**VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY**

**INTERNATIONAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING**

A blue and white logo

Description automatically generated

**TRAFFIC DENSITY DETECTION**

**FINAL REPORT**

Instructor: Dr. Nguyen Trung Ky

By

Nguyễn Bình Phương Huy – ITCSIU21189

Trần Thế Phong – ITCSIU21215

Đỗ Gia Thuỵ - ITCSIU21237

**ARTIFICIAL INTELLIGENCE**

**Our GitHub Link:**

<https://github.com/ChaoZiK/Traffic-Density-Detection>

Table of Contents

[1. Introduction 1](#_Toc186301990)

[1.1. Overview of the System 1](#_Toc186301991)

[1.1.1. Scope: 1](#_Toc186301992)

[1.1.2. Objective: 1](#_Toc186301993)

[1.1.3. Solution 1](#_Toc186301994)

[1.1.4. Data Collection and Preprocessing 2](#_Toc186301995)

[1.1.5. Model Architecture: 2](#_Toc186301996)

[1.2. Problem Statement 2](#_Toc186301997)

[1.2.1. Problem Analysis: 2](#_Toc186301998)

[1.2.2. Key problems in Traffic Density Detection: 2](#_Toc186301999)

[2. Related Work 3](#_Toc186302000)

[2.1. Existing work and solution 3](#_Toc186302001)

[2.1.1. Evaluation of the Application: 3](#_Toc186302002)

[2.1.2. Application Evaluation: 3](#_Toc186302003)

[2.1.3. Analysis and evaluation of the Tools and Techniques 3](#_Toc186302004)

[3. Methodology 5](#_Toc186302005)

[3.1. Proposed Method 5](#_Toc186302006)

[3.2. Data and tools 5](#_Toc186302007)

[3.3. System architecture 5](#_Toc186302008)

[3.4. Challenges and Assumptions 6](#_Toc186302009)

[4. Results and Evaluation 6](#_Toc186302010)

[4.1. Implementation and demo 6](#_Toc186302011)

[5. Evaluation and Conclusion 17](#_Toc186302012)

[5.1. Evaluation of the System: Accuracy of 81.25% 17](#_Toc186302013)

[5.2. Conclusion 17](#_Toc186302014)

# Table of Figures

[***Figure 1. Install Kaggle 6***](#_Toc186301810)

[***Figure 2. Upload dataset 6***](#_Toc186301811)

[***Figure 3. Data Loading and Processing 7***](#_Toc186301812)

[***Figure 4. Define CNN model 8***](#_Toc186301813)

[***Figure 5. Set up early stopping if the accuracy is not improved 9***](#_Toc186301814)

[***Figure 6. Prepare for training 10***](#_Toc186301815)

[***Figure 7. Training loop 11***](#_Toc186301816)

[***Figure 8. The results of training 12***](#_Toc186301817)

[***Figure 9. Plot the result 13***](#_Toc186301818)

[***Figure 10. Test model 14***](#_Toc186301819)

[***Figure 11. Reload model 14***](#_Toc186301820)

[***Figure 12. Run the model 15***](#_Toc186301821)

[***Figure 13. The result 16***](#_Toc186301822)

# Introduction

## Overview of the System

### Scope:

The scope of this project is to focus on creating a Traffic jams detection system based on Convolutional Neural Network (CNNs). This system classifies status into different categories based on the sparseness of the road. The system is created to adjust the number of vehicles stably which avoid the “overload” status.

The scope of the project:

**Dataset Consideration:** The system will rely on the large dataset about traffic congestion. The dataset will cover all types of traffic congestion in Kaggle.

**Technology platform:** The system is developed as a web-based platform. The primary technologies include Python, OpenCV, scikit-image, PyTorch, and matplotlib.

### Objective:

The primary objectives of this project are:

**Automated traffic congestion detection:** Develop an intelligence system which can automated detect traffic congestion to notice and adjust the number of vehicles which will go to the other roads in purpose reducing traffic congestion.

**High-performance model:** Utilize CNNs to build robust and high-performance traffic density detection models that can classify a large of traffic density.

**Accurate and Reliable Diagnosis:** Achieve high diagnostic accuracy to ensure that the system can be integrated in Maps-application to identify the status in a road. The system will provide confidence scores with predictions and make it easy to identify the density.

### Solution

The proposed solution to the problem of traffic density detection is deep learning-based image classification system that leverages CNNs to process and classify images of traffic congestion. Components of solution:

### Data Collection and Preprocessing

We will use a publicly available dataset in Kaggle, traffic\_dataset\_v1 which contains tested and trained images.

The images will undergo preprocessing steps (standard size, normalization of pixel values) to optimize the robustness of the model.

### Model Architecture:

A Convolutional Neural Network (CNN) will be used as the main of the classification system. We will use a pre-trained model which has been demonstrated to perform well on image classification tasks.

The model will have layers specifically to capture the status in the roads before analysis and identifying the density.

**Model Training and Evaluation:**

The model will be trained on a training set and evaluated using a test set of traffic density image.

## Problem Statement

### Problem Analysis:

The traffic conditions are one of the most important problems in the world, especially in urban areas. Every year, countries waste thousands of hours because of this problem. To solve the problem, this project has been created to adjust the number of vehicles, especially during rush hour. The system will notice the way which is having traffic jams, and the driver will receive that information to go to the low-density roads.

### Key problems in Traffic Density Detection:

**Choosing the best but not suitable way:** Driver, they always choose the shortest road because they think it is the best way to spend time most. However, there are too many vehicles that make the density increase and it will lead to traffic congestion.

**Density changes suddenly:** In some situations, the limited roads which is damaged or lockdown by some internal and external factors like festivals, limited roads or maintenance also causes density in a road decrease. If there is no notice before, it will lead to congestion.

# Related Work

## Existing work and solution

### Evaluation of the Application:

The Traffic Density Detection seeks to address the problems by automating the process of traffic density identification through IoT and machine learning. By providing the accurate and reliable dataset, this system can notice to driver timely and reduce the traffic congestion.

### Application Evaluation:

**Scalability**: The system is scalable and can handle large datasets. The more images are provided, the more accurate the system is. With large data, it will provide enough cases, and make the system get better knowledge.

**Practical Impact**: By offering an affordable and accessible solution, the system can significantly reduce wasted time for countries, ensuring that everyone who use it can get benefit. The driver can have more free time, and work effectively. Governments and companies also solve a dilemma.

### Analysis and evaluation of the Tools and Techniques

The heart of the traffic density detection system is computer vision and deep learning, specifically CNNs. This system has so many advantages like automatic feature extraction, highly accurate image recognition and classification, weight sharing, minimizing computation, use same knowledge across all image locations, ability to handle large dataset and hierarchical learning.

**Techniques**:

**The Convolutional Neutral Networks (CNNs)**: Used to classify density images into different categories. Their ability to automatically learn spatial hierarchies of features from images makes them ideal for tasks.

* These models have highly effective in tasks including image recognition, so it makes ideal in this project.

**Transfer Learning**: In this system, a training dataset is available divided from a large, general-purpose dataset. This approach reduces the amount of labeled data and speeds up the training process.

* This leads to classification being able to work at the low - cost and time required for training.

**Tools**:

**Python**: This programming language due to its widespread use in AI. It has a variety of library which help created system easily such as OpenCV, scikit-learn. Besides, it has simple and readable syntax, strong community support, platform independence, data analysis and visualization

* There are many advantages which can make completing project fast, effectively, and high accuracy.

**OpenCV**: short for Open-Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation.

* **OpenCV** is a great computer vision library when it comes to regular computer vision tasks and image processing. Its main advantages are speed, multiple language support, and a variety of image-processing functions that it implements.

Scikit-learn: One of the main benefits of Scikit-learn is its simplicity and consistency. Scikit-learn follows a clear and uniform API that makes it easy to use and understand. You can apply the same methods and conventions to different types of models and data, such as fit, predict, transform and score.

* Scikit-learn is a fantastic tool for exploring, transforming and classifying data. But it is optimized for learning algorithms.

**PyTorch**: PyTorch’s flexible and dynamic nature allows researchers to develop complex architectures.

* This framework maximizes both flexibility and performance for machine learning projects. Embrace its dynamic computation graph, explore its native optimization library, and leverage GPU acceleration to build efficient and effective deep learning models.

# Methodology

## Proposed Method

The methodology for developing the traffic density detection system involves designing a robust pipeline based on Convolutional Neural Networks (CNNs). The process includes data preparation, model design, training, and evaluation to ensure accurate and reliable detection of traffic congestion.

## Data and tools

The system relies on a dataset obtained from Kaggle's "traffic-density-Singapore" collection. The dataset includes categorized images representing varying levels of traffic density, providing a foundation for training and evaluation.

**Data Preprocessing**:

Resize all images to a consistent size (128 x 128 pixels) to ensure uniformity across the dataset.

Normalize pixel values to scale them between -1 and 1, enhancing the training stability.

**Tools and Frameworks:**

* **Programming Language**: Python.
* **Deep Learning Framework**: PyTorch.
* **Libraries**: OpenCV, matplotlib, and scikit-learn for preprocessing and visualization.

Apply data augmentation techniques, such as random horizontal flips, rotations, and crops, to improve model generalization and tackle dataset imbalances.

## System architecture

The system uses a CNN – based architecture tailored for image classification tasks. Key components include:

* **Convolutional Layers**: Extract spatial features from images.
* **Batch Normalization**: Standardize intermediate outputs to stabilize and accelerate training.
* **Pooling Layers**: Down-sample feature maps and reduce computational complexity.
* **Fully Connected Layers**: Combine features for final classification.

This architecture is designed to efficiently classify traffic density into multiple categories.

## Challenges and Assumptions

The project faces several challenges, including:

**Dataset Imbalance**: Traffic density categories may not be evenly distributed. To address this, oversampling of underrepresented classes and data augmentation are applied.

**Model Overfitting**: Implementing techniques such as dropout layers and early stopping ensures the model generalizes well to unseen data.

**Scalability**: The model must handle real-time traffic data efficiently. Future iterations will explore optimizations for inference speed.

# Results and Evaluation

## Implementation and demo

A screenshot of a computer

Description automatically generated

Figure 1. Install Kaggle

A screen shot of a computer

Description automatically generated

Figure 2. Upload dataset

A computer screen shot of a program

Description automatically generated

Figure 3. Data Loading and Processing

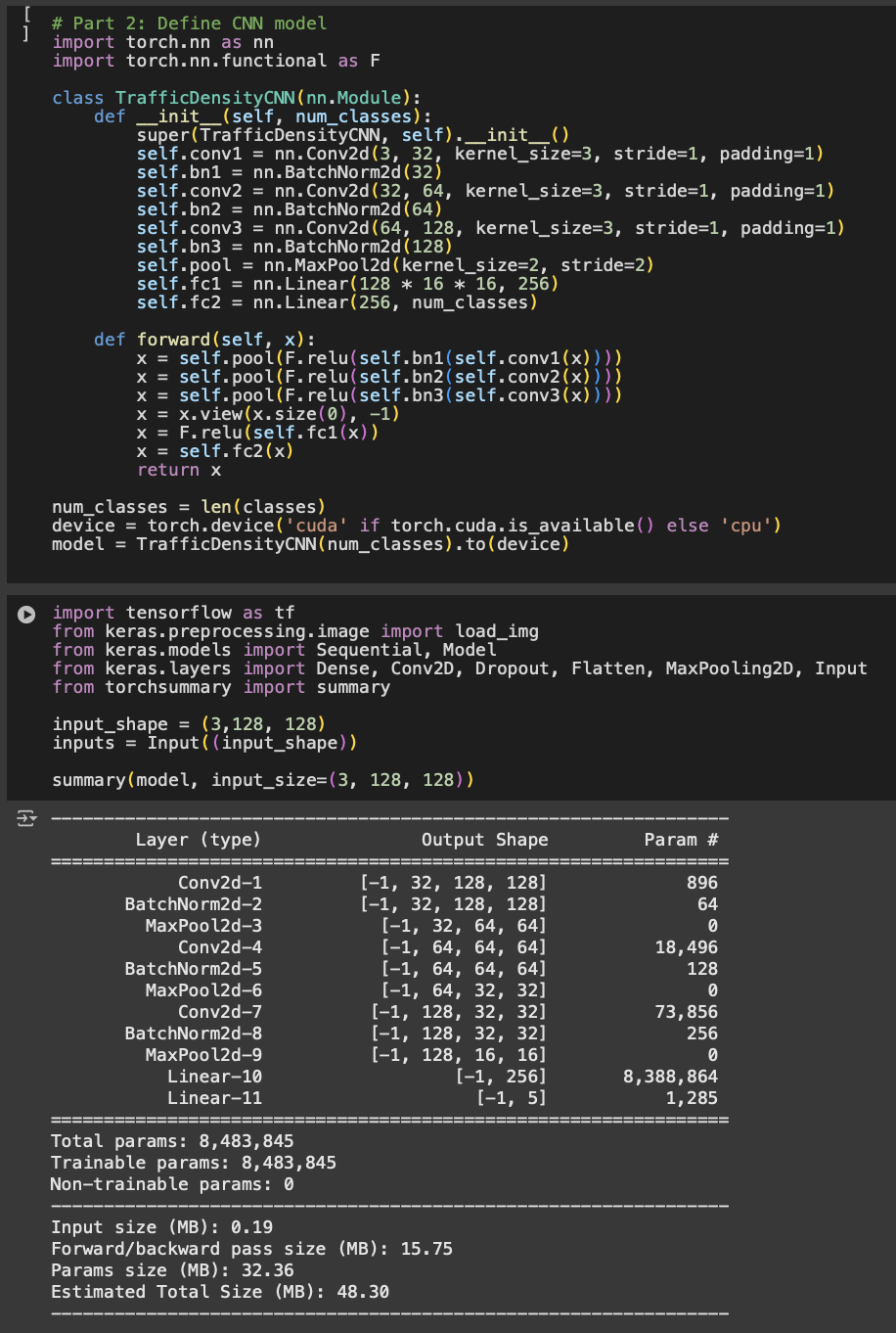


Figure 4. Define CNN model

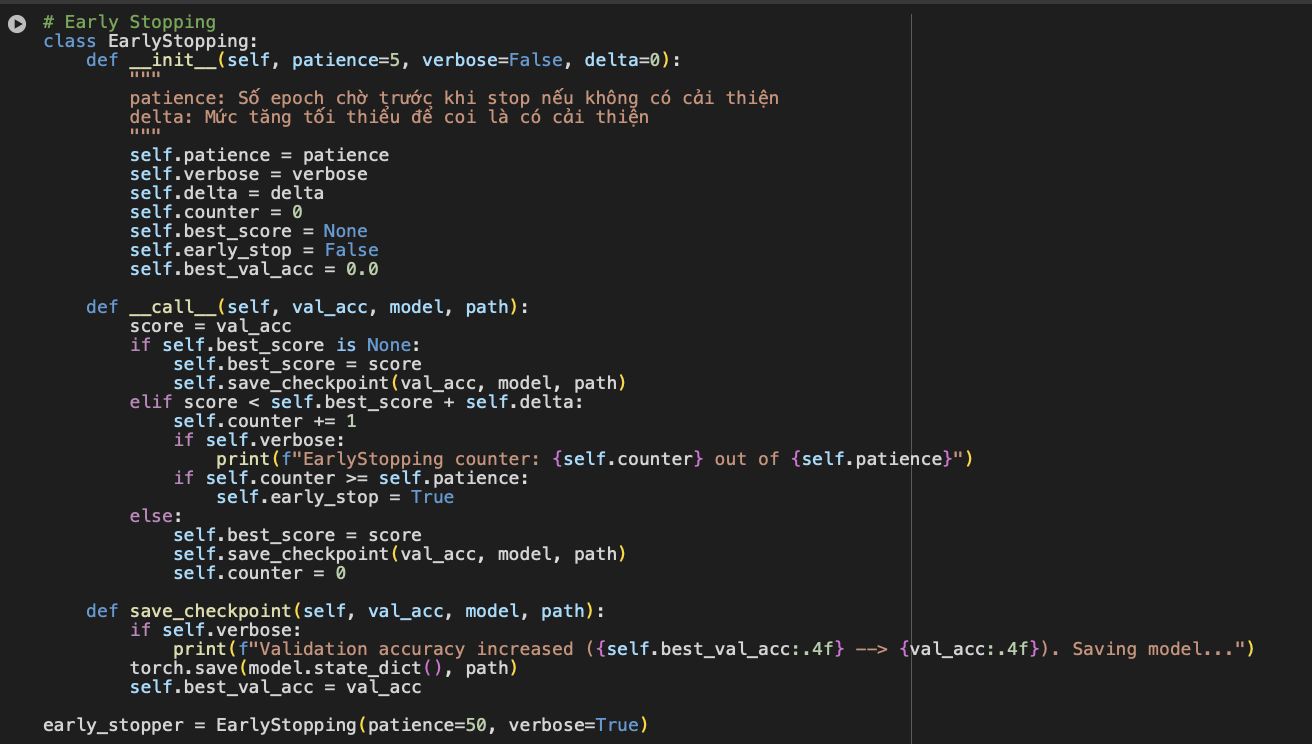


Figure 5. Set up early stopping if the accuracy is not improved

A screen shot of a computer program

Description automatically generated

Figure 6. Prepare for training

A screen shot of a computer program

Description automatically generated

Figure 7. Training loop

A screenshot of a computer

Description automatically generated

Figure 8. The results of training

A screenshot of a computer screen

Description automatically generated

Figure 9. Plot the result

A computer screen shot of a program code

Description automatically generated

Figure 10. Test model

A screenshot of a computer screen

Description automatically generated

Figure 11. Reload model

A computer screen shot of a program code

Description automatically generated

Figure 12. Run the model

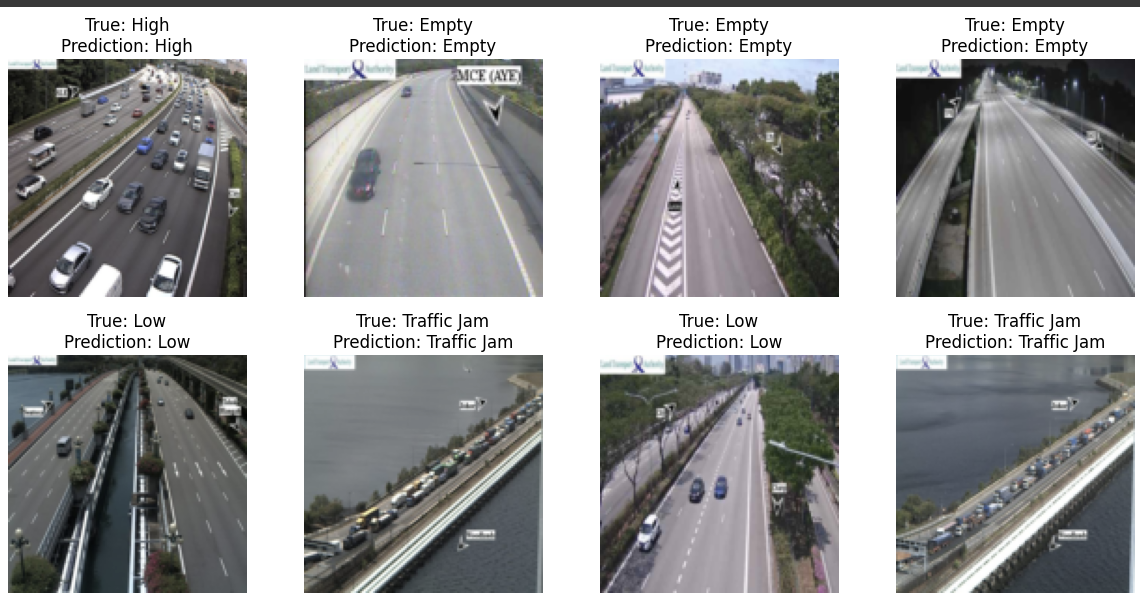


Figure 13. The result

# Evaluation and Conclusion

## Evaluation of the System: Accuracy of 81.25%

The accuracy of 81.25% indicates a moderate to string performance of the model on the given dataset. However, to reach the 90%, building or adjusting the models is the main method to improve the accuracy. Besides, the larger dataset is also necessary to get the higher result.

**The Overview**:

The Traffic Density Detection aims to detect the density in a road which will notice drivers to reduce congestion. The primary goal of this system is helping people save their time and work effectively for companies and government.

**Analysis of the Model Performance**

**Accuracy**: The model gets 81.25% accuracy on the test case. It shows that the models learn key features for density identification.

**Generalization**: Achieving 81.25% accuracy on the test case is indicative of good generalization to unseen data. The models have knowledge about density identification to give a good introduction.

## Conclusion

This achieved accurately shows that the model has a strong knowledge about traffic density, reflecting both generalization and the effective use of training models. However, there are still some problems that need to be improved:

Adjusting the models and using a better dataset to get higher accuracy.

Improve, apply and focus on precision, recall, and F1 score. Incorporating more diverse augmentation techniques and testing to ensure a robust relationship between theory and real-world problems.

The future work of this project is improving model performance, integrating with maps application, maintaining the reliable and predict the suitable way for driver early.