DeepFake Detection and Mitigation System

USING DEEP AND MACHINE LEARNING APPROACH

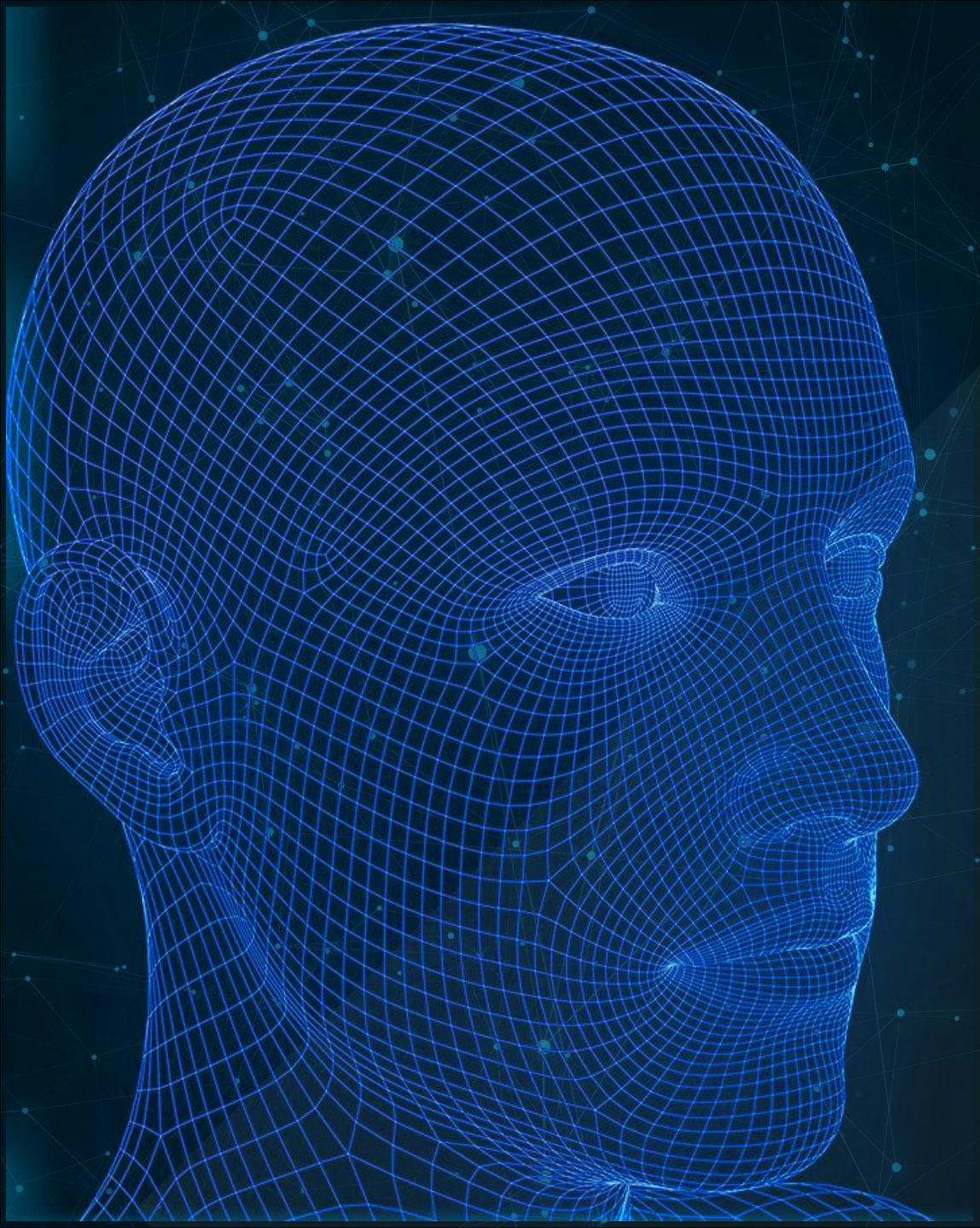
GRP: 38

Group Member's :

Praful Shrivastava
Piyush Pandey
Sumit Chourasiya
Abhay Kumar

Guided By:

Prof. Aditya Tiwari
Prof. Sanat Kumar Rana



Content's

- Introduction
- Problem's & Fact's
- Objective
- Dataset
- Work in Progress
  - Project Development Life Cycle
  - Code for I. V. T
- Conclusion

What’s Deepfake?

A deepfake is a fake video or audio created using advanced computer technology. It can make people appear to say or do things they never did. This technology raises concerns about misinformation and fraud.

Study of identify and counteract fake videos or audio created using advanced techniques. It helps spot and stop deceptive content, addressing concerns related to misinformation and fraud. And that technology known as deepfake detection & mitigation system.

INDIA TODAY DAILYO NORTHEAST MALAYALAM BUSINESS TODAY WEB3CAFE AAJ TAK LALLANTOP BANGLA GNT

INDIA TODAY

SIGN IN

Dark Mode

Premium

- Home
- Personalise
- Best Colleges 2023
- Live TV
- India
- World
- Business
- Technology
- Showbuzz
- Sports

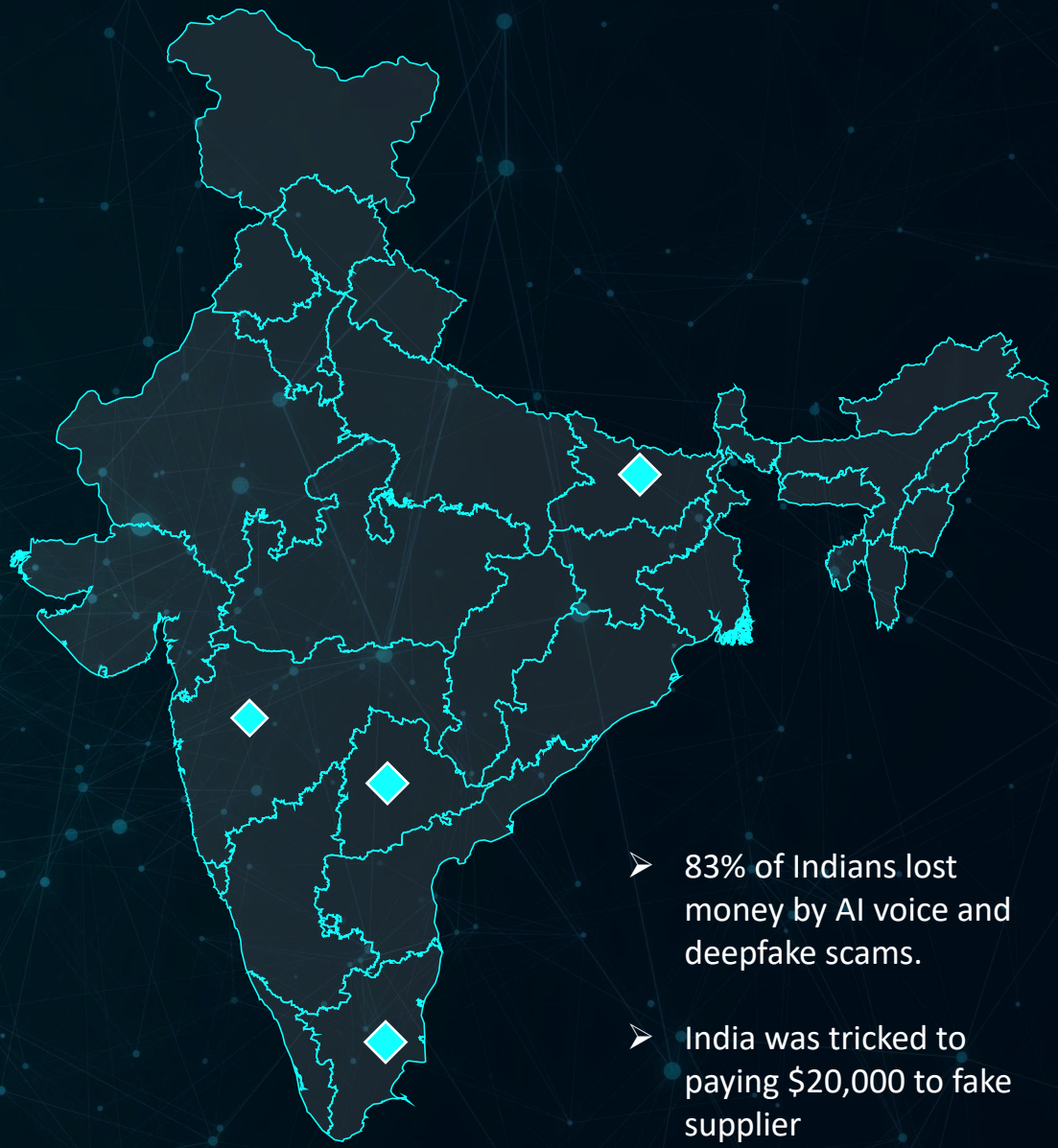
News / Technology / News / Kerala man loses Rs 40,000 in AI-based Deepfake WhatsApp fraud, all about the new scam

Kerala man loses Rs 40,000 to AI-based Deepfake WhatsApp fraud, all about the new scam

In a recent case of an online scam on WhatsApp, scammers used AI-based deepfake technology to dupe Rs 40,000 from a man from Kerala.

Listen to Story

Share



- 83% of Indians lost money by AI voice and deepfake scams.
- India was tricked to paying $20,000 to fake supplier

Objective's

1 Combat Deepfake Threats	Address the growing challenges posed by deepfake content, promoting media authenticity and trustworthiness.
2 Detection in 3 Board Ways	Achieve an high level detection accuracy on Image, video, Real-Time simulation
3 User-Friendly Interface	Design an intuitive and accessible user interface to ensure users of all technical levels can easily upload and analyze media for deepfake content.
4 Mitigate Misinformation	Contribute to the reduction of misinformation and disinformation by helping users identify manipulated media and promoting media authenticity.
5 Security & Credibility	Credibility as a reliable source for deepfake detection, becoming a go-to resource for media verification and Security measure.

Dataset's

For Collection of Dataset , We Use Corresponding Technique for their Utility

Image Based

- We Taken the Dataset from Kaggle over 2k+ images
- We can use the 1000 real and 1000 Fake images

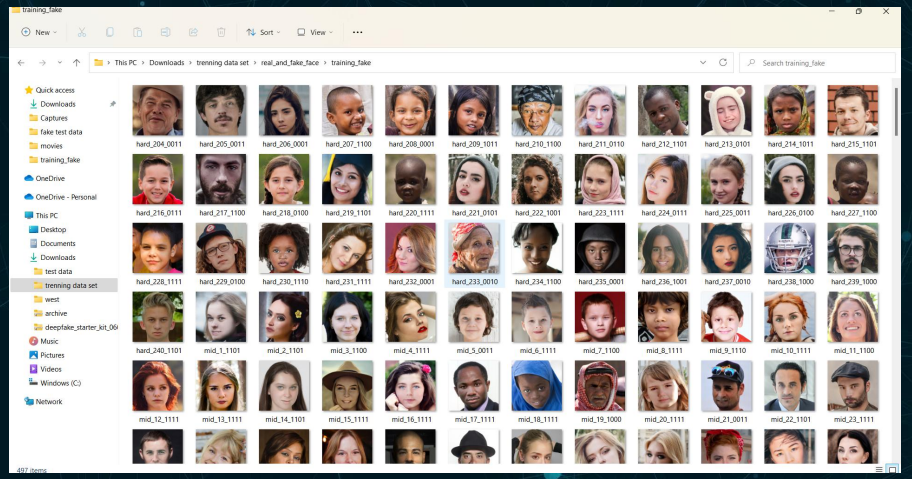
Real Time

- Use a Data collection script with Face Detection
- Maintain Equal Even ratio in Real and Fake images
- Yolo or MobileNetSSD Model (which are more reliable & considerable).

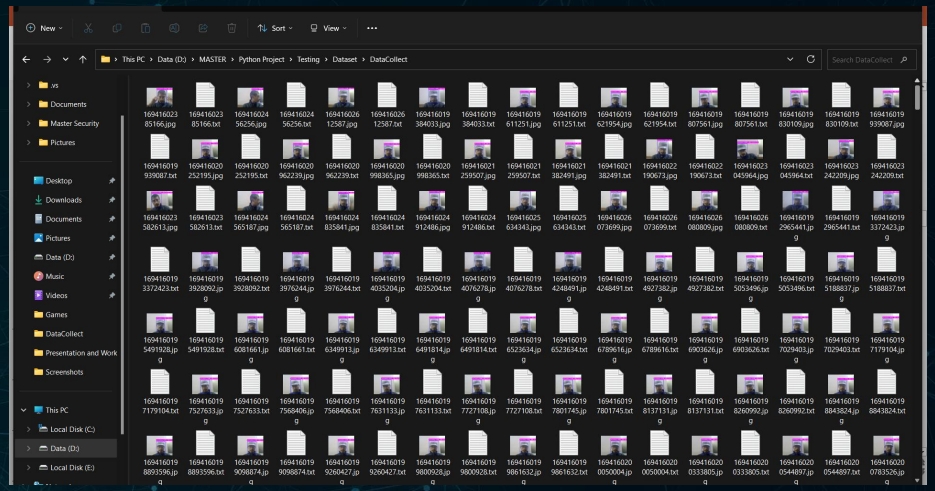
Video Based

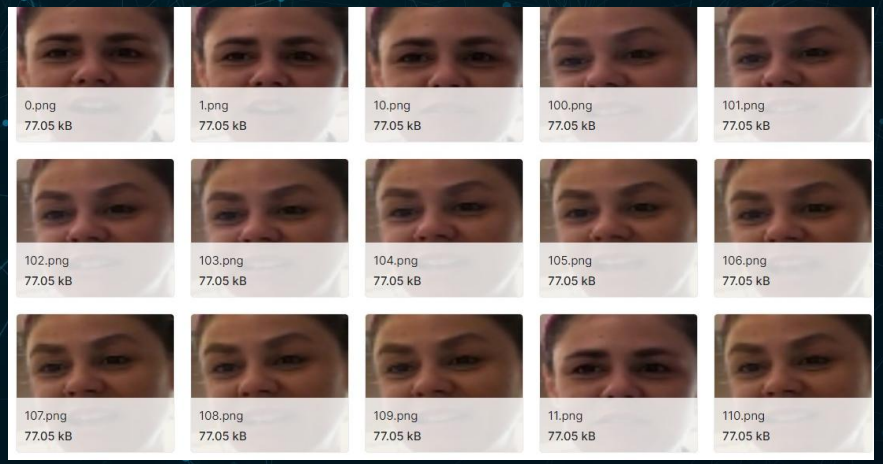
- For Video Based D.F.D we retrieve the dataset of image from FaceForensic++, Celebfake etc .
- We try to maintain the disciplined ratio for Train, Val , Test also.

For image



For Real Time



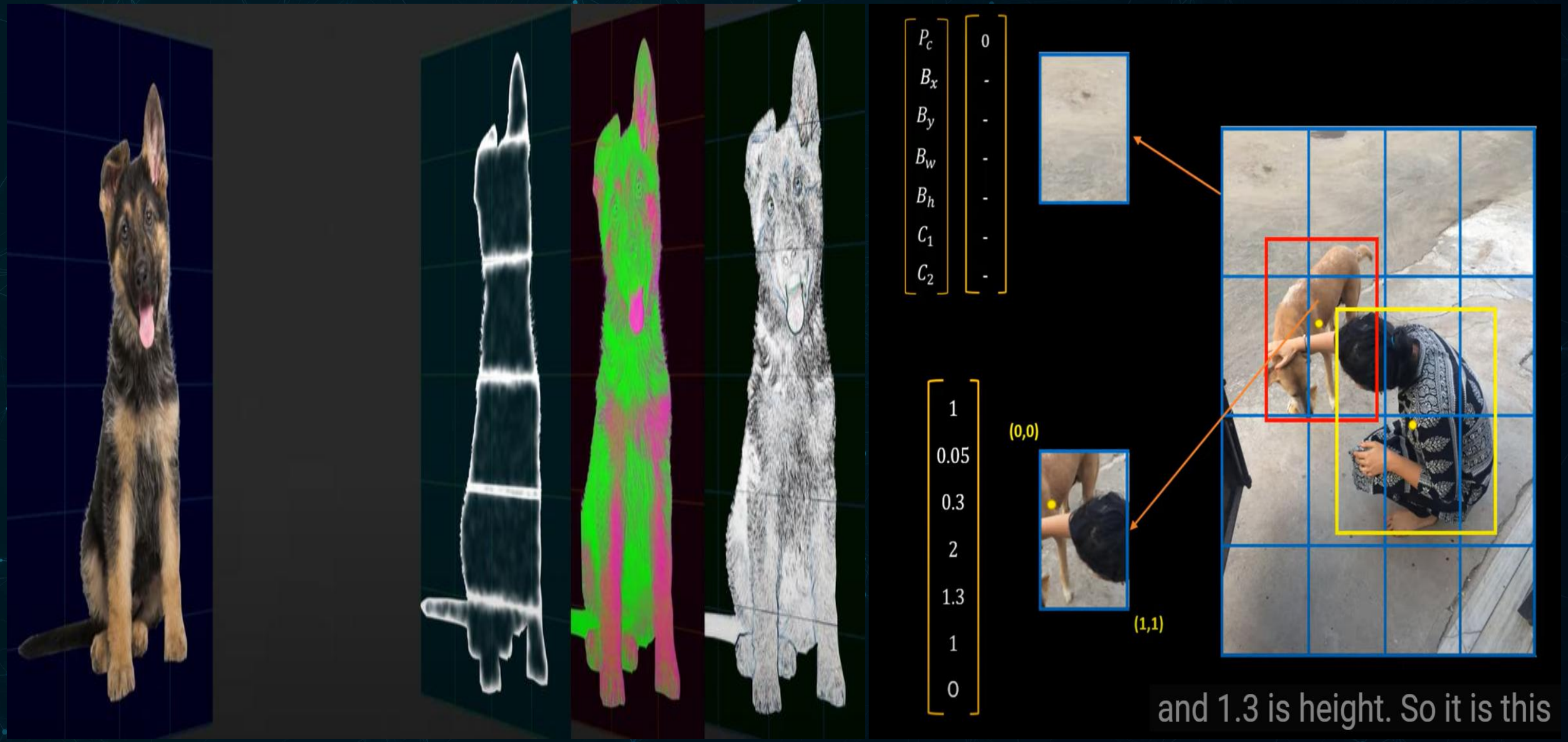


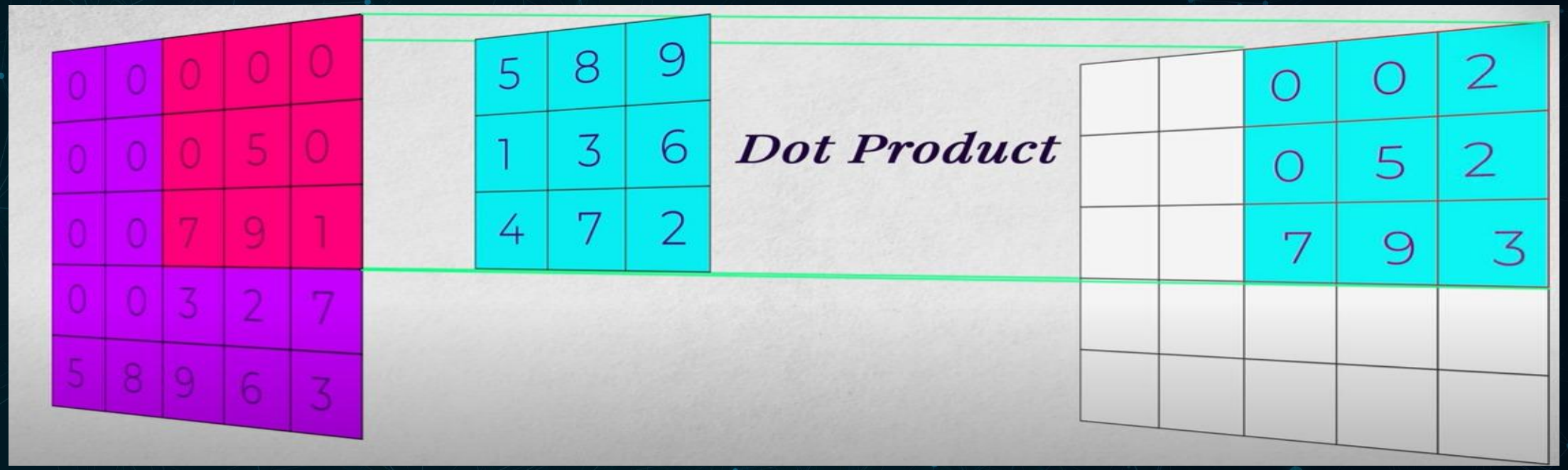
For video

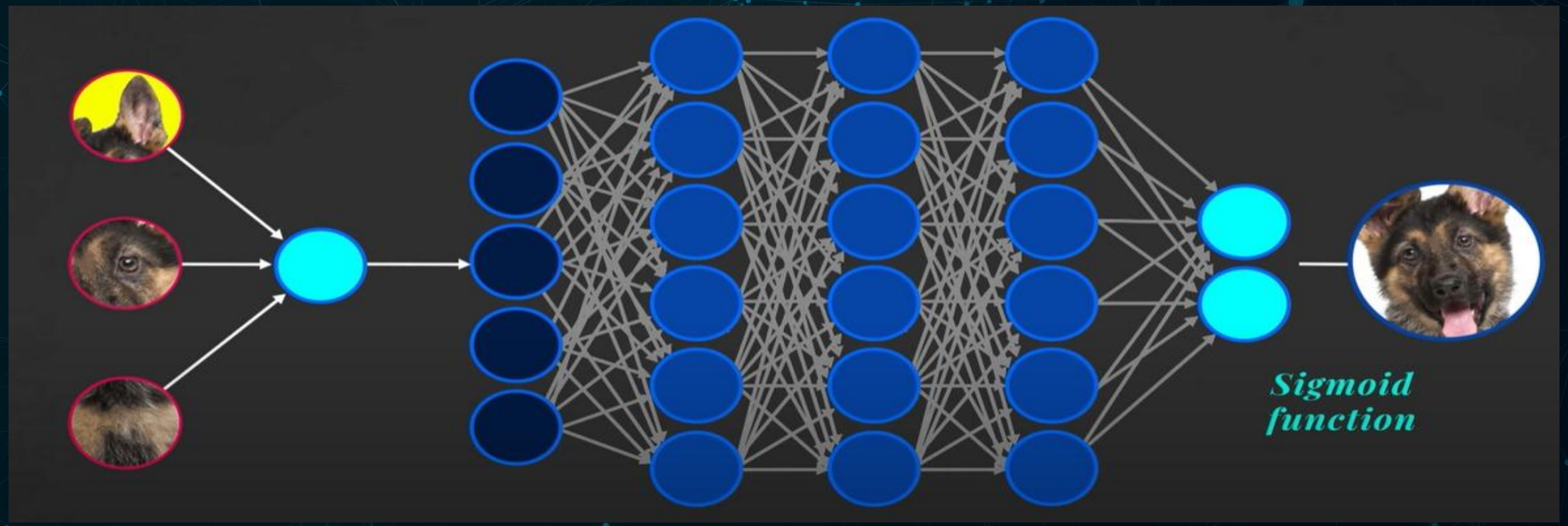
MODELS:

CNN

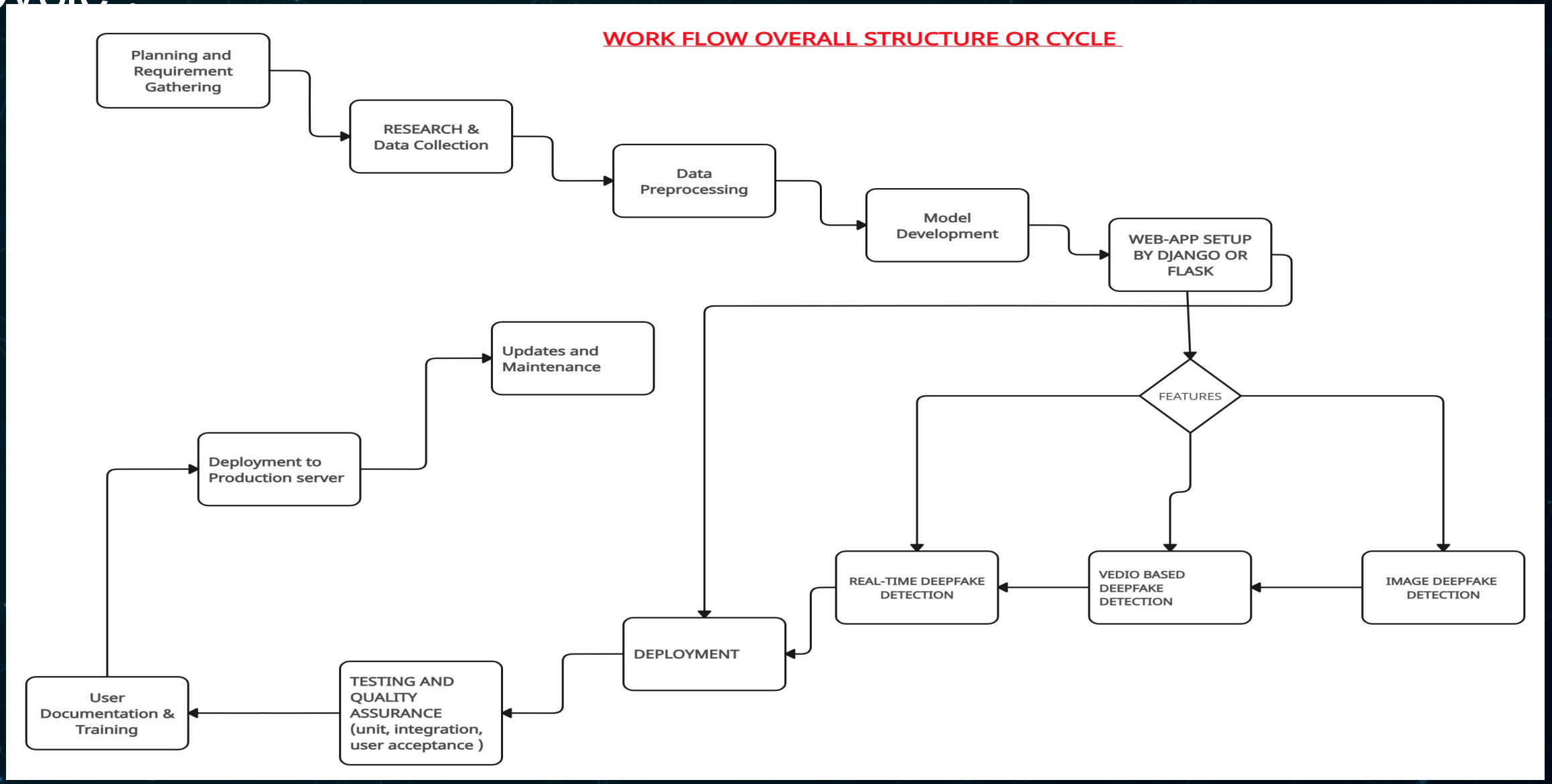
YOLO




0 0 0 0 0
0 0 0 5 0
0 0 7 9 1
0 0 3 2 7
5 8 9 6 3
5 8 9
1 3 6
4 7 2
Dot Product
0 0 2
0 5 2
7 9 3


Sigmoid function

➢WP - Project Development Cycle :



For Image Based

```
test_labels = []

with torch.no_grad():
    for images, labels in test_loader:
        outputs = model(images)
        predictions = (outputs >= 0.5).squeeze().cpu().numpy()
        test_predictions.extend(predictions)
        test_labels.extend(labels.cpu().numpy())

# Calculate accuracy and confusion matrix
accuracy = accuracy_score(test_labels, test_predictions)
conf_matrix = confusion_matrix(test_labels, test_predictions)

# Print and save the results
print(f"Accuracy: {accuracy * 100:.2f}%")
print("Confusion Matrix:")
print(conf_matrix)

# Save model
torch.save(model.state_dict(), 'deepfake_detection_model.pt')
```

```
Accuracy: 70.00%
Confusion Matrix:
[[13  7]
 [ 5 15]]
```

In [*]:

```
import torch
import torch.nn as nn
from torchvision import transforms
from PIL import Image


# Load the trained model
model = Net()
model.load_state_dict(torch.load('deepfake_detection_model.pt'))
model.eval()
```

Home Page - Select or create a n | image detection project in my ha

localhost:8888/notebooks/image%20detection%20project%20in%20my%20hand%20.ipynb

jupyter image detection project in my hand Last Checkpoint: 11 hours ago (autosaved)

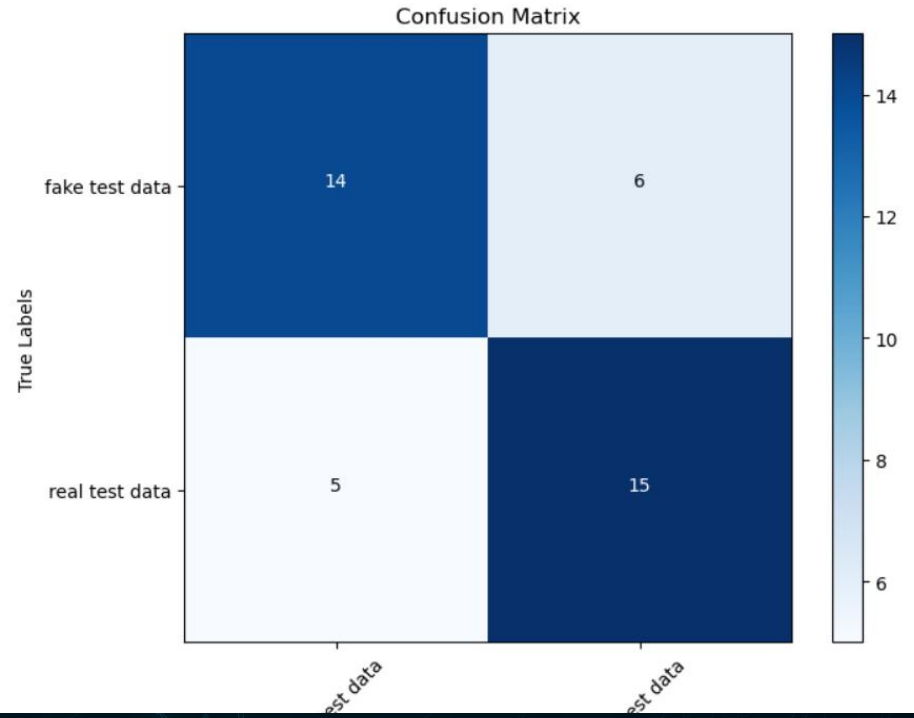
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Markdown

```
Accuracy: 72.50%
Confusion Matrix:
[[14  6]
 [ 5 15]]
```

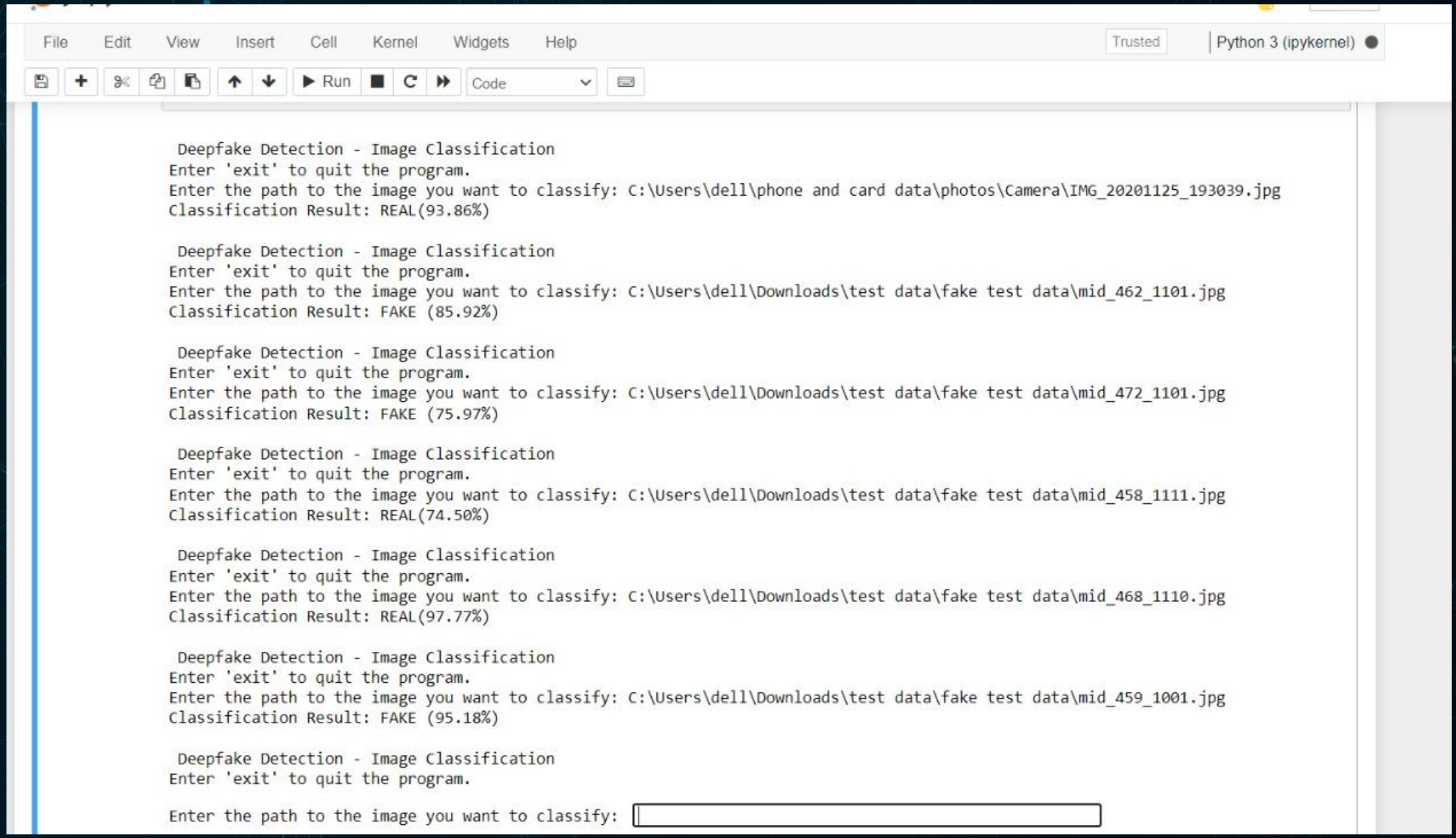


Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

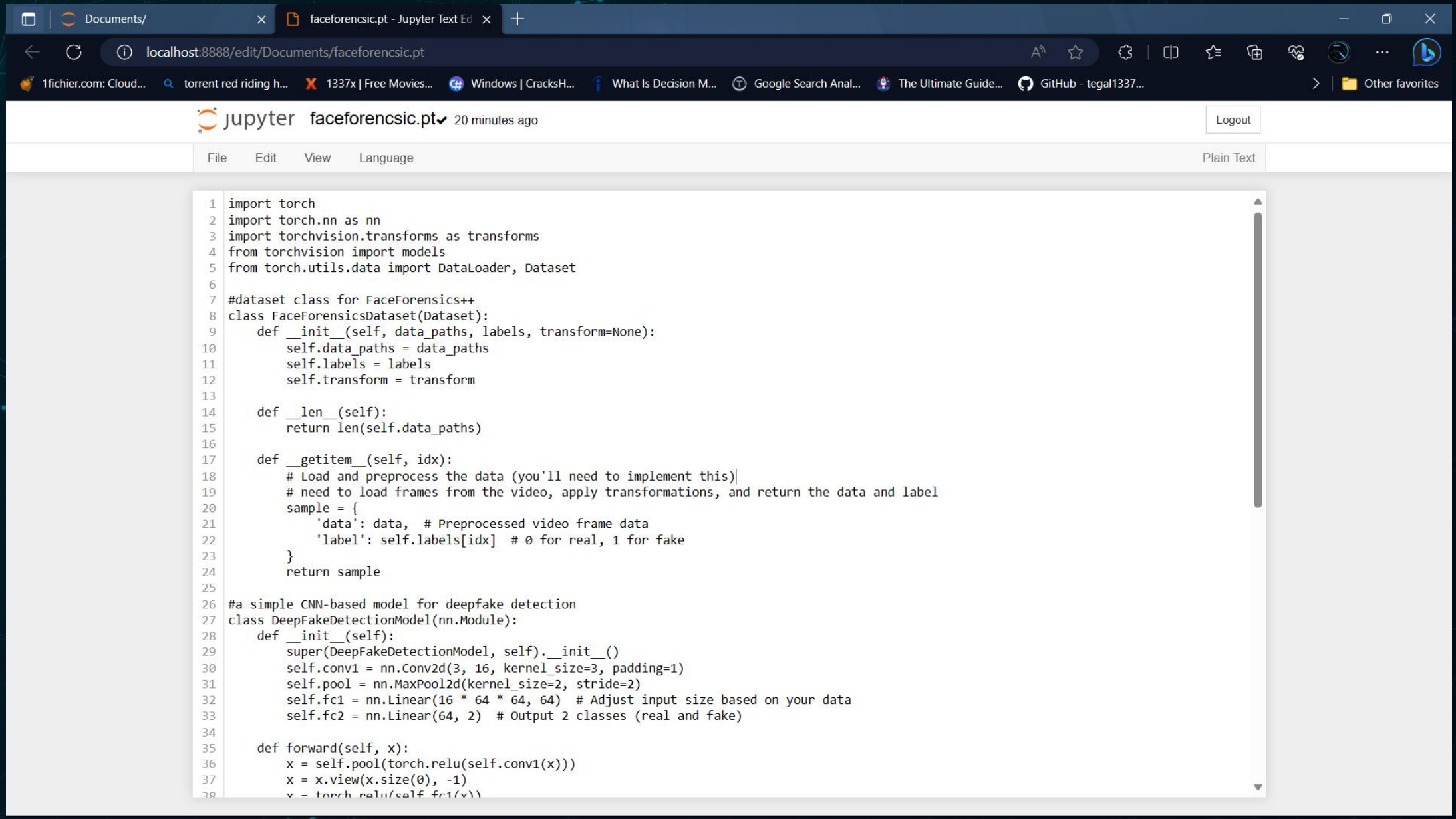
Run Code

```
    return result

if __name__ == '__main__':
    while True:
        print("\n Deepfake Detection - Image Classification")
        print("Enter 'exit' to quit the program.")

        image_path = input("Enter the path to the image you want to classify: ")

        if image_path.lower() == 'exit':
            break

        try:
            result = classify_image(image_path)
            print(f"Classification Result: {result}")
        except Exception as e:
            print(f"An error occurred: {str(e)}")
```

```
 Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\phone and card data\photos\Camera\IMG_20201125_193039.jpg
Classification Result: REAL(93.86%)

 Deepfake Detection - Image Classification
Enter 'exit' to quit the program.

Enter the path to the image you want to classify:
```


File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\phone and card data\photos\Camera\IMG_20201125_193039.jpg
Classification Result: REAL(93.86%)
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\Downloads\test data\fake test data\mid_462_1101.jpg
Classification Result: FAKE (85.92%)
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\Downloads\test data\fake test data\mid_472_1101.jpg
Classification Result: FAKE (75.97%)
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\Downloads\test data\fake test data\mid_458_1111.jpg
Classification Result: REAL(74.50%)
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\Downloads\test data\fake test data\mid_468_1110.jpg
Classification Result: REAL(97.77%)
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify: C:\Users\dell\Downloads\test data\fake test data\mid_459_1001.jpg
Classification Result: FAKE (95.18%)
Deepfake Detection - Image Classification
Enter 'exit' to quit the program.
Enter the path to the image you want to classify:

For Video Based



```
    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = x.view(x.size(0), -1)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Data preprocessing and augmentation
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Create datasets and dataloaders
train_dataset = FaceForensicsDataset(train_data_paths, train_labels, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

# Initialize the model
model = DeepFakeDetectionModel()

# Define a loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Training loop
num_epochs = 10
for epoch in range(num_epochs):
    for batch in train_loader:
        data, labels = batch['data'], batch['label']
        optimizer.zero_grad()
        outputs = model(data)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

torch.save(model.state_dict(), 'deepfake_detection_model.pth')
```

```
    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = x.view(x.size(0), -1)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Data preprocessing and augmentation
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Create datasets and dataloaders
train_dataset = FaceForensicsDataset(train_data_paths, train_labels, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

# Initialize the model
model = DeepFakeDetectionModel()

# Define a loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Training loop
num_epochs = 10
for epoch in range(num_epochs):
    for batch in train_loader:
        data, labels = batch['data'], batch['label']
        optimizer.zero_grad()
        outputs = model(data)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

torch.save(model.state_dict(), 'deepfake_detection_model.pth')
```

File Edit View Language

```
import torch
import torch.nn as nn

class DeepFakeDetectionCNN(nn.Module):
    def __init__(self, num_classes=2):
        super(DeepFakeDetectionCNN, self).__init__()

        # Convolutional layers
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv3 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.relu3 = nn.ReLU()
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)

        # Fully connected layers
        self.fc1 = nn.Linear(in_features=64 * 16 * 16, out_features=128)
        self.relu4 = nn.ReLU()
        self.dropout = nn.Dropout(0.5)

        self.fc2 = nn.Linear(in_features=128, out_features=num_classes)

    def forward(self, x):
        # Convolutional layers
        x = self.pool1(self.relu1(self.conv1(x)))
        x = self.pool2(self.relu2(self.conv2(x)))
        x = self.pool3(self.relu3(self.conv3(x)))

        # Flatten the output for the fully connected layers
        x = x.view(x.size(0), -1)

        # Fully connected layers
        x = self.relu4(self.fc1(x))
```

For Real Ti

DataCollection - 3.py > ...

```python
from cvzone.FaceDetectionModule import FaceDetector
import cv2
import cvzone
from time import time

#--------------------------------------------------------------
outputfolderpath = "Dataset/DataCollect"
classID = 0 # 0 fake , 1 real
Debug = True
confidence_value = 0.8
save = True
blurThreshold = 37 # Larger more focus
#--------------------------------------------------------------
offsetPercentageW = 10
offsetPercentageH = 20
camWidth, camHeight = 1920,1080
floatingPoint = 6
#--------------------------------------------------------------

cap = cv2.VideoCapture(0)
cap.set(3, camWidth)
cap.set(4, camHeight)
detector = FaceDetector()
while True:
    success, img = cap.read()
    imgout = img.copy()
    img, bboxs = detector.findFaces(img,draw=False)
```

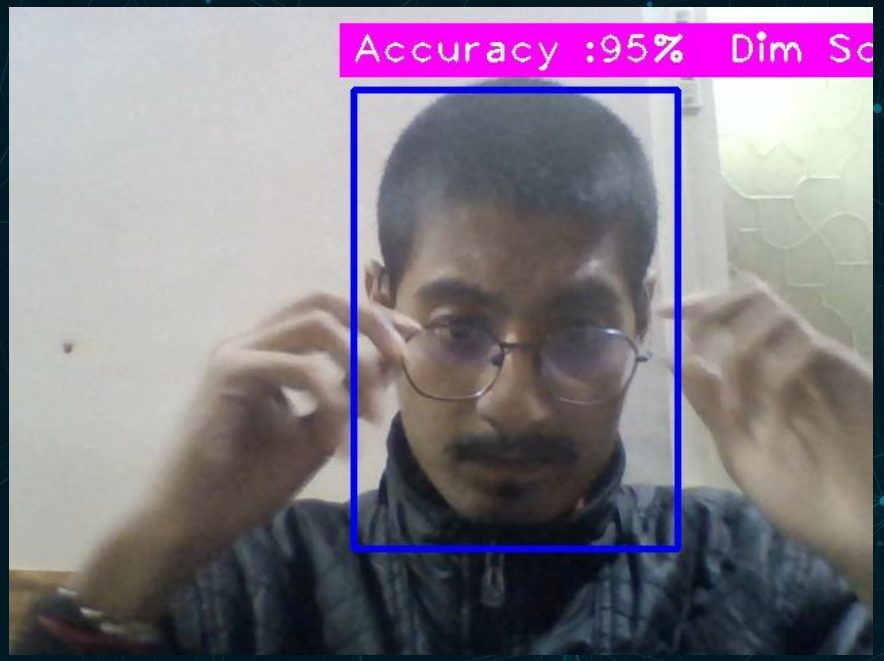
main.py | Yolo-Webcam.py | FaceDetector.py | DataCollection - 3.py

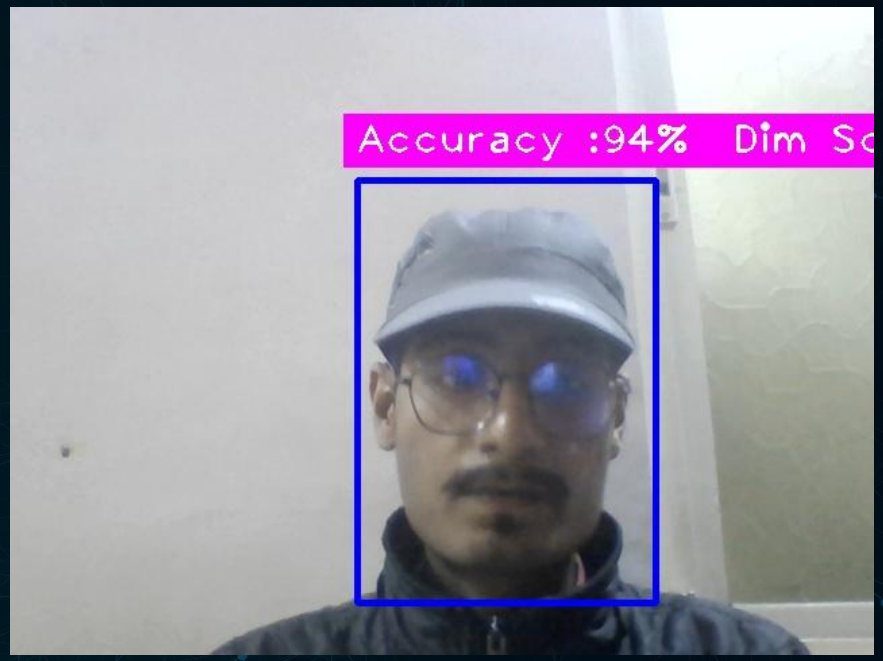
DataCollection - 3.py > ...

```
28
29      listofblur = [] # Indicates blur image in form of either true or false
30      listofinfo = [] # Contains information of class and other attributes
31      if bboxs:
32          # bboxInfo - "id","bbox","score","center"
33
34          for bbox in bboxs:
35              x, y, w, h = bbox["bbox"]
36              score = bbox["score"][0]
37              # for the check the required ratio : dimenions of bounding boxes
38              #print(x, y, w, h)
39
40              #++++++++++++++++++++++++ measuring score ++++++++++++++++++++++++++++++++
41              if score > confidence_value:
42
43                  #++++++++++++++++++++++++ Bounding Boxes sizes ++++++++++++++++++++++++++++++++
44                  #for x axis "Width"
45                  offsetW = (offsetPercentageW / 100) * w
46                  x = int(x - offsetW)
47                  w = int(w + offsetW * 2)
48
49                  #for y axis "Height"
50                  offsetH = (offsetPercentageH / 100) * h
51                  y = int(y - offsetH*3)
52                  h = int(h + offsetH * 3.5)
53
54                  #++++++++++++++++++++++++ Avoid face Detection cutting off BBoxs++++++++++++++++++++++++++++++++
55                  if x < 0 : x = 0
```

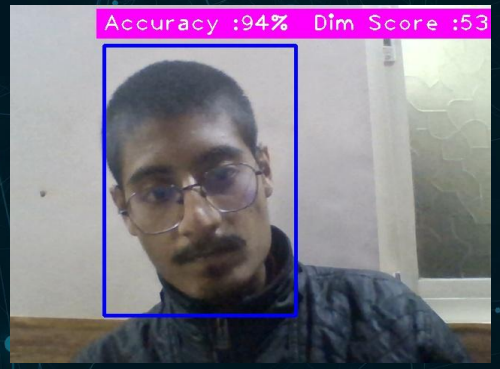
```
57                  if y < 0 : y = 0
58                      #pass
59                  if w < 0 : w = 0
60                  #pass
61                  if h < 0 : h = 0
62
63                  #++++++++++++++++++++++++++ Find bluriness ++++++++++++++++++++++++++++++++++
64                  imgface = img[y:y + h , x:x + w]
65                  #cv2.imshow("Face",imgface)
66                  blurValue = int(cv2.Laplacian(imgface, cv2.CV_64F).var())
67                  if blurValue > blurThreshold:
68                      listofblur.append(True)
69                  else:
70                      listofblur.append(False)
71
72                  #++++++++++++++++++++++++++++++ Normalize form ++++++++++++++++++++++++++++++++
73                  ih, iw,_ = img.shape
74                  xc, yc = x+w/2, y+h/2
75                  xcn, ycn = round(xc/iw,floatingPoint), round(yc/ih,floatingPoint)
76                  wn, hn = round(w/iw,floatingPoint), round(h/ih,floatingPoint)
77                  #print(xcn, ycn,wn, hn)
78
79                  #++++++++++++++++++++++++++++++ the value above 1 ++++++++++++++++++++++++++++++++
80                  if xcn > 1 : xcn = 1
81                  if ycn > 1 : ycn = 1
82                  if wn > 1 : wn = 1
83                  if hn > 1 : hn = 1
```

```
84
85                  listofinfo.append(f"{classID} {xcn} {ycn} {wn} {hn}\n")
86
87                  #+++++++++++++++++++++++++ Drawing a Boxes ++++++++++++++++++++++++++++++++++
88                  cv2.rectangle(imgout, (x, y, w, h), (255,0,0),3)
89                  cvzone.putTextRect(imgout, f"Score :{int(score*100)}%  Blur :{blurValue}",(x,y-20),scale=2,thickness=2)
90
91                  if Debug:
92                      cv2.rectangle(img, (x, y, w, h), (255,0,0),3)
93                      cvzone.putTextRect(img, f"Accuracy :{int(score*100)}%  Dim Score :{blurValue}",(x,y-20),scale=2,thick
94
95          if save:
96              if all(listofblur) and listofblur!=[]:
97                  #+++++++++++++++ save img +++++++++++++++++
98                  timeof = str(time())
99                  timeof =  timeof.replace(".","")
100                 cv2.imwrite(f"{outputfolderpath}/{timeof}.jpg",img)
101
102                 #+++++++++++++++ save Text file +++++++++++++++++
103                 for info in listofinfo :
104                     f = open(f'{outputfolderpath}/{timeof}.txt', 'a')
105                     f.write(info)
106                     f.close()
107
108     cv2.imshow("Image", imgout)
109     cv2.waitKey(1)
```


Accuracy :95% Dim Sc


Accuracy :94% Dim Sc


Accuracy :88%


Accuracy :94% Dim Score :53


Accuracy :90% Dim Score :63

Conclusion ~

In summary, our deepfake detection web application represents a significant step forward in addressing the pressing issue of deepfake content. With its intuitive interface, robust detection capabilities, and commitment to user education, our project aims to enhance media trustworthiness and combat deceptive practices online. By providing users with the tools to verify the authenticity of digital content, we contribute to a more vigilant and responsible digital society.

Thank You

For listening.