Traffic Signs Classification Using Convolutional Neural Networks

MD. ZULFIKER MAHMUD, PHD

Course Instructor

Professor, Dept. of CSE, JnU

NISHAT MAHMUD

B190305003

MD. WALIUL ISLAM RAYHAN

B190305034

Digital Image Processing Lab

CSEL-4104

Objective

- Develop a system to recognize and classify traffic signs using CNNs.
- Integrating DIP techniques and CNNs for accurate traffic sign classification.
- Essential for autonomous driving, smart driving assistance, and traffic safety analysis.

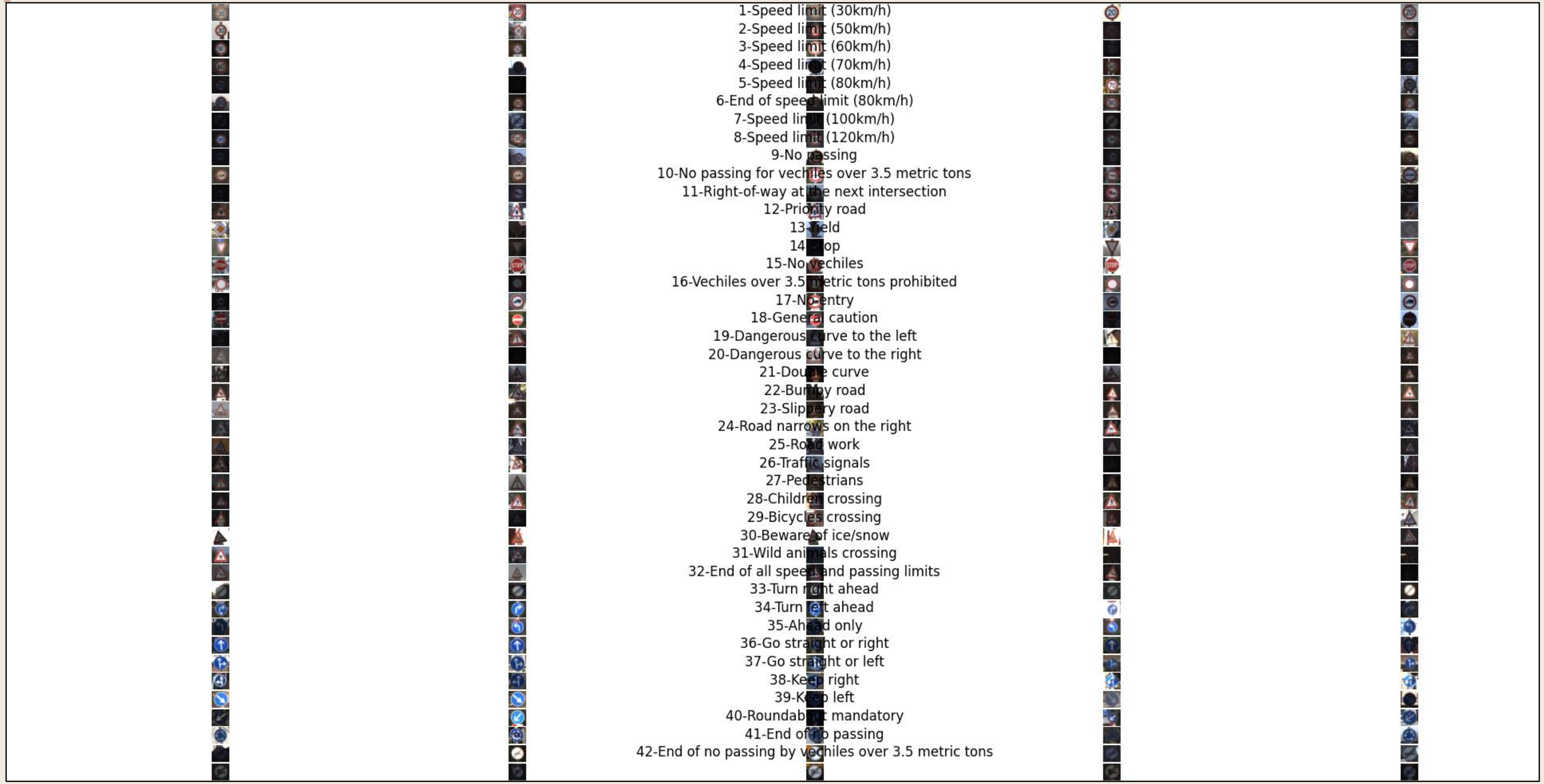
Motivation

- To make road navigation safer and more informed for both drivers and pedestrians.
- To enable seamless, on-the-go identification of traffic signs for all road users, not just vehicles.

Image Acquisition & Dataset

- **Dataset:** German Traffic Sign Recognition Benchmark (GTSRB)
- **Image Details:**
  - More than 35,000 images
  - Each sized 32x32 pixels
- **Classes:** Speed limits, warning signs, prohibitory signs, and other types, with a total of 43 different classes.

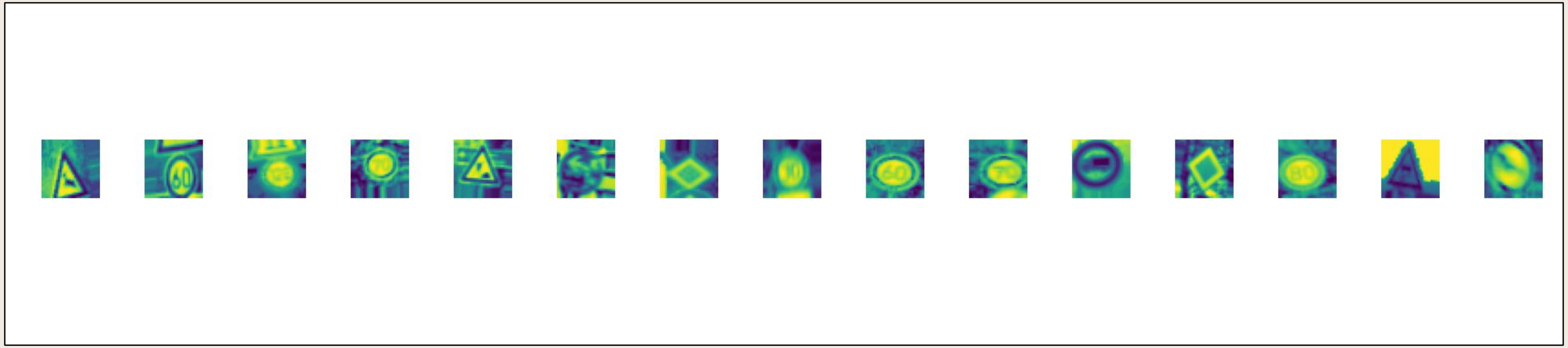
Dataset Overview



Data Preprocessing

- Grayscale conversion for simplified processing.
- Histogram equalization for better contrast.
- Data normalization to standardize data.
- Scaling pixel values to a range [0, 1] to enhance model performance.

Converted Gray Scale Images



Data Augmentation

- **Techniques Used:**
  - Width and height shifting
  - Zoom and shear transformations
  - Rotation for angle diversity
- **Purpose:** To improve model generalization by generating variations of the original images.

How CNN Utilizes DIP in This Project

- **Convolution layers** for feature extraction.
- **Pooling layers** for dimensionality reduction.
- **Dropout layers** to prevent overfitting.
- **Dense layers** for feature integration.
- **ReLU** and **Softmax** functions for activation and classification outputs.

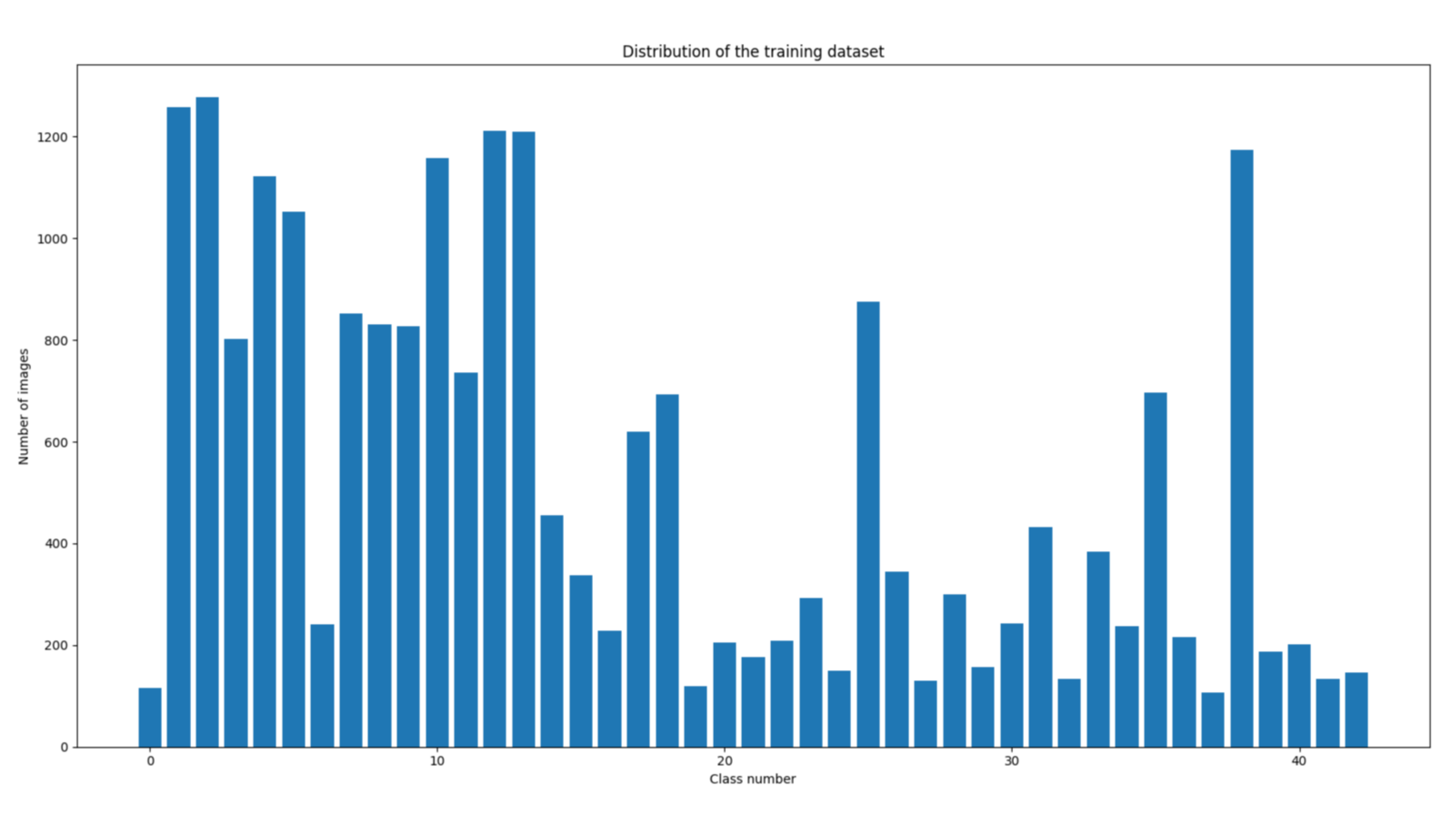
Process of Model Training

- **Data Splitting:**
    - **Training Set:** 80% of the dataset
    - **Validation Set:** 20% of the training set
- **Parameters:**
    - **Batch Size:** 2000
    - **Epochs:** 10

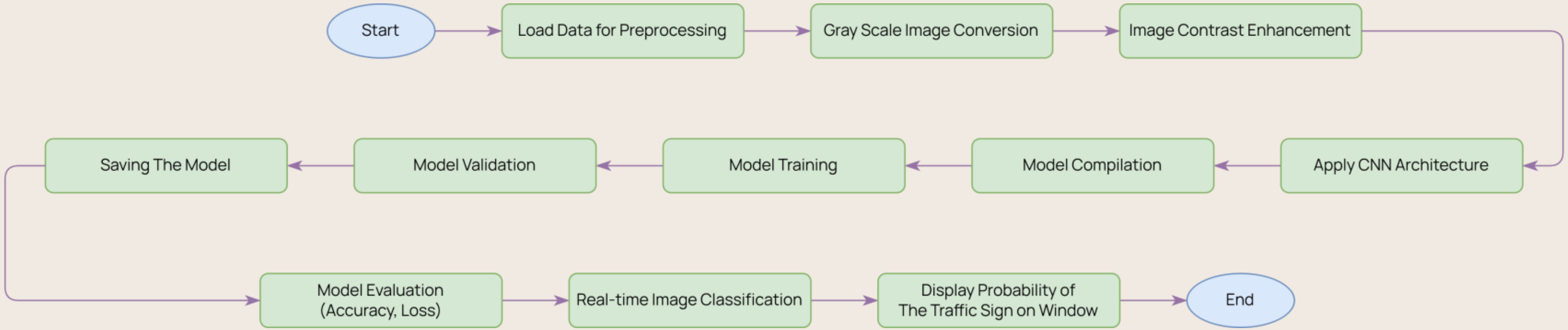
Process of Model Training (Cont.)

- **Training Process:**
  - Data augmentation applied before fitting data to the model.
  - Preprocessed images passed through CNN layers
- **Framework:** Keras and TensorFlow
- **Optimization:** Adam optimizer

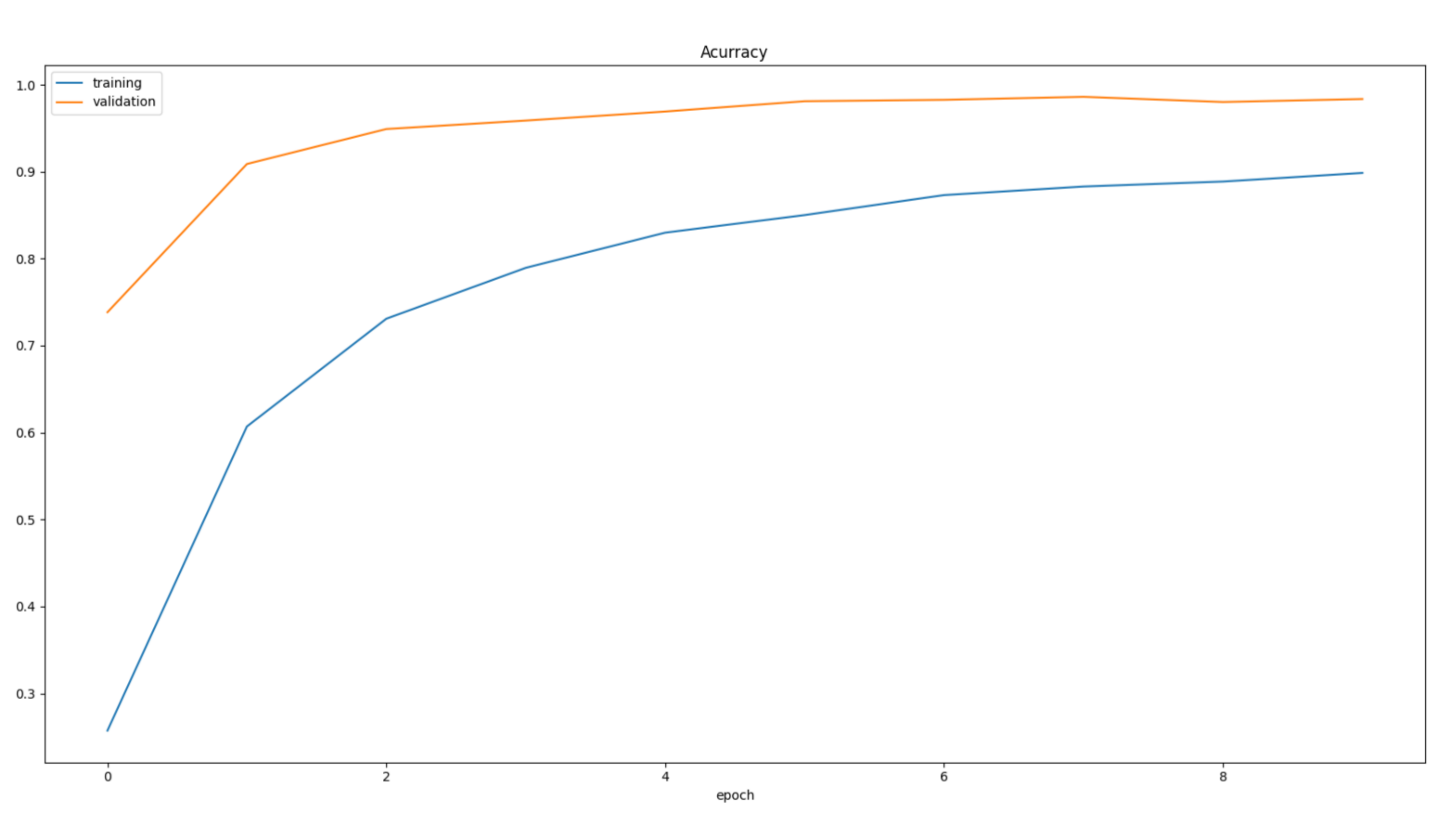
Distribution of the Training Dataset



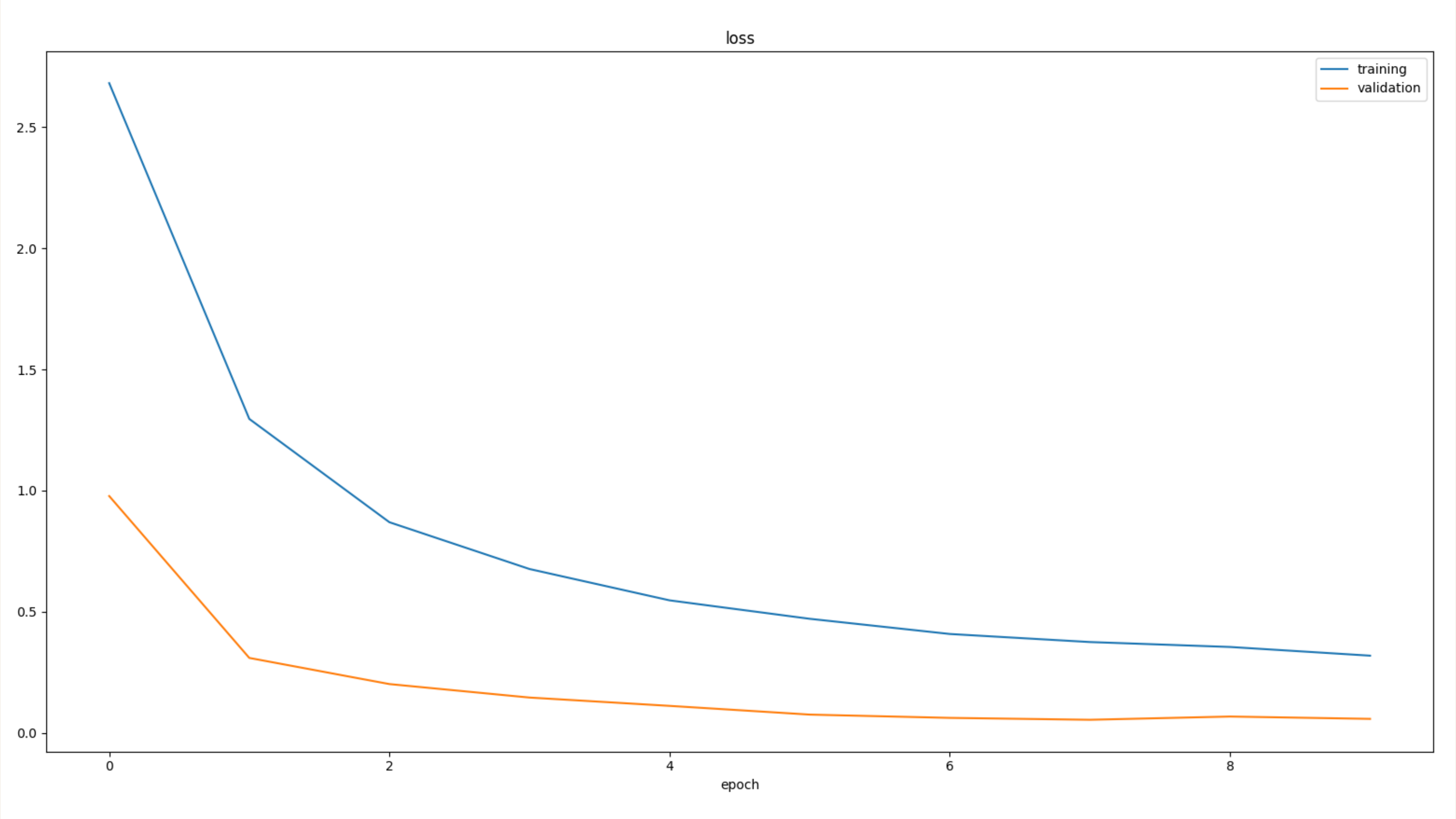
Project Workflow



Training vs Validation Accuracy Curve



Training vs Validation Loss Curve



Results & Evaluation

Test Accuracy	Test Score
98.29%	**6.19%**

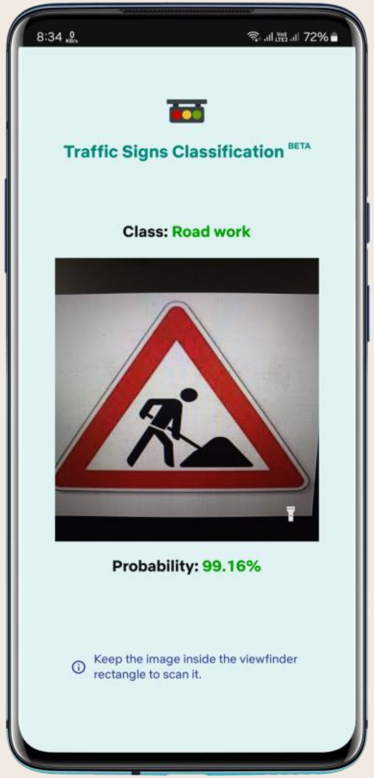
```
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 95s 47ms/step - accuracy: 0.5962 - loss: 1.3331 - val_accuracy: 0.9088 - val_loss: 0.3094
Epoch 3/10
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 49s 25ms/step - accuracy: 0.7250 - loss: 0.8862 - val_accuracy: 0.9490 - val_loss: 0.2015
Epoch 4/10
 446/2000 ━━━━━━━━━━━━━━━━━━━━ 2:48 109ms/step - accuracy: 0.7809 - loss: 0.7097983728: I tensorflow/core/framework/local
         [[{{node IteratorGetNext}}]]
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 52s 26ms/step - accuracy: 0.7875 - loss: 0.6838 - val_accuracy: 0.9587 - val_loss: 0.1459
Epoch 5/10
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 52s 26ms/step - accuracy: 0.8288 - loss: 0.5525 - val_accuracy: 0.9691 - val_loss: 0.1116
Epoch 6/10
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 54s 27ms/step - accuracy: 0.8498 - loss: 0.4713 - val_accuracy: 0.9810 - val_loss: 0.0756
Epoch 7/10
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 55s 27ms/step - accuracy: 0.8727 - loss: 0.4088 - val_accuracy: 0.9826 - val_loss: 0.0621
Epoch 8/10
 446/2000 ━━━━━━━━━━━━━━━━━━━━ 2:49 109ms/step - accuracy: 0.8796 - loss: 0.3782 status: OUT_OF_RANGE: End of sequenceow/
         [[{{node IteratorGetNext}}]]
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 52s 26ms/step - accuracy: 0.8821 - loss: 0.3757 - val_accuracy: 0.9860 - val_loss: 0.0542
Epoch 9/10
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 52s 26ms/step - accuracy: 0.8887 - loss: 0.3532 - val_accuracy: 0.9801 - val_loss: 0.0676
Epoch 10/10
2000/2000 ━━━━━━━━━━━━━━━━━━━━ 52s 26ms/step - accuracy: 0.8991 - loss: 0.3156 - val_accuracy: 0.9835 - val_loss: 0.0580
Test Score: 0.06193874403834343
Test Accuracy: 0.9829022884368896
```

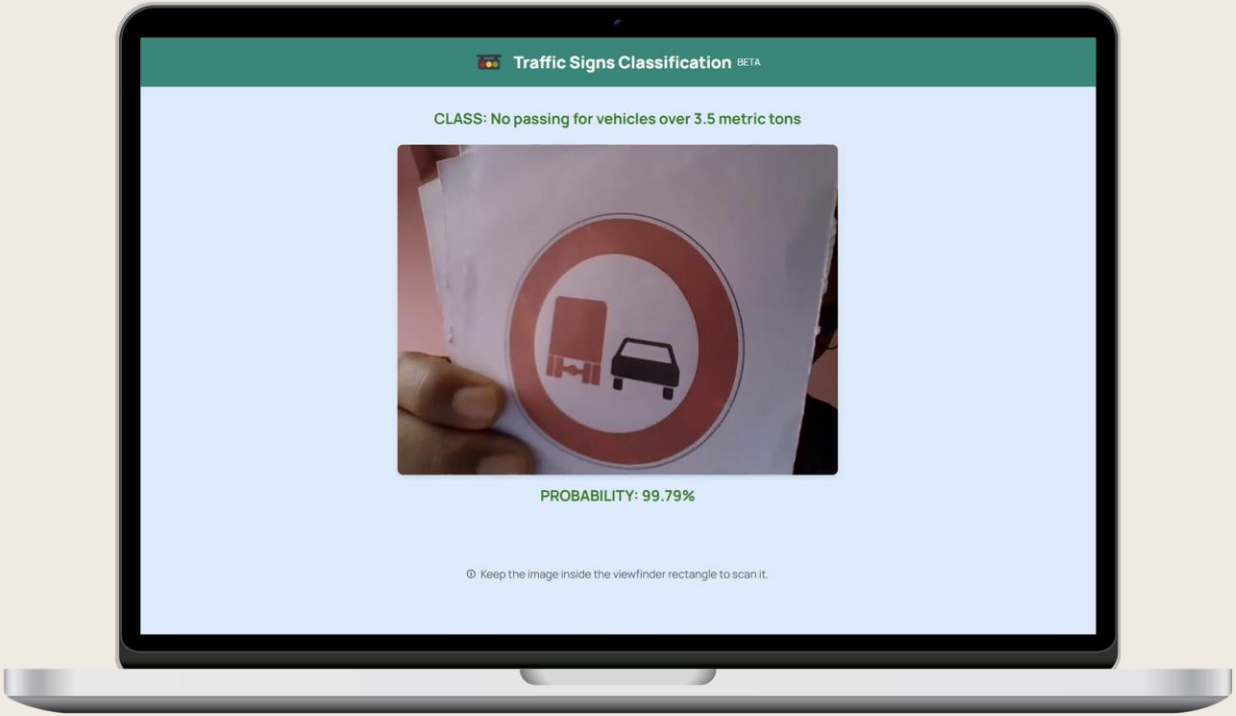
Live Testing Overview

Deployment

- **Android App:**
  - **Features:** On-the-go traffic sign recognition via the camera.
  - **Technologies:** TensorFlow Lite, OpenCV, Java, CameraX
- **Web Integration:**
  - **Features:** Allows users to classify traffic signs via the web app.
  - **Technologies:** Flask, HTML, CSS, JavaScript

Screenshots (App & Web)

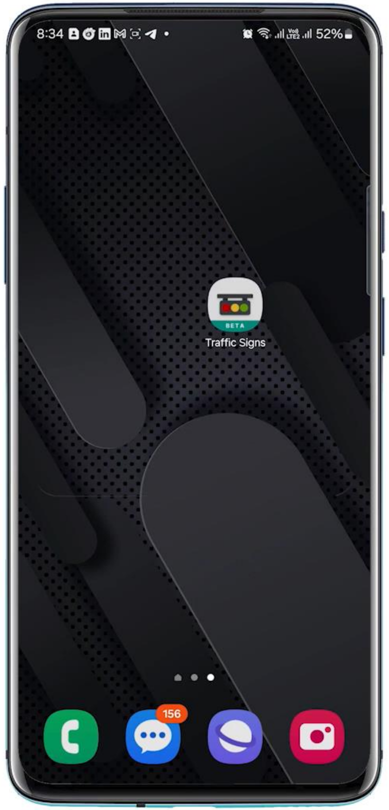




Future Work

- To collect a dataset of Bangladeshi traffic signs to enhance the model and make it more effective.
- To add audio alerts to tell users about detected traffic signs.
- Show detected signs on the screen in real-time using AR technology.

Video Demonstration



Android App

THANK YOU

Scan to learn more