Optical Character Recognition for Hand-Written Hindi Text

A PROJECT REPORT

Submitted by
19BCB0030 Rupin Patel
19BCB0034 Dipti Lulla
19BCB0052 Palak Thakur
19BCB0077 Vijeta Priya

Course Code: CSE3013
Course Title: Artificial Intelligence

Under the guidance of
Dr. S. Anto
Associate Professor, SCOPE,
VIT , Vellore.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2022

ABSTRACT

1. Introduction

We propose an Optical Character Recognition system based on deep learning algorithms to improve the accuracy and analyze each of the network architectures. This project outlines the many approaches that may be taken in order to create a deep learning solution for handwritten Hindi and English letters recognition using various methods. We are using convolutional neural networks to detect the handwritten text in image and focus on applications of automatic digital text identification . And this model will detect handwritten text, convert it to corresponding digital Hindi letters and paste the recognised text on the same coordinates of detection without changing the format of the document.

One of the most promising applications of character recognition in computer vision is optical character recognition and documents monitoring which is becoming increasingly important in our daily lives and as well as in our professional lives. Now also there are some forms and documents which are handwritten, and a person must fill in the details manually for storing on an online database which is a very time-consuming process. And specially in India some people fill handwritten forms and document in their native languages which is very difficult job for officers to recognize and understand. There are some present technologies for OCR but all are generally for English handwritten text. But as in India , Hindi being our national language which has very complex text formations and different letters it is almost impossible to use simple OCR. Since EasyOCR python library has an ability to recognise Hindi text from image but with constraint that text in image should be digitally printed, which is in-efficient for Hindi because people generally prefer handwritten Hindi format in forms and documents. Examples of present OCR apps: Office lens, Text-Fairy, Google Lens, Adobe Scan, etc. All these apps are based on OCR technology but generally providing more accuracy for English words and in that also generally which are digitally printed on scanned images or documents.

2. Literature Survey

2.1. Research Paper

2.1.1. Optical Character Recognition for Hindi

Proposed solution: In this research work different preprocessing operations like conversion of gray scale images to binary images, image rectification and segmentation are considered in order to design this system.

In the thinning algorithm eight neighborhood concepts are used to find the skeleton of the character. Instead of eliminating one pixel at a time the unwanted pixel of the same region is identified and then deleted at once which decreases the time required to find the skeleton of the image. Then the character is segmented to detect various lines.

Drawback:A simple form of database is used to recognize a character. With the current database the character recognition is good but to develop good quality OCR software the database should be fully organized.

2.1.2. Segmentation of Devanagari Handwritten Characters

Proposed Solution: This research work was directed towards development of a novel algorithm for Devanagari character segmentation for Hindi. Segmentation and Recognition of this particular language is difficult because of the presence of complex connected characters and the presence of shirorekha. The proposed approach is in the manner that it follows a particular order starting from scanning of devnagri script, noise reduction, skew correction, thinning, grayscale conversion, binarization, line segmentation, word segmentation, removing shirorekha and character segmentation.

Drawback: For segmentation of connected components accuracy is only 90%.

2.1.3. An improved faster RCNN model for handwritten character recognition

Proposed Solution: An effective and efficient HDR system, introducing a customized faster regional convolutional neural network (Faster-RCNN). This approach comprises three main steps. Initially, we develop annotations to obtain the region of interest. Then, an improved Faster-RCNN is employed in which DenseNet-41 is introduced to compute the deep features. Finally, the regression and classification layer is used to localize and classify the digits into ten classes. The performance of the proposed method is analyzed on the standard MNIST database, which is diverse in terms of changes in lighting conditions, chrominance, shape and size of digits, and the occurrence of blurring and noise effects, etc.

Drawback: Effective detection and classification of numerals is still a challenging task due to people's varying writing styles and the presence of blurring, distortion, light and size variations in the input sample. Also, if there doesn't exist a dictionary, we cannot use RCNN technology.

2.2. Problem Definition

One of the most promising applications of character recognition in computer vision is optical character recognition and documents monitoring which is becoming increasingly important in our daily lives and as well as in our professional lives. Now also there are some forms and documents which are handwritten, and a person must fill in the details manually for storing on an online database which is a very time-consuming process. And specially in India some people fill handwritten forms and documents in their native languages which is a very difficult job for officers to recognize and understand.

3. Overview of the Work

3.1. Objectives of the Project

Optical Character Recognition, or OCR, is a technology that enables you to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data. The objective of OCR is to achieve modification or conversion of any form of text or text-containing documents such as handwritten text, printed or scanned text images, into an editable digital format for deeper and further processing. Our OCR model will detect handwritten text, convert it to corresponding digital Hindi letters and paste the recognised text on the same coordinates of detection without changing the format of the document.

3.2. Software Requirements

- Python 3.8
- Anaconda
- Jupyter Notebook
- Tensorflow
- Pytorch
- Open-CV
- Roboflow(dataset labeling)
- CUDA

3.3. Hardware Requirements

- Any operating system namely Windows(preferable Windows 10), LINUX, MAC.
- RAM: 8 GB
- Processor: Intel Core i5 and above.
- GPU for performance

4. System Design

4.1. Algorithms

We are using convolutional neural networks to detect the handwritten text in image and focus on applications of automatic digital text identification. And this model will detect handwritten text,convert it to corresponding digital Hindi letters and paste the recognised text on the same coordinates of detection without changing the format of the document.

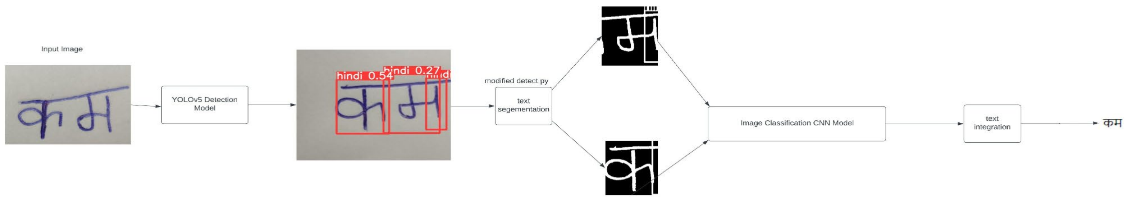
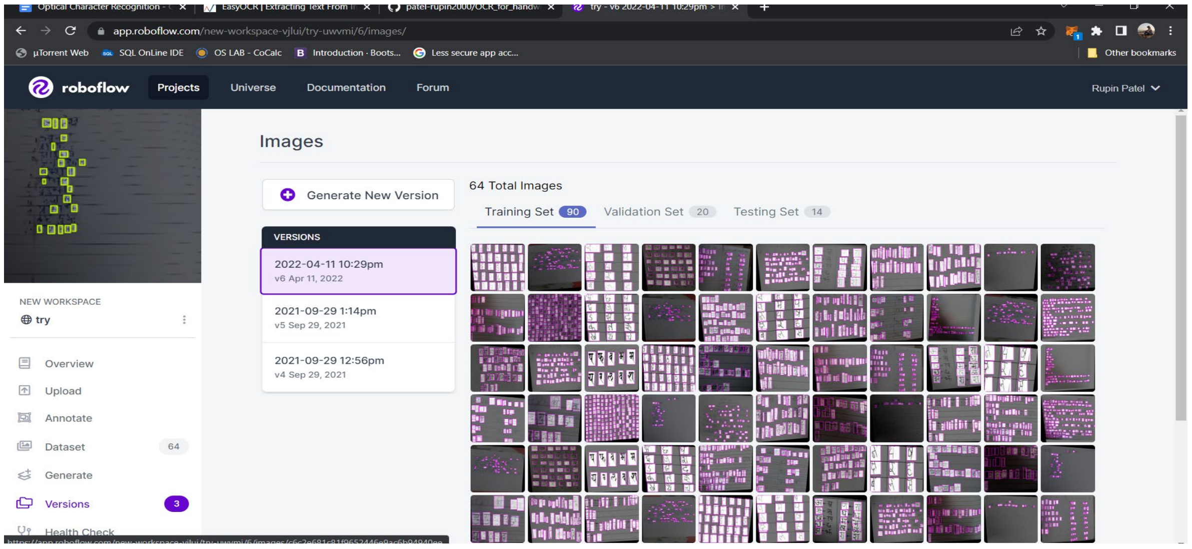
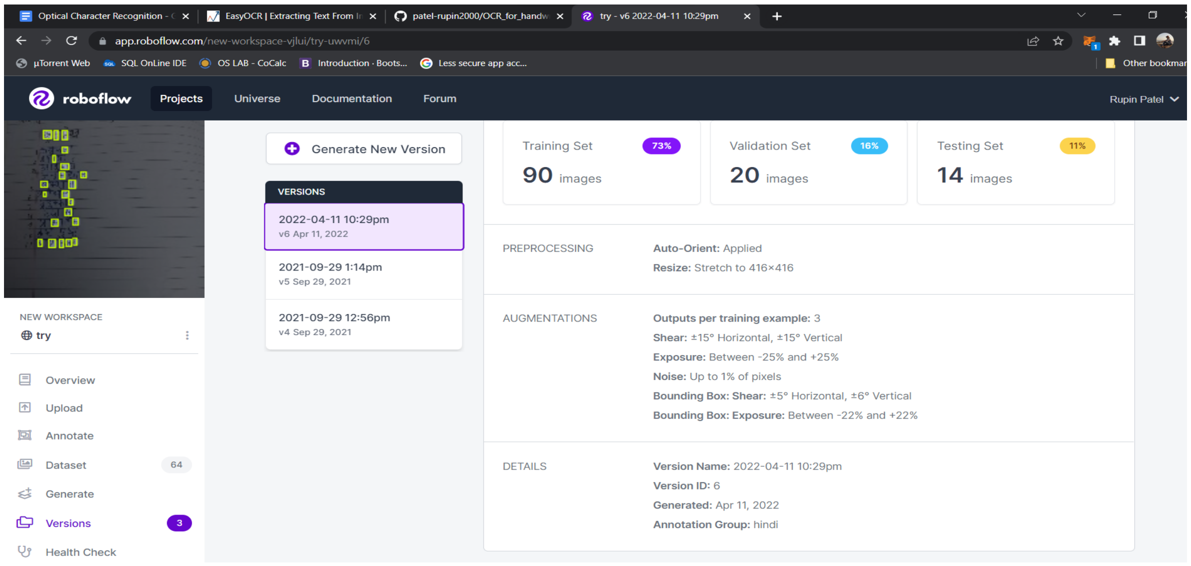


Fig-1 Overall Input Flow

Image object detection YOLO (You Only Look Once) - It is an object detection algorithm. It predicts classes and bounding boxes for the whole image in one run of the Algorithm.

Now for YOLO we must label dataset and we have collected 50 normal random handwritten hindi paragraph written on paper as image and label then on Roboflow.





Training Set 73% 90 images

Validation Set 16% 20 images

Testing Set 11% 14 images

PREPROCESSING
Auto-Orient: Applied
Resize: Stretch to 416×416

AUGMENTATIONS
Outputs per training example: 3
Shear: ±15° Horizontal, ±15° Vertical
Exposure: Between -25% and +25%
Noise: Up to 1% of pixels
Bounding Box: Shear: ±5° Horizontal, ±6° Vertical
Bounding Box: Exposure: Between -22% and +22%

DETAILS
Version Name: 2022-04-11 10:29pm
Version ID: 6
Generated: Apr 11, 2022
Annotation Group: hindi

We Split and add some filters by doing hit and trial on every ratio of distribution of test and train images.Applying filters such as shear, blur, noise and also exposure.

Image recognition Convolutional Neural Network (ConvNet/CNN) - It is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.This model is train on normal devanagari dataset available on Kaggle.CNN mainly consists of convolution layer and pooling layer connected to dense layer through flatten layer and finally to output layer for result.

Convolution layer:

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

Pooling Layer:

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

All the layers have activation function as ReLu except output layer .The Rectified Linear Unit (ReLu) has become very popular in last few years.It computes the function f(K)=max(0,K). In other word, the activation is simply threshold at zero.In comparison to sigmoid and tanh ,ReLu is more reliable and accelerates the convergence by six times.Unfortunately, a con is that ReLU can be fragile during training.A large gradient flowing through it can update it in such a way that the neuron will never get further updated. However, we can work with this by setting a proper learning rate.The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model. This is why softmax is sometimes referred to as a multinomial logistic regression.

4.2. Block Diagrams

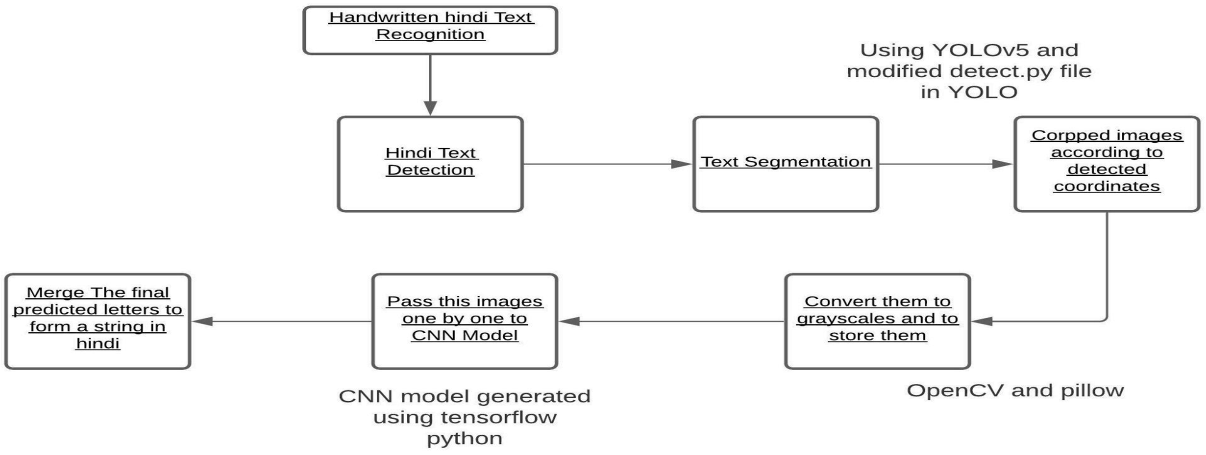

Handwritten hindi Text Recognition
Hindi Text Detection
Text Segmentation
Using YOLOv5 and modified detect.py file in YOLO
Corpped images according to detected coordinates
Convert them to grayscales and to store them
OpenCV and pillow
Pass this images one by one to CNN Model
CNN model generated using tensorflow python
Merge The final predicted letters to form a string in hindi

Fig-2 Data Flow

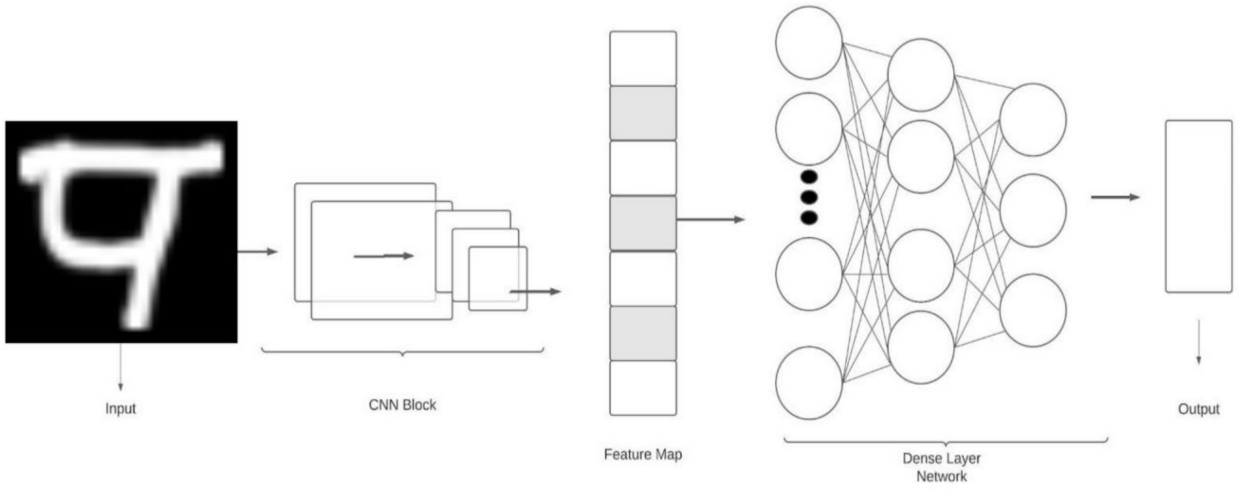


Fig-3 CNN Architecture

5. Implementation

5.1. Description of Modules/Programs

YOLOv5 and Pytorch:

YOLOv5 is a family of compound-scaled object detection models trained on the COCO dataset, and includes simple functionality for Test Time Augmentation (TTA), model ensembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite.PyTorch is an open source machine learning (ML) framework based on the Python programming language and the Torch library. It is one of the preferred platforms for deep learning research. The framework is built to speed up the process between research prototyping and deployment.PyTorch is similar to NumPy and computes using tensors that are accelerated by graphics processing units (GPU). Tensors are arrays, a type of multidimensional data structure, that can be operated on and manipulated with APIs. The PyTorch framework supports over 200 different mathematical operations.The popularity of PyTorch continues to rise as it simplifies the creation of artificial neural network (ANN) models. PyTorch is mainly used for applications of research, data science and artificial intelligence (AI).

Tensorflow:

TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.It is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research fascinating ideas on artificial intelligence, the Google team created TensorFlow.It is used for preprocessing and training and testing split.The main role of tensorflow it helps to create CNN layers and with integrations of activation functions and optimizers.

Open-CV:

OpenCV is the huge open-source library for 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 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.

Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Pandas:

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python.

5.2. Source Code

Yolov5 Model for hindi letter detection and segmentation:

```
# -*- coding: utf-8 -*-
"""yolov5.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1lAPSacvlub3w7WTBg7n5n81LFDsHOzf
4
"""

from google.colab import drive
drive.mount('/content/drive')

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/drive/MyDrive/
!curl -L "https://app.roboflow.com/ds/0RpDxXMddW?key=4atw0cK3tX" >
roboflow.zip; unzip roboflow.zip; rm roboflow.zip

!git clone https://github.com/ultralytics/yolov5

# Commented out IPython magic to ensure Python compatibility.
# %cat /content/drive/MyDrive/data.yaml
import yaml
with open("/content/drive/MyDrive/data.yaml",'r') as stream:
  num_classes=str(yaml.safe_load(stream)['nc'])

import torch
from IPython.display import Image, clear_output

clear_output()
print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")

# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard
# %tensorboard --logdir runs/train

# Commented out IPython magic to ensure Python compatibility.
# %pip install -q wandb
import wandb
wandb.login()

# Commented out IPython magic to ensure Python compatibility.
# %%time
# %cd /content/drive/MyDrive/yolov5
# !pip install requirements.txt
# !python train.py --img 832 --batch 16 --epochs 300 --data '/content/drive/MyDrive/data.yaml' --cfg "/content/drive/MyDrive/yolov5/models/yolov5s.yaml" --weights '' --adam --name yolo5s_results --nosave --cache

# Commented out IPython magic to ensure Python compatibility.
# %ls /content/drive/MyDrive/yolov5/runs/train/yolo5s_results6/weights

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/drive/MyDrive/yolov5
!pip install requirements.txt
!python detect.py --weights "/content/drive/MyDrive/yolov5/runs/train/yolo5s_results4/weights/last.pt" --img 416 --conf 0.2 --source "/content/"

import glob
from IPython.display import Image,display
```

```
for imageName in
glob.glob('/content/drive/MyDrive/yolov5/runs/detect/exp22/*.jpg'):
  display(Image(filename=imageName))
  print('\n')

import re
path='/content/darknet/result.txt'
myfile=open(path,'r')
lines=myfile.readlines()
pattern= "class_name"

for line in lines:
  if re.search(pattern,line):
    Cord_Raw=line
Cord=Cord_Raw.split("(")[1].split(")")[0].split("  ")
```

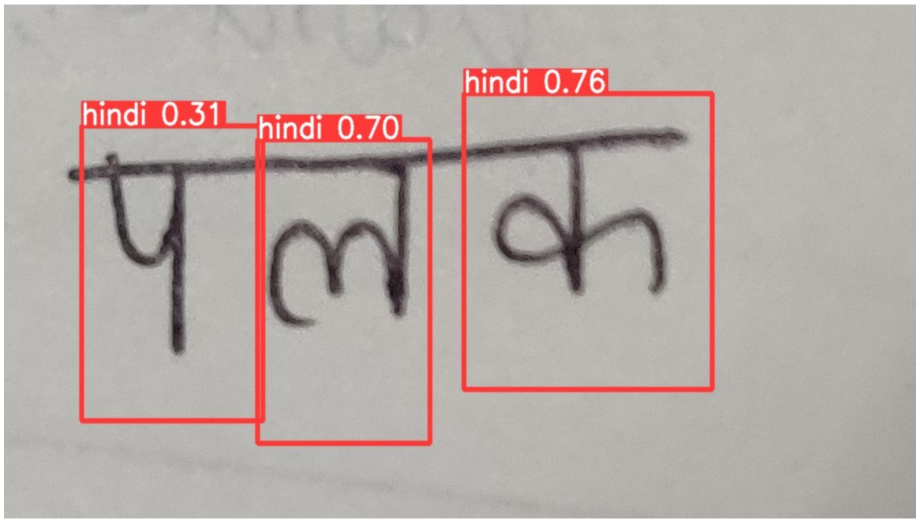


Fig-4 Text Detection

CNN Model Train on devanagari dataset for Image recognition:

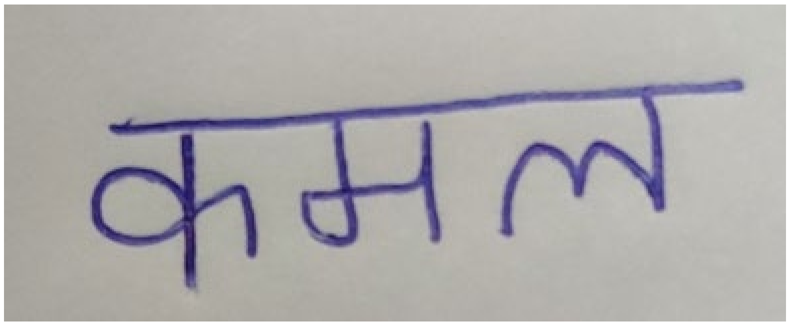
```
import pandas as pd
df = pd.read_csv('data.csv')
df.size
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
from sklearn.preprocessing import LabelBinarizer
binencoder = LabelBinarizer()
y = binencoder.fit_transform(y)
X_images = X.values.reshape((92000),32,32)
import matplotlib.pyplot as plt
```

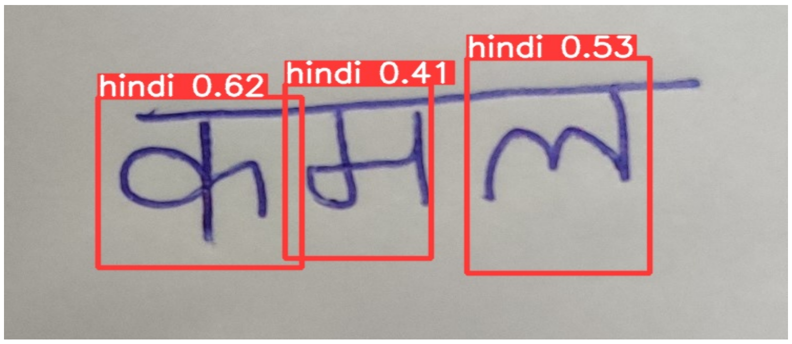
```
plt.imshow(X_images[0])
plt.show()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_images, y, test_size = 0.2,
random_state=90)
X_train = X_train/255
X_test = X_test/255

#change the dimension from 3 to 5
X_train = X_train.reshape(X_train.shape[0], 32,32,1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 32,32,1).astype('float32')
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dense,Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
from PIL import Image
import numpy as np
import os
model = Sequential()
model.add(Conv2D(32, (4, 4), input_shape=(X_train.shape[1], X_train.shape[2], 1),
activation='relu',kernel_initializer='he_uniform'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_uniform'))
model.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_uniform'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu',kernel_initializer='he_uniform'))
model.add(Dropout(0.5))
model.add(Dense(46, activation='softmax'))
opt = SGD(learning_rate=0.01, momentum=0.9)
model.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
history=model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100,
batch_size=32)
```

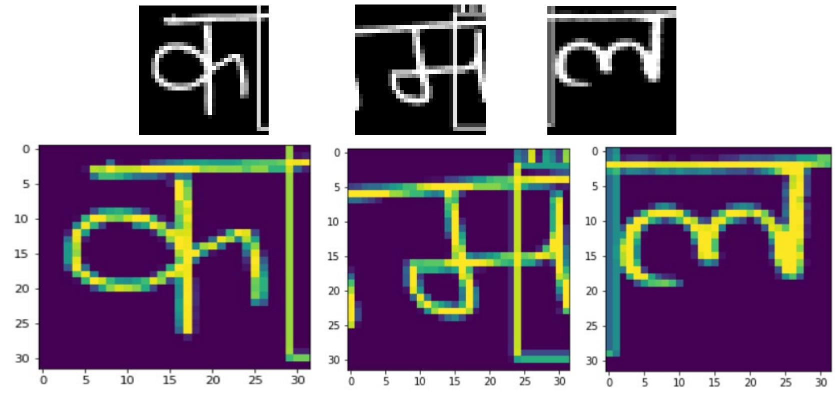
```
import numpy as np
np.save('history_002.npy',history.history)
model.save("OCR_002.h5")
from tensorflow.keras.models import load_model
model = load_model ("OCR_002.h5")
d={'character_01_ka':'क','character_25_ma':'म','character_21_pa':'प','character_28_la'
:'ल'}
from PIL import Image
import glob
import numpy as np
import matplotlib.pyplot as plt
image_list = []
s=""
for                              filename                              in
glob.glob(r'H:\Projects\Manthan\DevanagariHandwrittenCharacterDataset\cropped
_img/*.png'):
   img=Image.open(filename).convert('L')
   x = np.array(img)
   plt.imshow(x)
   plt.show()


   imgTrans =x.reshape(1,32,32,1)
   imgTrans.shape

   predictions = model.predict(imgTrans)
   print(d[binencoder.classes_[np.argmax(predictions)]])
   s=s+d[binencoder.classes_[np.argmax(predictions)]]
print(s[::-1])
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],label="train_accuracy")
plt.plot(history.history['val_accuracy'],label="test_accuracy")
plt.legend()
```

5.3. Test cases

1)



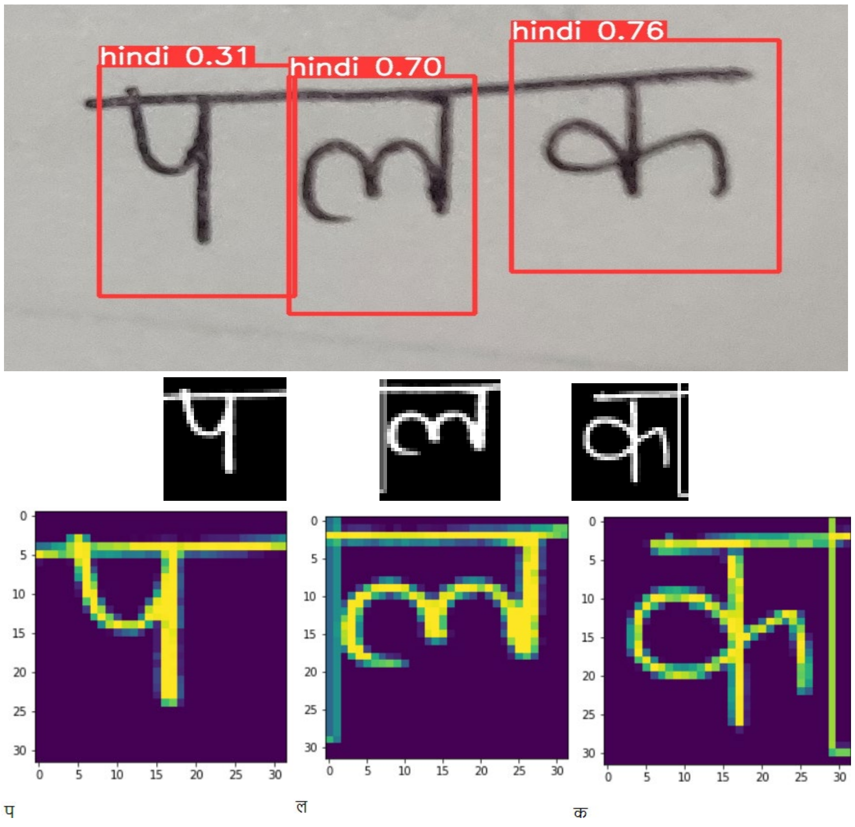




क म ल

कमल

2)

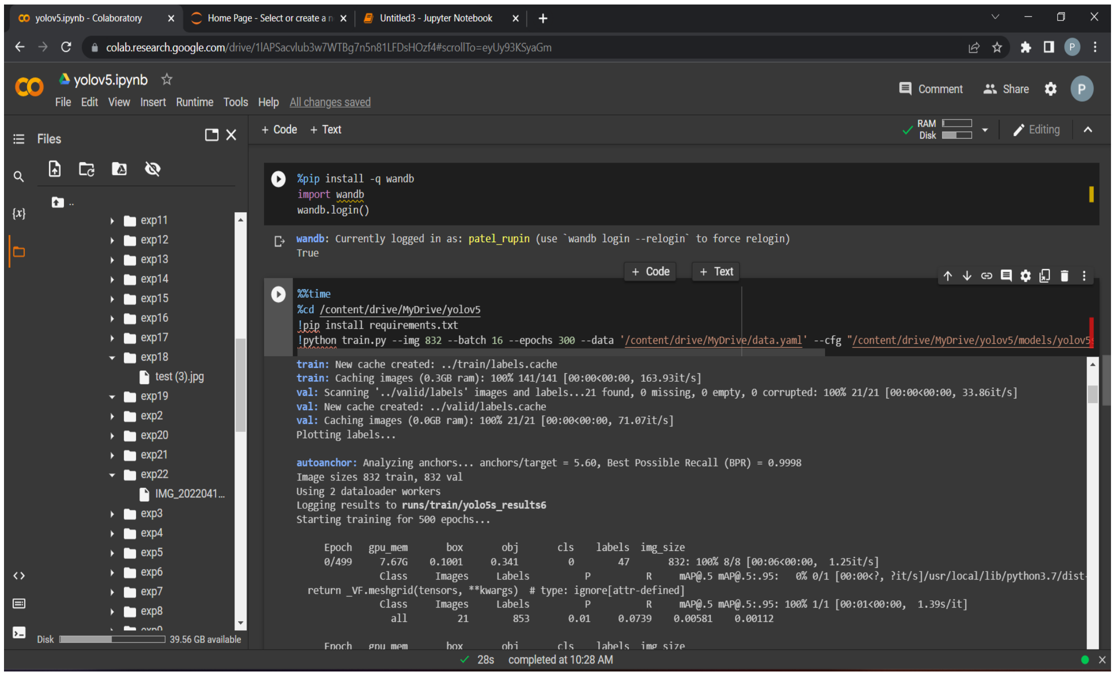


पलक

6. Output and Performance Analysis

6.1. Execution snapshots

YOLO:



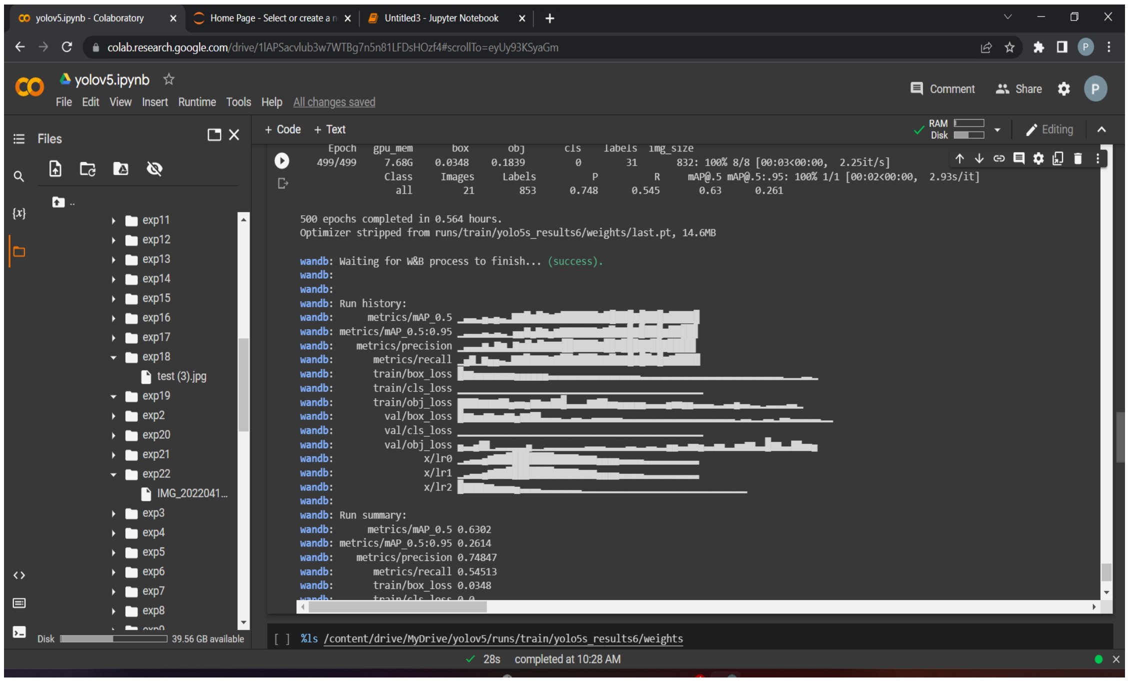
```
%pip install -q wandb
import wandb
wandb.login()
```

```
wandb: Currently logged in as: patel_rupin (use `wandb login --relogin` to force relogin)
True
```

```
%%time
%cd /content/drive/MyDrive/yolov5
!pip install requirements.txt
!python train.py --img 832 --batch 16 --epochs 300 --data '/content/drive/MyDrive/data.yaml' --cfg "/content/drive/MyDrive/yolov5/models/yolov5
```

```
train: New cache created: ../train/labels.cache
train: Caching images (0.3GB ram): 100% 141/141 [00:00<00:00, 163.93it/s]
val: Scanning '../valid/labels' images and labels...21 found, 0 missing, 0 empty, 0 corrupted: 100% 21/21 [00:00<00:00, 33.86it/s]
val: New cache created: ../valid/labels.cache
val: Caching images (0.0GB ram): 100% 21/21 [00:00<00:00, 71.07it/s]
Plotting labels...

autoanchor: Analyzing anchors... anchors/target = 5.60, Best Possible Recall (BPR) = 0.9998
Image sizes 832 train, 832 val
Using 2 dataloader workers
Logging results to runs/train/yolo5s_results6
Starting training for 500 epochs...

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     0/499     7.67G    0.1001     0.341         0        47       832: 100% 8/8 [00:06<00:00,  1.25it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95:   0% 0/1 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00,  1.39s/it]
                 all         21        853       0.01     0.0739    0.00581    0.00112

     Epoch   gpu_mem       box       obj       cls    labels  img_size
```

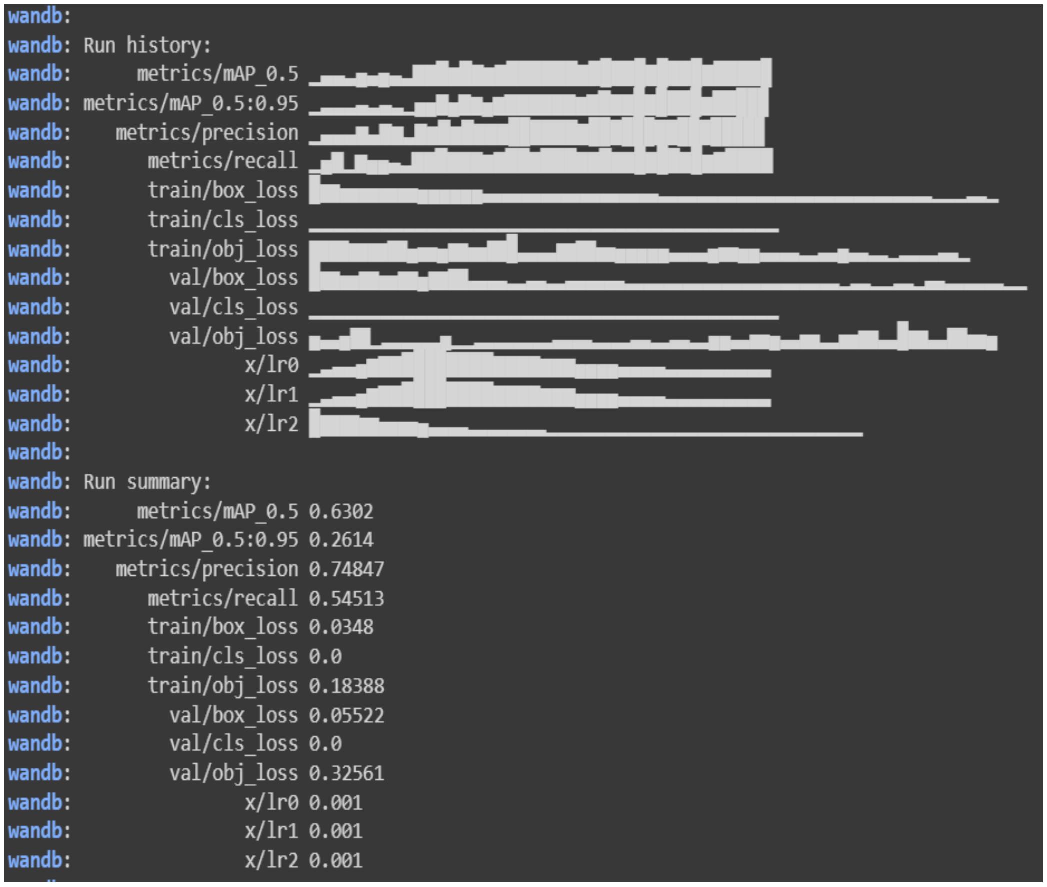
28s completed at 10:28 AM



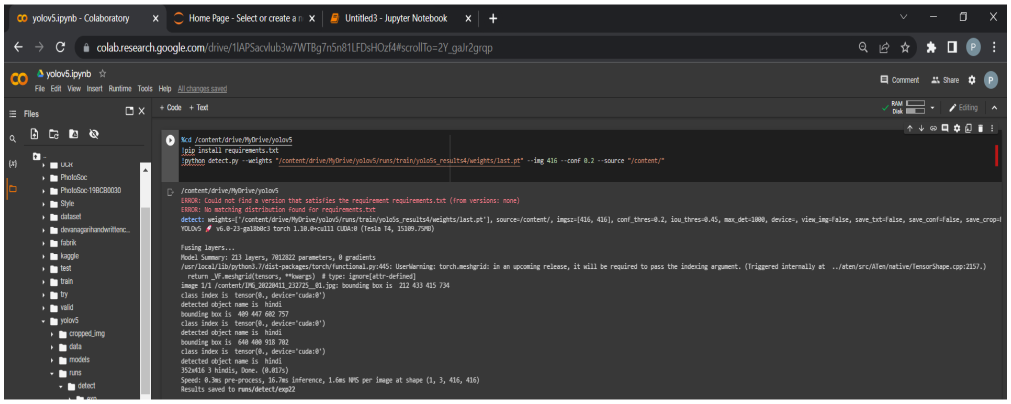
```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
   499/499     7.68G    0.0348    0.1839         0        31       832: 100% 8/8 [00:03<00:00,  2.25it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:02<00:00,  2.93s/it]
                 all         21        853      0.748      0.545       0.63      0.261

500 epochs completed in 0.564 hours.
Optimizer stripped from runs/train/yolo5s_results6/weights/last.pt, 14.6MB

wandb: Waiting for W&B process to finish... (success).
wandb:
wandb:
wandb: Run history:
wandb:      metrics/mAP_0.5
wandb: metrics/mAP_0.5:0.95
wandb:    metrics/precision
wandb:       metrics/recall
wandb:       train/box_loss
wandb:       train/cls_loss
wandb:       train/obj_loss
wandb:         val/box_loss
wandb:         val/cls_loss
wandb:         val/obj_loss
wandb:                x/lr0
wandb:                x/lr1
wandb:                x/lr2
wandb:
wandb: Run summary:
wandb:      metrics/mAP_0.5 0.6302
wandb: metrics/mAP_0.5:0.95 0.2614
wandb:    metrics/precision 0.74847
wandb:       metrics/recall 0.54513
wandb:       train/box_loss 0.0348
```

```
[ ] %ls /content/drive/MyDrive/yolov5/runs/train/yolo5s_results6/weights
```

28s completed at 10:28 AM

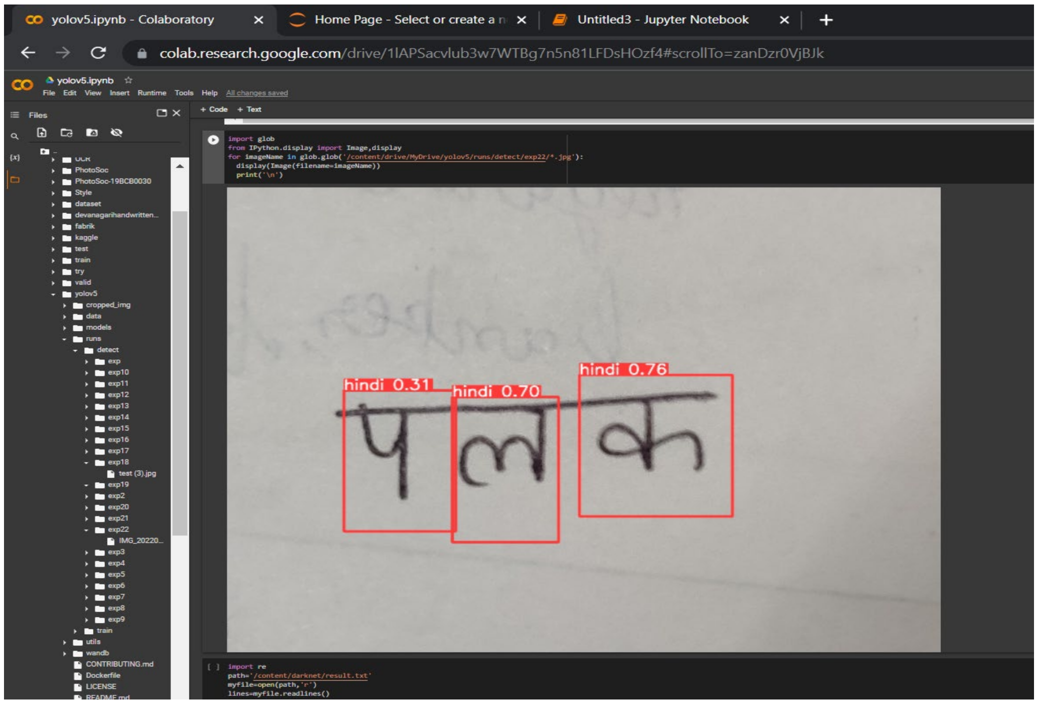


```
wandb:
wandb: Run history:
wandb:       metrics/mAP_0.5
wandb: metrics/mAP_0.5:0.95
wandb:    metrics/precision
wandb:       metrics/recall
wandb:       train/box_loss
wandb:       train/cls_loss
wandb:       train/obj_loss
wandb:         val/box_loss
wandb:         val/cls_loss
wandb:         val/obj_loss
wandb:                x/lr0
wandb:                x/lr1
wandb:                x/lr2
wandb:
wandb: Run summary:
wandb:       metrics/mAP_0.5 0.6302
wandb: metrics/mAP_0.5:0.95 0.2614
wandb:    metrics/precision 0.74847
wandb:       metrics/recall 0.54513
wandb:       train/box_loss 0.0348
wandb:       train/cls_loss 0.0
wandb:       train/obj_loss 0.18388
wandb:         val/box_loss 0.05522
wandb:         val/cls_loss 0.0
wandb:         val/obj_loss 0.32561
wandb:                x/lr0 0.001
wandb:                x/lr1 0.001
wandb:                x/lr2 0.001
```


yolov5.ipynb - Colaboratory
Home Page - Select or create a n
Untitled3 - Jupyter Notebook
colab.research.google.com/drive/1lAPSacvlub3w7WTBg7n5n81LFDsHOzf4#scrollTo=2Y_gaJr2grqp
yolov5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Editing
PhotoSoc
PhotoSoc-19BCB0030
Style
dataset
devanagarihandwrittenc...
fabrik
kaggle
test
train
try
valid
yolov5
cropped_img
data
models
runs
detect

```
%cd /content/drive/MyDrive/yolov5
!pip install requirements.txt
!python detect.py --weights "/content/drive/MyDrive/yolov5/runs/train/yolo5s_results4/weights/last.pt" --img 416 --conf 0.2 --source "/content/"
```

```
/content/drive/MyDrive/yolov5
ERROR: Could not find a version that satisfies the requirement requirements.txt (from versions: none)
ERROR: No matching distribution found for requirements.txt
detect: weights=['/content/drive/MyDrive/yolov5/runs/train/yolo5s_results4/weights/last.pt'], source=/content/, imgsz=[416, 416], conf_thres=0.2, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=F
YOLOv5 🚀 v6.0-23-ga18b0c3 torch 1.10.0+cu111 CUDA:0 (Tesla T4, 15109.75MB)

Fusing layers...
Model Summary: 213 layers, 7012822 parameters, 0 gradients
/usr/local/lib/python3.7/dist-packages/torch/functional.py:445: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the indexing argument. (Triggered internally at  ../aten/src/ATen/native/TensorShape.cpp:2157.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
image 1/1 /content/IMG_20220411_232725__01.jpg: bounding box is  212 433 415 734
class index is  tensor(0., device='cuda:0')
detected object name is  hindi
bounding box is  409 447 602 757
class index is  tensor(0., device='cuda:0')
detected object name is  hindi
bounding box is  640 400 918 702
class index is  tensor(0., device='cuda:0')
detected object name is  hindi
352x416 3 hindis, Done. (0.017s)
Speed: 0.3ms pre-process, 16.7ms inference, 1.6ms NMS per image at shape (1, 3, 416, 416)
Results saved to runs/detect/exp22
```

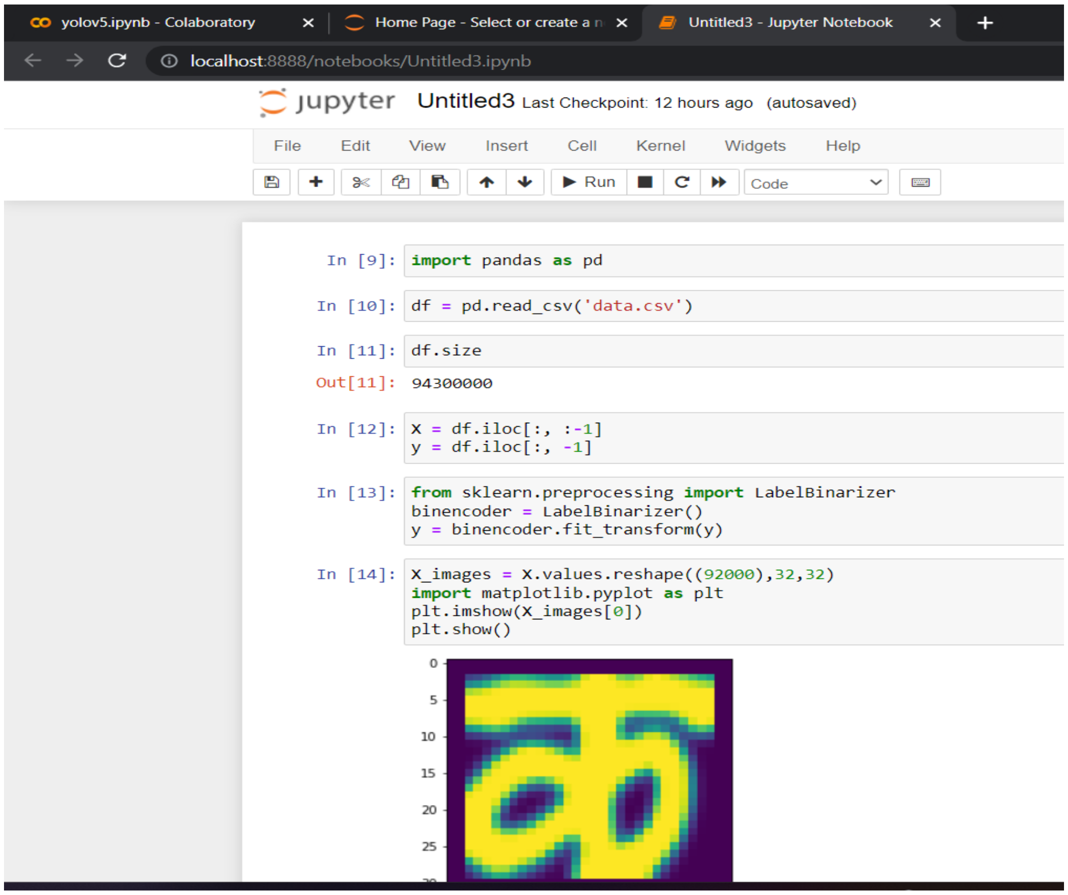


yolov5.ipynb - Colaboratory
Home Page - Select or create a n
Untitled3 - Jupyter Notebook
colab.research.google.com/drive/1lAPSacvlub3w7WTBg7n5n81LFDsHOzf4#scrollTo=zanDzr0VjBJk
hindi 0.31
hindi 0.70
hindi 0.76

```
import glob
from IPython.display import Image,display
for imageName in glob.glob('/content/drive/MyDrive/yolov5/runs/detect/exp22/*.jpg'):
  display(Image(filename=imageName))
  print('\n')
```

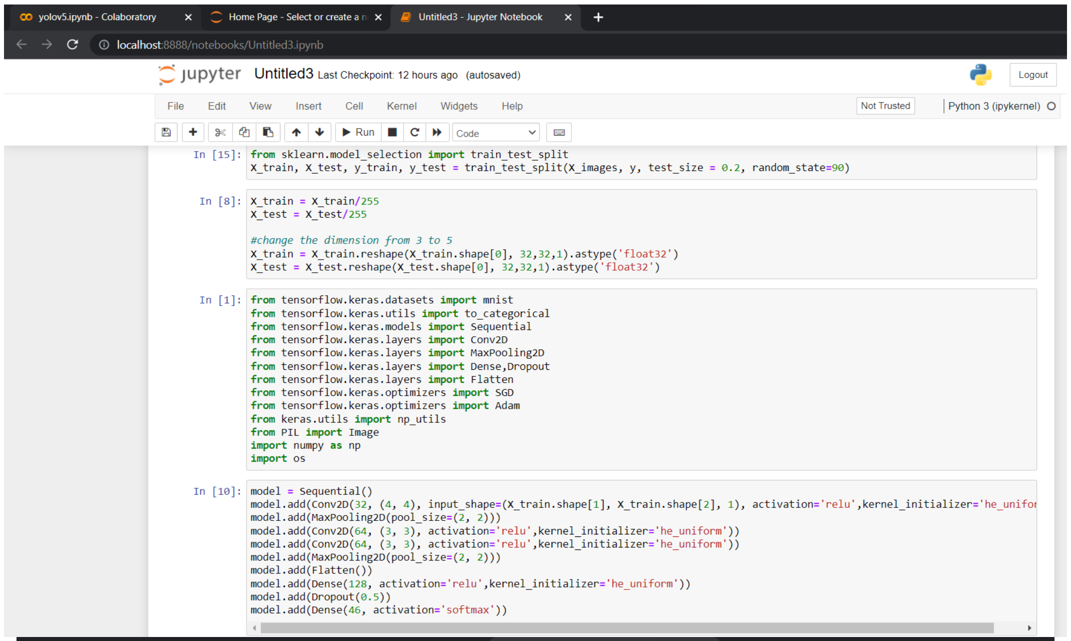
```
import re
path='/content/darknet/result.txt'
myfile=open(path,'r')
lines=myfile.readlines()
```

Image Recognition:

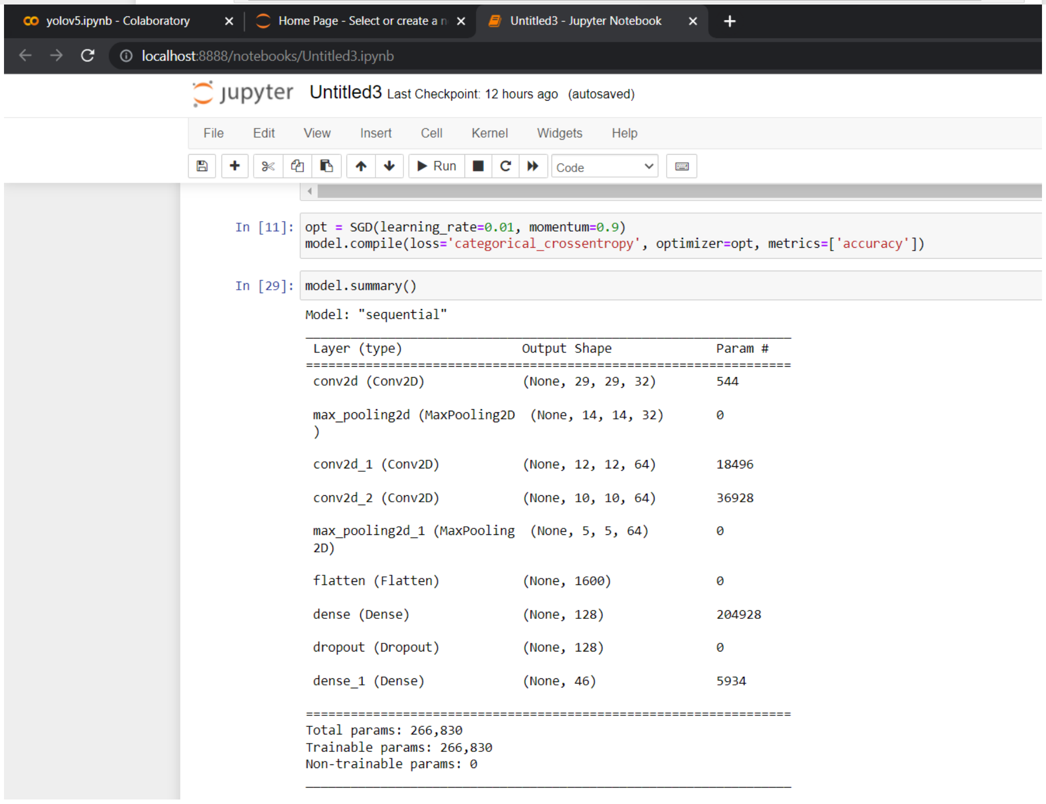


yolov5.ipynb - Colaboratory
Home Page - Select or create a n
Untitled3 - Jupyter Notebook
localhost:8888/notebooks/Untitled3.ipynb
jupyter Untitled3 Last Checkpoint: 12 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

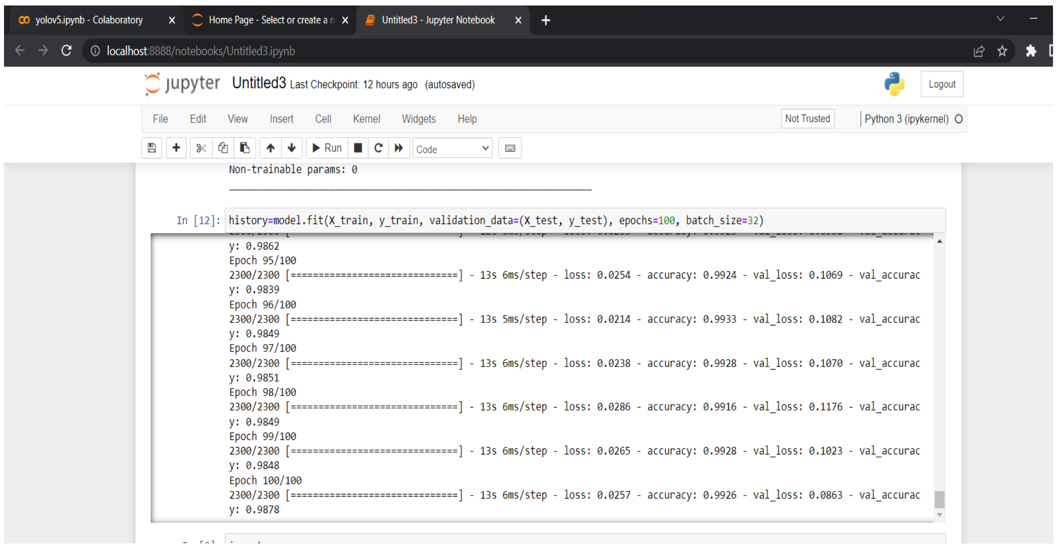
```
In [9]: import pandas as pd

In [10]: df = pd.read_csv('data.csv')

In [11]: df.size
Out[11]: 94300000

In [12]: X = df.iloc[:, :-1]
         y = df.iloc[:, -1]

In [13]: from sklearn.preprocessing import LabelBinarizer
         binencoder = LabelBinarizer()
         y = binencoder.fit_transform(y)

In [14]: X_images = X.values.reshape((92000),32,32)
         import matplotlib.pyplot as plt
         plt.imshow(X_images[0])
         plt.show()
```


yolov5.ipynb - Colaboratory
Home Page - Select or create a n
Untitled3 - Jupyter Notebook
localhost:8888/notebooks/Untitled3.ipynb
jupyter Untitled3 Last Checkpoint: 12 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [15]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X_images, y, test_size = 0.2, random_state=90)

In [8]:  X_train = X_train/255
         X_test = X_test/255

         #change the dimension from 3 to 5
         X_train = X_train.reshape(X_train.shape[0], 32,32,1).astype('float32')
         X_test = X_test.reshape(X_test.shape[0], 32,32,1).astype('float32')

In [1]:  from tensorflow.keras.datasets import mnist
         from tensorflow.keras.utils import to_categorical
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Conv2D
         from tensorflow.keras.layers import MaxPooling2D
         from tensorflow.keras.layers import Dense,Dropout
         from tensorflow.keras.layers import Flatten
         from tensorflow.keras.optimizers import SGD
         from tensorflow.keras.optimizers import Adam
         from keras.utils import np_utils
         from PIL import Image
         import numpy as np
         import os

In [10]: model = Sequential()
         model.add(Conv2D(32, (4, 4), input_shape=(X_train.shape[1], X_train.shape[2], 1), activation='relu',kernel_initializer='he_unifor
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_uniform'))
         model.add(Conv2D(64, (3, 3), activation='relu',kernel_initializer='he_uniform'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Flatten())
         model.add(Dense(128, activation='relu',kernel_initializer='he_uniform'))
         model.add(Dropout(0.5))
         model.add(Dense(46, activation='softmax'))
```


yolov5.ipynb - Colaboratory
Home Page - Select or create a n
Untitled3 - Jupyter Notebook
localhost:8888/notebooks/Untitled3.ipynb
jupyter Untitled3 Last Checkpoint: 12 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

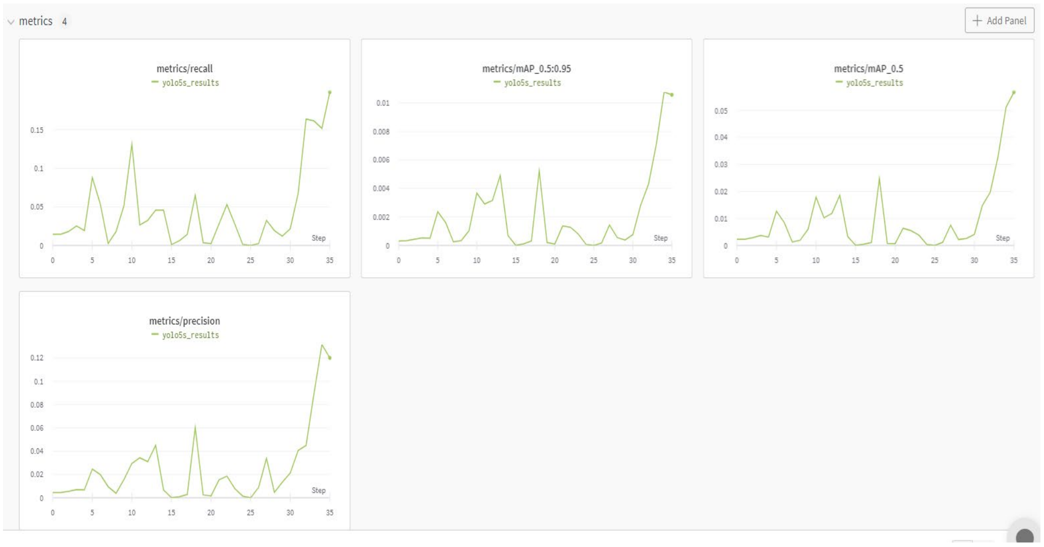
```
In [11]: opt = SGD(learning_rate=0.01, momentum=0.9)
         model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

In [29]: model.summary()

         Model: "sequential"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          conv2d (Conv2D)             (None, 29, 29, 32)        544

          max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
          )

          conv2d_1 (Conv2D)           (None, 12, 12, 64)        18496

          conv2d_2 (Conv2D)           (None, 10, 10, 64)        36928

          max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
          2D)

          flatten (Flatten)           (None, 1600)              0

          dense (Dense)               (None, 128)               204928

          dropout (Dropout)           (None, 128)               0

          dense_1 (Dense)             (None, 46)                5934

         =================================================================
         Total params: 266,830
         Trainable params: 266,830
         Non-trainable params: 0
         _________________________________________________________________
```


yolov5.ipynb - Colaboratory
Home Page - Select or create a n
Untitled3 - Jupyter Notebook
localhost:8888/notebooks/Untitled3.ipynb
jupyter
Untitled3 Last Checkpoint: 12 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
Non-trainable params: 0
_________________________________________________________________

In [12]: history=model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, batch_size=32)

y: 0.9862
Epoch 95/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0254 - accuracy: 0.9924 - val_loss: 0.1069 - val_accurac
y: 0.9839
Epoch 96/100
2300/2300 [==============================] - 13s 5ms/step - loss: 0.0214 - accuracy: 0.9933 - val_loss: 0.1082 - val_accurac
y: 0.9849
Epoch 97/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0238 - accuracy: 0.9928 - val_loss: 0.1070 - val_accurac
y: 0.9851
Epoch 98/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0286 - accuracy: 0.9916 - val_loss: 0.1176 - val_accurac
y: 0.9849
Epoch 99/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0265 - accuracy: 0.9928 - val_loss: 0.1023 - val_accurac
y: 0.9848
Epoch 100/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0257 - accuracy: 0.9926 - val_loss: 0.0863 - val_accurac
y: 0.9878
```

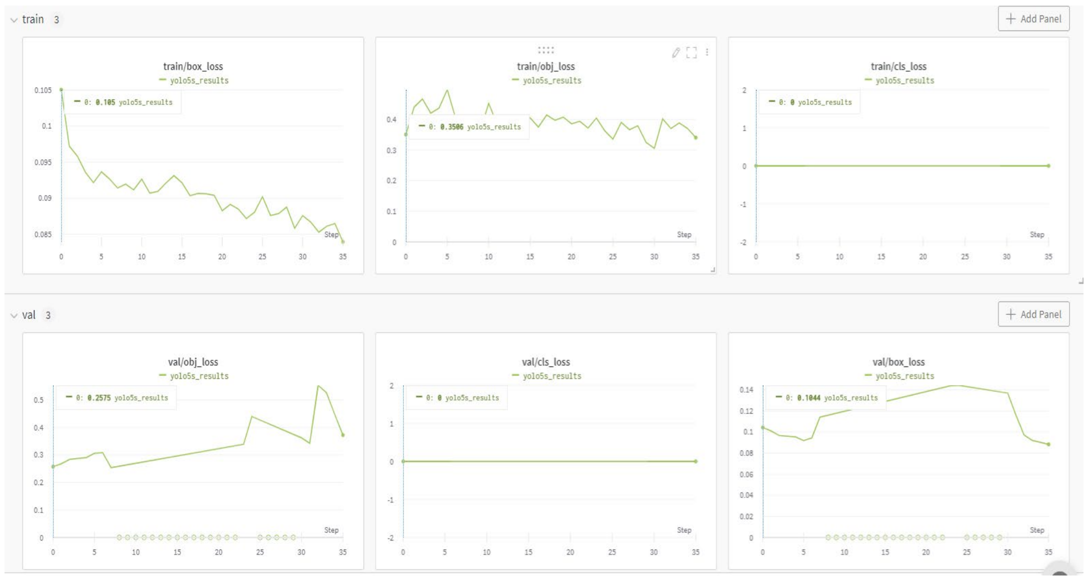
6.2. Output – in terms of performance metrics

YOLOv5 Model:



Metrics -Recall, Precision, mAP

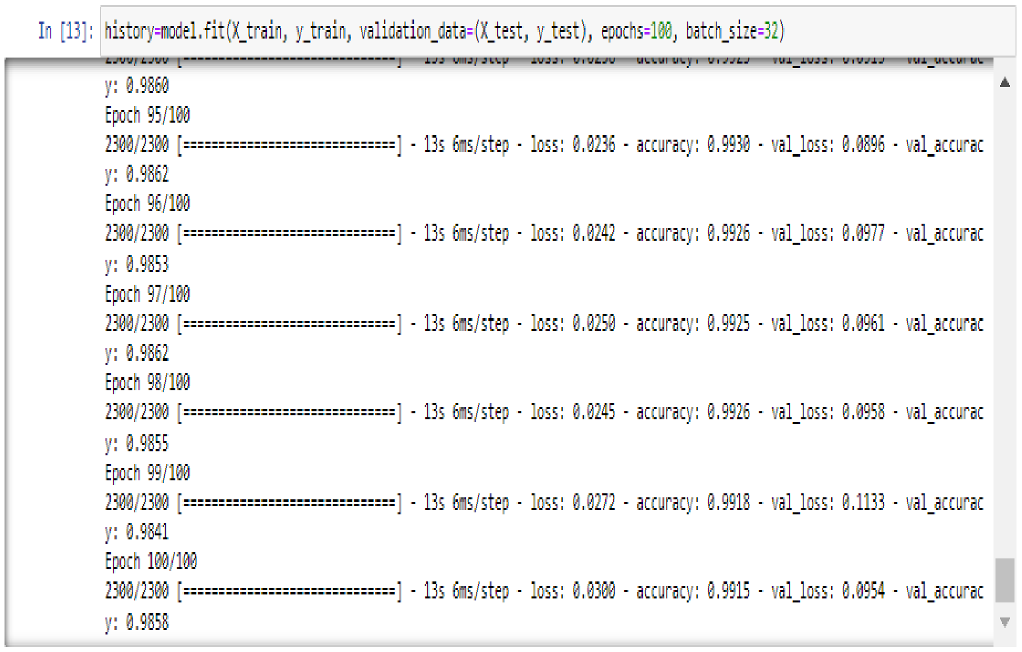
Achieved Precision of 0.7487 and Recall of 0.545



Training loss and validation loss

After achieving relevant precision on yolo model for text detection we move forward with text recognition part.

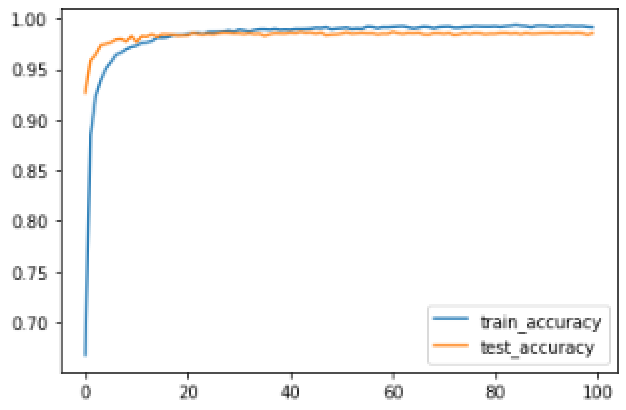
CNN model for Image Recognition using tensorflow:



```
In [13]: history=model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, batch_size=32)
```

```
y: 0.9860
Epoch 95/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0236 - accuracy: 0.9930 - val_loss: 0.0896 - val_accurac
y: 0.9862
Epoch 96/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0242 - accuracy: 0.9926 - val_loss: 0.0977 - val_accurac
y: 0.9853
Epoch 97/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0250 - accuracy: 0.9925 - val_loss: 0.0961 - val_accurac
y: 0.9862
Epoch 98/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0245 - accuracy: 0.9926 - val_loss: 0.0958 - val_accurac
y: 0.9855
Epoch 99/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0272 - accuracy: 0.9918 - val_loss: 0.1133 - val_accurac
y: 0.9841
Epoch 100/100
2300/2300 [==============================] - 13s 6ms/step - loss: 0.0300 - accuracy: 0.9915 - val_loss: 0.0954 - val_accurac
y: 0.9858
```

```
In [15]: import numpy as np
         print(history)
         import matplotlib.pyplot as plt
         plt.plot(history.history['accuracy'],label="train_accuracy")
         plt.plot(history.history['val_accuracy'],label="test_accuracy"
         plt.legend()

         <keras.callbacks.History object at 0x0000011A6341BF70>

Out[15]: <matplotlib.legend.Legend at 0x11aebee59a0>
```



Hence we can see here that model have achieved relevant accuracy and not only training accuracy but also validation accuracy .According to graph we can see that there are very less fluctuations in values and both training and validation accuracy are near two each other. So we can say that model have achieved better accuracy for text recognition.

6.3. Performance comparison with existing works

EasyOCR is actually a python package that holds PyTorch as a backend handler. EasyOCR like any other OCR(tesseract of Google or any other) detects the text from images but in my reference, while using it I found that it is the most straightforward way to detect text from images also when high end deep learning library(PyTorch) is supporting it in the backend which makes it accuracy more credible. EasyOCR supports 42+ languages for detection purposes. EasyOCR is created by the company named Jaided AI company. But there are some drawbacks also:

EasyOCR can only detect computerized text images Whereas our model is trained on handwritten text which can also detect handwritten text images.

EasyOCR detect text only from images whereas our model detect text from video therefore we can use it for real time applications also just like Google Lens.

According to accuracy achieved our model is giving relevant output then simple EasyOCR library.

7. Conclusion and Future Directions

In conclusion, we applied the knowledge of Convolution Neural Networks to identify the input image or document. We have achieved satisfactory accuracy for handwritten text recognition for both Hindi and English. Secondly, we will deploy our trained model as an online web app using Flask . Where the user uploads the photo or can take a snapshot and gets a digital converted text result based on the prediction without change in any format of document. We will print the same resulting text on the document from pixel coordinate from where it was detected with help of OpenCV and create another copy. Thus our final objective would be to create a Real time web app ,where users can use an online camera to scan documents, get their text recognised and digitize it.

8. References

[1] A. Srivastav και N. Sahu, "Segmentation of Devanagari Handwritten Characters", International Journal of Computer Applications, Vol 142, No. 14, pg.15–18, May, 2016.

[2] P. P. Bairagi, "Optical Character Recognition for Hindi",International Research Journal of Engineering and Technology (IRJET), May, 2018.

[3] S. Albahli, M. Nawaz, A. Javed, και A. Irtaza, 'An improved faster-RCNN model for handwritten character recognition', Arabian Journal for Science and Engineering, τ. 46, 03 2021.

[4] Russell, Stuart J. (Stuart Jonathan). Artificial Intelligence : a Modern Approach. Upper Saddle River, N.J. :Prentice Hall, 2010.

[5] A. Chaudhuri, K. Mandaviya, P. Badelia, και S. K. Ghosh, 'Optical character recognition systems', στο Optical Character Recognition Systems for Different Languages with Soft Computing, Springer, 2017, pg. 9–41.

[6] T. C. Wei, U. U. Sheikh, και A. A.-H. Ab Rahman, 'Improved optical character recognition with deep neural network', στο 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), 2018, pg. 245–249.

[7]Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR); IEEE(INSPEC Accession Number: 19973748)

[8]Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR); IEEE(INSPEC Accession Number: 19973748)

[9] A Complete OCR for Printed Hindi Text in Devanagari Script Year: 2001, Pages: 0800 DOI Bookmark: 10.1109/ICDAR.2001.953898

[10]Optical Character Recognition Techniques: A Review DOI: 10.1109/SCEECS54111.2022.9740911