Sprint 2

Date	29 October 2022
Team ID	PNT2022TMID25532
Project Name	Emerging Methods for Early Detection of Forest Fires

app.py

```
# Logout Page
@app.route('/logout')
def logout():
    session['name'] = None
    return redirect('/')


# About Page
@app.route('/about')
def about():
    if camera:
        camera.release()
    return render_template('about.html')


def detect(pred):
    li = ['Not Fire', 'Fire']
    return li[pred]


model = load_model(r'./static/model/ForestDetectionModel.h5')


def image_prediction(img_file):
    img = image.load_img(img_file, target_size=(150, 150))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    pred = model.predict(x)
    pred = detect(int(pred))
    return pred


def video_prediction(video_file):
    cap = cv2.VideoCapture(video_file)
    WIDTH = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    HEIGHT = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    FPS = 1
    fourcc = 0x00000021
    output = cv2.VideoWriter("./static/uploads/videos/output.mp4", fourcc,
FPS, (WIDTH, HEIGHT))
    def getFrame(sec):
        cap.set(cv2.CAP_PROP_POS_MSEC, sec * 1000)
        hasFrames, frame = cap.read()
        if hasFrames:
            cv2.imwrite('./static/uploads/images/pic.jpg', frame)
```

```
            img = image.load_img('./static/uploads/images/pic.jpg',
target_size=(150, 150))
            img = image.img_to_array(img)
            img = np.expand_dims(img, axis=0)
            pred = model.predict(img)
            if pred[0] == 1:
                cv2.circle(frame, (480, 55), 10, (0, 0, 255), -1)
                cv2.putText(frame, 'Fire Alert', (500, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2, cv2.LINE_AA)
            output.write(frame)
        return hasFrames
    sec = 0
    frameRate = 1
    count = 1
    success = getFrame(sec)
    while success:
        count = count + 1
        sec = sec + frameRate
        sec = round(sec, 2)
        success = getFrame(sec)
    output.release()
    cap.release()
    return 1


# Check the image and video extensions
def allowed_file(filename):
    image_extensions = {'jpg', 'jpeg', 'png', 'gif', 'webp'}
    video_extensions = {'webm', 'mp4', 'mov', 'avi', 'mkv'}
    ex = filename.rsplit('.', 1)[1].lower()
    if ex in image_extensions:
        return 'images'
    if ex in video_extensions:
        return 'videos'
    return None


# Predict Forest Fire using images and videos
@app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'POST':
        file = request.files['Pred_file']
        if not file:
            flash('No File Selected', 'image_video')
            return redirect('/')
        if not allowed_file(file.filename):
            flash('Invalid File Format', 'image_video')
            return redirect('/')
        if file and allowed_file(file.filename) == 'images':
            img_filename = secure_filename(file.filename)
            path = os.path.join(app.config['UPLOAD_FOLDER'], 'images',
img_filename)
            file.save(path)
            imageprediction = image_prediction(path)
            return render_template('home.html', img=img_filename,
img_prediction=imageprediction, link_image_video = True)
        if file and allowed_file(file.filename) == 'videos':
            vdo_filename = secure_filename(file.filename)
            path = os.path.join(app.config['UPLOAD_FOLDER'], 'videos',
vdo_filename)
            file.save(path)
```

```
            videoprediction = video_prediction(path)
            return render_template('home.html',
video_prediction=videoprediction, link_image_video = True)
    return redirect('/')



# Generate and predict using  each frames in realtime camera
def gen_frames():
    send = False
    while True:
        success, frame = camera.read()
        if success:
            try:
                cv2.imwrite('./static/uploads/videos/pic.jpg', frame)
                img = image.load_img('./static/uploads/videos/pic.jpg',
target_size=(150, 150))
                img = image.img_to_array(img)
                img = np.expand_dims(img, axis=0)
                pred = model.predict(img)
                if pred[0][0] == 1.0:
                    cv2.circle(frame, (480, 55), 10, (0, 0, 255), -1)
                    cv2.putText(frame, 'Fire Alert', (500, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2, cv2.LINE_AA)
                    mixer.init()
                    sound = mixer.Sound('./static/sound/alert1.ogg')
                    sound.play()
                    if not send:
                        alert_messge()
                        send = True
                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')
            except:
                pass


# render to cv camera to html file
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')


# Realtime camera page
@app.route('/camera_pred')
def camera_pred():
    global camera
    camera = cv2.VideoCapture(0)
    return render_template('/home.html', camera_start = True,
camera_prediction=True)


# camera stop method
@app.route('/camera_stop')
def camera_stop():
    camera.release()
    return render_template('/home.html', camera_start = False,
```

```
camera_prediction=True)


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

home.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Forest Fire Detection</title>
   <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename = 'css/home.css') }}">
   <link
href="https://fonts.googleapis.com/css2?family=Jost:wght@500&display=swap"
rel="stylesheet">
</head>
<body>
   <div id="header">
      <div class="title">
         <h1>Emerging Methods for Early Detection of Forest Fires</h1>
      </div>
      <div class="sections">
         <a href="{{url_for('home')}}">Image and video</a>
         <a href="{{url_for('camera_pred')}}">Camera</a>
         <a href="{{url_for('about')}}">About</a>
         <a href="{{url_for('logout')}}">Logout</a>
      </div>
   </div>
   {% if link_image_video %}
      <div id="imgpred">
         <h2>Image And Video Detection</h2><br>
         {% with msg =
get_flashed_messages(category_filter=['image_video']) %}
         {% if msg %}
         <p>{{ msg }}</p><br>
         {% endif %}
         {% endwith %}
         <form action="/predict" method="POST" enctype="multipart/form-
data">
               <input type="file" id="images" name="Pred_file" required>
               <button type="submit">Predict</button>
         </form>
      </div>
      {% if img_prediction and img %}
      <div class="img_frame">
         <h3>{{img_prediction}} Detected</h3>
         <img src="/static/uploads/images/{{img}}">
      </div>
      {% endif %}
      {% if video_prediction %}
      <div class="video_frame">
         <video height="500" width="700" autoplay="" muted="">
            <source src="../static/uploads/videos/output.mp4"
type="video/mp4">
         </video>
      </div>
      {% endif %}
```

```
   {% endif %}
   {% if camera_prediction %}
      <div>
         <h1 style="text-align:center;">RealTime Camera Detection</h1>
      </div>
      <div class="camera_actions">
         <form action="/camera_pred" method="GET">
            <button type="submit">Start Camera</button>
         </form>
         <form action="/camera_stop" method="GET">
            <button type="submit">Stop Camera</button>
         </form>
      </div>
      {% if camera_start %}
      <div class="camera_container">
         <div class="row">
            <div>
               <h3>Live Camera Forest Fire Detection</h3>
               <img src="{{ url_for('video_feed') }}" width="700px"
height="400px">
            </div>
         </div>
      </div>
      {% endif %}
   {% endif %}
   <div id="content">
      <p>
         Forest fires are a major environmental issue, creating economic
and ecological damage
         while endangering human lives. There are typically about 100,000
wildfires in the United
         States every year. Over 9 million acres of land have been
destroyed due to treacherous
         wildfires. It is difficult to predict and detect Forest Fire in a
sparsely populated forest
         area and it is more difficult if the prediction is done using
ground-based methods like
         Camera or Video-Based approach. Satellites can be an important
source of data prior to
         and also during the Fire due to its reliability and efficiency.
The various real-time
         forest fire detection and prediction approaches, with the goal of
informing the local
         fire authorities.
      </p>
   </div>
</body>
</html>
```

home.css

```
body{
   margin: 0;
   padding: 0;
   min-height: 100vh;
   font-family: 'Jost', sans-serif;
   color: #fff;
```

```
   background: linear-gradient(to bottom, #0f0c29, #302b63, #24243e);
}
#header{
    display:flex;
    position: relative;
    flex-direction: row;
    margin : 0.5em;
}
#header .title{
    text-align: left;
}
#header h1{
    font-size: 1.5em;
}

#header .sections{
    position: absolute;
    top:1em;
    right:0;
    font-size: 1em;
}
#header .sections a{
    margin-right : 1.5em;
    text-decoration: none;
    color: #fff;
}

#content p{
    text-align:center;
    font-size: 1em;
    text-align: justify;
    margin: 0.3em;
    margin-left:3em;
    margin-right:3em;
    padding: 0.5em;
    position: relative;
    bottom: -3em;
    line-height: 2;
}

input[type=file] {
  width: 350px;
  max-width: 100%;
  color: #444;
  padding: 5px;
  background: #fff;
  border-radius: 5px;
  border: 1px solid #555;
}
#imgpred{
    display:flex;
    flex-direction: column;
    margin:2em;
    align-items: center;
    justify-content: center;

}
input[type=file]::file-selector-button {
  margin-right: 20px;
  border: none;
  background: #573b8a;
```

```
  padding: 10px 20px;
  border-radius: 5px;
  color: #fff;
  cursor: pointer;
  transition: background .2s ease-in-out;
}

input[type=file]::file-selector-button:hover {
  background: #0d45a5;
}


button{
   width: 15em;
   height: 40px;
   margin: 10px auto;
   justify-content: center;
   display: block;
   color: #fff;
   background: #573b8a;
   font-size: 1em;
   font-weight: bold;
   margin-top: 20px;
   outline: none;
   border: none;
   border-radius: 5px;
   transition: .2s ease-in;
   cursor: pointer;
}
button:hover{
   background: #6d44b8;
}

.img_frame img, .video_frame video, .camera_container{
    display: block;
    margin-top:1em;
    margin-bottom: 1em;
    margin-left: auto;
    margin-right: auto;
    width:50%;
}
.img_frame h3, .camera_container h3{
    text-align: center;
}

.camera_actions{
    display: flex;
    justify-content:center;
    align-items:center;
}
.camera_actions button{
    margin-right:3em;
    marin-left: 3em;
}
```