Data Collection and Preprocessing Phase

Date	15 JUNE 2025
Team ID	xxxxxx
Project Title	CRIME VISION : Advanced Crime Classifaction Learning
Maximum Marks	6 Marks

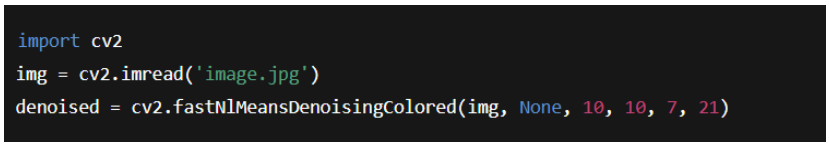
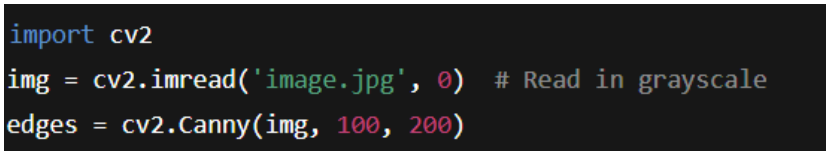
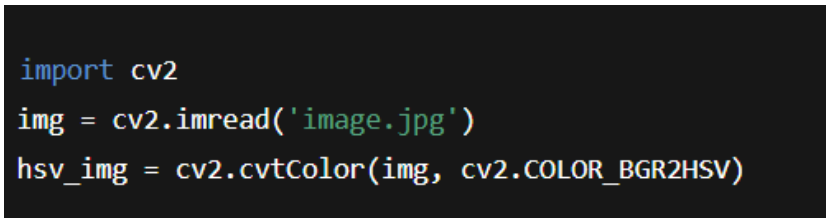
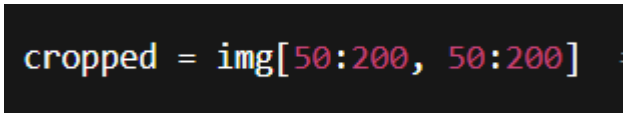
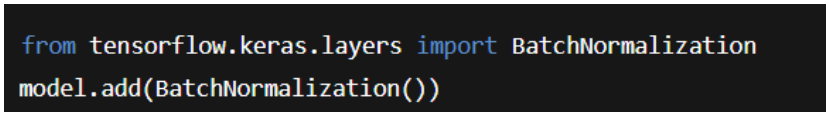
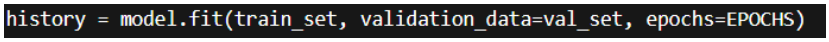
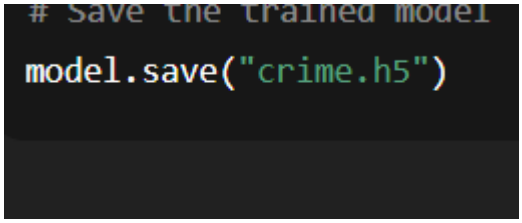
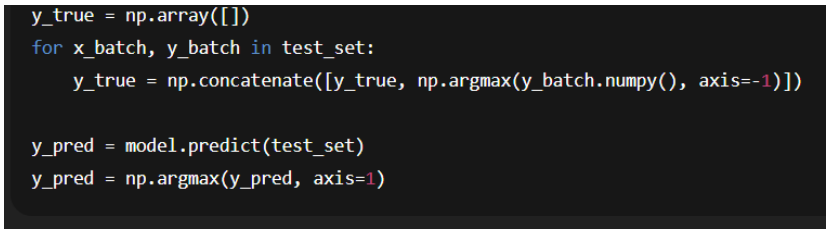
Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	**Description**
Data Overview	Load and explore image data from Train and Test directories, count class-wise images, and visualize dataset distribution using bar and pie charts.
Resizing	Resize all input images to a fixed target size of **64x64 pixels** for uniformity and compatibility with the neural network input.
Normalization	Normalize image pixel values to the range [0, 1] using the Rescaling layer, which improves model training performance and convergence.
Data Augmentation	(Implicit step in TensorFlow pipeline; can be added manually) Enhance dataset variability by applying techniques like flipping, rotating, etc.
Transfer Learning	Use **DenseNet121**, a pre-trained model on ImageNet, as the base model to extract features, reducing training time and improving accuracy.
Model Architecture	Build a custom Sequential model by stacking Rescaling, GlobalAveragePooling2D, Dense, and Dropout layers,

	followed by the final output layer.
Compilation	Compile the model using Adam optimizer and categorical_crossentropy loss function with accuracy as the evaluation metric.
Training	Fit the model on training data and validate using 20% of the training split for 5 epochs, and track the accuracy and loss.
Saving the Model	Save the trained model to disk (crime.h5) for later use in real-time prediction or deployment.
Model Evaluation	Load test dataset and evaluate the trained model by comparing predicted class indices with actual labels.
Single Image Prediction	Load individual test images, preprocess them (resize, convert to array, expand dims), predict using the model, and output the predicted crime label.
Web App Interface	Use Flask to build a web application with two routes: one to upload an image and one to display the predicted result.
Ngrok Deployment	Use pyngrok to expose the Flask server to the internet via a public URL for remote testing and demonstration.
Data Preprocessing Code Screenshots	
Resizing	`from tensorflow.keras.preprocessing import image` `img = image.load_img('path/to/image.jpg', target_size=(64, 64))`
Normalization	`from tensorflow.keras.layers import Rescaling` `model.add(Rescaling(1./255, input_shape=(64, 64, 3)))`

Data Augmentation	`from tensorflow.keras.preprocessing.image import ImageDataGenerator`
	`datagen = ImageDataGenerator(`
	`  rotation_range=30,`
	`  width_shift_range=0.1,`
	`  height_shift_range=0.1,`
	`  horizontal_flip=True`
	`)`

Denoising	
Edge Detection	
Color Space Conversion	
Image Cropping	
Batch Normalization	
Compilation	
Training	
Saving the Model	
Model Evaluation	

Single Image Prediction	```
	from tensorflow.keras.preprocessing import image
	
	img = image.load_img("path/to/image.png", target_size=(64, 64))
	x = image.img_to_array(img)
	x = np.expand_dims(x, axis=0)
	
	pred = np.argmax(model.predict(x), axis=1)[0]
	
	labels = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson',
	          'NormalVideos', 'Burglary', 'Explosion', 'Robbery', 'Abuse',
	          'Shooting', 'Shoplifting', 'RoadAccidents']
	
	print("Predicted Class:", labels[pred])
	``

Web App Interface	

```python
from flask import Flask, request, render_template_string
from werkzeug.utils import secure_filename
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

# Load model
model = load_model("crime.h5", compile=False)

# Flask app setup
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

labels = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson',
          'NormalVideos', 'Burglary', 'Explosion', 'Robbery', 'Abuse',
          'Shooting', 'Shoplifting', 'RoadAccidents']

@app.route('/')
def home():
    return render_template_string("""
        <h2>Upload an Image to Predict Crime</h2>
        <form action="/predict" method="post" enctype="multipart/form-data">
            <input type="file" name="image" required>
            <input type="submit" value="Predict">
        </form>
    """)

@app.route('/predict', methods=['POST'])
def predict():
    f = request.files['image']
    filename = secure_filename(f.filename)
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    f.save(filepath)

    img = image.load_img(filepath, target_size=(64, 64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    pred = np.argmax(model.predict(x), axis=1)[0]
    return f"<h3>Predicted Crime Category: <b>{labels[pred]}</b></h3>"
```

Ngrok Deployment	`from pyngrok import ngrok`
	
	`ngrok.set_auth_token("YOUR_NGROK_AUTH_TOKEN")  # Re`
	`public_url = ngrok.connect(5000)`
	`print("Ngrok URL:", public_url)`
	`app.run(port=5000)`