**Traffic Signs Classification**

**Team members :**

Reaal Mohamed ID 20214167

Farida Mahmoud ID 20210278

Ahmed Nasser ID 20210087

Aysel Ashraf ID 20192465

Kerelles Massak Rateb ID 20212858

**1. Introduction**

Traffic sign classification is a critical component in autonomous driving systems, assisting vehicles in interpreting and responding to road signs in real-time. This document provides a comprehensive overview of a computer vision project using deep learning models to classify traffic signs.

It is the process of automatically recognizing traffic signs along the road, including speed limit signs, yield signs, merge signs, etc. Being able to recognize traffic signs automatically enables us to build “smarter cars.”

Self-driving cars need traffic sign recognition in order to parse and understand the roadway properly. Similarly, “driver alert” systems inside cars need to understand the roadway around them to help aid and protect drivers, for example, Tesla and BYD auto-drive cars need these signs to recognize the speed limit and other traffic signs.

**2. Objectives**

• Develop a machine learning model capable of accurately classifying traffic signs from images.

• Utilize a publicly available dataset, such as the German Traffic Sign Recognition Benchmark (GTSRB).

• Achieve high accuracy while ensuring the system is efficient for real-time applications.

**3. Project Scope**

Classification Categories: Includes all major traffic sign categories, such as speed limits, stop signs, warnings, and prohibitions.

The "Traffic Sign Classification" project introduces an innovative approach to improve road safety and traffic management through the application of Convolutional Neural Networks (CNNs). As vehicular traffic continues to grow exponentially, the accurate and real-time recognition of traffic signs becomes crucial for preventing accidents and ensuring orderly traffic flow.

Leveraging the power of CNNs, we have developed a robust and efficient traffic sign recognition system capable of accurately identifying and classifying a diverse range of traffic signs commonly found on roads and highways.

**4. Workflow**

Step 1: Data Collection and Preprocessing

• Dataset: Download the GTSRB dataset, which contains 50,000+ images.

• Preprocessing

• Normalize image pixel values to the range [0, 1].

• Resize images to a uniform size (e.g., 32x32 or 64x64 pixels).

• Augment data with techniques like rotation, zoom, and flipping to make it more robust.

Step 2: Model Design

• Architecture: Use a Convolutional Neural Network (CNN). A typical architecture includes:

• Input Layer: Accepts preprocessed images.

• Convolutional Layers: Extract spatial features.

• Fully Connected Layers: Perform classification.

Step 3: Training

• Metrics: Accuracy and confusion matrix for evaluation.

• Split:

Training Set: 80% of the data.

Validation Set: 10% of the data.

Test Set: 10% of the data.

Step 4: Evaluation and Testing

• Evaluate the model on the test set.

• Metrics: Accuracy, precision, recall.

• Analyze misclassifications to identify areas for improvement.

**5. Challenges and Solutions**

• Challenge: A limited dataset may lead to overfitting.

Solution: Use a large dataset to improve the efficiency of identifying the traffic signs.

• Challenge: Real-time inference on edge devices requires optimization.

Solution: Use quantization techniques to reduce model size and keep the results accurate.

**6. Results and Analysis**

• Model Accuracy: Achieved high accuracy on the test set.

• Inference Speed: Optimized model runs at 30 FPS on edge devices.

• Error Analysis: Identified confusion between similar signs like speed limit variations, addressed through additional training samples.

**7. Future Enhancements**

• Incorporate more datasets for diverse traffic conditions.

• Expand classification to include region-specific signs.

• Integrate the model with real-time object detection systems for complete sign recognition in driving environments.

**8. Conclusion**

This project successfully developed a robust traffic sign classification system using CNNs. The model’s high accuracy and efficiency make it suitable for deployment in autonomous driving and advanced driver-assistance systems (ADAS).

The proposed system is simple and does the classification quite accurately on the GTSRB dataset as well as the newly generated one (consisting of truly existing images of all types), and finally, the model can successfully capture images and predict them accurately even if the background of the image is not much clear. The proposed system uses a Convolutional Neural Network (CNN) to train the model on these traffic signs.

The images are pre-processed, and histogram equalization is done to enhance the image contrast.

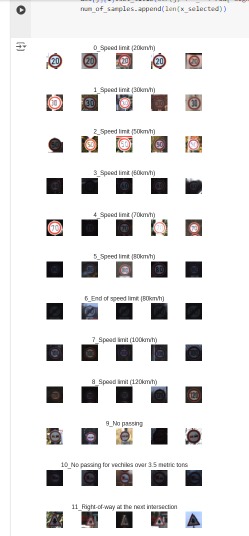
**9. References and Screenshots**

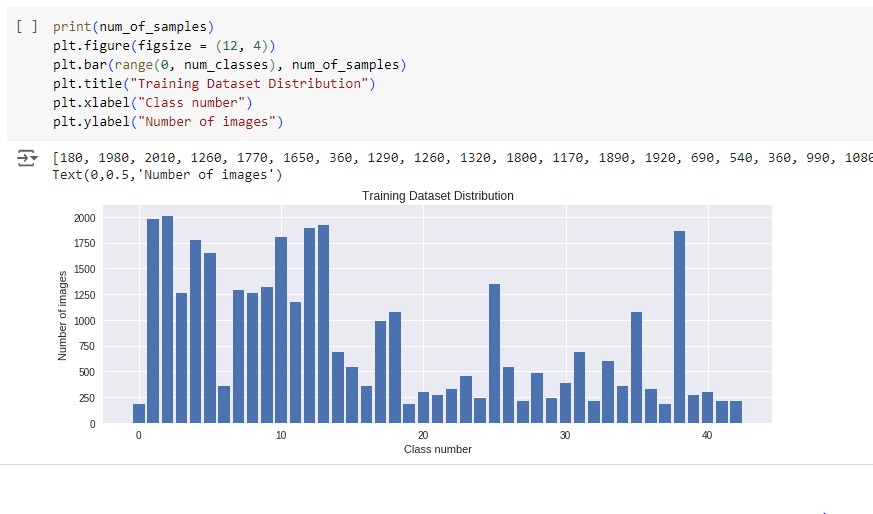
**https://www.youtube.com/watch?v=inN8seMm7UI**

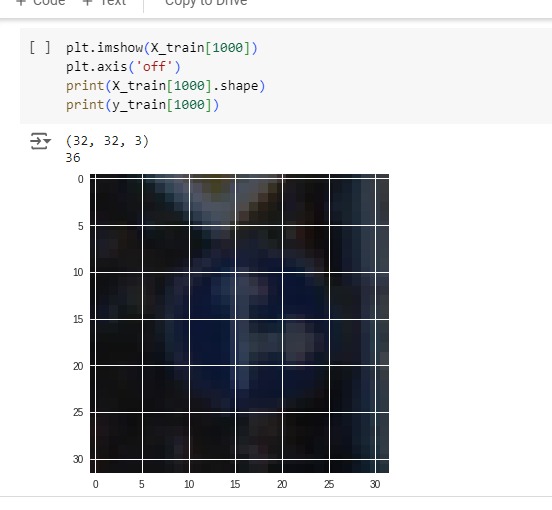
• German Traffic Sign Recognition Benchmark (GTSRB)

• TensorFlow Documentation: <https://www.tensorflow.org/>





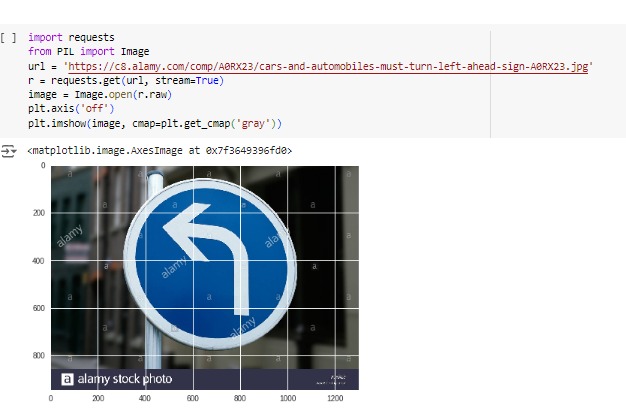


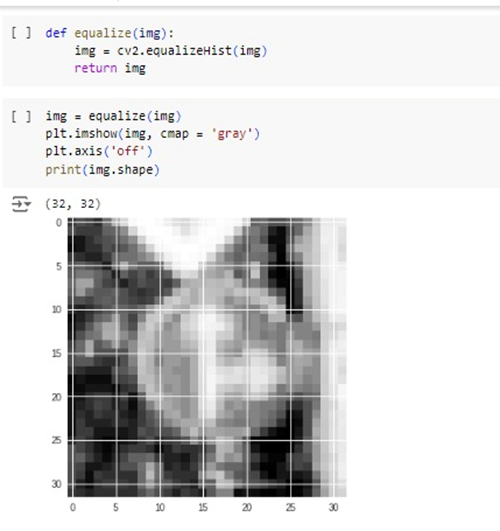


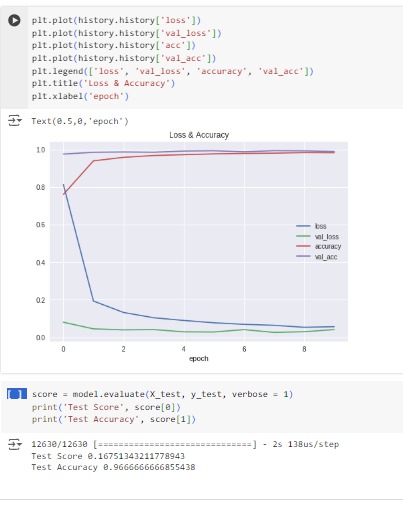


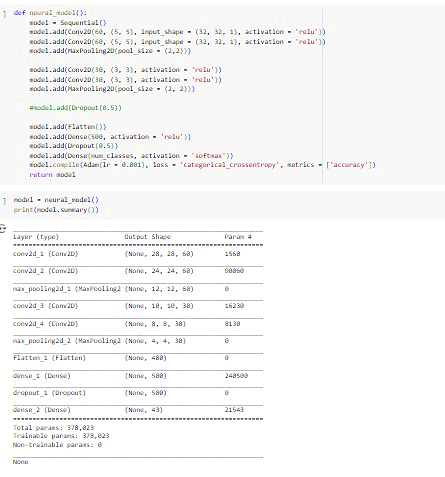


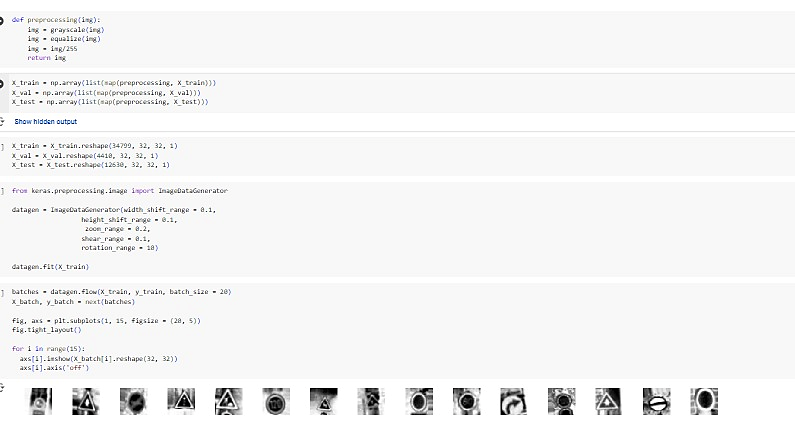












**Machine Learning part :**

**1. Project Setup and Data Collection**

* Define your goal: Specify the types of traffic signs you want to detect.
* Collect Data: Gather a dataset of images containing traffic signs. You can use existing datasets like GTSRB (German Traffic Sign Recognition Benchmark) or create your own by capturing images from the real world.

**2. Data Preprocessing**

* Annotation: Label the images with the corresponding traffic signs.
* Data Augmentation: Enhance your dataset by creating variations of the images (e.g., rotating, scaling, adding noise).

**3. Model Selection**

* Choose a suitable machine learning model for your task. Convolutional Neural Networks (CNNs) are commonly used for image recognition tasks.
* Consider pre-trained models like VGG16, ResNet, or MobileNet, which can save you time and resources.

**4. Model Training**

* Split Data: Divide your dataset into training, validation, and test sets.
* Training: Train your model using the training set and validate it using the validation set to tune hyperparameters.

**5. Evaluation and Testing**

* Evaluation: Assess your model's performance using the test set. Metrics like accuracy, precision, recall, and F1 score are useful here.
* Testing: Test your model on real-world data to ensure it performs well in practical scenarios.

**6. Deployment**

* Save Model: Once satisfied with the performance, save the model.
* Deploy: Implement your model into a real-world application or an embedded system to detect traffic signs in real-time.