RAMAIAH Institute of Technology

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Report submitted for the Subject

Introduction to Artificial Intelligence -AD35

Title:

Submitted by

Name	USN
Anuritha L	1MS21AD012
Shriya Srinivas	1MS21AD047
Shubhangi	1MS21AD048
Yashashwini Singh	IMS21AD061

Under the Supervision of

Dr. Meeradevi
Associate Professor

RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

BANGALORE-560054

www.msrit.edu, Nov 2022-Jan 2023

INDEX

ABSTRACT

Due to increased usage of digital technologies in all sectors and in almost all day to day activities to store and pass information, Handwriting character recognition has become a popular subject of research. In the field of Machine Learning, recognition of objects has become
most sought one. Handwriting character recognition refers to the computer's ability to detect and interpret intelligible.
Handwriting input from Handwriting sources such as touch screens, photographs, paper documents, and other sources. The development is based on an artificial neural network, which is a field of study in artificial intelligence. The use of neural networks for recognizing Handwriting characters is more efficient and robust compared with other computing techniques. Patterns are usually recognized with the help of large image data-set. Handwriting recognition is an application of pattern recognition through image. By using such concepts, we can train computers to read letters and numbers belonging to any language present in an image. Handwriting digits and character recognitions have become increasingly important in today's digitized world due to their practical applications in various day to day activities.. A good example is the use of automatic processing systems used in banks to process bank cheques. This algorithm can recognize the characters in image and convert them to text. Then the text can be converted to desired language of choice. With the help of hand-write recognition and AI, the answer scripts can be evaluated without human involvement. The challenge of visual pattern recognition is only apparent to develop a computer system to read handwriting. The artificial neural networks approach is considered as the best way to develop systems for recognizing handwriting. It allows machines to match Humans have different handwriting styles, some of which are difficult to read. The main aim of this paper is to develop a model that will be used to read Handwriting digits, characters, and words from the image using the concept of 'Convolution Neural Network'.

DATA SET DESCRPTION

For recognising handwritten forms, the very first step is to gather data in a considerable amount for training. The dataset for this project contains 372450 images of alphabets of 28×2, all present in the form of a CSV file
The dataset contains 26 folders (A-Z) containing handwritten images in size 28x*28 pixels, each alphabet in the image is centre fitted to 20x*20 pixel box. Each image is stored as Gray-level
Kernel **CSV_To_Images** contains script to convert .CSV file to actual images in.png format in structured folder. The images are taken from NIST(https://www.nist.gov/srd/nist-special-database-19) and NMIST large dataset and few other sources which were then formatted as mentioned above.

The **MNIST database** (*Modified National Institute of Standards and Technology database*[1]) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine to get an error rate of 0.8%.

Special Database 19 contains NIST's entire corpus of training materials for handprinted document and character recognition. It publishes Handprinted Sample Forms from 3600 writers, 810,000 character images isolated from their forms, ground truth classifications for those images, reference forms for further data collection, and software utilities for image management and handling.

The features of this database are:

- Final accumulation of NIST's handprinted sample data
- Full page HSF forms from 3600 writers
- Separate digit, upper and lower case, and free text fields
- Over 800,000 images with hand checked classifications

CSV file they were present as 784 columns of pixel data(784 attributes) and 9013 datapoints i.e ,9013 rows. All the labels are present in the form of floating point values, that we convert to integer values, & so we create a dictionary word_dict to map the integer values with the characters. We are using a labelled dataset for this project.

REQUIREMENTS

Requirements to create this Machine Learning Projects:

These are the libraries/frameworks which should be installed in your system:

- Python (latest version) installed.
- Jupyter notebook

Libraries:

- **Numpy (pip install numpy):** NumPy is a Python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- **OpenCV (pip install cv2):** OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.
- **Keras (pip install keras): Keras** is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.
- **Tensorflow (As we know that keras uses Tensorflow as backend) (pip install tensorflow):** TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.
- **Matplotlib (To visualize our Data) (pip install matplotlib): Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.
- **Pandas (pip install pandas): pandas** is a software library written for the Python programming language for data manipulation and analysis.[2] In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[3] The name is derived from the term "**pan**el **da**ta", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

ALGORITHM DESCRIPTION

STEP 1: Importing the necessary packages
We will import libraries that we have installed in our system, whenever we require them. So firstly we are only importing numpy , pandas and matplotlib to preprocess our data and to visualize that data.

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

STEP 2: Read the data
Now we will read our dataset that is a csv file using the pandas read_csv() method. We will read the data as data type float32 as the csv file is very large and it will be better to read it as float. Here we are only printing the first 10 images using **data.head(10).**

```
In [3]: data = pd.read_csv('A_Z Handwritten Data.csv').astype('float32')
        data.head(10)
```

Out[3]:

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642	0.643	0.644	0.645	0.646	0.647	0.648
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	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	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

10 rows × 785 columns

(The above image shows some of the rows of the dataframe data using the head() function of dataframe)

STEP 3: Split data into images and their labels
Splitting the data read into the images & their corresponding labels. The '0' contains the labels, & so we drop the '0' column from the data dataframe read & use it in the y to form the labels.

```
In [4]: X = data.drop('0',axis = 1)
        y = data['0']
```

STEP 4: Reshaping the data in the csv file so that it can be displayed as an image using sklearn

```
In [6]: from sklearn.model_selection import train_test_split
        from sklearn.utils import shuffle

        x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

        x_train = np.reshape(x_train.values, (x_train.shape[0], 28,28))
        x_test = np.reshape(x_test.values, (x_test.shape[0], 28,28))

        print("Shape of Training data: ", x_train.shape)
        print("Shape of Testing data: ", x_test.shape)

        Train data shape:  (297960, 28, 28)
        Test data shape:  (74490, 28, 28)
```

- In the above segment, we are splitting the data into training & testing dataset using train_test_split().

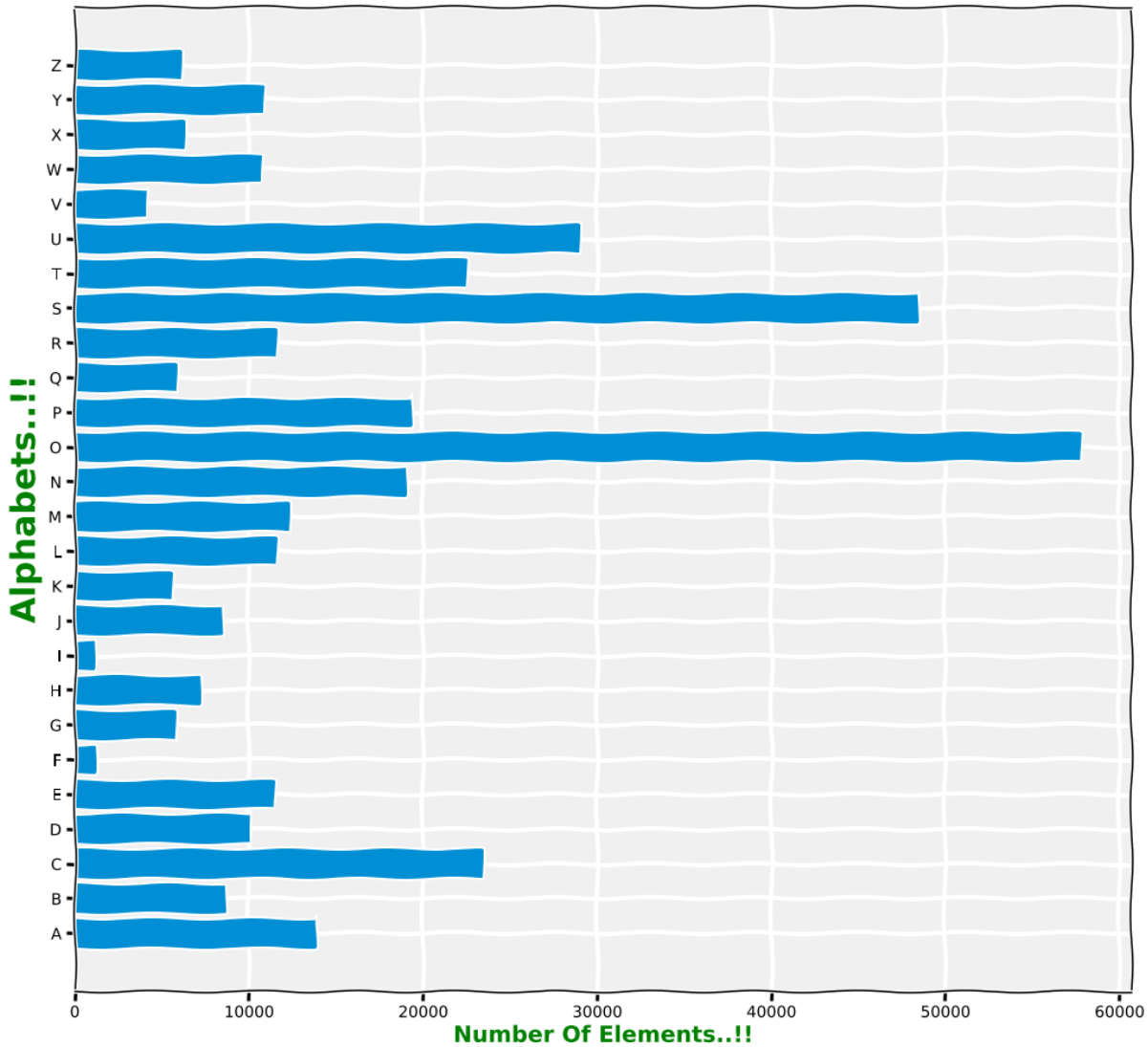
- Also, we are reshaping the train & test image data so that they can be displayed as an image, as initially in the CSV file they were present as 784 columns of pixel data. So we convert it to 28×28 pixels.

```
In [18]: words = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:'Q
```

- All the labels are present in the form of floating point values, that we convert to integer values, & so we create a dictionary words to map the integer values with the characters.

STEP 5: Plotting the number of alphabets in the dataset

- Here we are only describing the distribution of the alphabets.
- Firstly we convert the labels into integer values and append into the count list according to the label. This count list has the number of images present in the dataset belonging to each alphabet.
- Now we create a list – alphabets containing all the characters using the values() function of the dictionary.
- Now using the count & alphabets lists we draw the horizontal bar plot.



STEP 6: Shuffle the training data

```
In [1]: import cv2
        shuffle_data = shuffle(x_train)
```

- Now we shuffle some of the images of the train set.
- The shuffling is done using the shuffle() function so that we can display some random images.
- We then create 9 plots in 3×3 shape & display the thresholded images of 9 alphabets.

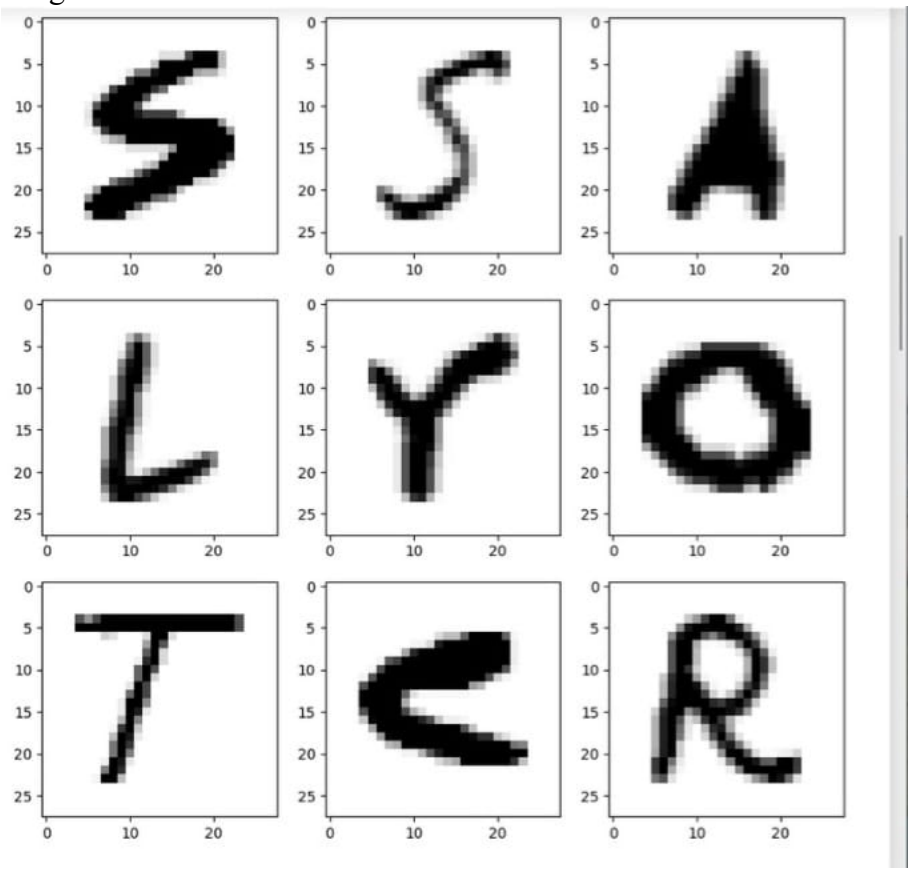
STEP 7: Visualize our training data

```
In [1]: import cv2
        shuffle_data = shuffle(x_train)

        fig, axes = plt.subplots(3,3, figsize = (10,10))
        axes = axes.flatten()

        for i in range(9):
            _, shu = cv2.threshold(shuffle_data[i], 30, 200, cv2.THRESH_BINARY)
            axes[i].imshow(np.reshape(shuffle_data[i], (28,28)), cmap="Greys")
        plt.show()
```

Now let's visualize our training data, and let's check that we have successfully converted them to view in image form.



STEP 8: Reshaping the data for our model

```
In [10]: import tensorflow
         from tensorflow.keras.utils import to_categorical

         y_training = to_categorical(y_train, num_classes = 26, dtype='int')
         y_testing = to_categorical(y_test, num_classes = 26, dtype='int')

         print("New shape of training labels: ", y_training.shape)
         print("New shape of testing labels: ", y_testing.shape)

         New shape of training labels:  (297960, 26)
         New shape of testing labels:  (74490, 26)
```

Now we reshape the train & test image dataset so that they can be put in the model.
New shape of train data: (297960, 28, 28, 1)
New shape of train data: (74490, 28, 28, 1)

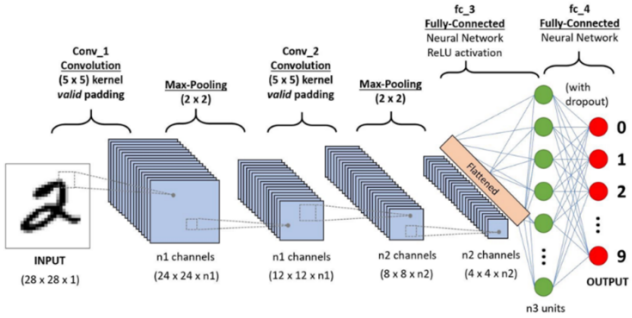
STEP 9: Converting to categorical
Now we have to convert our single float values to categorical values using to_categorical method given by tensorflow, keras

```
In [10]: import tensorflow
         from tensorflow.keras.utils import to_categorical
         y_training = to_categorical(y_train, num_classes = 26, dtype='int')
         y_testing = to_categorical(y_test, num_classes = 26, dtype='int')
```

STEP 10: Model Creation:
We will be creating a Convolutional Neural Networks (CNN) model which is very popular while classifying images as it extracts the features of images using several hidden layers or we say several layers of filters.

What is CNN?

CNN stands for Convolutional Neural Networks that are used to extract the features of the images using several layers of filters.



The convolution layers are generally followed by maxpool layers that are used to reduce the number of features extracted and ultimately the output of the maxpool and layers and convolution layers are flattened into a vector of single dimension and are given as an input to the Dense layer (The fully connected network).

```
In [11]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
         from tensorflow.keras.optimizers import SGD, Adam
         from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

The model we have created consist of :

- 3 convolutional layers(Conv2D) of 64,64,64 layers each, followed by MaxPool layers that are used to reduce the number of features extracted.
- And after flatten the layers, we created two fully connected layers (Dense layer) of 128,265 layers respectively,
- and at last we have created our output layer that is also a fully connected layer with softmax as an activation function.

```
In [13]: model = Sequential()

         model.add(Conv2D(64 , (3, 3), activation='relu', input_shape=(28,28,1)))
         model.add(MaxPool2D(2, 2))

         model.add(Conv2D(64, (3, 3), activation='relu'))
         model.add(MaxPool2D(2, 2))

         model.add(Conv2D(64, (3, 3), activation='relu'))
         model.add(MaxPool2D(2,2))

         model.add(Flatten())

         model.add(Dense(128,activation ="relu"))
         model.add(Dense(256,activation ="relu"))

         model.add(Dense(26,activation ="softmax"))
```

STEP 11: Model Summary
Now we are getting the model summary that tells us what were the different layers defined in the model & also we save the model using **model.save()** function.

```
In [17]: model.save(r'handwritten_character_recog_model.h5')
```

```
In [16]: model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_4 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_5 (Dense)              (None, 26)                6682
=================================================================
Total params: 122,522
Trainable params: 122,522
Non-trainable params: 0
_________________________________________________________________
```

STEP 12: Compiling and fitting Model

```
In [14]: model.compile(optimizer = Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

- Here we are compiling the model, where we define the optimizing function & the loss function to be used for fitting.
- The optimizing function used is Adam, that is a combination of RMSprop & Adagram optimizing algorithms.
- The dataset is very large so we are training for only a single epoch, however, as required we can even train it for multiple epochs (which is recommended for character recognition for better accuracy).

```
In [17]: model.save(r'handwritten_character_recog_model.h5')
```

STEP 13: Getting the Train & Validation Accuracies & Losses

In the above code segment, we print out the training & validation accuracies along with the training & validation losses for character recognition.

```
In [15]: history = model.fit(x_train, y_training, epochs=5,  validation_data = (x_test,y_testing))
```

```
Epoch 1/5
9312/9312 [==============================] - 816s 66ms/step - loss: 0.2159 - accuracy: 0.9384 - val_loss: 0.1639 -
val_accuracy: 0.9568
Epoch 2/5
9312/9312 [==============================] - 470s 50ms/step - loss: 0.1125 - accuracy: 0.9686 - val_loss: 0.1166 -
val_accuracy: 0.9693
Epoch 3/5
9312/9312 [==============================] - 445s 48ms/step - loss: 0.0981 - accuracy: 0.9730 - val_loss: 0.0964 -
val_accuracy: 0.9742
Epoch 4/5
9312/9312 [==============================] - 430s 46ms/step - loss: 0.0907 - accuracy: 0.9752 - val_loss: 0.1089 -
val_accuracy: 0.9718
Epoch 5/5
9312/9312 [==============================] - 425s 46ms/step - loss: 0.0859 - accuracy: 0.9771 - val_loss: 0.1150 -
val_accuracy: 0.9722
```

STEP 14: Prediction process

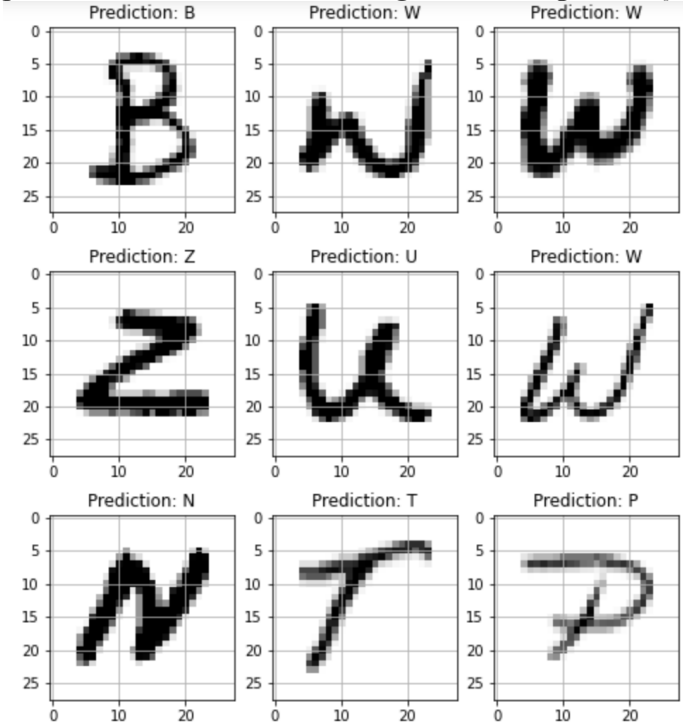
```
In [18]: words = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:'

fig, axes = plt.subplots(3,3, figsize=(8,9))
axes = axes.flatten()

for i,ax in enumerate(axes):
    image = np.reshape(x_test[i], (28,28))
    ax.imshow(image, cmap="Greys")

    pred = words[np.argmax(y_testing[i])]
    ax.set_title("Prediction: "+pred)
    ax.grid()
```

Here we are creating 9 subplots of (3,3) shape & visualize some of the test dataset alphabets along with their predictions, that are made using the **model.predict()** function for text recognition.

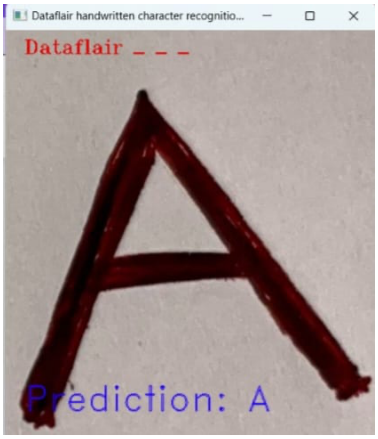


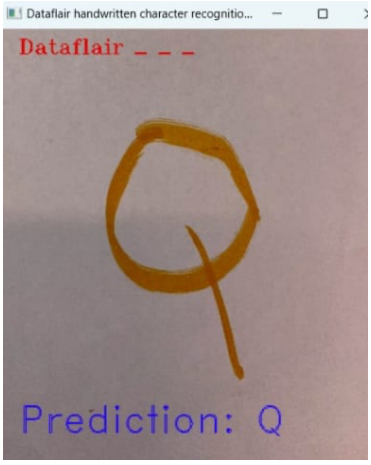
STEP 15: To visualize our model on Customize image

Lets visualize our model on a custom image that is able to predict the alphabet present in that image correctly or not.
This is the code in which you can also see the prediction just by setting the path of your image.

RESULT AND OUTPUT SCREENSHOTS

We have successfully developed our Machine learning project that is Handwritten character recognition (Alphabet Recognition). In this, we have created a CNN model which is working very accurately and efficiently in recognising the alphabet present in a particular image.
In this, we learned to create a Convolutional Neural Network model using Tensorflow, keras and learned how to use various other Machine Learning Libraries.





INFERENCE

Our handwriting recognition system utilized basic computer vision and image processing algorithms (edge detection, contours, and contour filtering) to segment characters from an input image.

Right now this model deals with recognizing only a single character and by widening the range this model could be utilized to recognize more number of characters at a time, this will prove to be an effective tool in several areas where high variance in handwriting styles across people and poor quality of the handwritten text compared to printed text pose significant hurdles in converting it to machine readable text. Nevertheless it's a crucial problem to solve for multiple industries like healthcare, insurance and banking.

Moreover the model only recognizes characters and does not recognise digits and is limited to only one language. The current model may be utilized by proper training model and availability of sufficient and good quality datasets.

Another limitation and huge challenge in image processing is reducing the noise and overcoming the noise and make accurate deductions.

Dealing with connected handwritten characters is still an open area of research ,although there have been significant developments in technology which help in better recognition of handwritten text, HTR is a far from a solved problem compared to OCR and hence is not yet extensively employed in industry. Nevertheless with the pace of technology evolution and with the introduction of models like transformers, we can expect HTR models to become a commonplace soon..

REFERENCES

https://paperswithcode.com/dataset/mnist

https://pyimagesearch.com/2020/08/24/ocr-handwriting-recognition-with-opencv-keras-and-tensorflow/

https://projectgurukul.org/handwritten-character-recognition-ml-project/

https://data-flair.training/blogs/handwritten-character-recognition-neural-network/

https://www.kaggle.com/code/mohammadkumail/handwritten-character-recognition-deep-learning

https://en.wikipedia.org/wiki/Digital_image_processing

https://en.wikipedia.org/wiki/MNIST_database

https://www.researchgate.net/publication/298808334

Handwriting Recognition using Artificial Intelligence Neural Network and Image Processing Sara Aqab[1], Muhammad Usman Tariq
Abu Dhabi School of Management Abu Dhabi, UAE

Machine Learning for Handwriting Recognition
Preetha S, Afrid I M, Karthik Hebbar P, Nishchay S K
Department of ISE,B.M.S. College of Engineering/ VTU, India

Full Page Handwriting Recognition via Image to Sequence Extraction
Sumeet S. Singh and Sergey Karayev
Turnitin, 2101 Webster St #1800, Oakland, CA 94612, USA