Smart Signature Verification Using Computer Vision

A

Project Report

Submitted for the partial fulfillment

Of B.Tech Degree In

INFORMATION TECHNOLOGY

By

Shivansh Purwar [1805213054]

Abhishek Kumar Batham [1805213003]

Saumitra Dhar Dubey [1805213051]

Kaushlendra Kumar [1805213026]

Under the supervision of

[Dr. Pawan Kumar Tiwari] **[Ms. Deepa Verma]**

Department of Computer Science & Engineering

Institute of Engineering & Technology, Lucknow**Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh.**

Contents

Declaration

We hereby declare that this submission is our own work and that, to the best of our belief and knowledge, it contains no material previously published or written by another person or material which to a substantial error has been accepted for the award of any degree or diploma of university or other institute of higher learning, except where the acknowledgement has been made in the text. The project has not been submitted by us at any other institute for requirement of any other degree.

Submitted by: - Date: - 14/05/2022

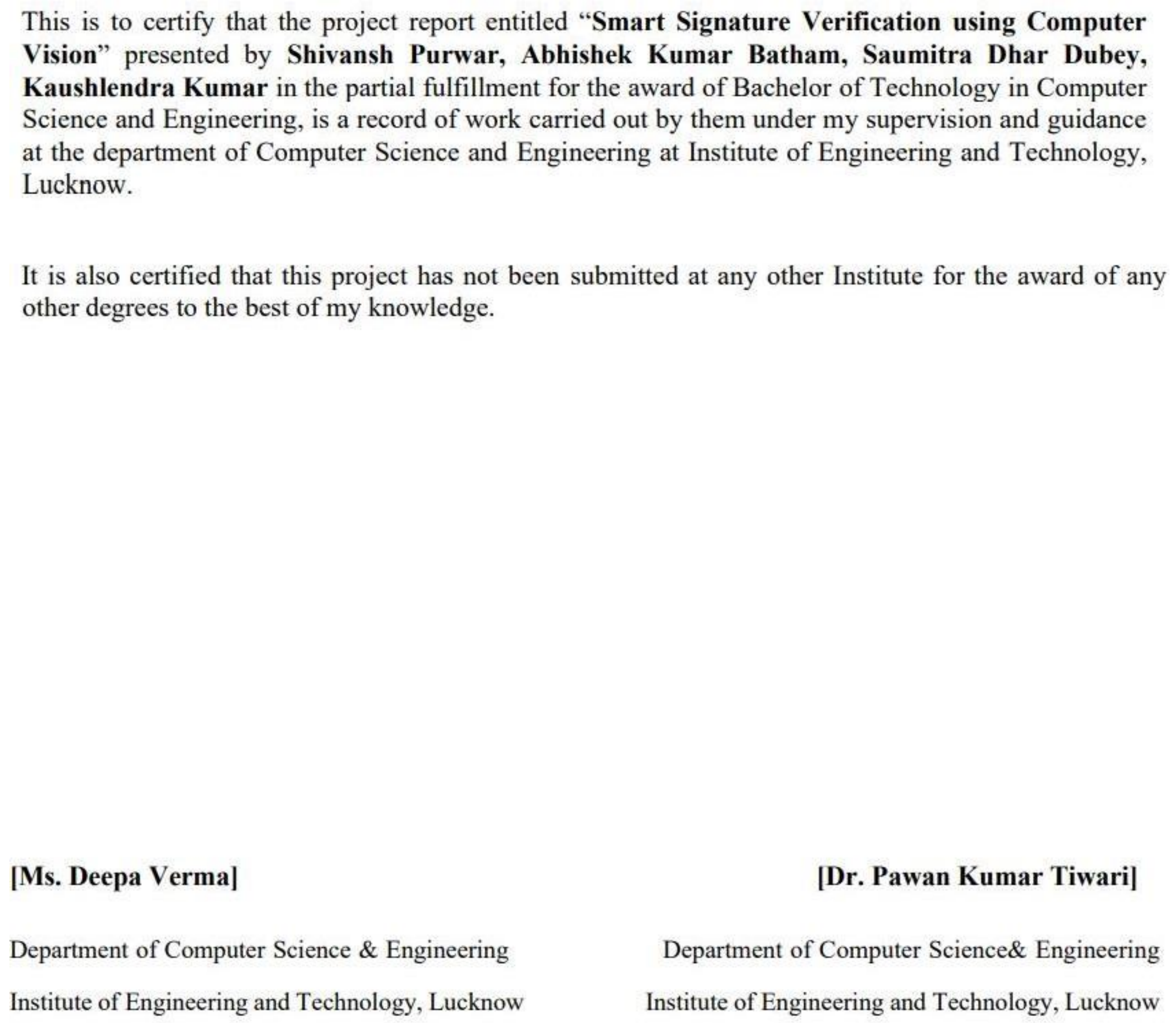
1. Name – Shivansh Purwar
   Roll No. – 1805213054
   Branch – Information Technology (IT)

2. Name – Abhishek Kumar Batham
   Roll No. – 1805213003
   Branch – Information Technology (IT)

3. Name – Saumitra Dhar Dubey
   Roll No. – 1805213051
   Branch – Information Technology (IT)

4. Name – Kaushlendra Kumar
   Roll No. – 1805213026
   Branch – Information Technology (IT)

Certificate



This is to certify that the project report entitled "**Smart Signature Verification using Computer Vision**" presented by **Shivansh Purwar, Abhishek Kumar Batham, Saumitra Dhar Dubey, Kaushlendra Kumar** in the partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering, is a record of work carried out by them under my supervision and guidance at the department of Computer Science and Engineering at Institute of Engineering and Technology, Lucknow.

It is also certified that this project has not been submitted at any other Institute for the award of any other degrees to the best of my knowledge.

[Ms. Deepa Verma]

Department of Computer Science & Engineering

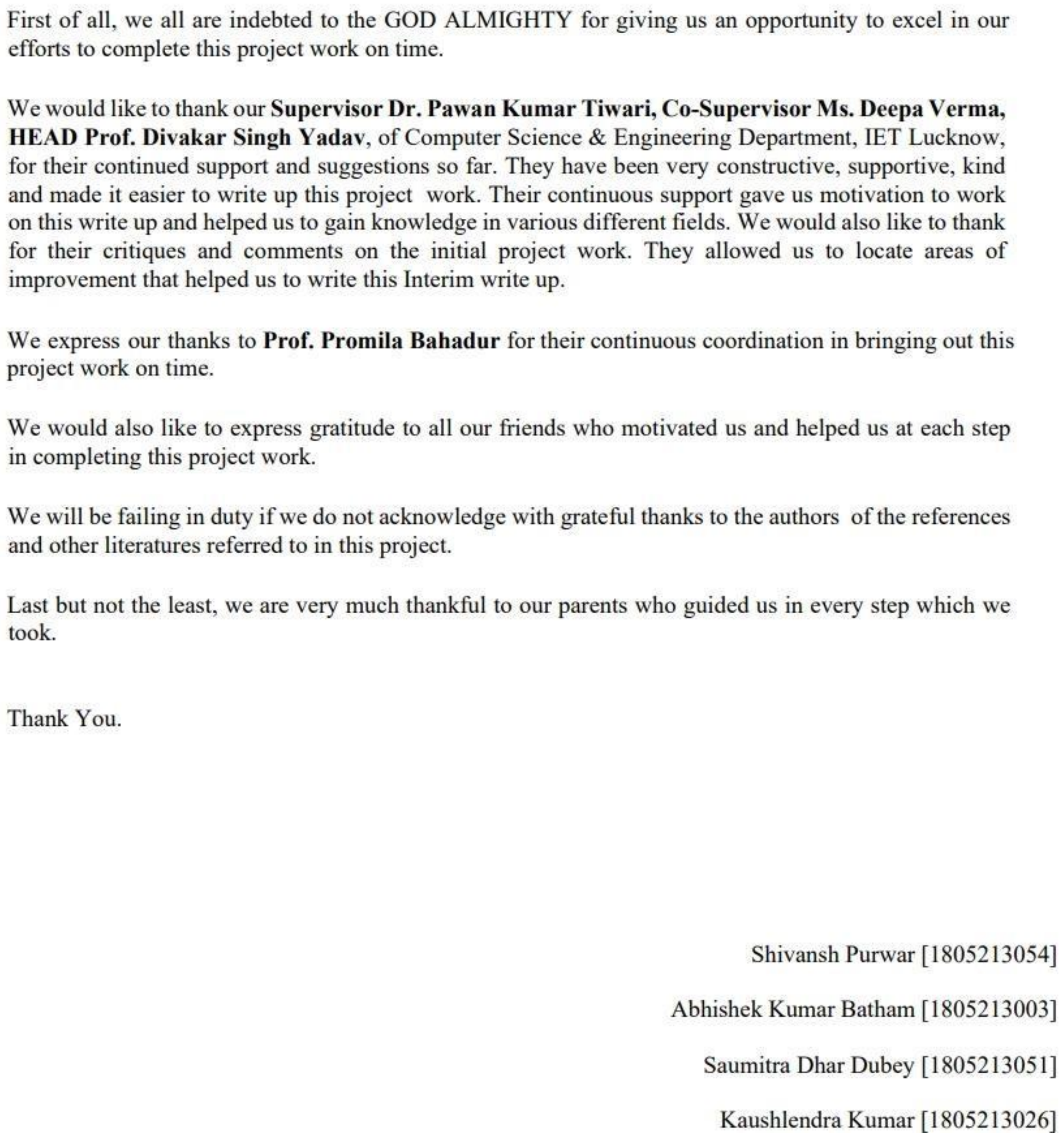
Institute of Engineering and Technology, Lucknow

[Dr. Pawan Kumar Tiwari]

Department of Computer Science& Engineering

Institute of Engineering and Technology, Lucknow

Acknowledgement



First of all, we all are indebted to the GOD ALMIGHTY for giving us an opportunity to excel in our efforts to complete this project work on time.

We would like to thank our **Supervisor Dr. Pawan Kumar Tiwari, Co-Supervisor Ms. Deepa Verma, HEAD Prof. Divakar Singh Yadav**, of Computer Science & Engineering Department, IET Lucknow, for their continued support and suggestions so far. They have been very constructive, supportive, kind and made it easier to write up this project work. Their continuous support gave us motivation to work on this write up and helped us to gain knowledge in various different fields. We would also like to thank for their critiques and comments on the initial project work. They allowed us to locate areas of improvement that helped us to write this Interim write up.

We express our thanks to **Prof. Promila Bahadur** for their continuous coordination in bringing out this project work on time.

We would also like to express gratitude to all our friends who motivated us and helped us at each step in completing this project work.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred to in this project.

Last but not the least, we are very much thankful to our parents who guided us in every step which we took.

Thank You.

Shivansh Purwar [1805213054]

Abhishek Kumar Batham [1805213003]

Saumitra Dhar Dubey [1805213051]

Kaushlendra Kumar [1805213026]

Abstract

Signature Verification remains an open research problem. The aim of automatic signature verification systems is to identify if a given signature is genuine or forged. This is a rigorous task, only in the offline screenplay, that uses scanned images of user's signatures, where the signing process of dynamic knowledge is not available. Many steps are taken for it and one of most famous of it to create another technology to defeat this problem, "Signature Verification".

Many scholars have created their own tools and technologies including many algorithms and explanations to defy this problem. In this project report, we have shown the accurate measures that how these problems can be handled, analyze the current advancements in this field, and the future scope of this project, many technologies have been developed from previous 5 to 10 years and more informative is that deep learning is used to learn the signatures of users.

The main aim of our project is to create a framework to identify Handwritten Signature fraud in checks and contracts in which when they are scanned, they can be easily pushed into an algorithm and after pushing it can verify whether the scanned document is authentic or a fraud. At the very first step, preprocessing of a. scanned image is done to isolate the signature part and to remove any spurious noise present.

List of Figures & Tables

Chapter 1
Introduction

Signature verification using Biometrics is used to verify hand written documentation for assignment in developing applications. The fact that the signature is extensively used as a means of personal verification emphasizes the need for an automatic verification.

Coping identification of automation of handwritten using verification of signature is in high time demand for the near future. In offline signature verification, the dynamic information of the signature is lost, and it is hard to design good feature extractors through which distinguish genuine signatures and skilled forgeries. One or more reasons for its affecting large number of people that process for collect signatures written by hands is not be invasive-, and people have to familiar with the use of signatures in their daily life.

We are proposing a novel formulation of the problem that including knowledge of the skill forgeries from a subset of users in a feature of learning a process, that aims to capture visual cues that have distinguish genuine signatures and forgeries regardless of the user. Conclude that , they are use to classify query signatures as genuine or forgeries. A substantial piece of eminent work has been done in the area of off-line signature verification over the past few decades. Verification can performed by two following ways Offline or Online based on the application. Online systems uses dynamic data of a signature captured when the signature is made. Offline systems work when the scanned image of a document.

We've created a framework to identify the Handwritten Signature fraud in checks and contracts in which, after being scanned, they can automatically be standardized and inserted into an algorithm that would verify if the analyzed document is authentic or a fraud. Initially, preprocessing of a scanned image is necessary to isolate the signature part and to remove any spurious noise present. The system is initially trained by using database of signatures obtained from those individuals whose signatures have to authenticate by the system. The details of preprocessing as well as the features depicted above are described throughout the discussion. Then neural network was used to verify and classify the signatures.

Signatures verification system aims to automatically discriminates if the biometric samples are indeed of a claimed individual. In sample way they are used to classify query signatures as they are authenticate or they are copied A great deal of period for 10 years work had done in area of offline signature verification over the past few years ago. Verification can performed offline and online based on the application. Online system use changing information of signature captured at the time when signatures is made. Offline systems work on the scanned image for gradual change in information of signature captured at the tie when signature is made. Offline systems work on the scanned image of document.

Chapter 2
Literature Survey

There exist no. of varieties of statistics strategies nowadays, costlier equipment are need to be installed by the Fingerprints and iris verification and therefore can't be used at day-to-day places. There’s good interest in authentication supported written signature verification system because it is that the most cost-effective thanks to manifest someone. Signatures are acknowledged by the Banks and Government bodies as a legal suggests that of authentication. Distinctive aspects of the signature are utilizes by the Signature verification technology to verify the identity of different people. Criminal consultants can't be utilized at each place and therefore there has been hefty effort towards developing computerized algorithms that might verify and manifest the individual’s identity [1,4].

Recently fingerprint verification used for iris scan only there adopted considerable interest in fingerprint verification used to detect online hand written signature verification utilizes in different aspects like banks forgeries. In order to verify and authenticate the individual identity we use the respective computer algorithm . The chapter aims to portray the present status along with the history of the problem to be addressed [1].

Modern forensic investigators signify the suspect by comparing different datasets. When signatures written at high speed it will led to forgeries. Compared with electronic signature verification which used patching of signature image by feature extraction and it is also easier to migrate from one place to another. And it also authenticates signatures which are no readable to human beings and helps to detect the signature belongs to that human being or not.

If we dig deeper into the matter and become a little more specific, we then tested the performance of the model inevery possible severe condition and we have also tested that how well the model can perform the process of transform learning for our CNN model to perform well. In order to identify the effect of the size of available data in accuracy, precision and recall, we finally merged the two datasets and performed 10- fold cross validation. Our report describes the virology field of medical science thoroughly as we all know that the structure of virus is ever changing so to find its cure and detect it is the most difficult task ever among any other pathogen found in human body.

However, the ultimate goal of our project is to simplify this task with higher precision an accuracy. The model is well-built only when the data provided to the model is accurate and of higher information value. Many a time the numerical large value of dataset only gets to loss of memory and does not provide any meaningful information to the model and the model collapses in the result of it or gets under fit or over fit, and both are highly undesirable in any of the cases. So, we have tried to give the numerical quantity of the dataset as well as the information value related to it as high as possible which in result allows our model to get an accuracy of about 99% which is very high enough to counter any such other test related to it [5].

hapter 3
Methodology

We solve the problem by matching the original signature of the user with signature on mandate form. First, we built models capable of extracting account number though which we extract the original signature from database. Next, we extract the signature from the mandate form. Next we build a model to match the original and questioned signature and give the confidence score in real time.

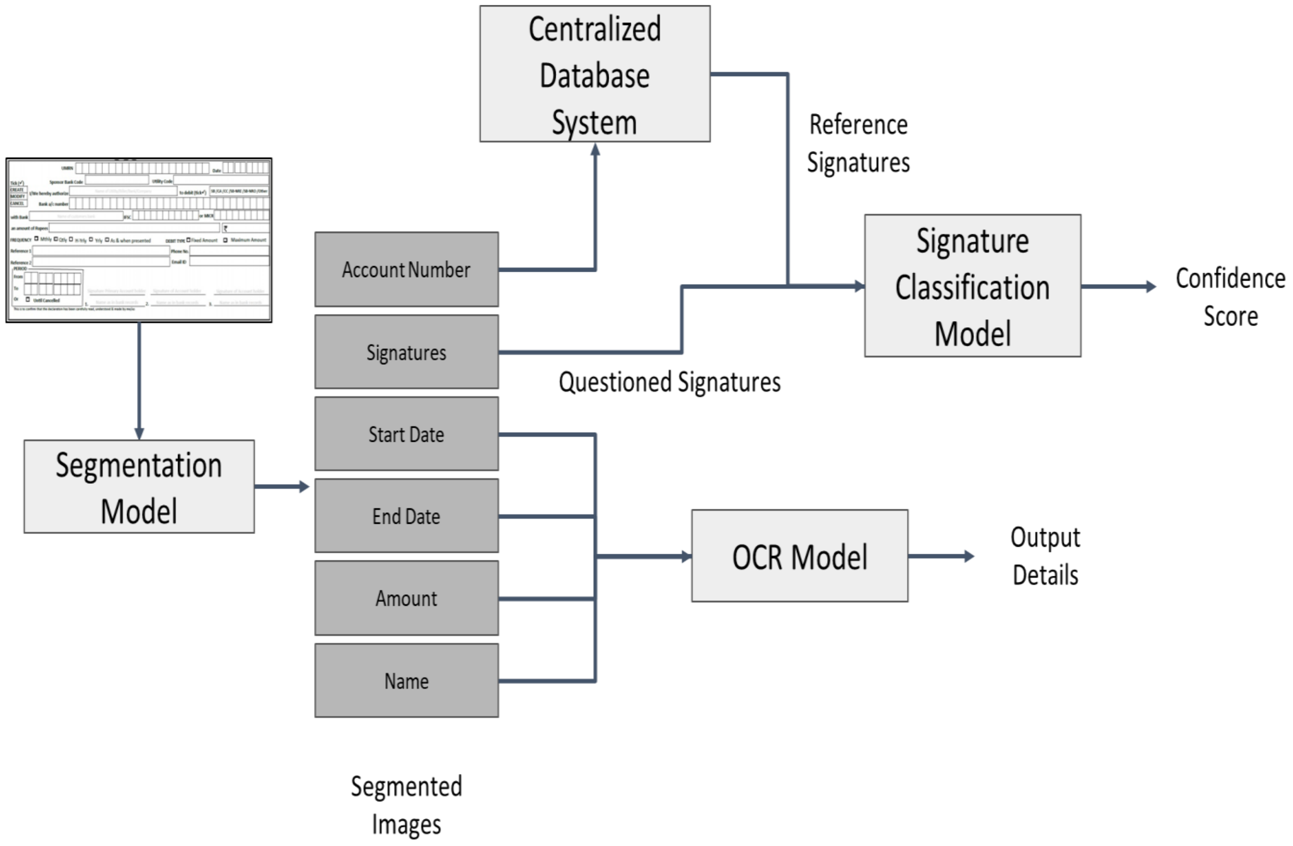


Fig.3.1: DFD Level-1

Through the above DFD level-1, and points given we can understand the working of our project.

- User triggers a request to the web app offline through the application user interface.
- User have to upload scanned cheque/mandate form and click on submit button.

- Segmentation & OCR model will extract different detail from document like Account Number, Questioned Signature, Account holder name etc.
- Request will be sent to centralized database to fetch the reference signature using Account Number fetch from the document.
- Reference signature and questioned signature goes through the signature classification model and output the confidence score.

<u>GUI (Graphical User Interface)</u>

The GUI, via which the person may be interacting with the net software are proven withinside the order of the appearance. The first web page or the beginning web page is proven below.

1. Click at the "Update Mandate" button and pick a record. The record is not anything however a scanned in addition to filled "Mandate Form".

2. Click at the Submit button. Refresh the web page to check every other form (clean is mandatory).

3. You can see the development bar moving. Once all of the code withinside the backend has been executed, the probability of two signature being matched gets displayed. It also displays the account number that has been fetched with the aid of OCR model.

DATASET

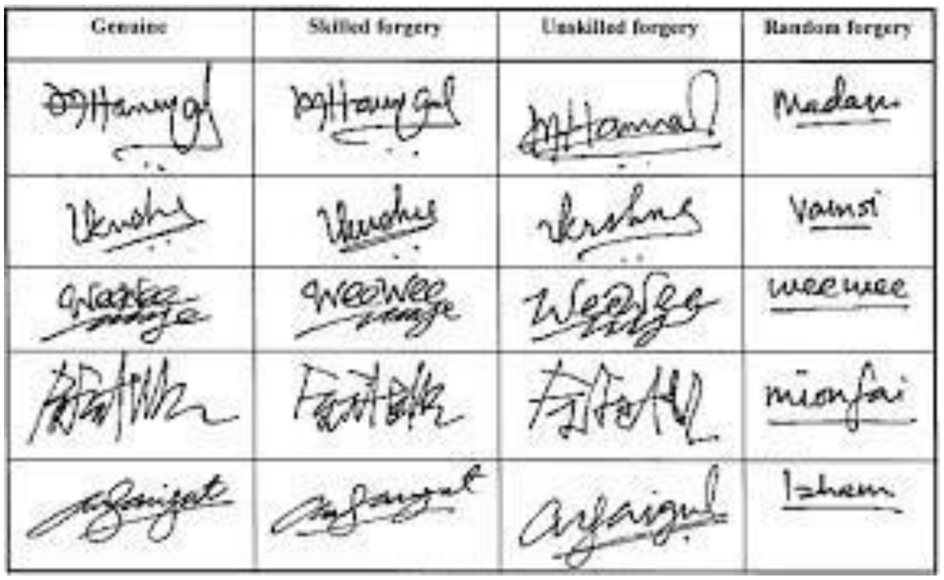

Genuine
Skilled forgery
Unskilled forgery
Random forgery

Fig.3.1.1: Sample images from the dataset

The dataset comprises of 28,000 labeled images in the training set, 3,500 labeled images in the development set, and 3,500 images in the test set. Each image in dataset is labeled as[8]

- Originals
- Forgery

MNIST Dataset

A MNIST dataset is a collection of datasets with integer between 0 to 9. Well-annotated media of signature is essential for training, testing, and validation of algorithms for the development of signature verification system[9].

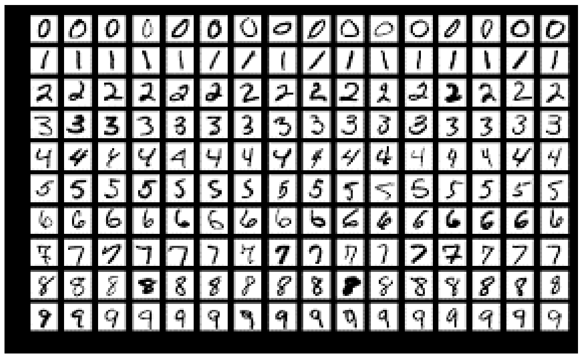


Fig.3.1.2: Sample images from the MNIST dataset>

Process of Verifying mandate form

This process includes four stages.

- Pre- processing – authenticating datasets in form of generalized algorithm.
- Detection stage – signature extraction from feature mapping stage.
- The last step signature classification step consists of implementing the Siamese neural network to predict the confidence score.

These stages are described using in a flowchart in Fig. 2:

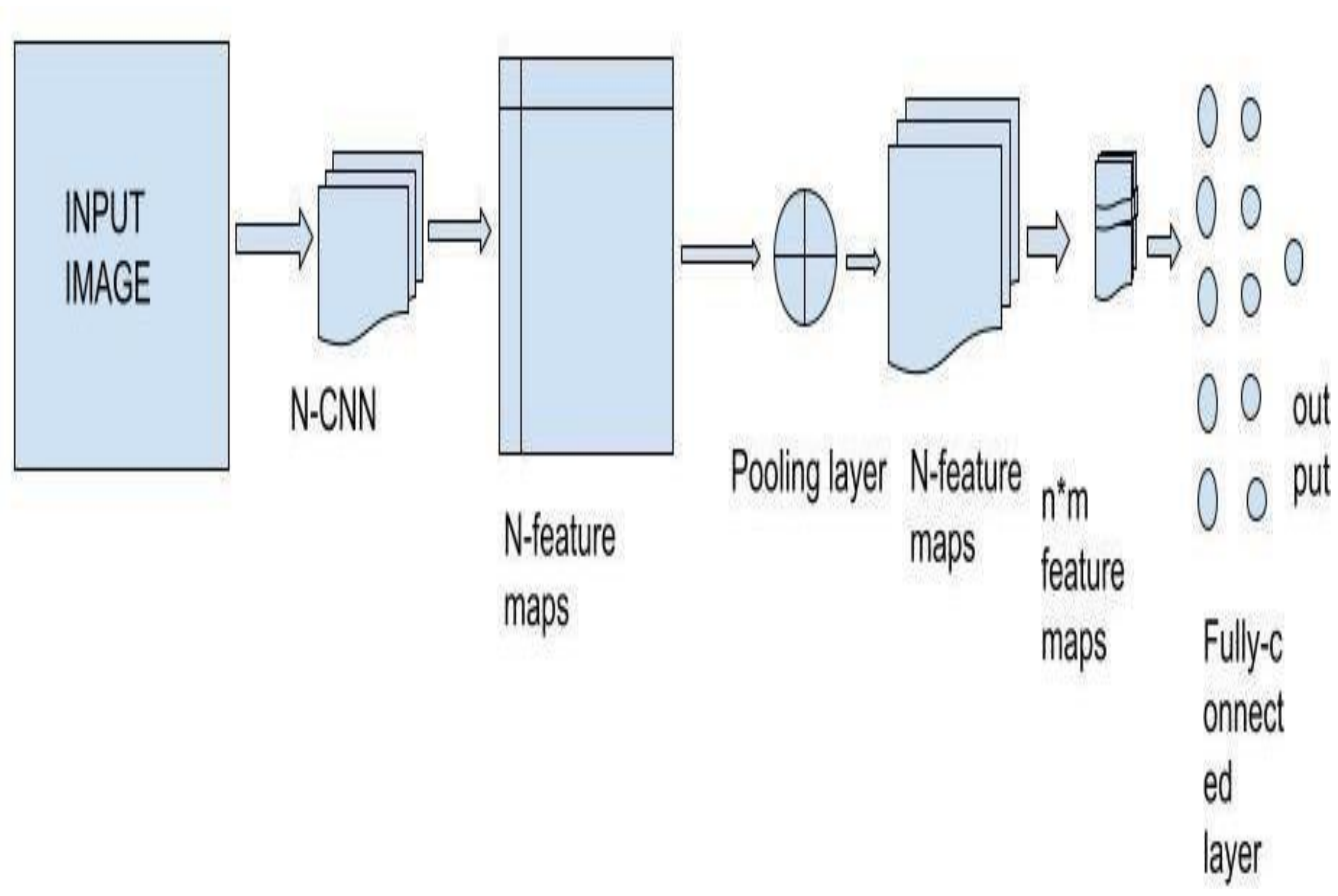


Fig.3.2: Process flow

Preprocessing

Pre-processing of the dataset is very important wherein have done the noise removal property adjustment part (angular rotation, resizing and exact position detection) and image augmentation.

- ***Normalization*** is done to remove illumination variations.
- ***Gray scaling*** is a process of converting a colored image input into an image whose pixel value depends on the intensity of light on the image. Gray scaling is done as colored images are difficult to process by an algorithm.
- ***Resizing*** is to remove the unnecessary parts of the image.
- ***Dilation*** is the morphological operation which increases the area and accentuate features.

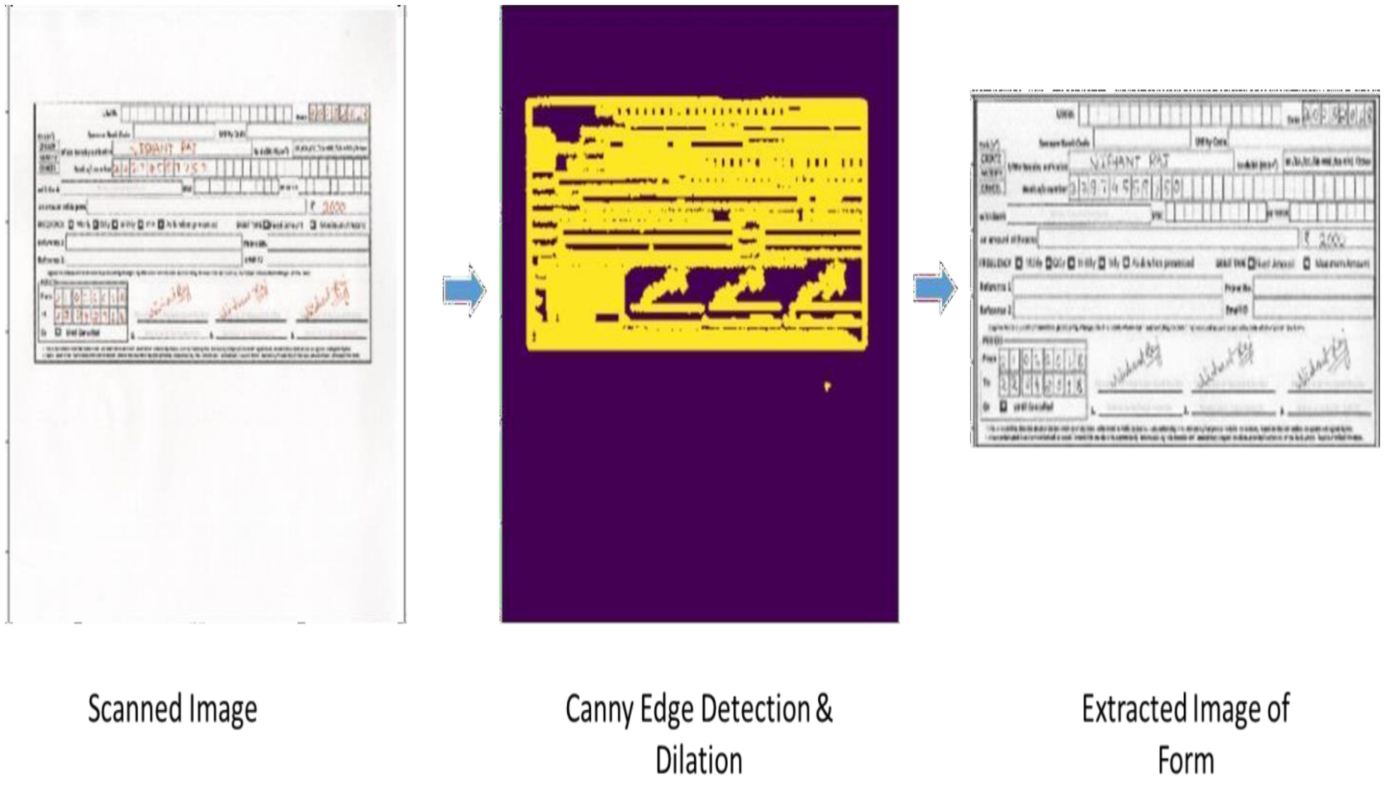
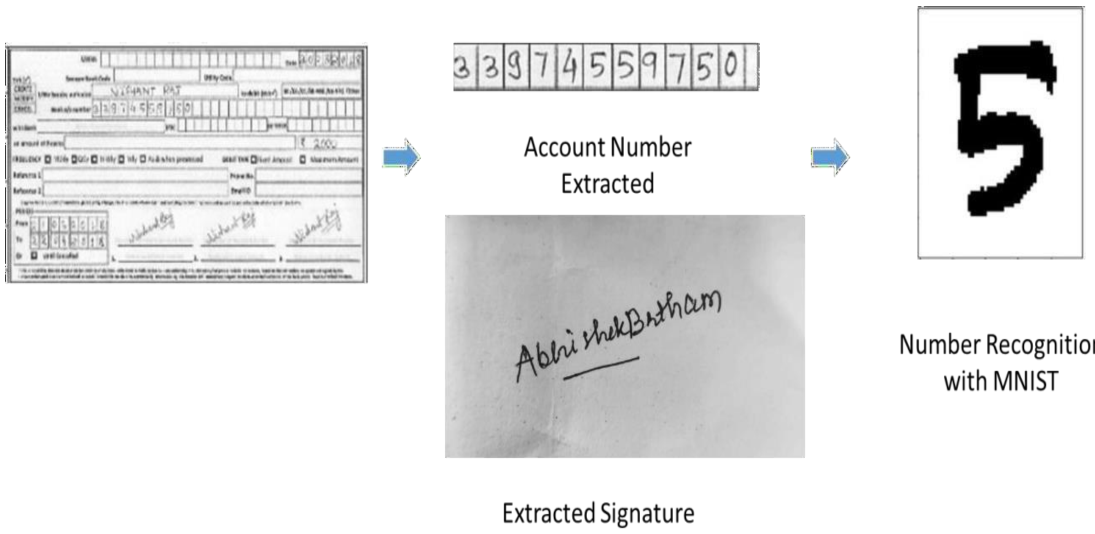


Fig. 3.2.1.1 Cheque Extraction



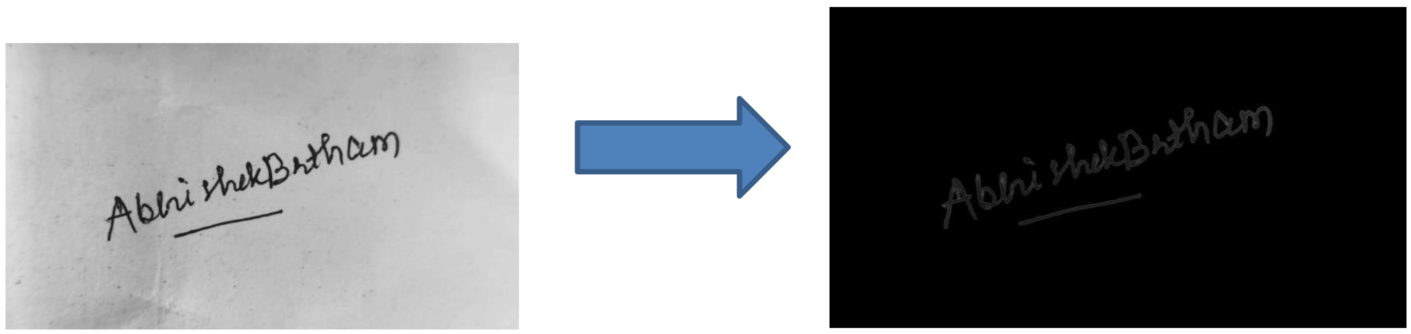
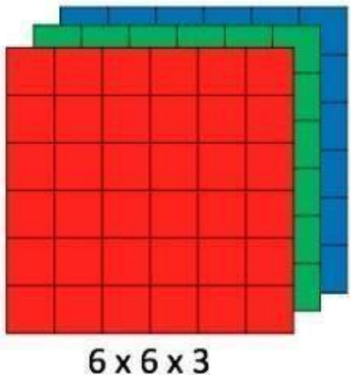


Fig. 3.2.1.2 Signature Extraction

<u>Account Number Recognition</u>

In this the system classifies each character in image into one of 0-9 integer. The training was done using CNN. The dataset was first split into training and test datasets, and then it was trained on the training set. MNIST dataset is used for training this model.

We can define convolution as the process, which consists of using a filter and sampling it over the image and hence used to extract only the desired information from the image while removing or we can say filtering the other with the use of kernel. If we say that image is represented in pixels in the form of a matrix then the filter is also represented as a matrix and moved through the entire image matrix covering the entire row at once and then increasing the column value by the desired unit each time, basically hovering it over to the image, this is also known as a stride. The next step typically involves the multiplication of the image as a matrix to the weighted matrix filter each time along the rows and incrementing the unit value of the column. When we reach the ending of a particular row, we hover the filter to the next row incrementing it by one.
When this operation is performed, we will get a filtered output image, which has the size that can be given by the formula: Output width = $(W - Fw + 2P) / Sw + 1$
Output Height = $(H - FH + 2P)/SH + 1$
W: width, H: height, FH: Filter Height, FW: Filter width, P: Padding



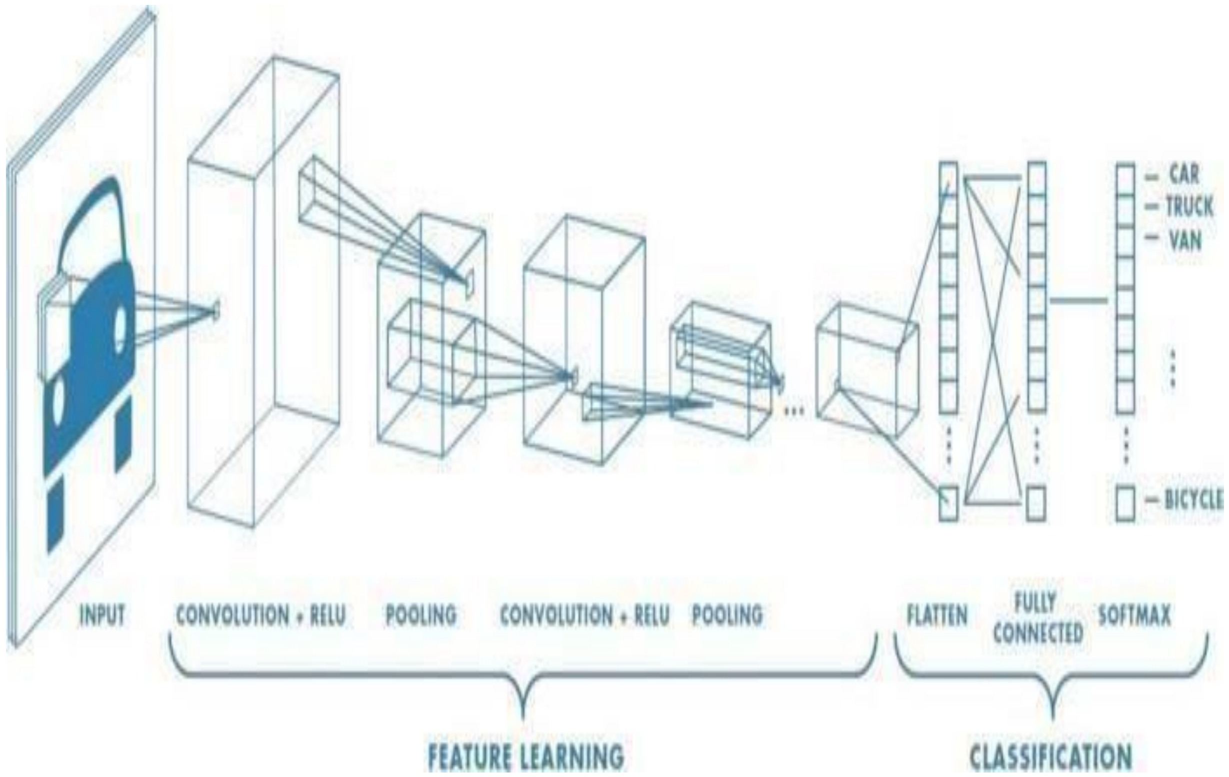


Fig.3.2.2.1.1: Neural network with many convolutional layers

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

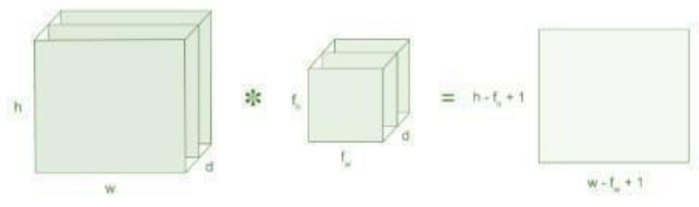
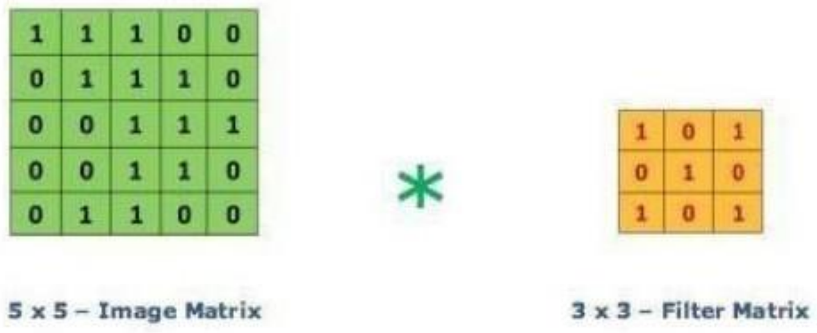
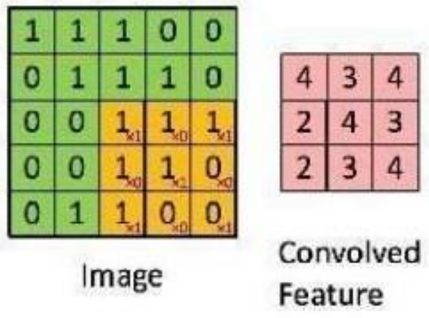


Fig.3.2.2.1.2 Image matrix multiplies kernel or filter matrix

Filter matrix 3 x 3 and 5 x 5 whose image pixels are 0,1 is taken into consideration as shown below.



Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called **"Feature Map"** as output shown in below :



1. **Stride** is the number of pixels shifts over the input matrix. When the stride is 1 then we mov filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a timean

2. **Padding** is the number of pixels added to an image when it is being processed by the kernel of a CNN. Valid padding is padding which keeps only valid part of the image.

Our model for image categorization is dependent on deep convolution neural networks.

1. Input: Input the images from the data set by the "University of Waterloo" having a collection of images labeled by classification tags – called a training set.

2. Learning: In this step, we use the training set to learn for predictable inputs- a step called learning a model.

3. Evaluation Classifier is used to predict the classification of images that are labeled and eventually evaluate the quality of the classifier. We compare the labels predicted by the classifier with the result obtained and judge the input the classification is true or not

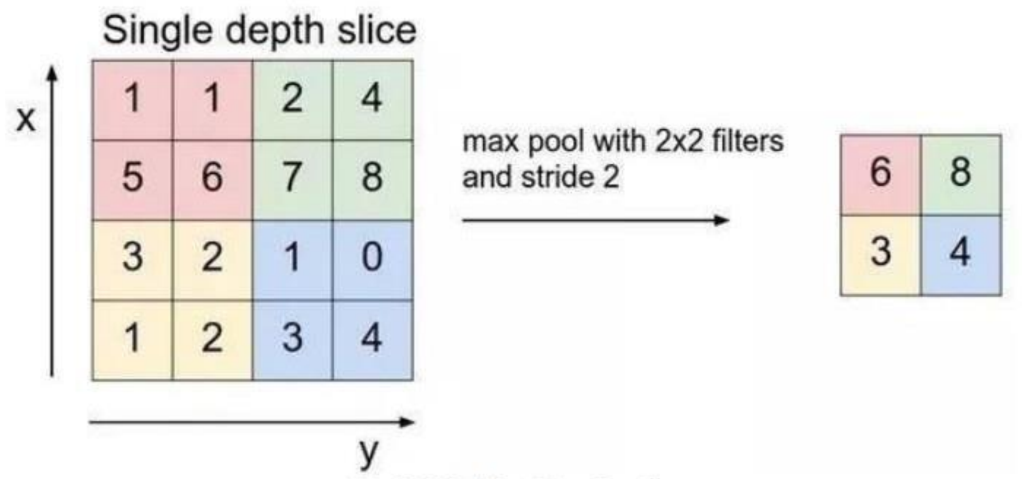


Fig.3.2.2.1.3 : Max Pooling

- The above approach followed was to experiment with different architectures on the CNN, to achieve better accuracy with the validation set, with minimum over-fitting. The emotion classification step consists of the following phases:

1) Splitting of Data

The dataset become break up into three classes consistent with the "Usage" label withinside the FER2013 dataset: Training, Public Test, and Private Test. The Training and Public Test set have been used for technology of a model, and the Private Test set become used for comparing the model.

2) Training and Generation of model

The neural community structure we used is including the subsequent layers:

i. Convolution Layer

In the convolution layer, a randomly instantiated learnable clear out out is slid, or convolved over the enter. The operation plays the dot product among the clear out out and every nearby vicinity of the enter. The output is a 3-D extent of more than one filters, additionally known as the function map.

ii. Max Pooling

The pooling layer is used to lessen the spatial length of the enter layer to decrease the dimensions of enter andthe computation cost.

iii. Fully related layer

In the entirely related layer, every neuron from the preceding layer is attached to the output neurons. The length of very last output layer is identical to the range of training wherein the enter photograph is to be classified.

iv. Activation characteristic

Activation capabilities are utilized in to lessen the overfitting. In the CNN structure, the ReLu activation characteristic has been used. The gain of the ReLu activation characteristic is that its gradient is constantly identical to 1, because of this that that maximum of the mistake is surpassed lower back for the duration of lower back-propagation.

$F(x) = \max(0, x)$ Equation 1: Equation of ReLu Activation Function

v. Softmax

The Softmax characteristic takes a vector of N actual numbers and normalizes that vector into a number values among (0, 1).

vi. Batch Normalization

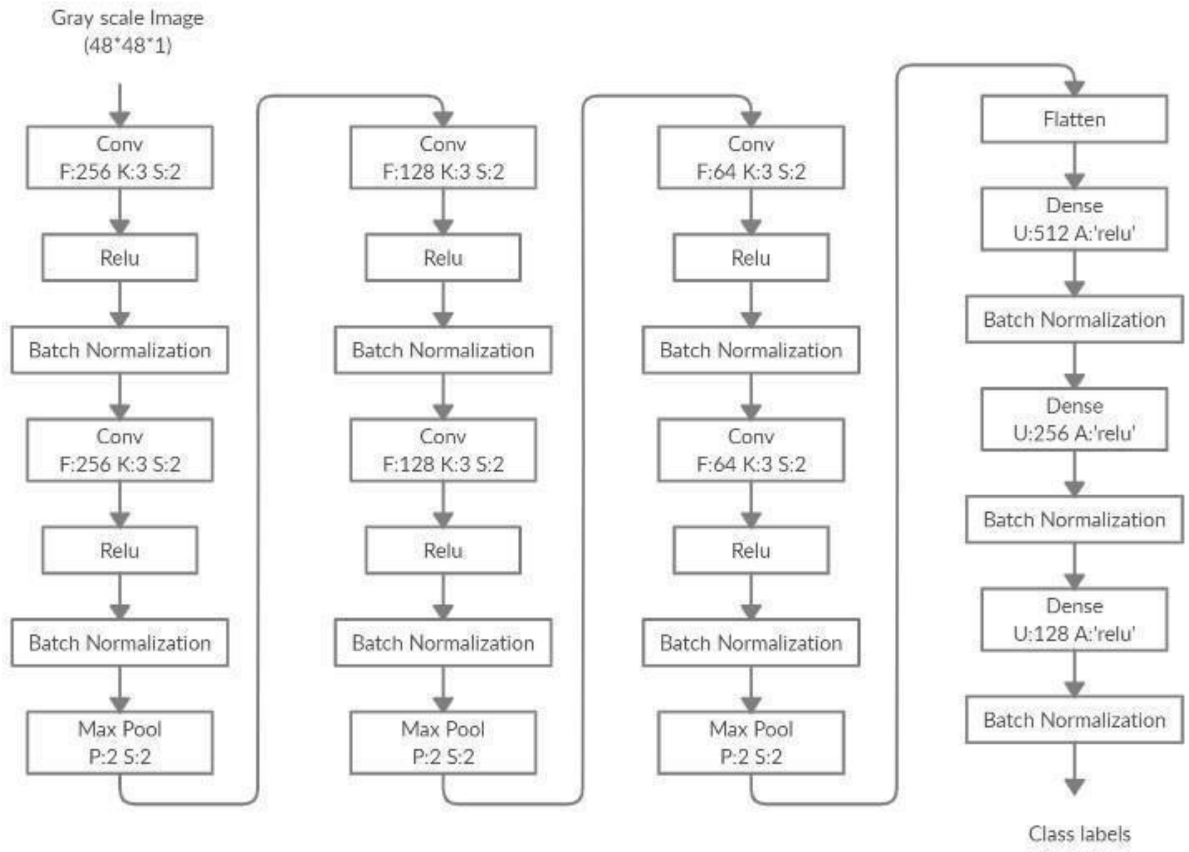
The batch normalize hurries up the schooling manner and applies a change that keeps the imply activation near zeroand the activation well- known deviation near 1.

1) Evaluation of version

The version generated for the duration of the schooling section became then evaluated at the validation set, which consisted of 3000 photos.

2) Using version to categorize actual time photos

The idea of switch studying may be used to discover person in photograph. The version generated for the duration



of the schooling manner includes pre-skilled weights and values, which may be used for implementation of a brand-new person recognition.

Fig.3.2.2.1.4 CNN architecture

Signature Classification

Let's recall this as a supervised studying challenge and put together the capabilities and labels. The Siamese community is fed with pairs of photos and their corresponding labels (comparable or distinctive). The information is

organized through looping over the dataset and forming an array of pairs of photos and their labels in every other array. This in the long run makes it a binary category problem!

Siamese Network Architecture:

Siamese community has a pile of convolutional and pooling layers and a very last completely linked layer with 128 neurons. The sister community takes at the identical weights and biases because the authentic community (basically method strolling the identical community twice). It takes the enter photograph pair and produces 128-D vectors as outputs.

The community learns to encode comparable enter photos nearer and distinctive ones, farther other than every different withinside the vector space. The traditional technique to educate it's miles through the usage of a contrastive loss feature or a triplet loss that takes 3 photos at once.

Architecture: -

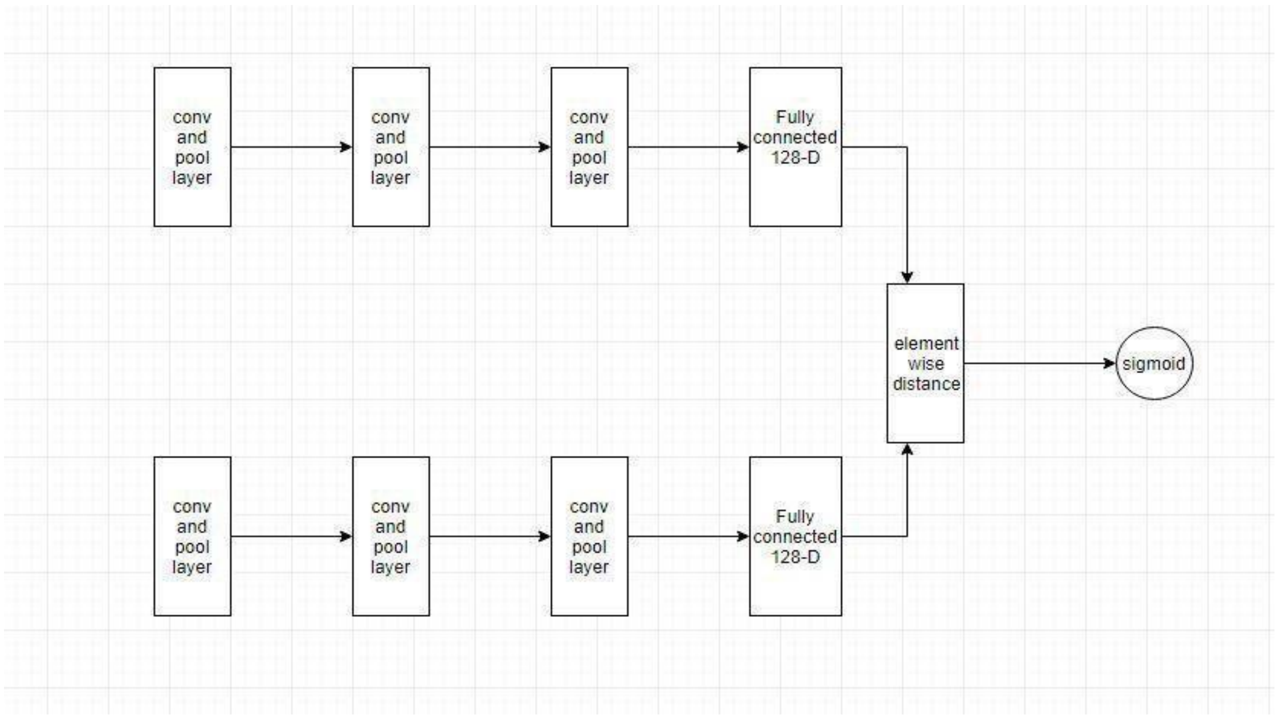


Fig. 3.2.1 Siamese network

The vectors are then subtracted element-smart to shape an unmarried 128-D vector (now no longer to be careworn with the L1 distance feature that yields only an unmarried scalar price as output).

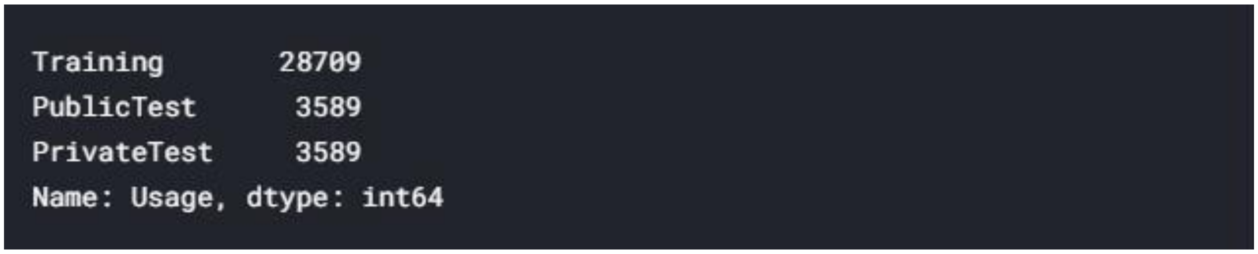
Well, we're nearly finished. The subsequent a part of its miles a completely linked community that takes the 128-D distance vector because the center and the output layer has a unmarried output neuron for category with sigmoid activation. The wide variety of hidden layers is a hyper-parameter and may be experimented with to get the fine results. Now, the community is skilled with the photograph pairs and labels that we accumulated earlier. The loss feature is 'binary-pass entropy' and the optimizer that fine fits is 'Adagrad'. Imposed an extra price at the version for making category errors at the terrible magnificence for the duration of schooling. This became performed preserving in thoughts that a wrong signature whilst labeled as accurate could bring about a more loss for the bank.

Chapter 4
Experimental Results

Examining dataset:

Signature Dataset

About 28,000 labeled images in the training set, 3,500 labeled images in the development set, and 3,500 images in the test set are comprised in the Signature dataset. Therefore, it can be seen that images are coupled for *80% training, 10% validation and 10% test*.**[8]**

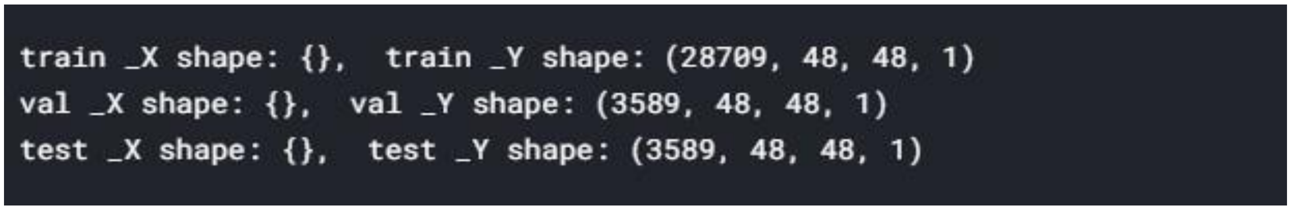


```
Training       28709
PublicTest      3589
PrivateTest     3589
Name: Usage, dtype: int64
```

After the above system, the subsequent three system have been done for the photo pre-processing:

1. Splitting dataset into three parts: train, validation, test
2. Convert strings to lists of integers
3. Reshape to 48x48 and normalize grayscale photo with 255.zero

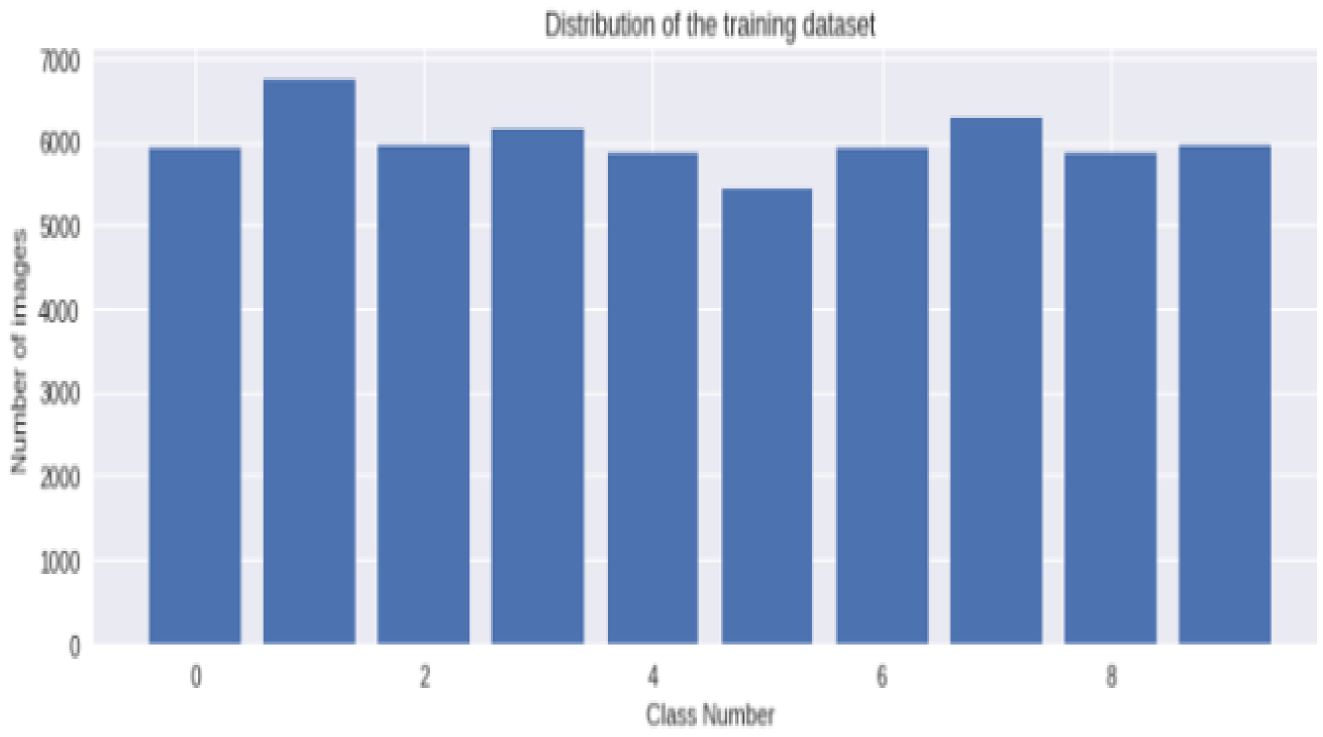
Results that have been achieved have been-



```
train _X shape: {},  train _Y shape: (28709, 48, 48, 1)
val _X shape: {},  val _Y shape: (3589, 48, 48, 1)
test _X shape: {},  test _Y shape: (3589, 48, 48, 1)
```

MNIST Dataset

About 80,000 labeled images in the training set, 8,000 labeled images in the development set, and 8,000 images in the test set are comprised in the Signature dataset. Therefore, it can be seen that images are coupled for *80% training, 10% validation and 10% test.*

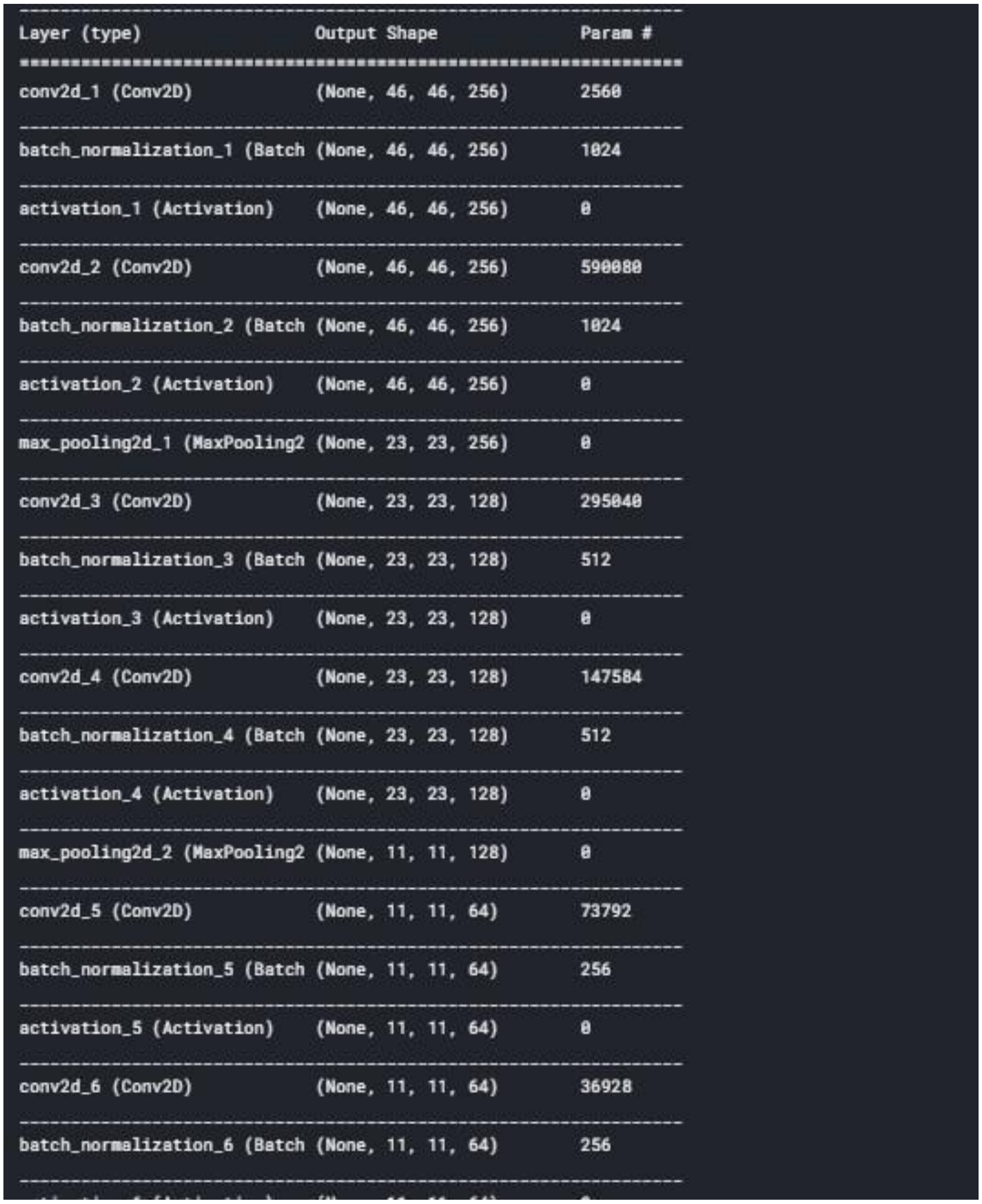


Bar graph output

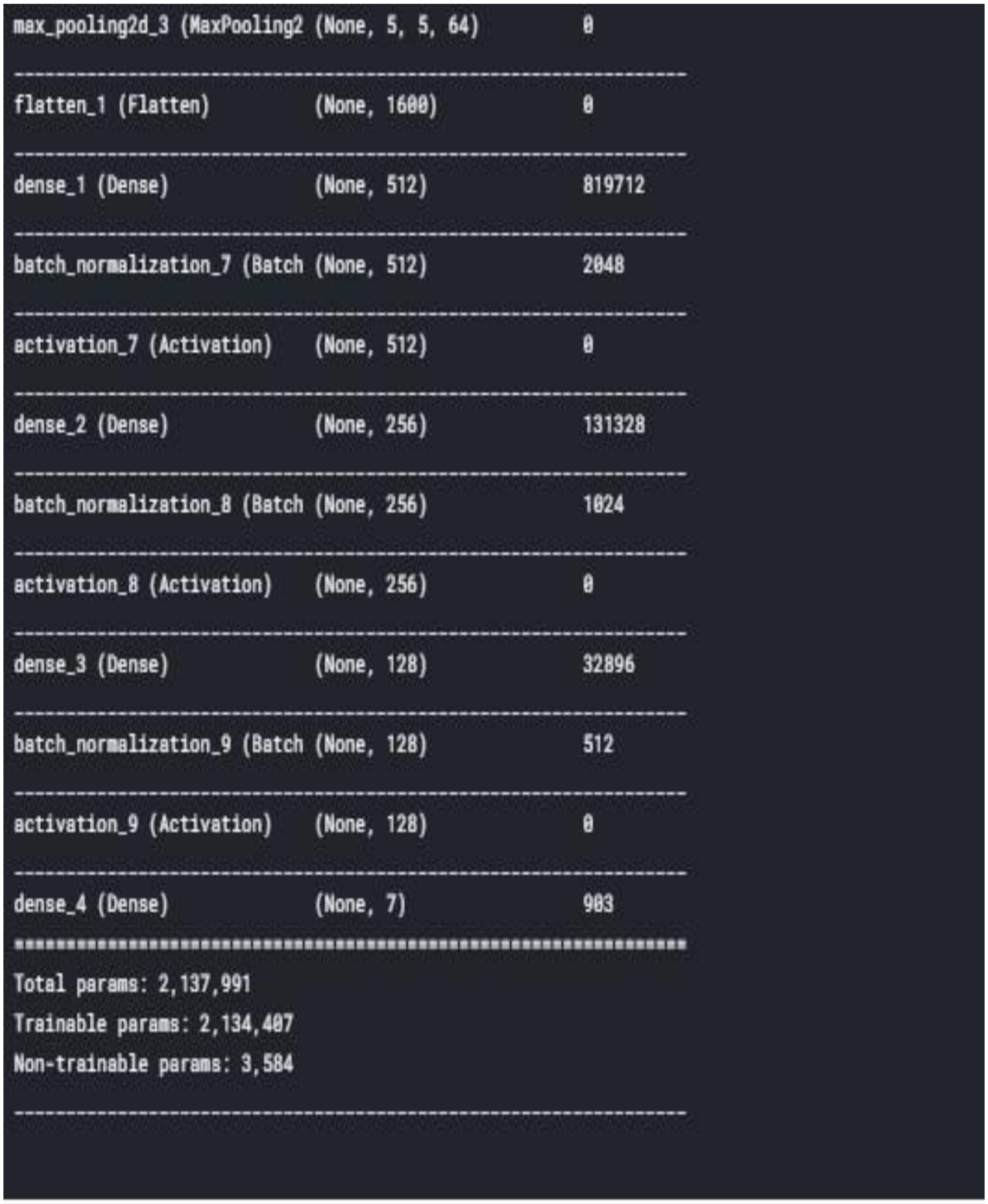
Building CNN model for Account Number Recognition

- ✓ Conv -> BN -> Activation -> Conv -> BN -> Activation -> MaxPooling
- ✓ Conv -> BN -> Activation -> Conv -> BN -> Activation -> MaxPooling ✓ Conv -> BN -> Activation -> Conv -> BN -> Activation -> MaxPooling
- ✓ Flatten
- ✓ Dense -> BN -> Activation
- ✓ Dense -> BN -> Activation
- ✓ Dense -> BN -> Activation
- ✓ Output layer

The results of building CNN model can be seen below:



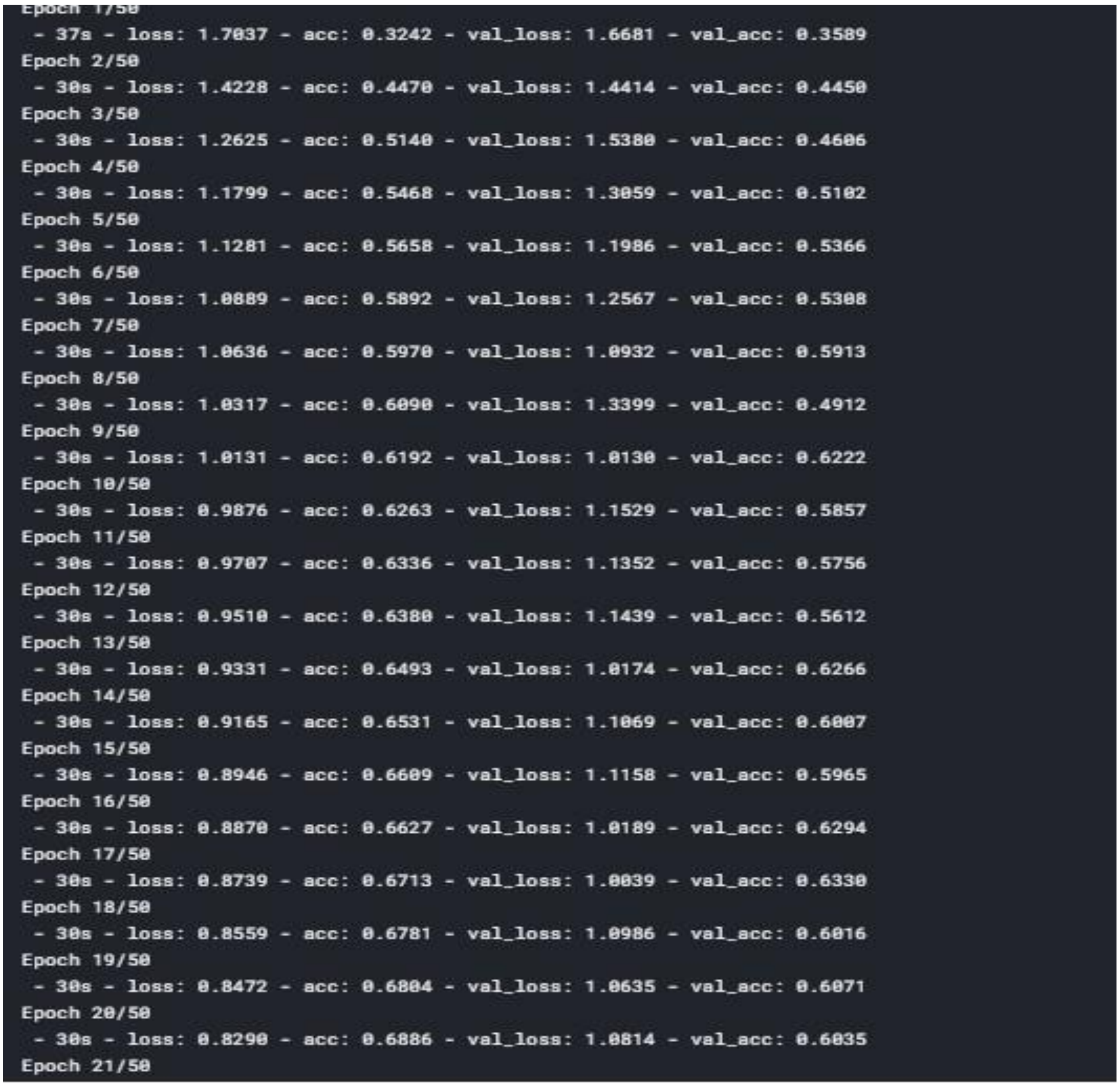
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 256)       2560
_________________________________________________________________
batch_normalization_1 (Batch (None, 46, 46, 256)       1024
_________________________________________________________________
activation_1 (Activation)    (None, 46, 46, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 256)       590080
_________________________________________________________________
batch_normalization_2 (Batch (None, 46, 46, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 46, 46, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 128)       295040
_________________________________________________________________
batch_normalization_3 (Batch (None, 23, 23, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 23, 23, 128)       147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 23, 23, 128)       512
_________________________________________________________________
activation_4 (Activation)    (None, 23, 23, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        73792
_________________________________________________________________
batch_normalization_5 (Batch (None, 11, 11, 64)        256
_________________________________________________________________
activation_5 (Activation)    (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
batch_normalization_6 (Batch (None, 11, 11, 64)        256
_________________________________________________________________
```



```
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)            0
_________________________________________________________________
dense_1 (Dense)              (None, 512)             819712
_________________________________________________________________
batch_normalization_7 (Batch (None, 512)             2048
_________________________________________________________________
activation_7 (Activation)    (None, 512)             0
_________________________________________________________________
dense_2 (Dense)              (None, 256)             131328
_________________________________________________________________
batch_normalization_8 (Batch (None, 256)             1024
_________________________________________________________________
activation_8 (Activation)    (None, 256)             0
_________________________________________________________________
dense_3 (Dense)              (None, 128)             32896
_________________________________________________________________
batch_normalization_9 (Batch (None, 128)             512
_________________________________________________________________
activation_9 (Activation)    (None, 128)             0
_________________________________________________________________
dense_4 (Dense)              (None, 7)               903
=================================================================
Total params: 2,137,991
Trainable params: 2,134,407
Non-trainable params: 3,584
_________________________________________________________________
```

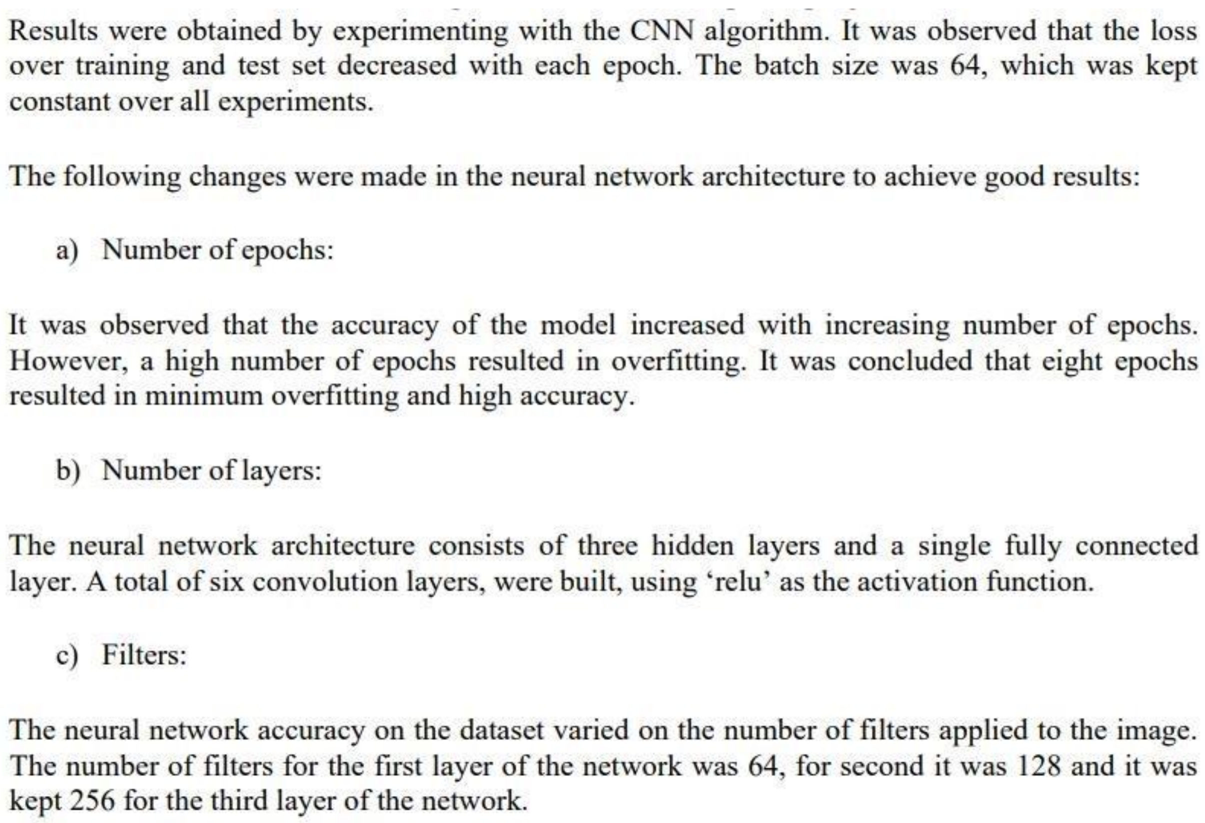
Fig.4.2.2.1 : Model Summary

The progress after successive epochs is displayed below:



```
Epoch 1/50
 - 37s - loss: 1.7037 - acc: 0.3242 - val_loss: 1.6681 - val_acc: 0.3589
Epoch 2/50
 - 30s - loss: 1.4228 - acc: 0.4470 - val_loss: 1.4414 - val_acc: 0.4450
Epoch 3/50
 - 30s - loss: 1.2625 - acc: 0.5140 - val_loss: 1.5380 - val_acc: 0.4606
Epoch 4/50
 - 30s - loss: 1.1799 - acc: 0.5468 - val_loss: 1.3059 - val_acc: 0.5102
Epoch 5/50
 - 30s - loss: 1.1281 - acc: 0.5658 - val_loss: 1.1986 - val_acc: 0.5366
Epoch 6/50
 - 30s - loss: 1.0889 - acc: 0.5892 - val_loss: 1.2567 - val_acc: 0.5308
Epoch 7/50
 - 30s - loss: 1.0636 - acc: 0.5970 - val_loss: 1.0932 - val_acc: 0.5913
Epoch 8/50
 - 30s - loss: 1.0317 - acc: 0.6090 - val_loss: 1.3399 - val_acc: 0.4912
Epoch 9/50
 - 30s - loss: 1.0131 - acc: 0.6192 - val_loss: 1.0130 - val_acc: 0.6222
Epoch 10/50
 - 30s - loss: 0.9876 - acc: 0.6263 - val_loss: 1.1529 - val_acc: 0.5857
Epoch 11/50
 - 30s - loss: 0.9707 - acc: 0.6336 - val_loss: 1.1352 - val_acc: 0.5756
Epoch 12/50
 - 30s - loss: 0.9510 - acc: 0.6380 - val_loss: 1.1439 - val_acc: 0.5612
Epoch 13/50
 - 30s - loss: 0.9331 - acc: 0.6493 - val_loss: 1.0174 - val_acc: 0.6266
Epoch 14/50
 - 30s - loss: 0.9165 - acc: 0.6531 - val_loss: 1.1069 - val_acc: 0.6007
Epoch 15/50
 - 30s - loss: 0.8946 - acc: 0.6609 - val_loss: 1.1158 - val_acc: 0.5965
Epoch 16/50
 - 30s - loss: 0.8870 - acc: 0.6627 - val_loss: 1.0189 - val_acc: 0.6294
Epoch 17/50
 - 30s - loss: 0.8739 - acc: 0.6713 - val_loss: 1.0039 - val_acc: 0.6330
Epoch 18/50
 - 30s - loss: 0.8559 - acc: 0.6781 - val_loss: 1.0986 - val_acc: 0.6016
Epoch 19/50
 - 30s - loss: 0.8472 - acc: 0.6804 - val_loss: 1.0635 - val_acc: 0.6071
Epoch 20/50
 - 30s - loss: 0.8290 - acc: 0.6886 - val_loss: 1.0814 - val_acc: 0.6035
Epoch 21/50
```

Fig.4.2.2.2: Successive epochs progress



Results were obtained by experimenting with the CNN algorithm. It was observed that the loss over training and test set decreased with each epoch. The batch size was 64, which was kept constant over all experiments.

The following changes were made in the neural network architecture to achieve good results:

a) Number of epochs:

It was observed that the accuracy of the model increased with increasing number of epochs. However, a high number of epochs resulted in overfitting. It was concluded that eight epochs resulted in minimum overfitting and high accuracy.

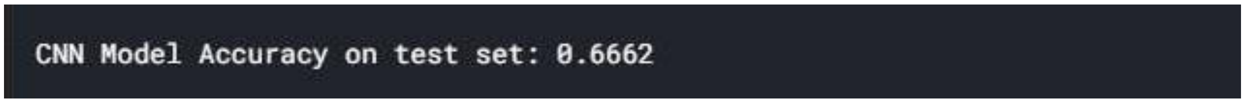
b) Number of layers:

The neural network architecture consists of three hidden layers and a single fully connected layer. A total of six convolution layers, were built, using 'relu' as the activation function.

c) Filters:

The neural network accuracy on the dataset varied on the number of filters applied to the image. The number of filters for the first layer of the network was 64, for second it was 128 and it was kept 256 for the third layer of the network.

The final, state-of-the-art-model gave a coaching accuracy of **seventy-nine percent** and a take a look ataccuracy **sixty- six percent** as shown below.



```
CNN Model Accuracy on test set: 0.6662
```

Fig.4.2.2.3: Accuracy Test Result

Siamese Network for Signature Classification:

Result of Siamese network after each epoch clearly seen in the figure:

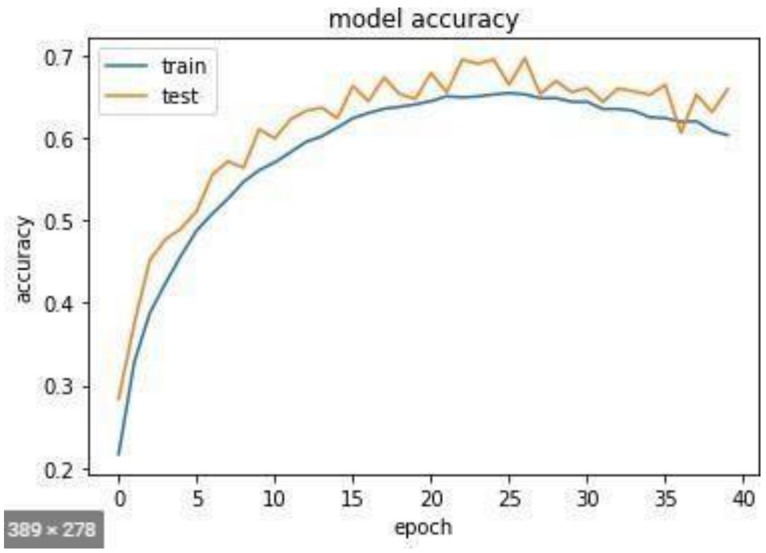


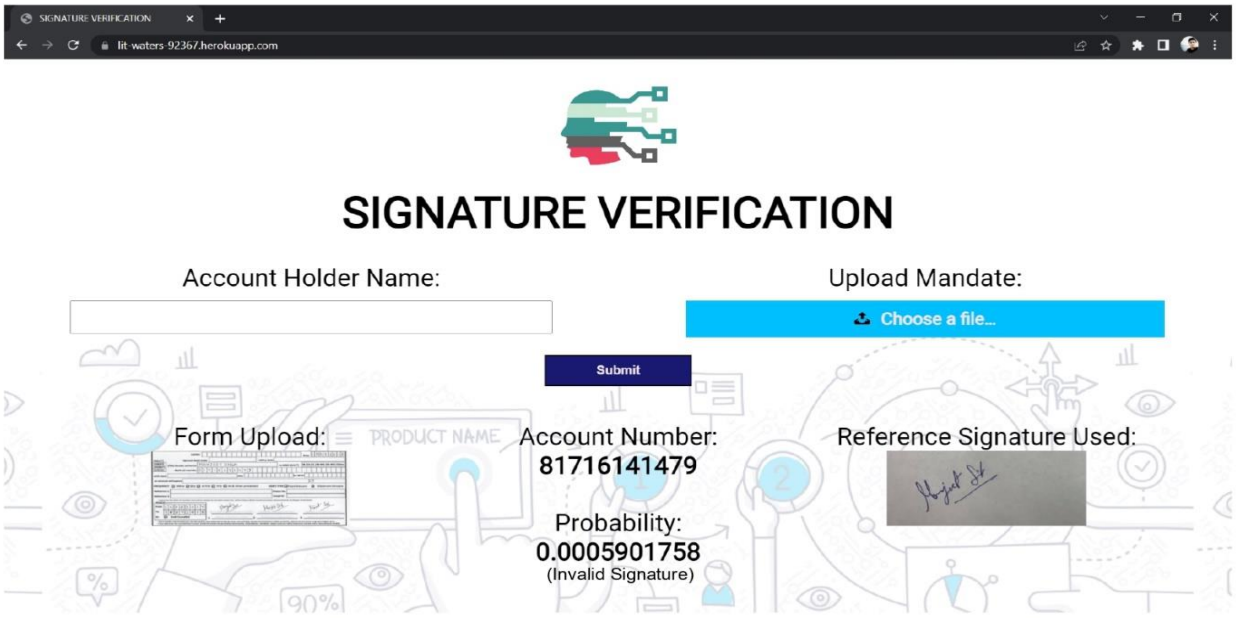
Fig.4.2.3.1: Successive epochs progress

Accuracy

- Extraction Accuracy (Numeric Fields): 99.6%
- Model Accuracy for Fake Signature Detection: 91% (present)
- Presently we are using Chinese and Dutch Signatures for training model
- With Indian Signature Dataset, model Accuracy can be increased to more than 85 %
- FAR (False Accuracy Rate) penalizes the model to ensure false doesn't gets classified as true.

Imposed an extra price at the version for making category errors at the terrible magnificence for the duration of schooling. This became performed preserving in thoughts that a wrong signature whilst labeled as accurate could bring about a more loss for the bank.

Forged Signature GUI


SIGNATURE VERIFICATION

lit-waters-92367.herokuapp.com

SIGNATURE VERIFICATION

Account Holder Name:

Upload Mandate:

Choose a file...

Submit

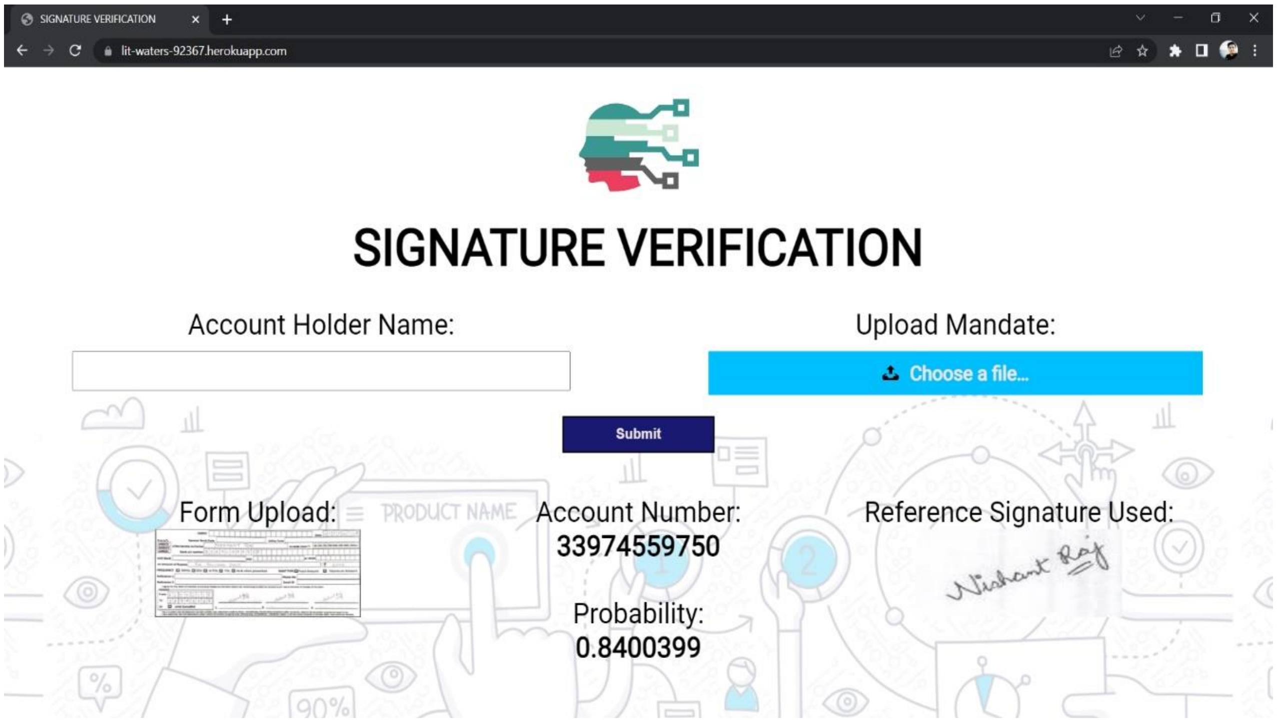
Form Upload:

Account Number:
81716141479

Probability:
0.0005901758
(Invalid Signature)

Reference Signature Used:

Genuine Signature GUI



SIGNATURE VERIFICATION

Account Holder Name:

Upload Mandate:

Form Upload:

Account Number:
33974559750

Probability:
0.8400399

Reference Signature Used:

Chapter 5
Conclusion

From the past few decade, scientists have suggested a many methods for Verification Of Offline Signature. For verifying signatures which are genuine as well as forged remainder a challenging task, fault rates have been fallen notably from some years , mainly due to development in Deep-Learning. Analyze fields contribution, it can be noticed that they centralize in the classes:

Get best features - Many new feature extractors have offered of the task. Texture–features (local binary variations), interest-point are matching :
scale-invariant feature transform and directional features Histogram Of Oriented Gradients
have auspicious used to improve the performance of Verification Systems of Offline Signature. More newly, methods for feature learning have been auspiciously applied to the task, to get the signatures of users learned by their subsets of signatures.

Elaborating classification of few samples - severe restrictions were given in applications, scientists improve performance by searching in cases where a few number of samples of user is available..

Increasing the datasets -Problem related to smaller number of datasets for per users , in developing forged signatures scientists work on number of datasets for training..

Model Building- To improve classification as well as accuracy, and strength of the few solution, scientists have explored the creation of both static as well as dynamic ensembles of classification.

<u>Chapter 6</u>

<u>Future Scope</u>

- It can be helpful in verifying Digital Signatures.
- It can reduce Fake Signature Cases.
- It can be a Big Step towards Digitalizing Banking Solutions and reducing hassles.
- In this, user can adopt without getting in intricacies of the model.
- Our model is highly user friendly for better customer experience.

References

1. B. Cozzens, R. Huang, M. Jay, K. Khem Bunjong, S.Paliskara, F. Zhan, M. Zhang, and S. Tayeb, "Signature Verification Using a Convolutional Neural Network,"BigData, pp. 2644–2651, 2017.

2. L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks,"Pattern Recognition,pp. 163–176, 2017.

3. Eman Alajrami , Belal A. M. Ashqar, Bassem S. Abu-Nasser, Ahmed J.Khalil, Musleh M. Musleh, Alaa M. Barhoom & Samy S.Abu-Naser, International Journal of Academic Multidisciplinary Research (IJAMR) ,3 (12):39-44, 2020.

4. Tolosana, R., Vera-Rodriguez, R., Fierrez J., & Ortega-Gracia, J, Deep Sign: Deep On-Line Signature Verification, IEEE Transactions on Biometrics, Behavior, and Identity Science, 3(2), 229-239, 2021.

5. Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks.", Pattern Recognition 70 : 163-176, 2017.

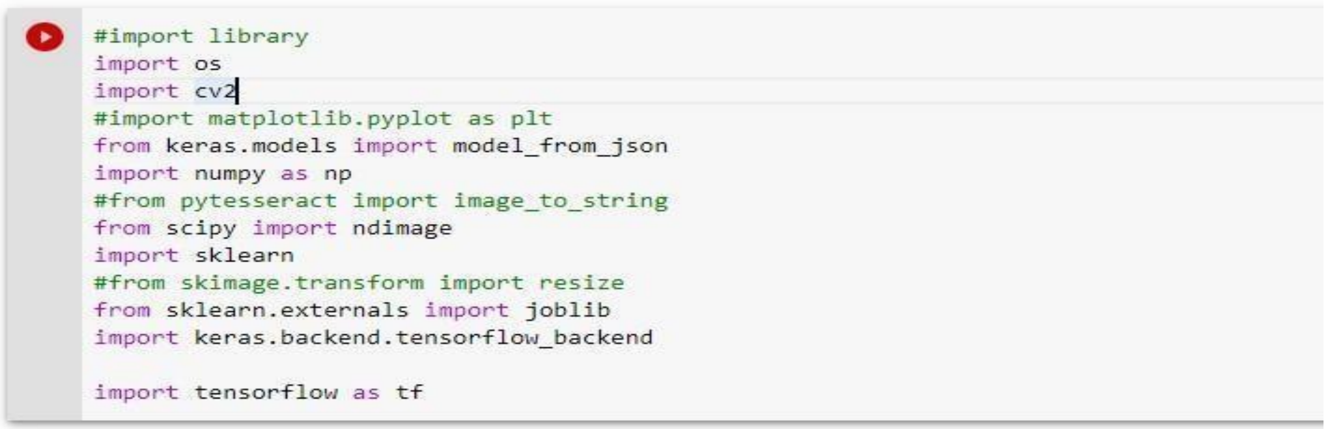
6. Kai Chen, Li Tian, Haisong Ding, Meng Cai, Lei Sun, Sen Liang, and Qiang Huo, based character model for online handwritten Chinese text recognition,IEEE, pages=1068–1073,2017.

7. Bouamra, W.; Djeddi, C.; Nini, B.; Diaz, M.; Siddiqi, I. Towards the design of an offline signature verifier based on a small number of genuine samples for training.182–195,2018.

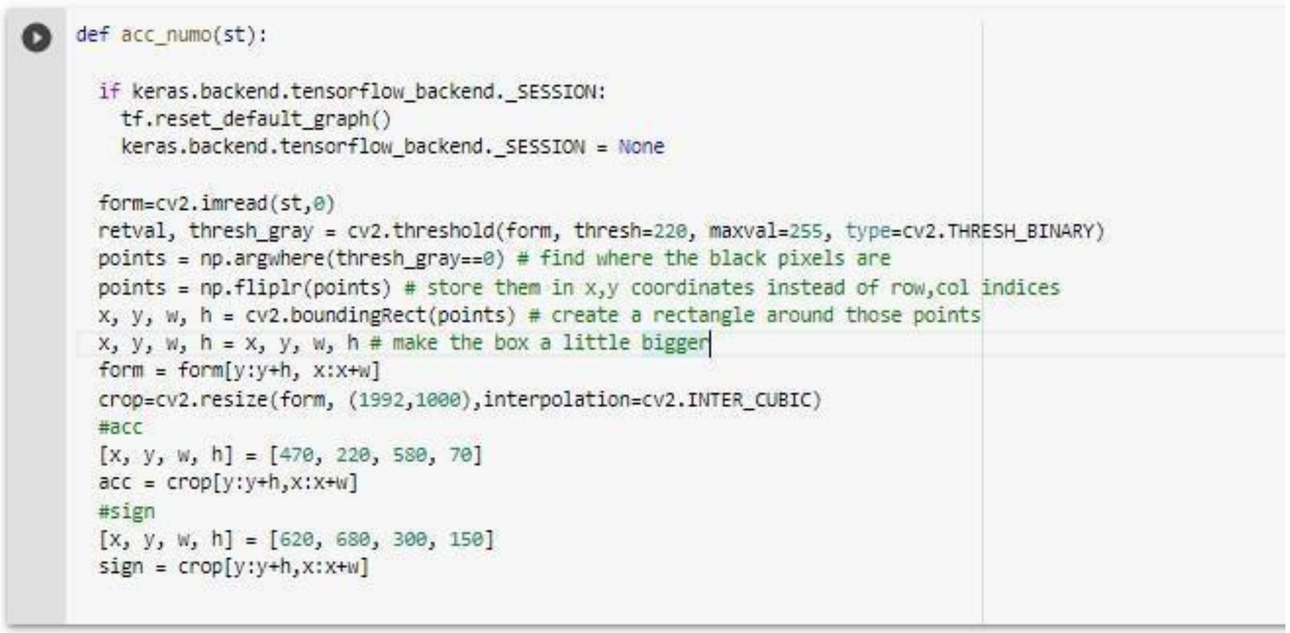
8.https://paperswithcode.com/dataset/cedar-signature
9.https://uwaterloo.ca/open-scholarship/open-data

Annexure *(Partial Codes)*

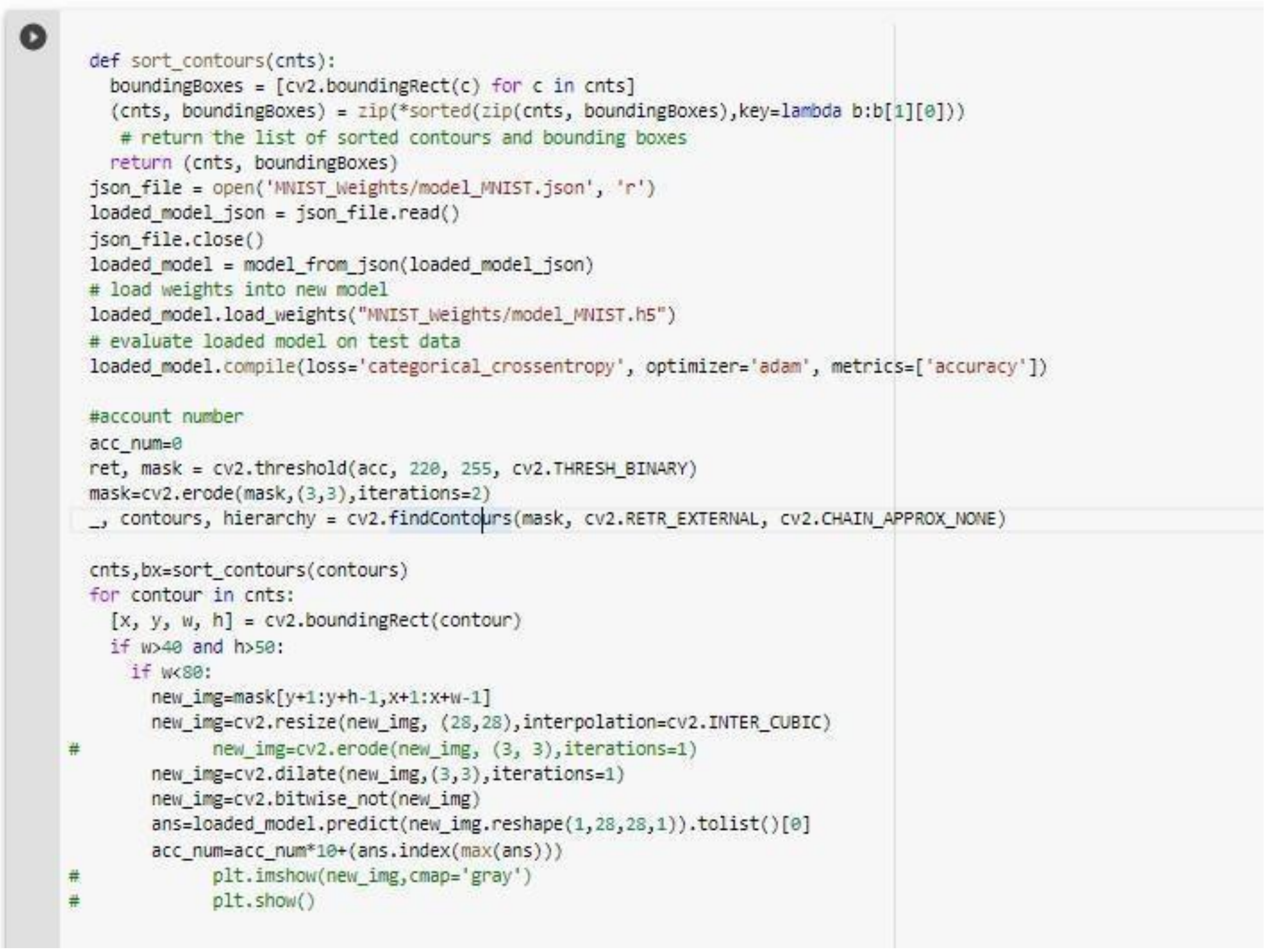
Signature and Account Number Extraction:



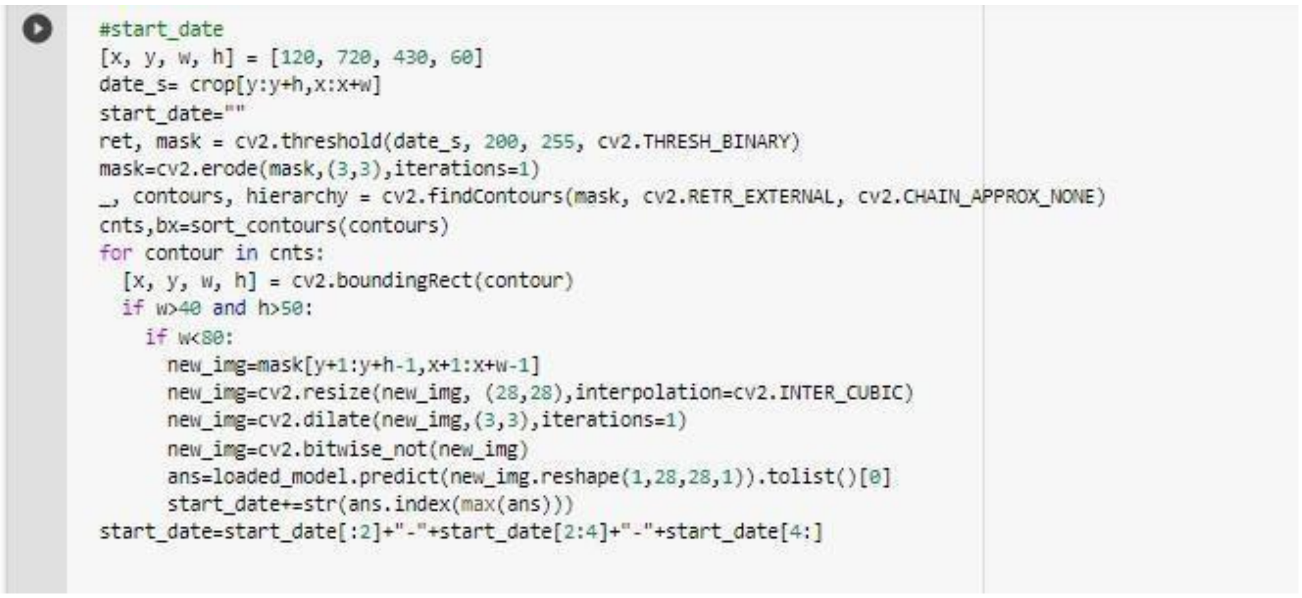
```
#import library
import os
import cv2
#import matplotlib.pyplot as plt
from keras.models import model_from_json
import numpy as np
#from pytesseract import image_to_string
from scipy import ndimage
import sklearn
#from skimage.transform import resize
from sklearn.externals import joblib
import keras.backend.tensorflow_backend

import tensorflow as tf
```



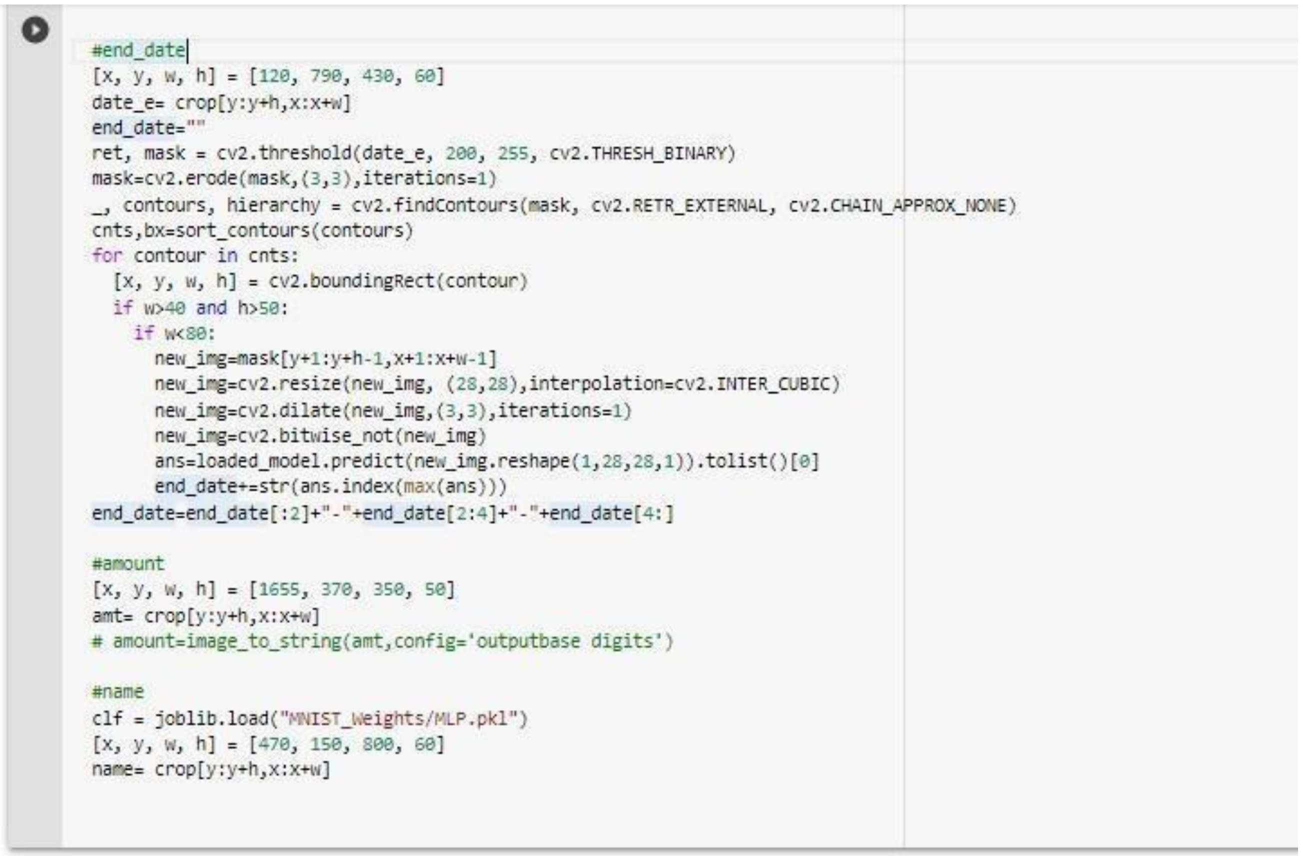
```
def acc_numo(st):

  if keras.backend.tensorflow_backend._SESSION:
    tf.reset_default_graph()
    keras.backend.tensorflow_backend._SESSION = None

  form=cv2.imread(st,0)
  retval, thresh_gray = cv2.threshold(form, thresh=220, maxval=255, type=cv2.THRESH_BINARY)
  points = np.argwhere(thresh_gray==0) # find where the black pixels are
  points = np.fliplr(points) # store them in x,y coordinates instead of row,col indices
  x, y, w, h = cv2.boundingRect(points) # create a rectangle around those points
  x, y, w, h = x, y, w, h # make the box a little bigger
  form = form[y:y+h, x:x+w]
  crop=cv2.resize(form, (1992,1000),interpolation=cv2.INTER_CUBIC)
  #acc
  [x, y, w, h] = [470, 220, 580, 70]
  acc = crop[y:y+h,x:x+w]
  #sign
  [x, y, w, h] = [620, 680, 300, 150]
  sign = crop[y:y+h,x:x+w]
```



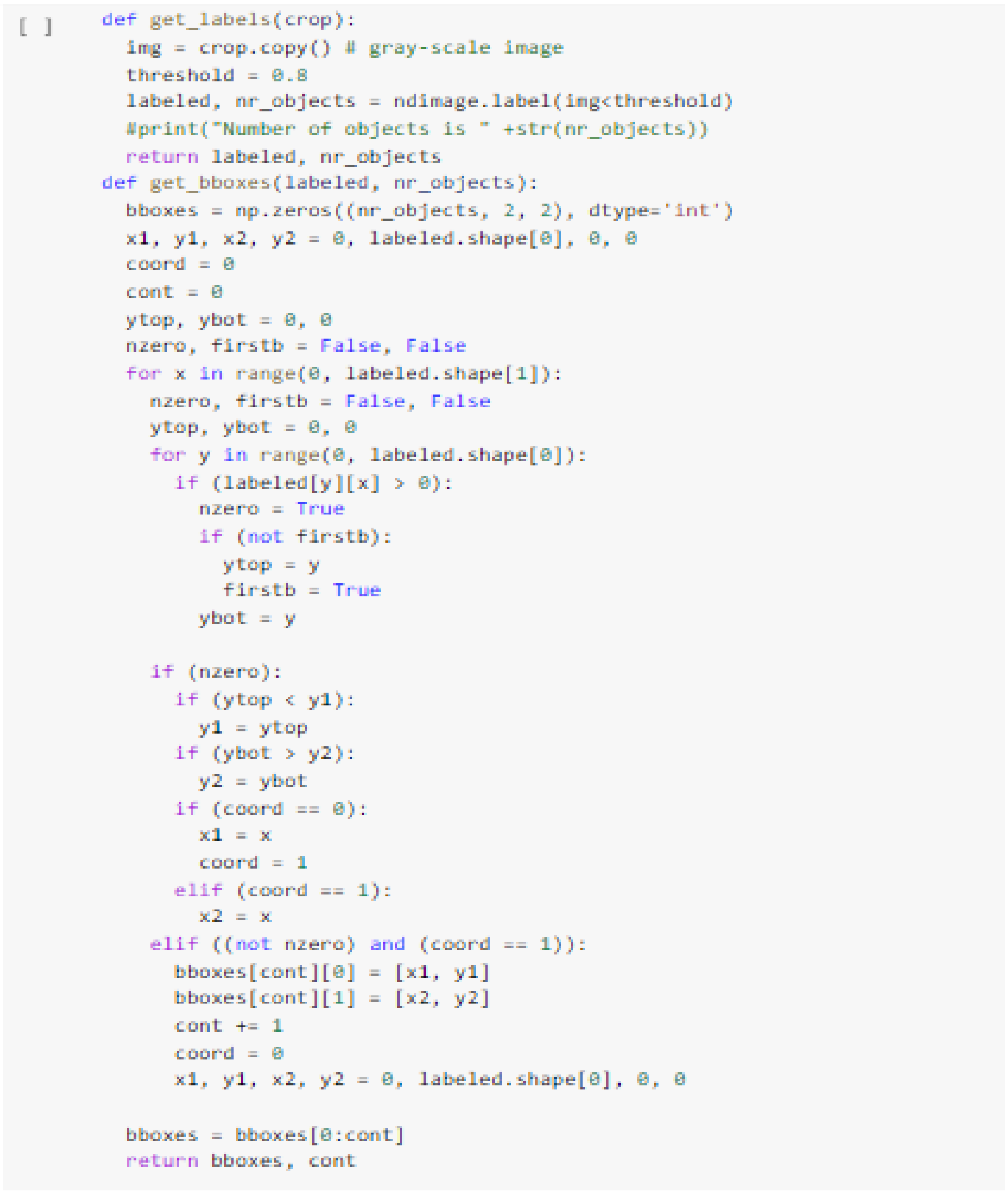
```
def sort_contours(cnts):
  boundingBoxes = [cv2.boundingRect(c) for c in cnts]
  (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),key=lambda b:b[1][0]))
   # return the list of sorted contours and bounding boxes
  return (cnts, boundingBoxes)
json_file = open('MNIST_Weights/model_MNIST.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("MNIST_Weights/model_MNIST.h5")
# evaluate loaded model on test data
loaded_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#account number
acc_num=0
ret, mask = cv2.threshold(acc, 220, 255, cv2.THRESH_BINARY)
mask=cv2.erode(mask,(3,3),iterations=2)
_, contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

cnts,bx=sort_contours(contours)
for contour in cnts:
  [x, y, w, h] = cv2.boundingRect(contour)
  if w>40 and h>50:
    if w<80:
      new_img=mask[y+1:y+h-1,x+1:x+w-1]
      new_img=cv2.resize(new_img, (28,28),interpolation=cv2.INTER_CUBIC)
#           new_img=cv2.erode(new_img, (3, 3),iterations=1)
      new_img=cv2.dilate(new_img,(3,3),iterations=1)
      new_img=cv2.bitwise_not(new_img)
      ans=loaded_model.predict(new_img.reshape(1,28,28,1)).tolist()[0]
      acc_num=acc_num*10+(ans.index(max(ans)))
#           plt.imshow(new_img,cmap='gray')
#           plt.show()
```



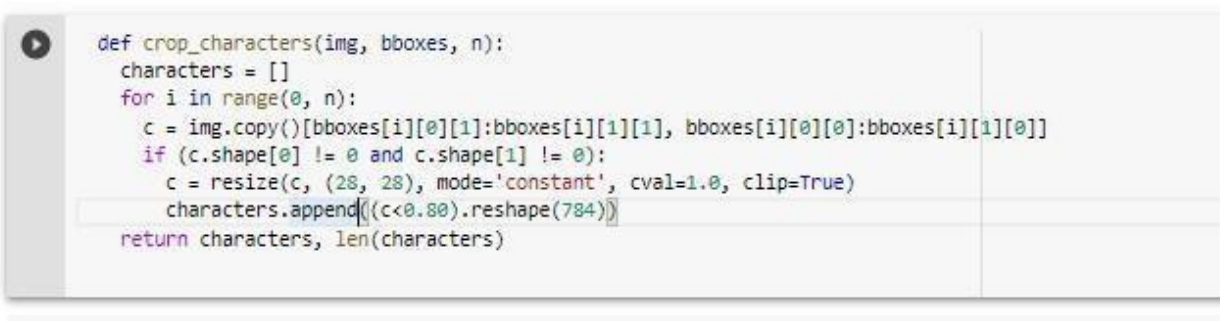
```
#start_date
[x, y, w, h] = [120, 720, 430, 60]
date_s= crop[y:y+h,x:x+w]
start_date=""
ret, mask = cv2.threshold(date_s, 200, 255, cv2.THRESH_BINARY)
mask=cv2.erode(mask,(3,3),iterations=1)
_, contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
cnts,bx=sort_contours(contours)
for contour in cnts:
  [x, y, w, h] = cv2.boundingRect(contour)
  if w>40 and h>50:
    if w<80:
      new_img=mask[y+1:y+h-1,x+1:x+w-1]
      new_img=cv2.resize(new_img, (28,28),interpolation=cv2.INTER_CUBIC)
      new_img=cv2.dilate(new_img,(3,3),iterations=1)
      new_img=cv2.bitwise_not(new_img)
      ans=loaded_model.predict(new_img.reshape(1,28,28,1)).tolist()[0]
      start_date+=str(ans.index(max(ans)))
start_date=start_date[:2]+"-"+start_date[2:4]+"-"+start_date[4:]
```



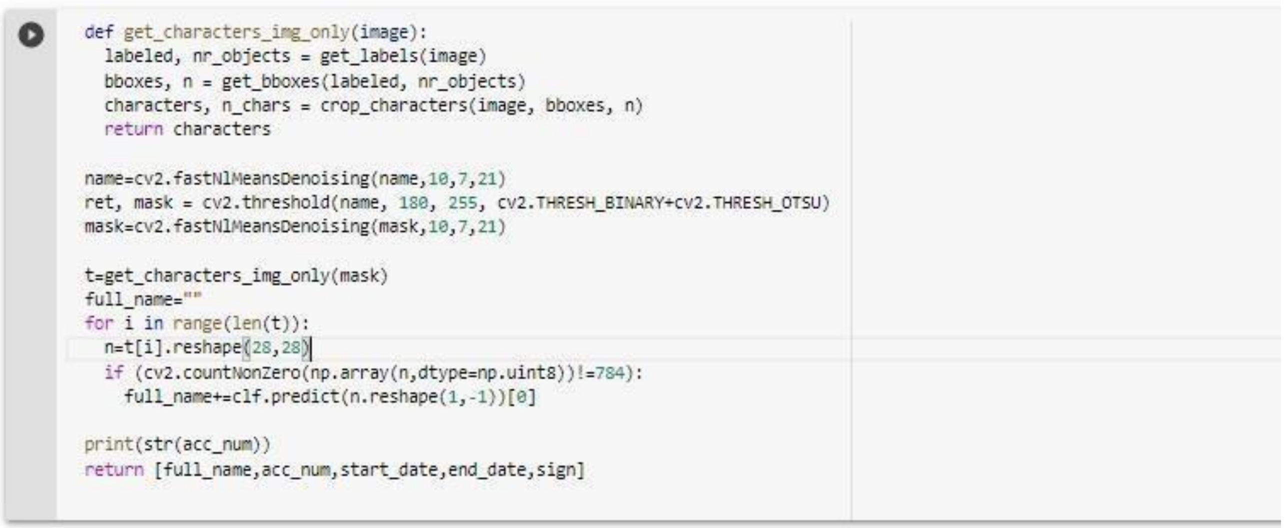
```
#end_date
[x, y, w, h] = [120, 790, 430, 60]
date_e= crop[y:y+h,x:x+w]
end_date=""
ret, mask = cv2.threshold(date_e, 200, 255, cv2.THRESH_BINARY)
mask=cv2.erode(mask,(3,3),iterations=1)
_, contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
cnts,bx=sort_contours(contours)
for contour in cnts:
  [x, y, w, h] = cv2.boundingRect(contour)
  if w>40 and h>50:
    if w<80:
      new_img=mask[y+1:y+h-1,x+1:x+w-1]
      new_img=cv2.resize(new_img, (28,28),interpolation=cv2.INTER_CUBIC)
      new_img=cv2.dilate(new_img,(3,3),iterations=1)
      new_img=cv2.bitwise_not(new_img)
      ans=loaded_model.predict(new_img.reshape(1,28,28,1)).tolist()[0]
      end_date+=str(ans.index(max(ans)))
end_date=end_date[:2]+"-"+end_date[2:4]+"-"+end_date[4:]

#amount
[x, y, w, h] = [1655, 370, 350, 50]
amt= crop[y:y+h,x:x+w]
# amount=image_to_string(amt,config='outputbase digits')

#name
clf = joblib.load("MNIST_Weights/MLP.pkl")
[x, y, w, h] = [470, 150, 800, 60]
name= crop[y:y+h,x:x+w]
```



```
[ ]  def get_labels(crop):
       img = crop.copy() # gray-scale image
       threshold = 0.8
       labeled, nr_objects = ndimage.label(img<threshold)
       #print("Number of objects is " +str(nr_objects))
       return labeled, nr_objects
     def get_bboxes(labeled, nr_objects):
       bboxes = np.zeros((nr_objects, 2, 2), dtype='int')
       x1, y1, x2, y2 = 0, labeled.shape[0], 0, 0
       coord = 0
       cont = 0
       ytop, ybot = 0, 0
       nzero, firstb = False, False
       for x in range(0, labeled.shape[1]):
         nzero, firstb = False, False
         ytop, ybot = 0, 0
         for y in range(0, labeled.shape[0]):
           if (labeled[y][x] > 0):
             nzero = True
             if (not firstb):
               ytop = y
               firstb = True
             ybot = y

         if (nzero):
           if (ytop < y1):
             y1 = ytop
           if (ybot > y2):
             y2 = ybot
           if (coord == 0):
             x1 = x
             coord = 1
           elif (coord == 1):
             x2 = x
         elif ((not nzero) and (coord == 1)):
           bboxes[cont][0] = [x1, y1]
           bboxes[cont][1] = [x2, y2]
           cont += 1
           coord = 0
           x1, y1, x2, y2 = 0, labeled.shape[0], 0, 0

       bboxes = bboxes[0:cont]
       return bboxes, cont
```

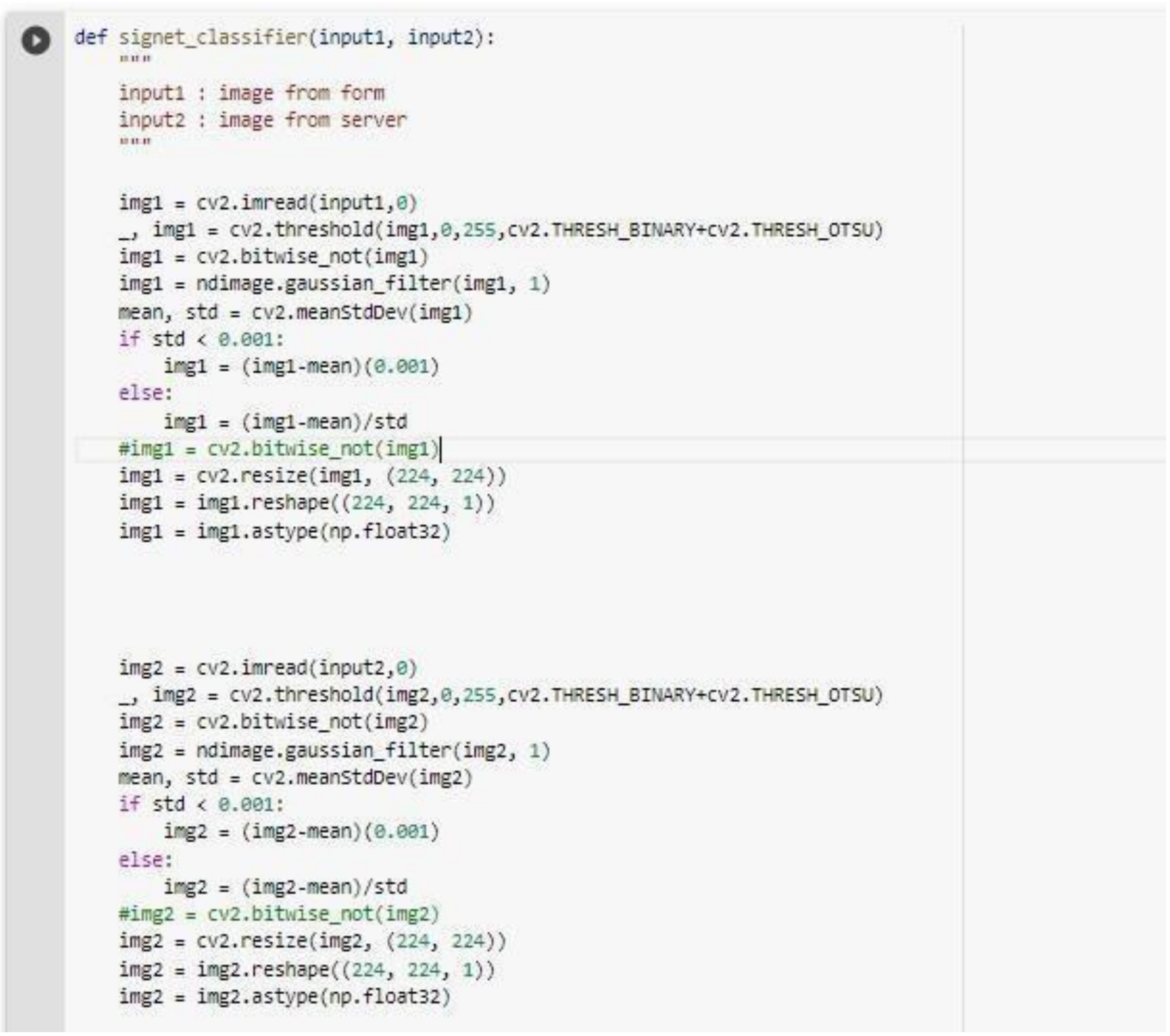


```
def crop_characters(img, bboxes, n):
  characters = []
  for i in range(0, n):
    c = img.copy()[bboxes[i][0][1]:bboxes[i][1][1], bboxes[i][0][0]:bboxes[i][1][0]]
    if (c.shape[0] != 0 and c.shape[1] != 0):
      c = resize(c, (28, 28), mode='constant', cval=1.0, clip=True)
      characters.append((c<0.80).reshape(784))
  return characters, len(characters)
```

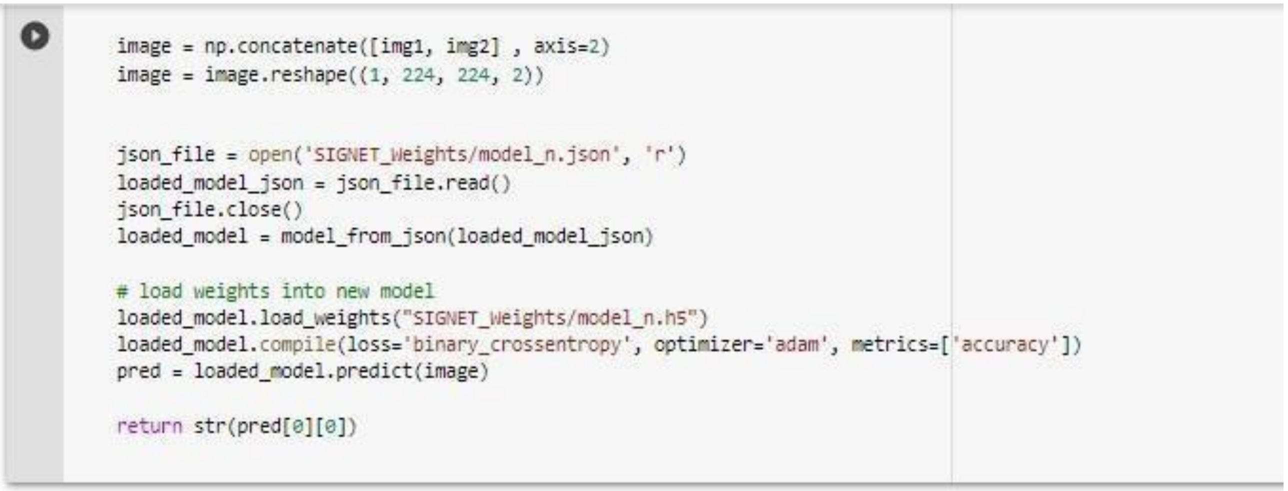


```
def get_characters_img_only(image):
  labeled, nr_objects = get_labels(image)
  bboxes, n = get_bboxes(labeled, nr_objects)
  characters, n_chars = crop_characters(image, bboxes, n)
  return characters

name=cv2.fastNlMeansDenoising(name,10,7,21)
ret, mask = cv2.threshold(name, 180, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
mask=cv2.fastNlMeansDenoising(mask,10,7,21)

t=get_characters_img_only(mask)
full_name=""
for i in range(len(t)):
  n=t[i].reshape(28,28)
  if (cv2.countNonZero(np.array(n,dtype=np.uint8))!=784):
    full_name+=clf.predict(n.reshape(1,-1))[0]

print(str(acc_num))
return [full_name,acc_num,start_date,end_date,sign]
```

Signature Classification:



```
def signet_classifier(input1, input2):
    """
    input1 : image from form
    input2 : image from server
    """

    img1 = cv2.imread(input1,0)
    _, img1 = cv2.threshold(img1,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    img1 = cv2.bitwise_not(img1)
    img1 = ndimage.gaussian_filter(img1, 1)
    mean, std = cv2.meanStdDev(img1)
    if std < 0.001:
        img1 = (img1-mean)(0.001)
    else:
        img1 = (img1-mean)/std
    #img1 = cv2.bitwise_not(img1)
    img1 = cv2.resize(img1, (224, 224))
    img1 = img1.reshape((224, 224, 1))
    img1 = img1.astype(np.float32)




    img2 = cv2.imread(input2,0)
    _, img2 = cv2.threshold(img2,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    img2 = cv2.bitwise_not(img2)
    img2 = ndimage.gaussian_filter(img2, 1)
    mean, std = cv2.meanStdDev(img2)
    if std < 0.001:
        img2 = (img2-mean)(0.001)
    else:
        img2 = (img2-mean)/std
    #img2 = cv2.bitwise_not(img2)
    img2 = cv2.resize(img2, (224, 224))
    img2 = img2.reshape((224, 224, 1))
    img2 = img2.astype(np.float32)
```



```
    image = np.concatenate([img1, img2] , axis=2)
    image = image.reshape((1, 224, 224, 2))


    json_file = open('SIGNET_Weights/model_n.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)

    # load weights into new model
    loaded_model.load_weights("SIGNET_Weights/model_n.h5")
    loaded_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    pred = loaded_model.predict(image)

    return str(pred[0][0])
```