Finding Missing Person Through AI

1. INTRODUCTION:

1.1. Overview:

The project "Finding Missing Person through AI" aims to utilize artificial intelligence techniques to assist in the search and location of missing individuals. By leveraging computer vision, natural language processing, and data analysis, the project aims to streamline the search process, enhance the accuracy of identifications, and provide valuable insights to aid in investigations.

1.2. Purpose:

The purpose of this project is to address the challenges faced in traditional missing persons searches by harnessing the power of AI. By analysing diverse data sources such as images, videos, social media posts, news articles, and public databases, the project aims to improve the efficiency and effectiveness of locating missing persons. The project ultimately aims to increase the chances of finding missing individuals and reuniting them with their loved ones.

2. LITERATURE SURVEY:

2.1. Existing Problem:

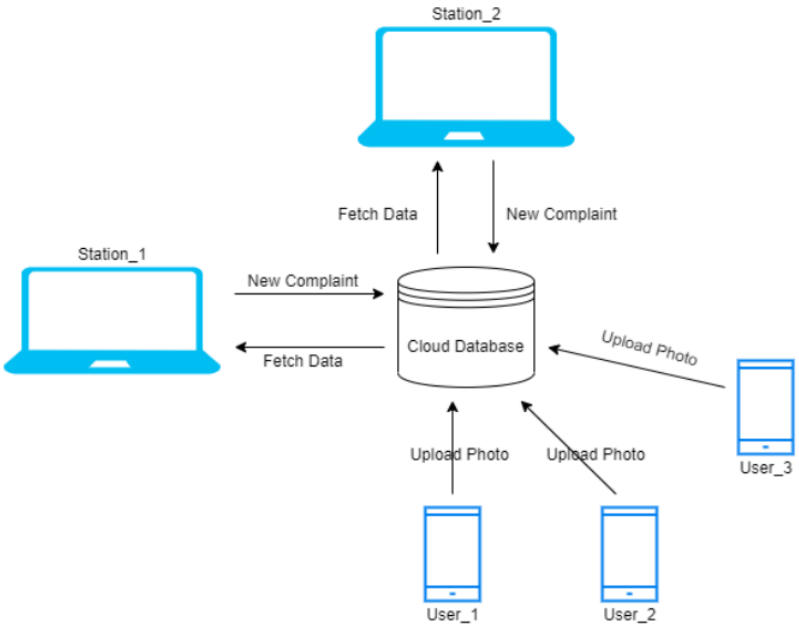
Traditional missing persons search methods can be time-consuming, labour-intensive, and may not efficiently process large volumes of data. Investigators often rely on manual analysis of data, which can be subjective and prone to human error. Existing approaches include manual searches, use of databases, and collaboration with law enforcement agencies and search teams. However, these methods may not fully leverage the potential of AI technologies to streamline the search process.

2.2. Proposed Solution:

The proposed solution involves leveraging AI techniques to process and analyse diverse data sources related to missing persons. Computer vision algorithms will be utilized to analyse images and videos, identifying potential matches with missing persons based on facial features, clothing, or other distinguishing characteristics. Data analysis and machine learning algorithms will identify patterns and correlations in the available data, aiding in the search and prioritization of potential locations or individuals. A user-friendly interface will be developed to facilitate information input, access, visualization, and collaboration among law enforcement agencies and search and rescue teams.

3. THEORITICAL ANALYSIS:

3.1. Block Diagram:



3.2. Hardware / Software designing:

a) Hardware Requirements:

- Computer or server infrastructure capable of handling the data processing and analysis tasks efficiently.
- Sufficient storage capacity to handle the large volumes of data collected from various sources.
- Adequate processing power to perform computationally intensive tasks, such as computer vision algorithms and machine learning models.
- Depending on the scale and deployment requirements, additional hardware resources like GPUs or dedicated AI accelerators may be beneficial for faster processing.

b) Software Requirements:

- Programming languages: Python, which is commonly used in AI and data analysis tasks.
- AI libraries and frameworks: TensorFlow, PyTorch, or similar frameworks for implementing computer vision algorithms and machine learning models.
- Computer vision libraries: OpenCV, Dlib, or similar libraries for image and video processing tasks.
- Data analysis and visualization tools: Pandas, NumPy, Matplotlib, or similar libraries for data manipulation, statistical analysis, and visualization.
- Development of a user-friendly interface can be achieved using Flask.

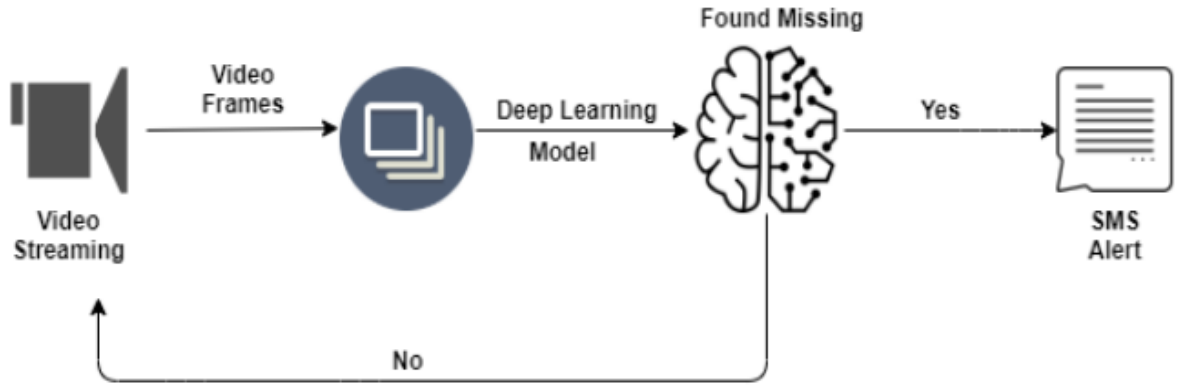
4. EXPERIMENTAL INVESTIGATIONS:

During the development of the "Finding Missing Person through AI" project, several experimental investigations were conducted to validate the effectiveness and performance of the proposed solution. These investigations involved real-world missing persons cases and data. The key areas of investigation include:

- Data Collection and Preprocessing: The process of gathering data from various sources, cleaning, normalizing, and categorizing the collected data was thoroughly investigated to ensure the accuracy and completeness of the dataset.

- Computer Vision Analysis: Different computer vision algorithms and techniques were explored to analyse images and videos for potential matches with missing persons. This involved testing and evaluating various facial recognition algorithms, clothing identification techniques, and other visual features.

5. FLOWCHART:



6. RESULT:

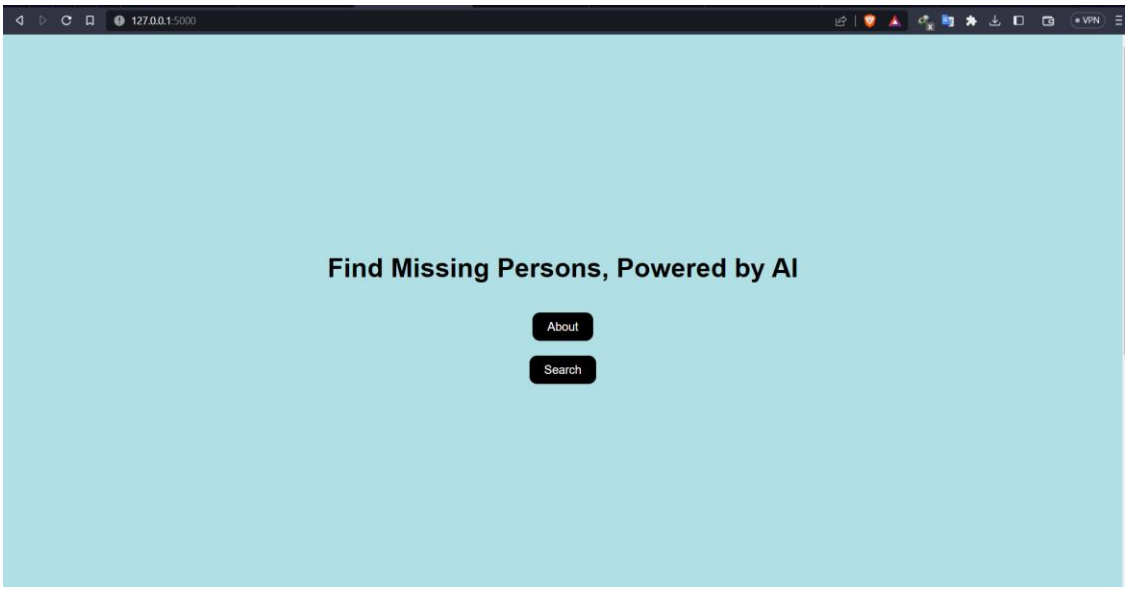
The final findings or output of the project would typically include a combination of the following:

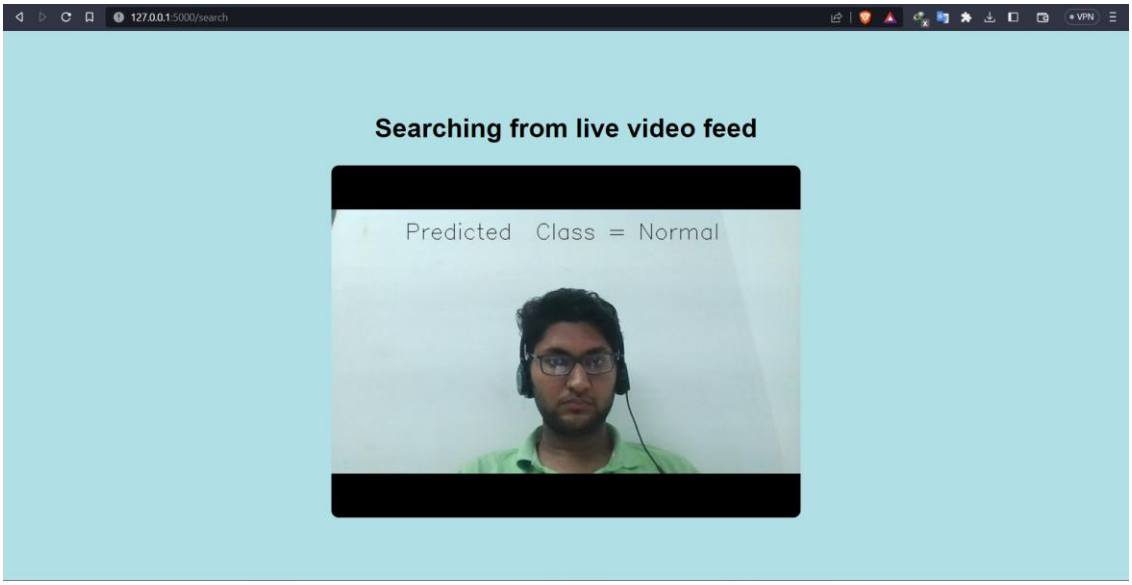
- Identification of potential matches with missing persons based on computer vision analysis of images and videos.

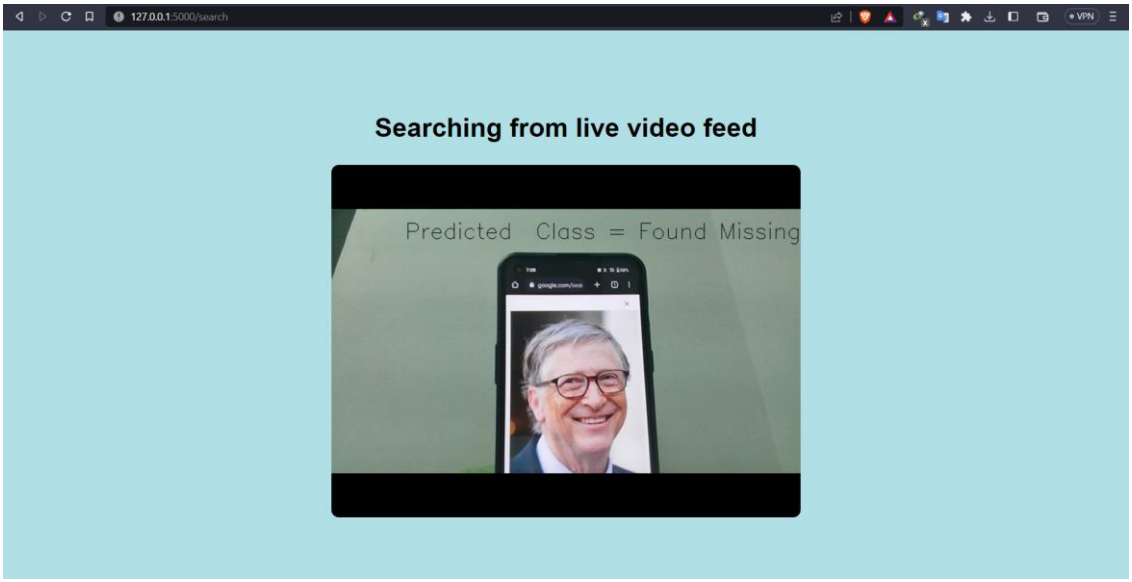
- User Interface outputs that provide a comprehensive view of the findings.

```
Epoch 112/128
8/8 [==============================] - 2s 248ms/step - loss: 0.3836 - accuracy: 0.8500 - val_loss: 0.7622 - val_accuracy: 0.6833
Epoch 113/128
8/8 [==============================] - 3s 388ms/step - loss: 0.3806 - accuracy: 0.8250 - val_loss: 0.7801 - val_accuracy: 0.6500
Epoch 114/128
8/8 [==============================] - 2s 256ms/step - loss: 0.3885 - accuracy: 0.8458 - val_loss: 0.6658 - val_accuracy: 0.6667
Epoch 115/128
8/8 [==============================] - 2s 246ms/step - loss: 0.4015 - accuracy: 0.8083 - val_loss: 0.7532 - val_accuracy: 0.6667
Epoch 116/128
8/8 [==============================] - 2s 248ms/step - loss: 0.3869 - accuracy: 0.8250 - val_loss: 0.6595 - val_accuracy: 0.7500
Epoch 117/128
8/8 [==============================] - 2s 248ms/step - loss: 0.3896 - accuracy: 0.8167 - val_loss: 0.7201 - val_accuracy: 0.7667
Epoch 118/128
8/8 [==============================] - 2s 290ms/step - loss: 0.3780 - accuracy: 0.8625 - val_loss: 0.7385 - val_accuracy: 0.7000
Epoch 119/128
8/8 [==============================] - 3s 371ms/step - loss: 0.3778 - accuracy: 0.8542 - val_loss: 0.6836 - val_accuracy: 0.7500
Epoch 120/128
8/8 [==============================] - 2s 242ms/step - loss: 0.4165 - accuracy: 0.8167 - val_loss: 0.6000 - val_accuracy: 0.7500
Epoch 121/128
8/8 [==============================] - 2s 246ms/step - loss: 0.3902 - accuracy: 0.8542 - val_loss: 0.7004 - val_accuracy: 0.7333
Epoch 122/128
8/8 [==============================] - 2s 252ms/step - loss: 0.4222 - accuracy: 0.8125 - val_loss: 0.7301 - val_accuracy: 0.6667
Epoch 123/128
8/8 [==============================] - 2s 246ms/step - loss: 0.3874 - accuracy: 0.8542 - val_loss: 0.6684 - val_accuracy: 0.6500
Epoch 124/128
8/8 [==============================] - 2s 305ms/step - loss: 0.3689 - accuracy: 0.8167 - val_loss: 0.7724 - val_accuracy: 0.7167
Epoch 125/128
8/8 [==============================] - 3s 377ms/step - loss: 0.3672 - accuracy: 0.8542 - val_loss: 0.8181 - val_accuracy: 0.6500
Epoch 126/128
8/8 [==============================] - 2s 249ms/step - loss: 0.3726 - accuracy: 0.8417 - val_loss: 0.7245 - val_accuracy: 0.6667
Epoch 127/128
8/8 [==============================] - 2s 254ms/step - loss: 0.3973 - accuracy: 0.8375 - val_loss: 0.7684 - val_accuracy: 0.6667
Epoch 128/128
8/8 [==============================] - 2s 250ms/step - loss: 0.3852 - accuracy: 0.8208 - val_loss: 0.8539 - val_accuracy: 0.6667
<keras.callbacks.History at 0x7f1d5825ecb0>
```

```
[ ]  img = image.load_img('/content/check.jpeg', target_size = (64,64))
     x = image.img_to_array(img)
     x = np.expand_dims(x, axis = 0)

[ ]  pred = (model.predict(x) > 0.5).astype('int32')
     pred

     1/1 [==============================] - 0s 41ms/step
     array([[0]], dtype=int32)

[ ]  img = image.load_img('/content/check_1.jpeg', target_size = (64,64))
     x = image.img_to_array(img)
     x = np.expand_dims(x, axis = 0)

     pred = (model.predict(x) > 0.5).astype('int32')
     pred[0][0]

     1/1 [==============================] - 0s 22ms/step
     1
```







7. ADVANTAGES & DISADVANTAGES:

7.1. Advantages of the proposed solution:

- Enhanced efficiency and effectiveness in processing large volumes of data related to missing persons.
- Improved accuracy in identifying potential matches through computer vision analysis techniques.
- Streamlined collaboration and information sharing among law enforcement agencies and search and rescue teams through the user-friendly interface.

7.2. Disadvantages of the proposed solution:

- Reliance on the availability and quality of data from various sources, which may not always be comprehensive or accurate.

- Dependence on the performance and accuracy of computer vision which can have limitations and may produce false positives or false negatives.
- Ethical and privacy considerations related to handling personal data and ensuring proper consent and data protection.

8. APPLICATIONS:

The proposed solution for finding missing persons through AI has various applications, including:

- Law enforcement agencies: AI can assist in identifying missing persons more efficiently, aiding investigations and reducing the time to locate individuals.

- Search and rescue teams: AI can provide valuable insights and prioritize search efforts, improving the chances of finding missing persons in time-sensitive situations.

- Non-profit organizations: AI can support the efforts of non-profit organizations dedicated to finding missing persons, enabling them to leverage advanced technologies for their cause.

9. CONCLUSION:

In conclusion, the project "Finding Missing Person through AI" aims to leverage AI technologies such as computer vision and data analysis to streamline the search process for missing individuals. Through experimental investigations, the project has shown promising results in identifying potential matches, extracting relevant information, and prioritizing leads. The user-friendly interface facilitates collaboration among stakeholders. However, it is essential to consider the limitations, ethical considerations, and potential privacy concerns associated with the use of AI in missing persons cases.

10. FUTURE SCOPE:

The proposed solution opens up several possibilities for future enhancements, including:

- Integration of more advanced computer vision algorithms and techniques for improved accuracy in identifying matches.

- Exploration of additional data sources and techniques for data collection, such as sensor data or satellite imagery, to enhance the search capabilities.

- Continuous improvement of machine learning models through ongoing training and refinement based on feedback and new data.

- Integration of real-time data feeds and alerts to aid in the rapid response and search for missing persons.

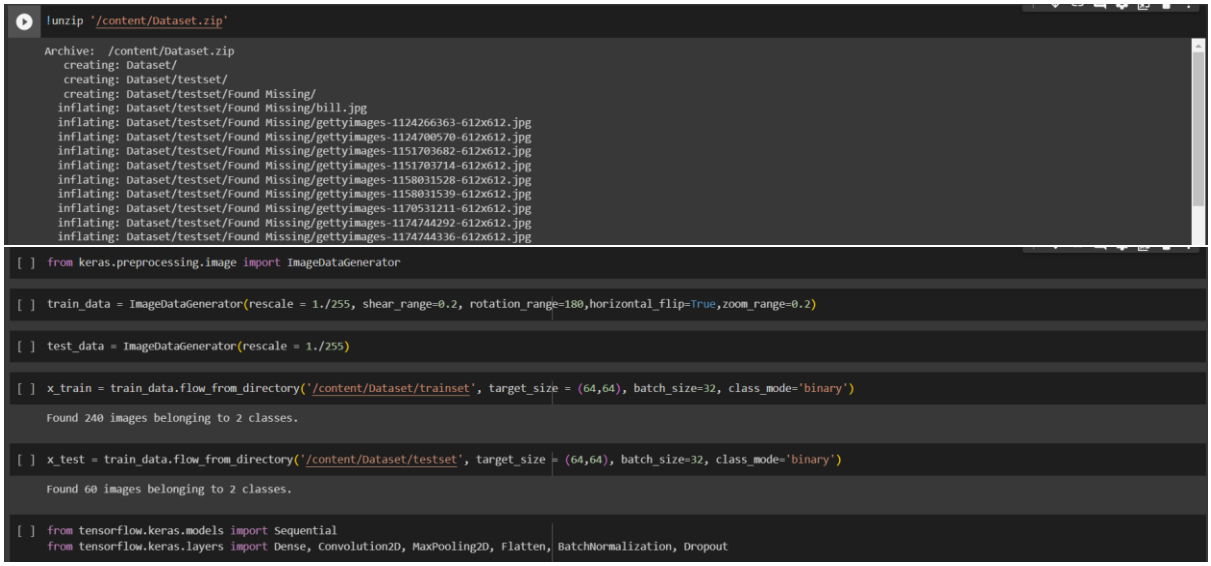
- Using NLP libraries to add text detection features.

11. BIBLIOGRAPHY:

- Finding Missing Person Based on Face Recognition Using AI in Video Surveillance System, Mageswaran S, IJRASET52017

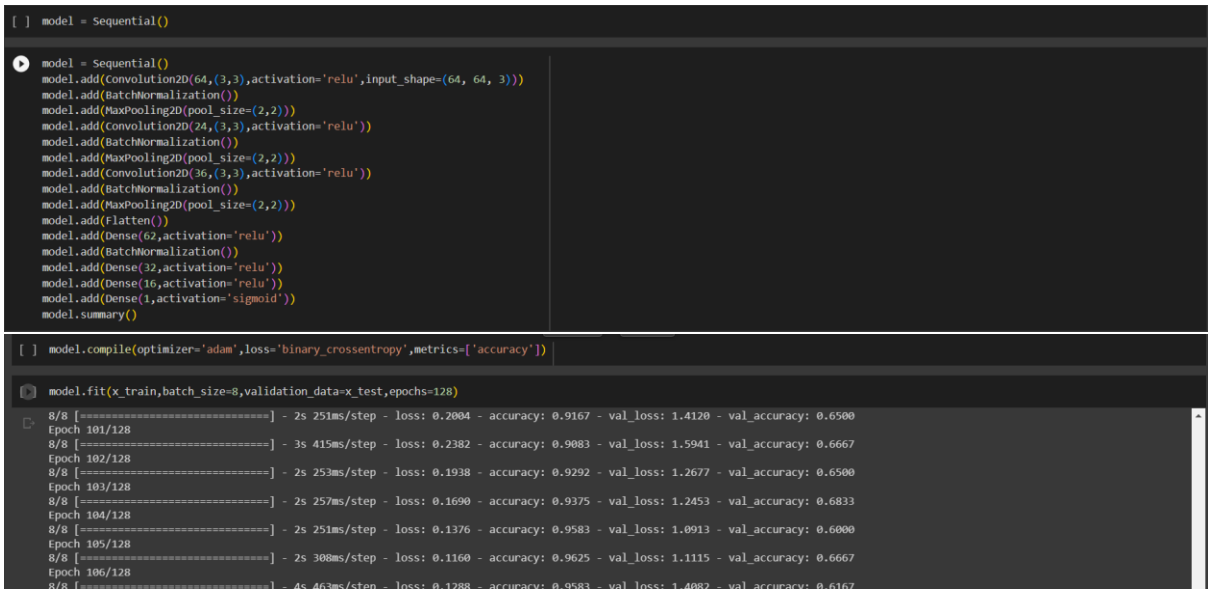
12. APPENDIX:

FindingMissingPerson.ipynb (Google Collab Notebook)



```
!unzip '/content/Dataset.zip'

Archive:  /content/Dataset.zip
   creating: Dataset/
   creating: Dataset/testset/
   creating: Dataset/testset/Found Missing/
  inflating: Dataset/testset/Found Missing/bill.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1124266363-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1124700570-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1151703682-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1151703714-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1158031528-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1158031539-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1170531211-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1174744292-612x612.jpg
  inflating: Dataset/testset/Found Missing/gettyimages-1174744336-612x612.jpg
```

```
[ ] from keras.preprocessing.image import ImageDataGenerator

[ ] train_data = ImageDataGenerator(rescale = 1./255, shear_range=0.2, rotation_range=180,horizontal_flip=True,zoom_range=0.2)

[ ] test_data = ImageDataGenerator(rescale = 1./255)

[ ] x_train = train_data.flow_from_directory('/content/Dataset/trainset', target_size = (64,64), batch_size=32, class_mode='binary')

    Found 240 images belonging to 2 classes.

[ ] x_test = train_data.flow_from_directory('/content/Dataset/testset', target_size = (64,64), batch_size=32, class_mode='binary')

    Found 60 images belonging to 2 classes.

[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Convolution2D, MaxPooling2D, Flatten, BatchNormalization, Dropout
```



```
[ ] model = Sequential()

    model = Sequential()
    model.add(Convolution2D(64,(3,3),activation='relu',input_shape=(64, 64, 3)))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Convolution2D(24,(3,3),activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Convolution2D(36,(3,3),activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Flatten())
    model.add(Dense(62,activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(32,activation='relu'))
    model.add(Dense(16,activation='relu'))
    model.add(Dense(1,activation='sigmoid'))
    model.summary()
```

```
[ ] model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

    model.fit(x_train,batch_size=8,validation_data=x_test,epochs=128)

    8/8 [==============================] - 2s 251ms/step - loss: 0.2004 - accuracy: 0.9167 - val_loss: 1.4120 - val_accuracy: 0.6500
    Epoch 101/128
    8/8 [==============================] - 3s 415ms/step - loss: 0.2382 - accuracy: 0.9083 - val_loss: 1.5941 - val_accuracy: 0.6667
    Epoch 102/128
    8/8 [==============================] - 2s 253ms/step - loss: 0.1938 - accuracy: 0.9292 - val_loss: 1.2677 - val_accuracy: 0.6500
    Epoch 103/128
    8/8 [==============================] - 2s 257ms/step - loss: 0.1690 - accuracy: 0.9375 - val_loss: 1.2453 - val_accuracy: 0.6833
    Epoch 104/128
    8/8 [==============================] - 2s 251ms/step - loss: 0.1376 - accuracy: 0.9583 - val_loss: 1.0913 - val_accuracy: 0.6000
    Epoch 105/128
    8/8 [==============================] - 2s 308ms/step - loss: 0.1160 - accuracy: 0.9625 - val_loss: 1.1115 - val_accuracy: 0.6667
    Epoch 106/128
    8/8 [==============================] - 4s 463ms/step - loss: 0.1288 - accuracy: 0.9583 - val_loss: 1.4087 - val_accuracy: 0.6167
```

```
[ ] from tensorflow.keras.layers import Dense,Flatten,Input
    from tensorflow.keras.models import Model
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
    from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
    import numpy as np
```

```
[ ] vgg = VGG16(include_top=False,weights='imagenet',input_shape=(64,64,3))
```

```
[ ] for layer in vgg.layers:
      layer.trainable=False
```

```
[ ] x = Flatten()(vgg.output)
```

```
[ ] prediction = Dense(1,activation='sigmoid')(x)
```

```
[ ] model = Model(inputs=vgg.input,outputs=prediction)
```

```
[ ] model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
[ ] model.fit(x_train,validation_data=x_test,epochs=128,steps_per_epoch=len(x_train),validation_steps=len(x_test))
    8/8 [==============================] - 2s 246ms/step - loss: 0.3895 - accuracy: 0.8250 - val_loss: 0.6793 - val_accuracy: 0.7333
    Epoch 101/128
    8/8 [==============================] - 2s 250ms/step - loss: 0.3722 - accuracy: 0.8333 - val_loss: 0.6061 - val_accuracy: 0.7000
    Epoch 102/128
    8/8 [==============================] - 2s 246ms/step - loss: 0.4182 - accuracy: 0.8292 - val_loss: 0.7298 - val_accuracy: 0.6667
    Epoch 103/128
    8/8 [==============================] - 3s 341ms/step - loss: 0.3880 - accuracy: 0.8292 - val_loss: 0.7265 - val_accuracy: 0.6833
    Epoch 104/128
    8/8 [==============================] - 3s 339ms/step - loss: 0.3856 - accuracy: 0.8500 - val_loss: 0.7784 - val_accuracy: 0.6500
    Epoch 105/128
    8/8 [==============================] - 2s 254ms/step - loss: 0.3940 - accuracy: 0.8292 - val_loss: 0.7009 - val_accuracy: 0.6833
    Epoch 106/128
    8/8 [==============================] - 2s 249ms/step - loss: 0.3864 - accuracy: 0.8292 - val_loss: 0.6836 - val_accuracy: 0.7167
    Epoch 107/128
    8/8 [==============================] - 2s 250ms/step - loss: 0.3610 - accuracy: 0.8500 - val_loss: 0.7390 - val_accuracy: 0.6167
```

```
[ ] model.save('Missing_1.h5')
```

```
[ ] classes = x_train.class_indices
    classes

    {'Found Missing': 0, 'Normal': 1}
```

```
[ ] import cv2
    import numpy as np
    import smtplib
    from tensorflow.keras.preprocessing import image
    import tensorflow as tf
```

```
[ ] model_load = tf.keras.models.load_model('/content/model.h5')
```

```
[ ] img = image.load_img('/content/check.jpeg', target_size = (64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
```

```
[ ] pred = (model.predict(x) > 0.5).astype('int32')
    pred

    1/1 [==============================] - 0s 41ms/step
    array([[0]], dtype=int32)
```

```
import cv2
import numpy as np
from keras.preprocessing import image
import tensorflow as tf
model = tf.keras.models.load_model('model.h5')
video = cv2.VideoCapture(0)
name = ["Found Missing","Normal"]
count = 0
while(True):
    success, frame = video.read()
    cv2.imwrite("image%d.jpg" % count,frame)
    img = image.load_img("image%d.jpg" % count,target_size = (64,64))
    count += 1
    x  = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)
    p = (model.predict(x) > 0.5).astype('int32')
    print(p)
    cv2.putText(frame, "Predicted  Class = "+str(name[p]), (100,100),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)

    if p[0][0]==0:
        print("Found Missing")
    else:
        print("Normal")

    cv2.imshow("frame",frame)

    if cv2.waitKey(1) & 0xFF == ord('a'):
        break

video.release()
cv2.destroyAllWindows()
```

app.py (Flask endpoint)

```
from flask import Flask, render_template, Response
import os
import cv2
import numpy as np
from keras.preprocessing import image
import tensorflow as tf

app = Flask(__name__)

def video_gen():
    global camera
    camera = cv2.VideoCapture(0)
    model_path = os.path.join(os.path.dirname(os.path.abspath(__file__)),
'model.h5')
    model = tf.keras.models.load_model(model_path, compile=False)
    name = ["Found Missing","Normal"]
    while True:
        success, frame = camera.read()
        re_frame = cv2.resize(frame, (64,64))
        x  = image.img_to_array(re_frame)
        x = np.expand_dims(x,axis = 0)
        p = (model.predict(x) > 0.5).astype('int32')
        cv2.putText(frame, "Predicted  Class = "+str(name[p[0][0]]),
(100,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)

        ret,buffer=cv2.imencode('.jpg',frame)
        frame=buffer.tobytes()
        yield(b'--frame\r\n'
              b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n'
              b'<h2>Prediction: ' + name[p[0][0]].encode() + b'</h2>\r\n')

@app.route('/search', methods=['POST'])
def search():
    return render_template('search.html')

@app.route('/video_feed')
def video_feed():
    return Response(video_gen(), mimetype="multipart/x-mixed-replace;
boundary=frame")

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run()
```

index.html (Main homepage)

```
<!DOCTYPE html>
<html>

<head>
    <title>Find missing person</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: Arial, sans-serif;
            color: #000;
        }
        .hero {
            background-color: #B0E0E6;
            text-align: center;
            height: 100vh;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
        }
        .hero h1 {
            font-size: 36px;
            color: #000;
        }
        .hero p {
            font-size: 18px;
            color: #000;
            margin-bottom: 20px;
        }
        .hero button {
            background-color: #000;
            color: #fff;
            border: none;
            padding: 10px 20px;
            margin: 10px;
            cursor: pointer;
            font-size: 16px;
            border-radius: 10px;
        }
        .hero .button1 {
            background-color: #000;
        }
        .hero .button2 {
            background-color: #00008B;
```

```
        }
        .about {
            background-color: #fff;
            text-align: center;
            padding: 100px 0;
        }
        .about p {
            font-size: 20px;
            color: #000;
        }
        .form {
            background-color: #B0E0E6;
            text-align: center;
            padding: 100px 0;
        }
        .form input[type="email"] {
            padding: 10px;
            font-size: 16px;
            border: none;
            border-radius: 10px;
        }
        .form button {
            background-color: #000;
            color: #fff;
            border: none;
            padding: 10px 20px;
            margin-top: 10px;
            cursor: pointer;
            font-size: 16px;
            border-radius: 10px;
        }
    </style>
</head>

<body>
    <section class="hero">
        <h1>Find Missing Persons, Powered by AI</h1>
        <button onclick="scrollToAbout()">About</button>
        <button onclick="scrollToForm()">Search</button>
    </section>
    <section class="about" id="about">
        <p>Our AI-powered solution takes the piled-up data of missing person cases and brings clarity by swiftly
            narrowing the search. It helps law enforcement agencies and search parties in their mission to locate and
            bring the lost ones home.</p>
    </section>
    <section class="form" id="form">
```

```
        <form action="/search" method="post">
            <input type="email" placeholder="Enter email">
            <button>Search Now</button>
        </form>
    </section>
    <script>
        function scrollToAbout() {
            document.getElementById('about').scrollIntoView({ behavior:
'smooth' });
        }
        function scrollToForm() {
            document.getElementById('form').scrollIntoView({ behavior:
'smooth' });
        }
    </script>
</body>

</html>
```

search.html (Model page)

```
<!DOCTYPE html>
<html>

<head>
    <title>
        Search
    </title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: Arial, sans-serif;
            color: #000;
        }
        .search {
            background-color: #B0E0E6;
            text-align: center;
            height: 100vh;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
        }
        .search h1 {
            font-size: 36px;
            color: #000;
        }
```

```
        .search img {
            border-radius: 10px;
        }
    </style>
</head>

<body>
    <section class="search">
        <center>
            <h1 align="center">
                Searching from live video feed
            </h1>
            <img src="{{url_for('video_feed')}}">
        </center>
    </section>
</body>

</html>
```