COURSEWORK 1:

I. INTRODUCTION

Maternal mortality is at a rate it simply should not be with the advances of modern medicine. According to WHO [1], in 2017, approximately 295,000 pregnant women died during pregnancy or following childbirth. Most instances belonging to low-income countries, and the majority of which could have been avoided. A lack of knowledge when it comes to maternal healthcare is a large contributing factor.

Data has been collected from varying sources, such as hospitals, clinics, and maternal health care centres through the Internet of Things –(IOT)- based health monitoring systems. This dataset contains attributes that affect maternal health. I am interested in investigating which of these health conditions have the strongest link/effective to health risks during pregnancy. The dataset also includes 3 class features. By using machine learning algorithms to classify these features into one of these three classes, hospital and other health institutions can use patients' health data classify their risk level during pregnancy. Therefore, appropriate precautions can be taken to reduce maternal mortality.

Four models were trained and tested and optimized as a part of this machine learning pipeline. These were logistic regression, K Nearest Neighbour (Knn), Decision tree and Support Vector Machine (SVM). Both Knn and Decision tree performed at an accuracy and precision score of 80%. Therefore, both classifiers are suitable, but decision tree is recommended over Knn.

II. THE DATA AND PRELIMINARY ANALYSIS

A. *The Dataset*

This dataset contains 6 attributes: Age, Systolic Blood Pressure, Diastolic BP, Blood sugar, Body Temperature and Heart rate. All these features are thought to be important risk factors in maternal mortality and are therefore important to monitor during pregnancy and childbirth. The dataset also includes a class variable 'Risk Level' which falls into 3 categories, 'High Risk', 'Mid Risk' and 'Low Risk'. I began this investigation through exploring this dataset. There are 1014 observations and 6 attributes in this dataset, providing enough data to be able to run classification algorithms on. The number of observations is not equally distributed between class variables with high risk posing the least number of observations (272/1014). This is something to consider when building the algorithms as an imbalanced dataset can lead to algorithmic bias where the algorithm is not able to classify high risk patients as well as mid and low risk patients.

B. *Preliminary Analysis*

Data visualisation is a very important aspect of preliminary analysis as it allows you to be able to understand the data better and make assumptions that will affect model design.

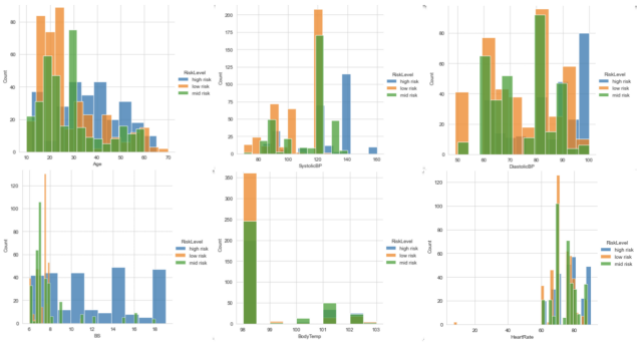


FIGURE 1: ATTRIBUTES VS CLASS VARIABLES

The first thing I wanted to visualise was each features relationship with the class variables (Fig.1). This allows visualisation of any correlation a feature may have with classes. It also contributes to attribute selection, allowing for early-stage removal of any attributes that seemingly have no effect on target. From this analysis of attributes, there seems to be a similar trend with the higher the value the stronger the association with a high-risk pregnancy. This is particularly clear with blood sugar levels. At this stage all 6 attributes seem like relevant features to influence the class. I also computed a correlation matrix (Fig2.) This matrix also allowed for comparison of all attributes to see if any attributes were related.

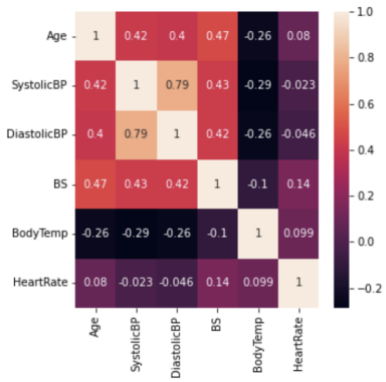


FIGURE 2: CORRELATION MATRIX ON FEATURES

There is seemingly no strong correlation between any of the attributes apart from Systolic and Diastolic BP which have a correlation of 0.79. BodyTemp seems to have very low correlation with all the attributes. However, as this is a correlation matrix between

features and is not comparing the correlation with class, this is not enough reason to remove body temp as an attribute. I have decided to keep all these attributes in my model building.

III. Methods

The Machine Learning approach deployed in this task is shown in Fig. 3. This pipeline is split into 3 main stages: Pre-processing, Model training and evaluation. Each of these stages include necessary steps that are usually taken in any machine learning pipeline.

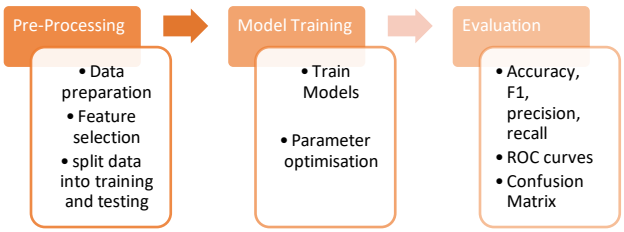


Figure 3: Machine learning pipeline

A. *Pre-processing*

This is the first step in my approach to solving this task of building an optimized classification model. This step includes three main stages. The first stage is data preparation which itself includes a couple steps such as familiarisation of the raw data, cleaning, encoding and normalisation. The second stage was Feature selection. To do this, I visualised each features relation with the class (Fig.1) and computed a confusion matrix to compare features (Fig.2). All the features seemed to show a trend with the class variables and with only 6 attributes I decided to move forward with them all. Finally, I split my data into training and testing data with a 30% hold out on data.

B. *Model Training*

This is the stage of my pipeline where I train 4 models that I have chosen for this experiment. Each model is trained with the training dataset and tested with the testing dataset from the split. This allows for the evaluation of the model performance on unseen data [2]. Once the model is tested on the testing data, we can run evaluation analysis and fine tune the parameters to make the optimal version of this model. This is applied to all 4 models used in this task.

C. *Evaluation*

This is the final step of this machine learning pipeline. Once the individual models have been fine-tuned and optimised, we run a final analysis on the models through the production of a classification report. This produces data on performance of each of the algorithms and allows the identification of the model which will be recommended for this classification task.

IV. Experiments

A. *Pre-Processing*

The raw data was imported into my editor of choice (Jupyter Notebook) and analysed to get a better understanding on the data. Data visualisation also contributed to feature selection, although no features were removed from this dataset. I also used a label encoder to ensure the class variables were numeric. Most algorithms expect numerical values to produce optimal results [3]. The class variable is this case are nominal and need to be changed to numeric values. I used a label encoder that changed the class variables; ‘High Risk’, ‘Mid Risk’ and ‘Low Risk’ to 0,1 and 2 respectively. Therefore, allowing for the data to be classified into one of these 3 values.

Next, I checked for missing data as this is an important part in any data preparation task. There was no missing data in this dataset and therefore I moved onto statistical analysis. The mean off all the attributes looked to fall within expected values. Finally, Normalisation/standardisation was considered for this dataset, but I decided to apply this to each model individually only if performance was improved.

B. *Model Training*

This is the main stage of this pipeline, where I will be using the cleaned and prepared data to train algorithms and optimize them. The four algorithms used in this task are: Logistic regression, Knn, SVM and decision tree. All these algorithms are suitable in supervised learning and for classification tasks.

1) *Logistic regression*

Regression is a type of supervised learning which involves the prediction of continuous outcomes. Regression analysis involves using data that includes variables and a target value to find patterns in the data that can help predict an outcome. Logistic regression involves a calculation of probability of dependent variable given an independent variable [4]. Before running this first algorithm on the testing data, I used Minmax scaler to normalize the data but realized it had no difference in performance. Therefore, I ran the model on the raw data for simplicity. The performance accuracy of this model was 61% and with this algorithm

there are not many critical hyperparameters to tune [5]. Therefore, I moved onto my next algorithm.

2) Knn Classifier

Knn is another supervised learning algorithm. It is a pattern recognition algorithm [4] and is fine-tuned by the value of K. K represents the number of nearest neighbours to include in the classification process. Knn uses a training dataset to classify datapoints into a category of its nearest neighbor. Another parameter in Knn is the weighting matrix. The weight value gives less or more weight to points nearby. Both parameters were fine tuned in the process of training this model. The weighting for both 'uniform' and 'distance' was firstly compared and the latter performed the better knn model in each case by almost 10% in accuracy. Also, the optimum K value was found to be K=10. Although the improvement in accuracy was not great. Normalisation of the data improved the accuracy of the model but not greatly. The optimal model was with k=10, weighting = 'distance' and normalisation applied.

3) Decision Tree Classifier

The next algorithm was a decision tree classifier. This is another classification algorithm. It works by ordering classes on a precise level. It is described to be like a 'flowchart structure' where each node represents a feature, and each leaf represents a class label [6]. I ran this model with and without normalization and it seemingly had no considerable effect on model performance. The first time I ran this model, I set the parameters criterion='gini', random_state=0, max_depth=3, min_samples_leaf=2. The accuracy with these parameters was at 65%. I then reran this algorithm with the default parameters of the decision tree in sklearn which improved the accuracy to 80%.

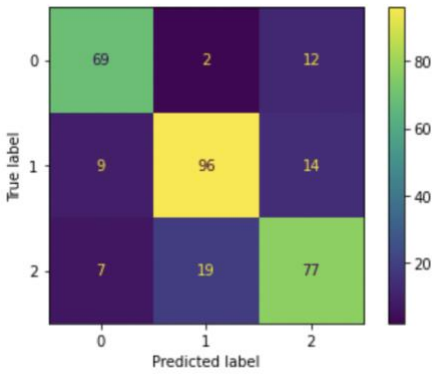


FIGURE 4: DECISION TREE CONFUSION MATRIX

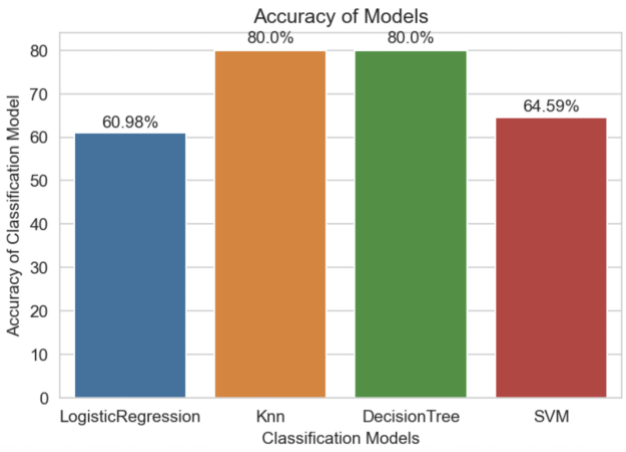
Fig. 4 shows the classification of data points into one of the 3 classes and the number of correctly classified instances vs incorrectly classified instances. It correctly classifies 69/83 high risk cases which is a precision score of 83.1%. It also classifies the other 2 classes at a similar level.

4) SVM Classifier

Finally, the last algorithm is the SVM classifier. This is a very effective and popular machine learning algorithm, and it provides many parameters to tune. The aim of this algorithm is to find a hyperplane in a dimensional space that can distinctively classify the data points. The most important parameter is the kernel and I created a for loop to test 'linear', 'poly' and 'rbf'. The order in performance accuracy was poly>linear>rbf. The accuracy with the poly kernel was 64.5%.

5) Comparison

These models were then applied to the testing dataset and their performance was evaluated through classification reports. The accuracy, precision, and F1 scores were compared to get ranking of the algorithms (Fig 3.).



Performance Metrics	Logistic Regression	Knn	Decision Tree	SVM
Accuracy	60.98%	80.00%	80.00%	64.59%
Precision	60.00%	80.03%	79.1%	65.1%
F1	60.00%	80.00%	79.1%	62.4%

FIGURE 5: MODEL PERFORMANCE COMPARISONS

The Figure above shows that the Knn algorithm and Decision tree algorithm performed very similarly in all aspects and they both performed marginally better than the other 2 algorithms. Although Knn and Decision tree performed similarly, it is the decision tree that will be

recommended as this classifier supports automatic feature interaction and is much faster than Knn [7].

C. Evaluation

The decision tree classifier is selected as the best performing algorithm that will be recommended for the classification of risk level during pregnancy. Fig 5. Shows the receiver operating characteristic (ROC) curve for the optimized decision tree which allows for evaluation of the model's performance.

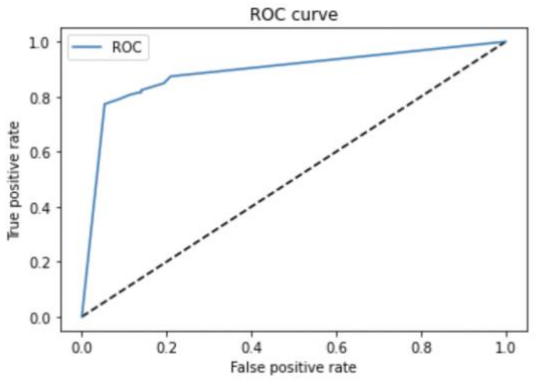


FIGURE 6: DECISION TREE ROC CURVE

The ROC curve plots the true positive rate vs false positive rate at various threshold values. The AUC score was 0.89. The AUC measures the ability of the classifier to categorise data into different classes. The closer this value is to 1, the better the performance of the model. This AUC score validates the high performance of this model.

V. REFLECTION

This machine learning pipeline provides detailed steps taken to produce a suitable and well performing classification model. All models are compared, and the decision tree classifier is proposed as it produced the highest accuracy (80%). The main problem I encountered during this task was fine tuning the parameters of the models to get a better performance. The decision tree originally gave an accuracy score of 65%. Changing the Max depth to the default parameter of none improved the accuracy to 80% however this means that the nodes expand until all the leaves are pure which can produce overfitted results. Therefore, next time I can try tuning other parameters whilst setting a max depth figure. Next time, I would explore further algorithms such as random forest classifier by incorporating ensemble learning. This learning method uses bagging, stacking, and boosting to obtain better predicative performances. I would also place a higher emphasis on fine tuning the hyperparameters. Overall, this decision tree classifier performed well.

COURSEWORK 2:

I. INTRODUCTION

This is a deep learning image classification task. An image dataset containing 10 classes of meaningful objects has been provided and I have used an existing convolutional neuro network (CNN) architecture to classify these images into the predefined classes.

A CNN is a class of neural networks that is typically used in image recognition and processing. It works by assigning importance to features in an image to be able to differentiate it from other images. Whilst traditional machine learning methods use feature engineering for feature selection in classification, CNN use neural networks to learn both feature extraction and classification without the need to do feature selection separately.

An existing CNN model has been adapted and optimised through data augmentation and dropout technique to produce a validation accuracy of 69% in classifying these images.

II. THE DATA AND PRELIMINARY ANALYSIS

A. The Dataset

The first thing I did was examine and understand the dataset provided. This allows me to build a better model. The dataset includes 10 classes of images consisting of Fish, Dog, Stereo, Chainsaw, Church, French horn, Truck, Petrol Pump, Golf Ball and Parachute images. The dataset was already split into training and validation datasets with a 70/30% split. The training dataset included 10009 images belonging to 10 classes and the validation dataset included 3945 images belonging to 10 classes.

B. Preliminary Analysis

Next I randomly visualised 9 images in the training dataset (Fig. 7). As can be seen there are different classes of images each. I can also observe that the images in each class are varying. For example, there are two different chainsaw images in this sample. The size of the objects in the image also varies which can be seen through the pictures of fish. This is an important observation as the

model will need to extract the features with high detail to achieve a high accuracy.

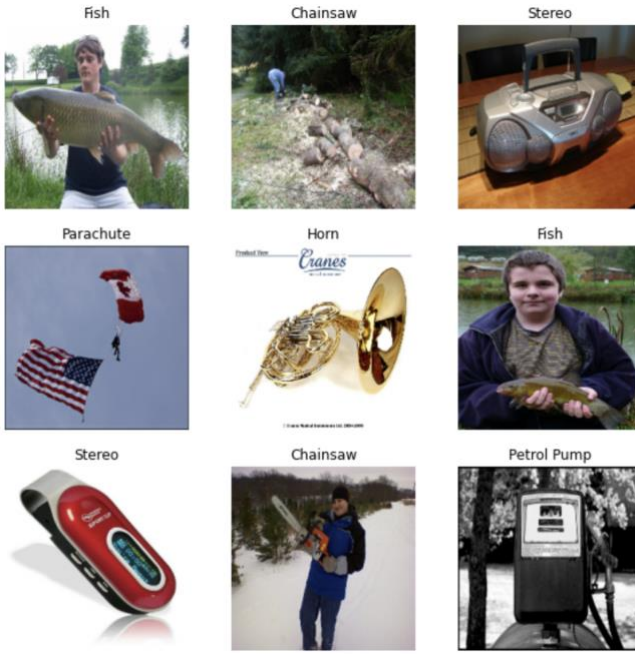


FIGURE 7: SAMPLE IMAGES FROM TRAIN DATASET

I have also configured the dataset for performance, specifically I used cache and prefetching to configure the dataset and avoid blocking during model training. I also standardised the data. Typically, the RGB values are in the range of [0, 255] and therefore I have computed a rescaling layer to standardise the pixel RBG values to a range of 0 to 1. This scale is more appropriate for neural networks

III. METHODS

The machine learning approach for solving this task follows a simple workflow. The first step involves examining and understanding the data like most machine learning tasks. The next stage is building and training the model. Once the CNN model is built (Fig.8), I will use the validation dataset to test the performance of the model and finally move onto improving the model and retesting.

I have used an existing TensorFlow Keras sequential model [8]. This model consists of three convolutional layers with max pooling layers in between followed by a fully connected layer. It is important to note that the optimization section of this task is not to achieve a high accuracy but rather to improve the training/validation gap by reducing overfitting. The two techniques used to reduce overfitting in this model are data augmentation and including a dropout layer in the model architecture.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 224, 224, 16)      448

 max_pooling2d (MaxPooling2D  (None, 112, 112, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 128)               6422656

 dense_1 (Dense)             (None, 10)                1290
```

Figure 8: Base CNN model architecture

A. *CNN Architecture*

Each layer in this CNN model is important in transforming the input data into output predictions. In this section I will briefly explain the use of each layer.

Convolutional layer

This layer is the core layer of the CNN which conducts the main computational load [9]. This layer creates a convolutional filter known as a kernel that slides across the height and width of the input image and produces an activation map that extracts features at each position of the image. This is how the model can recognise features of an image and later classify it. The output of the convolutional layer is a vector and is passed onto the next layer.

This CNN model uses a 3x3, 2D convolutional layer, the most common type of convolution and kernel size. This kernel size was already implemented in this model and is popular as a small kernel allows for better feature extraction, particularly as the image dataset I am using contains images of different sizes belonging to the same class. CNN are a highly non-linear feature extraction method and therefore an activation function is almost always necessary. It defines how the input signal is transformed into output and how

well the CNN is trained. The type of activation function used in the model has a great effect on the performance of the neural network. This CNN uses Rectified Linear Activation (ReLU) which more commonly used compared to Sigmoid and Tanh functions because of its simplicity to use and effectiveness. If the input value is negative, then a value of 0 is computed and the neuron in the model is not activated.

Pooling layer

This is the next layer in this CNN architecture design. The max pooling layers are in between the convolutional layers and work to calculate the maximum values for each region of the feature map covered by the kernel. It outputs a matrix with reduced parameters and is helpful in reducing overfitting of the model. This reduced matrix is containing the most prominent features of the original feature map which means that the no matter where the image is in the frame it can be recognised, therefore improving the model's classification accuracy. This max pooling layer also reduced the number of pixels in the output and therefore the computational load, allowing the overall CNN model to be for efficient.

Fully connected layer

The fully connected layer is the final layer of the CNN and computes the output of the convolutional and pooling layer to produce a final output. This is where the classification is computed. Prior to input into this layer the matrix is flattened to convert it to vector values for this fully connected layer. Now the model can distinguish between low level features in the images and classify them.

To compile this model an Adam optimiser is used along with a spare categorical cross entropy loss function.

B. Model Training

Now that the CNN is designed, through a series of epochs the model is ran with the training and validation data to train and test the model. In this task the model is initially ran over 5 epoch and then undergoes optimization and ran a second time with 10 epochs. With each epoch the accuracy and validation accuracy should improve.

C. Evaluation

The final stage of this task is evaluating the model's performance through the accuracy and loss values. Once an optimal model is created this can be tested on testing data including unseen images to classify the image type. Although a testing sample has not been created in this task due to the data size not being big enough.

IV. EXPERIMENTS

A. CNN Training

The CNN model used in this task was developed from a sequential model for classifying images [8]. The original model (Fig.8) was trained with the training and validation data with 5 epochs. The number of epochs relates to the number of times the model runs over the training dataset. The more times it runs through the dataset the better the accuracy in classification. I initially began with 5 epochs (Fig. 9) and in the optimized model I ran on 10 epochs. The number of training batches were 313 and 124 for the validation with a batch size of 32.

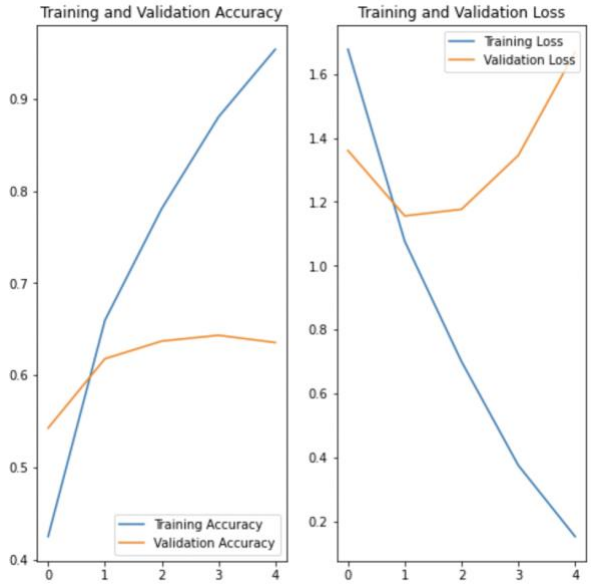


Figure 9: Original CNN model performance (5 epoch)

As can be seen from Fig. 9 the training accuracy drastically improves over the 5 epochs reaching a peak

accuracy of 95%. However, the validation accuracy is much lower peaking at only 64%. Similarly, the training loss falls to almost 0 whilst the validation loss falls and then begins to increase. What can be interpreted from this figure is that the model performs marginally greater on the training dataset compared to the validation dataset. This suggests that there is overfitting in this model and therefore it will equally not perform very well in classifying a test dataset of images. This task implements two techniques, data augmentation and dropout to reduce overfitting and improve the overall performance of the model. Again, the aim of this task is not to improve the accuracy but to improve the performance of the model.

Data Augmentation

Generally overfitting occurs because the training dataset is too small, and the model learns from the noise of the dataset and generally becomes too specific to the training dataset. To address this problem, data augmentation is implemented. This technique creates virtual training samples through random data augmentation such as random flip, random rotation, and random zoom. This way the model becomes familiar with more aspects of the images. Therefore, an augmentation layer was created to include in the optimized version of the model.

Dropout

The second technique used to reduce overfitting was introducing a dropout layer in the model. This layer randomly removes several output units through changing the activation to 0 before inputting into the fully connected layer.

B. Evaluation

A new optimised CNN model which includes the augmentation and dropout layers was produced. This was fitted on the training and validation datasets to test the change in accuracy and loss and overall performance of the model. This time the epoch was set to 10. The results are shown in Figure 10. It is clear here that the gap between the training accuracy and validation is reduced which shows that the overfitting of the model is also reduced. The overall validation accuracy from the previous model has also improved, this time reaching an optimal level of 69%.

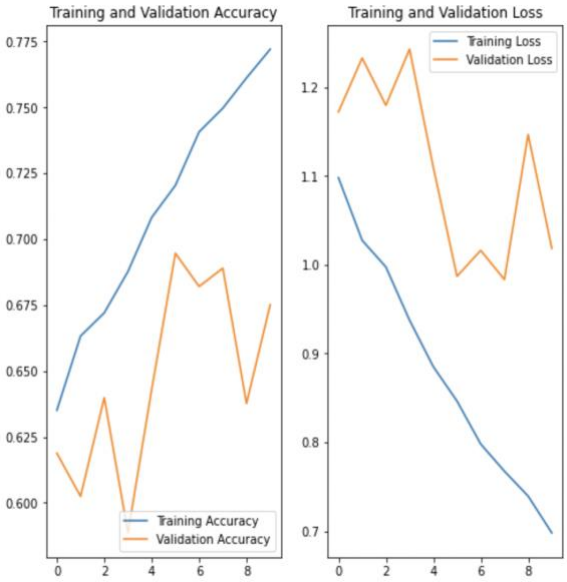


Figure 10: Optimized CNN model performance (10 epoch)

Similarly, the gap between the training and validation loss is smaller. This time rather than the validation loss increasing it is on a steady decline. Overall, this is an improvement to the model, and it would perform better if fitted to a testing dataset compared to the original CNN.

V. REFLECTION

The overall optimal performance I achieved through this CNN is a validation accuracy of 69%. This is a good level of accuracy considering the optimization in this task was not to improve the accuracy but rather to reduce the overfitting of the model. If I were to further develop this CNN, I would evaluate other parameters such as the kernel size and the number of convolutional layers. Another thing I would investigate further is the why the validation accuracy varies between the epochs. One problem I had during this task is managing the computational load. Therefore, the max epoch I used was 10. If I were to try again, like many other machine learning algorithms I would use a higher epoch to train the model better. Although there is room for improvement in terms of the classification accuracy of this model, I believe the overall performance and optimization of this model is satisfactory.

REFERENCES

[1] Who.int. 2019. *Maternal mortality*. [online] Available at: <https://www.who.int/news-room/fact-sheets/detail/maternal-mortality> [Accessed 13 March 2022].

[2] Brownlee, J., 2020. *Train-Test Split for Evaluating Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning

[3] Yadav, D., 2019. *Categorical encoding using Label-Encoding and One-Hot-Encoder*. [online] Medium. Available at: <https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd> [Accessed 13 March 2022].

[4] Wolff, R., 2019. [online] Available at: <https://monkeylearn.com/blog/classification-algorithms/> [Accessed 13 March 2022].

[5] Brownlee, J., 2020. *Tune Hyperparameters for Classification Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/> [Accessed 13 March 2022].

[6] H. Fang. (2022). Machine Learning [PowerPoint slides]. Available at <https://learn.lboro.ac.uk/>

[7] Varghese, D., 2018. *Comparative study on Classic Machine learning Algorithms*. [online] Medium. Available at: <https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222> [Accessed 13 March 2022].

[8] TensorFlow. n.d. *Image classification | TensorFlow Core*. [online] Available at: <https://www.tensorflow.org/tutorials/images/classification> [Accessed 17 March 2022].

[9] Mishra, M., 2020. *Convolutional Neural Networks, Explained*. [online] Medium. Available at: <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939> [Accessed 17 March 2022].

Appendix:

Google CoLab share links:

COURSEWORK 1:

https://colab.research.google.com/drive/14y-R6nN83NgNYXe62SVpqgHquFfm3Nnr?usp=sharing

COURSEWORK 2:

https://colab.research.google.com/drive/16EeyZ83sCnHEl0jLt_TPqgVVFJa_kM5B?usp=sharing

Note: I have also uploaded the ipynb files in case the links do not work.