Coin Counter Image Recognition Project

Final Report

APS360 - Introduction to Artificial Intelligence

Group 23

Word Count: 2489

Word Count Penalty: 0%

Maria Papadimitriou - 1003913624
Sara Nangini - 1004115209
Kimberly Shek - 1004108887
Amy Shi - 1004459338

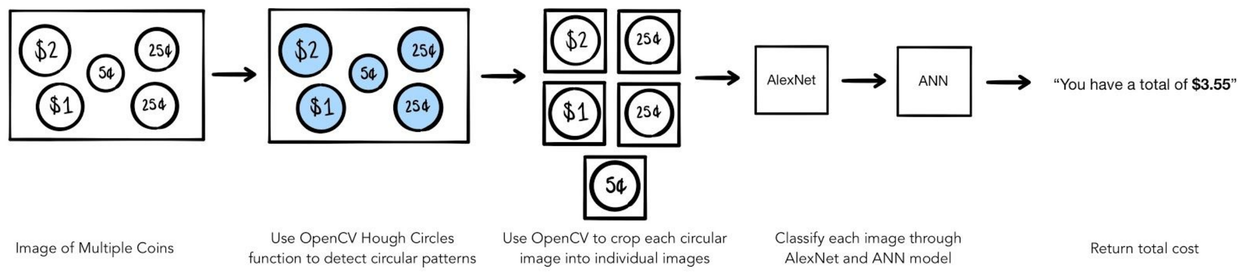
Table of Contents **Page**

1.0 Introduction

In retail industries, when consumers purchase items with cash, it is the responsibility of the cashier to count by hand in order to determine whether the item is paid for. There are many areas in this process which potentially increase retailer costs. For example, as the cashier is counting the cash, the service time of each payment increases. Increased waiting times are also a result of these service times. Additionally, manual counting has room for many errors, in which the miscounting of payments do not add up to the total inventory purchased.

Reducing manual cash counting through automation is extremely valuable as it increases worker productivity, reduces counting errors, and reduces overall costs. The goal of the project is to create a simple coin counter which will input an image of many coins and output the total sum. Machine learning is a reasonable approach as the methods utilized are able to iterate through all the identifiable coins, consistently update the total value of coins, and return the total sum, all within a speed arguably faster than the average human.

2.0 Illustration/Figure



$2
5¢
25¢
$1
25¢
AlexNet
ANN
"You have a total of **$3.55**"
Image of Multiple Coins
Use OpenCV Hough Circles function to detect circular patterns
Use OpenCV to crop each circular image into individual images
Classify each image through AlexNet and ANN model
Return total cost

Figure 2.1: Illustration of Overall Idea

3.0 Background & Related Work

Currently, field applications of coin counting are largely based on the weight and dimensions of the coins and require you to physically bring them to a counting machine. Coin counting is commonly done in one of two methods -- (1) by allowing coins to drop through holes placed in ascending order to only allow coins of a specific diameter to pass through or (2) by using a light beam [1][2]. The light beam method uses the time it takes to pass by in order to determine the size of the coin and its denomination.

There has been research done to apply artificial intelligence (AI) to identify coins through images. Research that has been performed deals with the identification and classification of individual coins and not the counting of multiple coins in one image. One research paper went through various steps to preprocess the image such as grayscale conversion, shadow removal, and cropping, then it was fed into a Artificial Neural Network (ANN), this paper was able to achieve a 97.74% accuracy [3]. Some other research that has been done includes additional steps, such as rotational invariance and edge detection [4]. Research has been done on individual coin recognition, but there is not a system which is able to take an image with multiple coins and identify how many of each coin there is. Our project will try to achieve a more comprehensive application of coin counting using AI.

4.0 Data Processing

Table 4.1: Data Collection Information

Class and Example	Information and Processing
Single Coins 	**Types of Coins**: nickel, dime, quarter, loonie, toonie **Number of Images per Coin**: 610 - 450 Training, 80 Validation, 80 Testing (74:13:13 ratio) placed into seperate folders organized by set and type of coin. - Half of these are made using processed images, rotated using the PIL library **Number of Single Coin Images in Total**: 3050 **Process**: - Take centered photo of the coin on the tail side on a solid background - Resize the image to 224 x 224 pixels - Label it (coin type)_#.jpg, increasing the number to match the number of images (e.g nickel_1.jpg) - Place the image in the corresponding Google Drive folder - Repeat with different coins of the same type to ensure the model can handle small differences such as the year the coin was made.
Grouping of Coins (Not to scale)	**Number of Grouping Images in Total**: 50 **Number of Images per Group**: 5 **Groups**: 1. Toonie, loonie, quarter, dime, nickel 2. Toonie, two loonies, quarter, two dimes 3. Quarter, loonie, toonie, two dimes, two nickels 4. Toonie, loonie, two quarters, dime, nickel 5. Toonie, loonie, quarter, two dimes, nickel 6. Toonie, loonie, quarter, dime, two nickels 7. Two toonies, two nickels, dime 8. Two dimes, quarter, loonie 9. Five nickels 10. Quarter, toonie **Process**: - Collect coins to make one of ten groups - Take a square photo of the grouping with a solid background 15cm above the coins while they are all on their tail side - Label it group#_#.png, depending on which group it is in and how

	many pictures have already been taken (e.g group1_1.png) - Place the image in folder that matches its value on Google Drive

5.0 Architecture

As the image of the group of coins is inputted, the Hough Circles function in OpenCV (a package in python) is used to detect circles within the image. Once these circles are detected, they are cropped using OpenCV at the coordinates of the edges, and converted into images of size 224x224. These cropped images are stored in a list as a PyTorch tensor, where they are then each passed through the final AlexNet and ANN classification model, where the resulting feature maps are passed to the linear layers which have 5 output channels for the 5 types of coins (nickel, dime, quarter, loonie, and toonie).

The pretrained convolutional neural network (CNN) AlexNet model contains 5 convolutional layers and 3 max pooling layers. An ANN was then added to the AlexNet model with 2 fully connected layers and 10 hidden units. The full classification model architecture can be seen below in Figure 5.1.

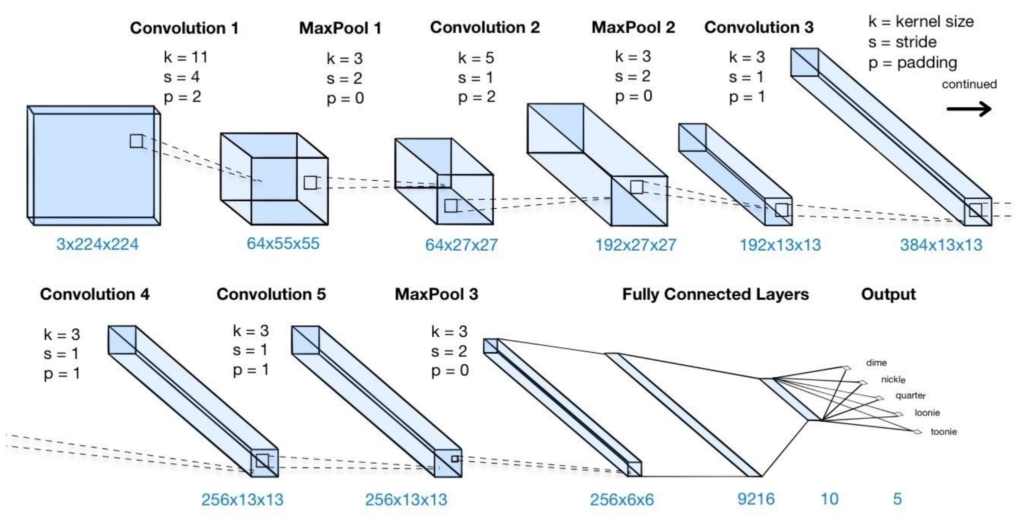


Figure 5.1: AlexNet Model Architecture and Fully Connected ANN Dimensions

The next hyperparameters determined were the learning rate, batch size, and number of epochs. After multiple iterations with different adjustments, the final model uses a learning rate of 0.001, batch size of 64, and number of epochs of 20. The final model parameters and further justification can be found in Appendix A, Table A1.

6.0 Baseline Model

An SVM baseline model is used to compare our neural network against. The SVM model was trained and tested with the same datasets as the AlexNet and ANN model. Figure 6.1 details the implementation steps for the SVM model.

The model was implemented with reference to the following online tutorial: [5]

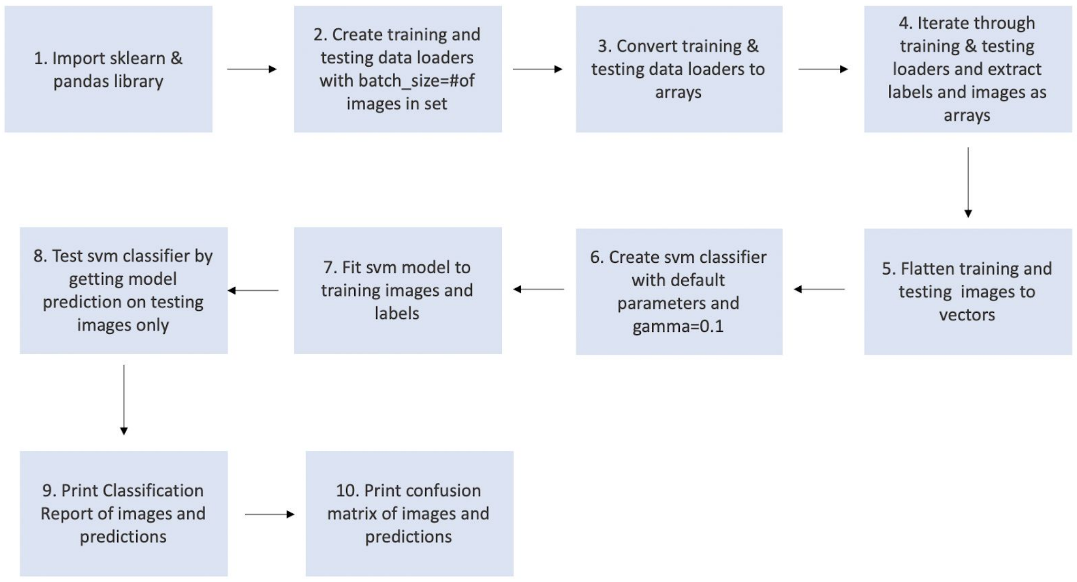
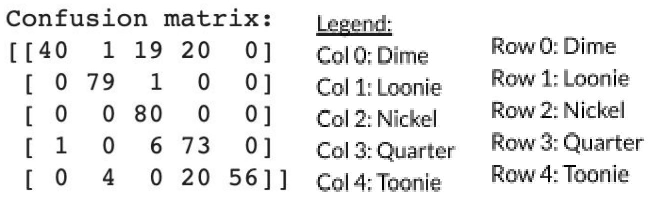

1. Import sklearn & pandas library
2. Create training and testing data loaders with batch_size=#of images in set
3. Convert training & testing data loaders to arrays
4. Iterate through training & testing loaders and extract labels and images as arrays
5. Flatten training and testing images to vectors
6. Create svm classifier with default parameters and gamma=0.1
7. Fit svm model to training images and labels
8. Test svm classifier by getting model prediction on testing images only
9. Print Classification Report of images and predictions
10. Print confusion matrix of images and predictions

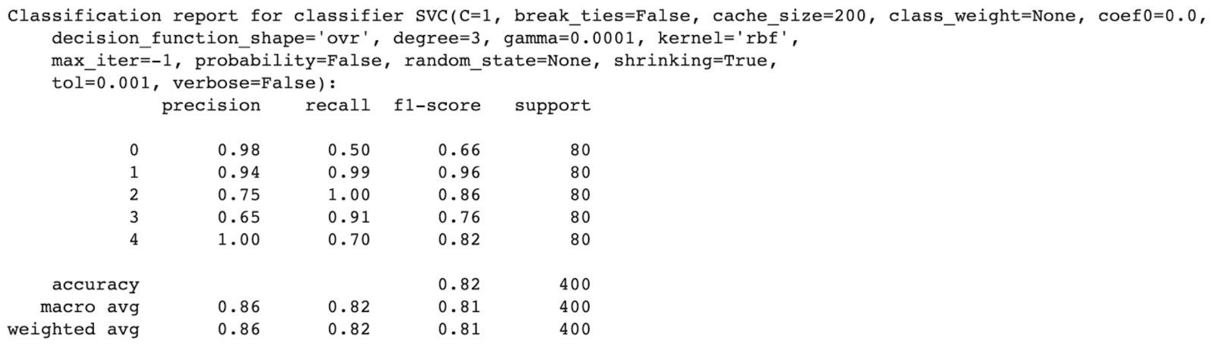
Figure 6.1: SVM Classifier Implementation Steps

The testing accuracy score for the SVM model was obtained through the classification report and confusion matrix available in the Sklearn packages shown in Figure 6.2 and 6.3 [5]. The final testing accuracy of the baseline model is 82%. The hyperparameters to obtain the final accuracy score are shown in Figure 6.3. Decreasing the gamma parameter to 0.0001 and utilizing an RBF kernel increased the precision for most classes and yielded the highest testing accuracy [6].



```
Confusion matrix:
[[40  1 19 20  0]
 [ 0 79  1  0  0]
 [ 0  0 80  0  0]
 [ 1  0  6 73  0]
 [ 0  4  0 20 56]]
```

Figure 6.2: Confusion Matrix for final SVM Classifier



```
Classification report for classifier SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.98      0.50      0.66        80
           1       0.94      0.99      0.96        80
           2       0.75      1.00      0.86        80
           3       0.65      0.91      0.76        80
           4       1.00      0.70      0.82        80

    accuracy                           0.82       400
   macro avg       0.86      0.82      0.81       400
weighted avg       0.86      0.82      0.81       400
```

Figure 6.3: SVM Classifier Classification Report

Based on the confusion matrix, the per class error rate on the testing data for the SVM classifier is shown in Table B1 in Appendix B. As per Table B1, dimes had the highest error rate out of the five classes, and were most commonly classified as quarters instead. This could be due to the similarity in colour between the dimes and quarters.

7.0 Quantitative Results

This section will provide quantitative results the team has compiled through training and testing the AlexNet and ANN model.

7.1.1 AlexNet and ANN Model Training Performance

Model achieves 99.8% training accuracy, please see Figure 7.1 below. The model was able to learn the training set well with negligible error of about 0.2%. As the number of epochs increased, the model loss converged toward 0 indicating that model parameters have been adjusted well enough to learn training data.

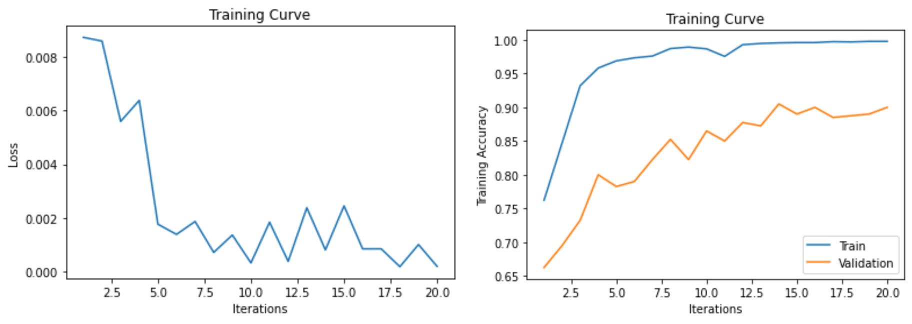


Figure 7.1: Training and Validation Accuracy Scores

7.1.2 AlexNet and ANN Model Validation and Testing Performance

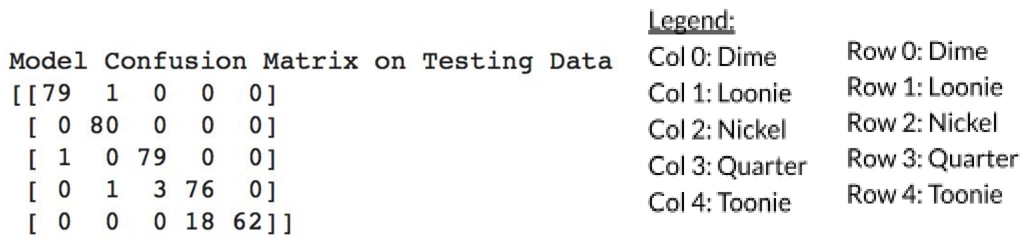
The model achieves a validation accuracy of 90% and a testing accuracy of 94%. The high testing score indicates that the model performance does not degrade with new images. This suggests a minimal amount of overfit from the model to the training data. However, the validation score is lower than the testing score. This could be due to more difficult images being present in the validation set rather than the testing set. These difficult images could be photos with a larger amount of glare or blurriness present in the validation set.

7.1.3 Baseline Model Testing Accuracy vs AlexNet and ANN model Testing Accuracy

The baseline model testing accuracy of 82% is significantly lower than the testing accuracy of the AlexNet and ANN model at 94%. Although both scores can be considered strong accuracies, the model had significantly lower error rates per class than the baseline model. This difference in score suggests that the model is functioning well and is complex enough to out perform a simple baseline model.

7.1.4 Testing Error in AlexNet and ANN Model is Consistent

As per the Confusion Matrix in Figure 7.2, Table C1 in Appendix C details the error rate present in the AlexNet and ANN model across classes on the testing data. A relatively low and consistent amount of error present in the model for each class, with an average error rate of 5.9%. These results show that the models' learning abilities are strong among classes. Toonies had the highest amount of error of 22.5%, with all incorrect predictions being quarters. This error could be a result of the similar colour of the outer ring of toonies and quarters. Variability in how close a testing photo was taken from the face of a coin could make the size of the coin more difficult to distinguish. An example of this comparison is shown in Figure 7.3 below.



```
Model Confusion Matrix on Testing Data
[[79  1  0  0  0]
 [ 0 80  0  0  0]
 [ 1  0 79  0  0]
 [ 0  1  3 76  0]
 [ 0  0  0 18 62]]
```

Legend:

Col 0: Dime	Row 0: Dime
Col 1: Loonie	Row 1: Loonie
Col 2: Nickel	Row 2: Nickel
Col 3: Quarter	Row 3: Quarter
Col 4: Toonie	Row 4: Toonie

Figure 7.2: Confusion Matrix for AlexNet and ANN model on Testing Data

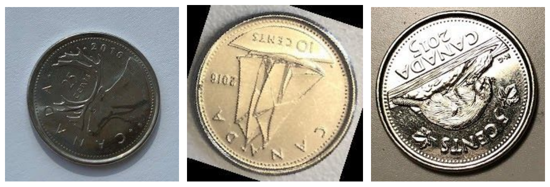
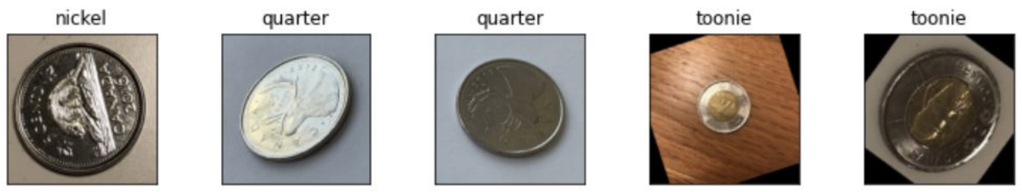


Figure 7.3: Example of Quarter, Dime and Nickel Testing images.

8.0 Qualitative Results

The team analyzed sample outputs from the AlexNet and ANN network on testing data to understand model predictions across classes that support the model accuracy and error scores.

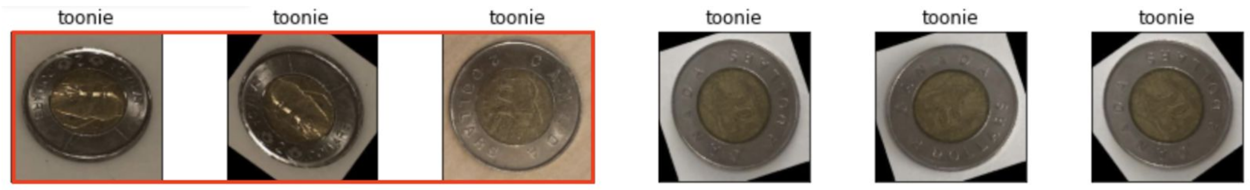
The sample shown in Figure 8.1 is a random selection of images from the testing set for each class. This sample consists of 2 randomly selected samples for each class from the testing data which the model correctly identifies. The variety of backgrounds and orientations shown in the sample, indicates that the model was able to handle variability between samples and does not rely on features other than those from the coin for classification.



```
Model predictions for Random Sampling Set:  ['dime', 'dime', 'loonie',
'loonie', 'nickel', 'nickel', 'quarter', 'quarter', 'toonie', 'toonie']
```

Figure 8.1 Random Testing Sample Predictions

As per Figure 7.2 and Table C1 in Appendix C, toonies had the highest amount of error where many toonies were misclassified as quarters. A random sample of toonies and respective predictions from the testing data are shown in Figure 8.2. The first three toonies were incorrectly classified as quarters, which correlates to the confusion matrix results. The misclassified toonie photos have significant glare and uneven lighting on the face of the coin. These lighting differences could have made it more difficult for the model to focus on the silver outer ring of the toonie rather than recognizing the golden inner face of toonies, resulting in a misclassification to quarters as they are silver coins of similar size.

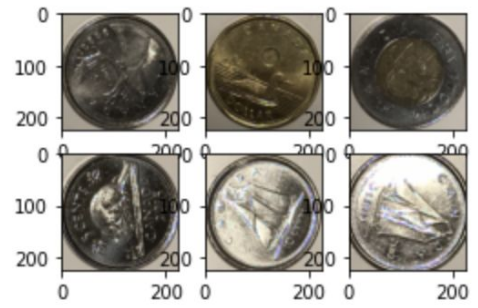


```
Model predictions for Random Sampling Set of Toonies:  ['quarter',
'quarter', 'quarter', 'toonie', 'toonie', 'toonie']
```

Figure 8.2: Random Toonie Testing Sample Predictions

9.0 Model Evaluation on New Data

New data can be obtained by taking a photo of one or more coins. The team used a different data set of 50 photos of coin groups to test how the model will perform with never before seen data. OpenCV is used to crop individual coins from the image which are fed into the model shown in Figure 9.1, these photos were not used in hyperparameter tuning.



```
['quarter', 'quarter', 'quarter',
    'nickel', 'dime', 'dime']
```

Figure 9.1 Model Results From Cropped Group Coins

Table D1 in Appendix D shows the results of running the model on the images of grouped coins. As a whole the model returns the correct balance 34% of the time, which is expected due to the nature of the model and problem. As the coin classification accuracy is not perfect, it can be expected that a coin is occasionally classified incorrectly; this causes the prediction for the entire group to be incorrect. In many cases for the images of groups of coins, when the model predicted the wrong monetary value, it was due to one coin being classified incorrectly. Looking at coins individually, the model correctly predicted 214/260 (82.3%) coins correctly. The accuracy with the new data is slightly lower than the validation accuracy we obtained while training, but this result is around what is expected given the difference in quality between the training and grouping images.

10.0 Discussion

The accuracy of the new data is lower than our validation and testing accuracy which is 90% and 94% respectively. This can be expected due to the quality of the group images for two reasons. First, the coin grouping images are not as well lit as glare became problematic when too much lighting was allowed, shown in Figure 9.1 where the dark loonie and toonie are misclassified. Second, since the image has multiple coins, less emphasis is placed on each coin, unlike the training/validation/testing images; so the new images may not convey the coins' details as well. The training dataset images are bright and clear to allow convolutional layers to recognize unique patterns with each coin. Thus, the grouping coins look different from what the model saw in training and testing. It was surprising to see how much of an impact the small details mentioned can have on the performance of the model.

From this, we learned that using never before seen data assists in detecting flaws in the model. In this case, the model did not predict loonies as well in the new data, but in the confusion matrix they were classified accurately. This also gives us insight on biases in training data, as our model was biased towards well lit and highly detailed coins. Biases in AI can be dangerous in other real life scenarios, such as Amazon's secret recruiting tool that showed a bias against women [7].

11.0 Ethical Considerations

When using the system, ethical concerns around privacy, security, and accuracy could come into question. With a model that deals with the collection of financial data, it is important to be conscious of user privacy as specific amounts of money users have in their possession will be collected. Concerns could also arise when dealing with the accuracy and security of the model. Since this system deals with money, there is always the possibility of misuse, such as the feeding of counterfeits into the model. Without the ability to guarantee 100% accuracy, there is always the possibility of miscounting which could lead to financial discrepancies, this could cause problems such as undue doubt towards employees handling the money.

12.0 References

[1] “How Do Coin Counters & Coin Sorters Work?,” OurWeigh, 06-Oct-2016. [Online]. Available: https://www.ourweigh.co.uk/blog/how-do-coin-counters-and-coin-sorters-work.html. [Accessed: 21-Feb-2020].

[2] J. Staughton, “How Does Cash Counting (Money Counting) Machine Work?,” Science ABC, 09-Jan-2016. [Online]. Available: https://www.scienceabc.com/innovation/how-do-currency-cash-note-money-counting-machines-work.html. [Accessed: 21-Feb-2020].

[3] S. Modi and S. Bawa, “Automated Coin Recognition System using ANN,” International Journal of Computer Applications, vol. 26, no. 4, Jul. 2011.

[4] S. Malik, P. Bajaj, and M. Kaur, “Sample Coin Recognition System using Artificial Neural Network on Static Image Dataset,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 1, Jan. 2014.

[5] Halien, “Simple Image classifier with SVM,” *Kaggle*, 25-Jun-2017. [Online]. Available: https://www.kaggle.com/halien/simple-image-classifer-with-svm. [Accessed: 20-Mar-2020].

[6] “(Tutorial) Support Vector Machines (SVM) in Scikit-learn,” DataCamp Community. [Online]. Available: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python. [Accessed: 20-Mar-2020].

[7] J. Dastin, “Amazon scraps secret AI recruiting tool that showed bias against women.” Reuters, 09-Oct-2019. [Online]. Available: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

13.0 Appendices

Appendix A

Table A1: Final Model Parameters

Parameter	Value	Reasoning
Number of Convolutional Layers (AlexNet)	5	Provided optimal train, validation, and testing accuracies.
Number of Fully Connected Layers (ANN)	2	Provided optimal train, validation, and testing accuracies.
Number of Hidden Units	10	Provided optimal train, validation, and testing accuracies.
Batch Size	64	Provided optimal train, validation, and testing accuracies.
Learning Rate	0.001	Provided for optimal accuracy in the validation and testing set, where it is not too small that it never reaches the minima, but also not too large where it skips over the minima.
Epochs	20	Validation accuracy remained consistent above 20 epochs, and performing more than 20 would be redundant as it provided no additional information and took a longer time to run.

Appendix B

Table B1: SVM Classifier Model Predictions per Class

*80 samples from each class were present in testing data.

Class	SVM Model Predictions per Class	Error (%)
Dime	40	50%
Loonie	79	1.25%
Nickel	80	0%
Quarter	73	8.75%
Toonie	56	30%
Average Error Rate		**18%**

Appendix C

Table C1: AlexNet and ANN Model Predictions per Class
*80 samples from each class were present in testing data.

Class	**Model Predictions per Class**	**Error (%)**
Dime	79	1%
Loonie	80	0%
Nickel	79	1%
Quarter	76	5%
Toonie	62	22.5%
Average Error Rate		**5.9%**

Appendix D

Table D1: Accuracy Results of Putting Each Group Coin Image Through the Model

	Number of Correct Predictions/Total Amount					
Group	**Overall Group Correct**	**Nickel**	**Dime**	**Quarter**	**Loonie**	**Toonie**
$0.25	1/5	19/25	-	-	-	-
$1.45	0/5	-	10/10	5/5	0/5	-
$2.25	5/5	-	-	5/5	-	5/5
$3.40	2/5	5/5	5/5	5/5	3/5	3/5
$3.45	3/5	9/10	5/5	5/5	5/5	4/5
$3.50	1/5	5/5	8/10	4/5	2/5	4/5
$3.55	1/5	10/10	8/10	5/5	3/5	3/5
$3.65	0/5	5/5	5/5	9/10	2/5	4/5
$4.20	3/5	9/10	5/5	-	-	8/10
$4.45	1/5	-	7/10	5/5	6/10	4/5
Total Accuracy	17/50 = 34%	62/70 = 88.6%	53/60 = 88.3%	43/45 = 95.6%	21/40 = 52.5%	35/45 = 77.8%

14.0 Google Colaboratory File

https://colab.research.google.com/drive/1VPyDdWm4VWLTP197lR07riN5RAaqBlf9