

# DeepFake Detection and Mitigation System

USING DEEP AND MACHINE LEARNING APPROACH

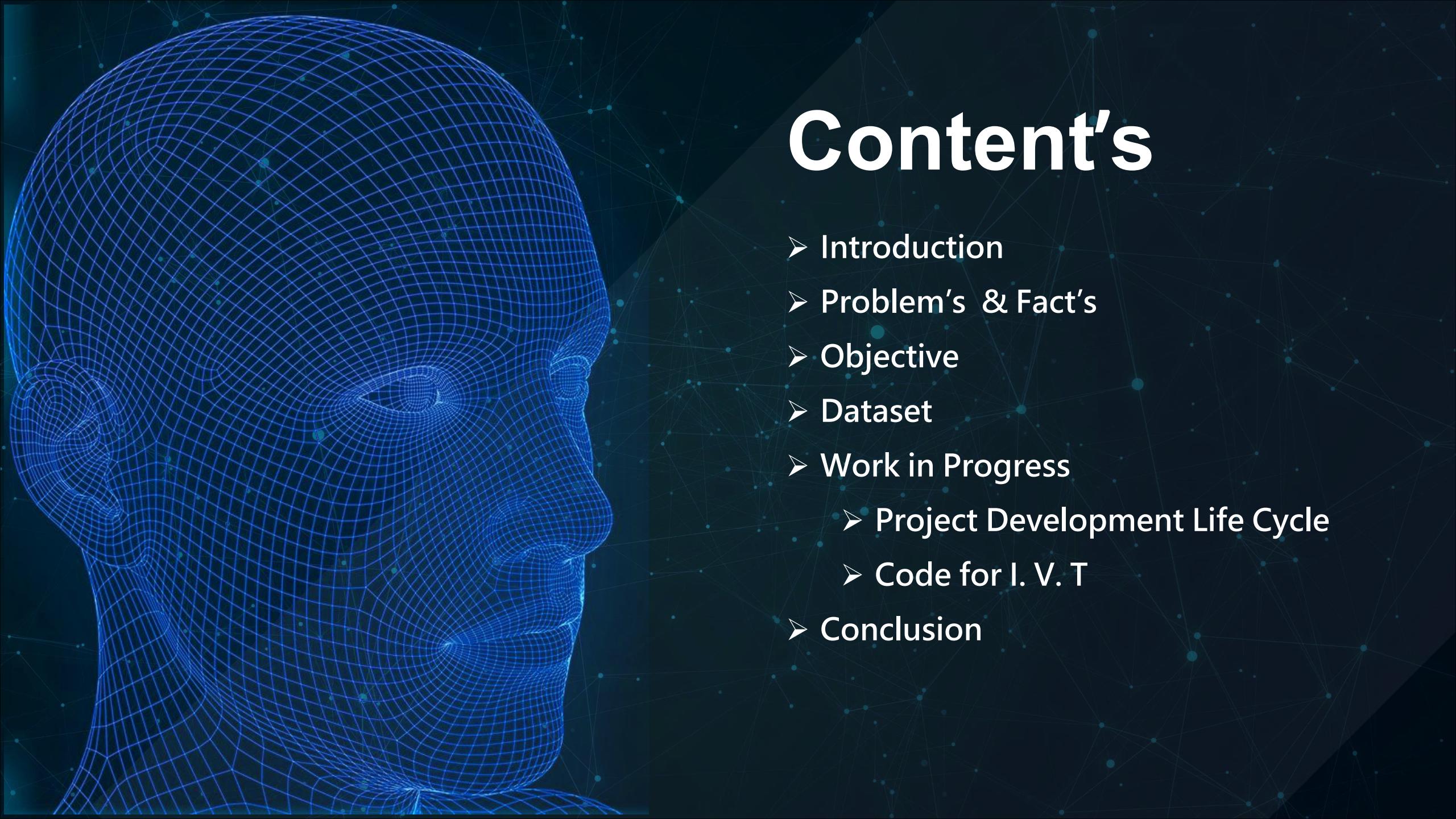
**GRP: 38**

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深偽 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

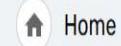


SIGN IN

Dark Mode



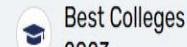
Premium



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

ADVERTISEMENT



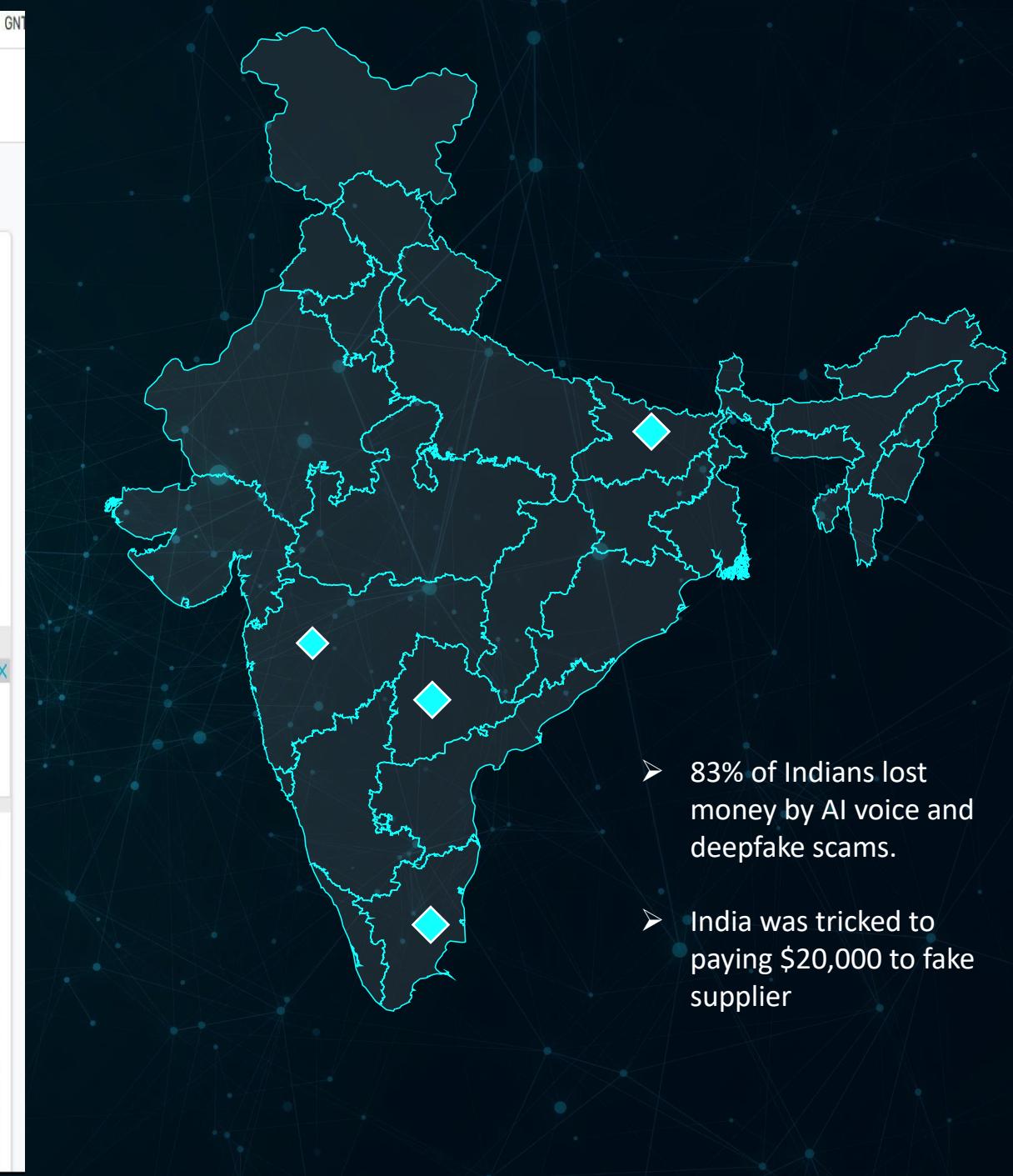
HitPaw AI Video Enhancer

Enhance Video Quality Easily. AI Video Upscaler. Convert Low Res Videos to 4K, 60FPS

HitPaw



Open &gt;



- 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

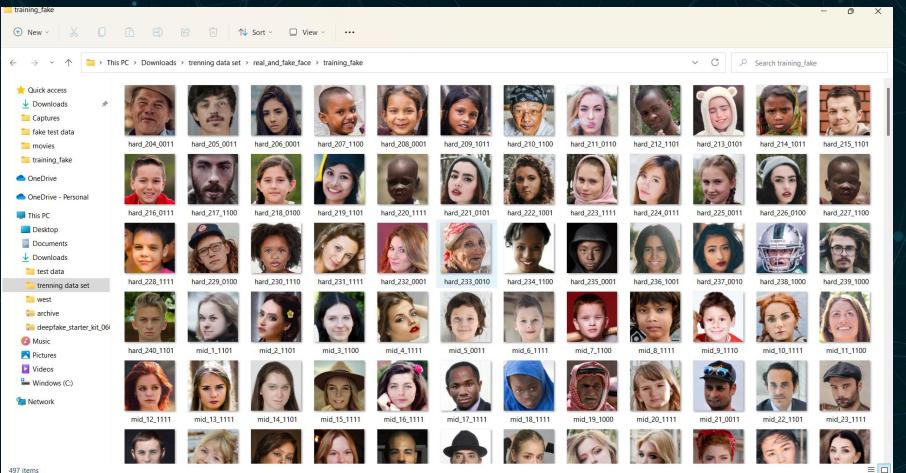
- Use a Data collection script with Face Detection
- Maintain Equal Even ratio in Real and Fake images
- Yolo or MobileNetSSD Model (which are more reliable & considerable ).



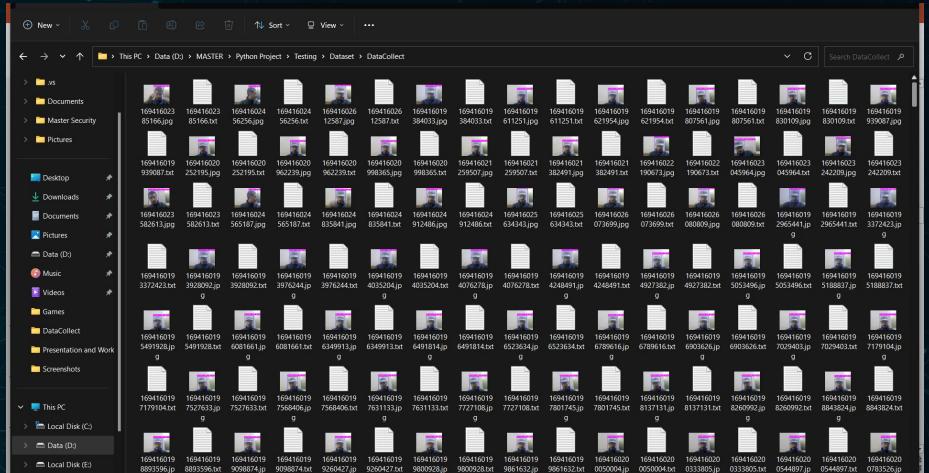
## Video Based

- 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.

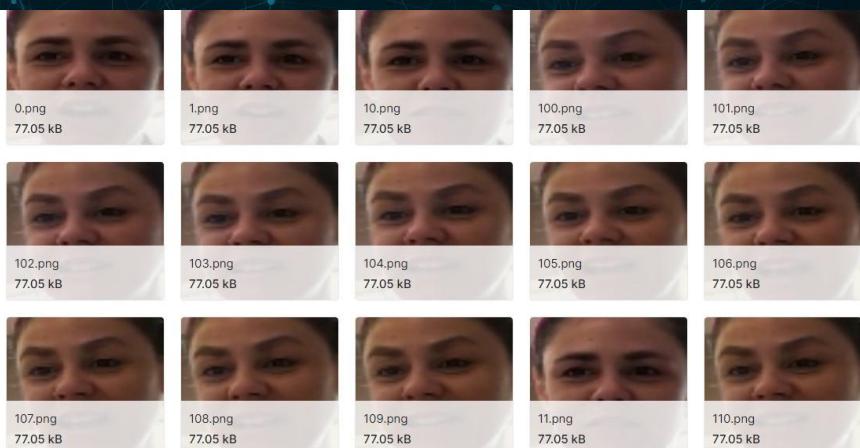
## For image



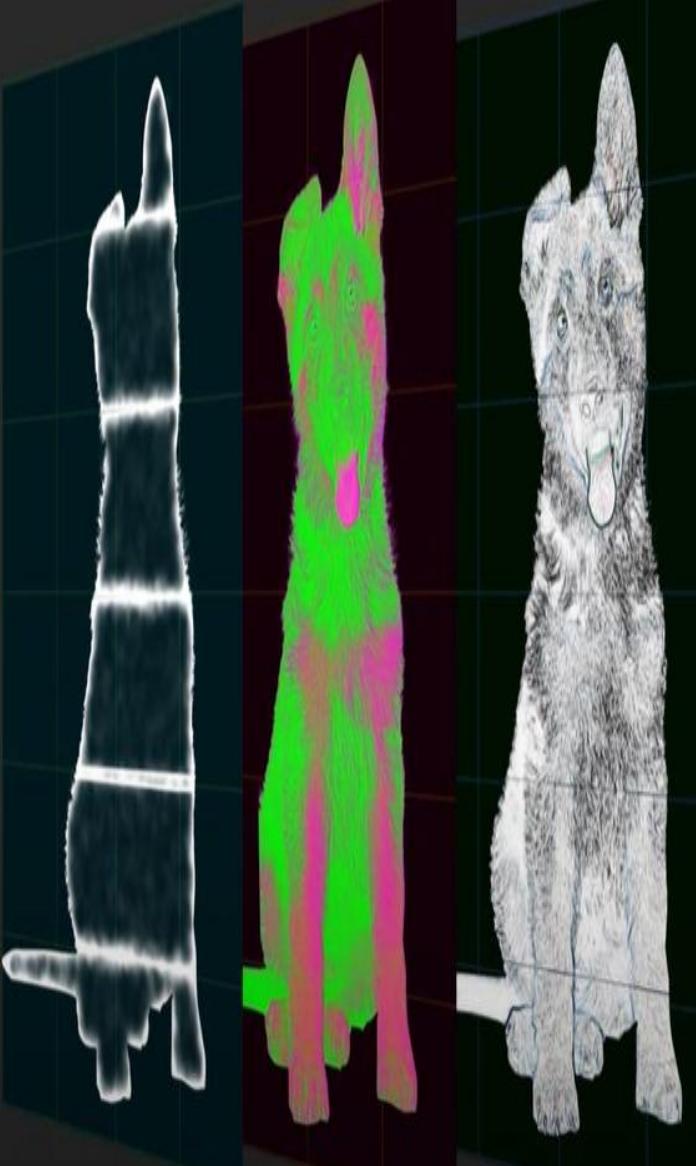
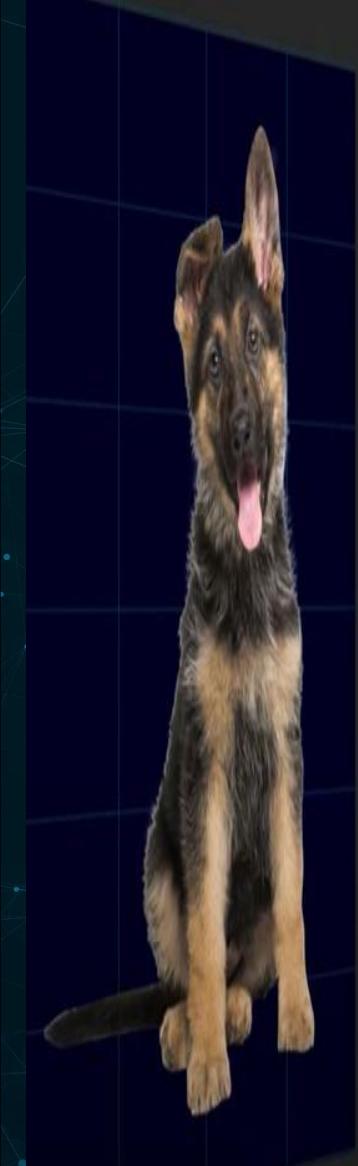
## For Real Time



## For video



# MODELS:



CNN

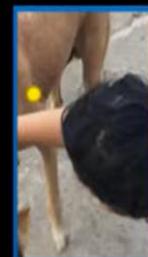
YOLO

$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix} \begin{bmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{bmatrix}$$

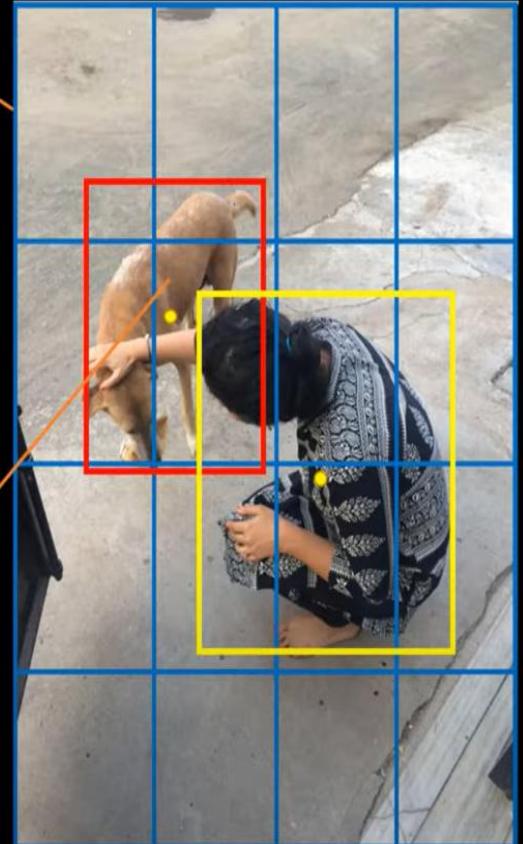
$$\begin{bmatrix} 1 \\ 0.05 \\ 0.3 \\ 2 \\ 1.3 \\ 1 \\ 0 \end{bmatrix}$$



(1,1)



(0,0)



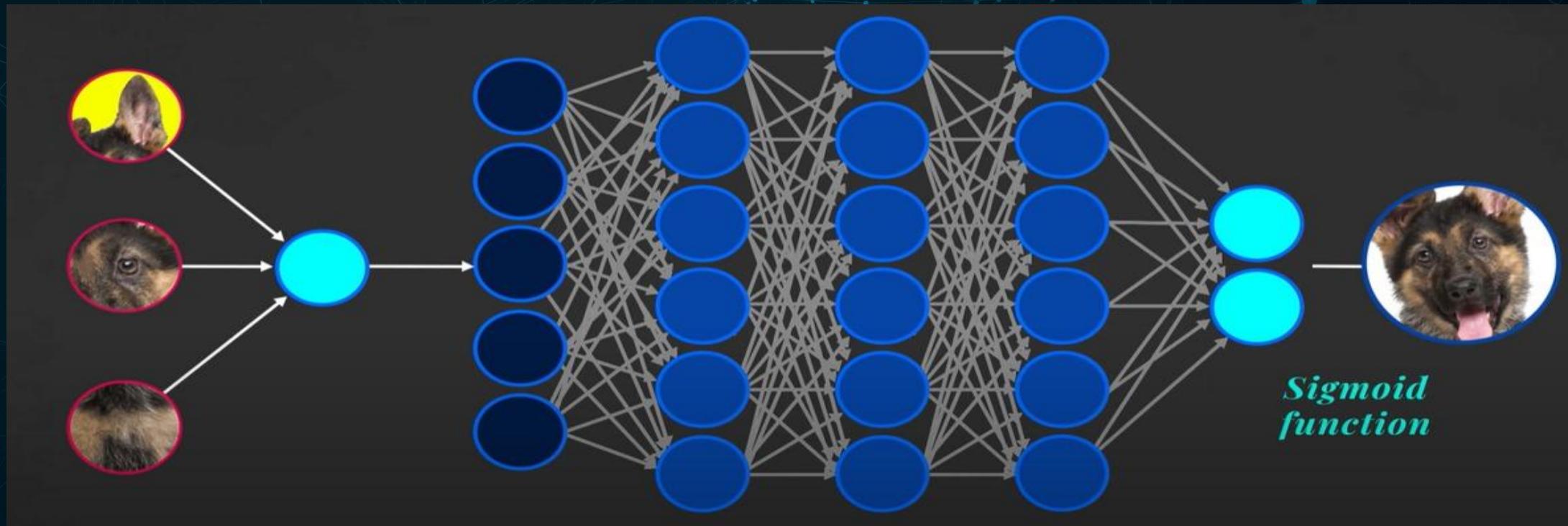
and 1.3 is height. So it is this

0	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	

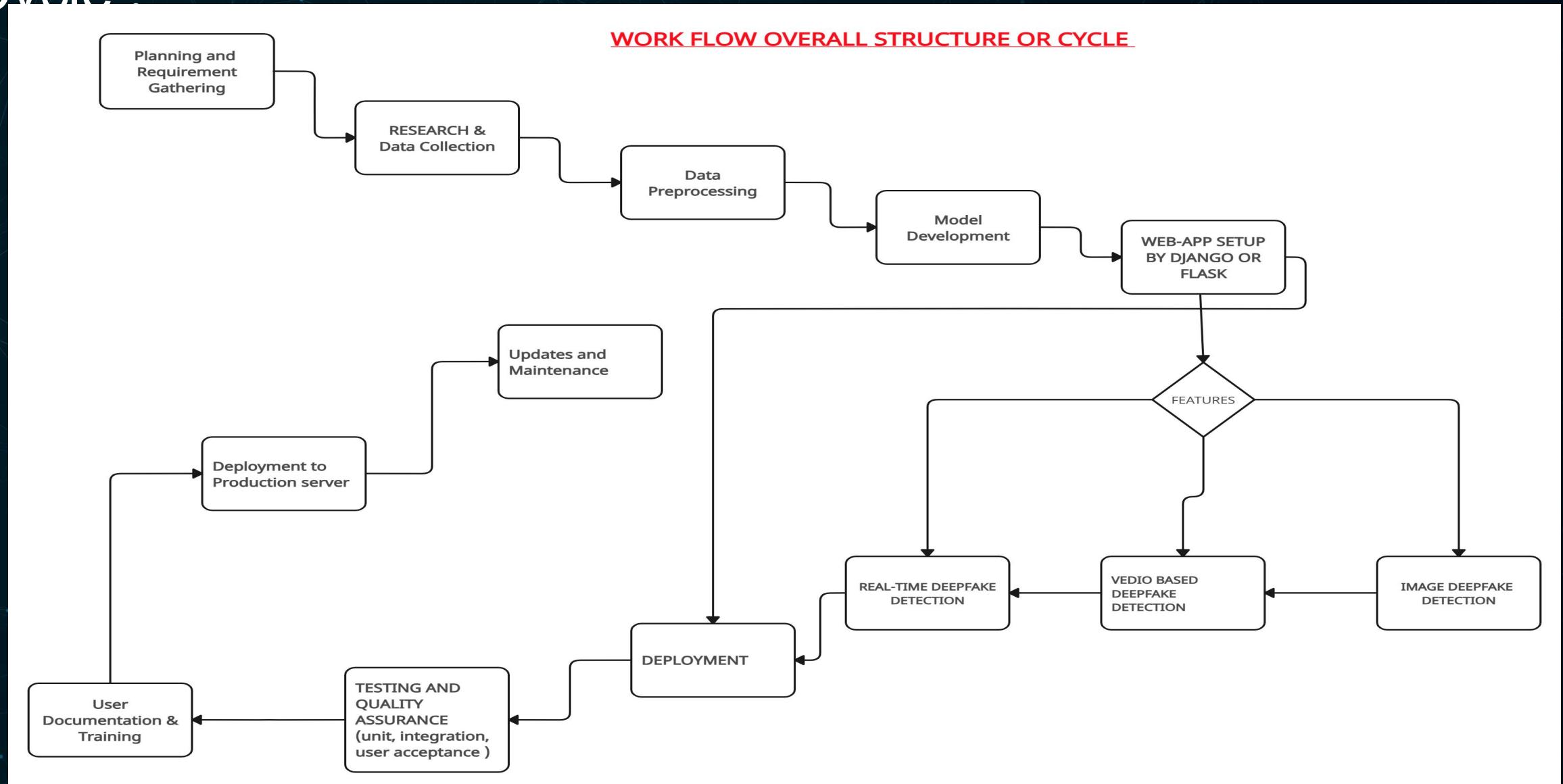
5	8	9
1	3	6
4	7	2

*Dot Product*

	0	0	2
	0	5	2
	7	9	3



# ➤WP - Project Development Cycle :



# For Image Based

The image shows a Jupyter Notebook interface with a dark blue background featuring a network graph pattern. The notebook has two main sections: a top cell and a bottom cell.

**Top Cell:**

```
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')
```

Output:

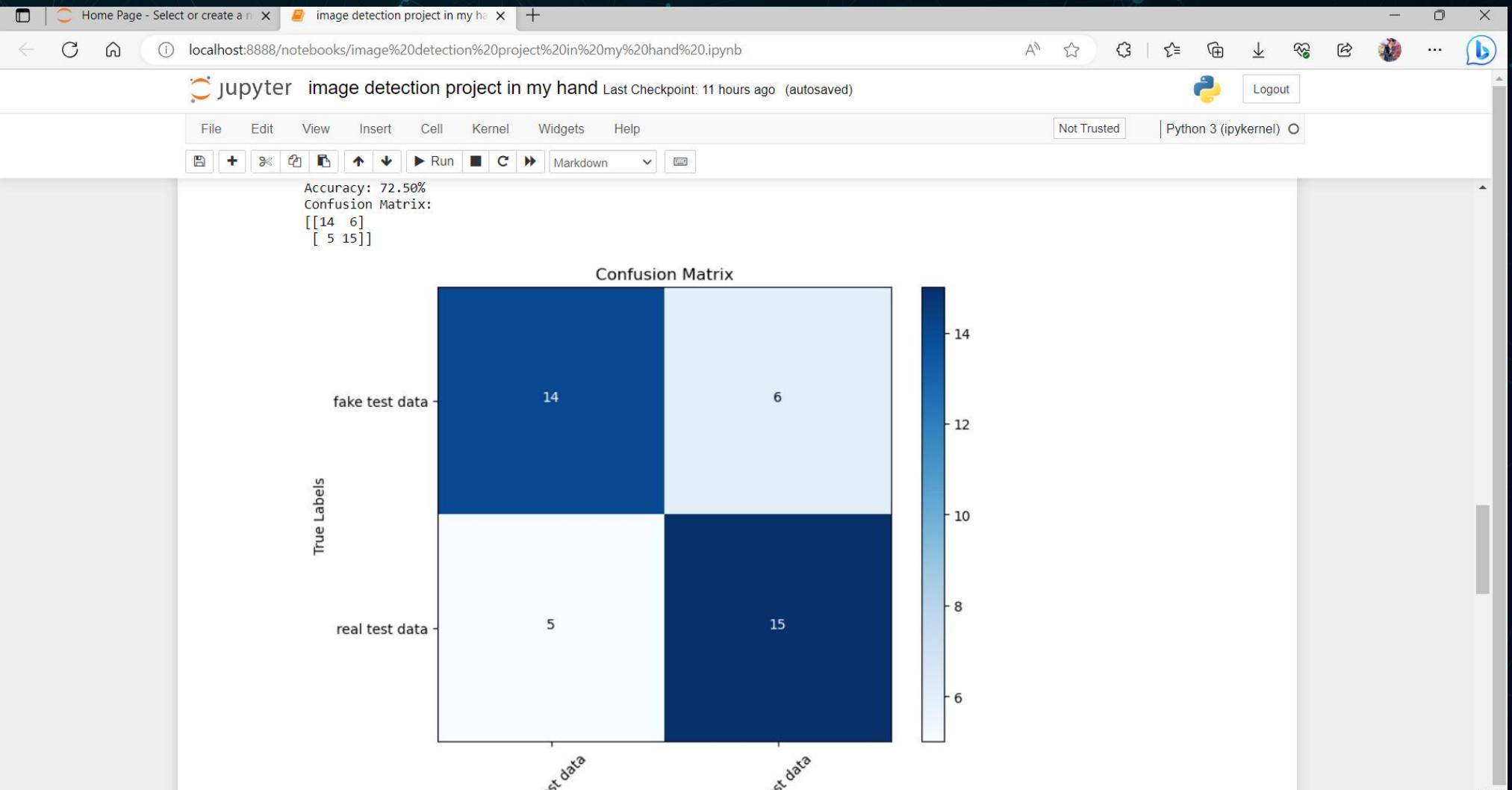
```
Accuracy: 70.00%
Confusion Matrix:
[[13  7]
 [ 5 15]]
```

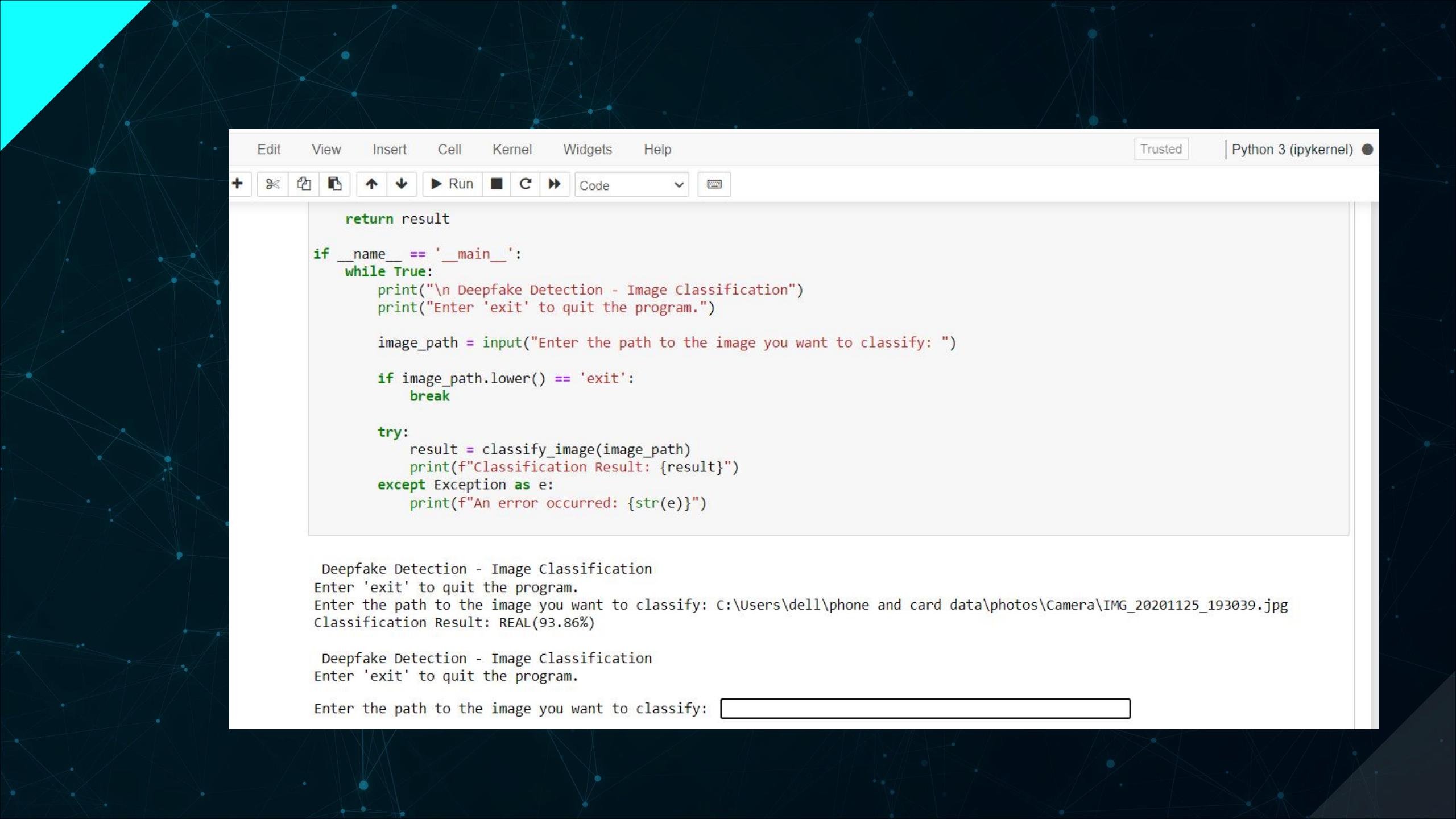
**Bottom Cell:**

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()
```





A Jupyter Notebook interface showing Python code for deepfake detection. The code uses the `classify_image` function to classify images and handles exceptions.

```
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)

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%)

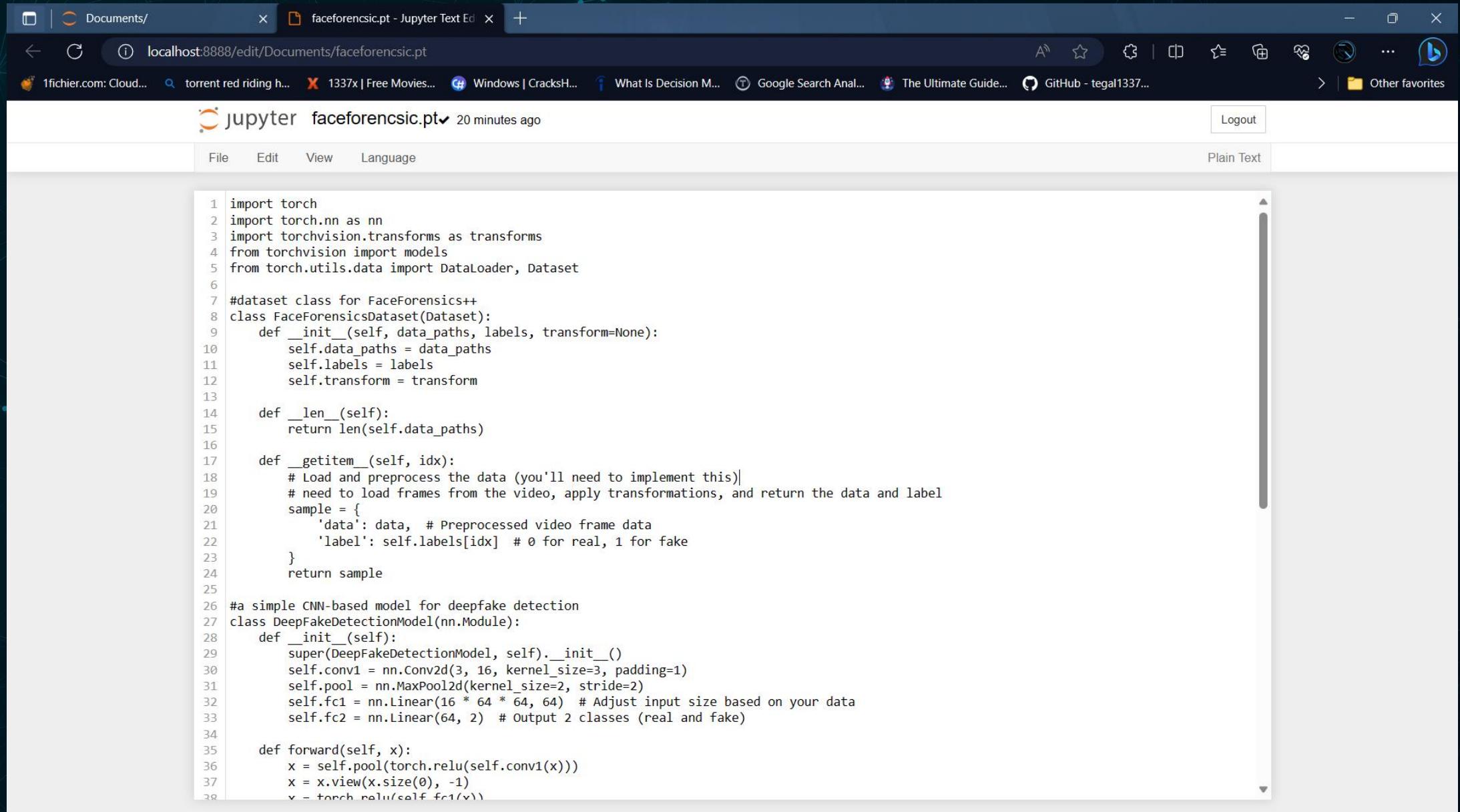
Deepfake Detection - Image Classification  
Enter 'exit' to quit the program.  
Enter the path to the image you want to classify: C:\Users\dell\Downloads\test data\fake test data\mid\_468\_1110.jpg  
Classification Result: REAL(97.77%)

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



The screenshot shows a Jupyter Notebook interface running on a local host. The title bar indicates the file is named "faceforencsic.pt". The notebook contains the following Python code:

```
1 import torch
2 import torch.nn as nn
3 import torchvision.transforms as transforms
4 from torchvision import models
5 from torch.utils.data import DataLoader, Dataset
6
7 #dataset class for FaceForensics++
8 class FaceForensicsDataset(Dataset):
9     def __init__(self, data_paths, labels, transform=None):
10         self.data_paths = data_paths
11         self.labels = labels
12         self.transform = transform
13
14     def __len__(self):
15         return len(self.data_paths)
16
17     def __getitem__(self, idx):
18         # Load and preprocess the data (you'll need to implement this)
19         # need to load frames from the video, apply transformations, and return the data and label
20         sample = {
21             'data': data, # Preprocessed video frame data
22             'label': self.labels[idx] # 0 for real, 1 for fake
23         }
24         return sample
25
26 #a simple CNN-based model for deepfake detection
27 class DeepFakeDetectionModel(nn.Module):
28     def __init__(self):
29         super(DeepFakeDetectionModel, self).__init__()
30         self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
31         self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
32         self.fc1 = nn.Linear(16 * 64 * 64, 64) # Adjust input size based on your data
33         self.fc2 = nn.Linear(64, 2) # Output 2 classes (real and fake)
34
35     def forward(self, x):
36         x = self.pool(torch.relu(self.conv1(x)))
37         x = x.view(x.size(0), -1)
38         x = torch.relu(self.fc1(x))
```

Documents/ CNN MODEL.pt - Jupyter Text Ed faceforencsic.pt - Jupyter Text Ed +

localhost:8888/edit/Documents/faceforencsic.pt

1fichier.com: Cloud... torrent red riding h... 1337x | Free Movies... Windows | CracksH... What Is Decision M... Google Search Anal... The Ultimate Guide... GitHub - tegal1337...

Logout

jupyter faceforencsic.pt 32 minutes ago

File Edit View Language Plain Text

```
35     def forward(self, x):
36         x = self.pool(torch.relu(self.conv1(x)))
37         x = x.view(x.size(0), -1)
38         x = torch.relu(self.fc1(x))
39         x = self.fc2(x)
40         return x
41
42 # Data preprocessing and augmentation
43 transform = transforms.Compose([
44     transforms.Resize((128, 128)),
45     transforms.ToTensor(),
46     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
47 ])
48
49 # Create datasets and dataloaders
50 train_dataset = FaceForensicsDataset(train_data_paths, train_labels, transform=transform)
51 train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
52
53 # Initialize the model
54 model = DeepFakeDetectionModel()
55
56 # Define a loss function and optimizer
57 criterion = nn.CrossEntropyLoss()
58 optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
59
60 # Training loop
61 num_epochs = 10
62 for epoch in range(num_epochs):
63     for batch in train_loader:
64         data, labels = batch['data'], batch['label']
65         optimizer.zero_grad()
66         outputs = model(data)
67         loss = criterion(outputs, labels)
68         loss.backward()
69         optimizer.step()
70
71 torch.save(model.state_dict(), 'deepfake_detection_model.pth')
72
```

Documents/ CNN MODEL.pt - Jupyter Text Ed faceforencsic.pt - Jupyter Text Ed +

localhost:8888/edit/Documents/faceforencsic.pt

1fichier.com: Cloud... torrent red riding h... 1337x | Free Movies... Windows | CracksH... What Is Decision M... Google Search Anal... The Ultimate Guide... GitHub - tegal1337...

Logout Plain Text

```
35     def forward(self, x):
36         x = self.pool(torch.relu(self.conv1(x)))
37         x = x.view(x.size(0), -1)
38         x = torch.relu(self.fc1(x))
39         x = self.fc2(x)
40         return x
41
42 # Data preprocessing and augmentation
43 transform = transforms.Compose([
44     transforms.Resize((128, 128)),
45     transforms.ToTensor(),
46     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
47 ])
48
49 # Create datasets and dataloaders
50 train_dataset = FaceForensicsDataset(train_data_paths, train_labels, transform=transform)
51 train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
52
53 # Initialize the model
54 model = DeepFakeDetectionModel()
55
56 # Define a loss function and optimizer
57 criterion = nn.CrossEntropyLoss()
58 optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
59
60 # Training loop
61 num_epochs = 10
62 for epoch in range(num_epochs):
63     for batch in train_loader:
64         data, labels = batch['data'], batch['label']
65         optimizer.zero_grad()
66         outputs = model(data)
67         loss = criterion(outputs, labels)
68         loss.backward()
69         optimizer.step()
70
71 torch.save(model.state_dict(), 'deepfake_detection_model.pth')
72
```

```
1 import torch
2 import torch.nn as nn
3
4 class DeepFakeDetectionCNN(nn.Module):
5     def __init__(self, num_classes=2):
6         super(DeepFakeDetectionCNN, self).__init__()
7
7     # Convolutional layers
8     self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=1, padding=1)
9     self.relu1 = nn.ReLU()
10    self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
11
12    self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=1)
13    self.relu2 = nn.ReLU()
14    self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
15
16    self.conv3 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1)
17    self.relu3 = nn.ReLU()
18    self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)
19
20    # Fully connected layers
21    self.fc1 = nn.Linear(in_features=64 * 16 * 16, out_features=128)
22    self.relu4 = nn.ReLU()
23    self.dropout = nn.Dropout(0.5)
24
25    self.fc2 = nn.Linear(in_features=128, out_features=num_classes)
26
27    def forward(self, x):
28        # Convolutional layers
29        x = self.pool1(self.relu1(self.conv1(x)))
30        x = self.pool2(self.relu2(self.conv2(x)))
31        x = self.pool3(self.relu3(self.conv3(x)))
32
33        # Flatten the output for the fully connected layers
34        x = x.view(x.size(0), -1)
35
36        # Fully connected layers
37        x = self.relu4(self.fc1(x))
38
```

# For Real

Ti

File DataCollection - 3.py > ...

```
1  from cvzone.FaceDetectionModule import FaceDetector
2  import cv2
3  import cvzone
4  from time import time
5
6  #-----
7  outputFolderPath = "Dataset/DataCollect"
8  classID = 0 # 0 fake , 1 real
9  Debug = True
10 confidence_value = 0.8
11 save = True
12 blurThreshold = 37 # Larger more focus
13 #-----
14 offsetPercentageW = 10
15 offsetPercentageH = 20
16 camWidth, camHeight = 1920,1080
17 floatingPoint = 6
18 #-----
19
20 cap = cv2.VideoCapture(0)
21 cap.set(3, camWidth)
22 cap.set(4, camHeight)
23 detector = FaceDetector()
24 while True:
25     success, img = cap.read()
26     imgout = img.copy()
27     img, bboxes = detector.findFaces(img, draw=False)
```

The screenshot shows a Python code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Testing
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Run.
- Code Area:** The active tab is "DataCollection - 3.py". The code is as follows:

```
28
29     listofblur = [] # Indicates blur image in form of either true or false
30     listofinfo = [] # Contains information of class and other attributes
31     if bboxes:
32         # bboxInfo - "id","bbox","score","center"
33
34         for bbox in bboxes:
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 BBoxes#####
55                 if x < 0 : x = 0
```

The code is part of a script for data collection, specifically handling bounding boxes (bbox), scores, and offsets for face detection. It includes logic to adjust bounding box dimensions based on a confidence threshold and to prevent detection from cutting off faces by adjusting offsets.

The screenshot shows a Python code editor interface with the following details:

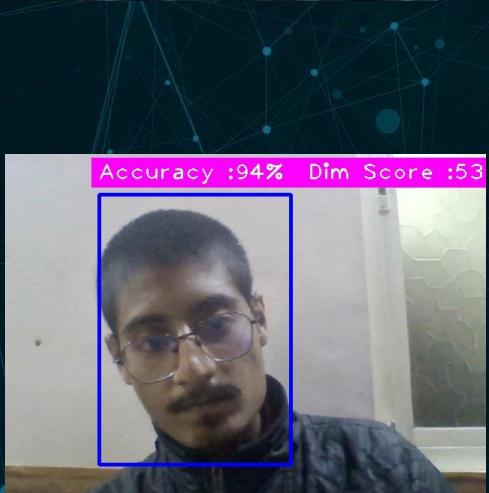
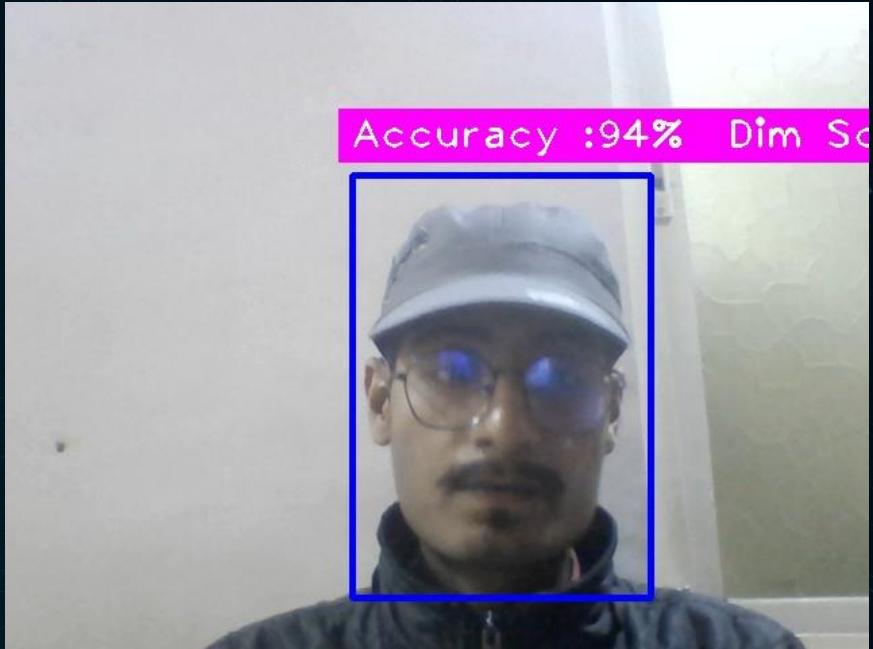
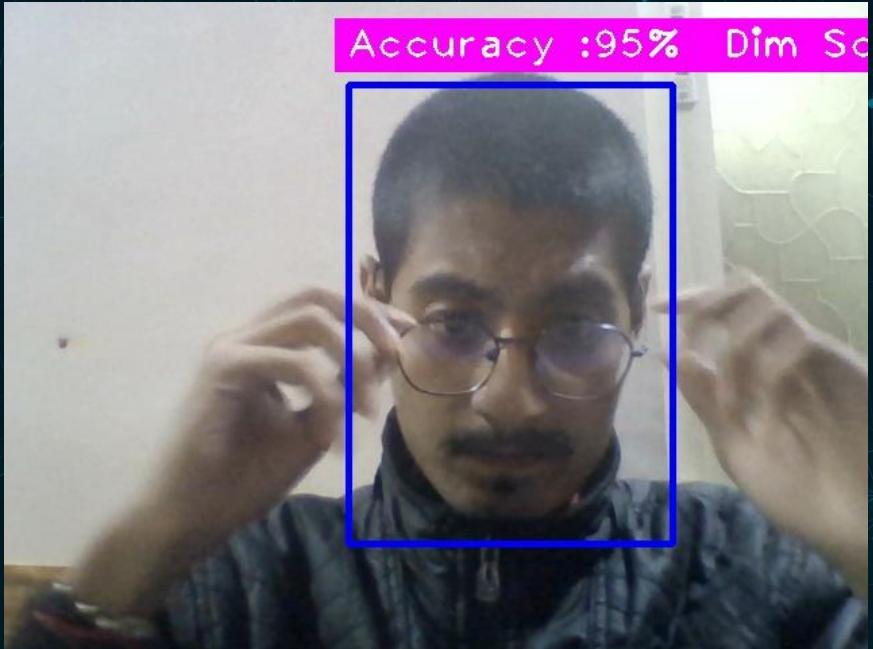
- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Testing
- Toolbar:** Includes icons for file operations (New, Open, Save, Find, Replace, etc.) and a search icon.
- Code Editor:** The active tab is "DataCollection - 3.py". The code itself is as follows:

```
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
```

**Status Bar:** Python extension loading... | Screen Reader Optimized | Ln 10, Col 23 | Spaces: 4 | UTF-8 | CRLF | Python 3.10.7 (.venv: venv) | Go Live | Prettier

```
main.py Yolo-Webcam.py FaceDetector.py DataCollection - 3.py X

DataCollection - 3.py > ...
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,thickness=2)
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)
```



# 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.