a single Gaussian prior, we find that up to 40% of the weights can be pruned, whereas for a GMM model 70 % of the weights can be removed.

We observe two well-separated peaks in the SNR density for the GMM prior, see Figure 3.6b. The pruning threshold of 70% corresponds to removing the peak with the low SNR values. Pruning the weights penalizes test performance more for the model trained with a single Gaussian prior since there exists a higher proportion of weights with higher SNR, as shown in Figure 3.6a. Our analysis of pruning results is consistent with the results of Blundell, who note that a model trained with a GMM prior allows a wide range of weight values to be learned, most of which can be successfully pruned without affecting the model performance.



(A) For the model trained with a Gaussian prior.



(B) For the model trained with a GMM prior.

FIGURE 3.6: SNR density with pruning thresholds

Chapter 4

Bayesian Classification of Radio Galaxies

4.1 MiraBest Data Set

The MiraBest data set (Porter, 2020) consists of 1256 images of radio galaxies pre-processed to be used specifically for deep learning tasks. The data set was constructed using the sample selection and classification described in Miraghaei & Best (2017), who made use of the parent galaxy sample from Best & Heckman (2012) and presented a sample of 18286 radio-loud active galactic nuclei (AGN). Optical data from data release 7 of Sloan Digital Sky Survey (SDSS DR7; Abazajian et al., 2009) was cross-matched with NRAO VLA Sky Survey (NVSS; Condon et al., 1998) and Faint Images of the Radio Sky at Twenty-Centimeters (FIRST; Becker et al., 1995) radio surveys using the techniques described in Best et al. (2005) and Donoso et al. (2009). The radio surveys were conducted at 1.4 GHz. A lower redshift cut of $z > 0.03$ was applied because of the large angular size of nearby sources. Only those objects were considered that were within the SDSS 'main galaxy' or 'luminous red galaxy' samples. A 40 mJy flux density cut-off was applied so that there would be sufficient SNR in any extended structures for morphological classification.

Parent galaxies were selected such that their radio counterparts had an active galactic nucleus (AGN) host rather than emission dominated by star formation. A combination of three methods was used to distinguish radio loud AGN from SF galaxies (Best & Heckman, 2012). The first method used the 4000 Å break strength and radio luminosity to stellar mass ratio (L_{rad}/M), since both of these factors are affected by the star formation

rate of the galaxy. Radio-loud AGN are separable on the plane of D_{4000} vs $L_{\rm rad}/M$ since they have higher values of $L_{\rm rad}$. Upto 83% of the sources in the parent sample could be classified using this method. The second method was based on emission line diagnostics since radio loud AGN and SF galaxies have different ionising radiation. Upto 30% of the sample was classifiable using this method. The third method made use of the Hα emission line luminosity and radio luminosity since radio loud AGN are expected to have higher $L_{\rm rad}$ to Hα ratio. Upto 80% of the sample was classifiable using this method. The final classification was done by examining all of the 27 possible classes generated by the results of these three methods. To enable classification of sources based on morphology, sources with multiple components in either of the radio catalogues were considered.

The morphological classification was done by visual inspection at three levels: (i) The sources were first classified as FRI/FRII based on the original classification scheme of Fanaroff & Riley (1974). Additionally, 35 *Hybrid* sources were identified as sources having FRI-like morphology on one side and FRII-like on the other. Of the 1329 extended sources inspected, 40 were determined to be unclassifiable. (ii) Each source was then flagged as 'Confident' or 'Uncertain' to represent the degree of belief in the human classification. (iii) Some of the sources which did not fit exactly into the standard FRI/FRII dichotomy were given additional tags to identify their sub-type. These sub-types include 53 Wide Angle Tail (WAT), 9 Head Tail (HT) and 5 Double-Double (DD) sources. To represent these three levels of classification, each source was given a three digit identifier as shown in Table 4.1.

To construct the machine learning data set, several pre-processing steps were applied to the data following the approach described in Aniyan & Thorat (2017) and Tang et al. (2019):

1. In order to minimise the background noise in the images, all pixels below the 3σ level of the background noise were set to 0. This threshold was chosen because among the classifiers trained by Aniyan & Thorat (2017) on images with 2σ, 3σ, and 5σ cut-offs, 3σ performed most well.

2. The images were clipped to 150×150 pixels, centered on the source.

3. All pixels outside the largest angular size of the radio galaxy were set to zero.

4. The images were normalised as follows:

$$\text{Output} = 255 \ \frac{\text{Input} - \text{Input}_{\text{min}}}{\text{Input}_{\text{max}} - \text{Input}_{\text{min}}} \,, \tag{4.1}$$

where Input refers to the input image, $\text{Input}_{\text{min}}$ and $\text{Input}_{\text{max}}$ are the minimum and maximum pixel values in the input image, and Output is the image after normalisation.

To ensure the integrity of the ML data set, the following 73 objects out of the 1329 extended sources identified in the catalogue were not included: (i) 40 unclassifiable objects; (ii) 28 objects with extent greater than the chosen image size of 150×150 pixels (iii) 4 objects which were found in overlapping regions of the FIRST survey (iv) 1 object in category 103 (FRI Confident Diffuse). Since this was the only instance of this category, it would not have been possible for the test set to be representative of the training set. The composition of the final data set is shown in Table 4.2. We do not include the sub-types in this table as we have not considered their classification.

In this work, we use the MiraBest Confident subset to train the BBB models. Examples of FRI and FRII galaxies from the MiraBest confident data set are shown in Figure 4.1.

Additionally, we use 49 samples from the MiraBest Uncertain subset to test the trained model's ability to correctly represent epistemic uncertainty, since these samples can be considered as being drawn from the same data generating distribution as the MiraBest Confident samples, but have a lower degree of belief in their classification. We also use 30 samples from the MiraBest Hybrid class to test the model's ability to quantify aleatoric uncertainty. The MiraBest Hybrid samples are a separate class of objects that was not included in the training and therefore might be denoted as being out-of-distribution by some measures; however, given that they are still a sub-population of the over-all radio galaxy population, and that they are defined as amalgams of the two classes used to train the model, they could also be considered to be in-distribution. Consequently, in this work we treat the MiraBest Hybrid test sample as being in-distribution. We note that there may be components of either type of uncertainty in the Uncertain and Hybrid samples, this is discussed in Section 4.5.3.

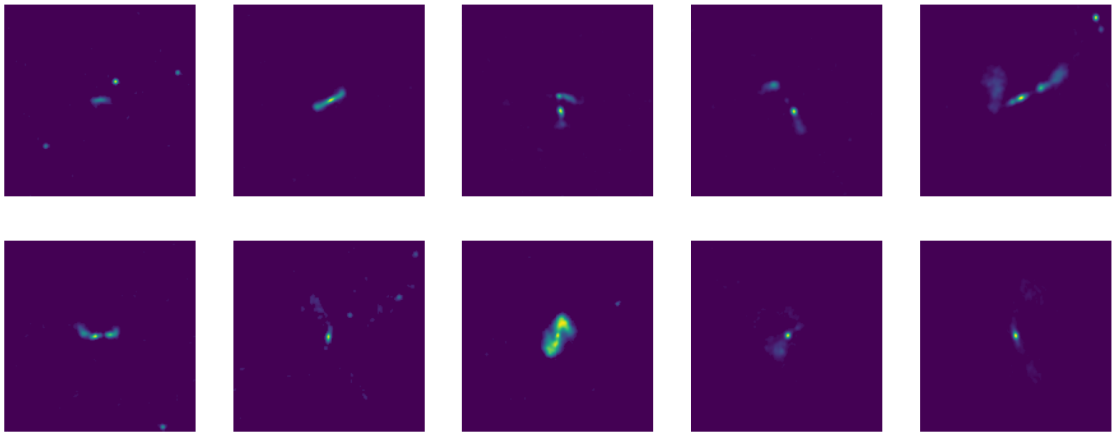
We note that all the previous work published using this data set uses some form of data augmentation, whereas we do not use any data augmentation. This allows us to

TABLE 4.1: Three digit identifiers for sources in Miraghaei & Best (2017).

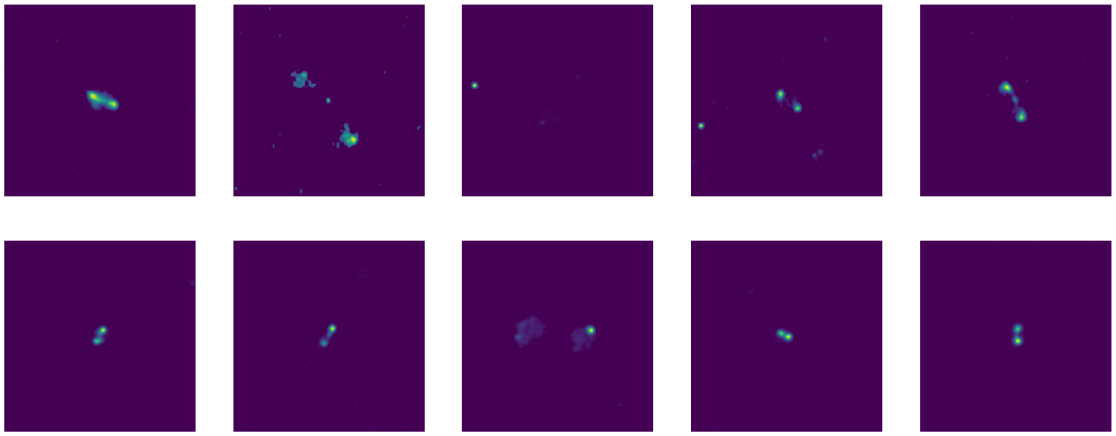
Digit 1	Digit 2	Digit 3
1: FRI	0: Confident	0: Standard
2: FRII	1: Uncertain	1: Double Double
3: Hybrid		2: Wide Angle Tail
4: Unclassifiable		3: Diffuse
		4: Head Tail

TABLE 4.2: MiraBest Class-wise Composition

Class	Confidence	No.
FRI	Confident	397
	Uncertain	194
FRII	Confident	436
	Uncertain	195
Hybrid	Confident	19
	Uncertain	15



(A) Examples of FRI galaxies from the MiraBest confident subset



(B) Examples of FRII galaxies from the MiraBest confident subset

FIGURE 4.1: MiraBest Confident data set

study the application of BBB on small data sets.

4.2 Uncertainty Quantification

We can broadly divide the sources of uncertainty in the predictions of neural network models into two categories: epistemic and aleatoric. Epistemic uncertainty quantifies how uncertain the model is in its predictions and this can be reduced with more data. Aleatoric uncertainty on the other hand represents the uncertainty inherent in the data and cannot be reduced. Uncertainty inherent in the input data along with model uncertainty is propagated to the output, which gives us predictive uncertainty (Abdar et al., 2021). BBB allows us to capture model uncertainty by defining distributions over model parameters.

Using Monte Carlo samples obtained from the posterior predictive distribution, we obtain N Softmax probabilities for each class c in the data set. Adapting Equation 3.47 for our supervised classification setting, we can recover N class-wise Softmax probabilities as follows:

$$q(y|x, D) = \frac{1}{N} \sum_{i=1}^{N} q(y = c|x, w^{(i)}), \tag{4.2}$$

where (x, y) are samples from the test set and D is the training data. Using these samples, we can quantify the uncertainties in the predictions using the metrics defined in the following subsections.

4.2.1 Predictive Entropy

The predictive entropy has combined contributions from both epistemic and aleatoric uncertainties. It is a measure of the average amount of information inherent in the distribution and is defined as:

$$\mathbb{H}(y|x, D) = -\sum_{c} q(y = c|x, w) \log q(y = c|x, w), \tag{4.3}$$

and can be approximated using MC samples as (Gal, 2016):

$$\mathbb{H}(y|x, D) = -\sum_{c} \left(\frac{1}{N} \sum_{i=1}^{N} q(y = c|x, w^{(i)}) \right) \log \left(\frac{1}{N} \sum_{i=1}^{N} q(y = c|x, w^{(i)}) \right). \tag{4.4}$$

We use the natural logarithm for all the equations described in this section. The entropy thus attains a maximum value of ~ 0.693 when the predictive entropy is maximum

and a minimum value close to zero.

4.2.2 Mutual Information

We use mutual information to quantify epistemic uncertainty. Mutual information is closely related to the entropy and can be calculated as follows:

$$\mathbb{I}(y, w|x, D) = \mathbb{H}(y|x, D) - \mathbb{E}_{q(w|D)}[\mathbb{H}(y|x, w)], \tag{4.5}$$

which can be approximated as (Gal, 2016):

$$\begin{aligned}\mathbb{I}(y, w|x, D) \quad = \quad & -\sum_c \left(\frac{1}{N}\sum_{i=1}^{N} q(y = c|x, w^{(i)})\right) \log\left(\frac{1}{N}\sum_{i=1}^{N} q(y = c|x, w^{(i)})\right) \\ & + \frac{1}{N}\sum_{c,N} q(y = c|x, w^{(i)}) \log q(y = c|x, w^{(i)}).\end{aligned} \tag{4.6}$$

4.2.3 Entropy of a single pass

We use the entropy of a single pass to capture aleatoric uncertainty associated with the data (Mukhoti et al., 2021):

$$\mathbb{H}(y|x, D) = -\sum_c q(y = c|x, w^{(i)}) \log q(y = c|x, w^{(i)}). \tag{4.7}$$

4.2.4 Overlap Index

We define two overlap indices: η_1, to quantify how much the distributions of predicted Softmax values for the two classes overlap; and η_2, to quantify how much the distributions of logits for the two classes overlap. A higher degree of overlap indicates a higher level of predictive uncertainty. The overlap parameters have contributions from both epistemic and aleatoric uncertainties.

Logits are the unnormalised outputs of the network, i.e., the outputs of the final layer before the Softmax function is applied. A distribution of logit values can be obtained in a manner similar to how MC samples of the Softmax distribution are obtained.

We calculate a distribution free overlap index as follows (Pastore & Calcagnì, 2019; Scaife & Porter, 2021):

1. For each class, estimate local density at location z using a Gaussian kernel density estimator:

$$f_{c_1}(z) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\beta\sqrt{2\pi}}\,\mathrm{e}^{-(z-c_1^N)^2/2\beta^2}\,, \tag{4.8}$$

$$f_{c_2}(z) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\beta\sqrt{2\pi}}\,\mathrm{e}^{-(z-c_2^N)^2/2\beta^2}\,, \tag{4.9}$$

where $\beta = 0.1$ and c_1, c_2 are the softmax/logit values of the two classes in the data set.

2. Calculate overlap index, η, using the local densities:

$$\eta = \sum_{i=1}^{M_z}\min[f_{c_1}(z_i), f_{c_2}(z_i)]\,\delta z\,, \tag{4.10}$$

where M_z defines the step size of z such that $\{z_i\}_{i=1}^{M_z}$ ranges from zero to one in M_z steps.

4.3 The cold posterior effect

It is observed that in order to get good predictive performance from Bayesian neural networks, the Bayesian posterior has to be down-weighted or tempered. This is known as the cold posterior effect and it has been observed by several authors (Zhang et al., 2019; Osawa et al., 2019; Ashukha et al., 2020; Wenzel et al., 2020). To temper the posterior, we use a weighting factor, T, in the cost function:

$$F_i(D_i,\theta) = \frac{T}{M}\mathrm{KL}[q(w|\theta)||P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(D_i|w)], \tag{4.11}$$

where $T \leq 1$ is the temperature. We also report this effect in our experiments with radio galaxies, see Figure 5.1, and discuss this effect further in Chapter 5.

4.4 Experiments with MiraBest

The MiraBest data set has a predefined training: test split. We further divide the training data into a ratio of 80:20 to create training and validation sets. The final split contains 584

TABLE 4.3: CNN architecture: MiraBest. Stride = 1 is used for all the convolutional and max pooling layers

Operation	Kernel	Channels	Padding
Convolution	5 x 5	6	1
ReLU			
Max Pooling	2 x 2		
Convolution	5 x 5	16	1
ReLU			
Max Pooling	2 x 2		
Convolution	5 x 5	26	1
ReLU			
Max Pooling	2 x 2		
Convolution	5 x 5	32	1
ReLU			
Max Pooling	2 x 2		
Fully-Connected		120	
ReLU			
Fully-Connected		84	
ReLU			
Fully-Connected		2	
Log Softmax			

training samples, 145 validation samples and 104 test samples.

4.4.1 Network Summary

The architecture used to classify MiraBest data set using BBB is shown in Table 4.3. We have added two additional convolutional layers to the LeNet-5 architecture.

The prior and posterior specifications are given below:

1. **Prior Specification:**
   - Gaussian Prior, $\sigma = 0.1$
   - Gaussian Mixture Model Prior, $\{3/4,\ 1,\ 9.\ 10^{-4}\}$
2. **Posterior Initialisation**
   - w_{mu} and b_{mu} are initialised from the uniform distribution $U(-0.1, 0.1)$
   - w_{rho} and b_{rho} are initialised from the uniform distribution $U(-5, -4)$

4.4.2 Training

All the models are trained for 500 epochs, with minibatches of size 50. We train the models using the Adam optimiser with a learning rate of 5.10^{-5}. A learning rate scheduler

is implemented which reduces the learning rate by 95 % if the validation likelihood cost does not improve for four consecutive epochs.

We found it necessary to temper our posterior in order to get a good performance from the Bayesian NN, without which the accuracy remains around 50%. We tempered the posterior for a range of temperature values, T, between $[10^{-5}, 1)$ and chose the largest value of T for which the accuracy was improved significantly. Thus, for all the experiments we use $T = 10^{-2}$ in Equation 4.11.

4.4.3 Uncertainty Quantification

For calculating MC samples from the Softmax and logit-space distributions, we calculate $N = 200$ samples (Equation 4.2).

The distribution-free parameter indices, η_1 and η_2, are calculated using $M_z = 100$ in Equation 4.10.

The MiraBest Uncertain data set samples contain 25 FRI and 24 FRII type galaxies. The Hybrid data set contains labels that indicate the confidence of the human classifier. Of the 30 samples in the data set, 17 are classified as Confident, while the remaining 13 are classified as Uncertain.

4.4.4 Alternative Prior

Laplace priors are used to induce sparsity in the network. We consider a Laplace prior and a Laplace Mixture Model prior with the following specifications:

1. **Prior Specification:**
    - Laplace Prior, $b = 1$
    - Laplace Mixture Model Prior, $\{0.75, 1, 10^{-3}\}$

We note that in case of this prior, we find the optimal model performance for a different initial learning rate of 10^{-4}. We used the same posterior initialisations as described for the Gaussian priors.

4.5 Results and Discussion

TABLE 4.4: Classification error on MiraBest using Bayesian-CNN

Method	#Parameters	Test Error
Gaussian Prior	464,888	$14.11 \pm 2.56\%$
GMM Prior	464,888	$12.80 \pm 2.60\%$

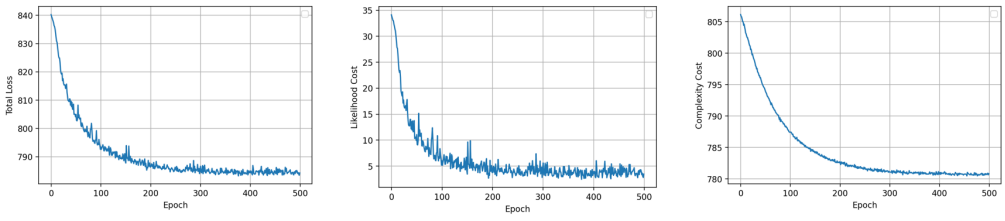


FIGURE 4.2: Training Loss Curves for BBB on MiraBest using a GMM Prior: The plots show the total loss, the likelihood cost and the complexity cost, in order.

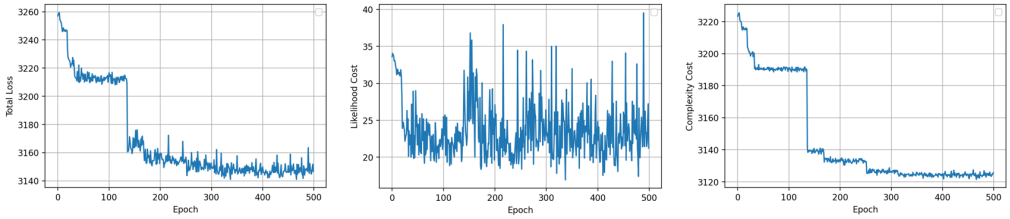


FIGURE 4.3: Validation Loss Curves for BBB on MiraBest using a GMM Prior: The plots show the total loss, the likelihood cost and the complexity cost, in order.

4.5.1 BBB Classifier Performance

The results of our classification experiments are shown in Table 4.4. The mean and standard deviation values of the test error are calculated by taking 100 samples from the posterior predictive distribution. Using the model trained with a GMM prior, we get a test error of $12.80 \pm 2.60\%$. The standard deviation values indicate the model's confidence in its prediction, and the large values can be attributed to the limited availability of training and test data. Bowles et al. (2021) who augment the MiraBest confident samples by 72 times, report a test error of 8%, whereas Scaife & Porter (2021) who use random rotations of the same data set as a function of epochs to augment the data, report a test error of $5.95 \pm 1.37\%$ with a LeNet-5 style CNN with Monte Carlo dropout and $3.43 \pm 1.29\%$ using D_{16} group-equivariant CNNs with Monte Carlo dropout. We again emphasise that the test error values we report are without any data augmentation.

The training and validation loss curves are shown in Figure 4.2 and Figure 4.3, respectively.

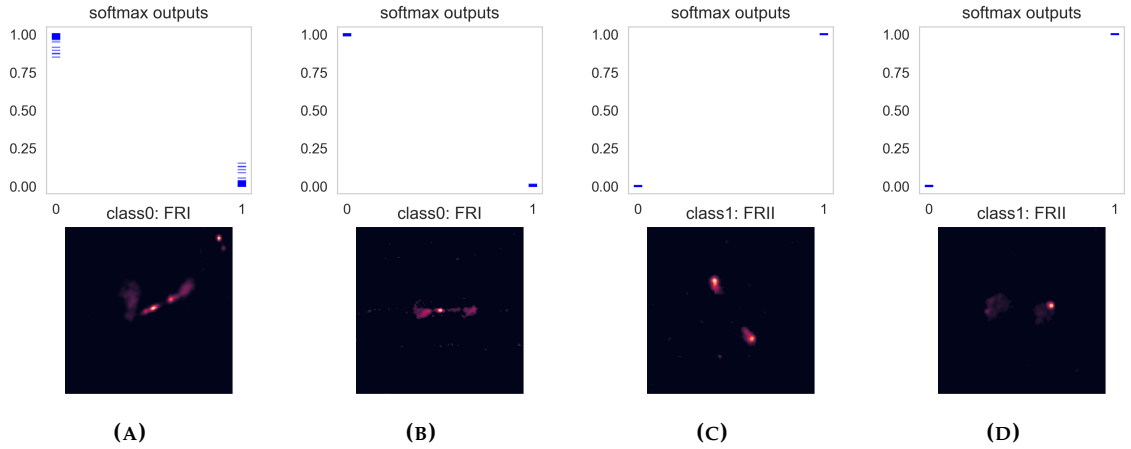


FIGURE 4.4: Examples of samples correctly classified with high predictive confidence. Top: softmax values for 200 forward passes through the trained model. Bottom: input data images.

4.5.2 Pruning Results

We use the methodology outlined in Section 3.5.3, with one major difference to adapt SNR-based pruning for a convolutional Bayesian NN: only the fully-connected layer weights of the model are pruned, instead of all the weights of the network. This is because the convolutional layer weights are shared weights, see Section 2.10, and removing even a small fraction may result in disastrous consequences for model performance. The fully-connected layers still make up $\sim$ 85% of the total weights of the network, so pruning methods are worth considering for convolutional BBB.

For the model trained on the MiraBest data set with a GMM prior, we find that up to 40% of the fully-connected layer weights can be pruned without a significant change in performance. The test error as a function of pruning threshold is shown in Figure 5.2, along with the results of alternative pruning methods based on Fisher information, which will be discussed in more detail in Section 5.2.1.

4.5.3 Uncertainty Quantification Results and Population Study

We first look at some specific examples of uncertainty quantification in the MiraBest Confident data set to build intuition about the uncertainty metrics used. We then study the uncertainties for the population of sources in MiraBest Confident, Uncertain and Hybrid data sets to see how well the trained model has captured the different types of uncertainties associated with these samples, see Figure 4.7 for an overview.

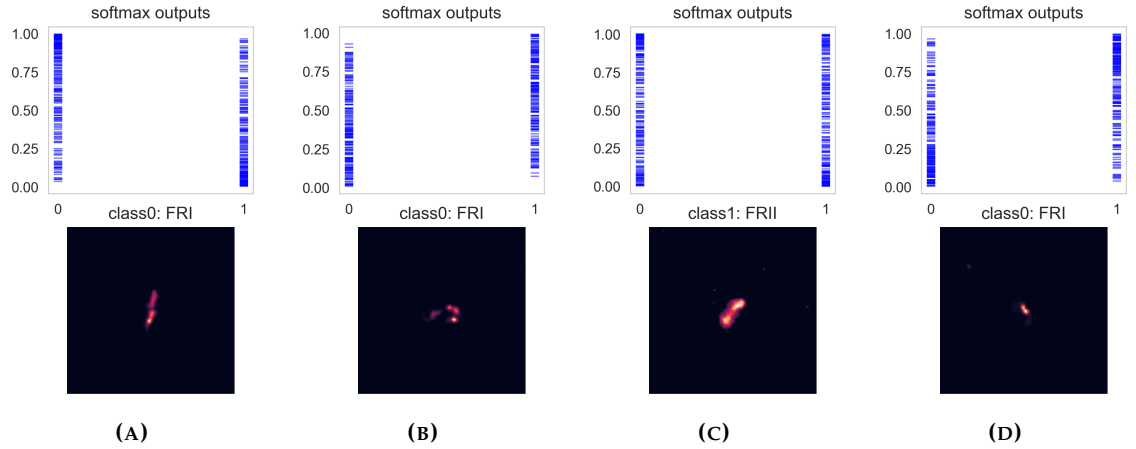


FIGURE 4.5: Samples with the highest entropy, mutual information, and high predictive uncertainty. Top: softmax values for 200 forward passes through the trained model. Bottom: input data images.

TABLE 4.5: Predictive entropy (PE), mutual information (MI) and overlap indices for Softmax (η_1) and logit-space (η_2) for samples incorrectly classified with low confidence shown in Figure 4.5.

Sample	PE	MI	η_1	η_2
A	0.64	0.20	0.56	0.13
B	0.67	0.14	0.70	0.11
C	0.64	0.16	0.54	0.13
D	0.68	0.28	0.74	0.16

We show some examples of galaxies that have been correctly classified with high confidence in Figure 4.4. These galaxies correspond to typical FRI/FRII classifications, and the predictive entropy and mutual information for all the samples shown is very low ($<$ 0.01 nats) for all samples except sample A, for which the predictive entropy is 0.04 nats and mutual information is 0.01 nats. The overlap indices η_1 and η_2 are $<< 10^{-5}$, which indicates that there is virtually no overlap, as can be seen in the distribution of Softmax probabilities.

We then consider galaxies for which the model uncertainty as well as the predictive uncertainty is high, as shown in Figure 4.5. These galaxies have the highest predictive entropy among the test samples of MiraBest confident data set and large values of overlap indices in both the softmax and logit space. These samples also have high mutual information, which indicates that the model's confidence in its classification is very low. The values of uncertainty metrics corresponding to these samples are shown in Table 4.5.

Finally in Figure 4.6, four examples where the model has incorrectly classified the test samples with high confidence are shown along with their softmax distributions. We

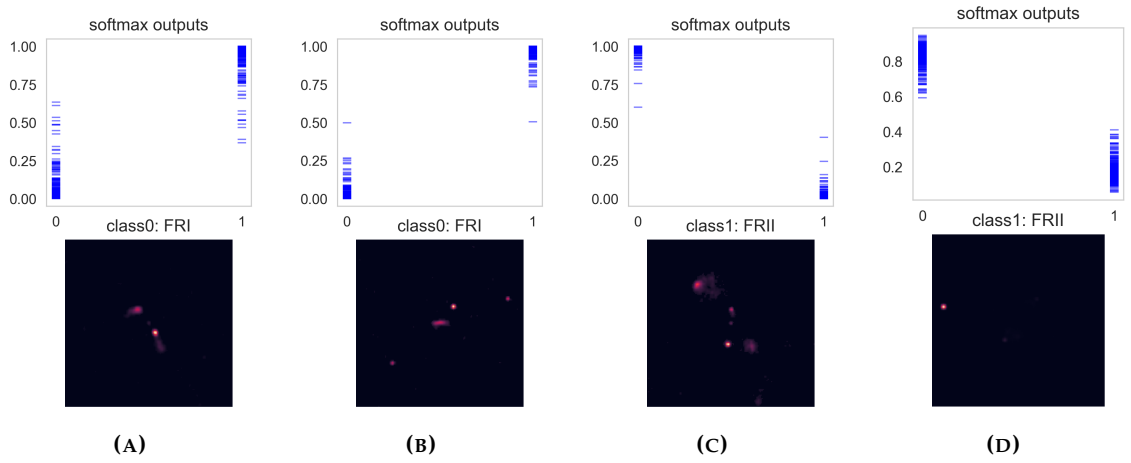


FIGURE 4.6: Samples that have been incorrectly classified with high predictive confidence. Top: softmax values for 200 forward passes through the trained model. Bottom: input data images.

can see that the galaxies either deviate from the typical FRI/FRII classification or their labels are somewhat ambiguous. For the galaxy shown in Figure 4.6a the model is able to identify the galaxy as an FRII, even when it has been mislabelled as an FRI in the dataset. The galaxy in Figure 4.6b has additional bright components at the edge along with bright components near the center. The model places high confidence on this galaxy being an FRII, but it is labelled FRI. In Figure 4.6c, we see a similar galaxy, which has been labelled an FRII and the model incorrectly classifies it as an FRI. Figure 4.6d shows a galaxy which has one bright component near its edge. The model's confidence in classifying this as an FRI can be attributed to a number of examples of FRI galaxies in the MiraBest Confident data set with a single bright component near the center. Since the underlying CNN architecture is equivariant to translation, the model could have learned to represent this feature as belonging to FRIs.

The predictive entropy and mutual information corresponding to these samples is shown in Table 4.6. In case of samples A and D, the predictive entropy is higher than that for samples B and C, while mutual information is similar for all the samples. This indicates that the uncertainty in prediction in relatively high for samples A and D, whereas the model's confidence in its prediction is high for all of these samples. Thus, in cases B and C, the bias introduced by the ambiguity in the definition of FRI and FRII and the ambiguity in the labels gives rise to uncertainty metrics that can be misleading.

TABLE 4.6: Predictive entropy (PE), mutual information (MI) and overlap indices for Softmax (η_1) and logit-space (η_2) for samples incorrectly classified with high confidence shown in Figure 4.6.

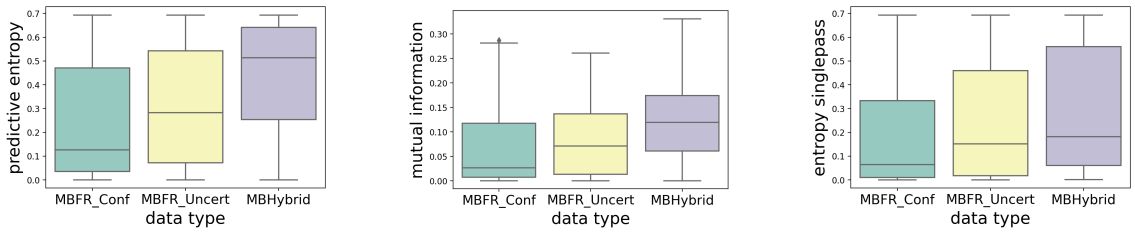
Sample	PE	MI	η_1	η_2
A	0.30	0.06	< 0.01	< 0.01
B	0.17	0.03	0.04	< 0.01
C	0.10	0.01	0.01	< 0.01
D	0.47	0.01	< 0.01	< 0.01

Analysis of Uncertainty Estimates on MiraBest Uncertain

We test the trained model's ability to capture epistemic uncertainty by calculating uncertainty metrics for the MiraBest Uncertain samples using the model trained on the MiraBest confident samples. This is done by loading in the saved model weights and making $N = 200$ forward passes through the model with the MiraBest Uncertain samples as test inputs. We use boxplots to study the distribution of uncertainty metrics. Some examples of boxplots can be seen in Figure 4.7a. The box shows the interquartile range of the distribution and the whiskers (grey bars) indicate the total extent of the distribution. The outliers (if any) are indicated using diamonds. The horizontal line inside the box indicates the median of the distribution. The values of uncertainty metrics are reported in nats, which is the natural unit of information. The uncertainty metrics thus attain a maximum value of 0.693 nats, when the entropy is maximum and a minimum value close to zero.

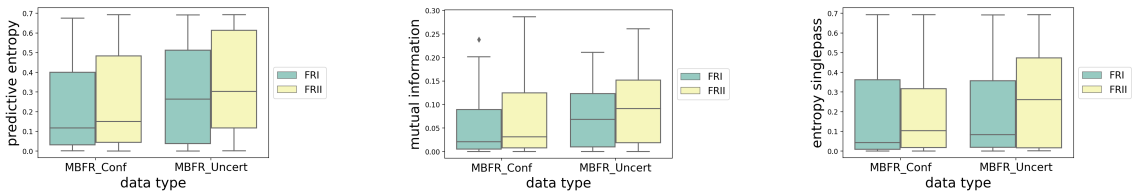
Predictive Uncertainty: We find that the predictive uncertainty is consistent with the degree of belief in the human classification, see Figure 4.7a, with uncertain samples being classified with a higher entropy than the confident samples. In Figure 4.8a we observe that the median values of the predictive entropy distribution are higher for both FRI and FRII classes in the MiraBest Uncertain data set and the interquartile range is also larger. The distribution is shifted toward higher values of predictive entropy, which indicates that a larger number of galaxies are being classified with higher predictive entropy. We note that even though the training set contains $\sim 7\%$ more FRIIs than FRIs, the predictive entropy distribution is slightly more spread out for FRIIs and has a higher median value.

Epistemic Uncertainty: The distribution of mutual information is shown in Figure 4.7b. The mutual information is also higher for samples from the MiraBest Uncertain set. This indicates a higher epistemic uncertainty in classifying these samples, which is consistent



(A) (B) (C)

FIGURE 4.7: Distributions of uncertainty metrics for MiraBest Confident (MBFR_Conf), Uncertain (MBFR_Uncert) and Hybrid (MBHybrid) data sets.



(A) (B) (C)

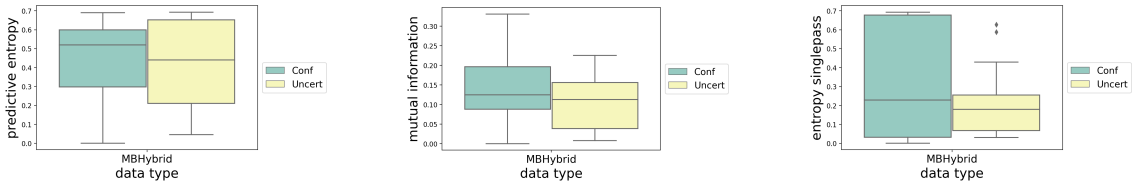
FIGURE 4.8: Class-wise distributions of uncertainty metrics for MiraBest Confident and MiraBest Uncertain data sets.

with how the data sets are defined. We also find that FRIIs have a higher degree of epistemic uncertainty than FRIs, see Figure 4.8b.

Aleatoric Uncertainty: Using the entropy of a single pass as a measure of aleatoric uncertainty, we see in Figure 4.8c that (i) for FRI type galaxies, the distribution of aleatoric uncertainty is very similar for the Confident and Uncertain subsets, and only the median value has shifted to a higher value by a small amount; (ii) for FRII type galaxies, the distribution is more spread out and it has a higher median value by $\sim$0.2 nats. These observations indicate that there is higher aleatoric uncertainty associated with FRII type galaxies. This could be due to the nature of images in the Uncertain subset where some FRIIs may be incorrectly labelled, i.e. the labels act as a source of noise that is inherent in the data set.

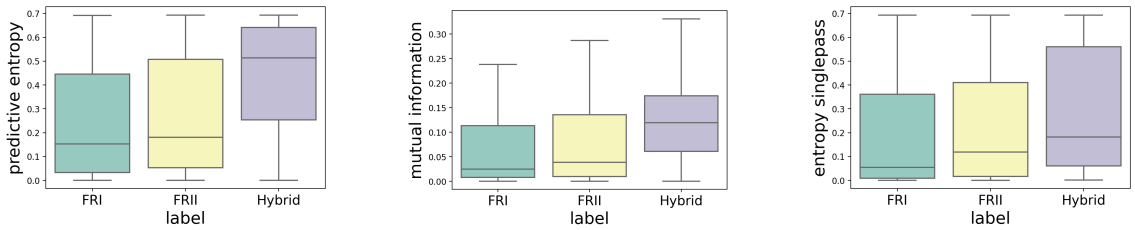
Analysis of Uncertainty Estimates on MiraBest Hybrid

We use the MiraBest Hybrid samples to test the model's ability to capture aleatoric uncertainty. This is done by loading in the saved model weights and making $N = 200$ forward passes through the model with the MiraBest Hybrid samples as test inputs.



(A) (B) (C)

FIGURE 4.9: Class-wise distributions of uncertainty metrics for MiraBest Hybrid data set.

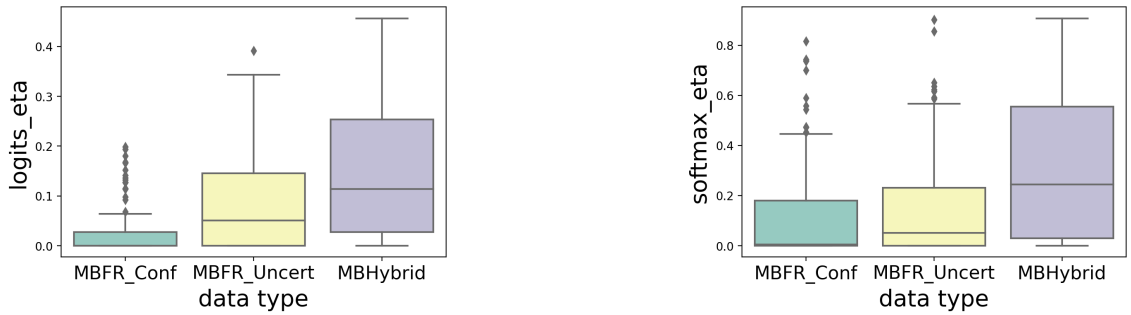


(A) (B) (C)

FIGURE 4.10: Morphology-wise distributions of uncertainty metrics for the MiraBest data set.

Predictive Uncertainty: The median predictive entropy of the Hybrid samples is higher than the MiraBest confident samples by 0.4 nats, as shown in Figure 4.7a. This indicates that there is a high degree of predictive uncertainty associated with the hybrid samples. This is expected as: (i) the training set does not contain any hybrid samples (ii) the hybrid samples represent a kind of noise at the source generating the data. In Figure 4.9a, we see the contributions of the sub-classes of the Hybrid set. The median values of the predictive entropy of the confident sub-set are higher than that of the uncertain sub-set, however the interquartile range is larger in case of the uncertain samples. We can also see form Figure 4.10a that the hybrid samples have higher uncertainty than FRI and FRII sources combined across the confident and uncertain samples.

Epistemic Uncertainty: In Figure 4.7b we see that the median value for the distribution of mutual information for the hybrid samples is close to the upper quartile of the MiraBest Confident subset. The high degree of epistemic uncertainty could be because the model did not see any hybrid samples during training. We also note that among the sub-classes of the Hybrid data set, the confidently classified samples have higher epistemic uncertainty than the uncertainly classified samples, as shown in Figure 4.9b. This could be because the uncertain samples are more like FRI/FRII galaxies that the model has seen during training, which could have been the reason they were classified as Uncertain by



(A) (B)

FIGURE 4.11: Distributions of overlap indices for the logit-space (logits_eta) and Softmax probability (softmax_eta) distributions for MiraBest Confident, Uncertain and Hybrid data sets.

a human classifier in the first place. The mutual information is also higher for the hybrid samples compared to the FRI and FRII samples combined across the confident and uncertain samples, see Figure 4.10b.

Aleatoric Uncertainty: The distribution of the entropy of a single pass has a higher median value and larger interquartile range, Figure 4.7c. While the plot shows that Hybrid samples have higher aleatoric uncertainty on average, in Figure 4.9c we can see how the aleatoric uncertainty is distributed among the classes in the hybrid samples. The confidently classified samples contribute a higher proportion to the total aleatoric uncertainty. The interquartile range is the largest, spanning almost the entire range of the entropy function between $(0, 0.693]$ nats. From Figure 4.10c we can see that the aleatoric uncertainty is higher for the hybrid samples compared to the FRI and FRII samples combined across the confident and uncertain samples.

Analysis of Uncertainty Estimates: Logits vs Softmax

In Figure 4.11 we show the distribution of Softmax and logit-space overlap indices η_1 and η_2. We can see the effect of the non-linear mapping between logit-space, Figure 4.11a, and Softmax-space, Figure 4.11b, on the uncertainty estimates for each subset of the MiraBest data set: for all the data sets, the Softmax overlap index distribution is broader than of logit-space, and it has more outliers towards higher values of the overlap index. We also note the difference in the scale of y-axis on both the plots. For the Softmax index distributions, the values span the whole range of η from zero to one, whereas for the

TABLE 4.7: Classification error on MiraBest Confident using Bayesian-CNN with Laplace priors

Method	#Parameters	Test Error
Laplace Prior	464,888	$11.39 \pm 2.14\%$
LMM Prior	464,888	$12.00 \pm 2.26\%$

logit-space overlap index, the values are less than 0.5. This gives us some insight into the 'squashing' operation of Softmax. Softmax probability scores alone can be misleading, which is one of the reasons why deterministic neural networks provide overconfident classifications.

4.5.4 Laplacian priors

The test errors for a model trained with a Laplace prior and a Laplace Mixture prior are shown in Table 4.7. We find that the classification performance is better than that for a Gaussian prior and the GMM prior, see Table 4.4 for comparison. This suggests that learning may benefit from sparser weights.

4.5.5 Discussion

In general we find that the method used in this work can correctly represent uncertainty in radio galaxy classification, and that this uncertainty is consistent with how human classifiers defined the MiraBest Confident, Uncertain and Hybrid qualifications. While we do not learn any new astrophysics from this, the methodology used in this work gives us learned posteriors. These posteriors can be used as a prior in future applications of Bayesian deep learning to radio galaxy classification.

While differences in performance are often used to choose a preferred model, however it is also the case that more accurate point-wise models make overconfident predictions (e.g. Nguyen et al., 2014). In particular this effect has been shown to lead to miscalibrated uncertainty in predictions, especially for data samples that are less similar to canonical examples of a class (Guo et al., 2017b; Hein et al., 2018). While previous works using the MiraBest dataset have achieved better classification accuracy on radio galaxy data, the methodology used in this work gives us uncertainty estimates through learned posteriors. Thus there is a trade-off between standard models, which are somewhat more

accurate, and our variational inference based Bayesian neural network, which is reasonably accurate while giving more reliable posteriors and therefore potentially more scientifically useful.

Chapter 5

Considerations for Improved Variational Inference

In this chapter, we present two considerations for improving variational inference as described in the previous chapters: (i) We examine the cold posterior effect and test a hypothesis for mitigating it. (ii) We consider an alternative pruning approach based on Fisher information and compare it to SNR-based pruning. We also examine the effect of pruning on uncertainty estimates.

5.1 The cold posterior effect

The cold posterior effect is shown in Figure 5.1 for our model trained on radio galaxies. We modified our cost function to down-weight the posterior in Equation 4.11 using a temperature term, T, and reported in the previous chapter that all experiments were performed using $T = 10^{-2}$. In Figure 5.1 we can see the effect of T on the test accuracy, and why we found it necessary to temper the posterior. These results suggests that some component of the Bayesian framework in the context of neural networks is misspecified and it becomes difficult to justify using a Bayesian approach to these models, whilst artificially reducing the effect of the components that make the learning Bayesian in the first place.

Finding an explanation for the cold posterior effect is an active area of research and several hypotheses have been proposed to explain this effect (Wenzel et al., 2020): use of

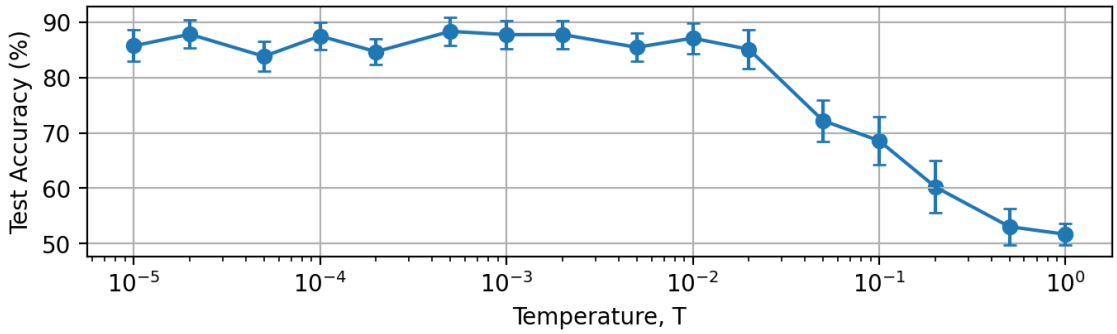


FIGURE 5.1: The cold posterior effect for a model trained with MiraBest Confident data set

uninformative priors, such as the standard Gaussian, which may lead to prior misspecification (Fortuin, 2021); model misspecification; data augmentation or data set curation issues which lead to likelihood misspecification (Aitchison, 2020; Nabarro et al., 2021).

In this section we examine whether the cold posterior effect in our results is due to model misspecification by optimising the model with a modified cost function, following the work of Masegosa (2019). The cost function is modified such that it minimises a second-order PAC-Bayesian bound on the cross entropy (CE) loss. We call this new objective function a 'masegosa posterior'.

5.1.1 Masegosa Posteriors

Generalised Variational Inference using PAC Bayes

PAC theory has its roots in Statistical Learning Theory and was first described in Valiant (1984) as a method for evaluating learnability, i.e. how well a machine learns hypotheses given a set of examples. It has now evolved into a formalised mathematical theory used to give statistical guarantees on the performance of machine learning algorithms by placing bounds on their generalisation performance.

According to the PAC theory, we can obtain an approximately correct upper bound on the generalisation performance, as measured by test loss for example, which holds true with an arbitrarily high probability as more data is collected, hence the name "Probably Approximately Correct". With high probability:

$$L(\theta) \leq \hat{L}(\theta) + \epsilon(\delta, D), \tag{5.1}$$

where δ is a confidence parameter that defines the probability that a sample in the training set is misleading, and $\epsilon(\delta, D_{\text{train}})$ is an upper bound on the generalisation gap, which is the difference between the theoretical loss, $L(\theta)$, and the empirical loss, $\hat{L}(\theta)$.

PAC started out as is a frequentist framework, but was soon combined with Bayesian principles. McAllester (1999) presented PAC-Bayesian inequalities which combine PAC-learning with Bayesian principles, which provide guarantees on the performance of *generalised* Bayesian algorithms. These algorithms are referred to as generalised because the PAC-Bayes framework has similar components to the Bayesian framework: a prior, π, defined over a set of hypotheses, $\theta \in \Theta$, and a posterior, ρ, which is updated using Bayes-rule style updates using samples from a data generating distribution, $\nu(x)$. But these bounds hold true for all choices of priors, whereas there is no guarantee on performance in Bayesian inference if the data set is not generated from the prior distribution i.e. if the prior assumptions are incorrect. The bounds also hold true for all choices of posteriors, so in principle we can have model-free learning. Traditionally, most of the PAC-Bayes bounds are only applicable to bounded loss functions. This makes it difficult to apply them to the unbounded loss functions which are typically used to train neural networks, however more recent works have introduced PAC Bayes bounds for unbounded losses as well (e.g. Alquier et al., 2016; Germain et al., 2016; Shalaeva et al., 2020). See Guedj (2019) for an overview of the PAC-Bayesian framework.

Learning in the PAC-Bayesian framework happens such that an optimal value of ρ^* is found that minimises the KL divergence between $\nu(x)$ and the posterior predictive distibution given by $\mathbb{E}_\rho[p(x|\theta)]$. Minimising this KL divergence is equivalent to minimising the cross entropy (CE) function,

$$\rho^* = \arg\min_\rho \text{CE}(\rho), \tag{5.2}$$

which is the expected log loss of the posterior predictive distribution, $p(x|\theta)$, with respect to the data generating distribution, $\nu(x)$:

$$\text{CE}(\rho) = \mathbb{E}_{\nu(x)}[-\log \mathbb{E}_{\rho(\theta)}[p(x|\theta)]]. \tag{5.3}$$

Thus, by minimising this CE function, we can find the optimal ρ^*. This cross entropy loss is bounded by the expected log loss of the posterior predictive distribution as:

$$CE(\rho) \leq \mathbb{E}_{\rho(\theta)}[L(\theta)]. \tag{5.4}$$

This is an example of what is known as an 'oracle' bound, since this inequality depends on the unknown data generating distribution, $\nu(x)$. It is also an example of a first-order Jensen inequality, which gives a linear bound such that:

$$\mathbb{E}_{\nu(x)}[-\log \mathbb{E}_{\rho(\theta)}[p(x|\theta)]] \leq \mathbb{E}_{\rho(\theta)}[\mathbb{E}_{\nu(x)}[-\log p(x|\theta)]], \tag{5.5}$$

which is an expansion of the terms given in Equation 5.4.

Germain et al. (2016) derived a first order PAC-Bayes bound for unbounded losses:

$$\mathrm{CE}(\rho) \leq \mathbb{E}_{\rho(\theta)}[L(\theta)] \leq \mathbb{E}_{\rho(\theta)}[\hat{L}(\theta, D)] + \frac{\mathrm{KL}(\rho, \pi)}{c_1} + c_2, \tag{5.6}$$

where c_1 and c_2 are constants. In the same work, Germain et al. (2016) also showed that under i.i.d. assumptions, the Bayesian posterior, $p(\theta|D)$, minimises this PAC-Bayes bound over the expected log loss, $\mathbb{E}_{\rho(\theta)}[L(\theta)]$, which bounds the cross-entropy loss.

Since variational inference is an approximation to the Bayesian marginal likelihood, we can use these bounds to optimise VI-based Bayesian NNs. By applying these bounds we deviate from the variational inference as defined in the Bayesian paradigm and move towards a more generalised variational inference algorithm.

The Variance Term

Masegosa (2019) showed that the Bayesian posterior minimises a PAC-Bayes bound over the CE loss only when the model is perfectly specified. When the model is misspecified, the minimum of the CE loss is not equal to minimum of the expected log loss, and thus optimising the Bayesian posterior does not give an optimal learning strategy. Since this is more often the case, they propose an alternative posterior by introducing a *variance term* that measures the variance of the posterior predictive distribution. They define a

second-order oracle and Jensen bound, which is given as:

$$CE(\rho) \leq \mathbb{E}_{\rho(\theta)}[L(\theta)] - \mathbb{V}(\rho), \tag{5.7}$$

where

$$\mathbb{V}(\rho) = \mathbb{E}_{\nu(x)}\left[\frac{1}{2\max_\theta p(x|\theta)^2}\mathbb{E}_{\rho(\theta)}[(p(x|\theta) - p(x))^2]\right] \tag{5.8}$$

is the variance term.

Since the true data generating distribution $\nu(x)$ is not known, the authors place an upper bound on Equation 5.7 using a second-order PAC-Bayes bound:

$$CE(\rho) \leq \mathbb{E}_{\rho(\theta)}[L(\theta] - \mathbb{V}(\rho) \leq \mathbb{E}_{\rho(\theta)}[\hat{L}(\theta, D)] - \hat{\mathbb{V}}(\rho, D) + \frac{\mathrm{KL}(\rho, \pi)}{c_1} + c_2, \tag{5.9}$$

where $\hat{L}(\theta, D)$ and $\hat{\mathbb{V}}(\rho, D)$ are the empirical loss and variance term, respectively.

This alternative posterior is compatible with VI and we can modify our cost function to test the hypothesis that the cold posterior effect observed in our work is due to model misspecification. Instead of optimising the ELBO function in Equation 3.14, we optimise the following function:

$$\arg\min_\theta \mathrm{KL}[q(w|\theta)|P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(D|w)] - \hat{\mathbb{V}}(q|D). \tag{5.10}$$

The empirical variance term, $\hat{\mathbb{V}}$, can be numerically calculated as follows:

$$\hat{\mathbb{V}}(w, w', D) = \exp(2\log P(D|w) - 2m_D) - \exp(\log P(D|w) + \log P(D|w') - 2m_D), \tag{5.11}$$

where w, w' are samples from the variational posterior, $q(w|\theta)$, D is the training data and m_D is given as:

$$m_D = \max_w \log P(D|w). \tag{5.12}$$

Results

The results of training our model with and without the variance term for $T = 1$ are shown in Table 5.1. We find that there is no significant improvement in the model performance with the variance term. The mean test error improves only by $\sim 2\%$. This suggests that

TABLE 5.1: Classification error on MiraBest using with variance term for temperature T = 1

Method	T	Test Error
without variance term	1	$48.30 \pm 1.89\%$
with variance term	1	$46.60 \pm 2.78\%$

model misspecification is not the primary cause of the cold posterior effect observed in our work.

5.2 Alternative pruning approaches

In this section, we discuss an alternative pruning approach based on Fisher information. We compare the performance of our BBB model trained on radio galaxies for different pruning methods and analyse the effect of pruning on uncertainty estimates.

We use the method described in Tu et al. (2016) for deterministic neural networks to prune our Bayesian NN trained on radio galaxy data. The Fisher information matrix (FIM) for a particular parameter of the network θ indicates how relevant the parameter is for producing the output of the network. The empirical FIM for a parameter θ can be calculated as follows:

$$F(\theta) = \mathbb{E}_y \left[\left(\frac{\partial \log L}{\partial \theta} \right) \left(\frac{\partial \log L}{\partial \theta} \right)^T \right] , \tag{5.13}$$

where L is the loss function. For our model this loss is the ELBO function, see Equation 3.11, whereas Tu et al. (2016) have used the log likelihood loss function. Using the Adam optimiser (Kingma & Ba, 2014) to train the models allows us to use the bias-corrected second raw moment estimate of the gradient to approximate the FIM diagonal. This is the value $\hat{\boldsymbol{r}}$ in Algorithm 3.

5.2.1 Fisher information method

The results of our pruning experiments are shown in Figure 5.2. As also observed by Tu et al. (2016), pruning the weights based on Fisher information alone does not allow for a large number of parameters to be pruned effectively because many values in the FIM diagonal are close to zero. We find this to be true for our model as well and that only $\sim$10% of the fully-connected layers can be pruned using Fisher information alone, see Figure 5.2.

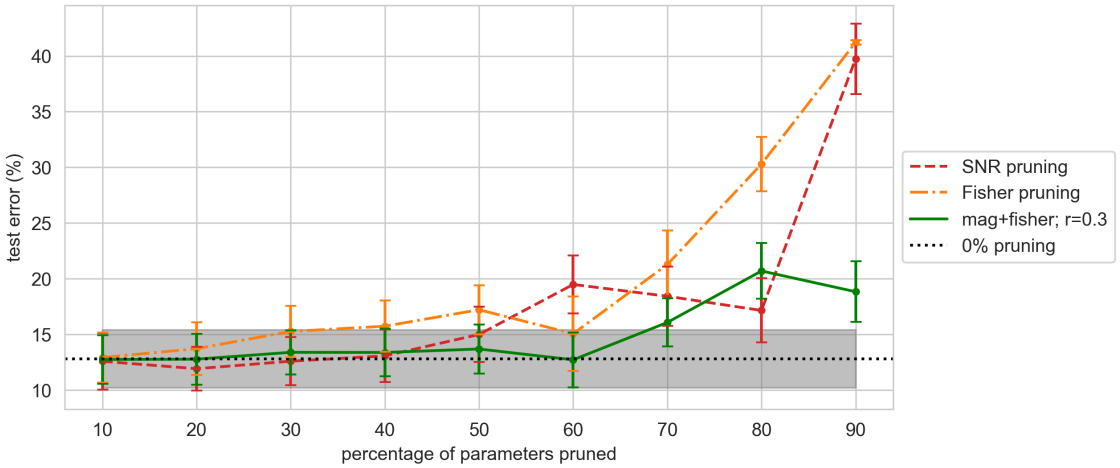
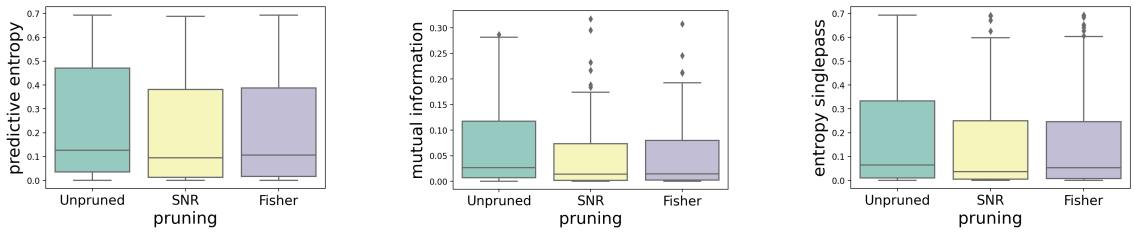


FIGURE 5.2: Comparison of model performance for different pruning methods based on: SNR, Fisher information, a combination of magnitude and Fisher information.

To remedy this, the authors suggest combining Fisher pruning with magnitude-based pruning, where magnitude refers to the absolute value of the weights. Following their approach, we define a parameter, r, to determine the proportion of weights that are pruned by either of these methods: to prune P parameters from the network, we perform the following steps in order: (i) remove the $P(1-r)$ weights with the lowest magnitude; (ii) remove the Pr parameters with the lowest FIM values. The parameter r is between $(0, 1)$ and is tuned like a hyper-parameter. We get an optimal pruning performance with $r = 0.3$. We find that up to 60% of the fully-connected layer weights can be pruned using this method, which is a 20% larger volume of weights than those pruned by the SNR-based method discussed in Section 4.5.2.

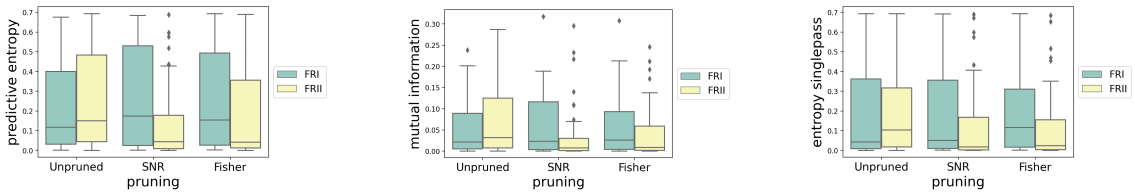
5.2.2 Analysis of Uncertainty Estimates for different pruning methods

The effect of pruning on uncertainty quantification is shown in Figure 5.3 for the MiraBest Confident data set. We plot uncertainty metrics for the two pruning methods discussed in this work: (i) based on SNR, with 40% pruning, and (ii) based on magnitude combined with Fisher information, with 60% pruning, and compare it to the metrics obtained for the unpruned model. From Figure 5.3a, we note that the distributions of predictive uncertainty for both the pruning methods have shifted towards lower values of entropy, with slightly smaller median values. A similar trend can be seen for epistemic uncertainty from the mutual information plot in Figure 5.3b and for aleatoric uncertainty in



(A) (B) (C)

FIGURE 5.3: Distributions of uncertainty metrics for different pruning methods for the MiraBest Confident data set.



(A) (B) (C)

FIGURE 5.4: Class-wise distributions of uncertainty metrics for different pruning methods for the MiraBest Confident data set.

Figure 5.3c. However, we observe that there are a few outliers for both epistemic and aleatoric uncertainties.

In Figure 5.4, we can see how these uncertainties are distributed among the FRI and FRII samples in the data set. From Figure 5.4a, we note that SNR pruning causes the interquartile range of the predictive uncertainty distribution to shift to lower values of entropy for FRIIs by a large amount compared to the unpruned distribution, by ~ 0.3 nats. A similar effect can be seen for Fisher-based pruning but the range shifts by a smaller amount. In case of FRIs, the opposite effect is seen i.e., the distributions become more spread out.

The epistemic uncertainty distributions also experience a similar effect, as shown in Figure 5.4b. These observations indicate that there is more epistemic uncertainty associated with the FRII samples, which may be improved by removing parameters that are noisy and contain less information. And removing the source of epistemic uncertainty also improves the predictive uncertainty.

Figure 5.4c indicates that pruning does not have a large effect on the aleatoric uncertainty distributions for FRIs. We note that the aleatoric uncertainty reduces significantly for FRIIs, which is unexpected and requires further examination. While the metric we

have used to estimate aleatoric uncertainty is adopted from Mukhoti et al. (2021), we note that they use it in addition to several other metrics to disentangle epistemic and aleatoric uncertainty for deterministic neural networks. Future work in this area could look at a different method for estimating aleatoric uncertainty.

Chapter 6

Conclusions and Future Work

In this work we have presented the first application of a variational inference based approach to morphological classification of radio galaxies, using a binary FRI/FRII classification. Using a Bayesian Convolutional Neural Network based on the Bayes by Backprop (BBB) algorithm, we showed that posterior uncertainties on the predictions of the model can be estimated by making a variational approximation to the posterior probability distribution over the model parameters. We also showed how this method overcomes the limitations of standard neural networks, which produce deterministic outputs using maximum likelihood estimates and are overconfident in their predictions due to the squashing nature of the Softmax function. This was also verified by comparing the logit-space uncertainties with Softmax uncertainties.

We showed that using a BBB model allows the test samples in the MiraBest Confident data set to be classified with an error of $12.80 \pm 2.60\%$ using a Gausssian Mixture Model (GMM) prior. We note that this is a larger error value than that obtained by other neural network based models trained on this data set, but emphasise that other works have used data augmentation to increase the size of the data set, whereas we have only used the original samples. This allowed us to study the use of VI-based neural networks on small data sets as well.

We also found that a model trained with a Laplace prior performs better than a Gaussian prior in terms of mean test error by $\sim 3\%$, and better than a GMM model prior by $\sim 1\%$, whereas the test error for a model trained with a Laplace Mixture Model prior also performs slightly better than the Gaussian priors, although not as well as a single Laplace prior. This suggests that learning may benefit from sparser weights. However,

the uncertainties associated with all the mean test error values are $\sim 2\%$ which means we cannot conclusively say which prior is better. We note that this work uses relatively simple priors and future extensions of this work will look more closely at prior specification and whether more informative priors can help learning.

The analysis of different components of uncertainties indicates that *model* uncertainty is correlated with the degree of belief of the human classifiers who originally created the morphological classification in the MiraBest data set. We find that a model trained on confidently classified radio galaxies is able to reliably estimate its confidence in predictions when presented with radio galaxies that have been classified with a lower degree of confidence. Notably, all measures of uncertainty are higher for the uncertainly classified samples. The model also made predictions with higher uncertainty for a sample of hybrid radio galaxies, which was expected as these samples were not present in the data set the model was trained on, but contained FRI/FRII like components nevertheless. Looking more closely at the class-wise distributions of uncertainties, we found that FRIIs are associated with a higher degree of uncertainty. Among the classes of the hybrid samples, we found that the uncertainty values are higher for the confidently classified samples compared to the uncertainly classified hybrid samples. This could be because the uncertain samples are more like the FRI/FRII samples the model was trained on, which is why the human classifiers were uncertain in their classification as a hybrid.

We also explored different weight pruning approaches with the motivation of reducing the storage and computation cost of these models at deployment. We find that using a SNR based method using posterior means and variances allows the fully-connected layers of the model to be pruned by up to 40%, but a method that combines Fisher information with weight magnitudes allows an even higher proportion of weights to be pruned, by up to 60%, without compromising the model performance. These two methods are based on fundamentally different methodologies: while SNR pruning takes into account noisy weights that are either too small in magnitude or have large posterior variances, the Fisher-information based method removes parameters based on their contribution to the gradients. If a parameter has smaller FIM values, this indicates that the gradients of the parameter did not change much during training. This means that the parameter contained less information and was less relevant to produce the output of the model. Thus one method may be better than the other in specific applications. We note here that since

our data set is small, the SNR metrics may be more noisy because model uncertainty was not able to be greatly reduced due to the limited availability of data. The effect of removing some of these noisy weights can also be seen on the uncertainty metrics. We found that while an unpruned model classified FRIIs with higher uncertainty, this is reduced when the model is pruned up to its threshold limit for both SNR and Fisher based metrics, but there is more improvement in case of SNR-based pruning. Future work in this area could make a comparison of these methods with augmented data to verify whether one method should be preferred over the other. Another possible extension could be re-training a pruned model to test whether pruning improves the generalisation performance of a network.

Finally, we considered the cold posterior effect and its implications for using Bayesian deep learning with radio galaxy data in future. We showed that our model required posterior tempering during training in order to have good predictive performance. While a model trained with an un-tempered posterior with $T = 1$ had a test error of $48.30 \pm 1.89\%$, reducing the temperature to $T = 10^{-2}$ allowed a model trained with a GMM prior to achieve $12.80 \pm 2.60\%$ test error. This is a significant difference in performance and calls into question the validity of the application of Bayesian principles to neural networks.

We considered the hypothesis that model misspecification may be causing the cold posterior effect and tested it by retraining our model with a modified cost function that has an additional variance term to account for the disagreement or variability in the posterior predictive distribution. This variance term comes from PAC-Bayesian theory which is used to place bounds on the generalisation performance of a learning algorithm. The modified cost function is a loose PAC-Bayes bound over the cross-entropy loss and leads to a new, generalised variational inference learning algorithm. However, we find that the model performance does not improve significantly when trained with the variance term and the mean test error only improves by $\sim 2\%$. Thus we conclude that model misspecification is not the major contributing factor to the cold posterior effect observed in our work. Further examination of this effect is required in order to find the component of Bayesian learning that is misspecified. Future studies in this area will investigate other hypotheses such as prior misspecification or data set curation issues. More specifically, the effect of data augmentation can be examined to test whether the model needs to over-count evidence via the tempered posterior in order to compensate for the limited

availability of data.

In this work we have considered a binary classification of morphology, but a diverse and complex population of galaxies exist in the radio universe. Understanding how populations of radio galaxies are distributed gives us insight into the effect of extrinsic and intrinsic factors that may have led to the morphologies, which in turn help shape our understanding of radio-loud AGN, their excitation and accretion modes, how they evolve and their relationship with their host galaxies and environments. Deep learning will play an important role in extracting scientific value from the next-generation of radio facilities. Understanding how neural network models propagate uncertainties will be crucial for deploying these models and a lot of work still needs to be done in this area.

Bibliography

Abazajian K. N., et al., 2009, Astrophysical Journal, Supplement, 182, 543

Abdar M., et al., 2021, Information Fusion

Aitchison L., 2020, in International Conference on Learning Representations.

Alger M. J., et al., 2018, Monthly Notices of the Royal Astronomical Society, 478, 5547

Alquier P., Ridgway J., Chopin N., 2016, The Journal of Machine Learning Research, 17, 8374

Aniyan A. K., Thorat K., 2017, The Astrophysical Journal Supplement Series, 230, 20

Ashukha A., Lyzhov A., Molchanov D., Vetrov D., 2020, arXiv preprint arXiv:2002.06470

Baldi R. D., Capetti A., Giovannini G., 2015, Astronomy and Astrophysics, 576, A38

Bastien D. J., Scaife A. M., Tang H., Bowles M., Porter F., 2021, Monthly Notices of the Royal Astronomical Society, 503, 3351

Becker R. H., White R. L., Helfand D. J., 1995, Astrophysical Journal, 450, 559

Best P. N., Heckman T. M., 2012, Monthly Notices of the Royal Astronomical Society, 421, 1569

Best P. N., Kauffmann G., Heckman T. M., Ivezić Ž., 2005, Monthly Notices of the RAS, 362, 9

Bicknell G. V., 1995, Astrophysical Journal, Supplement, 101, 29

Blei D. M., Kucukelbir A., McAuliffe J. D., 2016, arXiv e-prints, p. arXiv:1601.00670

Blundell C., Cornebise J., Kavukcuoglu K., Wierstra D., 2015, in Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15. JMLR.org, p. 1613–1622

Bowles M., Scaife A. M., Porter F., Tang H., Bastien D. J., 2021, Monthly Notices of the Royal Astronomical Society, 501, 4579

Cheng X. P., An T., 2018, Astrophysical Journal, 863, 155

Condon J. J., Cotton W. D., Greisen E. W., Yin Q. F., Perley R. A., Taylor G. B., Broderick J. J., 1998, Astronomical Journal, 115, 1693

Croston J. H., Ineson J., Hardcastle M. J., 2018, Monthly Notices of the RAS, 476, 1614

Donoso E., Best P. N., Kauffmann G., 2009, Monthly Notices of the RAS, 392, 617

Fanaroff B. L., Riley J. M., 1974, Monthly Notices of the Royal Astronomical Society, 167, 31P

Fortuin V., 2021, arXiv preprint arXiv:2105.06868

Gal Y., 2016, PhD thesis, University of Cambridge, http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf

Gal Y., Ghahramani Z., 2015, arXiv preprint arXiv:1506.02158

Garofalo D., Singh C. B., 2019, Astrophysical Journal, 871, 259

Gendre M. A., Best P. N., Wall J. V., Ker L. M., 2013, Monthly Notices of the Royal Astronomical Society, 430, 3086

George D., Shen H., Huerta E., 2018, Physical Review D, 97, 101501

Germain P., Bach F., Lacoste A., Lacoste-Julien S., 2016, arXiv preprint arXiv:1605.08636

Goodfellow I., Bengio Y., Courville A., 2016, Deep Learning. MIT Press

Gopal-Krishna Wiita P. J., 2000, Astronomy and Astrophysics, 363, 507

Graves A., 2011, in Proceedings of the 24th International Conference on Neural Information Processing Systems. NIPS'11. Curran Associates Inc., Red Hook, NY, USA, p. 2348–2356

Guedj B., 2019, arXiv preprint arXiv:1901.05353

Guo C., Pleiss G., Sun Y., Weinberger K. Q., 2017a, in International Conference on Machine Learning. pp 1321–1330

Guo C., Pleiss G., Sun Y., Weinberger K. Q., 2017b, in Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17. JMLR.org, p. 1321–1330

Hardcastle M. J., 2004, Astronomy and Astrophysics, 414, 927

Hardcastle M., 2018, Nature Astronomy, 2, 273

Hardcastle M., Croston J., 2020, New Astronomy Reviews, 88, 101539

Harwood J. J., Vernstrom T., Stroe A., 2020, Monthly Notices of the RAS, 491, 803

Hein M., Andriushchenko M., Bitterwolf J., 2018, CoRR, abs/1812.05720

Hinton G. E., van Camp D., 1993, in Proceedings of the Sixth Annual Conference on Computational Learning Theory. COLT '93. Association for Computing Machinery, New York, NY, USA, p. 5–13, doi:10.1145/168304.168306, https://doi.org/10.1145/168304.168306

Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R., 2012, arXiv e-prints, p. arXiv:1207.0580

Hochreiter S., 1991, Diploma, Technische Universität München, 91

Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J., et al., 2001, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies

Hornik K., Stinchcombe M., White H., 1989, Neural networks, 2, 359

Ineson J., Croston J. H., Hardcastle M. J., Kraft R. P., Evans D. A., Jarvis M., 2015, Monthly Notices of the Royal Astronomical Society

Janssen R. M. J., Röttgering H. J. A., Best P. N., Brinchmann J., 2012, Astronomy and Astrophysics, 541, A62

Jordan M. I., Ghahramani Z., Jaakkola T. S., Saul L. K., 1999, An Introduction to Variational Methods for Graphical Models. MIT Press, Cambridge, MA, USA, p. 105–161

Jospin L. V., Buntine W., Boussaid F., Laga H., Bennamoun M., 2020, arXiv preprint arXiv:2007.06823

Kaiser C. R., Schoenmakers A. P., Röttgering H. J. A., 2000, Monthly Notices of the RAS, 315, 381

Kingma D. P., Ba J., 2014, arXiv e-prints, p. arXiv:1412.6980

Kingma D. P., Welling M., 2013, arXiv preprint arXiv:1312.6114

Kingma D. P., Salimans T., Welling M., 2015, Advances in neural information processing systems, 28, 2575

Krizhevsky A., Sutskever I., Hinton G. E., 2012, Advances in neural information processing systems, 25, 1097

Krogh A., Hertz J. A., 1991, in Proceedings of the 4th International Conference on Neural Information Processing Systems. NIPS'91. Morgan Kaufmann Publishers Inc., p. 950–957

Kullback S., Leibler R. A., 1951, The annals of mathematical statistics, 22, 79

Laing R. A., Bridle A. H., 2014, Monthly Notices of the RAS, 437, 3405

LeCun Y., Bengio Y., 1998, Convolutional Networks for Images, Speech, and Time Series. MIT Press, Cambridge, MA, USA, p. 255–258

LeCun Y., Cortes C., 2010, MNIST handwritten digit database, `http://yann.lecun.com/exdb/mnist/`

LeCun Y., Denker J. S., Solla S. A., 1990, in Advances in neural information processing systems. pp 598–605

LeCun Y. A., Bottou L., Orr G. B., Müller K.-R., 2012, in , Neural networks: Tricks of the trade. Springer, pp 9–48

Lecun Y., Bottou L., Bengio Y., Haffner P., 1998, Proceedings of the IEEE, 86, 2278

Ledlow M. J., Owen F. N., 1996, Astronomical Journal, 112, 9

Lukic V., Brüggen M., Mingo B., Croston J. H., Kasieczka G., Best P. N., 2019, Monthly Notices of the RAS, 487, 1729

MacKay D. J. C., 1992a, Neural Computation, 4, 448

MacKay D. J. C., 1992b, Neural Computation, 4, 720

Mahatma V. H., Hardcastle M. J., Williams W. L., Lofar Surveys Team 2020, in Asada K., de Gouveia Dal Pino E., Giroletti M., Nagai H., Nemmen R., eds, Vol. 342, Perseus in Sicily: From Black Hole to Cluster Outskirts. pp 122–126, doi:10.1017/S1743921318005537

Masegosa A. R., 2019, arXiv preprint arXiv:1912.08335

Masters D., Luschi C., 2018, arXiv preprint arXiv:1804.07612

McAllester D. A., 1999, Machine Learning, 37, 355

McConnell D., et al., 2020, Publications of the Astron. Soc. of Australia, 37, e048

Mesarcik M., Boonstra A.-J., Meijer C., Jansen W., Ranguelova E., van Nieuwpoort R. V., 2020, Monthly Notices of the Royal Astronomical Society, 496, 1517

Mingo B., et al., 2019, Monthly Notices of the Royal Astronomical Society, 488, 2701

Miraghaei H., Best P. N., 2017, Monthly Notices of the Royal Astronomical Society, 466, 4346

Mukhoti J., Kirsch A., van Amersfoort J., Torr P. H., Gal Y., 2021, arXiv preprint arXiv:2102.11582

Nabarro S., Ganev S., Garriga-Alonso A., Fortuin V., van der Wilk M., Aitchison L., 2021, arXiv preprint arXiv:2106.05586

Naul B., Bloom J. S., Pérez F., van der Walt S., 2018, Nature Astronomy, 2, 151

Neal R. M., Hinton G. E., 1998, in , Learning in graphical models. Springer, pp 355–368

Nguyen A. M., Yosinski J., Clune J., 2014, CoRR, abs/1412.1897

O'Dea C. P., Owen F. N., 1985, Astronomical Journal, 90, 927

Osawa K., Swaroop S., Jain A., Eschenhagen R., Turner R. E., Yokota R., Khan M. E., 2019, arXiv preprint arXiv:1906.02506

Pastore M., Calcagnì A., 2019, Frontiers in psychology, 10, 1089

Peterson C., 1987, Complex systems, 1, 995

Polyak B. T., 1964, Ussr computational mathematics and mathematical physics, 4, 1

Porter F. A. M., 2020, MiraBest Batched Dataset, doi:10.5281/ZENODO.4288837, https://zenodo.org/record/4288837

Pratt L. Y., et al., 1993, Advances in neural information processing systems, pp 204–204

Prechelt L., 1998, in , Neural Networks: Tricks of the trade. Springer, pp 55–69

Qian N., 1999, Neural networks, 12, 145

Rosenblatt F., 1958, Psychological review, 65, 386

Rudnick L., Owen F. N., 1976, Astrophysical Journal, Letters, 203, L107

Rumelhart D. E., Hinton G. E., Williams R. J., 1986, Learning Internal Representations by Error Propagation. MIT Press, Cambridge, MA, USA, p. 318–362

Sabour S., Frosst N., Hinton G. E., 2017, arXiv preprint arXiv:1710.09829

Saripalli L., 2012, Astronomical Journal, 144, 85

Saul L. K., Jaakkola T., Jordan M. I., 1996, Journal of artificial intelligence research, 4, 61

Scaife A. M., Porter F., 2021, Monthly Notices of the Royal Astronomical Society, 503, 2369

Schoenmakers A. P., de Bruyn A. G., Röttgering H. J. A., van der Laan H., Kaiser C. R., 2000, Monthly Notices of the RAS, 315, 371

Shabala S. S., Ash S., Alexander P., Riley J. M., 2008, Monthly Notices of the RAS, 388, 625

Shalaeva V., Esfahani A. F., Germain P., Petreczky M., 2020, in Proceedings of the AAAI Conference on Artificial Intelligence. pp 5660–5667

Shridhar K., Laumann F., Liwicki M., 2019, arXiv e-prints, p. arXiv:1901.02731

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, The journal of machine learning research, 15, 1929

Tang H., Scaife A. M. M., Leahy J. P., 2019, Monthly Notices of the Royal Astronomical Society, 488, 3358

Tu M., Berisha V., Cao Y., Seo J.-s., 2016, in 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). pp 93–98

Valiant L. G., 1984, Communications of the ACM, 27, 1134

Wainwright M. J., Jordan M. I., 2008, Graphical models, exponential families, and variational inference. Now Publishers Inc

Wang Y.-C., Li M.-T., Pan Z.-C., Zheng J.-H., 2019, Research in Astronomy and Astrophysics, 19, 133

Wenzel F., et al., 2020, arXiv preprint arXiv:2002.02405

Wu C., et al., 2018, Monthly Notices of the Royal Astronomical Society, 482, 1211

Zhang R., Li C., Zhang J., Chen C., Wilson A. G., 2019, arXiv preprint arXiv:1902.03932