Fingerprint Recognition Using Deep Learning

Introduction

Biometrics technology is a way of personal identification using the phycological or the behavioural characteristics. Driven from the security needs for the electronically connected world, biometrics identification compensates some weaknesses of token- and knowledge-based identification in terms of loss, duplication and theft. Biometrics traits contain iris pattern, retinal scan, fingerprints, voice and signature. Fingerprint is one of the dominant biometrics traits that keeps spreading out because its uniqueness, acceptability, and low cost.

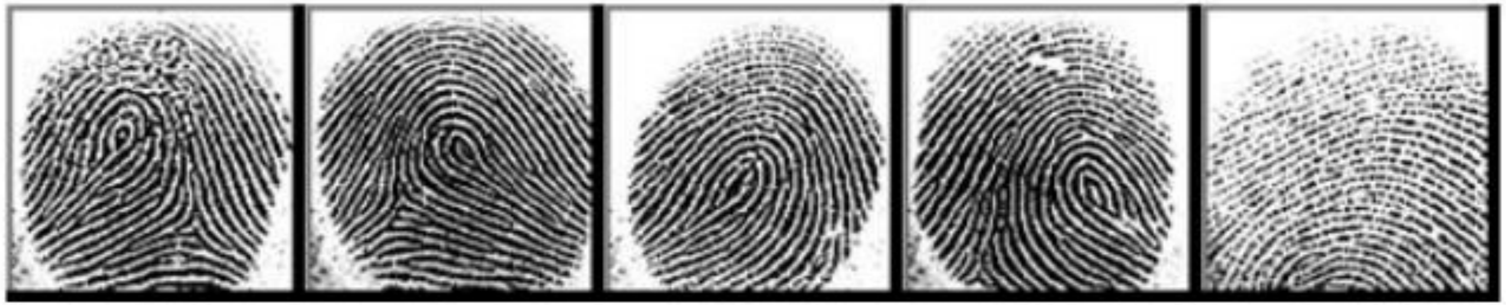
Abstract

Fingerprints play a significant role in many sectors. Nowadays, fingerprints are used for identification purposes in criminal investigations. They are also used as an authentication method since they are considered more secure than passwords. Fingerprint sensors are already widely deployed in many devices, including mobile phones and smart locks. Criminals try to compromise biometric fingerprint systems by purposely altering their fingerprints or entering fake ones. Therefore, it is critical to design and develop a highly accurate fingerprint classification. However, some fingerprint datasets are small and not sufficient to train a neural network. In this work a large Sokoto Coventry Fingerprint Dataset (SOCOFing), which contains 55,273 fingerprint images was used to create two datasets(SubjectID and Fingername), then build two convolutional neural networks and train the models with corresponding dataset. Finally, evaluate the method with a randomly picked fingerprint, when and only when both the predictions of ID and fingers name are correct then we can say the proposed method works.

Sokoto Coventry Fingerprint Dataset (SOCOFing)

The SOCOFing dataset, is made up of 6,000 real images belonging to 600 African subjects of age 18 years or older. Ten fingerprints from each subject have been captured. Three different levels of alterations for obliteration, central rotation, and z-cut have been applied to get synthetically altered versions of the real fingerprints. STRANGE toolbox is a framework that is used to generate a synthetic alteration on the fingerprint images. Easy, medium and hard parameter settings in the STRANGE toolbox have been applied to the images to produce the alterations. A total number of 17,934 images have been generated with easy parameter settings, 17,067 images with medium parameter settings, and 14,272 images with hard parameter settings. Therefore, the SOCOFing dataset contains 55,273 images in total .

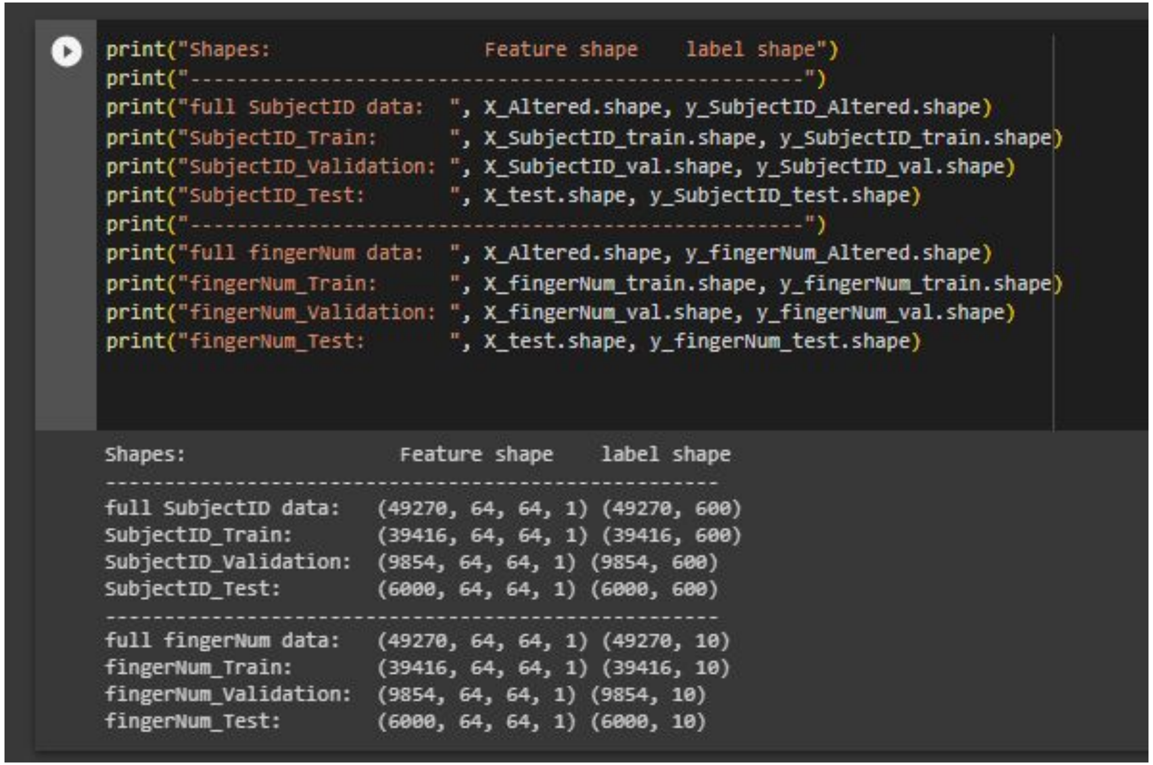
Sokoto Coventry Fingerprint Dataset (SOCOFing)


Sample illustration of five left hand fingerprints belonging to the same subject.

Preprocessing

The data preprocessing step is essential before feeding the data into the model. The purer the data is, the better the model learns. In this project, different data preprocessing techniques were applied to produce better-quality data. First, all the images in both datasets have been converted to grayscale images and have been resized into 32* 32(width * height). Smaller-sized images result in a faster training process. Then, all the data have been shuffled to avoid any chance of overfitting. and to get unique data for both training , validating and testing sets.

Preprocessing



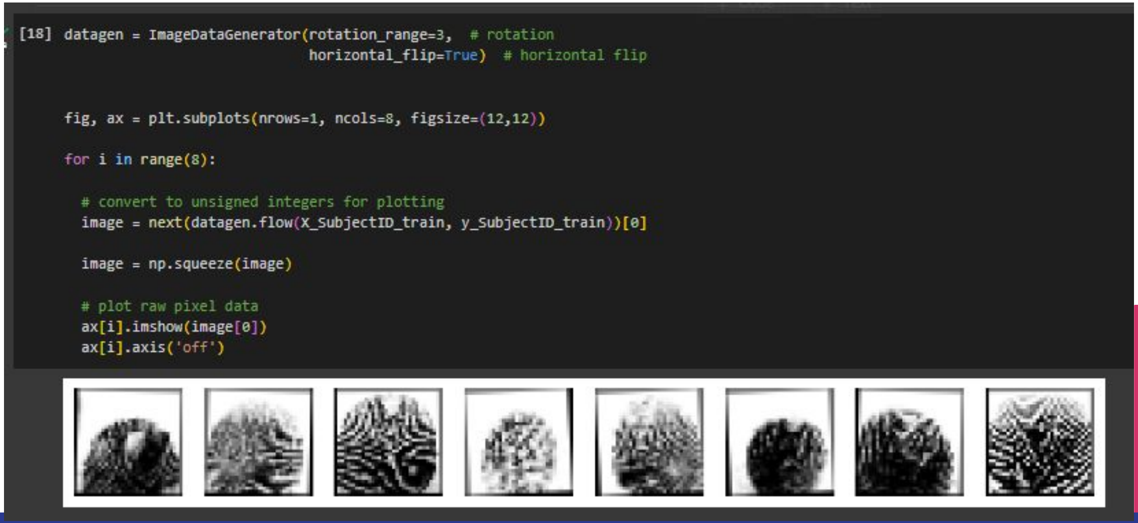
```
print("Shapes:                  Feature shape    label shape")
print("---------------------------------------------------")
print("full SubjectID data:  ", X_Altered.shape, y_SubjectID_Altered.shape)
print("SubjectID_Train:      ", X_SubjectID_train.shape, y_SubjectID_train.shape)
print("SubjectID_Validation: ", X_SubjectID_val.shape, y_SubjectID_val.shape)
print("SubjectID_Test:       ", X_test.shape, y_SubjectID_test.shape)
print("---------------------------------------------------")
print("full fingerNum data:  ", X_Altered.shape, y_fingerNum_Altered.shape)
print("fingerNum_Train:      ", X_fingerNum_train.shape, y_fingerNum_train.shape)
print("fingerNum_Validation: ", X_fingerNum_val.shape, y_fingerNum_val.shape)
print("fingerNum_Test:       ", X_test.shape, y_fingerNum_test.shape)
```

```
Shapes:                  Feature shape    label shape
---------------------------------------------------
full SubjectID data:    (49270, 64, 64, 1) (49270, 600)
SubjectID_Train:        (39416, 64, 64, 1) (39416, 600)
SubjectID_Validation:   (9854, 64, 64, 1) (9854, 600)
SubjectID_Test:         (6000, 64, 64, 1) (6000, 600)
---------------------------------------------------
full fingerNum data:    (49270, 64, 64, 1) (49270, 10)
fingerNum_Train:        (39416, 64, 64, 1) (39416, 10)
fingerNum_Validation:   (9854, 64, 64, 1) (9854, 10)
fingerNum_Test:         (6000, 64, 64, 1) (6000, 10)
```

SOCOFing Dataset After Being Divided into Training , Validating Testing Sets

Augmentation

Data augmentation is a technique used in machine learning to increase the size of a dataset by creating new data samples from existing ones. This can be done by applying various transformations to the original data, such as flipping, rotating, scaling, or adding noise. Data augmentation can help improve the performance of a machine learning pipeline in several ways. First, it can help prevent overfitting by providing more diverse training data for the model to learn from. By creating new variations of the original data, the model is exposed to a wider range of scenarios and is better equipped to handle new, unseen data.



```
[18] datagen = ImageDataGenerator(rotation_range=3,  # rotation
                                  horizontal_flip=True)  # horizontal flip


     fig, ax = plt.subplots(nrows=1, ncols=8, figsize=(12,12))

     for i in range(8):

       # convert to unsigned integers for plotting
       image = next(datagen.flow(X_SubjectID_train, y_SubjectID_train))[0]

       image = np.squeeze(image)

       # plot raw pixel data
       ax[i].imshow(image[0])
       ax[i].axis('off')
```

Convolutional Neural Network (CNN)

The CNN is made up of multiple consecutive layers. Each layer is formed of a set of artificial neurons. A neuron is a function that takes multiple inputs, calculates the inputs' weighted sum, and outputs an activation value

Convolutional Neural Network (CNN)

The CNN can be divided into two main parts:

- The hidden layers

The CNN first puts the input image into a series of hidden layers. The hidden layers are responsible for extracting the features. The layers are organized into 3 dimensions: height, width, and depth. The neurons in one layer do not connect to all the neurons of the previous layer. The network performs a series of convolution and pooling operations which result in detecting the features. Convolution is referred to as a mathematical combination of two functions to produce a third function. Convolution is usually associated with several attributes, which are:

Convolutional Neural Network (CNN)

1. Kernel size: Kernel is a filter that slides over the input. At each location, matrix multiplication is made, and the result is being summed onto the feature map. Common sizes for a kernel are 3x3 and 5x5.

2. Stride: Stride is a value that determines the step the convolution filter moves each time. If stride is equal to one, the filter moves pixel by pixel on the input. By increasing the stride, fewer overlap chances between cells may occur.

3. Padding: Since the feature map that is being generated after each layer is less in size than the input, padding can be performed by adding zeros to the input frame of the matrix. Padding helps in preventing the feature map from shrinking.

Convolutional Neural Network (CNN)

- The fully connected layers

  The fully connected layers work as a classifier. They consist of few connected layers where the neurons in one layer are connected to all the neurons in the previous layer. It gets the extracted features, one-dimensional data, as an input. Based on the activation map of the final convolution layer, the output of the classification layer is a set of values that indicate how likely the image belongs to a class.

Proposed Model

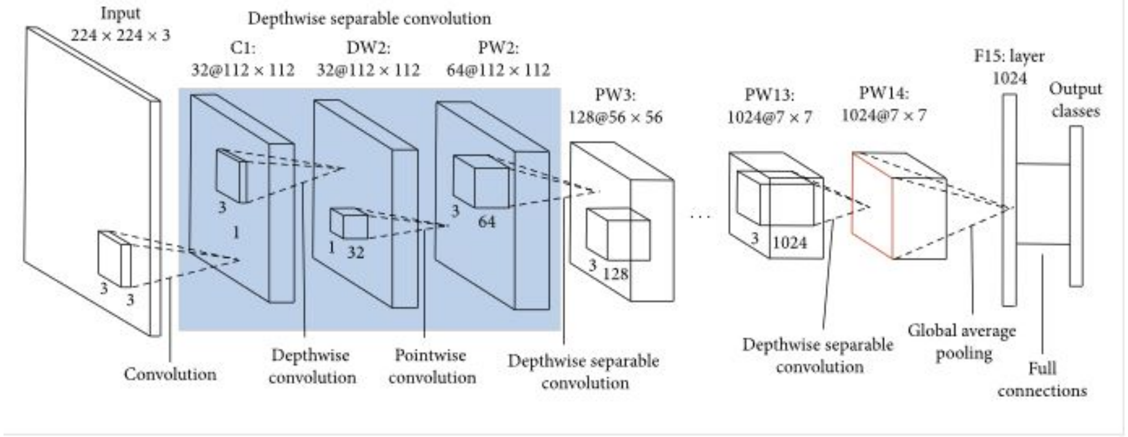
The proposed CNN model has three convolutional layers. Two layers have a kernel size of 5x5 and the last one have a kernel size of 3x3. No padding has been added. The output of every convolutional layer is shaped by the Rectifier Linear Model (ReLU) function. Max pooling is applied for the three layers with a size of 2x2. The convolutional layers are followed by two fully connected layers. A SoftMax activation function is then applied. The SoftMax function is used to map the output of the network to a probability distribution. Four probability values will be generated associated with each class. The biggest probability indicates to which class the image most likely belongs

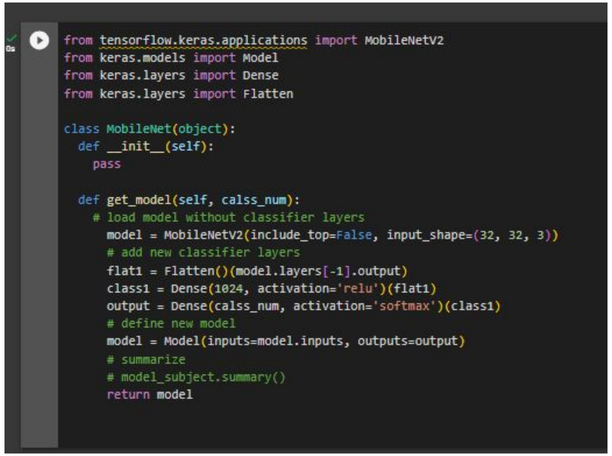
Models Architecture: MobileNetV2

MobileNetV2 is a convolutional neural network architecture that is designed for mobile and embedded devices with limited computational resources. It was developed by Google and released in 2018 as an improvement over the original MobileNet architecture.

MobileNetV2 uses a combination of depthwise separable convolutions and linear bottleneck layers to reduce the number of parameters and computations required by the model, while still maintaining high accuracy. The depthwise separable convolutions split the standard convolution operation into two separate operations: a depthwise convolution that applies a single filter to each input channel, followed by a pointwise convolution that applies a 1x1 filter to combine the output channels. The linear bottleneck layers further reduce the number of channels and computations by applying a 1x1 convolution followed by a 3x3 depthwise convolution and another 1x1 convolution.

MobileNetV2





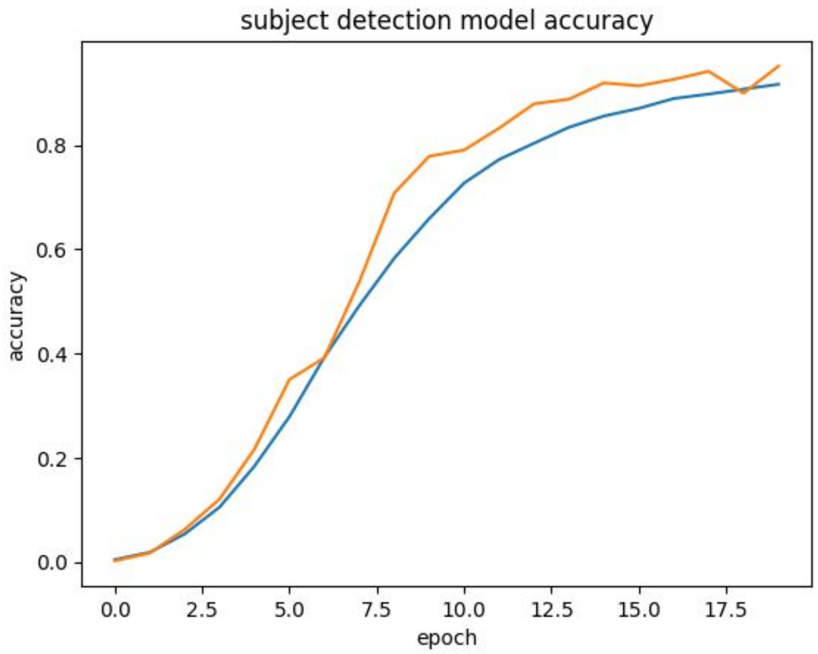
```python
from tensorflow.keras.applications import MobileNetV2
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten

class MobileNet(object):
  def __init__(self):
    pass

  def get_model(self, calss_num):
    # load model without classifier layers
      model = MobileNetV2(include_top=False, input_shape=(32, 32, 3))
      # add new classifier layers
      flat1 = Flatten()(model.layers[-1].output)
      class1 = Dense(1024, activation='relu')(flat1)
      output = Dense(calss_num, activation='softmax')(class1)
      # define new model
      model = Model(inputs=model.inputs, outputs=output)
      # summarize
      # model_subject.summary()
      return model
```

Training Process

The training set in the SOCOFing dataset, which consists of 49270 samples, has been divided into batches. A batch size refers to the number of images to run through before adjusting the weights of the neurons. Splitting the dataset into batches leads to a faster training process since the weights are getting updated after each propagation. The proposed model has a batch size of 32. The loss function used in the proposed model is Categorical Cross Entropy.

Adam is the replacement optimization algorithm that has been used to update the weights of the individual neurons. It can handle sparse gradients on noisy problems. The learning rate has been set to 0.0001. The learning rate defines the step size at each iteration while moving toward a minimum of a loss function.

Learning Curves



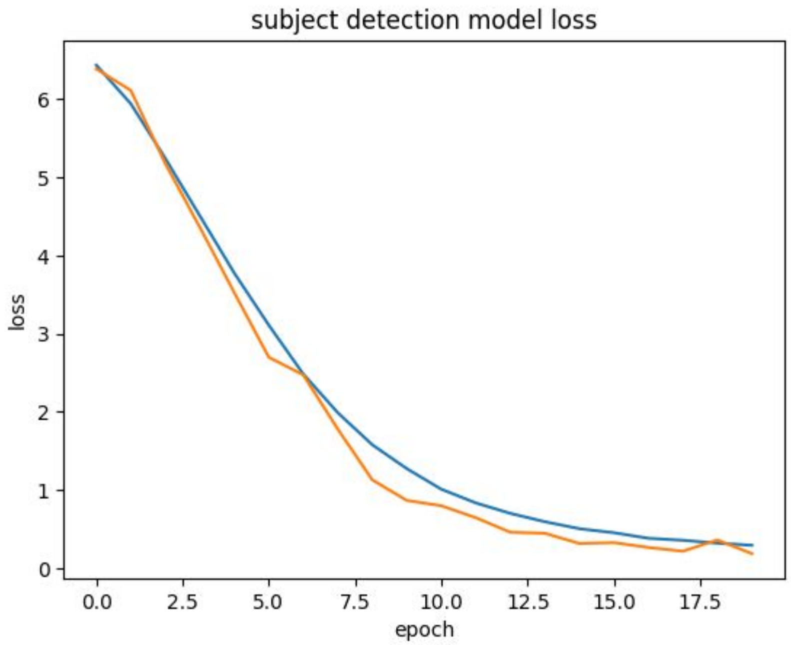
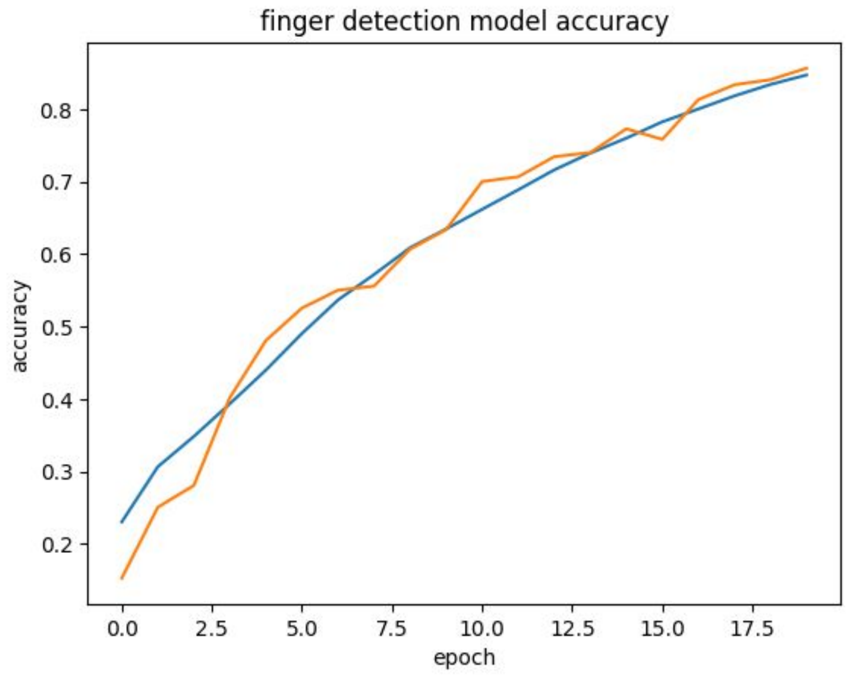


Figure shows that as the number of epochs increases, the loss value gets decreased.

Learning Curves



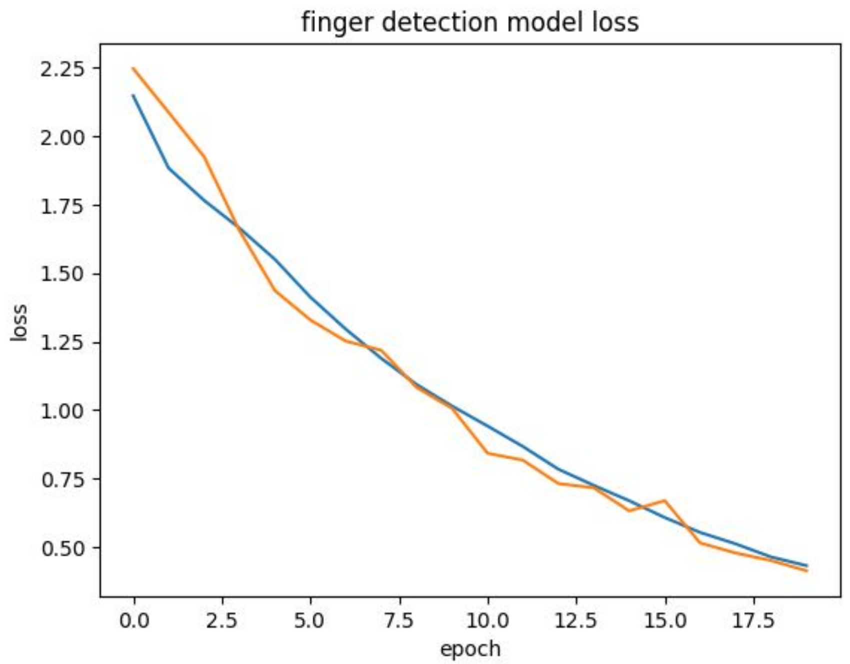


Figure shows that as the number of epochs increases, the loss value gets decreased.

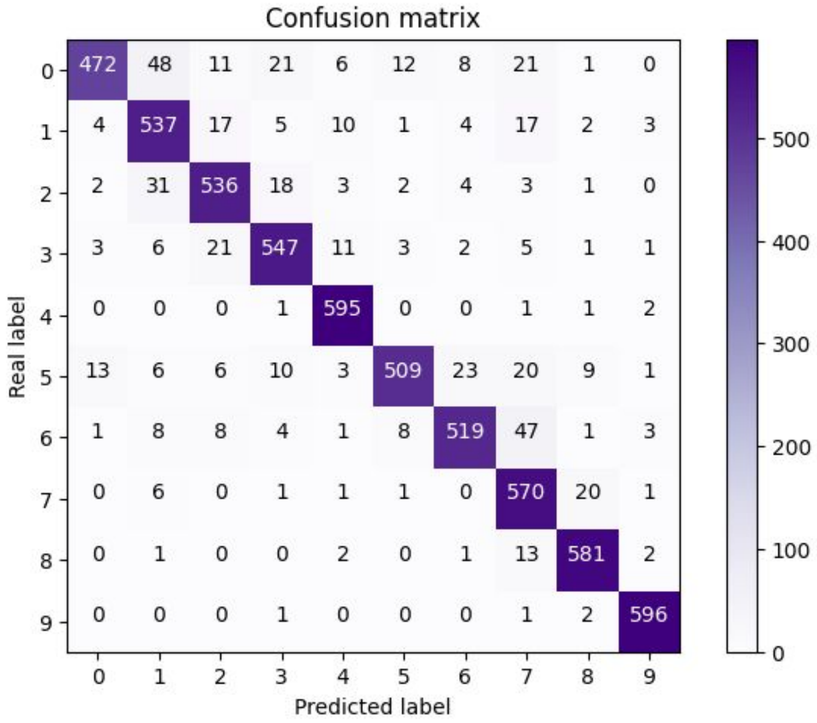
Evaluation

After training the model, it should be tested to evaluate its accuracy. Accuracy results indicate how the model is performed on unseen data. Testing is done by comparing the predicted class with the actual class the image belongs to.Both model has been tested on the testing dataset, which consists of 6000 samples.

Figure demonstrate the confusion matrix for finger number recognition model

Subject recognition accuracy: 98.05 %

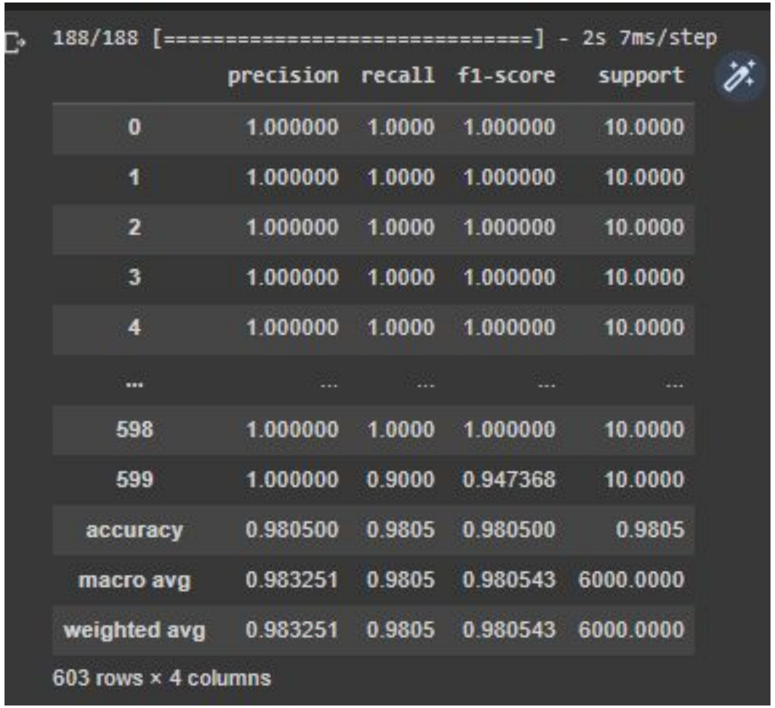
Finger recognition accuracy: 91.03 %



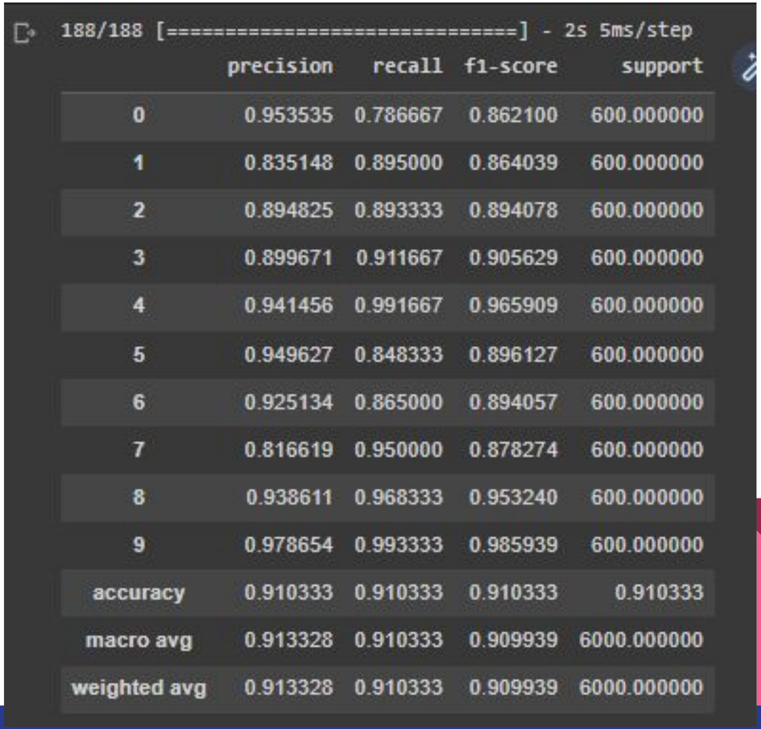
Real label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	472	48	11	21	6	12	8	21	1	0
1	4	537	17	5	10	1	4	17	2	3
2	2	31	536	18	3	2	4	3	1	0
3	3	6	21	547	11	3	2	5	1	1
4	0	0	0	1	595	0	0	1	1	2
5	13	6	6	10	3	509	23	20	9	1
6	1	8	8	4	1	8	519	47	1	3
7	0	6	0	1	1	1	0	570	20	1
8	0	1	0	0	2	0	1	13	581	2
9	0	0	0	1	0	0	0	1	2	596

Evaluation

Figure demonstrate the classification report of finger number and subject ID recognition models. 10 number of fingers and 600 subjects respectively



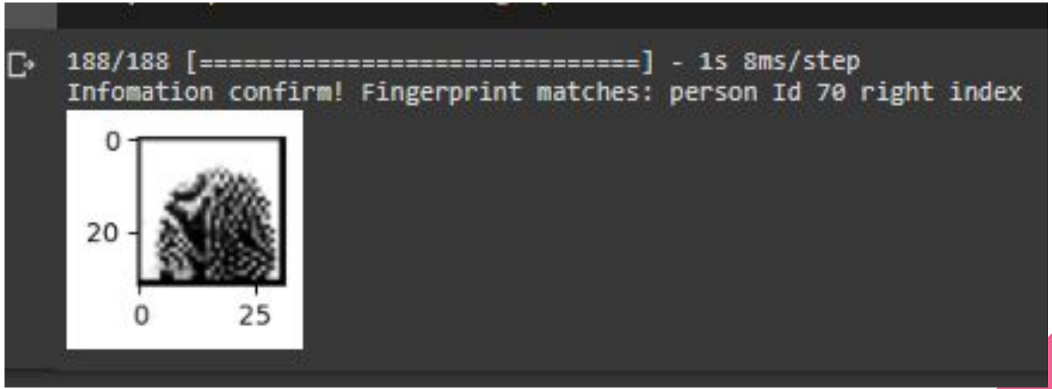
```
188/188 [==============================] - 2s 7ms/step
              precision  recall  f1-score    support
0              1.000000  1.0000  1.000000    10.0000
1              1.000000  1.0000  1.000000    10.0000
2              1.000000  1.0000  1.000000    10.0000
3              1.000000  1.0000  1.000000    10.0000
4              1.000000  1.0000  1.000000    10.0000
...                 ...     ...       ...        ...
598            1.000000  1.0000  1.000000    10.0000
599            1.000000  0.9000  0.947368    10.0000
accuracy       0.980500  0.9805  0.980500     0.9805
macro avg      0.983251  0.9805  0.980543  6000.0000
weighted avg   0.983251  0.9805  0.980543  6000.0000
603 rows × 4 columns
```



```
188/188 [==============================] - 2s 5ms/step
              precision    recall  f1-score      support
0              0.953535  0.786667  0.862100   600.000000
1              0.835148  0.895000  0.864039   600.000000
2              0.894825  0.893333  0.894078   600.000000
3              0.899671  0.911667  0.905629   600.000000
4              0.941456  0.991667  0.965909   600.000000
5              0.949627  0.848333  0.896127   600.000000
6              0.925134  0.865000  0.894057   600.000000
7              0.816619  0.950000  0.878274   600.000000
8              0.938611  0.968333  0.953240   600.000000
9              0.978654  0.993333  0.985939   600.000000
accuracy       0.910333  0.910333  0.910333     0.910333
macro avg      0.913328  0.910333  0.909939  6000.000000
weighted avg   0.913328  0.910333  0.909939  6000.000000
```

Biometric Pipeline

Finally, randomly pick a fingerprint from test data to predict both its Id and finger name. When both the predictions are correct, then we can say this method works great with two models. Then we verify a person's finger with his ID and the finger name.



```
188/188 [==============================] - 1s 8ms/step
Infomation confirm! Fingerprint matches: person Id 70 right index
```

Conclusion

The importance of classifying fingerprints increases every day. Fingerprint biometric systems face several threats that may lead to incorrect identification or authentication. Criminals tend to alter their fingerprints by making scars or surgical procedures to evade their identity. Hackers also try to adopt another individual’s identity by creating a fake fingerprint. This presentation proposed a CNN model that recognize finger number and subject based on fingerprint images. Sokoto Coventry Fingerprint Dataset (SOCOFing) was used to train and test the model. The first model achieved a testing accuracy of 91.033% in classifying the number of finger. The second model was able to recognize the subject id by the fingerprint image. An accuracy result of 98.05% was gained by second model.