HINDI HANDWRITTEN CHARACTER RECOGNITION

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted by

G.Deepak : 2011CS020143

Under the Guidance of

Dr. D. Thiyagarajan

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2023

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, "HINDI HANDWRITTEN CHARACTER RECOGNITION" submitted by G. Deepak (2011CS020143), B. Tech III year I semester, Department of CSE (AI &ML) during the year 2022-2023. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

INTERNAL GUIDE | **HEAD OF THE DEPARTMENT**

Dr. D. Thiyagarajan | **Dr.Thayyaba Khatoon**

CSE(AI&ML)

ABSTRACT

Optical Character Recognition is a technique by which you can automatically recognize the characters with an optical mechanism. OCR technology allows you the recognition of printed or handwritten text documents. Main aim of this research is to prepare a recognition system which can be used for the recognition of offline handwritten Hindi characters. For this proposed system Support Vector Machine is used as classifier and Diagonal feature extraction approach is used to extract features.

CONTENTS

INTRODUCTION

1.1 PROBLEM DEFINITION:

Handwriting recognition in images is a research area that attempts to develop a computer system to transform the text present in the paper format to electronic format. The high variance in handwriting styles across people and poor handwritten text quality compared to printed text poses significant hurdles in converting it to machine-readable text. In offline handwriting recognition, the text is analyzed after being written. *Machine Learning* has been widely used to recognize handwriting

1.2 OBJECTIVE OF PROJECT:

The handwritten digit recognition is the capability of computer applications to recognize the human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different shapes and sizes. The handwritten digit recognition **sy**stem is a way to tackle this problem which uses the image of a digit and recognizes the digit present in the image.

1.3 LIMITATIONS OF THE PROJECT:

- Using of learning model can be a difficult task when we have many images in the dataset.
- More number of images may also lead to less accuracy of the model.

2.ANALYSIS

2.1 INTRODUCTION:

It is easy to recognize typed characters but handwritten characters cannot be recognized with 100% accuracy by computer machine. So, Handwritten character recognition is still a difficult task. Handwritten character recognition is further divided into two domains i.e Offline handwritten character recognition and Online handwritten character recognition.Offline handwritten character recognition In case of offline character recognition, the typed/handwritten characters are scanned and then converted into binary or gray scale image. Then feature extraction and recognition process is carried over the binary image. Offline character recognition is a more challenging and difficult task as there is no timing information about character strokes is available. Therefore offline character recognition is considered as a more challenging task then its online counterpart.

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

2.2.1 Software Requirements:

- Jupyter Notebook.
- Anaconda navigator.
- Windows operating system will be used during development process. The system will be implemented using in python language. For image processing part we use SVM, NumPy and pandas' libraries will be used respectively

2.2.2 Hardware Requirements:

- RAM - 4 GB
- CPU - Intel i3 4th Generation
- GPU - Nvidia GeForce 740M (CUDA Compatible GPU)

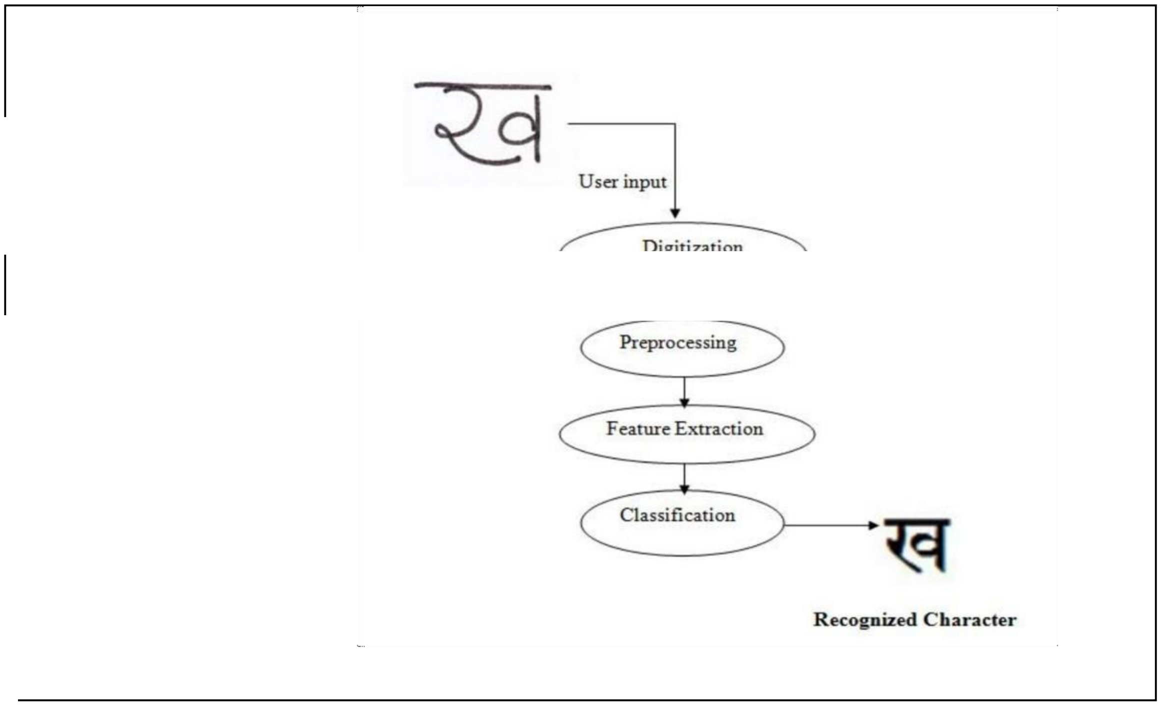
2.3 EXISTING SYSTEM

- Support vector machine is a supervised learning system and used for classification and regression problems.
- Support vector machine is extremely favored by many as it produces notable correctness with less computation power.
- It is mostly used in classification problems. We have three types of learning supervised, unsupervised, and reinforcement learning.

2.4 PROPOSED SYSTEM

- To check the usefulness of our method for extracting the features of the individual writers, we created our own database. Writer samples are collected, scanned and stored as image files.
- From each writer, 5 samples of handwriting is collected at different time of the day to take care of possible variation in their writing.

2.5 ARCHITECTURE

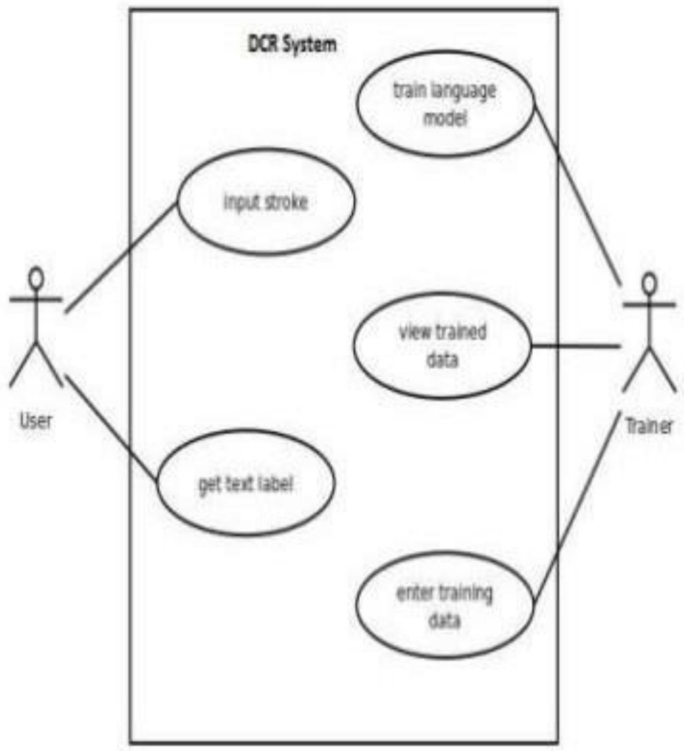


3.DESIGN

3.1INTRODUCTION

It is easy to recognize typed characters but handwritten characters cannot be recognized with 100% accuracy by computer machine. So, Handwritten character recognition is still a difficult task. Handwritten character recognition is further divided into two domains i.e Offline handwritten character recognition and Online handwritten character recognition

3.2UML/ USECASE DIAGRAM



3.3DATA SET DESCRIPTIONS

Devanagri is composed of two two Sanskrit words, “deva” and “nagri”. Deva means God and nagri means city. Hindi is written using Devanagari script. This script is used to write many other languages, such as Nepali, Marathi etc. Devanagari consists of 11 vowels and 33 consonants. There are no capital letters. Devanagari is written from left to right. [3] Hindi is an Indo-Aryan language which is written using devanagri script .

The column names are

क	ख	ग	घ	ङ			
ka	kha	ga	gha	nga			
च	छ	ज	झ				
cha	chh	ja	jhh				
ट	ठ	ड	ढ	ण			
ta	tha	da	dha	ṇa			
त	थ	द	ध	न			
ta	tha	da	dha	na			
प	फ	ब	भ	म			
pa	pha	ba	bha	ma			
य	र	ल	व	श	ष	स	ह
ya	ra	la	va	sh	shh	sa	ha

3.4DATA PREPROCESSING TECHNIQUES

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

It involves below steps:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

3.5METHODS AND ALGORITHMS

SVM of supervised machine learning algorithm are being using in implementing the project. They are

SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.SVMs are currently among the best performers for a number of classification tasks. Currently, SVM is widely used in object detection & recognition, text recognition, biometrics, speech recognition, etc. SVM is based on binary classification, means at a time it can classify two class groups.

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. Most classification tasks, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). [2] In all the experiments, the results have shown that at character level, SVM recognition rates are significantly better due to structural risk minimization implemented by maximizing margin of separation in the decision function

BUILDING A MODEL

Steps Involved:

- Importing Libraries
- Loading the train and test datasets
- Data Analysis like Descriptive statistics and Correlation
- Data Pre-processing (missing value replacement and dropping off the columns)
- Fitting the data to a classifier
- Predicting the attack of the test data
- Accuracy score

3.7Evaluation

The model is being evaluated based on the

1. Accuracy score
2. Classification Report
3. on the basis of samples of handwritten characters collected, the average accuracy of character recognition is 94.00%DEPLOYMENT AND RESULTS

4. DEPLOYMENT AND RESULTS

4.1INTRODUCTION

It is easy to recognize typed characters but handwritten characters cannot be recognized with 100% accuracy by computer machine. So, Handwritten character recognition is still a difficult task. Handwritten character recognition is further divided into two domains i.e Offline handwritten character recognition and Online handwritten character recognition.

4.2SOURCE CODE

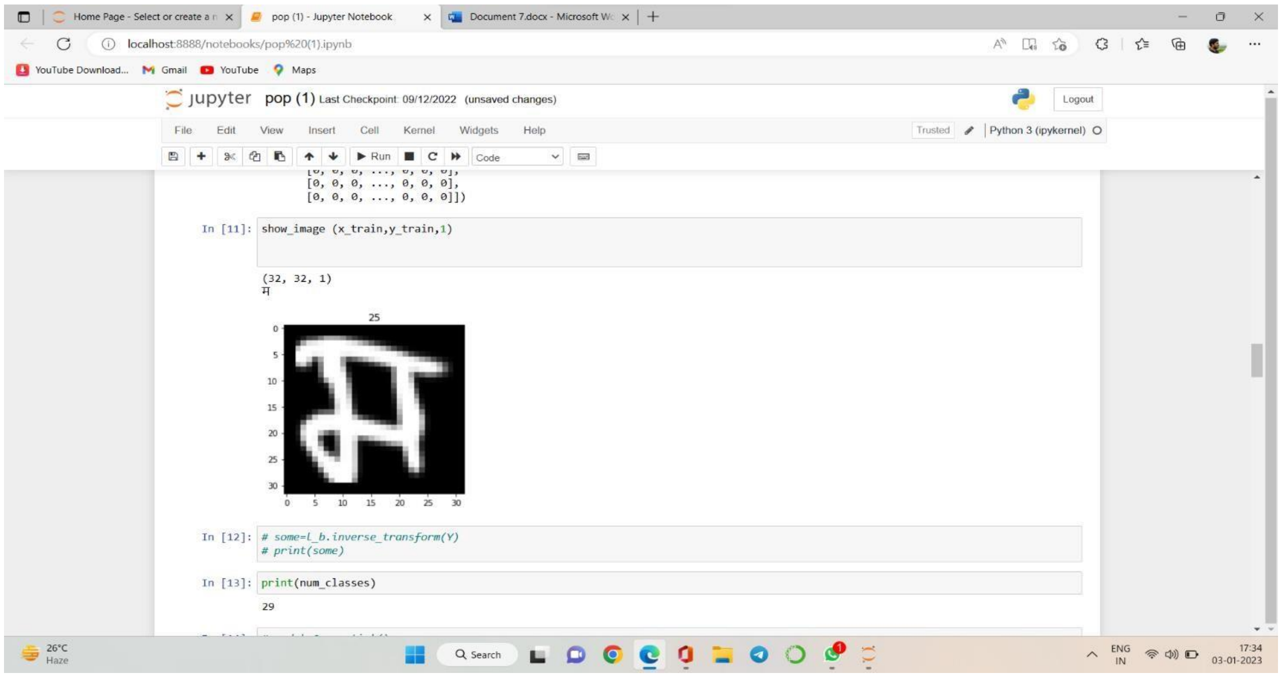
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelBinarizer as lb
#kera
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten,BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils
import sklearn.metrics as metrics
import seaborn as sns

label=pd.read_csv("label_consonants_vowels.csv")
label.drop(columns='Unnamed: 0',inplace=True)
df=pd.read_csv('Final_data.csv')
df.drop(columns='Unnamed: 0',inplace=True)
# df=df.sample(frac=1)
# train_x=np.transpose(df[:,1:])
```

```
labeling = dict(zip(label.Key,label.Hindi))

# labeling=labeling[:,-1]
print(labeling)

def show_image(df_x,df_y,n):
    k=(df_x[n]*255).astype(int)

    print(k.shape)

    plt.imshow(k, cmap=plt.get_cmap('gray'))
    la=l_b.inverse_transform(df_y[n:n+1])
    la=list(la)
    k=labeling[la[0]]
    print(k)
    plt.title(la[0]



train_x=np.transpose(df.values[:,1:].reshape(len(df),32,32,1),axes=[0,1,2,3])

train_y=df['character']

num_classes=train_y.nunique()

train_y=np.asarray(train_y)


total=len(train_y)

total=int(total/50)

print(total)


l_b=lb()

Y=l_b.fit_transform(train_y)


Y


show_image (x_train,y_train,1)


print(num_classes)
```

```
# model=Sequential()
# model.add(Conv2D(32, kernel_size = 3, activation='relu', input_shape = (28, 28, 1)))
# model.add(BatchNormalization())
# model.add(Conv2D(32, kernel_size = 3, activation='relu'))
# model.add(BatchNormalization())
# model.add(Conv2D(32, kernel_size = 5, strides=2, padding='same', activation='relu'))
# model.add(BatchNormalization())
# model.add(Dropout(0.4))

# model.add(Conv2D(64, kernel_size = 3, activation='relu'))
# model.add(BatchNormalization())
# model.add(Conv2D(64, kernel_size = 3, activation='relu'))
# model.add(BatchNormalization())
# model.add(Conv2D(64, kernel_size = 5, strides=2, padding='same', activation='relu'))
# model.add(BatchNormalization())
# model.add(Dropout(0.4))

# model.add(Conv2D(128, kernel_size = 4, activation='relu'))
# model.add(BatchNormalization())
# model.add(Flatten())
# model.add(Dropout(0.4))
# model.add(Dense(29, activation='softmax'))

# model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
# model.summary()

model = Sequential()
model.add(Conv2D(input_shape=(32,32,1),filters=64,kernel_size=(3,3),padding="same",
activation="relu"))
model.add(BatchNormalization())
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.4))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.4))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))

model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.4))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(Flatten())
model.add(Dropout(0.4))
model.add(Dense(512,activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(num_classes,activation="softmax"))

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.summary()

checkpoint = ModelCheckpoint('/content/drive/MyDrive/train/Model.h5',
                             monitor='val_acc',
                             verbose=1,
                             save_best_only=False,
                             mode='auto')

training_data_generator = data_generator_aug.flow(x_train,y_train,batch_size=200, subset='training')
validation_data_generator = data_generator.flow(x_train,y_train,batch_size=200,subset='validation')
history = model.fit(training_data_generator, epochs=1,
validation_data=validation_data_generator,validation_steps=1,callbacks=[checkpoint])

x_test.shape

t=model.predict(x_train[3:4])
k=l_b.inverse_transform(t)
print(k)

labeling[k[0]]

show_image(x_train,y_train,3)

p=y_test[0:1]
c=l_b.inverse_transform(p)

q = len(history.history['accuracy'])

plt.figsize=(10,10)
```

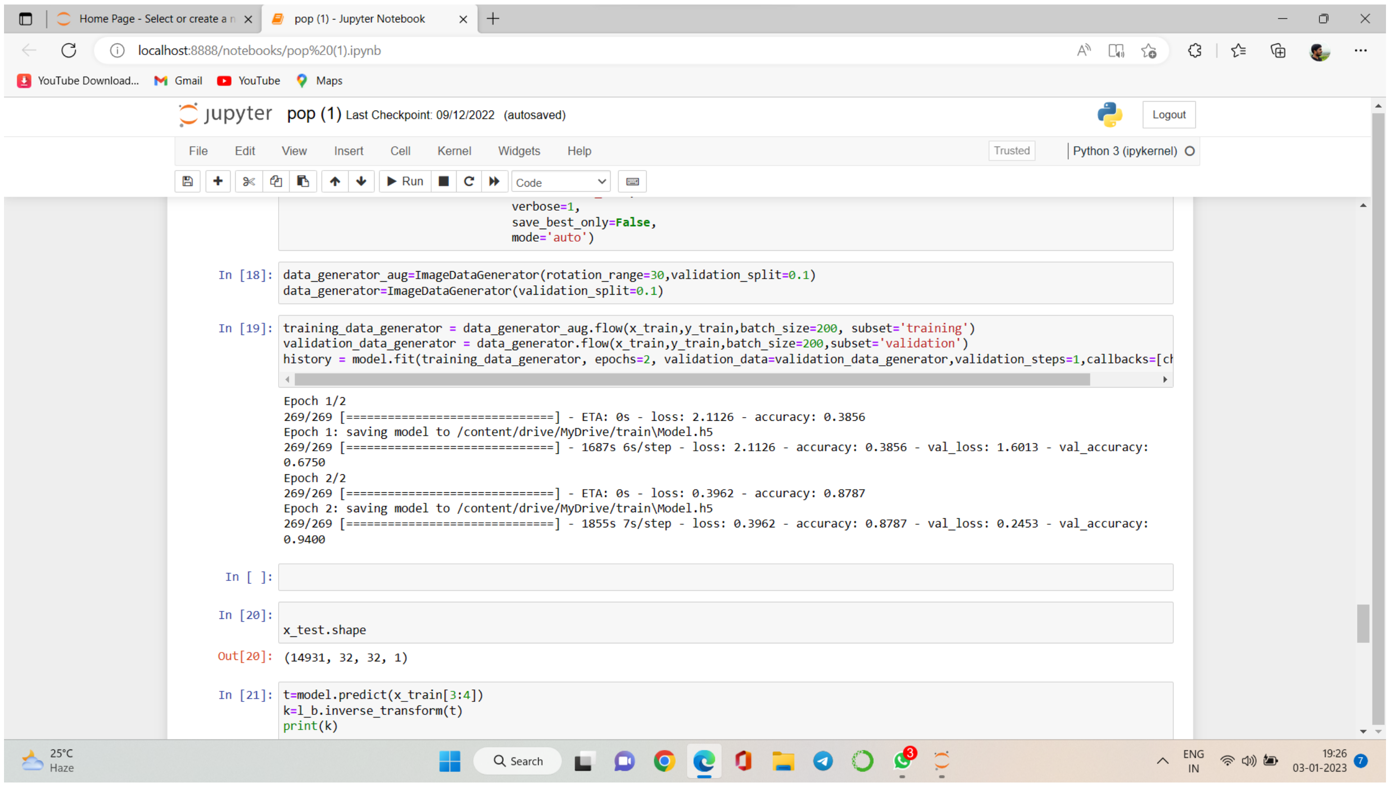
```
sns.lineplot(x = range(1,1+q),y = history.history['accuracy'], label='Accuracy')
sns.lineplot(x = range(1,1+q),y = history.history['val_accuracy'], label='Val_Accuracy')
plt.xlabel('epochs')
plt.ylabel('Accuray')score=model.evaluate(x_test,y_test,verbose=1)

q = len(history.history['accuracy'])

plt.figsize=(10,10)
sns.lineplot(x = range(1,1+q),y = history.history['accuracy'], label='Accuracy')
sns.lineplot(x = range(1,1+q),y = history.history['val_accuracy'], label='Val_Accuracy')
plt.xlabel('epochs')
plt.ylabel('Accuray')
```

4.3FINAL RESULTS



```
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])

In [11]: show_image (x_train,y_train,1)

(32, 32, 1)
म

In [12]: # some=l_b.inverse_transform(Y)
         # print(some)

In [13]: print(num_classes)

29
```

Home Page - Select or create a n
pop (1) - Jupyter Notebook
Document 7.docx - Microsoft W
localhost:8888/notebooks/pop%20(1).ipynb
YouTube Download...
Gmail
YouTube
Maps
jupyter pop (1) Last Checkpoint: 09/12/2022 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
25
26°C Haze
Search
ENG IN
17:34 03-01-2023



```
                                  verbose=1,
                                  save_best_only=False,
                                  mode='auto')
```

```
In [18]: data_generator_aug=ImageDataGenerator(rotation_range=30,validation_split=0.1)
         data_generator=ImageDataGenerator(validation_split=0.1)
```

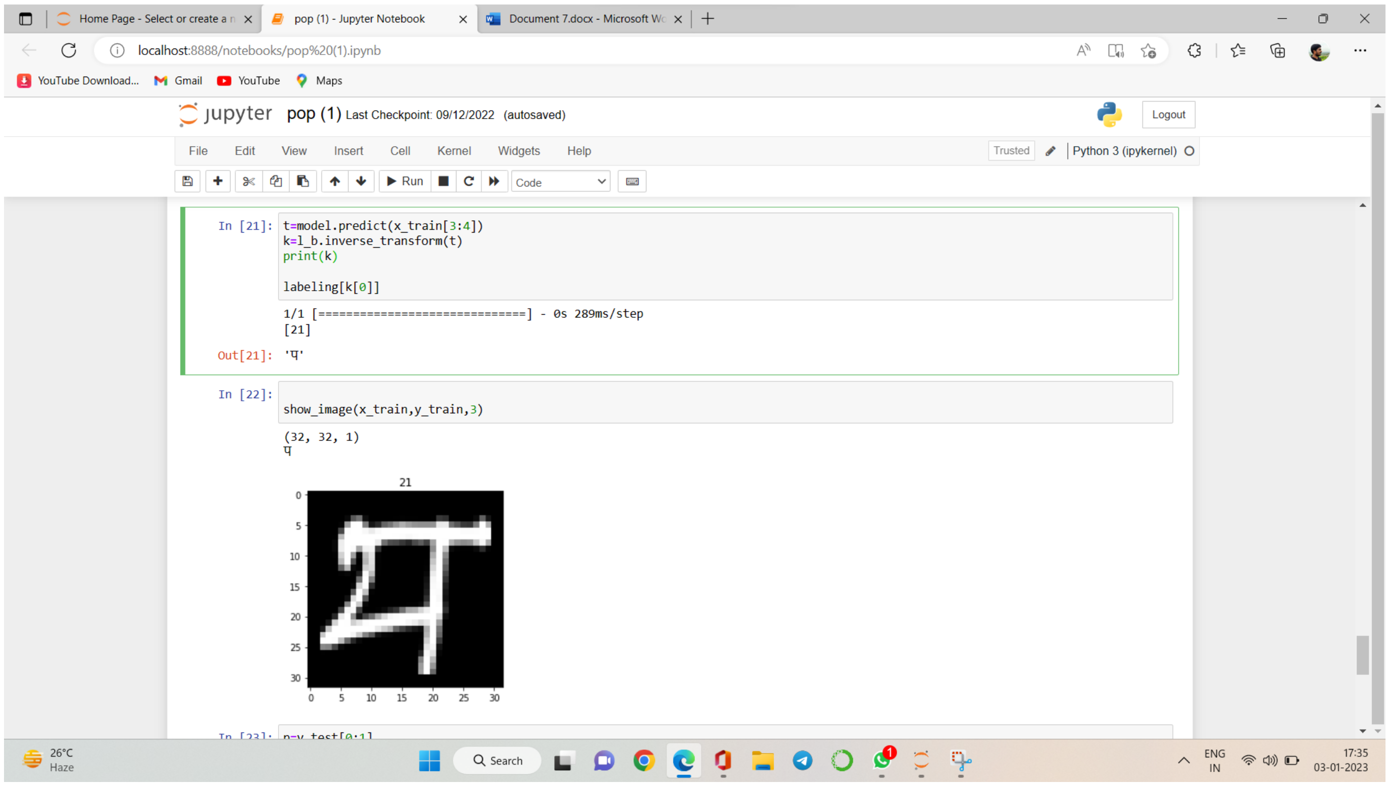
```
In [19]: training_data_generator = data_generator_aug.flow(x_train,y_train,batch_size=200, subset='training')
         validation_data_generator = data_generator.flow(x_train,y_train,batch_size=200,subset='validation')
         history = model.fit(training_data_generator, epochs=2, validation_data=validation_data_generator,validation_steps=1,callbacks=[ch
```

```
Epoch 1/2
269/269 [==============================] - ETA: 0s - loss: 2.1126 - accuracy: 0.3856
Epoch 1: saving model to /content/drive/MyDrive/train\Model.h5
269/269 [==============================] - 1687s 6s/step - loss: 2.1126 - accuracy: 0.3856 - val_loss: 1.6013 - val_accuracy:
0.6750
Epoch 2/2
269/269 [==============================] - ETA: 0s - loss: 0.3962 - accuracy: 0.8787
Epoch 2: saving model to /content/drive/MyDrive/train\Model.h5
269/269 [==============================] - 1855s 7s/step - loss: 0.3962 - accuracy: 0.8787 - val_loss: 0.2453 - val_accuracy:
0.9400
```

```
In [ ]:
```

```
In [20]: x_test.shape

Out[20]: (14931, 32, 32, 1)
```

```
In [21]: t=model.predict(x_train[3:4])
         k=l_b.inverse_transform(t)
         print(k)
```



```
In [21]: t=model.predict(x_train[3:4])
         k=l_b.inverse_transform(t)
         print(k)

         labeling[k[0]]

         1/1 [==============================] - 0s 289ms/step
         [21]

Out[21]: 'प'
```

```
In [22]: show_image(x_train,y_train,3)

         (32, 32, 1)
         प
```

```
In [23]: p=y_test[0:1]
```

5.CONCLUSION

5.1PROJECT CONCLUSION

From the results it can be concluded that combination of SVM classifier and diagonal feature extraction approach is best method for the recognition of handwritten characters.

5.2FUTURE SCOPE

The Future work may involve the recognition of words and complete sentences as well as speech can be synthesized for the individual character that is recognized by the system.

6.REFERENCES:

[1] Anuj Sharma, R.K. Sharma, Rajesh Kumar, “Online Handwritten Gurmukhi Character Recognition”, Ph.D. Thesis, Thapar University, 2009 [Online].

[2] Abdul Rahim Ahmad, Christian Viard-Gaudin, Marzuki Khalid, Emilie Poisson (2004) “Online Handwriting Recognition using Support Vector Machine” TENCON 2004 IEEE Region 10 Conference, Vol. No. 1 , pp 311-314.

[3] J. Pradeep, E.Srinivasan, S.Himavathi (Oct 2010) “Diagonal Feature Extraction Based Handwritten Character System Using Neural Network.” International Journal of Computer Applications (0975 – 8887)

[4] C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):121– 167, 1998.1, 2.2, 2.2

[5] N. Cristianini and J. Shawe−Taylor. Support Vector Machines. Cambridge University Press, 2000. 1, 2.2, 2.2

Websites

https://www.javatpoint.com

https://github.com

http://hindilanguage.info/devanagari