

Final Report

Crime Vision Advanced Crime Classification with Deep Learning

Team ID: 591975

Team Members:

1. **Akella Sumanasri (21BCE8937)**
2. **Sanghamitra Nayak (21BCE7579)**
3. **Saathvika Allamshetti (21BCE8863)**
4. **Samikshya Dash (21BCE7157)**

Project Report Format

1. **INTRODUCTION**
 - Project Overview
 - Purpose
2. **LITERATURE SURVEY**
 - Existing problem
 - References
 - Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
 - Empathy Map Canvas
 - Ideation & Brainstorming
4. **REQUIREMENT ANALYSIS**
 - Functional requirement
 - Non-Functional requirements
5. **PROJECT DESIGN**
 - Data Flow Diagrams & User Stories
 - Solution Architecture
6. **PROJECT PLANNING & SCHEDULING**
 - Technical Architecture
 - Sprint Planning & Estimation
 - Sprint Delivery Schedule
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
 - Feature 1
 - Feature 2
 - Database Schema (if Applicable)
8. **PERFORMANCE TESTING**
 - Performance Metrics
9. **RESULTS**
 - Output Screenshots
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
 - Source Code
 - GitHub & Project Demo Link

Project Description:

Crime identification using deep learning is a technique that involves applying deep learning techniques, specifically deep learning, to analyze images and video footage of crime scenes or incidents and identify and classify different types of crimes. Deep learning involves training neural networks on large amounts of data to recognize patterns and make predictions or decisions. By using deep learning, it is possible to analyze images and video footage of crime scenes or incidents and classify different types of crimes based on the type of activity depicted in the images. This can be useful in a variety of criminal justice and law enforcement contexts, including crime scene investigation, forensic analysis, and surveillance. Deep learning algorithms can be trained to recognize patterns and features in images and video that are relevant to identifying different types of crimes. They can also be used to analyze large amounts of data, such as surveillance footage, to identify trends and patterns in crime data. This can allow law enforcement agencies to develop strategies and interventions to prevent crime.

Ideation Phase

Empathize & Discover

Date	18 th October 2023
Team ID	PNT2023TMID591975
Project Name	Project – Crime Vision
Maximum Marks	4 Marks

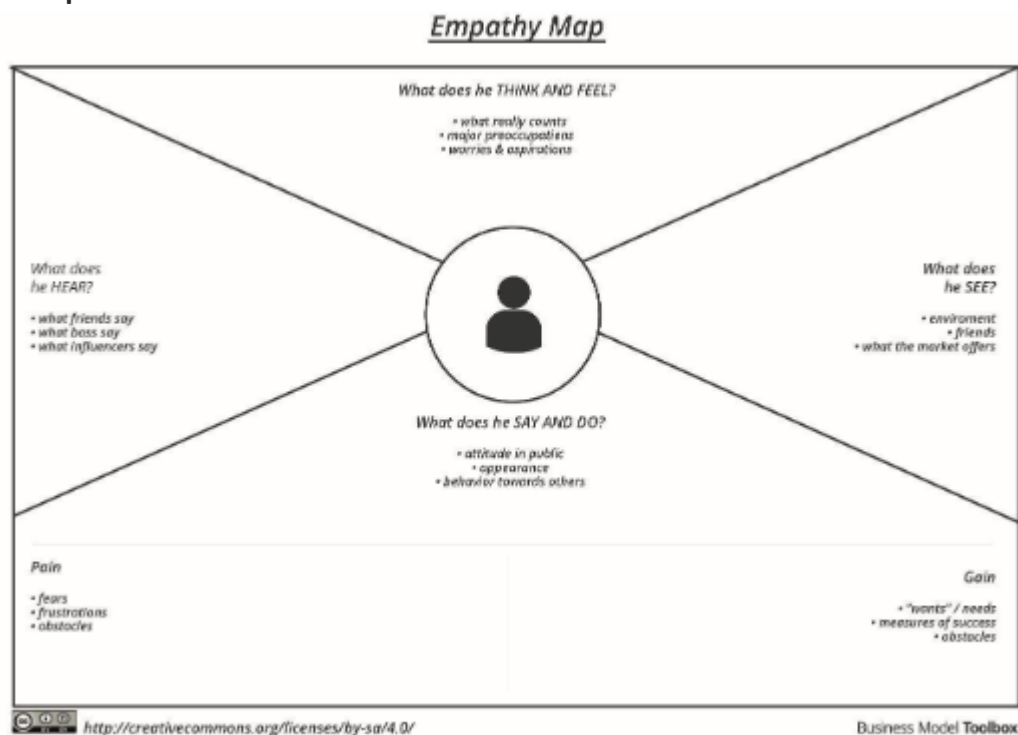
Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to help teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Example:



Reference: <https://www.mural.co/templates/empathy-map-canvas>

Project Description:

Template

Empathy map canvas

Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.

© 2016, Google LLC. All rights reserved.

2

Develop shared understanding and empathy

To enhance the ways you have gathered related to the people that are impacted by your work, it will help your generation ideas, problem features, or discuss decisions.

WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they're in?
What is their role in the situation?

WHAT do they FEEL?
What are they feeling right now?
What are they feeling for (fear, joy, etc.)?
What are they feeling for (fear, joy, etc.)?
What are they feeling for (fear, joy, etc.)?

WHAT do they THINK and FEEL?
What are they thinking?
What are they feeling?
What are they thinking?
What are they feeling?

WHAT do they NEED to DO?
What do they need to do?
What do they need to do?
What do they need to do?
What do they need to do?

WHAT do they SAY?
What do they say?
What do they say?
What do they say?
What do they say?

WHAT do they DO?
What do they do?
What do they do?
What do they do?
What do they do?

GOAL
What do they want to achieve?
What do they want to achieve?
What do they want to achieve?
What do they want to achieve?

CHALLENGES
What are the challenges?
What are the challenges?
What are the challenges?
What are the challenges?

OPPORTUNITIES
What are the opportunities?
What are the opportunities?
What are the opportunities?
What are the opportunities?

ASSUMPTIONS
What are the assumptions?
What are the assumptions?
What are the assumptions?
What are the assumptions?

CONTEXT
What is the context?
What is the context?
What is the context?
What is the context?

STAKEHOLDERS
Who are the stakeholders?
Who are the stakeholders?
Who are the stakeholders?
Who are the stakeholders?

RESOURCES
What are the resources?
What are the resources?
What are the resources?
What are the resources?

CONSTRAINTS
What are the constraints?
What are the constraints?
What are the constraints?
What are the constraints?

KEY INSIGHTS
What are the key insights?
What are the key insights?
What are the key insights?
What are the key insights?

RECOMMENDATIONS
What are the recommendations?
What are the recommendations?
What are the recommendations?
What are the recommendations?

CONCLUSIONS
What are the conclusions?
What are the conclusions?
What are the conclusions?
What are the conclusions?

Next steps
What are the next steps?
What are the next steps?
What are the next steps?
What are the next steps?

Share template feedback

Need some inspiration?
See how we've used the canvas to solve real-world problems.

See examples

Ideation Phase Brainstorm & Idea Prioritization Template

Date	18 th October 2023
Team ID	591975
Project Name	Crime Vision
Maximum Marks	4 Marks

Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Crime Vision (Team ID: 591975)

Project Description:

This project uses a special kind of computer technology called deep learning to help identify different types of crimes from pictures and videos. Deep learning is like teaching a computer to recognize patterns in images. We can use it to look at pictures or videos of crime scenes and figure out what kind of crime happened. This is really helpful for police and investigators. It helps them investigate crime scenes better and analyze evidence. It can also be used to watch a lot of video footage and spot any unusual or suspicious activities. This way, they can plan ways to stop crimes from happening in the future.

Team Gathering, Collaboration and Select the Problem Statement

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare
🕒 1 hour to collaborate
👤 2-8 people recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes



A Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.



B Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.



C Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →



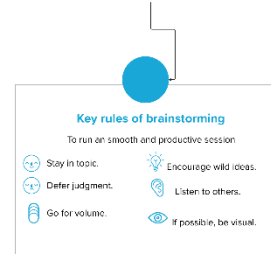
Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes



Crime identification using deep learning is a cutting-edge approach that leverages deep learning techniques to analyze images and video footage from crime scenes or incidents for the purpose of classifying and identifying various types of crimes. Our project utilizes deep learning to automatically analyze crime scene images and videos, classifying and identifying various crimes. This approach enhances investigations, aids forensic analysis, and supports surveillance in the criminal justice system. We develop deep learning algorithms to recognize crime patterns and process surveillance data to identify trends, empowering law enforcement for proactive crime prevention and response.



Need some inspiration?

See a finished version of this session to kickstart your work.

[Open example](#) →

Brainstorm, Idea Listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

Identify primary data sources, like surveillance and body cameras.

Model Choice based on nature of data

Privacy & Ethics

Person 2

Plan for accommodating increasing data volumes and emerging crime trends.

Performance Metrics

Design an Intuitive User Interface

Person3

Integration Strategy

Create a testing framework for real-world performance assessment.

Resource Requirements

Person 4

Budget Estimation

Create a pipeline for model training on the labeled dataset, including validation steps.

Legal and Regulatory Compliance



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Data Sources and Collection

Identify primary data sources, like surveillance and body cameras.

Model Choice based on nature of data

Plan for accommodating increasing data volumes and emerging crime trends.

Model Development and Performance

Performance Metrics

Create a pipeline for model training on the labeled dataset, including validation steps.

Budget Estimation

Create a testing framework for real-world performance assessment.

User Interface and Integration

Design an Intuitive User Interface

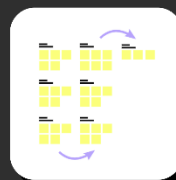
Integration Strategy

Other Necessities

Privacy & Ethics

Resource Requirements

Legal and Regulatory Compliance



Idea Prioritization

4

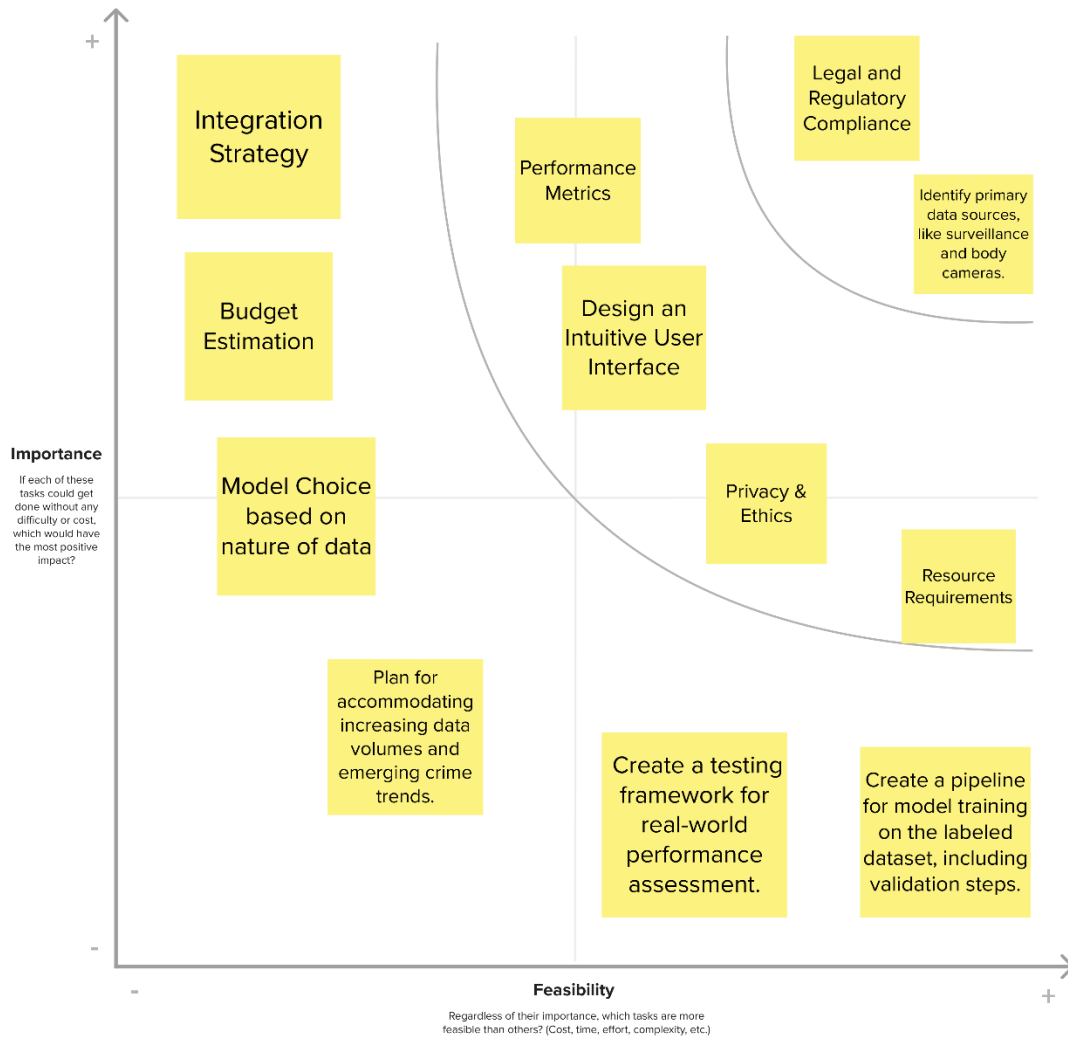
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H** key on the keyboard.



Project Design Phase-I
Proposed Solution Template

Date	23 rd October 2023
Team ID	PNT2022TMID591975
Project Name	Project – Crime Vision
Maximum Marks	2 Marks

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In today's fast-paced world, effective crime identification and classification are crucial for maintaining public safety and ensuring justice. Traditional methods of analyzing crime scenes and incidents often fall short in accurately identifying various types of crimes. To address this challenge, we propose a solution that leverages deep learning techniques.
2.	Idea / Solution description	The goal of this project is to develop a robust system capable of analyzing images and video footage from crime scenes or incidents. The system should be able to accurately identify and classify different types of crimes based on the activities depicted. This technology will find applications in various domains of criminal justice and law enforcement, including crime scene investigation, forensic analysis, and surveillance.
3.	Novelty / Uniqueness	This project uniquely applies deep learning to swiftly and accurately identify various types of crimes from visual data. It offers automation, data-driven insights, and ethical considerations, making it a cutting-edge tool for law enforcement. The inclusion of a feedback loop ensures continuous improvement, making it a powerful and innovative tool for law enforcement and criminal justice.

4.	Social Impact / Customer Satisfaction	<p>Social Impact:</p> <ul style="list-style-type: none"> • Enhances public safety, leading to a safer society. • Enables faster response times, preventing further harm. • Optimizes resource allocation for more efficient investigations. • Empowers investigators with advanced tools for solving crimes.
		<p>Customer Satisfaction:</p> <ul style="list-style-type: none"> • Ensures accurate and reliable crime identification. • Improves efficiency with real-time responses. • Offers user-friendly interfaces for ease of use. • Adapts and evolves over time for ongoing effectiveness.
5.	Business Model (Revenue Model)	<p>Business Model: Crime Identification Service</p> <ol style="list-style-type: none"> 1. Subscription Licensing: Law enforcement agencies pay for access tiers based on usage and features. 2. Support and Maintenance: Provide ongoing technical support, updates, and maintenance. 3. Partnerships and Collaborations: Integrate with existing law enforcement platforms. 4. Data Security Services: Provide compliance-related services.
6.	Scalability of the Solution	<p>Scalability Strategies</p> <ol style="list-style-type: none"> 1. Parallel Processing: Distribute workload for faster processing. 2. Cloud Infrastructure: Utilize scalable cloud platforms. 3. Load Balancing: Ensure even distribution of tasks. 4. Monitoring and Metrics: Track performance for proactive adjustments.

Project Design Phase-I Solution Architecture

Date	23 rd October 2023
Team ID	NM2023TMID591975
Project Name	Project – Crime Vision
Maximum Marks	5 Marks

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Example - Solution Architecture Diagram:

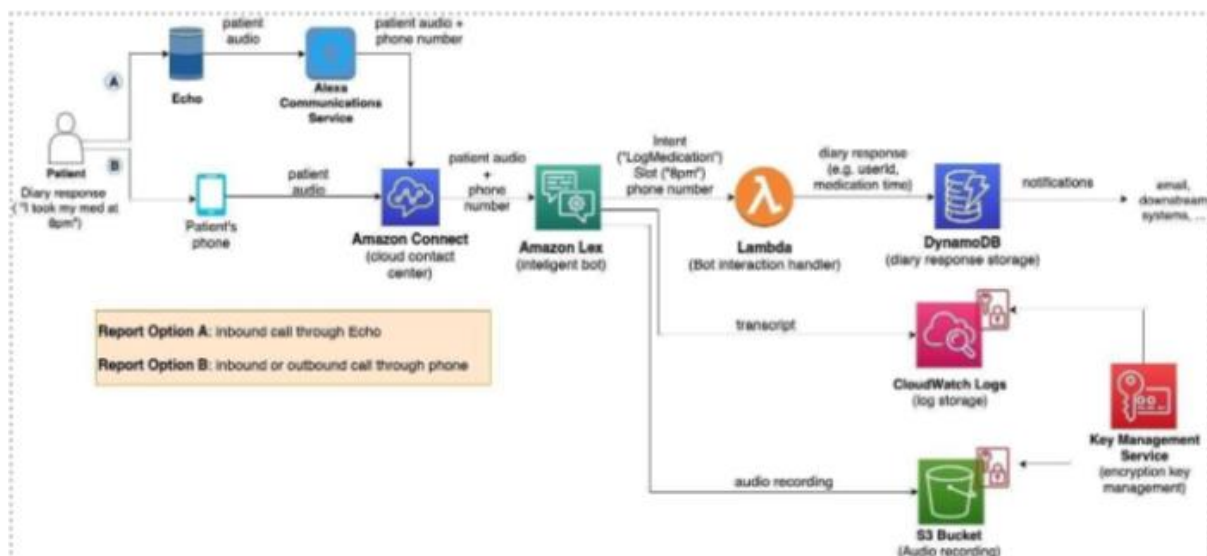


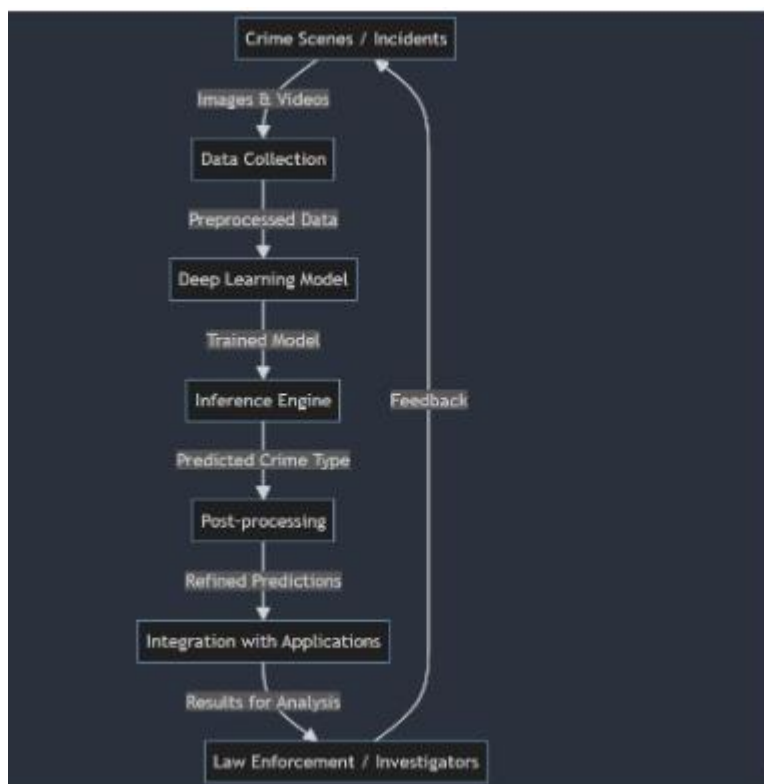
Figure 1: Architecture and data flow of the voice patient diary sample application

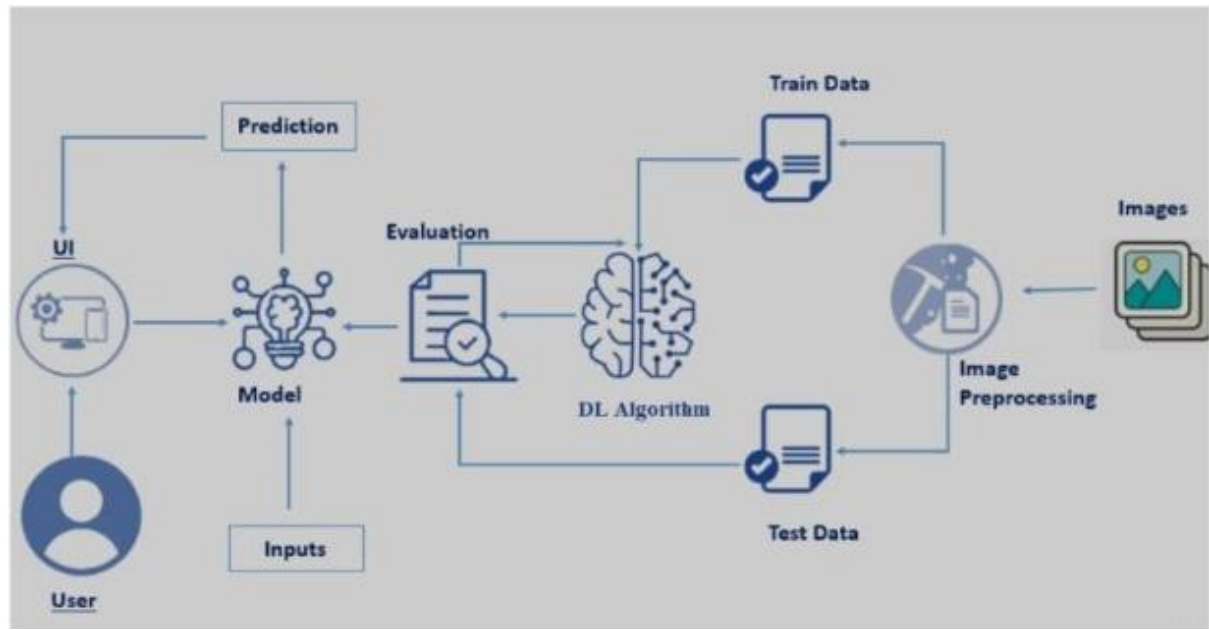
Crime Vision (Team ID: 591975)

Project Description:

This project uses a special kind of computer technology called deep learning to help identify different types of crimes from pictures and videos. Deep learning is like teaching a computer to recognize patterns in images. We can use it to look at pictures or videos of crime scenes and figure out what kind of crime happened. This is really helpful for police and investigators. It helps them investigate crime scenes better and analyze evidence. It can also be used to watch a lot of video footage and spot any unusual or suspicious activities. This way, they can plan ways to stop crimes from happening in the future.

Solution Architecture Diagram:

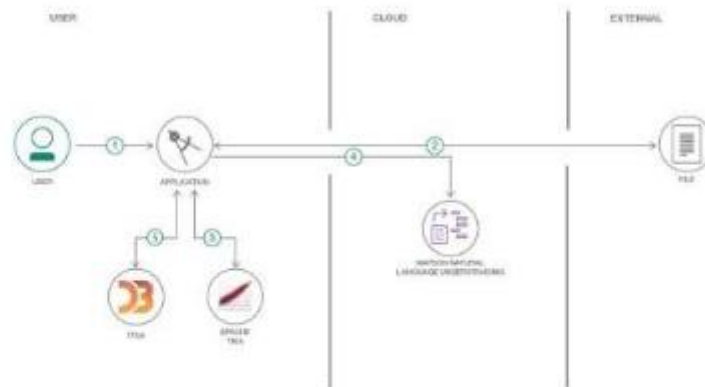




Project Design Phase-II Data Flow Diagram & User Stories

Date	23 rd October 2023
Team ID	NM2023TMID 591975
Project Name	Project - Crime Vision: Advanced Crime Classification with Deep Learning

Flow

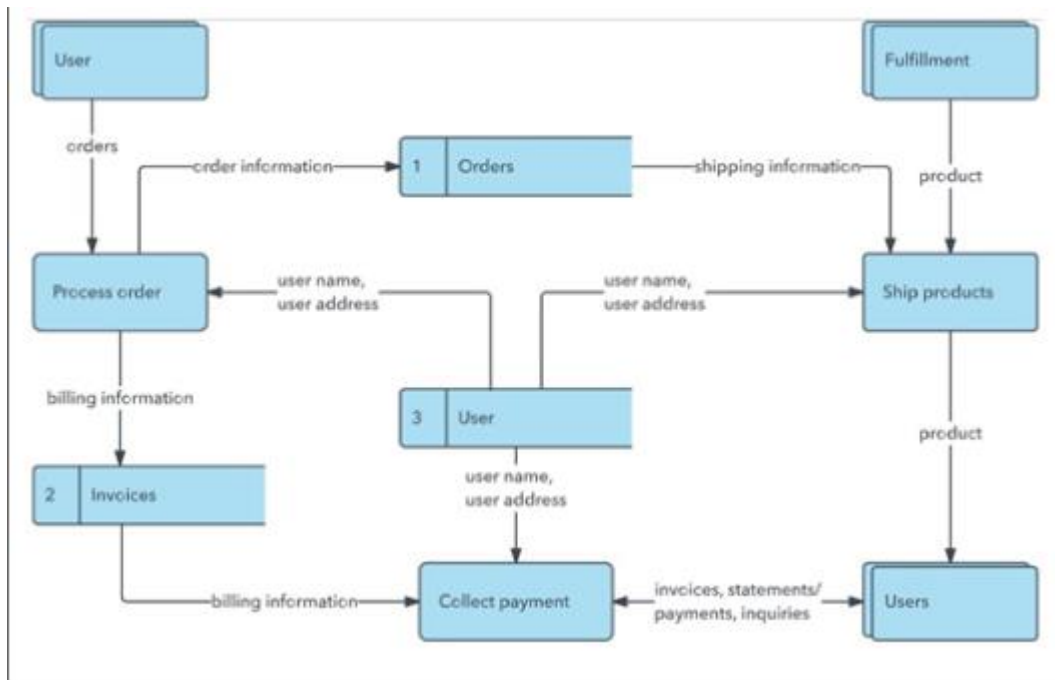


1. User configures credentials for the Watson Natural Language Understanding service and starts the app.
2. User selects data file to process and load.
3. Apache Tika extracts text from the data file.
4. Extracted text is passed to Watson NLU for enrichment.
5. Enriched data is visualized in the UI using the D3.js library.

Data Flow Diagrams:

Example: DFD Level 0 (Industry Standard)

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



Example: [\(Simplified\)](#)

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sumana
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Samikshya
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sanghamitra
		USN-4	As a user, I can register for the application through Gmail		Medium	Sathvika
	Login	USN-5	As a user, I can log into the application by		High	Sumana

			entering email & password			
	Dashboard					
Customer (Web user)						
Customer Care Executive						
Administrator						

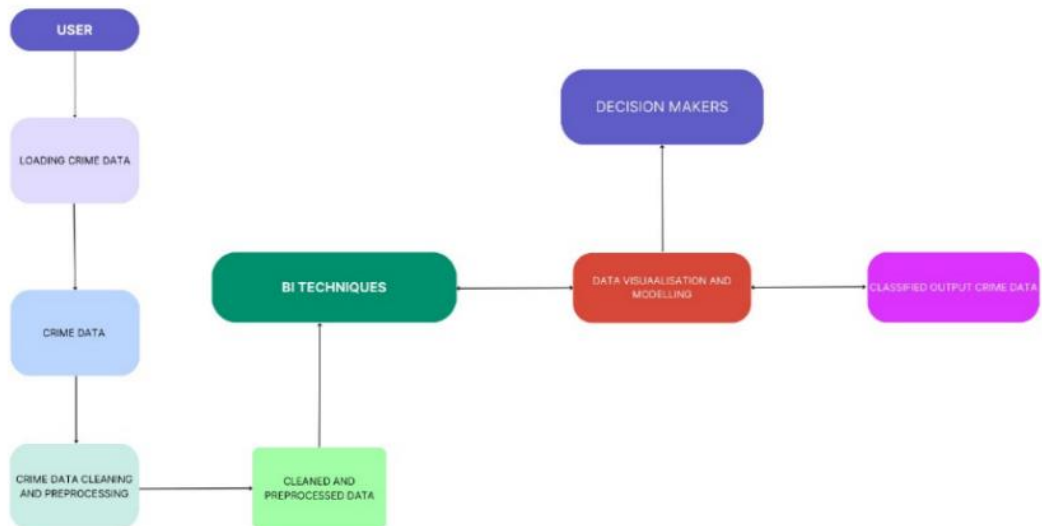
Crime Vision (Team ID: 591975)

Project Description:

This project uses a special kind of computer technology called deep learning to help identify different types of crimes from pictures and videos. Deep learning is like teaching a computer to recognize patterns in images. We can use it to look at pictures or videos of crime scenes and figure out what kind of crime happened. This is really helpful for police and investigators. It helps them investigate crime scenes better and analyse evidence. It can also be used to watch a lot of video footage and spot any unusual or suspicious activities. This way, they can plan ways to stop crimes from happening in the future.

Data Flow Diagram

DATA FLOW DIAGRAM



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Law Enforcement Officer	Crime Identification	USN-1	To submit various types of multimedia evidence, such as images and videos, to the deep learning system	<ul style="list-style-type: none"> - Able to upload images or videos. -potential criminal activities can be identified and investigated efficiently. 	High	Sprint-1
Forensic Analyst	Forensic Analysis	USN-2	To retrieve detailed analysis reports generated by the deep learning model	<ul style="list-style-type: none"> - Can access the uploaded media. - Use the information to support forensic investigations . -Can review and validate the identified objects and patterns. 	High	Sprint-1
System Administrator	User Management	USN-3	To ensure the secure and seamless flow of data within the system	<ul style="list-style-type: none"> -The deep learning algorithms can operate effectively and reliably. - Add, remove, or 	Medium	Sprint-2

Detective	Crime Details Extraction	USN-4	To receive real-time alerts when the system identifies potential matches with known criminals	- Can respond promptly to emerging threats.	Medium	Sprint-1
Criminal Database Manager	User Management	USN-5	Regularly update and maintain the database integrated with the deep learning system	- The system has the most recent and relevant data for accurate crime detection.	Medium	Sprint-2
Privacy Officer	Security Management	USN-6	To ensure that the system adheres to privacy regulations and guidelines	-The rights of individuals are protected during the data analysis process	Low	Sprint-3
System Developer	Developer Duties	USN-7	Continuously improve and update the deep learning algorithms based on feedback and emerging trends	-The system remains effective in adapting to new challenges and technologies.	Medium	Sprint-2
Citizen	Prime User	USN-8	To have a user-friendly interface to report crimes or suspicious activities	-Can actively contribute to community safety through the crime detection system.	High	Sprint-1

**Project Design Phase-II
Technology Stack (Architecture &
Stack)**

Date	03 October 2022
Team ID	PNT2022TMID591975
Project Name	Project - Crime Vision: Advanced Crime Classification with Deep Learning
Maximum Marks	4 Marks

Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI etc.	HTML, CSS, Python etc.
2.	Application Logic-1 (Data Collection and Preprocessing)	1. Utilize IP cameras or video streams as data sources. 2. Preprocess video frames to ensure consistency.	Python Libraries Like OpenCV
3.	Application Logic-2 (Anomaly Detection and Alert Generation)	1. Employ pre-trained deep learning models for object detection. 2. Utilize facial recognition models for identifying individuals. 3. Generate real-time alerts when anomalies are detected.	TensorFlow, deep Learning Frameworks.
4.	Application Logic-3 (User Interface, Logging, and Monitoring)	1. Update the web-based user interface to display video feeds with anomaly indicators. 2. Maintain a database to store information about detected anomalies, including timestamps and descriptions. 3. Implement cloud or server clusters for scalability.	Flask, load balancing
5.	Database	Varchar, Int, Float etc.	Kaggle
6.	File Storage	File storage requirements	Kaggle, RAM, ROM
7.	External API-1	NA	NA.
8.	External API-2	NA	NA
9.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.

10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc.
-----	---------------------------------	--	--

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Visual Analysis	The project primarily focuses on the analysis of images and video footage from crime scenes, relying on visual data for crime identification.	Python with Deep Learning
2.	Pattern Recognition	The system learns to recognize intricate patterns and features associated with different types of crimes, enabling accurate classification.	Python with Deep Learning
3.	Surveillance and Prevention	It can analyze large volumes of surveillance footage to identify trends and patterns, enabling proactive crime prevention strategies.	Python with Deep Learning
4.	Data Privacy and Security	The project places a strong emphasis on protecting sensitive data, ensuring compliance with privacy regulations and standards.	Python with Deep Learning
5.	Adaptive and Evolving	The system can potentially improve over time through feedback loops and continuous model training, adapting to new patterns and emerging crime trends.	Python with Deep Learning
6.	Ethical Considerations	The project acknowledges and addresses ethical and privacy concerns, ensuring responsible use of technology in law enforcement.	Python with Deep Learning

Project Planning Phase
Project Planning Template (Product Backlog, Sprint
Planning, Stories, Story points)

Date	18 October 2023
Team ID	PNT2022TMID591975
Project Name	Project – Crime Detection using Deep Learning
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Crime Identification	USN-1	To submit various types of multimedia evidence, such as images and videos, to the deep learning system	2	High	Sumana Sanghamitra
Sprint-2	Forensic Analysis	USN-2	To retrieve detailed analysis reports generated by the deep learning mode	1	High	Samikshya
Sprint-3	User Management	USN-3	To ensure the secure and seamless flow of data within the system	2	Low	Sathvika Sumana
Sprint-4	Crime Details Extraction	USN-4	To receive real-time alerts when the system identifies potential matches with known criminals	2	Medium	Sanghamitra Samikshya
Sprint-5	Security Management	USN-5	To ensure that the system adheres to privacy regulations and guidelines	2	High	Sathvika Sanghamitra
Sprint-6	Developer Duties	USN-6	Continuously improve and update the deep learning algorithms based on feedback and emerging trends	2	Medium	Samikshya Sumana

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	4 Days	28 Oct 2023	31 Oct 2023	20	31 Oct 2023
Sprint-2	20	3 Days	31 Oct 2023	02 Nov 2023	20	02 Nov 2023
Sprint-3	20	4 Days	02 Nov 2023	06 Nov 2023	20	07 Nov 2023
Sprint-4	20	6 Days	07 Nov 2023	13 Nov 2023	20	13 Nov 2023
Sprint-5	20	4 Days	13 Nov 2023	17 Nov 2023	20	17 Nov 2023
Sprint-6	20	2 Days	17 Nov 2023	19 Nov 2023	20	19 Nov 2023

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



Crime Vision: Advanced Crime Classification with Deep Learning

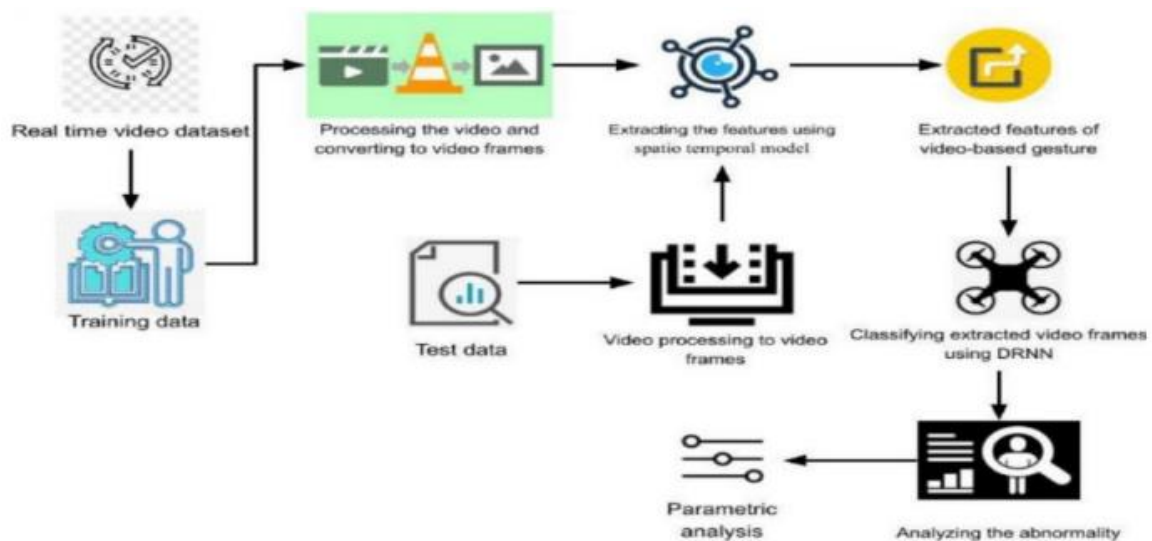
Project Description:

Crime identification using deep learning is a technique that involves applying deep learning techniques, specifically deep learning, to analyze images and video footage of crime scenes or incidents and identify and classify different types of crimes. Deep learning involves training convolutional neural networks on large amounts of data to recognize patterns and make predictions or decisions.

By using deep learning, it is possible to analyze images and video footage of crime scenes or incidents and classify different types of crimes based on the type of activity depicted in the images. This can be useful in a variety of criminal justice and law enforcement contexts, including crime scene investigation, forensic analysis, and surveillance.

Deep learning algorithms can be trained to recognize patterns and features in images and video that are relevant to identifying different types of crimes. They can also be used to analyze large amounts of data, such as surveillance footage, to identify trends and patterns in crime data. This can allow law enforcement agencies to develop strategies and interventions to prevent crime.

Technical Architecture:



Project Flow:

- The user interacts with the UI to choose an image.
- The chosen image is processed by a transfer learning deep learning model.
- The transfer learning model is integrated with a Flask application.
- The transfer learning model analyzes the image and generates predictions.
- The predictions are displayed on the Flask UI for the user to see.
- This process enables users to input an image and receive accurate predictions quickly.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
 - o Create a Train and Test path.
 - o Image Pre-processing.
 - o Import the required library
 - o Configuration of Images and preprocessing
 - o Apply Image_Dataset_from_directory functionality to Train set and Test set
- Model Building
 - o Create Transfer Learning Function
 - o Adding Dense Layer
 - o Configure the Learning Process
 - o Train the model
 - o Save the Model
 - o Test the model
 - o Application Building
 - o Create an HTML file
- Build Python Flask Code

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

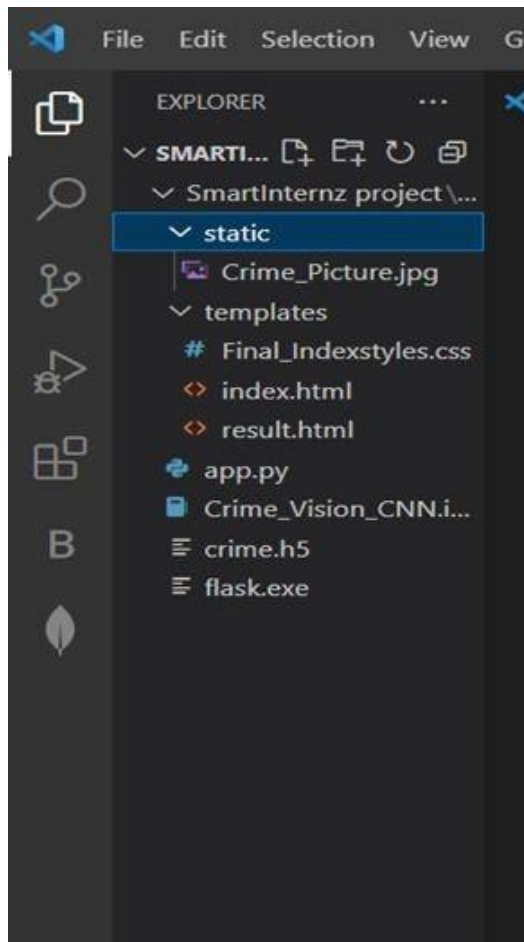
Deep Learning Concepts

- **CNN:** <https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add>
- **Transfer Learning:** <https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/>
- **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

Link: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create a Project folder which contains files as shown below



- The Dataset folder contains the training and testing images for training our model.
- For building a Flask Application we need HTML pages stored in the **templates** folder, CSS for styling the pages stored in the static folder and a python script **app.py** for server side scripting
- The IBM folder consists of a trained model notebook on IBM Cloud.
- Training folder consists of crime_classification.ipynb model training file & crime.h5 is saved model

Milestone 1: Data Collection

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

Activity 1: Download the dataset

The dataset contains images extracted from every video from the UCF Crime Dataset.

Every 10th frame is extracted from each full-length video and combined for every video in that class. All the images are of size 64*64 and in .png format
The dataset has a total of 14 Classes :

You can download the dataset used in this project using
the below link Dataset:- :

<https://www.kaggle.com/datasets/odins0n/ucf-crime-dataset>

Note: For better accuracy train on more images

We are going to build our training model on Google colab so we have to upload a dataset zip file on Google colab.

To upload a dataset zip file to Google Colab and then unzip it, you can follow these steps:

- Open Google Colab and create a new notebook.
- Click on the "Files" icon on the left-hand side of the screen.
- Click on the "Upload" button and select the zip file you want to upload.
- Wait for the upload to complete. You should see the file appear in the "Files" section.
- To unzip the file, you can use the following command:

```
!unzip /content/ucf-crime-dataset.zip
```

Streaming output truncated to the last 5000 lines.

```
inflating: Train/Vandalism/Vandalism035_x264_230.png
inflating: Train/Vandalism/Vandalism035_x264_240.png
inflating: Train/Vandalism/Vandalism035_x264_250.png
inflating: Train/Vandalism/Vandalism035_x264_260.png
inflating: Train/Vandalism/Vandalism035_x264_270.png
inflating: Train/Vandalism/Vandalism035_x264_280.png
inflating: Train/Vandalism/Vandalism035_x264_290.png
inflating: Train/Vandalism/Vandalism035_x264_30.png
inflating: Train/Vandalism/Vandalism035_x264_300.png
inflating: Train/Vandalism/Vandalism035_x264_310.png
inflating: Train/Vandalism/Vandalism035_x264_320.png
inflating: Train/Vandalism/Vandalism035_x264_330.png
inflating: Train/Vandalism/Vandalism035_x264_340.png
inflating: Train/Vandalism/Vandalism035_x264_350.png
inflating: Train/Vandalism/Vandalism035_x264_360.png
inflating: Train/Vandalism/Vandalism035_x264_370.png
inflating: Train/Vandalism/Vandalism035_x264_380.png
inflating: Train/Vandalism/Vandalism035_x264_390.png
inflating: Train/Vandalism/Vandalism035_x264_40.png
inflating: Train/Vandalism/Vandalism035_x264_400.png
inflating: Train/Vandalism/Vandalism035_x264_410.png
inflating: Train/Vandalism/Vandalism035_x264_420.png
inflating: Train/Vandalism/Vandalism035_x264_430.png
inflating: Train/Vandalism/Vandalism035_x264_440.png
...
inflating: Train/Vandalism/Vandalism050_x264_870.png
inflating: Train/Vandalism/Vandalism050_x264_880.png
inflating: Train/Vandalism/Vandalism050_x264_890.png
inflating: Train/Vandalism/Vandalism050_x264_90.png
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

Activity 2: Create training and testing dataset

To build a DL model we have to split training and testing data into two separate folders.

But In the project dataset folder training and testing folders are presented. So, in this case we just have to assign a variable and pass the folder path to it.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255, zoom_range = 0.2, horizontal_flip = True)
```

```
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
x_train = train_datagen.flow_from_directory('/content/Train', target_size = (224,224), class_mode='categorical', batch_size=32)
x_train
```

Found 1266345 images belonging to 14 classes.

<keras.src.preprocessing.image.DirectoryIterator at 0x7ed04f735750>

```
x_test = train_datagen.flow_from_directory('/content/Test', target_size = (224,224), class_mode='categorical', batch_size=32)
x_test
```

Milestone 2: Image Preprocessing

In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense
```

To understand the above imported libraries:-

- **Image_dataset_from_directory** : is a function in the tensorflow.keras.preprocessing module of TensorFlow, which allows you to create a TensorFlow Dataset from a directory containing image files. This function can be useful for training deep learning models on large image datasets.
- **Keras**: Keras is a high-level neural network API written in Python that allows for fast experimentation and prototyping of deep learning models.
- **DenseNet121**: It has been trained on large-scale image classification datasets such as ImageNet and has achieved state-of-the-art performance on a range of benchmark tasks. It has also been used as a pre-trained model for transfer learning in various computer vision applications.
- **Global average pooling 2D (GAP 2D)**: It is a type of pooling operation commonly used in convolutional neural networks (CNNs) for image classification tasks.
- **Dense Layer**: A dense layer in neural networks is a fully connected layer where each neuron in the layer is connected to every neuron in the previous layer, and each connection has a weight associated with it.
- **Flatten Layer**: A flatten layer in neural networks is a layer that reshapes the input tensor into a one-dimensional array, which can then be passed to a fully connected layer.
- **Input Layer**: The input layer in neural networks is the first layer of the network that receives the input data and passes it on to the next layer for further processing.
- **Maxpooling 2D** : is a downsampling operation that reduces the spatial dimensions (height and width) of an input tensor while preserving the number of channels. The operation takes a window of a fixed size, typically 2x2, and outputs the maximum value within that window. Max Pooling helps reduce the computational cost of the network while also increasing its robustness to small translations of the input.
- **Convolution 2D**: It is a linear operation that applies a set of learnable filters (also called kernels or weights) to an input tensor to extract features. The filters slide over the input tensor, computing a dot product between their values and the values of the input tensor at each position. The output of a convolutional layer is a set of

feature maps, each corresponding to a specific filter. Convolution 2D is the main building block of CNNs and is used to learn representations of the input data.

- **image:** from tensorflow.keras.preprocessing import image imports the image module from Keras' tensorflow.keras.preprocessing package. This module provides a number of image preprocessing utilities, such as loading images, converting images to arrays, and applying various image transformations.
- **load_img:** load_img is a function provided by the tensorflow.keras.preprocessing.image module that is used to load an image file from the local file system. It takes the file path as input and returns a PIL (Python Imaging Library) image object.
- **Dropout :** is a regularization technique that randomly drops out a fraction of the neurons in a neural network during training. This helps to prevent overfitting, which is when a model performs well on the training data but poorly on new data. Dropout forces the network to learn more robust features by preventing any one neuron from becoming too important in the network's predictions.
- **Numpy:** It is for performing mathematical functions
- **Matplotlib:** Matplotlib is a data visualization library in Python that is widely used for creating high-quality, publication-ready plots and charts.
- **clear_output:** command is used to clear the output of a Jupyter notebook cell. This can be useful when you want to update the output of a cell with new information, or when you want to remove previous output that is no longer relevant.

Activity 2: Configuration of Images and preprocessing

```
model = Sequential()
model.add(Convolution2D(32,(3,3), activation = 'relu', input_shape = (224,224,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(150, activation = 'relu'))
model.add(Dense(14, activation = 'softmax'))
```

directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.

batch size: Size of the batches of data which is 64. **target size:** Size to resize images after they are read from disk.

Learning Rate (LR): The learning rate is a hyperparameter that determines the step size at each iteration during gradient descent optimization. Gradient descent is the most common optimization algorithm used in machine learning to update the weights of a neural network during training. The learning rate controls how quickly or slowly the weights are updated in response to the error gradient. A high learning rate can cause the algorithm to converge too quickly, whereas a low learning rate can cause slow convergence or even prevent the model from converging.

Seeds: Seeds are used in machine learning to ensure that results are reproducible. A seed is a random number that is used to initialize the random number generator before training the model. By setting the seed value, we can ensure that the same sequence of random numbers is generated every time the code is run. This is important when developing models as it allows us to compare results between different runs and ensure that any changes to the model or hyperparameters are actually improving performance.

Now it is time to Build input and output layers for Transfer Learning model

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

Hidden layers freeze because they have trained sequence, so changing the input and output layers

Activity 3: Apply Image_Dataset_from_directory functionality to Train set and Test set

`ImageDataset.from_directory()` is a function from the TensorFlow library used to load images from a directory and create a dataset object that can be used for machine learning tasks, such as image classification or object detection.

The function takes the following arguments:

- **directory:** A string representing the directory where the images are located.
- **labels:** A list of strings representing the class labels for the images.
- **batch_size:** An integer representing the number of images to load in each batch.
- **image_size:** A tuple representing the size to which the images should be resized.

```
train_datagen = ImageDataGenerator(rescale =1./255, zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale =1./255)
```

```
x_train = train_datagen.flow_from_directory('/content/Train', target_size = (224,224), class_mode='categorical', batch_size=32)
x_train
```

Found 1266345 images belonging to 14 classes.

<keras.src.preprocessing.image.DirectoryIterator at 0x7ed04f735750>

[+ Code](#) [+ Markdown](#)

```
x_test = train_datagen.flow_from_directory('/content/Test', target_size = (224,224), class_mode='categorical', batch_size=32)
x_test
```

Found 111308 images belonging to 14 classes.

<keras.src.preprocessing.image.DirectoryIterator at 0x7ed0ce1fe080>

Milestone 3: Model Building

Activity 1: Create Transfer Learning Function

Now, let us create transfer learning function with DenseNet121 with parameters include_top, and weights imagenet with mentioned input shape. Also, we are setting threshold at 149.

DenseNet is a convolutional neural network where each layer is connected to all other layers that are deeper in the network, that is, the first layer is connected to the 2nd, 3rd, 4th and so on, the second layer is connected to the 3rd, 4th, 5th and so on.

- **include_top:** whether to include the fully-connected layer 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_shape:** optional shape tuple, only to be specified if include_top is False (otherwise the input shape has to be (224, 224, 3))

Activity 2:

Adding Dense Layers

```
model = Sequential()
model.add(Convolution2D(32,(3,3), activation = 'relu', input_shape = (224,224,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Convolution2D(32,(3,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(150, activation = 'relu'))
model.add(Dense(14, activation = 'softmax'))
```

A **dense** layer is a deeply connected neural network layer. It is the most common and frequently used layer.

The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities. Understanding the model is a very important phase to properly use it for training and prediction purposes.

Keras provides a simple method, summary to get the full information about the model and its layers.

```
model=create_model()

model.compile(optimizer="adam",
              loss='categorical_crossentropy',
              metrics = ['accuracy'])

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_29084464/29084464 [=====] - 2s 0us/step
Model: "sequential"
```

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 2, 2, 1024)	7037504
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0
dense (Dense)	(None, 256)	262400
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131584
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1024)	525312
dense_3 (Dense)	(None, 14)	14350

```
=====  
Total params: 7,971,150  
Trainable params: 6,386,894  
Non-trainable params: 1,584,256
```

Activity 3: Configure the Learning Process

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer

```

<ipython-input-11-7aab2fc0a681>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`
model.fit_generator(x_train, steps_per_epoch=len(x_train)/50, epochs = 10, validation_data=x_test, validation_steps=len(x_test))
Epoch 1/10
791/791 [=====] - 1982s 2s/step - loss: 0.9879 - accuracy: 0.7653 - val_loss: 2.1247 - val_accuracy: 0.5568
Epoch 2/10
791/791 [=====] - 1986s 3s/step - loss: 0.6666 - accuracy: 0.8221 - val_loss: 2.1643 - val_accuracy: 0.4938
Epoch 3/10
791/791 [=====] - 1917s 2s/step - loss: 0.5468 - accuracy: 0.8503 - val_loss: 2.3249 - val_accuracy: 0.5048
Epoch 4/10
791/791 [=====] - 1905s 2s/step - loss: 0.4540 - accuracy: 0.8742 - val_loss: 2.3809 - val_accuracy: 0.5581
Epoch 5/10
791/791 [=====] - 1861s 2s/step - loss: 0.4050 - accuracy: 0.8871 - val_loss: 2.2491 - val_accuracy: 0.5249
Epoch 6/10
791/791 [=====] - 1843s 2s/step - loss: 0.3608 - accuracy: 0.8979 - val_loss: 2.7502 - val_accuracy: 0.5177
Epoch 7/10
791/791 [=====] - 1813s 2s/step - loss: 0.3365 - accuracy: 0.9067 - val_loss: 2.3852 - val_accuracy: 0.5094
Epoch 8/10
791/791 [=====] - 1834s 2s/step - loss: 0.3196 - accuracy: 0.9108 - val_loss: 2.6323 - val_accuracy: 0.4869
Epoch 9/10
791/791 [=====] - 1833s 2s/step - loss: 0.2919 - accuracy: 0.9187 - val_loss: 2.5938 - val_accuracy: 0.5249
Epoch 10/10
791/791 [=====] - 1915s 2s/step - loss: 0.2757 - accuracy: 0.9216 - val_loss: 2.6197 - val_accuracy: 0.4669

```

Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

ACTIVITY 4: Train the model:

Now, let us train our model with our image dataset. The model is trained for 5 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch.

fit_generator functions used to train a deep learning neural network

Arguments:

- **steps_per_epoch**: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of **steps_per_epoch** as the total number of samples in your dataset divided by the batch size.

```

<ipython-input-11-7aab2fc0a681>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit_model_with_generator` instead.
model.fit_generator(x_train, steps_per_epoch=len(x_train)/50, epochs = 10, validation_data=x_test, validation_steps=len(x_test))
Epoch 1/10
791/791 [=====] - 1982s 2s/step - loss: 0.9879 - accuracy: 0.7653 - val_loss: 2.1247 - val_accuracy: 0.5568
Epoch 2/10
791/791 [=====] - 1986s 3s/step - loss: 0.6666 - accuracy: 0.8221 - val_loss: 2.1643 - val_accuracy: 0.4938
Epoch 3/10
791/791 [=====] - 1917s 2s/step - loss: 0.5468 - accuracy: 0.8503 - val_loss: 2.3249 - val_accuracy: 0.5048
Epoch 4/10
791/791 [=====] - 1905s 2s/step - loss: 0.4540 - accuracy: 0.8742 - val_loss: 2.3809 - val_accuracy: 0.5581
Epoch 5/10
791/791 [=====] - 1861s 2s/step - loss: 0.4050 - accuracy: 0.8871 - val_loss: 2.2491 - val_accuracy: 0.5249
Epoch 6/10
791/791 [=====] - 1843s 2s/step - loss: 0.3608 - accuracy: 0.8979 - val_loss: 2.7502 - val_accuracy: 0.5177
Epoch 7/10
791/791 [=====] - 1813s 2s/step - loss: 0.3365 - accuracy: 0.9067 - val_loss: 2.3852 - val_accuracy: 0.5094
Epoch 8/10
791/791 [=====] - 1834s 2s/step - loss: 0.3196 - accuracy: 0.9108 - val_loss: 2.6323 - val_accuracy: 0.4869
Epoch 9/10
791/791 [=====] - 1833s 2s/step - loss: 0.2919 - accuracy: 0.9187 - val_loss: 2.5938 - val_accuracy: 0.5249
Epoch 10/10
791/791 [=====] - 1915s 2s/step - loss: 0.2757 - accuracy: 0.9216 - val_loss: 2.6197 - val_accuracy: 0.4669

```

- Epochs: an integer and number of epochs we want to train our model for.
- validation_data can be either:
 - an inputs and targets list
 - a generator
 - an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

Accuracy of the model after 5 epochs.

Milestone 4: Save the Model

The model is saved with .h5 extension as follows

```
[ ] #Saving our model

model.save('crime.h5')

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format
saving_api.save_model(
```

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

Testing the model:

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.

```
[ ] #Testing the model
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

[ ] import tensorflow as tf

[ ] model=tf.keras.models.load_model(r"/content/crime.h5",compile=False)

img=image.load_img('/content/Train/RoadAccidents/RoadAccidents003_x264_1140.png',target_size=(224,224))
img
```



Load the saved model using load_model

```
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred = np.argmax(model.predict(x))
```

Python

```
1/1 [=====] - 0s 31ms/step
```

```
op=['Abuse','Arrest','Arson','Assault','Burglary','Explosion','Fighting','NormalVideos','RoadAccidents','Robbery','Shooting','Shoplifting','St
op[pred]
```

Python

```
'NormalVideos'
```

Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks

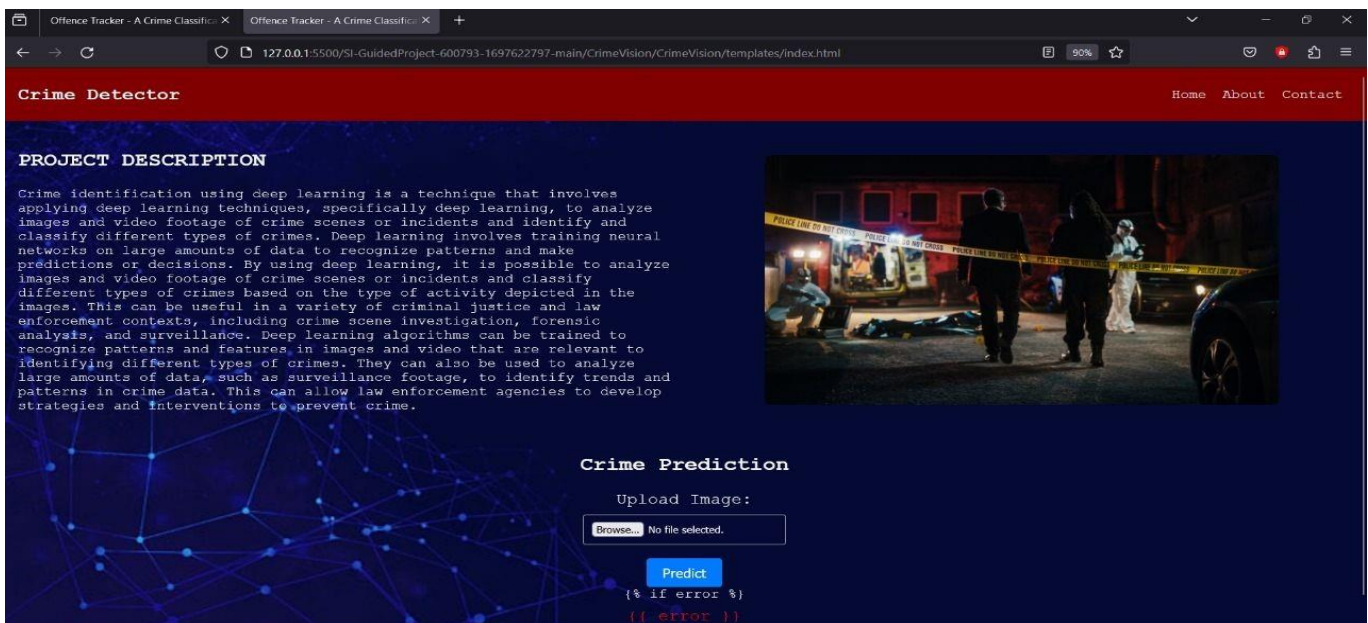
- Building HTML Pages • Building python code

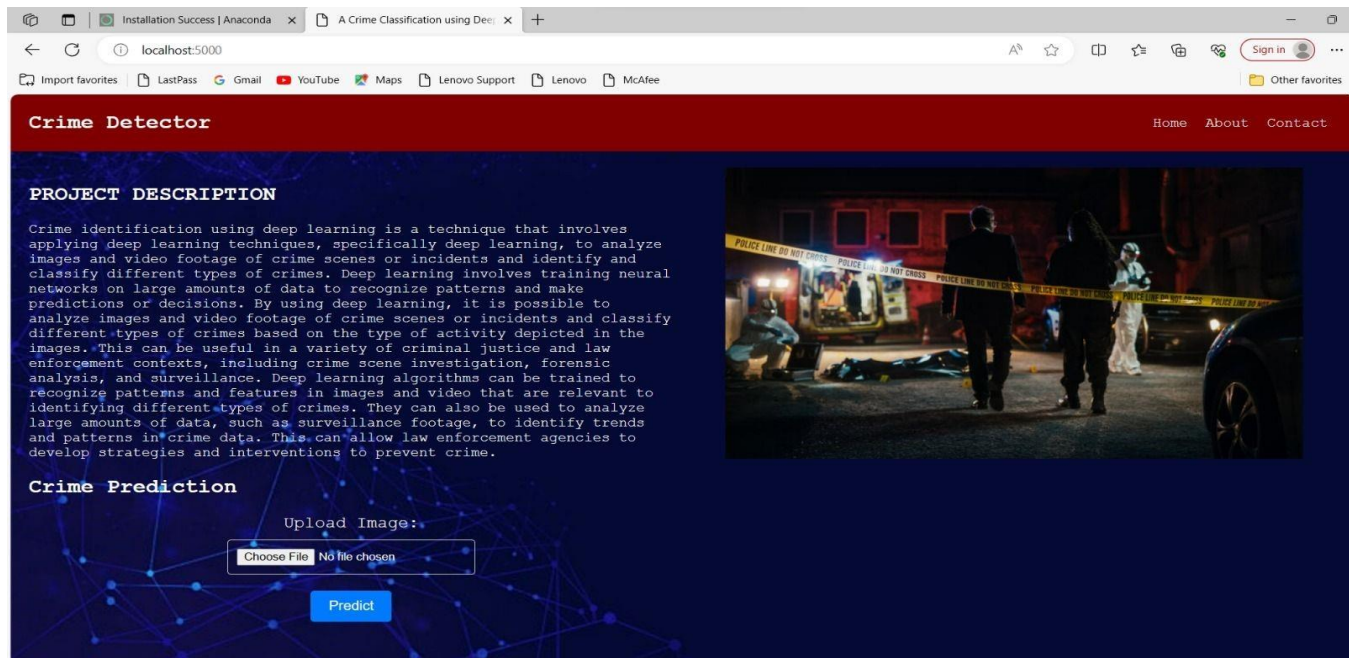
Activity1: Building Html Pages:

For this project create one HTML file namely

- index.html
- result.html

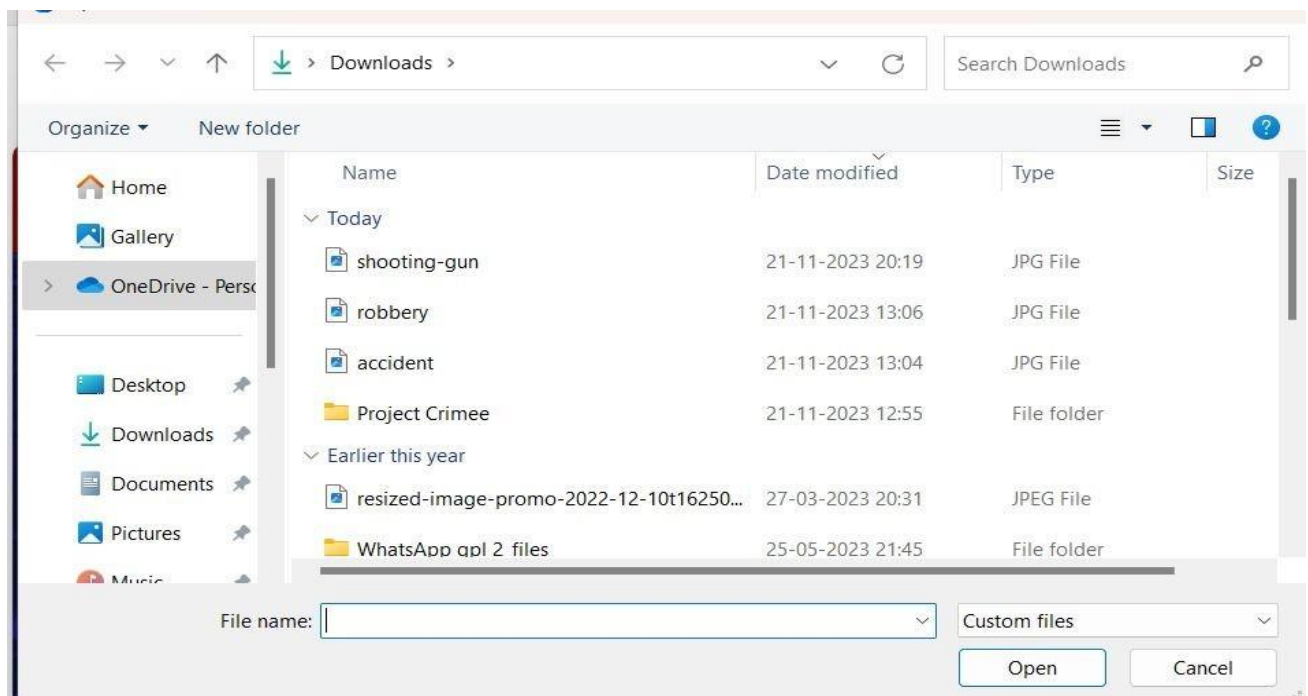
Let's see how our home.html page looks like:





When you click on the Choose file button, it will redirect you to the below page:

From here you can choose different images for getting prediction



Activity 2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, request, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import io
```

Loading the saved model and initializing the flask app

```
app = Flask(__name__)

# Load the trained model
model = load_model('crime.h5')
```

Render HTML Pages:

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

@app.route('/predict', methods=['POST'])
def predict():
    # Check if the POST request has the file part
    if 'file' not in request.files:
        return render_template('index.html', error='No file part')
```

Once we uploaded the file into the app, then verifying the file uploaded properly or not. Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with prediction.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/predict', methods=['POST'])
def predict():
    # Check if the POST request has the file part
    if 'file' not in request.files:
        return render_template('index.html', error='No file part')

    file = request.files['file']

    # If the user does not select a file, submit an empty part without filename
    if file.filename == '':
        return render_template('index.html', error='No selected file')

    # If the file is allowed and properly uploaded
    if file and allowed_file(file.filename):
        # Make prediction
        prediction = make_prediction(file)

        return render_template('result.html', filename=file.filename, prediction=prediction)
    else:
        return render_template('index.html', error='File type not allowed')
```

```
def make_prediction(file):
    try:
        # Load image using io.BytesIO
        img = image.load_img(io.BytesIO(file.read()), target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        pred = model.predict(x)
        predicted_class = crime_categories[np.argmax(pred)]
        return predicted_class
    except Exception as e:
        print(f"Error making prediction: {e}")
        return "Error making prediction"
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model. Predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier

Main Function:

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

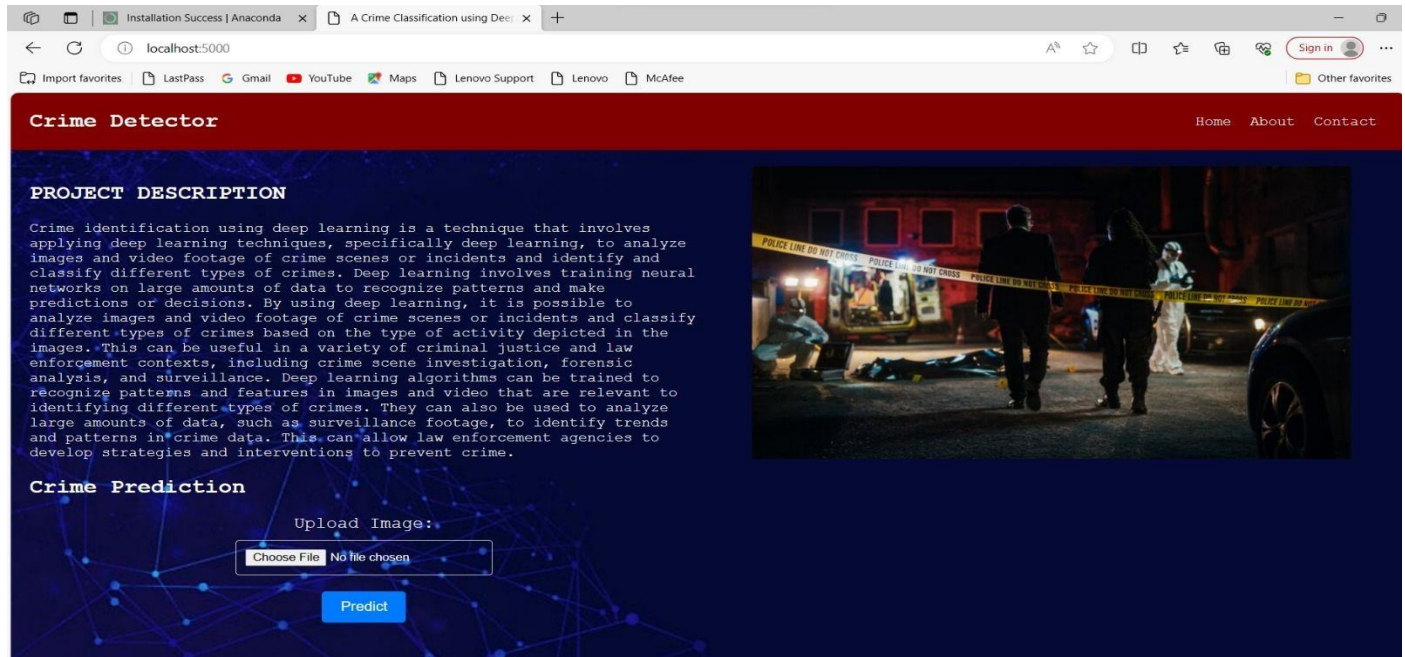
Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

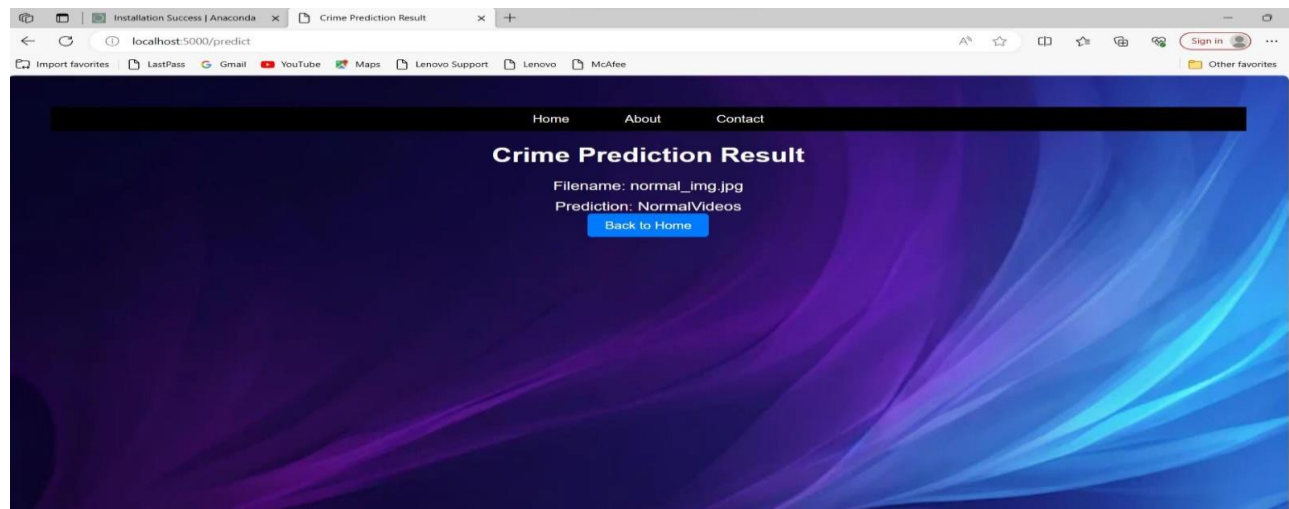
```
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations,  
rebuild TensorFlow with the appropriate compiler flags.  
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat
```

Now, Go the web browser and write the localhost url
(http://127.0.0.1:5000) to get the below result

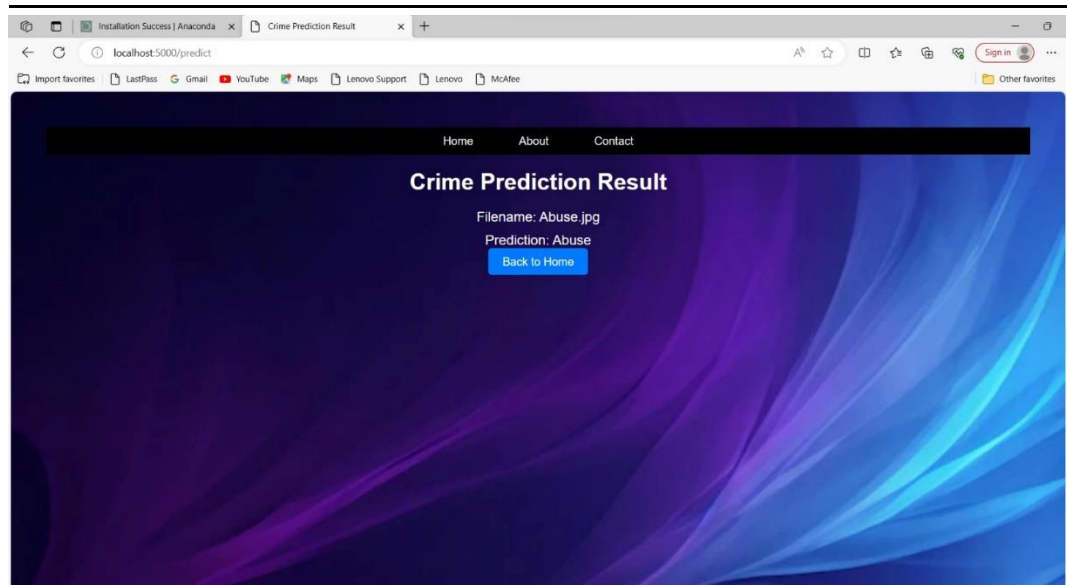
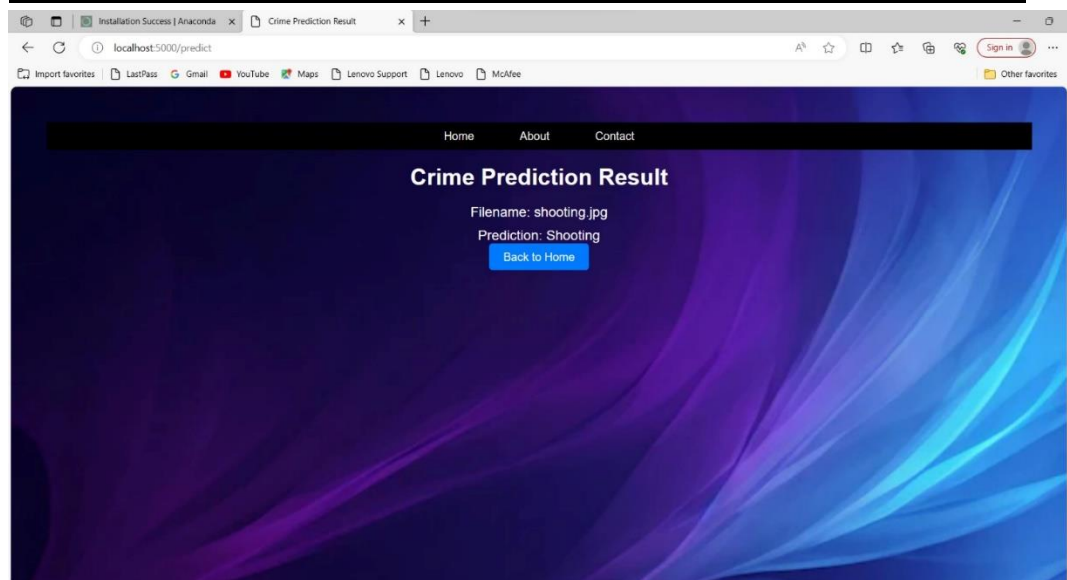
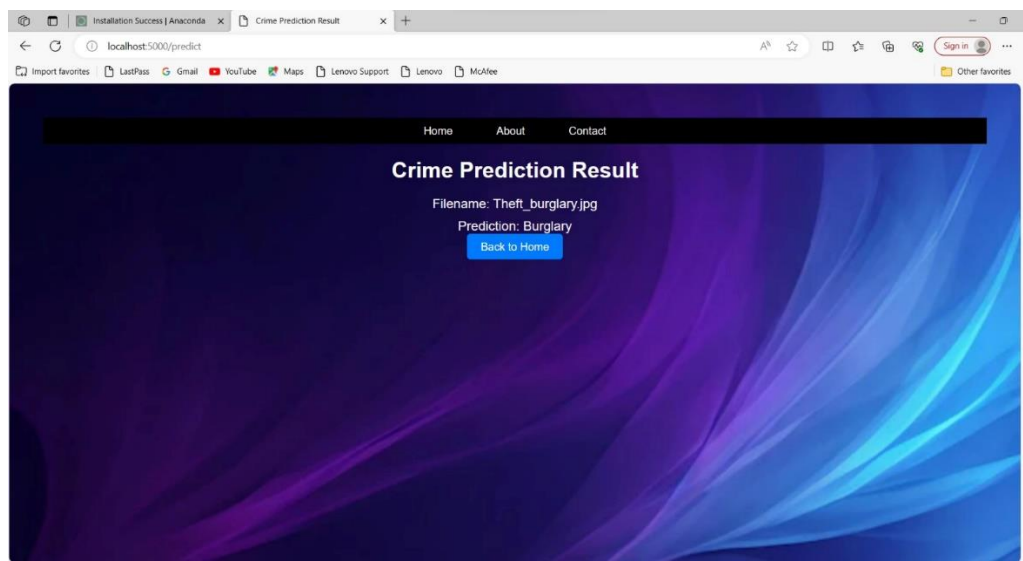
Index page:



PREDICTION PAGE:



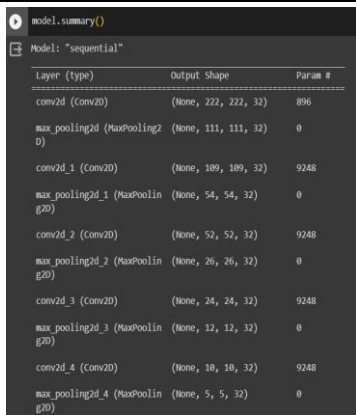
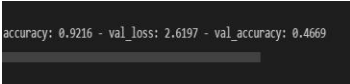
Predicting with various input images



Date	10 November 2022
Team ID	PNT2022TMID591975
Project Name	Project - Crime Vision: Advanced Crime Classification with Deep Learning
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	-	 <pre> model.summary() Model: "sequential" Layer (type) Output Shape Param # ----- conv2d (Conv2D) (None, 222, 222, 32) 896 max_pooling2d (MaxPooling2D) (None, 111, 111, 32) 0 conv2d_1 (Conv2D) (None, 109, 109, 32) 9248 max_pooling2d_1 (MaxPooling2D) (None, 54, 54, 32) 0 conv2d_2 (Conv2D) (None, 52, 52, 32) 9248 max_pooling2d_2 (MaxPooling2D) (None, 26, 26, 32) 0 conv2d_3 (Conv2D) (None, 24, 24, 32) 9248 max_pooling2d_3 (MaxPooling2D) (None, 12, 12, 32) 0 conv2d_4 (Conv2D) (None, 10, 10, 32) 9248 max_pooling2d_4 (MaxPooling2D) (None, 5, 5, 32) 0 max_pooling2d_3 (MaxPooling2D) (None, 12, 12, 32) 0 conv2d_4 (Conv2D) (None, 10, 10, 32) 9248 max_pooling2d_4 (MaxPooling2D) (None, 5, 5, 32) 0 flatten (Flatten) (None, 800) 0 dense (Dense) (None, 150) 120150 dense_1 (Dense) (None, 14) 2114 Total params: 160152 (625.59 KB) Trainable params: 160152 (625.59 KB) Non-trainable params: 0 (0.00 Byte) </pre>
2.	Accuracy	Training Accuracy – 0.921 Validation Accuracy – 0.46	 <pre> accuracy: 0.9216 - val_loss: 2.6197 - val_accuracy: 0.4669 </pre>
3.	Confidence Score (Only Yolo Projects)	Class Detected - NA Confidence Score - NA	NA

Date	10 November 2022
Team ID	PNT2022TMID591975
Project Name	Project - Crime Vision: Advanced Crime Classification with Deep Learning
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

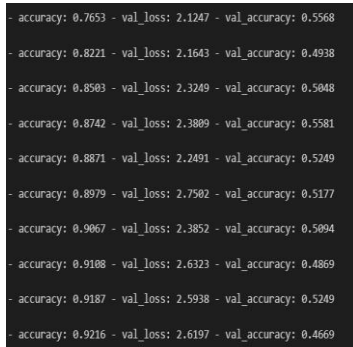

S.No.	Parameter	Screenshot / Values
1.	Dashboard design	No of Visualizations / Graphs - 0
2.	Data Responsiveness	<p>1. Data storage: The location and organization of data storage can significantly impact its responsiveness. Efficient data storage systems, such as data warehouses and data lakes, can quickly retrieve and analyze large datasets. Data access: The ease of accessing data is another crucial factor. Data should be accessible through a variety of methods, including web interfaces, APIs, and data visualization tools.</p> <p>2. Data processing: Data processing pipelines should be optimized to minimize latency and ensure timely data delivery. This may involve using parallel processing, caching, and other techniques.</p> <p>3. Data analysis: Data analysis tools should be easily accessible and user-friendly to allow for quick and efficient analysis of data. They should also be able to handle large datasets and provide insights in real-time.</p>
3.	Amount Data to Rendered (DB2 Metrics)	<p>1.The type of queries that are being executed</p> <p>2.The complexity of the queries</p> <p>3.The amount of data that is being returned by the queries</p> <p>4.The efficiency of the database server</p>
4.	Utilization of Data Filters	NA
5.	Effective User Story	No of Scene Added - 100

6.	Descriptive Reports	No of Visualizations / Graphs - 0
----	---------------------	-----------------------------------

Date	9 November 2022
Team ID	PNT2022TMID591975
Project Name	Project - Crime Vision: Advanced Crime Classification With Deep Learning
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

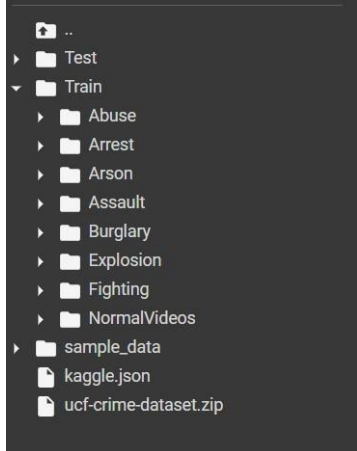
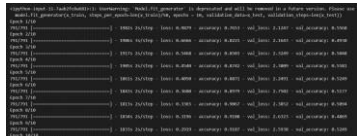
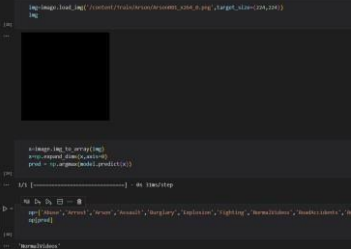
S.No.	Parameter	Values	Screenshot
1.	Metrics	Regression Model: MAE - NA, MSE -NA , RMSE - NA, R2 score - NA Classification Model: Confusion Matrix - , Accuray Score- & Classification Report -	
2.	Tune the Model	Hyperparameter Tuning - Validation Method - CNN	

Project Development Phase Model Performance Test

Date	9 November 2023
Team ID	PNT2022TMID591975
Project Name	Project – Crime Vision: Advanced Crime Classification with Deep Learning
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information when working for VAPT testing for a target .

S.No.	Parameter	Values	Screenshot
1.	Information gathering	Footprinting - Reconicessines -	
2.	Scanning the target	Scanning info - Risk factors -	
3.	Gaining access	Access process - Vulnerability found -	
4	Maintaining access - Automation (AI implementation)	AI tools used - Automation implemented -	NA

5	Covering Tracks & Report	<p>Vulnerability risk factors -</p> <p>VAPT report –</p> <p>Software Versions: Used Google Colab Software.</p> <p>Network Configuration: Internet-facing</p> <p>Client Requirements or Constraints: The Photos should be uploaded and our Model will predict the type of the crime.</p>	NA
---	--------------------------	---	----

APPENDIX:

SOURCE CODE:

App.py code:

```
from flask import Flask, render_template, request, url_for
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import io

app = Flask(__name__)

# Load the trained model
model = load_model('crime.h5')

# Define the list of crime categories
crime_categories = ['Abuse', 'Arrest', 'Arson', 'Assault', 'Burglary', 'Explosion',
'Fighting', 'NormalVideos', 'RoadAccidents', 'Shooting', 'Shoplifting', 'Robbery',
'Stealing', 'Vandalism']

# Define the upload folder
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['STATIC_FOLDER'] = 'static' # Add this line for serving static files

# Define allowed extensions for uploaded files
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

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

@app.route('/predict', methods=['POST'])
def predict():
    # Check if the POST request has the file part
```

```

if 'file' not in request.files:
    return render_template('index.html', error='No file part')

file = request.files['file']

# If the user does not select a file, submit an empty part without filename
if file.filename == "":
    return render_template('index.html', error='No selected file')

# If the file is allowed and properly uploaded
if file and allowed_file(file.filename):
    # Make prediction
    prediction = make_prediction(file)

    return render_template('result.html', filename=file.filename,
prediction=prediction)

else:
    return render_template('index.html', error='File type not allowed')

def make_prediction(file):
    try:
        # Load image using io.BytesIO
        img = image.load_img(io.BytesIO(file.read()), target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        pred = model.predict(x)
        predicted_class = crime_categories[np.argmax(pred)]
        return predicted_class
    except Exception as e:
        print(f"Error making prediction: {e}")
        return "Error making prediction"

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

```

index.html

```

<!DOCTYPE html>
<html lang="en">

```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>A Crime Classification using Deep Learning </title>
  <link rel="stylesheet" href="Final_Indexstyles.css">
</head>
<style>
  body {
    font-family: 'Courier New', Courier, monospace;
    text-align: center;
    color:white;
    margin: 0;
    padding: 0;
    background:
url("https://png.pngtree.com/thumb_back/fh260/background/20210604/pngtree-network-cable-abstract-digital-technology-background-image_724746.jpg") no-repeat center center fixed;
    background-size: cover;
  }
.navbar {
  background-color:#800000;
  color: #fff;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 20px;
}

.logo {
  font-size: 24px;
  margin: 0;
}

nav ul {
  list-style: none;
  padding: 0;
  display: flex;
}

nav li {
```



```
    margin: 0 10px;
}

nav a {
    text-decoration: none;
    color: #fff;
}

.content {
    display: flex;
    margin: 20px;
}

.description {
    width: 50%;
    padding-right: 20px;
    text-align: left; /* Align text to the left */
}

.crime-picture {
    width: 50%;
}

.crime-picture img {
    max-width: 100%;
}

.container {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.title {
    font-size: 2em;
    margin-bottom: 20px;
    color: #333;
}
```

```
}
```

```
.upload-form {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
}
```

```
.file-label {  
  font-size: 1.2em;  
  margin-bottom: 10px;  
}
```

```
input[type="file"] {  
  padding: 10px;  
  margin-bottom: 20px;  
  border: 1px solid #ccc;  
  border-radius: 4px;  
}
```

```
.predict-button {  
  padding: 10px 20px;  
  font-size: 1em;  
  background-color: #007bff;  
  color: #fff;  
  border: none;  
  border-radius: 4px;  
  cursor: pointer;  
}
```

```
.predict-button:hover {  
  background-color: #0056b3;  
}
```

```
.error-message {  
  color: #ff0000;  
  font-size: 1.2em;  
  margin-top: 10px;  
}
```

```
</style>
<body>
  <header class="navbar">
    <h1 class="logo">Crime Detector</h1>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>

  <main class="content">
    <section class="description">
      <h2>PROJECT DESCRIPTION</h2>
      <p>Crime identification using deep learning is a technique that involves applying deep learning techniques, specifically deep learning, to analyze images and video footage of crime scenes or incidents and identify and classify different types of crimes. Deep learning involves training neural networks on large amounts of data to recognize patterns and make predictions or decisions. By using deep learning, it is possible to analyze images and video footage of crime scenes or incidents and classify different types of crimes based on the type of activity depicted in the images. This can be useful in a variety of criminal justice and law enforcement contexts, including crime scene investigation, forensic analysis, and surveillance. Deep learning algorithms can be trained to recognize patterns and features in images and video that are relevant to identifying different types of crimes. They can also be used to analyze large amounts of data, such as surveillance footage, to identify trends and patterns in crime data. This can allow law enforcement agencies to develop strategies and interventions to prevent crime.</p>

      <h1>Crime Prediction</h1>

      <form action="/predict" method="post" enctype="multipart/form-data" class="upload-form">
        <label for="file" class="file-label">Upload Image:</label>
        <input type="file" id="file" name="file" accept=".png, .jpg, .jpeg, .gif" required>
        <button type="submit" class="predict-button">Predict</button>
```

```

</form>

{% if error %}
  <p class="error-message">{{ error }}</p>
{% endif %}
</section>

<section class="crime-picture">
  
</section>

</main>

</body>
</html>

```

Result.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Crime Prediction Result</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin: 50px;
      background-image: url("https://img.freepik.com/premium-
photo/abstract-background-images-wallpaper-ai-generated_643360-
21667.jpg");
      background-size: cover;
      color: white; /* Set default text color to white */
    }
    nav {
      background-color: #000;
      color: #fff;

```

```
padding: 10px 0;
margin-bottom: 20px;
}

nav a {
  color: #fff;
  text-decoration: none;
  padding: 10px 20px;
  margin: 0 10px;
}

nav a:hover {
  background-color: #333;
}

h1 {
  font-size: 2em;
  margin-bottom: 20px;
}

p {
  font-size: 1.2em;
  margin: 10px 0;
}

a.button {
  text-decoration: none;
  background-color: #007BFF;
  color: #FFFFFF;
  padding: 10px 20px;
  font-size: 1em;
  border-radius: 5px;
}

a.button:hover {
  background-color: #0056b3;
}
</style>
</head>
<body>
```

```
<nav>
  <a href="/">Home</a>
  <a href="#">About</a>
  <a href="#">Contact</a>
</nav>

<h1>Crime Prediction Result</h1>

<p>Filename: {{ filename }}</p>
<p>Prediction: {{ prediction }}</p>

  <a class="button" href="/">Back to Home</a>
</body>
</html>
```

GITHUB & PROJECT DEMO LINK:

<https://github.com/smartinternz02/Sl-GuidedProject-612389-1699422569>

PROJECT DEMO LINK:

https://drive.google.com/file/d/1is8dxX-tNjyAQ0ot_DkqnZLY5jfRw-If/view?usp=sharing