Machine Learning Assignment

Task One

I. Introduction

A German bank has found that it frequently makes bad loans decisions, either by offering loans to unreliable debtors or withholding loans from reliable debtors. Since the bank's current system too often misevaluates potential debtors, they would like to improve their selection process with machine learning. The bank has provided a dataset of previous credit decisions so that it may be used for a supervised machine learning model that decides whether a loan application be granted.

Seven models were trained and optimised, Naïve Bayes proving best with an f2 score of 0.7.

II. The Data

I. The Dataset

The provided dataset contains information on previous debtors such as their loan purpose, credit history, job, age, etc. It has 1000 instances across 20 attributes. The target variable is the binary 'credit_risk', with a 0 for good credit and 1 for bad credit. A cost matrix also accompanies the dataset, stating that it "is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1)." The bank therefore incurs a greater loss by giving credit to a bad debtor than by withholding credit from a good debtor.

Preliminary exploration of the dataset revealed that the categorical variables take an Axxx format, where xxx are numbers representing various labels. The continuous variables are on different scales, and the target variable is heavily unbalanced. The distribution is 70% negative class (0/good credit) and 30% positive class (1/bad credit). There are no missing data or duplicate values, no obviously incorrect data entries, and the datatypes are formatted correctly.

Through a column transformer, the nominal attributes were one hot encoded, the ordinal features ordinal encoded, and the continuous variables standardised.

II. Data Exploration

Some exploratory analysis was undertaken to gain a deeper understanding of the dataset. Histograms revealed that the continuous variables are somewhat tail-heavy, although not to a necessarily problematic degree. Most of the categorical variables are evenly distributed, with the exception of 'foreign_worker', 'other_installment_plans', and 'other_debtors'.

Since fewer attributes reduce computational load and, in some cases, improve accuracy, all features were tested for importance. The continuous variables were ranked with the ExtraTreesClassifier, F-statistic, and mutual information. Of the three continuous features, 'age' ranked very low, while 'amount' and 'duration' ranked quite high. The categorical attributes were ranked according to their ANOVA F-value, chi-squared, and mutual information. The chi-squared and ANOVA results were very similar, with mutual information not too far off. Initial observation shows that 'status' and 'savings' are the most important categorical features.

III. Methods

The pipeline for this project is outlined in the following figure:

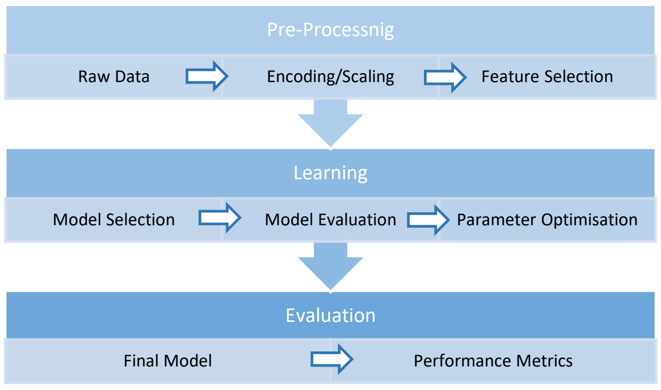


Figure 1: Methodology

A. Pre-Processing

As outlined in Fig 1, the pre-processing stage involves three main steps: acquisition and familiarisation of the data, encoding and scaling the data, and selecting the features for the model. Steps one and two were undertaken in *Part II: The Data,* as well as a provisional feature ranking. To determine the final set of features, different combinations of the top attributes from all rankings were tested in the model selection and evaluation stages.

Because the dataset is relatively small, instead of a train test split, k-fold cross validation was performed with k = 10. Due to the class imbalance in the dataset, each fold required stratification. Each cross validation was also repeated three times to prevent lucky outcomes.

B. Learning

The learning stage is about building the most optimal model for the job. An iterative process, it involved evaluating seven models against the desired performance metrics. For example, logistic regression is selected as a potential model, it is trained and tested with 10-fold cross validation, its parameters tuned, then cross validated again until the optimal model has been built.

Evaluation

The best models from the previous stage are now compared in order to choose the overall best model for the task. The final model's metrics are then evaluated.

IV. Experimental Results

A. Pre-Processing

With the data properly formatted, encoded, and scaled, a variety of attribute combinations were tested. The best performing attributes were 'status', 'credit_history', 'purpose', 'savings', 'employment_duration', 'personal_status_sex', ' other_debtors', 'property', 'other_installment_plans', 'housing', 'foreign_worker', 'age', 'amount', and 'duration'. However, despite being the performing combination, these attributes did not yield strong F scores.

B. Learning

The models tested were logistic regression, linear discrimination analysis, naïve Bayes, Gaussian process classification, support vector machine, random forest, and ridge classifier. A pipeline was built that begins by normalizing the continuous data with MinMaxScaler(), one hot encoding the nominal data with OneHotEncoder(), and encoding the ordinal data with OrdinalEncoder(), before cross validating on each of the seven classifiers. The best performing classifiers were then put through GridSearchCV in order to determine the optimal parameters. Overall, the best classifiers were naïve Bayes, random forest, and ridge classifier.

Recall that the cost matrix shows that classifying bad debtors as good is more damaging to the bank than classifying good debtors as bad. In other words, false negatives are worse than false positives. This must be taken into account when evaluating model performance. Since we want to identify the models that best minimize false negatives, the F2 score is a suitable metric. The F2 score places more emphasis on recall than on precision, and therefore on false negatives.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Due to the imbalanced distribution of the target variable (70/30), accuracy is not a suitable metric, since it "does not distinguish between the numbers of correctly classified examples of different classes" [1]. Because the emphasis is on reducing false negatives, the F1 score is not suitable either, as it gives equal weight to precision and recall. The F2 score is therefore the main metric used to evaluate the models for this task, as well as confusion matrixes and individual precision and recall scores.

Before experimenting with any models, a baseline result was determined by running the DummyClassifier on the dataset. The baseline F2 score was 0.68. The classifiers were compared against this baseline score; models with an F2 score higher than 0.68 are considered to have skill.

1) Naïve Bayes

Naïve Bayes is a simple algorithm that uses prior information to make probability based predictions. Given its simplicity, its hyperparameters cannot be tuned like with other machine learning algorithms, so it did not undergo parameter optimization.

2) Random Forest

Random forest uses an ensemble of individual decision trees. Using a technique called bagging, where each decision tree is built from different data samples, the random forest constructs an uncorrelated group of decision trees who then "vote" on the overall predictions.

The random forest was put through GridSearchCV with various possible parameters. The optimal value for max_features, the number of features an individual decision tree can try out, was 0.7. The optimal min_samples_split, the minimum samples necessary to split internal nodes, was found to be 6. The optimal number of trees required before voting, n_estimators, was 25. Max depth remained at the default of 'None'.

3) Ridge Classifier

The ridge classifier works by converting the target variables into -1's and +1s and using regression to predict the class. The optimal parameters after using GridSearchCV were as follows:
alpha=0.1
fit_intercept=False
solver='lsqr'

4) Comparison

Metric	Naïve Bayes	Random Forest	Ridge Classifier
Precision	0.49	0.61	0.65
Recall	0.72	0.48	0.46
F2	0.65	0.47	0.49

The results can be seen in the previous table. Although Naïve Bayes is the best performer, all of the models underperformed compared to the baseline F2 score of 0.68. The other two classifiers performed quite badly. Overall, none of the models have skill. This may be due to the severely skewed class distribution, the relatively small dataset of 1000 cases, and perhaps because there is a lot of noise around the decision boundary between the target variables.

5) *Undersampling*

In order to better deal with this unbalanced dataset, undersampling was performed. Undersampling works by reducing the size of the majority class while maintaining the size of the minority class. Undersampling may remove instances of the majority class around the decision boundary that confuse the classifiers. There are several approaches to this. For this task, the following methods were tested with the Naïve Bayes model:

- Tomek Links
- One Sided Selection
- Neighbourhood Cleaning Rule

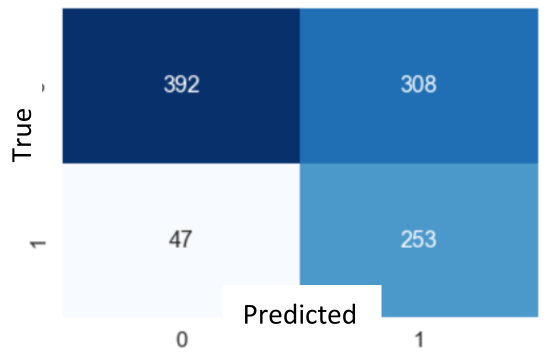
The results are as follows:

Tomek Links	One Sided Selection	Neighbourhood Cleaning Rule
0.67	0.67	0.7

Neighbourhood Cleaning Rule improved the F2 score of the Naïve Bayes model by 5%. With this undersampling technique, Naïve Bayes outperforms the baseline model.

C. Evaluation

The final model is the undersampled Naïve Bayes model. It achieved the highest F2 score of all he models.



As we can see from this confusion matrix, the model was quite successful with regards to minimising false negatives. Recall was very high at 0.89 (47 false negatives), while precision was relatively low at 0.56 (308 false positives).

Although the model as a whole did not perform amazingly, it was successful with regards to the main objective of limiting false negatives. The cost matrix gave false negatives a penalty of 5, while false positives only a penalty of 1. With the Naïve Bayes model, the bank will greatly limit the number of loans offered to bad debtors, protecting them from financial loss. However, while the model will save the bank from losing money, it will not help the bank make money by giving out loans to good debtors. If the bank wishes to take a conservative approach to their loan deals, then this model may very well be useful. Yet if they wish to take more risk, another model is needed, one that would reduce false positives at the cost of false negatives.

V. Reflection

For this task, a pipeline was built and tested for the purpose of helping a bank decide who to offer loans to. This involved formatting the raw data, selecting for important variables, training various classifiers, optimising the most successful classifiers, and implementing an undersampling technique on the best classifier. Comparing the models, Naïve Bayes was found to be the superior model. With a relatively good F2 score of 0.7, this model was suggested to the bank.

The large degree of imbalance in the dataset proved problematic, although this was mitigated slightly with the Neighbourhood Cleaning Rule. Whether the model is successful depends on the bank's goals. For instance, if the bank's only concern is reducing the number of loans offered to bad debtors (as might well be the case given the cost matrix), a model that could significantly reduce false negatives would be considered successful. Although not perfect, the final model offered here achieves this reasonably well. On the other hand, however, if the bank was also interested in increasing profits, then the model would not be considered successful because it classes many good creditors as bad creditors (i.e. too many false positives).

Perhaps a different combination of attributes would have reduced noise around the decision boundary. PCA was not attempted because it is generally not advised to perform PCA on discrete variables and one hot encoded categorical variables [2], but perhaps it was worth trying. An oversampling technique was tested on the Naïve Bayes model, but this improved the model only slightly, so it seems that undersampling was the correct choice.

Task Two

I. Introduction

We have been given a dataset comprising images of various objects, each object belonging to one of ten classes. The goal of this task is to build a deep learning model that can identify the objects in the images.

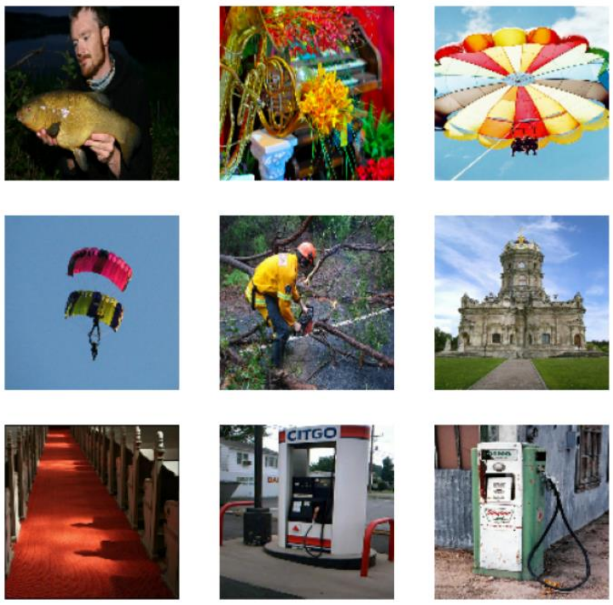
The kind of deep learning algorithm used for this task is known as a Convolutional Neural Network (CNN). Modelled on the human brain, deep learning is a form of machine learning that arranges algorithms in an "artificial neural network". CNNs are a neural network architecture that are particularly adept at image processing, making them ideal for this task.

A very simple, shallow CNN model has been built and adjusted to classify the objects in the given dataset. The model has an accuracy of ~73% with a loss of 0.89.

II. The Data

A. The Dataset

The dataset comes split into a training set and validation set. Each set contains ten folders, each folder containing images of a particular class. The ten kinds of objects are as follows: fish, dogs, stereos, chainsaws, churches, brass horns, garbage trucks, gas pumps, golf balls, and parachutes. In total, there are 13,394 images divided into 9,469 training images and 3,925 validation images. A random selection of photos can be viewed in the following Fig 2:

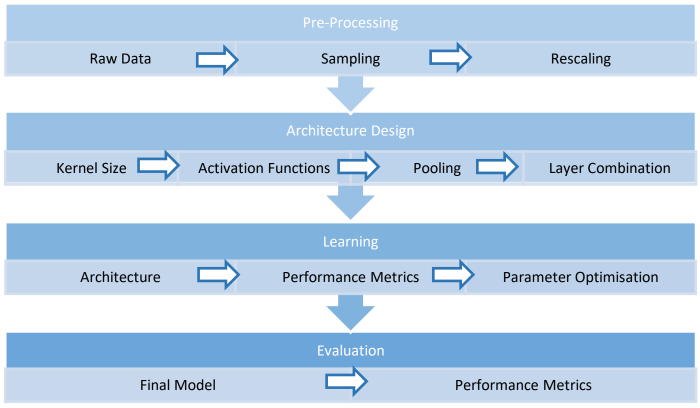


B. Preliminary Analysis

A random sample of nine images was extracted in order to get a sense of what the dataset contained. As can be seen by the parachutes and churches in Fig 2, the images for each class can be quite diverse. One image is of a church from the outside, while the other image is of inside the church; and while one parachute is a single, large parachute, the other is of two smaller ones. Additionally, many of the photos in the fish category include humans holding them, as can be seen in Fig 2. Such diversity in image content will cause problems for the CNN and likely inhibit its ability to classify the images well. This should be taken into account when deciding on the kernel size, since a smaller kernel will capture the detail when the photo's object is more obscure. For instance, the chainsaw in the middle photo is very small and obscure, so a small kernel would help identify that object. However, too small a filter and the model will fail to find larger objects like the two gas pumps.

III. Methods

The pipeline for this task is summarised in the following figure:



A. Pre-Processing

The raw data is already seperated into training and validation sets (70%/30%). As well as setting the image and batch sizes, the images need to be shuffled when imported from the directory so that they are randomized. CNNs use stoachastic gradient descent algorithms that allow them to adjust model weights based on the error gradient. Because batch size determines the error gradient, an appropriate batch size is important.

The images must be standardised in order for the CNN to handle them. They will already share the same size when imported, but they must then be rescaled. The RGB channels are in the [0, 255] range and will need to be rescaled to [0, 1].

B. Architecture Design

After pre-processing, the CNN must be designed and constructed. The principal features of the architecture to consider are kernel size, padding, activation functions, pooling layers, and dense layers.

A simple, pre-designed CNN is used as the base model and then built upon for optimisation [3]. This model is composed of three convolution blocks with a max pooling layer in each block. This is followed by a dense layer with 128 units activated by a ReLU activation function. With this design as the base model, various modifications are required in order to optimise its performance. This involves techniques such as adding extra layers, regularisation, hyperparameter tuning, and data augmentation.

C. Learning

After running the base model on the training and validation sets, the aforementioned optimisation techniques are to be implemented. CNNs are trained over a series of "epochs", where the accuracy of each epoch changes. If the accuracy begins to fall or stagnate, an early stopping callback function stops the training. This can improve efficiency as it is not necessary to wait for the CNN to complete its set number of epochs when it is clear that it is not performing well. For this task, accuracy and loss are the performance metrics. Accuracy is the usual metric that compares predictions with true values, while loss is a summation of the errors made. Optimization will take place iteratively and be guided by both accuracy and loss. The Adam optimiser is selected for this task.

D. Evaluation

The best performing CNN is selected as the final model. It is then evaluated according the aforementioned metrics.

IV. Experiments

A. Pre-Processing

The raw data was imported and shuffled, with a batch size of 32 and a small image size of 130 x 130 to reduce computational load. The images were rescaled to [0, 1] in a scaling layer.

B. Architecture Design & Learning

The base model had three convolution layers with 32 filters and 3x3 kernels, each followed by a max pooling layer, and then two dense layers. After ten epochs, the base models accuracy was 59% and its loss was 2.45. As well as a relatively poor performance, the model was extremely overfitted to the training data, which had an accuracy of 95% and loss of 0.14.

The first optimisation method tested was data augmentation. This involves randomly rotating or flipping the images in the dataset to expose the model to different features in the photos, which can reduce overfitting. Indeed, this technique drastically reduced the overfitting, but only improved the model marginally.

A regularization technique called dropout was implemented after the data augmentation. Another technique that can help with overfitting, it works by randomly dropping out nodes during training. A dropout of 0.2 proved successful and reduced overfitting by increasing the accuracy. With this degree of dropout, validation accuracy was 65% while training accuracy was at 74%.

Next, two extra convolution layers were added, which can help the model extract more features. The base model only had three convolutional layers, meaning that it may struggle to identify enough useful features. The first additional layer had 32 filters, followed by a layer with 64 filters. This produced an accuracy of 72% on the validation set and 75% on the training set, while the validation set's loss was at 0.88. More layers than this either reduced the accuracy or maintained the same level of accuracy and were therefore redundant. Various numbers of filters were experimented with, but it was found that the two layers with 32 and 64 filters performed best.

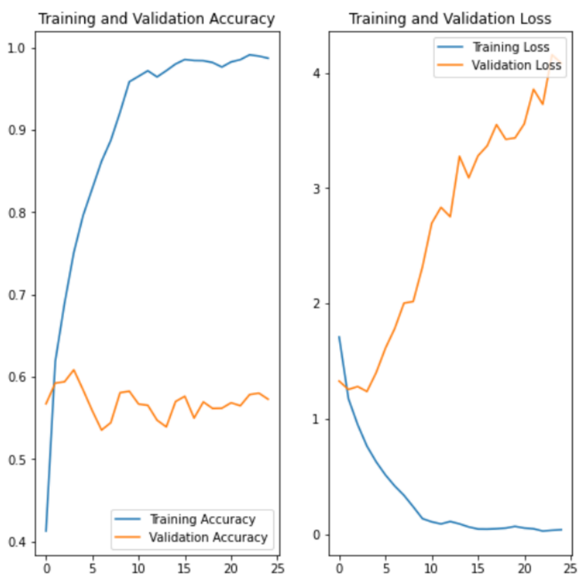
Different kernel sizes were also tested, as kernel size determines which features are extracted. The base model's kernel size of 3x3 proved to be most optimal.

C. Evaluation

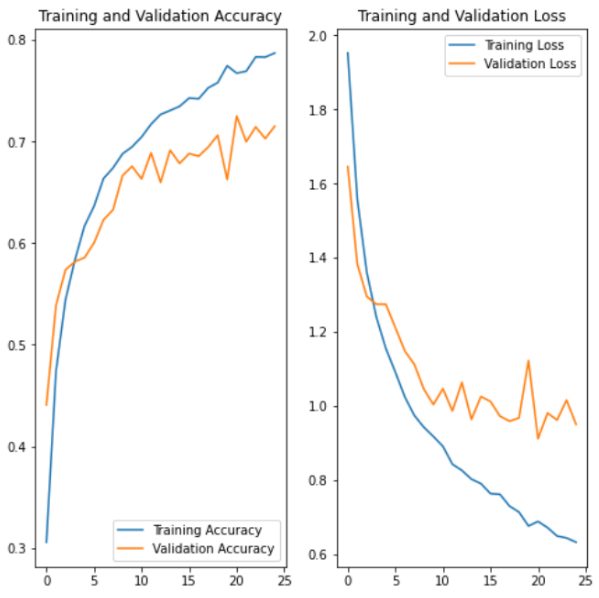
The performance of the final model compared to the base model can be viewed in the following Fig:

Performance Metric		**Base CNN**	**Optimal CNN**
Accuracy	Training	0.98	0.79
	Validation	0.57	0.72
Loss	Training	0.03	0.63
	Validation	4.04	0.95

The models were then run with 25 epochs, as loss on the optimised model continued to decrease until this point (and increase in the case of the base model). The following two graphs show the accuracy and loss changes across all 25 epochs for base model and optimal CNN model:



Baseline Model

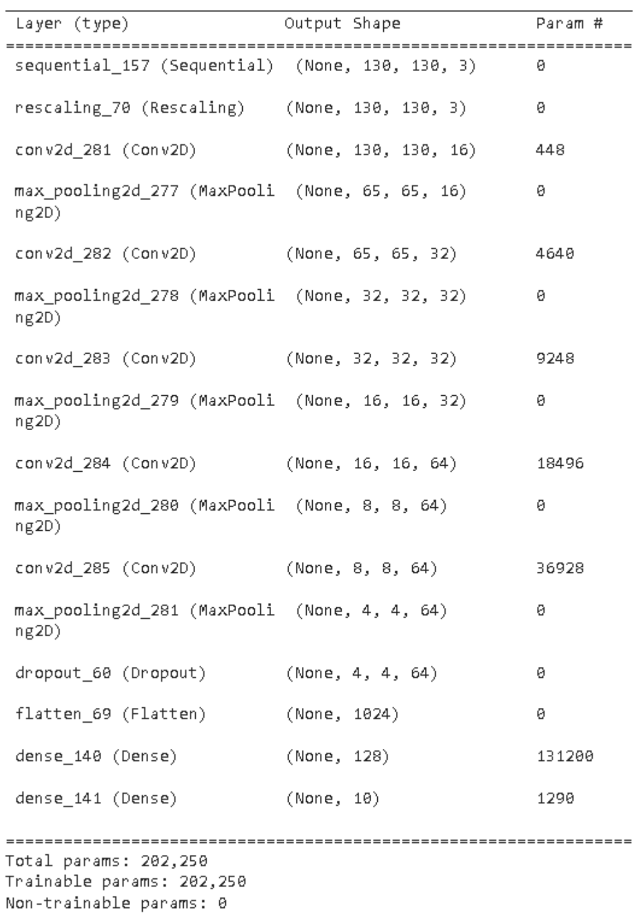


Optimised Model

As we can see from the table and graphs, the optimised model considerably outperforms the baseline model. It had a 72% accuracy on the validation set, compared to the base model's 59%. The optimised CNN's loss was also considerably lower at 0.95, while the baseline's was at 3.2. In addition to better performance, the optimised CNN largely reduced overfitting. Although the optimised model still had some overfitting, with the training set's accuracy at 79% and the validation set's accuracy at 72%, the overfitting was quite small - much smaller than the baseline model, whose training accuracy was 98% and validation accuracy 59%. This reduction in overfitting is clearly visible in the graphs. So not only does the optimised model outperform the base model, but it will also perform better on unseen data.

V. Reflection

A pipeline has been built that imports a set of images, cleans and formats them, before building a CNN model to classify them. With a simple model as the baseline, an optimised model was built by modifying the base model. The following graphic details the final architecture:



```
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_157 (Sequential)  (None, 130, 130, 3)      0

 rescaling_70 (Rescaling)    (None, 130, 130, 3)       0

 conv2d_281 (Conv2D)         (None, 130, 130, 16)      448

 max_pooling2d_277 (MaxPooli  (None, 65, 65, 16)       0
 ng2D)

 conv2d_282 (Conv2D)         (None, 65, 65, 32)        4640

 max_pooling2d_278 (MaxPooli  (None, 32, 32, 32)       0
 ng2D)

 conv2d_283 (Conv2D)         (None, 32, 32, 32)        9248

 max_pooling2d_279 (MaxPooli  (None, 16, 16, 32)       0
 ng2D)

 conv2d_284 (Conv2D)         (None, 16, 16, 64)        18496

 max_pooling2d_280 (MaxPooli  (None, 8, 8, 64)         0
 ng2D)

 conv2d_285 (Conv2D)         (None, 8, 8, 64)          36928

 max_pooling2d_281 (MaxPooli  (None, 4, 4, 64)         0
 ng2D)

 dropout_60 (Dropout)        (None, 4, 4, 64)          0

 flatten_69 (Flatten)        (None, 1024)              0

 dense_140 (Dense)           (None, 128)               131200

 dense_141 (Dense)           (None, 10)                1290

=================================================================
Total params: 202,250
Trainable params: 202,250
Non-trainable params: 0
```

This model produced an accuracy of 72% and a loss of 0.95. While this is not particularly strong compared to many CNN models, it is still a big step up from the baseline model. CNN models often have much more than five convolutional layers, which generally allows them to identify more features in the images. The trade off tends to be a higher computational load, which the proposed model is not burdoned with. In this particular case, adding more layers did not help the model's performance. Next time, I would tweak the parameters of the extra layers and play with more filter and kernel sizes and so on. Extra layers with the right combination of parameters would likely improve the accuracy of this model.

Bibliography

[1] A. F. E. B. H. B. a. F. H. M. Galar, “"A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. vol. 42, no. no. 4, pp. pp. 463-484, 2012.

[2] B. Walker, “PCA Is Not Feature Selection,” [Online]. Available: https://towardsdatascience.com/pca-is-not-feature-selection-3344fb764ae6#:~:text=PCA%20is%20a%20rotation%20of,encoded%20variables%2C%20you%20should%20not.. [Accessed 16 03 2022].

[3] TensorFlow, “Baseline CNN Model,” 2018. [Online]. Available: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb#scrollTo=WcUTyDOPKucd.

Appendix

The code used in the two tasks outlined in this report can be found on Google Colab at the following web addresses:

Task One:

https://colab.research.google.com/drive/1pfYyfJUnLe2ab9FU1WxRaEkys5lNZsoN?usp=sharing

Task Two:

https://colab.research.google.com/drive/19yMwg21__x4-jUYm-HUrFwfit2_Dl29k?usp=sharing

The dataset for task one can be found here:

UCI Machine Learning Repository: South German Credit (UPDATE) Data Set