Towards Understanding Regularisation Techniques in Various Machine Learning Models for Classifying Diabetic Retinopathy

Boqian Ma (z5260890), Haoran Hu (z5366098), Jing Deng (z5213505), Kinto Wan (z5353871)
{boqian.ma, haoran.hu1, jing.deng, kinto.wan}@student.unsw.edu.au

August 3, 2022

Abstract

Machine learning techniques have been widely used for disease diagnosis in the medical industry. In this paper, we investigate different image classification methods in the context of diabetic retinopathy. Inspired by recent advancements in the field of deep learning, we compare results from traditional methods with an extended version of LeNet as an extension of this paper. Results show that traditional machine learning methods are on par of LeNet on classifying diabetic retinopathy.

1 Introduction

Diabetic Retinopathy (DR) is the world's leading cause in vision impairment and blindness. In 2015, it has affected an estimated 415 million people worldwide, with an expectancy to rise to 642 million by 2040. This is likely due to the rising prevalence in diabetes and the increase in life expectancy of diabetics [MGW07, Sha08, SBC+19]. With this disease being so prevalent, it is important to have an effective diagnosis and hence providing correct treatment plans. However it is quite difficult as the disease tends to show few symptoms until it is in later stages. Five stages of DR have been proposed to use for diagnosis. Each of these stages shows gradually worsen symptoms. Traditional approaches of identifying DR have mostly been manual inspection based on the clinician's experience. Such approaches are labor intensive and error prone. Therefore it is essential to consider cheaper and more accurate alternatives.

An opportunity to apply machine learning in this context, with an abundance in labeled data images, can be found on Kaggle's Diabetic Retinopathy Detection challenge [kag].

The goal of this challenge is to find a method, using machine learning techniques or otherwise, to accurately classify and different retina images of the eye into their respective classes (0 – No DR, 1 – Mild, 2 – Moderate, 3 – Severe, 4 – Proliferative DR).

In this paper, we review different machine learning classification models and their regularisation techniques in the context of classifying DR images. We first compare the effectiveness of traditional machine learning algorithms such as multi-class logistic regression, supported vector machines and Naive Bayes. We use these results as baselines to a more modern classification technique - convolution neural networks. More specifically, we focus on extending on Yann LeCun's LeNet [LBBH98] by applying modern regularisation techniques. We also discuss implementation details such as the data pre-processing, and evaluation metrics used for this experiment.

This paper is divided into the follow sections:

1. We first introduce related work about existing models and techniques in Section 2.
2. Next, implentation details are discussed in Section. 3.
3. Then, detailed exploration of individual models and their respective regularisation techniques are conducted.
4. Finally, we conclude the project with discussion of of the results in Section. 8

2 Background and Related Work

2.1 Machine Learning Algorithms for Image Classification

2.1.1 Mutinomial Logistic Regression

Multinomial logistic regression [Böh92] is a simple extension of binary logistic regression that allows more than two classes of dependent variable. It also uses maximum likelihood estimation to determine the probability of classes.

2.1.2 Supported Vector Machines

A supported vector machine (SVM) [P+99]is a supervised ML model capable of performing linear or non-linear classification. The model takes in training data points what outputs the hyperplane (decision boundary) that best separates different classes. SVMs models usually requires limited data and can be trained efficiently.

2.1.3 Naive Bayes

The Naive Bayes [R+01] algorithm is a classification technique based on Bayes theorem with an assumption of independence among predictor. Naive Bayes classifiers have high scalability and requires a number of parameters linear in the number of variables in a learning problem.

2.1.4 Convolution Neural Networks

Convolutional networks (convolutioanl neural networks or CNNs) [LLY+21]are a type of neural networks designed for learning sequential grid-like data such as time-series data (1-D), images (2-D), volumes (3-D). CNNs employs the mathematical operation called **convolution** to process data. CNNs have been widely used to solve computer vision problems in areas such as classification, object detection, motion estimation, etc.

2.2 Overfitting and regularisation

Overfitting is a common challenge when training machine learning models [GZM20]. It happens when the model training process fits its parameters to well to the training data, which causes failure to generalise to the testing data. Hence receiving lower testing accuracy.

Regularisation is a set of techniques used to reduce overfittiing in the training process. Many of the techniques are model-specific. Applying the correct regularisation techniques can improve the model training process and hence achieve better results.

2.3 Evaluation Metric: Accuracy

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated by dividing the number of correct predictions by the number of total predictions.

$$\text{Accuracy} = \frac{\text{\# of Correct Predictions}}{\text{\# of Total Prediction}}. \tag{1}$$

3 Diabetic Retinopathy Dataset

The data set we used contains a total of 8408 labelled instances spreading across 5 classes. Each class represents the severity of DR (0 being no symptom and 5 being proliferative DR). An example of each class is shown in Figure. 1.

Inspecting these images, we noticed that diagnosing DR through simply by inspection is a tedious process. During the early data exploration stages, the first problem we encountered was the uneven distribution of each example. (see table). Initially, we sampled uniformly for our training set. However, this approach is does not reflect on the skewed distribution in testing time, where the majority of retina images belong to the "No DR" class. Therefore, we decided use the distribution of the test set for our training set as it reflects the reality more closely.

Class	Number of Examples
0 (No DR)	6150
1 (Mild)	588
2 (Moderate)	1283
3 (Severe)	221
4 (Proliferative)	166

Table 1: Training set data example count.

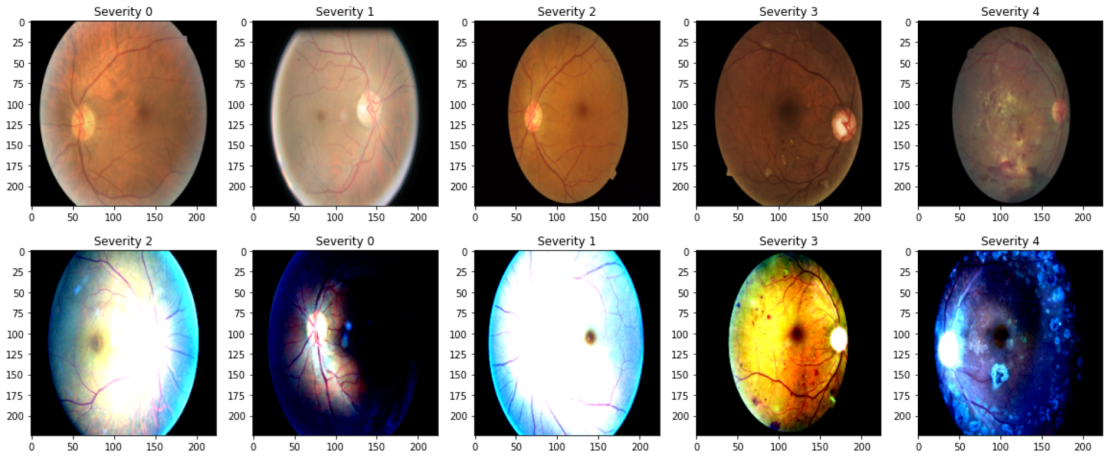


Figure 1: Some samples taken from the dataset. Top row shows the unnormalized samples. Bottom row shows featured-wise normalised samples.

3.1 Data Pre-processing

We followed standard image pre-processing techniques for our experiments. Note that each example image has three color channels (red, green, and blue) First, we resized all examples to 224×224 pixels for consistency purposes. Secondly, while training, we apply a random horizontal and vertical flipping outlined in [SK19] in an attempt to reduce overfitting by providing more training examples [PW17]. Random flipping is applied to every image in a training batch with a 50% probability. Then, we apply feature normalisation and standardisation on the training and testing. All images are normalised and standardised using the mean and variance of the training set. Keep in mind that each color channel is normalised and standardised independently. Lastly, we used a 80/20 split for our training and testing datasets.

4 Multinomial Logistic Regression

What is Logistic Regression

Logistic regression [LaV08, DeM95] is one of the machine learning techniques belonging to the regression family, it is defined by equation 2

$$y = \frac{e^(\beta_0 + \beta_1 x)}{1 + e^(\beta_0 + \beta_1 x)} \tag{2}$$

where x is the input value, y is the predict output, β_0 is the bias or intercept term and β_1 is the coefficient for the single input value (x).

The regression technique family is known to be flexible in the field of medical research due to their ability to measure, draw connections between observations and predicting results. Logistic regression is particularly efficient when it is working with a range of independent variables on a binary classification outcome to examine each variable's weighting in the model. While logistic is a very powerful technique,

its output accuracy is heavily dependent on variable selection and suitable model building logic as well as validation method.

Why Logistic Regression

Knowing the strength and limitations of logistic regression along with other options for multi-class image classification models, we still consider logistic regression to be a perfect fit for a baseline model for several reasons.

1 Simplicity

Logistic regression algorithm being a light weight model, it can be relatively easier to implement compared to other models, in fact external packages such as **sklearn** provides built-in functions to efficiently interrupt and train the model given data. Along with the logistic regression model's simplicity, it comes with relatively lower computation power. This not only enables us to quickly produce a baseline model, it also gives us the flexibility to test any pre-processing methods and up-scale our data size without any concern.

2 Multi-class Classification

While logistic regression is primarily used for binary classification, it can easily be modified to support multi-class classification problems which is particularly helpful for our selected problem as we will be looking to classify between 5 classes.

3 Model output

While the logistic regression model is not particularly effective for non-linear problems, it provides a reasonable prediction accuracy considering its resource and time consumption. The logistic regression model is also capable of producing well-calibrated probabilities alongside classification prediction, which can be used to effectively determine if certain training examples are more accurate for the problem. This gives the logistic regression model an edge over other complex models which only output predictions.

4.1 Hyper-parameter Tuning

In multi-class logistic regression models, there are four three hyper-parameters that are worth investigating namely penalty, solver,b and the C parameter.

Solver

Solver refers to the algorithm used to solve the given optimization problem. Which means it is responsible for calculating the coefficient that goes into the model given a set of features and class. There are a range of solver algorithm we can choose from, but given we will be training our logistic regression through **sklearn** we will only consider four types of solver algorithms namely 'newton-cg', 'lbfgs', 'sag' and 'saga'.
The **newton-cg** solver algorithm is based on the well known Newton's method of estimation[LWK08] as shown in the equation 3.

$$x^{k+1} = x^k - (H(x^k))^{-1}\nabla f(x^k), \quad k = 0, 1, 2, ...n \tag{3}$$

Where x^k is the current estimation, x^{x+1} is the next estimation, $H(x^k)$ is the Hessian Matrix of the model shown in equation 4 computed using current estimation values and $\nabla f(x^k)$ is the first derivative of the model.

$$Hf(x_1, x_2, ..., x_n) = \begin{bmatrix} (\frac{\partial^2 f}{\partial x_1^2}) & (\frac{\partial^2 f}{\partial x_1 \partial x_2}) & \cdots & (\frac{\partial^2 f}{\partial x_1 \partial x_n}) \\ (\frac{\partial^2 f}{\partial x_2 \partial x_1}) & (\frac{\partial^2 f}{\partial x_2^2}) & \cdots & (\frac{\partial^2 f}{\partial x_2 \partial x_n}) \\ \vdots & \vdots & \ddots & \vdots \\ (\frac{\partial^2 f}{\partial x_n \partial x_1}) & (\frac{\partial^2 f}{\partial x_n \partial x_2}) & \cdots & (\frac{\partial^2 f}{\partial x_n^2}) \end{bmatrix} \tag{4}$$

While the Newton method is capable of converging to roots accurately and is compatible to multi-dimensional problems, its efficiency decrements exponentially as the dimensions of the problem increase, since the Hessian matrix requires both first and second derivatives to be computed for every features as shown above in equation 4.

The **lbfgs** solver algorithm on the other hand is based on the coordinate descent approach[YHL11] shown in equation 5.

$$\arg\min_{x} f(x) = f(x_1, x_2, ..., x_n) \tag{5}$$

Where $x \in \mathbb{N}^n$ is the vector variable of f, x_i are scalar coordinates of x and function $f : \mathbb{R}^n \to \mathbb{R}$ is continuous.

The coordinate descent algorithm aimed to minimize a multi-variate function by progressively solving the optimization problem. While the coordinate decent algorithm is particularly convenient in certain type of problems, it has a few limitations. Firstly, due to the nature of coordinate descent the algorithm will try to reach the minimum by moving towards one direction at a time which mean the algorithm could fail to reach the minimum. Additionally, the coordinate descent algorithm does not support parallel training which mean this approach will be extremely time consuming for large data sets. Most importantly, the coordinate descent algorithm is not compatible to problems outside of multi-class logistic regression.

Lastly, both **sag** and **saga** solver algorithms are both stochastic average gradient descent based approaches. The stochastic average gradient descent algorithm [ML18] is a well known variation of the gradient descent[JT18] shown in equation 6.

$$w = w - \alpha \nabla_w J \tag{6}$$

Where w is the weight of the model, α is the chosen learning rate and J is the cost function of the model.

Compared to standard gradient descents algorithm, stochastic average gradient descent features lower time and memory consumption while delivering accurate results. This mean **sag** and **saga** solver algorithms are exceptionally beneficial for problems with large data sets. The **saga** solver algorithm is slightly superior to the **sag** solver algorithm due to the additional L1 regularization support

Applying these different solver algorithm in our logistic regression model with all other parameters set to default, we observe the results shown in figure 2.

Investigating the results, we found that while the Newton method solver failed to converge (as the number of dimensions in image classification problems are generally higher than ordinary classification problems), we were able to obtain relatively similar results use other coordinate and gradient descent based solver algorithm. Considering the differences between the obtained results and time taken to produce these results, these observations smoothly aligned with all the properties we outlined above. However, as the differences between the obtained accuracy are too marginal, we were not able to claim if a certain algorithm is more superior than the others for the given problem. Therefore, we decided to proceed with the **lbfgs** solver algorithm and **sag** solver algorithm for further hyper parameter turnings. Although the **saga** solver algorithm did come close, it was consistently under performing compared to the **sag** solver algorithm after a few rounds of experiments.

Penalty

Penalty is the type of regularization. As we have chosen to use the **lbfgs** and the **sag** solver algorithm above, we are restricted to use penalty terms that are compatible with it. In other words, we can either use the L2 penalty term or not use any. Upon applying both penalty terms into our models, we were able to observe the results shown in figure 3.

Which is what we expected prior to the test, as L2 is the typical regularization used by ridge regression which keeps coefficients non-zero while shrinking them, it might not be the perfect fit for the type of problem we are solving but it was sufficient to outperform models that do not adjust coefficients at all.

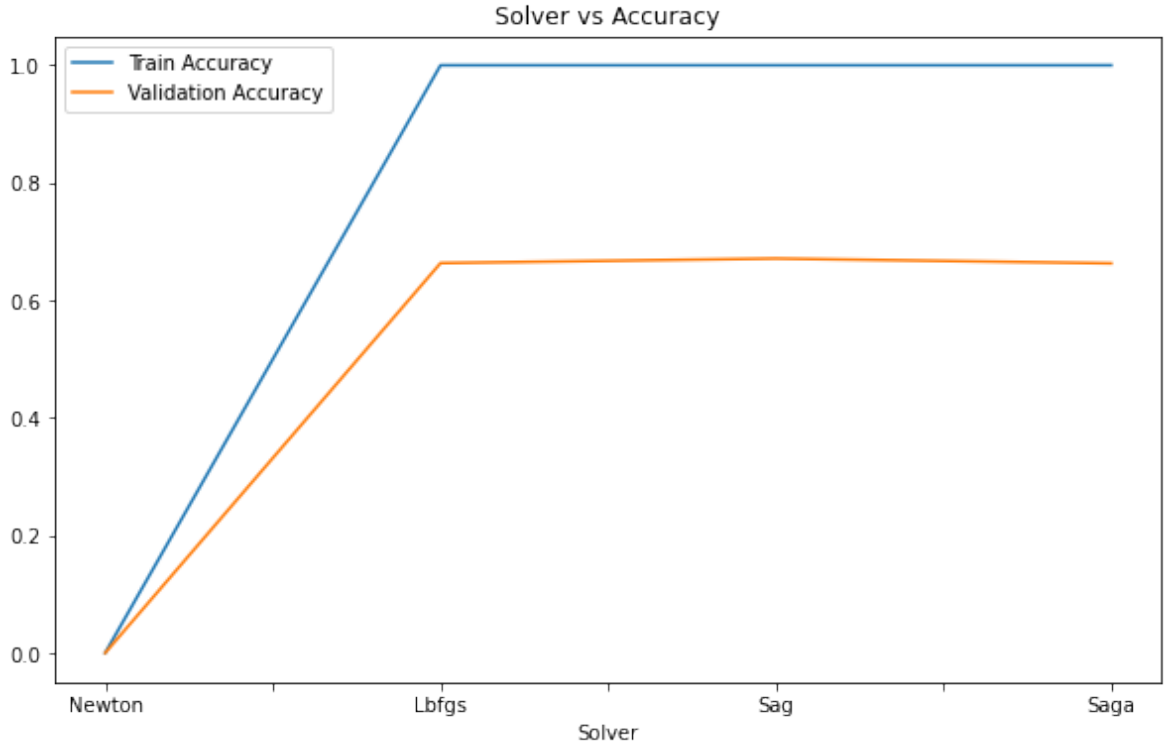


Figure 2: Accuracy achieved by different Solvers

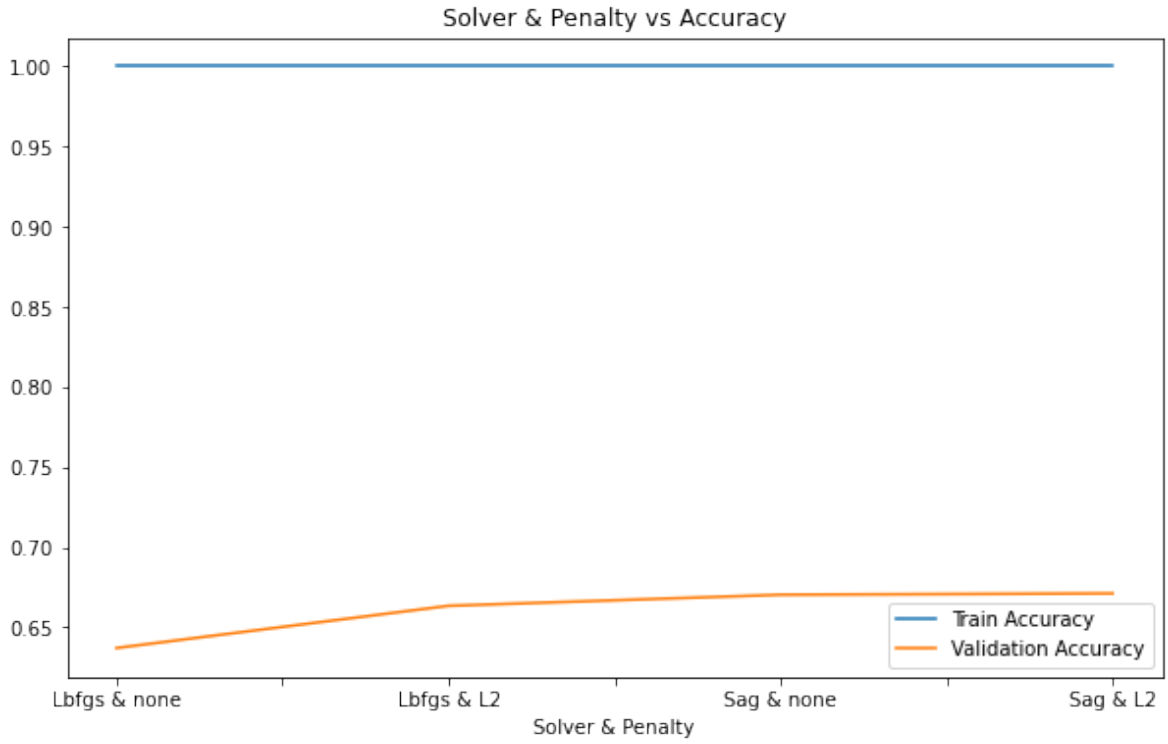
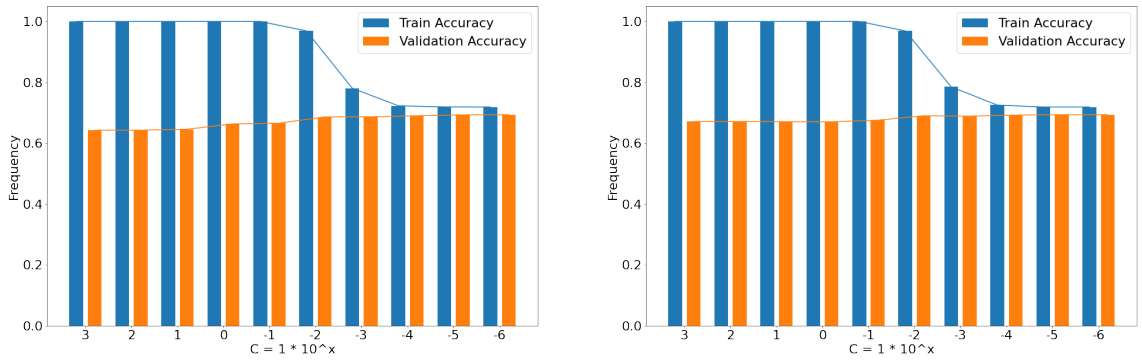


Figure 3: Accuracy achieved by different Penalty Solver combinations

C

The C parameter denotes the inverse of regulation strength in a logistic regression model, where



(a) Lbfgs Solver (b) Sag Solver

Figure 4: Accuracy achieved by different Solver C combinations

regulation is the process of constraining the size of the model coefficient. This is particularly helpful for models that are over-fitting training data (such as the model we currently have as suggested by our train and test accuracy shown above), since the over-fitting issue can be minimized by simply reducing variance of the model which can ultimately result in a model that is overall more likely to generalize future data, hence a better model.

We can increase regulation by simply changing the value of C, by convention the C value is usually tuned in power of 10s. To investigate which C value best fits our problem, we start with the C value of '1000.0' and compare the performance of the model by tuning it twelve times, and were able to observe the results shown in figure 4.a and 4.b

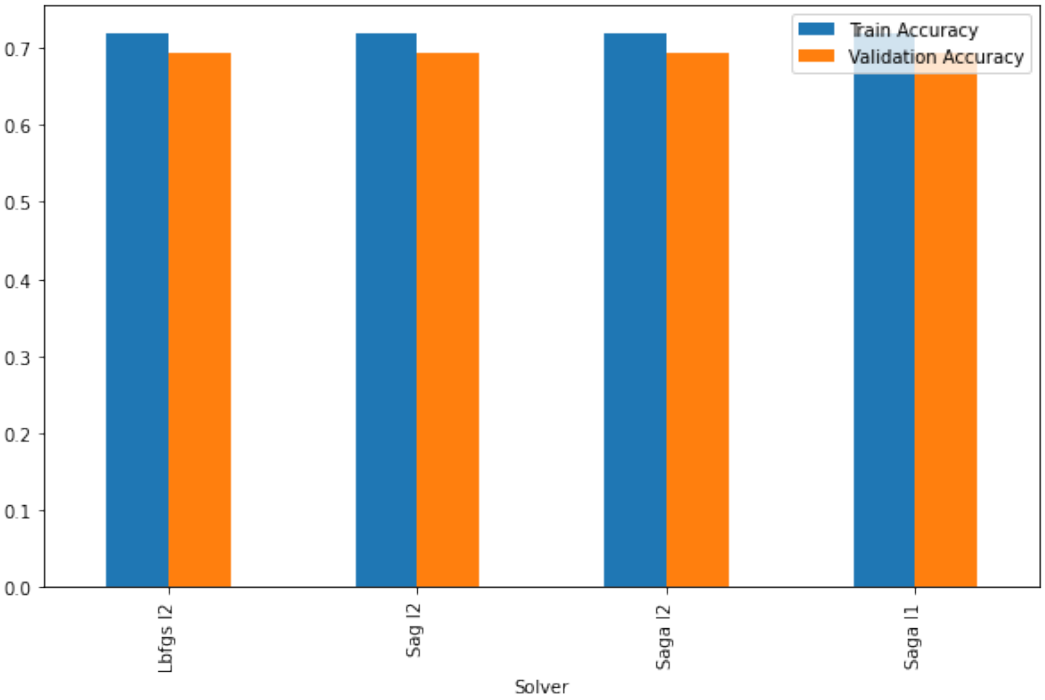


Figure 5: Accuracy achieved by different penalty solver combinations when C $=1 \times 10^{-6}$

Investigating the results, it is clear that the over fitting issue is getting less apparent as the C parameter kept decreasing for both solver algorithms, this is particularly obvious when the C parameter has the value of 1×10^{-3}. Furthermore, we were able to observe that both training and validation accuracy for **lbfgs** and **sag** solver algorithm are approaching 70% as the value of C parameter continue to decrease,

and eventually stop changing when the C parameter has the value of 1×10^{-6}. Taking all tuning procedures and corresponding results into considerations, we were able to identify increasing the scale of regularization has given us most success. Therefore we re-ran logistic regression models with **lbfgs**, **sag** and **saga** along with C value of 1×10^{-6} and we were able to observe the results shown in figure 5.

5 Supported Vector Machine

The SVM was a mathematical model introduced by Corinna Cortes and Vladimir Vapik [AKYN19]. It takes in data as input, transforms it into high-dimensional space and generates hyper-planes to separate the dataset into two or more classes. The general idea is to maximise the margin between the data and hyperplanes to create more reliable results [AKYN19]. The equation of the primal problem can be represented as :

$$\min_{\omega,\beta,\zeta} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\zeta_i, \tag{7}$$

where

$$\zeta_i = max(0, 1 - y_i(\omega^T x_i - b)) \tag{8}$$

w is the normal vector to the plane, constant b is the offset, and C is the regularisation parameter.

Ideally, we are to maximise the margins through minimising . When a sample is incorrectly classified, we penalise the function through the penalty term, C. C defines the strength of the penalty and acts as an inverse of the regularisation parameter. On top of this, the SVM model can be applied with different kernels, where selecting the correct kernel plays a crucial role in the classification accuracy. The kernel is essentially a mathematical function that splits the dataset in different ways and returns the input data in its desired form. Often, the linear kernel is best for datasets with lots of features, there is also the polynomial kernel, the radial basis function (rbf) kernel (typically for non-linear models), and sigmoid kernel (typically for neural network). Lastly, in terms of properties, the SVM model is known for to its strong ability to generalise data, despite noise which makes it strong against overfitting data [AZ21, SC21, BC20]. In addition, it also has the following properties :
- Effectiveness in high dimensional space,
- Memory efficient in the sense that it only uses a subset of training points in the decision function.
- Its versatility allowing for different kernel functions
From this, we have deemed it suitable to include the SVM model as one of our classification methods.

HyperParameter - Tuning
For the SVM, several hyper-parameters were applied for adjusting. These include: Different kernel selections = linear, radial bias function (rbf), polynomial (poly), sigmoid On top of these kernels, some of these allow for further parameter adjustments:
- The rbf, poly and sigmoid require further parameter adjustment through gamma, the kernel coefficient. The parameter defines generally how much of an influence a single training data can reach.
- The poly kernel also allows for further adjustment through the degree of the polynomial used. This generally defines the flexibility of the decision boundary. Higher value refers to a more lenient decision boundary. C : In SVM, C is the regularisation parameter. Lower C means less penalty for mis-classification. The more noisy the data, the smaller or closer to 0 it should be. Class-weight = None, Balanced. This refers to the weight of each class. If balanced then the weight of the classes would be dependent on the frequency of data in each class.

Results
Multiple experiments were conducted and the following observations were noted: For different kernels, it was found that for C-values in 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10, C-value

of 0.1 gave the highest testing accuracy results. For the Kernel: Poly and rbf gave results where the training accuracy was low, and testing around 68%. However, after checking the confusion matrix, all of the predictions were 100% skewed and were all in class 0, which was clearly incorrect. The linear kernel provided the best results. Originally, the model was found to be over-fitting with accuracy close to 100%. However, with more dataset, we found that the linear kernel stopped over-fitting and gave the best training and test accuracy. Gamma-value: auto, scale, 0.001, 0.01, 0.1, 0.5, 1.0. From comparing all of the existing values, the auto value was found to give the best results for different kernels. However, with the linear kernel giving the best result, the gamma-value is irrelevant to the model. On top of this, data normalization was applied to the dataset, which increased the final test accuracy by about 6%.
The results of the optimised model is shown in Table.2

Training Acc.	Testing Accuracy	Acc	Prec	Recall	F1	Prec	Macro Recall	Macro F1
0.936	0.736	0.736	0.736	0.736	0.736	0.659	0.479	0.538

Table 2: Metrics result of the SVM model

6 Naive Bayes

Naive Bayes classifier assumes that each feature is unrelated and independent to each other[R+01]. Although it is a poor assumption, it often works well with more sophisticated classifiers. Moreover, the accuracy of Naive Bayes is not directly correlated with the degree of feature dependencies.

Since features of the objects are assumed to be independent, if we have $X = (x_1, x_2, ..., x_n)$ be the feature vector and X is the sample that we need to classify. [Leu07]In Bayesian terms, X is considered "evidence". Let H be some hypothesis, such as X belongs to class C where $C = (C_1, C_2, ...C_k)$ indicating that there are k classes exist. For this project or any other classification problem, the main goal is to determine $P(H|X)$ which is called the posterior probability of H conditioned on X. Based on Bayes' Theorem, this probability can be expressed as $P(H|X) = \frac{P(X|H)P(H)}{P(X)}$, where $P(X|H)$ is the posterior probability of X conditioned on H, $P(H)$ is the prior probability of H and $P(X)$ is the prior probability of X. Hence, for Naive Bayes classifier, we need to maximize $P(C_i|X)$, in order to do that, we need to maximize $P(X|C_i)P(C_i)$ since $P(X)$ is the same for all samples. The classifier will predict the class label of X for each C_i, and finally it will select a class under the circumstance that $P(C_i|H)P(C_i)$ is maximized.

6.1 Advantages of Naive Bayes[APT13]

- Not sensitive to irrelevant features
- Fast, easy to implement
- Less parameters to set
- Not data-intensive
- Highly scalable

For this project, two methods are applied to implement Naive Bayes classification, namely OpenCV and PIL, the former approach will be described in detail in the following context.

6.2 ORB in OpenCV

OpenCV [BK08](Open Source Computer Vision Library) is an open-source library that contains many computer vision algorithms. It includes some core functionality that are useful for this project. For

example, ORB (Oriented FAST and Rotated BRIEF), which is a **FAST** key-point detector and after all key points are extracted, **BRIEF** descriptions of images can in turn be generated for further use of classification.

Some key operations of ORB have been involved in this project are: create, detect and compute. Following is an example output of the key points for an image after implementing these operations.

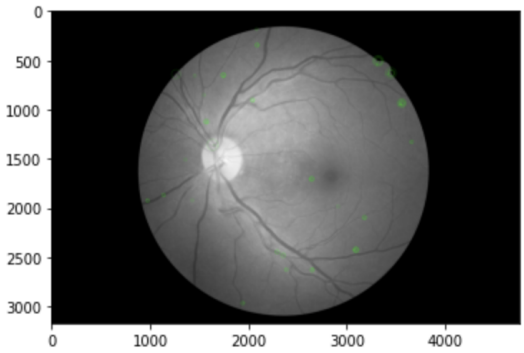


Figure 6: Key Points Detected by ORB

Note the size of the image shown in the graph is the original size in the dataset, not the size we use for this project.

Green dots are representing key points, can be observed from the graph and as we can see, they are all over the eyeball. As we proceed to operate on all training images, we notice that the number of key points for different images in not identical.

After all images have been processed, two more essential elements can be resulted, namely codebook and image-feature matrix. Examples are shown below.

```
codebook:  [[231.51502146 193.19849785 155.74785408 ... 221.15879828 189.93133047
   37.8583691 ]
 [129.91814947 195.21174377 123.62633452 ... 203.40035587 220.25266904
  224.38967972]
 [161.40730337 179.28651685 161.03370787 ... 234.89185393 139.64044944
   31.97331461]
 ...
 [209.12255407 188.58702369 157.33985582 ... 109.75798146 122.46343975
   26.75180227]
 [224.97536946 215.77668309 151.97044335 ... 121.20853859 178.10673235
   31.20689655]
 [176.52022059 183.77757353 174.52205882 ...  88.51102941 155.37132353
  149.73713235]]
```

Figure 7: Codebook Generated by Detecting Set of Images

```
img_features:  [[ 6.  1.  2.  4.  0.]
 [10.  8.  3. 15.  0.]
 [ 7.  2.  0.  8.  0.]
 ...
 [ 1.  2.  0.  3.  2.]
 [ 9.  7.  4.  7.  2.]
 [13.  5.  0. 19.  5.]]
```

Figure 8: Image Feature Matrix

Codebook is calculated based on k-means algorithm. More specifically, k-means algorithm requires a cluster size (for this project, multiple options of cluster size is tested), all data that within the cluster size is considered to be similar. In this case, we can say all key points within the cluster size will be grouped together.

After the codebook is generated, we can proceed to obtain the image feature matrix according to the codebook. The image matrix is achieved applying vector quantization, which takes codebook and the brief descriptors as inputs, each descriptor (observation) is compared with all groups in the codebook, and the codebook will assign a code to the descriptor to which its closest.

Once the codebook and the feature matrix are achieved, we can use the matrix and the training labels to fit the model, and get the estimator for that model. Then, we can use it to predict the class label of each sample and calculate the accuracy.

6.2.1 Drawbacks of ORB

Since the codebook is generated using k-means algorithm, which requires clustering, so it will not work well when the data is not well-separated. For this project, the key points of each image in the data sets can be very different, so it is not easy to determine if those points are well-separated. Also, it is not an easy job to select the optimal k value, inappropriate k values can result in empty clusters, so multiple values of k are required in order to enhance the accuracy. Besides, K-means algorithm is very sensitive to irrelevant data and noise, and in this case, key points are randomly placed, so the shape of clusters are not deterministic.

6.3 Results

For this project, different values of k have been tested, not surprisingly, both training accuracy and testing accuracy get boosted by selecting more reasonable k values.

Cluster Size	Training Accuracy	Test Accuracy
200	0.207	0.286
100	0.206	0.261
50	0.228	0.278
5	0.498	0.537
2	0.667	0651
1	0.673	0.656

Table 3: Training and testing accuracy for different cluster sizes

6.4 Discussion

For this project, all the images in the image set do not have the same size, so before ORB starts to detect and compute descriptor, all images that are being processed will be resized to the same size, which has been mentioned in the pre-processing part. Certain proportion of the images contain no information, as shown above, thus those areas cannot be be will utilized since no key points would be extracted detecting those areas, because of that, the detect operation might not be very effective. More importantly, the key-point set for each image is not identical, some images have more key points while others might only have a few. This is the primary reason that different cluster sizes can hugely affect the accuracy. If an image only has very few key point and the cluster size is set to be 200, then those points are basically the same point, thus when making predictions, the features become ambiguous which result in low accuracy. By emphasizing on smaller region of key points, more and more key points would actually be focused on, then image features are sophisticated so that higher matching accuracy can be achieved.

7 Extending LeNet with modern techniques

LeNet [LBBH98] was one of the first successful attempts of using neural networks to classify images. It uses stochastic gradient descent and back propagation to learn the parameters. Since then, many improved techniques have been proposed in this field with the aim to achieve better results.
Inspired by LeNet and recent advancements, we decided to modify the origional LeNet to fit our usecase, with a couple of improvements. First, we consider the ADAM optimizer [KB14]. Then, we apply two regularisation techniques. Namely, Dropout [SHK+14] and Batch Normalisation [IS15]. We study the effectiveness of these improvements by empirically by comparing with the original LeNet model.
Further, since LeNet was designed for gray scale images (1 color channel), we also modified the network to fit RGB color images with 3 color channels.
We begin by introducing these improvements individually.

7.1 ADAM optimizer

We compare the performance of stochastic gradient descent (SGD) with ADAM. Unlike SGD, which maintains a single learning rate for all weight updates, ADAM computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients[KB14]. It combines the advantages of Adaptive Gradient Algorithm [DHS11] and Root Mean Square Propagation [Gra13].

7.2 Regularisation Techniques

Two challenges arose in training deep learning models are overfitting and long convergence time. Dropout and batch normalisation are two well-recognised approaches to tackle these two challenges.

7.2.1 Dropout

Dropout is proposed to randomly change the network architecture, to minimize the risk that the learned weight values are highly fitted to the given training data, and therefore causing overfiting. Intuitively, as the name suggests, dropout changes modifies fully connected layers by "turning off" certain neurons given by a probability. It closely simulates ensemble learning but without specifically specifying individual models.

7.2.2 Batch Normalisation

Batch normalisation [IS15] significantly reduces training time by normalising the input of each layer in the network, not only the input layer (that is normalised before being fed into the model). This approach allows the use of higher learning rates, which in turn reduces the number of training steps the network need to converge. Batch normalisation can be implemented simply by adding an additional layer after the layers of choice. It has shown regularisation great effects in the training process [LWSP18].

7.3 Results

The empirical results are show in Table.4. After 20 epochs, we observed the following:

1. The baseline model showed the highest variance between training and testing accuracy, this is reasonable because it has no additional regularisation techniques applied to it.

2. Models with ADAM optimiser converged quicker than those that used SGD because ADAM implements adaptive learning rate updates that allows for faster convergence.

3. The model with SGD and dropout showed the least variance in training and testing. The model with the second lowest variance is ADAM with dropout. This showed that dropout is a more effective regularisation technique than batch normalisation in this case.

Optimiser	technique	Train Accuracy	Test Accuracy	Variance in accuracy
SGD	N/A	**0.8536**	0.6912	0.1624
SGD	dropout	0.7217	**0.6921**	**0.0296**
SGD	batch norm	0.7246	0.6825	0.0412
ADAM	dropout	0.7221	0.6904	0.0317
ADAM	batch norm	0.7259	0.6784	0.0475

Table 4: Training and testing accuracy of each of the LeNet-based models after 20 epoches. The best scores in each column are bolded.

It is also important to note that the images in our dataset is bigger in size than what LeNet was designed for. LeNet was designed for 32×32 instead of 224×224. We suspect this is the reason for the high bias and variance nature of the results from these models.
Overall, dropout and batch normalisation did not show big difference in terms of model accuracy. However, they did show good regularisation effects on the training process.

8 Results Discussions

Overall, despite our best efforts, none of the models in the project achieved promising results. Multinomial logistic regression, SVM, and modified LeNet achieved similar testing accuracy with SVM achieving the highest and Naive Bayes achieving the lowest out of all models.
With careful regularisation parameters selections, we were able to generate models with low variance in training and testing accuracy with the exception of the SVM model. Further investigation is required to decrease the variance.
It was unexpected that the modified-LeNet did not perform significantly better than other models. This most likely because the LeNet architecture was designed for much smaller images, which in turn means it requires more parameters and layers to better fit our data set.

Models Name	Train Accuracy	Test Accuracy	Variance in accuracy
Multinomial Logistic Regression	0.719	0.694	0.025
SVM	**0.937**	**0.736**	0.201
Naive Bayes	0.673	0.656	**0.017**
Modified LeNet with dropout	0.7217	0.6921	0.0296

Table 5: Result comparison of different models. The best score from each column is bolded.

9 Conclusion

In this paper, we explored different classification machine learning algorithms in the context of diabetic retinopathy. More specifically, we focused on exploring regularisation techniques to increase model performance.
However, none of the models showed promising performance. This is mainly due to the lack of computational resource we were provided.
Overall, the results were of satisfaction. Our best model performed with 0.736 test accuracy. If we were to compare this with other contestants in the Kaggle competition, we would receive a silver medal (coming 26th out of 661 different teams). This is a very reasonable attempt given that we had computational limitations, such only being able access to only 1 of the 8 datasets.

10 Further work

There are several areas of improvements interested readers could explore

1. In the data preprocessing stage, we resized each examples to 224×224 due to time constraints. This could potentially lead to loss of important features in training time. Examining different image sizes could be useful.

2. For our project, due to computational constraint, we only used one set of data. However, there is potential for improvement in classification accuracy if all 8 dataset were used.

3. The SVM could have been further improved upon. For example, if there were faster computers, the probability = True parameter could have been applied, which would take much longer, however the accuracy would most likely see improvement.

References

[AKYN19] Nur Izzati Ab Kadera, Umi Kalsom Yusofa, and Syibrah Naima. Diabetic retinopathy classification using support vector machine with hyperparameter optimization. *Int. J. Advance Soft Compu. Appl*, 11(3), 2019.

[APT13] Ahmad Ashari, Iman Paryudi, and A Min Tjoa. Performance comparison between naïve bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(11), 2013.

[AZ21] Mohamed M Abdelsalam and MA Zahran. A novel approach of diabetic retinopathy early detection based on multifractal geometry analysis for octa macular images using support vector machine. *IEEE Access*, 9:22844–22858, 2021.

[BC20] Manoj Kumar Behera and S Chakravarty. Diabetic retinopathy image classification using support vector machine. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–4. IEEE, 2020.

[BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[Böh92] Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the institute of Statistical Mathematics*, 44(1):197–200, 1992.

[DeM95] Alfred DeMaris. A tutorial in logistic regression. *Journal of Marriage and the Family*, pages 956–968, 1995.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[Gra13] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[GZM20] Christian Garbin, Xingquan Zhu, and Oge Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, 79(19):12777–12815, 2020.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[JT18] Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*, 2018.

[kag] Diabetic retinopathy detection.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LaV08] Michael P LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Leu07] K Ming Leung. Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*, 2007:123–156, 2007.

[LLY+21] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[LWK08] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust region newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9(4), 2008.

[LWSP18] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*, 2018.

[MGW07] Quresh Mohamed, Mark C Gillies, and Tien Y Wong. Management of diabetic retinopathy: a systematic review. *Jama*, 298(8):902–916, 2007.

[ML18] Gunasekaran Manogaran and Daphne Lopez. Health data analytics using scalable logistic regression with stochastic gradient descent. *International Journal of Advanced Intelligence Paradigms*, 10(1-2):118–132, 2018.

[P+99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[PW17] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[R+01] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[SBC+19] Charumathi Sabanayagam, Riswana Banu, Miao Li Chee, Ryan Lee, Ya Xing Wang, Gavin Tan, Jost B Jonas, Ecosse L Lamoureux, Ching-Yu Cheng, Barbara EK Klein, et al. Incidence and progression of diabetic retinopathy: a systematic review. *The lancet Diabetes & endocrinology*, 7(2):140–149, 2019.

[SC21] Nagur Shareef Shaik and Teja Krishna Cherukuri. Lesion-aware attention with neural support vector machine for retinopathy diagnosis. *Machine Vision and Applications*, 32(6):1–13, 2021.

[Sha08] Chirag A Shah. Diabetic retinopathy: A comprehensive review. *Indian journal of medical sciences*, 62(12):500–519, 2008.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[SK19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[YHL11] Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1):41–75, 2011.