Applied Machine Learning Coursework

I. Introduction

The main purpose of this report is to present the approach taken when solving a machine learning task as well as a deep learning task. The first task involved predicting whether a driver would accept a coupon based on certain conditions such as the destination they are going to, the current weather and temperature as well as their occupation and education, among other features [1]. The main application area of this machine learning task is the business area. It is important for a business to know the optimum conditions necessary for a potential customer to accept a coupon and use it at their business outlet. If the business is able to predict the time and weather in which they will have maximum success at handing out coupons, then they can focus more of their efforts at that specific time and weather. If the business is also able to predict whether a customer will accept a coupon based on their characteristics, then the business will put more focus on their target customers with the most favourable characteristics. Ultimately, the main goal of the business is to increase sales revenue hence the importance of machine learning in this situation. Since the main aim is to predict whether a driver would accept a coupon, then the machine learning category this falls under is classification. The machine learning algorithms used in this task were K-Nearest Neighbours (KNN), Support Vector Machines (SVM), Decision Tree Classifiers and Adaptive boosting (AdaBoost) classifier. It was found that the SVM classifier model was the most effective predictor, having the highest accuracy at 75%. The KNN classifier, however, performed almost equally as good as the SVM, with an accuracy of 73%. The AdaBoost then followed with an accuracy of 68% and the Decision Tree classifier performed poorest with an accuracy of 63%.

The second machine learning task involved classifying images into a set of pre-defined classes. The deep learning algorithm used to classify the images was a convolutional neural network (CNN). The classifier achieved low accuracies of about 10% to 20% on test data, and the most likely reason for the low accuracy was the dataset not being large enough.

II. Task 1

A. Data and Preliminary Analysis

The first task involved the use of an in-vehicle coupon recommendation data set from UCI machine learning repository [2]. The data set was uploaded as a comma separated value (CSV) file onto the repository on 15th September 2020 and has 12684 instances. The dataset was then analyzed using Pandas library [3] in the Python programming language [4]. The dataset had 26 features and 12684 rows. The features included driver characteristics, such as gender, age, marital status, number of children the driver has, occupation, education, income and whether the driver was alone in the car or with friends. Other features also included were the driver's destination, the weather, time of day, the coupon type, coupon expiration and whether the bar or restaurant for which the coupon is being used is in the opposite direction or the same direction the driver is headed to. Drivers were also asked questions relating to how many times they went to a bar, coffeehouse and restaurant in a month and these were included as features. The average amount of money spent by a driver in a restaurant, the car type as well as how often the driver orders take-away food were also features of the dataset.

The dataset contained missing values as shown in *Fig 1*.

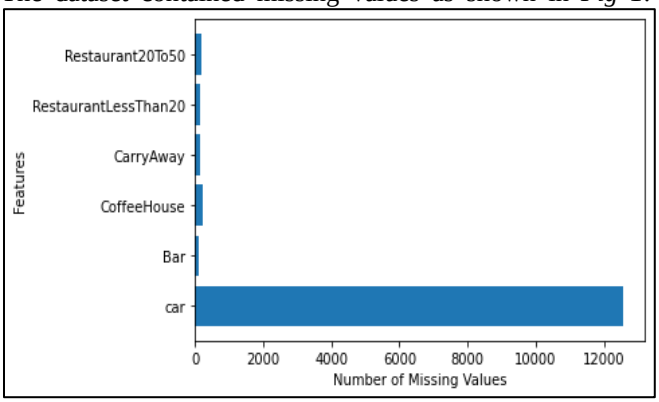

Fig 1: Missing Values.

It was noted that the car feature had 12576 missing values, yet the dataset only has 12684 instances. This was the highest number of missing values compared to other features. The rest of the features in the dataset not shown in *Fig 1* had no missing values.

It was worth noting that most of the features contained categorical values, regardless of whether the data type was a string or an integer. For example, the temperature column had only three values which were 80, 30 and 55. The time column had 5 values, which were 2PM, 7AM, 10AM, 10PM and 6PM. After displaying the summary statistics of the dataset, it was noted that one feature, toCoupon_GEQ5min, had all instances of its values as 1.

Based on the observations of the dataset, it can be concluded that before creating a machine learning model, the data should be one-hot encoded during pre-processing as all of the data is categorical data. Since the target feature, 'Y', has only 0 and 1 as its values, it can be concluded that the classification task is binary.

B. Methods

In order to solve the task, a machine learning pipeline as shown in the machine learning flowchart in *Fig 2* was adopted. The first step was to load the data into a data frame. Next the data was cleaned and missing values were removed. Since the 'car' feature in the dataset had too many missing values as shown in *Fig 1*, the whole feature was dropped from the dataset as it would not add much value to the machine learning models. It was then decided that remaining rows which contained missing values should be removed. This is

because the features with missing values already contained a unique category called ‘never’ which implied that these values were equivalent to zero. If a zero was used to replace the missing values, then they would clash with the intended meaning of the data collected. Therefore, it was necessary to remove the rows containing missing numbers.

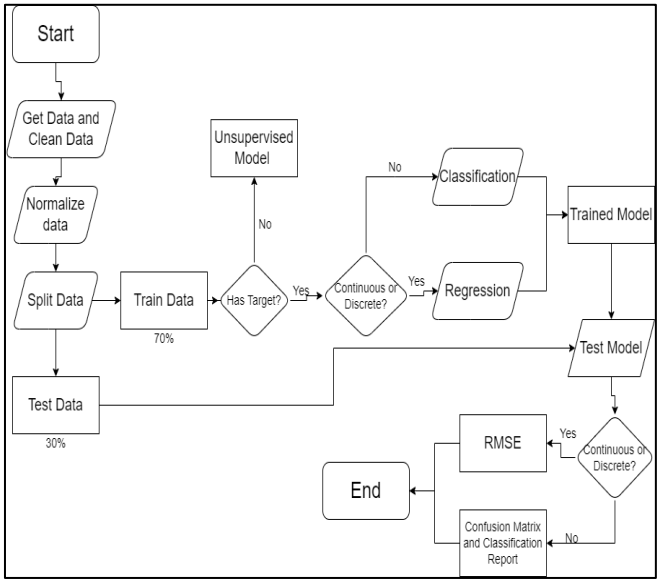

Fig 2: Machine Learning Pipeline.

After ensuring that the data had no more missing values, the last step in cleaning the data was dropping the ‘toCoupon-GEQ5min’ feature which had only ‘1’ as the data values. It was then necessary to move on to data pre-processing and normalization.

Since all of the features contained categorical values, the best method of data pre-processing was one-hot encoding. Features that were of string type, as well as temperature, were one hot encoded so that the resulting dataset contained only 1s and 0s. One hot encoding assigns binary variables to each category of each feature. In this case, one-hot encoding was preferred to label encoding since the latter may result in poor performance of the final model. This is because it is highly likely that the model will detect an erroneous relationship between features, for example, that sunny weather is less than windy weather [5].

After pre-processing the data so that it is fully numerical, it was then necessary to separate the target variable from the rest of the features. The features were then split into a training set and a testing set, where the training set was 70% of the data and the testing set was 30%. This was done using the train_test_split() command from the scikit-learn machine learning library in Python [6]. In order to get reproducible results, the random state was set to 42.

After splitting the data into training and testing sets, it was then necessary to choose the machine learning models that would be used. A KNN classifier, a SVM classifier, a Decision Tree classifier and an AdaBoost classifier were chosen as the machine learning models to be tested. The data would be trained based on these classification models, and the results from training the data would be compared against each model so that the best model would be identified. In the KNN classifier, the hyperparameter that was varied so that the best model would be identified is the number of neighbours. The evaluation metric used in deciding the best value for the number of neighbours was the accuracy score. In the SVM classifier, the hyperparameter varied was the kernel type, and the accuracy score would also be used to select the best model. Similarly, in the Decision Tree classifier, the best model would be chosen from varying the maximum number of depths hyperparameter. Lastly, the AdaBoost Classifier would also be included as a model to be tested, using a Logistic Regression classifier as the base estimator.

C. Experiments

The first model tested was the KNN classifier. The training data was used to train the model and the testing data was used to test the accuracy of the model. 80 nearest neighbour values were tested in order to find the optimum nearest neighbour value, as shown in *Fig 3*.

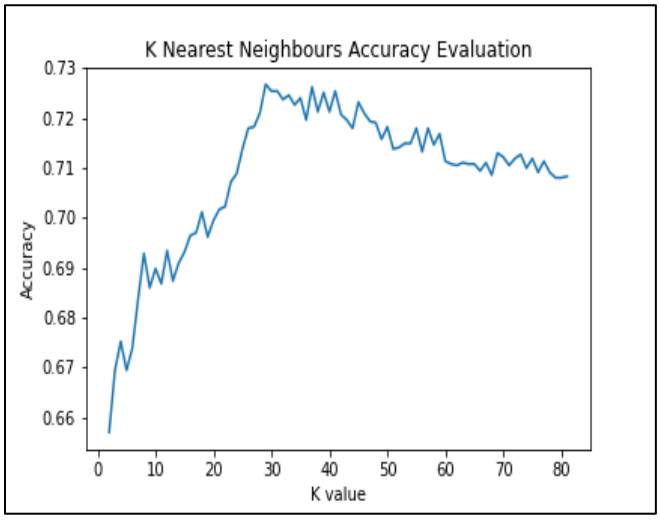

Fig 3: KNN Accuracy vs Nearest Neighbours.

It is evident from *Fig 3* that the optimum number of nearest neighbours was 29, leading to a model accuracy of 73%. The confusion matrix for the final model is as shown in *Table 1*.

	TRUE(1)	**FALSE(0)**
TRUE(1)	898	645
FALSE(0)	345	1736

Table 1: KNN Confusion Matrix

The Receiver Operarating Characteristic (ROC) curve of the model is shown in *Fig 4*. The curve shows that the KNN classifier is better than a random classifier, but less effective compared to a perfect classifier. The Area Under Curve (AUC) is 0.78.

For the SVM classifier, the model was trained and tested using the ‘linear’, ‘rbf’ and ‘poly’ kernels. The accuracies of each of the three kernels were compared and the ‘rbf’ kernel

was identified as the kernel with the highest accuracy at 75%, as shown in *Fig 5*.

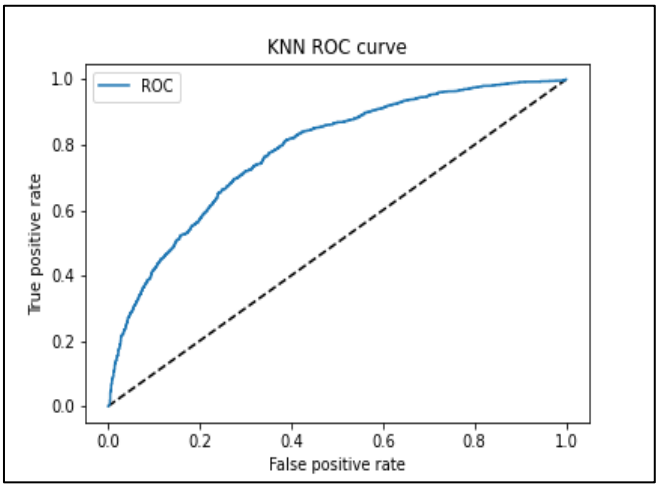

Fig 4: KNN ROC Curve

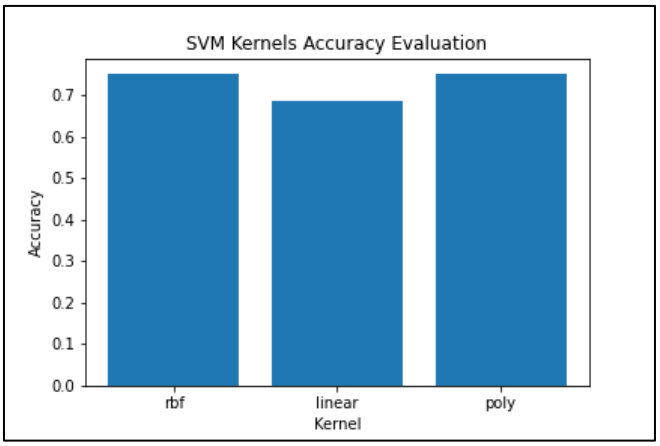

Fig 5: SVM Kernel Accuracy.

The confusion matrix for the final SVM model is as shown in *Table 2*.

	TRUE(1)	**FALSE(0)**
TRUE(1)	1021	522
FALSE(0)	374	1707

Table 2: SVM Confusion Matrix

The ROC curve of the model is as shown in *Fig 6*. The AUC of the model is 0.82, which is better than the KNN's AUC.

The Decision Tree Classifier model was fed with the training and testing data. Different values of maximum depth were used and the accuracies were obtained as shown in *Fig 7*. It is evident that a maximum depth value of 6 gave the highest accuracy of 63%. The confusion matrix is as shown in *Table 3*.

	TRUE(1)	**FALSE(0)**
TRUE(1)	478	1065
FALSE(0)	280	1801

Table 3: Decision Tree Classifier Confusion Matrix

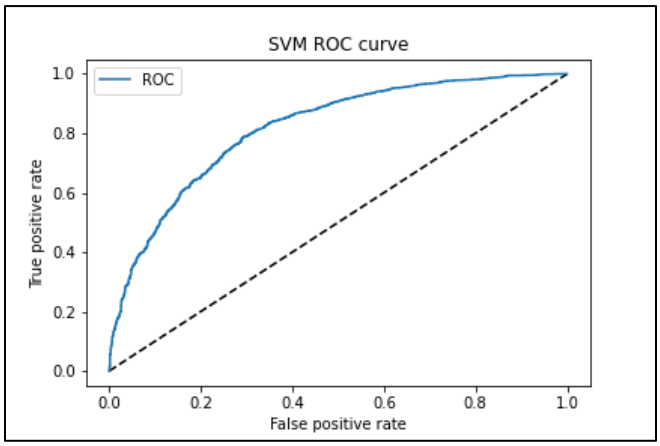

Fig 6: SVM ROC Curve

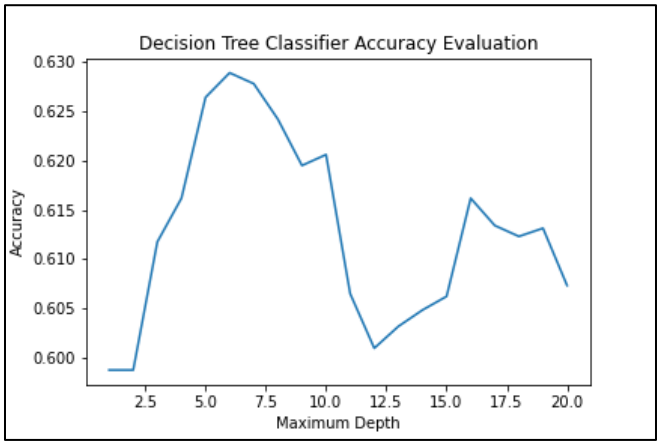

Fig 7: Decision Tree Maximum Depth Accuracy.

The ROC curve of the Decision Tree Classifier model is as shown in *Fig 8*. The AUC score is 0.63.

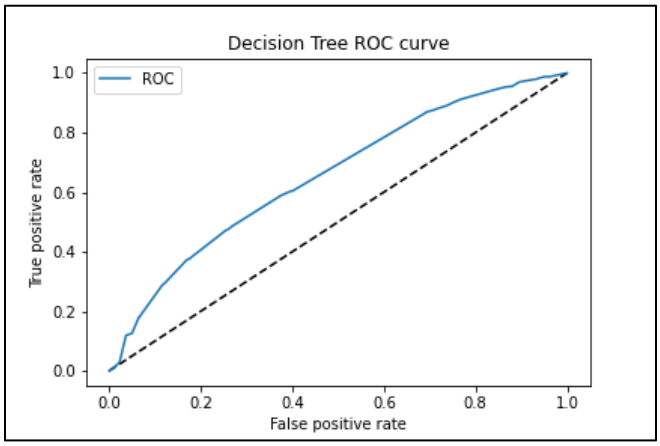

Fig 8: Decision Tree ROC Curve.

Lastly, the AdaBoost classifier model was trained and tested using a Logistic Regression classifier as the base estimator and the resultant accuracy was 68%. The ROC curve is as shown in *Fig 9*. The AUC score was 0.74. The confusion matrix for the model is as shown in *Table 4*.

	TRUE(1)	FALSE(0)
TRUE(1)	869	674
FALSE(0)	487	1594

Table 4: AdaBoost Confusion matrix

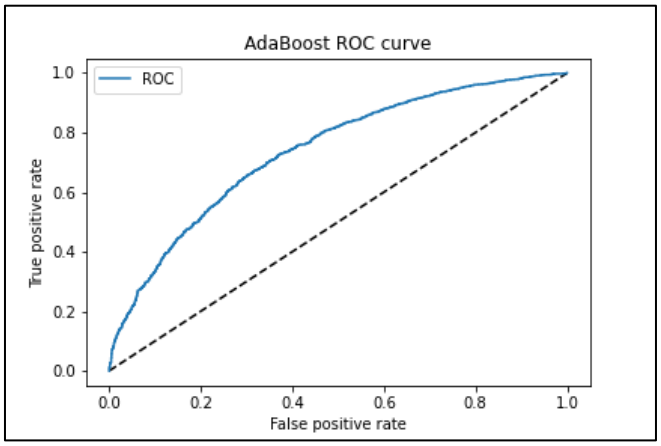

Fig 9: AdaBoost Classifier ROC curve.

The classification report for the four models is as shown in *Table 5*.

Classifier	F1 (macro avg)	Recall	Precision	Accuracy	CV
KNN	0.71	0.71	0.73	0.73	0.64
SVM	0.74	0.74	0.75	0.75	0.69
Decision Tree	0.57	0.59	0.63	0.63	0.59
AdaBoost	0.67	0.66	0.67	0.68	0.67

Table 5: Evaluation Metrics Scores.

It is evident from *Table 5* that the best classifier is the SVM classifier. This is because it has the highest scores in all categories. As evident in the confusion matrix in *Table 2*, the SVM classifier was the best predictor of both true positives and true negatives. One would therefore conclude that the SVM is the most preferred model.

The poorest performing classifier was the Decision Tree classifier, as all its scores were the lowest in all categories. The KNN classifier performed second best, while the third best performance was the AdaBoost classifier. In order to determine the performance of models on the whole dataset, k-fold cross validation (CV) scores were calculated and are as shown in *Table 5*. The SVM had the best cross-validation score, followed by the AdaBoost classifier, then the KNN and finally the Decision Tree classifier had the worst cross-validation score. The number of folds that were used to determine the average CV scores were 5.

III. Task 2

A. Data and Preliminary Analysis

The second task involved the classification of images into their correct classes using a convolutional neural network. The dataset provided was composed of two folders, the first folder being the training data containing 9469 photos and the second folder being the validation data containing 3925 photos. Both datasets contain 10 classes of photos. The library used to load the photos and make a deep learning model is the PyTorch [7] library in Python.

B. Methods

In order to solve the task, an appropriate deep learning library had to be selected first. In this case, the library selected was PyTorch. A suitable transform that would resize and randomly flip the images was then chosen. Next, both training and validation sets were loaded into separate variables, with the batch size being 32 and with shuffle being turned on. It was important for the pictures to be shuffled so that the neural network model doesn't learn from the order of pictures. Both the training and validation data were then separated from their labels.

The deep learning algorithm that was used was the convolutional neural network. The algorithm would be defined [8], and the number of convolutional layers would be selected. The training and evaluation of the model would then follow, and a suitable number of epochs would be selected based on a learning curve that ensured both the training set and validation set had minimal losses. The process is as illustrated in *Fig 10*.

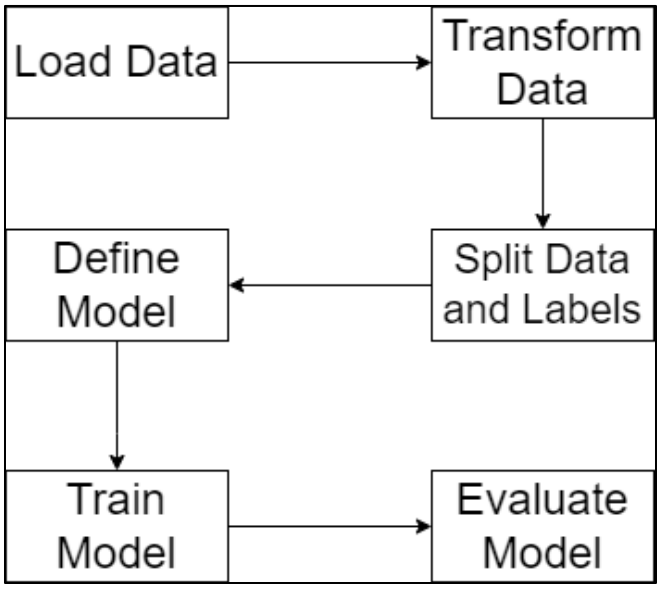

Fig 10: Deep Learning Task Methods

C. Experiments

The model tested was a convolutional neural network that had three layers. The first layer had 3 input and 12 outputs, the second layer had 12 inputs and 20 outputs while the last layer had 20 inputs and 32 outputs. All of the layers had a kernel size of 3, a stride size of 1 and a padding size of 1. Each layer was followed by a rectified linear unit (ReLU) and a maximum pooling layer.

The CNN model was trained and optimized during each epoch. Training losses as well as validation losses were calculated for each epoch and they were used to produce a learning curve as shown in *Fig 11*.

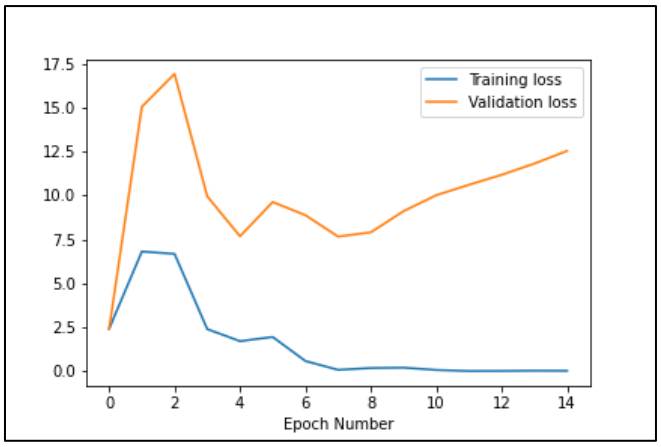

Fig 11: Learning Curve

The training and evaluation of the data was not reproducible, and different values of losses were recorded each time the model was trained and tested, as evidenced in the learning curve in *Fig 12* which differs from the learning curve in *Fig 11*.

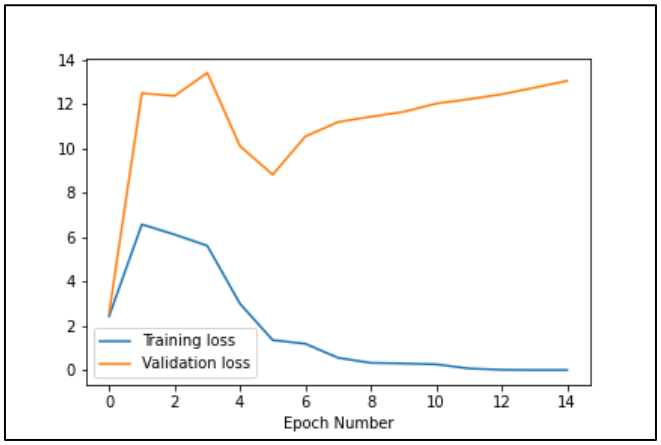

Fig 12: Learning Curve

From *Fig 11*, it is evident that the ideal number of epochs for both minimal training and validation losses would be 7. However, *Fig 12* it suggests that 5 epochs would be ideal for to minimize all losses. Since the losses are not the same each time the code is run, an epoch number of 6 was selected and the code was run. The average accuracy for the training data was 69% and the average accuracy of the validation data was 22%. However, these accuracies would vary as the results are sometimes irreproducible when the code was run.

IV. Reflections

A. Task 1

Out of the tested classifiers, the best classifier was found to be the SVM classifier, as it had the highest accuracy, F1, recall, precision, AUC and cross validation scores. The worst classifier was the Decision Tree classifier as it had the lowest accuracy and evaluation scores. The main good finding is the 75% accuracy score of the SVM. This is because if applied in a business context, it means 3 out of 4 of drivers approached would accept a coupon if the model is used for prediction. This is already a significant improvement from a random classifier with an accuracy of 50%. The SVM classifier would therefore be beneficial for the business seeking to distribute coupons.

The main challenge encountered when making the classifier model was that the SVM code took longer than the rest of the classifiers to run. Getting a k-fold cross validation score was therefore difficult as this took a long time to run when different folds were trained and tested. Another challenge encountered was the difficulty in surpassing the 75% accuracy mark. This, however, can be attributed to the nature of the dataset. More datapoints may be required for training so that the accuracy may be increased as a result of the model improving. In terms of the lengthy time taken to run the code, a computer with a higher processing speed may be useful to cut short the time needed to run the code.

B. Task 2

The use of 6 epochs was found to be the most effective in obtaining low values of validation losses. However, the accuracy for both training and validation data was low, at 69% and 22% respectively. This is largely as a result of a small amount of data. For higher accuracy, the model may need to be trained with hundreds of thousands of photos. Another challenge encountered is the use of TensorFlow library [9]. Images were loaded using TensorFlow and a model was created, but when time came to train the model, the code chunk took too long to run. Therefore, PyTorch was used instead of TensorFlow. Another challenge was when more CNN layers were used, the code computation became even slower. One way of solving this challenge is the use of compute unified device architecture (CUDA) which is a parallel computing platform [10]. PyTorch can detect whether the device uses CUDA and would run its algorithms with faster computing if this is the case.

V. References

[1] T. Wang and C. Rudin, “in-vehicle coupon recommendation dataset,” Sep. 15, 2020. https://archive.ics.uci.edu/ml/datasets/in-vehicle+coupon+recommendation (accessed Mar. 18, 2022).

[2] “UCI Machine Learning Repository.” https://archive.ics.uci.edu/ml/index.php (accessed Mar. 18, 2022).

[3] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed Mar. 18, 2022).

[4] “Welcome to Python.org.” https://www.python.org/ (accessed Mar. 18, 2022).

[5] “Categorical Encoding | One Hot Encoding vs Label Encoding.” https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/ (accessed Mar. 18, 2022).

[6] “scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation.” https://scikit-learn.org/stable/ (accessed Mar. 18, 2022).

[7] “PyTorch.” https://pytorch.org/ (accessed Mar. 18, 2022).

[8] "Convolutional Neural Network Pytorch | CNN Using Pytorch." https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/ (accessed Mar. 18, 2022).
[9] "Introduction to TensorFlow." https://www.tensorflow.org/learn (accessed Mar. 18, 2022).
[10] "CUDA Toolkit | NVIDIA Developer." https://developer.nvidia.com/cuda-toolkit (accessed Mar. 18, 2022).

VI. Appendices

A. Appendix A: Link to Task 1 Google Collaborate Code

https://colab.research.google.com/drive/1Y_tIQ5tXf6-Lr2ySGuEYep82A4Ue1G_L?usp=sharing

B. Appendix B: Link to Task 2 Google Collaborate Code

https://colab.research.google.com/drive/1QIhLsON-LddeGCRiTmlCHuOlZn0dKFOm?usp=sharing