EC3712 Product Development Laboratory

Name Of the Project – Suspicious and Criminal Activity Detection

Group Leader – Satya Ranjan Dutta

Group – 8

Deep Learning Model - Satya Ranjan Dutta
Deep Learning Model - Aditya Milland Kukarni

back End - Siddhant Hota
Front End - Manasi Gharai
Front End - Brijesh Khatua

Introduction –

Human behavior recognition in the real-world environment finds plenty of applications including intelligent video surveillance, shopping behavior analysis. Video surveillance has vast application areas especially for indoor outdoor and places. Surveillance is an integral part of security. Today security camera becomes part of life for the safety and security purposes. Today, manual monitoring of all the events on the CCTV (Closed Circuit Television) camera is impossible. Even if the event had already happened, searching manually the same event in the recorded video wastes a lot of time. Analyzing abnormal events from video is an emerging topic in the domain of automated video surveillance systems. Human behavior detection in video surveillance system is an automated way of intelligently detecting any suspicious activity. Number of efficient algorithms is available for the automatic detection of human behavior in public areas like airports, railway stations, banks, offices, examination halls etc Video surveillance is the emerging area in the application of Artificial Intelligence, Machine Learning and Deep Learning. Artificial intelligence helps the computer to think like human. In machine learning, important components are learning from the training data and make prediction on future data. Nowadays GPU (Graphics Processing Unit) processors and huge datasets are available, so the concept of deep learning is used. Deep Neural Networks is one of the best architectures used to perform difficult learning tasks. Deep Learning models automatically extract features and builds high level representation of image data. This is more generic because the process of feature extraction is fully automated. From the image pixels, convolutional neural network (CNN) can learn visual patterns directly. In the case of video stream, long short-term memory (LSTM) models are capable of learning long term dependencies. LSTM network has the ability to remember things. The proposed system will use footage obtained from CCTV camera for monitoring the human behavior in a campus and gently warn when any suspicious event occurs. The major components in intelligent video monitoring are event detection and human behavior recognition. The entire process of training a surveillance system can be summarized in to three phases: data preparation, training the model and inference

Objective –

This system camera will detect the Suspicious and criminal activities than it will give an alert (Siren and Red light) to the person so also user able to know it in his phone by alert notification (through website/app).

Outcome of the project –

This alert system will improve the safety of the locality by monitoring and preventing various Suspicious and criminal activities.

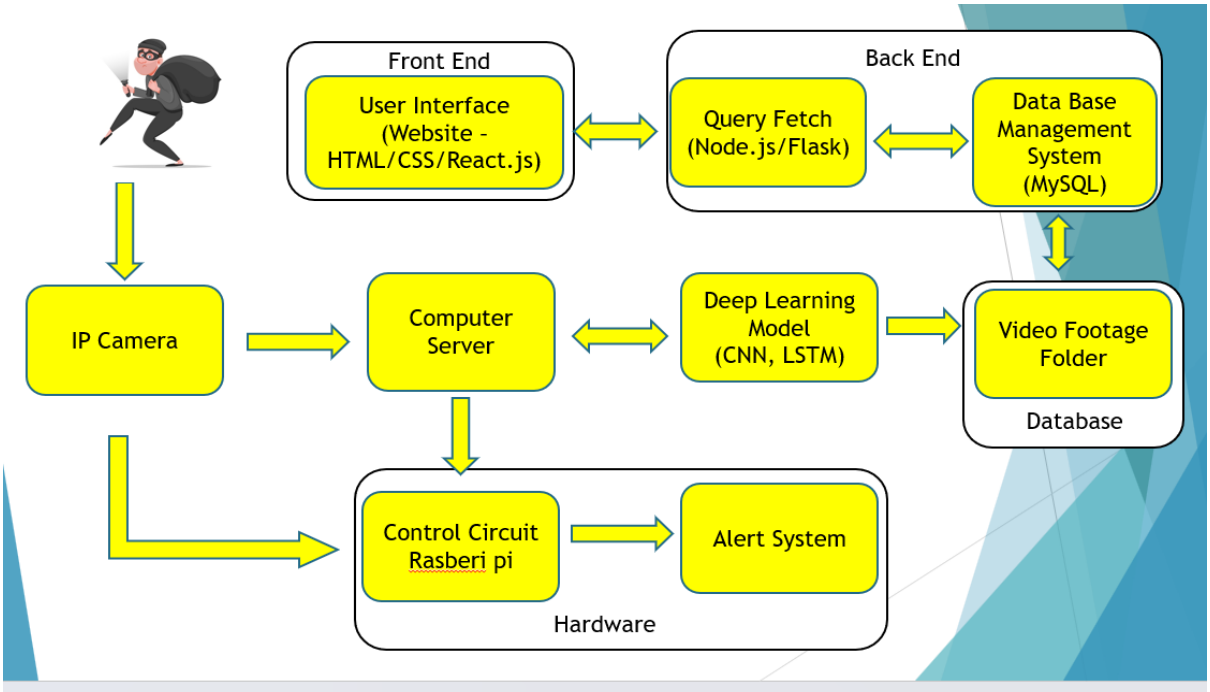
Features of the Project –

1. user friendly (Both in software and hardware)
2. Low Cost
3. easy to Implement
4. Highly reliable (provide App/Web)
5. better detection and alert system (Sound and light)

Basic principle Behind the Project –

- The IP camera is installed at the desired location along with the monitoring system, Red light and Alarm for alert system.
- The system is already trained by the various dataset of Suspicious and criminal activities from the data base with high efficiency.
- When any type of activity is detected by the camera it will predict it is a Suspicious and criminal activity or not, if it is then it will be signaled to the backend server which will run the front end so that user can see alert message in provided app or web.
- Along with this the alert system will also activate automatically providing alarm sound and red light which will help to prevent the crime to be escalate.
- All this will happen in few of seconds after the detection of activity.

Project Block Diagram -



Computer Server:

It will receive live footage from the IP camera and process it using the deep learning model. When a suspicious activity is detected, the video clip is sent to be stored in a According to the folder (different folder for different suuspicous activity) known as DATA BASE which is will ready for the user to show in the User Inter face (While query is fetched).

Deep Learning Model:

It is the most Important part of the product to detect the suspicious Activity.

language – We are using the Python Language as it already contains most of the librarie szand

mostly used for Machine Learning and Deep Learning.

IDE To be used – We are using Jupyter Notebook and Google Colab to write and Debug the code

as here we can excute the line by line code of the python program.

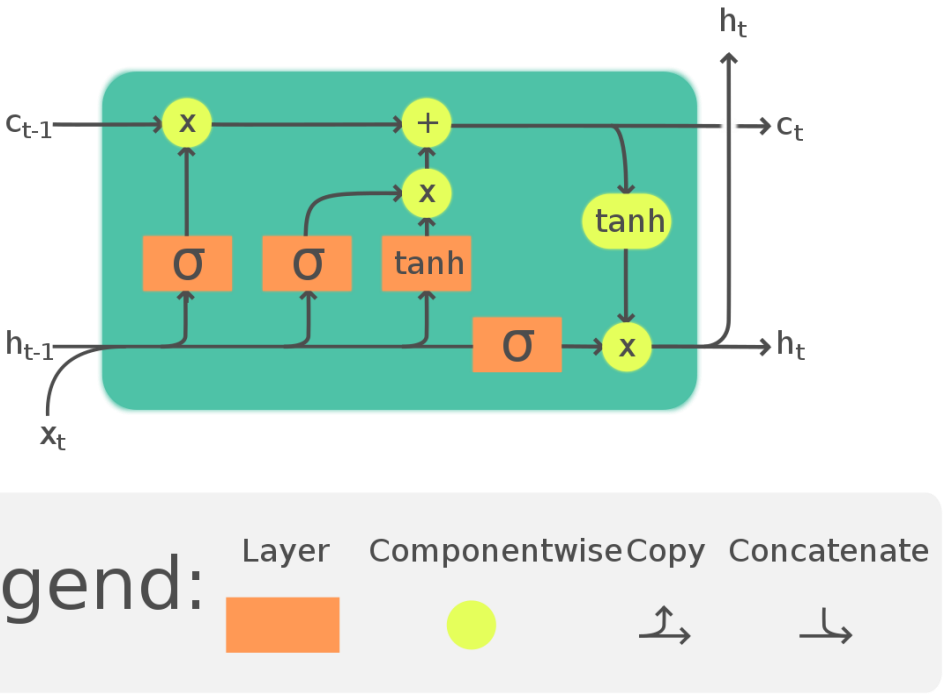
In google Colab we already get all the Libraies pre Installed and GPU and TPU server in free.

Model To Be Used –

we are using **LSTM and CNN** as both are good for real Video Processing and Continuous data flow to train the model. As in our model simultaniously video will be detected as to the model so it will give the better accuracy.

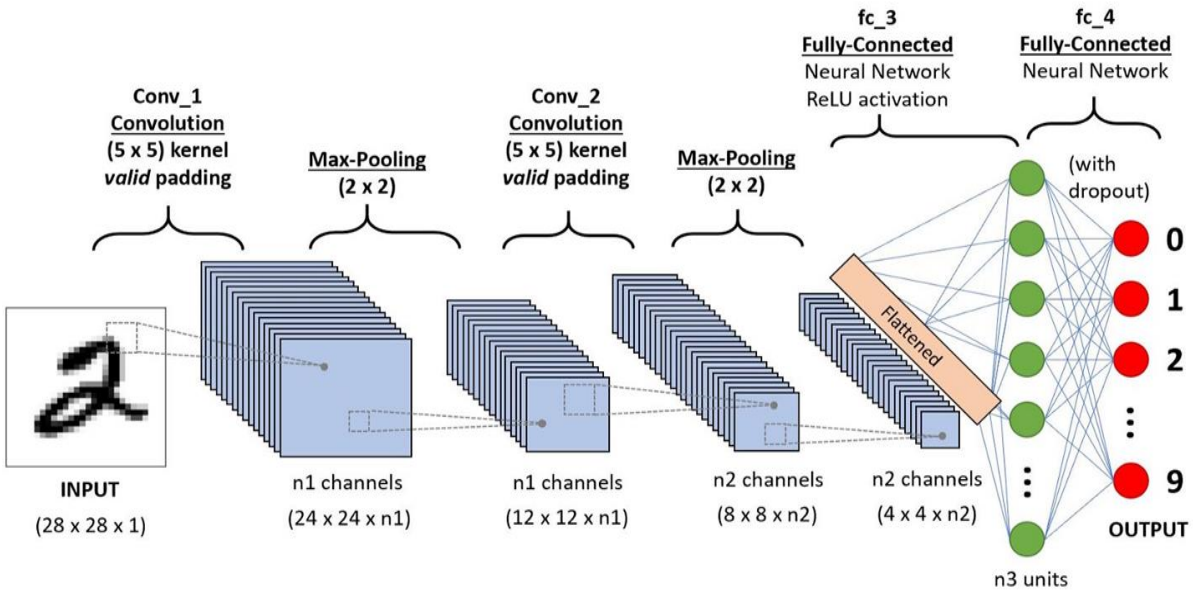
LSTM – Long Short Term Memory -

It is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). This characteristic makes LSTM networks ideal for processing and predicting data



CNN –

A convolutional neural network (CNN or convnet) is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data.



TensorFlow, Keras, Numpy , Pandas, Matplot Lib, SKLearn –

To Implement the CNN and LSTM Model we are using Various Libraries.

Tensorflow and Keras – TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it's built-in Python.

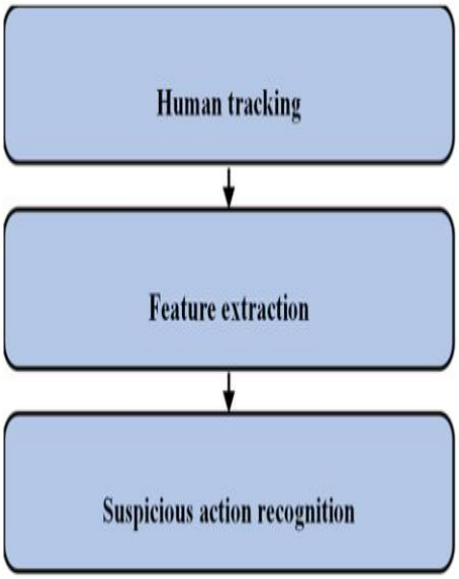


Figure 1. Overview of suspicious-action detection.

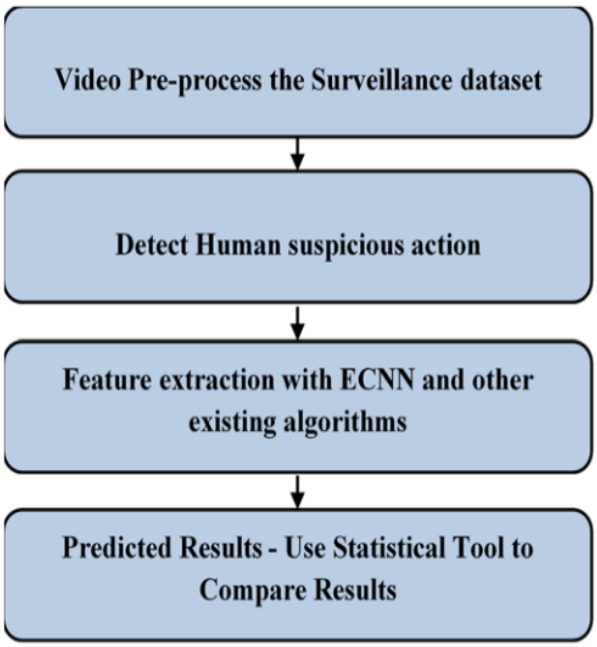
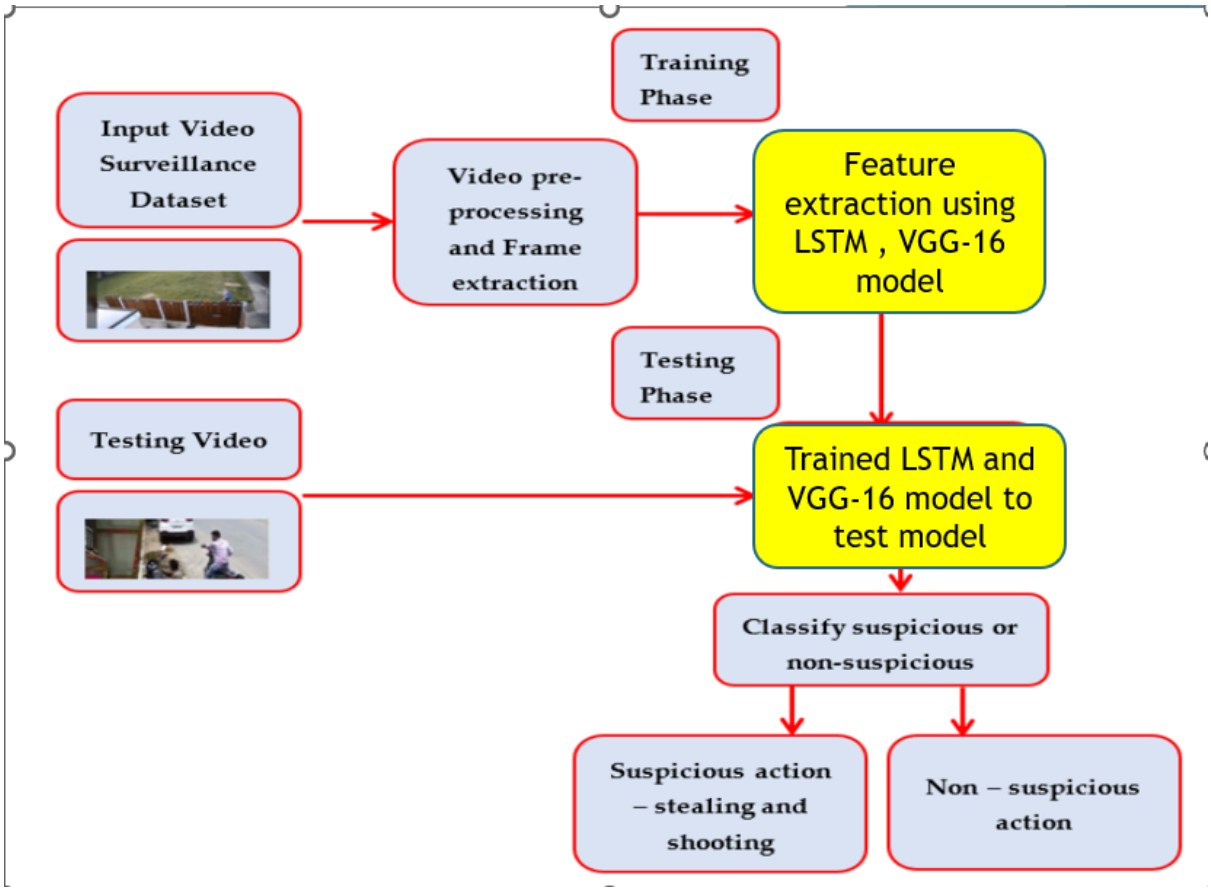


Figure 2. Suspicious-action (shooting-and-stealing) detection steps.

Suspicious and non-suspicious action with training and testing procedure



Code For The Project –

First we are importing the Important Libraries – ,

Open CV , Math , Numpy, Pandas, Mathplotlib, Seaborn, Sklearn, TensorFlow, Keras, Collection, datetime, OS, cv,

```
import numpy as np
import os
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten, LSTM, TimeDistributed, Dropout
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import cv2
import os
import datetime
import time
```

Defining the parameters –

Frame size, batch size, number of epochs, data directory, carogories of the data cllasifacation.

```
# Define the size of the frames
frame_size = (224, 224)

# Define the number of frames per sequence
frames_per_sequence = 16

# Define the batch size
batch_size = 32

# Define the number of epochs
# num_epochs = 10
num_epochs = 20


# Define the path to the video dataset directory
data_dir = "Dataset"

# Define the list of categories
# categories = ["Fighting","Kidnap"]
categories = ["Fighting","Kidnap", "Murder", "Robbery", "Shoplifting"]
```

Now creating the Deep Learning Model using LSTM and VGG 16 –

```
# Load VGG16 model without the top layers
vgg16 = VGG16(weights='imagenet', include_top=False, input_shape=(frame_size[0], frame_size[1], 3))

# Freeze the layers
for layer in vgg16.layers:
    layer.trainable = False

# Define the LSTM model
model = Sequential()
model.add(TimeDistributed(vgg16, input_shape=(frames_per_sequence, frame_size[0], frame_size[1], 3)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(256, return_sequences=False))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dense(len(categories), activation='softmax'))

# Compile the model
optimizer = Adam(learning_rate=1e-4, decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

VGG-16 –

VGG-16 is **a convolutional neural network that is 16 layers deep**. You can load a pretrained version of the network trained on more than a million images from the ImageNet database.

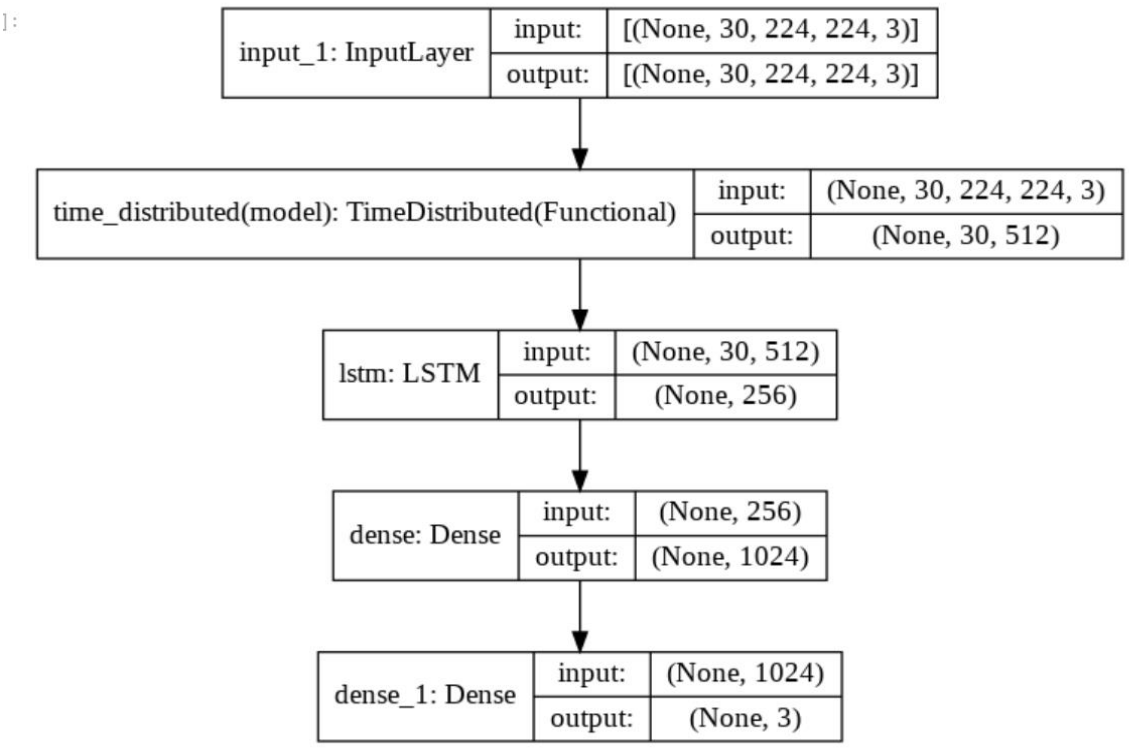
- **include_top**: whether to include the 3 fully-connected layers at the top of the network.
- **weights**: one of None (random initialization), 'imagenet' (pre-training on ImageNet), or the path to the weights file to be loaded.
- **input_tensor**: optional Keras tensor (i.e. output of layers.Input()) to use as image input for the model.
- **input_shape**: optional shape tuple, only to be specified if include_top is False (otherwise the input shape has to be (224, 224,
  3) (with channels_last data format) or (3, 224,
  224) (with channels_first data format). It should have exactly 3 input channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value.
- **pooling**: Optional pooling mode for feature extraction
  when include_top is False. - None means that the output of the model will be the 4D tensor output of the last convolutional block. - avg means

- that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor. - max means that global max pooling will be applied.
- **classifier_activation**: A str or callable. The activation function to use on the "top" layer. Ignored unless include_top=True.
  Set classifier_activation=None to return the logits of the "top" layer.
  When loading pretrained weights, classifier_activation can only
  be None or "softmax".

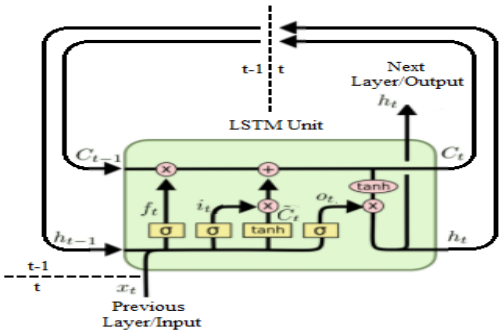
LSTM –

Long short-term memory (LSTM) is **an artificial neural network used in the fields of artificial intelligence and deep learning**. Unlike standard feedforward neural networks, LSTM has feedback connections.

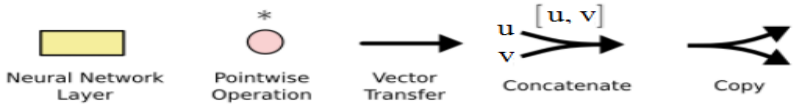
Tree architecture using in LSTM –



Understanding LSTM Networks



$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}
$$




```
data = []
labels = []

for category in categories:
    path = os.path.join(data_dir, category)
    label = categories.index(category)

    for filename in os.listdir(path):
        filepath = os.path.join(path, filename)

        # Initialize a list to store the frames
        frames = []

        # Open the video file
        cap = cv2.VideoCapture(filepath)

        # Loop through the frames and add them to the list
        while True:
            ret, frame = cap.read()
            if not ret:
                break

            # Resize the frame
            frame = cv2.resize(frame, frame_size)

            # Convert the frame to an array and normalize the pixel values
            frame = img_to_array(frame)
            frame = frame.astype('float32') / 255.0

            frames.append(frame)

        # Release the video file
        cap.release()

        # If the number of frames is less than the required number, pad the list with zeros
        if len(frames) < frames_per_sequence:
            frames.extend([np.zeros(frame_size) for _ in range(frames_per_sequence - len(frames))])

        # Keep only the required number of frames
        frames = frames[:frames_per_sequence]

        # Add the list of frames to the data list and the corresponding label to the labels list
        data.append(frames)
        labels.append(label)
```

In The above code snippet Extracting frame by frame data from the video file –

- first going to the data directory
- capturing the video frame
- resizing the video frames and scaling them by dividing by 255
- appending the video frmae to the numpy array for future use
- creating data set data and lebel
- categorical encoding
- train testsplit
-

Now converting to catogorincal levels and making train test split -

```
# Convert the data and labels lists to arrays
data = np.array(data)
labels = np.array(labels)

# Convert the labels to categorical format
labels = to_categorical(labels, num_classes=len(categories))

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
```

Now fitting the model on data set -

```
history = model.fit(X_train, y_train, batch_size=batch_size, epochs=num_epochs, validation_data=(X_test, y_test))
```

```
Epoch 1/20
```

```
loss, accuracy = model.evaluate(X_test, y_test)

print("Test Loss: ", loss)
print("Test Accuracy: ", accuracy)
```

```
1/1 [==============================] - 135s 135s/step - loss: 0.4593 - accuracy: 0.7586
Test Loss:  0.45931559801101685
Test Accuracy:  0.7586206793785095
```

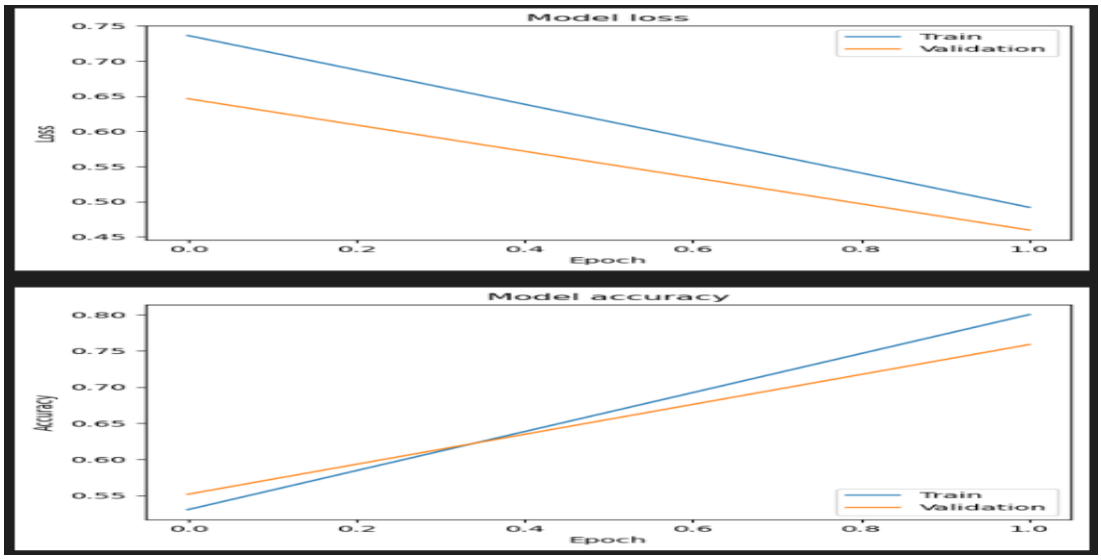
Verifying test loss and test acuuracy

Now saving the model and ploting model lossand model accuracy.

```
model.save("f1_model.h5")
```

```
import matplotlib.pyplot as plt


# Plot the training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()

# Plot the training and validation accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')
plt.show()
```

Graphs -



Now creating predict catogry function to predict the o/p from the input video and creating a output folder and storing the output in text document.

```
def predict_category(video_path):
    # Load the video file and extract frames
    cap = cv2.VideoCapture(video_path)
    frames = []
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv2.resize(frame, (224, 224))
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) # Convert to RGB
        frame = frame.astype('float32') / 255.0 # Normalize pixel values
        frames.append(frame)
    cap.release()

    # Pad or truncate frames to desired length
    frames_per_sequence = 16
    if len(frames) < frames_per_sequence:
        frames.extend([np.zeros((224, 224, 3)) for _ in range(frames_per_sequence - len(frames))])
    else:
        frames = frames[:frames_per_sequence]

    # Convert frames to numpy array
    frames = np.array(frames)

    # Add batch dimension and change number of channels to 3
    frames = np.expand_dims(frames, axis=0)
    frames = np.expand_dims(frames,  axis=-1)

    # Make prediction
    predicted_probs = model.predict(frames)

    # Get the index of the category with the highest probability
    pred_idx = np.argmax(predicted_probs)
    predicted_category = categories[np.argmax(predicted_probs)]
```

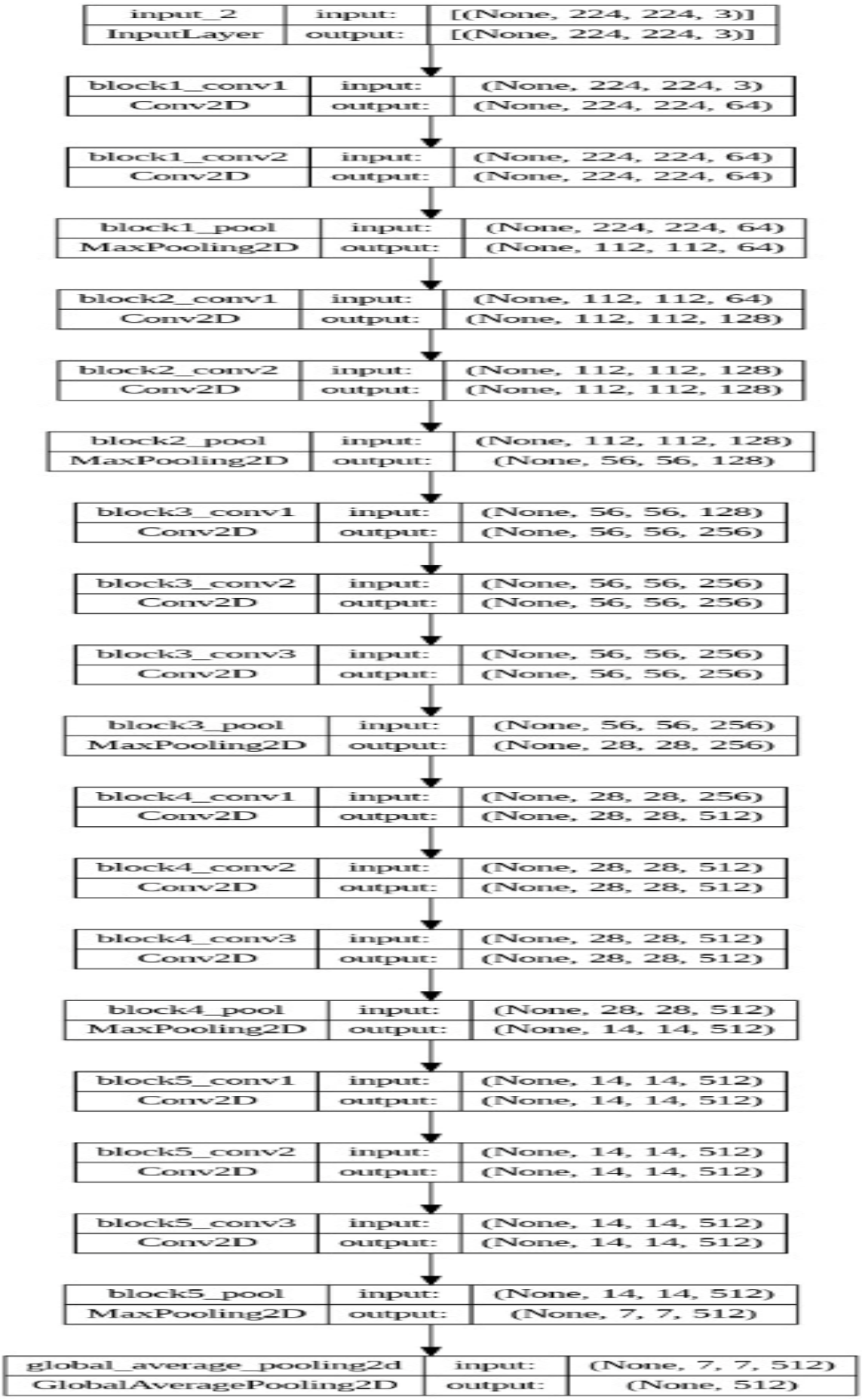
Calling predict function using the sample input video-

```
    # Get the corresponding probability
    prob = predicted_probs[0][pred_idx]

    # Print the predicted category with the corresponding probability
    print(f"Predicted category: {predicted_category}")
    print(f"Probability: {prob:.2f}")


    output_directory_path = 'output_prediction'
    predicted_output_directory_path = os.path.join(output_directory_path, predicted_category)

    # Create the directory if it doesn't exist.
    if not os.path.exists(predicted_output_directory_path):
        os.makedirs(predicted_output_directory_path)

    timestamp = time.strftime('%Y%m%d_%H%M%S')

    # Write the predicted action and confidence to the output file in the predicted output directory.
    output_file_path = os.path.join(predicted_output_directory_path, f'{predicted_category}_output_{timestamp}.txt')

    # Write the predicted action and confidence to a file in the output directory corresponding to the predicted class.
    # output_file_path = os.path.join("output_file", f'{predicted_class_name}.txt')


    # Get the current time
    now = datetime.datetime.now()
    # Format the timestamp in the desired format
    date_time_str = now.strftime("Date %Y-%m-%d, Time %H:%M:%S")


    # Write the predicted action and confidence to the output file.
    with open(output_file_path, 'w') as f:
        f.write(f'Action Predicted: {predicted_category}\nAccuracy: {prob:.2f}\n{date_time_str}')
```

```
predict_category("sql demo\v_Robbery_g001_c001.mp4")
```

```
1/1 [==============================] - 2s 2s/step
Predicted category: Kidnap
Probability: 0.58
```

VGG 16 tree -


input_2 | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
block1_conv1 | input: | (None, 224, 224, 3)
Conv2D | output: | (None, 224, 224, 64)
block1_conv2 | input: | (None, 224, 224, 64)
Conv2D | output: | (None, 224, 224, 64)
block1_pool | input: | (None, 224, 224, 64)
MaxPooling2D | output: | (None, 112, 112, 64)
block2_conv1 | input: | (None, 112, 112, 64)
Conv2D | output: | (None, 112, 112, 128)
block2_conv2 | input: | (None, 112, 112, 128)
Conv2D | output: | (None, 112, 112, 128)
block2_pool | input: | (None, 112, 112, 128)
MaxPooling2D | output: | (None, 56, 56, 128)
block3_conv1 | input: | (None, 56, 56, 128)
Conv2D | output: | (None, 56, 56, 256)
block3_conv2 | input: | (None, 56, 56, 256)
Conv2D | output: | (None, 56, 56, 256)
block3_conv3 | input: | (None, 56, 56, 256)
Conv2D | output: | (None, 56, 56, 256)
block3_pool | input: | (None, 56, 56, 256)
MaxPooling2D | output: | (None, 28, 28, 256)
block4_conv1 | input: | (None, 28, 28, 256)
Conv2D | output: | (None, 28, 28, 512)
block4_conv2 | input: | (None, 28, 28, 512)
Conv2D | output: | (None, 28, 28, 512)
block4_conv3 | input: | (None, 28, 28, 512)
Conv2D | output: | (None, 28, 28, 512)
block4_pool | input: | (None, 28, 28, 512)
MaxPooling2D | output: | (None, 14, 14, 512)
block5_conv1 | input: | (None, 14, 14, 512)
Conv2D | output: | (None, 14, 14, 512)
block5_conv2 | input: | (None, 14, 14, 512)
Conv2D | output: | (None, 14, 14, 512)
block5_conv3 | input: | (None, 14, 14, 512)
Conv2D | output: | (None, 14, 14, 512)
block5_pool | input: | (None, 14, 14, 512)
MaxPooling2D | output: | (None, 7, 7, 512)
global_average_pooling2d | input: | (None, 7, 7, 512)
GlobalAveragePooling2D | output: | (None, 512)

Front End –

Using Streamlit Python module to make Front end.

The Front end contains 5 main pages –

1. Login/ log out page
2. Home
3. Predict action
4. About
5. Contact

The web interface is synchronized with the MY Sql data base which stores-

1. User credentials for login/log out
2. New user creation
3. Storing output data and video file path time stamp

About page to download documentation -

```python
import streamlit as st
# import base64
# from pdf2image import convert_from_path

def docoments():
    # Add a link to a PDF file
    st.title(f"Download the docomentation :page_facing_up:")
    st.markdown("[Download PDF](https://example.com/example.pdf)")
```

Contact page / synchronized with the gmail –

```python
def contact_us():
    st.title(f"contact us here:e-mail:")
    # Define the recipient email address
    recipient = "srd2802@gmail.com"
    # Add a way to contact us directly
    st.write('If you encounter any issues with the form or need additional assistance, please email us at srd2802@gmail.com')

    # Construct the mailto: URL
    subject = "Contact Form Submission"
    body = "Please enter your message here."
    mailto_url = f"mailto:{recipient}?subject={subject}&body={body}"
    # Open the user's default email client
    #  import webbrowser
    # webbrowser.open(mailto_url)
```

The Predict category Page is using the trained h5 file for prediction it uses some data preprocessing and than predicted the output -

```python
# Define the predict_category function
def predict_category():
    # File upload
    video_file = st.file_uploader("Upload video", type=["mp4", "avi"])
    predicted = st.button("predict")
    st.video(video_file)
    if predicted:

        if video_file is not None:
            # save the file to a local directory
            with open("temp.mp4", "wb") as f:
                f.write(video_file.getbuffer())

            # open the video file
            cap = cv2.VideoCapture("temp.mp4")

            frames = []
            while True:
                ret, frame = cap.read()
                if not ret:
                    break
                frame = cv2.resize(frame, (224, 224))
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) # Convert to RGB
                frame = frame.astype('float32') / 255.0 # Normalize pixel values
                frames.append(frame)
            cap.release()

            # remove the temporary file
            os.remove("temp.mp4")

            # Pad or truncate frames to desired length
```

The home page contsins a basic in fo of the project –

```python
st.set_page_config(page_title="Activity Prediction", page_icon=":tada:")
def intro():
    # Define the page title and description
    st.title("Welcome to my Web App! :derelict_house_building:")

    # Add a header for the app description
    st.header("About the Web App")
    st.write("This web app is designed to help you detect the suspicious activity. It was created using Streamlit, a Python fra

    # Add a header for the model description
    st.header("About the Model")
    st.write("The model used in this web app is a activity detector. It was trained on kaggle data set and has achieved an accu

    # Add any additional content you'd like to include on the home page
```

The login page create a session state such that as long as the user is loged in he can access any features till log out .It also contain an Username password window , for new user ther is an sign up window to create new user id.

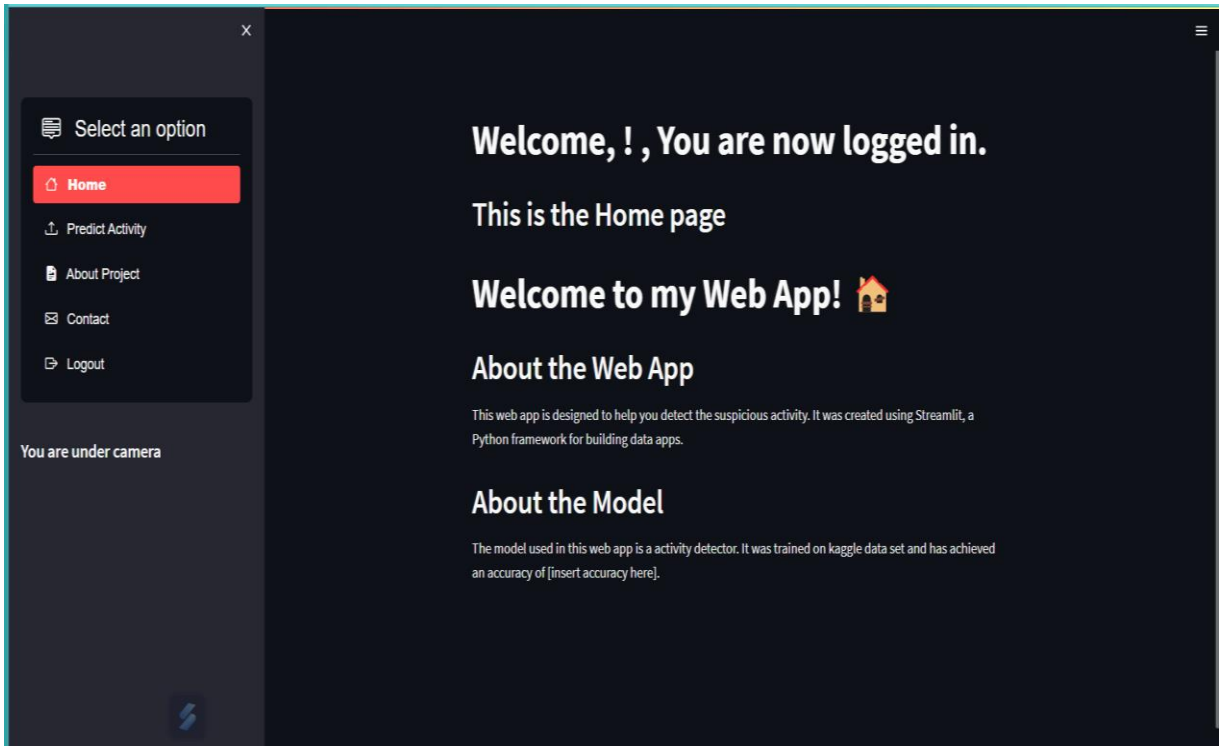
```python
# Define a login function
def login(session_state):
    st.header("Login")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")
    if st.button("Login"):
        # Check if user exists in the database
        cursor.execute("SELECT * FROM users WHERE username=%s AND password=%s", (username, password))
        result = cursor.fetchone()
        if result:
            session_state.logged_in = True
            session_state.user = username
            st.success("Logged in successfully.")
            trialerror.main()  # call main function after successful login
        else:
            st.error("Invalid username or password. Create new account by sign up")
            # Create a sign-up form
            st.write("## Sign Up")
            new_username = st.text_input("Enter a new username:")
            new_password = st.text_input("Enter a new password:", type="password")
            if st.button("Sign Up"):
                if create_user(new_username, new_password):
                    st.success("You have successfully signed up!")
                else:
                    st.error("Sorry, there was an error signing you up.")
```

All this pages are attached to the main function which runs the entire web server pages and activates the navbar having all features.Also it connects to the Local y sql data Base and the web interface creating local data base management system.

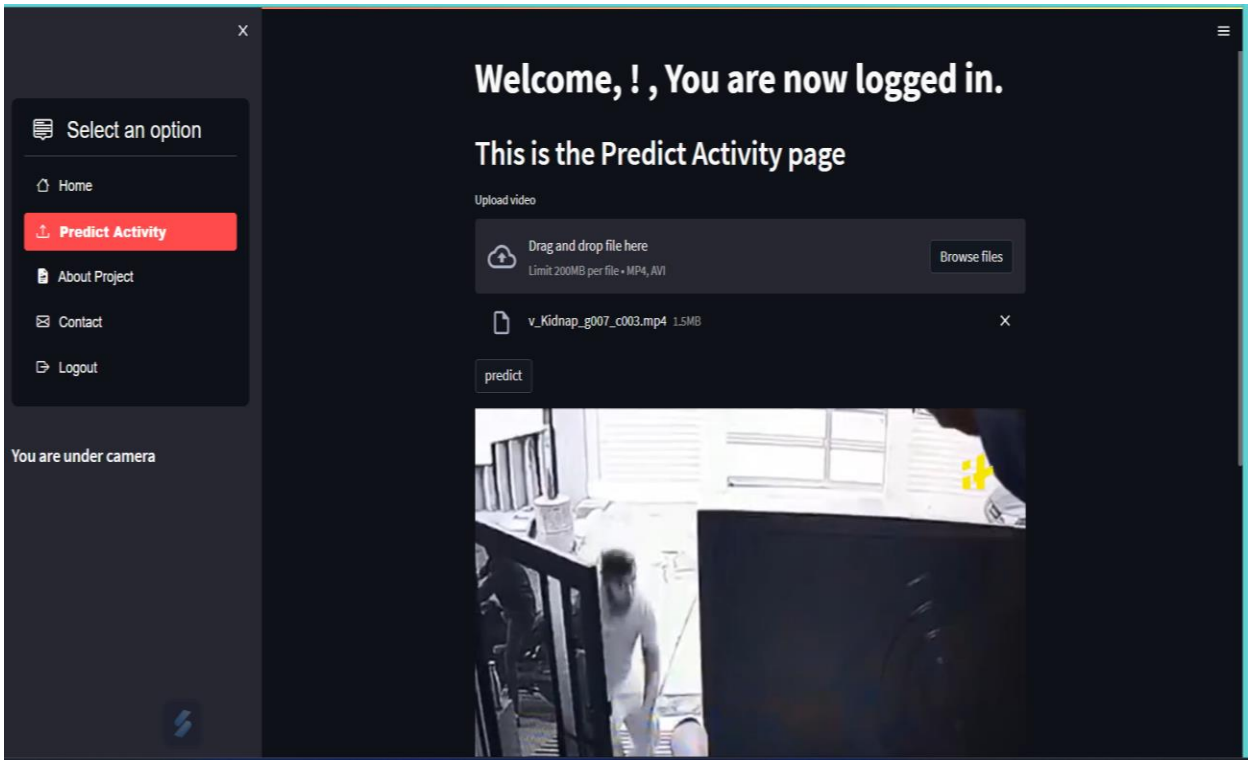
```python
# Connect to the database
db = mysql.connector.connect(
    (function) user: Any
    user="root",
    password="Satya@123",
    database="labproject"
)

# Create a cursor object to execute SQL queries
cursor = db.cursor()

# Define a query to fetch all data from the users table
query = "SELECT * FROM users"

# Execute the query and fetch the results
cursor.execute(query)
results = cursor.fetchall()
```

```python
def main():
    # Set up session state
    session_state = st.session_state
    if not hasattr(session_state, "logged_in"):
        session_state.logged_in = False
        session_state.user = ""

    # If not logged in, show login form
    if not session_state.logged_in:
        trialerror_signup.login(session_state)
    # If logged in, show the main app
    else:
        st.title(f"Welcome, {session_state.user}! , You are now logged in.")
        u = ["Home", "Predict Activity", "About Project", "Contact", "Logout"]

        with st.sidebar:
            choice = som.option_menu(
                "Select an option",
                u,
                icons=["house","upload","file-earmark-richtext-fill","envelope","box-arrow-right"],
            )
        with st.sidebar:
            st.header("You are under camera")
            st_lottie(lottie_anim,height=300,key="cameraanim")
        if choice == "Home":
            st.header("This is the Home page")
            trialerror_home.intro()
```

Home page –



Predict Page -



Conclusion –

we success fully completed the project with front end back end, deep learning model, database.

Links –

Git Hub - https://github.com/Srdcode/Criminal-Activity-detection

Data Set - https://drive.google.com/drive/folders/1clYNuSaCv8djyoiYYopeItCQbbVjTuD9?usp=share_link

On going topic – Working with IP Camera