Predicting the heteroscedastic aleatoric uncertainty of the images of radio galaxies

Mohammad Kordzanganeh

10137275

School of Physics and Astronomy

University of Manchester

MPhys Report

January 2021

This project was performed in collaboration with *Aydin Utting*

Abstract

A convolutional neural network was trained to predict the heteroscedastic aleatoric uncertainties of images as a learned parameter. This was done by introducing a normal prior distribution over the output layer whose standard deviation was derived from a secondary output of the network. This network was then trained on a data set including 50x50 images of radio galaxies, and by the end of training it predicted the aleatoric uncertainties of input images. This was tested by augmenting the test data and measuring the change in the distribution of uncertainty pre- and post-augmentation. Then, the training data set was split into five quintiles based on uncertainty and separate models were trained on them. The models were compared with each other as well as with a model trained on randomly pruned data by using their validation accuracy and loss curves. The second lowest uncertainty quintile had the highest validation accuracy and the lowest validation loss, outperforming the rest of the models. This was identified to be due to its spread of data between low uncertainty data and some uncertain data where more new features are displayed.

Contents

1 Introduction

Machine learning has grown rapidly in the past decade and it has solved many problems that we previously thought were intractable. Supervised machine learning utilises neural networks and the provided training data to make an approximate model for the problem. However, both of these requirements have an uncertainty associated with them. This project aims to explore the uncertainty on the latter, namely the data provided to the model or the heteroscedastic aleatoric uncertainty, and specifically, the uncertainty on a categorisation due to the noise in the input data.

To explore this uncertainty, we shall work with images of radio galaxies. B. Fanroff and J. Riley made a survey of radio galaxies in 1974 and reported that there was a clear distinction between two sets of radio galaxies. First set, named type-I, were fainter and more spread across the sky, and the second set, type-II, had strong active cores. [1] We shall use a data set made by H. Miraghaei and P. Best - often referred to as *Mirabest* - to train a classifying neural network that also predicts the aleatoric uncertainty of each image.

2 Theory

To understand the theory for this report[1] we, first, need to establish what a statistical hypothesis is. The theory of machine intelligence relies on the fine tuning the parameters of a hypothesis.

2.1 Statistical hypothesis

The hypothesis of machine learning is that given an entry, a model can predict accurately the correct outcome. This model uses its parameters to map the entries into this desired outcome. To put this abstract concept into more understandable terms we shall entertain some intuitive use cases. For example, we can hypothesise that a given bird is a macaw if she 1)is colourful and 2)has a long tail. The latter, i.e the set of conditions, is a model and the hypothesis is that for an entry X - a tuple of two binary statements about her colourfulness and whether she has a long tail e.g. (0,1) - the model can predict whether the given bird is a macaw - output Y. In other words,

$$f(x_1, x_2) = \begin{cases} 1, & \text{if } x_1 = x_2 = 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where f is the prediction which is 1 if the bird is a macaw and 0 if not. This is a zeroth-order effort at this model, but we can do better. Instead of a tuple of boolean numbers, e.g. $(0, 1)$, we can assign an extent to each of these variables. The bird with the longest tail is the ribbon-tailed

[1] It is worth noting that this report shall only tackle classifying neural networks, and so is adapted to the concepts in *logistical regression*.

astrapia with a tail of length of about $100cm$ [2], and the most colourful bird is the scarlet macaw with up to 5 distinct colour regions [3] - we shall assume these numbers are the maximum possible values. We can normalise our entries using these figures, e.g if a bird has a 20 cm long tail and 3 colour regions the entry will be $(3/5, 20/100) = (0.6, 0.2)$. To consolidate these numbers, we need to introduce a weighting regime such that w_1 and w_2 are the respective weights of the tuple variables. We will also need to introduce a set of criteria for which the bird is most likely to be a macaw, for example

$$f(x_1, x_2, w_1, w_2) = \begin{cases} 1, & \text{if } (x_1, x_2) \cdot (w_1, w_2) > C \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where C is a number defining our criterion. We have now formed a better model for the hypothesis. This can be expanded further by applying knowledge from information theory.

2.2 Information theory and cross entropy

In this section, we shall briefly venture into the domain of information theory. We start by considering the concept of communication. As suggested by C. E. Shannon: *"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point."* [4]

This fundamental problem can be further reduced to the question [4]: how much information is needed to uniquely identify one possibility among a finite number of possibilities. For example, think of four-digit binary numbers. The probability space consists of 16 possibilities, spanning from 0000 up to 1111. Now, imagine we have chosen one possibility, say 1010, how much information do we need to transfer to another person familiar with this probability space to retrieve correctly this chosen number? Intuitively, one could say 4 pieces of information: whether each of the digits is 0 or 1.

R. Hartley suggested in his 1928 paper that the amount of information could be measured by taking the logarithm of the size of the probability space [5]. Shannon formalised this further, giving the following three reasons for his choice of the logarithm [4]:

1. its practical basis, in engineering many variables are proportional with the logarithm of the number of possibilities;
2. human intuition is closer to a logarithmic basis, as per the example above with the number of binary digits to transfer information;
3. and its mathematical properties that would otherwise limit proper calculation.

Therefore, we get

$$h = log_x(\Omega), \tag{3}$$

where Ω is the size of the probability space, and x is the bandwidth of the information unit - e.g. 2 in binary communication. Then, suppose a different question: what is the expected amount of information that we need to send to communicate the outcome of a given probability distribution?

As an example, let's imagine we have N consecutive coin tosses, and we are interested in the number of tails in any N-long string of heads and tails. There are $\Omega = 2^N$ possible combination of these strings, so the amount of information needed to specify a string is:

$$h_1 = log_2(2^N) = N \tag{4}$$

Now imagine we have an extra piece of information: our string has i tails. Take n_i to be the number of combinations that have i tails. If we communicate that we have i tails, to further communicate which exact string has occurred , we need to send more information by $log(n_i)$, as n_i is the size of our probability sample here. We find by taking the expectation value:

$$h_2 = \Sigma_i p_i log_2(n_i), \tag{5}$$

where p_i is the probability of getting i tails. Therefore, providing the extra information of the value of i saves us an expected amount of information:

$$\delta h = log_2(2^N) - \Sigma_i p_i log_2(n_i) = \Sigma_i p_i (log_2(2^N) - log_2(n_i)) \tag{6}$$

and if different combinations are equally likely to occur, $\frac{n_i}{2^N} = \frac{n_i}{\Omega} = p_i$, so

$$\delta h = -\Sigma_i p_i log_2(p_i). \tag{7}$$

Equation 7 is known as Shannon entropy. It shows the expected amount of information saved if we know a feature of the system - i in this case.[2] It is often also interpreted as the measure of uncertainty of obtaining a result. [6] It is immediately evident that Equation 7 is maximum when the distribution is uniform and equal to zero when the distribution is deterministic - often referred to as a one-hot vector [7].

We have, so far, considered a probability distribution $p(X)$ that explains the behaviour of our system. We can extrapolate the concept of Shannon entropy to *relative entropy* to find the compatibility of two distributions with each other. [6] Let's assume we have two probability distributions $p(X)$ and $q(X)$ over the same possibility space Ω. We want to determine the probability of a realisation of X, such as x, belonging to the distributions $p(X)$ and $q(X)$.
We can use our knowledge of Bayes' theorem to calculate the ratio of the conditional probabilities of x belonging to each distribution [8]:

$$\frac{P(x|p)}{P(x|q)} = \frac{p(x)}{q(x)} = \frac{P(p|x)/P(p)}{P(q|x)/P(q)} \tag{8}$$

[2]For a more complete proof please refer to [4] Appendix II.

From earlier, we know that the measure of information is logarithmic. So, encoding a result x in p and q requires different amounts of information given by

$$\delta h(x) = log(p(x)) - log(q(x)) = log\left(\frac{p(x)}{q(x)}\right). \tag{9}$$

This can also be interpreted as the difference in the amount of information that p provides over q for a given event x. To take the expectation we need to select a source distribution. S. Kullback and R. A. Liebler suggested [8] that if the true source of the event is distributed as p and the experimental distribution is in q, then the degree by which the two distributions are separated is given by the expectation of this distance, which by assumption is being sourced from p:

$$H(p, q) = \Sigma_i p_i log\left(\frac{p_i}{q_i}\right). \tag{10}$$

Equation 10 is often referred to as the Kullback-Leibler distance [6] (or KL divergence [9]), and used as a measure of the separation between two distributions that describe the same phenomenon. We will use this to find out how a distribution described by our hypothesis compares with the true distribution of data. If the true distribution of data is a one-hot vector, its entropy is zero, therefore we can rewrite the KL divergence as:

$$H(p, q) = -\Sigma_i p_i log(q_i), \tag{11}$$

which is also known as the cross entropy from the distribution p to q [6].

2.3 Gradient descent

Now that we have the tools to measure the goodness of a hypothesis, let's hypothesise a model to describe a natural phenomenon. Imagine D is a set of data that we have collected from a phenomenon. For example, images from extra-galactic radio sources with active nuclei which belong to two categories of type-I and type-II. [1] For each image i that belongs to type-I, we have $p(I|D_i) = 1$ and $p(II|D_i) = 0$ (and vice versa). We can write this in vector form:

$$\boldsymbol{p_i} = \begin{bmatrix} p(I|D_i) \\ p(II|D_i) \end{bmatrix}. \tag{12}$$

This is a one-hot vector, and the true probability distribution is completely deterministic[3]. This means that the Shannon entropy of $\boldsymbol{p_i}$ is zero for all values of i. Now, we hypothesise that there is a function $\boldsymbol{f_i} \equiv \boldsymbol{f}(D_i)$ that can approximate this probability distribution. The average difference between these distributions can be calculated using Equation 11:

$$L(p, q) = -\frac{1}{N}\Sigma_i \Sigma_j p_{ij} log(f_{ij}), \tag{13}$$

[3]Here, we have assumed that these radio sources can be perfectly classified into either type-I or type-II categories, i.e. perfect information will certainly result in perfect categorisation

where j is used to index the vectors, and N is the number of data points. This quantity, L, shall be referred to as the loss function, and showcases the goodness of f in describing p. The lower the loss function is, the closer f will be to p. Therefore, it is desirable to minimise L by assuming a shape for f. Let's assume $\boldsymbol{f_i} = \boldsymbol{f}([w_1, w_2, \cdots, w_n], D_i)$. This means that the shape of the function f is now characterised by a number of independent parameters $[w_1, w_2, \cdots, w_n]$. The loss function also changes to take the form:

$$L(p, q) \equiv L(\boldsymbol{w}) = -\frac{1}{N}\Sigma_i\Sigma_j p_{ij} log(f_{ij}([w_1, w_2, \cdots, w_n])) \tag{14}$$

where $\boldsymbol{w} = [w_1, w_2, \cdots, w_n]$. This gives us n degrees of freedom. We can change these parameters to minimise the loss function. An efficient way of doing this would be to find its gradient:

$$\boldsymbol{\nabla} L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \cdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix}. \tag{15}$$

The gradient of a function is a vector that points to the direction of the maximal increase. Therefore, to minimise the function, we need to move in the opposite direction to the gradient vector. Thus, we can normalise $-\boldsymbol{\nabla} L$ to find a relative scale (or direction) to change our parameters. This is a direction, but we also need a magnitude. We can define a scale to amplify this direction by - we shall call this learning rate, δl. We get:

$$\boldsymbol{\delta w} = -\left(\frac{\boldsymbol{\nabla} L}{\mid \boldsymbol{\nabla} L \mid}\right) \delta l. \tag{16}$$

Adding this to the previous parameters gives:

$$\boldsymbol{w'} = \boldsymbol{w} + \boldsymbol{\delta w} \tag{17}$$

$$\boldsymbol{f_i'} = \boldsymbol{f}(\boldsymbol{w}, D_i). \tag{18}$$

Thus we have improved our model and it is now more expressive of p than before. Doing this iteratively can improve our model even more. This method of improving a model is called gradient descent [10].[4] Next, we need to find the explicit form of the function $\boldsymbol{f}(\boldsymbol{w}, D_i)$.

2.4 Neural networks

A particular form of the function f is a series of mappings - see Figure 1. Before any mathematical calculation, we need to find a way to mathematically represent the data set D_i. We made an attempt at this in the macaw example in Section 2.1. There are many ways for the pre-processing of data. A popular method for turning images to vectors, for example, is using convolutional neural networks (CNNs). We have used this, but it is beyond the purpose of general introduction. From this point onward, we shall assume that we have found a way to turn the abstract concept

[4] There are many extensions to this, some of them outlined in [10] chapters 3, 4, and 5. As will be discussed later, we have used the adaptive moment estimation (Adam) in our particular implementation.

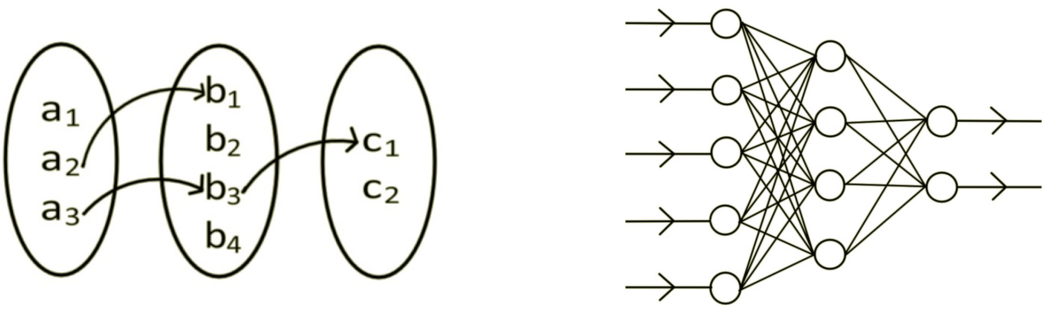


Figure 1: Two series of mappings. The right mapping shows 2 layers of fully connected maps.

of D_i to a vector of numbers $\boldsymbol{D_i}$ with elements D_{ik}. A fully connected linear mapping connects every element from class A to every element in class B. Each connection has an associated weight, therefore all weights can be represented in a matrix, w_{mn}, connecting element m of class A to element n of the class B. We get

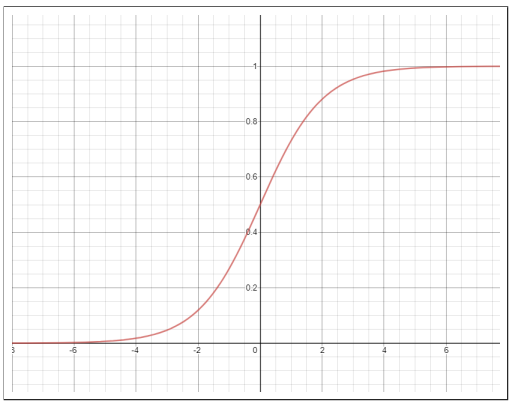
$$\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ b_n \end{bmatrix}. \tag{19}$$

The goal is to get $\boldsymbol{D_i}$ in vector form as the input, and map it to a category distribution vector $\boldsymbol{q}(y_j|D_i)$, where y_j is the category in question (type-I or -II for the radio galaxy types). Just as with the macaw example is Section 2.1, we need to normalise every element, i.e. turn it into a value between 0 and 1. This is usually done by using activation functions - see Fig 2. These are functions that introduce non-linearity into the system as well as normalising the element[5]. A neural network, for our purposes, is a series of linear mappings normalised by activation functions - see Figure 2. There is a different class of activation functions that aim to normalise not only the element but the entire in that class. These turn outputs - known as logits - into a probability distribution. The example we have used is the softmax function:

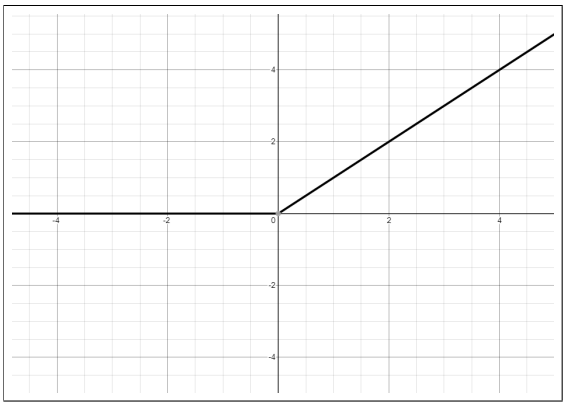
$$p(x_i) = softmax(x_i) = \frac{e^{x_i}}{\Sigma_j e^{x_j}}, \tag{20}$$

where x_i is a logit, and $p(x_i)$ is the probability that the input data belongs to the class i. If we now take the input vector $\boldsymbol{D_i}$ and perform multiple stages of mapping-activation pairs, we can map the data D_i into an approximate distribution of data point $\boldsymbol{q}(type|D_i)$. We can then use gradient descent described in Equation 16 to improve this result. The process of mapping is referred to by forward propagation and the gradient descent by back propagation. To get the vocabulary right, we shall hereafter refer to classes as layers, and the elements as neurons. The first layer containing the data vector $\boldsymbol{D_i}$ is known as the input vector and $\boldsymbol{q}(type|D_i)$ as

[5]A discussion on activation functions can be found in [11]. In the interest of brevity an in-depth discussion is not included.



(a) The sigmoid activation function



(b) the rectified linear activation function

Figure 2: Two activation functions. Note how they both bring non-linearity to the system.

the output layer. Finally, the collection of the particular shape of mappings and the weights is called the *model* and the process of iteratively improving the weights is known as learning. In summary, learning contains:

1. setting up a hypothesis and the explicit structure of the model,
2. turning the data set into a vector,
3. initialising the weights randomly,
4. forward propagating all the data points,
5. calculating the loss function for each data point using Equation 14,
6. back propagating through the network and finding the gradient for each data point,
7. averaging the gradients for all data points and applying the average to the weights, and
8. re-trying steps 4-7 and improving until no noticeable difference is made to the loss value.[6]

After the training, we expect the network to be able to categorise new input data, though not always perfectly. There are uncertainties associated with the result of a network.

2.5 Uncertainties

There are two types of uncertainties in results of a network: *1) epistemic and 2) aleatoric.* Epistemic uncertainty describes the error of the model. This includes the uncertainty in the explicit structure of the model, the extent of the correlation of data with the outcomes using this model, and whether using a different set of parameters could give a more accurate prediction.

[6]Every iteration is called an *epoch*.

Given infinite training data points, we can completely explain away this uncertainty.

The aleatoric uncertainty refers to the information that our data cannot explain. [12] [13] This is further split into homoscedastic and heteroscedastic uncertainties. Heteroscedastic uncertainty is the error which depends on the input data, for example a blurry image has high heteroscedastic uncertainty, especially if the network is trained using very sharp images. Homoscedastic uncertainty is the task-dependent error of the system[7].

2.6 Heteroscedastic uncertainty as a learned parameter

It is possible to obtain the heteroscedastic aleatoric uncertainty as a learned parameter in the network [14]. To do this, we can set up a classification network that learns to infer the heteroscedastic uncertainty on the classification distribution. To even begin measuring this uncertainty, we need to assume that logit values are spread normally about a mean - which is their current value -, and with a standard deviation corresponding to the heteroscedastic uncertainty. In other words, we assume that the logit values we obtain are realisations of an ensemble of values with a normal distribution[8]:

$$\hat{\boldsymbol{y}}_{\boldsymbol{i}} \sim \mathcal{N}(\boldsymbol{y}_{\boldsymbol{i}}, \sigma_i^2) \tag{21}$$

where $\boldsymbol{y}_{\boldsymbol{i}}$ is the current values of the logit vector predicted by the classifier. Note that σ_i is not a vector, this is because the effect of the aleatoric uncertainty is meant to be data dependent, so does not change for different elements in the logit vector. The loss value, therefore, can be the mean of the loss function of the ensemble:

$$= -\frac{1}{N} \int_{-\infty}^{+\infty} \Sigma_i \Sigma_j \mathcal{N}(y_{ij}, \sigma_i^2) p_{ij} log(softmax(y_{ij})) dy_{ij}. \tag{22}$$

Equation 22 is computationally expensive to integrate with each iteration. To simplify this, we can use the Gaussian re-parameterisation trick, where instead of using the original Gaussian distribution we scale the standard deviation by another normally distributed parameter ϵ, and then add the product to the mean, $y_i j$:

$$\hat{y_{ij}} \approx y_{ij} + \sigma_i \epsilon_{ij}^k, \tag{23}$$

$$\epsilon_{ij}^k \sim \mathcal{N}(0, 1) \tag{24}$$

Then, L can be approximated using a Monte Carlo integration:

$$L \approx -\frac{1}{N} \frac{1}{T} \Sigma_i^N \Sigma_j \Sigma_k^T p_{ij} log(softmax(y_{ij} + \sigma_i \epsilon_{ij}^k)), \tag{25}$$

[7] In a multi-tasking network, all input data will have the same homoscedastic uncertainty, but this uncertainty will vary between different tasks, for example depth perception and semantic segmentation in [13]

[8] This sets a *prior* distribution on the logits, much similar to Bayesian neural networks where unique normal distributions are set as priors to introduce a more probabilistic view. See Appendix 5.3 for a discussion on Bayes' theorem.

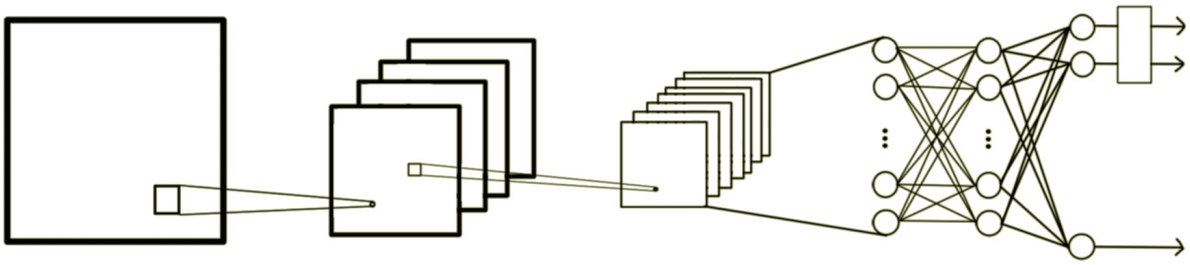


Figure 3: LeNet-KG network. The image is initially passed into 2 convoutional layers

where the ϵ_{ij}^{k} is chosen at random from the distribution in Equation 24 [15][9]. Finally, we need to determine the value of σ_i. The 2017 paper by A. Kendall and Y. Gal suggested that we could have a logit, s_i, which is passed through the softplus function and provides us with the variance σ_i^2 [14]:

$$\sigma_i^2 = softplus(s_i) = ln(1 + e^{s_i}). \tag{26}$$

3 Particular implementation

3.1 Network model

The network used to produce this work was a modified version of the LeNet [16] convolutional network with a kernel size of 5[10] - see Figure 3. This network has two series of outputs: the usual classifier *logits*, and an extra neuron s, whose softplus provides the heteroscedastic aleatoric uncertainty of the output associated with the input image.

3.2 Data sets in use

The primary data set used in this report was the reduced[11] Mirabest database. It includes two classes of radio galaxies based on the Fanaroff–Riley categorisation [18] [1]. This data set includes 729 images of 50x50 pixels. In each training, 80% of this data was used to train the model and the rest was kept for validation. Figure 4 shows some of the images in each of the two classes, FR-I and FR-II.

[9]This is not exactly the same as the loss function in [15]. The superposition of losses in this paper suggests that the network is being trained to tackle two tasks in one, but the structures and the weights are essentially the same so it does not warrant having a superposition of the loss function. For more on this superposition, please refer to 5.3.

[10]The code for this particular implementation could be found in [17].

[11]The complete Mirabest database includes more than two classes of images which is in use in this report. This is explained fully in [18].

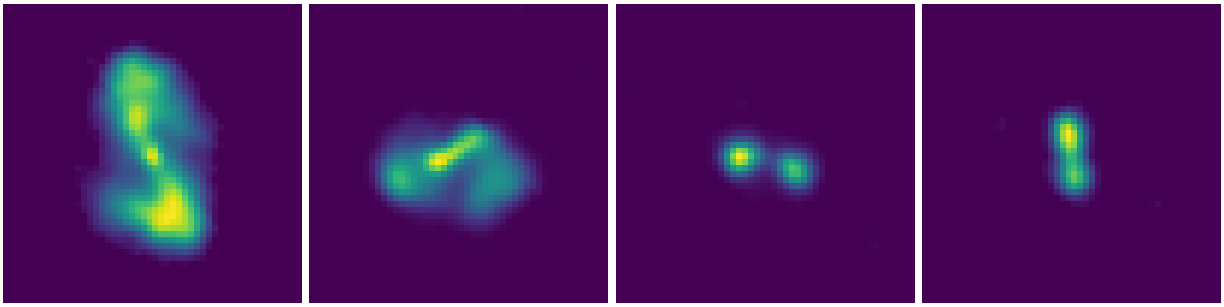


Figure 4: A sample of Mirabest images; the two on the left belong to the FR-I category while the two on the right are FR-II radio galaxies. It is clear from their shapes that type-I galaxies are more spread and have lower received core luminosities. In contrast, FR-II have two dense cores and are of higher received core luminosities. This is consistent with the observation table and conclusions in [1].

3.3 Grid search

A grid search was performed to determine the optimal *hyper-parameters* for this network. The hyper-parameters of a network are the parameters that need to be fine-tuned before the start of the training. We have already encountered an example of this: the learning rate, δl, in gradient descent. There can be many hyper-parameters in a network; in our modified LeNet-5 network, these are:

- batch size (32)[12]
- momentum (0.9)
- scheduler gamma (0.8)
- weight decay (1×10^{-2})
- scheduler step size (5)
- learning rate (1×10^{-3})

We shall refrain from explaining these in detail in the interest of brevity, but a detailed explanation can be found in [19]. This search takes in a range of values for each combination of the hyper-parameters.

3.4 Transformations

A set of transformations were applied to the images, and these were applied every time images were loaded[13]. These transformations were:

- centre crop: crops the image to fit the input size of the network,
- normalise: normalises the image intensity values, and
- rotation: randomly rotates the image - cropping the edges to do so.

[12]The values in parenthesis show the optimal values obtained.
[13]The *data loader* object in PyTorch applies transformations with every call to the object.

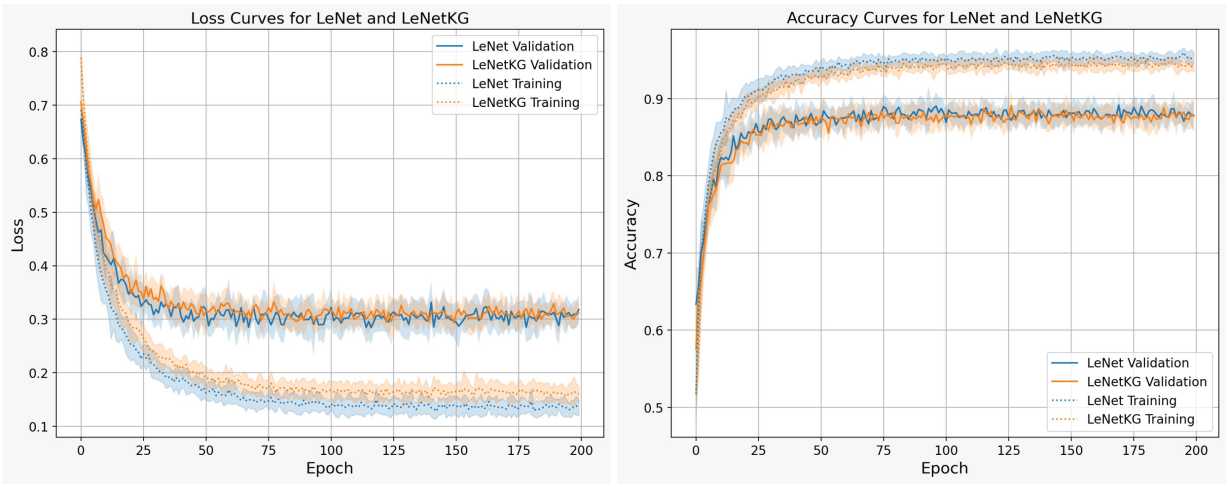


Figure 5: The loss and accuracy curves for the standard LeNet-5 and LeNet-KG. In each case, the emboldened line shows the mean of the distribution and the faint cloud on the sides represents the standard deviation of 20 runs.

4 Experimentation and Results

4.1 Deviation from the original LeNet-5

Since the LeNet-5 network was modified, an important step was to see the effect on the accuracy of the models. A measure often used to describe the training of a model is the validation accuracy. This is where a set of data previously unbeknown to the network is forward propagated through the network and the results compared with the true classification. The portion of this *validation* data that the network correctly predicts is known as the validation accuracy. Figure 5 shows a comparison between the validation accuracy of the original and the modified LeNet-5 (LeNet KG[14]) networks.

We can see from the accuracy curves, that the accuracy of the model is unaffected by the changes. Furthermore, it is evident from the training and validation curves for both networks that validation losses are consistent with each other, but the LeNet-KG has a significantly larger training loss.

4.2 Augmentation of data

To see the behaviour of the aleatoric uncertainty, it was appropriate to augment the data and observe how various sources of noise changed the distribution of the aleatoric uncertainty on the outputs. Figure 7 shows the distribution of the aleatoric uncertainties before any augmentation, and Figure 6 shows the images with the highest and the lowest uncertainties in each group.

[14]This is referred to as LeNet KG in the figures. **KG** is to attribute this particular modification to A. **K**endall and Y. **G**al in [14].

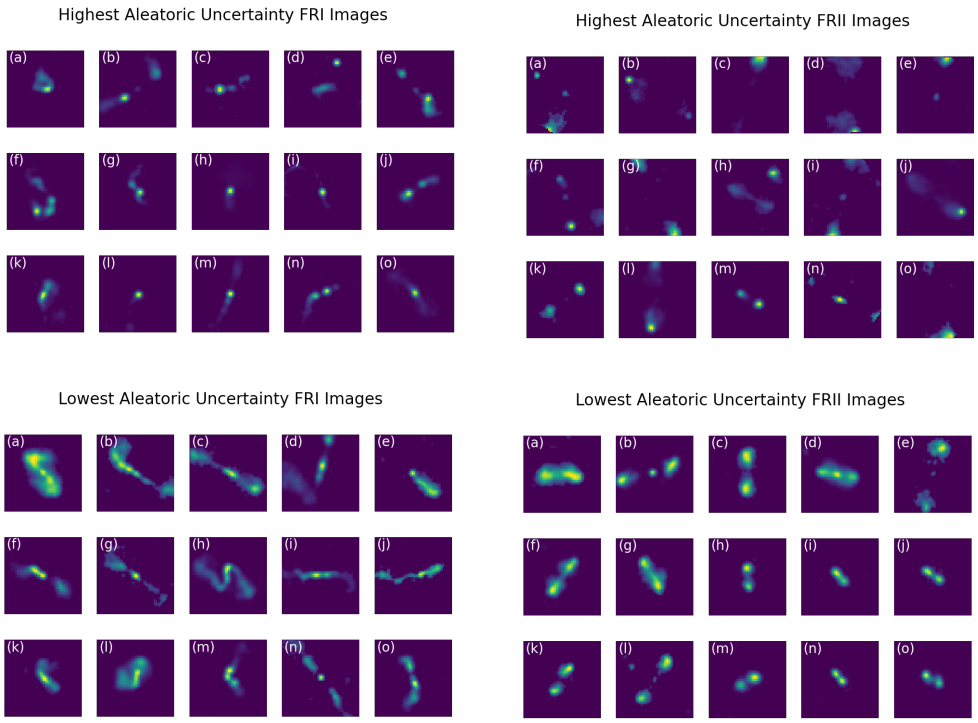


Figure 6: Mirabest images with the highest and lowest aleatoric uncertainties. As illustrated, the highest FR-I and FR-II types are difficult to distinguish.

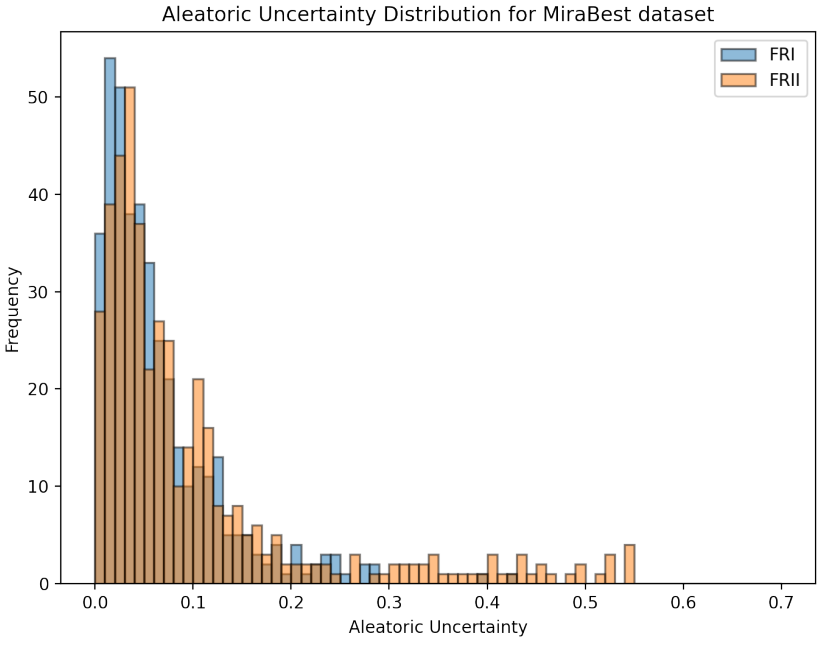


Figure 7: The distribution of the aleatoric uncertainty after 200 epochs of training. It is clear that the network has identified the FRII type to have higher aleatoric uncertainties than FRI.

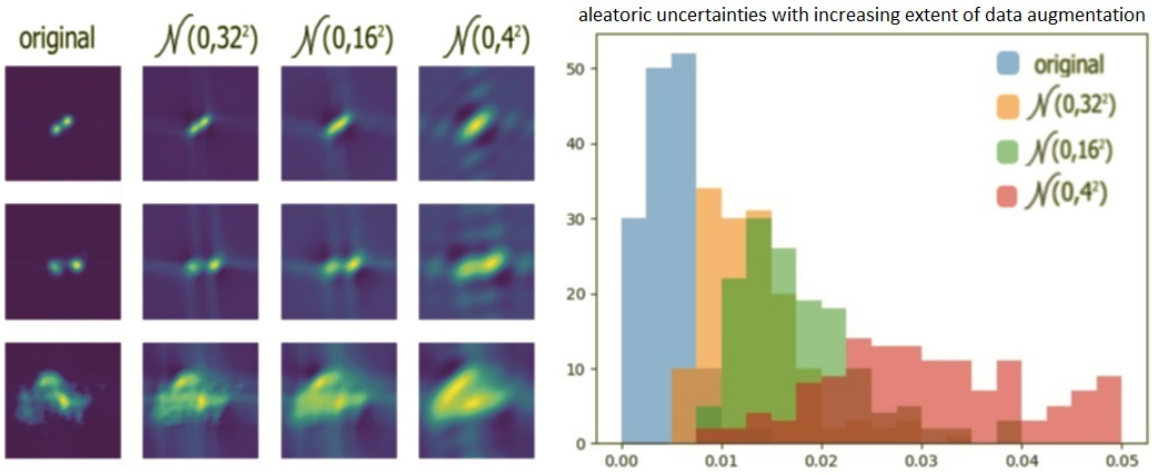


Figure 8: Increasing degree of blurriness both spread and increased the distribution of the aleatoric uncertainty.

4.2.1 Low pass filter

For this section, the images were passed through a low-pass filter to decrease the sharpness of the edges in the images. This is similar to removing higher orders of the Fourier transform of a function to keep the general shape but decrease the extent of description within it. For this purpose, the images were Fourier transformed, resulting in real and imaginary parts. These image parts were then separately convolved[15] with the same Gaussian distribution to diminish the effect of the higher order Fourier coefficients. These two parts were then combined into a blurrier image of the radio galaxy. The results in Figure 4.2.1 show examples of this augmentation as well as the change in the distribution of the aleatoric uncertainty.

4.2.2 Gaussian noise

In this section, a Gaussian noise was added to the image to distort the features of the image, therefore intuitively increasing the aleatoric uncertainty. Figure 9 shows how this affected the results. We can see that with increasing noise on the data, the distribution of aleatoric uncertainty is becoming less spread and closer to 0.

4.2.3 Kernel blurring

Next, kernel blurring was investigated as a different blurring mechanism to the low pass filter. The adopted version of this blurring was the box blur. A box blur with a kernel of size n takes the average of the neighbours a distance of $\frac{(n-1)}{2}$ pixels away from the middle pixel. Thus, the bigger the size of the kernel, the more pronounced the blurring effect will be. The particular implementation for this was using the python library *Albumentations* [20].

[15]This experiment was slightly modified from a complete cut-off of high frequencies to reduce the *egg-tray* effect evident on the images.

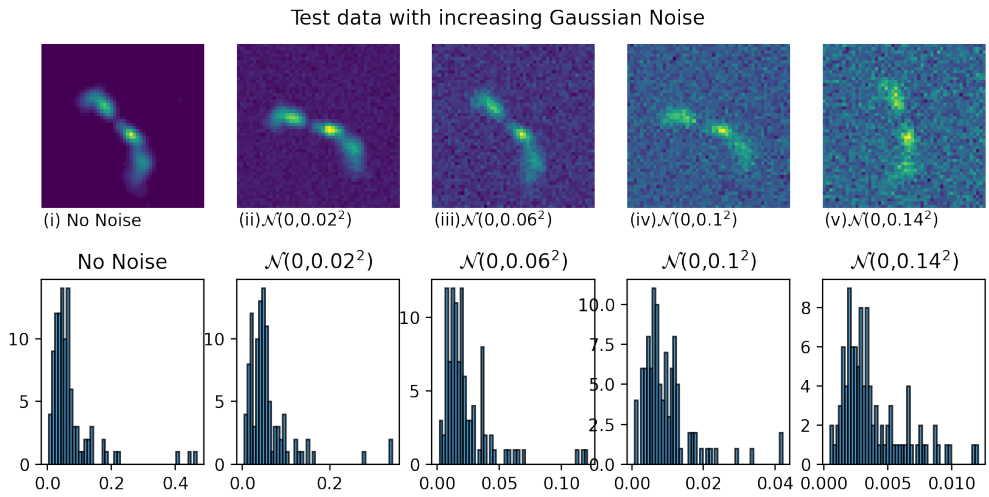


Figure 9: Increasing degree of Gaussian noise added to the images. The effect of rotation is evident in the set of images.

The results of this blurring are shown in Figure 10. It is clear that the spread and the mean of the distribution is increasing with the increasing extent of the blur.

4.3 Pruning the data

After about 200 epochs, we have a network that can predict the aleatoric uncertainty of its classification. We shall now explore the ways to use this information to include or reject each piece of training data.

A natural step is to prune the training data based on this aleatoric uncertainty, and then use this pruned data set to train a new standard LeNet-5 model from scratch. To create a reference point for this model, we need a network to compare it with. For this purpose, we shall train the same structure on the randomly pruned data set. This ensures that we can capture the significance of the selective pruning. Figure 11 shows the results of this exercise for various pruning protocols. The training data set was split into 5 quintiles based on their aleatoric uncertainties. Models were trained on these and compared with a model based on a random selection of 20% of the data. The important features are:

1. the second lowest has the lowest validation loss and the highest accuracy,
2. the worst performing by far is the highest quintile,
3. the validation loss of the highest quintile increases instead of decreasing, and
4. the validation loss of the lowest quintile starts to increase at around epoch 27 after decreasing. At the same time its validation accuracy is decreased, too.

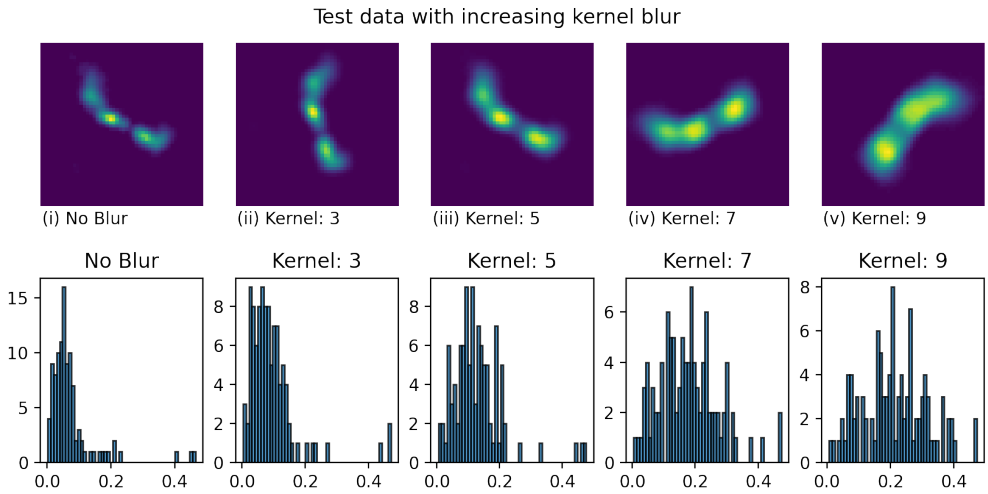


Figure 10: Augmented data using the box blur.

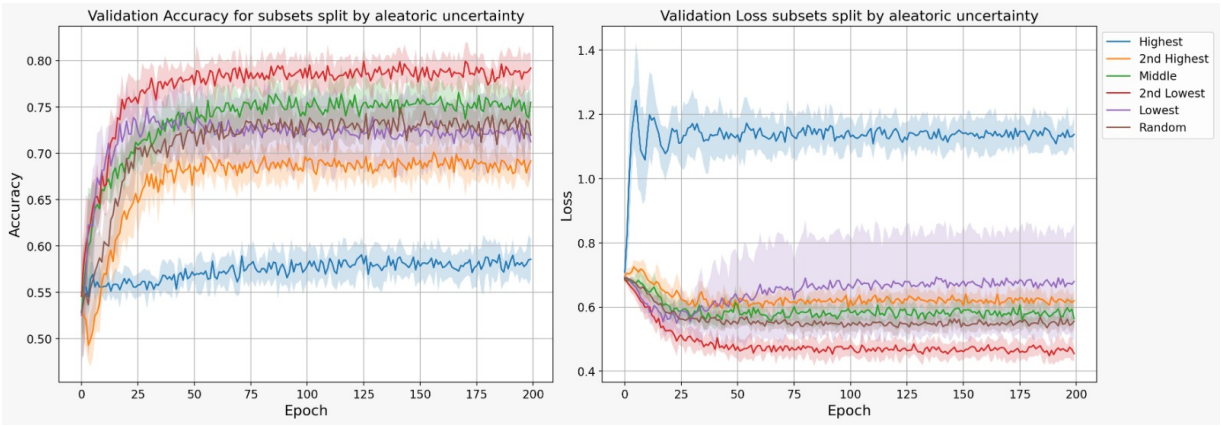


Figure 11: LeNet-KG when trained with each of the 5 quintiles of the aleatoric uncertainty distribution.

5 Discussion

5.1 Deviation from standard LeNet-5

As we saw in Section 4.1, the validation accuracy curves for the two networks are consistent, suggesting that there is no difference in their performance. However, the training loss of the standard LeNet-5 is significantly less than that of LeNet-KG. This is also evident in the training loss curves. This result is likely due to the averaging over the ensemble. The logits are deviated by the prior distribution described in Section 2.6, and thus the network can not train as well as a standard network that trains for a deterministic value. In other words, the ensemble average of the deviated logit is less predictable than the value of the logit itself and therefore it is harder for the network to minimise it.

5.2 Augmentations

The results from Sections 4.2.1 and 4.2.3 are consistent. They both indicate an increase in the aleatoric uncertainty with an increase in the extent of augmentation, as expected. However, the Gaussian noise in Section 4.2.2 displays the opposite behaviour. A speculation could be that the Gaussian noise is making the prominent features of the radio galaxies more distinguished post normalisation.

5.3 Pruning

The results obtained in Section 4.3 were unexpected. Especially the second lowest quintile performs significantly better than all the rest. This result is likely due to the convenient middle-ground split of data: the data has a generally low aleatoric uncertainty to generalise the model but also there is enough variety of feature in this quintile to show the network some of the more difficult cases. In the future, it would be interesting to test if this quintile could replace the entirety of data to increase the learning efficiency.

The highest quintile is the worst performing by far, and this is likely because the data in this batch is so similar that any inference would be difficult - see Figure 6. The validation loss for this quintile skyrockets from the start. Following the same logic, this might be due to the validation sample being radically different to the data in this quintile, and so, training on this quintile makes the network learn categorisation features that are not necessarily representative to what the categories physically mean, and when it is evaluated by the validation sample it provides semi-random results, as evident from its accuracy curve.

Finally, the validation loss of the lowest quintile takes an upward turn. This is likely due to the possibility that the data points in this bracket are easy to categorise, so it overfits the categories to the extent that it struggles to recognise edge cases in the validation sample.

These observations indicate that sampling based on the aleatoric uncertainty has a physical significance. They also indicate that for the Mirabest data set training based on the data with only low or high aleatoric uncertainty is likely not a good strategy.

References

[1] B. L. Fanaroff and J. M. Riley, "The Morphology of Extragalactic Radio Sources of High and Low Luminosity," *Monthly Notices of the Royal Astronomical Society*, vol. 167, pp. 31P–36P, 04 1974.

[2] S. N. O. D. Grzimek, B., *Grzimek's animal life encyclopedia*. American Zoo and Aquarium Association, 2003.

[3] W. Lantermann, *Encyclopedia of macaws*. TFH Publications, 1996.

[4] C. E. SHANNON, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, p. 379–423, 1948.

[5] R. V. L. Hartley, "Transmission of information," *Bell System Technical Journal*, vol. 7, no. 3, pp. 535–563, 1928.

[6] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 2015.

[8] R. A. L. S. Kullback, "On information and sufficiency," *The Annals of Mathematical Statistics*, 1948.

[9] J. A. T. THOMAS M. COVER, *ELEMENTS OF INFORMATION THEORY*. John Wiley Sons Inc, 2006.

[10] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley Sons Inc, 2003.

[11] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[12] Y. Gal, "Uncertainty in deep learning - phd thesis," *Oxford University*, 2016.

[13] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," 2018.

[14] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," 2017.

[15] E. Kochkina and M. Liakata, "Estimating predictive uncertainty for rumour verification models," 2020.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] "Predicting aleatoric uncertainties for images of radio galaxies." Last modified on 10th Jan 2021: https://github.com/kordham/aleatoric-uncertainty.

[18] H. Miraghaei and P. N. Best, "The nuclear properties and extended morphologies of powerful radio galaxies: the roles of host galaxy and environment," *Monthly Notices of the Royal Astronomical Society*, vol. 466, pp. 4346–4363, 01 2017.

[19] M. Feurer and F. Hutter, *Hyperparameter Optimization*, pp. 3–33. Cham: Springer International Publishing, 2019.

[20] "Albumentations github repository." Accessed in Nov 2020: https://github.com/albumentations-team/albumentations.

[21] "Desmos, the online graphing calculator." Accessed in Jan 2021: https://www.desmos.com/.

[22] R. M. B. communicated by Mr. Price in a letter to John Canton, "An essay towards solving the problem in the doctorine of chances.." Philosophical transactions of the Royal Society of London., 1763.

Acknowledgements

A special thanks to Professor A. Scaife for the opportunity of doing this project and guiding me along the way, and to my MPhys partner and friend A. Utting for his expertise and professionalism.

Aside from providing Figures 5, 7, and 11, Aydin performed the Sections 4.2.2 and 4.2.3 and kindly provided me with the results.

Thanks to the JBCA team for lending the Lofar-5 machine to us so we could run the heavier trainings on their capable GPUs.

Thanks to the lovely artist, P. Blum, who kindly drew Figures 1 and 3 for me.
Finally, Figure 2 was produced using the online graphing tool Desmos [21].

APPENDIX

APPENDIX I: Conditional probabilities and Bayes' theorem

Conditional probabilities form an important part of our understanding of statistics. As before, we shall employ an example to better grasp the concept. Imagine you are a medical professional at a Covid ward, and a patient tells you that she has developed a continuous cough. Furthermore, given the latter information you are interested in determining the probability of whether this person has Covid-19. This is denoted as $P(covid|cough)$, where | indicates the conditioning of the problem to the happening to its right - coughing, in this case. Intuitively, we know that this is given by:

$$P(covid|cough) = \frac{P(covid \cap cough)}{P(cough)}. \tag{27}$$

This is because the condition has shrunk the possibility space to all the ones where *cough* is present. We can also write $P(cough|covid)$ in similar terms:

$$P(cough|covid) = \frac{P(cough \cap covid)}{P(covid)}. \tag{28}$$

We know that $P(cough \cap covid) = P(covid \cap cough)$. Therefore, by equating this term in both equations we get:

$$P(covid|cough) = \frac{P(cough|covid)P(covid)}{P(cough)}. \tag{29}$$

This is known as the Bayes' theorem [22]. All of these terms play important roles:

- $P(cough)$: Evidence - prob. that the a person coughs for any reason.
- $P(covid)$: Prior - prob. that any given person has Covid.
- $P(cough|covid)$: Likelihood - prob. that a coughing person has Covid.
- $P(covid|cough)$: Posterior - prob. that a Covid patient has the coughing symptom.

Bayes' theorem is useful because often likelihood is more accessible than the posterior probability distribution, whereas the latter is more desirable.

APPENDIX II: Multi-tasking networks

In general, if there are N tasks to learn, the overall loss function becomes:

$$L_{tot} = \Sigma_k \alpha_k L_k, \tag{30}$$

where L_k is the loss function associated with the specific task k, and α_k is its respective coefficient. [13]