Deep learning for biological image classification

Jacob Holmshaw

School of Mathematics and Statistics
University of Sheffield

The University Of Sheffield.

Dissertation submitted as part of the requirements for the award of
MSc in Statistics, University of Sheffield, 2020–2021

Acknowledgements

Thank you to my supervisor, Alex, for providing guidance and feedback on this project. I would also like to thank my co-supervisor, Ian, for devoting much of his time and support to me throughout the project. Finally, thanks to my support worker, Megan for providing writing support throughout the project.

Lay Summary of the Dissertation

This project aimed to create a computational model to classify between images of chicken embryos. These images were divided into three classes of development, in which there are small changes between them. Therefore, it is difficult to distinguish these images into a specific class by human visualisation. Using computational resources with machine learning has the potential to distinguish these images efficiently and to high accuracy. Achieving this is important as it would allow further research into the developing hypothalamus of the chicken. Many aspects of the development of the chicken are strongly conserved with humans. Thus, any knowledge gained will apply to human development. This knowledge is important in furthering the understanding and developing of treatment of hypothalamus related diseases.

The data consisted of 151 images. Despite the potential of using machine learning for this project. Common problems arise with image classification. For example, when the original data is small in number, the model performs well on data that it uses to train a model but far worse on unseen data. Techniques to reduce this difference in performance were explored. This problem can be caused by feature differences in the images which are not useful in identifying the stage of the embryo. Therefore, creating copies of the images by transforming them, (such as rotating, flipping, cropping the image), can remove the differences between the images. Removing the differences means the model does not focus on these differences and use them to make classifications. Various transformations were tested based on visualising differences in the images. The model has various parameters that can be changed, which can alter the results. Therefore, many different parameters were tested to find the ones that maximised results. Additionally, pre-processing the images to reduce differences were also tested in some scenarios where it was appropriate to alter the original data.

Grayscaling each image and applying 5° rotations (up to 355°) improved the accuracy with which the model can classify images correctly. The improvement was because removing colour and angle differences in the images allowed the model to identify other features to perform classification. This metric was increased further by tweaking the model's parameters. This tweaking highlighted that the data could cause very different results if specific images were used to test the accuracy of the model instead of being used to train the model. Analysis of the images that were commonly classified wrongly revealed that applying

histogram equalization (a modification that improves the contrast of images) improved the validation accuracy further. Testing further transformations to the data found that randomly covering up parts of the image with a grey box, known as cutout, caused the accuracy to increase.

The project proved that a model can classify microscopic images into stages of development, with an accuracy of 76%. Further research could focus on improving the accuracy of this model, which would enable further knowledge of the hypothalamus to be discovered. This accuracy can be achieved with more access to computational resources, exploring more transformations of the images and finding more methods for removing insignificant features in the data. Analysing saliency maps can show which areas of the image were paid attention to. This analysis could reveal feature differences that could be eliminated through transformations, which would improve the choices of augmentation.

Contents

Acknowledgements i

Lay Summary of the Dissertation iii

1 Introduction 3
- 1.1 Background, motivation and previous work . . . 3
  - 1.1.1 The hypothalamus . . . 3
- 1.2 Machine learning and deep learning . . . 4
  - 1.2.1 Machine learning . . . 4
- 1.3 Overfitting . . . 6
  - 1.3.1 The overfitting problem . . . 6
  - 1.3.2 Data augmentation . . . 6
  - 1.3.3 Regularization . . . 6
- 1.4 Hyperparameter optimisation . . . 7
- 1.5 Cross-validation . . . 8
- 1.6 Aims . . . 8

2 Methods 9
- 2.1 Data . . . 9
- 2.2 Standard neural network . . . 9
- 2.3 Model construction . . . 10
  - 2.3.1 Pre-processing of data . . . 10
  - 2.3.2 Model specification . . . 11
  - 2.3.3 Compiling the model . . . 11
  - 2.3.4 Training the model . . . 11
- 2.4 Base augmentation . . . 12
- 2.5 Base model hyperparameter optimisation . . . 12
- 2.6 Data augmentation . . . 14
  - 2.6.1 Choice of augmentations and values . . . 14
  - 2.6.2 Augmentation adjustment . . . 15
- 2.7 Augmented model optimisation . . . 16
- 2.8 Error analysis . . . 16
- 2.9 Testing further pre-processing . . . 16
  - 2.9.1 Histogram equalization . . . 16
  - 2.9.2 Pre-processing less magnified images . . . 17

2.10 Obtaining final metric 17
2.11 Python packages used 17

3 Results **19**

3.1 Rotation augmentation and optimisation improves classification accuracy of neural network-based image classifier 19
3.2 Histogram equalization improves classification accuracy of neural network-based image classifier 20
3.3 Cutout and optimisation improve classification accuracy of neural network-based image classifier 21
3.4 K-fold cross-validation of augmentations shows cutout classification accuracy of neural network-based image classifier 24

4 Discussion **27**

4.1 Aims 27
4.2 Discussion 27
4.2.1 Base model 27
4.2.2 Base model optimisation 27
4.2.3 Data splits/folds 28
4.2.4 Histogram equalization 29
4.2.5 Augmentations 30
4.2.6 Optimisation of cutout data set 32
4.2.7 Overall accuracy of classifier 33
4.3 Further work 33
4.3.1 Computational resources 33
4.3.2 Medical image enhancement 34
4.3.3 Improving background elimination 34
4.3.4 Saliency analysis 35

5 Conclusion **37**

A Research Ethics Approval **39**

Chapter 1

Introduction

1.1 Background, motivation and previous work

1.1.1 The hypothalamus

Developmental biology is the study of how humans and organisms grow from a single cell. In this area, we focus on the hypothalamus. This region of the brain maintains stable internal, physical and chemical conditions. Understanding hypothalamic development is fundamental to its function in health and disease (Fu et al., 2017). Yet, this understanding is insufficient, as the embryonic origins of the hypothalamus are unknown (Fu et al., 2019). We seek a greater understanding of this development to further research. This research would improve the treatment of hypothalamus-related diseases.

We use chicken embryos to study brain development. This is because they are easily accessible and there are fewer ethical issues. Furthermore, the embryo is conserved between humans and chickens. To study an embryo, it is important to know where the embryo is at in its development. This is relevant for both understanding and experimental interpretation. Staging guides are currently used for this purpose, which is the process of creating stages for tracking key events in a cell's development. Thus, a biologist can determine the stage of an embryo by referring to the staging guide. This method of determination is reliant on the precision of the staging guide. Additionally, it is prone to human error. Thus, a more efficient method is required.

In this project, we use images of chicken embryos from stage 10 of the Hamburger-Hamilton staging system. This is a conventional staging guide used for chicken embryos which tracks the key stages of development. We want to improve the precision by focusing on the development within these stages. The time-point focused on here is when the embryo has a high level of dynamism in its development. This is between 33-38 hours post-incubation. Within stage 10, the collective structures of embryonic brains were assessed. This assessment suggested creating three sub-stages of development, termed 10.1, 10.2 and 10.3.

These sub-stages are very close in their timescale of development, thus, there are very few differences between images in the different sub-stages. As a result, it is difficult to determine differences by human judgement. To know how far through stage 10 an embryo is, we must improve the precision of the staging system. The solution to this is to use machine learning. This is the study of allowing computers to learn patterns from data. Machine learning classifiers can extract deep features from an image, features that a human is unlikely to detect. Thus, this will give the more precise and efficient staging system that is desired.

1.2 Machine learning and deep learning

1.2.1 Machine learning

Machine learning is the process of solving a given problem by teaching computers to learn using real-world data (Alpaydin, 2020). One associated problem is classifying items into different categories, known as classification. Additionally, values can be predicted through regression. Of particular interest, in this project, are artificial (ANNs) and convolutional neural networks (CNNs). These models are both employed for image classification tasks (Deng, 2014).

Deep learning for image classification

An artificial neural network is a set of interconnected basic processing units that operate on given inputs to process the information and generate desired outputs (Khan et al., 2018). The multi-layer perceptron is a neural network used for making predictions. It has applications across many regression and classification problems applications. A layer is a structure within the network that performs a function on the data. The first layer takes the raw data input and the final layer outputs the prediction (Milewski and Świrski, 2009). The layers between are known as the hidden layers. Each layer has nodes that are given input and deliver output. The feature, the weight and the activation function determine the output from a node. The weight is a learned parameter that decides the importance of a feature if present. If a feature is important to a specific class, the weight will encourage the prediction towards that classification. The activation function maps the weighted input to the output. The output is taken to the next layer or as the prediction if this is the last layer. Deep learning improves on ANNs by compromising more layers. These additional layers allow a higher level of feature abstraction that can improve the predictions made by the model (LeCun et al., 2015).

CNNs perform feature recognition in images (O'Shea and Nash, 2015). They differ from ANNs by being more specific in their use. The main feature of CNNs is their convolutional layers. These layers take an input feature map for each pixel and generate an output feature map (Xie et al., 2020). The feature map indicates the precise location of where the features are in the image. CNNs also make use of pooling layers. Pooling layers operate on blocks of the input feature

map and combine the feature activations (Khan et al., 2018). Max pooling will reduce a structure by replacing each 2 by 2 array of pixel values with the maximum value. This operation keeps the general locations of the features in the image. The advantage is that it reduces the size of the input feature map to the next layer. CNNs usually feature a dense layer as the last layer in the model. This layer maps the feature information to the output prediction (Zhang et al., 2018). These layers manage to optimise the network parameters by learning collectively to improve the network's performance during training.

Gradient descent

Gradient descent (Cauchy et al., 1847) is the process of minimizing an objective function. The objective here is the loss function of the model. The loss function of our model is the predicted error of the network when making predictions. Recall, the derivative gives the slope of a function $f(x)$ at x. For minimizing the loss function, it tells us how to make a small improvement in y by changing x. Here, y is the loss of the model, and x is a feature of the input. Gradient descent is reducing y by moving x in small steps with the opposite sign of the derivative (Goodfellow et al., 2016). The learning rate determines the size of the steps.

Design decisions

To deploy a neural network, there are required decisions on the values of parameters. The loss function gives a metric for the loss of information in the training process. The cross-entropy between the actual and predicted classes is appropriate for our problem (Goodfellow et al., 2016).

Another decision for deploying a neural network is the choice of an optimizer. Optimizers are algorithms that attempt to minimise the loss function. Effectively, optimizers adjust the weights and learning rate to try and reduce the loss of the network (Khan et al., 2018). Recall, the weights decide the importance of features in the data. Adjusting these weights will change how the model processes the features, changing the outcome. Additionally, adjusting the learning rate adjusts the amount the weights are updated. Updating the weights too much will not allow the model to learn all the features in the data. There are mainly two classes of optimizers, adaptive optimizers and gradient descent optimizers. Adaptive optimizers include Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), RMSprop (Graves, 2013) and Adam (Kingma and Ba, 2014). Whilst gradient descent algorithms include batch, stochastic and mini-batch gradient descent. Adaptive optimizers automatically tune the learning rate instead of requiring the user to tune this.

The batch size is the number of examples from the training data set used to make predictions at one time. A larger size decreases the time for the network to converge, whereas smaller batches reduce overfitting and increase computational speed (Radiuk et al., 2017). The larger size causes the weight updates to be

larger, so convergence is quicker. However, smaller sizes give smaller weight updates, which discourages learning a more complex model.

An epoch is one complete pass through the training data. Too few epochs can cause underfitting and inaccuracy of results (Meulenkamp and Grima, 1999). A lack of weight updates causes this, as it does not allow the network to learn all the relevant features. Too many epochs are take longer to run and can cause overfitting (Singh and Singh, 2005), which is defined below.

1.3 Overfitting

1.3.1 The overfitting problem

Embryos are rarely available in high numbers. Additionally, labelling and imaging are time-consuming (Pond et al., 2021). As a result, obtaining a large number of embryo images is difficult. CNNs tend to overfit with a small number of sample images. Overfitting means that the classifier performs well on the given data set but worse on any new data introduced. Various methods exist to overcome these problems, outlined below.

1.3.2 Data augmentation

A common method to reduce overfitting on image data is to artificially enlarge the data set using label-preserving transformations (Simard et al., 2003). This method of transformation is known as data augmentation. Augmentation reduces overfitting by creating new samples by altering the raw image slightly in a few different ways. For example, the image could be rotated, translated, cropped, or flipped. These transformations reduce overfitting as they attempt to normalise the feature variance in the data set. They stop the network from using unimportant features to distinguish between images. Automated augmentation is possible through algorithms such as AutoAugment (Cubuk et al., 2018). AutoAugment found a reduction of error rates of the best-published results of several benchmark data sets. This result suggests automated methods are effective at reducing the feature variance, whilst also being time-efficient. One aspect of the project will be testing different types of data augmentations due to the success demonstrated in previous research. Previous work on augmentation of microscopic images is scarce. For that reason, we can find augmentation policies that are effective on microscopic data.

1.3.3 Regularization

Regularization is modifying a learning algorithm to reduce its generalisation error without reducing the test error. (Goodfellow et al., 2016). Introducing bias or penalties can reduce the variance of predictions which reduces overfitting and increases generalisation (Johansen, 1997). It is crucial to include these techniques to reduce the error in testing due to our small number of samples.

Dropout

Dropout is a technique that randomly drops neurons from the neural network during training. We then train the model on the remaining neurons. We repeat using a different subset of neurons every time before averaging the predictions during testing. This method reduces overfitting as it stops units from accounting for the other mistakes. Less co-adaptation causes a neuron to be less specialised to a specific feature, improving generalisation (Srivastava et al., 2014).

L^2 regularization

L^2 regularization adds a regularization term to the objective function (Goodfellow et al., 2016). This term is a fraction (λ) of the squared magnitude of weights (β), shown below, $\lambda \sum_{j=1}^{p} \beta_j^2$ and we use it as a penalty term added to the loss function. The regularization penalises large weights, reducing the complexity of the model. This penalty reduces overfitting as it stops complex features from being learnt. Here, λ controls how much we penalise complex models.

Early stopping

We also incorporate the concept of early stopping into our models, which stops the training of the model if a metric of the model is no longer changing. For example, if the validation loss is no longer decreasing, training is halted. Gradient descent tends to learn more complex functions that will not generalise with more epochs, as stated in Section 1.2.1. Therefore, we reduce overfitting as well as increase time efficiency.

1.4 Hyperparameter optimisation

Another method that will be key to solving our problem is hyperparameter optimisation. A hyperparameter is a variable of a machine learning model set before training. Different hyperparameter values change the results of the model. Therefore, finding the hyperparameters which maximise results is desirable. This process is hyperparameter optimisation. Implementation can be achieved through a grid or random search. Grid search uses several specified values from a grid and trains a model using each one. A random search uses a random value from a parameter space and trains a model. These two methods are extremely time-consuming. In addition, there is no informed way of choosing the next value. Bayesian optimisation improves this by building a surrogate probability model of an objective function. A surrogate probability model can explore the search space for the hyperparameters to estimate how the performance varies. The values that perform best on the surrogate to perform a training iteration. The training produces an attributed accuracy with the hyperparameters. These results are used to update the surrogate model. This occurs recursively until a specified number of iterations is reached. This method considers previous results when selecting values of the next iteration and so can

approach the optimal point more effectively compared to random sampling (Shin et al., 2020). There are many Python libraries that can be used to achieve this optimisation, such as `hyperopt` (Bergstra et al., 2013), `scikit-optimize` (Head et al., 2020), `Gpyopt` (González and Zhenwen, 2016), `SigOpt` (Hayes et al., 2019) and Sequential Model-Based optimisation for General Algorithm Configuration (SMAC) (Lindauer et al., 2017).

1.5 Cross-validation

A small validation set implies statistical uncertainty of the validation accuracy (Goodfellow et al., 2016). When this occurs, it is difficult to claim that one model is performing better than the other. Cross-validation is a technique to reduce uncertainty. This method repeats the training and validation computation on different randomly chosen subsets of the original data (Fushiki, 2011). K-fold cross-validation is the process of partitioning the data into k non-overlapping subsets. Then, repeating for all k, taking one of the k subsets as the validation set and the other $k - 1$ subsets as the training set. Averaging the validation accuracy across all trials then gives a more certain estimate.

1.6 Aims

The main goal of the dissertation is to create a robust image classifier for classifying chicken embryos. Achieving this goal will mean we have successfully created sub-stages of the Hamburger-Hamilton staging system. To achieve this, we must overcome the issue of overfitting, which requires the completion of secondary goals. These goals include maximally exploring augmentation and regularization techniques. Furthermore, optimising hyperparameters allows us to maximise the results of the classifier. As a result, we hope to provide evidence of using deep learning to create practical tools for use in developmental biology research.

Chapter 2

Methods

2.1 Data

The raw data includes 151 images of chicken embryos at the traditional Hamburger-Hamilton (HH) stage 10. These images have been assessed to be at different levels of development. These different levels are 10.1, 10.2 and 10.3. There are 54 images labelled as 10.1, 55 as 10.2 and 42 as 10.3. 10.0 is the traditional HH stage 10 and 10.1, 10.2, and 10.3 are the sub-stages before HH stage 11. These images are JPEG files of size approximately 10MB and 1000 by 1000 pixels. There is a mixture of colours among the images. Most are dark images with gray colouring, whereas others are brighter, appearing brown/orange in colour with a white background. Some images contain bright green or red dots which is where fluorescent dye has been inserted into the embryo in an unrelated experiment. These embryos are microscopic, so require the image to be taken through a microscope lens, where varying levels of magnification was used for different images. The data was obtained by taking digital images of the chicken embryos in the Placzek lab, situated at the University of Sheffield's School of Biosciences. When loading the data, image dimensions of 200 by 200 pixels are used. This smaller image size compresses the original higher resolution data set. Due to reduced dimensions, this ensures that our training and processing of the model is sufficiently fast and required less computer memory. Concurrently, the dimensions are large enough that the network does not miss out on any relevant features of the image.

2.2 Standard neural network

To gather a baseline, we first constructed a simple neural network of three convolutional layers with 16, 32 and 64 nodes (refer to Section 1.2.1). Also, with max-pooling (refer to Section 1.2.1) and a 64 node dense layer (refer in Section 1.2.1). The model finishes with the output layer with softmax activation. Softmax is a type of activation that is used as the output of the classifier and

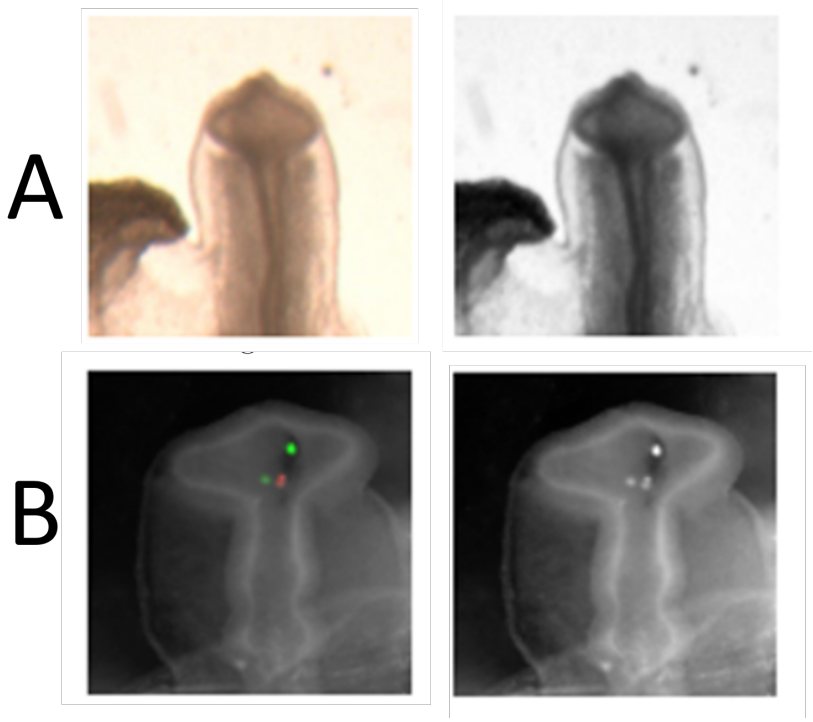

Figure 2.1: The effect of applying grayscale to an image. Showing the bright colouring is reduced in the first image (A) and the fluorescent dye is less apparent in the second image (B).

represents the probability of predicting the image into each class (Goodfellow et al., 2016). We ran this model on 80% of the 151 images with 20% validation.

2.3 Model construction

2.3.1 Pre-processing of data

To assist reproducibility, we fully outline our data pre-processing. One feature evident in the data set was the difference in the colour of some images. Most images are gray, but several are orange and green. As stated in Section 2.1, there are some fluorescent circles in some of the embryos, evidenced by Figure 2.1. Applying grayscale as pre-processing to all the images aids in the classifier not picking these out as features. These colour differences are due to data acquisition and are known to not affect the development of the hypothalamus. Therefore, these features are undesirable for our network to pick out. The result of applying grayscale to an image is in Figure 2.1.

The data were randomly shuffled after loading. Shuffling ensures all the images in the same class are not next to each other. As a result, this causes similar proportions of classes in both the training and validation splits. The data was split into training and validation sets using 80% for the former and 20% for the latter. After this, the sets were split into x and y arrays. The image array is

given to x and the associated class label to y, giving `X_train`, `X_val`, `y_train` and `y_val`. These arrays were divided by 255 to rescale the pixels. Rescaling caused the pixel scale to change from 0-255 to 0-1. So that 0 is still 0 but 1 now corresponds to 255. This was because the network convergence was slow without this rescaling (LeCun et al., 2012). The x arrays were reshaped into an array with the number of images, image dimensions and colour channel value (gray here). This reshaping results in an array of the following dimensions: `(151, 200, 200, 1)`. The y arrays were converted to categorical arrays, meaning classification of 10.1 would now be `[1, 0, 0]`. This is because the network was set up to output the predictions in this format. Therefore, the y arrays need to be in the same format for comparison to the predictions.

2.3.2 Model specification

This model specification was found by manually tuning parameters and judging results. Convolutional layers were added until a decrease in validation loss was observed. As a result, 7 convolutional layers with units up to 1024 with max pooling. The first layer had 16 units and each subsequent layer had double the number of units of the previous layer. This design is motivated by the design of VGG blocks. This design is the process of doubling units in consecutive convolutional layers with max-pooling (Simonyan and Zisserman, 2014). A dropout of 20% after each layer, as suggested by Park and Kwak (2016) is used. Each layer used rectified linear unit (ReLU) activation (Agarap, 2018) and L^2 regularization (Cortes et al., 2012) with $\lambda = 0.001$. The model then has a 512 dense layer before a dropout layer of 50% (Srivastava et al., 2014). The final layer is a 3-unit dense layer with softmax activation.

2.3.3 Compiling the model

Compiling the model is where we define the choices of the optimizer, loss function and metric to monitor. The optimizer used is Adam, for reasons stated in Section 1.2.1. We use categorical cross-entropy as the loss function. This choice was because this function is useful for multi-class classification tasks, such as three classes. Finally, we monitor the accuracy of the model, which will give us the loss and accuracy of the training and validation sets. Validation accuracy is the metric of interest due to the interest in increasing the generalisation of the model.

2.3.4 Training the model

When training models, we used 50 epochs to allow smaller learning rates to converge to the maximum validation accuracy. Manual tuning of the learning rate and checking the epoch of convergence revealed this number. We used early stopping to reduce overfitting and to improve the speed of multiple runs of training. The validation loss was monitored with a change of less than 0.01 (`min_delta`) judged to be unsatisfactory. We used a patience parameter of 10

epochs. Consequently, the training stopped if the validation loss did not reduce by more than 0.01 for 10 consecutive epochs. A minimum delta larger than 0.01 stopped training when the network was still improving. However, a smaller value often meant failure to stop at appropriate points. Similarly, a larger patience value would waste computational time and power, whereas a smaller value caused the training to stop when improvement was evident. The choice of values for both was deemed a compromise between the two problems for each parameter. Validation loss instead of accuracy was monitored as this appeared less sensitive to changes than the accuracy.

32 training samples will be tested in the network at a time. The results of which will provide an estimate for the error gradient. Following this, the weights of the neurons will be updated for the next batch to be tested on the new weights.

2.4 Base augmentation

From the initial results, we applied a base augmentation to the data before training more networks. This is so we could first optimise a model that could act as a baseline to compare with the other augmentations. As well as this, we wanted to build from a model with a respectable validation accuracy to optimise. If we optimized a model on the 151 images, we were likely to overfit and not get hyperparameters that were useful for the augmentation step. Rotations of 5° (from 0° to 355°) were chosen as the base augmentation to the data. Rotation was chosen as it is the most variant feature in the original data set. Moreover, the rotation has no bearing on the development of the hypothalamus. Thus, this feature should not be identified by the network for classification. This base augmented data set was used to manually tune a model which preceded optimisation of hyperparameters with the Bayesian optimisation package skopt (Head et al., 2020). This optimized model was a base for testing further augmentations on top of this base augmented data set. The variance of results was also considered as the augmentation that was effective for the most split of data was preferable. The results of this can be seen in Section 3.1.

2.5 Base model hyperparameter optimisation

We looked to optimise the following hyperparameters of the base model from Section 2.3. Firstly, the number of convolutional layers and the number of nodes in the first layer. We tested the number of layers because each convolutional layer identifies a different feature in the images. Therefore, we wanted to optimise the number of features the model is identifying.

Secondly, the choice of optimizer from Adam, Adagrad, Rmsprop, Adadelta, Adamax and SGD. We gave this option as different optimizers can perform better for different problems. The learning rate was given a range of $(1e^{-6}, 1e^{-1})$ as it controls how fast the model learns the problem. Jacobs (1988) states that a

smaller learning rate causes a small adjustment of the weight. This means that more epochs are needed to reduce the error significantly. However, the model is less likely to learn a feature that does not generalise to the validation set. As a result, overfitting is reduced at the cost of more steps to achieve a significant reduction in error.

We optimized the percentage of dropout. We defined this in Section 1.4 as randomly dropping a percentage of units from the network to stop units compensating for the mistakes of other units. This technique reduces overfitting, so finding the optimal value would improve our model. Research has found dropout with values of 50% at the end of the final dense layer to be effective (Srivastava et al., 2014). In addition, more recent findings have discovered that including dropout of much smaller percentages in between convolutional layers to be valuable in improving the network (Park and Kwak, 2016). Thus, we specify the range for the percentage of dropout within layers from a range of $[0.05, 0.25]$. The percentage of dropout at the end of the model was given a range of $[0.3, 0.8]$.

Recall from Section 1.3.3 that λ is the fraction of penalty we add to the loss in L^2 regularization. If λ is 0, we add no penalty to learning complexity. If λ is too large, the model will be too simple, and not learn enough features to allow accuracy to improve. Therefore, we give a range of 0.000001 to 0.001.

Activation was described in Section 1.2.1 as the function which decides whether the node will fire or not when given an input. We employed ReLU and sigmoid as options here. Firstly, ReLU is the default activation function recommended for use (Goodfellow et al., 2016). This is because it is simple to implement and is less susceptible to the vanishing gradients problem. This problem is where the gradient is very small and prevents the weights from updating, halting learning (Bengio et al., 1994). We included sigmoid in the event of the network experiencing exploding activation (Eidnes and Nøkland, 2017), or the nodes failing to learn, which can occur from using ReLU.

To test the effectiveness of smaller or larger batches, different batch sizes were tested between the choices of 8, 16, 32 and 64. Here, we want to see which choice maximises results. Smaller sizes reduce overfitting whilst larger sizes estimate the gradient more accurately. Therefore, an optimal choice between these two extremes is sought.

We define defaults for the algorithm to start with so we can direct it towards what we suspect could be the best parameters. These are 6 layers with 16 nodes in the first layer. The chosen optimizer was `Adam`, with a learning rate of `5e-05`. The dropout values were 20% after each layer and 50% after the penultimate dense layer. These are the parameters chosen for the base model.

We ran 30 iterations of optimisation to give us the parameters that maximised the validation accuracy. We included early stopping to reduce overfitting and speed up the optimisation process.

five instances of training were performed for each iteration of optimisation. This

means there were five different training/validation splits of the data for each iteration. The average validation accuracy over these five splits was used as the metric for the optimisation to minimise. We ran 15 iterations of the optimisation process; therefore we trained a total of 75 models for this step with different random data splits for each model.

2.6 Data augmentation

Data augmentation is artificially enlarging the data set using label-preserving transformations as specified in Section 1.3.2. To investigate this, several augmentations were tested. We evaluated the performance judgement of the validation accuracy and validation loss metrics.

Our method was to use the previously optimized model in Section 2.5. Each augmentation was applied once to each image. We ran ten repeats of training with different random splits, recording the average validation accuracy. The training and validation sets were augmented separately after splitting the data. This was to ensure no mixing of copies of images, which can cause false results. The augmented images were added to the original image sets in each respective data set. We aimed to understand the best augmentation regime to apply to the data for this problem. In the following section, we outline the specific augmentations that were used and the specific values.

2.6.1 Choice of augmentations and values

Below we state the augmentations that were tested. These were identified by feature differences in the data. Some augmentations were implemented due to past research. The method included augmentation of the baseline data set and addition of these to the baseline set, therefore doubling the number of images. The set of augmentation regimes that were tested are shown in Figure 2.2 and described below.

Due to the variability in the position of the embryo in the images in the data, we decided to apply 5° rotations (2.2B) to each image up to 355°, as stated in Section 2.4. Variability in the magnification of the embryo in some images motivated the use of cropping. We tested this as a lone augmentation and with translation. A crop of 10% (2.2C) was first tested. In addition, a translation of 20% in both y-directions along with a 30% crop was utilized (2.2D). Applying grayscale to images causes some of the embryos to appear very dark in colour. As a result, we attempted to brighten this by using gamma correction with a magnitude of ten (2.2E). This technique can brighten the darker embryos without hindering the already bright embryos. Blurring the images was tested with a magnitude of 5 (2.2F) to try and reduce the variability in the backgrounds of the images. This was also aimed to be solved by complete background removal (2.2G). To perform background removal, the image was first converted to grayscale and blurred (with magnitude 5). A threshold value was then determined by Otsu's

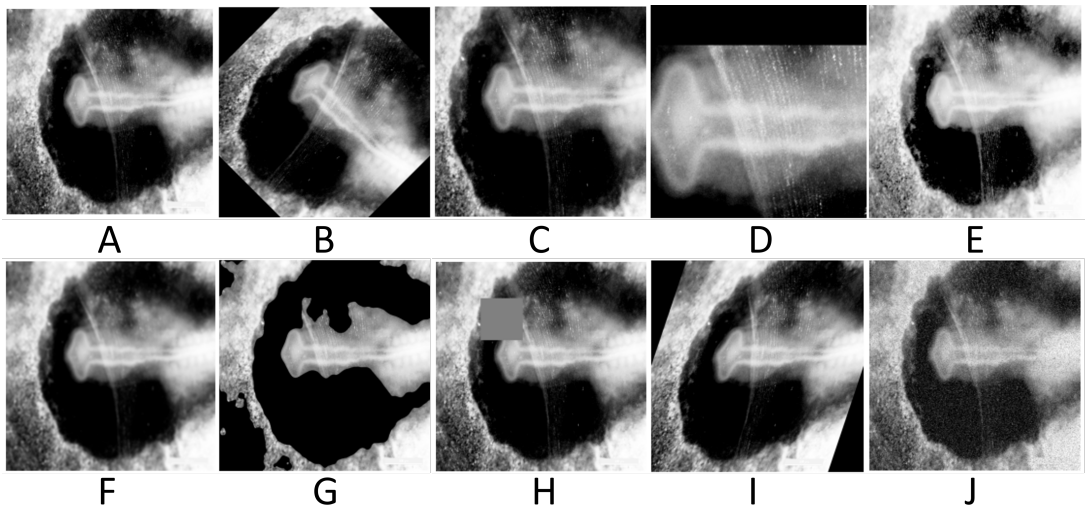

Figure 2.2: An original image from the data set along with the result of the application of 9 different augmentations. These images used histogram equalization as pre-processing which is mentioned in more depth in Section 2.9.1, causing a greater brightness in the image. The augmentations were as follows: A) Original image, B) Rotation by 45°, C) Crop by 10%, D) Crop by 30% followed by -20 translation, E) Gamma correction by 20, F) Blur by $\sigma = 5$, G) Background removal, H) Cutout, I) Shear by 20°, J) Salt and pepper by 20%.

adaptive thresholding (Kurita et al., 1992). If the pixel value is below this threshold, the pixels are given a value of zero (black) and a value of one (white) otherwise. This results in a threshold black/white image that can be used as a mask to apply on top of the image that should cover the background in black. As mentioned in this section, some embryos are darker, which causes the embryo to be removed as opposed to the background. Because of this, we included a statement to check the value of the centre pixel, (where the embryo should be). If this value was less than 100, the bright areas of the image were removed instead of the dark areas. Further augmentations included cutout (2.2H) as it is a state-of-the-art augmentation in image classification (DeVries and Taylor, 2017). As a result, cutout was applied once to each image. Shear (2.2I) was implemented with an angle of 20° in the x-direction. Shear had been found to have great performance on the CIFAR-10 data set (Hu et al., 2019). Lastly, a 20% salt and pepper augmentation (2.2J) was employed once on each image to compare cutout to other occlusion based augmentations. Therefore, showing whether this subset of augmentation is useful for microscopic data.

2.6.2 Augmentation adjustment

To ensure consistency of training/validation sets, we performed 10-fold cross-validation. This problem of uncertainty of results was made evident later in Section 3.3. As we had ten folds, 90% of the data was now used for training and

10% for validation. We ensured that no copies were mixed into another set, as this can give false results. To achieve this, the data was split randomly into ten splits. The ten splits were then augmented separately and added to the split. We then created ten training and validation sets. We then iteratively made one of the subsets the validation set and the other nine subsets the training set. Note that 2 augmentations were eliminated from consideration. Both were removed due to poor performance in the first run of augmentation. Therefore, we reduced computational costs by considering the other augmentations only.

2.7 Augmented model optimisation

The best performing configuration of augmentations based on the results was optimized using Bayesian optimisation. The method was again to take ten splits of the data for each iteration and take the average of the maximum validation accuracy over each split. We used 15 iterations here.

2.8 Error analysis

To analyse potential differences between images, the probabilities of predictions were analysed to reveal feature differences. Here, the optimized model, defined in Section 2.5 with hyperparameters from Table 3.1, was used on three different splits of data obtained through the optimisation process. This included a split with a high validation accuracy (82.18%) and two low validation accuracy splits (54.17% and 40.46%). The method to identify feature differences in the images was to visualise and compare the images predicted with a probability of less than 50% and then those with 90% probability. The numbers in these brackets were analysed for the total in each class, to analyse the performance within classes.

2.9 Testing further pre-processing

2.9.1 Histogram equalization

Histogram equalization is a contrast enhancement method for both natural images and medical and other initially nonvisual images (Pizer et al., 1987). It adjusts the contrast of an image using its histogram. This technique modifies the image so that the resulting histogram is uniform and is efficient for grayscale images (Trahanias and Venetsanopoulos, 1992). This modification causes areas of the image with low contrast to gain a higher value. It has enhanced the quality and diagnostic ability of x-ray images in Attia (2016). The method was effective on dark, noisy and low in contrast images (Attia, 2016). Therefore, this technique is likely to remove contrast differences from the data set. The images are comparable to x-ray imaging which could cause success with equalization.

2.9.2 Pre-processing less magnified images

As crop and translation caused more issues than solutions in Section 4.2.5, we proposed an alternative method. That is to identify the less magnified images in the data and replace these with cropped and translated versions. This is a pre-processing step to see the importance of magnification in the data set. Embryos were categorised based on the direction their embryo was pointing. This was because the translation step is required first and so translating in the right direction will centre the embryo, as opposed to moving it off the frame. After these image sets were identified, the augmentation was carried out. The aim was to find a compromise in the augmentation values so that most images in the set were similar in position and size. This resulted in translation values of 0.2, -0.1 in the x-direction and crop values of 20% for embryos facing left and right. The embryos facing upward had y translation of 0.1 and 20% crop. An example of this result can be found in Figure 2.3. The results of this method can be viewed in Section 3.3.

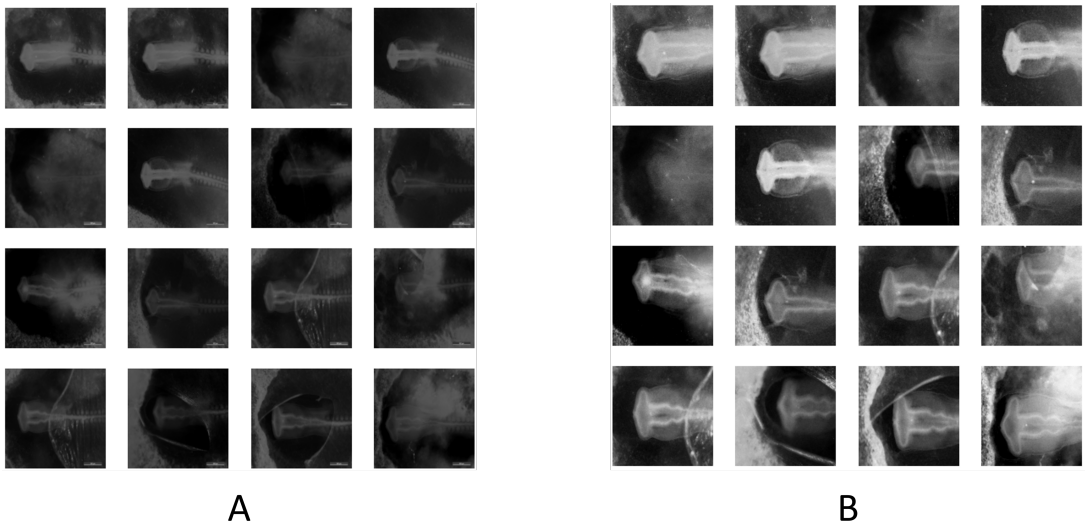

Figure 2.3: The images facing left that were manually identified as less magnified in the 10.1 class (A) and the attempt to increase the magnification of all images simultaneously (B), showing a compromise was made to successfully improve the visualisation of all embryos at the same time.

2.10 Obtaining final metric

To obtain a final metric, the previous best model was trained without early stopping for 100 epochs to allow convergence. The training used the ten splits of data used previously and can give a final overall metric for the project.

2.11 Python packages used

Package	**Reference**	**Utilisation**
imgaug	Jung et al. (2020)	Used for implementing most augmentations including shear, crop, gaussian blur, cutout, salt and pepper, rotation, gamma contrast and translation.
Keras	Chollet et al. (2015)	Specifying the model layers, providing optimizers such as Adam, pre-processing and early stopping.
Matplotlib	Hunter (2007)	Creating plots and the figures with images of embryos.
Numpy	Harris et al. (2020)	Mathematical computations throughout the project.
OpenCV	Bradski (2000)	Loading the data and histogram equalization.
scikit-optimize	Head et al. (2020)	Hyperparameter optimisation.
scikit-image	van der Walt et al. (2014)	Background removal.
Tensorflow	Abadi et al. (2015)	For neural network model building, compiling and training.

Table 2.1: The python packages used in this project, along with their references and the specific tasks they were used for.

Chapter 3

Results

3.1 Rotation augmentation and optimisation improves classification accuracy of neural network-based image classifier

The average validation accuracy over 50 repeats of the training process with different data splits was 36.3%. This was obtained using the original 151 images only. After applying 5° rotations to each image using the model mentioned in Section 2.3, this improved to 51.53%.

Hyperparameter	**Value/category**
Activation	ReLU
Batch size	32
Layer dropout	20%
Final dropout	50%
λ	0.0001
Learning rate	0.0001
Optimizer	Adam
Average validation accuracy	64.1%
Minimum	43.6%
Maximum	81.5%

Table 3.1: The optimal hyperparameters, as determined by Bayesian hyperparameter optimisation.

Furthermore, after optimizing this base model and optimisation, the average validation accuracy improved to 64.1%. The best hyperparameters that maximised the average validation accuracy are shown in Table 3.1. The range of values for these results were from 43.6% up to 81.5%. This range shows disparity amongst results based on the data split.

The histograms of the optimizer(top left), activation (bottom) and batch size (top right) choices are displayed in Figure 3.1. These show the sample counts between the choices in the parameter space. Bayesian optimisation chooses its next value to use in the parameter space by the one that performs best on the objective function in the surrogate model. Therefore, a higher sample count implies that this choice gave better results as this directly influences how often a category is chosen. As Adam, ReLU and 32 were used much more often than the other categories for that parameter, we know that these produce better results for the problem. As a result, in further optimisation, these 3 parameters were removed from the parameter space to maximise computational efficiency.

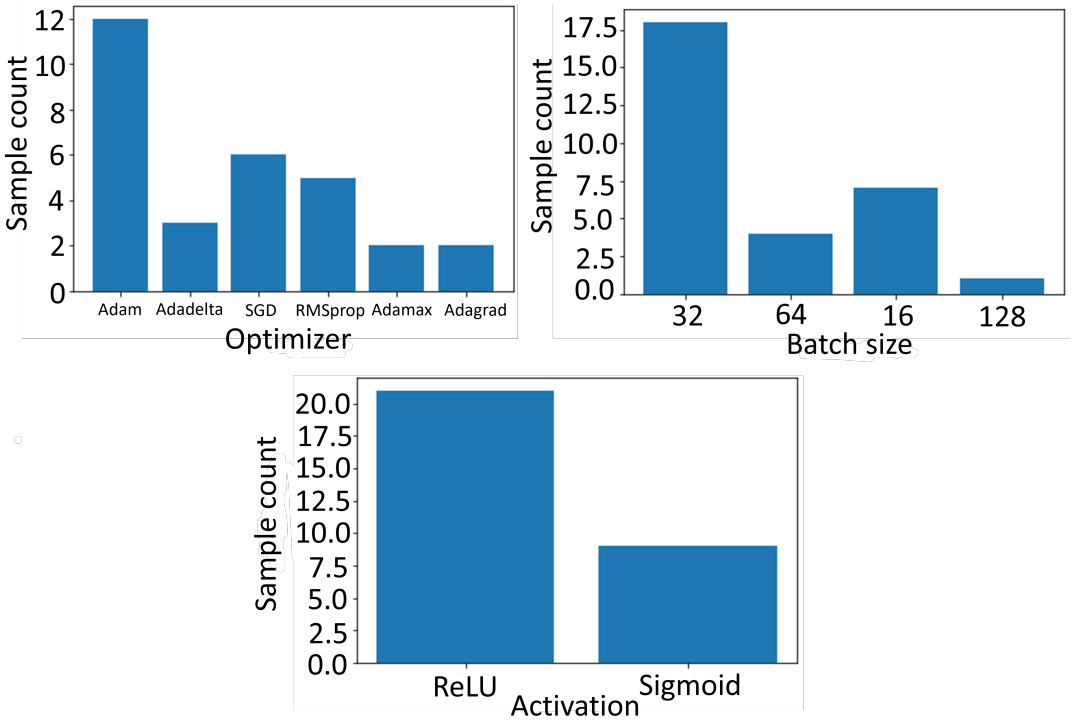

Figure 3.1: Results from optimisation, showing the sample counts of optimizer (top left), activation (bottom) and batch size (top right) during the optimisation process.

3.2 Histogram equalization improves classification accuracy of neural network-based image classifier

As seen in the previous section, the results from running the model repeatedly produced very different results. The only difference between these methods was the randomness of the train and validation split. This randomness suggests that the inclusion of some images in the validation set and not the training set can

cause the results to perform worse. Poor performance may be attributed to important features lacking from the training set cause the model to reduce its accuracy. Using the methods set out in Section 2.8, a difference in the data seeming to affect the classifier was the difference in brightness of images. The predictions with the most uncertainty in all three data splits were those in the 10.3 class. In the good split, 4% of images had a probability of less than 50%. However, there were only 26% with over 90%. This result compared to 80% and 69% proportions in the other two classes were very low. In the bad split, these numbers were worse with 10.1% and 8%.

To solve this problem, the histogram equalization (`equalizeHist`) from Bradski (2000) was used on every image. A more detailed explanation of this method can be found in Section 2.8. ten data splits produced an average validation accuracy of 79.45%. These ten values also had a standard deviation of 5.237 which is much lower than the previous result. The lowest validation accuracy across a split was brought up to 68.7% from 43.6%, with a highest of 86.9% achieved across splits.

3.3 Cutout and optimisation improve classification accuracy of neural network-based image classifier

	Average (%)	**St. Dev**	**Min (%)**	**Max (%)**
Baseline	80.3	6.7	64.7	91.6
Crop	74.8	8.4	59.9	82.3
Shear	78.4	4.8	68.1	84.8
Gaussian blur	80.3	4.9	71.9	89.3
Cutout	83.6	4.6	76.1	91.5
Salt pepper	74.4	7.4	61.9	87.6
Gamma correction	78.7	6.5	67.2	87.7
Crop and translation	53.5	8.9	39.2	74.9
Background removal	73.1	6.2	65.0	87.2

Table 3.2: Showing the average validation accuracy, standard deviation of accuracies, minimum accuracy and maximum accuracy across ten splits of applying different augmentations to the baseline data. Underlined is the best values for each column.

The results of applying data augmentation to ten random splits of data and training model gave the results in Table 3.2. Cutout provided the best validation accuracy with an improvement of up to 83.6%. Based on this, we then used cutout for further optimisation. Cutout increases the average validation accuracy and reduces the variation in the splits. The pre-processed data set that aimed to increase the magnification of embryos as outlined in Section 2.9.2 was

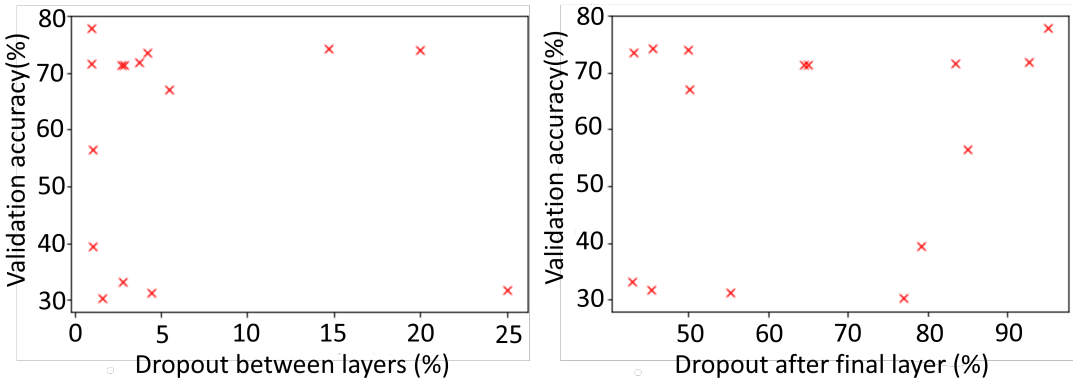

Figure 3.2: Plots of the dropout hyperparameters and the average validation accuracy achieved with these values. This establishes that the validation accuracy is not dependent on these two hyperparameters.

also tested whilst carrying out these augmentations. This resulted in an average validation accuracy of 76.4% which did not improve on the baseline result here.

Hyperparameter	**Value/category**
Layer dropout	1%
Final dropout	95%
λ	0.000004
Learning rate	0.00009
Average validation accuracy	77.7%
Standard deviation	3.2
Minimum	30.3%
Maximum	77.7%

Table 3.3: The optimal hyperparameters from using Bayesian hyperparameter optimisation. Showing an increase in validation accuracy, with different hyperparameters causing results as low as 30.3% and as high as 77.7%.

Optimisation of the cutout and rotated data set resulted in a decrease in validation accuracy to 77.7%, and a standard deviation of 3.2. Therefore, these parameters resulted in a much lower variation amongst results. It is evident how much lower these values are compared to the augmentation results. Also, over 15 iterations, the process never got close to the 83% observed earlier.

The values of the dropout hyperparameters chosen and the average validation accuracy achieved with this value can be observed in Figure 3.2. The most obvious feature of these graphs is that the dropout values chosen have no close relationship to the validation accuracy. This lack of relationship can be seen as the values across the x-axes for both dropout values show a variation of accuracies.

In contrast to dropout, by examining Figure 3.3, we see that the results are

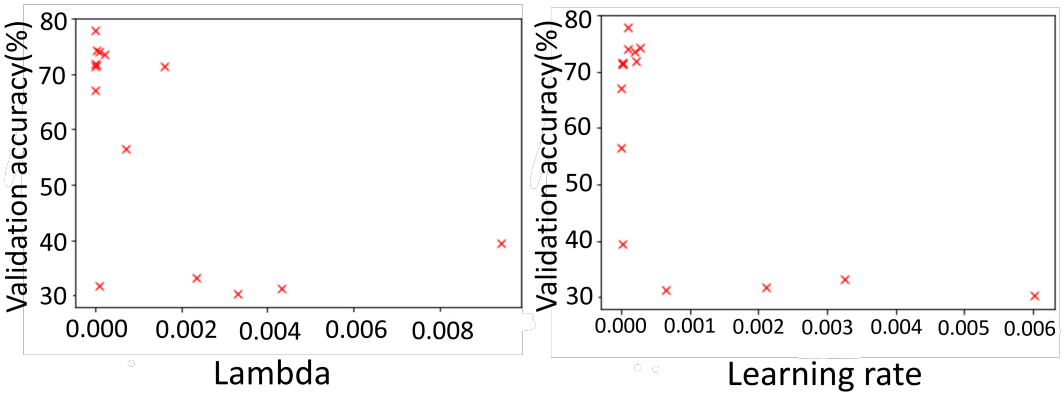

Figure 3.3: Plots of the learning rate and λ values chosen and the average validation accuracy achieved with these values. Implying that the results are very dependable on the value of the learning rate and that there is some dependence between the learning rate and λ.

dependent on the learning rate and λ. Firstly, the learning rate performs best with a value between 0.000018 and 0.0002. There is low accuracy at the start of the axis for values 0.0000206, 0.000006, 0.00001 and 0.000015 of the learning rate. The low result validation accuracy achieved with the learning rate of 0.0000206 and the fact that values similar to this achieved better validation accuracy, suggests that λ is also required to be in an optimal range. The value of λ is optimal between 0.000001 and 0.0002. However, λ is very dependent on the value of the learning rate, shown by the low result achieved with λ as 0.00009 and the learning rate as 0.002. This result was because the learning rate was too high, suggesting that having an optimal value of λ alone is not enough for a high validation accuracy.

Figure 3.4 shows partial dependence plots for learning rate and λ. The partial dependence plot shows the marginal effect one or two features have on the predicted outcome of a machine learning model (Friedman, 2001). Bayesian optimisation builds a surrogate model of the search space and then searches this model instead of the real search space as it is faster. These plots show the results of the last surrogate model. The yellow regions indicate where a hyperparameter had more of an effect on the predicted outcome and the blue areas show less of an effect. The plot suggests the choice of learning rate and λ both have a strong effect on the outcome. This effect is a result of the optimal values being close to a yellow region. It also suggests that any value of λ less than 10^{-4} influences the outcome. Furthermore, a value between 10^{-4} and 10^{-3} for the learning rate is influential. These results suggest limiting the parameter space to these values for future optimisation. Figure 3.5 suggests the unimportance of the dropout between layers, as this whole region is yellow. The graph implies that the range of values for the final layer dropout should be reduced, as the range of values were between 40% and 90%, a further run would suggest exploring values below 40%. The dropout values found to be the best are in the top left corner of

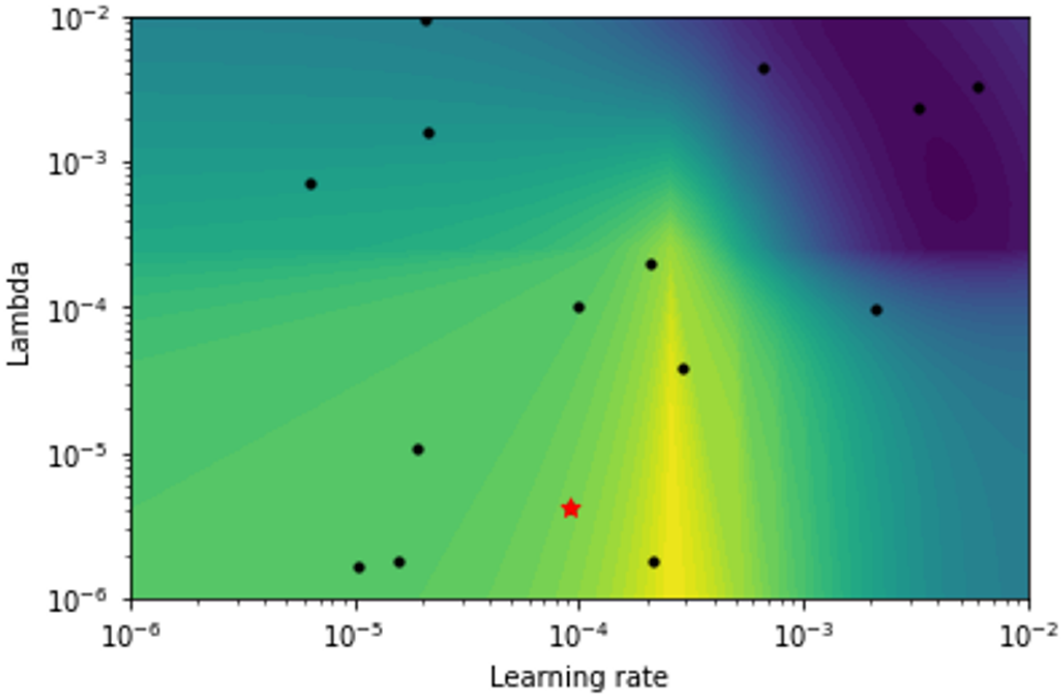

Figure 3.4: Partial dependence plot of the learning rate and λ (lambda). The points on the graph are the choices of the learning rate and λ, the red star indicates the values of both hyperparameters that resulted in the highest average validation accuracy. The yellow regions indicate where the hyperparameters had more of an effect on the predicted outcome and the blue areas show less of an effect. This shows the optimal learning rate value is somewhere between 10^{-4} and 10^{-3}, whereas a value of 10^{-4} or lower for λ (lambda) is favourable.

the graph, which is a blue region. The blue region further adds to the earlier thought that the results in this optimisation process depended on the other two hyperparameters and not these dropout values.

3.4 K-fold cross-validation of augmentations shows cutout classification accuracy of neural network-based image classifier

Due to inconsistency of results, we performed 10-fold cross-validation. Note that two augmentations were eliminated from consideration as stated in 2.6.2. We removed crop and translation due to poor performance. Furthermore, salt and pepper performed much worse than cutout. As an occlusion based augmentation, we decided that cutout is the optimal occlusion augmentation. The results of the training can be seen in Table 3.4. The best result on average across the ten folds was cutout, as also seen in the results of the previous method. Cutout gained the highest validation accuracy on four folds, as well as the lowest standard

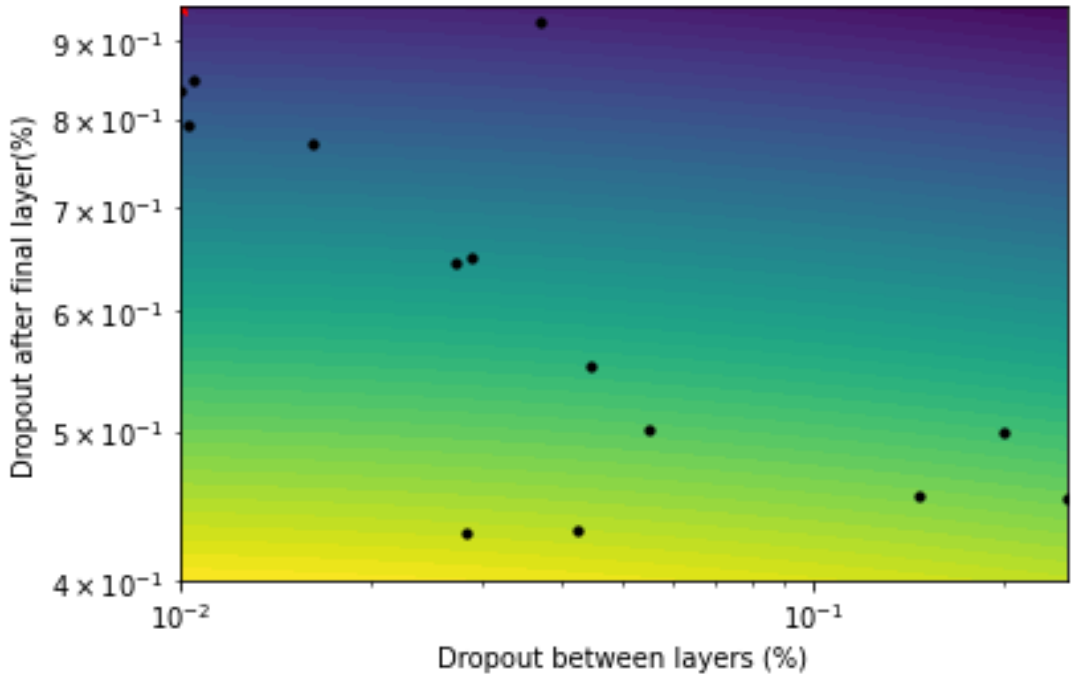

Figure 3.5: Partial dependence plot of the dropout hyperparameters. The points on the graph are the choices of dropout for between layers and the final layer. The red star indicates the values of both hyperparameters that resulted in the highest average validation accuracy, shown in the very top left of the graph. The yellow regions indicate where the hyperparameters had more of an effect on the predicted outcome and the blue areas show less of an effect. This demonstrates that the dropout between layers is not important and that the range of values for the dropout after the final layer should be reduced. The location of the red star indicates that the dropout after the final layer was unimportant in gaining the maximum validation accuracy as it is in a blue region.

deviation of ten.

After proceeding as outlined in Section 2.10, the average validation accuracy improved from 75.6 to 76.7. The results of each fold can be seen in Table 3.5, demonstrating that most of the folds improved with more training and that folds six and ten improved by a large margin. Therefore, the maximum validation accuracy achieved in this project was on average 76.7%.

Aug	**Validation accuracy on fold (%)**										**Avg**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	85.2	75.4	85.7	77.4	82.3	70.2	69.3	47.8	58.5	69.1	72.1
2	78.2	89.1	89.2	73.9	74.7	68.9	75.1	51.4	66.4	71.9	73.9
3	71.3	86.2	85.7	78.7	79.9	68.8	78.5	51.4	60.1	75.0	73.6
4	77.8	92.2	89.4	74.7	78.0	66.3	74.8	45.8	66.3	76.0	74.1
5	79.6	91.9	89.7	74.1	75.9	64.4	76.3	56.8	69.9	77.1	75.6
6	72.6	86.2	87.9	74.6	80.0	60.4	68.5	48.5	68.4	68.0	71.5
7	75	87	85.5	86	65.5	68.1	79.1	51.5	58.2	66.9	72.3
Avg.	77.5	86.8	87.9	75.6	78.5	66.5	73.8	50.3	64.9	72.9	

Table 3.4: The augmentation results across ten folds, also showing the average validation accuracy and standard deviation for each augmentation as well as the average for each fold. The augmentations (aug) are as follows; 1) **Baseline**, 2) **Shear**, 3) **Crop**, 4) **Gaussian blur**, 5) **Cutout**, 6) **Gamma correction**, 7) **Background removal**. Augmentations 2-7 are in addition to the baseline data. Here, avg. stands for average. The best value for each fold is underlined.

Validation accuracy on fold (%)										**Avg**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
80.0	89.8	89.7	75.5	78	70.28	75.2	56.2	69.1	83.4	76.7

Table 3.5: The results of running 100 epochs without early stopping on the cutout and rotated data set and optimized model. Showing improvements on most folds and large improvements in folds six and ten.

Chapter 4

Discussion

4.1 Aims

Recall that the main aim of the dissertation was to create a robust image classifier for classifying chicken embryos. This was approached by attempting to achieve secondary objectives, which were to implement strategies to reduce overfitting. Firstly, through regularization and secondly, by implementing the most effective data augmentations. Regularization was approached by utilising l^2 regularization in addition to dropout and early stopping. Data augmentation was investigated by identifying feature differences in the images and then testing their effectiveness. Attempts were made to reduce the effects of the original data set, by attempting to maximally augment the data set. The number of images increased from 151 to around 9000 images. Beyond this, an aim was to use Bayesian hyperparameter optimisation to maximise results in the shortest computational time. This was completed by optimizing the initial rotated data set and later the cutout data set. Overall, the results showed that a successful image classifier was created. However, for completion of the sub-staging, a greater validation accuracy is required.

4.2 Discussion

4.2.1 Base model

Section 3.1 showed that rotating all the images by 5° improved the validation accuracy from 36.3% to 51.53%. This result confirmed our assumption that the image rotation would be a classifying feature without any augmentation.

4.2.2 Base model optimisation

In Section 3.1 Adam to be the most efficient optimizer. We know that Adam is robust to hyperparameter choice (Kingma and Ba, 2014). Thus, Adam was less

sensitive to the hyperparameter changes, giving more consistent results. This consistency may have caused the higher sample count. However, Kingma and Ba (2014) found that Adam lowers the loss of training and improves convergence. This result was based on the CIFAR-10 data set. These two points suggest Adam would reduce the training cost in the fewest epochs. As training used 50 epochs, other optimizers may have taken longer to converge. Even if there was convergence, we expect Adam to reduce training loss the most. Thus, Adam was important in maximising the validation accuracy. The batch size of 32 was a satisfactory compromise between overfitting and convergence as suggested by the superior sample count. Also, ReLU solved the vanishing gradients problem of sigmoid activation. Furthermore, it did not encounter the problems we alluded to in Section 2.5. As a result, we justified the use of these hyperparameters after this point.

4.2.3 Data splits/folds

The most intriguing part of the data is that splitting the data into training and validation sets differently before every model training could cause very different results. This was judged in Section 3.3 which showed the minimum and maximum values for each result were far different from their average. Furthermore, the 10-fold cross-validation results in Section 3.4 showed even more variance in the results when compared to using random splitting of the data. These results included the 8^{th} fold having an average validation accuracy of 50.3% over all the results, whereas fold 3 had an average of 87.9%. It is interesting that all the validation accuracies decreased compared to the first augmentation and that the variance of results increased. Firstly, the difference in results is most likely because of some features in only a select amount of images. When these images are in the validation set and not the training set, the validation accuracy increases as the model has not learnt these features during training. Contrarily, the validation accuracy decreases when the model does learn these features during training.

Recall that the validation set used 20% of the data in Section 3.3 (first augmentation). However, the folds of cross-validation in Section 3.4 whereas only 10% of the data (second augmentation). This percentage means that there was much fewer data in the validation set of the 10-fold cross-validation tests and more images in the training set. Subsequently, there should have been more features present in the training set. More features should therefore have increased generalisation. Thus, it does not make sense that the average validation accuracies in the 10-fold cross-validation were less than those in the first augmentation. A reason for this is that there could be a small number of images that cause the poor generalisation error. Therefore, when the validation set increases in number, such as in the first augmentation, the number of correctly classified images increase.

Figure 4.1 shows the images in the third (4.1A) and eighth (4.1B) folds. When

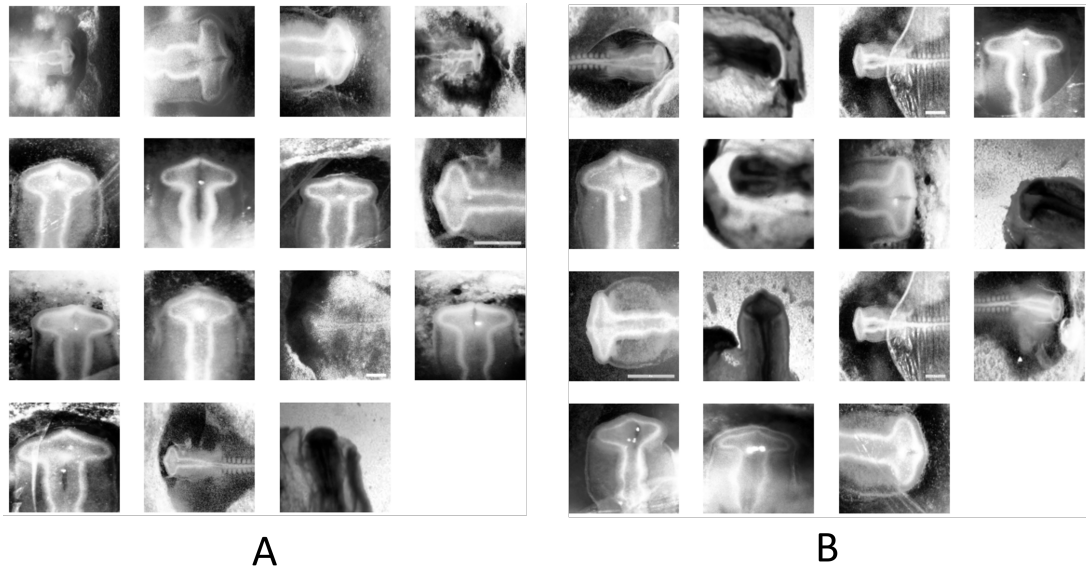

Figure 4.1: The 15 images in fold 3 (A) and the 15 images in fold 8 (B). Demonstrating that it is difficult to find a reason for the discrepancy in validation accuracy by manual inspection of the two sets.

the third fold was not in the training set, the validation accuracy was low. However, when this was instead the eighth fold, the validation accuracy was 37.9% better. In terms of feature differences, there are four dark embryos in fold eight compared to one in fold three. Also, fold eight has a duplicate image. This duplicate is because one of the images had fluorescent dye inserted in it compared to the other which did not. Although, when grayscale is applied, they look similar. This suggests deleting one of these images, as well as manual inspection of the full data set for more instances of this replication. Other than this, the two sets seem equal in terms of distorted background and the number of less magnified images.

4.2.4 Histogram equalization

As folds of training and validation data were unequal in their brightness, histogram equalization was applied. As demonstrated in Section 3.2, increasing the contrast of images with this method improved the average validation accuracy from 64.1% to 79.5%. We surmise that this improved the classifier's ability to pick out features of the data. Recall that this function causes lighting alterations. This alteration appeared to normalise all the images towards the same lighting of backgrounds and embryos. This transformation may have helped remove these differences as a feature or the features may have become more visible. This technique is used in x-ray imaging for improving the clarity of features (Attia, 2016). Thus, the classifier was improved due to the increased ability to identify patterns, increasing generalisation.

4.2.5 Augmentations

In both tests of augmentation, cutout increased the validation accuracy the most. Cutout applies a fixed size zero mask in a random area of the image. DeVries and Taylor (2017) stated that the motivation behind cutout was to reduce focus on prominent features to promote generalisation. This suggests that the network focused less on features that were not generalisable. However, this is difficult to confirm as the method did not improve every single fold. Some folds were not improved because the classifiers had already identified features that generalised well to these validation sets. Cutout reduced the accuracy in these cases as it covered up parts of the images that the network was reliant on. The improvement in most folds was possibly due to the classifier focusing less on certain features due to it being covered up, causing a smaller generalisation error.

Overall, folds one, five and six were not improved by any augmentation. This lack of improvement means that the augmentations caused the features chosen to be generalise worse to the validation set. This result could be because the classifier could not learn the patterns in the data, due to the augmentation causing the identification of generalisable features to be more difficult. Alternatively, there may not have been enough copies for the classifier to learn. The second option is more likely, as there was only one augmentation applied to each image and one value of this augmentation. This differs to the initial rotation augmentation, which gave many rotated copies of the image. This number of copies needs to be repeated for the other augmentations to have a similar effect. However, this is limited by the number of images we can use for training at once without causing a memory error.

The removal of backgrounds of the images using thresholding had varying effects on the results based on the fold. Mostly, removal reduced the maximum validation accuracy or performed similarly to the other augmentations. Removal had the biggest improvement on the fourth fold, improving the result to 86% from 77.4%. Looking at Figure 4.2, background removal (4.2B) appears to keep most of the embryo intact, except for image ten. Apart from this positive result, it also gave the best result on fold seven, but by a much smaller margin. This result of fold four suggests the background removal can enhance results significantly if removal is executed efficiently. Execution was inefficient in other folds, which revealed varying issues with removal. One key issue was that some embryos were removed from the image instead of the background, despite the attempts to circumvent this. This error suggests that a more successful technique could be beneficial to improving the classifier. The issue in the current method lies with determining the value for changing the operator. Recall that in Section 2.6.1 the centre pixel value was used to determine the brightness of the embryo. This value decided the choice of the operator to use,

$$\begin{cases} >, & \text{if centre pixel} > 100 \\ <, & \text{otherwise} \end{cases} \tag{4.1}$$

which would determine whether the dark or bright pixels of the image were removed. Unfortunately, in some cases, this still removed the embryo instead of the background. The failure could be because the embryo was not centred or that the choice of 100 does not split the dark and bright images into separate groups perfectly. This error in value choice is evident as increasing the value solved the incorrect removal for some images but caused a different issue with removing the incorrect sections in another image. Consequently, there may not be a perfect value for the centre pixel. The fact that there are rarely perfectly segmented embryos produced means the method could be improved. Improvement could be obtained through object detection or by manually choosing the threshold values for each image.

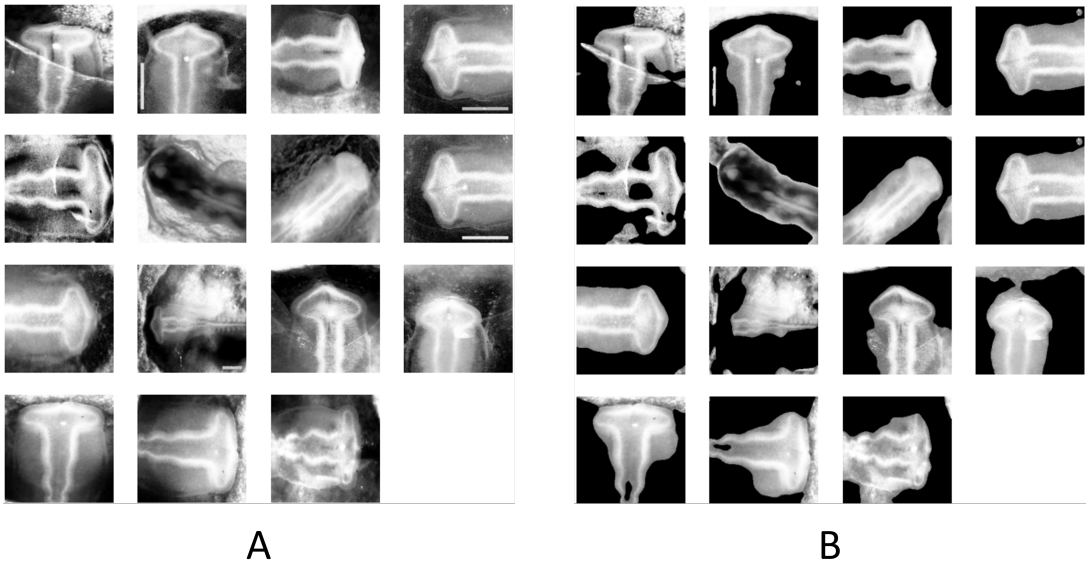

Figure 4.2: The original 15 images in fold 4 (A) and the images in fold 4 with background removal applied (B). Showing that the background was successfully removed in all images to a good standard.

Figure 4.3 shows a potential issue with the crop and translation method. The combination that is effective for the first image (4.3A) also is very ineffective for the second image (4.3B). This ineffectiveness will also be true for other images, as images that were originally at a greater magnification will be cropped further and will fail to show the full embryo shape. As a result, this method will likely cause more harm to the classifier as it creates augmented images without their labels preserved. Failure of this method motivated applying this method to the original data, by manually identifying the images that needed more magnification. The results of which were revealed in Section 3.3, providing a 76.4% accuracy. Although this did not improve the baseline, it also did not reduce the results by much. This result suggests further investigation into this method to try and eliminate differences in magnification from the data. This difference in magnification may not be important, the magnification could be causing the model to be more robust.

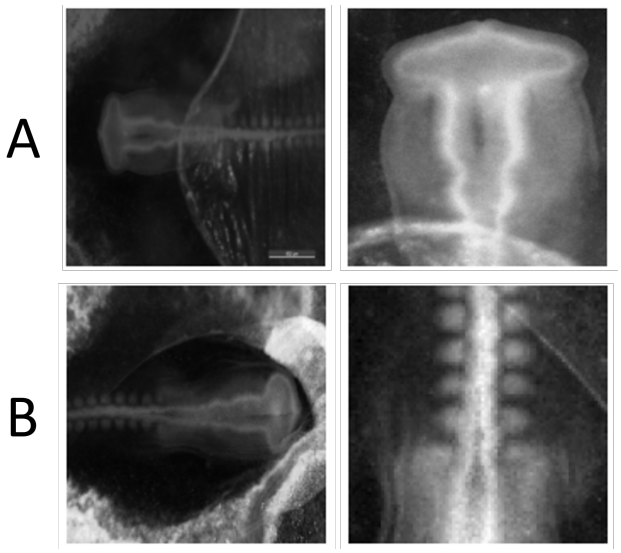

Figure 4.3: Two example images from the data set before and after applying crop and translation. One image shows combining rotation, translation and cropping can result in a desirable augmentation (A). However, from we can see that this can also result in an image that is not label preserving (B). So, particular care needs to be applied to ensure that every image has all label defining features as part of the image.

Overall, there is a compromise with augmentation reducing the feature differences in the data but not causing internalisation of false features. Although augmentation can significantly improve the generalisation, we have also proved that some methods can cause the opposite effect. Furthermore, removing every single feature has shown that some features make the model more robust to changes. Recall that the values for the selected augmentations were chosen by visualising the augmentations and judging which would be most beneficial. Therefore, this introduced human error into the methods and different values may well produce better results. However, the use of human judgement will be improved upon by automated algorithms.

4.2.6 Optimisation of cutout data set

The results of optimizing the cutout data sets appeared to show that the key contributor to results is the choice of the learning rate and λ in l^2 regularization, where having one of these values too large whilst having the other too small would cause poor results. Even further, it seems that the dropout values for both between layers and after the final layer were insignificant. Judging by the partial dependence plot in Figure 3.5, any value of the dropout between layers would give similar results. The yellow region suggests that lower values of the final layer dropout were better, which implies decreasing the range of values for the parameter space. We chose a value of 50% initially due to previous research from (Srivastava et al., 2014), but the parameter space of 50% up to

95% was decided by manual tuning. This decision was because larger dropout values seemed to increase the validation accuracy in early model exploration. The parameter space should have been centred at 50%. This error could have caused the hyperparameter to be insignificant, but further testing is required for verification.

4.2.7 Overall accuracy of classifier

Despite the overall average validation accuracy of 76.7%, the project proved that the validation accuracy could reach 92.2%. This result was achieved with the second fold in Section 3.4 with the blur augmentation. Also, this fold had a high percentage throughout. This suggests that it is possible to achieve high accuracy with this data set. An improved average could be accomplished by further work to eliminate more of the feature differences. This would improve the reliability of the high accuracy results.

4.3 Further work

4.3.1 Computational resources

The project utilised convolutional neural networks and large amounts of image processing. This processing required a graphics processing unit (GPU) to complete this work in a realistic timeframe. We gained access to a GPU over the cloud through Google Colab. Colab caused various issues in terms of access to resources that were not anticipated at the outset. The limiting factor for many issues of the project was the access to RAM, which would often cause computer crashes. Crashes occurred if the model used had a large number of layers or images. The lack of layers limited the amount that could be achieved with this project, as a deeper model could have identified up more image features. Additionally, more augmentations could have improved the classifier. Here, the project was limited to using one augmentation on top of the base rotated augmentation. The full extent of augmentations was not tested in this project. The feature differences that were obvious from manual inspection in the images were instead tested. But it was shown that augmentations such as shear and cutout are effective. Testing more augmentations and combinations could improve results. Beyond that, more computational power will allow the use of deeper models. These resources would allow the detection of improved features that is likely to improve the validation accuracy further. We aimed to solve the limited computational resources through the university's high-performance computer as an alternative to Colab. However, we encountered issues with the interactivity of the python platforms available as well as some resource issues. With more time, we could solve these problems.

Even with more computational power, manually testing many augmentations and combinations is very time-consuming. At the outset of the project, we hoped to use recent research with AutoAugment to find the best augmentation policies

for the data. However, this package is not open source. There are a limited number of these packages based on these findings but none were compatible with Google Colab or the data. That meant that time was consumed in testing many different augmentations along with testing values to use on the data set. In addition to this, augmentations were falsely thought to perform well by split variation. Proceeding with more splits however resulted in an average time of around 40 minutes to test one augmentation based on a data set with 10,000 images. This meant that optimisation was also limited, as to run 15 iterations of these augmented data sets would take over four hours to run. If this process was sped up, more iterations could be used which would give a greater search of the parameter space. It also meant that parameters such as the optimizer function were taken out of consideration as it was producing results that were wasting computational time.

In Section 2.6.1, we chose values for augmentation by manual visualisation and subsequent presumptions about what would be best for the data set. This method could be improved by performing optimisation on the augmentation values. This optimisation would include specifying a range of values for the augmentation in the parameter space, then the optimisation process testing this value recursively and giving the validation accuracy. Due to the variability, this would require cross-validation with five or ten folds. Then using multiple iterations for each of the ten folds would mean a high computational cost. Completing this for every augmentation would be valuable, but very costly in terms of computational resources.

4.3.2 Medical image enhancement

One of the key findings in this work was applying histogram equalization to the images which improved classification. As mentioned, this technique is used to improve the diagnostic ability of x-ray images in Section 2.9.1. Based on this, we should further investigate the application of medical image enhancement techniques on this data. This is because the images are similar to an x-ray in terms of colour and the skeletal frame. Consequently, any transformations that improve feature identification in x-rays are likely to improve the classifier.

4.3.3 Improving background elimination

The backgrounds of the images are very different across the data set. However, these backgrounds are not useful to the decision that the classifier makes. The classifier may improve if the embryo was segmented or the background blurred. This segmentation is likely to require object detection to remove the correct parts. However, this is outside the scope of this project. Additionally, manual segmentation could be performed but would be time-consuming.

4.3.4 Saliency analysis

Given the variance of folds in the results, a method to investigate the reasons for these differences could be through saliency analysis. Saliency shows what areas of an image the network is paying attention to classify. Saliency would determine whether the CNN is using relevant or irrelevant features. This would be useful for further analysis on fold eight and would give much more insight into the reasons this fold performed poorly, as well as showing the feature each layer in the network was using. This information would provide insight into the different features picked up for the different folds. If these features that saliency identifies cannot be removed by augmentation, we could delete these images.

Chapter 5

Conclusion

In conclusion, the project succeeded in answering the main research question regarding if it was possible to create an image classifier that can classify chicken embryos well, despite the limitations caused by the small data set. The drawbacks were minimised by augmentation, regularization and identifying pre-processing steps. However, more work is required to ensure the success of this classifier which further research into the hypothalamus.

To achieve this improvement, we require greater computational resources to perform more augmentations and use deeper models. Moreover, the improvement of automated augmentation may allow increased efficiency that is hard to attain manually. More medical image enhancement methods may improve the feature detection ability of the classifier. Furthermore, pre-processing images to reduce data acquisition differences could improve results, such as improving the work on background elimination. Finally, saliency analysis on the layers in the model should highlight the features that are being used to make classifications, which could lead to identifying why some data folds perform very well. As a result, we could improve all the folds to the same standard. Overall, the results proved that the classifier can perform to a high standard.

In terms of new findings, the project proved that augmentation is effective for improving classification accuracy for microscopic data. The method of identifying feature differences in the data to reduce through augmentation allows the accuracy to be improved to a certain point. However, further improvement in this area would benefit from saliency analysis. Furthermore, testing augmentations based on research can improve classification, shown by the results of cutout. Additionally, hyperparameter optimisation was shown to be effective for increasing the validation accuracy. As well as this, histogram equalization is an effective method for improving the classification accuracy on microscopic data, as it makes features of the images more apparent. Overall, this project provides evidence that deep learning can be used practically for furthering research in developmental biology.

Appendix A

Research Ethics Approval

The research ethics approval process for the work described in this dissertation was completed in March 2021, before any work using data commenced. The data collected does not involve any human participation or analysis of secondary data, therefore, formal research ethics approval was deemed unnecessary. The declaration is evidenced here by the inclusion of the declaration form in Figure A.1.

Form 1: Student declaration (for research that does not involve human participation or analysis of secondary data)

School of Mathematics and Statistics

Research Ethics Review for Postgraduate Taught Students

Dissertation title: Deep learning for biological image classification

In signing this student declaration, I am confirming that:

My project does **not** involve people participating in research either directly (e.g. interviews, questionnaires) and/or indirectly (e.g. people permitting access to data).

My project does not therefore require an ethics review and I have not submitted a Research Ethics Application Form.

Name of student: Jacob Holmshaw

Signature of student: Jacob Holmshaw Date: 15/03/21

Name of supervisor: Alexander G Fletcher

Signature of Supervisor: Alexander G Fletcher Date: 15/03/21

Figure A.1: The declaration form that ethics approval was not needed for the work outlined in this dissertation, following compliance with the University of Sheffield official research ethics processes.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Attia, S. J. (2016). Enhancement of chest x-ray images for diagnosis purposes. *Advances in Physics Theories and Applications*, 54:20–3.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Bergstra, J., Yamins, D., and Cox, D. D. (June 2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Available at: `http://proceedings.mlr.press/v28/bergstra13.pdf`.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cauchy, A. et al. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2012). L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*, pages 109–116.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, pages 1–14.

Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:1–29.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, pages 1–8.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7):2121–2159.

Eidnes, L. and Nøkland, A. (2017). Shifting mean activation towards zero with bipolar activation functions. *arXiv preprint arXiv:1709.04054*, page 7.

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

Fu, T., Pearson, C., Towers, M., and Placzek, M. (2019). Development of the basal hypothalamus through anisotropic growth. *Journal of neuroendocrinology*, 31(5):12727.

Fu, T., Towers, M., and Placzek, M. A. (2017). Fgf10+ progenitors give rise to the chick hypothalamus by rostral and caudal growth and differentiation. *Development*, 144(18):3278.

Fushiki, T. (2011). Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, 21(2):137–146.

González, J. and Zhenwen, D. (2016). GPyOpt: A Bayesian Optimization framework in python. `http://github.com/SheffieldML/GPyOpt`.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, pages 1–43.

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., del Río, J. F., 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(7825):357–362.

Hayes, P., Anderson, D., Cheng, B., Spriggs, T. J., Johnson, A., and McCourt, M. (2019). SigOpt documentation. Technical Report SO-12/14 – Revision 1.07, SigOpt, Inc.

Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., and Shcherbatyi, I. (2020). scikit-optimize/scikit-optimize. https://doi.org/10.5281/zenodo.4014775.

Hu, B., Lei, C., Wang, D., Zhang, S., and Chen, Z. (2019). A preliminary study on data augmentation of deep learning for image classification. *arXiv preprint arXiv:1906.11887*, page 2.

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

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307.

Johansen, T. A. (1997). On tikhonov regularization, bias and variance in non-linear system identification. *Automatica*, 33(3):441–446.

Jung, A. B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F.-M., Weng, C.-H., Ayala-Acevedo, A., Meudec, R., Laporte, M., et al. (2020). imgaug. https://github.com/aleju/imgaug. Online; accessed 01-Feb-2020.

Khan, S., Rahmani, H., Shah, S. A. A., and Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, pages 1–15.

Kurita, T., Otsu, N., and Abdelmalek, N. (1992). Maximum likelihood thresholding based on population mixture models. *Pattern recognition*, 25(10):1231–1240.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Lindauer, M., Eggensperger, K., Feurer, M., Falkner, S., Biedenkapp, A., and Hutter, F. (2017). Smac v3: Algorithm Configuration in Python. https://github.com/automl/SMAC3.

Meulenkamp, F. and Grima, M. A. (1999). Application of neural networks for the prediction of the unconfined compressive strength (ucs) from equotip hardness. *International Journal of rock mechanics and mining sciences*, 36(1):29–39.

Milewski, J. and Świrski, K. (2009). Modelling the sofc behaviours by artificial neural network. *International journal of hydrogen energy*, 34(13):5546–5553.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, page 2.

Park, S. and Kwak, N. (2016). Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision*, pages 189–204. Springer.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., and Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368.

Pond, A. J. R., Hwang, S., Verd, B., and Steventon, B. (2021). A deep learning approach for staging embryonic tissue isolates with small data. *Plos one*, 16(1):e0244151.

Radiuk, P. M. et al. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, 20(1):20–24.

Shin, S., Lee, Y., Kim, M., Park, J., Lee, S., and Min, K. (2020). Deep neural network model with Bayesian hyperparameter optimization for prediction of NOx at transient conditions in a diesel engine. *Engineering applications of artificial intelligence*, 94:103761.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings*, volume 2003-, pages 958–963. IEEE.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, pages 1–14.

Singh, T. and Singh, V. (2005). An intelligent approach to prediction and control ground vibration in mines. *Geotechnical & Geological Engineering*, 23(3):249–262.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Trahanias, P. and Venetsanopoulos, A. (1992). Color image enhancement through 3-d histogram equalization. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis*, volume 3, pages 545–548. IEEE Comput. Soc. Press.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453.

Xie, Z., Zhang, Z., Zhu, X., Huang, G., and Lin, S. (2020). Spatially adaptive inference with stochastic feature sampling and interpolation. In *European Conference on Computer Vision*, pages 531–548. Springer.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, pages 1–6.

Zhang, Q., Zhuo, L., Li, J., Zhang, J., Zhang, H., and Li, X. (2018). Vehicle color recognition using multiple-layer feature representations of lightweight convolutional neural network. *Signal Processing*, 147:146–153.