Project Report: Traffic Density Classification Using Convolutional Neural Networks and MongoDB Integration

1-Abdulziz Alsidias

2-Hassan Abid

3-Majed Abdulrahman Balubaid

4-hamed binnumay

1. Introduction

1.1 Project Overview

This project focuses on developing a convolutional neural network (CNN) model for classifying traffic density using image data. The dataset, structured in a hierarchical format within a MongoDB database, is retrieved, processed, and then used to train a deep learning model. The project aims to accurately classify images into five categories: Empty, Low, Medium, High, and Traffic Jam.

1.2 Objectives

The main objectives of this project include:
- Integrating MongoDB for efficient dataset management.
- Developing a CNN model for traffic density classification.
- Optimizing the model using hyperparameter tuning.
- Evaluating the model's performance and visualizing the results.

2. Data Handling and Preprocessing

2.1 MongoDB Integration

2.1.1 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.

2.1.2 File 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.

2.2 Directory Structure

2.2.1 Dataset Organization
The images are organized into three main folders: Training, Validation, and Testing.

2.2.2 Folder Hierarchy
Each of these folders has five subfolders corresponding to the traffic density classes: Empty, Low, Medium, High, and Traffic Jam.

2.3 Data Augmentation

2.3.1 Image Preprocessing with ImageDataGenerator
The TensorFlow ImageDataGenerator class is used to preprocess the images by rescaling pixel values to a range of [0, 1]. This step ensures that the model receives consistent data inputs.

3. Model Development

3.1Convolutional Neural Network (CNN) Architecture

3.1.1 Model Structure
The CNN model is built using the Keras and TensorFlowl libraries. The architecture includes several convolutional layers, followed by MaxPooling layers to reduce spatial dimensions, and dense layers for classification.

3.1.2 Hyperparameter Tuning
The model's hyperparameters, such as the number of filters and learning rate, are tuned using the Keras Tuner library. A Random Search approach is used to explore the hyperparameter space and find the best combination for maximizing validation accuracy.

4. Model Training and Evaluation

4.1 Training

4.1.1 Training Process
The model is trained using the training dataset, with validation performed after each epoch to monitor the model's performance.

4.1.2 Epochs and Validation

The training process includes 10 epochs, and the model's weights are updated based on the categorical cross-entropy loss function.

4.2 Evaluation

4.2.1 Testing Dataset Performance

After training, the model is evaluated on the testing dataset to assess its final accuracy and generalization ability.

4.2.2 Accuracy Metric

The accuracy metric is used as the primary measure of performance.

5. Predictions and Visualization

5.1 Random Image Predictions

5.1.1 Prediction Methodology

The model's predictive capabilities are demonstrated by selecting random samples from the testing dataset and plotting them alongside the predicted class labels.

5.1.2 Visualization of Predictions

A grid of 15 images (3 rows x 5 columns) is plotted, displaying both the model's predicted class and the true class for each image.

6. Conclusion

6.1 Summary of Results

This project successfully integrates MongoDB for dataset management and TensorFlow for deep learning, resulting in a CNN model capable of classifying traffic density with high accuracy.

6.2 Future Work and Applications

This approach can be extended to other image classification tasks with similar data structures, and can be integrated with various projects.

Contents