Model Development Phase Template

Date	1 JUNE 2025
Team ID	SWTID1749902640
Project Title	CRIME VISION:ADVANCED CRIME CLASSIFICATION LEARNING
MARKS	5 Marks

Model Selection Report

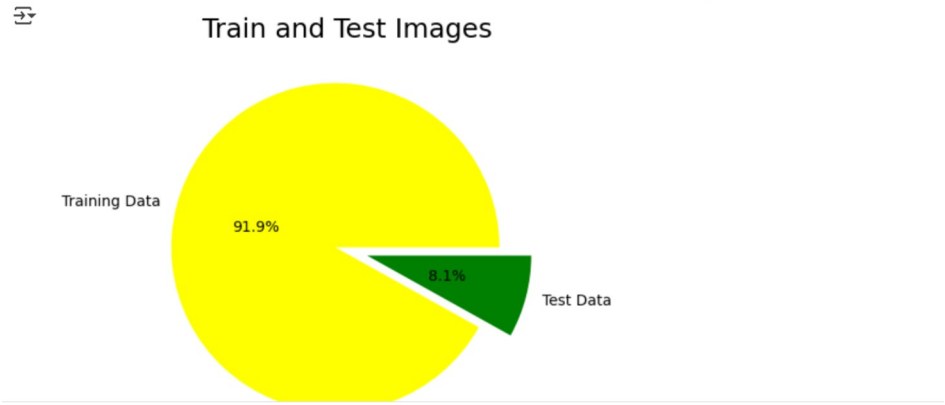
In the model selection report for future deep learning and computer vision projects, various architectures, such as CNNs or RNNs, will be evaluated. Factors such as performance, complexity, and computational requirements will be considered to determine the most suitable model for the task at hand.

Model	Description
Model 1: Simple CNN	Baseline CNN model with 2 convolutional layers and dense layers for classification.
Model 2: DenseNet121 (Final)	Transfer learning using pre-trained DenseNet121. Lightweight and accurate with frozen convolutional base. Selected as final model.

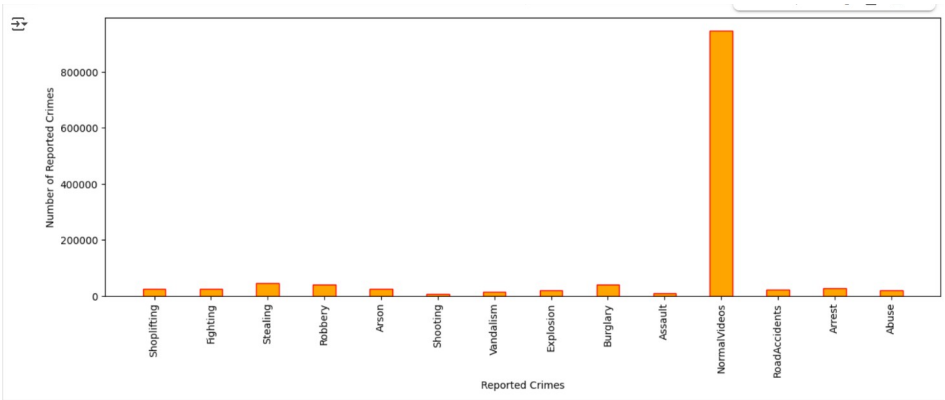
Model Selection Report:

Model	Description
Model 1	Training Setup Code & Visualizations :

```
[ ] # Visualizations
    plt.figure(figsize=(8, 5))
    plt.pie(x=np.array([train, test]), autopct="%.1f%%", explode=[0.1, 0.1],
            labels=["Training Data", "Test Data"], pctdistance=0.5, colors=['yellow', 'green'])
    plt.title("Train and Test Images ", fontsize=18)
    plt.show()
```




```
plt.figure(figsize=(15, 5))
plt.bar(list(crimes.keys()), list(crimes.values()), width=0.4, align="center",
        edgecolor=['red'], color=['orange'])
plt.xticks(rotation=90)
plt.xlabel("Reported Crimes")
plt.ylabel("Number of Reported Crimes")
plt.show()
```



Model 2	Dataset Loading & Augmentation :

```
train_set = image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=True,
    seed=SEED,
    validation_split=0.2,
    subset="training",
)
```

```
Found 1266345 files belonging to 14 classes.
Using 1013076 files for training.
```

```
val_set = image_dataset_from_directory(
    train_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=True,
    seed=SEED,
    validation_split=0.2,
    subset="validation",
)
```

```
Found 1266345 files belonging to 14 classes.
Using 253269 files for validation.
```

```
test_set = image_dataset_from_directory(
    test_dir,
    label_mode="categorical",
    batch_size=BATCH_SIZE,
    image_size=IMG_SHAPE,
    shuffle=False,
    seed=SEED,
)
```

```
Found 111308 files belonging to 14 classes.
```

Model 3	**Model Architecture Code :**

```
def transfer_learning():
    base_model = DenseNet121(include_top=False, input_shape=(*IMG_SHAPE, 3), weights="imagenet")
    base_model.trainable = False  # Freeze all layers
    return base_model
```

```
def create_model():
    model = Sequential([
        Rescaling(1./255, input_shape=(*IMG_SHAPE, 3)),
        transfer_learning(),
        GlobalAveragePooling2D(),
        Dense(256, activation="relu"),
        Dropout(0.2),
        Dense(512, activation="relu"),
        Dropout(0.2),
        Dense(1024, activation="relu"),
        Dense(n, activation="softmax")
    ])
    model.summary()
    return model
```

Model 4

Training the Model :

```
# Train the model
history = model.fit(x = train_set, validation_data = val_set,epochs = EPOCHS)
```

```
Epoch 1/5
7915/7915 ━━━━━━━━━━━━━━━━━━━━ 1301s 161ms/step - accuracy: 0.9211 - loss: 0.2912 - val_accuracy: 0.9858 - val_loss: 0.0537
Epoch 2/5
7915/7915 ━━━━━━━━━━━━━━━━━━━━ 1155s 139ms/step - accuracy: 0.9772 - loss: 0.0819 - val_accuracy: 0.9918 - val_loss: 0.0329
Epoch 3/5
7915/7915 ━━━━━━━━━━━━━━━━━━━━ 1128s 143ms/step - accuracy: 0.9823 - loss: 0.0641 - val_accuracy: 0.9927 - val_loss: 0.0272
Epoch 4/5
7915/7915 ━━━━━━━━━━━━━━━━━━━━ 1128s 138ms/step - accuracy: 0.9845 - loss: 0.0570 - val_accuracy: 0.9933 - val_loss: 0.0260
Epoch 5/5
7915/7915 ━━━━━━━━━━━━━━━━━━━━ 1102s 139ms/step - accuracy: 0.9862 - loss: 0.0524 - val_accuracy: 0.9938 - val_loss: 0.0234
```

```
# Save model
```

'ariables Terminal

Model 5

Evaluation on Test Set :

```
y_true = np.array([])

for x, y in test_set:
    y_true = np.concatenate([y_true, np.argmax(y.numpy(), axis=-1)])
```

```
y_pred = model.predict(test_set)
```

```
870/870 ━━━━━━━━━━━━━━━━━━━━ 1975s 2s/step
```

```
y_pred
```

```
array([[0.02731416, 0.0285832 , 0.07593239, ..., 0.1252327 , 0.0470111 ,
        0.1947365 ],
       [0.03004663, 0.02984005, 0.09710822, ..., 0.12563553, 0.04533564,
        0.22680359],
       [0.03198503, 0.02882746, 0.08044955, ..., 0.14730105, 0.06031731,
        0.22366147],
       ...,
       [0.04685552, 0.01129376, 0.18159631, ..., 0.10865402, 0.02512256,
        0.09404041],
       [0.05456919, 0.00962798, 0.14960171, ..., 0.11969905, 0.03055681,
        0.11526216],
       [0.05312692, 0.01975181, 0.17439762, ..., 0.11273936, 0.02866976,
        0.14421938]], dtype=float32)
```

```
y_true
```

```
y_true
```

```
array([ 0.,  0.,  0., ..., 13., 13., 13.])
```

```
y_pred = np.argmax(y_pred, axis=1)
```

Model 6

Individual Image Predictions :

```
from tensorflow.keras.preprocessing import image

img = image.load_img('/content/Test/RoadAccidents/RoadAccidents001_x264_0.png', target_size=(64,64))  # Reading image
x = image.img_to_array(img)  # Converting image into array
x = np.expand_dims(x, axis=0)  # Expanding Dimensions
pred = np.argmax(model.predict(x))  # Predicting the higher probability index

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

op[pred]  # List indexing with output
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
'Stealing'
```

```
img = image.load_img('/content/Test/Shoplifting/Shoplifting001_x264_0.png', target_size=(64,64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
pred = np.argmax(model.predict(x))
op = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson', 'NormalVideos', 'Burglary',
      'Explosion', 'Robbery', 'Abuse', 'Shooting', 'Shoplifting', 'RoadAccidents']
op[pred]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step
'RoadAccidents'
```

```
img = image.load_img('/content/Test/Explosion/Explosion002_x264_0.png', target_size=(64,64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
pred = np.argmax(model.predict(x))
op = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson', 'NormalVideos', 'Burglary',
      'Explosion', 'Robbery', 'Abuse', 'Shooting', 'Shoplifting', 'RoadAccidents']
op[pred]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step
'Stealing'
```

```
img = image.load_img('/content/Test/Burglary/Burglary005_x264_0.png', target_size=(64,64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
pred = np.argmax(model.predict(x))
op = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson', 'NormalVideos', 'Burglary',
      'Explosion', 'Robbery', 'Abuse', 'Shooting', 'Shoplifting', 'RoadAccidents']
op[pred]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step
'Shooting'
```

```
img = image.load_img('/content/Test/Robbery/Robbery048_x264_0.png', target_size=(64,64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
pred = np.argmax(model.predict(x))
op = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson', 'NormalVideos', 'Burglary',
      'Explosion', 'Robbery', 'Abuse', 'Shooting', 'Shoplifting', 'RoadAccidents']
op[pred]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step
'Shooting'
```

Model 7

Web App Deployment Code :

```
import re
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from flask import Flask, app,request,render_template
from tensorflow.keras import models
from tensorflow.keras.preprocessing import image
from tensorflow.python.ops.gen_array_ops import concat
from tensorflow.keras.models import load_model
```

```
[ ] #Loading the model
    model = load_model(r"crime.h5", compile=False)

    app = Flask(__name__)
```

```
[ ] #home page
    @app.route('/')
    def home():
        return render_template('home.html')

    #prediction page
    @app.route('/prediction')
    def prediction():
        return render_template('predict.html')
```

```
!pip install pyngrok
```

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

# Initialize Flask app
app = Flask(__name__)

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

# Define labels based on your training
labels = ['Fighting', 'Arrest', 'Vandalism', 'Assault', 'Stealing', 'Arson',
          'NormalVideos', 'Burglary', 'Explosion', 'Robbery', 'Abuse',
          'Shooting', 'Shoplifting', 'RoadAccidents']

# Configure upload folder
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Home route
@app.route('/')
def home():
```

```python
    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>
    """)

# Prediction route
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        # Retrieve the uploaded file
        f = request.files['image']
        filename = secure_filename(f.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        f.save(file_path)

        # Preprocess the image
        img = image.load_img(file_path, target_size=(64, 64))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        # Make prediction
        pred = np.argmax(model.predict(x), axis=1)[0]
        result = labels[pred]
```

```python
        result = labels[pred]

        return f"<h3>Predicted Crime Category: <b>{result}</b></h3>"
```

Model 8-	**Ngrok Public URL Setup :**

```
# Run the app
if __name__ == "__main__":
    # Set up ngrok for public URL
    ngrok.set_auth_token("2yeEMU5DS4DJwLKV63mXeUzdwp3_4NxbciuXYywDJ2juqybE")
    public_url = ngrok.connect(5000)
    print("Public URL:", public_url)
    app.run(port=5000)
```

```
Public URL: NgrokTunnel: "https://28f1-34-125-217-112.ngrok-free.app" -> "http://localhost:5000"
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [17/Jun/2025 21:07:32] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Jun/2025 21:07:33] "GET /favicon.ico HTTP/1.1" 404 -
INFO:werkzeug:127.0.0.1 - - [17/Jun/2025 21:14:17] "POST /predict HTTP/1.1" 400 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step
INFO:werkzeug:127.0.0.1 - - [17/Jun/2025 21:14:20] "POST /predict HTTP/1.1" 200 -
```