From Pixels to Prediction: Exploring Statistical Machine Learning for the Classification of Medical Image Data

A thesis submitted to the University of Manchester for the degree of Bachelor of Science in the Faculty of Science and Engineering

15th May 2024

Jack Hodgkinson
Department of Mathematics

Contents

List of Figures

List of Tables

Abstract

How can we enhance the accuracy and efficiency of medical image diagnosis, considering the challenges posed by image quality, complexity, and the limitations of human perception?

The paper proposes an approach of utilising statistical machine learning techniques to assist with medical image classification. An extensive study of current methods of classification from the subjects of statistics and machine learning, along with an analysis of their capabilities, is completed on three different datasets from MedMNIST+ (Yang et al., 2021, 2023, 2024). Although the methods have the potential for success in other data modalities, the research concentrates on data from the domain of oncology. The analysis evaluates how image structure, including resolution and colour, affects the performance of methods. This forms the basis of conclusions regarding optimal data format to optimise classification performance. Whilst only images sourced H&E stained colorectal tissue samples, breast ultrasound scans and dermatoscopy are evaluated, the results may form the basis for future research on other types of medical images.

There is optimism for a successful future of statistical machine learning for the benefit of patient outcomes and the role of clinicians.

Declaration

I declare this thesis is my original work. No portion of the work referred to in the thesis 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 extend my sincere gratitude to my project supervisor Korbinian Strimmer. In addition to being an excellent project supervisor, he has served as a mentor for me in my final year of my BSc in Mathematics. His support with both the thesis and applications to future study has proved to be invaluable, and I believe that the project's success speaks highly of his commitment. The guidance received on this project opened my eyes to the brilliant world of statistical machine learning and motivated me to explore related topics in the future.

I would like to give thanks to my incredible family who have provided unwavering support throughout the thesis. Firstly my parents have inspired me to persevere in times of hardship and periods of academic intensity. I appreciate the support more than ever at this time, not forgetting the regular tea refills provided during writing at home! Secondly, I would like to thank all four of my grandparents for being my biggest supporters, championing me to work harder and do myself and them proud.

I would like to thank all of my incredible friends who have been nothing short of amazing. Whilst all of my friends deserve recognition, I would especially like to thank my friends Sina and Ella for their constant support during this year, sharing both the good and bad experiences. Furthermore, I want to extend my thanks to my friend Liv, a qualified diagnostic radiographer, for helping me to comprehend the many forms and applications of medical imaging in a clinical setting. Ensuring I produce the best possible paper, I want to thank my friend Katie, whose support in proofreading the thesis has been pivotal for my success.

Finally, I want to dedicate my thesis to everyone who has been affected or continues to be impacted by cancer. Hopefully, this research demonstrates the effectiveness of statistical machine learning in medical image classification and that some of the techniques discussed in this paper will prove invaluable in the field.

1. Introduction

Image data is used widely in the field of medicine to aid in diagnostic decisions. Whilst analysis of the images is conducted by experienced health professionals, the nature of humankind means that errors are possible. In addition, when multiple candidates are involved in classification, disagreements may arise due to the large range of possibilities. As medical images tend to be specific and detailed, unexpected image features may confuse clinicians, leading to incorrect classification. This has a huge impact on patient outcomes, in which the consequence of misdiagnosis could lead to serious deteriorations in health, and result in fatalities.

One of the largest burdens to human health is cancer. Ferlay et al. (2021) reported that there were an estimated 19.3 million new cases of cancer and almost 10 million deaths as a result of the disease worldwide in 2020. Whilst there are attempts to reduce these numbers, the disease is often detected late into its development, due to the unpredictability of symptoms. Patients may present as asymptomatic or dismiss symptoms as being indicative of less serious conditions.

In cancer diagnosis, early detection of abnormalities can enhance both treatment possibilities and prospects. Early detection allows for disease management strategies to be less invasive and aggressive, resulting in an improved quality of life due to reductions in the physical and mental impact. As a result, mortality rates are reduced and the burden on healthcare systems lessens. Furthermore, early detection reduces the risk of metastasis, resulting in longer and healthier lives for patients.

Identifying early symptoms of cancer can be difficult, and both general practitioners and oncology specialists frequently overlook or misdiagnose the disease. This is a result of the intricacy of the human body, which is difficult to depict in a single image. This is particularly true for low-resolution images, where the numerous small structures of the human body are unable to be represented accurately.

However, this cannot just be blamed on the modality of the data. Wise (2022) reported that in the United Kingdom, the National Health Service (NHS) is short of 12,000 hospital doctors and more than 50,000 nurses and midwives. Because of this, hospitals are largely understaffed, adding extra strain on clinicians and increasing their levels of stress and fatigue. This will undoubtedly have an impact on the capabilities of healthcare professionals, meaning that misjudgements are more likely to occur

Something must be done to avoid these errors in cancer misdiagnosis. Whilst statistical approaches have long been used in classification, their full capabilities in medicine have only recently been acknowledged In the last decade, the revolution of machine learning and predictive modelling techniques, both of which are built

on foundations from statistics, has provided the field of medicine with hope of a solution to this problem. Even though the field is still emerging, previous studies such as Tang et al. (2009); Shen et al. (2017) prove the potential of the techniques in medical practice.

The research aims to identify a high-performing classification model to aid clinicians in cancer diagnosis using medical imaging. The analysis will investigate the success of different statistical and machine learning methodologies in generating precise and dependable classifications. Numerous measures, including accuracy, F1 score and log loss will be used for evaluation. The prediction score, a newly created metric for the thesis, will supplement the existing metrics. This equips the study with a set of robust performance measures to cover crucial elements of diagnosis, including predictive accuracy and model confidence.

The analysis focuses on breast, skin and colorectal cancers. I will utilise the BreastMNIST, DermaMNIST and PathMNIST datasets from the MedMNIST+ collection by Yang et al. (2021, 2023, 2024). The wider MedMNIST+ collection has been created to facilitate the development of novel machine learning approaches for the classification of both 2D and 3D medical images, providing users with validated, standardised and labelled data for experimentation. The collection includes various data modalities including chest X-rays, breast ultrasounds and dermatoscopic images. The release of MedMNIST+ (Yang et al., 2024) introduced higher resolution data, with image resolutions of 64×64, 128×128 and 224×224 being made available. Prior to this, the datasets were only available as MNIST-like Deng (2012) 28×28 resolution images. I focused my analysis on a subset of 2D data from the collection, meaning no analysis of 3D data was completed. Examination of the effect of increasing resolution and colour transformations on the performance of the classification is completed.

The paper inspects the performance of four different classification methods: linear discriminant analysis, logistic regression, convolutional neural network and XGBoost. Each model will be assessed by standardised metrics and an attempt to optimise performance will be conducted through tuning.

The thesis will be organised as follows. The background provides a brief overview of elementary concepts in statistical machine learning for those who are not familiar with the foundational mathematics. The methods section will then describe the detailed mathematics behind each method, with an added focus on the crucial developments in history to derive the techniques used in the paper. The datasets and experimental design chapter provides further background on the datasets used in the research and describes the exact configuration used for modelling. The later chapters discuss the outcomes from the modelling, balancing both the successes and limitations of the investigation. All findings will be summarised in the conclusion, providing readers with a concise reflection on the successes of the paper.

2. Background

2.1. Overview

This section serves as an introduction to some of the concepts which will be discussed throughout the thesis. For those readers unfamiliar with elementary mathematical concepts behind statistical machine learning techniques, this section provides a brief introduction to crucial topics in the paper.

The thesis assumes knowledge of elementary mathematical and statistical concepts such as vector and matrix notation, partial differentiation, the chain rule, linear algebra, maximum likelihood estimation and linear regression.

2.2. Optimisation

Mathematical optimisation is the process of minimising or maximising a function concerning variable constraints (Nocedal and Wright, 1999). It is an important step in the modelling process to ensure that the correct solution is achieved. Optimisation can be completed through numerous methods, known as algorithms, for the computation of a function's minimum or maximum value. In statistical machine learning, often we are concerned with minimisation problems, such as minimising the loss function to construct an accurate predictive model.

For an arbitrary function $f(\boldsymbol{\theta})$, $\boldsymbol{\theta} = \{\theta_1, ..., \theta_n\} \in \boldsymbol{\Theta}$, subject to constraint functions $c_i(\boldsymbol{\theta}) = 0, c_i(\boldsymbol{\theta}) \geq 0$, one can define an optimisation problem by the following notation:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} f(\boldsymbol{\theta}) \quad \text{subject to} \quad \begin{cases} c_i(\boldsymbol{\theta}) = 0 \\ c_i(\boldsymbol{\theta}) \geq 0 \end{cases}$$

Optimisation problems can take many forms, dependent on decision variables, constraint functions and the function being minimised. Decision variables represent the choices or actions that can be adjusted to optimise the objective function. These may be linear, non-linear, convex, non-convex, or involve integer values. Often in non-linear optimisation, convergence to the global solution is difficult. Thus the local solution, a minimum point that is not the minimum of the whole function, is located instead.

Furthermore, algorithms can be classified into two main categories concerning uncertainty. Deterministic optimisation deals with problems when the model is completely known, meaning that the parameters and constraints of the model are known and constant throughout the refinement process. On the other hand, stochastic optimisation handles probabilistic outcomes and allows variability with the parameters and constraints of the model. For further details on determin-

istic optimisation, please refer to Nocedal and Wright (1999) and for stochastic optimisation, refer to Birge and Louveaux (2011).

2.3. Gradient Descent

Gradient descent is a first-order deterministic optimisation algorithm that aims to minimise a function by "travelling" in the direction of steepest descent. In machine learning, gradient descent is an iterative process where values of model parameters are edited to fit a model which optimises a loss function. Gradient descent utilises the entirety of the training dataset to compute both the value of the loss function and its gradient, enabling all parameters in the data to be updated in each iteration.

Model parameters are initially assigned by one of the following methods: assignment to the value 0, random determination, assignment due to the data domain, assignment influenced by current literature or initialisation from a pre-trained model (James et al., 2023). By taking the derivative of the loss function with respect to the parameters simultaneously, one can take "steps" towards the optimal value. The size of the step is determined from the slope and the learning rate η, a pre-specified, constant parameter that controls the rate of convergence of the optimisation algorithm, reducing the risk of overfitting. A smaller learning rate implies that a larger number of iterations are required, but results in a better fit of parameter values. The gradient descent algorithm converges for a step size equal to or extremely close to zero, such as 0.001. The method can also limit the number of iterations it completes to find the optimal parameter value. Whilst this could lead to non-optimal parameters, it can significantly reduce model training times.

For a parameter vector $\boldsymbol{\theta}$, the gradient descent algorithm can be applied to the individual parameters θ_i simultaneously to minimise the loss function $\mathcal{L}(\boldsymbol{\theta})$ until it converges below a threshold ψ by the following process:

Algorithm 1 Gradient Descent

Initialise the values of $\boldsymbol{\theta}_0$, select a learning rate η and convergence threshold ψ.
while $\eta\nabla\mathcal{L}(\boldsymbol{\theta}) > \psi$ **do**
    $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \eta\nabla\mathcal{L}(\boldsymbol{\theta}_{i-1})$
    if $\eta\nabla\mathcal{L}(\boldsymbol{\theta}_{i-1}) \leq \psi$ **then**
        terminate
    else
        $i = i + 1$
    end if
end while

2.4. Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an adaptation of gradient descent. It computes gradients on randomly drawn training examples from the wider training set, instead of utilising the whole set (Murphy, 2022). This is advantageous in cases of

observation-rich and high-dimensional data, as gradient calculations would be required for every feature in every sample in ordinary gradient descent.

Stochastic gradient descent chooses one or a small number of samples for each step of derivative calculations, dramatically reducing the required number of calculations. The subset of samples selected is called a batch, and the batch size is often a parameter that can be specified in models.

For a parameter vector $\boldsymbol{\theta}$, the stochastic gradient descent algorithm can be applied to the individual parameters θ_i simultaneously to minimise the loss function $\mathcal{L}_b(\boldsymbol{\theta})$ using b randomly selected samples until it converges to a threshold ψ by the following element-wise process:

Algorithm 2 Stochastic Gradient Descent

Initialise the values of $\boldsymbol{\theta}_0$, select a learning rate η and a convergence threshold ψ.
while $\eta \nabla \mathcal{L}_b(\boldsymbol{\theta}) > \psi$ **do**
    Randomly select a subset of b samples from the dataset of size M:
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_{i-1} - \eta \nabla \mathcal{L}_b(\boldsymbol{\theta}_{i-1})$
    if $\eta \nabla \mathcal{L}_b(\boldsymbol{\theta}_{i-1}) \leq \psi$ **then**
        terminate
    else
        $i = i + 1$
    end if
end while

2.5. Newton-Raphson Method for Optimisation

The Newton-Raphson method, named after Isaac Newton and Joseph Raphson, is a technique to find approximations of the roots of a function. Also known as Newton's method, it is an iterative method to find a solution of $f(x)$ defined by:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

where $f'(x_k)$ is the first derivative of the function $f(x)$ (Galã, 2000). The method is repeated until the absolute difference between predictions reaches a threshold $\psi \in \mathbb{R}$ i.e. $\|x_{k+1} - x_k\| < \psi$ or when a maximum number of iterations T has been reached.

Newton's method can be applied to unconstrained optimisation problems to minimise a function $f(x)$. The goal is to find a value of x that minimises the derivative $f'(x)$. Newton's method for optimisation problems can be defined as:

$$x_{k+1} = x_k + \frac{f'(x_k)}{f''(x_k)}, \ k \geq 0$$

The convergence criteria remain the same. Newton's method exhibits quadratic convergence close to a local minimum when the initial guess is near the optimal

solution. Hence, the prediction error decreases quadratically with each iteration contributing to rapid convergence. However, global convergence is not guaranteed. This depends on the properties of the function being minimised and the continuity of the derivatives.

Whilst explanations here have been limited to one-dimensional cases, the methods can be extended to higher dimensions, by applying the same process element-wise.

3. Methods

3.1. Overview

The methods section explores both probabilistic and non-probabilistic techniques from statistics and machine learning for the classification of medical images. All methods were tested using the experimental design as discussed in 4.3. The methods are examples of supervised learning, hence the data has been partitioned into training, validation and test sets. Furthermore, the analysis data is comprised of continuous features and discrete, categorical outputs hence the methods will be formulated specifically for this case. For other data types, please consult relevant literature. This section explains the mathematics behind each method used in the analysis.

3.2. Statistical Methods

3.2.1. Linear Discriminant Analysis

Overview

Linear Discriminant Analysis is a probabilistic classification method used in supervised learning that uses a linear combination of the predictors to separate the data into categories, each of which has a probability associated with a potential class allocation. Fisher (1936) first proposed the method as a technique to discriminate between two different types of flowers. It is an approximation of the Bayesian discriminant rule, or Bayes classifier, which is a classification technique based on the famous probability theorem from Reverend Thomas Bayes in his posthumous essay *An Essay towards solving a Problem in the Doctrine of Chances* (Bayes, 1763).

Bayes' Theorem and Bayesian Statistics

Bayes' Theorem is a method to calculate conditional probabilities. It calculates the likelihood of an event X happening given another event Y has already occurred. The theorem assumes that the event Y has occurred, i.e. the probability of event Y occurring cannot be zero, $\mathbb{P}(Y) \neq 0$. Hence the basic theorem states:

$$\mathbb{P}(X|Y) = \frac{\mathbb{P}(X)\mathbb{P}(Y|X)}{\mathbb{P}(Y)} \tag{3.1}$$

where $\mathbb{P}(X)$ is the marginal probability of event X occuring and $\mathbb{P}(Y|X)$ is the conditional probability of event Y happening given X has already happened (González-Mejía et al., 2015). $\mathbb{P}(Y)$ is the marginal probability of the event Y occurring. $\mathbb{P}(X)$ is referred to as the *prior probability*, the prior knowledge that an event will happen and $\mathbb{P}(X|Y)$ the *posterior probability*, the updated probability that

X will happen given Y will happen.

This particular theorem gave rise to a new perspective on statistics. The development of Bayesian statistics viewed probability as a degree of belief, where prior knowledge is updated based on new information. This opposes the frequentist school of thought, where probability is considered a measure of frequency.

This offers a new approach to statistical inference. Bayesian statistics estimates parameter values based on the data and assumes that each parameter is a random variable, providing an alternative method to estimating parameters based on the whole population as in maximum likelihood estimation. For the single parameter case, the theorem in 3.1 can be modified to utilise the probability mass/density function (p.m/d.f) $f(\theta|D)$, dependent on the data D and parameter θ, to generate the Bayes theorem used in Bayesian statistics:

$$f(\theta|D) = \frac{f(D|\theta)f(\theta)}{f(D)} \tag{3.2}$$

Now, $f(\theta)$ represents the p.m/d.f of the prior distribution, indicative of current beliefs on the possibility of parameter values. The prior probability distribution can be elicited from reviewing previous literature of similar studies or subjective beliefs of the priors. Alternative approaches include using weakly informative priors when minimal knowledge is available. The probabilities of parameter values are considered before data collection and modelling.

Frequentist statisticians are concerned with the likelihood function, denoted $f(D|\theta)$ to represent the joint probability of the data conditioned on a specified parameter value. However, Bayesian statisticians utilise the likelihood to check the chance of the data coming from specified parameter values and update the prior distribution with relevant information. This in turn influences the posterior distribution.

The marginal probability mass/density function of observing the data is denoted by $f(D)$. This represents all possible ways of generating the data using all possible parameters $\theta \in \Theta$. This is normally completed post-data collection so that the probability can be represented as a real number. Derivation of the marginal p.m/d.f involves integrating the prior and likelihood functions, i.e.:

$$f(D) = \int_{\Theta} f(D|\theta)f(\theta)d\theta, \ \theta \in \Theta$$

This is the posterior distribution, which is a portrayal of the updated beliefs of likely and unlikely parameter values based on data observation. The p.m/d.f of the posterior distribution is used in conducting inference.

Whilst the formula appears simple, the calculation is not. The distribution of the likelihood may not be chosen as it is decided by the data, but statisticians can carefully choose the prior distribution to aid in finding the posterior. The aim is to choose a conjugate prior, i.e. when the prior and posterior come from the same family of distributions. Another option of prior estimation is through sampling, which is an approach used in Markov Chain Monte Carlo (MCMC) algorithms.

A Brief Introduction to Mixture Models

A mixture model is a statistical model that represents the probability of a random variable as a mixture of two or more component distributions. A weighted sum of the component distributions, with the weights denoting the chances of each component being chosen, is how the overall probability distribution is expressed.

For a finite mixture model based on observed data $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_M)$ containing M independent and identically distrbuted (i.i.d) samples, there is a known number K classes, where the probability of class $k \in \{0, ..., K-1\}$ is represented by $\mathbb{P}(k) = \pi_k,\ \sum_{k=0}^{K-1} \pi_k = 1$. Note that each class k is modelled by its own distribution, denoted F_k with individual parameters θ_k and density $f_k(\mathbf{x}) = f(\mathbf{x}|k)$. Each class k has conditional means and variances, where $\mathbb{E}(\mathbf{x}|k) = \boldsymbol{\mu}_k$ and $\text{Var}(\mathbf{x}|k) = \boldsymbol{\Sigma}_k$. These properties generate the mixture density defined by:

$$f_{mix}(\mathbf{x}) = \sum_{k=1}^{K} \pi_k f_k(\mathbf{x})$$

In supervised learning, the class labels $\mathbf{y} = (y_1, ..., y_M)$ is known meaning a mixture model can be fitted by estimating parameters of the probability density function using the data. In predictive modelling, this is done by using a portion of the data named the training dataset to avoid overparameterisation. Aside from the estimation approaches discussed, another method is to use the Expectation-Maximisation (E-M) algorithm to estimate the parameters by maximising the expected complete data log-likelihood. For further detail on the E-M algorithm, please refer to Izenman (2008).

Bayes Classifier

The Bayes classifier is a popular technique in pattern recognition that aims to minimise the average probability of classification error. It is based on mixture models, however with changes to the names of the components to align with the theory of Bayesian statistics. Recall that one can use Bayes' theorem to calculate the probability of the data having a certain label as in equation 3.2, i.e. being in a certain class. For a classification task of K classes, We can divide the labelled data of M samples $\mathbf{x} = \{\mathbf{x}_1, y_1), ..., (\mathbf{x}_M, y_M)\}$ into two distinct datasets: $\{\mathbf{x}^{train}, y^{train}\}$ representing a sample of M_{train} labelled training data and $\{\mathbf{x}^{test}, y^{test}\}$ representing a sample of M_{test} test data. Typically in machine learning, a third dataset called the validation dataset is used to prevent overfitting, denoted $\{\mathbf{x}^{val}, y^{val}\}$ which contains M_{val} samples for $1 \leq M_{train} \leq M_{test} \leq M_{val} < M$.

The model learns patterns on the training data, by first calculating the prior probability π_k of a sample being in class k, $k \in \{0, ..., K-1\}$. Then, the likelihood function $f_k(\mathbf{x})$ is derived based on the density of the mixture model with estimated parameters based on the data.

Utilising the labelled training data enables learning of the discriminant function, denoted $h(\mathbf{x}^{train})$. The discriminant function uses Bayes' theorem to compute the posterior probabilities of each class given the observed features. It selects the

class with the highest posterior probability as the predicted class label. The function learns which properties of the data correspond to a particular class as it iterates through the training data. The posterior probability of a class $k \in \{0, ..., K-1\}$ using equation 3.2 can be represented by:

$$\mathbb{P}(k|\mathbf{x}^{train}) = \frac{\pi_k f_k(\mathbf{x})}{f(\mathbf{x})}$$

The likelihood density of the data belonging to class k is denoted by $f_k(\mathbf{x})$ and $f(\mathbf{x})$ is the marginal density of the data, which for data with discrete categorical labels can be represented by:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k f_k(\mathbf{x})$$

The discriminant function below is then used to calculate the logarithm of the posterior probability.

$$d_k(\mathbf{x}) = \log \pi_k + \log f_k(\mathbf{x}) - \log f(\mathbf{x})$$

Simplification is conducted to remove terms not involving k, implying the $\log f(\mathbf{x})$ term can be removed as it is a constant. Therefore the Bayes discriminant function is defined as:

$$d_k(\mathbf{x}) = \log \pi_k + \log f_k(\mathbf{x}) \tag{3.3}$$

The output of the discriminant function can be used for class prediction using the test dataset. A hard classification can be made by classifying the sample with the label that maximises the discriminant function, known as the predictor function. The predictor function takes the form:

$$\hat{y}^{test} = h(\mathbf{x}^{test}) = \arg\max_k d_k(\mathbf{x}^{test}) \tag{3.4}$$

The accuracy of the discriminant function is analysed by assessing the error associated with the prediction of a sample m is calculated using a loss function such as the log loss. This is then averaged over the whole sample to get an overall predictive accuracy of using a Bayes classifier on the dataset.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is an approximation to the Bayes classifier, however with the added assumption that the distribution of the d-dimensinal data of each class is drawn from a multivariate normal distribution with a class-specific mean and common covariance matrix (James et al., 2023). This is represented by:

$$f_k(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

$$f_k(\mathbf{x}) \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_d \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \sigma_{2,1} & \dots & \sigma_{d,1} \\ \sigma_{1,2} & \sigma_2^2 & \dots & \sigma_{d,2} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1,d} & \sigma_{2,d} & \dots & \sigma_d^2 \end{pmatrix}\right)$$

where each predictor $\mathbf{x}_i$ of $\mathbf{x}$ has its own univariate normal distribution:

$$f_k(\mathbf{x}_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

Recall that the probability density function of the multivariate Normal distribution on data $\mathbf{x}$ in class k with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}$ is defined as:

$$f_k(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

Assumptions of LDA specify that every class share a common covariance matrix $\boldsymbol{\Sigma}$ and each class has a specific mean $\boldsymbol{\mu}_k$, therefore the linear discriminant function can be derived from the Bayes discriminant function in equation 3.3 as:

$$\begin{aligned}
d_k(\mathbf{x}) &= \log \pi_k - \frac{d}{2}\log 2\pi - \frac{1}{2}\log(\det \boldsymbol{\Sigma}) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \\
&= \log \pi_k - \frac{1}{2}(\mathbf{x}^T - \boldsymbol{\mu}_k^T)\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \\
&= \log \pi_k - \frac{1}{2}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\
\text{Remove non-}k\text{ terms} \implies & \log \pi_k - \frac{1}{2}(-\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \boldsymbol{\mu}_k\boldsymbol{\Sigma}^{-1}\mathbf{x}^T + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\
\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k = \boldsymbol{\mu}_k\boldsymbol{\Sigma}^{-1}\mathbf{x}^T \implies & \log \pi_k - \frac{1}{2}(-2\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\
&= \log \pi_k + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k)
\end{aligned}$$

This is a linear function as all values represent scalars of order 1. The discriminant function finds a linear combination of all features that separate the classes whilst maximising between-class variance and minimising within-class variance. Essentially, LDA fits a decision boundary between two of the K classes $i, j \in \{0, ..., K-1\}$ such that $d_i(\mathbf{x}) = d_j(\mathbf{x}),\ i \neq j$.

Classification of data is made using the same predictor function as in the Bayes classifier, and class selection is done by the same process of maximisation of the predictor (see equation 3.4).

The intuition behind the method of LDA is to determine a subspace of lower dimension in which the data becomes separable (Xanthopoulos et al., 2013). For a dataset with M samples each with dimension d, let $\mathbf{x}_1, ..., \mathbf{x}_M \in \mathbb{R}^d$ be a set of M samples belonging to K classes. Let M_k denote the number of samples belong to the class k, $k \in \{0, ..., K-1\}$. Denote the sample means for each class as $\bar{\mathbf{x}}_k$, the total mean vector can be calculated as:

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{\mathbf{x}_i \in k} M_k \bar{\mathbf{x}}_k$$

The sample means are utilised in the calculation of the positive semi-definite within-class scatter matrices $\mathbf{S}_k$ representing the variability between different classes.

$$\mathbf{S}_k = \sum_{k=1}^{K} M_k(\bar{\mathbf{x}}_k - \bar{\mathbf{x}})(\bar{\mathbf{x}}_k - \bar{\mathbf{x}})^T$$

The between-class scatter matrix can be defined as the sum of the within-class scatter matrices i.e.:

$$\mathbf{S} = \sum_{k=1}^{K} \mathbf{S}_k$$

To minimise the variability between classes, the data must be transformed to a lower-dimensional subspace by a matrix **U** which maximises the between-class scatter and minimises the within-class. This can be formed by the following eigenvalue problem for class k:

$$\mathbf{S}_k\mathbf{U} = \lambda\mathbf{S}\mathbf{U}$$

where λ denotes the eigenvalue.

Derivation of the transformation matrix **U** can be completed using multiple methods. One way to approximate this is through singular value decomposition, which is a computationally inexpensive way to estimate the transformation matrix. This is the default method for linear discriminant analysis in sklearn, due to its fast speed on high-dimensional data. This is also employed in my analysis to estimate the parameters as using alternative methods such as eigenvalue decomposition and least squares estimation require computation of the covariance matrix, which for the high-dimensional abundant data from MedMNIST+ would potentially take a matter of days. For further information on eigenvalue decomposition and least squares estimation please consult Hastie et al. (2009).

Singular value decomposition occurs on the within-class scatter matrix to derive the transformation matrix. SVD fragments $\mathbf{S}_k$ into three individual matrices:

$$\mathbf{S}_k = \mathbf{V}\mathbf{Q}\mathbf{W}^T$$

where **V** contains the eigenvectors that define the transformation, **Q** contains the singular values and **W** is an orthogonal matrix. To reduce the data to c dimensions, the top c eigenvectors are chosen from **V** and defined as the transformation matrix **U**. This method avoids calculation of the covariance matrix.

3.2.2. Logistic Regression

Overview

Logistic regression is a classification technique based on principles from regression modelling. It is a special case of a Generalised Linear Model (GLM) with a Binomial conditional distribution and logit link function. Instead of modelling the relationship between the features and outcome of the data through a straight line or hyperplane, logistic regression fits an "S" shaped logistic function from probabilistic outputs that represent whether an instance belongs to one of two binary encoded classes 0 and 1. Similar to linear regression, logistic regression can work with continuous and discrete data and inferences can be made on the significance of parameters on model performance. The categorisation of the input is made based on probability.

General Linear Model

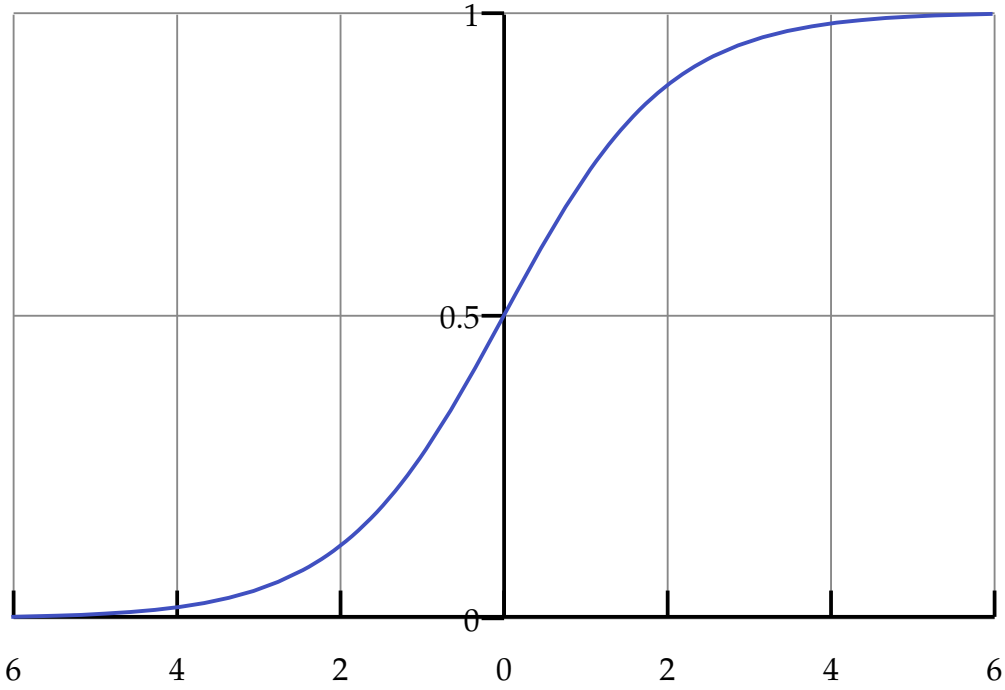
Recall that a general linear model using p explanatory variables/predictors $x_1, ..., x_p$ to predict the response Y can be defined as:

$$Y = \mathbb{E}(Y) + \epsilon = \beta_1 x_1 + ... + \beta_p x_p + \epsilon \tag{3.5}$$

It is required that $\beta_i,\ \forall i$ are linear, and that they can be estimated through least squares estimation or maximum likelihood. In the presence of qualitative responses with greater than 2 classes, a general linear model is not appropriate as encoding a qualitative response using quantitative values implies ordering of levels. This in practice would create different general linear models based on the ordering of the levels. Furthermore, for binary responses, there is the potential that estimates may exceed the [0,1] interval required for probability.

Logistic Regression

Figure 3.1.: The logistic function plotted within the range of −6 to 6, showcasing the characteristic S-shaped curve.



To mitigate the problems of general linear models, the method of logistic regression is employed for data with a categorical outcome variable. Logistic regression works by modelling the probability that the outcome variable Y belongs to a particular category, and allocates the sample to a category based on a threshold (normally 0.5). For a binary classification problem, with classes 0 and 1, I define the positive class to be 1. For p predictors, the probabilities are calculated by manipulating the output of a general linear model to a real number between 0 and 1 as seen in figure 3.1. The transformed values represent the probability that the data is contained within the positive class 1. This is called the logistic function which has the following formula:

$$\mathbb{P}(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X+\ldots+\beta_p X_p}} \tag{3.6}$$

Although there are now probabilities representing the likelihood of class allocation, which is suitable for classification, the relationship between the relationship between the predictor variables and the probability of class allocation must be

modelled. This is to align with the aim of regression modelling. Therefore, the probabilities must be transformed into a linear form with values being in the space of real numbers. To do this, the probability is transformed to the logarithm of the odds using the logit function.

The odds in binary classification represent the probability of the sample being in the positive class against the probability of the sample not being in the positive class (Kleinbaum et al., 2002). The odds can be defined as:

$$\text{Odds}(Y = 1) = \frac{\mathbb{P}(Y = 1|X)}{1 - \mathbb{P}(Y = 1|X)}$$

Taking the logarithm of this produces the log odds, which is calculated by the logit function. The log odds are symmetric around 0, meaning that interpreting the value from the logit function gives a clear indication of the change in event likelihood. The logit function is defined as:

$$\text{Logit} = \log \text{Odds}(Y = 1) = \log\left(\frac{\mathbb{P}(Y = 1|X)}{1 - \mathbb{P}(Y = 1|X)}\right) \tag{3.7}$$

Substitution of the logistic function into the logit function derives the standard form of a multiple logistic regression model. For simplification, let $Z = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$:

$$\begin{aligned}
\log\left(\frac{\mathbb{P}(Y = 1|X)}{1 - \mathbb{P}(Y = 1|X)}\right) &= \log\left(\frac{\frac{e^Z}{1+e^Z}}{1 - \frac{e^Z}{1+e^Z}}\right) \\
&= \log\left(\frac{e^Z}{1 + e^Z}\right) - \log\left(1 - \frac{e^Z}{1 + e^Z}\right) \\
&= \log(e^Z) - \log(1 + e^Z) - \log\left(\frac{1 + e^Z}{1 + e^Z} - \frac{e^Z}{1 + e^Z}\right) \\
&= \log(e^Z) - \log(1 + e^Z) - \log\left(\frac{1}{1 + e^Z}\right) \\
&= \log(e^Z) - \log(1 + e^Z) - \log 1 + \log 1 + e^Z \\
&= \log(e^Z) \\
&= Z \\
\text{Therefore} \log\left(\frac{\mathbb{P}(Y = 1|X)}{1 - \mathbb{P}(Y = 1|X)}\right) &= \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p
\end{aligned}$$

For multi-class logistic regression, there are two approaches. For K classes, one option is to formulate $K - 1$ individual binary classification tasks. This is known as the *One vs Rest/One vs All* method. On each task, $k \in \{0, \ldots, K - 1\}$ is treated as the positive class, and a probability of the sample being in class k is calculated using Equation 3.6. The class with the maximum probability is selected using a predictor function to allocate the sample to one of the classes. In multi-class

logistic regression, this is the same as the predictor function used in LDA as seen in equation 3.4.

$$\hat{y}^{test} = \arg\max_{k} \mathbb{P}(Y = k|X)$$

In my analysis, for non-binary classification tasks, I employed the One vs Rest approach to logistic regression due to its support of parallel computing enabling quicker run times. The second method, known as multinomial logistic regression, involves changes to the logistic function and the use of the softmax function for classification. For further details, please refer to James et al. (2023).

As in linear regression, estimation of model coefficients is required. For data with few predictors, directly employing maximum likelihood estimation yields coefficient estimates. For large datasets with many features, this would be an arduous process and approximation would be difficult. From an optimisation perspective, logistic regression aims to minimise the loss function. Hence, optimisation of the parameter/coefficient estimates is imperative for the reduction of the loss function $\mathcal{L}(\boldsymbol{\theta})$. Some optimisation algorithms utilised in sklearn will now be discussed that derive optimal estimates to minimise the loss function.

Optimisation Algorithms

In sklearn, there is an option of six different optimisation algorithms to minimise the loss function. In practice, the methods available in sci-kit learn often converge much faster than gradient descent. My analysis only analysed three of the methods: LBFGS, SAG and SAGA.

BFGS Algorithm The Broyden-Fletcher-Goldfarb-Shanno algorithm is an optimisation method used in unconstrained, non-linear optimisation problems. It belongs to the class of quasi-Newton methods, which are an alternative to the Newton-Raphson method in which the Hessian is not calculated or inverted due to the high computational cost or system incapacity. The BFGS algorithm is an iterative optimisation method that approximates Newton's method to solve non-linear unconstrained optimisation problems. The process involves approximating the Hessian of a function $f(\boldsymbol{\theta})$ i.e. $\mathbf{B}_k \approx \nabla^2 f(\boldsymbol{\theta})$.

In the case of logistic regression, the BFGS algorithm starts with a guess of the initial parameter values $\boldsymbol{\theta}_0$ and approximates the Hessian of te loss function based on the initial parameters $\mathbf{B}_0 \approx \nabla^2 \mathcal{L}(\boldsymbol{\theta}_0)$. The step size α_i is derived using a line-search algorithm until certain conditions such as a sufficient decrease in $\mathcal{L}(\boldsymbol{\theta})$ or the Wolfe conditions are met. Consult Bonnans et al. (2006) for further information on line-search algorithms and the Wolfe conditions.

To find optimal parameters that minimise the loss function below a threshold ψ, the BFGS algorithm can be formulated as:

Algorithm 3 BFGS Algorithm

Initialise the values of $\boldsymbol{\theta}_0, \mathbf{B}_0$
while $\|\nabla \mathcal{L}(\boldsymbol{\theta}_i)\| > \psi$ **do**
    $\mathbf{p}_i = \mathbf{B}_i^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}_i)$
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i \mathbf{p}_i$
    $\mathbf{s}_i = \alpha_i \mathbf{p}_i = \boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i$
    $\mathbf{y}_i = \nabla \mathcal{L}(\boldsymbol{\theta}_{i+1}) - \nabla \mathcal{L}(\boldsymbol{\theta}_i)$
    $\mathbf{B}_{i+1} = \mathbf{B}_i + \dfrac{\mathbf{y}_i \mathbf{y}_i^T}{\mathbf{y}_i^T \mathbf{s}_i} - \dfrac{\mathbf{B}_i \mathbf{s}_i \mathbf{s}_i^T \mathbf{B}_i}{\mathbf{s}_i^T \mathbf{B}_i \mathbf{s}_i}$
end while

LBFGS Algorithm Limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm is an estimate of the BFGS algorithm using limited computer memory. A disadvantage of the BFGS algorithm is that the algorithm must store $\frac{n(n+1)}{2}$ matrices for n variables. This is unattainable for many large datasets. Due to this problem, Nocedal (1980) created an adaptation of the algorithm that uses a limited amount of storage and updates the Hessian approximation continuously. By placing a constraint on the memory size r, the search direction $\mathbf{p}_i$ is calculated at iteration i using a specific algorithm that reduces memory usage. This algorithm, which I will name the L-BFGS search direction algorithm, performed on the loss function is defined as (Nocedal and Wright, 1999):

Algorithm 4 L-BFGS Search Direction Algorithm

$\mathbf{q}_i = \nabla \mathcal{L}(\boldsymbol{\theta}_i)$
for $j = i-1, \ldots, i-r$ **do**
    $\alpha_j = \dfrac{\mathbf{s}_j^T \mathbf{q}_j}{\mathbf{y}_j^T \mathbf{s}_j}$
    $\mathbf{q}_{j-1} = \mathbf{q}_j - \alpha_j \mathbf{y}_j$
end for
$\mathbf{p}_i = [\mathbf{B}_i^0]^{-1} \mathbf{q}_i$
for $j = i-r, i-r+1, \ldots, i-1$ **do**
    $\beta_j = \dfrac{\mathbf{y}_j^T \mathbf{p}_i}{\mathbf{y}_j^T \mathbf{s}_j}$
    $\mathbf{p}_{j+1} = \mathbf{p}_j + \mathbf{s}_j(\alpha_j - \beta_j)$
end for $\mathbf{p}_i = [\mathbf{B}_i]^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}_i)$

Utilising the above algorithm, the L-BFGS algorithm to minimise the loss function is:

Algorithm 5 L-BFGS Algorithm

Initialise the values of $\boldsymbol{\theta}_0$, $\mathbf{B}_0$ and r (memory size)
while $|\nabla \mathcal{L}(\boldsymbol{\theta}_i)| > \psi$ **do**
    Choose $\mathbf{B}_i^0$ using $\mathbf{B}_i^0 = \gamma_i \mathbf{I}$ where $\gamma_i = \dfrac{\mathbf{s}_{i-1}^T \mathbf{y}_{i-1}}{\mathbf{y}_{i-1}^T \mathbf{y}_{i-1}}$
    $\mathbf{p}_i = [\mathbf{B}_i]^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}_i)$ from Algorithm 4
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i \mathbf{p}_i$
    if $i > r$ **then**
        Discard the vector pair $\{\mathbf{s}_{i-r}, \mathbf{y}_{i-r}\}$ from storage.
    end if
    $\mathbf{s}_i = \alpha_i \mathbf{p}_i = \boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i$
    $\mathbf{y}_i = \nabla \mathcal{L}(\boldsymbol{\theta}_{i+1}) - \nabla \mathcal{L}(\boldsymbol{\theta}_i)$
end while

SAG Algorithm The Stochastic Average Gradient (SAG) algorithm is a variation of stochastic gradient descent. The SAG algorithm works particularly well on optimisation problems involving finite sums, such as logistic regression (Schmidt et al., 2017). The key difference between traditional stochastic gradient descent and the Stochastic Average Gradient (SAG) algorithm is the way gradients are computed and updated. In SAG, a running average of gradients is maintained, reducing the variance of gradient estimates and leads to faster convergence.

The SAG algorithm can be defined as:

Algorithm 6 Stochastic Average Gradient (SAG)

Initialize the values of $\boldsymbol{\theta}_0$, the average gradient $\mathbf{G} = \mathbf{0}$, where $\mathbf{G}$ has the same dimension as $\boldsymbol{\theta}$, select a learning rate η and convergence threshold ψ.
while $\|\mathbf{G}\| > \psi$ **do**
    Randomly select a sample m from the dataset of size M.
    Compute the gradient of sample m: $\mathbf{G}_m = \nabla \mathcal{L}_m(\boldsymbol{\theta})$
    Update the average gradient:

$$\mathbf{G} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{G}_m$$

    Update the parameter vector:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \mathbf{G}$$

end while

SAGA Algorithm The Stochastic Average Gradient Accelerated (SAGA) algorithm is an extension of the SAG algorithm incorporating adaptive step-size selection which accelerates convergence (Defazio et al., 2014). This involves updating the parameter vector for each parameter based on a criterion regarding the gradients of the previously selected sample. The SAGA algorithm can be defined as:

Algorithm 7 Stochastic Average Gradient Accelerated (SAGA)

1: Initialize the values of $\boldsymbol{\theta}_0$, the average gradient $\mathbf{G} = \mathbf{0}$, where $\mathbf{G}$ has the same dimension as $\boldsymbol{\theta}$, select a learning rate η and convergence threshold ψ.
2: **while** $\|\mathbf{G}\| > \psi$ **do**
3: Randomly select a sample m from the dataset of size M.
4: Compute the gradient of sample m: $\mathbf{G}_m = \nabla \mathcal{L}_m(\boldsymbol{\theta})$
5: Update the average gradient:

$$\mathbf{G} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{G}_m$$

6: Update the parameter vector for each parameter $\theta^{(j)}$ individually:

$$\boldsymbol{\theta}_{i+1}^{(j)} = \begin{cases} \boldsymbol{\theta}_i^{(j)} - \eta \mathbf{G}_m^{(j)} & \text{if } m \neq m* \\ \boldsymbol{\theta}_i^{(j)} - \eta \mathbf{G}^{(j)} & \text{if } m = m* \end{cases}$$

where $m*$ is the previously selected sample.
7: **end while**

Regularisation

Regularisation involves placing constraints on coefficient estimates to shrink estimates towards zero, reducing the variance (James et al., 2023). Regularisation introduces constraints on the parameters, adapting the loss function. The strength of regularisation is controlled by a parameter λ. In my analysis, I remained with the default regularisation strength where $\lambda = 1$. This may be changed in sklearn by altering the C parameter in logistic regression, where $C = \frac{1}{\lambda}$. There are two main methods of regularisation explored: Ridge regularisation and LASSO regularisation.

Ridge Regularisation Ridge regularisation, known as L2 regularisation in sklearn, involves the inclusion of the sum of squared parameter estimates on the loss function, $\lambda \sum_{j=1}^{p} \beta_j^2$. This has the effect of shrinking the estimates of β_j towards zero when the coefficient estimates are already close to zero. This ensures that all p predictors remain in the model.

LASSO Regularisation LASSO regularisation, known as L1 regularisation in sklearn, involves the inclusion of the sum of the absolute value of parameter estimates on the loss function, $\lambda \sum_{j=1}^{p} |\beta_j|$. This shrinks the estimates of β_j towards zero and also allows for $\beta_j = 0$ when λ is sufficiently large, meaning automatic variable selection is performed.

3.3. Machine Learning Algorithms

3.3.1. Convolutional Neural Network

Overview

One of the most popular techniques for image classification in deep learning is the Convolutional Neural Network (CNN). This is a multilayer network that is trained using backpropagation, usually on imaging data, and reduces the number of parameters in the network. Whilst similar models were previously used, the first example of this type of network utilised on images instead of feature vectors was by LeCun et al. (1989) on a task of recognising handwritten zip codes. In a CNN, the hidden layers perform further operations in addition to activation. In this section, I will discuss the foundations of the neural network, from the discovery of the artificial neuron to the modern-day CNN and describe the architecture of the CNN used in the analysis.

Foundations of the artifical neuron

Neural networks are fundamental to deep learning. A neural network is a model that is designed based on the behaviour of biological neurons. Like in neuroactivity, information is fed through multiple layers until the output is received, which in the case of a brain would be a reflex or activity (Peter et al., 2019).

The idea of a neural network for computation was proposed by Mcculloch and Pitts (1943) at the University of Chicago. By building on early ideas of artificial intelligence discussed by Turing (1937), they discovered that connecting simple elements in a network that mimicked a brain neuron could have high computational power. Mcculloch and Pitts (1943) constructed a simple mathematical model to describe the workings of a single artificial neuron that inputs and outputs binary values. By taking in N inputs, where $\mathbf{x}$ is the vector of binary input nodes, the model calculates a net input z by summing the input values. Furthermore, a threshold/bias $\theta \in \mathbb{R}$ parameter is introduced to shift the decision boundary away from the origin. The net input is calculated by the following formula:

$$z = \sum_{i=1}^{N} x_i - \theta \tag{3.8}$$

The decision boundary to determine the output of the artificial is decided by an activation function $\phi(z)$. An activation function simply determines if a neuron should be "activated". The McCulloch-Pitts model maps the net input to the predicted outputs $\hat{y}$ which take binary $0/1$ values based on the threshold. The activation function can be defined with the following formula:

$$\hat{y} = \phi(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{3.9}$$

This is equivalent to the Heaviside step function.

The Perceptron

Models based on a collection of biological neurons were introduced by Rosenblatt (1958), who defined the idea of the perceptron. For the first time, a method was developed to enable a computer to learn from data by utilising artificial neurons, paving the way for future groundbreaking discoveries in supervised learning. The invention of the perceptron allowed for classification tasks to be completed on linearly separable data - data that can be separated by the hyperplane $z = 0$.

The perceptron model uses the foundations of the Mcculloch and Pitts (1943) neuron but expands the idea to account for non-binary input values and introduces parameter weights. This allows the model to analyse the importance of each input. By introducing a new vector namely $\mathbf{w}$ to represent the weight of each input into equation 3.8, the formula for net input can be rewritten as:

$$z = \mathbf{w} \cdot \mathbf{x} - \theta \tag{3.10}$$

Analogous to the McCulloch-Pitts neuron, the net input is taken through an activation function to determine the output of the neuron. Rosenblatt also utilised the Heaviside step-function as seen in Equation 3.9 as the activation function for his perceptron.

Probably the largest breakthrough with the Rosenblatt perceptron is the notion of the perceptron learning algorithm. This rule adjusts the weights of the inputs to optimise classification by learning errors. This is the first time a model-training technique has been introduced to enhance model performance and generalisability. Whilst the initial weights of the model can be randomly chosen, subsequent model iterations adjust weights using an algorithm to minimise the discrepancy between the predicted output and the actual output.

The rule is a variation of equations 3.9 and 3.10 but with the added notation of $^{(m)}$ to denote the m-th sample, where $1 \leq m \leq M$. Let $\mathcal{L}^{(m)}$ denote the loss function of the m-th sample, $y^{(m)}$ the true label or output on the m-th sample and η the learning rate, typically $\eta = 0.5$. Let $_j$ denote the iteration of the algorithm. The perceptron learning rule uses the following algorithm on a single training example to choose weights to minimise output difference and improve accuracy (Du and Swamy, 2019):

Algorithm 8 Perceptron Learning Rule

1: **for** $1 \leq m \leq M$ **do**
2: $\quad z_j^{(m)} = \mathbf{x}^{(m)} \cdot \mathbf{w}_j^{(m)} - \theta_j^{(m)}$
3: $\quad \hat{y}_j^{(m)} = \phi(z_j^{(m)}) = \begin{cases} 1, & z_j^{(m)} \geq 0 \\ 0, & z_j^{(m)} < 0 \end{cases}$
4: $\quad \mathcal{L}_j^{(m)} = y^{(m)} - \hat{y}_j^{(m)}$
5: $\quad \mathbf{w}_j^{(m+1)} = \mathbf{w}_j^{(m)} + \eta \mathbf{x} \mathcal{L}_j^{(m)}$
6: $\quad \theta_j^{(m+1)} = \theta_j^{(m)} + \eta \mathcal{L}^{(m)}$
7: **end for**

The method works by changing the weight vector to point towards input vectors labelled with 1 (Shi et al., 2023). Rosenblatt demonstrated that the algorithm converged four years later in his Principal Convergence Theorem (Rosenblatt et al., 1962). Consequently, it could be deduced that the perceptron learning rule trained the model for optimal performance, i.e. when the loss function was minimised below a threshold ψ or to $\mathcal{L} = 0$.

Even though the perceptron was revolutionary at the time of discovery, the model would prove to be limited as it is only applicable to binary classification problems.

The Problematic Perceptron

Research into neural networks was halted in the late 1960s, as Minsky and Papert (1969) unveiled issues with Rosenblatt's perceptron. The two MIT mathematicians highlighted that the perception was unable to comprehend problems involving exclusive OR operators. This is because a linear hyperplane cannot be used to solve these types of problems. Whilst they offered further information on how multilayer structures may address these issues, no solution nor a promise of one was proposed. This induced the first AI winter to begin.

Backpropagation and the Multilayer Perceptron

Algorithm training for better performance had received little attention up to this point in history. However, this was changed by the thesis of a Finnish Master's student Linnainmaa (1970) who first described a method called the reverse mode of automatic differentiation. Linnainmaa showed that the costs associated with forward activation equal the costs of backward derivative calculation when trying to minimise errors in complex optimisation problems (Schmidhuber, 2015). He discovered that exploiting the chain rule could lead to the automatic computation of derivatives. This idea was further developed when a nonlinear version of optimising gradient descent by back-calculating derivatives through a neural system, equivalent to the generalised delta rule, was created by Werbos (1974).

Rumelhart et al. (1986b) popularised this strategy in neural networks by formalising the current method through an experimental analysis. The technique of backwards calculation of derivatives became known as backpropagation - a process used in supervised learning of sending derivatives of loss function backwards to previous layers to guide the training of the weights and biases. The method uses calculus to calculate the gradient of the loss function which is in gradient descent to find optimal model parameters. The paper expanded on a single-layer network by introducing hidden layers and collections of neuron nodes, alluding for a multilayer network. This removed the problem of linear separability in XOR operators highlighted by Minsky and Papert (1969), as the introduction of hidden layers enabled differentiable and non-linear activation functions, allowing neurons to take on non-binary values. Whilst previous models utilising multiple layers had been created (Ivakhnenko, 1968; Fukushima, 1980), this was the foundation of the multilayer perceptron (MLP) model used in the modern day.

In multilayer perceptrons, inputs are fed forward through the network, traversing

the hidden layers, activation functions and output layers to predict the output. As there are now multiple layers each with multiple neurons, every neuron in the previous layer is connected to every neuron in the next layer by all possible connections. Similar to the Rosenblatt perceptron, the output is then compared to the true output and an error is calculated. Extending the notation from equations 3.8 - 3.10 and algorithm 8 enables multiple layers to be accounted for. Except now, the weights and biases are updated using the gradients calculated through backpropagation of the loss function, in which updates to the weights and biases are derived from gradient descent.

To exploit the advantages of multiple layers, non-linear activation functions will be introduced. For backpropagation, the activation functions must be continuously differentiable. This implies that backpropagation makes no progress on the binary step activation function as it is not differentiable at 0, and its derivative is 0 at all other values (Snyman, 2005). A commonly used non-linear activation function for binary classification problems is the sigmoid function, which transforms continuous real numbers into the range of $(0, 1)$. Denoting the binary value of 1 as the positive class, the sigmoid function produces a probability of the sample belonging to the positive class. This can be defined as:

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Using ideas from the perceptron learning rule allows for the construction of the process for the forward pass through multiple layers. This represents how information traverses through a network to generate a prediction of the output. For a binary classification problem on M samples, let there be L layers in the network, and N_l denote the number of neurons contained in layer l, $l \in \{1, ..., L\}$. To denote the layer l, I will utilise the notation $_{[l]}$.

To represent a forward pass for a sample $m \in M$, I will define a new notation. Due to the increased complexity, instead of explaining the process in vector notation, I will define things element-wise. Let:

- $a^{(m)}_{i;[1]}$ be the input of the i-th feature input to neuron i in layer 1
- $w^{(m)}_{(i,j);[l-1,l]}$ denote the weights connecting neuron i in layer $l-1$ and neuron j in layer l
- $\theta^{(m)}_{j;[l]}$ be the bias of neuron j in layer l:
- $a^{(m)}_{j;[l]}$ represent the output of the activation function $\phi(z^{(m)}_{j;[l]})$ on neuron j in layer l
- $z^{(m)}_{j;[l]}$ denote the linear combination of weights, the previous later's activation output and the bias.

$$z^{(m)}_{j;[l]} = \sum_{i=1}^{N_{l-1}} w^{(m)}_{(i,j);[l-1,l]} a^{(m)}_{i;[l-1]} - \theta^{(m)}_{j;[l]}$$

- $\hat{y}^{(m)}_{1;[L]}$ is the output of the activation function in the final layer, representing the probability that the input sample belongs to the positive class 1.

In the case of a single layer, the error associated with misclassification was calculated by taking the difference between the predicted class of the input and the input's true label. In a binary classification problem, $y^{(m)}$ can take on the value of 0 or 1. A commonly used error/loss function in classification tasks is the log loss. Further detail of this loss function is provided in Section 4.3.8. The log loss for the output of sample m can be defined as:

$$\mathcal{L}^{(m)} = -\left[y^{(m)} \log\left(\hat{y}^{(m)}_{1;[L]}\right) + \left(1 - y^{(m)}\right) \log\left(1 - \hat{y}^{(m)}_{1;[L]}\right)\right]$$

Once all samples have been fed forward through the network, the final classification error is obtained by taking the average of the loss function over all M samples i.e.

$$E = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}^{(m)}$$

For multi-class classification, a few modifications can be made to the forward pass. In the output layer, the softmax function is employed as the activation function instead of the sigmoid function. For a classification problem involving K classes, the softmax function in neuron k of layer L, where $k \in \{0, \ldots, K-1\}$ is defined as:

$$\phi(z^{(m)}_{k;[L]}) = \frac{e^{z^{(m)}_{k;[L]}}}{\sum_{k=1}^{K} e^{z^{(m)}_{k;[L]}}} \tag{3.11}$$

For neuron k in layer L, the softmax function converts the value of the pre-activation output $z^{(m)}_{k;[L]}$ to a probability, which represents the probability that the input belongs to the class k corresponding with neuron k. This is denoted by $\hat{y}^{(m)}_{k;[L]}$. As the activation outputs are probabilities, by definition:

$$\sum_{k=0}^{K-1} \hat{y}^{(m)}_{k;[L]} = 1$$

Furthermore, the form of the loss function changes when there are multiple classes. The loss function for an output layer of K neurons representing K classes is:

$$\mathcal{L}^{(m)} = -\sum_{k=0}^{K-1} y^{(m)} \log\left(\hat{y}^{(m)}_{k;[L]}\right)$$

Hence the process for a forward pass in a multilayer network is represented by the below algorithm:

Algorithm 9 Forward Pass

for sample $1 \leq m \leq M$ in a classification problem of K classes **do**

Begin with n features with the i-th feature being $a^{(m)}_{i;[1]}$.

Input the i-th feature into neuron i.

Calculate the linear combination of weights and bias between the input layer (layer 1) at neuron i and neuron j in layer 2:

$$z^{(m)}_{j;[2]} = \sum_{j=1}^{N_2} w^{(m)}_{(i,j);[1,2]} a^{(m)}_{i;[1]} - \theta^{(m)}_{j;[2]}$$

Calculate the output of the activation function for neuron j in layer 2:

$$a^{(m)}_{j;[2]} = \phi(z^{(m)}_{j;[2]}) = \frac{1}{1 + e^{-z^{(m)}_{j;[2]}}}$$

for neuron c in layer $l-1$ and neuron d in layer l, $2 \leq l \leq L-1$ **do**

$$z^{(m)}_{d;[l]} = \sum_{c=1}^{N_{l-1}} w^{(m)}_{(c,d);[l-1,l]} a^{(m)}_{c;[l-1]} - \theta^{(m)}_{d;[l]}$$

$$a^{(m)}_{d;[l]} = \phi(z^{(m)}_{d;[l]}) = \frac{1}{1 + e^{-z^{(m)}_{d;[l]}}}$$

end for

Calculate the linear combination of weights and bias between neuron p of layer $L-1$ and the pre-activation output in layer L:

$$z^{(m)}_{1;[L]} = \sum_{p=1}^{N_{L-1}} w^{(m)}_{(p,1);[L-1,L]} a^{(m)}_{p;[L-1]} - \theta^{(m)}_{1;[L]}$$

if K = 2 **then**

Calculate the output post activation in layer L, i.e. the probability that the input belongs to the positive class using the sigmoid function:

$$\hat{y}^{(m)}_{1;[L]} = \phi(z^{(m)}_{1;[L]}) = \frac{1}{1 + e^{-z^{(m)}_{1;[L]}}}$$

Calculate the classification error per sample by comparing the actual output to the predicted output using the binary log loss function:

$$\mathcal{L}^{(m)} = -\left[y^{(m)} \log\left(\hat{y}^{(m)}_{1;[L]}\right) + \left(1 - y^{(m)}\right) \log\left(1 - \hat{y}^{(m)}_{1;[L]}\right) \right]$$

end if

15: **if** K > 2 **then**

16: Calculate the output post activation of neuron k in layer L, i.e. the probability that the input belongs to the corresponding class $k \in \{0, ..., K-1\}$ using the softmax function:

$$\hat{y}^{(m)}_{k;[L]} = \phi(z^{(m)}_{k;[L]}) = \frac{e^{z^{(m)}_{k;[L]}}}{\sum_{k=0}^{K-1} e^{z^{(m)}_{k;[L]}}}$$

17: Calculate the classification error per sample by comparing the actual outputs to the predicted probabilities using the categorical log loss function:

$$\mathcal{L}^{(m)} = -\sum_{k=0}^{K-1} y^{(m)} \log\left(\hat{y}^{(m)}_{k;[L]}\right)$$

18: **end if**

19: **end for**

20: Calculate the total classification error across all samples by taking the average of each sample's error:

$$E = \frac{1}{M}\sum_{m=1}^{M} \mathcal{L}^{(m)}$$

Deriving the method of backpropagation

The Principal Convergence Theorem demonstrated that the perception learning rule's weight and bias changes may be made repeatedly over M samples until the error value converges to either 0 or a threshold ψ. In the case of a multilayer network, how may these be adjusted to lower the error? This can be accomplished through gradient descent on each weight and bias. But how is the gradient found? This is done through backpropagation. As the computation of the gradient that minimises the loss function is extremely difficult for high-dimensional vectors, the method of backpropagation automates this.

For sample m in a binary classification task, the loss function $\mathcal{L}^{(m)}$ is in terms of $\hat{y}^{(m)}_{1;[L]}$ and $y^{(m)}$. Furthermore, $\hat{y}^{(m)}_{1;[L]}$ can be represented in terms of $z^{(m)}_{1;[L]}$ through the sigmoid function. Also, $z^{(m)}_{1;[L]}$ is defined in terms of $w^{(m)}_{(p,1);[l-1,l]}$, $\hat{y}^{(m)}_{1;[L]}$ and $\theta^{(m)}_{1;[L]}$. Upon closer inspection, it becomes evident that each parameter of the model is related when moving backwards through the network. This can be represented by the diagram below.

$$\mathcal{L}^{(m)} \overset{\text{in terms of}}{\Longrightarrow} \hat{y}^{(m)}_{1;[L]} \overset{\text{in terms of}}{\Longrightarrow} z^{(m)}_{1;[L]} \overset{\text{in terms of}}{\Longrightarrow} \begin{cases} w^{(m)}_{(p,1);[L-1,L]} \\ a^{(m)}_{p;[L-1]} \\ \theta^{(m)}_{1;[L]} \end{cases} \overset{\text{in terms of}}{\Longrightarrow} z^{(m)}_{p;[L-1]} \cdots$$

Although the weights and bias can be changed at each layer with respect to the loss function through partial differentiation, it proves difficult to find all partial

derivatives. Using the chain of association shown above invites the use of the chain rule, which hugely simplifies the process!

For a binary classification task, I will derive the partial derivative of the loss function for sample $m \in M$ with respect to the activation output in the final layer L:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}^{(m)}_{1;[L]}} = -\frac{y^{(m)} - \hat{y}^{(m)}_{1;[L]}}{\hat{y}^{(m)}_{1;[L]}\left(1 - \hat{y}^{(m)}_{1;[L]}\right)}$$

This derivative can then be used to calculate the derivative of the loss function w.r.t the pre-activation output of layer L denoted as $z^{(m)}_{1;[L]}$:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial z^{(m)}_{1;[L]}} = \frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}^{(m)}_{1;[L]}} \frac{\partial \hat{y}^{(m)}_{1;[L]}}{\partial z^{(m)}_{1;[L]}} = \frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}^{(m)}_{1;[L]}} \left(\frac{\partial}{\partial z^{(m)}_{1;[L]}} \phi\left(z^{(m)}_{1;[L]}\right) \right)$$

In binary classification, $\phi(z^{(m)}_{1;[L]})$ is the sigmoid function. By the quotient rule, the derivative of the sigmoid function is:

$$\frac{\partial \hat{y}^{(m)}_{1;[L]}}{\partial z^{(m)}_{1;[L]}} = \phi'\left(z^{(m)}_{1;[L]}\right) = \phi\left(z^{(m)}_{1;[L]}\right)\left(1 - \phi\left(z^{(m)}_{1;[L]}\right)\right) = a^{(m)}_{1;[L]}\left(1 - a^{(m)}_{1;[L]}\right) \tag{3.12}$$

Hence an important quantity is derived:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial z^{(m)}_{1;[L]}} = \frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}^{(m)}_{1;[L]}} \hat{y}^{(m)}_{1;[L]}\left(1 - \hat{y}^{(m)}_{1;[L]}\right) = -\frac{y^{(m)} - \hat{y}^{(m)}_{1;[L]}}{\hat{y}^{(m)}_{1;[L]}\left(1 - \hat{y}^{(m)}_{1;[L]}\right)} \hat{y}^{(m)}_{1;[L]}\left(1 - \hat{y}^{(m)}_{1;[L]}\right) = -\left(y^{(m)} - \hat{y}^{(m)}_{1;[L]}\right)$$

This term represents the sensitivity of the loss function with respect to the changes in the pre-activation output. This is often denoted by $\delta^{(m)}_{1;[L]}$ which is defined as:

$$\delta^{(m)}_{1;[L]} = \frac{\partial \mathcal{L}^{(m)}}{\partial z^{(m)}_{1;[L]}} \tag{3.13}$$

The next step of backpropagation is to calculate the partial derivatives of the loss function with respect to the weights between neuron p in layer $L-1$ and the output in layer L and also the bias of layer L. This is done by utilising the chain rule and previous derivatives. Finding the partial derivative of z w.r.t the weight and w.r.t the bias below:

$$\frac{\partial z^{(m)}_{1;[L]}}{\partial w^{(m)}_{(p,1);[L-1,L]}} = \frac{\partial}{\partial w^{(m)}_{(p,1);[L-1,L]}} \left(\sum_{p=1}^{N_{L-1}} w^{(m)}_{(p,1);[L-1,L]} a^{(m)}_{p;[L-1]} - \theta^{(m)}_{1;[L]} \right) = a^{(m)}_{p;[L-1]}$$

$$\frac{\partial z^{(m)}_{1;[L]}}{\partial \theta^{(m)}_{1;[L]}} = \frac{\partial}{\partial \theta^{(m)}_{1;[L]}} \left(\sum_{p=1}^{N_{L-1}} w^{(m)}_{(p,1);[L-1,L]} a^{(m)}_{p;[L-1]} - \theta^{(m)}_{1;[L]} \right) = 1$$

But how does the pre-activation output of layer L depend on the activation output of layer $L-1$? As shown earlier, this can be derived by partial differentiation of z:

$$\frac{\partial z^{(m)}_{1;[L]}}{\partial a^{(m)}_{p;[L-1]}} = \frac{\partial}{\partial a^{(m)}_{1;[L]}} \left(\sum_{p=1}^{N_{L-1}} w^{(m)}_{(p,1);[L-1,L]} a^{(m)}_{p;[L-1]} - \theta^{(m)}_{1;[L]} \right) = w^{(m)}_{(p,1);[L-1,L]} \tag{3.14}$$

Hence the partial derivatives of the loss function with respect to the weight between neuron p in layer $L-1$ and the output in layer L and with respect to the bias at the output in layer L can be written as:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w^{(m)}_{(p,1);[L-1,L]}} = \delta^{(m)}_{1;[L]} a^{(m)}_{p;[L-1]}$$

$$\frac{\partial \mathcal{L}^{(m)}}{\partial \theta^{(m)}_{1;[L]}} = \delta^{(m)}_{1;[L]}$$

The form of this derivative remains the same for all neurons in all layers. Hence, a formula for $\delta^{(m)}_{c;[l-1]}$ of the neuron c in layer $l-1$ can be derived. From the forward pass, the value of $a^{(m)}_{c;[l-1]}$ is know. I will show this by using the definition of $\delta^{(m)}_{c;[l]}$ using equation 3.12 - 3.14 below:

$$\begin{aligned}
\delta^{(m)}_{c;[l-1]} &= \frac{\partial \mathcal{L}^{(m)}}{\partial a^{(m)}_{c;[l-1]}} \frac{\partial a^{(m)}_{c;[l-1]}}{\partial z^{(m)}_{c;[l-1]}} \\
&= \frac{\partial \mathcal{L}^{(m)}}{\partial z^{(m)}_{d;[l]}} \frac{\partial z^{(m)}_{d;[l]}}{\partial a^{(m)}_{c;[l-1]}} \frac{\partial a^{(m)}_{c;[l-1]}}{\partial z^{(m)}_{c;[l-1]}} \\
\text{Using 3.12 - 3.14:} \quad &= \delta^{(m)}_{d;[l]} w^{(m)}_{(c,d);[l-1,l]} a^{(m)}_{c;[l-1]} \left(1 - a^{(m)}_{c;[l-1]}\right)
\end{aligned}$$

Therefore the value of the delta for a neuron in the previous layer can be calculated using previously derived delta values and values found in the forward pass! The process showcases the power of automatic differentiation.

Considering a multi-class classification task, the output layer requires calculation of the partial derivative of the loss function with respect to the softmax output of neuron k in layer L, and the derivative of the softmax output of neuron k in layer L with respect to its pre-activation output. Partial differentiation of the categorical log loss w.r.t the output of the softmax function in neuron k of layer L produces:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}^{(m)}_{q;[L]}} = \sum_{k=0}^{K-1} \frac{y^{(m)}}{\hat{y}^{(m)}_{k;[L]}}$$

I will write the derivative of the softmax function in the following form:

$$\frac{\partial \hat{y}^{(m)}_{k;[L]}}{\partial z^{(m)}_{k;[L]}} = \phi'\left(z^{(m)}_{k;[L]}\right)$$

Using the quotient rule on the derivative of the softmax function:

$$\frac{\partial \hat{y}_{k;[L]}^{(m)}}{\partial z_{k;[L]}^{(m)}} = \frac{e^{z_{k;[L]}^{(m)}} \sum_{k=0}^{K-1} e^{z_{k;[L]}^{(m)}} - e^{z_{k;[L]}^{(m)}} e^{z_{k;[L]}^{(m)}}}{\left(\sum_{k=0}^{K-1} e^{z_{k;[L]}^{(m)}}\right)^2} = \frac{e^{z_{k;[L]}^{(m)}}}{\sum_{k=0}^{K-1} e^{z_{k;[L]}^{(m)}}} \left(\frac{\sum_{k=0}^{K-1} e^{z_{k;[L]}^{(m)}} - e^{z_{k;[L]}^{(m)}}}{\sum_{k=0}^{K-1} e^{z_{k;[L]}^{(m)}}}\right) = \hat{y}_{k;[L]}^{(m)} \left(1 - \hat{y}_{k;[L]}^{(m)}\right)$$

Which is the same as the derivative of the sigmoid activation function with respect to the pre-activation output! This value can therefore be used to find $\delta_{k;[L]}^{(m)}$:

$$\delta_{k;[L]}^{(m)} = \frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}_{k;[L]}^{(m)}} \frac{\partial \hat{y}_{k;[L]}^{(m)}}{\partial z_{k;[L]}^{(m)}} = -\sum_{k=0}^{K-1} \frac{y^{(m)}}{\hat{y}_{k;[L]}^{(m)}} \hat{y}_{k;[L]}^{(m)} \left(1 - \hat{y}_{k;[L]}^{(m)}\right) = -\sum_{k=0}^{K-1} y^{(m)} \left(1 - \hat{y}_{k;[L]}^{(m)}\right)$$

The same update rule for $\delta_{k;[L]}^{(m)}$ can be used as in the binary case due to the sigmoid and softmax activation functions having the same derivative. As the pre-activation outputs have not changed form, one can use the previously derived results in backpropagation for multi-class classification with minor notational updates.

Therefore, the backpropagation algorithm for a classification task on data with M samples is defined as:

Algorithm 10 Backpropagation

1: **for** sample $1 \leq m \leq M$ in a classification problem of K classes **do**
2: Begin with the output of the forward pass on sample m, $\mathcal{L}^{(m)}$
3: **if** K = 2 **then**
4: Calculate $\delta_{1;[L]}^{(m)}$ from the partial derivative of the binary loss function with respect to the pre-activation output:

$$\delta_{1;[L]}^{(m)} = \frac{\partial \mathcal{L}^{(m)}}{\partial z_{1;[L]}^{(m)}} = \frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}_{1;[L]}^{(m)}} \frac{\partial \hat{y}_{1;[L]}^{(m)}}{\partial z_{1;[L]}^{(m)}} = -\left(y^{(m)} - \hat{y}_{1;[L]}^{(m)}\right)$$

5: Calculate the partial derivative of the loss function with respect to the weight between neuron p in layer $L-1$ and the output in layer L and the bias of the output:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w_{(p,1);[L-1,L]}^{(m)}} = \delta_{1;[L]}^{(m)} a_{p;[L-1]}^{(m)}$$

$$\frac{\partial \mathcal{L}^{(m)}}{\partial \theta_{1;[L]}^{(m)}} = \delta_{1;[L]}^{(m)}$$

6: **end if**

if K > 2 **then**

Calculate $\delta^{(m)}_{k;[L]}$ from the partial derivative of the categorical loss function with respect to the pre-activation output of neuron k in layer L:

$$\delta^{(m)}_{k;[L]} = \frac{\partial \mathcal{L}^{(m)}}{\partial z^{(m)}_{k;[L]}} = \frac{\partial \mathcal{L}^{(m)}}{\partial \hat{y}^{(m)}_{k;[L]}} \frac{\partial \hat{y}^{(m)}_{k;[L]}}{\partial z^{(m)}_{k;[L]}} = -\sum_{k=0}^{K-1} y^{(m)} \left(1 - \hat{y}^{(m)}_{k;[L]}\right)$$

Calculate the partial derivative of the loss function with respect to the weight between neuron p in layer $L-1$ and neuron k in layer L and the bias of neuron k:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w^{(m)}_{(p,k);[L-1,L]}} = \delta^{(m)}_{k;[L]} a^{(m)}_{p;[L-1]}$$

$$\frac{\partial \mathcal{L}^{(m)}}{\partial \theta^{(m)}_{k;[L]}} = \delta^{(m)}_{k;[L]}$$

end if

$l = L - 1$

for neuron d in layer l, $1 \leq l \leq L-1$ **do**

Calculate $\delta^{(m)}_{d;[l]}$ based on $\delta^{(m)}_{e;[l+1]}$ from neuron e in layer $l+1$:

$$\delta^{(m)}_{d;[l]} = \delta^{(m)}_{e;[l+1]} w^{(m)}_{(d,e);[l,l+1]} a^{(m)}_{d;[l]} \left(1 - a^{(m)}_{d;[l]}\right)$$

Calculate the partial derivative of the loss function with respect to the weights between neuron c in layer $l-1$ and neuron d in layer l and the bias of neuron d in layer l:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w^{(m)}_{(c,d);[l-1,l]}} = \delta^{(m)}_{d;[l]} a^{(m)}_{c;[l-1]}$$

$$\frac{\partial \mathcal{L}^{(m)}}{\partial \theta^{(m)}_{d;[l]}} = \delta^{(m)}_{d;[l]}$$

$l = l - 1$

end for

end for

After calculating the derivatives of each weight and bias in the network, the values are free to be updated using gradient descent. This is done for each sample in the data and generates a total error of the sample. This process is repeated until convergence of the loss function, i.e. when the total error E reaches 0 or a defined threshold ψ.

The work of Rumelhart et al. (1986a) fuelled encouragement and enthusiasm for the research of artificial neural networks. Another paper from Rumelhart et al. (1986b) created momentum regarding the use of neural networks and their integration into image recognition tasks.

Convolutional Neural Network

Then came the convolutional neural network, a neural network whose name is derived from the matrix operation of convolution (Albawi et al., 2017). As in the multilayer perception, the CNN introduces hidden layers, however, they can perform more operations than simply just activation. The prediction of a convolutional neural network is similar to that of a perceptron, but this is contained in a single layer at the end named the fully connected layer.

The hidden layers in a CNN augment the data through feature extraction through the creation of feature maps and down-sampling. Although they are referred to as convolutional layers, each convolutional layer contains multiple operations. The different components of a convolutional layer include:

Convolution This involves sliding a small filter, known as a kernel, across the entire image to compute the dot product of the pixel values in the region of the kernel and the weights of each input. This is done a total of ζ_l times, where l is the layer index. As the network progresses through the layers, the number of such operations may increase to further reduce the spatial dimensions of the data. The values inside the kernel are randomly initialised but adapted as part of the model training process. The values of the kernel are determined at each batch through backpropagation, adjusting to minimise the loss function.

In the case of non-grayscale data, multiple input channels are responsible for the depth dimension of the input data. For example in coloured image data, the number of input channels is often 3 to denote the red, green and blue channels of an RGB image. In this case, the filter would slide over each of the three colour channels independently, and generate separate feature maps for each input channel.

In addition, adding extra border pixels before applying convolution can be applied. This is done to control the spatial dimensions of the output of the feature map so that convolution does not affect the dimension of the image data. This is useful to control the spatial resolution of feature maps throughout the network.

Batch Normalisation This process normalises the output from convolution before it is fed through an activation function. Often in neural networks, data is fed through the network in randomly allocated batches to update model parameters more frequently, resulting in a lower computation time and memory efficiency. As batch learning introduces stochasticity into the model and improves generalizability, the process involves subtracting the mean pixel value from the batch and dividing by the standard deviation to result in pixel values that follow an approximately Normal distribution. This stabilises the distribution of inputs to each layer, leading to more stable training.

Activation As seen in perceptron models, features are fed through an activation function to introduce non-linearity into the network. As image data contains many features, often the Rectified Linear Unit (ReLU) activation function is used to set negative pixel values to zero, adding sparsity to the data and reducing complexity

(Nair and Hinton, 2010). The ReLU activation function is completed element-wise over the pre-activation output z in layer l, which has p_l features located in sample m. ReLU can be defined as:

$$\phi(z^{(m)}_{p_l;[l]}) = \max(0, z^{(m)}_{p_l;[l]})$$

By providing a constant gradient of 1 for positive inputs, the problem of vanishing gradients is mitigated for the forward pass, enabling simpler backpropagation.

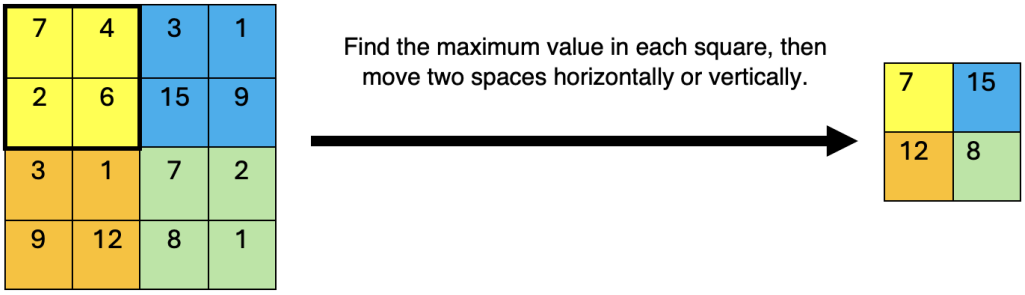


Figure 3.2.: An arbitrary example of Max Pooling using a 2×2 grid with stride 2.

Pooling Once features have been fed through an activation function, there is an option to complete pooling to reduce the spatial dimension of the feature maps. Whilst there are different types of pooling operations, I used Max Pooling in my model architecture. This involves sliding a 2×2 filter over the feature map and taking the maximum value in each feature map. This retains the most significant features and discards the least significant ones. The pooling layer then converts the output of the activation function into smaller feature maps, reducing spatial dimensions. The step size of the filter can be controlled too, which is called a stride. For example, a stride of 2 means that the filter moves 2 pixels at a time to further reduce the spatial dimensions. A visualisation of this can be seen in figure 3.2

A series of the above operations are repeated until the features are arranged in feature maps that are of a small dimension, which can be easily flattened to a one-dimensional vector. Then, the features proceed forward as in a multilayer perceptron to perform a prediction of the correct classification. These layers are called fully connected layers and involve a linear transformation of the data with respect to the weights and biases until the number of classes is reached. Then, a prediction is made based on the output and compared to the actual output in the loss function.

Several parameters can control the learning of a neural network. As previously mentioned, the process of information through the network is done in batches to optimise training. The term to describe when all the batches of data have traversed the network once is an epoch. It is standard practice to assess the loss function after each epoch. This evaluation is completed on both the training and validation datasets to ensure that the model does not fit exclusively to the training data. Furthermore, a method called early stopping can be applied to a neural network to ensure that the loss function converges close to the minimum. This ensures that

the model will stop trying to optimise the parameters when the loss function has not improved in I epochs, with a value I to be specified.

In sklearn, to ensure that the final model is fitted for testing on the test data, a checkpoint can be employed which saves the model with the lowest log loss on the validation dataset. This ensures that future uses of the model are utilising the best combination of parameters from training and validation.

Optimisation Algorithm

The optimisation algorithm used in the neural network to train the model parameters is the Adaptive Moment Estimation (Adam) algorithm. Created by Kingma and Ba (2015), the Adam algorithm is an extension of stochastic gradient descent with the addition of adaptive learning rates, momentum, Root Mean Square Propagation (RMSProp) and bias correction. The gradients used in the Adam optimisation algorithm are derived from backpropagation. The algorithm has proved successful across a wide range of architectures. Adam works over mini-batches that are run together as part of an epoch.

The Adam algorithm introduces moments to the algorithm for faster convergence. Adam optimisation combines the benefits of gradient-based updates with momentum and variance estimation to result in an efficient, adaptive, and stable algorithm. The inclusion of momentum in gradient descent was introduced by Qian (1999), who extended the formula of gradient descent to include a momentum parameter $\rho \in [0, 1]$ that multiplied the change in weights at the last iteration. The term "momentum" is used to describe the effect of past gradients on the current update step. This is defined as the first moment. Hence, the update on a parameter vector θ_i on iteration i to minimise the loss function including momentum can be defined as:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_{i-1}) + \rho(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{i-2})$$

$$\Delta\boldsymbol{\theta}_i = \boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1} \implies \Delta\boldsymbol{\theta}_i = -\eta \nabla \mathcal{L}(\boldsymbol{\theta}_{i-1}) + \rho \Delta\boldsymbol{\theta}_{i-1}$$

RMSProp is an unpublished algorithm proposed by Hinton et al. (2012) in a course on the online learning platform Coursera. It is an extension of the resilient backpropagation (RProp) algorithm from Riedmiller and Braun (1993) to work more efficiently with mini-batches. This involves keeping a moving average of the squared gradient for each weight at each iteration i called the mean squared average of gradients denoted by v. Each mean square is dependent on decay rate β and learning rate η. RMSProp automatically adjusts the learning rate η to more efficiently minimise the loss function. Riedmiller and Braun (1993) referenced an unpublished recommendation from Tijmen Tieleman that dividing the gradient of the loss function with respect to the weights by the moving average makes the learning rate work much better, hence this is incorporated into RMSProp. To avoid zero denominators, a small constant $\epsilon \to 0^+$ is added to the learning rate update to ensure numerical stability. Hence the process of RMSprop to estimate the second moment used in gradient descent to optimise the parameters of the loss function can be defined as:

$$v_i = \beta v_{i-1} + (1 - \beta)(\nabla \mathcal{L}(\boldsymbol{\theta}_i)^2$$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \frac{\eta}{\sqrt{v_i} + \epsilon} \nabla \mathcal{L}(\boldsymbol{\theta}_i)$$

The final enhancement of the Adam algorithm is bias correction. As the initial estimates are biased towards zeros, the first and second moments are divided by $1 - \text{moment}^i$ at each iteration i to correct the accuracy of the estimates. This ensures more reliable training of the model parameters.

The Adam algorithm brings together all these ideas in a single optimisation algorithm. However, there are changes to the momentum, such as the learning rate not being included in the momentum step of Adam as it is used later in the parameter updates. Instead, $1 - \beta_1$ replaces the learning rate η, where β_1 is the momentum decay rate used instead of ρ. In addition, to distinguish between the decay rates in momentum and RMSProp, the exponential decay rate in the derivation of the second moment will be referred to as β_2. Hence the Adam optimisation algorithm to minimise the loss function can be defined as:

Algorithm 11 Adam

1: Start with a differentiable loss function $\mathcal{L}(y, \hat{y})$, learning rate η, momentum decay rates β_1, β_2, a small constant ϵ, number of epochs I and/or convergence criterion ψ.
2: Initialise a parameter vector $\boldsymbol{\theta}_0$
3: Let $m_0 = 0, v_0 = 0$
4: **while** $i < I$ or $E_i^{val} > \psi$ **do**
5: Shuffle the training data
6: Divide the dataset into b batches $\{B_1, ..., B_b\}$
7: **for** $j \in \{1, ..., b\}$ **do**
8: Calculate the gradient $g_{j;i} = \nabla \mathcal{L}_j(\boldsymbol{\theta_{j;i-1}})$
9: Calculate the first moment $m_{j;i} = \beta_1 m_{j;i-1} + (1 - \beta_1) g_{j;i}$
10: Calculate the second moment $v_{j;i} = \beta_2 v_{j;i-1} + (1 - \beta_2) g_{j;i}^2$
11: Perform bias correction for the first and second-moment estimates

$$\hat{m_{j;i}} = \frac{m_{j;i}}{1 - \beta_1^i}$$
$$\hat{v_{j;i}} = \frac{v_i}{1 - \beta_2^i}$$

12: Update the model parameters:

$$\boldsymbol{\theta}_{j;i} = \boldsymbol{\theta}_{j;i-1} - \eta \frac{\hat{m_{j;i}}}{\sqrt{\hat{v_{j;i}}} + \epsilon}$$

13: Calculate the value of the loss function for the batch j: $\mathcal{L}_j(\boldsymbol{\theta}_{j;i})$
14: **end for**
15: Calculate the total classification error on the validation dataset:

$$E_i^{val} = \sum_{j=1}^{b} \mathcal{L}_j(\boldsymbol{\theta}_{j;i})$$

16: **end while**

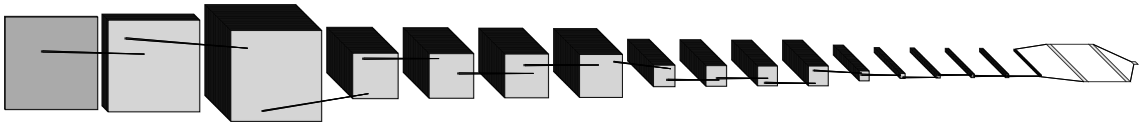


Figure 3.3.: The convolutional neural network architecture employed to classify an RGB image of dimension 224×224 into one of 9 classes.

The architecture used for the convolutional neural network in my analysis was based on the model used in MedMNIST V2 (Yang et al., 2021, 2023, 2024). However, updates were made to the architecture. The updates include the introduction of if/else conditions to trigger extra hidden layers for higher-dimensional data. This was implemented as the neural network used in Yang et al. (2021, 2023) only works for images with dimension 28×28. I added extra convolutional layers with added padding and max pooling to process higher dimensional images, which were only triggered if the resolution was a specific value. This ensured that all images regardless of dimension went into the fully connected layer in 64 feature maps of dimension 4×4. I used the ReLU activation for all layers of my data. Whilst a brief description of the model is found in table 3.1, further details can be found by exploring the **mach_learn.py** source code in my GitHub repository `https://github.com/jackhodgkinson/modules`.

Table 3.1.: A comprehensive overview of the differences in the Convolutional Neural Network architectures for the different image resolutions of Yang et al. (2024)

Image Res.	# of Layers	Kernel Size	Max Pooling	Padding	Adaptation for Resolution
28×28	6	3×3	2×2 filter with a 2-pixel stride in Layer 2, 5	1 row and 1 column of zeros added to either side of the image in Layer 5	Standard NN from Yang et al. (2023)
64×64	9	3×3	2×2 filter with a 2-pixel stride in Layer 2, 5, 8	1 row and 1 column of zeros added to either side of the image in Layer 8	No padding in layer 5
128×128	11	3×3	2×2 filter with a 2-pixel stride in Layer 2, 5, 8, 10	1 row and 1 column of zeros added to either side of the image in Layer 10	No padding in layers 5 or 8
224×224	13	3×3	2×2 filter with a 2-pixel stride in Layer 2, 5, 8, 10, 12	1 row and 1 column of zeros added to either side of the image in Layer 8, 12	No padding in layer 5

Following the original analysis of Yang et al. (2023), I employed a learning rate $\eta = 0.001$ and a batch size of 128. However, I changed the number of epochs to 500. I trained my model on both a test and validation dataset to ensure that the model did not overfit. To accelerate the convergence of the model, I employed 10 early stopping rounds, meaning that the model will stop training if there is no improvement in the validation loss in 10 epochs. I analysed my model using the log loss for both binary and multi-class classification problems. I utilised the Adam optimisation algorithm due to its impressive performance in previous convolutional neural networks.

3.3.2. eXtreme Gradient Boosting (XGBoost)

Overview

Decision Trees

A decision tree is a predictive hierarchical model which maps observations to a conclusion through the stratification of the predictor space into regions by decisions (Tan, 2015). Analogous to a physical tree, the decision tree consists of a root, branches and leaves. Whilst there are many types of decision trees such as multiway trees and random forests, the focus of this section is related to Classification and Regression Trees (CART). CART decision trees allow the segmentation of data into relevant categories by using recursive binary splitting - splitting the data based on questions or statements with only two possibilities. When a decision tree categorises inputs into discrete groups or classes, this is known as a classification tree.

A CART consists of a root node, branches and leaf nodes. Data is often split into a simple decision tree by Boolean expressions, which can be based on both continuous and categorical data. The root node contains the initial Boolean expression to separate the data into two groups. Branches further split the data based on either new or more specific expressions. For example, a branch may separate input data based on a population by age into > 60 and ≤ 60, or by their sex into Male and Female. The leaf nodes represent a final classification of data into a certain class. Note that there can be duplicate leaf nodes in a CART representing the same class. Below is a basic classification tree modelled off grouping participants on the UK TV show *The X Factor* into the relevant judging categories:

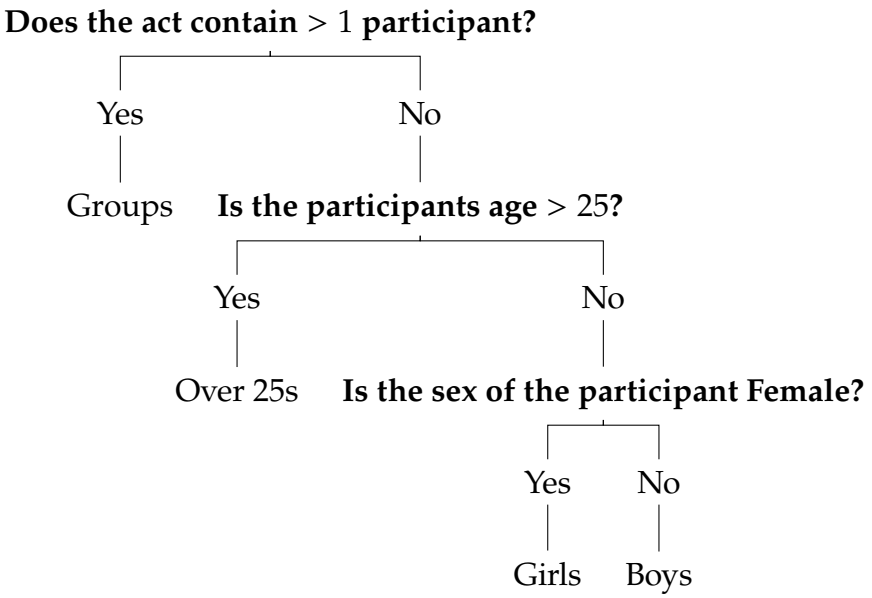


Figure 3.4.: Classification of X Factor Participants into Judges' Categories using a Decision Tree

Boosting

Boosting is a method used to improve, or "boost", the predictive accuracy of an algorithm. The process involves growing trees sequentially, meaning that each

new tree is grown using information from previously grown trees (James et al., 2023). The process of boosting builds strong predictive models by combining the predictions of multiple weak learners, such as decision trees. A weak learner can be defined as a classifier where the error rate of the learner is only slightly better than random guessing (Hastie et al., 2009). Whilst the question of whether weak learning algorithms can be "boosted" to a singular strong learning algorithm was originally posed by Kearns (1988); Kearns and Valiant (1994), the first provable boosting algorithm was created by Schapire (1990) who is credited as one of the founders of the widely used AdaBoost algorithm (Freund and Schapire, 1996). AdaBoost involves iteratively building trees consisting of only an input node and two leaves, called stumps, from the training data until a maximum number of stumps is reached or the trees fit the data perfectly. Each stump's influence is scaled in the actual classification based on how well it compensated for the previous errors, and the errors of that stump influence the building of the next stump.

Gradient Boosting

Gradient Boosting is an algorithm used to fit a predictive model to training data, with applications in both classification and regression problems. Gradient boosting in classification involves initially assigning a prediction to a single leaf for each sample. The initial predictions are derived from the log odds value calculated using the logit function seen in equation 3.7, to represent a preliminary estimate for the likelihood of each sample belonging to the positive class.

Subsequently, gradient boosting proceeds by constructing decision trees of user-defined size that incorporate information on the previous errors, similar to AdaBoost. Each tree's contribution to the final prediction is weighted based on the performance during training. The weights, denoted by $\mathbf{w}$, play a crucial role in determining the optimal way to split nodes in each tree. The algorithm iteratively refines the predictions by building new trees, incorporating all previous errors and the weighted contribution of each previous tree. This repeats until a maximum number of trees/iterations is reached or until performance converges on a validation dataset. The term for the collection of trees is called an ensemble.

During each iteration, the algorithm computes pseudo-residuals for each sample, which represent the errors that need correcting in subsequent iterations. The pseudo-residuals are derived from the negative gradient of the loss function with respect to the predictions of the previous ensemble. The algorithm leverages gradient descent to update model parameters to minimise the loss function, with respect to the predictions made by the tree ensemble. The algorithm utilises the gradient to adjust the direction of the steepest descent of the loss function. This is carried out proportionally to the pseudo-residuals and moderated due to the learning rate parameter η. The adaptive step size ensures that predictions take small steps in the right direction, resulting in lower variance (Friedman, 2001). As the algorithm proceeds through each iteration, the log odds values from newly added trees are cumulatively summed with those from previous trees in the ensemble. The final sum represents the contribution of each tree's prediction.

During the construction of each decision tree, the algorithm determines the

optimal splits for each node by assessing which splits minimise the loss function. This partitions the data into subgroups related to each class with the highest accuracy. This process ensures that the future trees constructed focus on areas where previous models performed poorly.

Gradient boosting can iteratively assess the performance of the model on a validation dataset to remove the risk of overfitting to the training data. Training can be terminated when either the maximum number of iterations I has been reached or the validation loss meets a criterion ψ.

Once the training process has been completed, the model is ready for prediction on unseen data. The predicted probability of a sample belonging to a class k is calculated using the cumulative sum of the log odds. For binary classification, the probabilities are calculated using the logistic function from equation 3.6 on the sum of predictions from all trees in the ensemble whereas in multi-class classification, the probabilities are obtained from the softmax function in equation 3.11. Once the loss function has been optimised, the predicted classification is based on the class that takes the maximum probability, in the same fashion as LDA and Logistic Regression.

The algorithm of gradient boosting to minimise the loss function is the following iterative process:

Algorithm 12 Gradient Boosting

1: Start with M_{train} samples of training data $(\mathbf{x}_i, y_i)_{t=1}^{M_{train}}$ and M_{val} samples of validation data $(\mathbf{x}_j, y_i)_{j=1}^{M_{val}}$ split into K classes, a differentiable loss function $\mathcal{L}(y, \hat{y})$, learning rate η, number of iterations I and/or convergence criterion ψ.
2: Calculate initial predictions using the log odds of class k being nwthe positive class:
$$F^{(0)}(\mathbf{x}) = \log \frac{\mathbb{P}(Y = k|\mathbf{X})}{1 - \mathbb{P}(Y = k|\mathbf{X}}$$
3: **while** $i < I$ or $E_i^{val} > \psi$ **do**
4: Compute the pseudo-residual
$$r_t^{(i)} = -\left[\frac{\partial \mathcal{L}(y_t, F^{(i-1)}(\mathbf{x}_t))}{\partial F^{(i-1)}(\mathbf{x}_t)}\right] \text{ for } t \in \{1, ..., M_{train}\}$$
5: Fit a tree $h(\mathbf{x}, \mathbf{w}^{(i)})$ to the $r_t^{(i)}$ values where
$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \sum_{t=1}^{M_{train}} L(y_t, F_{i-1}(\mathbf{x}_t) + h(\mathbf{x}_t, \mathbf{w}))$$
6: Update the model:
$$F_i(\mathbf{x}) = F_{i-1}(\mathbf{x}) + \eta h(\mathbf{x}, \mathbf{w}_i)$$
7: Calculate the weights $\mathbf{w}_i$ for the tree $h(\mathbf{x}, \mathbf{w}^{(i)})$ where:
8: Compute the validation error for iteration i on the validation dataset:
$$E_i^{val} = \frac{1}{M_{val}} \sum_{j=1}^{M_{val}} \mathcal{L}(y_j, F_i(\mathbf{x}_j^{val}))$$
9: **end while**

XGBoost

XGBoost is an advanced implementation of gradient boosting that introduces several new parameters and techniques to improve training speed and generalisability. Created by Chen and Guestrin (2016), the model has proved extremely successful in the data science community, with 17 out of the 29 machine learning tasks posted on Kaggle in 2015 being won using XGBoost (Ogunleye and Wang, 2019).

In supervised learning, XGBoost operates on labelled datasets consisting of M samples and p features, partitioned into training, validation and test data of the form $(\mathbf{x}_m, y_m)_{m=1}^{M_{\text{train/val/test}}}$. It commences with an initial prediction probability by calculating the log odds of belonging to the positive class, serving as an initial calculation of the value for the loss function.

Trees are built iteratively to minimise an objective function, which is an extension of the loss function to account for regularisation. The objective function for XGBoost on the sum of the loss functions over M samples and the i-th tree of I iterations/trees is defined as:

$$\begin{aligned} O_i &= \sum_{m=1}^{M} \mathcal{L}^{(m)}(y, \hat{y}) + \Omega(F_i(\mathbf{x})) \\ &= \sum_{m=1}^{M} \mathcal{L}^{(m)}(y, \hat{y}_{i-1} + F_i(\mathbf{x}_m)) + \Omega(F_i(\mathbf{x})) \end{aligned}$$

where $F_i(\mathbf{x})$ is the i-th tree in the ensemble on the data $\mathbf{x}$ and $\Omega(F_i(\mathbf{x}))$ is the regularisation term which is penalising the complexity of individual trees. The regularisation term is expressed as:

$$\Omega(F_i(\mathbf{x})) = \gamma l + \lambda_1 \sum_{j=1}^{p} |w_j| + \lambda_2 \sum_{j=1}^{p} w_j^2 \tag{3.15}$$

This incorporates the hyperparameters λ_1, λ_2, representing the regularisation parameters of L1 and L2 regularisation respectively, and w_j the weight associated with the j-th feature. The weights are adjusted at each iteration. Each tree is penalised individually by the γ parameter, where values of $\gamma > 0$ encourage tree pruning during model building. By default, XGBoost sets $\gamma = 0$ and $\lambda_1 = 0$ meaning that the model performs Ridge regularisation. I remained with the original regularisation in my analysis. For more information on tree pruning, please refer to James et al. (2023).

Whilst gradient boosting takes advantage of first-order methods to minimise the loss function, XGBoost utilises a second-order Taylor approximation of the loss function, working similarly to the Newton-Raphson method for optimisation as discussed in Section 2.5. This contributes to the rapid convergence of XGBoost. Furthermore, whilst gradient boosting works on a by-sample basis, XGBoost works in batches. Unlike CNN, the batch size is calculated automatically to minimise memory usage and maximise computational efficiency. Formally, the objective function at tree i can be approximated by a second-order Taylor expansion of the

loss function:

$$O_i \approxeq \sum_{m=1}^{M} \left[\mathcal{L}^{(m)}(y, \hat{y}_{i-1}) + \frac{\partial L^{(m)}}{\partial \hat{y}_{i-1}} F_i(\mathbf{x}_m) + \frac{1}{2} \frac{\partial^2 L^{(m)}}{\partial \hat{y}_{i-1}^2} F_i^2(\mathbf{x}_m) \right] + \Omega(F_t(\mathbf{x}))$$

To simplify, constants will be removed and let $g_m = \frac{\partial L^{(m)}}{\partial \hat{y}_{t-1}}$ denote the gradient and $h_m = \frac{\partial^2 L^{(m)}}{\partial \hat{y}_{t-1}^2}$ denote the Hessian. This gives the following simplified objective function:

$$\tilde{O}_t = \sum_{m=1}^{M} \left[g_m F_t(\mathbf{x}_i) + \frac{1}{2} h_m F_t^2(\mathbf{x}_i) \right] + \Omega(F_t(\mathbf{x}))$$

XGBoost trains its model using the method of gradient boosting as seen in algorithm 12 however with the objective function rather than the loss function. It iteratively builds new trees until the objective function is minimised or a maximum number of trees is reached.

Many optimisations are performed to speed up the process. As XGBoost splits on nodes without considering the future splits of the leaves, XGBoost is defined as a greedy algorithm because it does not look forward to whether the current split is optimal longer term. This allows the splitting criteria to be assumed hence the construction of trees is fast. However, when there are many features and observations, the algorithm becomes slow as it needs to look through every possible threshold for splitting. Hence, XGBoost has the option to employ an approximate greedy algorithm, which segments the data into quantiles and uses the quantiles to test thresholds, not every single feature. For a large dataset, simple tasks can take a large amount of time, so a sketch algorithm splits data into segments and completes computations in parallel across different cores. The quantile sketch algorithm combines the values from each computation to make an approximate histogram, which calculates approximate quantiles. XGBoost uses a weighted quantile sketch to distribute the number of features in each quantile with the weights being derived from the Hessian. In Python, the model allows for an automatic selection of the splitting method which is most appropriate to the input data and parameters, which I employed in my analysis. For further details on the algorithms used to split the nodes of the trees, please refer to Chen and Guestrin (2016).

When values are missing, XGBoost splits the dataset into two groups: one with missing features and one without. Although normal thresholding takes place on the complete data, the missing values are allocated into each split node in a way that minimises the loss function. This process is called Sparsity-Aware Split Finding.

XGBoost provides users with the option of specifying the objective function of the model. Similar to gradient boosting, the performance of the model can be analysed using a validation dataset to ensure that the model does not overfit to the training data. However, in XGBoost, there is an additional parameter to control convergence speed. One can employ the method of early stopping as in a CNN, meaning that training concludes when there has been no improvement in the value of the objective function for a number of rounds. In my analysis, I set the early stopping rounds to 10 to follow consistency with the neural network.

I specified the objective for binary classification to be logistic regression so that class probabilities are output. Similarly for multi-class classification, I specified the objective to be softprob which uses the softmax objective function and converts the outputs to predicted probabilities.

I set the maximum number of trees/iterations to 500 to align with both the CNN and Logistic Regression iteration numbers and kept the maximum tree depth as 6. XGBoost has a default learning rate of $\eta = 0.3$. A review of literature on the use of XGBoost in medical image classification such as Liew et al. (2021); Maleki et al. (2023) used high learning rates of 0.5 and 0.6 respectively, but I chose to approach the classification tasks with XGBoost using a slower but more conservative and stable approach, setting $\eta = 0.1$.

4. Datasets and Experimental Design

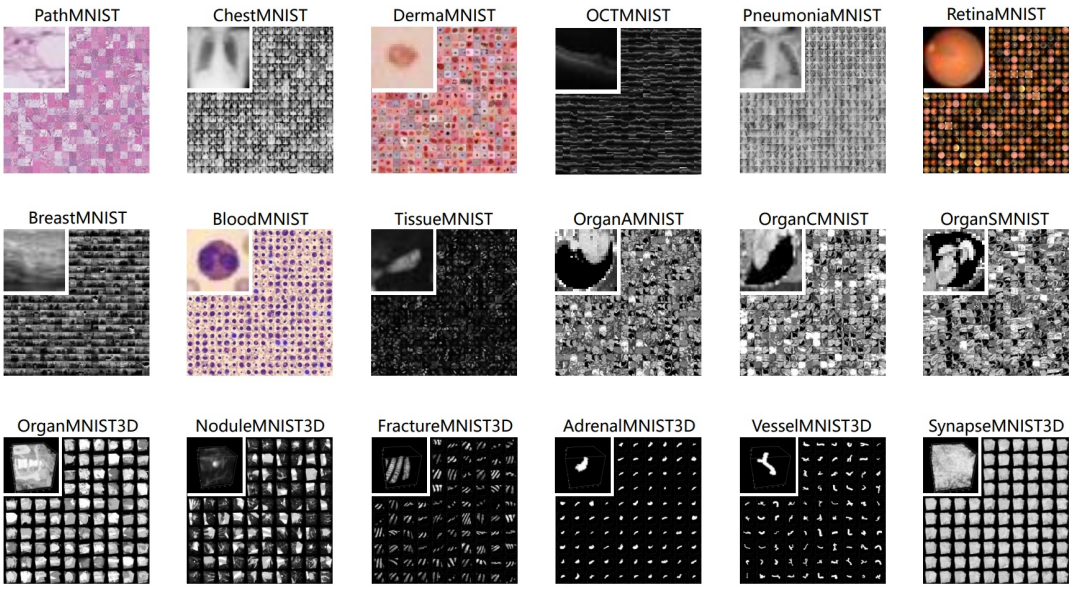
4.1. Overview

This section serves as an introduction to the datasets utilised in the analysis, offering a comprehensive summary detailing the design of experiments conducted. The section encompasses the information represented by each dataset, as well as methods of data collection, data structure and allocation, The experimental design section elaborates on the implementation of techniques completed as part of the analysis such as data pre-processing. The methods conducted for model tuning are described in detail as well as defining the metrics used to analyse the performance of the models.

4.2. Datasets

4.2.1. MedMNIST

Figure 4.1.: Full collection of MedMNIST V2 datasets reproduced from Yang et al. (2023)

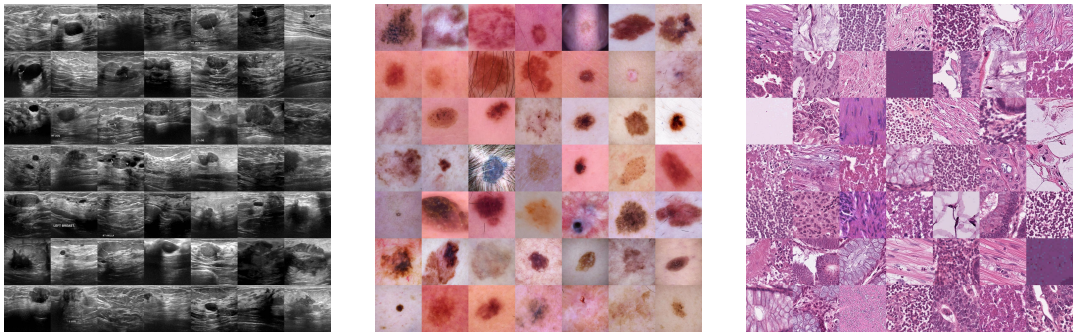


The MedMNIST+ datasets, derived by Yang et al. (2021, 2023, 2024) is a collection of large-scale MNIST-like (Deng, 2012) datasets containing standardised medical images, including both 2D and 3D images as seen in Figure 4.1. Originally in MedMNIST V1 and MedMNIST V2, the data had been pre-processed into a smaller resolution, with each dimension being equal to size 28, aligning with the resolution of the samples in the MNIST dataset.

However, the latest release of the MedMNIST+ datasets, published on January 20th 2024 (Yang et al., 2024), provides higher resolution images to complement the exisiting 28x28 size images (or 3x28x28 for colour images) in MedMNIST V1 and V2. MedMNIST+ introduced resolution dimensions of 64, 128 and 224 to the collection, enabling analysis to take place in higher dimensions.

With the decision to focus on data related to oncology for the paper, the thesis focuses on three datasets: BreastMNIST, DermaMNIST and PathMNIST. High-resolution samples of these datasets can be seen in Figure 4.2.

Figure 4.2.: Fourty-nine high-resolution random samples

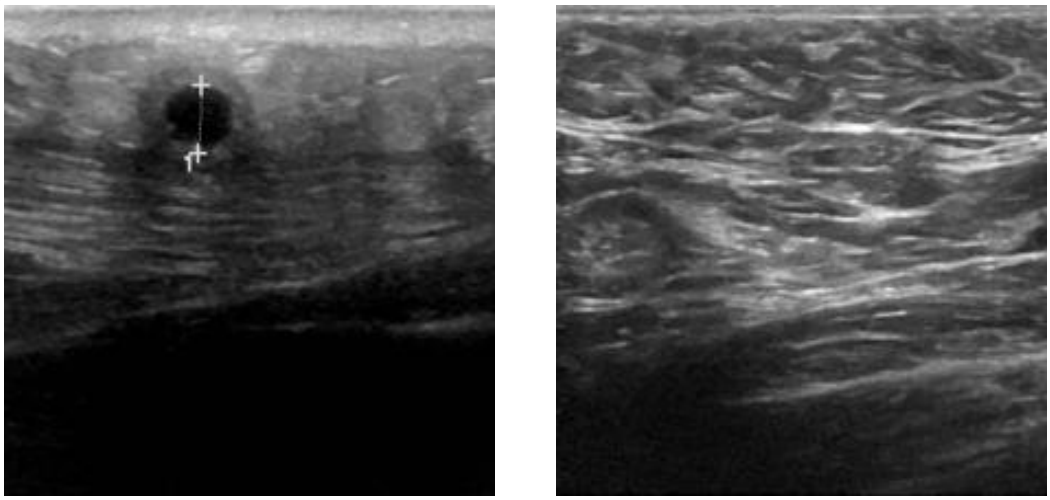


(a) BreastMNIST (b) DermaMNIST (c) PathMNIST

The datasets provide opportunities for binary and multi-class classification relating to cancerous features. Each of the datasets comes in the four resolutions released in MedMNIST+, of which I will investigate classification on each dataset at all the provided resolutions.

4.2.2. BreastMNIST

Figure 4.3.: Two high-resolution BreastMNIST samples of each of the data classes



(a) Malignant (b) Normal, benign

BreastMNIST contains 780 samples of breast ultrasound images from Al-Dhabyani et al. (2020). The data has been collected from 600 female patients aged between

25 and 75 years old. The original data had been classified into three classes: *normal*, *benign* and *malignant*. However, as the images have been resized down from 500×500, the MedMNIST paper simplifies the classification by combining both the *normal* and *benign* classes into a single class named *normal, benign*. This creates a binary classification problem, which assists with the low resolution of the resized images in comparison to the original data. Visualisation of a malignant and normal/benign sample can be seen in Figure 4.3.

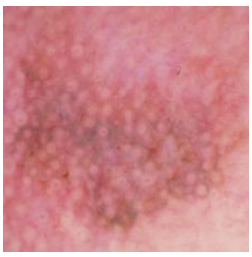
Breast cancer is one of the most common causes of death in those of the female sex worldwide. Early detection of such cancerous lesions can cause a significant reduction in the amount of deaths. Ultrasound scans are used in breast cancer diagnosis to differentiate solid lesions as benign or malignant, assisting with the staging and management of breast cancers (Svensson, 1997). Ultrasound scans have proved successful in determining benign lesions as seen in Stavros et al. (1995), allowing for easier follow-up as opposed to the traditional method of biopsy. The paper also identified that malignant lesions could be identified with a high degree of confidence with negative predictive values as high as 99.5% and sensitivity as high as 98.4%.

The original dataset from Al-Dhabyani et al. (2020) was collected and stored at the Baheya hospital in Egypt, where the collection and labelling of the images were completed by radiologists in a timeframe of around one year.

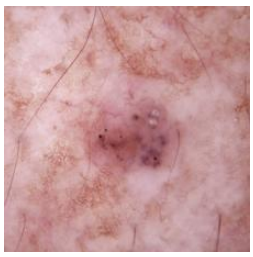
The BreastMNIST dataset has been pre-allocated into a training set, validation set and test set in the Yang et al. (2021, 2023, 2024) paper. The training dataset contains 546 samples, the validation set 78 samples and the test set 156 samples. Therefore, the split adheres to a ratio of 7:1:2.

4.2.3. DermaMNIST

Figure 4.4.: Seven high-resolution DermaMNIST samples representing each of the data classes

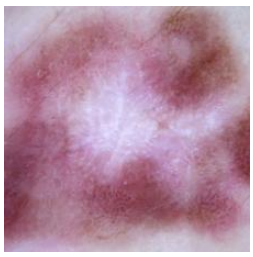


(a) Actinic keratoses and intraepithelial carcinoma

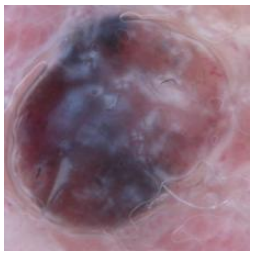


(b) Basal cell carcinoma

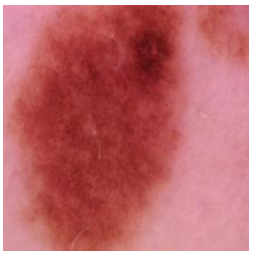
(c) Benign keratosis-like lesions



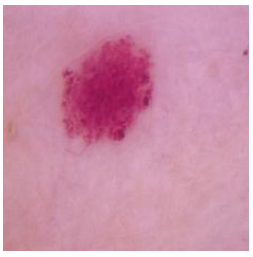
(d) Dermatofibroma



(e) Melanoma



(f) Melanocytic nevi



(g) Vascular lesions

The DermaMNIST dataset contains 10,015 samples based on the HAM10000 (Human Against Machine with 10000 training images) dataset from Tschandl et al. (2018), a large collection of multi-source dermatoscopic images of common pigmented skin lesions. The dataset is a formulation of images from different populations acquired and stored in different data modalities. Due to this, the data has been acquired and cleaned using different methods, been semi-automated by specifically trained neural networks.

The HAM10000 dataset was created to address the problem of bias towards melanocytic lesions in dermatoscopic data. As a result of this bias, previous research focused mainly on the differentiation between melanoma and nevus. This meant that non-melanocytic pigmented lesions were often disregarded, although they are commonly observed in clinical practice.

The data was collected over 20 years from two different sites and stored in different ways. The first source is from the Department of Dermatology at the Medical University of Vienna, Austria. This set of images consists of lesions from patients who have been referred to a tertiary European referral centre that specialises in the early detection of melanoma for high-risk groups, normally those with a high number of nevi and a personal or family history of melanoma. They started to collect images before the era of the digital camera, hence images were first stored as diapositives that were digitised, cropped and corrected into JPEG images then later samples were captured using digital technology such as the DermLiteTM or MoleMaxHD systems. The skin practice of Cliff Rosendahl in Queensland, Australia was the second source of dermatoscopic images. The image

set includes lesions from patients at a primary care facility in an area of high risk of skin cancer. These images were captured solely by Rosendahl with either a DermLite Fluid or DermLite DL3 with immersion fluid. The images were collected from 2008 until May 2017.

As the data was sourced from medical facilities, extraction of only dermatoscopic images was required as the samples also contained clinical close-ups and overviews. From the Australian data collection, 1501 image files were labelled into groups of *overviews, close-ups* and *dermatoscopy*. This was used as a training set in a neural network to classify the images according to their type. This achieved an accuracy of 98.68%, accelerating the filtering of dermatoscopic images from the collection.

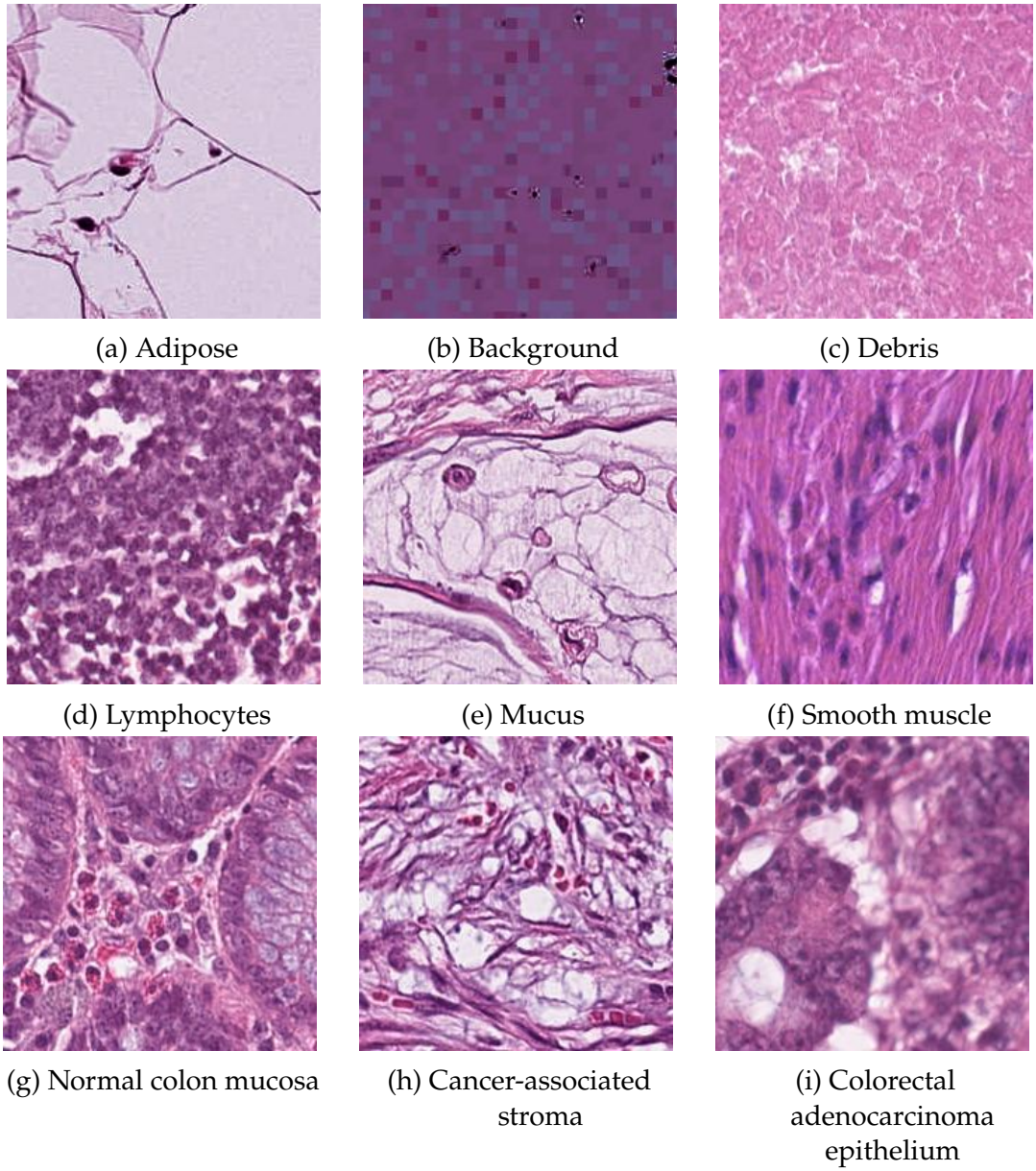
Once the data had been collated into one set, the source images were resized from 3x600x450 into the respective resolutions of MedMNIST+.

The dataset has been formulated into a multi-class classification problem. There are 7 classes of lesions present in the data, two of which are cancerous. Whilst the class labels were chosen to be generic for simplicity, they are representative of over 95% of all lesions encountered in clinical dermatology practice. All important diagnostic categories for pigmented lesions are covered in the classes. Except for vascular lesions, all lesions have variants devoid of pigment. More than 50% of lesions have been confirmed by pathology, with the rest confirmed by either follow-up, expert consensus, or confirmation by in-vivo confocal microscopy. Non-pigmented lesions have a larger number of possible diagnoses hence samples of this type are not included in the DermaMNIST dataset. The classes that are present are: *actinic keratoses and intraepithelial carcinoma, basal cell carcinoma, benign keratosis-like lesions, dermatofibroma, melanoma, melanocytic nevi* and *vascular lesions*. Visualisation of a singular sample of each type of lesion can be seen in Figure 4.4.

The DermaMNIST dataset has been pre-allocated into a training set, validation set and test set. The training set contains 7007 samples, the validation set 1003 samples and the test set 2005 samples. As with BreastMNIST, the split adheres to an approximate 7:1:2 ratio.

4.2.4. PathMNIST

Figure 4.5.: Nine high-resolution PathMNIST samples representing each of the data classes



(a) Adipose (b) Background (c) Debris

(d) Lymphocytes (e) Mucus (f) Smooth muscle

(g) Normal colon mucosa (h) Cancer-associated stroma (i) Colorectal adenocarcinoma epithelium

PathMNIST is a dataset of 107,180 samples of coloured histological images displaying human colorectal cancer (CRC) and healthy tissue (Kather et al., 2019). For almost every CRC patient, histological slides of tumour tissue are routinely available, making these images optimal for training models for classification.

The dataset has been created by combining two similar datasets. The first dataset is the *NCT-CRC-HE-100K*, which provides 100,000 non-overlapping image patches from histological images stained by hematoxylin and eosin (H&E). These images were manually extracted from 86 H&E stained human cancer tissue slides obtained from formalin-fixed paraffin-embedded (FFPE) tissue samples collected by the NCT Biobank (National Center for Tumor Diseases, Heidelberg, Germany) and the UMM pathology archive (University Medical Center Mannheim, Mannheim,

Germany). The tissue samples contain CRC primary tumour slides and tumour tissue from CRC liver metastases. Normal tissue classes were augmented with non-tumorous regions from gastrectomy specimen to increase variability. H&E staining is completed on the colorectal tissue samples to highlight areas of potential cancerous growth.

The second dataset used for the construction of PathMNIST is the *CRC-VAL-HE-7K* dataset. This dataset contributes 7,180 image patches from 50 patients with colorectal adenocarcinoma. The tissue samples were provided by the NCT tissue bank, and have no overlap with NCT-CRC-HE-100K.

The images in the dataset have been resized from 3×224×224 to the respective resolutions. As the highest resolution image in the analysis is 3x224x224, this means the PathMNIST samples at resolution 3x224x224 align in resolution with the source images.

The source images were colour normalised using the Macenko method (Macenko et al., 2009), which addresses the variations in colour and intensity across histological samples. The variations would make consistent analysis difficult, so Macenko et al. (2009) addressed this by developing a technique involving colour normalisation to achieve a standardised representation of histological images. The normalisation was done due to subtle differences in the red and blue hues in the original images, which when tested on a convolutional neural network resulted in a biased classification.

There are nine types of tissue classes present in the data, of which two types display properties related to colorectal cancer. The classes are defined as: *adipose, background, debris, lymphocytes, mucus, smooth muscle, normal colon mucosa, cancer-associated stroma* and *colorectal adenocarcinoma epithelium*. Visualisations of each class can be seen in Figure 4.5.

For the training, testing and validation set split of the PathMNIST, the sets have been obtained differently from that of BreastMNIST and DermaMNIST. As PathMNIST is a large dataset of 107,180 samples obtained from two individual datasets, NCT-CRC-HE-100K and CRC-VAL-HE-7K, the pre-defined splits sourced from Yang et al. (2021, 2023, 2024) take a unique approach. The training and validation sets have been sourced from NCT-CRC-HE-100K, with a 9:1 ratio of training to validation being provided, meaning that the training set has 89996 samples and the validation set has 10004 samples. However, the test dataset is comprised of the CRC-VAL-HE-7K dataset, which contains 7180 samples, meaning that the test and training/validation datasets are obtained from different data sources.

4.3. Experimental design

4.3.1. Implementation

The analysis for this project has been completed using the Python v3.11.2 (Van Rossum and Drake, 2009) programming language on the JupyterLab v3.5.3

interface of the Minerva server owned by the Statistics group at the University of Manchester's Department of Mathematics. All analysis was completed on data sourced from the package MedMNIST v3.0.1 (Yang et al., 2021, 2023, 2024).

The statistical analysis completed in this paper was done using the Scikit-learn (sklearn) v1.2.1 package (Pedregosa et al., 2011). The method of Linear Discriminant Analysis was implemented using the *LinearDiscriminantAnalysis* function from the *discriminant_analysis* module in sklearn and Logistic Regression was implemented using the *LogisticRegression* function from the *linear_model* module.

To build the convolutional neural network (CNN), PyTorch v2.2.1 (Paszke et al., 2019) was used to construct the model architecture. Whilst utilising the existing neural network architecture taken from Yang et al. (2021, 2023, 2024), further edits were required to make the model compatible with higher resolution data by introducing extra convolution and pooling layers. The Skorch v0.15.0 (Tietz et al., 2017) package was used to train the CNN, allowing PyTorch models to be used in a sklearn compatible format. The architecture was specified within the *NeuralNetClassifier* function alongside callback functions to aid model training implemented from the *callbacks* module. These include: *EpochScoring* to score the performance on the training and validation data after each epoch, *EarlyStopping* to stop model training if the loss doesn't improve for a specified number of rounds to prevent under or over-fitting and *Checkpoint* to save the parameters of the optimal model. By default, the function *NeuralNetClassifier* performs cross-validation on the training dataset. To mitigate this, I took advantage of the *predefined_split* function from Skorch's *helper* module to implement the pre-defined validation data of MedMNIST+ for the model. This was encapsulated inside the *Dataset* wrapper in the *dataset* module, which is normally used for inputting data into the Dataloader form for PyTorch.

The XGBoost classification model was implemented using the XGBoost v2.0.3 (Chen and Guestrin, 2016) package, which also supports a sklearn interface. Unlike CNN, many of the necessary callbacks are already a part of the core XGBoost functions, such as the *early_stopping_rounds* option inside the model. XGBoost allows user-specified training and validation data, using the *eval_set* parameter inside the *XGBClassifier.fit()* call. For this to be successful, each pair of features and labels must be inside a tuple, and all evaluation datasets must be inside a single list.

The code for Linear Discriminant Analysis, Logistic Regression and XGBoost were run on the CPU of the Minerva server. This can run 40 processes simultaneously, with a total server CPU memory of 1.48TB (1480GB).

To accelerate run-time, the *device='cuda'* option was used for the convolutional neural network. Compute Unified Device Architecture (CUDA) is a parallel computing platform that allows for the use of graphics processing units (GPUs) for simultaneous data computation, hence decreasing model runtime. Developed by NVIDIA in 2007 based on the work of Buck et al. (2004), CUDA has empowered data scientists to build and train the most accurate deep learning models. This differs from running models on computer processing units (CPU), which can only handle tasks one at a time. Utilising CUDA for CNN has allowed my analysis of

deep learning models to be successful in the short project timescale, as running these models on CPUs would require more time due to the high computational cost. Although my original plan intended to run the XGBoost model using GPU, the 16GB memory capacity on the GPU meant that the larger datasets did not have adequate memory available for XGBoost to be successful. In this paper, the following NVIDIA CUDA libraries in Python were used: *nvidia-cublas-cu11 v11.11.3.6*, *nvidia-cuda-cupti-cu11 v11.8.87*, *nvidia-cuda-nvrtc-cu11 v11.8.89*, *nvidia-cuda-runtime-cu11 v11.8.89*, *nvidia-cudnn-cu11 v8.7.0.84*, *nvidia-cufft-cu11 v10.9.0.58*, *nvidia-curand-cu11 v10.3.0.86*, *nvidia-cusolver-cu11 v11.4.1.48*, *nvidia-cusparse-cu11 v11.7.5.86*, *nvidia-nccl-cu11 v2.19.3* and *nvidia-nvtx-cu11 v11.8.86*.

To automate the running of the code, I have created different Python files containing main and nested functions that are run from a master file. The master file is run from the terminal, and is dependent on many mandatory inputs such as dataset, resolution and method but also takes some optional inputs including colour, to change RGB colour channels into grayscale, and stage of model i.e. training or testing of the statistical methods. Whilst each method uses the same functions for preprocessing, each method has its own wrapper function which controls the data handling, feature selection (if required), model fitting, prediction, metric calculation and generation of results outputs including text files and plots.

To aid the wrapper functions written for the analysis, the NumPy v1.24.2 package (Harris et al., 2020) was used to perform data organisation, manipulation and summary and for the implementation of random seeds. To assist with outputting the results in the wrapper function, the Pandas v1.5.3 package (McKinney et al., 2010) was used. The in-built Python modules *datetime*, *functools*, *itertools*, *math*, *os*, *random*, *re* and *sys* were used for other functionalities in the wrapper functions.

To access all code used for the paper, please visit my personal GitHub repository `https://github.com/jackhodgkinson`, which has public access. The analysis script written for this paper and the master file for running all code can be found in the **statml_medmnistplus** repository and any wrapper functions discussed inside the **modules** repository.

4.3.2. Data preprocessing

Data pre-processing has been completed using the package Torchvision v0.17.1 (TorchVision maintainers and contributors, 2016), an image and video extension for PyTorch deep learning. To align with the analysis of Yang et al. (2021, 2023, 2024), the data was transformed by converting first to tensors and then values were normalised.

A tensor is a multi-dimensional array that can store data in any dimension. An example of a frequently used tensor in mathematics is a matrix, which is a 2^{nd}-order tensor as it has two dimensions. The dimension of a tensor is referred to as the rank, hence a vector is a rank 1 tensor and a matrix is a rank 2 tensor. Whilst lower-dimensional data can be represented using NumPy arrays, machine learning models in Python make use of storing data in tensors as they are backed by GPU acceleration. The *ToTensor* function from the *transforms* module can convert

NumPy arrays or PIL images into tensors. As the features in MedMNIST are PIL images and the labels are NumPy arrays, the transformation of the data to tensors allows for consistent data storage.

After conversion to tensors, the data was normalised. Normalisation involves scaling the data values to lie numerically in the same interval or scale, therefore all values having the same importance (Tahvili and Hatvani, 2022). Normalisation can take a variety of forms, such as Z-score normalisation (also known as standardisation) and min-max scaling. In this paper, Z-score normalisation was applied to the pixel values, as done by Yang et al. (2021, 2023, 2024). For the i-th pixel, the pixel value denoted $p_{\text{old}}^{(i)}$ is normalised through the following formula

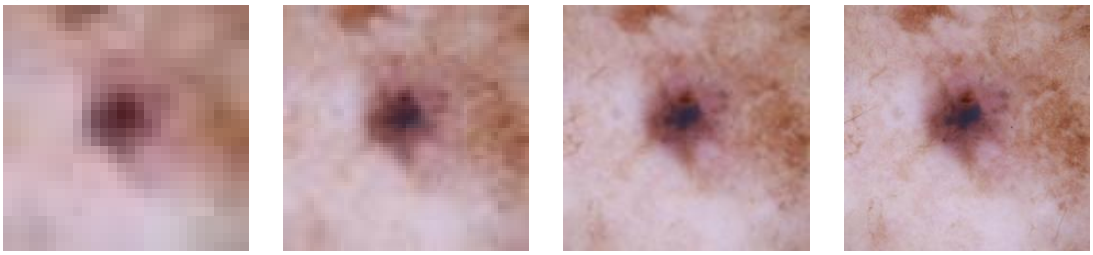
$$p_{\text{new}}^{(i)} = \frac{p_{\text{old}}^{(i)} - \mu}{\sigma}$$

where μ is the mean pixel value and σ is the standard deviation.

In the original data of MedMNIST+, the pixel values lied in the range of $[0, 1]$. Normalisation transformed the pixel values to be in the range $[-1, 1]$, meaning that the mean pixel value is $\mu \approx 0$ and the standard deviation is $\sigma \approx 0$. The pixel values now follow an approximate normal distribution. This helps to stabilise and accelerate classification in statistical machine learning methods.

4.3.3. Resolution

Figure 4.6.: A comparison of image resolution on a singular DermaMNIST sample



(a) Resolution: 28x28 (MNIST-like) (b) Resolution: 64x64 (c) Resolution: 128x128 (d) Resolution: 224x224

With the release of MedMNIST+, the analysis has been extended into comparing the effect of resolution on medical image classification. To achieve this, I will compare each of the methods at the different resolutions provided by MedMNIST+: $28 \times 28, 64 \times 64, 128 \times 128, 224 \times 224$. This can be visualised in figure 4.6, which shows how a sample's appearance differs with resolution.

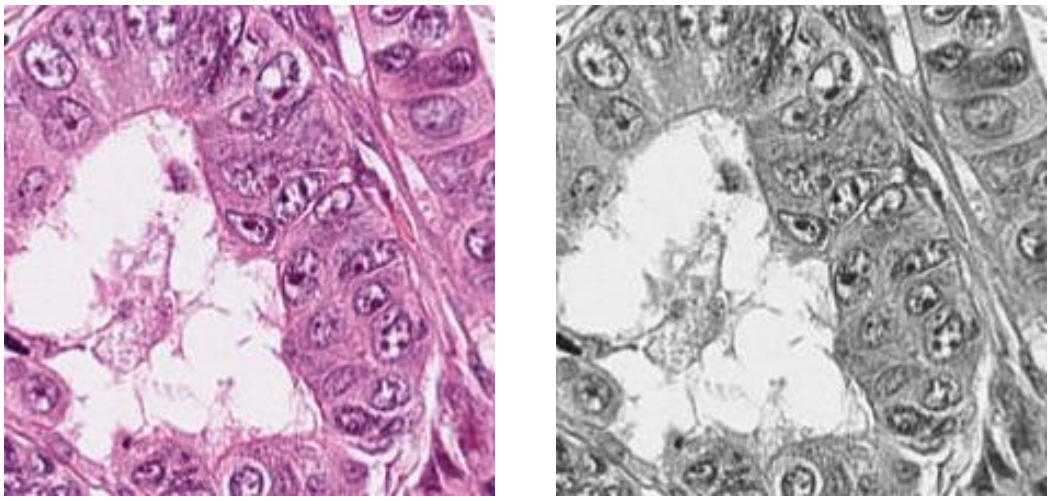
Although there is no dimensional rescaling of images, the structure width, height and channel dimensions must be encapsulated into a single dimension for the data to be compatible with certain methods.

4.3.4. Data partitioning

To assist with model optimisation, I will be making use of the training/validation/test data split provided by Yang et al. (2021, 2023, 2024). Whilst much of the current machine learning and predictive modelling research encourages the use of cross-validation, I have decided against this approach to maximise the amount of data the models train with. The MedMNIST V2 paper and accompanying package have split the data already, which my programme accesses, meaning that no manual splitting of data is completed.

4.3.5. Colour

Figure 4.7.: Comparison of a single PathMNIST high-resolution sample with RGB channels before conversion against the same sample in grayscale post-conversion



To compare the effect of colour on classification, I analysed data that had been converted from having RGB colour channels to grayscale, using the *rgb_to_grayscale* function from the *functional* module inside of *torchvision.transforms*. To do this, I had to convert the original data, that is in NumPy array format, into PIL images using the *fromarray* function of the *Image* module in the *Pillow* v9.4.0 package Clark (2015). Pillow is a fork for the original Python Imaging Library (PIL), that was discontinued in 2011, with added Python3 support. Pillow provides Python with image editing capabilities, whilst the Image module provides a class with the same name which is used to represent a PIL image in Python3. Post colour conversion, the data was transformed back into a NumPy array and subsequently back into tensors. This allows for investigation of the important features representing colour on model performance.

4.3.6. Feature selection

For feature selection, the *feature_selection* module from sklearn was used. This was done through two methods: the removal of constant features and the selection of features by Mutual Information scores.

Firstly, constant features were removed from the data. To test if features were constant among samples, the variance of each feature was analysed. Those features that had zero variance were removed, as this would imply no difference in the data values of the feature among all samples thus constant. The *VarianceThreshold* function in Python managed this, which when set with a threshold of 0 removes all features that have no variability.

Further feature selection was done by utilising Mutual Information scores, which were calculated using the *mutual_info_classif* function. Mutual Information (MI) is a probabilistic measure of the amount of information that is shared between two variables, say x and y (Doquire and Verleysen, 2013). MI works similarly to the R^2 coefficient of determination in providing numeric values on the relationship between variables, however, MI extends beyond the linear and continuous data restrictions of R^2, acting as a measure for non-linear, categorical data. Introduced by Battiti (1994) as a method for feature selection, it has been used extensively in statistical machine learning as a metric to remove redundant features, as it provides a determinant of which features are most influential in accurately predicting labels.

For two variables/features denoted x, y, the Mutual Information between x and y can be calculated by

$$MI(x, y) = \mathbb{E}_{F_{x,y}} \log\left(\frac{f(x, y)}{f(x)f(y)}\right) \tag{4.1}$$

where $F_{x,y}$ is the joint distribution of x and y.

Mutual Information is an application of the Kullback-Leibler (KL) divergence to joint and product distributions. Taking x as the features and y as the labels, the marginal distribution of y is denoted F_y and the conditional distribution of the labels y given the features x is $F_{y|x}$. Recall that the Kullback-Leibler divergence of conditional and marginal distributions is defined as:

$$D_{KL}(F_{y|x}, F_y) = \mathbb{E}_{F_{y|x}} \log\left(\frac{f(y|x)}{f(y)}\right) \tag{4.2}$$

where $f(.)$ denotes the probability density function. Taking the expectation of the KL divergence in equation 4.2 with respect to the marginal distribution of x, denoted F_x, and as the joint distribution is the product of the conditional and marginal distributions from probability theory derives the mutual information equation as in equation 4.1:

$$\begin{aligned}
\mathbb{E}_{F_x} D_{KL}(F_{y|x}, F_y) &= \mathbb{E}_{F_x}\left[\mathbb{E}_{F_{y|x}} \log\left(\frac{f(y|x)}{f(y)}\right)\right] \\
&= \mathbb{E}_{F_x}\mathbb{E}_{F_{y|x}} \log\left(\frac{f(y|x)f(x)}{f(y)f(x)}\right) \\
&= \mathbb{E}_{F_{x,y}} \log\left(\frac{f(x, y)}{f(x)f(y)}\right) \\
&= MI(x, y)
\end{aligned}$$

where $F_{x,y}$ is the joint distribution of x and y.

For the feature selection in my analysis, I created a function called *select_best_features_pct* to select the features that corresponded with the top k% of scores. When the MI scores are derived for a dataset, they are saved in a NumPy file, and loaded when they are required in further feature selection. Although I used mutual information scores to select the best features in my analysis, the wrapper function does work with other scoring metrics.

4.3.7. Hyperparameter tuning

For the statistical methods, hyperparameter tuning was completed. Hyperparameter tuning involves testing different combinations of model parameters to find the optimal combination of parameters to optimise model performance. Whilst this can be done automatically by utilising methods such as Grid Search, Random Search and Bayesian Optimisation, I completed the tuning manually due to the high computational cost of automatic methods and the project time constraint.

For linear discriminant analysis, I ran the model only using the Singular Value Decomposition (SVD) solver and did not employ shrinkage due to the high cost of computation for large, high-resolution datasets for the use of other solvers that enabled shrinkage. This is because the other methods required inversion of the covariance matrix, which for high-dimensional data with many samples would take a very long time! For logistic regression, I ran the model again using different solvers and experimented with the use of different regularisation techniques to penalise the model. Further details of solvers and penalties can be found in the Results chapter.

Extensive hyperparameter tuning was not undertaken for the machine learning methods as they have a much higher number of model parameters, meaning that tuning would have been extremely time-consuming. Due to the constraints of the project length, this was not prioritised in my analysis.

4.3.8. Performance measures

To test the models, the *metrics* module from sklearn was used to test all of the models. When required, visual displays of model performance were created using the *pyplot* module from the package matplotlib v3.6.3 (Hunter, 2007). These were automatically created from the wrapper functions for each method. The following metrics were used for the analysis of model performance:

1. **Accuracy Score** The *accuracy_score* function was used to calculate the accuracy score. The accuracy score is a value between 0 and 1 that denotes the fraction of correct predictions.

   If M denotes the number of samples, y_i is the true label of the i-th sample of features and $\hat{y}_i$ is the predicted label of the i-th sample, then the accuracy is:

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{m=1}^{M} I(y_i = \hat{y}_i)$$

where $I(.)$ is the indicator function that takes the value 1 when the condition is true and 0 when the condition is false. In the case of accuracy:

$$I(y_i = \hat{y}_i) = \begin{cases} 1, y_i = \hat{y}_i \\ 0, y_i \neq \hat{y}_i \end{cases}$$

2. **F1 Score** The F1 score was derived using the *f1_score* function. The F1 score is the harmonic mean of precision and recall with values between 0 and 1 with 1 denoting the best score, a0 the worst.

   Precision can be defined as the proportion of accurately predicted labels (true positives) from all predicted labels, regardless of the correctness. Recall is the proportion of accurately predicted labels (true positives) from all correct labels, i.e. the completeness of positive predictions among the set of actual positives. They are calculated through the following formulae:

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$

   where TP denotes the number of true positives, FP denotes number of false positives and FN denotes number of false negatives.

   As F1 is the reciprocal of the arithmetic mean, the equation can be derived below:

$$\begin{aligned} F1 &= \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \\ &= \frac{2 \times \frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \\ &= \frac{2 \times TP^2}{(TP + FP)(TP + FN)} \times \frac{(TP + FP)(TP + FN)}{TP(TP + FN) + TP(TP + FP)} \\ &= \frac{2 \times TP}{2 \times TP + FP + FN} \end{aligned}$$

   This formula applies to binary classification problems only. To account for multi-class classification problems, there are different extensions to the formula:

   - Micro average - count of total true positives, false negatives and false positives.
   - Macro average - calculated by finding the unweighted mean of the F1 for each different class.
   - Weighted average - The same as macro average but accounting for class imbalances.

   I have chosen to use the Weighted F1 score for the datasets with greater than 2 classes, which can be found by adjusting the F1 score for different weights

$$\text{Weighted } F1 = \frac{\sum_{k=0}^{K-1} w_k \times F1_k}{\sum_{i=1}^{N} w_i}$$

where K is the total number of classes and w_k is the weight of the k-th class that depends on the number of true labels in each class.

3. **Log Loss** The log loss, also known as the logarithmic loss, is a non-negative metric that evaluates the probabilistic outputs of a classifier instead of the discrete predictions. Calculated in sklearn using the *log_loss* function, it can be used to evaluate the performance of classification models that contain two or more labels. The logarithmic loss, derived from taking the negative logarithm of the likelihood function, is a measure of the expected amount of surprise that the predictor experiences when encountering the known labels (Rezaei et al., 2020).

   It provides a metric to analyse how far away the prediction probability is from the true label, hence model predictions are found using the *predict_proba* call to the model instead of the *predict* call used for all other metrics in this paper. Furthermore, it adds a heavy penalty when the model predicts incorrectly with confidence. The lower the log loss value is, the better the model is as a classifier.

   For data with M samples that are labelled by K classes, let $p_{m,k}$ denote the predicted probability that sample $m \in [1, M]$ is located in class $k \in [0, K-1]$. To denote if a prediction is correct, let $y_{m,k}$ be an indicator function such that

$$y_{m,k} = \begin{cases} 1, & \text{if } k \text{ is the correct label for } m \\ 0, & \text{if } k \text{ is not the correct label for } m \end{cases}$$

   Hence log loss can be calculated using the following formula (Bishop, 2006)

$$\text{Log Loss} = L_{log}(y, p) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=0}^{K-1} y_{m,k} \log(p_{m,k})$$

4. **Prediction Score** This metric serves as a single score that acts as a generalisation of the accuracy and F1 scores and adjusts for log loss. The prediction score can be defined as:

$$\text{Prediction} = \frac{\text{Accuracy}(y, \hat{y}) + F1}{2 * L_{log}(y, p)}$$

   Scores lie in the range of values $[0, \infty)$, with a higher score denoting a better-performing model.

5. **Convergence** A model is said to converge when the loss function is minimised and further model iterations do not improve performance. Models that do not converge are considered to not have reached their full predictive potential, and often will not generalise well to new data. To accurately understand if a model has performed to its maximum capacity, I have measured convergence for all iterative methods. For cross-comparative purposes, I have set the maximum number of iterations (or epochs in the case of neural networks) to 500. Note, that there is no functionality to set a maximum iteration number in LDA as it is not iterative. To measure convergence, I have used each of the model's capabilities to return the number of iterations and epochs, and if this is greater than or equal to the maximum iterations then the code prints "N".

6. **Training Time** This measures the time of a method to run through model training and prediction. Training time was measured using the *datetime* function from the *datetime* module, using the *.now()* call. This was done by calling the function before model fitting and after the prediction end time, and the training time was derived by subtracting the start time from the end time.

4.3.9. Reproducibility

To ensure code and result reproducibility, seeds have been used throughout the programming. This has been done by allocating the number 28 to the **random_seed** variable in my master script, to ensure that all random number generators produce consistent values. My code inputs this value into the functions when there is support for a *random_state* parameter. When this is not available, the value is input into the *np.random.seed* and *random.seed* functions for all methods and the *torch.manual_seed*, *torch.Generator.manual_seed*, *torch.cuda.manual_seed* functions where applicable.

When a method runs in a multi-process environment, an additional *seed_worker* function has been created to seed the random number generators. To ensure that the behaviour of the machine learning models was deterministic and hence reproducible when utilising GPU, the following settings were applied: *torch.backends.cudnn.deterministic = True* and *torch.backends.cudnn.benchmark = False*.

5. Results and Discussion

5.1. Overview

This section will discuss the results from training, validation and testing of each of the methods discussed on the BreastMNIST, DermaMNIST and PathMNIST datasets of MedMNIST+.

Due to the extensive model and hyperparameter tuning and feature selection completed for LDA and Logistic Regression, the in-depth results of optimising each model on the validation dataset can be found in Appendix A. When selecting a model for these methods, all scores were analysed considering result validity due to the potential addition of randomness during feature selection. As discussed previously, feature selection was done by removing constant features and then selecting features with the highest percentage of mutual information scores. However, in the analysis, it appears that no constant features were present as no features were removed during this step across all three datasets.

A conservative approach has been taken for model selection, where feature selection has only been considered worthwhile if scores for all metrics improved. Although scores have been rounded to 3 decimal places, marginal differences in scores were further analysed in the original results due to their high precision.

In the case of logistic regression, when choosing the optimal combination of regularisation, feature selection and optimiser, run time was taken into consideration. If several top-performing models present almost equivalent scores, the model with the shortest run time was chosen due to its computational efficiency.

For the PathMNIST dataset, repeated attempts at computation failed for the methods of LDA and Logistic Regression for the colour 128×128 images and the grayscale 224×224 images. This could be a result of the high dimensional and high number of observations of the data, meaning that finding linear discriminants is a highly difficult task. Due to the time limitations of the project, an investigation into the reason for the failures was not able to be conducted in-depth.

For the machine learning methods, no manual hyperparameter tuning took place. Whilst the models tune themselves using the training and validation datasets during optimisation, in-depth training or validation metrics are not reported. Only the log loss values for both methods and accuracy for CNN were reported during training. As the log loss provides the best indication of model improvement, plots of the scores across the training and validation datasets can be found in the Appendix A. No matter the method of model testing, all performance metrics are calculated and reported. Due to the nature of the machine learning methods, the methods train and test in the same pro-

gram, and hence training metrics are found in the Testing section of Results. For further analysis of the results at the highest precision, please refer to `https://github.com/jackhodgkinson/statml_medmnistplus/tree/main/Results`

This section provides an analysis of the methods used. When hyperparameter tuning has been completed manually, a comparison of the full model (model with default parameters) to the chosen optimised model has been provided. Note that the best score for each metric across resolution and colour has been highlighted in bold, and the model that I believe performs best has been highlighted. There will also be a discussion of which model configuration is best for each data domain, and limitations of the analysis and models will be discussed.

5.2. Linear Discriminant Analysis

5.2.1. BreastMNIST

Table 5.1.: Results of Linear Discriminant Analysis on the BreastMNIST dataset at different resolutions

Res.	Colour	Model	Feat.%	Solver	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	Full/Best	100	SVD	Y	31.3s	0.720	0.796	9.240	0.082
64×64	Gray	Full/Best	100	SVD	Y	2m 7s	0.769	0.845	1.712	0.471
128×128	Gray	Full/Best	100	SVD	Y	3m 18s	**0.801**	**0.865**	0.868	0.960
224×224	Gray	Full/Best	100	SVD	Y	5m 17s	0.795	0.863	**0.661**	**1.254**

Analysing Table 5.1, the full model was chosen as the optimal LDA model at all resolutions for the BreastMNIST dataset. This is due to the random and large fluctuations in all metrics as features are removed, implying that the removal of features results in the model not fitting correctly to the data. The model could likely be over or underfitting when features are removed. More in-depth training results can be analysed in tables A.1 - A.4 and visualised in figures A.1 - A.4. Whilst one would expect a reduction in log loss when features are removed, in all resolutions except 224×224 the log loss increased when feature selection began. However, for the highest resolution, there was a reduction in accuracy and F1 when feature selection began.

The data shows that an increase in the resolution of the data leads to a reduction in the log loss. Whilst accuracy and F1 scores are the highest for the 128×128 resolution images, the log loss is the lowest for the 224×224. Furthermore, the discrepancy between accuracy and F1 scores between 128×128 and 224×224 is minimal, with less than 1 percentage point difference.

Overall, for the classification of the BreastMNIST dataset using linear discriminant analysis, using the highest resolution images possible without removing any

features produces the best-performing model. The model achieves a classification accuracy of 79.5%, which is 30% higher than random guessing, and an F1 score of 86.3%. This implies the model has a good balance between making accurate class predictions and capturing all relevant positive instances. The log loss value of 0.661 indicates that the model's predictions are reasonably accurate but are not fully certain.

5.2.2. DermaMNIST

Table 5.2.: Results of Linear Discriminant Analysis on the DermaMNIST dataset at different resolutions

Res.	Colour	Model	Feat.%	Solver	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	Full/Best	100	SVD	Y	3m 50s	0.627	0.580	**1.669**	**0.364**
	RGB	Full/Best	100	SVD	Y	19m 25s	0.604	0.600	2.761	0.218
64×64	Gray	Full	100	SVD	Y	41m 55s	0.490	0.514	8.579	0.059
		Best	90	SVD	Y	26m 18s	0.510	0.527	6.909	0.075
	RGB	Full/Best	100	SVD	Y	1h 20m 8s	0.540	0.553	10.570	0.052
128×128	Gray	Full/Best	100	SVD	Y	1h 11m 1s	0.545	0.541	8.539	0.064
	RGB	Full	100	SVD	Y	1h 44m 54s	0.625	0.604	5.172	0.119
		Best	90	SVD	Y	1h 43m 58s	0.626	0.605	5.284	0.117
224×224	Gray	Full/Best	100	SVD	Y	2h 26m 2s	0.606	0.576	5.322	0.111
	RGB	Full/Best	100	SVD	Y	4h 22m 47s	**0.645**	**0.614**	3.788	0.166

Table 5.2 provides insight into the results of classification on the DermaMNIST dataset using linear discriminant analysis on images both in original colour and post-transformation to grayscale.

As in the case of BreastMNIST, the accuracy and F1 scores are generally the best with the highest-resolution images. However, there are drops in these scores for the 64×64 resolution images which could be indicate anomalous results. This resolution also has the highest log loss value, hence indicating that the classifier is performing very poorly.

In all cases except grayscale 64×64 images and colour 128×128 images, the full model provided the best separation of data. In the exceptions, the top 90% of features were kept, with the features that scored in the lowest 10% for mutual information removed. In the case of the grayscale 64×64, the accuracy and F1 scores only improved by approximately 2 percentage points whilst log loss had a large reduction. On the contrary, for the colour 128×128 images, the removal of features damaged the performance of the LDA model. This suggests that at a higher resolution, crucial features that were removed may not have been as influential for the classification in a lower dimension.

An interesting observation for the DermaMNIST dataset is that the log loss and prediction score were optimal for the grayscale 28×28 image data. This is surprising considering that the images were transformed from colour. This implies that for linear discriminant analysis, the added features from the inclusion of colour may not be necessary to accurately classify dermatoscopic images.

In conclusion, the model that I believe is optimal for multi-class classification of the DermaMNIST dataset into the seven provided categories is by using the lowest resolution images possible 28×28 transformed to grayscale. The accuracy score of 62.7% is 48% higher than random guessing, which is a considerable improvement. The F1 score of 58.5% implies the model has a moderate balance between accurate predictions and avoiding false negatives. The log loss value of 1.669 implies that there is a degree of uncertainty in the model's prediction, meaning that it may not generalise well to unseen data.

5.2.3. PathMNIST

Table 5.3.: Results of Linear Discriminant Analysis on the PathMNIST dataset at different resolutions

Res.	Colour	Model	Feat.%	Solver	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	Full/Best	100	SVD	Y	2m 24s	0.414	0.371	1.644	0.239
	RGB	Full	100	SVD	Y	13m 31s	0.575	0.561	1.191	0.477
		Best	90	SVD	Y	11m 20s	**0.579**	**0.563**	**1.185**	**0.482**
64×64	Gray	Full/Best	100	SVD	Y	57m 2s	0.401	0.374	1.784	0.217
	RGB	Full	100	SVD	Y	4h 10m 42s	0.496	0.485	1.776	0.276
		Best	80	SVD	Y	2h 9m 24s	0.512	0.501	1.605	0.316
128×128	Gray	Full/Best	100	SVD	Y	5h 38m 29s	0.356	0.336	2.560	0.135

The results from running linear discriminant analysis on the PathMNIST dataset follow a different trend than the other datasets. However, the failure of LDA for the grayscale 224×224 images and colour 128×128 images means that an accurate analysis of the performance at the highest resolution is impossible.

The scores in table 5.3 are highest for all metrics when the data is in the lowest resolution, in the original colour and feature selection of the features corresponding to the highest 90% of mutual information scores is conducted. Note that there is only a slight improvement with removing the lowest 10% of mutual information scores, with accuracy and F1 score improving by under 0.5 percentage points.

An interesting observation is the large difference in scores and log loss between the grayscale images and the full-colour images, with full-colour images outperforming the grayscale ones. Considering that the samples are from H&E stained

tissue, this result is expected as the method of staining highlights crucial features in the images. Furthermore, the scores regarding the grayscale images seem to steadily deteriorate with increasing resolution. This further implies that in the PathMNIST dataset, colour is an important feature for class separation.

The scores are lower for the higher-resolution data when in colour too. But on the 64×64 data, the scores all improve when the features related to the lowest 20% of mutual information scores are removed. This suggests that when the data is in a higher dimension, many of the features are redundant and may confuse the class separation, leading to less accurate predictions.

An important observation is that in all cases when resolution increases, the log loss also increases. In theory, the model struggles to differentiate the classes for higher resolution images more than lower resolution images due to the increase in the features of the data. This could be a result of increased data complexity or the addition of noise to the samples.

Considering all points, the optimal LDA model for the PathMNIST dataset is on the 28×28 resolution images with the features related to the lowest 10% of mutual information scores removed. This achieved an accuracy of 57.9% which is an improvement of approximately 46.8% from random guessing, suggesting that LDA as a classifier is much superior to random selection. The F1 score of 56.3% suggests that the model achieves a reasonable balance between precision and recall, meaning the model has reasonable accuracy in detecting false positives and negatives. The log loss score of 1.185 implies that the predicted probabilities of the model are relatively close to the actual class probabilities, however, there is again room for improvement.

5.2.4. Model Summary

Figure 5.1.: Summary of performance of Linear Discriminant Analysis across all datasets, resolutions and metrics with varying colour

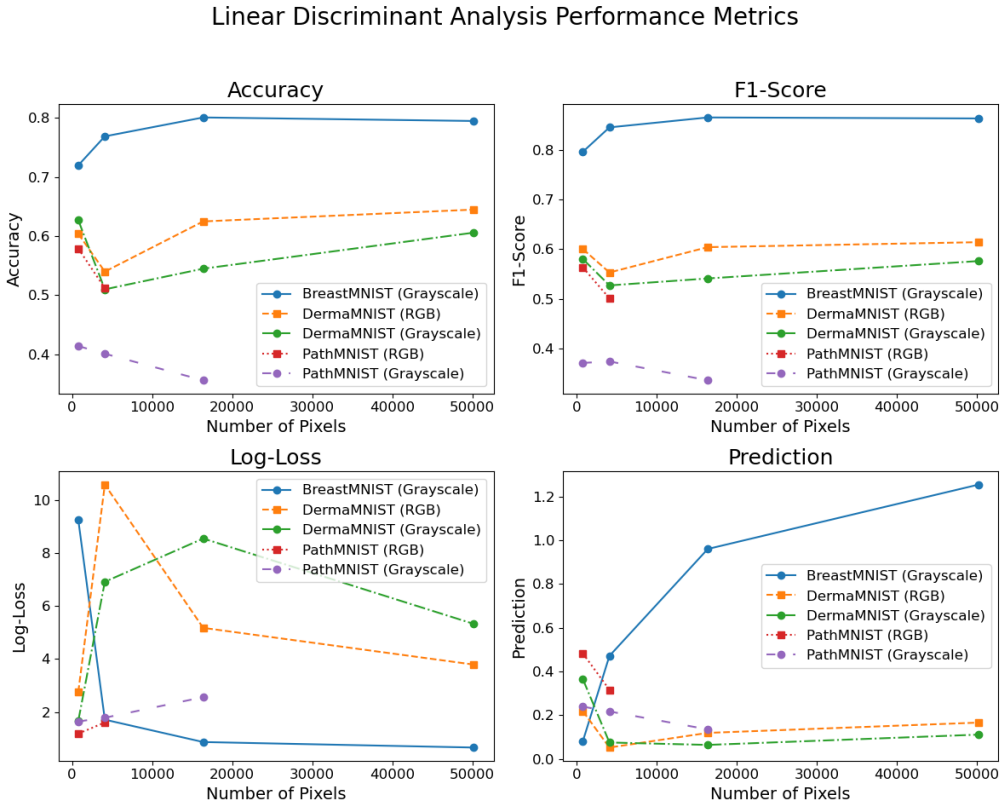


Figure 5.1 summarises the findings from the classification of the selected MedMNIST datasets using linear discriminant analysis. The data suggests that for the binary classification problem of BreastMNIST, all scores improve considerably when the image resolution is increased. As ultrasound images are grayscale, low-resolution images prove harder to detect malignant tumours in breasts. Increasing image resolution proves to considerably increase the predictive accuracy and F1 scores whilst lowering the log loss, implying that the LDA model generalises best to unseen data on high-resolution samples. Higher-resolution images allow for finer details and subtle abnormalities to be detected more easily and smaller structures to be more visible. This likely means that the detection of a malignant tumour is easier in higher dimensional data, so the LDA model provides a more accurate prediction.

For the multi-class classification tasks, the plots show that in general, accuracy is higher for the full-colour images than the grayscale images. Similarly, the F1 score is also higher for the colour images. This suggests that when colour is available in medical imaging, it should be utilised and not transformed to grayscale as important information is provided by colour. Log loss on the other hand seems to have a more stochastic nature in multi-class classification, and also evidences much higher values in comparison to binary classification when resolution increases.

For the DermaMNIST dataset, the accuracy is marginally higher in the grayscale

images for the lowest resolution and then drops below the full-colour images with increasing resolution. The F1 score has higher scores in the colour images as opposed to the grayscale images. Regarding log loss, the grayscale images have a lower log loss for the 28×28 and 64×64 images than the colour images but then for the higher resolutions colour images generate a lower log loss value. However, the log loss for the DermaMNIST dataset rapidly increases when the resolution is increased, meaning that the model is less confident in its predictions on higher-resolution images. Log loss does become lower in the colour images for the two highest resolutions, suggesting that when there are more features, colour plays an important role in dermatoscopic imaging.

Whilst the PathMNIST dataset has few observations due to computational failure, the accuracy and F1 scores are all higher for the coloured images as opposed to the grayscale images, and both decrease linearly for an increase in resolution. Furthermore, the log loss is also lower in the coloured images, however, for both colour and grayscale the log loss increases as resolution increases. This suggests that the classification of the PathMNIST dataset using LDA is optimal when the original colour is kept and a low-resolution tissue sample is used. The model's performance was hindered by limitations in the capabilities of the server, but it is conceivable that the model could exhibit more robust and efficient performance when deployed on more advanced systems or hardware configurations.

5.3. Logistic Regression

5.3.1. BreastMNIST

Table 5.4.: Results of Logistic Regression on the BreastMNIST dataset at different resolutions

Res.	Colour	Model	Feat.%	Reg.	Solver	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	Full	100	L2	LBFGS	N	6.6s	0.756	0.833	2.563	0.310
		Best	100	L1	SAGA	N	4.2s	0.776	0.851	**0.543**	1.497
64×64	Gray	Full	100	L2	LBFGS	N	8.6s	0.782	0.851	1.866	0.437
		Best	100	L1	SAGA	N	21.3s	0.801	0.869	0.552	1.513
128×128	Gray	Full	100	L2	LBFGS	Y	15.9s	0.795	0.861	1.861	0.445
		Best	100	L1	SAGA	N	1m 20s	**0.827**	**0.886**	0.562	**1.525**
224×224	Gray	Full	100	L2	LBFGS	Y	31.6s	0.801	0.866	1.937	0.430
		Best	95	L2	SAGA	N	2m 5s	0.821	0.881	0.560	1.520

As seen in Table 5.4, the introduction of LASSO regularisation to the model seemed to provide the most promising results at all resolutions except the $224x224$ resolution images, where Ridge regression outperformed LASSO. However, the best performing method of optimisation among all resolutions in the BreastMNIST dataset was the SAGA algorithm, implying that binary classification of medical

images benefits from using the SAGA algorithm for optimising the parameters of Logistic Regression.

Although LASSO regularisation has the potential to automatically remove features, no features were removed at any resolution on this dataset. For the 224×224 dataset, ridge regression performed the best when 95% of the features were kept, with the lowest 5% of mutual information scores being removed from the data before classification. This could be due to the large number of features for this resolution, 50176 to be exact, meaning that 2509 features were removed that probably added little to nothing to the image information. The features could have simply just been noise.

Although all logistic regression models were tested with a maximum iteration number of 500, only the full model converged for the images with resolution 128×128 and 224×224. This suggests that logistic regression could potentially operate better for binary classification tasks with image data with a high number of features. Also, a much higher number of iterations are likely required for convergence, meaning that the scores found likely indicate the performance of the models not at their highest capabilities.

The best models managed to increase predictive accuracy by at least 2%, implying that hyperparameter tuning improves the classification models. F1 score was increased by at least 1.5%, meaning that the optimised models reduce the probability of Type I and II errors in classification, leading to more trustworthy models. The biggest improvement was in the log loss, where hyperparameter tuning significantly reduced the log loss. The log loss was the smallest for the 28×28 resolution data. Whilst the log loss scores are not drastically different across the different resolutions, there is quite a large variation across accuracy and F1 scores at each resolution.

Taking the metrics into context, the optimal image resolution for binary image classification on the BreastMNIST data using logistic regression is the 128×128 resolution. Images of this resolution marginally display the best prediction score, as the high accuracy and F1 scores are penalised by an increase in the log loss. The model accurately classified 82.7% of samples and has 88.6% accuracy in avoiding false positives. Whilst the log loss was not the lowest at this resolution, it is only marginally worse than the other resolutions. A log loss score of 0.562 implies that the model's probability estimates are reasonably accurate, however there is further room for improvement.

5.3.2. DermaMNIST

Table 5.5.: Results of Logistic Regression on the DermaMNIST dataset at different resolutions

Res.	Colour	Model	Feat.%	Reg.	Solver	Conv.	Time	Acc.	F1	Log Loss	Pred.
28 × 28	Gray	Full	100	L2	LBFGS	N	15.3s	0.658	0.586	1.193	0.521
		Best	100	L1	SAGA	N	1m 14s	0.669	0.580	1.041	0.600
	RGB	Full	100	L2	LBFGS	N	35.5s	0.675	0.635	1.191	0.550
		Best	95	L2	SAGA	N	2m 25s	0.690	0.636	**0.880**	0.754
64 × 64	Gray	Full	100	L2	LBFGS	N	59.1s	0.623	0.594	2.300	0.265
		Best	100	L1	SAGA	N	8m 38s	0.670	0.585	1.037	0.605
	RGB	Full	100	L2	LBFGS	N	2m 35s	0.650	0.631	3.420	0.187
		Best	100	L2	SAGA	N	8m 47s	0.690	0.643	0.885	0.754
128 × 128	Gray	Full	100	L2	LBFGS	N	4m 0s	0.620	0.592	3.272	0.185
		Best	100	L2	SAGA	N	13m 20s	0.674	0.593	1.037	0.611
	RGB	Full	100	L2	LBFGS	N	11m 12s	0.656	0.630	3.471	0.185
		Best	100	L1	SAGA	N	1h 7m 14s	**0.694**	**0.647**	0.885	**0.758**
224 × 224	Gray	Full	100	L2	LBFGS	N	12m 26s	0.621	0.592	4.591	0.132
		Best	100	L2	SAGA	N	51m 34s	0.676	0.593	1.034	0.614
	RGB	Full	100	L2	LBFGS	N	33m 9s	0.659	0.629	6.079	0.106
		Best	100	L1	SAGA	N	3h 23m 48s	**0.694**	**0.647**	0.884	**0.758**

Table 5.5 provides insight into the performance of a logistic regression model for classifying the DermaMNIST data into the seven provided classes. Across the board, accuracy declined when the original coloured data was transformed into grayscale when comparing the full models at each resolution. The same pattern emerges across the best-performing models at each resolution, however, the comparison here must be carefully considered as the best-performing models differ between resolutions and colours. This implies that colour is an important property in dermatoscopy and that transforming the data into a grayscale format hinders classification accuracy. As there is a large cost associated with misclassification, I would suggest that coloured samples be kept in their original colour.

The SAGA optimisation algorithm proves to be the most successful across all cases, with improvements being made from the full model across all metrics. Whilst this algorithm has a much longer training time, the benefit of the improvement in accuracy and F1 score significant decrease in log loss outweighs the time extension.

In only one case, the grayscale 28×28 sample, does a score decrease when the best model is used. The F1 score decreases by 0.6 percentage points. Whilst this is a very minimal decrease and likely does not have a huge effect overall, it is an important point to highlight.

In the best models, there is an even split of which method of regularisation was used. Whilst LASSO regularisation automatically removes redundant features, none were removed in any case. The only time that manual feature selection was conducted is for the best model in the original colour 28×28, where the features related to the lowest 5% of mutual information scores were removed.

Regarding convergence, none of the models converged to the minimum log loss with 500 iterations. This provides further evidence that the amount of iterations for optimising logistic regression needs to be much higher and scaled according to the number of samples, features and classes in the problem.

Considering the optimal situation for the logistic regression model, choosing the best model was difficult as the performance of the best models on coloured data is not vastly different across the different resolutions. The accuracy scores only differ by a range of 0.4 percentage points, the F1 scores by 1.1 percentage points and the log loss by a value of 0.5. Due to the minimal differences, in selection, I considered the run times.

Taking everything into account, the optimal model for the classification of the DermaMNIST dataset is the best model on the 28×28 resolution data in original colour. This uses L2/Ridge regularisation and optimises using the SAGA algorithm, with 95% of the features selected. This model achieves a prediction accuracy of 69%, and an F1 score of 63.6%, meaning that the model adequately predicts correctly and identifies the true positives. The log loss of 0.880 is the lowest among the logistic regression models tested on this dataset, meaning that the model's predicted probabilities are relatively close to the actual class labels, however, there is still room for improvement. The run time is relatively short at 2 minutes and 25 seconds for the model to test the testing set, which is much shorter than the other models with similar performance.

5.3.3. PathMNIST

Table 5.6.: Results of Logistic Regression on the PathMNIST dataset at different resolutions

Res.	Colour	Model	Feat.%	Reg.	Solver	Conv.	Time	Acc.	F1	Log Loss	Pred.
28 × 28	Gray	Full/ Best	100	L2	LBFGS	N	2m 47s	0.159	0.137	2.200	0.067
	RGB	Full	100	L2	LBFGS	N	9m 4s	0.416	0.382	1.717	0.232
		Best	100	L1	SAGA	N	1h 8m 47s	**0.422**	**0.386**	**1.701**	**0.237**
64 × 64	Gray	Full	100	L2	LBFGS	N	15m 38s	0.134	0.128	2.321	0.056
		Best	100	L2	SAGA	N	54m 16s	0.135	0.127	2.310	0.057
	RGB	Full	100	L2	LBFGS	N	49m 43s	0.393	0.373	1.883	0.203
		Best	80	L2	SAGA	N	2h 4m 17s	0.408	0.383	1.806	0.219
128 × 128	Gray	Full	100	L2	LBFGS	N	58m 56s	0.135	0.137	2.660	0.051
		Best	95	L2	SAGA	N	3h 12m 37s	0.134	0.130	2.471	0.053

Table 5.6 displays the results from model tuning a logistic regression model for the different colours and resolutions of PathMNIST. Note that computation failed for the grayscale 224×224 and the colour 128×128 and 224×224 resolutions of data. This could be a result of the large number of features at higher resolutions and the very large sample size of PathMNIST. Because of this, the analysis of logistic regression on the PathMNIST dataset will be limited.

The first observation from the results is that the coloured data resulted in better performance than the grayscale images. The grayscale images displayed extremely low accuracy and F1 scores, and a high log loss value. The accuracy scores are only slightly better than random guessing, meaning that logistic regression on grayscale data for the PathMNIST dataset provides little improvement to the random allocation of classes. The coloured image samples still display low accuracy and F1 scores, but they are significantly better than the grayscale data. The coloured data predicts samples with an accuracy 28% higher than random guessing.

For the samples in original colour, accuracy and F1 had an improvement when the best model was used as opposed to the full model. Whilst improvements were limited, the metrics did increase implying that hyperparameter tuning is beneficial for logistic regression. Interestingly, in some cases, the accuracy and F1 scores decreased when using the best model for the grayscale data. In all models, when selecting the best model, the log loss value decreased.

For the grayscale 28×28 data, the full model performed the best. This is unusual, as in all other examples of logistic regression, the full model has not ranked as superior. One reason for this could be that the tuned models were too complex for the data. At the low resolution of this data, grayscale samples are difficult to classify due to important information being lost. The SAGA algorithm proved

triumphant as the highest-scoring optimisation algorithm in all cases except from the grayscale 28×28 images.

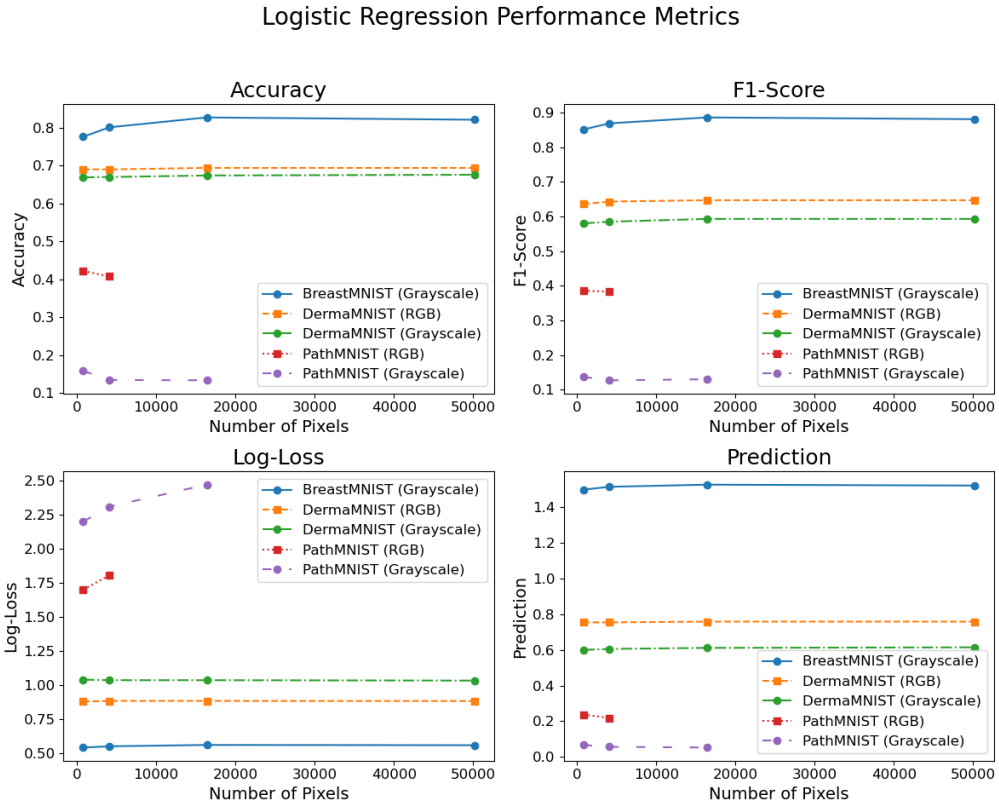
The method of ridge regularisation was the clear winner for the PathMNIST dataset, being the choice of regularisation for the best models in all cases except one case, where LASSO was used. Feature selection was only used for the 64×64 resolution coloured images, where the lowest scoring 20% of features were removed and for the 128×128 resolution grayscale images, where the lowest scoring 5% of features were removed. One could infer from this that in the PathMNIST dataset when more features are present in the data, fewer are required to make the optimal accurate predictions.

None of the models converged within the 500 maximum iterations. This suggests that the images from PathMNIST require a higher number of iterations for the model to converge due to the large number of features and samples. As a result of non-convergence, likely, the models are not performing to their full predictive potential, hence the scores would potentially improve with more iterations.

Analysing the results, the 28×28 resolution images in original colour resulted in the best performance across all metrics when LASSO regularisation was employed as part of the SAGA optimisation algorithm. Despite the relatively longer runtime of 1 hour, 8 minutes, and 47 seconds, the higher accuracy achieved in prediction makes the computational investment worthwhile. The precision gained enhances the reliability of the predictions, which in clinical settings would improve patient outcomes through more accurate diagnoses and treatment decisions. The model achieves a predictive accuracy of 42.2%, 31% higher than random guessing, and an F1 score of 38.6%. Given the data domain, the scores do not provide confidence in the method of logistic regression for the classification of human colorectal cancer. Furthermore, the log loss of 1.701 suggests that there is considerable divergence of the model's predicted probabilities from the true class labels, implying that a logistic regression model will likely not generalise well to unseen data in this domain.

5.3.4. Model Summary

Figure 5.2.: Summary of performance of Logistic Regression across all datasets, resolutions and metrics with varying colour.



The data from figure 5.2 suggests that for the binary classification problem associated with BreastMNIST, the accuracy and F1 score on the best-performing logistic regression model improves as the image resolution increases. Whilst there is a gradual improvement as resolution increases, moving from the 128×128 resolution to the 224×224 appears to have minimal effect on both accuracy and log loss, with either no improvement or a very small decrease. The accuracy and F1 score are also much higher for the BreastMNIST dataset than the other datasets, which is to be expected due to the problem being a binary classification task. The log loss appears to deteriorate ever so slightly when resolution is increased, however, the change is very minimal and likely does not have a huge effect on the overall model performance. In summary, the predictive capabilities of higher-resolution images seem to improve in the task of binary classification using logistic regression, which can be seen by the increase in the prediction score.

For the multi-class classification problems of DermaMNIST and PathMNIST, there appears to be an improvement in all metrics when original colour images are used as opposed to images transformed to grayscale. This suggests that for both dermatoscopic classification tasks and colorectal cancer detection using H&E stained images, colour is an important feature not to be discarded in logistic regression. When resolution is increased, the effect on the metrics is dependent on the specific dataset. This shows the importance of considering the data domain

before deciding on models and image resolution.

The DermaMNIST dataset seems to score similarly across all resolutions, however, with higher scores on colour images. There appears to be a slight improvement in F1 scores as the number of pixels increases, meaning that with a higher number of features, the model identifies false positives with more accuracy. In all other metrics, the scores appear to remain relatively constant. From this, it could be concluded that for the DermaMNIST dataset, classification using logistic regression does not depend on the number of features. Whilst the highest resolution images perform the best, this is only a very small improvement from the lowest resolution image. The smaller training and testing time of the lowest resolution images outweighs the minuscule performance enhancement. The extra time could potentially be spent on verification of the predictions by a health professional.

However, the results from the classification task on the PathMNIST data do not follow the above pattern. There is an obvious decrease in all metrics when the resolution is increased, and the coloured images perform drastically better than the grayscale images. By considering the purpose of H&E staining in colorectal cancer detection, these results are logical. However, the analysis of this dataset is limited due to the failure of computation for the logistic regression model on the higher-resolution images. I encourage further investigation into the use of logistic regression on the PathMNIST dataset in the highest resolutions to diagnose this computational issue.

5.4. Convolutional Neural Network

5.4.1. BreastMNIST

Table 5.7.: Results of classification using a Convolutional Neural Network on the BreastMNIST dataset at different resolutions

Resolution	Colour	Num. of Epochs	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	9	Y	29.9s	**0.878**	**0.920**	0.657	1.369
64×64	Gray	9	Y	31.4s	0.872	0.912	0.638	1.398
128×128	Gray	10	Y	32.8s	0.821	0.886	0.964	0.885
224×224	Gray	6	Y	39.5s	0.833	0.887	**0.569**	**1.512**

The results from Table 5.7 display varying patterns. For the 128×128 resolution images, there is a sharp drop in accuracy and F1 score from the 64×64 resolution data, and an increase in log loss. Whilst the log loss improves when increasing resolution from 28×28 to 64×64, there is a minor reduction in the accuracy and F1 score, suggesting that the neural network's performance is decreasing proportionally to an increase in resolution. The log loss is the smallest for the highest resolution data, with the largest being for the second highest resolution data. As

all of the metrics do not follow the observed patterns at the 128×128 resolution, it could be assumed that the results are anomalous and further investigation should be completed.

Further to this, the accuracy and F1 scores do not seem to drastically improve when adjusting resolution from 128×128 to 224×224. Whilst log loss does drop significantly, the accuracy and F1 scores only marginally increase by 1.2 and 0.1 percentage points respectively.

At all resolutions, the models converged quickly in less than 10 epochs. An interesting observation is that the number of epochs for the highest resolution data was the lowest, which personally is surprising considering that the data traverses through the largest number of hidden layers at the highest resolution.

Unlike the statistical methods, the run time does not appear to have a dramatic increase with higher-resolution data. This could be a result of CNN being run using GPUs or the efficiency of the Adam optimiser.

The decision on which model is optimal needs to be considered carefully in the context of the data. As the BreastMNIST dataset is used to classify malignant breast tumours, I would say that confidence estimation is extremely important. Due to the high cost associated with misclassification, I would personally select the model with the lowest log loss, the CNN on the 224×224. This achieved a predictive accuracy of 83.3% and correctly distinguished between true and false positives 88.7% of the time. The log loss score of 0.569 suggests that the probability estimates produced by the model are relatively accurate, but the model could improve. One way to potentially achieve this would be with the removal of early stopping.

5.4.2. DermaMNIST

Table 5.8.: Results of classification using a Convolutional Neural Network on the DermaMNIST dataset at different resolutions and colours

Resolution	Colour	Num. of Epochs	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	2	Y	1m 44s	0.664	0.578	1.405	0.442
	RGB	7	Y	1m 5s	0.728	0.699	0.886	0.806
64×64	Gray	6	Y	3m 2s	0.701	0.637	1.171	0.572
	RGB	14	Y	5m 43s	0.710	0.667	1.267	0.543
128×128	Gray	10	Y	7m 34s	0.711	0.648	1.295	0.525
	RGB	17	Y	46m 57s	0.726	0.693	0.979	0.725
224×224	Gray	3	Y	32m 18s	0.409	0.474	1.578	0.280
	RGB	11	Y	1h 19m 59s	**0.732**	**0.705**	**0.829**	**0.866**

Table 5.8 shows that on the DermaMNIST dataset, all metrics produce better scores for the coloured images than the grayscale images. This is analogous to the methods of linear discriminant analysis and logistic regression, further highlighting the importance of colour for detecting cancerous lesions on images of human skin. The results do appear to be fairly consistent across all resolutions, with some exceptions. However, the increase in resolutions doesn't have a linear relationship with the improvement in scores. Instead, the lowest and highest resolution images perform the best, with the 64×64 images performing worst for the colour images and 128×128 for the grayscale images. The lowest-resolution images may capture the fundamental details or patterns required for classification and the highest-resolution images likely provide the model with the most information. The middle-resolution images potentially do not provide the necessary balance required between clarity and simplicity for effective classification. This emphasises that added data complexity doesn't always provide useful details to improve model performance.

The data implies that there could be a trend relating the number of epochs required and resolution, as an increase in resolution appears to require more epochs. More epochs are required for the coloured images rather than the grayscale data. This is logical as higher resolution and coloured data samples possess more features, hence the model has to learn more information. However, it is interesting that the number of epochs for the 224×224 resolution datasets is the second lowest among the resolutions. Furthermore, the 224×224 resolution data for the images in original colour performs the best across all metrics. The grayscale images do not follow the same pattern, with a large reduction in the accuracy and f1 scores and an increase in log loss when the resolution increases.

The coloured data has similar accuracy and F1 scores across the different resolutions, with a maximum difference of 2.2 and 3.8 percentage points respectively. The log loss values seem to have more variability. In the case of the coloured data, the lowest and highest resolution images perform the best across all metrics, with the scores plateauing for the 64×64 and 128×128 resolution images. Whilst one may think an increase in resolution leads to better classifications due to the added wealth of information, sometimes this can introduce noise to the model and so the detail is overshadowed. This could be a potential explanation for the dip in scores.

The grayscale images provide a more linear improvement in accuracy and F1 scores, with both metrics improving up to the 128×128 resolution images. A similar pattern emerged for the log loss, but log loss increased for the 128×128 grayscale data. In all metrics, the scores deteriorate rapidly for the 224×224 grayscale images, with accuracy and F1 scores dropping over 20%. However, the log loss value is not vastly greater than the 28×28 grayscale images, meaning that although the predictions and identification of false positives in the model rapidly declined, the prediction of probabilities did not decline drastically. As the coloured data outperforms the grayscale data at all resolutions in each metric, I would advise against transforming the DermaMNIST data to grayscale.

When choosing the optimal combination of resolution and colour for the prediction

of the DermaMNIST dataset in a convolutional neural network, the 224×224 resolution images in original colour have the optimal scores across all metrics. Whilst, in theory, this would be the logical selection, the model training and testing time of 1 hour 19 minutes and 59 seconds is rather high. Comparing this to the 28×28 resolution data, the accuracy and F1 scores are only marginally less but the model run time is only 1 minute and 5 seconds. However, the moderate improvement in log loss sways my vote towards the 224×224 data as a lower log loss often indicates better generalisability to unseen data. As making correct predictions in the data domain is crucial, I believe the added security of a model that generalises well to new data outweighs the short training time of a model on lower-resolution images. I believe that patients and clinicians would value a more accurate and confident predicting model that takes longer to produce results than quick results that are not as accurate. This is especially true for the case of potentially cancerous skin lesions, as the cost of making a wrong classification on the outcome of the patient is high.

Therefore for the DermaMNIST dataset, the optimal resolution and colour to use for the highest chance of accurate classification is the 224×224 resolution images in the original colour. This achieves a predictive accuracy of 73.2% and an F1 score of 70.5%, which indicates quite a good balance between precision and recall. The log loss score of 0.829 implies that the predicted probabilities of the model are relatively close to the true class labels. As this model has a small number of epochs, I believe that the removal of early stopping would further improve the scores and ensure the model has more confidence in its predictions of unseen data.

5.4.3. PathMNIST

Table 5.9.: Results of classification using a Convolutional Neural Network on the PathMNIST dataset at different resolutions and colours

Resolution	Colour	Num. of Epochs	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	6	Y	30m 29s	0.747	0.756	1.045	0.718
	RGB	1	Y	15m 14s	0.430	0.408	3.813	0.110
64×64	Gray	6	Y	48m 55s	0.716	0.717	1.442	0.497
	RGB	11	Y	58m 13s	0.786	0.763	1.567	0.494
128×128	Gray	17	Y	1h 40m 20s	0.857	0.860	**0.728**	1.179
	RGB	17	Y	2h 22m 14s	0.838	0.837	1.047	0.800
224×224	Gray	15	Y	2h 15m 56s	0.763	0.770	1.299	0.590
	RGB	35	Y	4h 36m 10s	**0.882**	**0.884**	0.739	**1.195**

The results of a convolutional neural network on the PathMNIST dataset can be seen in table 5.9. Interestingly, there is no clear winner between colour and grayscale images. Whilst colour provides better scores for the 64×64 and 224×224 resolution images, the grayscale image performs better on the 28×28 and 128×128 resolution images.

The number of epochs required for the model to converge again appears to increase when the image resolution increases. Also, the number of epochs required is higher for the coloured data than the grayscale data. However, in the case of the coloured 28×28 samples, the number of epochs required for convergence is lower than in the grayscale images. This result does not seem legitimate, as a singular epoch is unlikely to converge to the model's optimal capabilities. This is further evidence of the limitations of early stopping, as I believe that extra epochs on the 28×28 colour images would have drastically improved the scores. All metrics for this data are significantly worse than the other cases, suggesting that at this resolution crucial information for classification could potentially be lost in the operations completed in the convolutional layers.

For the colour images, all metrics appear to improve as resolution increases. Whilst there is a significant improvement moving from the 28×28 resolution images to the 64×64 resolution images, the improvement becomes more gradual with further resolution increases. The same trend does not apply to the grayscale images, with the scores deteriorating when moving from 28×28 to 64×64, but improving significantly when increasing resolution to 128×128. The metrics for the grayscale 128×128 resolution images are a close second for the best in the PathMNIST task using a CNN, with this data achieving the lowest log loss. This is the first time across the datasets and methods that the grayscale images have performed exceptionally well across all metrics. From this, one could infer that for the PathMNIST dataset, colour is not a critical feature of the image and the structures depicted in the image are more important.

When assessing all metrics, the optimal data structure for the classification of the PathMNIST dataset using a convolutional neural network emerges as the 224×224 resolution images in original colour. This achieved an extremely high accuracy score of 88.2%, which considering this dataset has a very large number of samples and is classified into 9 classes is extremely impressive. The F1 score of 88.4% suggests that the model balances precision and recall very well, meaning the number of false positives is low. Whilst the log loss is not the best across the colours and resolutions, it is only a value of 0.011 worse than the lowest log loss score which is minimal. The log loss value of 0.739 suggests that the model's predicted probabilities are reasonably close to the true labels, hence the model is well generalised to unseen data. Although the run time is high at 4 hours 36 minutes and 10 seconds, when considering patient outcomes a model that takes longer to run is insignificant compared to its impressive classification results.

5.4.4. Model Summary

Figure 5.3.: Summary of performance of the Convolutional Neural Network across all datasets, resolutions and metrics with varying colour.

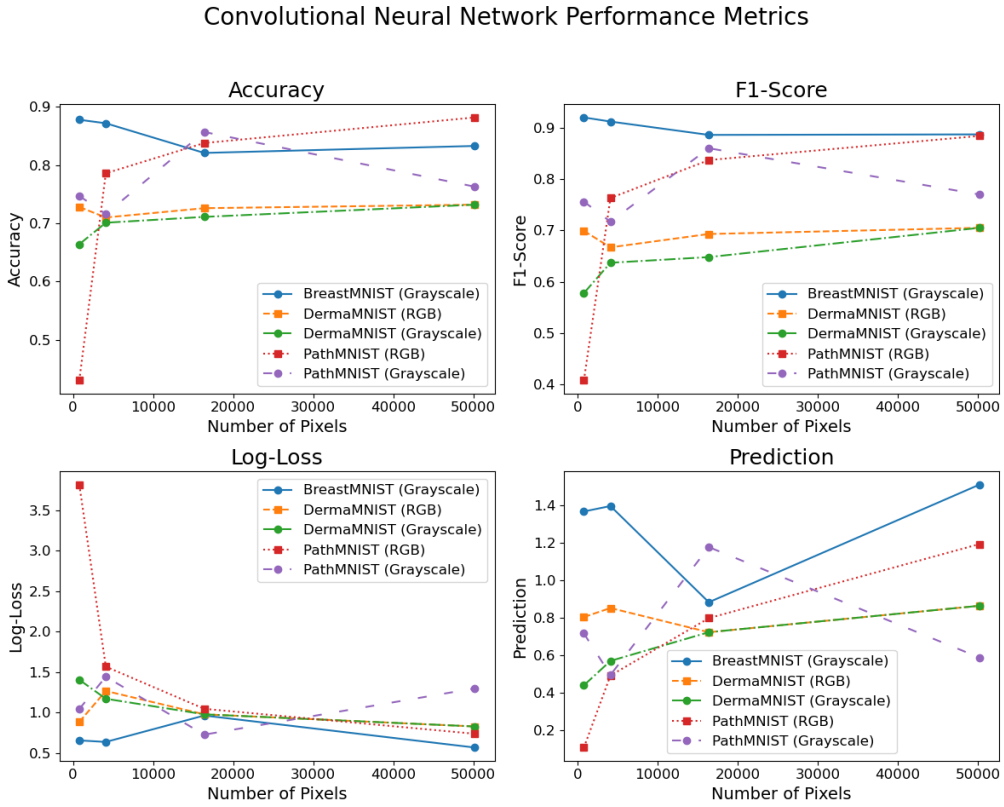


Figure 5.3 indicates the performance of convolutional neural networks across varying classification problems on different data domains at increasing resolution and both colour and grayscale images.

The binary classification task on the BreastMNIST dataset appears to have the best accuracy and F1 score when the resolution is the lowest and slowly decreases as the resolution is increased. The log loss however appears to improve when resolution is increased from 28×28 to 64×64 and then deteriorates for the 128×128 data. The log loss reaches its minimum for the 224×224 resolution images. Whilst the accuracy and F1 scores are not the highest for the high-resolution data, the graph displays that there is minimal difference between scores as resolution increases. As the log loss is lowest for the highest resolution and there is not a huge discrepancy in the accuracy scores, the prediction score is therefore maximised for the high-resolution images. This implies that for a binary classification problem on grayscale ultrasound images, higher resolution results in a more accurate prediction in a CNN. However as I have only tested this on the BreastMNIST dataset, the result cannot be generalised to any other data modalities than breast ultrasounds.

In the multi-class classification tasks, it appears that in general, the colour images display better performance than the grayscale images. There are some exceptions

to this though, with the grayscale 28×28 and 128×128 resolution images of PathMNIST outperforming their coloured image counterparts. Furthermore, the log loss for DermaMNIST is lower for the grayscale data at the 64×64 resolution. Whilst this could indicate that at certain resolutions transformation to grayscale will result in better classifications, this could be due to the inherent characteristics of the samples. For example, certain features relevant to classification may appear more pronounced in grayscale images at specific resolutions. The effectiveness of colour information in classification tasks can vary depending on the nature of the data and the complexity of the classes being distinguished. Therefore, while grayscale images may outperform colour images in some cases, it's essential to consider the context of the dataset and the requirements of the classification task. From analysing all metrics at other resolutions, I would still argue that coloured images result in more accurate and confident predictions in multi-class classification tasks using a convolutional neural network on the DermaMNIST and PathMNIST datasets.

For the DermaMNIST datasets, it appears that there is little variability in the metrics between grayscale and colour. Whilst there is a slight improvement in the F1 score in the coloured images, as resolution increases the scores from both grayscale and coloured images seem to become very similar. This suggests that as resolution increases, colour is not as important of a feature in the classification of dermatoscopic images in a convolutional neural network. As there is more information in the coloured images at higher resolution, I would still recommend using the highest resolution colour images for the classification of the DermaMNIST dataset using a CNN.

The scores for the PathMNIST dataset do not follow a simple pattern. For the coloured images, as resolution increases it appears that all metrics improve, with a large improvement when increasing resolution from 28×28 to 64×64. The metrics still improve as resolution increases, but the level of improvement lessens. It is interesting to note that in the PathMNIST dataset, the full-colour high-resolution images outperform any other resolution and image in all metrics, but specifically the log loss. This implies that the higher resolution provides the model with the necessary information to predict the probabilities close to the true labels. The grayscale scores fluctuate a lot more. The scores for 128×128 appear to be similar to the coloured 224×224 images. Considering the purpose of H&E staining, I would advise that a high-resolution coloured image is used for the classification of the PathMNIST datasets to enhance the accuracy and generalisability of the model.

5.5. XGBoost

5.5.1. BreastMNIST

Table 5.10.: Results of classification using XGBoost on the BreastMNIST dataset at different resolutions

Resolution	Colour	Num. of Trees	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	47	Y	1.1s	**0.872**	**0.918**	**0.350**	**2.556**
64×64	Gray	17	Y	3.6s	0.821	0.885	0.443	1.925
128×128	Gray	19	Y	14.2s	0.795	0.867	0.442	1.879
224×224	Gray	27	Y	49.0s	0.846	0.902	0.397	2.200

The results of XGBoost on the BreastMNIST data differ from the other methods analysed. In the other methods, higher resolutions lead to more accurate predictions and a lower log loss. However, XGBoost's accuracy, F1 and log loss appear to deteriorate with increasing resolution. This is surprising considering the structure of the BreastMNIST data, as one would assume that the discovery of smaller structures with increasing resolution would make classifying malignant tumours easier.

Whilst all scores seem to decline when resolution is increased, there is an unexpected upturn in the results for the 224×224 resolution images. Whilst this could be a result of the higher resolution images revealing things that were missed or appeared as noise at lower resolutions, the result could potentially be identified as anomalous as it does not follow the overarching pattern.

An interesting observation is that the number of trees for model converges does not follow a set trend regarding resolution. However, there does seem to be a pattern whereby an increase in the number of trees results in a higher accuracy and F1 score and a lower log loss. This is relatively analogous to the results from the model training and validation as seen in figures **??-??**. This suggests that employing early stopping is potentially restricting the model from performing to its highest ability. Research should be conducted into the optimal number of trees to enhance the metrics through experimentation or literature review.

Unlike CNN, there is a notable increase in run time as the resolution increases. This is logical as increasing resolution results in a higher number of features for the model to analyse. The run time is also very short for the lowest resolution, with the model training and testing in 1.1 seconds.

Considering all metrics, the obvious choice for which resolution to use in the XGBoost model for the classification of the BreastMNIST dataset is the 28×28 resolution images. This accurately classified 87.2% of the samples, and an F1 score of 91.2% implies the model is making a large proportion of accurate positive predictions while capturing a large proportion of actual positives. The log loss

value of 0.350 is the lowest seen across the models on the BreastMNIST dataset, suggesting that XGBoost is the best model to calibrate the predicted probabilities and the true probabilities.

5.5.2. DermaMNIST

Table 5.11.: Results of classification using XGBoost on the DermaMNIST dataset at different resolutions and colours

Resolution	Colour	Num. of Trees	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	79	Y	1m 27s	0.698	0.637	0.887	0.752
	RGB	65	Y	1m 1s	0.723	**0.690**	**0.752**	**0.943**
64×64	Gray	56	Y	7m 4s	0.694	0.628	0.888	0.744
	RGB	63	Y	6m 21s	**0.727**	0.685	0.773	0.913
128×128	Gray	69	Y	21m 50s	0.701	0.638	0.892	0.751
	RGB	60	Y	31m 39s	0.717	0.672	0.769	0.903
224×224	Gray	53	Y	26m 31s	0.700	0.637	0.887	0.753
	RGB	63	Y	55m 49s	0.723	0.677	0.768	0.912

The metrics reported in table 5.11 appear to remain relatively constant across the different resolutions. However, it can be concluded that the performance of XGBoost is best when the images are kept in original colour rather than being transformed to grayscale. This suggests that colour is a very important feature for XGBoost to classify accurately and confidently, hence I would recommend that the classification of dermatoscopic images using XGBoost refrain from colour transformation to grayscale.

The number of trees required for convergence does not follow a specific trend. One might expect that as resolution increases, more trees need to be built due to the increase in the number of features. However, this is not the case, with the number of trees generally decreasing for both the coloured and grayscale images. There are a few anomalies though, with there being an increase in the number of trees required at the grayscale 128×128 images and the coloured 224×224 images.

One metric that does vary drastically between resolutions is the run time, hence considering the similarities of scores I would personally favour a model with the shortest run time.

Across all performance metrics in the grayscale images, there is little variation in scores. For accuracy, the scores differ by 0.7 percentage points and for F1 score by 0.9 percentage points. This emphasises that in XGBoost, resolution is not a contributing factor for the correct classification of dermatoscopic images. For the log loss, there is an even smaller difference across resolutions with a range of 0.5. This suggests that at all possible resolutions of grayscale images, XGBoost is predicting the probabilities with consistent accuracy and is relatively close to the

true labels, as the values of log loss range between 0.887 and 0.892. Fortunately, the accuracy and log loss scores indicate that the predictions of XGBoost are moderately accurate.

The scores do not follow a set pattern, with accuracy increasing when resolution is raised to 64×64 but plateauing for the 128×128 and 224×224 resolution images. F1 score is a different story, with the scores worsening as the resolution is increased however increasing again slightly for the highest resolution data. Log loss seems to increase when resolution increases, however, there is a slight decrease from 128×128 to 224×224 by a value of 0.1.

In the coloured images, there are again no consistent trends among the performance metrics. The accuracy score appears to improve from 28×28 to 64×64 resolutions, but then drops down for the 128×128 resolution images and finally comes back up for the highest resolution. The F1 score seems to get worse as resolution increases, except for an upturn for the highest resolution. Log loss appears to get worse initially in the initial resolution increase, but then from that point improves ever so slightly.

The higher scores for coloured images mean that I would not recommend transforming the colour of the DermaMNIST samples when using XGBoost. As the scores are not drastically different across resolutions, I decided to choose the model that has one stand-out metric value. This led me to conclude that I would utilise the coloured 28×28 images in an XGBoost model to classify skin lesions as the log loss for this resolution is much lower than the others. As correct classification is important due to the data domain, going with minimum log loss ensures that the model will adapt best to unseen data. The scores for all metrics except accuracy were the best for these images, where accuracy is only 0.4 percentage points less than the highest accuracy score.

The model on this data manages to achieve an accuracy of 72.3% and an F1 score of 69% within the 1 minute 1 second run time of the model. This means that the model does a moderately good job of classifying the samples correctly with a relatively good balance between precision and recall. The log loss value of 0.752 suggests that the model's predicted probabilities are reasonably close to the true class labels, however, there is still room for improvement. One way to action this improvement would be to remove the early stopping rounds and increase the maximum number of trees from 500.

5.5.3. PathMNIST

Table 5.12.: Results of classification using XGBoost on the PathMNIST dataset at different resolutions and colours

Resolution	Colour	Num. of Trees	Conv.	Time	Acc.	F1	Log Loss	Pred.
28×28	Gray	500	N	30m 58s	0.652	0.647	0.932	0.696
	RGB	500	N	1h 40m 59s	**0.820**	**0.820**	**0.513**	**1.598**
64×64	Gray	500	N	1h 23m 51s	0.684	0.680	0.838	0.814
	RGB	500	N	5h 42m 59s	0.815	0.814	0.522	1.560
128×128	Gray	500	N	6h 6m 23s	0.686	0.681	0.841	0.813
	RGB	500	N	1d 7h 27m 42s	0.791	0.789	0.563	1.404
224×224	Gray	500	N	10h 40m 23s	0.690	0.686	0.841	0.818
	RGB	500	N	1d 6h 35m 8s	0.779	0.776	0.594	1.309

The metrics reported in table 5.12 explore the classification of the PathMNIST dataset using XGBoost at different resolutions and colours. The first thing I noticed analysing the metrics is that in none of the cases did the XGBoost model converge within the maximum of 500 trees. This suggests that for data with highly complex features, an abundance of samples and multiple options of class allocation, the number of trees must be higher. In-depth data analysis and a review of current literature should be conducted before model fitting if exploring the use of a similar model in a clinical setting.

There is an obvious increase in the model run times as the number of features increases, with the grayscale images running quicker than the colour images. This is expected due to the grayscale images having a lower number of features. Whilst the increase appears on the surface to be relatively linear, there is a slight improvement in the run time when resolution is increased from 128×128 to 224×224, with the model run-time improving by roughly 52 minutes.

For the grayscale images, an increase in resolution has a linear relationship with the increase in accuracy and F1 score. In both of these metrics, as the resolution increases the scores for these two metrics increase, though the increases in some instances are very small. The log loss improves when the image resolution is increased from 28×28 resolution to 64×64, however further resolution increase results in the log loss value deteriorating slightly by 0.3. The log loss is approximately the same for the 128×128 and 224×224 resolution images, implying that distinguishing between which resolution allows for better model performance should be analysed considering the other metrics.

The colour images follow quite a different pattern from the grayscale images. An increase in the resolution results in worsening accuracy, F1 and log loss scores. This suggests that when resolution is increased, the noise added to the image distracts the XGBoost model from predicting with the highest accuracy.

Analysing all results, there is a clear winning resolution and colour for the classification of PathMNIST using XGBoost. The 28×28 images in original colour emerged as the clear winner. They achieved a predictive accuracy and F1 score of 82%, meaning that not only are the predictions highly accurate, but the model does very well at distinguishing between false and true positives. The predictive accuracy is 70% higher than random guessing, suggesting that the model displays great accuracy for this data domain. I believe the scores are impressive considering that there are 9 possible class allocations for this dataset. The log loss score of 0.513 suggests that the model predicts probabilities close to the true labels, meaning that the model will generalise well to unseen data.

5.5.4. Model Summary

Figure 5.4.: Summary of performance of XGBoost across all datasets, resolutions and metrics with varying colour.

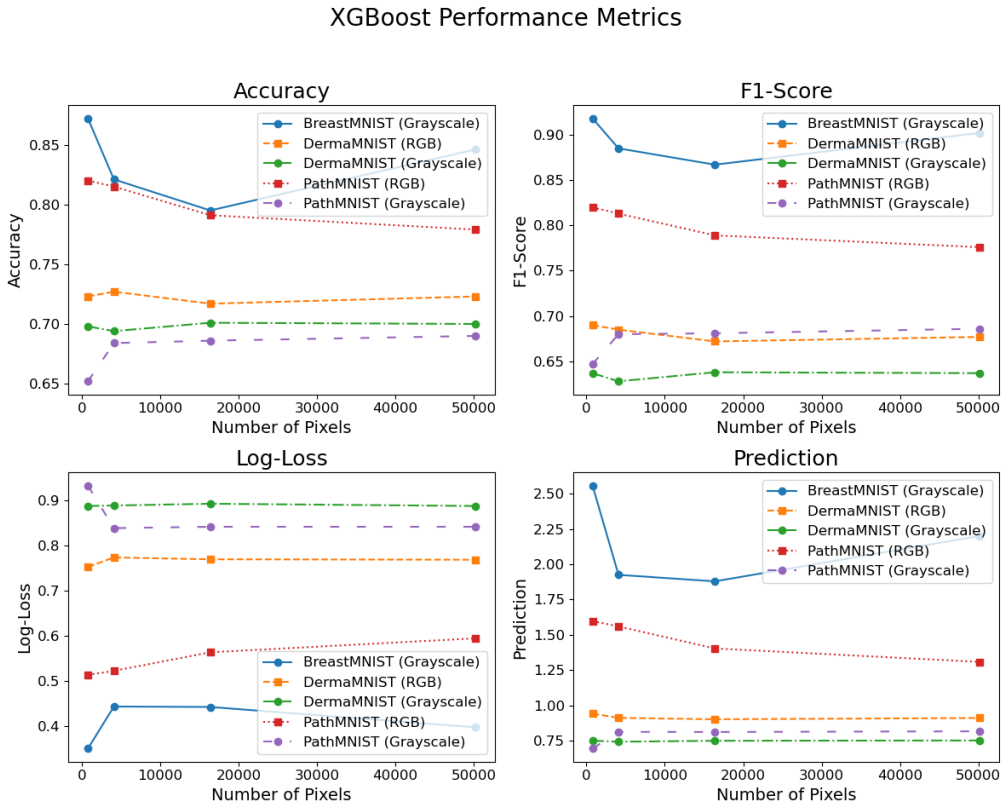


Figure 5.4 brings together the scores for all datasets to provide a visual comparison of the performance of XGBoost among the different data modalities.

For the binary classification task on the BreastMNIST dataset, the accuracy and F1 scores appear to deteriorate as resolution increases. However, there is then a sharp increase in the scores when the resolution increases from 128×128 to 224×224. For the log loss, there is an increase in the log loss when the resolution increases initially, but then the log loss starts to decrease as the resolution increases. In all

cases, the lowest resolution dataset in BreastMNIST performed the best in XGBoost. This suggests that for a binary classification task using breast ultrasound images, that higher resolution can damage the model performance.

The multi-class classification tasks on both the DermaMNIST and PathMNIST datasets appear to follow different trends. There is not a consistent pattern for the change in scores as resolution increases. The data suggests that for a multi-class classification task using an XGBoost model, coloured images result in a more accurate and confident model as opposed to grayscale images. Whilst this can only be confirmed for the PathMNIST and DermaMNIST datasets, the consistently better performance of the coloured images among the other methods suggests this could be the general case.

For the DermaMNIST data, the only consistency is that the coloured images outperformed the grayscale images. The accuracy and F1 scores remained reasonably constant among the resolutions, with the majority of changes in scores being a reduction. The lower-resolution images performed marginally better than the high-resolution images in colour. However, for the PathMNIST datasets, the lowest resolution yielded the best performance for the coloured data, whereas performance increased as resolution increased in the grayscale data.

Analysing the average performance with the prediction score, it appears that for XGBoost the lower resolution images without a colour transform perform optimally, both in scores and run time.

5.6. Optimal Model Configurations

Table 5.13.: Complete results of classification on the BreastMNIST dataset at different resolutions

Dataset	Res.	Colour	Method	Conv.	Time	Acc.	F1	L. Loss	Pred.
Breast MNIST	28×28	Gray	LDA	Y	31.3s	0.720	0.796	9.240	0.082
			LogReg	N	4.2s	0.776	0.851	0.543	1.497
			CNN	Y	29.9s	**0.878**	**0.920**	0.657	1.369
			XGBoost	Y	1.1s	0.872	0.918	**0.350**	**2.556**
	64×64	Gray	LDA	Y	2m 7s	0.769	0.845	1.712	0.471
			LogReg	N	21.3s	0.801	0.869	0.552	1.513
			CNN	Y	31.4s	0.872	0.912	0.638	1.398
			XGBoost	Y	3.6s	0.821	0.885	0.443	1.925
	128×128	Gray	LDA	Y	3m 18s	0.801	0.865	0.868	0.960
			LogReg	N	1m 20s	0.827	0.886	0.562	1.525
			CNN	Y	32.8s	0.821	0.886	0.964	0.885
			XGBoost	Y	14.2s	0.795	0.867	0.442	1.879
	224×224	Gray	LDA	Y	5m 17s	0.795	0.863	0.661	1.254
			LogReg	N	2m 5s	0.821	0.881	0.560	1.520
			CNN	Y	39.5s	0.833	0.887	0.569	1.512
			XGBoost	Y	49.0s	0.846	0.902	0.397	2.200

The choice of the optimal model for binary classification on the BreastMNIST dataset is dependent on the metric of focus and the resolution of the data available.

The statistical methods of LDA and logistic regression follow a similar pattern for the accuracy and F1 scores as resolution increases. The accuracy and F1 scores increase as resolution increases, and they peak at the 128×128 resolution images. There is a small reduction in the scores when resolution increases from 128×128 to 224×224, however for both methods, the reduction in the accuracy and F1 scores is minimal at under 1 percentage point in both cases. However, the log loss does not follow similar patterns. For linear discriminant analysis, the log loss appears to improve considerably as the resolution is increased, with the log loss value dropping from 9.240 at the 28×28 resolution images to 0.661 at the 224×224 resolution images. Logistic regression follows a completely different pattern, with log loss increasing as resolution increases followed by a very small decrease for the 224×224 resolution data. This is a surprising result, as one would expect the confidence in the model's predictions to increase as the number of features increases. As the changes in log loss are relatively small for logistic regression, the prediction score for both LDA and logistic regression appears to improve as resolution increases except for a small decrease for logistic regression on the 224×224 data. This implies that the overall capabilities of the model in terms of predictive accuracy, precision, recall and generalisability improve with higher-resolution data.

The machine learning methods follow quite a different pattern than the statistical methods on the BreastMNIST dataset. The accuracy and F1 scores appear to decrease as the resolution increases, with the metrics upturning for the highest-resolution images. For all resolutions except the 224×224 resolution images, the convolutional neural network outperforms XGBoost in accuracy and F1 scores. However, XGBoost has a much smaller log loss value than CNN in all resolutions and has the lowest log loss out of all methods at each resolution. This influences the high prediction scores that XGBoost has in comparison to the convolutional neural network.

If the goal of one's analysis is to maximise the predictive accuracy and balance between precision and recall for the BreastMNIST data, then the data suggests that a convolutional neural network on the lowest resolution images would achieve this. However, if the analysis aims to find the model that is accurate and generalisable to unseen data, then the XGBoost model for the lowest resolution data is the optimal model.

Considering the domain of the BreastMNIST dataset, it is important to train a model that is not only highly accurate and avoids false positives, but can perform just as well to new data. This is because, in breast cancer diagnosis, the cost of misclassification is very high on patient outcomes. In addition, the model must be confident in its predictions. The model that delivers this for the BreastMNIST dataset is XGBoost on the lowest dimensional data. This model manages to predict with high accuracy and classifies a low amount of false positives whilst achieving a low log loss value, instilling confidence in the model's ability to generalise well and confidently to new data. As a result of the impressive scores, the prediction score for this model is the highest out of all analyses in the paper.

Considering all factors, I believe that the optimal model for the binary classification task on the BreastMNIST dataset is the XGBoost model on the 28×28 resolution images. This model achieves the second-highest accuracy and F1 scores, only 0.6 and 0.2 percentage points below the highest scores. The log loss score is very low in comparison to the other methods and resolutions. Whilst not a crucial factor, the run time of XGBoost is incredibly fast at 1.1 seconds. Although model performance should not be compromised for time, it is a bonus that the fastest-running model for BreastMNIST also performs the best I think this model would be the best to apply to a clinical setting involving identifying breast cancer. However, as with all models in this analysis, its potential has only been explored regarding the identification of tumour malignancy.

Table 5.14.: Complete results of classification on the DermaMNIST dataset at different resolutions

Dataset	Res.	Colour	Method	Conv.	Time	Acc.	F1	L. Loss	Pred.
Derma MNIST	28×28	Gray	LDA	Y	3m 50s	0.627	0.580	1.669	0.364
			LogReg	N	1m 14s	0.669	0.580	1.041	0.600
			CNN	Y	1m 44s	0.664	0.578	1.405	0.442
			XGBoost	Y	1m 27s	0.698	0.637	0.887	0.752
		RGB	LDA	Y	19m 25s	0.604	0.580	1.669	0.218
			LogReg	N	2m 25s	0.690	0.636	0.880	0.754
			CNN	Y	1m 5s	0.728	0.699	0.886	0.806
			XGBoost	Y	1m 1s	0.723	0.690	**0.752**	**0.943**
	64×64	Gray	LDA	Y	26m 18s	0.510	0.527	6.909	0.075
			LogReg	N	8m 38s	0.670	0.585	1.037	0.605
			CNN	Y	3m 2s	0.701	0.637	1.171	0.572
			XGBoost	Y	7m 4s	0.694	0.628	0.888	0.744
		RGB	LDA	Y	1h 20m 9s	0.540	0.553	10.570	0.052
			LogReg	N	8m 47s	0.690	0.643	0.885	0.754
			CNN	Y	5m 43s	0.710	0.667	1.267	0.543
			XGBoost	Y	6m 21s	0.727	0.685	0.773	0.913
	128×128	Gray	LDA	Y	1h 11m 1s	0.545	0.541	8.539	0.064
			LogReg	N	13m 20s	0.674	0.593	1.037	0.611
			CNN	Y	7m 34s	0.711	0.648	1.295	0.525
			XGBoost	Y	21m 50s	0.701	0.638	0.892	0.751
		RGB	LDA	Y	1h 44m 54s	0.625	0.604	5.172	0.119
			LogReg	N	1h 7m 14s	0.694	0.647	0.885	0.758
			CNN	Y	46m 57s	0.726	0.693	0.979	0.725
			XGBoost	Y	31m 39s	0.717	0.672	0.769	0.903
	224×224	Gray	LDA	Y	2h 26m 2s	0.606	0.576	5.322	0.111
			LogReg	N	51m 34s	0.676	0.593	1.034	0.614
			CNN	Y	32m 18s	0.409	0.474	1.578	0.280
			XGBoost	Y	55m 49s	0.700	0.637	0.887	0.753
		RGB	LDA	Y	4h 22m 47s	0.645	0.614	3.799	0.166
			LogReg	N	3h 23m 48s	0.694	0.647	0.884	0.758
			CNN	Y	1h 19m 59s	**0.732**	**0.705**	0.829	0.866
			XGBoost	Y	55m 49s	0.723	0.677	0.768	0.912

The decision of the optimal image resolution, colour and method for the classification of DermaMNIST becomes more difficult as more factors are involved.

For the majority of resolutions and methods, the coloured images outperformed the grayscale images in terms of performance. Whilst there are a few instances of the opposite, the consensus is that coloured images are superior to grayscale images for the classification of dermatoscopic images. In my opinion, this makes clear sense as often skin abnormalities first appear with a change in appearance, normally through a colour change. Whilst grayscale images can still highlight

intensity, dark colours may appear very similar when different colours could represent quite different conditions. Due to this, I would recommend that a colour transformation is not completed on the DermaMNIST dataset or other related images. Hence, the preceding discussion will focus on coloured images only.

For the statistical methods, clear patterns do not appear. As resolution increases, the accuracy and F1 scores for LDA appear to improve apart from a plateau at 64×64 resolution. However, log loss appears to become much worse originally when the resolution is increased to 64×64, increasing from 1.669 to 10.570. Although the log loss does decrease again as resolution increases, the values are still very high for LDA. This suggests that for the DermaMNIST dataset, there could be potentially poor class separation or an abundance of misclassified samples, meaning the LDA model would likely perform poorly on unseen data. For logistic regression, all metrics remain relatively similar across the resolutions, with a difference of only 0.4 percentage points in accuracy, 1.1 percentage points in F1 score and a value of 0.005 in log loss. From this, it could be concluded that resolution has minimal influence on model performance for the DermaMNIST dataset, hence the resolution that allows the model to run most efficiently should be chosen. In all cases, logistic regression outperforms LDA significantly. I would not recommend the use of linear discriminant analysis for classification on the DermaMNIST dataset or similar data.

For accuracy and F1 score, the machine learning methods prove triumphant again as a clear improvement is evidenced. The convolutional neural network appears to have a slight fluctuation in accuracy and F1 scores with a slight decrease when resolution is upped to 64×64, followed by my score improvements for higher resolutions. Concerning accuracy, CNN performs the best at the highest resolution images, albeit only slightly better than the lowest resolution images. For log loss, the CNN appears to have a spike at the 64×64 resolution images which then decreases to its minimum at 224×224. The convolutional neural network performs optimally on the DermaMNIST dataset at the highest resolution.

For XGBoost, the results also appear to be quite consistent across all metrics. There appear to be fluctuations in the accuracy and F1 score in the intermediate resolutions, however these are minimal and within 2 percentage points. The log loss is also very consistent, with the lowest log loss achieved at the lowest image resolution. The difference between the log loss values is only 0.021, suggesting that the XGBoost model displays a similar predictive accuracy and generalisability across all resolutions.

Analysing all the available information, I propose that the XGBoost model on the 28×28 resolution data is the optimal model for the classification of the DermaMNIST dataset. While the method does not achieve the maximum accuracy and F1 scores, it minimises log loss, implying that the model would perform ideally among the other methods on new data. As the accuracy and F1 scores are still relatively high, the minimal log loss produces the best prediction score on the DermaMNIST analysis. Further to this, XGBoost demonstrates the quickest run time which is ideal for fast diagnosis in clinical settings, eliminating the long diagnosis period in current skin cancer practice. I am confident that out of all the methods analysed, this setup would prove to be the most robust on dermatoscopic

images.

Table 5.15.: Complete results of classification on the PathMNIST dataset at different resolutions

Dataset	Res.	Colour	Method	Conv.	Time	Acc.	F1	L. Loss	Pred.
Path MNIST	28×28	Gray	LDA	Y	2m 24s	0.414	0.371	1.644	0.239
			LogReg	N	2m 47s	0.159	0.137	2.200	0.067
			CNN	Y	30m 29s	0.747	0.756	1.045	0.718
			XGBoost	N	30m 58s	0.652	0.647	0.932	0.696
		RGB	LDA	Y	11m 20s	0.579	0.563	1.185	0.482
			LogReg	N	1h 8m 47s	0.422	0.386	1.701	0.237
			CNN	Y	15m 14s	0.430	0.408	3.813	0.110
			XGBoost	N	1h 40m 59s	0.820	0.820	**0.513**	**1.598**
	64×64	Gray	LDA	Y	57m 2s	0.401	0.374	1.784	0.217
			LogReg	N	54m 16s	0.135	0.127	2.310	0.057
			CNN	Y	48m 55s	0.716	0.717	1.442	0.497
			XGBoost	N	1h 23m 51s	0.684	0.680	0.838	0.814
		RGB	LDA	Y	2h 9m 24s	0.512	0.501	1.605	0.316
			LogReg	N	2h 4m 17s	0.408	0.383	1.806	0.219
			CNN	Y	58m 13s	0.786	0.763	1.567	0.494
			XGBoost	N	5h 42m 59s	0.815	0.814	0.522	1.560
	128×128	Gray	LDA	Y	5h 38m 29s	0.356	0.336	2.560	0.135
			LogReg	N	3h 12m 37s	0.134	0.130	2.471	0.053
			CNN	Y	1h 40m 20s	0.857	0.860	0.728	1.179
			XGBoost	N	6h 6m 23s	0.686	0.681	0.841	0.813
		RGB	CNN	Y	2h 22m 14s	0.838	0.837	1.047	0.800
			XGBoost	N	1d 7h 27m 42s	0.791	0.789	0.563	1.404
	224×224	Gray	CNN	Y	2h 15m 56s	0.736	0.770	1.299	0.590
			XGBoost	N	10h 40m 23s	0.690	0.686	0.841	0.818
		RGB	CNN	Y	4h 36m 10s	**0.882**	**0.884**	0.739	1.195
			XGBoost	N	1d 6h 35m 8s	0.779	0.776	0.594	1.309

Table 5.15 displays the results of the highest performing models for each method at the different resolutions and colours. As previously addressed, computational limitations prevented the LDA and logistic regression methods from being successful on the higher-resolution data.

In the majority of cases, the coloured images prove more successful than the grayscale images. When analysing the properties of the sample, this result does not surprise me as transformation to grayscale effectively abolishes the purpose of H&E staining. Although, likely, the crucial features are still detected in grayscale images due to the high intensity, the remainder of the features of the image may prove less distinguishable. My recommendation would be to refrain from transforming the PathMNIST or colorectal H&E stained tissue samples to grayscale, as colour is an important feature of the data. Therefore I will focus my decision on the optimal

model for the coloured images.

Whilst the data for the statistical methods on the coloured samples is limited, there are identifiable patterns. As resolution increases, it appears that all metrics decline with an increase in resolution. Interestingly, unlike the other two datasets, logistic regression performs worse than linear discriminant analysis. This could imply that the PathMNIST dataset has a data distribution close to a multivariate normal distribution meaning that the LDA classifier performs better. Another possibility is multicollinearity in the data, which would be effectively handled during the dimensionality reduction of LDA whereas logistic regression is sensitive to this.

The machine learning methods in general perform better. This does have exceptions, however, where the coloured 28×28 resolution images perform rather terribly in the convolutional neural network. Considering the performance of the convolutional neural network at higher resolutions, this does appear to be an anomalous result. On the contrary, this could be a result of low dimensions in the PathMNIST dataset losing key structural details of the tissue that are important for classification. Noise in the images may have confused the neural network during training.

Whilst all metrics for the CNN improve when resolution is increased, the scores for XGBoost appear to deteriorate. Like the other two datasets, the convolutional neural network proves to perform the best regarding accuracy and F1 score at the highest resolution and XGBoost minimises log loss at the lowest resolution image by considerably more than any other method. However, an important observation is that the XGBoost model does not converge at any resolution on the PathMNIST data. Whilst some may argue that this is a limitation of the method, I believe that this demonstrates that the method has the potential to improve its performance if the maximum number of trees is increased.

Whilst the convolutional neural network on the 224×224 resolution images would be the method of choice for those wanting to optimise predictive accuracy solely, I suggest that the XGBoost model on the 28×28 coloured images is the best classifier for the PathMNIST dataset. Whilst it does not converge within the maximum number of 500 trees, I believe that the method has the potential to improve further if the number of trees is increased. The method displays high accuracy and F1 scores, which although not the highest are still high considering that the problem is concerned with nine classes. XGBoost again proves to minimise the log loss the most, further reaffirming that the method is the most generalisable to unseen data. The low log loss score shows confidence in the model's predictions, which is especially important considering the data domain. XGBoost also displays a much faster run time than CNN, with the model training and testing in 1 hour 40 minutes and 59 seconds. I believe that this method shows great potential in a clinical setting regarding colorectal cancer due to its high accuracy and confidence.

5.7. Limitations

5.7.1. Non-convergence

Non-convergence proved to be a problem across logistic regression and XGBoost. Whilst this was only the case for XGBoost on the PathMNIST dataset, the non-convergence of logistic regression was unfortunately present in the majority of simulations.

Non-convergence implies that the algorithms were not able to obtain optimal parameter estimates that minimise the loss function. Because of this, likely, the estimates are not truly representative of the underlying relationships between the features and labels. This could result in inaccurate classifications, and falsely suggest better model performance than it achieves. This could result in inflated model confidence. Due to the datasets selected being in the oncology domain, misclassification risk is extremely high and potentially fatal.

The lack of convergence in logistic regression could be a result of data patterns known as complete separation, where the maximum likelihood estimate does not exist (Allison, 2008). This can be due to the lack of a maximum value of the likelihood function, meaning that parameter estimates cannot be derived. Other possibilities could be small sample size or highly correlated explanatory variables. Complete separation means that the predictor function performs perfect prediction therefore all classes are separated flawlessly. However, this produces infinite coefficient estimates and standard errors meaning convergence is not possible (Clark et al., 2023). A weaker form of complete separation where the predictor function predicts a subset of the response variables perfectly is called quasi-complete separation (Sauter and Held, 2016).

A method proposed by Firth (1993) to reduce bias in the maximum likelihood estimates, known as penalised maximum likelihood estimation, was shown by Heinze and Schemper (2002) to produce finite parameter estimates when applied to the problem of separation. Please consult the literature referenced above for further details.

A simple approach to try and rectify this problem for both XGBoost and logistic regression would be to increase the maximum number of iterations. Whilst this appears more unlikely to work with logistic regression due to the lack of convergence across all analysed datasets, this may work for XGBoost as the algorithm converged for the other data. This suggests that in the case of XGBoost, as the number of samples and classes increases, more trees are required.

Further solutions to attempt to achieve convergence in XGBoost include experimentation with different learning rates, regularisation strengths and tree depths.

5.7.2. Computational failure

For the coloured 128×128 and 224×224 resolution samples and grayscale 224×224 resolution samples of the PathMNIST dataset, the methods of linear discriminant analysis and logistic regression failed. Limited time was available to diagnose the problem due to the short time frame of the project.

This could be a consequence of a multitude of reasons. The most likely reason for the failure is the constraints on the server. As the server is shared, full access to the maximum computational power of the server is rarely possible. The methods of logistic regression and LDA on the high-dimensional PathMNIST data, which has nearly 100,000 samples, likely did not have access to the required amount of memory during the running of the techniques. Due to time constraints, I was not able to experiment with running the methods again.

The failure of the methods means that there is potentially added bias towards the machine learning methods on the PathMNIST data as they were successful in computation. There is a possibility that the failure of these methods introduced unintentional bias in model selection, as data was not available to analyse the performance of the statistical techniques at all resolutions and colours.

In future, one solution for mitigation could be to perform more rigorous feature scaling. As all the data was standardised before being pre-processed in MedMNIST, I naively assumed that the data had been scaled to a satisfactory level. Whilst I did perform some basic normalisation, this could have potentially not been rigorous enough to succeed in running high-dimensional data with a large sample size in sklearn.

Furthermore, another way to avoid this problem would be to secure a more powerful computational setup. By ensuring sole access to a high-memory, powerful machine, in theory, computational failure should not occur.

5.7.3. Model excecution times

Another limitation of the analysis is the varying computational times. As discussed previously, the Minerva server at the University of Manchester is shared amongst colleagues and students in the statistics group. Due to the unpredictable levels of traffic on the server, the execution times of the models in both training and testing have probably been influenced.

The processors used for the methods were also not consistent. All models except CNN were run on the server's CPU, but the CNN was run on GPU. As a result of this, the interpretability of the execution times may be limited as not all models are run across a consistent hardware configuration.

A solution to this limitation could be to ensure that all models are run using a consistent hardware configuration that is used by only the researcher. This would allow for a fairer comparison of execution times which showcase the model's capabilities rather than that of the hardware.

5.7.4. Bias of Prediction Score

The prediction score metric that I created appears to be biased towards methods that optimise log loss over accuracy and F1 score. Whilst this may be beneficial if that is the purpose of one's analysis, for studies wanting to optimise predictive accuracy, this proves as a limitation of the metric.

The prediction score could potentially favour a model that is low in predictive accuracy and balance of precision and recall but has a really low log loss. This could occur in cases of non-convergence. I would advise researchers to consider all metrics and not just the prediction score for this reason.

A potential solution to this could be to introduce weights and biases to the metric so that models are not penalised or favoured when there is an abnormal score.

5.7.5. Limited project time

The short time frame of the project meant that any anomalous results or computational failure could not be deeply investigated or re-run. Due to this, there is a potential that some of the results do not accurately represent the capabilities of the model. This could be a result of errors in programming or issues with the server.

As the project was short and an individual effort, quality control of the code used for the project was not completed. This means that there is a chance of errors in the code, which could influence the results of the modelling.

6. Conclusions

6.1. Summary

The analysis investigated statistical machine learning techniques for the classification of two-dimensional medical images from a selection of data from the MedMNIST+ collection at varying resolutions and colours. The investigation aims to assess the potential of methods in statistics and machine learning in the identification of cancerous features from medical images. This has the potential to aid clinical diagnoses, providing a second opinion to the expert conclusions drawn by clinicians.

The conclusion that I have derived is that machine learning techniques are superior to traditional statistical methods for classification. On average, the machine learning techniques achieved a higher and more confident predictive accuracy and a lower log loss, meaning that the machine learning models more accurately reflect the likelihood of each class.

For the DermaMNIST and PathMNIST datasets, the influence of colour on the classification capability was considered. This was done by comparing the performance of the samples before and after transformation to grayscale. The decline in performance when the images were transformed was unanimous, solidifying the fact that in these datasets colour is an important feature for correct classification. Whilst in practice this is computationally expensive, the performance improvement outweighs the computation cost as the risk of misclassification decreases. Considering the oncology domain of the datasets, misclassification has potentially severe consequences on patient outcomes.

For LDA and logistic regression, an increase in image resolution appeared to have positive consequences for the BreastMNIST and DermaMNIST datasets, but not the PathMNIST dataset. The methods appeared to perform optimally at the highest resolutions for these datasets, For the DermaMNIST dataset, the pattern occurred for the coloured images, with minor fluctuations, but was not as linear with the grayscale images. The grayscale images in DermaMNIST appear to have minimal improvement as the resolution increases, with the scores remaining fairly constant. The computational failure in PathMNIST meant that analysing the effect of an increase in resolution on the two statistical methods proved difficult. From the limited information available, it appears that the scores deteriorated when resolution increased. Whilst this is the pattern that emerged for the successful computations, speculation on the overall trend is not possible.

Feature selection proved to damage model performance rather than improve it. I do find this quite surprising, as intuition would lead me to believe that a model would perform better on a focused subset of data. On reflection, I believe

that the feature selection I completed had too large of a step size. In future analysis, I would remove features at a more gradual rate to reassess the potential of the technique in optimising classification.

For the BreastMNIST and DermaMNIST datasets, logistic regression outperformed linear discriminant analysis. I believe this to be a result of the regularisation that was introduced into the logistic regression models. However, this was not the case for PathMNIST, where LDA proved to be considerably better than logistic regression. My intuition believes that this could be a result of either multicollinearity in the data or the PathMNIST dataset being closer to a normal distribution, however, testing of this belief is yet to be conducted.

No regularisation was introduced to LDA due to time constraints. To apply shrinkage LDA in sci-kit learn the matrix of transformation to the lower-dimensional feature space must be explicitly calculated instead of estimated through singular value decomposition. For the high-dimensional data with an abundance of samples, this calculation has the potential to take a large amount of time hence for the short time frame of the project, this was not implemented. I predict that LDA with regularisation may perform similarly to regularised logistic regression.

The convolutional neural network appeared to perform best on average across all datasets at the highest resolutions. Whilst this may not be the case for individual metrics, a holistic consideration of the performance evidences my claim. Whilst this does result in longer execution times, given the potential fatality of misclassification, the effect of a slightly longer run time should not play a contributing factor in model analysis.

On the contrary, XGBoost has optimal performance on the lowest resolution data - the metrics decrease as resolution increases. A potential reason for this may be that XGBoost overfits data with a high number of features, as the inclusion of noise may sway the model to learn noise rather than the important patterns.

This analysis has not been without problems. The non-convergence of logistic regression and partial non-convergence of XGBoost have hindered the analysis, meaning that evaluation of the full potential of these methods has not been possible for all the datasets. It is still unclear whether the issues regarding convergence in logistic regression are a result of the perfect predictor function or due to not enough iterations being run.

The decision of the optimal model for the classification of medical images in the field of oncology is not as straightforward of an answer as one may expect. Consideration of the resolution of the data, number of classes and the severity of misclassification must be extensively considered before choosing a model for the detection of cancerous features.

If the medical images available are high resolution and the goal is to maximise predictive accuracy and reduce the risk of detecting false positives, then my research points towards the use of a convolutional neural network. This consistently outperformed the other methods at resolution resolutions in terms of accuracy and

F1 score. The CNN proved to successfully predict the class labels to a relatively high accuracy for the highest resolution data, proving much better than random guessing. Whilst the scores are not perfect, there is potential to further improve the models with more extensive hyperparameter tuning.

For maximising model confidence on low-resolution medical images, XGBoost proved to be victorious. The model displayed the lowest log loss score out of all the metrics, being minimised at the lowest image resolution across all three datasets. Because of the low log loss, XGBoost often performed the best in terms of the prediction score. Whilst there is a potential that the prediction score has been biased by log loss, a review of the accuracy and F1 scores shows that the method performed well across all metrics.

As I completed a comprehensive analysis of the best model across all metrics, I concluded that the optimal model for medical image classification is the XGBoost model on the lowest-resolution images. This consistently proved to minimise log loss the most and resulted in all datasets in the best prediction score. Furthermore, the method only fell slightly short of performing best in terms of accuracy and F1 score, implying that the model performs the best all around.

Whilst the results of this paper may be used as a benchmark for medical imaging, experiments must be conducted on other types of images. My analysis only concludes this result for the BreastMNIST, DermaMNIST and PathMNIST datasets. Whilst the results are promising in the areas of breast cancer, skin cancer and colorectal cancer detection, it is important to recognise that the dataset is for academic research only and may not result in the same success in clinical settings. Although the images in the analysis are originally sourced from medical facilities, the extent of pre-processing could potentially result in images which are not realistic in clinical practice. Because of this, I advise that before a method is decided, an in-depth investigation into the data structure and modality is conducted.

6.2. Recommendations

6.2.1. Further hyperparameter tuning

A limited amount of hyperparameter tuning was completed due to the strict time constraints of the project.

For LDA, I would encourage further research into the effect of shrinkage on classification performance. Although this is likely to be computationally expensive as the transformation matrix is unable to be estimated through singular value decomposition in sklearn, I anticipate the improved performance of linear discriminant analysis in classification when shrinkage is introduced. I believe this could reduce or even close the gap between the performances of LDA and logistic regression.

Whilst a moderate amount of tuning was undertaken for the logistic regression model, there is still the potential for more to be completed. For example, the maximum number of iterations could be increased to analyse whether this was

a cause for non-convergence, as well as changing the stopping criteria tolerance. Experimentation of Elastic-Net regularisation I believe would be worthwhile, as successes of both LASSO and Ridge regularisation on the logistic regression models mean that a combination of the two may perform best. Regarding regularisation, an interesting experiment would be to test different levels of regularisation strength on model performance. Not all of the available optimisation algorithms were employed in my analysis of logistic regression. Testing Newtonian methods such as Newton Conjugate Gradient or Newton-Cholesky optimisation would provide a useful exploration of the power of Newton's method on the optimisation of large datasets. Finally, the multi-class classification problems in logistic regression were solved using method one versus the rest, essentially creating multiple binary classification problems. Approaching multi-class classification using multinomial logistic regression could potentially produce different results.

For the convolutional neural network, there are many further parameters to be tuned. In my analysis, most of the parameter values were based on the original convolutional neural network used in Yang et al. (2021, 2023, 2024). Varying parameters such as the learning rate and batch size would likely alter performance.

The XGBoost model used in the analysis remained with many of the default parameter values. Assessment of changes in the effect of the maximum tree depth, boosting algorithm, tree method, learning rate, regularisation terms and the introduction of tree pruning on performance could be an avenue of exploration to further improve the performance of XGBoost in the domain.

Although manual hyperparameter tuning is possible in all cases, this could become labour-intensive. An alternative to this could be to use a method of hyperparameter tuning such as random search. Whilst in practice the algorithms take a long time to find the optimal parameters, the increase in model performance would be beneficial for medical image classification. For further details on the random-search algorithm, please consult Bergstra and Bengio (2012).

6.2.2. GPU support in scikit-learn

Whilst graphics processing units have been used since the 1970s for gaming, and their use in machine learning is highlighted in the PhD thesis of Buck et al. (2004), their use for inference and training in machine learning was heavily popularised by Raina et al. (2009) at Stanford University. These systems were seen as more effective for parallel processing in repeatable, identical computations. This accelerated the computation time for the models against CPUs that process multiple, more complex computations at the same time.

Currently, scikit-learn is only compatible with CPUs. I believe that support for GPUs in sklearn would provide a wealth of benefits, such as faster execution times for large datasets and support for parallel processing. Specifically, I believe that hyperparameter tuning would benefit greatly in sklearn with GPU support, as multiple combinations of parameters could be tested simultaneously. This would reduce overall model training times, allowing for simplified model building in sklearn.

6.2.3. Implementation of logistic regression in PyTorch

The setup of PyTorch appears to allow the construction of a logistic regression model. As the package supports linear layers and the sigmoid function, in theory, the construction of a logistic regression model is possible. Due to the GPU support of PyTorch, this could rapidly improve the execution time for logistic regression models in Python. I invite investigation of this methodology and analysis of the performance in comparison to sklearn.

6.2.4. Experimentation of other colour space conversions

Although my analysis was limited to the assessment of colour on the effect of classification using both RGB and grayscale images, there is a wealth of colour transformations available for imaging data. Investigating transformation to other colour spaces such as hue, saturation and value (HSV), which is inherently easier for humans to understand, could potentially also be easier for a model to interpret, leading to increased performance.

6.2.5. Data Augmentation

To increase the diversity of samples in the training and validation dataset, data augmentation could be applied to allow the model to improve generalisation and reduce overfitting. Examples of augmentation include cropping, rotation and flipping of images as well as further changes to image colour such as transformation to a different colour space. By ensuring that the data used for training the model has increased variability, the model likely learns the important features for classification better rather than focusing on uninformative features. I believe this could improve model performance, and an analysis of this kind on the MedMNIST datasets would be intriguing.

6.3. Closing Remarks

The paper provides confidence in the potential of statistical machine learning as an aid for medical image classification. Whilst the models are not performing perfectly currently, I am sure that one can achieve close to this with further tuning and augmentation. However, the risk of misclassification is likely to always be present. Misclassification can have disastrous consequences for the outcome of patients with cancer. Therefore, I do not see machine learning methods as a replacement for clinicians but rather a complement. The methods proposed in the paper could act as a second or third opinion for clinicians, more as a verifier than a decision-maker.

I hope that the analysis and results presented instil faith rather than fear in the potential of machine learning techniques in medicine. I believe that they are only going to enhance the field by providing further security and confidence in diagnoses. I believe that one day, similar models to what is discussed in the paper will be utilised heavily in medicine, enhancing workflows and leading to faster clarification of human health concerns. This has the potential to revolutionise the future of healthcare.

Bibliography

Al-Dhabyani, W., Gomaa, M., Khaled, H., and Fahmy, A. (2020). Dataset of breast ultrasound images. *Data in Brief*, 28:104863.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6.

Allison, P. D. (2008). Convergence failures in logistic regression. In *SAS Global Forum*, volume 360, page 11.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550.

Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, 53:370–418.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, 1 edition.

Bonnans, J.-F., Gilbert, J. C., Lemaréchal, C., and Sagastizábal, C. A. (2006). *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media.

Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., and Hanrahan, P. (2004). Brook for gpus: Stream computing on graphics hardware.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Clark, A. (2015). Pillow (pil fork) documentation.

Clark, R. G., Blanchard, W., Hui, F. K., Tian, R., and Woods, H. (2023). Dealing with complete separation and quasi-complete separation in logistic regression for linguistic data. *Research Methods in Applied Linguistics*, 2:100044.

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

Doquire, G. and Verleysen, M. (2013). Mutual information-based feature selection for multilabel classification. *Neurocomputing*, 122:148–155.

Du, K.-L. and Swamy, M. N. S. (2019). *Perceptrons*, pages 81–95. Springer London, London.

Ferlay, J., Colombet, M., Soerjomataram, I., Parkin, D. M., Piñeros, M., Znaor, A., and Bray, F. (2021). Cancer statistics for the year 2020: An overview. *International Journal of Cancer*, 149(4):778–789.

Firth, D. (1993). Bias reduction of maximum likelihood estimates. *Biometrika*, 80(1):27–38.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Galã, A. (2000). The theory of newton's method.

González-Mejía, A., Vance, L., Eason, T., and Cabezas, H. (2015). Recent developments in the application of fisher information to sustainable environmental management. *Assessing and Measuring Environmental Impact and Sustainability*, pages 25–72.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585:357–362.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Heinze, G. and Schemper, M. (2002). A solution to the problem of separation in logistic regression. *Statistics in Medicine*, 21(16):2409–2419.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95.

Ivakhnenko, A. G. (1968). The group method of data handling; a rival of the method of stochastic approximation. *Soviet Automatic Control*.

Izenman, A. J. (2008). *Modern multivariate statistical techniques*, volume 1. Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *Springer Texts in Statistics An Introduction to Statistical Learning with Applications in Python*. Springer Cham, 1 edition.

Kather, J. N., Krisam, J., Charoentong, P., Luedde, T., Herpel, E., Weis, C.-A., Gaiser, T., Marx, A., Valous, N. A., Ferber, D., Jansen, L., Reyes-Aldasoro, C. C., Zörnig, I., Jäger, D., Brenner, H., Chang-Claude, J., Hoffmeister, M., and Halama, N. (2019). Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLOS Medicine*, 16(1):1–22.

Kearns, M. (1988). Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory*.

Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA.

Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., and Klein, M. (2002). *Logistic regression*. Springer.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Liew, X. Y., Hameed, N., and Clos, J. (2021). An investigation of xgboost-based algorithm for breast cancer classification. *Machine Learning with Applications*, 6:100154.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. PhD thesis, Master's Thesis (in Finnish), Univ. Helsinki.

Macenko, M., Niethammer, M., Marron, J. S., Borland, D., Woosley, J. T., Guan, X., Schmitt, C., and Thomas, N. E. (2009). A method for normalizing histology slides for quantitative analysis. In *Proceedings of the 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Boston, MA, USA, June 28 - July 1, 2009*, pages 1107–1110. IEEE.

Maleki, A., Raahemi, M., and Nasiri, H. (2023). Breast cancer diagnosis from histopathology images using deep neural network and xgboost. *Biomedical Signal Processing and Control*, 86.

Mcculloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Minsky, M. and Papert, S. (1969). An introduction to computational geometry. *Cambridge tiass., HIT*, 479(480):104.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

Ogunleye, A. and Wang, Q.-G. (2019). Xgboost model for chronic kidney disease diagnosis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17:2131–2140.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Peter, S. C., Dhanjal, J. K., Malik, V., Radhakrishnan, N., Jayakanthan, M., and Sundar, D. (2019). Quantitative structure-activity relationship (qsar): Modeling approaches to biological applications. In Ranganathan, S., Gribskov, M., Nakai, K., and Schönbach, C., editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 661–676. Academic Press, Oxford.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12:145–151.

Raina, R., Madhavan, A., and Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880.

Rezaei, A., Fathony, R., Memarrast, O., and Ziebart, B. (2020). Fairness for robust log loss classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5511–5518.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591 vol.1.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rosenblatt, F. et al. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, volume 55. Spartan books Washington, DC.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71:599–607.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sauter, R. and Held, L. (2016). Quasi-complete separation in random effects of binary response mixed models. *JOURNAL OF STATISTICAL COMPUTATION AND SIMULATION*, 86:2781–2796.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5:197–227.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schmidt, M., Roux, N. L., Bach, F., and Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162:83–112.

Shen, D., Wu, G., and Suk, H.-I. (2017). Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248.

Shi, M., Cheung, G., and Shahamiri, S. R. (2023). Speech and language processing with deep learning for dementia diagnosis: A systematic review. *Psychiatry Research*, 329.

Snyman, J. A. (2005). *Practical Mathematical Optimisation*. Springer, 1 edition.

Stavros, A. T., Thickman, D., Rapp, C. L., Dennis, M. A., Parker, S. H., and Sisney, G. A. (1995). Solid breast nodules: use of sonography to distinguish between benign and malignant lesions. *Radiology*, 196(1):123–134.

Svensson, W. E. (1997). A review of the current status of breast ultrasound. *European Journal of Ultrasound*, 6:77–101.

Tahvili, S. and Hatvani, L. (2022). Benefits, results, and challenges of artificial intelligence. *Artificial Intelligence Methods for Optimization of the Software Testing Process*, pages 161–172.

Tan, L. (2015). Code comment analysis for improving software quality. *The Art and Science of Analyzing Software Data*, pages 493–517.

Tang, J., Rangayyan, R. M., Xu, J., El Naqa, I., and Yang, Y. (2009). Computer-aided detection and diagnosis of breast cancer with mammography: recent advances. *IEEE transactions on information technology in biomedicine*, 13(2):236–251.

Tietz, M., Fan, T. J., Nouri, D., Bossan, B., and skorch Developers (2017). *skorch: A scikit-learn compatible neural network library that wraps PyTorch*.

TorchVision maintainers and contributors (2016). Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`.

Tschandl, P., Rosendahl, C., and Kittler, H. (2018). Data descriptor: The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5.

Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Thesis (Ph. D.). Appl. Math. Harvard University*. PhD thesis, Harvard University.

Wise, J. (2022). Persistent understaffing of the nhs is putting patients at risk, say mps.

Xanthopoulos, P., Pardalos, P. M., Trafalis, T. B., Xanthopoulos, P., Pardalos, P. M., and Trafalis, T. B. (2013). Linear discriminant analysis. *Robust data mining*, pages 27–33.

Yang, J., Shi, R., and Ni, B. (2021). Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195.

Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., and Ni, B. (2023). Medmnist v2 - a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10.

Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., and Ni, B. (2024). [MedMNIST+] 18x Standardized Datasets for 2D and 3D Biomedical Image Classification with Multiple Size Options: 28 (MNIST-Like), 64, 128, and 224.

Glossary

Actinic Keratoses (Solar Keratoses) and Intrapithelial Carcinoma (Bowen's Disease) Common non-invasive variants of squamous cell carcinoma induced by UV-light, with the surrounding skin often severely sun damaged. In the case of Bowen's disease, the lesions are caused by the human papilloma virus (HPV). The lesions may progress to an invasive squamous cell carcinoma, which is often non-pigmented. Actinic keratoses are more common on the face and intraepithelial carcinoma on other body sites. 67

Adipose Commonly known as body fat, it is a connective tissue that is mainly composed of fat cells called adipocytes. 69

Basal Cell Carcinoma A common variant of epithelial skin cancer that rarely metastasises but grows destructively if untreated. 67

Benign A condition/tumour/growth that is not cancerous. 65, 127, 128

Benign keratosis-like lesion A generic class of lesions that include seborrheic keratoses, solar lentigo and lichen-planus like keratoses (LPLK). The appearance of the lesions is different dermatoscopically but display similar biological properties and are reported under the same term histopathologically. LPLK often mimic features of melanoma hence are often examined through or excised. 67

Colorectal Adenocarcinoma The third most common cancer which begins in the cells that make and release mucus and other fluids, i.e. in the glands that line the organs. This type is specifically related to colorectal cancer. 69

Dermatofibroma A benign skin lesion regarded as a benign growth or inflammatory reaction to minimal trauma. 67

Dermatoscopic Relating to the method of dermatoscopy. 66

Dermatoscopy A diagnostic technique used to improve the diagnosis of benign and malignant pigmented skin lesions. 127

Epithelium A type of body tissue that covers all internal and external surfaces of the body and lines body cavities and hollow organs. It is the major tissue found in glands. 69

Formalin-Fixed Paraffin-Embedded (FFPE) tissue A commonly used method for preserving tissues by fixing them using formalin embedding in paraffin wax. This allows for long-term preservation of tissue samples. 68

Gastrectomy A type of surgery to remove all or part of the stomach. 69

Hematoxylin and Eosin (H&E) staining A laboratory method that uses two dyes called hematoxylin and eosin that clarifies parts of a cell under a microscope. Hematoxylin shows the ribosomes and genetic material within the nucleus and other structures as a deep blue-purple colour, whereas eosin shows the cytoplasm, collagen, connective tissue and other cell-supporting structures as an orange-pink-red colour. Aids the identification of cell types, patterns and structure in a sample. 68

Histological Relating to the discipline of histology. 68, 69

Histology The microscopic study of tissues and organs through sectioning, staining and examination. 128

Lesion An area of abnormal or damaged tissue caused by injury, infection or disease. 65–67, 127

Lymphocyte A type of immune cell that is made in the bone marrow and found in blood and lymph tissue. 69

Malignant A condition/tumour/growth that is classed as cancerous. 65, 127, 128

Melanocyte A type of cell which produce the skin-darkening pigment melanin. 128

Melanocytic Relating to the type of cells called melanocytes. 66

Melanocytic nevi Usually symmetric, benign neoplasms of melanocytes that appear in many variants. 67

Melanoma A malignant neoplasm derived from melanocytes that can be invasive or non-invasive. They are often classified as chaotic.. 66, 67, 127

Metastasis The spread of malignant cells from the place of original formation to another part of the body. 21, 69

Neoplasm Abnormal mass of tissue that forms when cells grow and divide more than they should or do not die when they should. 128

Nevus A benign skin growth that is formed by a cluster of melanocytes. Commonly referred to as a mole.. 66

Normal colon mucosa The inner lining of the colon. 69

Stroma Cells and tissues that support and give structure to organs, glands and other tissues in the body. May be involved in the growth and spread of cancerous cells. 69

Vascular lesion A range of skin lesions caused by blood vessels including cherry angiomas, angiokeratomas, pyogenic granulomas and haemorrhages. 67

A. Model Training and Tuning

The purpose of this appendix is to provide readers with access to the full training data in the form of figures and tables. Note that all scores have been rounded to 3 decimal places and all times have been rounded to 1 decimal place if under 1 minute and to the nearest second for any larger times. Times and scores to a higher precision are available at `https://github.com/jackhodgkinson/statml_medmnistplus/tree/main/Results`.

A.1. BreastMNIST

A.1.1. Linear Discriminant Analysis

The following tables and figures display the results from training and tuning a Linear Discriminant Analysis model for the binary classification of training examples using the BreastMNIST dataset.

Figure A.1.: Plot of scores of LDA model tuning on the BreastMNIST dataset at 28×28 resolution

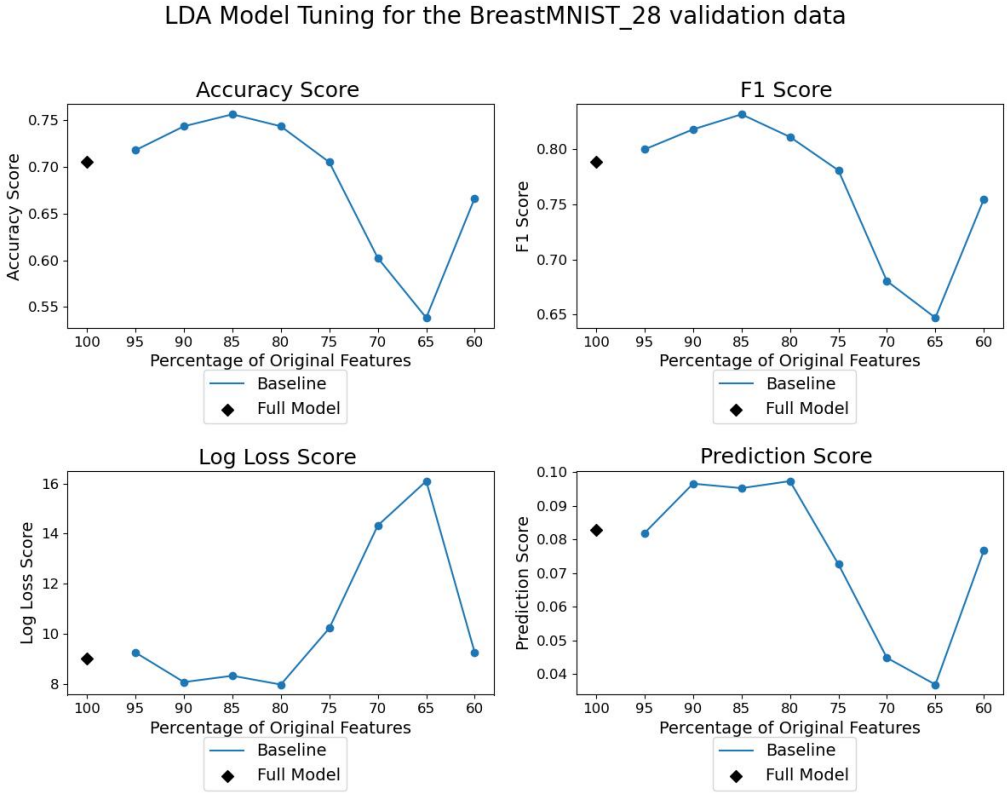


Table A.1.: Scores from LDA model tuning on BreastMNIST dataset at 28×28 resolution

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	8.4s	0.705	0.789	9.023	0.083
95	Y	4.6s	0.718	0.8	9.263	0.082
90	Y	7.8s	0.744	0.818	8.083	0.097
85	Y	4.3s	0.756	0.832	8.335	0.095
80	Y	4.2s	0.744	0.811	7.985	0.097
75	Y	4.8s	0.705	0.781	10.231	0.073
70	Y	3.5s	0.603	0.680	14.325	0.045
65	Y	7.8s	0.538	0.647	16.096	0.037
60	Y	3.5s	0.667	0.755	9.263	0.077

Figure A.2.: Plot of scores of LDA model tuning on the BreastMNIST dataset at 64×64 resolution

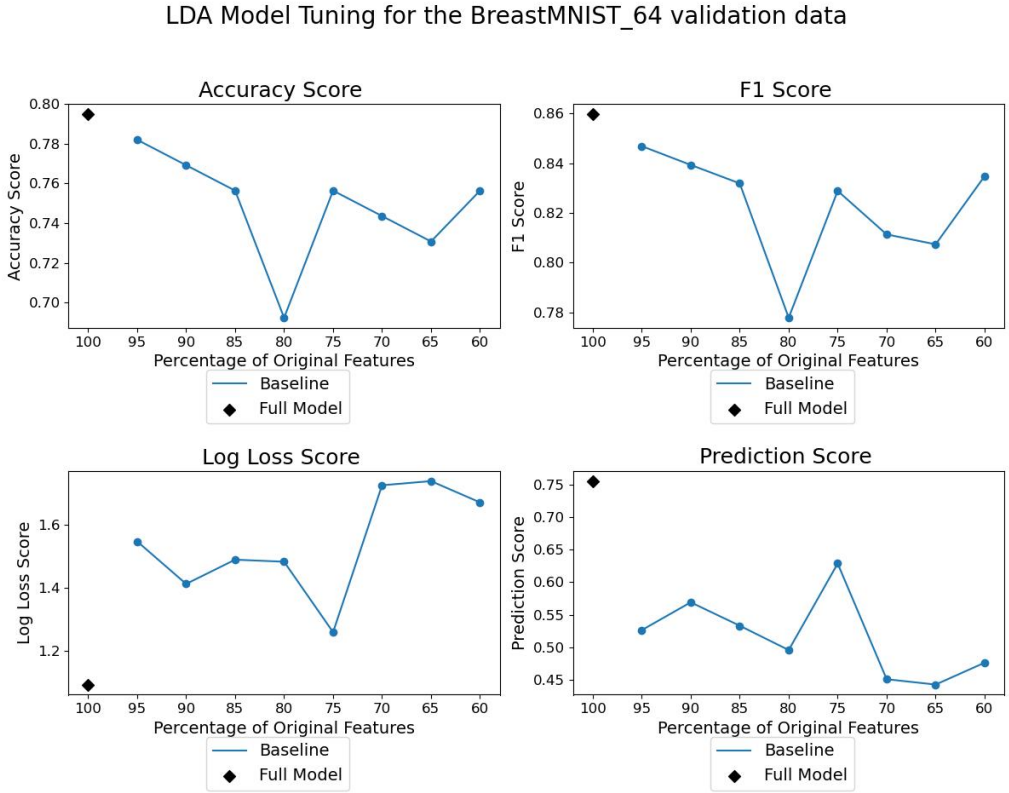


Table A.2.: Scores from LDA model tuning on BreastMNIST dataset at 64×64 resolution

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	2.2s	0.795	0.860	1.095	0.756
95	Y	2.8s	0.782	0.847	1.548	0.526
90	Y	1.8s	0.769	0.839	1.413	0.569
85	Y	2.0s	0.756	0.832	1.490	0.533
80	Y	1.4s	0.692	0.778	1.484	0.495
75	Y	1.2s	0.756	0.823	1.260	0.629
70	Y	3.2s	0.744	0.811	1.726	0.450
65	Y	1.7s	0.731	0.807	1.739	0.442
60	Y	1.5s	0.756	0.835	1.672	0.476

Figure A.3.: Plot of scores of LDA model tuning on the BreastMNIST dataset at 128×128 resolution

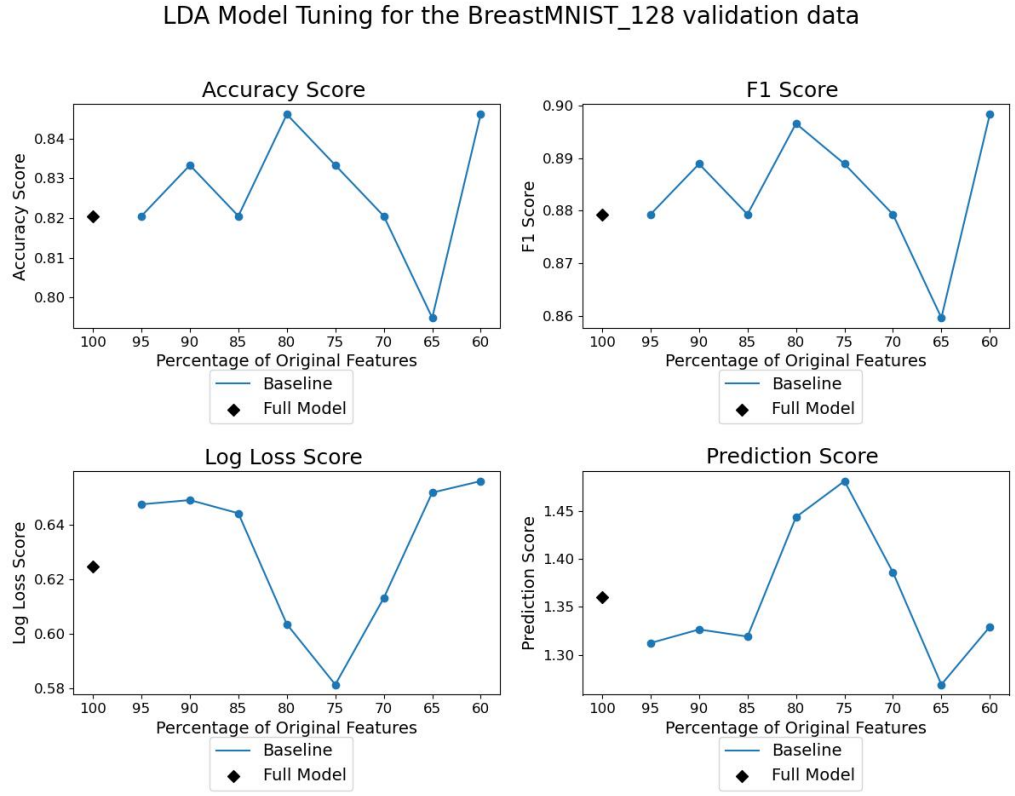


Table A.3.: Scores from LDA model tuning on BreastMNIST dataset at 128×128 resolution

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	6.4s	0.821	0.879	0.643	1.361
95	Y	5.2s	0.821	0.879	0.648	1.312
90	Y	5.4s	0.833	0.889	0.649	1.326
85	Y	3.9s	0.821	0.879	0.644	1.319
80	Y	4.1s	0.846	0.897	0.604	1.444
75	Y	4.2s	0.833	0.889	0.581	1.481
70	Y	4.7s	0.821	0.879	0.613	1.386
65	Y	4.4s	0.795	0.860	0.652	1.269
60	Y	3.7s	0.846	0.898	0.656	1.329

Figure A.4.: Plot of scores of LDA model tuning on the BreastMNIST dataset at 224×224 resolution

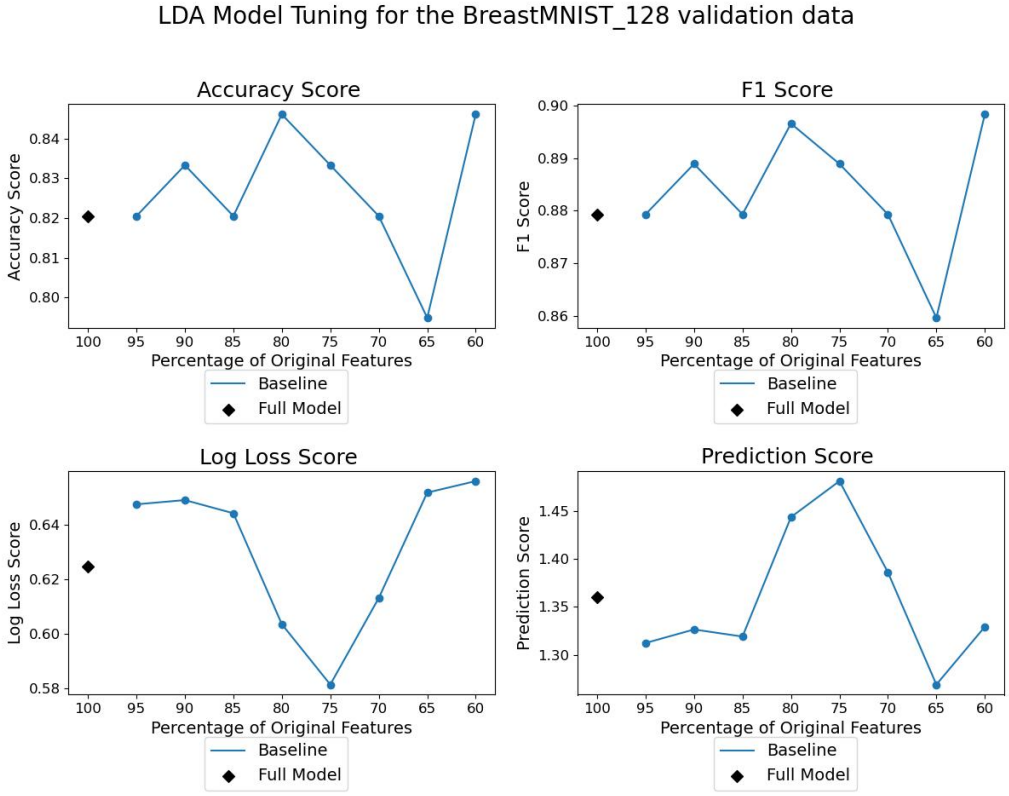


Table A.4.: Scores from LDA model tuning on BreastMNIST dataset at 224×224 resolution

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	13.1s	0.846	0.897	0.488	1.786
95	Y	11.6s	0.833	0.889	0.467	1.843
90	Y	12.4s	0.833	0.889	0.477	1.806
85	Y	13.1s	0.833	0.889	0.467	1.845
80	Y	12.0s	0.833	0.889	0.463	1.859
75	Y	10.6s	0.833	0.889	0.464	1.855
70	Y	9.7s	0.833	0.889	0.477	1.806
65	Y	9.0s	0.833	0.889	0.481	1.791
60	Y	8.9s	0.833	0.889	0.482	1.785

A.1.2. Logistic Regression

The following tables and figures display the results from training a Logistic Regression model for the classification of training examples using the BreastMNIST dataset at different resolutions alongside assessing the effect feature selection by testing the model on the validation dataset.

Figure A.5.: Plot of scores of Logistic Regression model tuning on the BreastMNIST dataset at 28×28 resolution

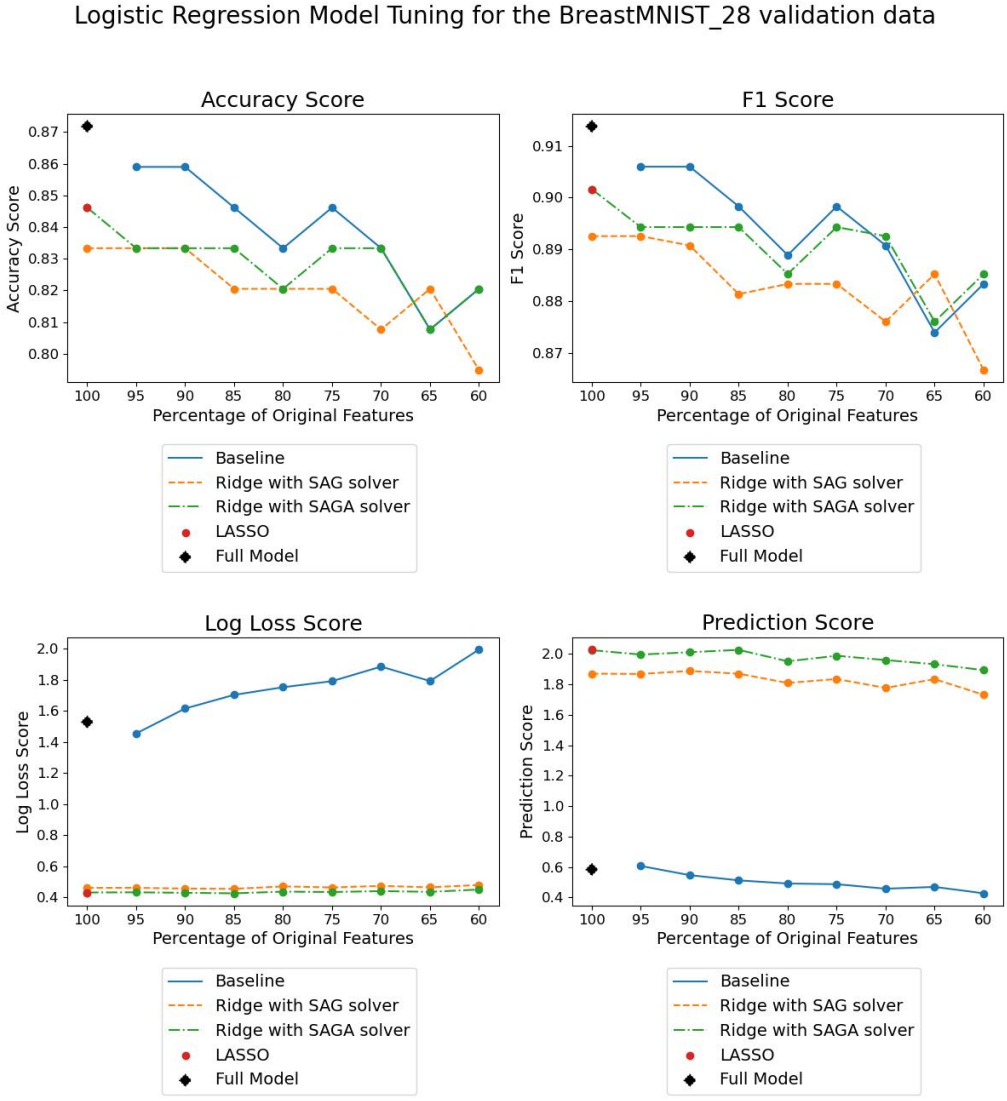


Table A.5.: Scores of Logistic Regression model tuning on the BreastMNIST dataset at 28×28 resolution

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2.0s	0.872	0.914	1.529	0.584
95	N	0.9s	0.859	0.906	1.455	0.606
90	N	0.9s	0.859	0.906	1.616	0.546
85	N	0.9s	0.846	0.898	1.704	0.512
80	N	0.9s	0.833	0.889	1.753	0.491
75	N	0.9s	0.846	0.898	1.792	0.487
70	N	0.8s	0.833	0.891	1.886	0.457
65	N	0.8s	0.808	0.874	1.793	0.469
60	N	0.8s	0.821	0.883	1.996	0.427
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1.7s	0.833	0.893	0.462	1.870
95	N	1.7s	0.833	0.893	0.462	1.868
90	N	1.6s	0.833	0.891	0.457	1.870
85	N	1.7s	0.821	0.881	0.455	1.870
80	N	1.4s	0.821	0.883	0.471	1.809
75	N	1.3s	0.821	0.883	0.464	1.835
70	N	1.2s	0.807	0.876	0.474	1.776
65	N	1.1s	0.821	0.885	0.465	1.834
60	N	1.1s	0.795	0.867	0.480	1.731
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2.0s	0.846	0.902	0.432	2.024
95	N	1.9s	0.833	0.894	0.433	1.995
90	N	2.0s	0.833	0.894	0.430	2.011
85	N	1.8s	0.833	0.894	0.426	2.026
80	N	1.7s	0.821	0.885	0.437	1.951
75	N	1.6s	0.833	0.894	0.435	1.987
70	N	1.5s	0.833	0.893	0.441	1.959
65	N	1.4s	0.808	0.876	0.436	1.932
60	N	1.2s	0.821	0.885	0.451	1.892
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	4.15s	0.846	0.902	0.431	2.028

Figure A.6.: Plot of scores of Logistic Regression model tuning on the BreastMNIST dataset at 64×64 resolution

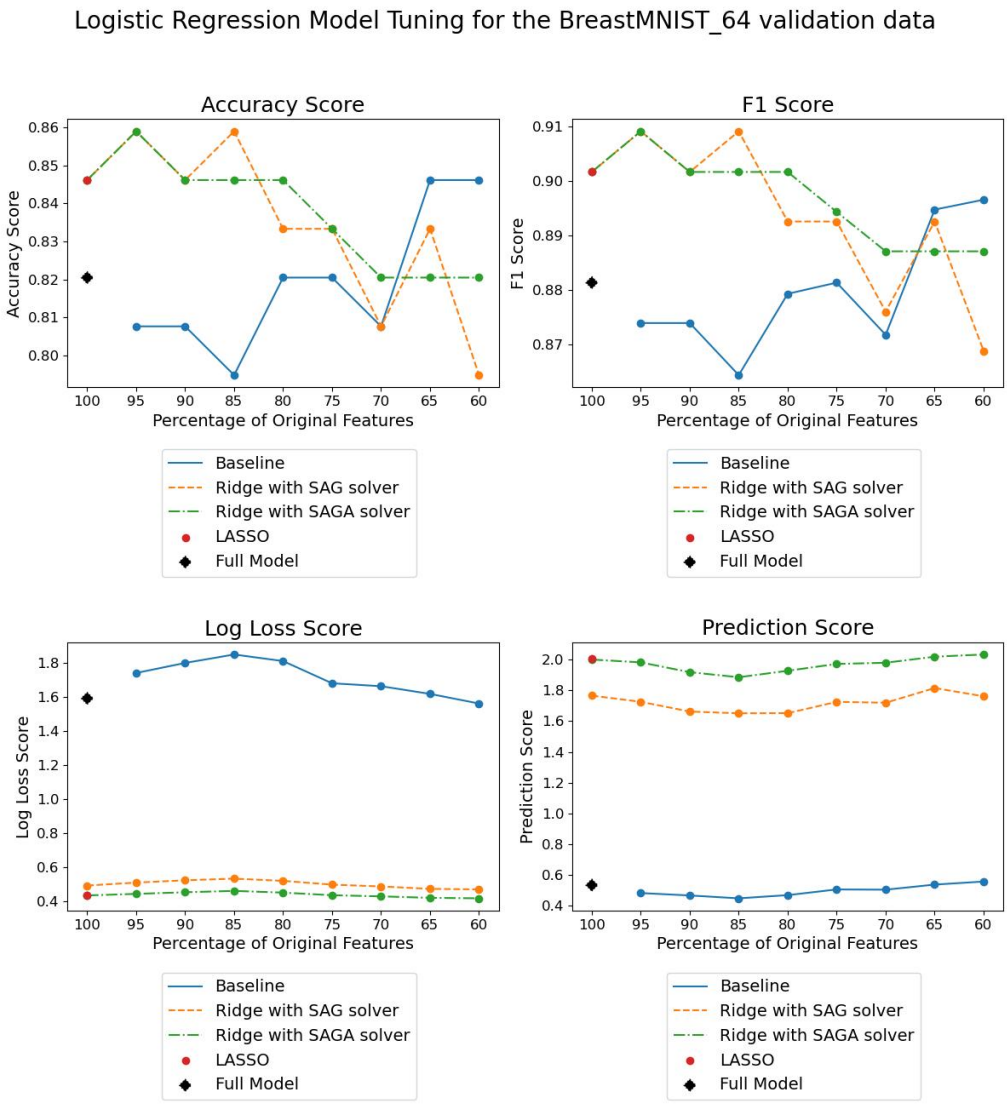


Table A.6.: Scores of Logistic Regression model tuning on the BreastMNIST dataset at 64×64 resolution

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	3.8s	0.821	0.881	1.594	0.534
95	N	2.7s	0.808	0.874	1.743	0.483
90	N	2.8s	0.808	0.874	1.801	0.467
85	N	2.4s	0.795	0.864	1.851	0.448
80	N	2.3s	0.821	0.879	1.813	0.469
75	N	2.2s	0.821	0.881	1.682	0.506
70	N	2.1s	0.808	0.872	1.665	0.504
65	N	2.0s	0.846	0.895	1.620	0.537
60	N	1.8s	0.846	0.897	1.563	0.557
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	9.2s	0.846	0.902	0.495	1.766
95	N	8.4s	0.859	0.909	0.512	1.726
90	N	8.0s	0.846	0.902	0.526	1.662
85	N	7.5s	0.859	0.909	0.535	1.651
80	N	7.3s	0.833	0.893	0.522	1.652
75	N	6.7s	0.833	0.893	0.500	1.726
70	N	6.2s	0.808	0.876	0.489	1.720
65	N	5.8s	0.833	0.893	0.475	1.815
60	N	5.3s	0.795	0.869	0.472	1.762
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	10.8s	0.846	0.902	0.437	2.001
95	N	10.3s	0.859	0.909	0.446	1.982
90	N	9.6s	0.846	0.902	0.455	1.919
85	N	9.1s	0.846	0.902	0.463	1.886
80	N	9.4s	0.846	0.902	0.453	1.928
75	N	8.1s	0.833	0.894	0.438	1.972
70	N	7.5s	0.821	0.887	0.431	1.980
65	N	7.3s	0.821	0.887	0.423	2.019
60	N	6.9s	0.821	0.887	0.420	2.037
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	21.2s	0.846	0.902	0.435	2.007

Figure A.7.: Plot of scores of Logistic Regression model tuning on the BreastMNIST dataset at 128×128 resolution

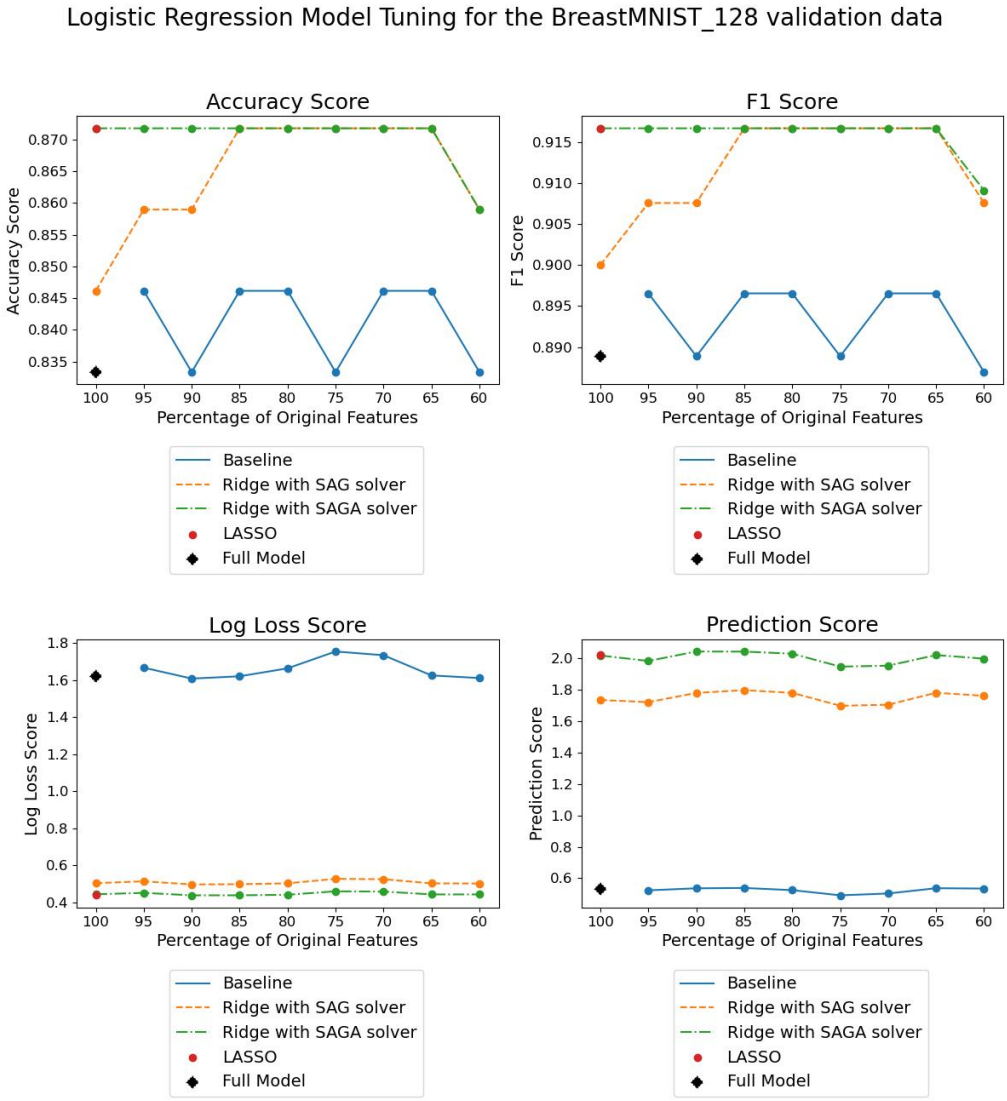


Table A.7.: Plot of scores of Logistic Regression model tuning on the BreastMNIST dataset at 128×128 resolution

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	11.7s	0.833	0.889	1.621	0.531
95	Y	10.3s	0.846	0.897	1.666	0.523
90	Y	9.5s	0.833	0.889	1.608	0.536
85	Y	8.4s	0.846	0.897	1.612	0.538
80	Y	8.9s	0.846	0.897	1.663	0.524
75	Y	7.8s	0.833	0.889	1.734	0.491
70	Y	6.9s	0.846	0.897	1.734	0.503
65	Y	6.7s	0.846	0.897	1.625	0.536
60	Y	5.9s	0.833	0.887	1.611	0.534
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	36.1s	0.846	0.900	0.503	1.736
95	N	34.2s	0.859	0.908	0.513	1.722
90	N	32.5s	0.859	0.908	0.496	1.781
85	N	30.8s	0.872	0.917	0.497	1.798
80	N	29.3s	0.872	0.917	0.502	1.781
75	N	27.2s	0.872	0.917	0.526	1.699
70	N	25.3s	0.872	0.917	0.524	1.705
65	N	23.5s	0.872	0.917	0.502	1.781
60	N	21.7s	0.859	0.908	0.501	1.762
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	43.3s	0.872	0.917	0.443	2.019
95	N	41.0s	0.872	0.917	0.451	1.984
90	N	38.6s	0.872	0.917	0.437	2.044
85	N	36.5s	0.872	0.917	0.438	2.044
80	N	34.4s	0.872	0.917	0.440	2.030
75	N	32.2s	0.872	0.917	0.459	1.948
70	N	29.8s	0.872	0.917	0.457	1.954
65	N	27.9s	0.872	0.917	0.442	2.022
60	N	26.0s	0.859	0.909	0.442	1.999
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1m 20s	0.872	0.917	0.441	2.026

Figure A.8.: Plot of scores of Logistic Regression model tuning on the BreastMNIST dataset at 224×224 resolution

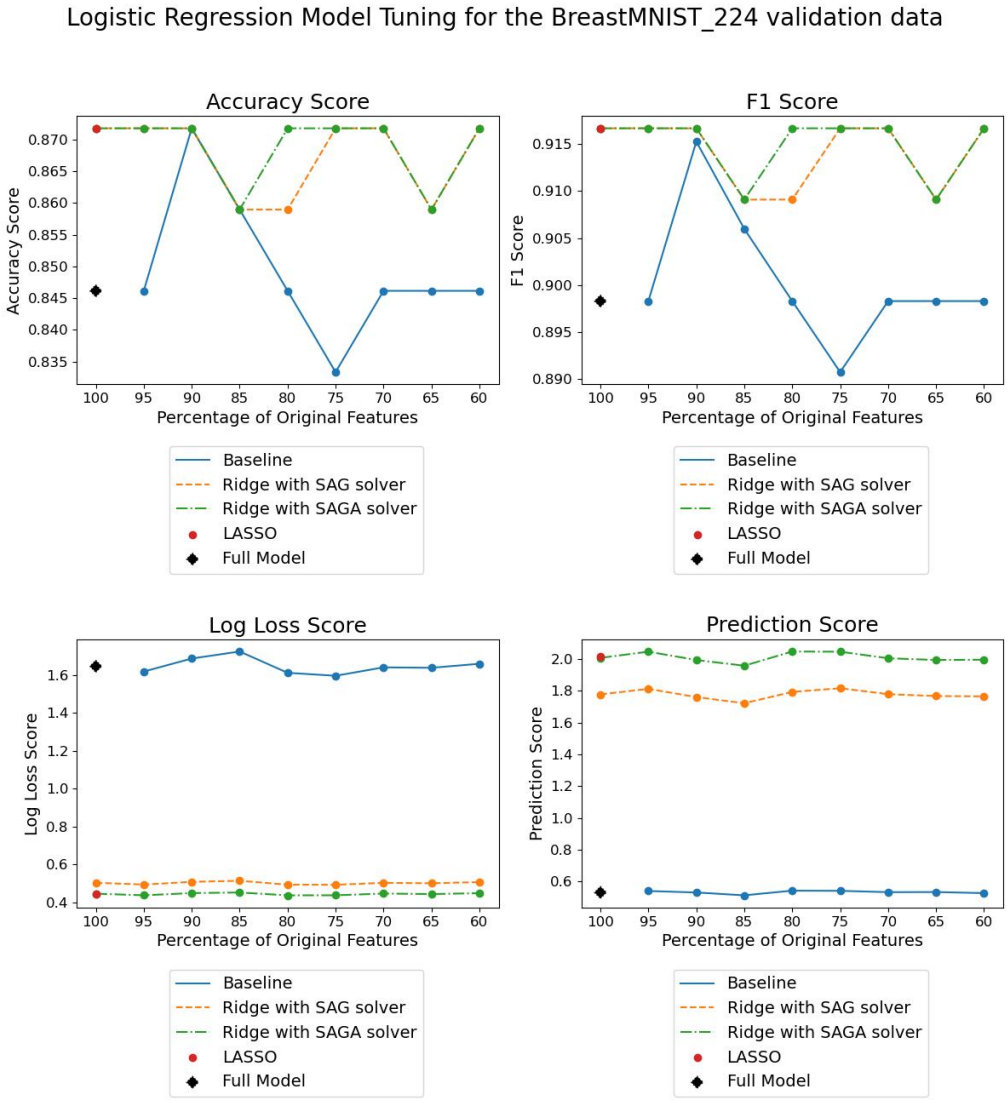


Table A.8.: Scores of Logistic Regression model tuning on the BreastMNIST dataset at 224×224 resolution

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	34.0s	0.846	0.898	1.645	0.530
95	Y	32.3s	0.846	0.898	1.617	0.539
90	Y	29.1s	0.872	0.915	1.686	0.530
85	Y	27.1s	0.850	0.906	1.723	0.512
80	Y	20.2s	0.846	0.898	1.611	0.542
75	Y	20.9s	0.833	0.891	1.595	0.540
70	Y	22.7s	0.846	0.898	1.639	0.532
65	Y	23.6s	0.846	0.898	1.637	0.533
60	Y	21.5s	0.846	0.898	1.658	0.526
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1m 51s	0.872	0.917	0.503	1.778
95	N	1m 45s	0.872	0.917	0.493	1.813
90	N	1m 40s	0.872	0.917	0.508	1.761
85	N	1m 34s	0.859	0.909	0.513	1.723
80	N	1m 29s	0.859	0.909	0.493	1.793
75	N	1m 23s	0.872	0.917	0.492	1.817
70	N	1m 18s	0.872	0.917	0.502	1.780
65	N	1m 12s	0.872	0.917	0.506	1.766
60						
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 12s	0.872	0.917	0.445	2.008
95	N	2m 5s	0.872	0.917	0.437	2.047
90	N	1m 59s	0.872	0.917	0.448	1.995
85	N	1m 52s	0.859	0.909	0.451	1.959
80	N	1m 45s	0.872	0.917	0.437	2.048
75	N	1m 39s	0.872	0.917	0.437	2.047
70	N	1m 32s	0.872	0.917	0.446	2.006
65	N	1m 26s	0.858	0.909	0.443	1.995
60	N	1m 19s	0.872	0.917	0.448	1.997
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	3m 59s	0.872	0.917	0.444	2.015

A.1.3. Convolutional Neural Network

The following figures display the results from training and tuning a Convolutional Neural Network for the classification of training examples using the BreastMNIST dataset at different resolutions.

Figure A.9.: Plot of log loss during training and validation of the Convolutional Neural Network model on the BreastMNIST dataset at 28×28 resolution in Grayscale

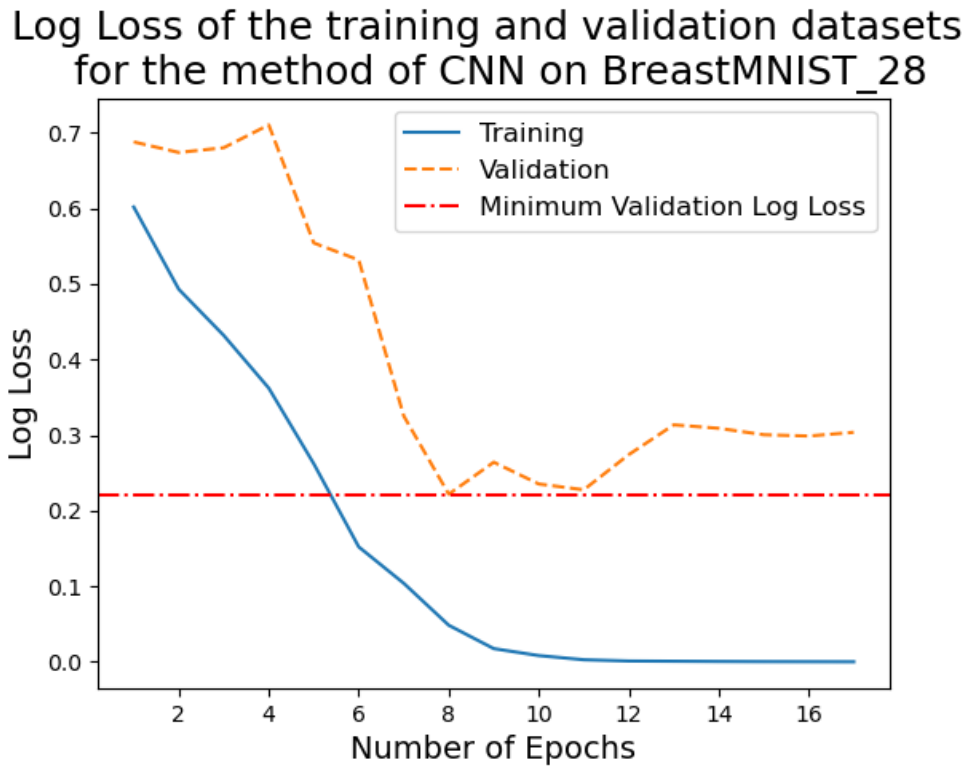


Figure A.10.: Plot of log loss during training and validation of the Convolutional Neural Network model on the BreastMNIST dataset at 64 × 64 resolution in Grayscale

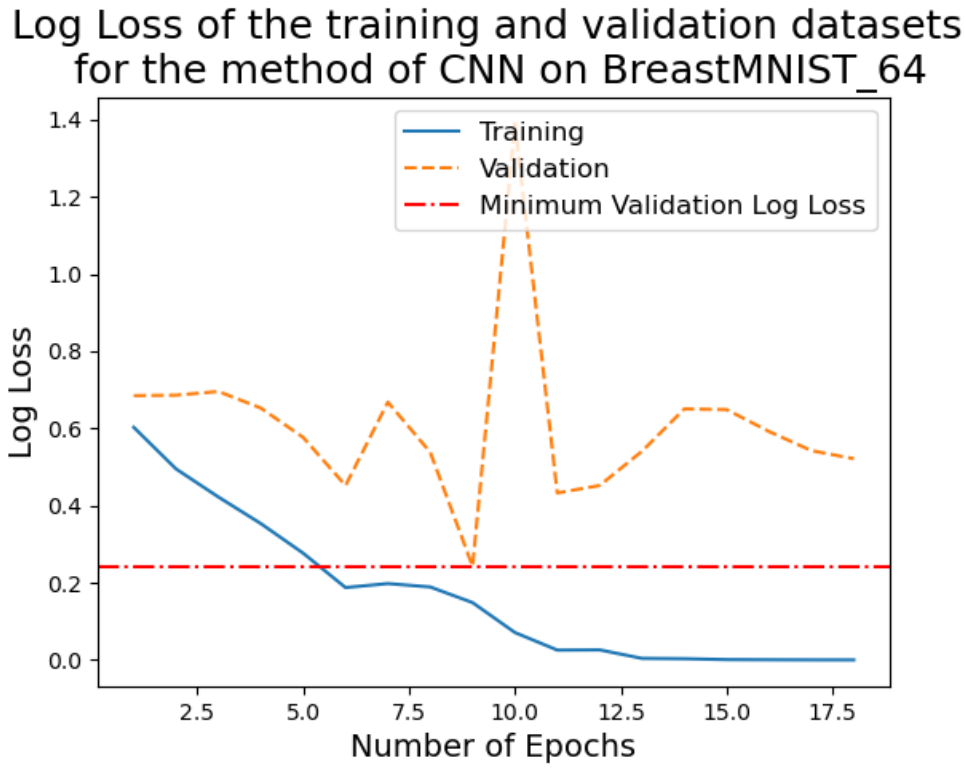


Figure A.11.: Plot of log loss during training and validation of the Convolutional Neural Network model on the BreastMNIST dataset at 128 × 128 resolution in Grayscale

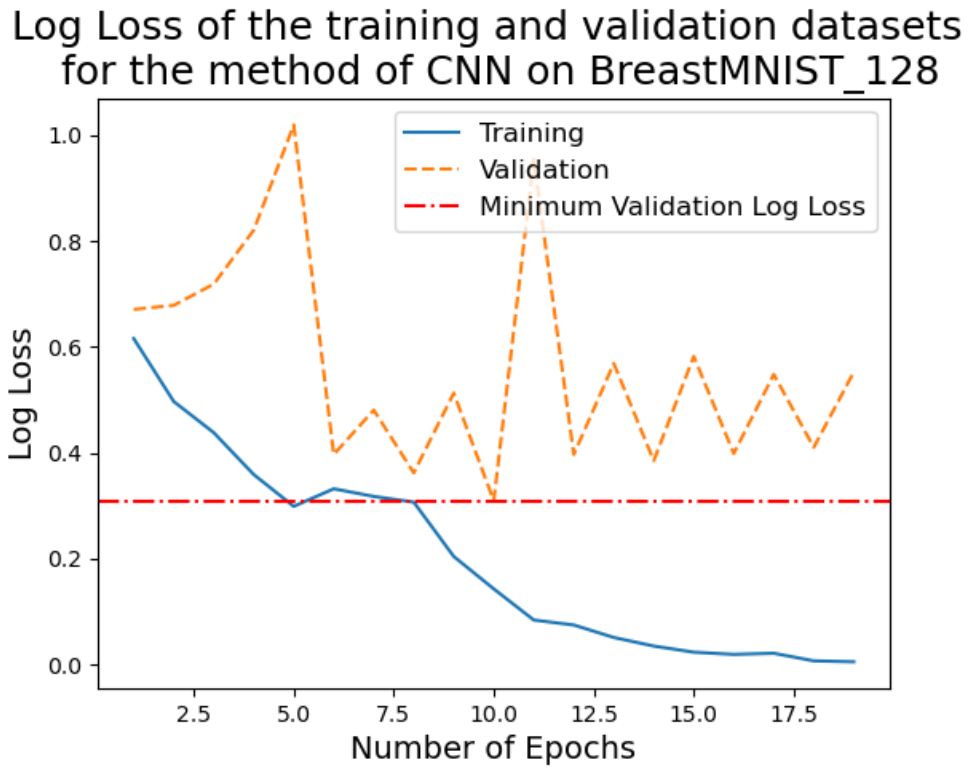
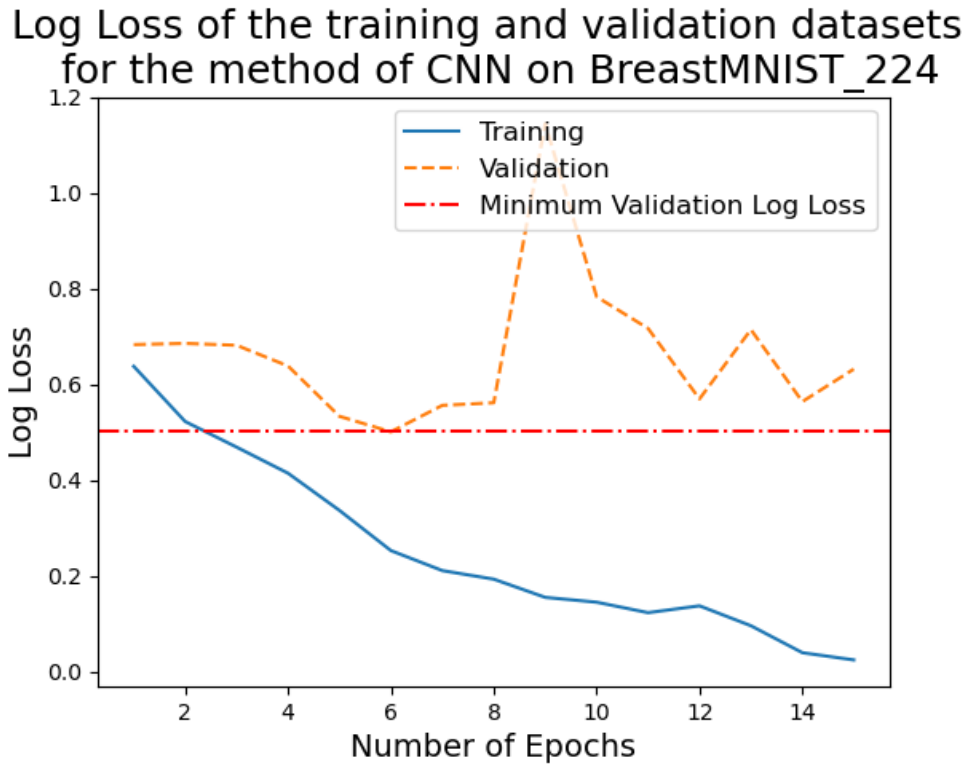


Figure A.12.: Plot of log loss during training and validation of the Convolutional Neural Network model on the BreastMNIST dataset at 224 × 224 resolution in Grayscale



A.1.4. XGBoost

The following figures display the results from training and tuning an XGBoost model for the classification of training examples using the BreastMNIST dataset at different resolutions.

Figure A.13.: Plot of log loss during training and validation of the XGBoost model on the BreastMNIST dataset at 28×28 resolution in Grayscale

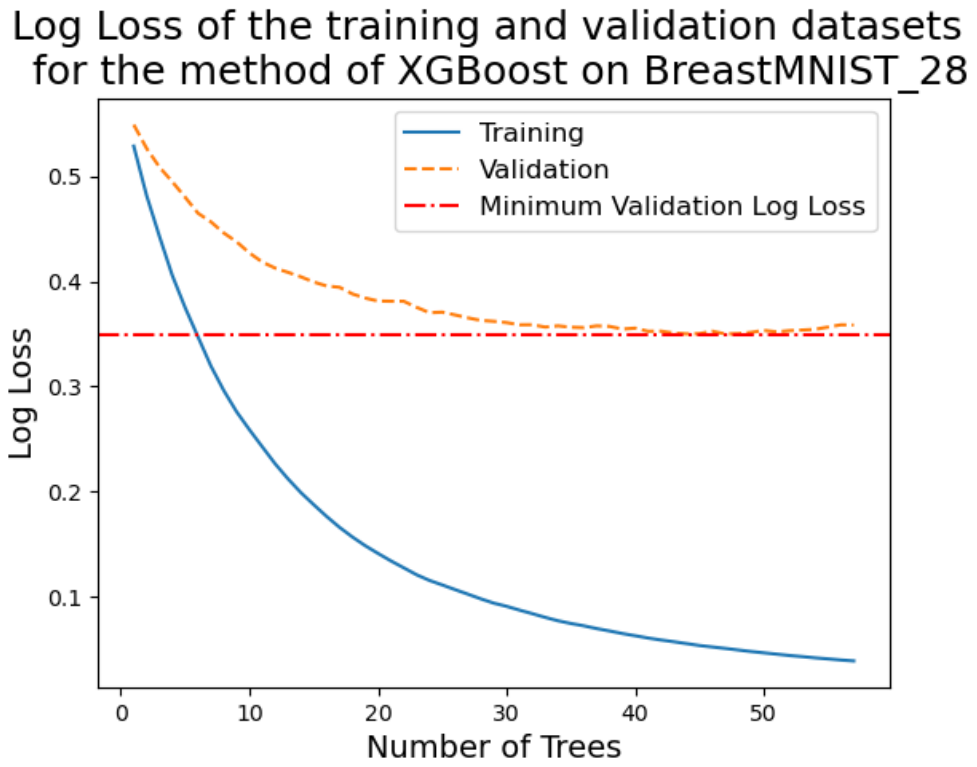


Figure A.14.: Plot of log loss during training and validation of the XGBoost model on the BreastMNIST dataset at 64×64 resolution in Grayscale

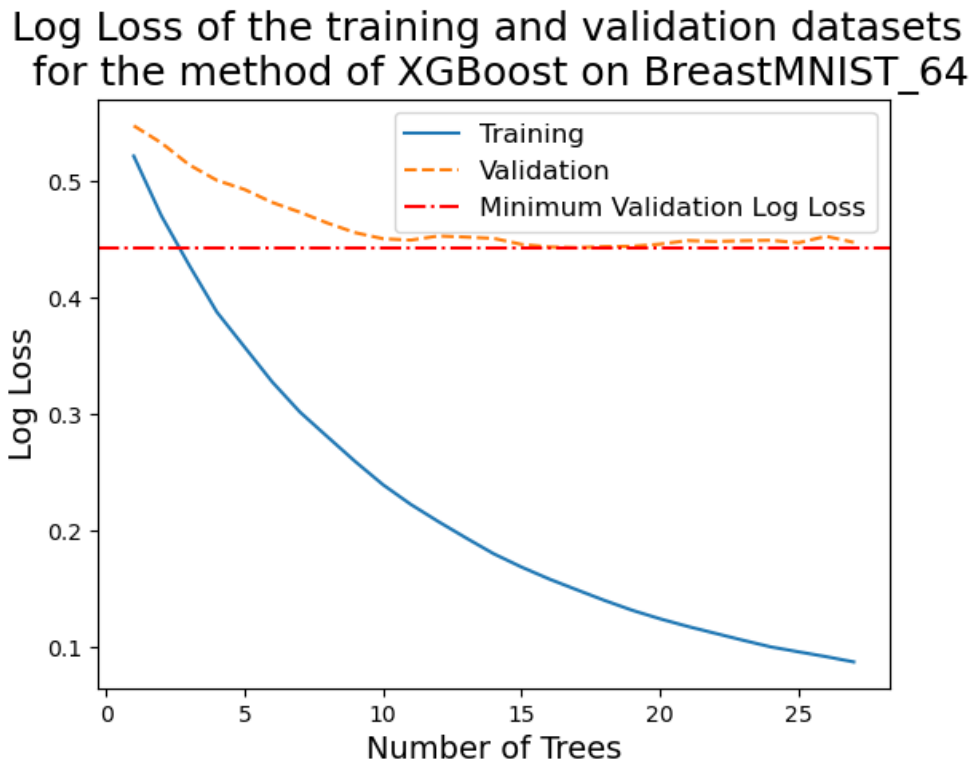


Figure A.15.: Plot of log loss during training and validation of the XGBoost model on the BreastMNIST dataset at 128 × 128 resolution in Grayscale

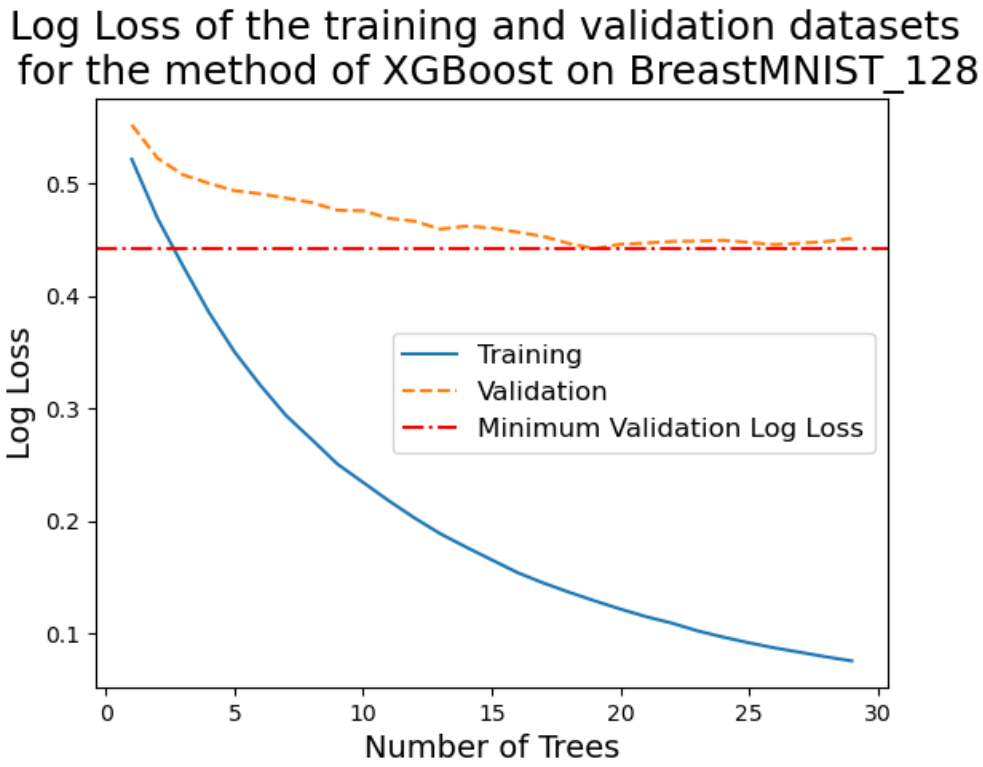
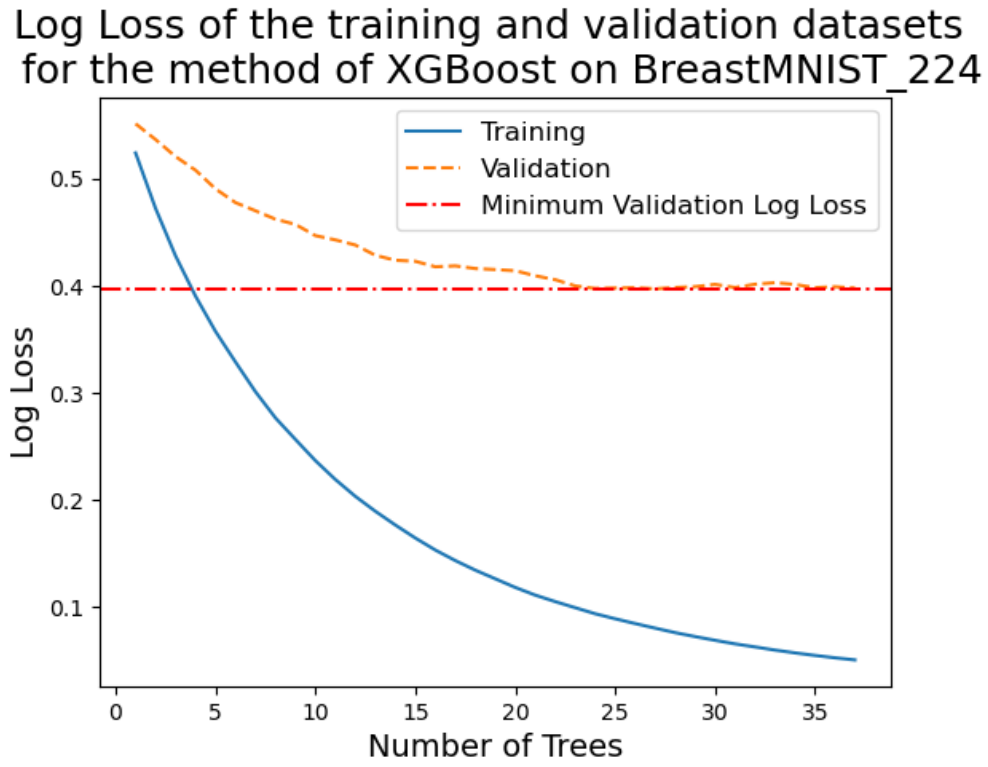


Figure A.16.: Plot of log loss during training and validation of the XGBoost model on the BreastMNIST dataset at 224 × 224 resolution in Grayscale



x

A.2. DermaMNIST

A.2.1. Linear Discriminant Analysis

Grayscale

The following tables display the results from training and tuning a Linear Discriminant Analysis model for the classification of training examples using the DermaMNIST dataset at different resolutions in Grayscale.

Figure A.17.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 28×28 resolution in Grayscale

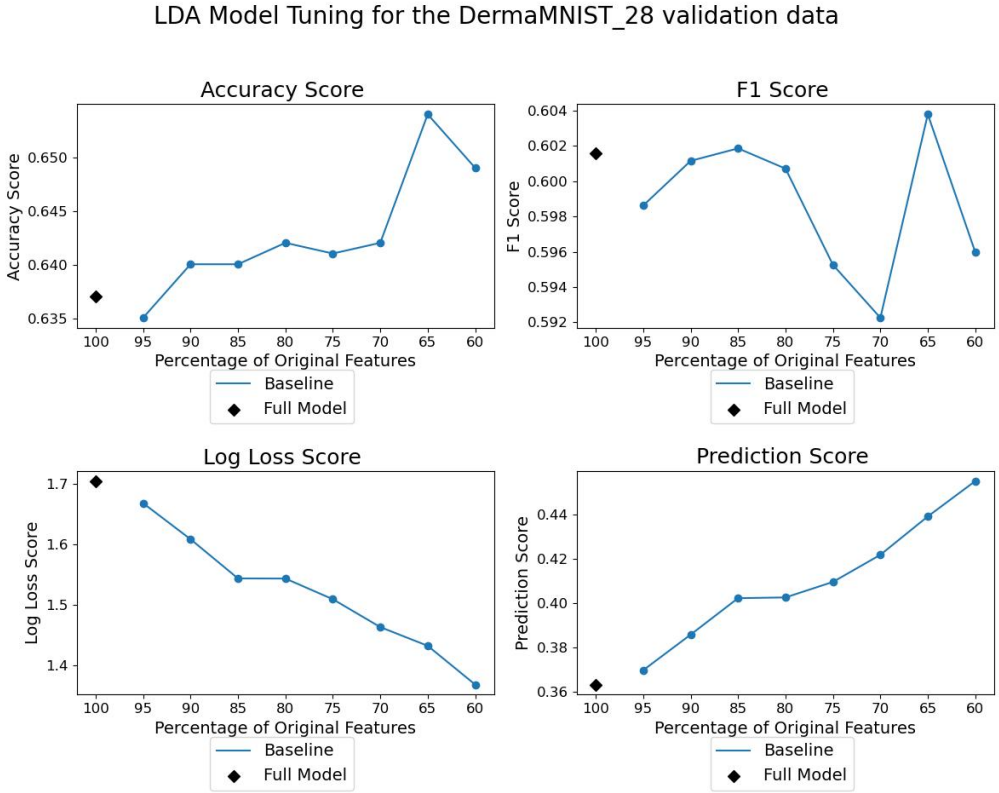


Table A.9.: Scores of LDA model tuning on the DermaMNIST dataset at 28×28 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	22.5s	0.637	0.602	1.705	0.363
95	Y	22.5s	0.635	0.599	1.668	0.370
90	Y	17.9s	0.640	0.601	1.608	0.386
85	Y	23.0s	0.640	0.602	1.543	0.402
80	Y	16.7s	0.642	0.601	1.543	0.402
75	Y	17.2s	0.641	0.595	1.509	0.410
70	Y	12.3s	0.642	0.592	1.462	0.422
65	Y	8.6s	0.654	0.604	1.432	0.439
60	Y	9.8s	0.649	0.596	1.367	0.455

Figure A.18.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 64×64 resolution in Grayscale

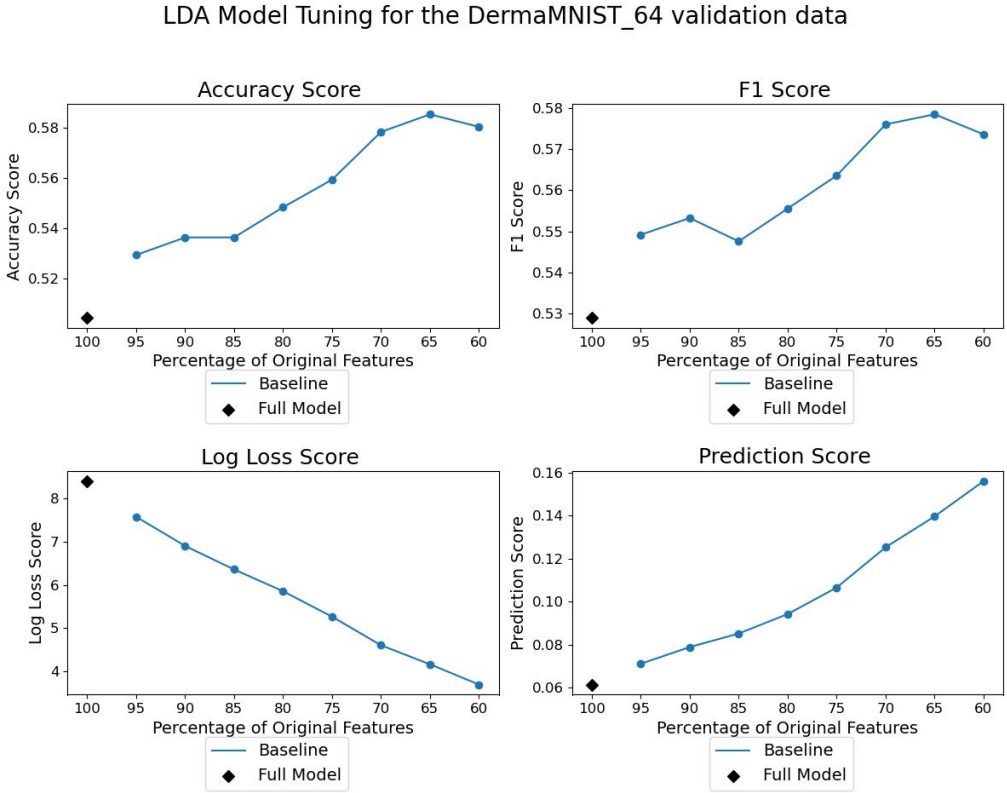


Table A.10.: Scores of LDA model tuning on the DermaMNIST dataset at 64×64 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	4m 18s	0.504	0.529	8.404	0.061
95	Y	3m 59s	0.529	0.549	7.579	0.071
90	Y	3m 48s	0.536	0.553	6.904	0.079
85	Y	3m 50s	0.536	0.548	6.360	0.085
80	Y	2m 39s	0.548	0.556	5.859	0.094
75	Y	3m 2s	0.559	0.564	5.267	0.107
70	Y	2m 46s	0.578	0.576	4.605	0.125
65	Y	2m 22s	0.585	0.576	4.165	0.140
60	Y	2m 1s	0.580	0.574	3.696	0.156

Figure A.19.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 128×128 resolution in Grayscale

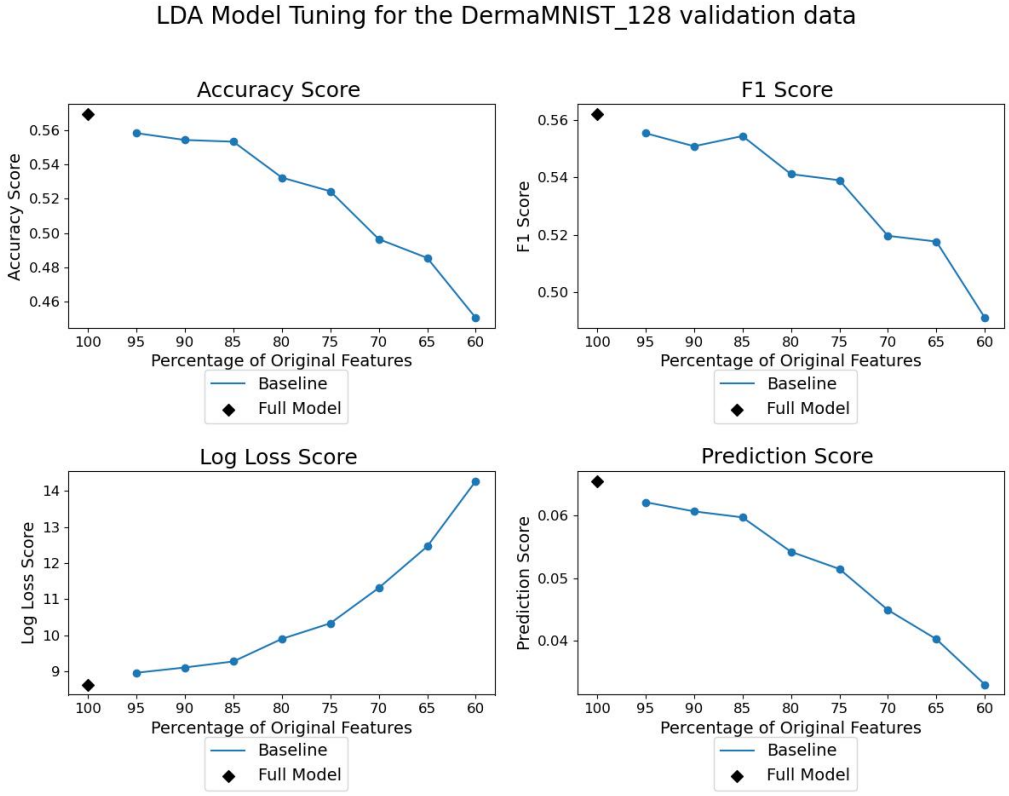


Table A.11.: Scores of LDA model tuning on the DermaMNIST dataset at 128×128 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	3m 28s	0.569	0.562	8.632	0.066
95	Y	3m 19s	0.558	0.555	8.959	0.062
90	Y	3m 23s	0.554	0.551	9.105	0.061
85	Y	3m 18s	0.553	0.554	9.273	0.060
80	Y	3m 16s	0.532	0.541	9.990	0.054
75	Y	4m 14s	0.524	0.539	10.330	0.051
70	Y	3m 54s	0.497	0.520	11.312	0.045
65	Y	3m 21s	0.486	0.518	12.461	0.040
60	Y	3m 17s	0.451	0.491	14.273	0.033

Figure A.20.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 224×224 resolution in Grayscale

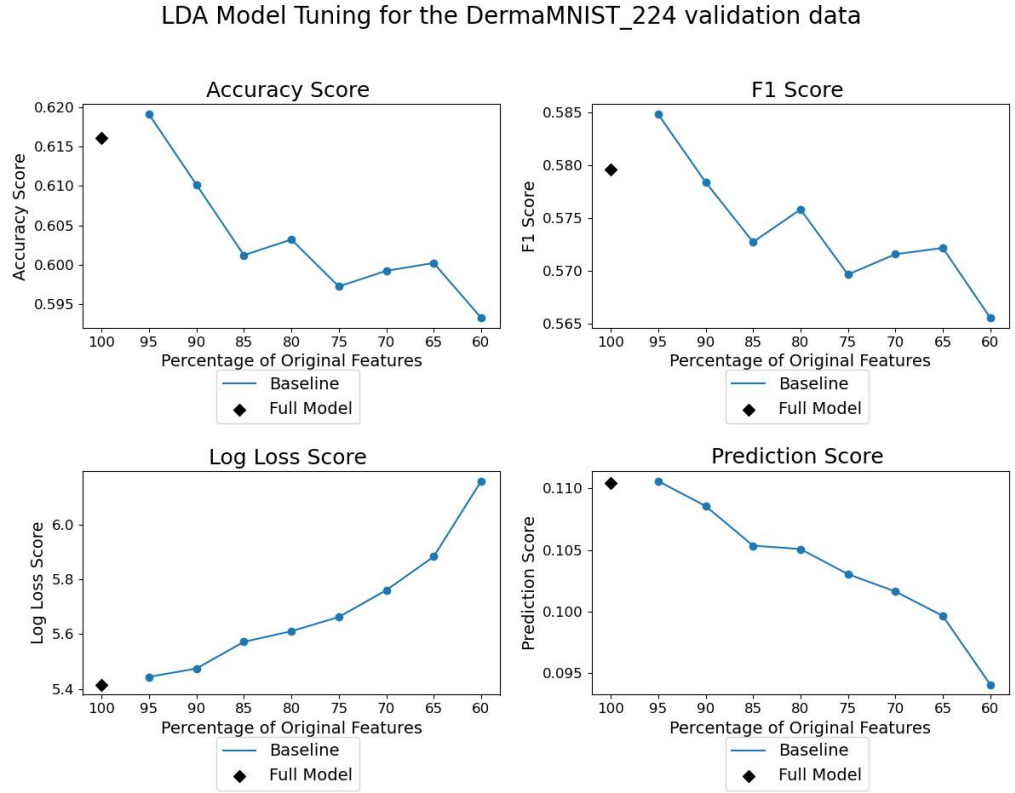


Table A.12.: Scores of LDA model tuning using the DermaMNIST dataset at 224×224 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	7m 13s	0.616	0.580	5.416	0.110
95	Y	7m 8s	0.619	0.585	5.444	0.111
90	Y	6m 35s	0.610	0.578	5.474	0.109
85	Y	7m 53s	0.601	0.573	5.572	0.105
80	Y	9m 30s	0.603	0.578	5.610	0.105
75	Y	9m 16s	0.597	0.570	5.662	0.103
70	Y	8m 38s	0.599	0.572	5.760	0.102
65	Y	7m 56s	0.600	0.572	5.883	0.100
60	Y	7m 20s	0.593	0.566	6.158	0.094

Colour (RGB)

The following tables display the results from training and tuning a Linear Discriminant Analysis model for the classification of training examples using the DermaMNIST dataset at different resolutions in the original colour.

Figure A.21.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 28×28 resolution in Colour (RGB)

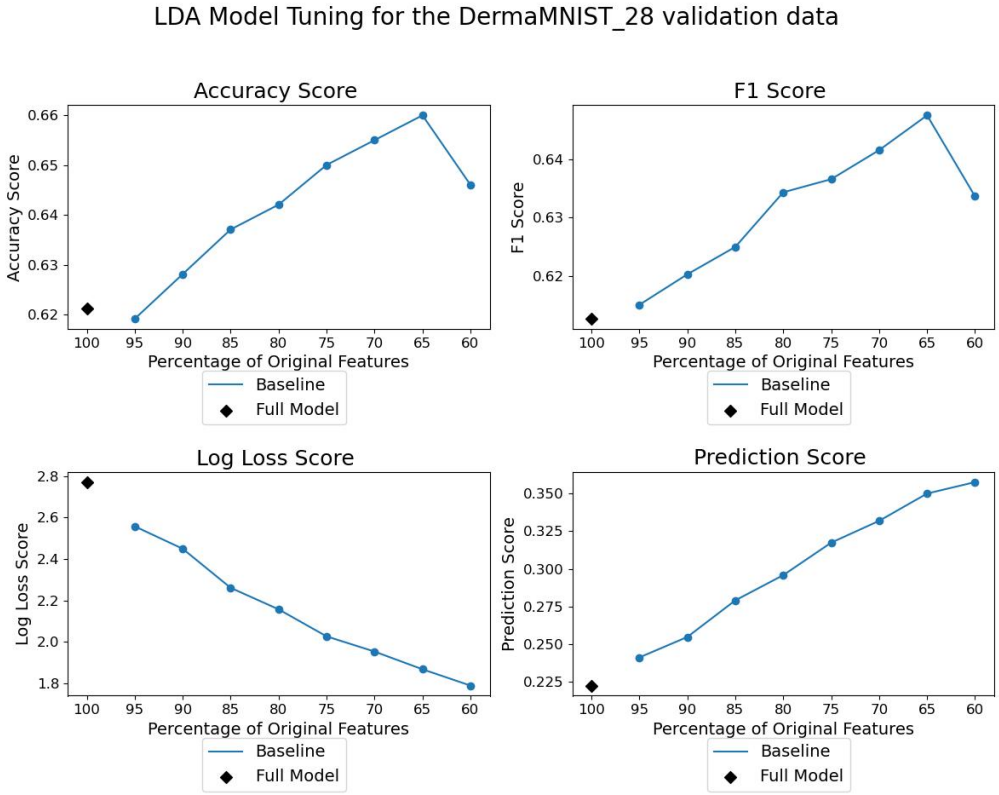


Table A.13.: Scores of LDA model tuning on the DermaMNIST dataset at 28×28 resolution in Colour (RGB)

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	13.0s	0.621	0.613	2.771	0.223
95	Y	10.9s	0.619	0.615	2.558	0.241
90	Y	11.1s	0.628	0.620	2.449	0.255
85	Y	11.4s	0.637	0.625	2.261	0.279
80	Y	8.9s	0.642	0.634	2.157	0.295
75	Y	9.3s	0.650	0.637	2.027	0.317
70	Y	7.4s	0.655	0.642	1.954	0.332
65	Y	6.2s	0.660	0.648	1.868	0.350
60	Y	5.9s	0.646	0.634	1.790	0.357

Figure A.22.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 64×64 resolution in Colour (RGB)

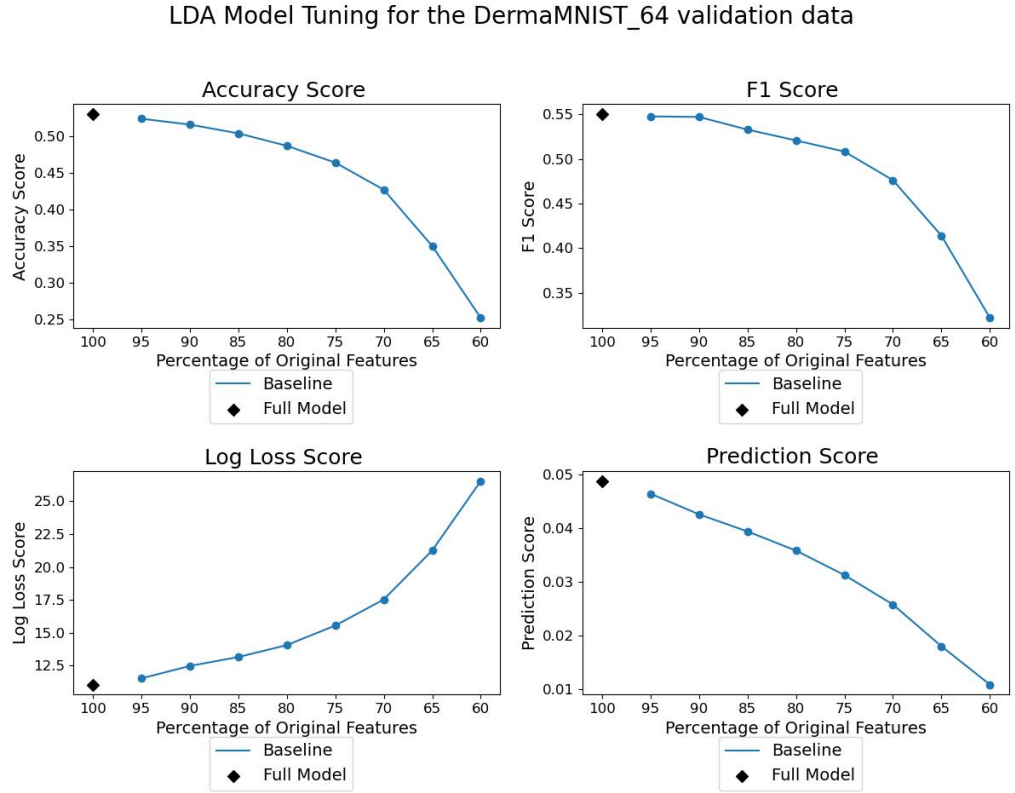


Table A.14.: Scores of LDA model tuning on the DermaMNIST dataset at 64×64 resolution in Colour (RGB)

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	3m 51s	0.529	0.550	11.070	0.049
95	Y	3m 58s	0.523	0.548	11.540	0.046
90	Y	3m 47s	0.515	0.547	12.484	0.042
85	Y	3m 40s	0.503	0.533	12.162	0.039
80	Y	3m 6s	0.487	0.521	14.067	0.036
75	Y	2m 52s	0.467	0.508	15.553	0.031
70	Y	2m 40s	0.427	0.476	17.431	0.026
65	Y	2m 14s	0.350	0.414	21.257	0.018
60	Y	2m 8s	0.252	0.322	26.521	0.011

Figure A.23.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 128 × 128 resolution in Colour (RGB)

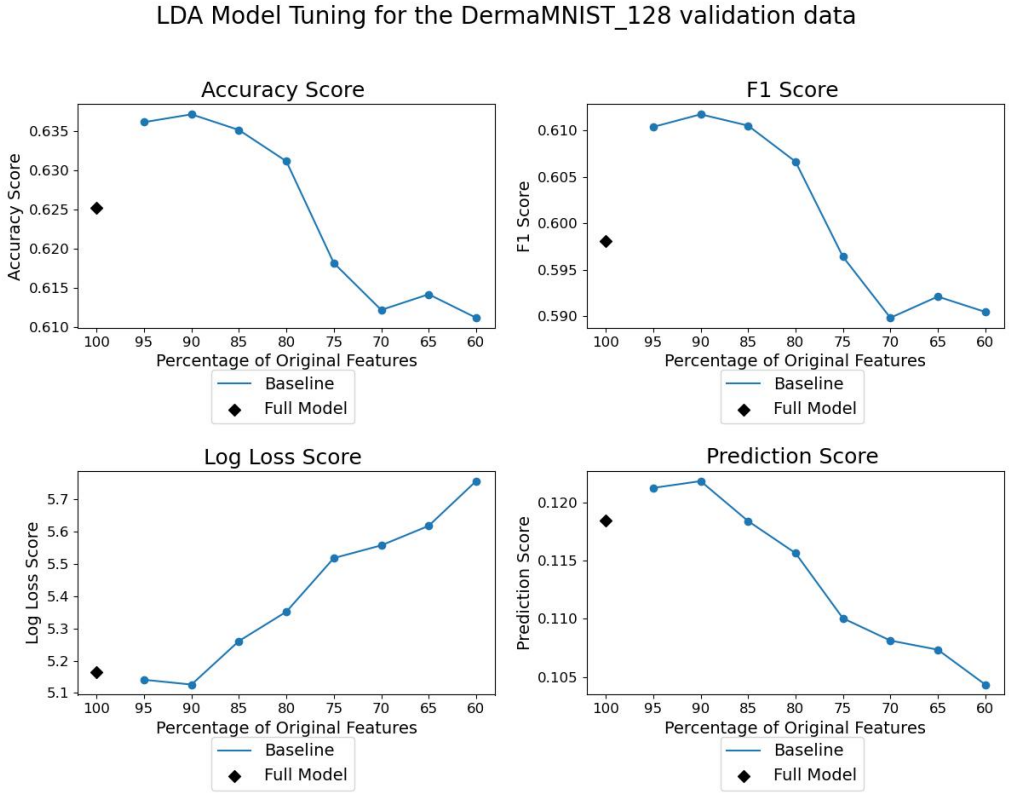


Table A.15.: Scores of LDA model tuning on the DermaMNIST dataset at 128 × 128 resolution in Colour (RGB)

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	8m 0s	0.625	0.598	5.165	0.118
95	Y	7m 13s	0.636	0.610	5.141	0.121
90	Y	9m 3s	0.637	0.612	5.126	0.122
85	Y	8m 33s	0.635	0.611	5.261	0.118
80	Y	7m 24s	0.631	0.607	5.352	0.116
75	Y	11m 0s	0.618	0.596	5.518	0.110
70	Y	10m 26s	0.612	0.590	5.557	0.108
65	Y	10m 5s	0.614	0.592	5.618	0.107
60	Y	7m 8s	0.611	0.590	5.757	0.104

Figure A.24.: Plot of scores of LDA model tuning on the DermaMNIST dataset at 224×224 resolution in Colour (RGB)

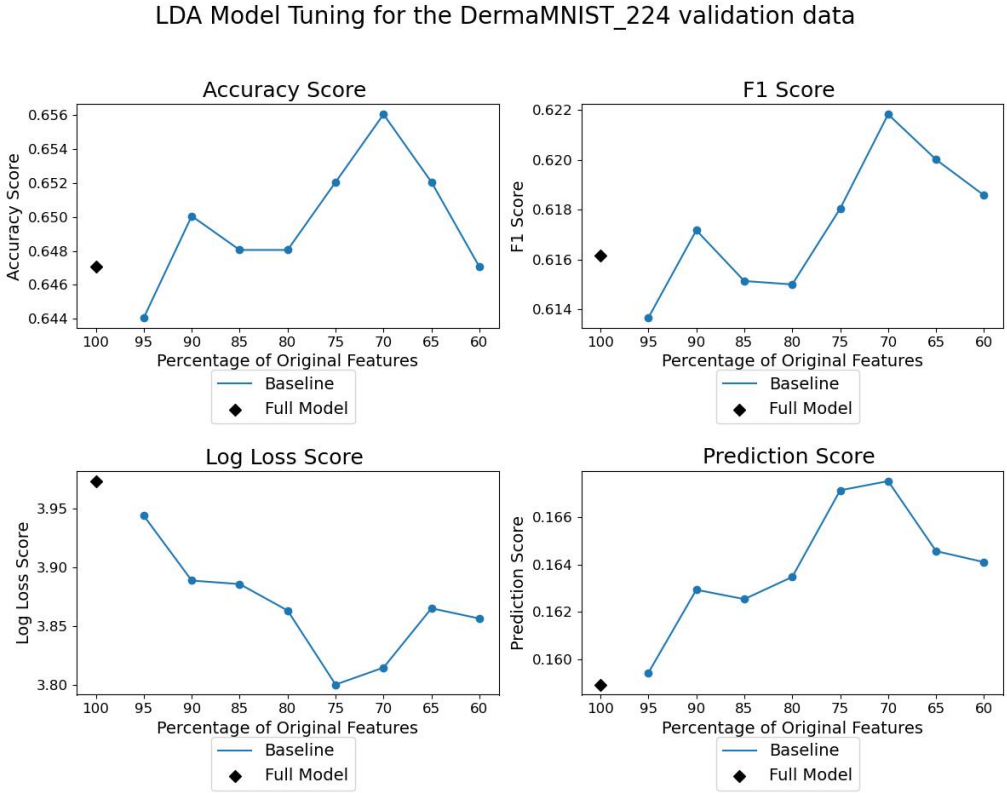


Table A.16.: Scores of LDA model tuning on the DermaMNIST dataset at 224×224 resolution in Colour (RGB)

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	24m 23s	0.647	0.616	3.973	0.159
95	Y	28m 11s	0.644	0.614	3.944	0.159
90	Y	25m 35s	0.650	0.617	3.889	0.163
85	Y	22m 10s	0.648	0.615	3.886	0.163
80	Y	22m 22s	0.648	0.615	3.863	0.163
75	Y	23m 36s	0.652	0.618	3.800	0.167
70	Y	19m 59s	0.656	0.622	3.815	0.167
65	Y	20m 58s	0.652	0.620	3.865	0.165
60	Y	18m 40s	0.647	0.619	3.856	0.164

A.2.2. Logistic Regression

Grayscale

The following tables and figures display the results from training and tuning a Logistic Regression model for the classification of training examples using the DermaMNIST dataset at different resolutions in Grayscale.

Figure A.25.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 28×28 resolution in Grayscale

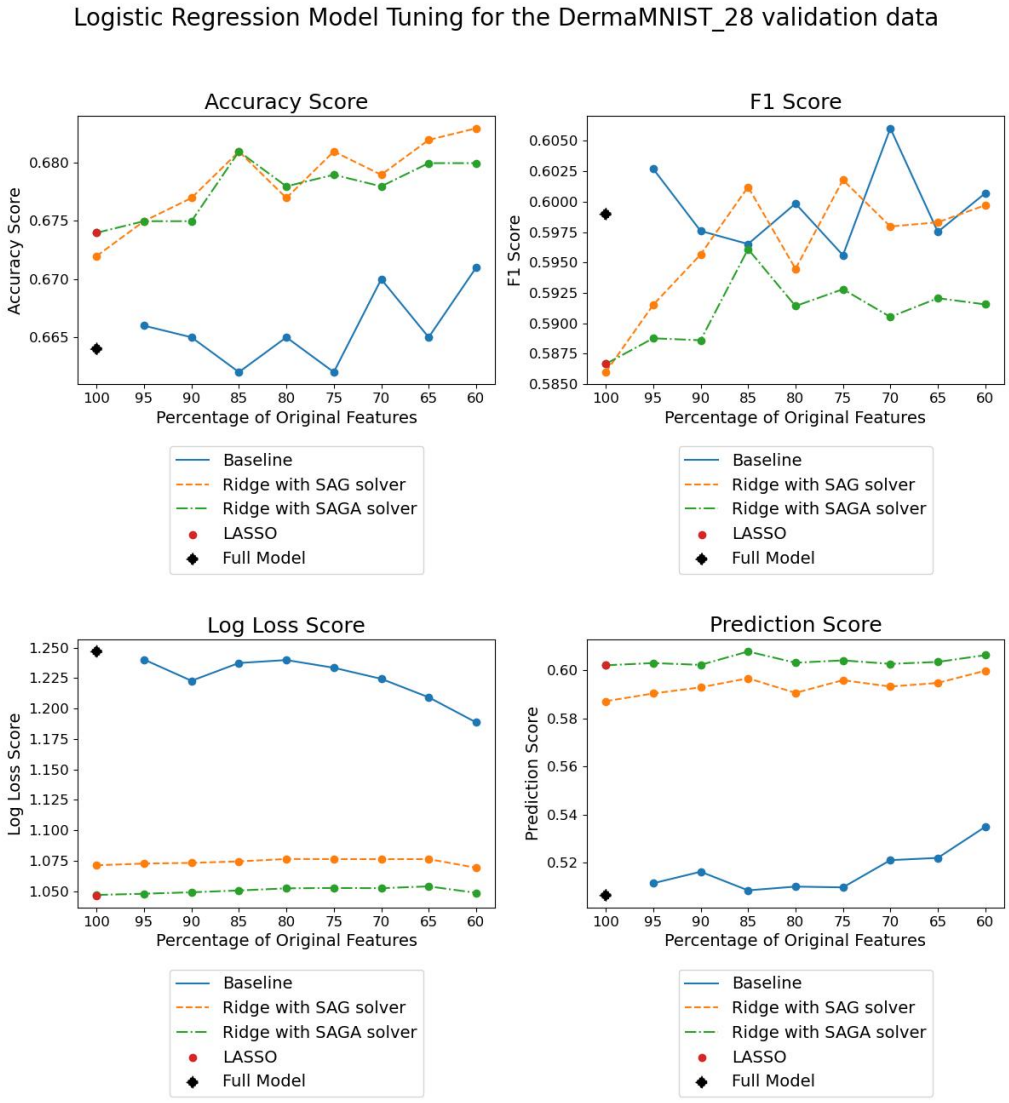


Table A.17.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 28×28 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	9.9s	0.664	0.599	1.247	0.506
95	N	8.4s	0.666	0.603	1.240	0.511
90	N	8.4s	0.665	0.598	1.223	0.516
85	N	7.8s	0.662	0.597	1.237	0.509
80	N	7.4s	0.665	0.600	1.240	0.510
75	N	7.7s	0.662	0.596	1.233	0.510
70	N	6.8s	0.670	0.606	1.224	0.521
65	N	6.7s	0.665	0.598	1.209	0.522
60	N	6.3s	0.670	0.601	1.189	0.535
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	41.8s	0.672	0.586	1.071	0.587
95	N	39.1s	0.675	0.592	1.073	0.590
90	N	38.2s	0.677	0.596	1.073	0.593
85	N	32.8s	0.681	0.601	1.075	0.597
80	N	34.0s	0.677	0.594	1.076	0.591
75	N	31.4s	0.681	0.602	1.076	0.596
70	N	29.2s	0.679	0.598	1.076	0.593
65	N	26.6s	0.682	0.598	1.076	0.595
60	N	25.2s	0.683	0.600	1.069	0.600
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	50.6s	0.674	0.587	1.047	0.602
95	N	45.3s	0.675	0.589	1.048	0.603
90	N	45.1s	0.675	0.589	1.049	0.602
85	N	41.2s	0.681	0.596	1.051	0.608
80	N	41.0s	0.678	0.591	1.052	0.603
75	N	38.0s	0.679	0.593	1.053	0.604
70	N	34.6s	0.678	0.591	1.053	0.603
65	N	34.9s	0.678	0.592	1.054	0.603
60	N	30.1s	0.678	0.592	1.049	0.606
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1m 35s	0.674	0.587	1.047	0.602

Figure A.26.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 64 × 64 resolution in Grayscale

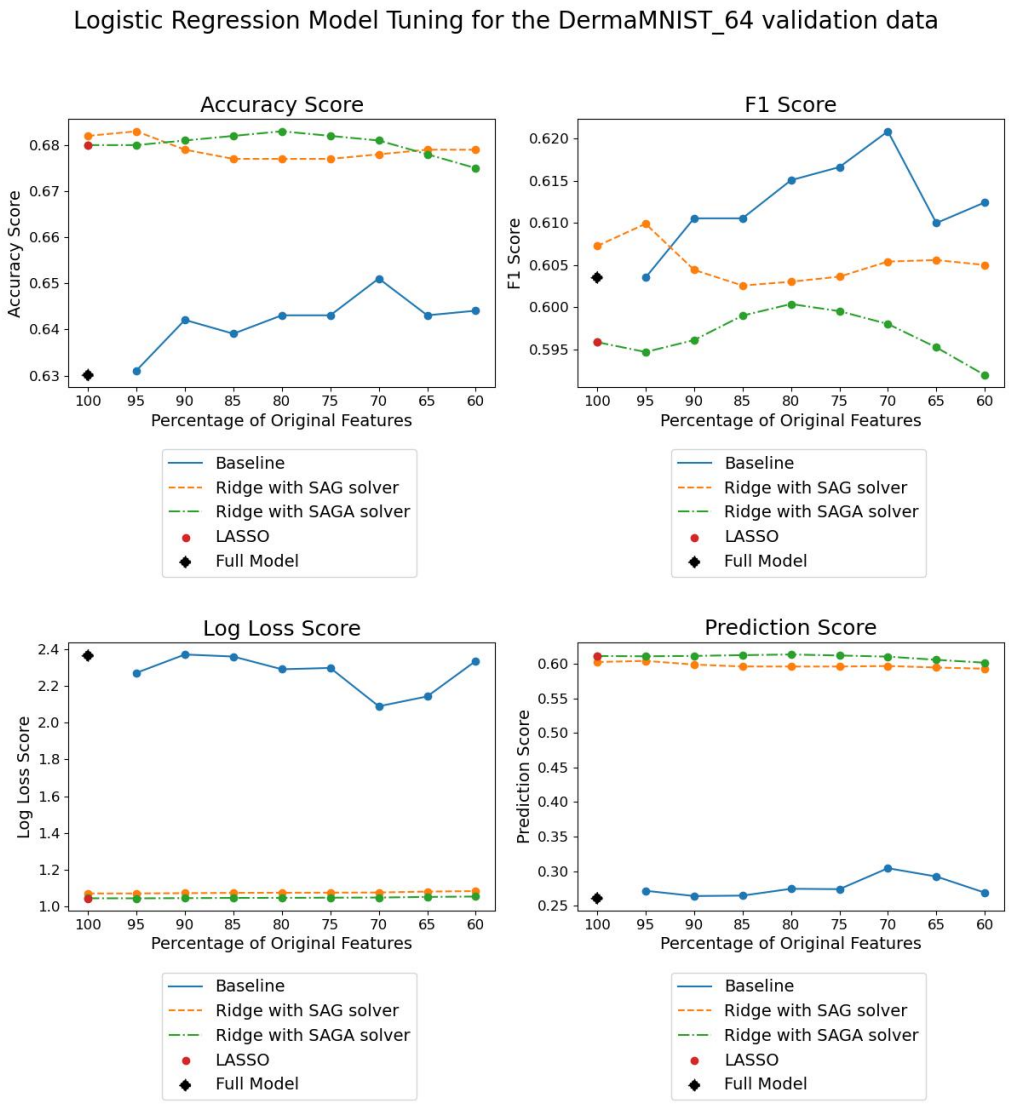


Table A.18.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 64×64 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	39.0s	0.630	0.604	2.366	0.261
95	N	36.6s	0.631	0.604	2.273	0.272
90	N	35.9s	0.642	0.611	2.373	0.264
85	N	34.2s	0.639	0.611	2.362	0.265
80	N	30.8s	0.643	0.615	2.292	0.274
75	N	30.4s	0.643	0.616	2.299	0.274
70	N	28.8s	0.651	0.621	2.090	0.304
65	N	26.5s	0.643	0.610	2.144	0.292
60	N	25.9s	0.644	0.612	2.337	0.269
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 14s	0.682	0.608	1.070	0.603
95	N	2m 9s	0.683	0.610	1.070	0.604
90	N	1m 55s	0.679	0.604	1.071	0.599
85	N	1m 49s	0.677	0.603	1.073	0.596
80	N	1m 43s	0.677	0.603	1.073	0.596
75	N	1m 38s	0.677	0.603	1.074	0.596
70	N	1m 32s	0.678	0.605	1.075	0.597
65	N	1m 25s	0.679	0.606	1.080	0.595
60	N	1m 18s	0.679	0.605	1.082	0.593
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 34s	0.680	0.596	1.043	0.611
95	N	2m 27s	0.680	0.595	1.043	0.611
90	N	2m 19s	0.681	0.596	1.044	0.612
85	N	2m 10s	0.682	0.599	1.045	0.613
80	N	2m 3s	0.683	0.600	1.046	0.614
75	N	1m 57s	0.682	0.600	1.047	0.612
70	N	1m 48s	0.681	0.598	1.047	0.611
65	N	1m 42s	0.678	0.595	1.051	0.606
60	N	1m 33s	0.675	0.592	1.053	0.602
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	5m 16s	0.680	0.596	1.043	0.612

Figure A.27.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 128×128 resolution in Grayscale

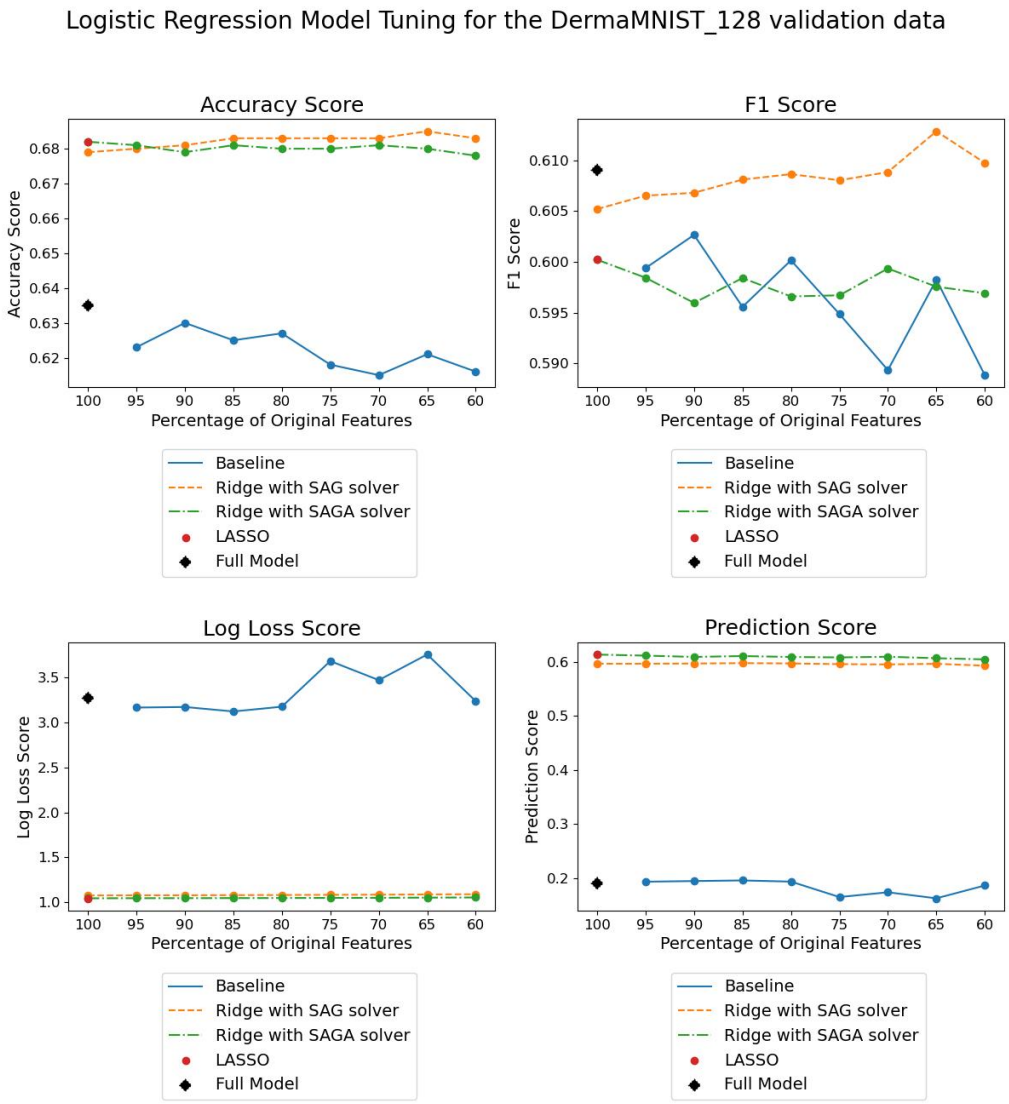


Table A.19.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 128 × 128 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 51s	0.635	0.609	3.273	0.190
95	N	2m 31s	0.623	0.599	3.167	0.193
90	N	2m 24s	0.630	0.603	3.173	0.194
85	N	2m 23s	0.625	0.596	3.123	0.195
80	N	2m 11s	0.627	0.600	3.176	0.193
75	N	2m 7s	0.618	0.595	3.684	0.165
70	N	1m 55s	0.615	0.589	3.472	0.173
65	N	1m 46s	0.621	0.598	3.757	0.162
60	N	1m 39s	0.616	0.589	3.239	0.186
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	8m 52s	0.679	0.605	1.077	0.596
95	N	8m 24s	0.680	0.607	1.079	0.596
90	N	7m 59s	0.681	0.607	1.079	0.597
85	N	7m 13s	0.683	0.608	1.080	0.597
80	N	6m 46s	0.683	0.609	1.082	0.597
75	N	6m 21s	0.683	0.608	1.084	0.596
70	N	5m 56s	0.683	0.609	1.085	0.595
65	N	5m 32s	0.685	0.613	1.088	0.596
60	N	5m 6s	0.683	0.608	1.090	0.593
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	10m 16s	0.682	0.600	1.045	0.613
95	N	9m 56s	0.681	0.598	1.046	0.611
90	N	9m 7s	0.679	0.596	1.047	0.609
85	N	8m 37s	0.681	0.598	1.048	0.611
80	N	8m 8s	0.680	0.597	1.048	0.609
75	N	7m 37s	0.680	0.597	1.050	0.608
70	N	7m 7s	0.681	0.599	1.051	0.609
65	N	6m 36s	0.680	0.598	1.053	0.607
60	N	6m 7s	0.680	0.597	1.055	0.604
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	20m 36s	0.682	0.600	1.045	0.613

Figure A.28.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 224×224 resolution in Grayscale

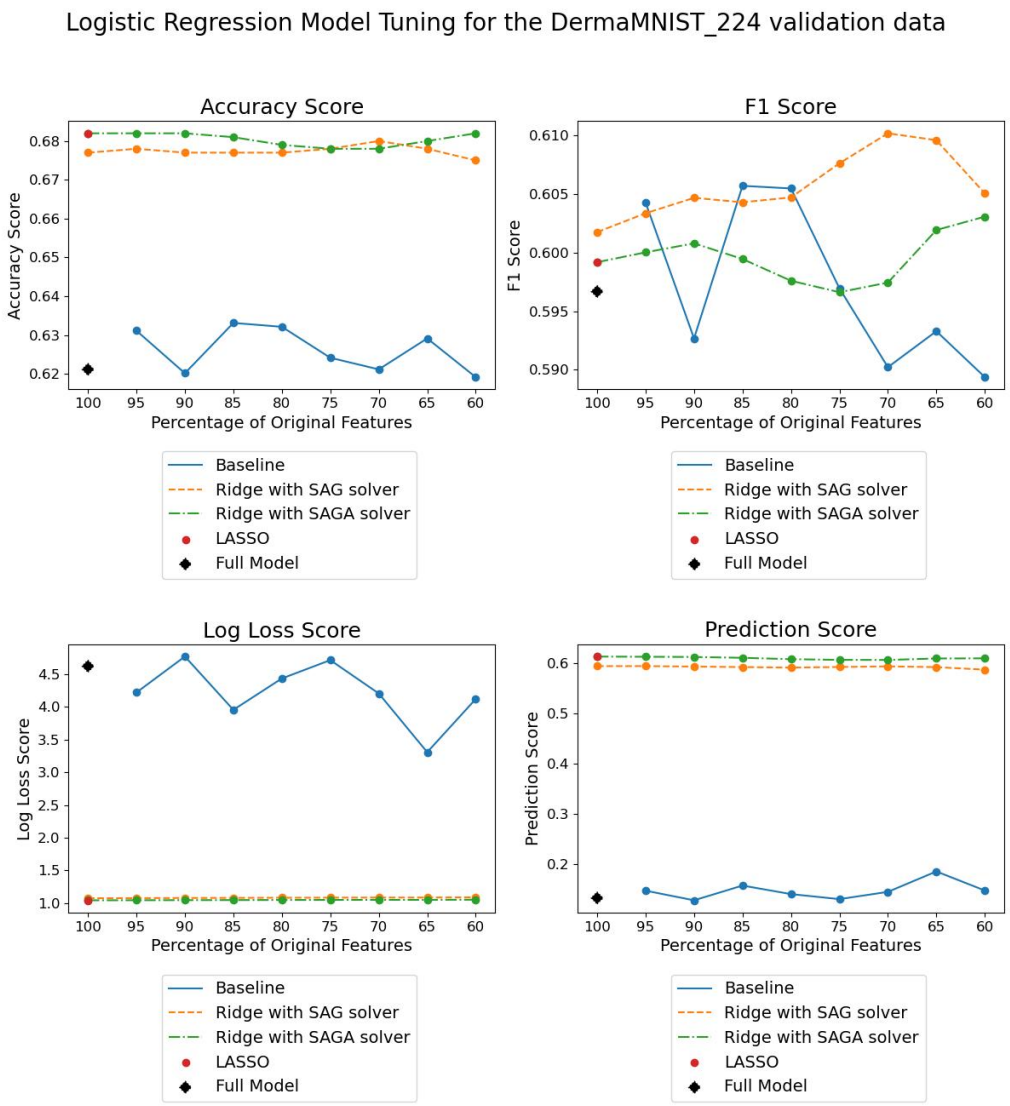


Table A.20.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 224 × 224 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	8m 50s	0.621	0.597	4.624	0.132
95	N	8m 9s	0.631	0.604	4.221	0.146
90	N	7m 59s	0.621	0.593	4.771	0.127
85	N	7m 45s	0.633	0.606	3.956	0.157
80	N	7m 43s	0.632	0.605	4.435	0.140
75	N	7m 5s	0.624	0.597	4.718	0.129
70	N	6m 30s	0.621	0.590	4.206	0.144
65	N	5m 55s	0.629	0.593	3.308	0.185
60	N	5m 9s	0.519	0.589	4.121	0.147
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	26m 9s	0.678	0.602	1.075	0.595
95	N	24m 44s	0.678	0.603	1.077	0.595
90	N	23m 23s	0.677	0.605	1.079	0.594
85	N	22m 9s	0.677	0.604	1.081	0.593
80	N	20m 49s	0.677	0.605	1.083	0.592
75	N	19m 28s	0.678	0.608	1.084	0.593
70	N	18m 5s	0.678	0.610	1.086	0.594
65	N	16m 57s	0.678	0.610	1.086	0.593
60	N	15m 45s	0.675	0.605	1.089	0.588
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	32m 54s	0.682	0.599	1.044	0.614
95	N	29m 49s	0.682	0.600	1.045	0.613
90	N	28m 33s	0.682	0.600	1.046	0.613
85	N	26m 53s	0.681	0.599	1.047	0.611
80	N	25m 12s	0.679	0.598	1.049	0.609
75	N	23m 52s	0.678	0.597	1.050	0.607
70	N	21m 53s	0.678	0.597	1.051	0.607
65	N	20m 18s	0.680	0.602	1.051	0.610
60	N	18m 42s	0.682	0.603	1.053	0.610
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1h 2m 0s	0.682	0.599	1.044	0.614

Colour (RGB)

The following tables and figures display the results from training and tuning a Logistic Regression model for the classification of training examples using the DermaMNIST dataset at different resolutions in the original colour.

Figure A.29.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 28×28 resolution in Colour (RGB)

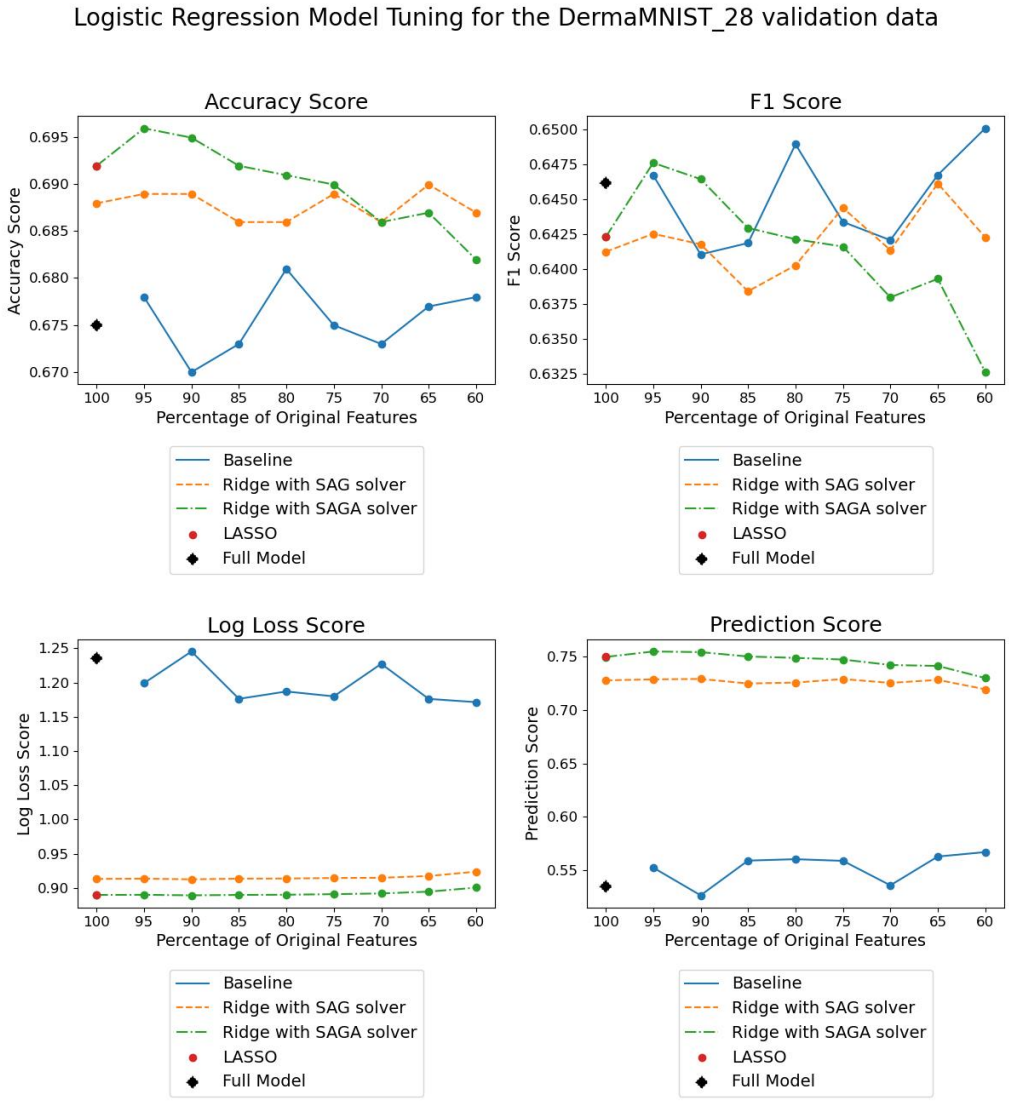


Table A.21.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 28×28 resolution in Colour (RGB)

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	22.7s	0.675	0.646	1.236	0.535
95	N	20.3s	0.678	0.547	1.199	0.552
90	N	18.9s	0.670	0.641	1.245	0.526
85	N	17.4s	0.673	0.642	1.176	0.559
80	N	16.7s	0.681	0.649	1.187	0.560
75	N	16.3s	0.675	0.643	1.180	0.559
70	N	15.3s	0.673	0.642	1.227	0.536
65	N	14.15s	0.678	0.648	1.176	0.563
60	N	13.3s	0.678	0.650	1.171	0.567
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1m 12s	0.688	0.641	0.913	0.728
95	N	1m 11s	0.689	0.643	0.913	0.729
90	N	1m 7s	0.689	0.642	0.912	0.729
85	N	1m 3s	0.686	0.638	0.913	0.725
80	N	1m 0s	0.686	0.640	0.914	0.723
75	N	54.6s	0.690	0.644	0.914	0.729
70	N	51.8s	0.686	0.641	0.915	0.726
65	N	48.7s	0.690	0.646	0.917	0.728
60	N	43.9s	0.687	0.642	0.924	0.720
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1m 25s	0.692	0.642	0.890	0.750
95	N	1m 23s	0.696	0.648	0.890	0.755
90	N	1m 17s	0.695	0.646	0.889	0.754
85	N	1m 14s	0.692	0.643	0.890	0.750
80	N	1m 6s	0.691	0.642	0.890	0.749
75	N	1m 2s	0.690	0.642	0.891	0.747
70	N	59.3s	0.686	0.638	0.892	0.742
65	N	54.3s	0.687	0.639	0.894	0.741
60	N	51.8s	0.682	0.633	0.900	0.730
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 53s	0.692	0.642	0.889	0.750

Figure A.30.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 64×64 resolution in Grayscale

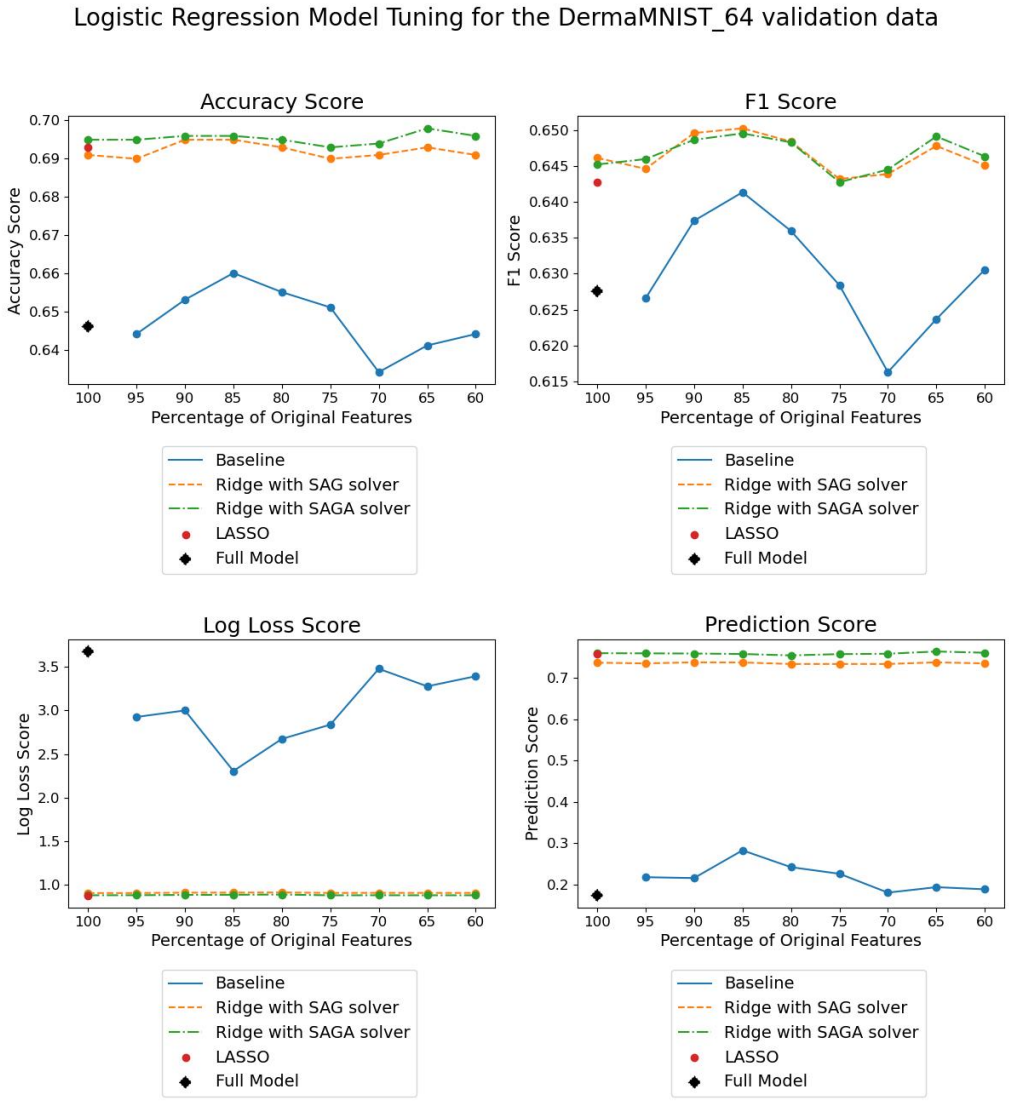


Table A.22.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 64 × 64 resolution in Colour (RGB)

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 5s	0.646	0.628	3.675	0.173
95	N	2m 2s	0.634	0.615	3.477	0.180
90	N	1m 54s	0.641	0.624	3.276	0.193
85	N	1m 39s	0.644	0.631	3.391	0.188
80	N	1m 38s	0.644	0.627	2.924	0.217
75	N	1m 29s	0.653	0.637	2.999	0.215
70	N	1m 28s	0.660	0.641	2.305	0.282
65	N	1m 16s	0.655	0.636	2.673	0.241
60	N	1m 9s	0.651	0.628	2.837	0.226
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	6m 3s	0.690	0.643	0.909	0.734
95	N	5m 44s	0.691	0.644	0.910	0.734
90	N	5m 25s	0.693	0.648	0.909	0.738
85	N	5m 8s	0.691	0.645	0.909	0.735
80	N	4m 51s	0.691	0.646	0.907	0.737
75	N	4m 31s	0.690	0.645	0.908	0.735
70	N	4m 15s	0.695	0.650	0.911	0.738
65	N	3m 58s	0.695	0.650	0.912	0.737
60	N	3m 38s	0.693	0.648	0.914	0.734
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	7m 9s	0.693	0.643	0.882	0.758
95	N	6m 50s	0.694	0.644	0.882	0.758
90	N	6m 22s	0.698	0.649	0.882	0.764
85	N	5m 59s	0.696	0.646	0.882	0.761
80	N	5m 42s	0.695	0.645	0.882	0.760
75	N	5m 25s	0.695	0.646	0.883	0.759
70	N	5m 3s	0.696	0.649	0.886	0.759
65	N	4m 42s	0.696	0.650	0.888	0.758
60	N	4m 20s	0.695	0.648	0.890	0.754
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	14m 45s	0.693	0.643	0.881	0.758

Figure A.31.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 128 × 128 resolution in Colour (RGB)

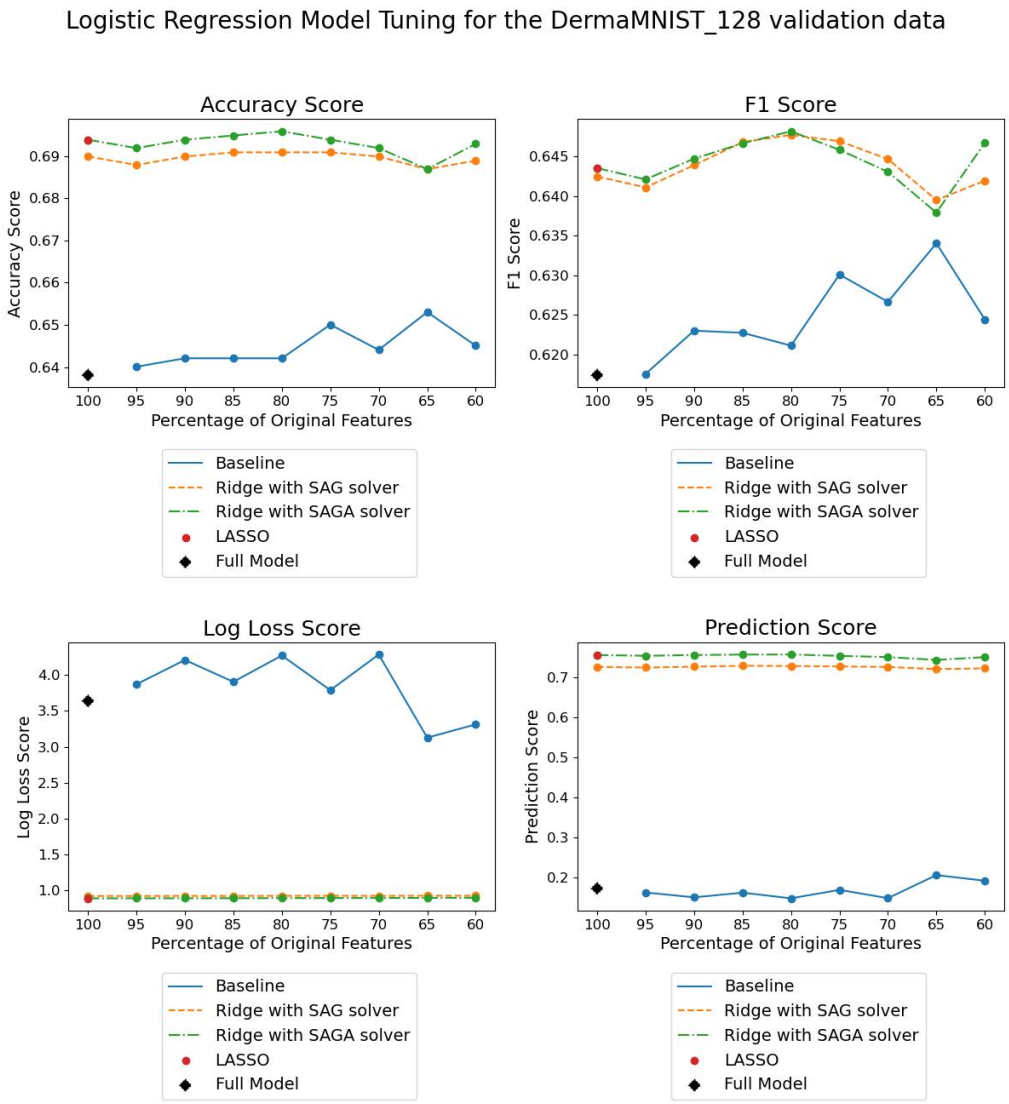


Table A.23.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 128 × 128 resolution in Colour (RGB)

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	8m 32s	0.638	0.617	3.640	0.172
95	N	8m 2s	0.640	0.618	3.872	0.162
90	N	7m 34s	0.642	0.623	4.211	0.150
85	N	7m 14s	0.642	0.623	3.906	0.162
80	N	6m 42s	0.642	0.621	4.274	0.148
75	N	6m 18s	0.650	0.630	3.790	0.169
70	N	5m 47s	0.644	0.627	4.288	0.148
65	N	5m 24s	0.653	0.634	3.127	0.206
60	N	5m 10s	0.645	0.624	3.312	0.192
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	29m 7s	0.690	0.642	0.918	0.726
95	N	29m 53s	0.688	0.641	0.918	0.724
90	N	26m 35s	0.690	0.643	0.918	0.726
85	N	29m 0s	0.691	0.647	0.918	0.728
80	N	27m 36s	0.691	0.648	0.920	0.727
75	N	26m 0s	0.691	0.647	0.920	0.727
70	N	24m 29s	0.690	0.645	0.920	0.726
65	N	22m 23s	0.687	0.639	0.921	0.720
60	N	20m 30s	0.689	0.642	0.922	0.722
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	43m 9s	0.694	0.644	0.885	0.755
95	N	39m 39s	0.692	0.642	0.885	0.753
90	N	37m 49s	0.694	0.645	0.886	0.755
85	N	33m 53s	0.695	0.647	0.887	0.756
80	N	22m 57s	0.696	0.648	0.888	0.757
75	N	22m 18s	0.694	0.646	0.889	0.753
70	N	19m 59s	0.692	0.643	0.890	0.750
65	N	18m 31s	0.687	0.638	0.892	0.743
60	N	17m 10s	0.693	0.647	0.893	0.750
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	56m 14s	0.694	0.644	0.885	0.756

Figure A.32.: Plot of scores of Logistic Regression model tuning on the DermaMNIST dataset at 224 × 224 resolution in Colour (RGB)

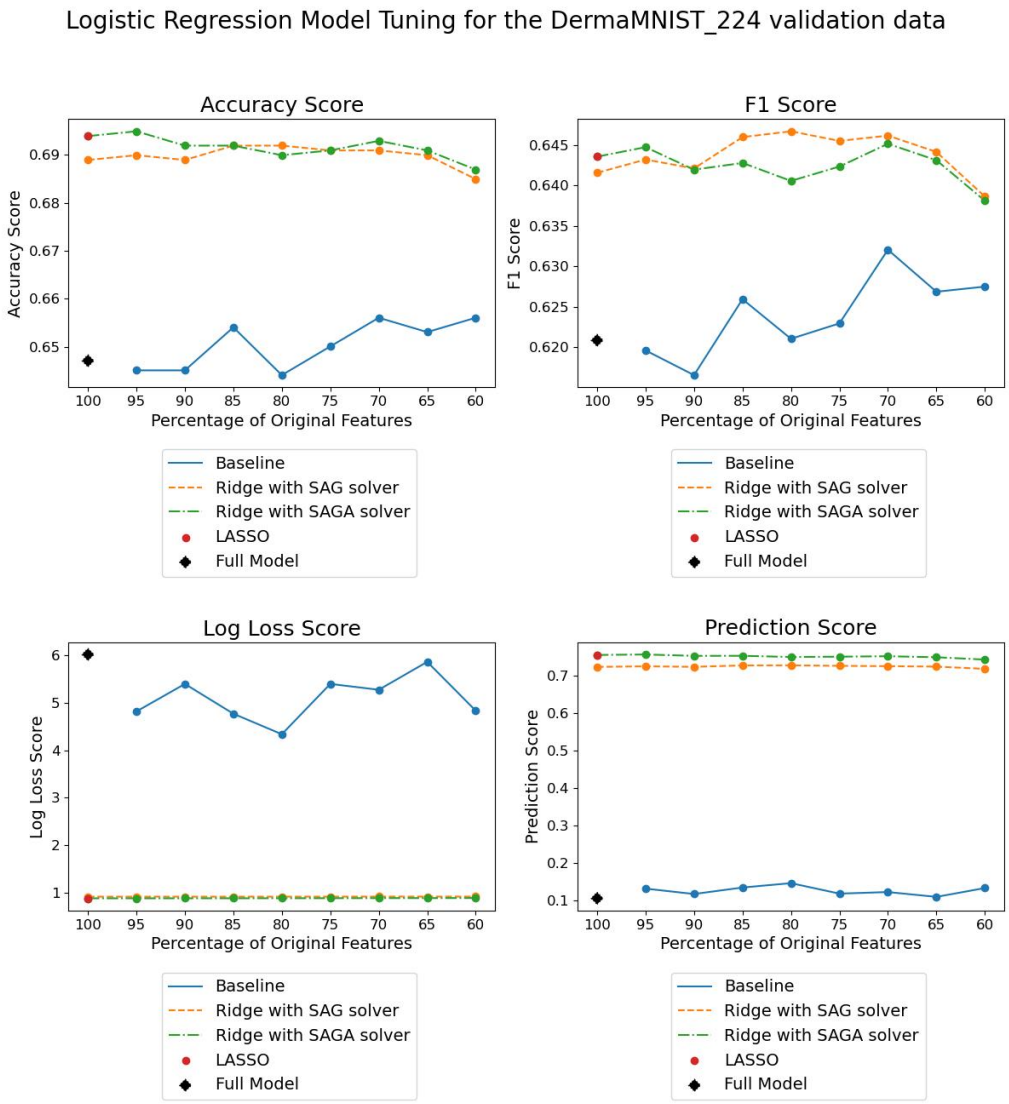


Table A.24.: Scores of Logistic Regression model tuning on the DermaMNIST dataset at 224 × 224 resolution in Colour (RGB)

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	30m 23s	0.647	0.621	6.013	0.105
95	N	31m 23s	0.645	0.620	4.815	0.131
90	N	29m 37s	0.645	0.617	5.394	0.117
85	N	33m 38s	0.654	0.623	4.767	0.134
80	N	26m 34s	0.644	0.621	4.336	0.146
75	N	27m 26s	0.650	0.623	5.395	0.118
70	N	21m 52s	0.656	0.632	5.271	0.122
65	N	19m 26s	0.653	0.627	5.858	0.109
60	N	17m 37s	0.656	0.628	4.832	0.133
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2h 4m 26s	0.689	0.642	0.920	0.723
95	N	1h 56m 14s	0.690	0.643	0.920	0.725
90	N	1h 48m 53s	0.690	0.642	0.920	0.723
85	N	1h 46m 12s	0.691	0.646	0.920	0.727
80	N	1h 43m 10s	0.692	0.647	0.920	0.727
75	N	1h 36m 45s	0.691	0.646	0.921	0.726
70	N	1h 28m 59s	0.691	0.646	0.922	0.725
65	N	1h 19m 56s	0.690	0.644	0.922	0.724
60	N	1h 12m 19s	0.685	0.639	0.922	0.718
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2h 11m 37s	0.694	0.644	0.886	0.755
95	N	1h 34m 41s	0.695	0.645	0.886	0.756
90	N	1h 28m 16s	0.692	0.642	0.886	0.752
85	N	1h 22m 6s	0.692	0.643	0.887	0.753
80	N	1h 17m 6s	0.690	0.641	0.888	0.750
75	N	1h 11m 59s	0.691	0.642	0.888	0.750
70	N	1h 5m 42s	0.693	0.645	0.890	0.752
65	N	1h 1m 19s	0.691	0.643	0.891	0.749
60	N	58m 15s	0.687	0.638	0.892	0.743
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	3h 3m 43s	0.694	0.644	0.886	0.755

A.2.3. Convolutional Neural Network

Grayscale

The following figures display the results from training and tuning a Convolutional Neural Network model for the classification of training examples using the DermaMNIST dataset at different resolutions in Grayscale.

Figure A.33.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 28×28 resolution in Grayscale

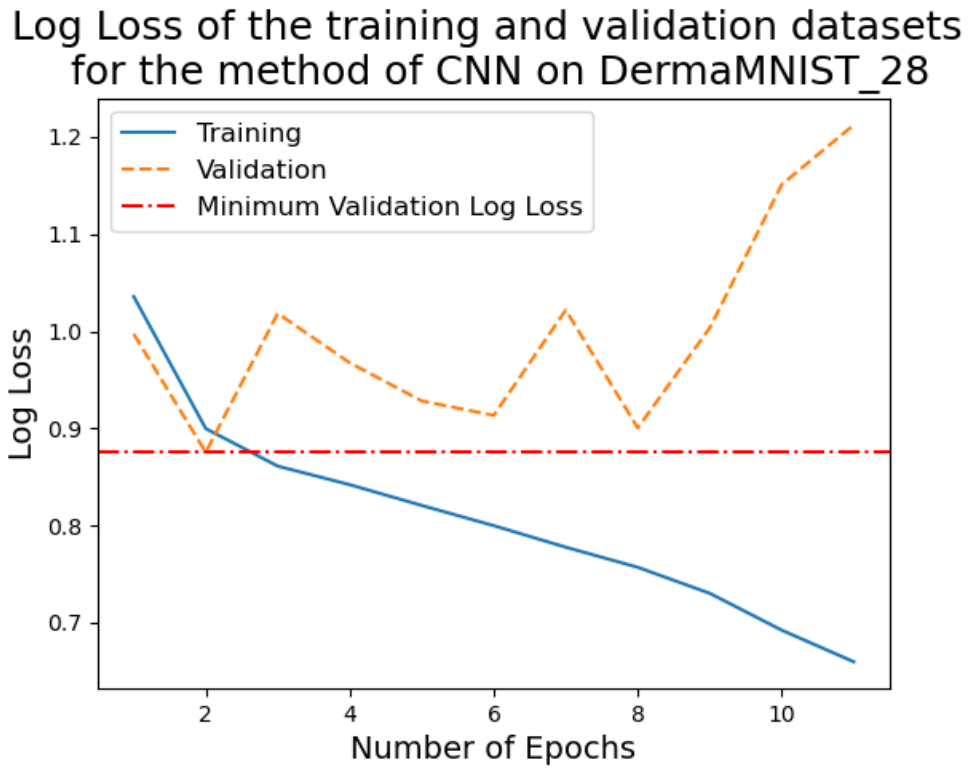


Figure A.34.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 64×64 resolution in Grayscale

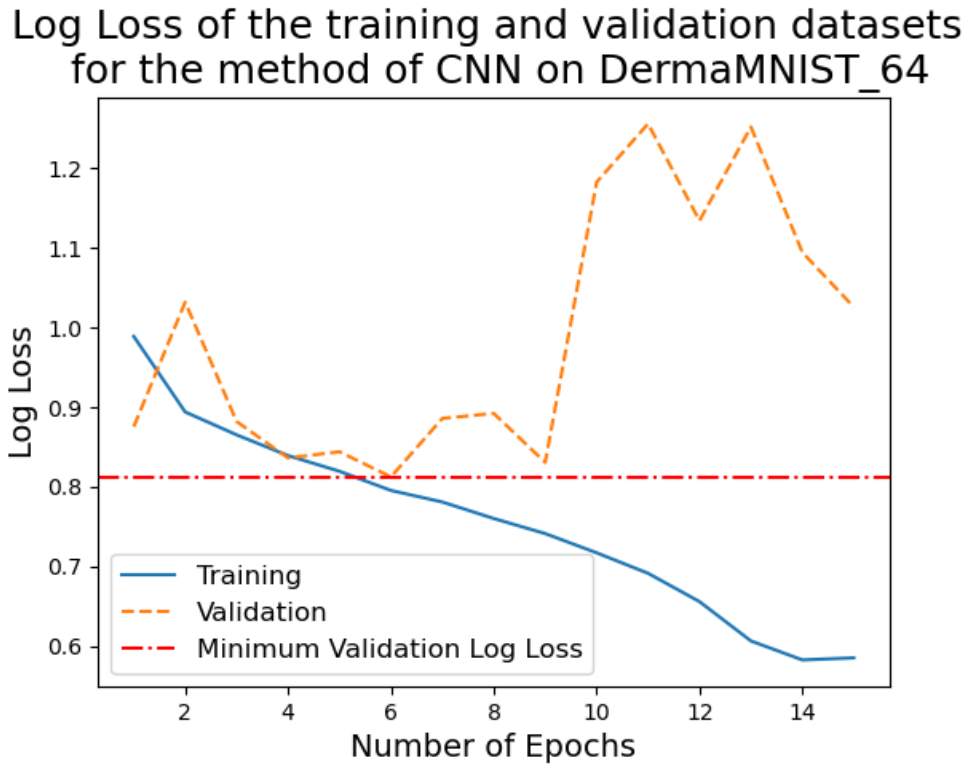


Figure A.35.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 128 × 128 resolution in Grayscale

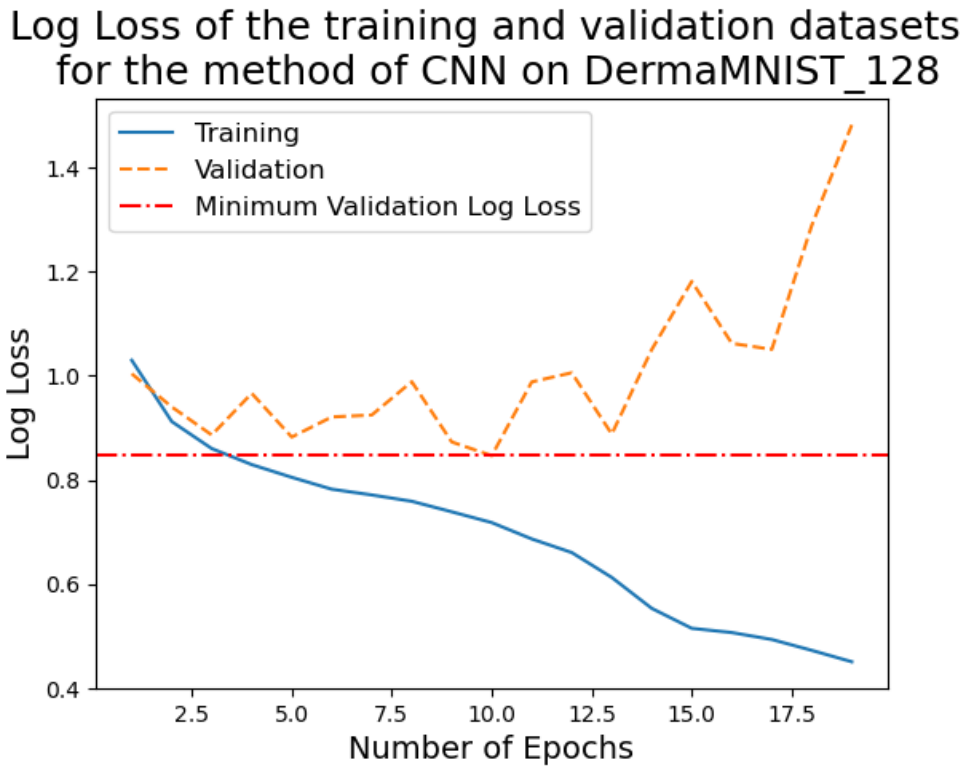
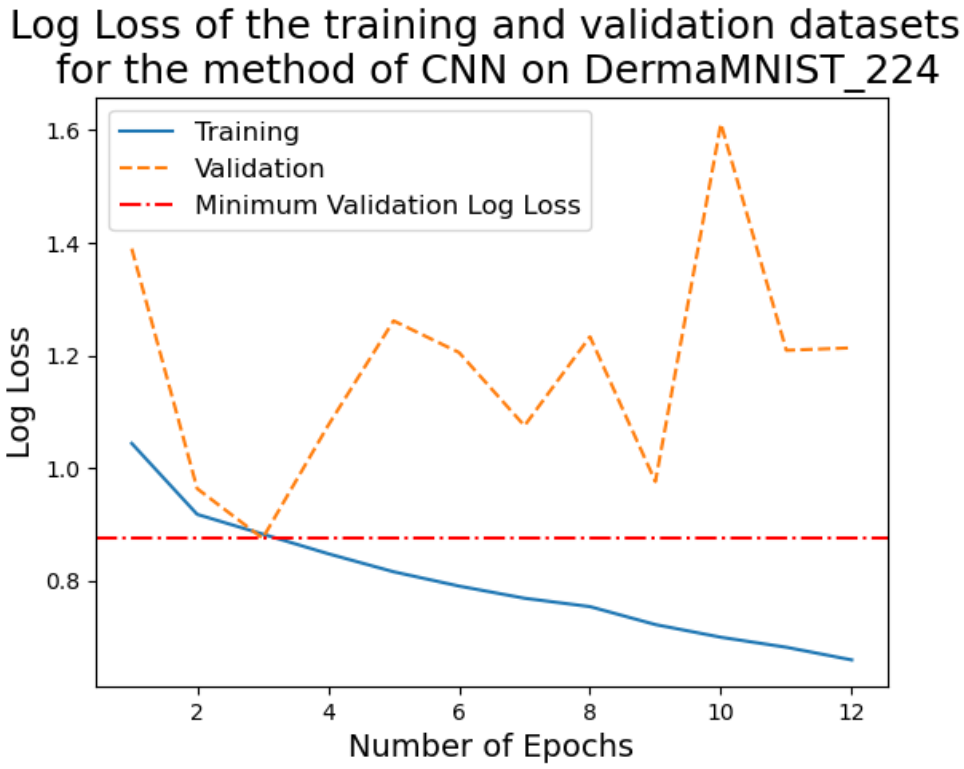


Figure A.36.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 224 × 224 resolution in Grayscale



Colour (RGB)

The following figures display the results from training and tuning a Convolutional Neural Network model for the classification of training examples using the DermaMNIST dataset at different resolutions in the original colour.

Figure A.37.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 28×28 resolution in Colour (RGB)

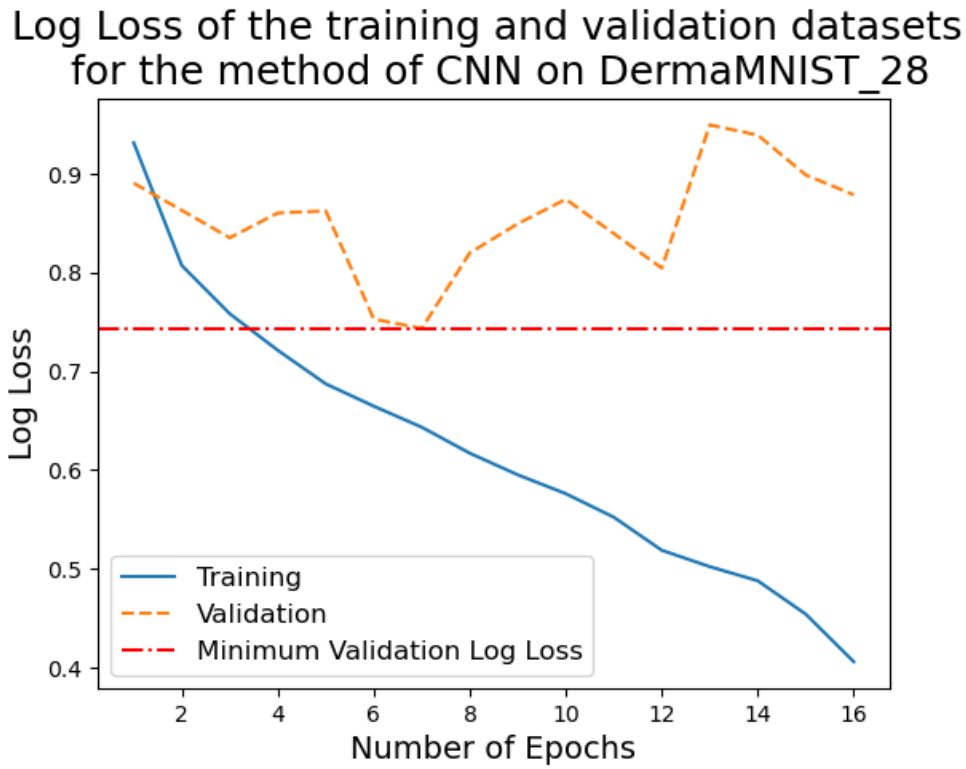


Figure A.38.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 64×64 resolution in Colour (RGB)

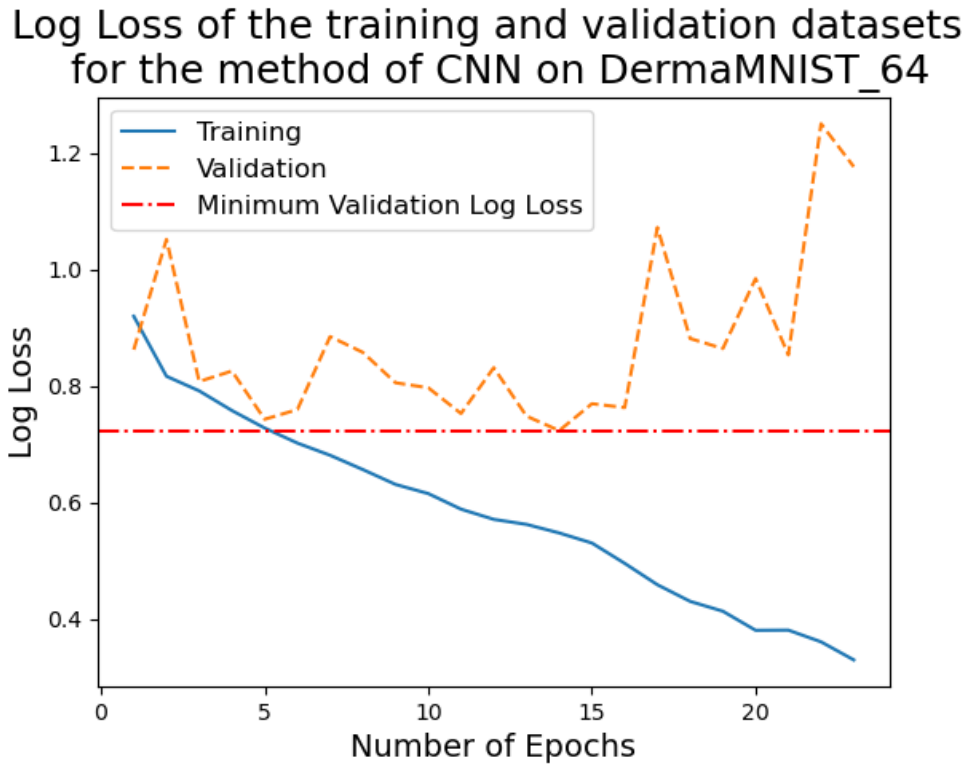


Figure A.39.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 128 × 128 resolution in Colour (RGB)

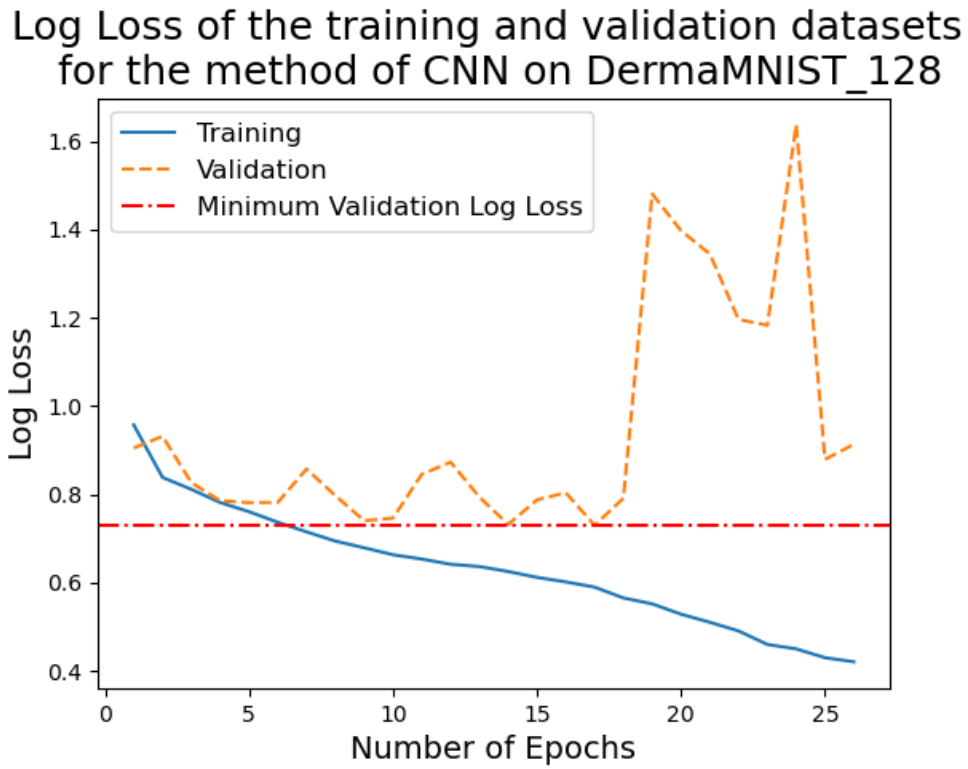
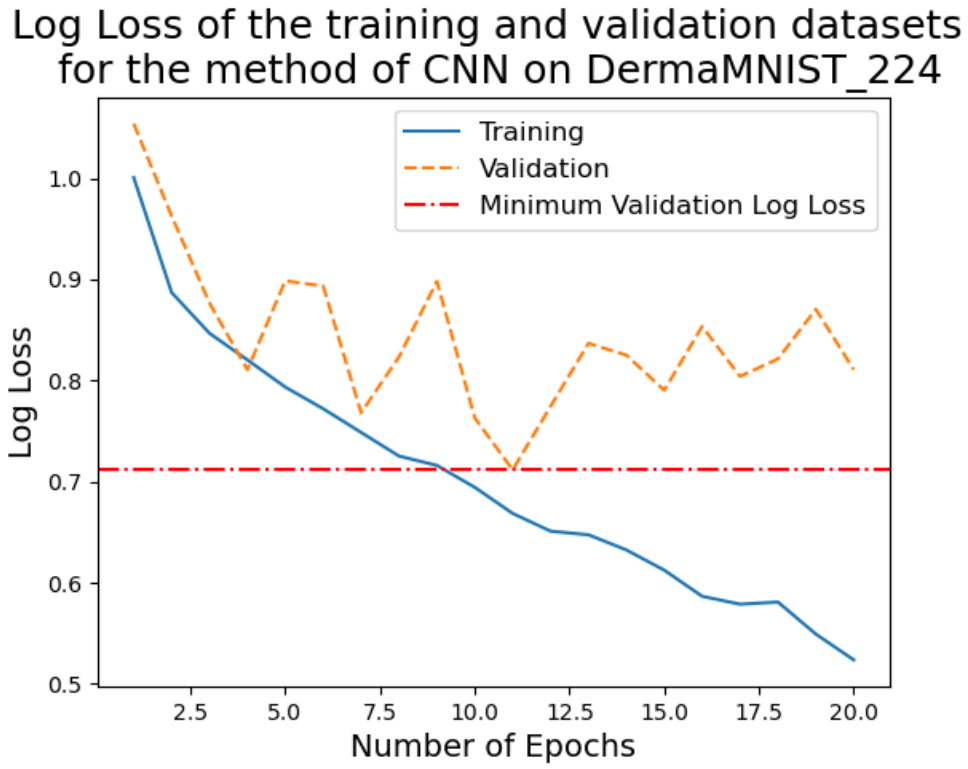


Figure A.40.: Plot of log loss during training and validation of the Convolutional Neural Network model on the DermaMNIST dataset at 224 × 224 resolution in Colour (RGB)



A.2.4. XGBoost

Grayscale

The following figures display the results from training and tuning an XGBoost model for the classification of training examples using the DermaMNIST dataset at different resolutions in Grayscale.

Figure A.41.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 28×28 resolution in Grayscale

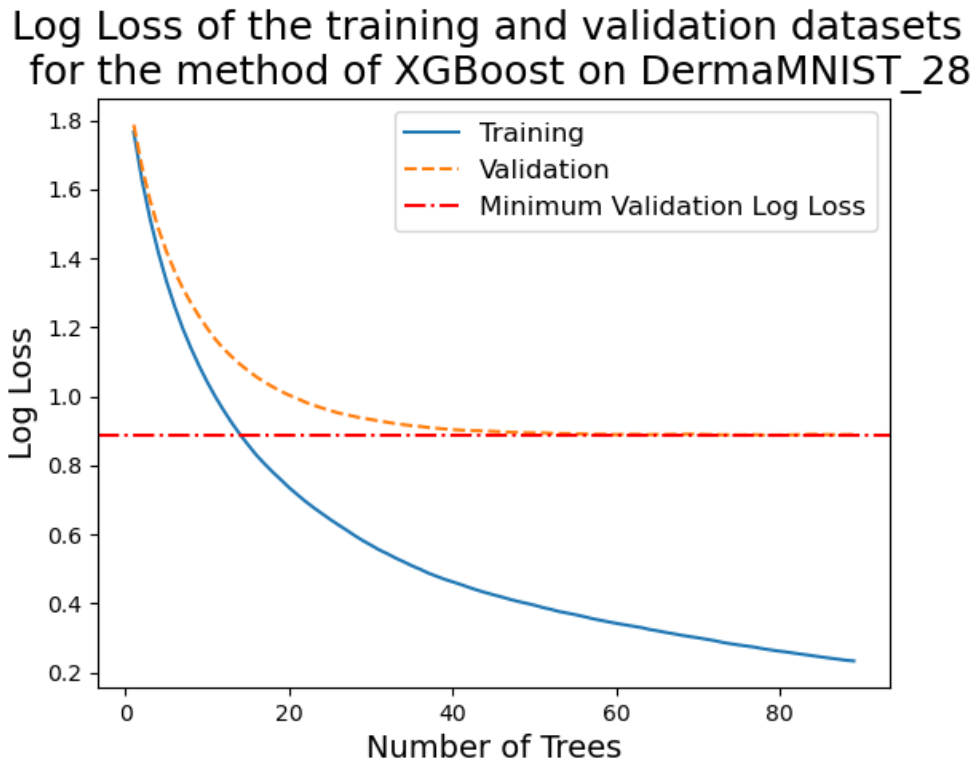


Figure A.42.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 64×64 resolution in Grayscale

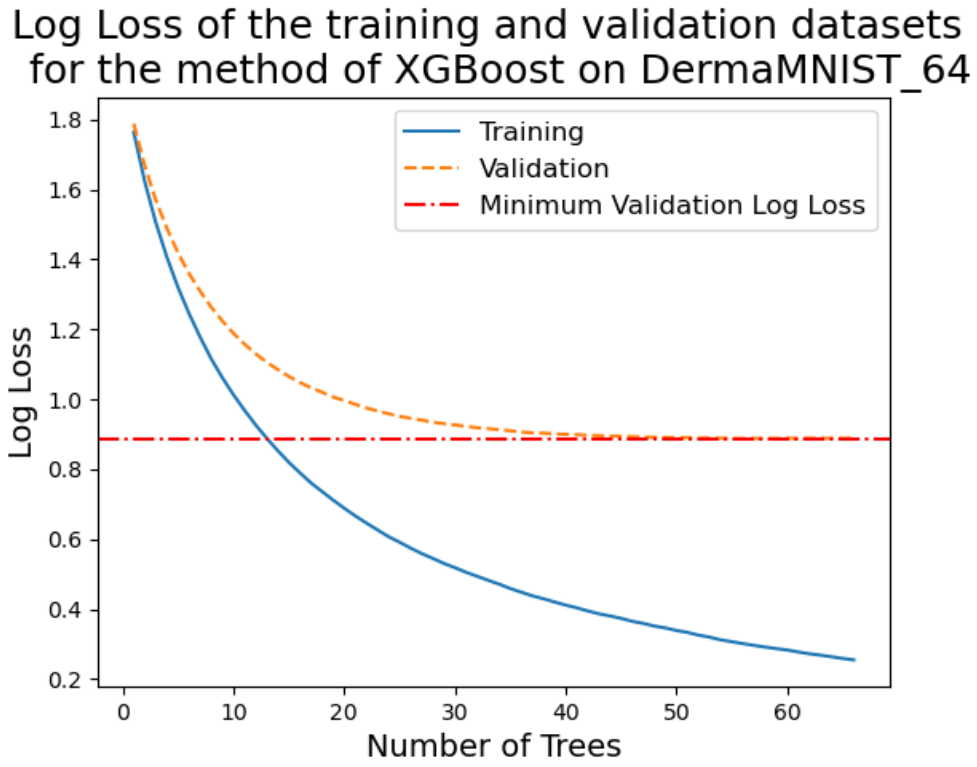


Figure A.43.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 128 × 128 resolution in Grayscale

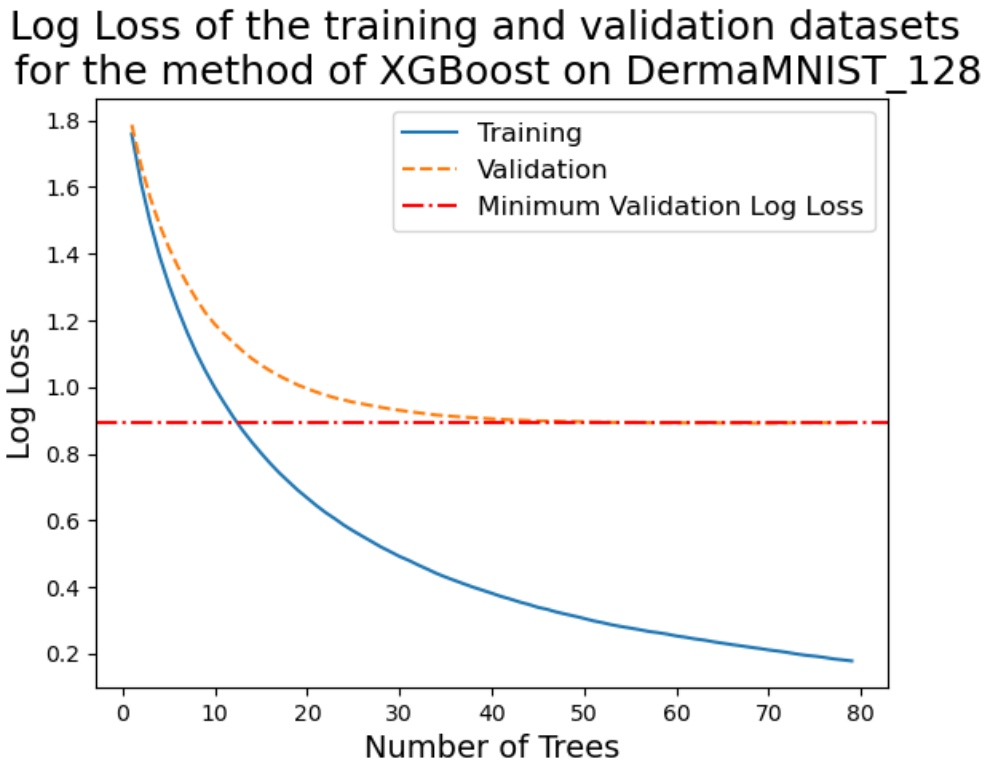
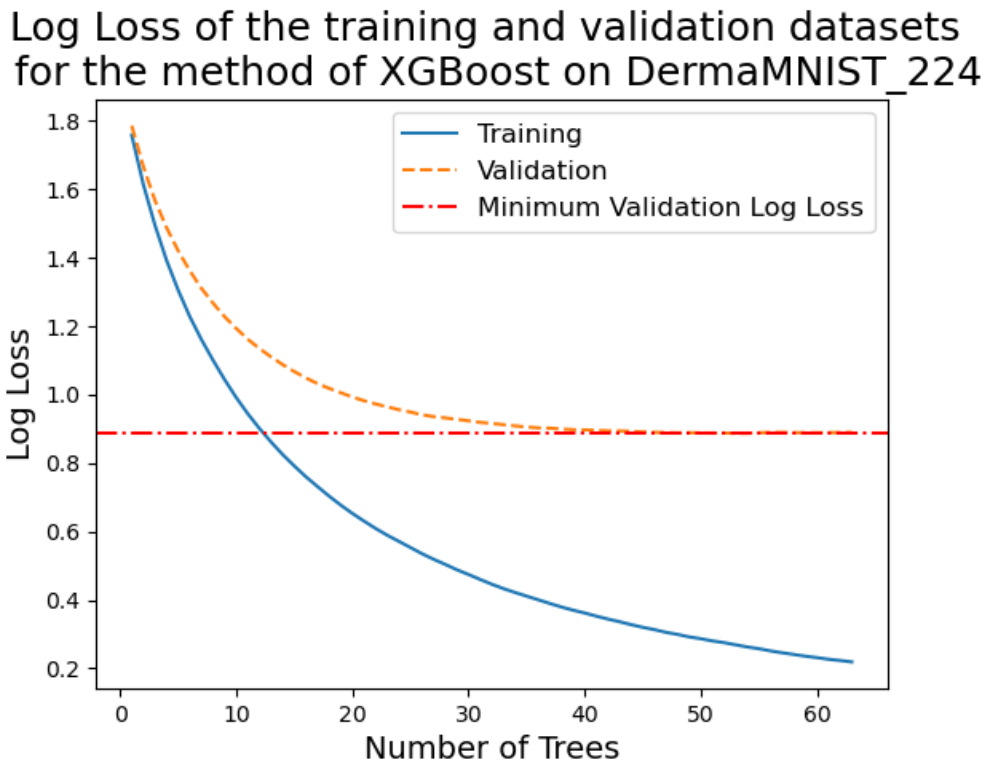


Figure A.44.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 224 × 224 resolution in Grayscale



Colour (RGB)

The following figures display the results from training and tuning an XGBoost model for the classification of training examples using the DermaMNIST dataset at different resolutions in the original colour.

Figure A.45.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 28 × 28 resolution in Colour (RGB)

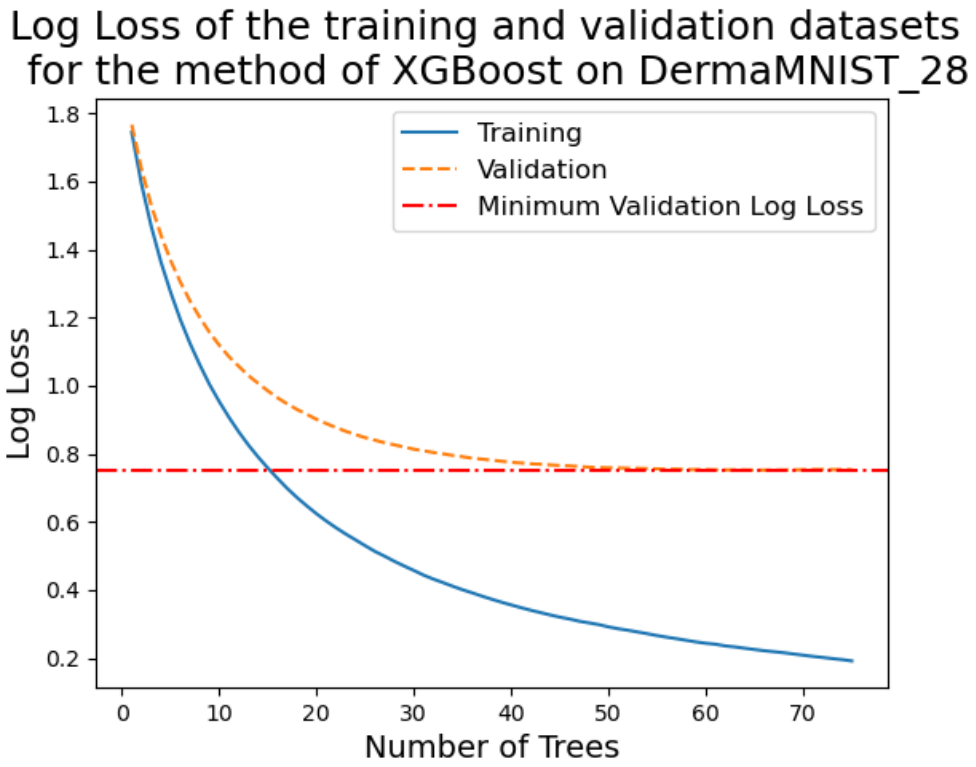


Figure A.46.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 64 × 64 resolution in Colour (RGB)

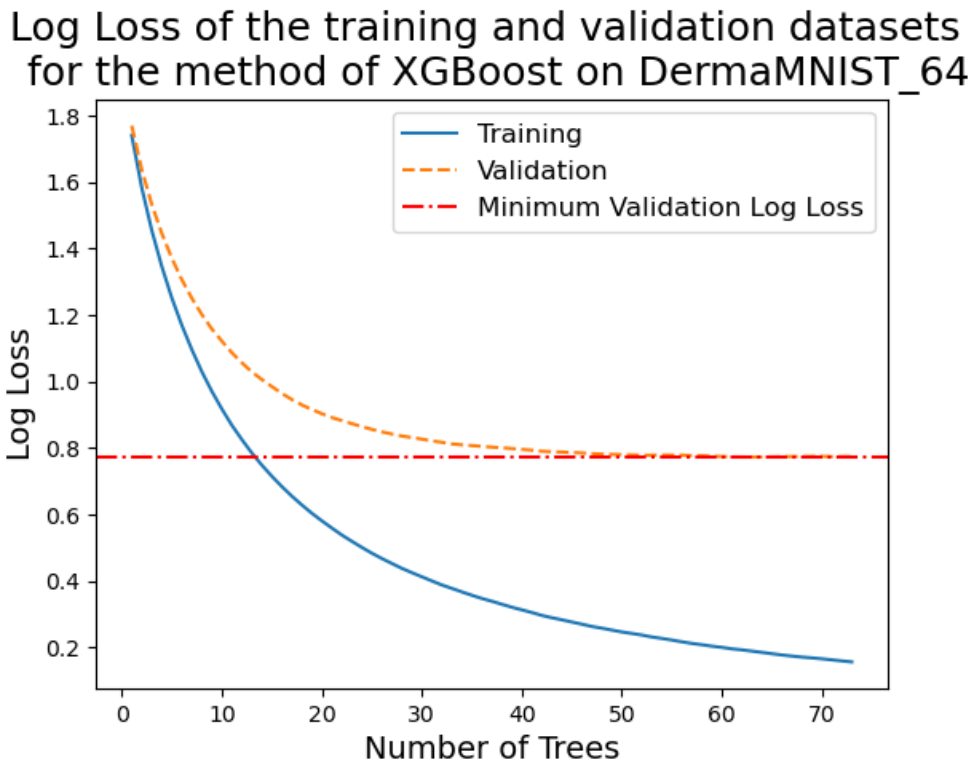


Figure A.47.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 128 × 128 resolution in Colour (RGB)

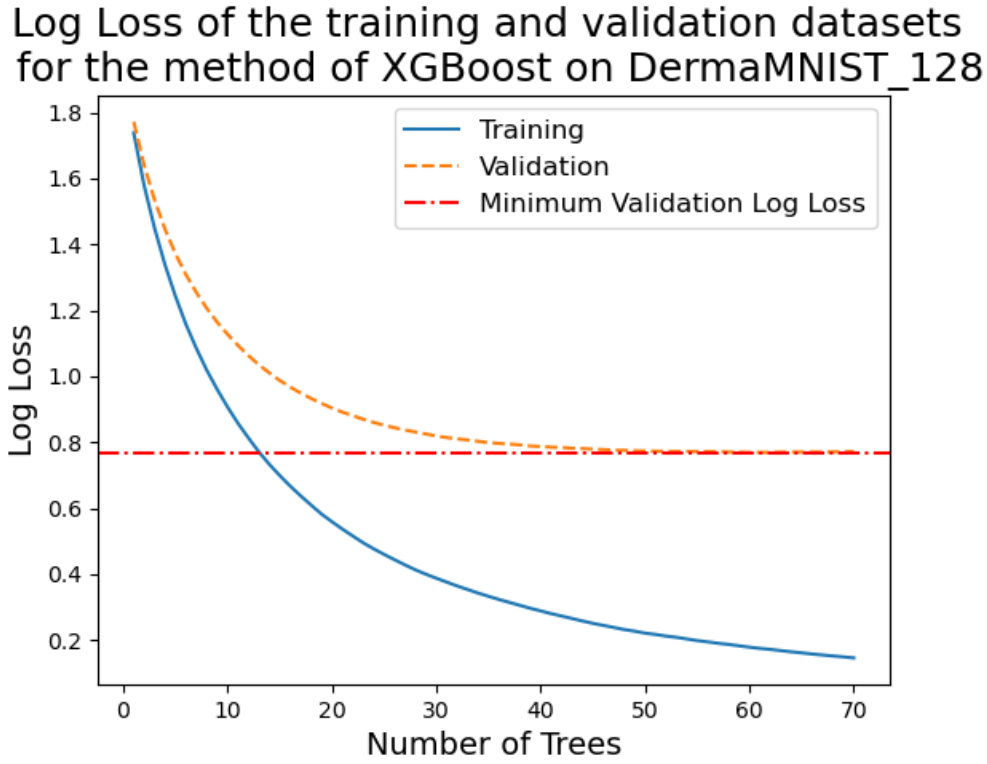
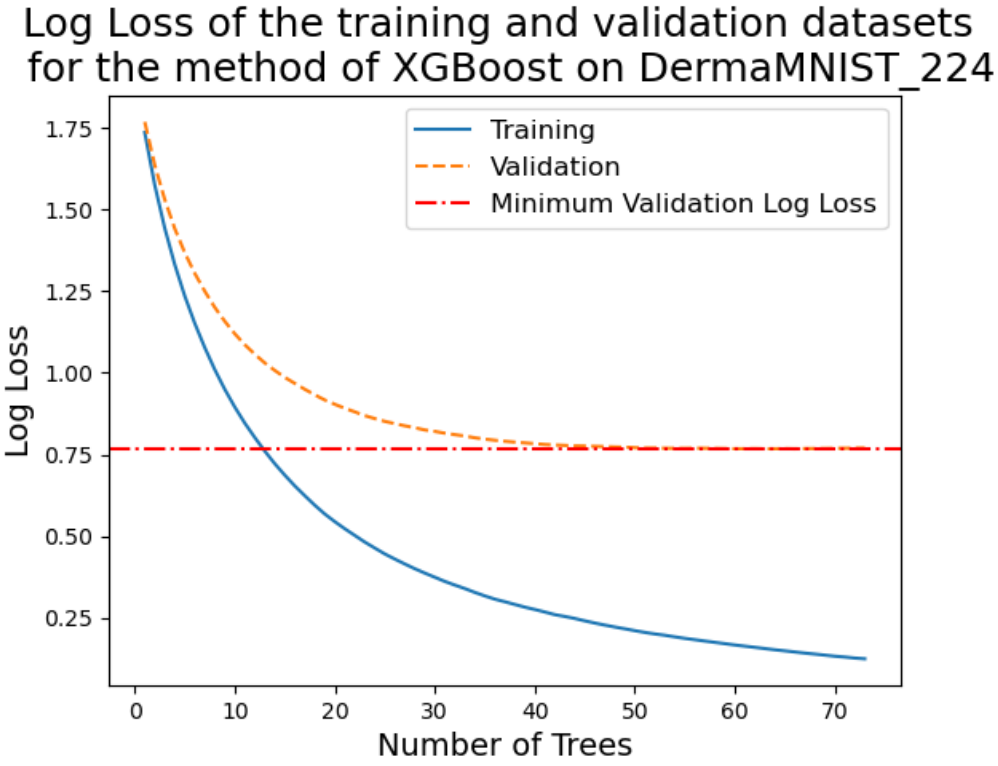


Figure A.48.: Plot of log loss during training and validation of the XGBoost model on the DermaMNIST dataset at 224 × 224 resolution in Colour (RGB)



A.3. PathMNIST

A.3.1. Linear Discriminant Analysis

Grayscale

The following tables display the results from training and tuning a Linear Discriminant Analysis model for the classification of training examples using the PathMNIST dataset at different resolutions in Grayscale.

Figure A.49.: Plot of scores of LDA model tuning on the PathMNIST dataset at 28×28 resolution in Grayscale

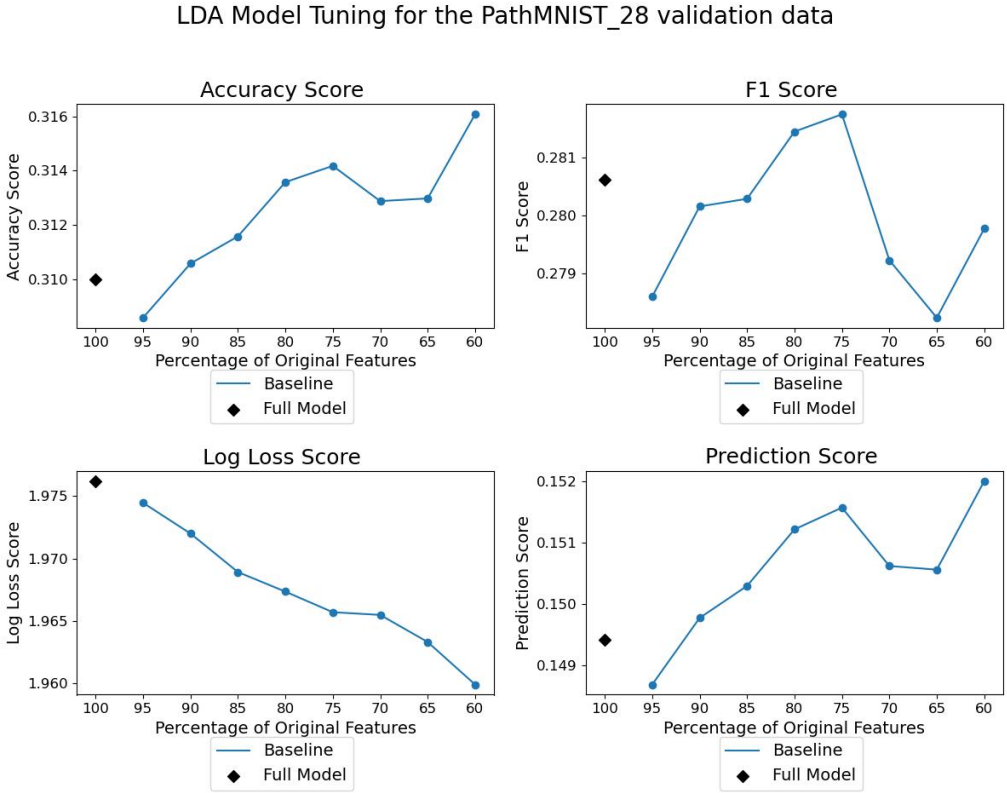


Table A.25.: Scores of LDA model tuning on the PathMNIST dataset at 28×28 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	4m 43s	0.310	0.281	1.976	0.149
95	Y	4m 15s	0.309	0.279	1.947	0.149
90	Y	3m 46s	0.311	0.280	1.972	0.150
85	Y	3m 42s	0.312	0.280	1.970	0.150
80	Y	3m 24s	0.314	0.281	1.967	0.151
75	Y	3m 3s	0.314	0.282	1.966	0.152
70	Y	2m 50s	0.313	0.279	1.965	0.151
65	Y	2m 25s	0.313	0.278	1.963	0.151
60	Y	2m 7s	0.316	0.280	1.960	0.152

Figure A.50.: Plot of scores of LDA model tuning on the PathMNIST dataset at 64×64 resolution in Grayscale

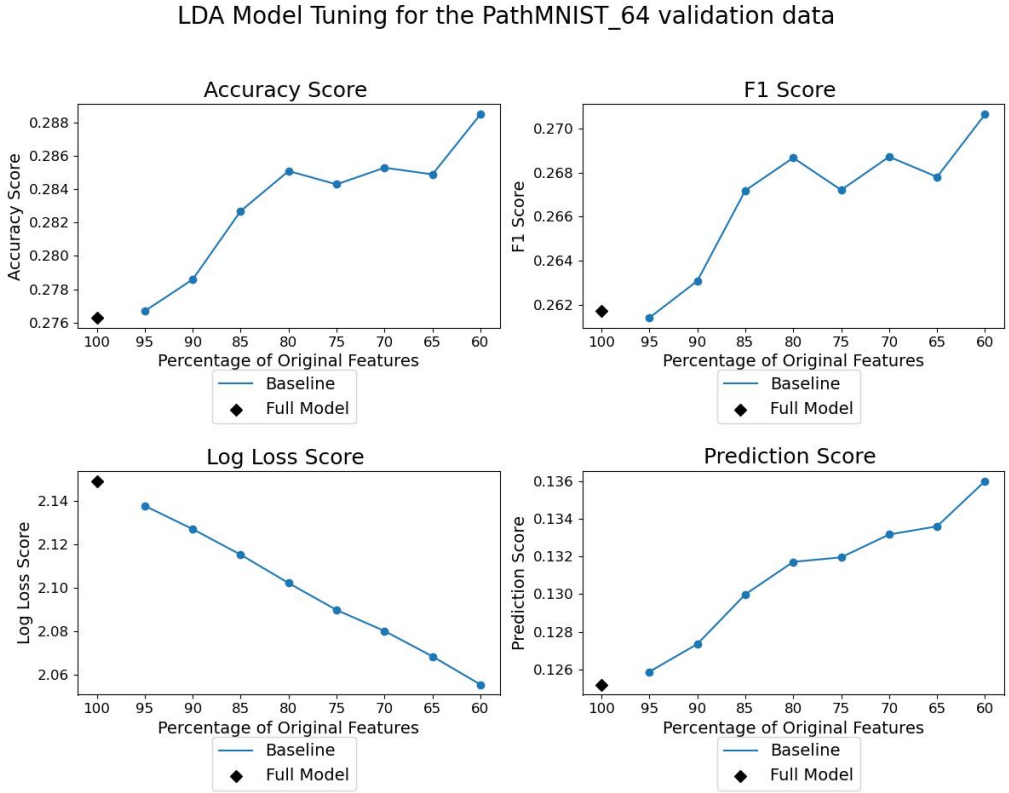


Table A.26.: Scores of LDA model tuning on the PathMNIST dataset at 64×64 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	29m 12s	0.276	0.262	2.149	0.125
95	Y	22m 26s	0.277	0.261	2.138	0.126
90	Y	25m 4s	0.279	0.261	2.138	0.126
85	Y	23m 15s	0.283	0.267	2.115	0.130
80	Y	22m 31s	0.285	0.269	2.102	0.132
75	Y	19m 41s	0.284	0.267	2.090	0.132
70	Y	17m 24s	0.285	0.269	2.080	0.133
65	Y	16m 35s	0.285	0.268	2.068	0.134
60	Y	15m 39s	0.288	0.271	2.055	0.136

Figure A.51.: Plot of scores of LDA model tuning on the PathMNIST dataset at 128×128 resolution in Grayscale

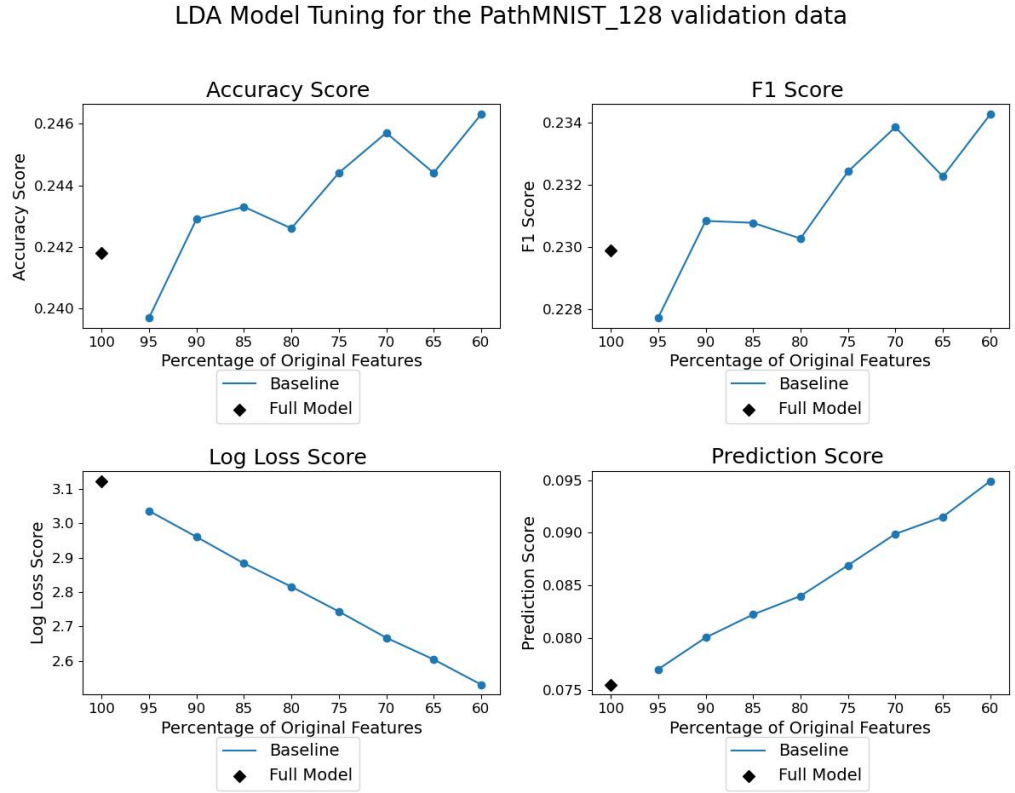


Table A.27.: Scores of LDA model tuning on the PathMNIST dataset at 128×128 resolution in Grayscale

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	2h 55m 37s	0.242	0.230	3.122	0.076
95	Y	2h 45m 3s	0.240	0.228	3.036	0.077
90	Y	2h 31m 44s	0.243	0.231	2.960	0.080
85	Y	2h 22m 4s	0.243	0.230	2.883	0.082
80	Y	2h 23m 3s	0.243	0.230	2.815	0.084
75	Y	5h 43m 33s	0.244	0.232	2.744	0.087
70	Y	5h 39m 34s	0.246	0.234	2.667	0.090
65	Y	5h 0m 43s	0.244	0.232	2.604	0.092
60	Y	4h 19m 50s	0.246	0.234	2.531	0.095

Colour (RGB)

The following tables display the results from training and tuning a Linear Discriminant Analysis model for the classification of training examples using the PathMNIST dataset at different resolutions in the original colour.

Figure A.52.: Plot of scores of LDA model tuning on the PathMNIST dataset at 28×28 resolution in Colour (RGB)

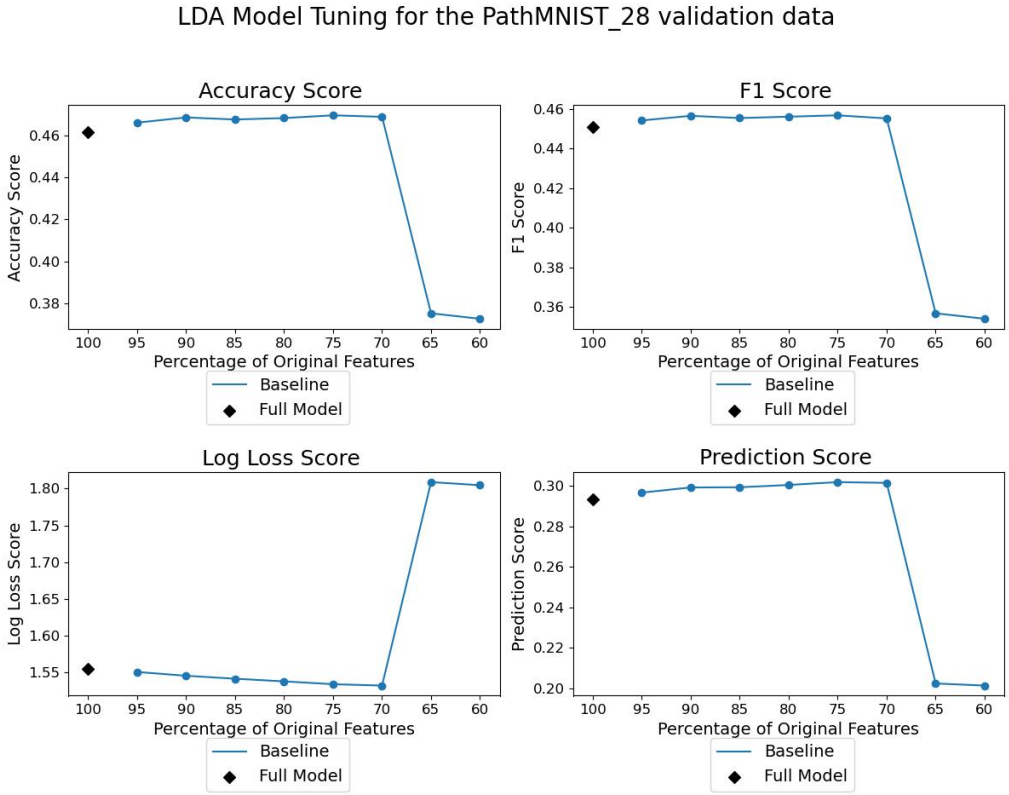


Table A.28.: Scores of LDA model tuning on the PathMNIST dataset at 28×28 resolution in Colour (RGB)

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	1m 38s	0.462	0.451	1.554	0.294
95	Y	1m 24s	0.466	0.454	1.551	0.297
90	Y	1m 30s	0.469	0.457	1.546	0.299
85	Y	1m 12s	0.468	0.455	1.542	0.299
80	Y	1m 7s	0.468	0.456	1.538	0.300
75	Y	1m 4s	0.470	0.457	1.534	0.302
70	Y	1m 7s	0.469	0.455	1.532	0.302
65	Y	50.9s	0.375	0.357	1.809	0.202
60	Y	43.2s	0.373	0.354	1.805	0.201

Figure A.53.: Plot of scores of LDA model tuning on the PathMNIST dataset at 64×64 resolution in Colour (RGB)

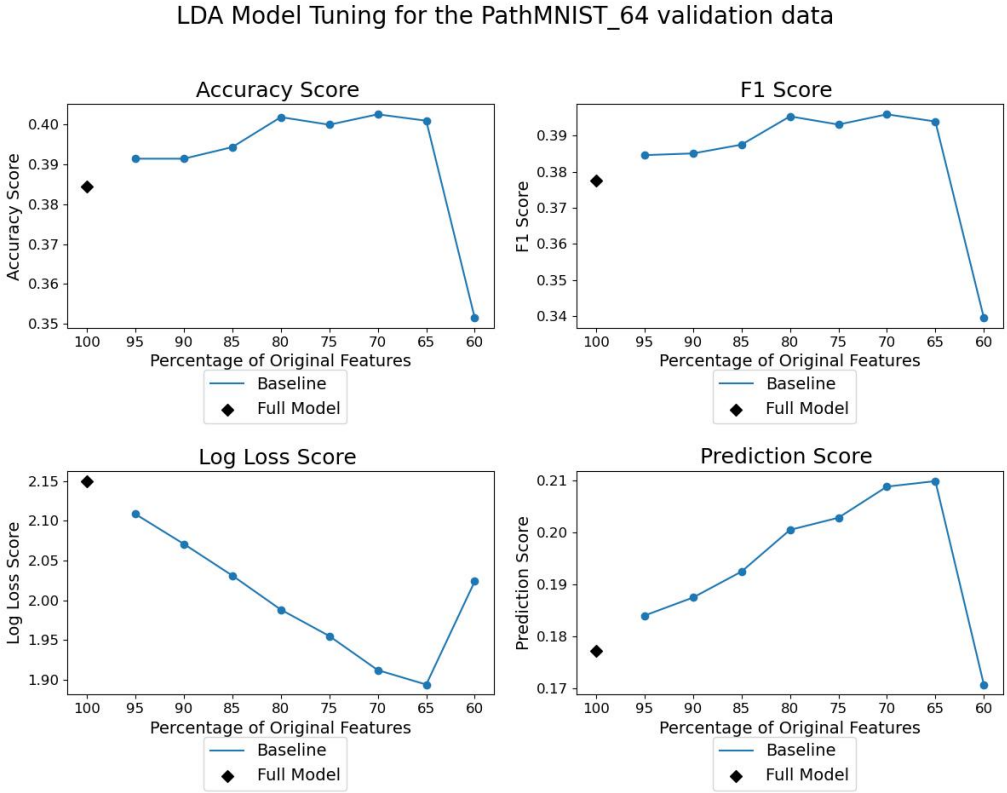


Table A.29.: Scores of LDA model tuning on the PathMNIST dataset at 64×64 resolution in Colour (RGB)

Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	25m 25s	0.384	0.378	2.150	0.177
95	Y	22m 1s	0.391	0.385	2.108	0.184
90	Y	21m 19s	0.391	0.385	2.071	0.187
85	Y	18m 47s	0.394	0.388	2.031	0.192
80	Y	18m 51s	0.402	0.395	1.988	0.200
75	Y	17m 57s	0.400	0.393	1.955	0.203
70	Y	1h 15m 41s	0.402	0.396	1.912	0.209
65	Y	1h 23m 54s	0.401	0.394	1.894	0.210
60	Y	1h 15m 58s	0.352	0.340	2.024	0.170

A.3.2. Logistic Regression

Grayscale

The following tables display the results from training and tuning a Logistic Regression Analysis model for the classification of training examples using the PathMNIST dataset at different resolutions in Grayscale.

Figure A.54.: Plot of scores of Logistic Regression model tuning on the PathMNIST dataset at 28×28 resolution in Grayscale

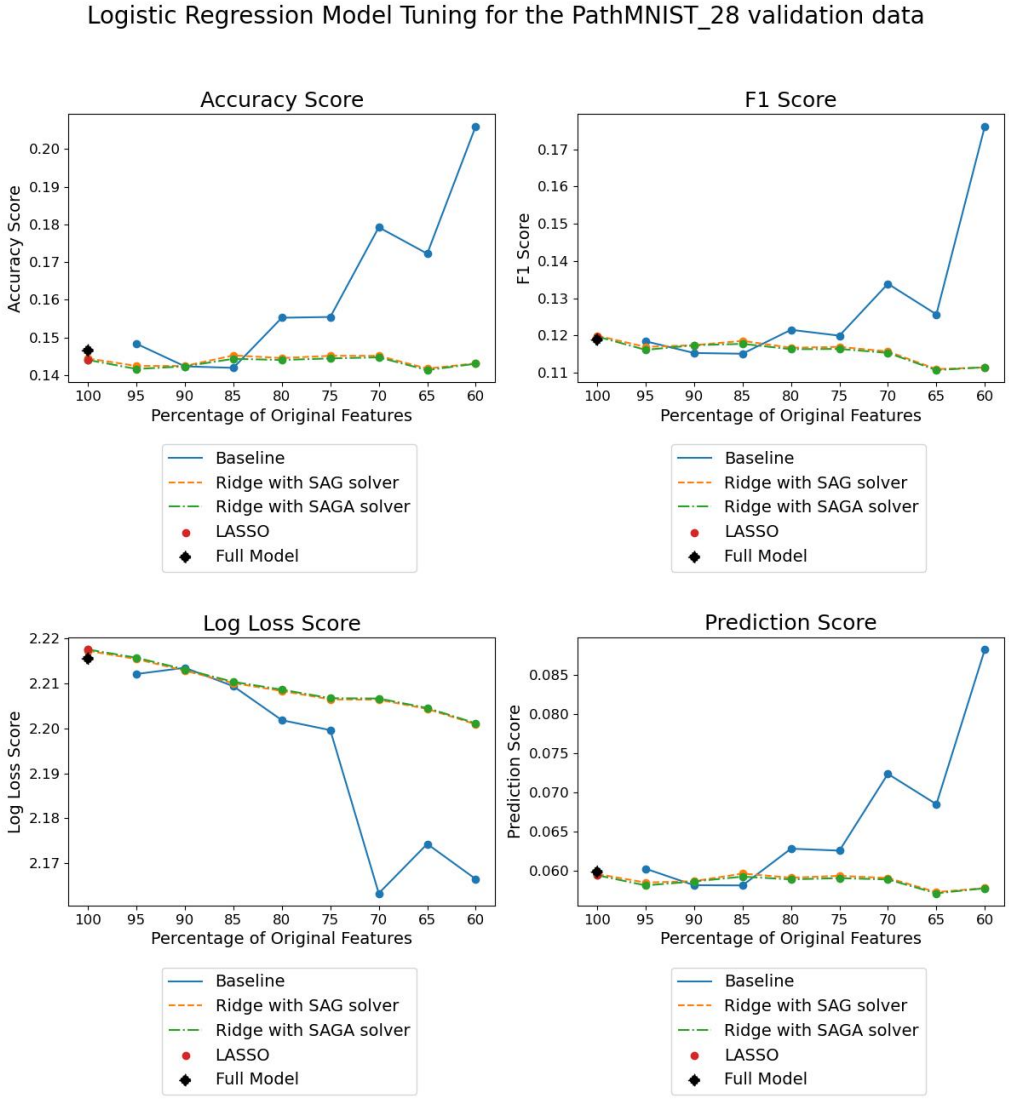


Table A.30.: Scores of Logistic Regression model tuning on the PathMNIST dataset at 28×28 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2m 47s	0.147	0.119	2.216	0.060
95	N	2m 33s	0.148	0.118	2.212	0.060
90	N	2m 26s	0.142	0.115	2.214	0.058
85	N	2m 16s	0.142	0.115	2.209	0.058
80	N	2m 11s	0.155	0.122	2.202	0.063
75	N	2m 0s	0.155	0.120	2.200	0.063
70	N	1m 47s	0.179	0.134	2.163	0.072
65	N	1m 47s	0.172	0.126	2.174	0.069
60	N	1m 34s	0.206	0.176	2.166	0.088
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	4m 55s	0.144	0.120	2.217	0.060
95	Y	4m 29s	0.142	0.117	2.215	0.059
90	Y	4m 51s	0.142	0.117	2.213	0.059
85	Y	3m 44s	0.145	0.119	2.210	0.060
80	Y	3m 19s	0.145	0.117	2.208	0.059
75	Y	2m 44s	0.145	0.117	2.207	0.059
70	Y	2m 26s	0.145	0.116	2.206	0.059
65	Y	2m 5s	0.142	0.111	2.204	0.057
60	Y	1m 52s	0.143	0.111	2.201	0.058
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	9m 27s	0.144	0.120	2.218	0.059
95	Y	8m 34s	0.142	0.116	2.216	0.058
90	Y	7m 53s	0.142	0.117	2.213	0.059
85	Y	7m 26s	0.144	0.118	2.210	0.059
80	Y	6m 38s	0.144	0.116	2.209	0.059
75	Y	5m 17s	0.144	0.116	2.207	0.059
70	Y	4m 40s	0.145	0.115	2.207	0.059
65	Y	4m 14s	0.141	0.111	2.205	0.057
60	Y	3m 34s	0.143	0.111	2.201	0.058
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	16m 50s	0.144	0.120	2.218	0.059

Figure A.55.: Plot of scores of Logistic Regression model tuning on the PathMNIST dataset at 64×64 resolution in Grayscale

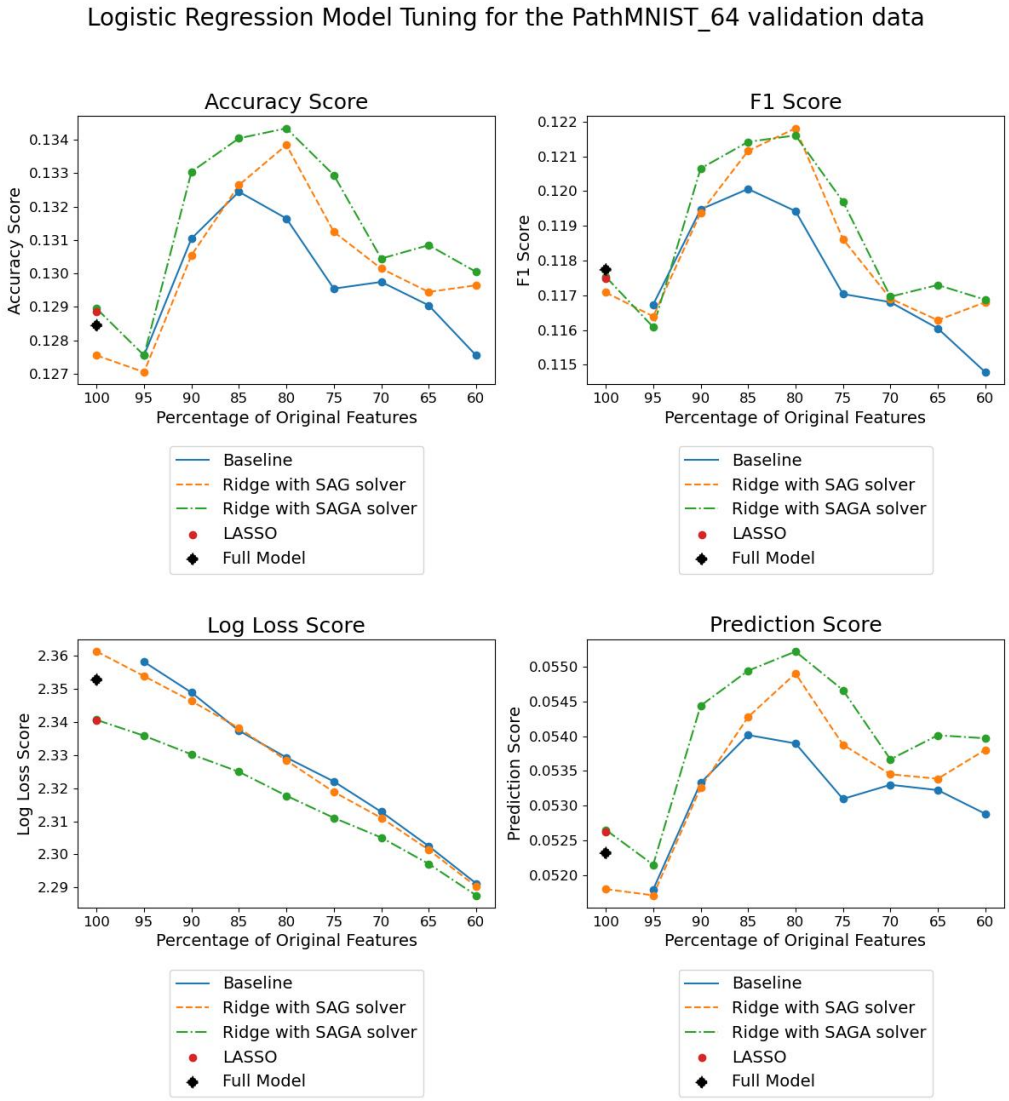


Table A.31.: Scores of Logistic Regression model tuning on the PathMNIST dataset at 64 × 64 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	14m 59s	0.128	0.118	2.353	0.052
95	N	14m 24s	0.128	0.117	2.358	0.052
90	N	13m 35s	0.131	0.119	2.349	0.053
85	N	12m 53s	0.132	0.120	2.337	0.054
80	N	12m 14s	0.132	0.119	2.329	0.054
75	N	11m 1s	0.130	0.117	2.322	0.053
70	N	11m 43s	0.130	0.117	2.313	0.053
65	N	11m 3s	0.129	0.116	2.302	0.053
60	N	11m 0s	0.128	0.115	2.291	0.053
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	42m 45s	0.128	0.117	2.361	0.052
95	N	40m 54s	0.127	0.116	2.354	0.052
90	N	38m 6s	0.131	0.119	2.346	0.053
85	N	36m 20s	0.133	0.121	2.338	0.054
80	N	34m 6s	0.134	0.122	2.328	0.055
75	N	32m 51s	0.131	0.119	2.319	0.054
70	N	30m 43s	0.130	0.117	2.311	0.053
65	N	27m 58s	0.129	0.116	2.301	0.053
60	N	26m 6s	0.130	0.117	2.290	0.054
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	50m 48s	0.129	0.118	2.341	0.053
95	N	48m 42s	0.128	0.116	2.336	0.052
90	N	45m 9s	0.134	0.121	2.330	0.054
85	N	43m 4s	0.134	0.121	2.325	0.055
80	N	40m 59s	0.134	0.122	2.318	0.055
75	N	39m 1s	0.133	0.120	2.311	0.055
70	N	35m 25s	0.130	0.117	2.305	0.054
65	N	33m 42s	0.131	0.117	2.297	0.054
60	N	30m 33s	0.130	0.117	2.288	0.054
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	1h 59m 26s	0.129	0.117	2.340	0.053

Figure A.56.: Plot of scores of Logistic Regression model tuning on the PathMNIST dataset at 128 × 128 resolution in Grayscale

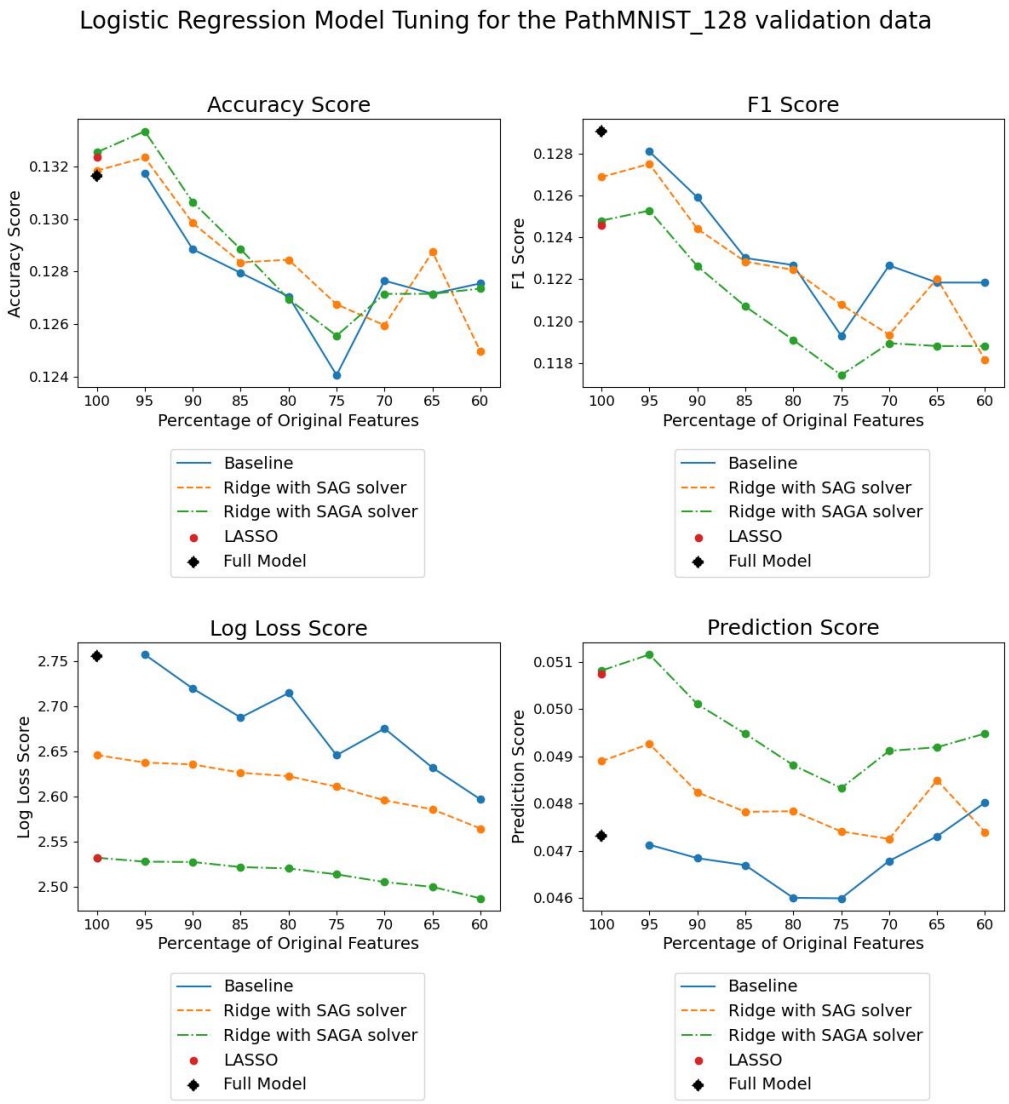


Table A.32.: Scores of Logistic Regression model tuning on the PathMNIST dataset at 128 × 128 resolution in Grayscale

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1h 1m 21s	0.132	0.129	2.755	0.047
95	N	58m 14s	0.132	0.128	2.757	0.047
90	N	54m 11s	0.129	0.126	2.719	0.047
85	N	51m 17s	0.128	0.123	2.687	0.047
80	N	48m 37s	0.127	0.123	2.714	0.046
75	N	48m 6s	0.124	0.119	2.646	0.046
70	N	42m 42s	0.128	0.123	2.675	0.047
65	N	40m 16s	0.127	0.122	2.632	0.047
60	N	36m 14s	0.128	0.122	2.597	0.048
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2h 50m 55s	0.132	0.127	2.646	0.049
95	N	2h 38m 19s	0.132	0.128	2.638	0.049
90	N	2h 31m 14s	0.130	0.124	2.636	0.048
85	N	2h 25m 45s	0.128	0.123	2.626	0.048
80	N	2h 20m 17s	0.128	0.122	2.623	0.048
75	N	2h 6m 12s	0.127	0.121	2.611	0.047
70	N	1h 56m 30s	0.126	0.119	2.596	0.047
65	N	1h 50m 25s	0.129	0.122	2.585	0.048
60	N	1h 40m 11s	0.125	0.118	2.565	0.047
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	3h 21m 41s	0.133	0.125	2.532	0.051
95	N	3h 7m 19s	0.133	0.126	2.528	0.051
90	N	2h 58m 57s	0.131	0.123	2.528	0.050
85	N	2h 47m 44s	0.129	0.121	2.522	0.049
80	N	2h 39m 26s	0.127	0.119	2.521	0.049
75	N	2h 27m 4s	0.126	0.117	2.514	0.048
70	N	2h 18m 27s	0.127	0.119	2.505	0.049
65	N	2h 8m 42s	0.127	0.119	2.500	0.049
60	N	1h 58m 40s	0.127	0.119	2.488	0.049
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	Y	8h 1m 7s	0.132	0.125	2.532	0.051

Colour (RGB)

The following tables display the results from training and tuning a Logistic Regression Analysis model for the classification of training examples using the PathMNIST dataset at different resolutions the in original colour.

Figure A.57.: Plot of scores of Logistic Regression model tuning on the PathMNIST dataset at 28 × 28 resolution in Colour (RGB)

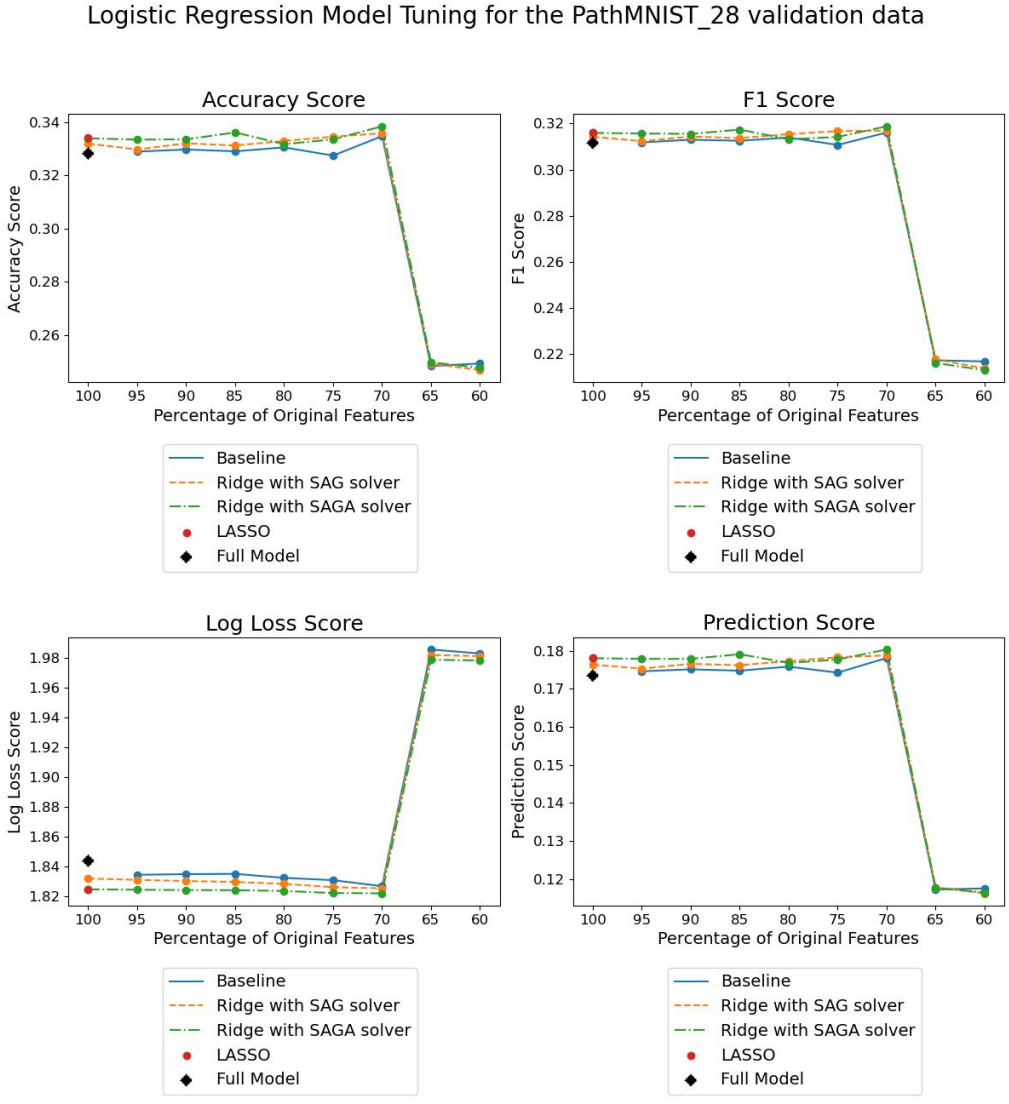


Table A.33.: Scores of Logistic Regression model tuning on the PathMNIST dataset at 28 × 28 resolution in Colour (RGB)

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	9m 1s	0.328	0.312	1.844	0.174
95	N	8m 24s	0.329	0.312	1.834	0.175
90	N	7m 45s	0.330	0.313	1.835	0.175
85	N	7m 18s	0.329	0.313	1.835	0.175
80	N	6m 49s	0.330	0.314	1.832	0.176
75	N	6m 20s	0.327	0.311	1.831	0.174
70	N	5m 59s	0.335	0.316	1.827	0.178
65	N	5m 26s	0.248	0.217	1.985	0.117
60	N	4m 59s	0.249	0.217	1.983	0.118
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	25m 24s	0.332	0.314	1.832	0.176
95	N	24m 16s	0.330	0.312	1.831	0.175
90	N	22m 50s	0.332	0.314	1.830	0.177
85	N	21m 54s	0.331	0.314	1.830	0.176
80	N	20m 30s	0.333	0.315	1.828	0.177
75	N	19m 15s	0.334	0.318	1.826	0.178
70	N	18m 1s	0.334	0.317	1.825	0.179
65	N	17m 10s	0.249	0.218	1.982	0.118
60	N	15m 42s	0.247	0.214	1.981	0.116
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	30m 2s	0.334	0.316	1.824	0.178
95	N	28m 38s	0.333	0.316	1.824	0.178
90	N	27m 16s	0.333	0.316	1.824	0.178
85	N	25m 32s	0.336	0.317	1.824	0.179
80	N	24m 20s	0.331	0.313	1.823	0.177
75	N	22m 59s	0.333	0.314	1.822	0.178
70	N	20m 59s	0.338	0.319	1.822	0.180
65	Y	19m 34s	0.250	0.216	1.979	0.118
60	Y	18m 31s	0.248	0.213	1.978	0.116
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	1h 6m 5s	0.334	0.316	1.824	0.178

Figure A.58.: Plot of scores of Logistic Regression model tuning on the PathMNIST dataset at 64×64 resolution in Colour (RGB)

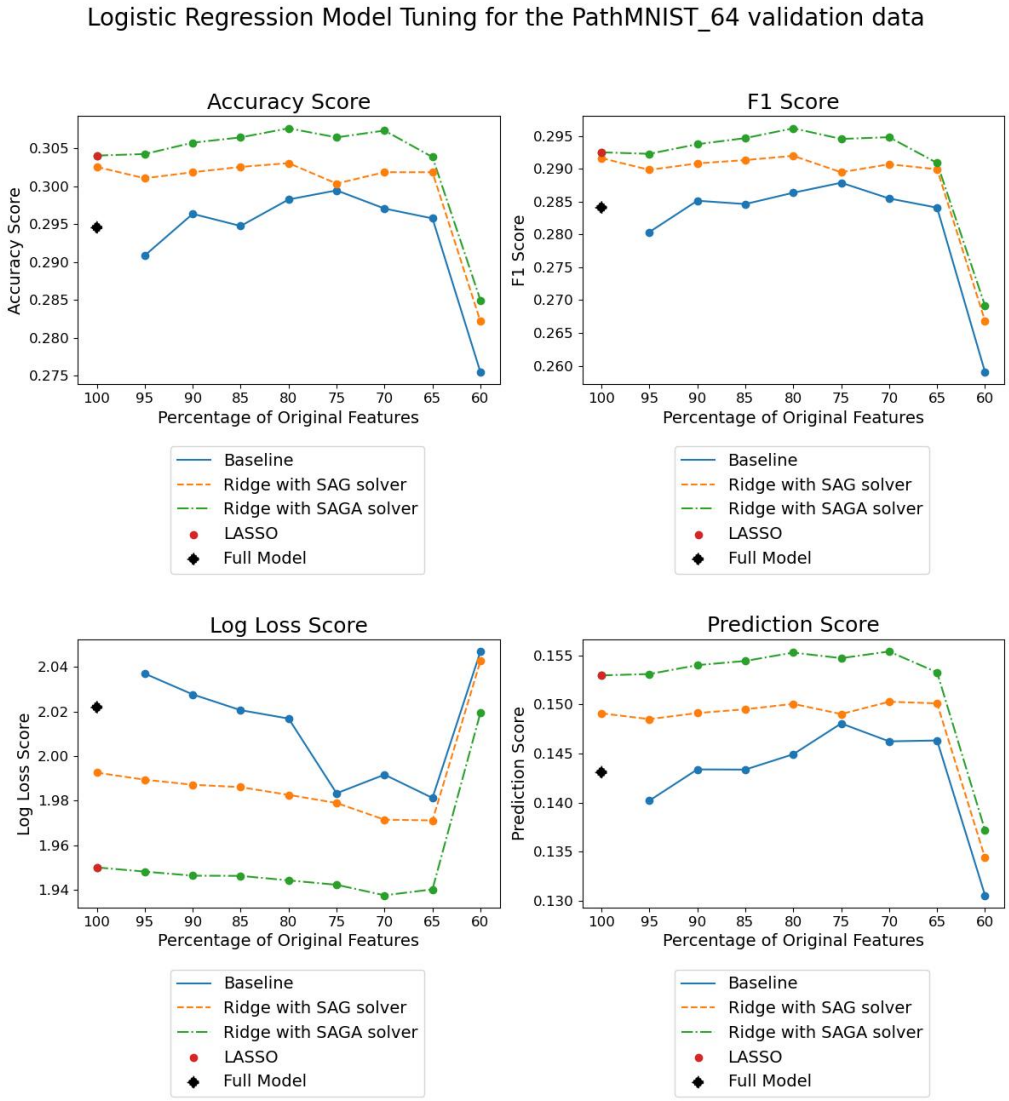


Table A.34.: Scores of Logistic Regression model tuning on the PathMNIST dataset at 64 × 64 resolution in Colour (RGB)

LBFGS Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	49m 45s	0.295	0.284	2.022	0.143
95	N	44m 58s	0.291	0.280	2.037	0.140
90	N	42m 47s	0.296	0.285	2.028	0.143
85	N	40m 34s	0.295	0.285	2.021	0.143
80	N	38m 9s	0.298	0.286	2.017	0.145
75	N	36m 12s	0.299	0.288	1.983	0.148
70	N	33m 30s	0.297	0.285	1.992	0.146
65	N	30m 55s	0.296	0.284	1.981	0.146
60	N	28m 31s	0.275	0.259	2.047	0.131
SAG Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2h 16m 29s	0.303	0.292	1.993	0.149
95	N	2h 4m 19s	0.301	0.290	1.989	0.149
90	N	1h 57m 41s	0.302	0.291	1.987	0.149
85	N	1h 50m 55s	0.303	0.291	1.986	0.150
80	N	1h 43m 37s	0.303	0.292	1.983	0.150
75	N	1h 37m 13s	0.300	0.289	1.979	0.149
70	N	1h 30m 20s	0.302	0.291	1.971	0.150
65	N	1h 23m 31s	0.302	0.290	1.971	0.150
60	N	1h 18m 55s	0.282	0.267	2.043	0.134
SAGA Solver with L2/Ridge Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	2h 40m 30s	0.304	0.292	1.950	0.153
95	N	2h 32m 55s	0.304	0.292	1.948	0.153
90	N	2h 18m 29s	0.306	0.294	1.946	0.154
85	N	2h 8m 18s	0.306	0.295	1.946	0.154
80	N	2h 1m 11s	0.308	0.296	1.944	0.155
75	N	1h 51m 27s	0.306	0.295	1.942	0.155
70	N	1h 44m 45s	0.307	0.294	1.937	0.155
65	N	1h 36m 45s	0.304	0.291	1.940	0.153
60	N	1h 30m 3s	0.285	0.269	2.019	0.137
SAGA Solver with L1/LASSO Regularisation						
Feature %	Convergence	Run Time	Accuracy	F1	Log Loss	Pred. Score
100	N	5h 47m 24s	0.304	0.292	1.950	0.153

A.3.3. Convolutional Neural Network

Grayscale

The following figures display the results from training and tuning a Convolutional Neural Network model for the classification of training examples using the PathMNIST dataset at different resolutions in Grayscale.

Figure A.59.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 28×28 resolution in Grayscale

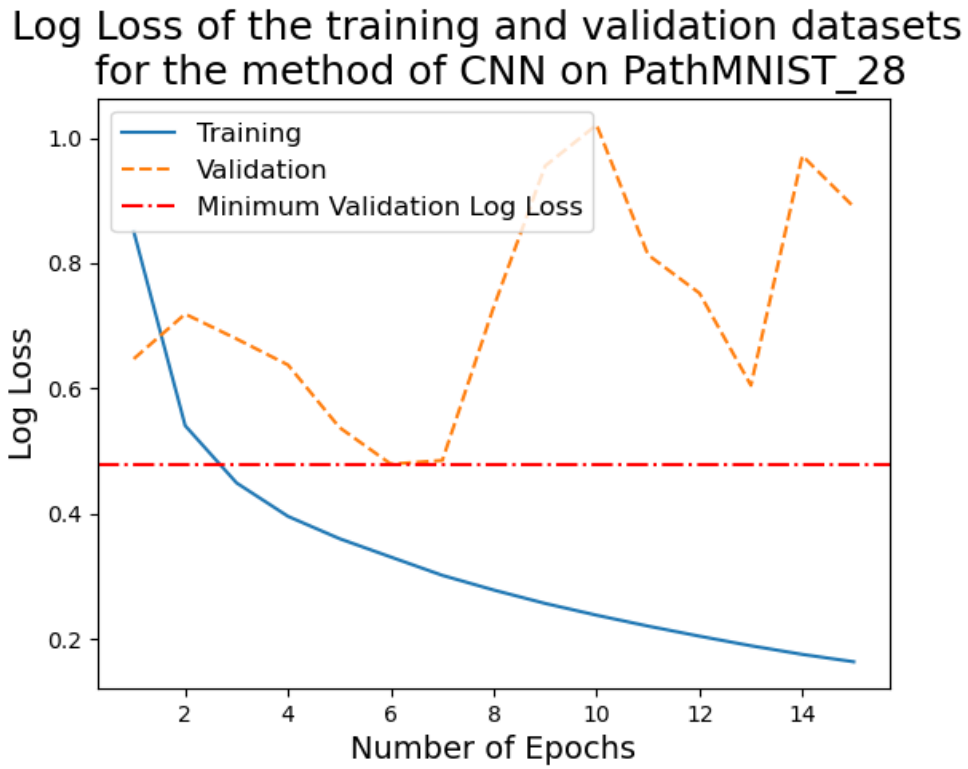


Figure A.60.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 64×64 resolution in Grayscale

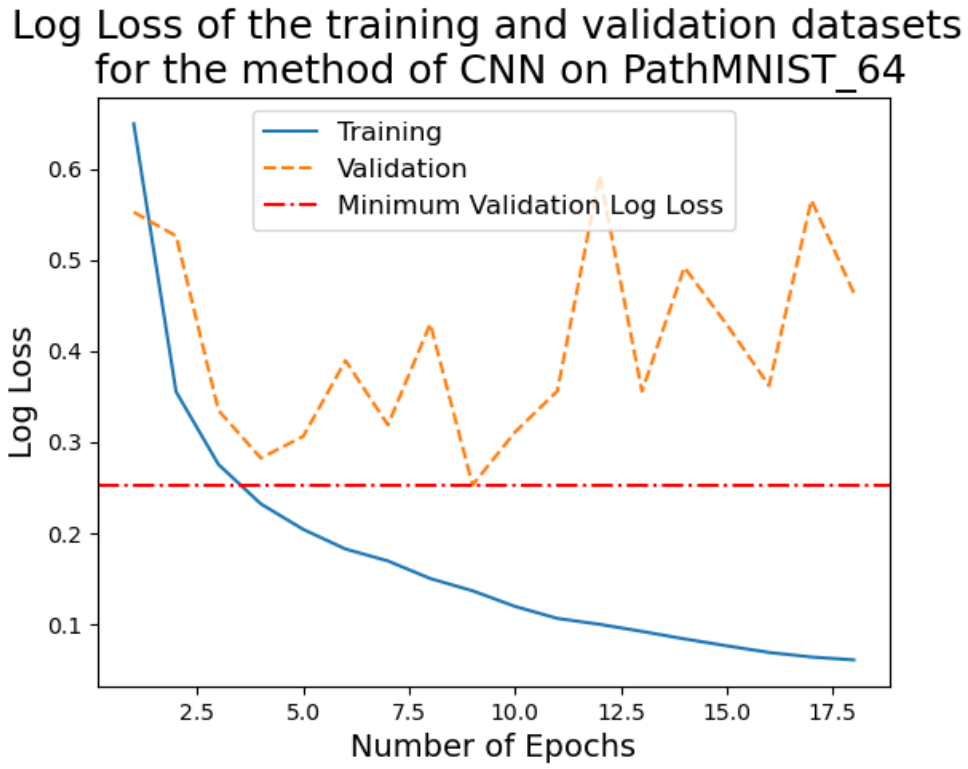


Figure A.61.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 128 × 128 resolution in Grayscale

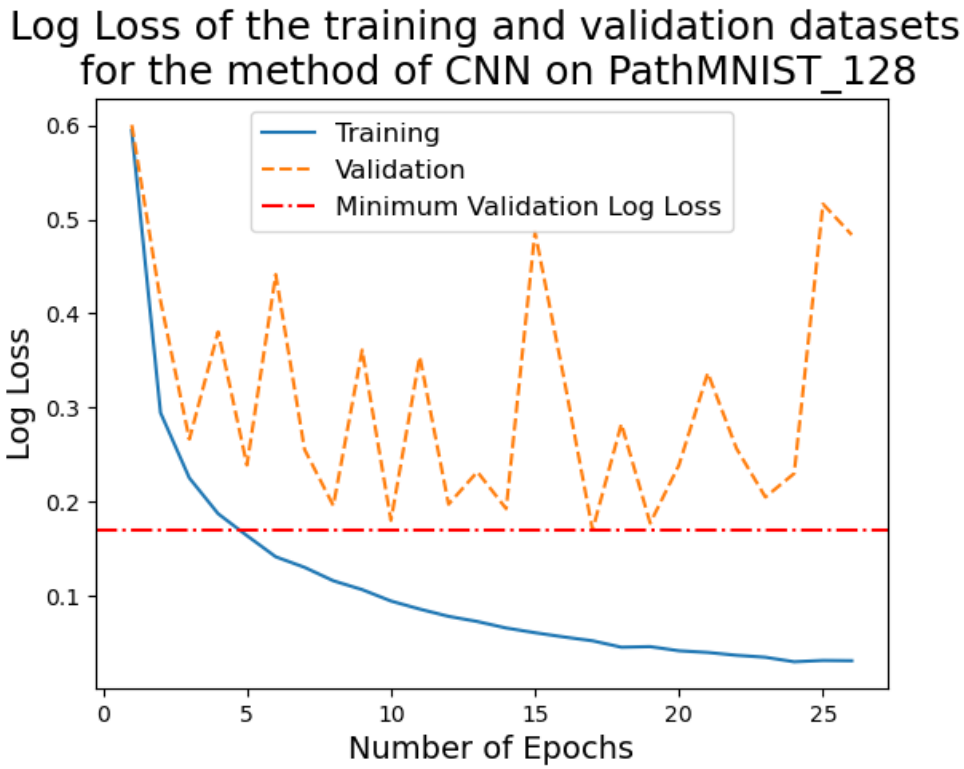
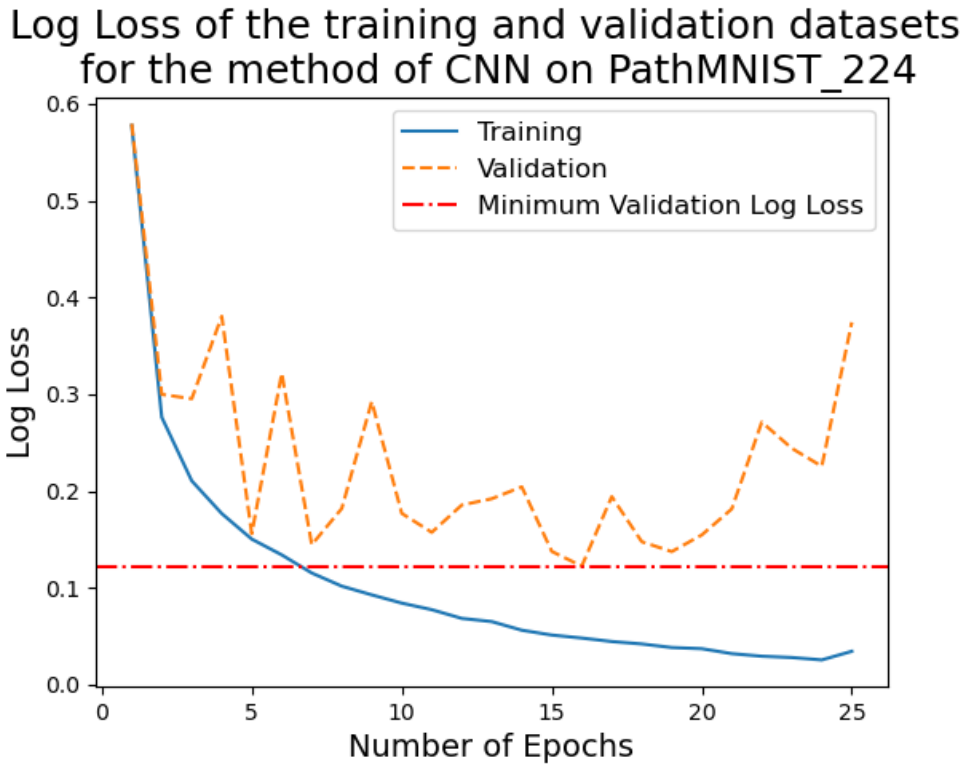


Figure A.62.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 224 × 224 resolution in Grayscale



Colour (RGB)

The following figures display the results from training and tuning a Convolutional Neural Network model for the classification of training examples using the PathMNIST dataset at different resolutions in the original colour.

Figure A.63.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 28×28 resolution in Colour (RGB)

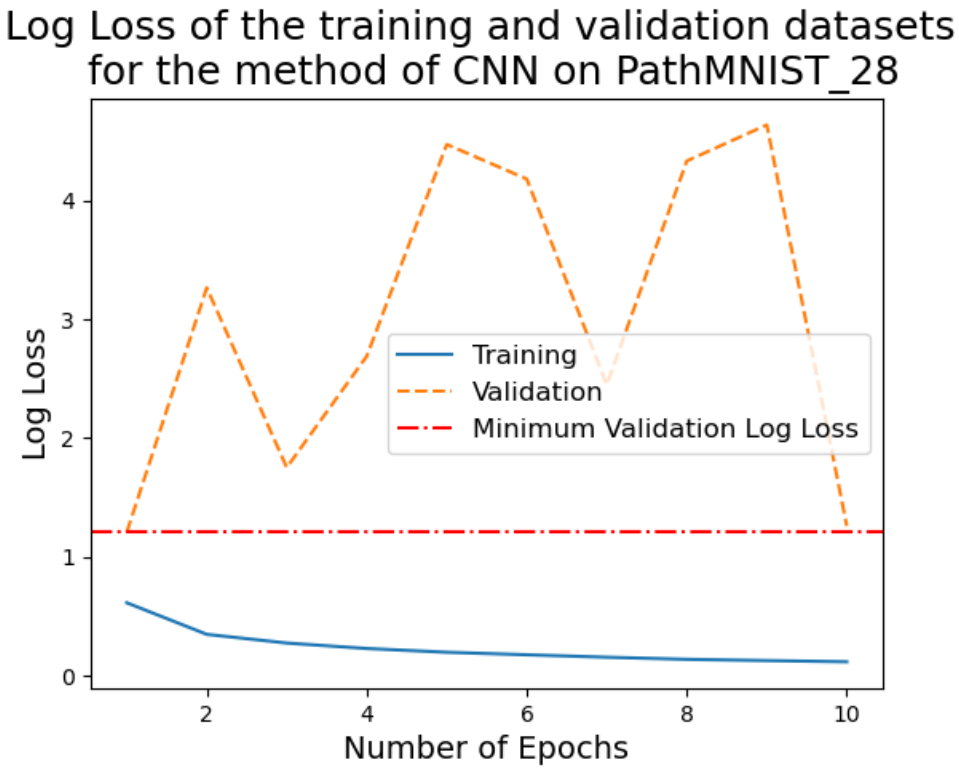


Figure A.64.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 64×64 resolution in Colour (RGB)

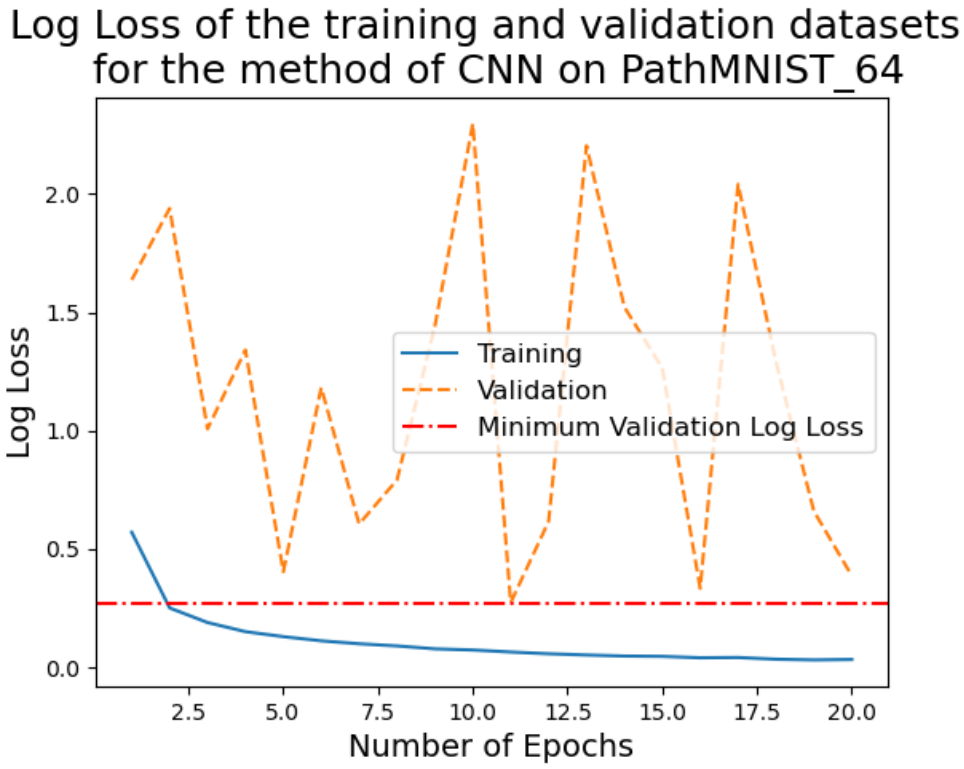


Figure A.65.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 128 × 128 resolution in Colour (RGB)

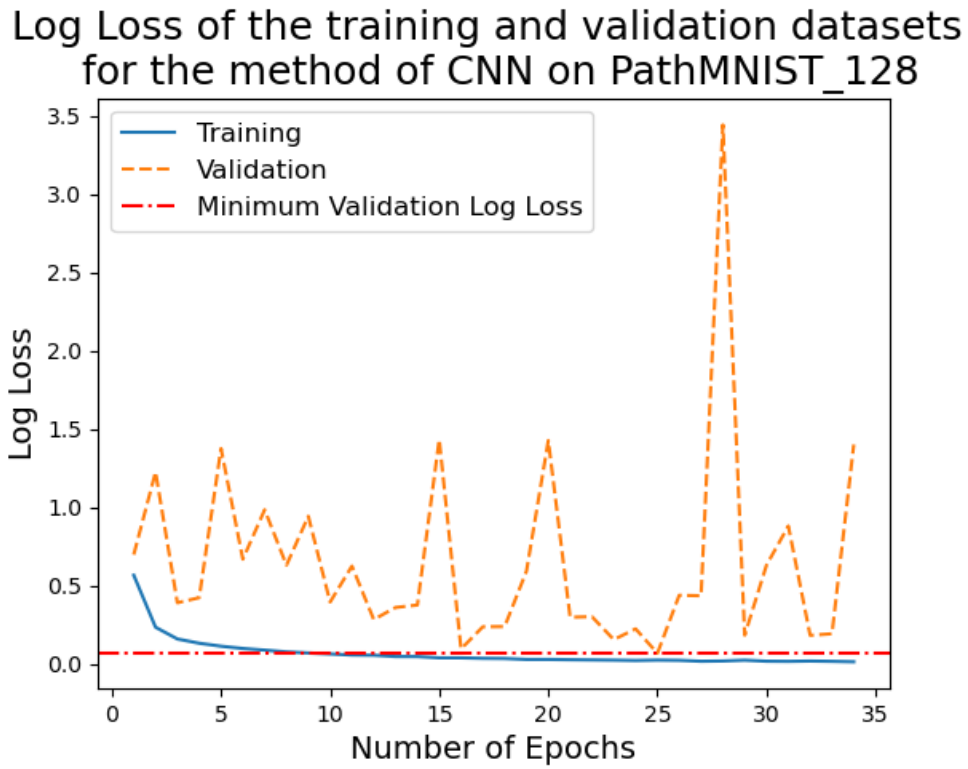
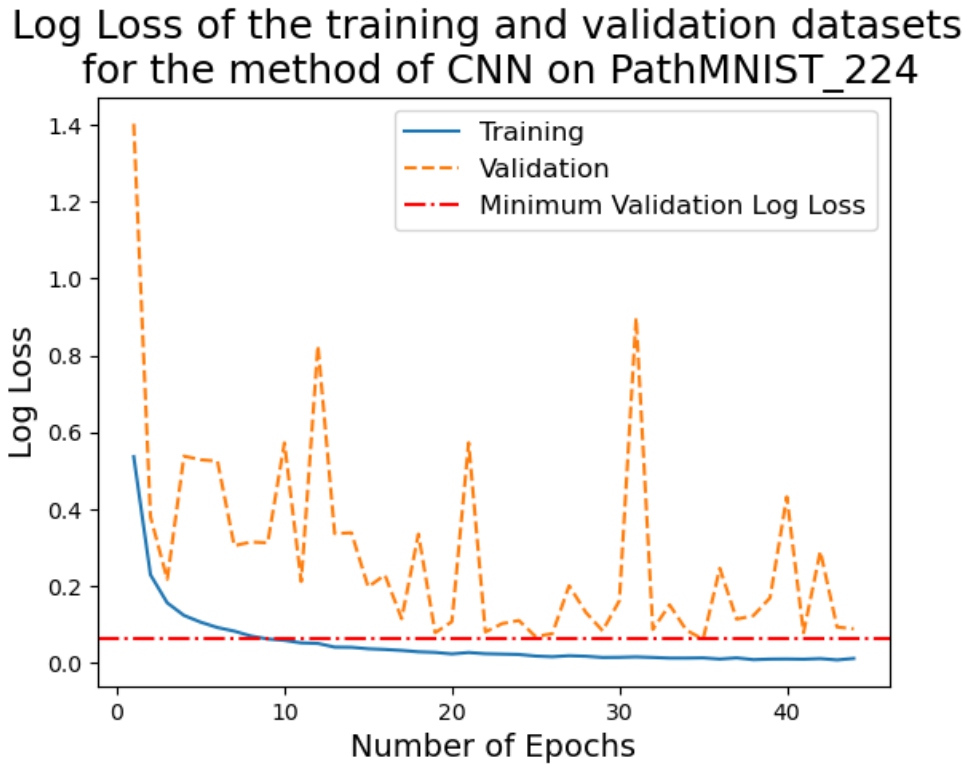


Figure A.66.: Plot of log loss during training and validation of the Convolutional Neural Network model on the PathMNIST dataset at 224 × 224 resolution in Colour (RGB)



A.3.4. XGBoost

Grayscale

The following figures display the results from training and tuning an XGBoost model for the classification of training examples using the PathMNIST dataset at different resolutions in Grayscale.

Figure A.67.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 28×28 resolution in Grayscale

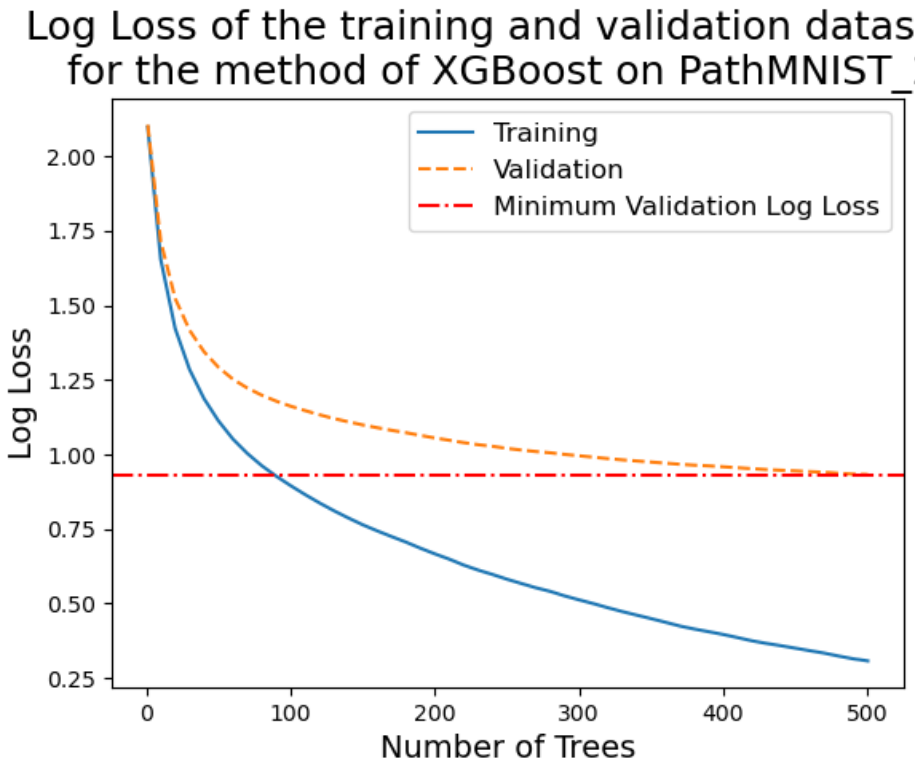


Figure A.68.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 64×64 resolution in Grayscale

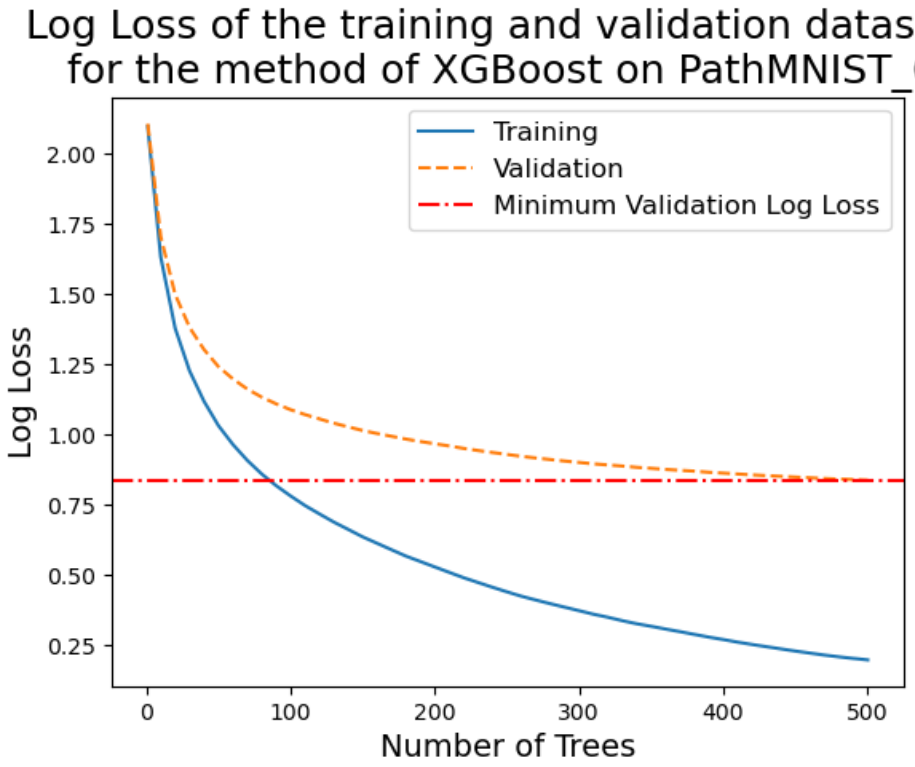


Figure A.69.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 128 × 128 resolution in Grayscale

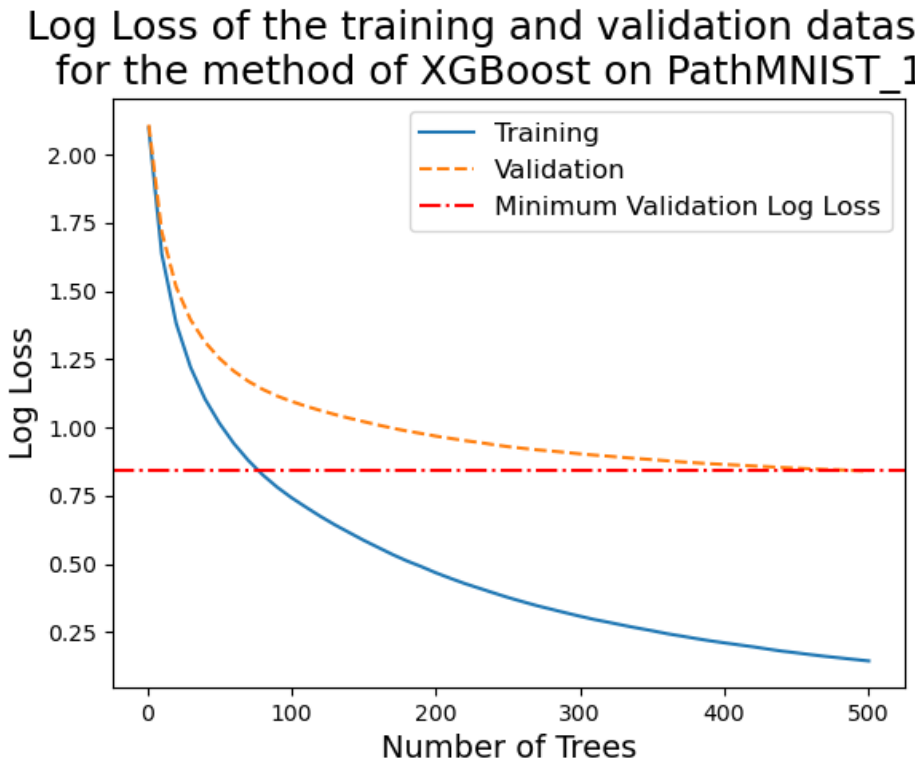
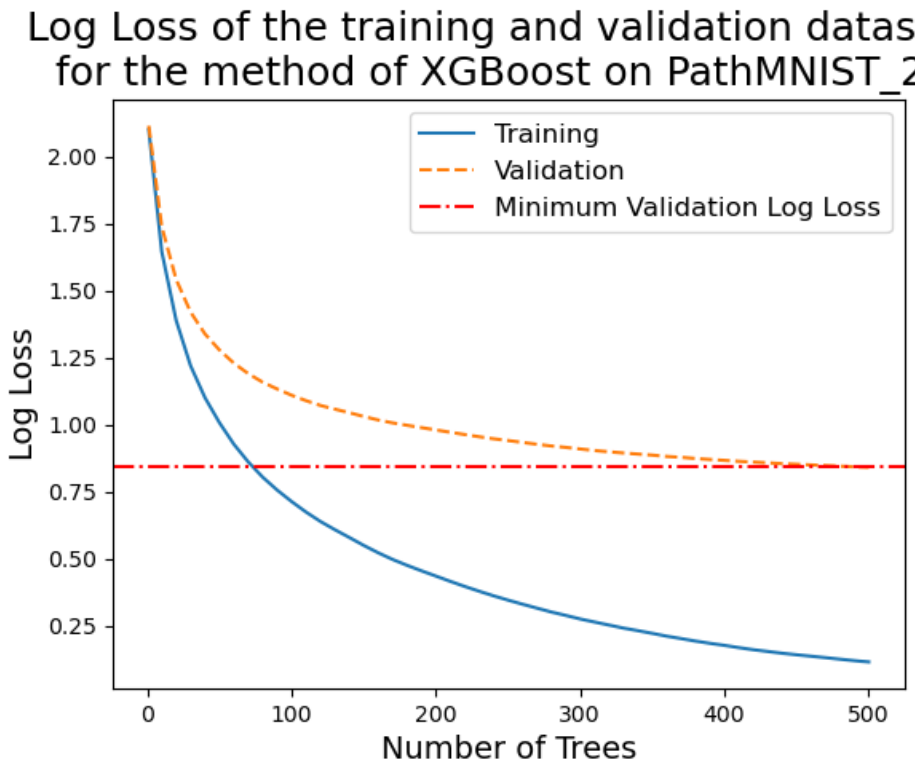


Figure A.70.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 224 × 224 resolution in Grayscale



Colour (RGB)

The following figures display the results from training and tuning an XGBoost model for the classification of training examples using the PathMNIST dataset at different resolutions in the original colour.

Figure A.71.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 28 × 28 resolution in Colour (RGB)

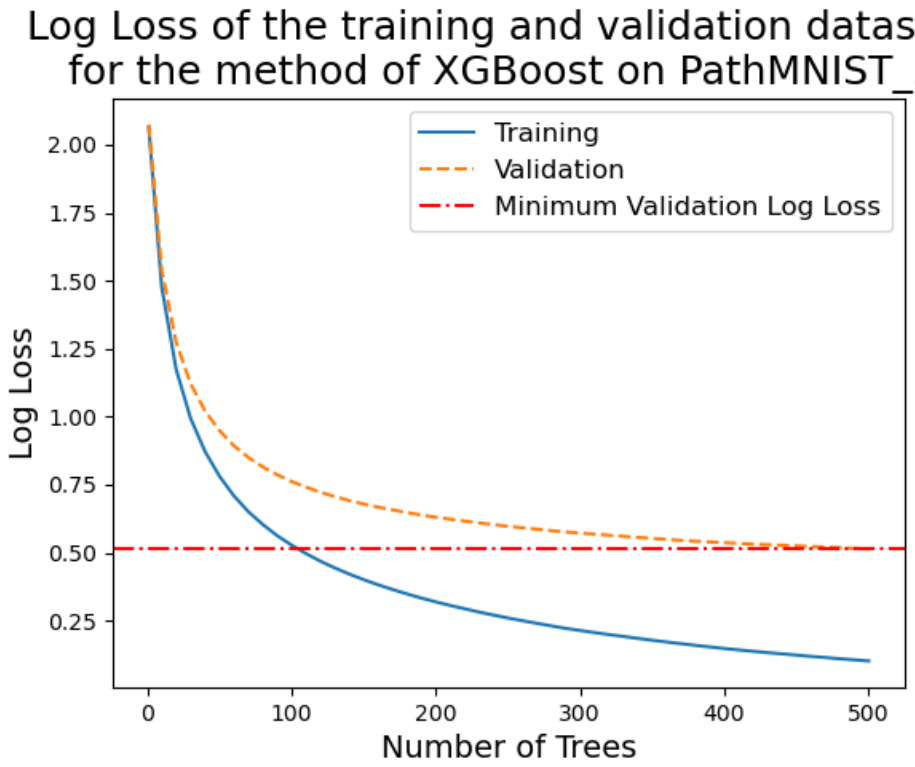


Figure A.72.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 64 × 64 resolution in Colour (RGB)

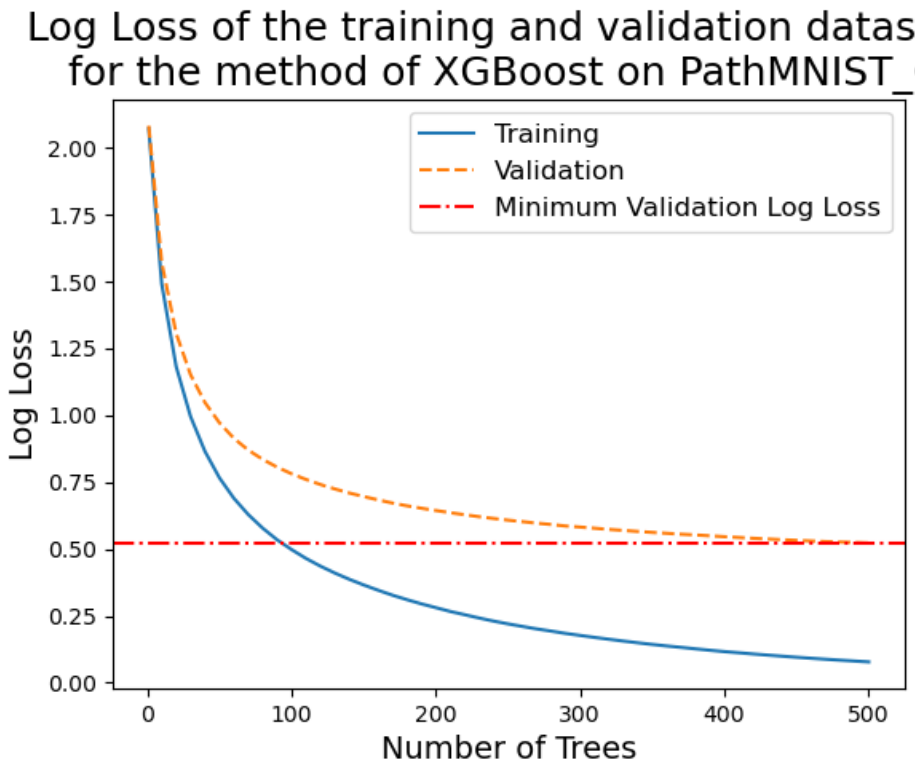


Figure A.73.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 128 × 128 resolution in Colour (RGB)

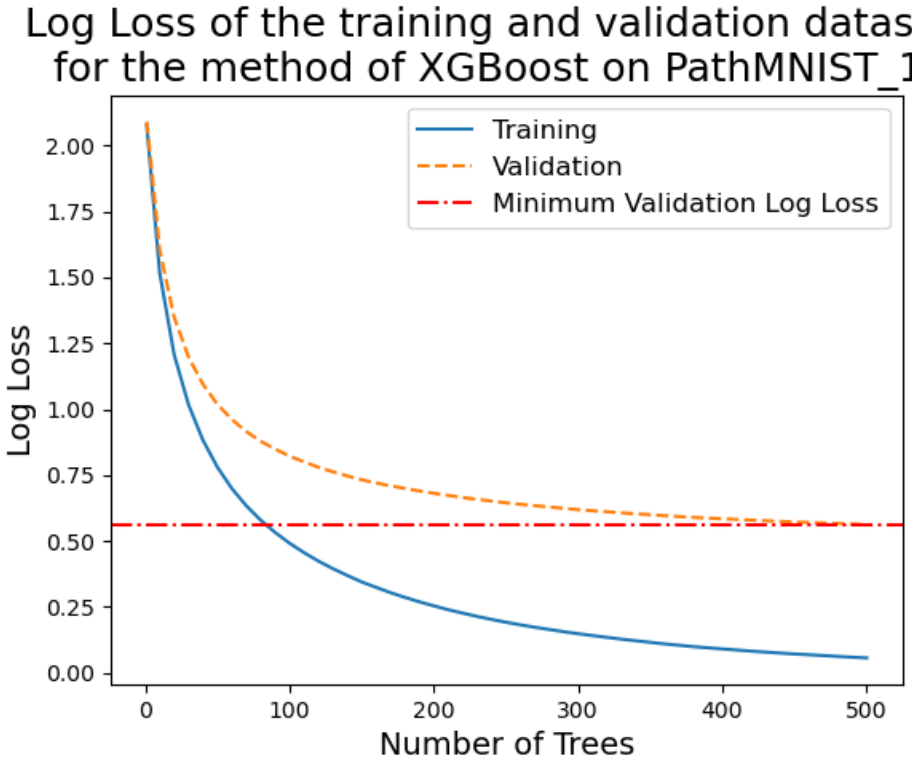


Figure A.74.: Plot of log loss during training and validation of the XGBoost model on the PathMNIST dataset at 224 × 224 resolution in Colour (RGB)