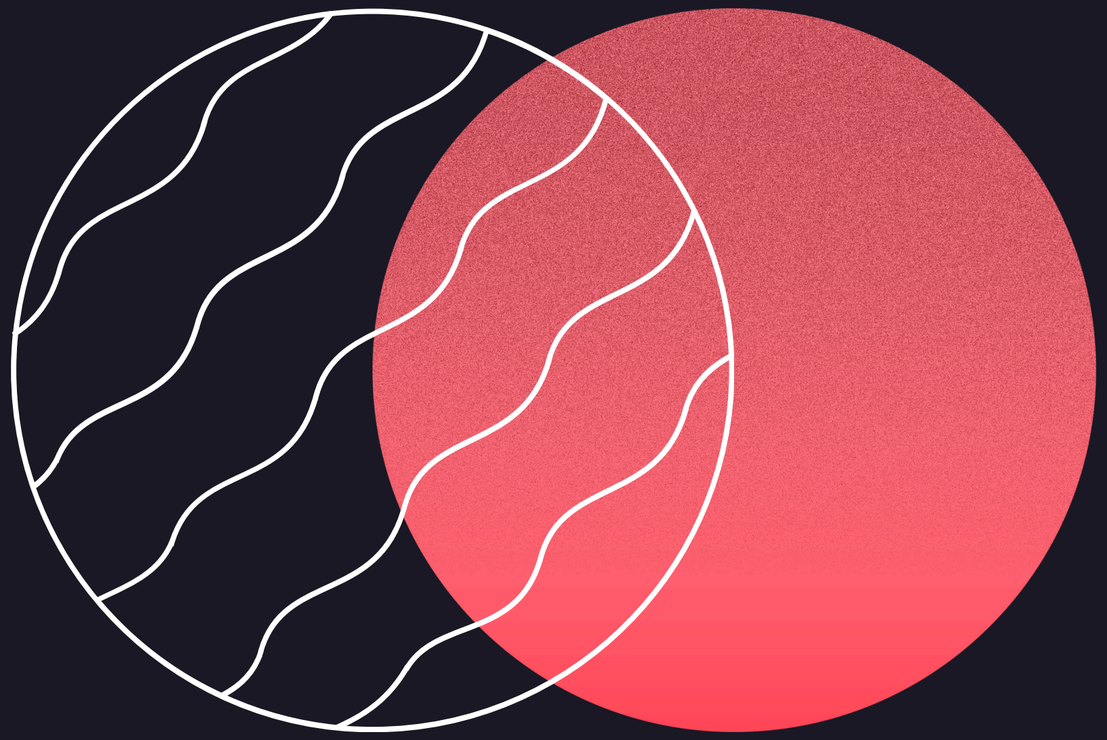
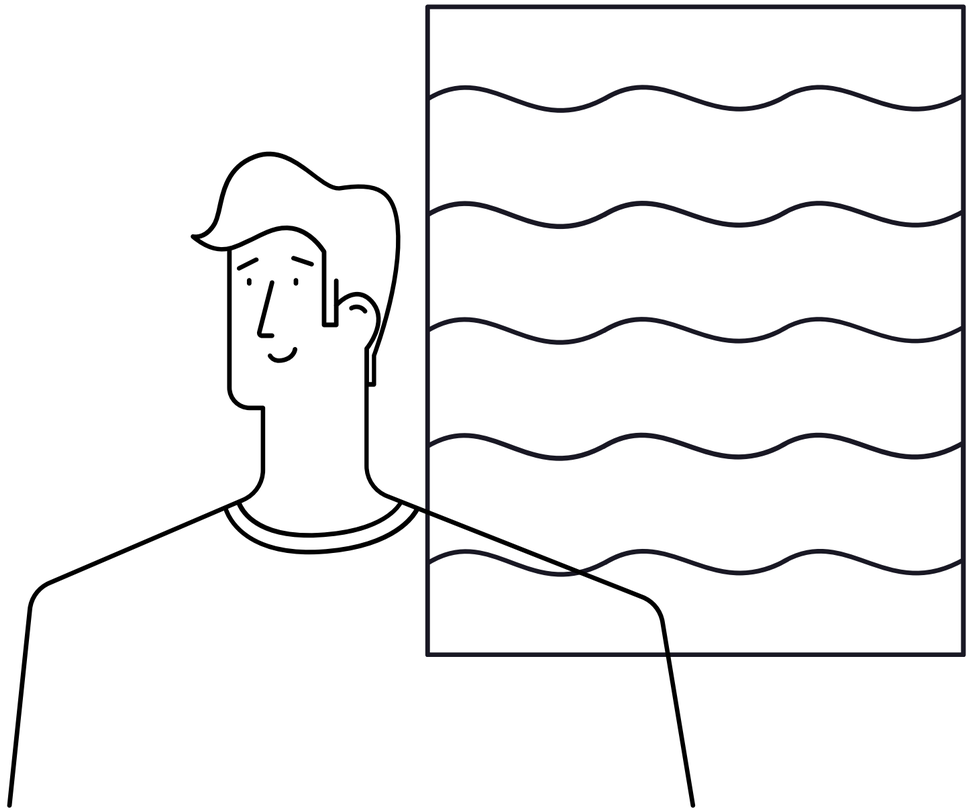


Traffic Density Classification Using CNN

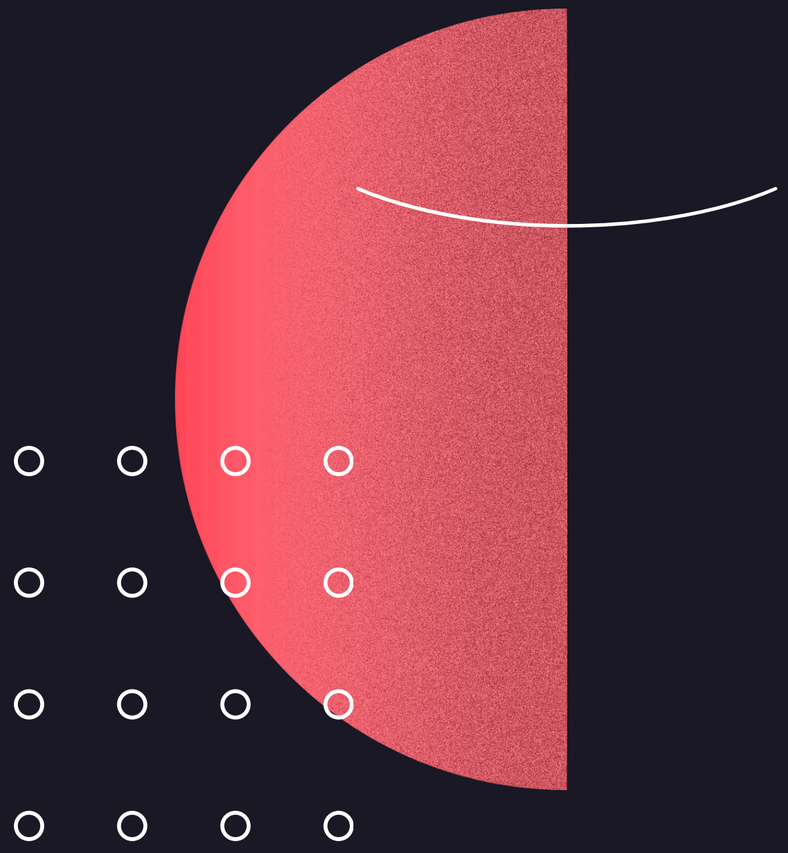
Tuwaiq Academy

Tuwaiq MiniProject

Maj. Abdulrahman Hassan Baluaid	Preprocessing, Model Selection, and implementation, Presentation.
Hassan Abid	Data Collection and management, Debugging, reporting.
Abdulaziz Alsidias	Implementation with Hypertunning and training,Debugging
Hamad Binnumay(evaluation)	Evalutation of the model

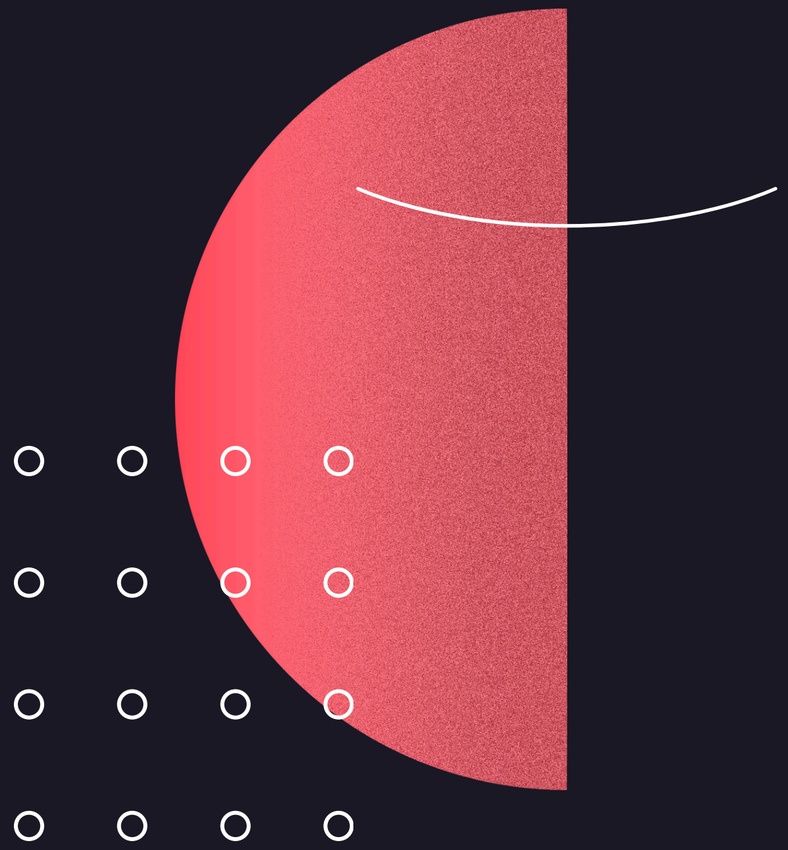


Problem statement



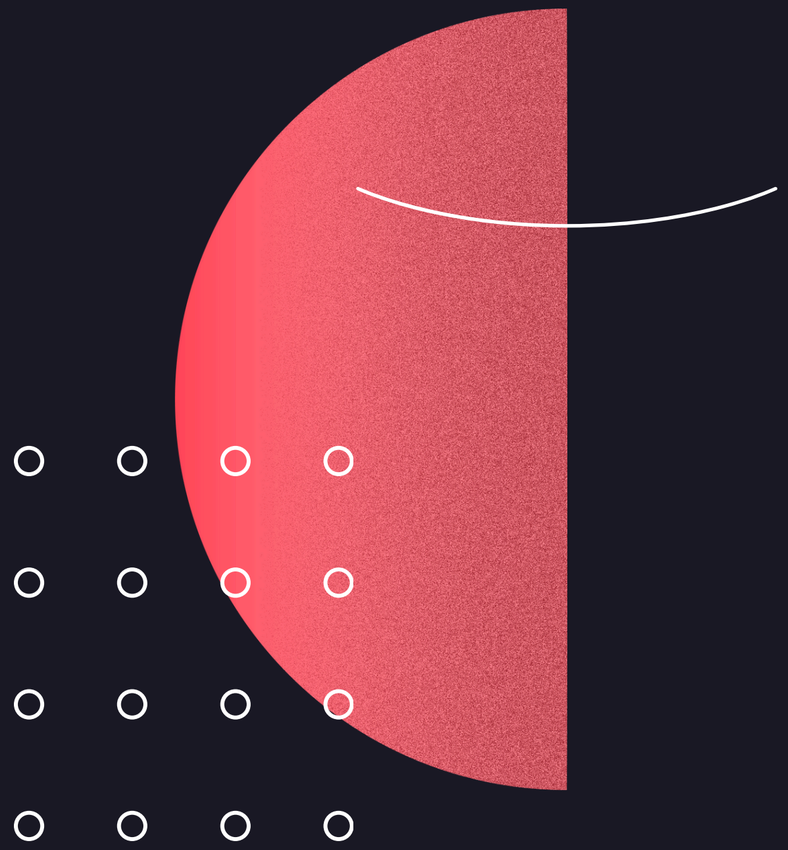
Urban traffic management relies on real-time analysis of traffic conditions to reduce congestion and improve safety. This project aims to develop a Convolutional Neural Network (CNN) model to classify traffic density levels (low, medium, high, very high) from camera images. The model will assist in optimizing traffic flow and decision-making, enhancing overall urban mobility.

Data Collection



- Data source: Kaggle (https://www.kaggle.com/datasets/rahat52/traffic-density-singapore)
- Images represent traffic density in different areas.
- Labels include: Empty, Low, Medium, High, Traffic Jam.

MongoDB connection



The project begins by connecting to a MongoDB Atlas database using the PyMongo library. The connection string (uri) allows secure access to the database, which stores the image dataset.

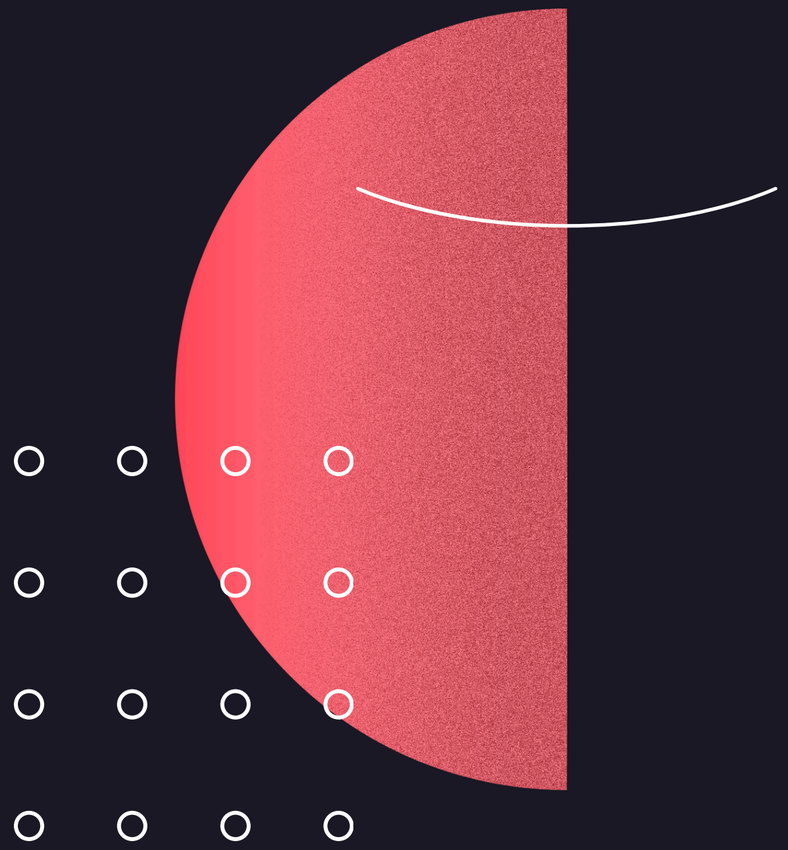
```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

uri = "mongodb+srv://hssanabid7:ez1VpZw6TXw38de2@cluster0.os8e2ax.mongodb.net

# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))

# Send a ping to confirm a successful connection
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

Data uploading and retrieval



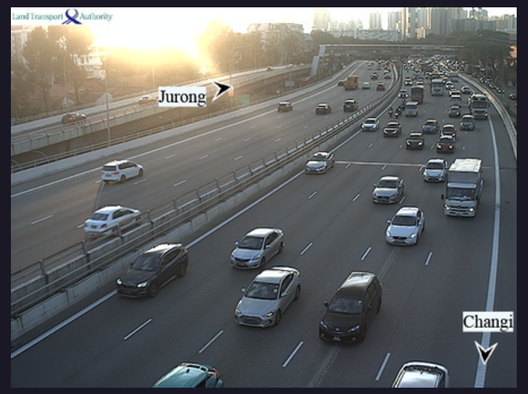
The dataset is stored in GridFS, a MongoDB specification for storing large files. Images are retrieved from the database and saved locally into a structured directory format suitable for training, validation, and testing.

```
db = client['traffic_density']
fs = gridfs.GridFS(db)
output_root_dir = '/content/Final Dataset'
os.makedirs(output_root_dir, exist_ok=True)
for grid_out in fs.find():
    filename = grid_out.filename
    main_folder = grid_out.main_folder
    sub_folder = grid_out.sub_folder
    output_dir = os.path.join(output_root_dir, main_folder, sub_folder)
    os.makedirs(output_dir, exist_ok=True)
    output_path = os.path.join(output_dir, filename)
    with open(output_path, 'wb') as f:
        f.write(grid_out.read())
        print(f'Saved {filename} to {output_path}')
```

DATA DETAILS

Dataset Type: Classification

Data Type: Images

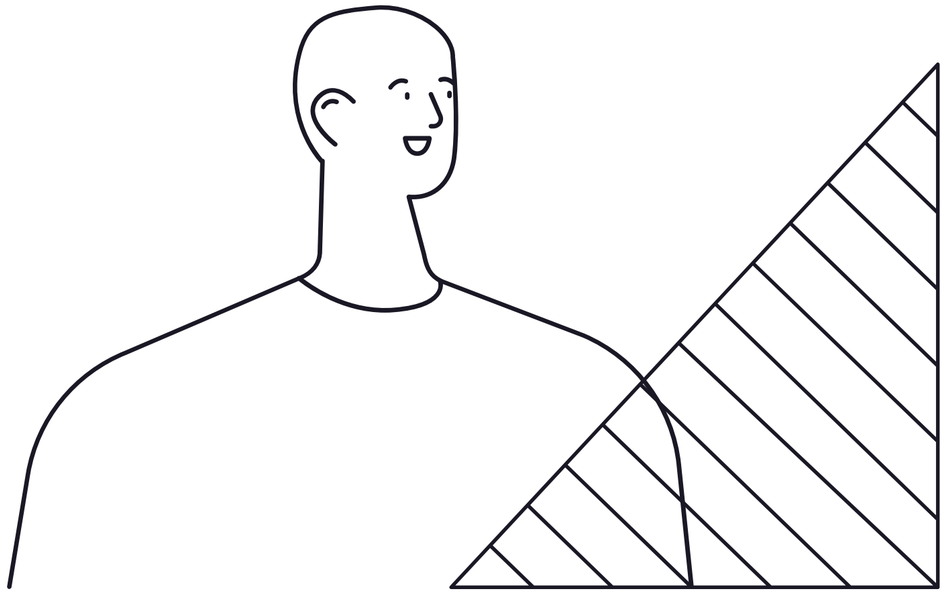


Preprocessing

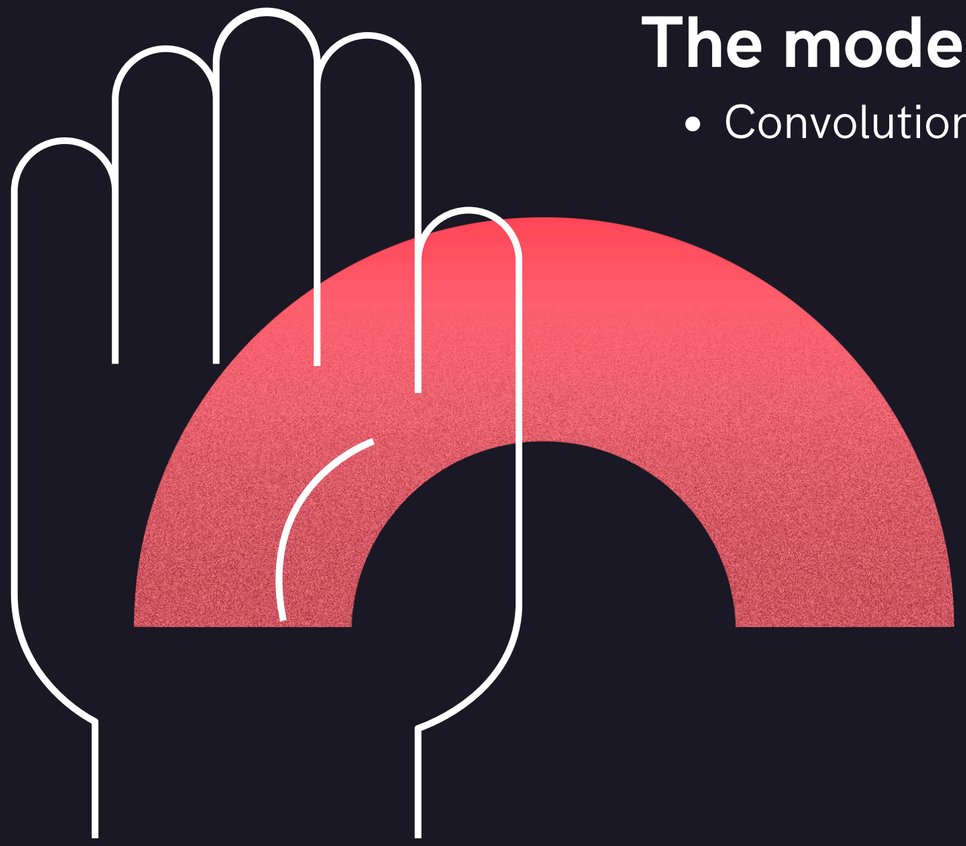
We started with a CNN simple model and we saw that we don’t need to increase the samples as the accuracy was good,
then we started increasing the complexity and increasing the layer to get a good accuracy.

- Rescaling: Adjusts pixel values from 0-255 to a 0-1 range to normalize the data and improve model performance.

DATA PREPARATION SUMMARY



MODELS TRAINING



The models that we used:

- Convolutional Neural Network

WE USED A CNN BECAUSE OF ITS SUPERIOR PERFORMANCE IN IMAGE CLASSIFICATION TASKS AND ITS ABILITY TO AUTOMATICALLY EXTRACT RELEVANT FEATURES FROM COMPLEX TRAFFIC IMAGES, MAKING IT A PERFECT FIT FOR OUR PROJECT.

HYPERTUNNING OUR MODEL

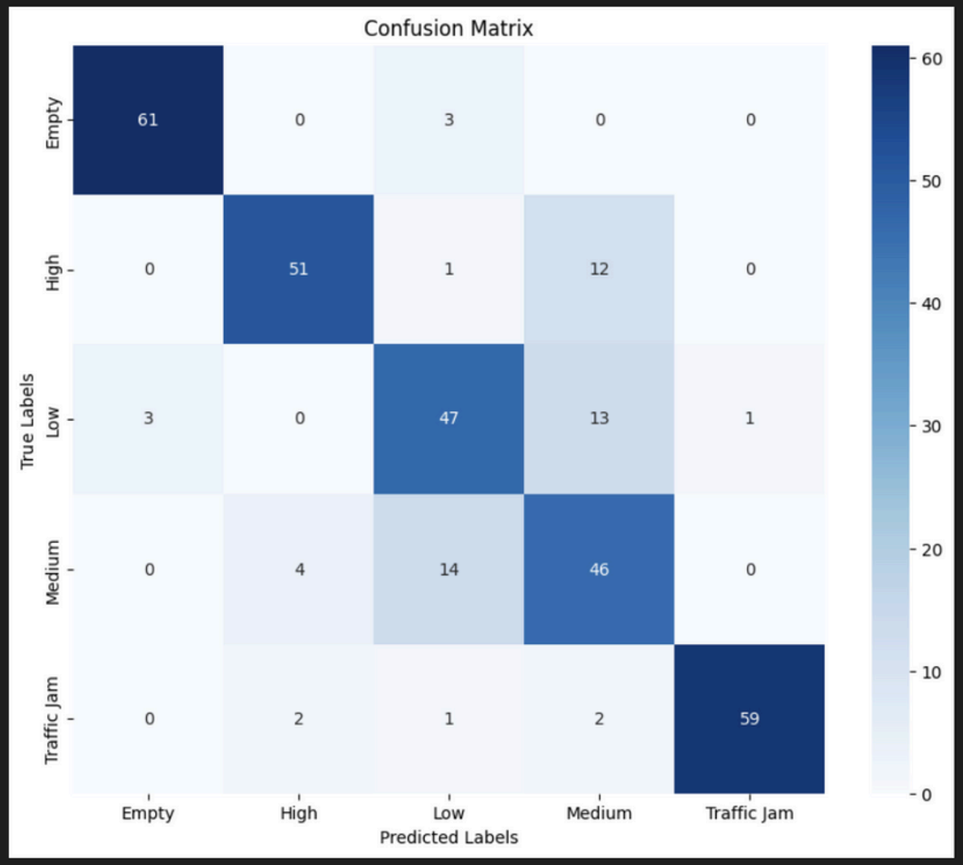
We did three things

1- We Define a tuner that uses RandomSearch

2- Then we found the best accuracy paramters

3- Lastly we saved our hypertunning parameters

MODEL RESULTS




```
Classification Report:
              precision    recall  f1-score   support

       Empty       0.95      0.95      0.95        64
        High       0.89      0.80      0.84        64
         Low       0.71      0.73      0.72        64
      Medium       0.63      0.72      0.67        64
 Traffic Jam       0.98      0.92      0.95        64

    accuracy                           0.82       320
   macro avg       0.83      0.82      0.83       320
weighted avg       0.83      0.82      0.83       320
```

```
model.evaluate(test_generator)
```

```
 3/16 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 0.7750 - loss: 0.5152
c:\Users\azo0o\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\tra
  self._warn_if_super_not_called()
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 61ms/step - accuracy: 0.8161 - loss: 0.4667
```

PREDICTIONS RESULTS

CHALLENGES

We faced challenges such as:

- Overfitting
- Underfitting
- Ensuring dataset's diversity
- Maintaining Labeling accuracy
- Dealing with varying image quality.

OUR SOLUTIONS

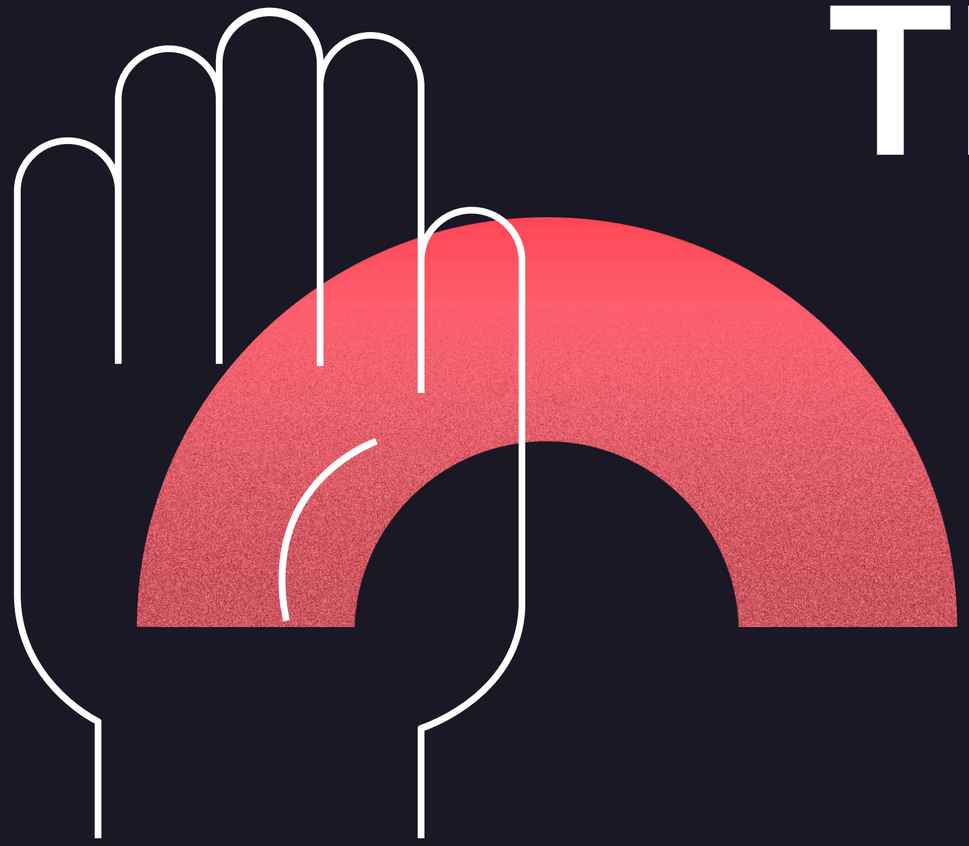
- For over and under fitting we used hyperparameter tuning to find the best model arch while adding dropout layers to prevent overfitting and early stopping
- Data Augmentation to increase data diversity and rescaling, resizing the images

REASONS FOR USING OUR MODEL

- Model Simplicity: We selected an architecture that balances complexity and computational efficiency, ensuring it could be trained within a reasonable timeframe while still achieving high accuracy.
- Precedent Success: The chosen architecture has been successfully applied in similar image classification tasks, providing a strong foundation for our traffic density classification model.

FUTURE WORK

- **Real-Time Prediction: Implementing the model in a real-time system to provide live traffic density updates, integrating the model with other models to get more valuable insights and results.**



Thank You

For Listening