Applied Machine Learning Coursework

Task 1: Machine Learning Pipeline

I. Introduction

Absenteeism at work has long been widely understood as one of the primary drivers for financial loss in the working world, and it is on a steady incline [2]. as workplace expectations intensify to compete with an ever-growing competitive landscape.

Coronavirus allowed for this financial problem to fade into the back of the minds of management teams across the globe. For many organisations the productivity of staff teams spoke for itself as now organisations such as Facebook aim to enable a post-covid WFH (work-from-home) culture [1] However, for the UK government and many other businesses initiatives to bring people back into the office have been all but inconspicuous.

The dataset used here is based on employees from a courier company in Brazil and is available within the UCI Machine Learning Repository [3]. Whilst the physical nature of the role may not translate directly into the office environment it is my hypothesis that many of the psychological factors recorded within the attributes measured can be of value for HR management. I hope that future studies will build onto my findings here. I will be using multiple supervised machine learning techniques as to identify which attributes an employee may carry best predict absenteeism. Through this exploration I will offer insight into the challenges faced by employees whilst they are physically present at work – aiming to support management teams with controlling staff motivation; as well as potentially highlighting opportunities where offering WFH solutions may alleviate rises in absenteeism through classification techniques.

This report will be structured as follows; section 1 for introduction, section 2 for preliminary analysis and assumptions, section 3 for methodology, section 4 for experimentation and section 5 for my reflections.

II. Preliminary Analysis and preprocessing

As mentioned earlier the dataset is a stratified sample taken from a Brazilian based courier organization, when an employee was absent, they would complete a form detailing reason for absence.

The dataset contains 740 instances and 21 attributes, there are no missing data. The information recorded to predict absenteeism is made up of a variety of potential direct, e.g., travel expense, and indirect, e.g., smoking habits, influences. Table 1 [4] provides an overview for attribute types and their purposes.

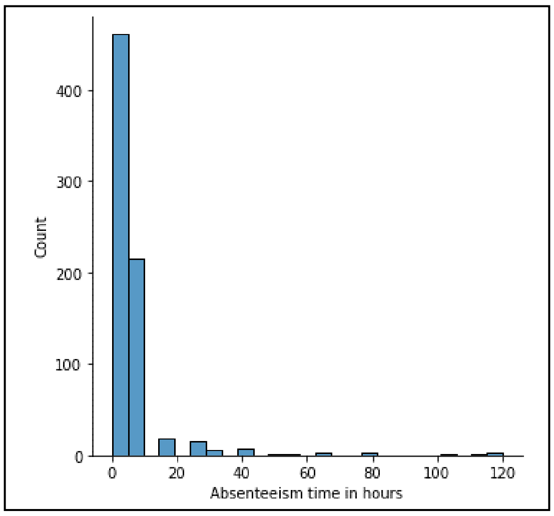
TABLE 1

Attribute	Data Type	Description
ID	Integer	Employee ID
Reason	Integer	1-28 range based on reason list
Month	Integer	1-12 range based on which month
Weekday	Integer	1-7 range based on day of week
Season	Integer	1-4 range based on season
Travel_Cost	Integer	Cost of transport
Distance	Integer	Distance to work
Service_Time	Integer	Months of service
Age	Integer	Age of employee
Workload	Integer	Avg workload/day
Work_Target	Integer	Target for employee
Disciplinary	Boolean	1 = Past failure, else = 0
Education	Integer	1-4 Range based on level
Children	Integer	Number of children
Drinker	Boolean	1 = social drinker, else = 0
Smoker	Boolean	1 = social smoker, else = 0
Pet	Integer	Number of pets
Weight	Integer	Weight (in kg)
Height	Integer	Height (in cm)
BMI	Integer	BMI
Absence_Time	Integer	Time in hours

Employee ID is not useful for analysis and so I have removed it. Several attributes are either Boolean or integer types which are not suitable; for example, reasons for absence may be assigned inappropriate weighting by algorithms ‘Day = 1’ (Monday) should have no statistical difference to ‘Day = 6’.

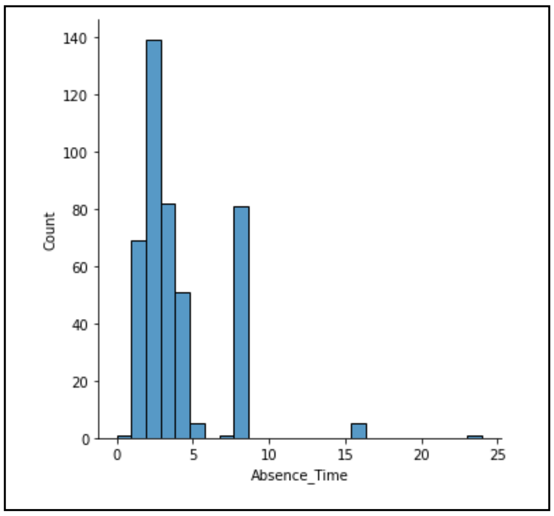
Month of absence, day of the week, education and seasons have been converted into categorical types with category names used from the UCI data description; names are self-explanatory and can be seen in code appendices.
The reason for absence attribute is particularly interesting to discuss. An initial frequency distribution can be seen below in figure 1.

Figure 1 (Full dataset)



As seen, it is clear most of the absence occurs in the 0-20 hours range, the mean of this data is 6.9, and the median is 3.0. This vast range created by the outliers on the upper end can be easily explained via the 'reason for absence' attribute. As an overview there are two main categories of reasoning; reasons attested by the International Code of Diseases (ICD) and those outside the ICD. If you filter the data to exclude entries covered by ICD the following result is produced (Figure 2).

Figure 2 (Reduced Dataset)



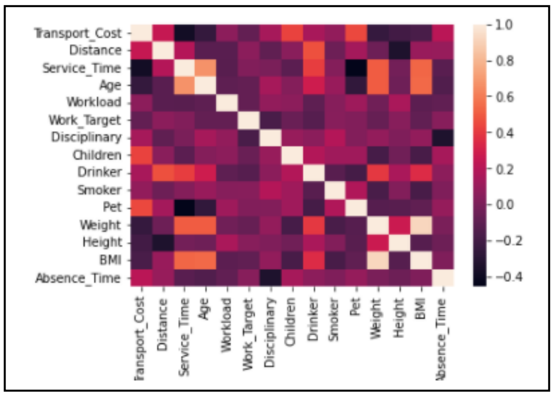
Here we retain most instances, and the range of the data has drastically reduced. The dataset drops to 478 instances from 740. The mean dropping down to 3.1 and median to 2; indicating more symmetry in our data distribution than figure 1.

As the purpose of my study is to explain reasons for absenteeism to HR management as for them to adjust workplace practice, I do not feel it appropriate to include ICD records; it should now be obvious to modern HR in any organization that long term medical conditions will have drastic impacts on absenteeism, and many examples illustrate this relationship e.g. [5]. This essentially makes these attribute dimensions noise which may hide some of the lesser-known nuance behind absenteeism. Their removal will potentially reduce the bias created through the limited number of observations seen in these subsets of the reason category. Remaining subsets are now amended to be categorical instead of numeric. As such experiments will be run with and without the reason filtering.

This does however make a small dataset even smaller and thus it will not be appropriate to use any deep learning techniques for analysis.

Pairwise correlations (see appendix) and a heatmap (figure 3) illustrate our data distribution. Immediately it appears that past disciplinary action, age, and service time have the greatest influence on future absenteeism.

Figure 3 (Heatmap for reduced dataset)



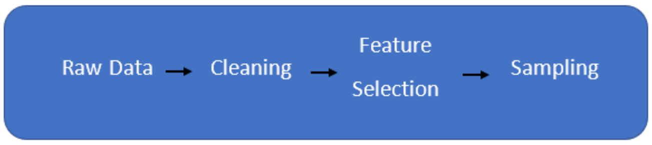
Additionally, to facilitate classification the continuous Absence_Time variable has been split into three categories for both experiment datasets. For the full dataset the categories are as follows: 'Low' = 0, 'Medium' = 1-15, 'High' = 16-120. For the reduced dataset: 'Low' = 0, 'Medium' = 1-4, 'High' = 5-24. Bin boundaries created using proportional min/max values for each dataset.

To tackle the imbalance created through this categorization a random oversampling technique was used. Splitting datasets into high/mid/low datasets and oversampling the high and low datasets to match the number of entries to the mid dataset.

III. Method

A simple machine learning pipeline method is used for this task, the below pipeline will guide you through the process.

Stage 1: pre-processing:

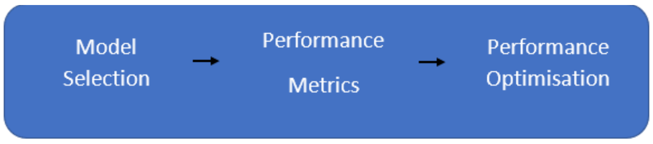


The initial step of this stage requires collecting the data from the repository and loading using an appropriate reading tool. Cleaning is largely covered during the preliminary analysis/pre-processing section of this report. It involved amending attribute names, standardisation and one-hot encoding to categorise continuous attributes with multiple (3+) categories as well as an oversampling technique.

Feature selection here involved observation of data distributions seen in the heatmap/frequency tables to interpret the weighting/noise of attributes. My selections here (to split data into experiments 1 & 2) was largely domain-based knowledge/curiosity.

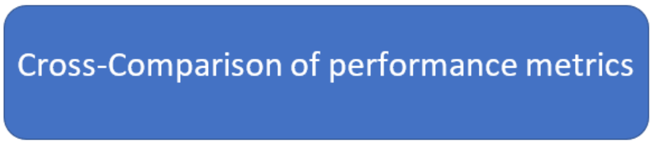
The dataset is finally sampled to produce training, testing and validation sets. There are one of each type of dataset for both experiment types. Training sets are used to train models, testing sets are used to gain performance metrics used in cross-comparison across models. Validation sets are used to split the class attribute (Absence_Time) from the rest of the dataset as to not interfere with model prediction.

Stage 2: experimentation:



This stage involves selecting models; in our case: decision tree, SVM and KNN. Each model is evaluated using testing data sets and then placed through iterative processing involving dimension changes to optimise each specific model. For example, the decision tree classifier had pre-pruning iterations (changing number of nodes) and post-pruning adjustment using cost complexity pruning. Prediction performance is tested using validation datasets and the optimised models have their optimised results recorded in the results table seen later in the reflections section.

Stage 3: evaluation:



The optimised algorithms have each of their performance metrics compared against the optimised results of each other classifier. The best model can then be used for final prediction.

Stage 4: Prediction:

This stage would be available to use for HRM management teams to predict the levels of absenteeism given current staff. This could lead to a data initiative which could be used to convince leadership to offer different working options such as WFH.

IV. Experiments

As mentioned in the preliminary data discussion two experiments will be run, a full dataset (A) and a reduced dataset, removing ICD diagnosis, (B).

Training, testing and validation datasets were used to complete experiments. Experiments were ran using decision tree, support vector machines and K nearest neighbor algorithms. StandardScaler() standardization was applied prior to KNN and SVM experiments.

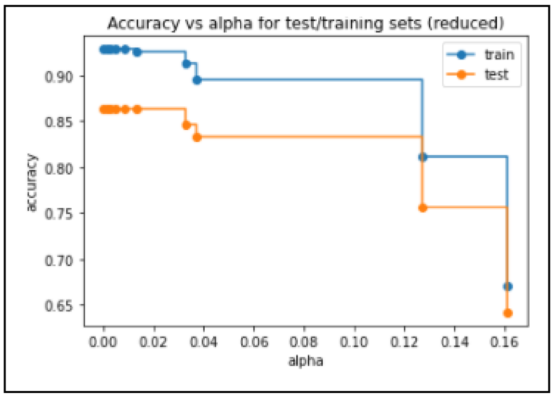
A – Decision Tree

The first classifier used was decision tree. This method aims to create a series of questions which can split the dataset in a way which creates the highest amount of 'pure' final leaves (essentially highest accuracy classification). Each attribute is tested with splitting and then optimal is used for the root (first question) and nodes (following questions). [6]

The method was optimized using pre- techniques (tuning max_depth (number of nodes) and min_sample_leaf (number of samples at a destination post question, to prevent overfitting). A post-pruning method of 'cost complexity pruning' was then applied. Cost complexity pruning calculates the sum of squared residuals and multiplying that value by a penalty value (alpha) for the size of the tree. [7]

Below figure 3 illustrates how a set of alpha values are tested and then the optimal accuracy score can be used to prune the tree, following a repetition of this task and using a mean found through cross-validation. The alpha values are quite small for this decision tree, however marginal increases in model performance were still achieved.

Figure 3 (Alpha for reduced experiment)



B - SVM

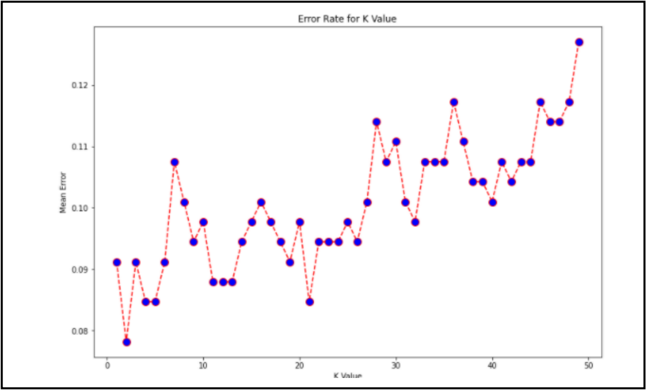
The second tests were ran using support vector machines. Support vector machines work by transforming data which doesn't have obvious splits for classification into a higher dimensional plane as to easier split the data [8], for example Age may be plotted against Age squared to better observe where classes congregate. Optimization is found by altering the kernels which change the method of transformation used on the dataset. Our 19-dimensional data is transformed into a higher dimension to better observe data splits. Using a polynomial kernel achieved the optimal results for the datasets.

C – KNN

KNN is a simple classifier which simply memorizes the training set and during prediction it assigns class based on the 'nearest' memorized values in the training set. Fine tuning is achieved by altering the 'k' metric, which is the number of nearby values used when calculating class (a majority rule is applied to assign class).

A loop was running to test different K values for both sets. Optimal K was found to be 1 but using this value would carry bias toward the training data and so the next best value was used which was 3. Across our data we can see that the more neighbors used the worse prediction becomes. This also makes it quite easy to understand why uniform was the lesser performing weighting as opposed to distance weighting (nearest values are just more important).

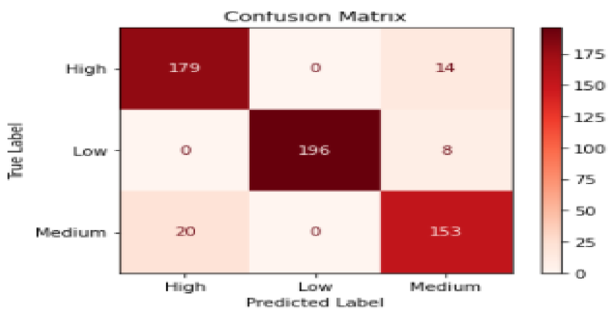
Figure 4: Full set K-value testing



Experiment	Precision	Recall	F1	AUC	Accuracy
Full (DT)	0.773	0.770	0.771	0.911	0.777
Full (SVM)	0.925	0.924	0.924	0.986	0.926
Full (KNN)	0.962	0.954	0.955	0.967	0.958
Red(DT)	0.882	0.867	0.867	0.929	0.863
Red(SVM)	0.906	0.898	0.899	0.971	0.896
Red(KNN)	0.927	0.923	0.923	0.947	0.922

Overall, the reduced dataset saw a higher mean score, however; this can be attributed to the weak performance of the decision tree on the full dataset. This may be due to the decision trees greedy nature in attempting to use multiple reasons for absence in classification and once those options are taken away it improves. Nonetheless KNN performed well on both datasets. Due to the small size of the dataset KNN is given a chance to shine. However, KNN may also be performing well here due to the bias/variance trade off playing in its favor [9]; exacerbated by the overfitting method I used. For this reason, SVM's comparably high results make it the optimal choice for classification.

My prior expectations of the full model being a weaker method for prediction seem unwarranted. These were largely dealt with during oversampling. It is understandable how KNN performed better with a wider range of neighbors to use. SVM for both models also had notably high performance. Particularly the AUC score; indicating almost no false positive/negatives. Potentially highlighting SVMs strength in utilizing the wide range of attributes. You can find the finalized confusion matrix for SVM prediction here:



Reflections

Overall, I have been able to produce a strong optimal prediction algorithm for HR management teams to use (SVM). My pipeline could be slightly optimized with the use of standardization at the initial preprocessing stage. Had I better understood oversampling I would not have had to run dual tests throughout the experiment stage and focused on optimal feature selection instead.

I also could've spent more time evaluating the different performance metrics as I did not know how to weigh up the different results for each metric. Generally, the optimal classifiers had strong results across the board with only marginal differences so this was not much of an issue, but it could present a problem given a different dataset.

To improve performance in the future I could look to use 10-fold cross validation which would help me catch out bias issues like that which I suspect from the KNN algorithm. There was also quite a lot of oversampling necessary to balance the high and low classes, I could evaluate different oversampling methods or test with different categorizations of absence_time in the future.

References

[1] BBC News (2021) *Facebook remote working plan extended to all staff for long term.* Available at: https://www.bbc.co.uk/news/technology-57425636 . (Accessed: 09 /03/2022.)

[2] Kocakulah, M. C., Kelley, A. G., Mitchell, K. M. & Ruggieri, M. P. (2016) "Absenteeism Problems And Costs: Causes, Effects And Cures", *International Business & Economics Research Journal (IBER)*, 15(3), pp. 89–96. doi: 10.19030/iber.v15i3.9673.

[3] Martiniano, A., Ferreira, P, R. & Sassi, J, R., 2018. *UCI Machine Learning Repository*, Available at: https://archive.ics.uci,edu/ml.

[4] Skorikov, M., Hussain, A, M., Khan, R, M. & Akbar, K, M. (2020) 'Prediction of Absenteeism at work using Data Mining Techniques'. *5th International Conference on Information Technology Research (ICITR).* doi: 10.1109/ICITR51448.2020.9310913.

[5] Zhang, W., McLeod, C. & Koehoorn, M. (2016) 'The relationship between chronic conditions and absenteeism and associated costs in Canada.' *Scandinavian Journal of Work, Environment & Health.* 42(5), pp. 413-422.

[6] Starmer, J. (2021) *'Decision and Classification Trees, Clearly Explained!!!'.* Available at: https://www.youtube.com/watch?v=_L39rN6gz7Y. . (Accessed: 12/03/2022.)

[7] Starmer, J. (2019) *'How to Prune Regression Trees, Clearly Explained!!!'.* Available at: https://www.youtube.com/watch?v=D0efHEJsfHo&t=610s&ab_channel=StatQuestwithJoshStarmer. .(Accessed: 12/03/2022.)

[8] Starmer, J. (2019) *'Support Vector Machines Part 1 (of 3): Main Ideas!!!'* Available at: https://www.youtube.com/watch?v=efR1C6CvhmE&ab_channel=StatQuestwithJoshStarmer. (Accessed: 12/03/2022.)

[9] Hastie, T., Tibshirani, R. & Friedman, J. (2009) *'The elements of statistical learning'.* 2nd edition. Springer Series

Task 2: Deep Learning for image classification

I. Introduction

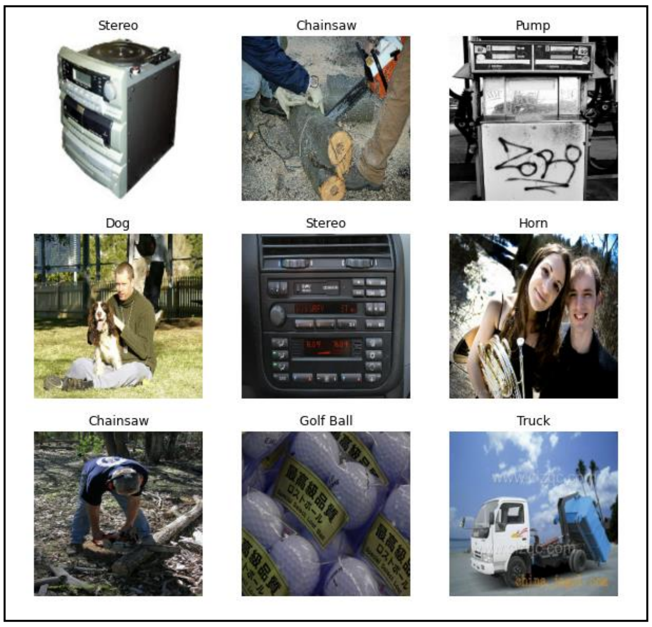
This task was completed using a provided image dataset. Images belong to distinct classes and the overall aim of this task will be to identify an object and correctly classify it.

This will be performed using a deep learning algorithm; convolutional neural network (CNN). This works by assigning importance, known as weights/biases, to various aspects and objects in images to differentiate them from one another. [1] A CNN will take a filter (subset of pixels of an image) and create a feature map to cross-compare different images, making it able to recognize small differences in objects without a need for exact matches, making it a very useful classifier for this task.

I have produced a simple CNN model and optimized it achieving a result of X% accuracy when predicting object class.

The dataset used is made up of a training and validation set. 10 folders are present in each set containing the following list of 10 different objects: Fish, Dogs, Stereos, Chainsaws, Churches, Horns, Trucks, Petrol Pumps, Golf Balls and Parachutes. An example of images available can be seen below in figure 1.

Figure 1: Image Subset



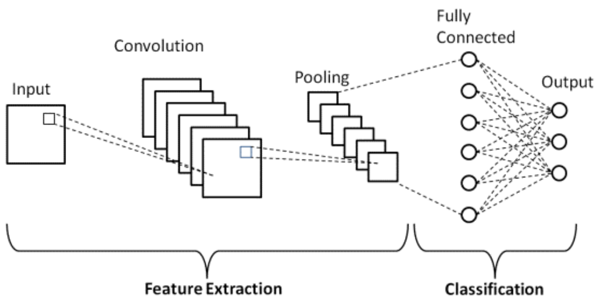
II. Preliminary Analysis and preprocessing

The dataset consisted of 13,404 data entries and has already been split into testing and validation sets: 70% testing and 30% validation respectively. The preprocessing necessary was to provide standardized image sizes to be used during processing of the CNN to utilize. This was achieved by rescaling all images by diving the pixel values by 252.

III. Method

Workflow followed a standard machine learning approach of understanding the data, building an input pipeline (both of which covered during prelim), building the model, training the model, testing the model and improving the model.

The finalized model was created using TensorFlow tutorial. A pre-existing 'Sequential' model [6] was used for the task. A basic illustration of the process can be seen below.



The input stage is simply the input images which are then used to build the convolutional layer. Within the convolutional layer a filter (kernel) is created and applied. This is essentially a subset of pixels from the input image; it is then slid across each image to test pixel matches. Through Boolean representation of pixel matching a 'feature map' is created. Generally, smaller size kernels are preferred as they can identify specific features of an image easier as well as require a smaller computational load to work; a 3x3 filter is used in this case. [3]

Padding is an additional option to improve accuracy; padding adds a 1-pixel extension to images during filtering. This allows for the filter to iterate over pixels in the corner multiple times as opposed to just once with no padding. [4] Padding = same was used in this case which allows the output map to retain the same size as the original image, maximizing information gain.

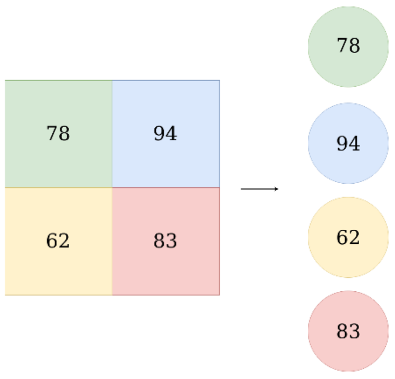
Another step during the convolutional layer is the activation function used throughout; I opted for ReLu. It is a simple activation function with the formula:

$$\mathbf{y = max\ (0, x)}.$$

This simplicity allows for fast computation, highly beneficial for our large dataset and multiple hidden layers; it has been the optimal choice for activation for several years. SoftMax was used for the output node as it can provide probabilities for each class. [2]

During the pooling layer the aim is to reduce the size of the convolutional layer whilst retaining maximum information to optimize computing costs. In this method Max Pooling has been used. This method looks at blocks of the image and aims to pick out most notable features; this is beneficial as it reduced computational load and can help prevent overfitting. This may prove useful for this dataset as images for each class vary widely (‘golf ball’ may show the object or somebody playing golf).

The fully connected layer begins classification. A flattening process must be applied to take the result from the pooling and convolutional layers (a matrix) and transform those values into a vector. Illustrated below. Flattening allows every output image to be connected to every node in the fully connected layer. The fully connected layer will not only classify images but learn to associate features with a particular class; this allows the layer to computer weight/bias values to process future images. The final connected layer will then use a SoftMax activation function to classify the image.

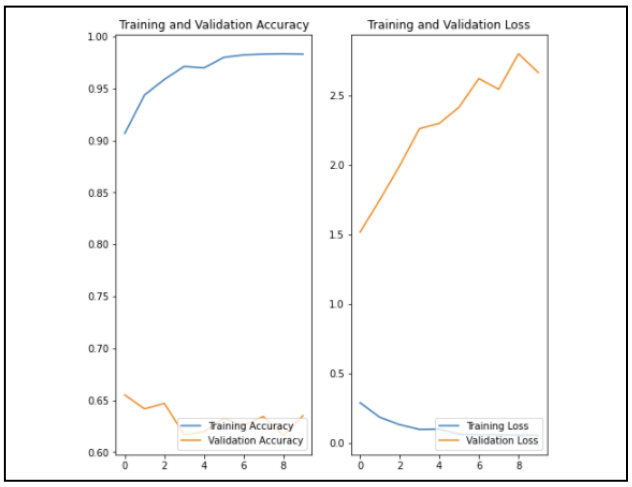


IV. Experiments

As previously mentioned, this model was taken from the TensorFlow tutorial website. No tuning measures were applied.

‘Batch-Size’ pertains to the number of images processed through the network at one time. After each batch the model parameters will be updated. This hyperparameter allows for the dataset to be processed without overloading computation. Batch sizes are optimized through trial and error but for our premade model 16, 32, 64, 128 are common sizes used.

Figure 9: Loss and Accuracy



Epochs are the number of iterations the full dataset is processed through the network; for the sake of time 10 epochs were used in my testing, but greater epochs will result in greater results.

Figure 9 illustrates a plot of the accuracy and loss presented during the experiment. Accuracy represents the number of images being correctly classified and loss represents how well the model is performing after each iteration of optimizing weights/biases. Low loss is better, high accuracy is better.

Observing figure 9 we can see that the training set has high accuracy and low loss, the validation set is the complete opposite, this indicates a high level of overfitting throughout the dataset.

V. Reflections

As I didn’t have time to evaluate further models, I will discuss options for improvement across the board here.

If overfitting was present, I would have the option to run data augmentation, one method of doing this is to create duplicates of images but having them rotated to improve generalizations the model makes. These would be included within the existing layer framework. Another technique could be ‘dropout’ this randomly eliminates nodes from the network; meaning that output classification can’t be overfit to any specific feature it has over relied upon.

Other aspects to test could include trialing a different pooling method such as average pooling or min pooling. Average pooling particularly looks at the average features of images rather than sharp specifics; this may prove useful for our dataset which has many vague images within each class.

VI. Refrences

[1] Saha, S. (2018) ‘A ocmprehensive guide to convolutional neural networks – the ELI5 way’. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 . (Accessed: 15/03/2022).

[2] Brownlee, J. (2021) *‘How to choose an activation function for Deep Learning’*. Available at: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (Accessed: 15/03/2022).

[3] Sahoo, S. (2018) *‘Deciding optimal kernel size for CNN.’* Available at: https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363 (Accessed: 15/03/2022).

[4] Verma, Y. (2021) ‘*Guide to different methods of padding for CNN models.*’ Available at: https://analyticsindiamag.com/guide-to-different-padding-methods-for-cnn-models/ (Accessed: 15/03/2022.)

[5] Brownlee, J. (2019) *‘A Gentle Introduction to Pooling Layers in Convolutional Neural Networks’*. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/ (Accessed: 16/03/2022).

[6] TensorFlow (2020) ‘Image Classification’. Available at: https://www.tensorflow.org/tutorials/images/classification (Accessed: 14/03/2022)

Task 1 Code Link
https://colab.research.google.com/drive/17K0jzD4pywBQVuPbcez2dEnOjvinHq99?usp=sharing

Task 2 Code Link
https://colab.research.google.com/drive/1jFAG7jkLT56S3FQLJWSt6dxH6dAvMNfg?usp=sharing