**GEORGE BROWN COLLEGE**

**APPLIED AI SOLUTIONS DEVELOPMENT (T431)**

****

**FULL STACK DATA SCIENCE SYSTEMS**

**GROUP 8 REPORT**

**‘TRAFFIC SIGN RECOGNITION PROJECT’**

Aditya Koshti - 101504567

Ayushi Patel - 101450798

Mahesh Dadheech - 101501168

Sakshi Sareen - 101471328

Shivani – 101504534

**Introduction**

Traffic Sign Recognition (TSR) is a vital component in the development of autonomous driving systems, where the ability to accurately detect and classify traffic signs in real-time is essential for ensuring road safety. With the increasing adoption of autonomous vehicles, a robust TSR system becomes crucial in preventing accidents and ensuring compliance with traffic regulations. This report outlines the development, deployment, and evaluation of a TSR system, built using deep learning techniques, containerized with Docker, and deployed on the cloud using Microsoft Azure.

**Problem Statement**

The challenge in traffic sign recognition lies in the accurate identification and interpretation of various traffic signs in real-time. This task is complicated by factors such as varying lighting conditions, occlusions, weather conditions, and the presence of non-standard or damaged signs. Misinterpretation of traffic signs can lead to accidents or traffic violations, highlighting the need for a reliable and robust TSR system.

**Methodology**

The project began with data collection using the German Traffic Sign Recognition Benchmark (GTSRB) dataset, a widely recognized benchmark in the field of computer vision. The dataset includes over 50,000 images of traffic signs, categorized into 43 different classes.

A Convolutional Neural Network (CNN) was chosen as the model architecture due to its proven effectiveness in image classification tasks. The CNN was trained to automatically extract features from the images, allowing it to classify traffic signs with a high degree of accuracy. Key preprocessing steps, such as data augmentation, were employed to enhance the model's ability to generalize across different environmental conditions.

**Model Deployment**

Once the model achieved satisfactory performance, the focus shifted to deployment. The deployment process involved several key steps:

* Transition from Notebook to Script: The initial model development was conducted in a Jupyter notebook. For deployment, the code was refactored into a Python script, following best practices for modularity and maintainability. The Flask framework was used to create a web application that allows users to upload images and receive real-time predictions.
* Dockerization: To ensure that the application could run consistently across different environments, it was containerized using Docker. A Dockerfile was created to define the application's environment, including the base image, dependencies, and instructions for running the Flask app. Containerization provides the added benefit of environment isolation, ensuring that the application behaves identically in development, testing, and production environments.
* Cloud Deployment: The Docker container was deployed on Microsoft Azure using Azure Container Instances (ACI). This service was chosen for its ease of use and scalability, allowing the application to be accessed publicly via a web interface. Azure’s monitoring tools were integrated to track the application's performance, including metrics such as CPU usage, memory usage, and response time.

**Evaluation and Benchmarking**

The model was evaluated using several performance metrics. The final CNN model achieved an accuracy of 75.8% on the test set, significantly outperforming traditional image processing methods and comparable to state-of-the-art deep learning models. The inference time they averaged 15ms per image, making it suitable for real-time applications in autonomous driving systems.

To further validate the model’s performance, a confusion matrix was analyzed to identify any potential misclassifications. The model showed strong performance across all major traffic sign categories, with minor misclassifications occurring between visually similar signs (e.g., speed limit signs with different limits).

**Challenges and Solutions**

During the deployment phase, several challenges were encountered. Configuring the environment to run TensorFlow within a Docker container required resolving dependency conflicts and ensuring all necessary packages were installed. Additionally, optimizing the model for cloud deployment was crucial to balancing model size and inference time. These challenges were addressed through careful dependency management and the use of post-training quantization to reduce the model size without sacrificing accuracy.

Another significant challenge was ensuring real-time performance in the cloud environment. Azure’s monitoring tools played a critical role in identifying bottlenecks and optimizing the application's performance.

**Conclusion**

The Traffic Sign Recognition project successfully demonstrates the application of deep learning techniques in a real-world scenario. The project not only highlights the importance of accurate traffic sign recognition for autonomous driving systems but also showcases the end-to-end process of developing, deploying, and optimizing a machine learning model in a cloud environment. The use of Docker and Azure for deployment ensures that the system is scalable, reliable, and accessible, making it a viable solution for integration into modern autonomous driving technologies.