Machine Learning Project

Instituto Superior Técnico
Group 102

Leonardo Alves 93392, Paulo Machado 93977

1 Part 1 - Regression - First problem

In this assignment we were given a training set, and a test set. We were tasked with training a predictor $f(x)$, based on the training data, in order to submit a vector of outcomes $\hat{y} = f(x_{test})$, with x_{test} being the values of the features provided to test our predictor.

1.1 Choosing the model

In order to choose the model, we first resorted to experimentation. Initially we trained a linear model with all the data provided, and the resulting mean squared error was quite small, leading us to believe that using a polinomial model would lead to a higher risk of overfitting, while not improving our results in a significant manner. We tested different methods of Linear Regression, first without regularization, and then with regularization imposed by the two methods covered in the theoretical lectures: Ridge Regression and LASSO. As such, our estimates will be given by $\hat{y} = \beta x + \beta_0$

1.1.1 Regularization

The aim of regularization is to constrain the coefficient estimates towards zero. This discourages the training of an unnecessarily complex model, which, in turn, reduces the risk of overfitting. Ridge regression works by modifying the cost function, adding a penalty $(\lambda\|\beta\|^2)$. This penalty serves the purpose of discouraging large coefficients. The LASSO method works in a similar fashion, but the added penalty is $(\lambda\|\beta\|_1)$. This results in key differences between the two models, which will become evident when training and cross validating the different models. In order to choose the best possible value of λ for each of the two models we resorted to the GridSearchCV method of the *scikit-learn* Python library. This method allows us to cycle through different values of λ between an upper and a lower limit, incrementing it by a given value. For each value it executes cross validation (which is covered in a later section). The ideal λ will be the one that corresponds to the model that scores the lowest mean squared error.

1.1.2 Centering the data

When using LASSO or Ridge Regression it is advised to center the data, and training the data without the use of an intercept term. Removing one coefficient from the training model may lead to better results when cross validating, therefore, in order to test this hypothesis and decide wether or not the data should be centered, we tested each model two times. One of the times we trained the model using an intercept parameter, without centering the data, while in the other version of the model we manually centered each feature as well as the outcomes and trained the model without using an intercept term.

1.1.3 Cross Validation

In order to reduce the risk of overfitting through the choice of an inappropriate model, Repeated K-fold Cross Validation was performed on the training set. This type of cross validation consists in repeating the standard K-fold Cross Validation method n times, with different splits at every repetition. This method was chosen as it is expected to be more accurate than only doing a single K-fold Cross Validation, while also being less computationally expensive or prone to variability than the Leave One Out or Leave P Out methods. The Repeated K-fold CV was performed by splitting the data set into 10 folds (k=10) and performing 10 repetitions. A larger value of k would have led to a potentially insufficient amount of points being left aside for validation purposes, while a smaller value would have caused a reduction in the amount of data used for the training itself. The choice of performing no more than 10 repetitions was made due to the fact that any further repetitions had minimal effects on the results while increasing the computational burden. The implementation of this method was achieved through the use of the $RepeatedKFold$ function of the $scikit - learn$ package. This method computes the average value of the mean squared error of each test, which we used as the criterion to choose the best model.

1.1.4 Cross Validation Results

In this section we will summarize the results we obtained for each model. The code used will be available as an appendice. In order to estimate the value of λ for each of the versions of LASSO or Ridge Regression we started with a broad spectrum of values and narrowed the acceptable values down to a point where the iteration step was reasonably small. Then we ran the same cross validation method for every model and compared the results. The resulting mean squared error is the average of the mean squared error for each run of the cross validation.

Model	Linear	Centered Linear	Ridge	Centered Ridge	Lasso	Centered Lasso
MSE	0.016107	0.015631	0.016103	0.015627	0.015930	0.015459
λ			0.031	0.030	0.002	0.002

Table 1: Results obtained for each regression model

We can instantly determine that centering the data results in a better cross validation score, which indicates that it is a good procedure to center the data, removing the existence of an intercept parameter (β_0). Also, we can see that adding regularization parameters results in a better cross validation score, leading us to believe it is correct to apply regularization in this problem. Furthermore, we were able to conclude that LASSO has a slightly better score than Ridge Regression. This result may occur due to the fact that LASSO has the ability to zero out coefficients, effectively removing them from the model completely, while ridge can only scale them down. This means that LASSO, in this particular case, may be more efficient when it comes to minimizing overfitting risk, since it can set certain coefficients to zero. Therefore, we chose the LASSO method for estimating the test outcomes, and we decided to center the test data as well. After determining that this method was the superior one, we reduced the search interval for λ, and reduced the step, in order to reach a more exact value for the best model possible, having obtained $\lambda = 0.00191$. After obtaining this value, we trained the model using all 100 samples, resulting in the final β estimates to be used in the estimation of $\hat{y}$.

1.2 Estimating Test Outcomes

Finally, we estimated the test outcomes by using the LASSO method, with $\lambda = 0.00191$, after centering the test data for each feature. After obtaining the $\hat{y}$ predictions, we add the average of the outcomes given to us in the test data, in order to ”de-center” the data. By doing these steps, we hope to achieve the best possible estimates of $\hat{y}$, having reduced the chances of overfitting. The values of the parameters are shown in the following tables.

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9	β_{10}
0.018	0.000	0.075	0.328	-0.680	1.695	0.047	1.810	0.005	-0.013
β_{11}	β_{12}	β_{13}	β_{14}	β_{15}	β_{16}	β_{17}	β_{18}	β_{19}	β_{20}
-1.454	-0.707	0.030	-0.615	0.007	-0.375	-0.126	-1.362	-1.272	0.958

Table 2: Estimated parameters

We can see that, as proposed in a previous section, one of the parameters was set to zero by the regularization term. The predicted values for $\hat{y}$ were saved in the file ”$Ytest\_Regression\_Part1.npy$”.

1.3 Result analysis

After submitting the $\hat{y}$ predictions to the teachers, the MSE of those predictions was 0.016192650273241647. This is higher than anticipated, due to a mistake in the pre-processing of the data. The issue lies in the fact that the test data for x had been centered by subtracting it's own mean, when the correct procedure would have been to subtract the mean of the training data, such that the same pre-processing has been applied to both the test and training sets. This was determined to be the cause of the higher than expected MSE. However the regression as a whole can still be considered to be reasonably accurate and appropriate for the purpose of predicting outcomes for other sets of data.

2 Part 2 - Regression - Second problem

In this assignment, we were once again provided with a training set and test set. However, this time, there are an unknown number (less than 10%) of training examples that have not been generated with the same model as the majority of the data. We were tasked with once again training a predictor for the data in order to submit a vector of outcomes. This time, however, outlier detection methods must first be used to remove the points that do not follow the model.

2.1 Choosing an outlier detection method

Before training a predictor, it is necessary to attempt to remove the outliers, while taking care not to unnecessarily remove valid points. For this purpose, a variety of outlier detection methods were explored in order to select the most appropriate. The first methods that were tested were the Local Outlier Factor, Isolation Forest and Eliptic Envelope. However, using any of these three methods, improvements were noted to be minimal. This is due to the fact that they are not the most appropiate for outlier detection when the data is distributed as in this problem, as they only consider the density of points in x to decide on wether or not a point is an outlier. This can lead to outliers that have an output y very far from the expected value for the model not to be removed if the input values x are similar to those corresponding to the rest of the points. Since in this case the input values in x for the outliers are similar to the one for inliers, and it is the model used to generate the corresponding output y that is different, the aforementioned detection methods are ineffective. It was therefore necessary to explore alternative methods of outlier detection.

2.1.1 Comparison of Squared Error of each observation

As a first step, we decided to train a simple Linear Regression model to the entire set of training data. We hoped to find points that deviated significantly from the obtained curve, which would indicate that they may be outliers.

After training the model and predicting the values $\hat{y}$, we then calculated the values of the squared error for each observation, according to the expression:

$$e_i^2 = (\hat{y}_i - y_i)^2 \tag{1}$$

We then calculated the mean and the standard deviation (σ) of the squared errors, and considered any value that is located over 3σ away from the mean to be a potential outlier. We then remove these points from the dataset. This method is appropriate when the outliers are very distanced from the inliers, in which case they will most likely have the largest errors, however, since the outliers make the model adjust itself to them, as the observations become less disperse, they may not necessarily have the largest error.

2.1.2 Determining outliers based on Cook's Distance

The other method we considered was based on Cook's Distance. The Cook's Distance is a measure of the influence of a given point when performing a regression. This method aims to pinpoint certain points which may be causing the model to shift when they are included in the dataset. If a point is removed and the model predicts similar values, it means that said point probably was well integrated with the model, so its presence does not affect the estimated parameters too much. However, if we remove an outlier, the model will not compensate for its presence, and will therefore shift to parameters that better approximate the present values. This measure of "influence" (D_i) of a given point "i" is given by:

$$D_i = \frac{\sum_{j=1}^{n} (\hat{y}_j - \hat{y}_{j(i)})^2}{ps^2} \tag{2}$$

Where $\hat{y}_{j(i)}$ corresponds to the predicted value of the model trained without the point i, $\hat{y}_j$ corresponds to the predicted value of the model trained with the observation i and $p = 20$ is the number of features per observation. This means that D_i is the sum of all the changes in the regression model when the observation "i" is removed from it. Therefore, observations with large D scores when compared to the majority of observations will probably be an outlier. There is a commonly used threshold of $I = 4/n$, with n being the number of observations, such that observations with $D_i > I$ are considered to be potential outliers. However, we are advised to not stick to this threshold with no flexibility, as the primary indicator that an observation may be an outlier is the fact that its D score is much larger than the other observations.

2.1.3 Applying the methods to the dataset

After training the linear model to the whole dataset with 100 observations, we calculated the influence (D) and the square error (e^2) for each point, having obtained the following results:

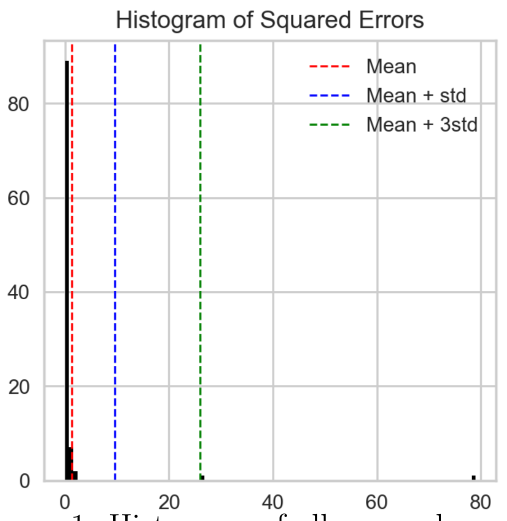

Figure 1: Histogram of all squared errors.

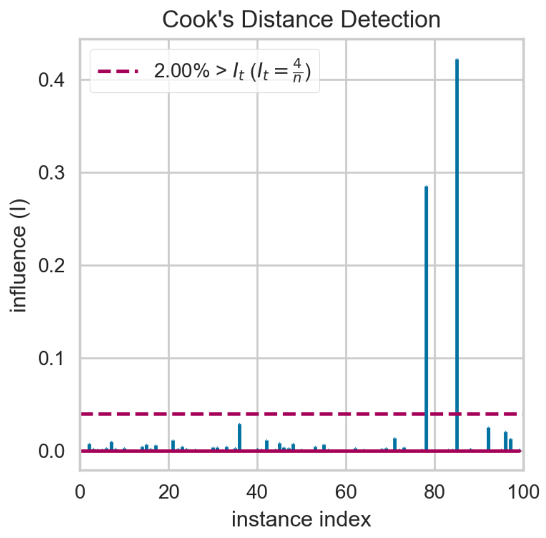

Figure 2: Cook's Distance for each observation.

We can clearly see that there are two abnormal values of (e^2) and of (D_i), being that these values are far superior to the average, and are too dispersed from the rest. After obtaining these values we compared the index of the abnormal points, and concluded that these corresponded to the same observations in each case. Therefore, we removed both the outliers, corresponding to observations 78 and 85.

After removing these observations, there was still a possibility that these values were so far from the model prediction, that they may have reduced the influence of other points that may still be outliers. In order to further evaluate the existence of outliers, we repeated the previous steps, that is, we trained a linear regression to the data, and analized the errors and D_i factors of the remaining 98 points. We obtained the following results.

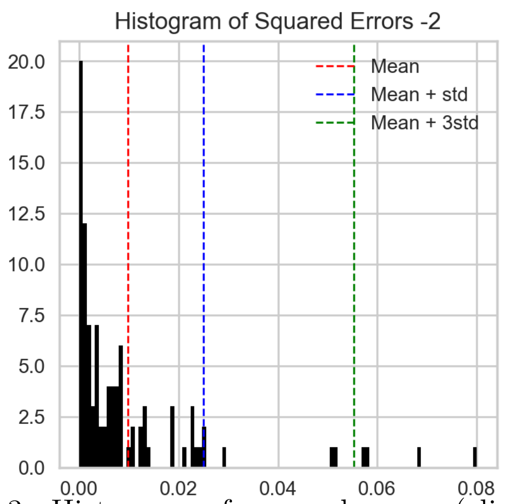

Figure 3: Histogram of squared errors (eliminated i=78,85).

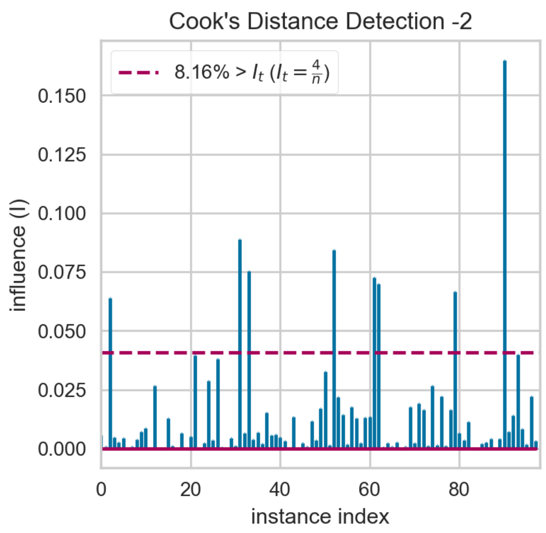

Figure 4: Cook's Distance for each observation (eliminated i=78,85).

As we can see, there are still points with larger errors than others, albeit with a much smaller difference, and we still have a point with a much larger D_i than the rest of the dataset. Having decided that we should remove the observation corresponding to the largest influence, we determined this point to be i=90. This point does not have the largest squared error ($e^2_{90} = 0.005$), however, it has one of the largest e^2, and the highest influence. Therefore, we choose to remove this observation, and repeat the process.

On our third and final iteration of our process we obtained the following results:

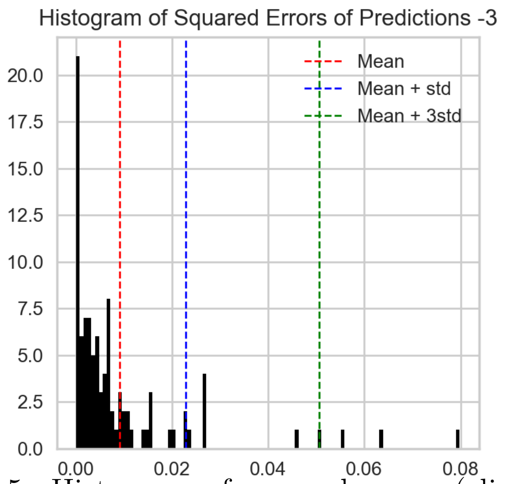

Figure 5: Histogram of squared errors (eliminated i=78,85,90).

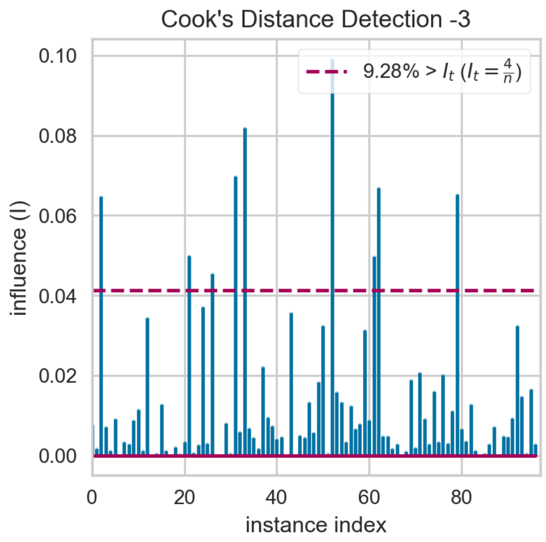

Figure 6: Cook's Distance for each observation (eliminated i=78,85,90).

Despite the fact that there are points with a larger error than the rest, and the fact that some of the values of D are higher than the supposed threshold for determining outliers, we decided not to remove any more observations. This is due to the fact that at this point larger influence does not immediately correlate to larger error, and the fact that there is a substantial amount of points with similar values of D, this leads us to conclude that no more points can be flagged as obvious outliers.

2.2 Estimating Test Outcomes

Once a satisfactory outlier detection method had been implemented, the resulting regression problem was similar in nature to the first one. For this reason, the process used will not be described in exhaustive detail.

However, it is worth mentioning that regression in this case was preformed through a Linear Regression model, as attemps to use Ridge or LASSO led to identical results for ideal λ, determined to be $\lambda = 0$. This is due to the fact that the data in this case did not benefit from regularization. Having obtained an average MSE of approximately 0.015, for a Repeated K-fold Cross Validation with k=10 and n=10 (k = number of folds and n = number of repetitions), we opted not to remove more observations, in order to avoid the risk of removing points that do, in fact, belong to the dataset.

2.3 Result analysis

After submitting the $\hat{y}$ results to the teachers, their MSE was calculated to be 0,013941596. While this is a good result, there was still some possible improvement. The main source of improvement might have been the removal of further points, as there were still points that seemed like they could be outliers, but were not removed as it was still reasonable to consider them as possibly being part of the dataset. While our approach might be correct to ensure no valid points are omitted, it may also be the case that the adverse effects of taking the risk of including an outlier outweigh the potential benefits from utilizing every single valid point. In this case it would have been correct to remove more points that seemed to deviate from the model to be completely certain of not including any outliers.

3 Part 2 - Classification - First problem

In this assignment, we were provided with a dataset of grayscale 50x50 face images, and tasked with creating a classification model to predict the gender of the subject of each picture, and return the label 0 for male or 1 for female.

3.1 Building the architecture

It was decided to accomplish this by creating and training a convolutional neural network (CNN), since this method is widely accepted to be the most appropriate for computer vision tasks. The specific architecture was determined by researching popular existing models, such as VGG and LeNet, and methods which were then adapted to our specific needs. After some experimentation with different configurations, too many to list extensively, we adopted a standard strategy of alternating convolutional layers and pooling layers, followed by a flattening layer and then a sequence of dense layers, which produce the final result. The *relu* function was used as the activation function, as this is also the standard choice for modern neural networks. At this point, adding more convolutions or pooling layers resulted in a negligible change in the validation accuracy, but cost us large amounts of time. We also added data augmentation, as well as Dropout and BatchNormalization layers, which will be discussed in section 3.2. All things considered, we settled for this structure:

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9	Layer 10
Conv2D(32)	MaxPooling	Dropout(0.2)	BatchNorm	Conv2D(64)	MaxPooling	Dropout(0.2)	BatchNorm	Conv2D(128)	Dropout(0.2)
Layer 11	Layer 12	Layer 13	Layer 14	Layer 15	Layer 16	Layer 17	Layer 18	Layer 19	
BatchNorm	Flatten	Dense(256)	Dropout(0.2)	BatchNorm	Dense(128)	Dropout(0.2)	BatchNorm	Dense(2, activation = softmax)	

Table 3: Model: CNN-1

The final layer has a *softmax* activation function in order to return the probabilities of belonging to any of the classes. The data was also normalized before any of the other layers, by dividing it's values by 255, so that all values fit into a range of zero to one. In between every layer a batch normalization layer was added to increase training speed and the stability of the network. The addition of the batch normalization layers seemed to help with overfitting as well.

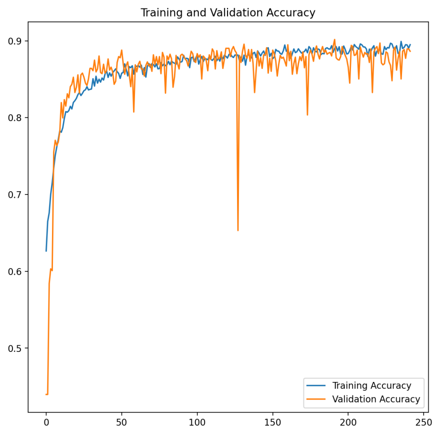

Figure 7: CNN-1

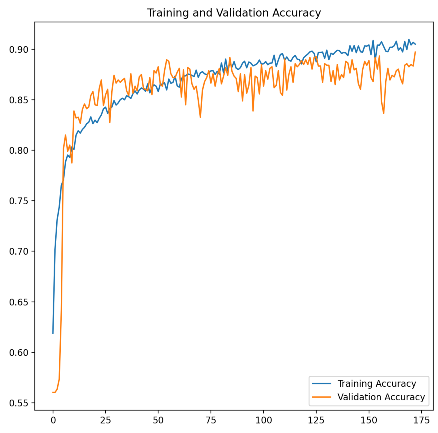

Figure 8: CNN-1 without batch normalization

In the following section 3.2 we will discuss the impact of the Dropout layers, as well as the use of Data Augmentation. The model CNN-1 will be used as the baseline, and compared with a model where the Dropout layers were excluded, or where data augmentation was not applied on the training set.

3.2 Reducing overfitting

With the basic model set up, we were faced with the problem of overfitting, which results as a consequence of training the model on a limited amount of data for a large amount of epochs, causing it to become overly specialized for the training dataset and therefore less adequate to predict outcomes for new sets of data. The most important tool used to counter overfitting was the use of a validation set. A validation set is a set of training data that is separated from the rest before training, so that the accuracy of the model can be checked against data that was not used in it's training. Monitoring the validation accuracy instead of the training accuracy therefore gives us a better idea of how well the model performs on unseen data.

Training done using a validation set was performed to determine the optimal amount of epochs to then train the full set of data on. The training and validation accuracy and loss were plotted, which makes it easy to see how many epochs it takes them to stabilize. An early stopping criterion was used to determine the exact epoch at which the best validation loss had been obtained, and terminate training after no improvement in the validation loss for 50 epochs

3.2.1 Data augmentation

The first step taken towards the reduction of overfitting was to introduce data augmentation. Data augmentation works by applying random transformations to the images, to increase the amount of data available for training, without damaging the facial features that our CNN attempts to extract. In this case, random rotations and zooms were utilized to create modified versions of the existing images, creating new, distinct training examples without the need for additional external data. It is essential that the data augmentation is only applied to the training set, and not the validation set. In the following images we demonstrate the impact of introducing data augmentation in our model.

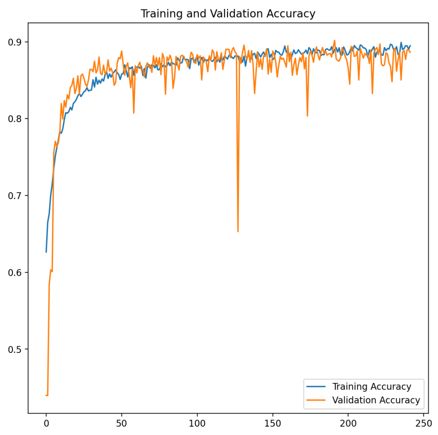

Figure 9: CNN-1

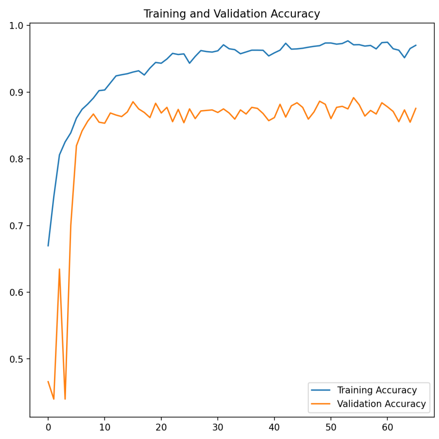

Figure 10: CNN-1 without performing data augmentation

It is clear to see that the use of a data augmentation layer helps to reduce overfitting to the training data. By using data augmentation, the early stopping criterion is only activated much later, and we achieve validation accuracy on par with the training accuracy for many more epochs.

3.2.2 Dropout layers

Another measure to counter overfitting was the introduction of dropout layers after every dense or convolutional layer. Dropout layers randomly set input units to zero. In this case, the dropout layers were configured to set 20% of the input units to zero each layer. After introducing these layers we noticed that the validation accuracy and training accuracy did not diverge as significantly.

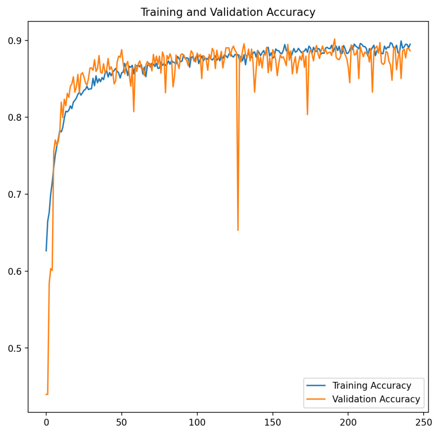

Figure 11: CNN-1

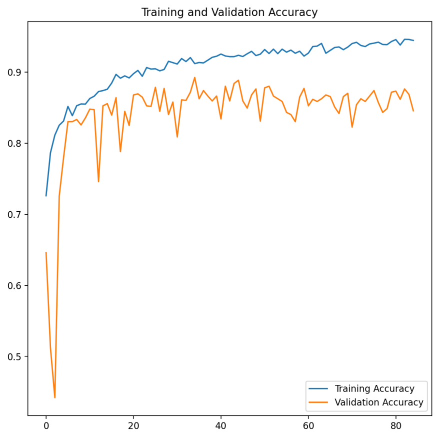

Figure 12: CNN-1 with no Dropout layers

It is also noticeable that the use of Dropout layers enables us to train the model for many more epochs without the risk of overfitting, which in turn allows us to achieve better accuracy scores.

3.3 Final model training

After testing extensively on the CNN-1 structure, using different methods to combat overfitting and to achieve higher accuracies, we were confident on the structure of our model. However, as a final test we attempted to change the number of feature maps produced by each convolution layer, and widening the dense layers. A "wider" network, that is, one that produces more feature maps per convolution layer, and has more nodes in the dense layers seemed to produce higher validation accuracy.

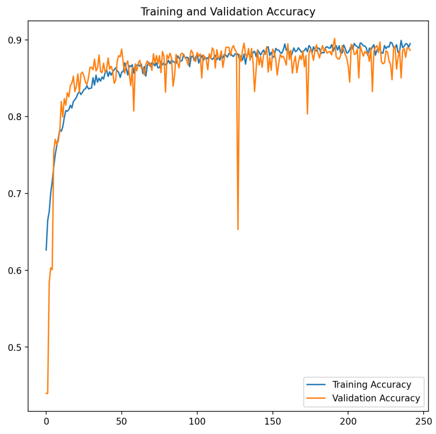

Figure 13: CNN-1: Validation Accuracy: 90.2%; Balanced Accuracy: 89.9%

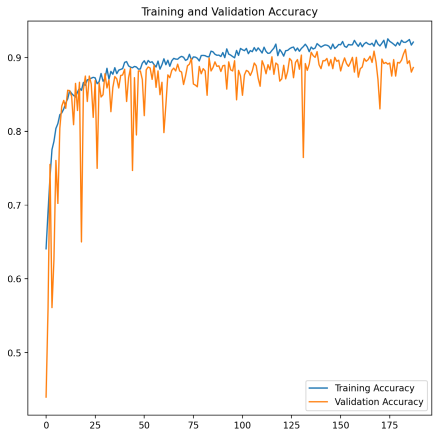

Figure 14: CNN-2: Validation Accuracy: 91.1%; Balanced Accuracy: 90.8%

Our final structure was:

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9	Layer 10
Conv2D(64)	MaxPooling	Dropout(0.2)	BatchNorm	Conv2D(128)	MaxPooling	Dropout(0.2)	BatchNorm	Conv2D(256)	Dropout(0.2)
Layer 11	Layer 12	Layer 13	Layer 14	Layer 15	Layer 16	Layer 17	Layer 18	Layer 19	
BatchNorm	Flatten	Dense(512)	Dropout(0.2)	BatchNorm	Dense(256)	Dropout(0.2)	BatchNorm	Dense(2, activation = softmax)	

Table 4: Model: CNN-2

Analizing the graphics it is possible to see that the network starts to overfit around the time where the early stopping stops the model from training further. This is to be expected since a more complex model is more likely to exhibit overfitting issues. However, a more complex model can also be better at extracting features from images, and, seeing as the accuracy and balanced accuracy obtained from the CNN-2 model were superior, we decided to use the CNN-2 model for our final submission.

The model was left to run for up to 1000 epochs, with an early stopping criterion set up. Afterwards, the predictions were made not through the model obtained after the final epoch, but through the model with the best accuracy, which was saved to a '.h5' file through the model_checkpoint function.

As the optimal number of epochs seemed to be around 100 after repeated trials, this was the final value chosen to train using the entire data set. The final training was done without a validation set or any kind of early stopping, as these had served their purpose of helping to determine the optimal amount of epochs to train for. Having obtained the final predictions, these were submitted.

3.4 Result analysis

After submitting the predicted outcomes to the teachers, these were rated as having a balanced accuracy of 0.905672184. This is a quite satisfactory result, though there still remains the possibility of obtaining even better results. One possibility would have been to submit an average of the results of multiple trainings under the previously described conditions, as this might help reduce the influence of statistical fluctuations due to the stochastic nature of the method. Additionally, it is possible that a somewhat different architecture would have yielded even better results, as every data set is distinct, and the sensibility of the results to changes in the amount or type of layers is unpredictable. However, as the results were generally positive, we considered our model and it's implementation to have been successful.

4 Part 2 - Classification - Second problem

In the final assignment we were presented with a similar dataset as in the first part and tasked with creating a classification model to predict the ethnicity of the subject of each picture.

4.1 Building the architecture

The choice of architecture for this problem was very similar to the previously developed one, as we were dealing with a similar dataset. However, a few alternatives were explored again to confirm wether or not the previously tested optimizations were still preferable in this case. For the sake of brevity and legibility, the graphs for cases that yielded the same conclusions reached in the previous case have been omitted.

As a starting benchmark upon which to iterate, the previous model (Model 1) was utilized, with the final dense layer now set to output four values instead of two. Then, a few variations of it were tested to see if they would improve or worsen the balanced accuracy. The data for each of these models is shown in table 5, at the end of the section.

As the new problem involved a more complex situation, with four potential outputs instead of two, our first hypothesis was to increase the complexity of the model by adding an aditional convolutional layer (Model 2).

Another potential source of innacuracy was deemed to be the amount of dropout layers used, as the data is very unbalanced in it's distribution, having a lot more subjects with the caucasian label than any of the others, with a remarkably low rate of africans. For this reason, having too much dropout could result in eliminating a large amount of the results stemming from the african data samples, lowering the balanced accuracy. Therefore, a model was trained with half the amount of dropout layers of the original one (Model 3) or with the same amount of layers but set to only drop out 10% of the data each (Model 4).

To ensure that the model was not too complex, it was also attempted to reduce the amount of filters in both the convolution and dense layers by half (Model 5).

Finally, we tried to introduce an l2 layer weight regularizer to the kernels of the convolution layers which applies penalties on layer parameters during optimization. (Model 6).

Model	1	2	3	4	5	6
BAcc	0.794	0.776	0.767	0.791	0.774	0.722

Table 5: Balanced accuracy of each model

Analyzing the results, the two clear favorites were determined to be models 1 and 4. Both increasing and decreasing the complexity of the model seems to have adverse effects. This is due to the fact that while there are now four outputs, the data that is being used for training is of the same type as the data used in the previous problem, and the same type of architecture is therefore best suited to handling it, despite the increase in labels. However the difference in balanced accuracy between models 1 and 4 remains too close to be able to pick a clear favorite, as the difference could very well be affected by stochastic fluctuations. For this reason, graphs comparing their training accuracy and validation accuracy were generated to compare how much the influence of overfitting could be felt for each model.

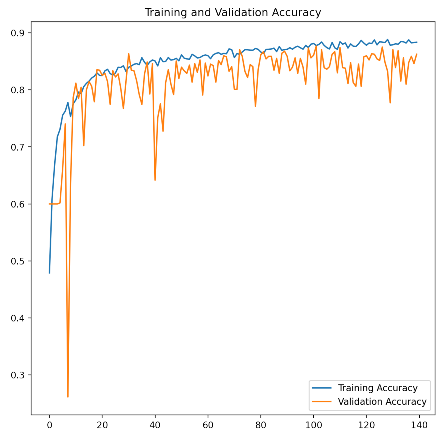

Figure 15: Model 1 - Training Accuracy vs Validation Accuracy

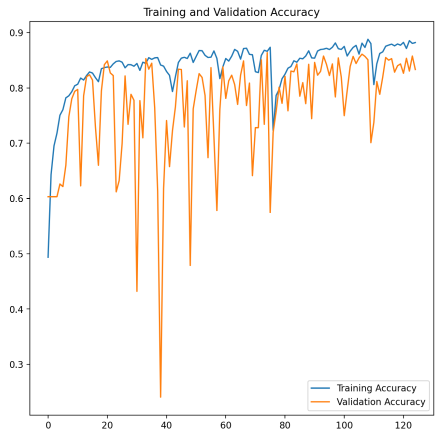

Figure 16: Model 4 - Training Accuracy vs Validation Accuracy

By analyzing the graphs as well as the numerical values for the accuracies, Model 1 was determined to be less prone to overfitting, and therefore chosen as the final model. The model with decreased dropout rates resulted in inconsistent results for the validation accuracy and was discarded, as decreasing the chance to dropout valuable information was not worth the trade off of increasing inconsistency and overfitting.

4.2 Dataset and Prediction Analysis

In this part 2 of the Classification problem, there was a much larger disparity between the balanced accuracy and the validation accuracy. This is to be expected, since the distribution between classes is very uneven. Upon analysis of the dataset we found the proportion of each ethnicity. We also calculated the validation accuracy for each of them.

	Caucasian	African	Asian	Indian
Proportion (%)	60.3	4.7	18.8	16.2

	Caucasian	African	Asian	Indian
Accuracy (%)	91.5	61.6	86.0	74.5

There seems to be a correlation between the amount of data for each class and the accuracy of the predictions in each of them. This makes sense, since having more data in a class can both make the model more adequate to predict said class, since correct predictions of that class will further diminish the loss when training. This results in a model that best identifies one of the classes while neglecting the less frequent ones. We attempted to create our own dataset, with the same number of subjects from each class, but it was too small to yield any results worth discussing.

4.3 Result analysis

After submitting the predictions to our teachers, they were rated as having a Balanced Accuracy of 0.81021464. This is a very good results given the small size of the dataset and it's imbalance. This method was therefore deemed successful.