Machine Learning Challenge 35

Introduction

This project aims to create an algorithm to recognize sign language images and to classify them correctly into different groups. Two different Machine Learning models, Convolutional Neural Network and Decision Trees, have been built for the project.

Sign language involves the usage of different parts of the body, such as fingers, hand, arm, head, body, and facial expression. One class of sign languages, known also as fingerspelling, is the representation of the letters of the writing system, and sometimes numerical systems, using only the hands. These manual alphabets (also known as finger alphabets or hand alphabets) have often been used in deaf education and have subsequently been adopted as a distinct part of a number of sign languages.[1] There are different types of sign languages across the world, but this project uses only The American Sign Language which is widely used by the majority of the deaf community in the US and Canada. The American Sign Language (ASL) is a natural language with a structure quite different from spoken English. It is not a manual- gestural representation of spoken English, nor is it a pantomime. Instead, ASL is a full language, with all of the properties of spoken natural languages, but one that has developed independently of and differently from English.[2]

Data description

The American Sign Language letter dataset of hand gestures consists of 24 classes of letters and represents a multiclass classification problem. In general, each label is illustrated as a 0-25 value that corresponds to the location of the value in the English alphabet, for example, A=0, B=1, C=3. Two classes of letters excluded (J=9 and Z=25) because of the gesture motions. The dataset itself is divided into training data and test data. The number of training cases is 27,455 (includes images and labels), and the number of test cases is 7172 (includes images and labels). All the images are in the form of numpy arrays and labels are indexed according to their positions. Each instance in both subsets is a 789 dimensional vector that can be reshaped to an image a size of 28 to 28 with greyscale values between 0-255.

After loading the training and the test data, a shape of the training data, unique labels in the training data and a balance between classes have been explored. Next, the elements of the training data and test data have been converted to the same type. In order to reduce the effect of illuminations' differences a grayscale normalisation has been introduced. Moreover, the CNN converges faster on [0...1] data than on [0...255].[3].Afterwards, the data has been reshaped from 1-D to 3-D as it was a required input for the first layer of CNN.[4]

In the next step, a one-hot encoding has been used to convert integer labels to binary form. In Machine Learning some algorithms require input to be a numeric value, which implies they cannot directly work with categorical data. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.[5] In the given dataset every label is represented as single values from 1 to 24. When these labels are used in the CNN output layer, there will be 24 nodes since there are 24 different classes. However, after the implementation of the one-hot encoding, each label will be in the form of an array that has a length of 24 with corresponding label being 1 and the rest are 0, such as if y=5 the array is [0 0 0 0 1 …0].

Finally, the dataset has been split into three parts, a training set, a validation set and a test set with the ratio 60%, 20% and 20% respectively.

Task 1

Model 1: Convolutional Neural Network (CNN) model

Reason

The first model chosen for the assignment is the Convolutional Neural Network (CNN) machine learning model. CNNs are commonly used for image classification and recognition because of its high accuracy.[6] To briefly explain, CNN is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.[7]

Input to classifier

Before the CNN model was built, the data given had to be changed in order to fit a CNN model. First the data was normalised and then reshaped with an extra dimension to make it fit in the CNN model. Finally, the train data labels were transformed through one hot encoding. This made it easier to categorise the different possible hand sign pictures. After extracting and transforming the train data, it was finally possible to split the train data into a train/validation set.

The train data was split 80% train data and 20% validation data. Commonly a 25% margin is used to split the data but the test data is already split in a different data set. The train en test data set is divided into 80/20 respectively and 0.25 x 0.8 = 0.2, so the validation set is 20% of the train data. After splitting the train data, the CNN model can be built.

Hyperparameter(HP) tuning and model training

To start off with building the model, different CNN models were compared and looked at in kaggle and from comparing these models, a base model was created. The base model already has some form of hyper-parameter tuning. The base model already makes use of dropout and batch normalisation, as well as optimizers and loss functions (more on that in the next paragraph). For testing purposes, most of these options were excluded and this test produced a model with a test accuracy of 0.9498. This basi test is not included in the notebook anymore as the model with data augmentation and a bit of hyper-parameter will be used as a base model.

The data augmentation for the base model was used from a reference in kaggle.[8] The use of these data augmentations and the bit of hyper-parameter tuning made the model more robust against overfitting. After completing the base model, a tuned model was created using the Keras tuner on the base model.[9] The Keras tuner tried to find the optimal parameter settings for the model and those settings were then used to create the tuned model. Furthermore, the epochs and batch size were changed to see which fitted best on the model. The results of the models are mentioned in the next paragraph.

After creating the tuned model, the l2-regularizer tried to train the model, but it only decreased the accuracy and increased the loss of the model. No special optimizer or loss function were further tried to improve the model, as it was functioning very well without them.

Results Report performance and confusion matrix

The metrics which the model should focus on was accuracy. The test accuracy of our base model was 0.9598, the test loss was 0.0137, and the MAE was 0.0023. For the tuned model, the test accuracy was 0.9262, the test loss was 0.2830, and the MAE was 0.0069. Surprisingly, the accuracy and loss performed better with the base model than the tuned model. This is the reason why the base model is chosen to perform task 2. The summary of the base model is found in figure 1 in the appendix, the confusion matrix of the base model can be seen in figure 2 , and the accuracy per class can be seen in figure 3. The class which was most difficult to classify for the model was class 17.

Model 2: Decision tree model

Reason

A decision tree model is a simple method for classifying objects based on their features. It has a tree structure that is hierarchical and consists of a root node, branches, internal nodes, and leaf nodes.[10] The model is also fast and efficient. Other models were also implemented for the task. A KNN model, for example, was also chosen, but the kernel died when attempting to implement it. So, the decision tree model had the advantage of being fast and efficient when comparing it to other classification models. A Decision tree model is also very intuitive and simple to explain.[11]

Input to classifier

The accuracy of the model before doing any feature engineering is 24%, however the decision tree model has the advantage of not requiring feature transformation when dealing with non-linear data because decision trees do not consider multiple weighted combinations at the same time. So, feature engineering is not applied. The data was split in the same way as in the CNN model.

Hyperparameter (HP) tuning

The parameters that are tuned and the candidate values for each hyper parameter are the following:

```
parameters = {'max_depth' : (3,5,7,9,10,15,20,25) , 'criterion' : ('gini', 'entropy') , 'max_features' : ('auto', 'sqrt', 'log2') , 'min_samples_split' : (2,4,6) }
```

To see the impact the hyper parameter values have on performance different values are used for the decision tree model. One of these hyper parameter values are the following:

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, max_features='log2', min_samples_split= 2, random_state=123)
```

The accuracy in this case are as follows:

```
Train Accuracy - : 0.235
Test Accuracy - : 0.229
```

The accuracy in this case is very low, which is not ideal because we would like a model with both a high accuracy and precision. If we change the hyper parameter 'max_depth':15 to 25 and max_features:'sqrt' to 'auto' the accuracy changes as follows:

```
Train Accuracy - : 1.000
Test Accuracy - : 0.869
```

As we can see accuracy is a lot higher for both the train and test data when we change these values. Since the accuracy is 100% and the test accuracy is lower this can mean there is overfitting.

Additional information about training

To deal with overfitting and to see which values are the best for each hyper parameter DT_grid.best_params_ was used. The result of this is that the best parameters across all the candidate values are printed. This can be seen as follows:

```
The best parameters across ALL searched params:
{'min_samples_split': 2, 'max_features': 'sqrt', 'max_depth': 15, 'criterion': 'entropy'}
```

These values are also used to train the model.

Results Report performance and confusion matrix

The accuracy after hyperparameter tuning with best values for the train and test data are as follows:

```
Train Accuracy - : 0.983
Test Accuracy - : 0.859
```

Precision is 0.97, Recall is 0.96 and the confusion matrix is as follows:

```
[[221   2]
 [  3  65]]
```

.

The letter A is the simplest to categorise. This can be determined by creating a table in which the diagonal elements represent the number of correct classifications for each class. A is mostly correctly classified in this table.

Comparison of ML model 1 and ML model 2

The overall accuracy of the CNN model on the test set is 0.9598 and the overall accuracy of the decision tree model is 0.8590. The CNN model accuracy is higher than the accuracy of the decision tree. This means that the CNN model is better at identifying relationships and patterns between variables in a dataset based on the training data. The CNN model is also better at predicting, which can be more valuable than the decision tree model.[12] The decision tree model, on the other hand, has a much shorter training time than the CNN model. If the training data set is large, this may be a reason to use the decision tree model rather than the CNN model. However, we can conclude that the CNN model is better than the decision tree model for image classification. The macro average precision for the decision tree model is 0.1474 and the macro average precision for the CNN model is 9657. Since the macro average precision for the CNN model is a lot higher than the macro average precision of the decision tree model and the accuracy of the CNN model is also high this means that the CNN model is more reproducible and that the true values and predicted values are similar.[13]

Task 2

Abstract

For task 2, we built an image pre-processing function of 8 steps to process the given images. Then a cropping image function was built to crop the processed images. Finally , a loop function was created to run the preprocessing image and cropping image. Then the cropped images were predicted for the Top 5 accuracies based on the built CNN model of the task 1.

Image pre-processing

The images in test_images_task2.npy contain noises that heavily influenced the accuracy of image detection.In order to resolve the problems caused by image noise, 8 steps of image pre-processing techniques (Table 1) were applied to enhance the accuracy of prediction.

Firstly, the thresholding and the median denoise filter were used to remove noises. After denoised, two contrast stretching techniques were applied to adjust the contrast of each image in order to enhance the features of the image. After contrast stretching, we tried to remove the small object again. Then we tried to extract the features by applying the White Tophat filtering. Binary dilation technique was applied to enhance the extracted features. Finally, we applied the remove small objects function to determine the cropping size as a sanity check for the prediction.

The model generalisation is the most challenging and time-consuming part for the image processing task of task 2. Due to the images in the dataset not having similar contrast and similar features, slightly adjusted parameters of certain image pre-processing techniques may result in the failure of cropping images. Removing the unwanted shadows in the processed images is the most demanding part. While removing a certain shadow also means removing some subtle features of the image, it is a trade-off to remove the shadow or keep the subtle features of the image.

The image pre-processing process was operated many times because some images were not applicable for the step of cropping images, especially when there is only one letter or two letters in the image, such as the 9586th image (Figure 4).It is very time-consuming to find the optimal image processing model for all the images in the dataset of task 2. We had to adjust the hyperparameters of the image processing function and the cropping image function when we couldn't pass this problematic image. Due to the 9586th image being the last few images of the given dataset we must rerun the loop function of task 2 when we were very close to the finish line. The final version is the most optimal model among the other failures of image processing and cropping image models.

Cropping the images

Before cropping images, an equal padding of black pixels was added horizontally and vertically. After that, the function of connected-component labelling was applied to build a model for cropping images. First, the labelling of the regions of the pre-processing image. Then the regions of images are resized as (28, 28, 1), in order to pass the image recognition model.

Top-5-accuracy prediction

Top 5 accuracy of prediction were processed after applied image pre-processing and cropped the images. The function of predicting the top 5 predictions was applied to return the top 5 accuracies. Due to each image not containing the same number of images, it took some time to write a function that fit the task of Top-5 accuracy prediction. We had to transpose the array in order to yield the desired result. Furthermore, padding leading zero for the predicted number was also processed because the predicted number was a list. In order to meet the requirements of the assignment, we took some time to write a function that transforms the list to string.

References:

1. Fingerspelling. Wikipedia. Available online: https://en.wikipedia.org/wiki/Fingerspelling
2. Encyclopaedia Britannica, “Sign Language”. Available online: https://www.britannica.com/topic/sign-language
3. CNN using Keras. Available online: https://www.kaggle.com/code/madz2000/cnn-using-keras-100-accuracy
4. Sign Language Classification. Available online: https://www.kaggle.com/code/sayakdasgupta/sign-language-classification-cnn-99-40-accuracy/notebook
5. Why One-hot encode data in Machine Learning?Available online: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

6.Overview of Convolutional Neural Network in image classification. Available online:https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/#:~:text=CNNs%20are%20used%20for% 20image,visual%20perception%20of%20recognizing%20things

7. A comprehensive guide to Convolutional Neural Networks - the ELI5 way. available online:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

8. CNN using Keras. Available online: https://www.kaggle.com/code/madz2000/cnn-using-keras-100-accuracy

9. Hyperparameter tuning of Neural Networks using Keras tuner. Available online:https://www.analyticsvidhya.com/blog/2021/08/hyperparameter-tuning-of-neural-networks-using-keras-tuner/

10. What is the decision tree? Available online:https://www.ibm.com/topics/decision-trees

11. Decision tree advantages and disadvantages. Available online:https://www.educba.com/decision-tree-advantages-and-disadvantages/

12. Machine Learning model accuracy. Available online:https://www.datarobot.com/wiki/accuracy/

13. What is the difference between accuracy and precision? Available online:https://www.thoughtco.com/difference-between-accuracy-and-precision-609328

References for task 2:

1. Brownlee, J. (2020). How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras. Retrieved 20 May 2021,from machinelearningmastery: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

2. Brownlee, J. (2020). Why One-Hot Encode Data in Machine Learning? Retrieved 20 May 2021, from machinelearningmastery: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

3. Brownlee, J. (2020). How to Evaluate Pixel Scaling Methods for Image Classification With CNNs. Retrieved 20 May 2021, from machinelearningmastery: https://machinelearningmastery.com/how-to-evaluate-pixel-scaling-methods-for-image-classification/

4. Maladkar, K. (2018). Overview Of Convolutional Neural Network In Image Classification. Retrieved 15 June 2022, from analyticsindiamag:https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/#:~:text=CNNs%20are%20used%20for%20image,visual%20perception%20of%20recognizing%20things

5. Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Retrieved 15 June 2022, from towardsdatascience:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

6. Mathur, M. (2020). CNN using Keras(100% Accuracy). Retrieved 5 May 2022, from Kaggle: https://www.kaggle.com/code/madz2000/cnn-using-keras-100-accuracy/notebook

7. Singh, A. (2021). Hyperparameter Tuning Of Neural Networks using Keras Tuner. Retrieved 16 june 2022, from analytics vidhya: https://www.analyticsvidhya.com/blog/2021/08/hyperparameter-tuning-of-neural-networks-using-keras-tuner/

8. KUMAR V. (2000) Hands-On Guide To Sign Language Classification Using CNN. Retrieved 20 May 2021 from ANALYTICS INDIA MAGAZINE PVT LTD https://analyticsindiamag.com/hands-on-guide-to-sign-language-classification-using-cnn/

9.Lindsay, G. W. (2020). Convolutional neural networks as a model of the visual system: past, present, and future. Journal of cognitive neuroscience, 1-15.

10.O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458. Daume, H. (2017). A course in Machine Learning. (2nd ed.).

11.Pedregosa, F., Varogaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnav, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

The list of deliverables:

1. Task 1 – "Model 1": EDA, data preprocessing, data augmentation, building a CNN model, hyperparameter tuning and confusion matrix.

Thomas Brinkman, Kafia Elmi and Madina Makhmudova

2. Task 1 – "Model 2": Building a Decision tree model, checking for the accuracy, precision and recall.

Thomas Brinkman, Kafia Elmi and Madina Makhmudova

3. Task 2 –"Model 2": Image processing, image cropping, padding, slicing and getting top 5 predictions.

Chi Hung

4. Report :

Introduction - Madina Makhmudova

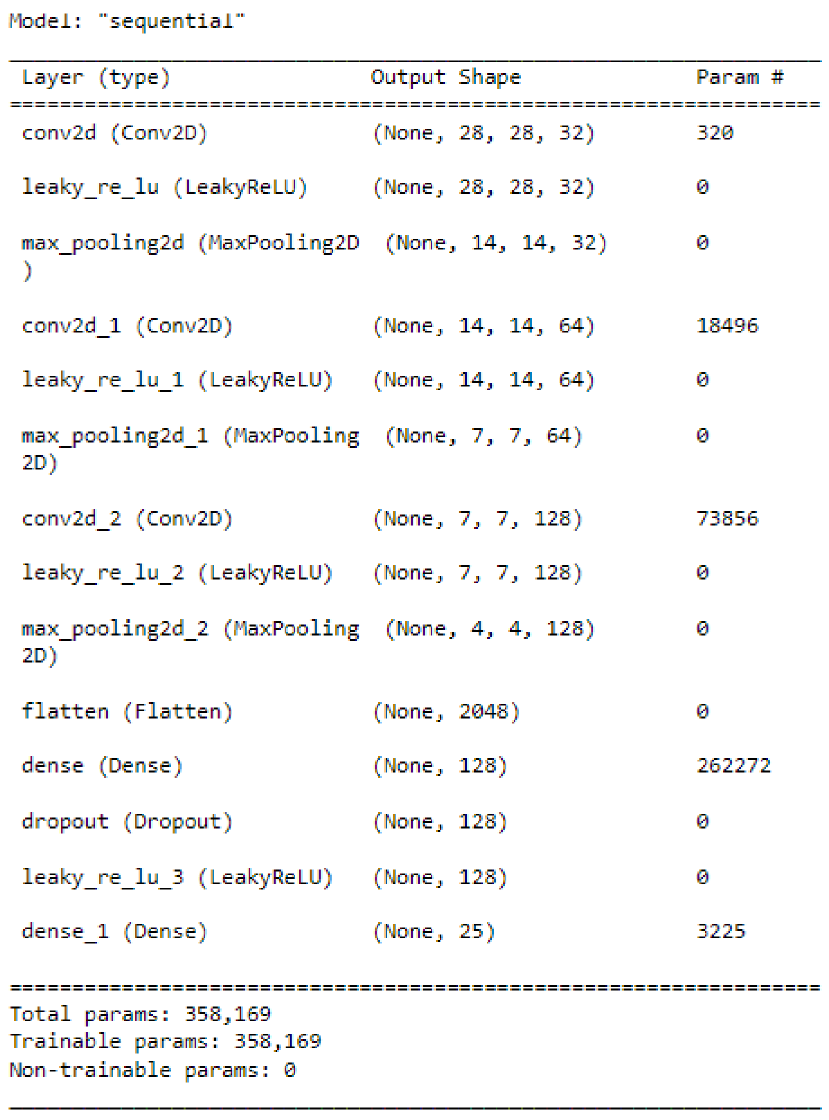
Task 1 – "Model 1" – Thomas Brinkman

Task 1 – "Model 2" – Kafia Elmi

Comparison of two models - Kafia Elmi

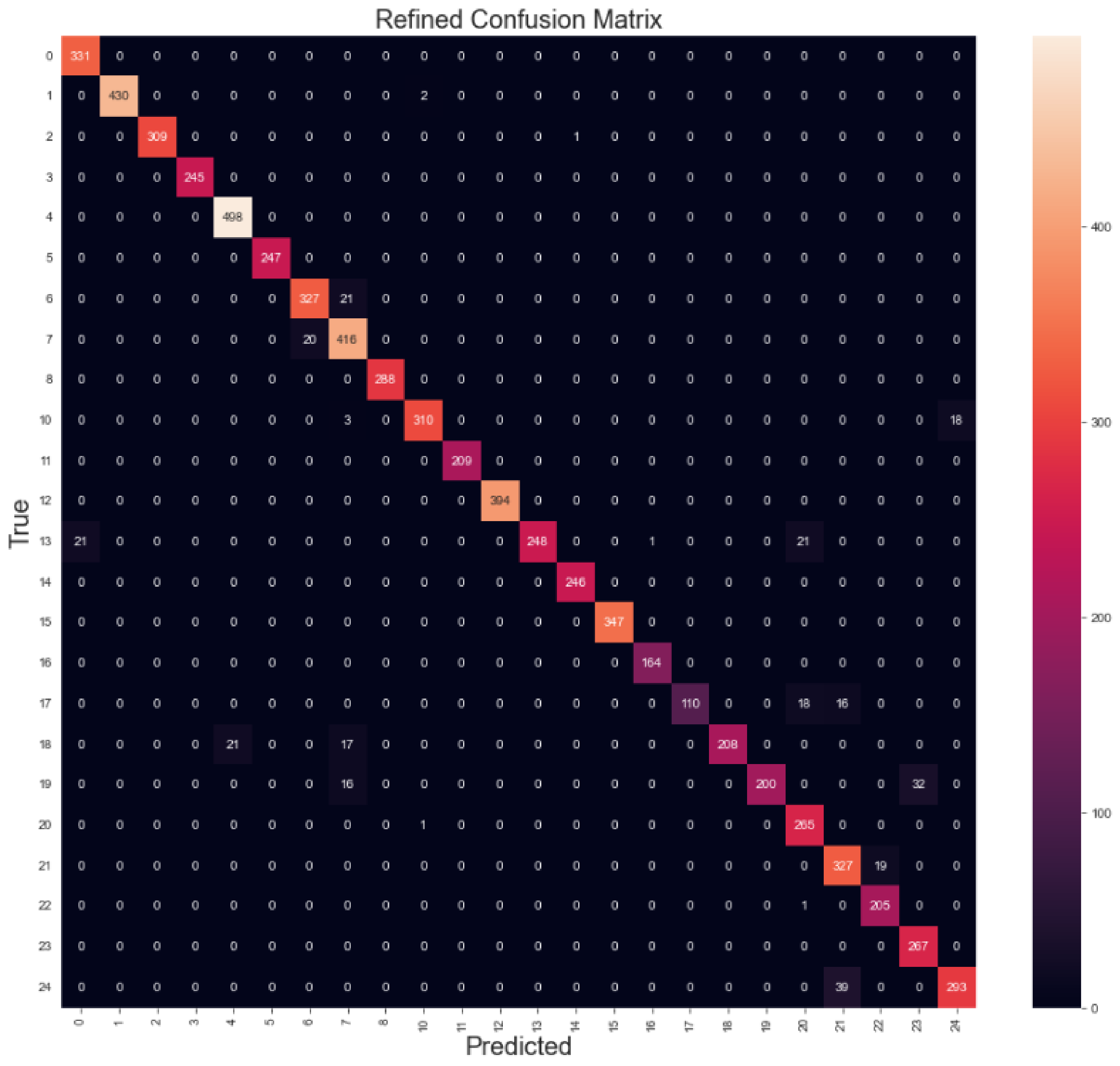
Task 2 – Chi Hung

Appendices



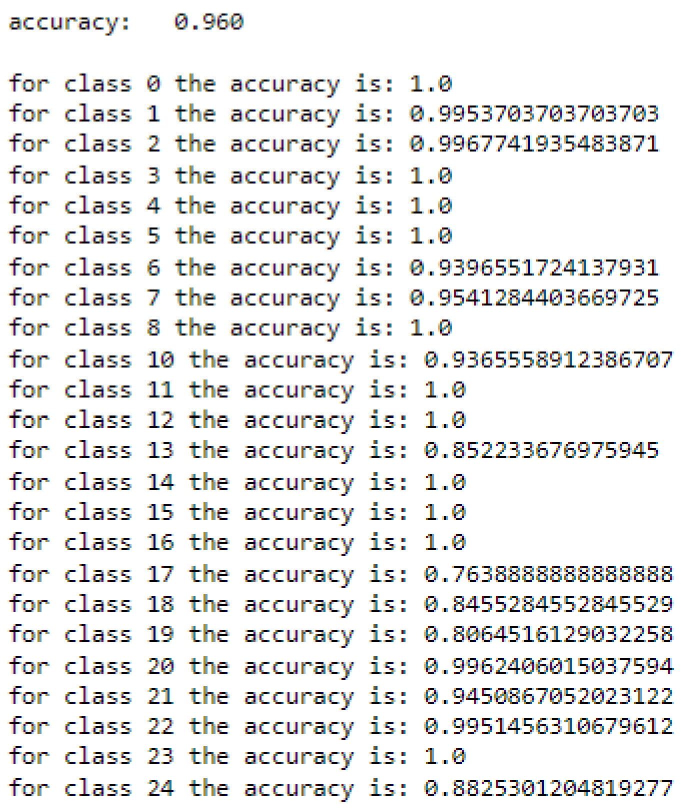
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 leaky_re_lu (LeakyReLU)     (None, 28, 28, 32)        0         
                                                                 
 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496     
                                                                 
 leaky_re_lu_1 (LeakyReLU)   (None, 14, 14, 64)        0         
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 7, 7, 128)         73856     
                                                                 
 leaky_re_lu_2 (LeakyReLU)   (None, 7, 7, 128)         0         
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)        0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 2048)              0         
                                                                 
 dense (Dense)               (None, 128)               262272    
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 leaky_re_lu_3 (LeakyReLU)   (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 25)                3225      
                                                                 
=================================================================
Total params: 358,169
Trainable params: 358,169
Non-trainable params: 0
_________________________________________________________________
```

Figure 1 The Architecture of the CNN model



True \ Predicted	0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	331	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	430	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	309	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	245	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	498	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	327	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	20	416	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	3	0	310	0	0	0	0	0	0	0	0	0	0	0	0	0	18
11	0	0	0	0	0	0	0	0	0	0	209	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	394	0	0	0	0	0	0	0	0	0	0	0	0
13	21	0	0	0	0	0	0	0	0	0	0	0	248	0	0	1	0	0	0	21	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	347	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	110	0	0	18	16	0	0	0
18	0	0	0	0	21	0	0	17	0	0	0	0	0	0	0	0	0	208	0	0	0	0	0	0
19	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	200	0	0	0	32	0
20	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	265	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	327	19	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	205	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	267	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	293

Figure 2 Confusion matrix



```
accuracy:   0.960

for class 0 the accuracy is: 1.0
for class 1 the accuracy is: 0.9953703703703703
for class 2 the accuracy is: 0.9967741935483871
for class 3 the accuracy is: 1.0
for class 4 the accuracy is: 1.0
for class 5 the accuracy is: 1.0
for class 6 the accuracy is: 0.9396551724137931
for class 7 the accuracy is: 0.9541284403669725
for class 8 the accuracy is: 1.0
for class 10 the accuracy is: 0.9365558912386707
for class 11 the accuracy is: 1.0
for class 12 the accuracy is: 1.0
for class 13 the accuracy is: 0.852233676975945
for class 14 the accuracy is: 1.0
for class 15 the accuracy is: 1.0
for class 16 the accuracy is: 1.0
for class 17 the accuracy is: 0.7638888888888888
for class 18 the accuracy is: 0.8455284552845529
for class 19 the accuracy is: 0.8064516129032258
for class 20 the accuracy is: 0.9962406015037594
for class 21 the accuracy is: 0.9450867052023122
for class 22 the accuracy is: 0.9951456310679612
for class 23 the accuracy is: 1.0
for class 24 the accuracy is: 0.8825301204819277
```

Figure 3 The accuracy per class for the CNN model

Step	**Function**	**Purpose**
1	Apply thresholding	To remove noise
2	Median filter	To remove noise
3	Adaptive Histogram Equalisation	To enhance features
4	Contrast stretching	To adjust the contrast of the images
5	Remove small object	To remove noise
6	White Tophat filtering	To extract the elements and features
7	Binary Dilation	To remove noise
8	Remove Small Objects	To determine the cropping size as a sanity check

Table 1 Steps of image pre-processing

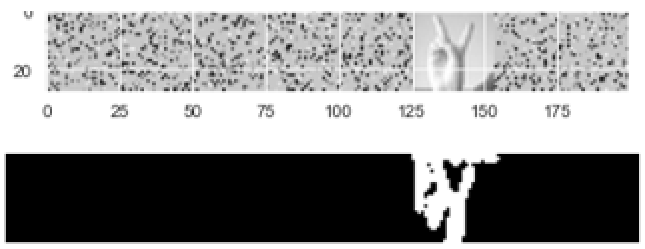


Figure 3 Comparison of the original image and processed image of the 9586th Image

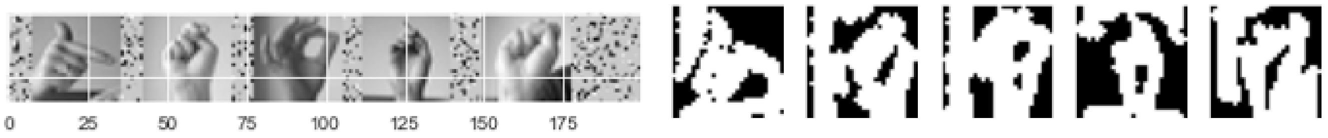


Figure 4 The Comparisons of the original images and the processed images