# Traffic Sign Detection App

# Vision Document

## Version 1.1

## Team Members:

Hayden Jones

Shreya Komarabattini

Rishigesh Rajendrakumar

Aidan Mao

Jacob Heffelmire

|  |  |
|---|---|
| Title: Traffic Sign Recognition Application | |
| Version: 1.1 | Date: 10/3/25 |

## Revision History

| Date | Version | Description | Name |
|---|---|---|---|
| 10/3/25 | 1.0 | First draft | Hayden Jones |
| 10/31/25 | 1.1 | Not using Tensorflow anymore, just C++. Updated to reflect that. | Hayden Jones |

## Table of Contents

# 1. Introduction

## 1.1. Overview

The Traffic Sign Recognition (TSR) Application is a project designed to improve road safety by assisting drivers in recognizing and responding to traffic signs in real time. The system leverages deep learning and computer vision to classify traffic signs from a live camera feed or uploaded images. It is primarily intended for deployment on smartphones, ensuring accessibility and portability for a broad user base.

This vision document defines the problem, identifies stakeholders and users, and outlines the scope, features, and requirements of the TSR application.

## 1.2 Definitions, Acronyms, and Abbreviations

- **CNN** – Convolutional Neural Network, a type of deep learning model effective for image recognition tasks.
- **TSR** – Traffic Sign Recognition.
- **GTSRB** – German Traffic Sign Recognition Benchmark, a public dataset for training/testing TSR models.
- **OpenCV** – Open Source Computer Vision library used for image processing and real-time camera handling.
- **UI** – User Interface.
- **FPS** – Frames Per Second, measure of real-time performance.

- **YOLO** - Stands for You Only Look Once. It is a detection algorithm that is especially good at inference of real time video.

## 1.3 References

- **GTSRB Dataset:** http://benchmark.ini.rub.de/
- **OpenCV Documentation:** https://docs.opencv.org/

# 2. Problem Statement

**Problem:**

Drivers often miss or misinterpret traffic signs due to distraction, low visibility, or environmental conditions. This can lead to overspeeding, missed stops, or unsafe maneuvers, contributing to accidents.

Current in-car assistance systems are costly and limited to newer vehicles, leaving most drivers without affordable access.

**Proposal:**

The goal of this project is to design a lightweight, smartphone-based TSR application that can provide real-time sign detection and alerts to improve driver awareness and safety. The app can highlight the signs via camera feed and also give audio alerts to users keeping them aware of road conditions while driving.

## 3. Stakeholder and User Descriptions

### 3.1. Stakeholder Summary

| Name | Description | Responsibilities |
|---|---|---|
| Capstone Instructor/Advisor Dr. Mohammadreza Hajiarbabi | Project Sponsor, Project Advisor | Provide project requirements, evaluate project deliverables, evaluate prototype system |
| Team Members | Developers of the TSR application | Design, build, test, and present the project |

### 3.2. User Summary

| Name | Description |
|---|---|
| Drivers | Users who want real time sign recognition and alerts while driving |
| Students/Researchers | Users that test the system in a controlled environment, using uploaded images and video |

### 3.3. User Environment

**Driver**

Drivers can open the app on their Android phone and use their camera to detect road signs. Users can enter settings to change the region that they are located in, as well customize the voice and sign highlighting output.

**Developer**

Developers can open the app and upload images and videos to test the sign detection.

## 3.4. Operating Environment

- Mobile deployment on Android/iOS.
- Core implementation in C++ using Cuda for speeding up training
- OpenCV for video processing and detection
- Must run efficiently on mid-range smartphones.

## 3.5. Key Stakeholder or User Needs

| Need | Priority | Concerns | Proposed Solution |
|---|---|---|---|
| Create a functioning prototype for project sponsors | Critical | Prototype must be created from scratch | Use Agile method to ensure continuous progress and adaptability to changes |
| Drivers need accurate, fast real time recognition of signs | High | Model may not be fast enough to accurately recognize signs while driving, especially under poor lighting or high speed conditions. | Optimize the CNN model using techniques such as quantization and pruning to reduce latency. |
| Model must be lightweight enough to run on smartphones | High | The model may not be small enough to run efficiently on smartphones with limited processing power and memory. | Export our model to TensorFlow Lite for model compression. |
| Users need a simple and mostly hands free interface to use while driving | High | Drivers could be distracted by the app while driving | The application will primarily alert users through audio, and require only a single press to start the app before driving. |
| App should protect user privacy and not upload data without consent | High | Continuous camera use may raise privacy concerns regarding image storage or transmission. | All image processing is done locally on the user's device, so no sensitive data is stored. |
| Developers need testing and evaluation modes with metrics like accuracy | Medium | Without proper evaluation tools, it will be difficult to measure model performance and compare improvements. Lack of standardized testing may | Implement a built-in testing mode that logs predictions and computes metrics such as accuracy using sample images or validation data. Provide summary reports |

| | | | |
|---|---|---|---|
| | | also cause inconsistent results across datasets or devices. | after each test session. |
| Voice alerts and detection must stay synchronized during rapid or multiple sign detections | High | If several signs are recognized in quick succession, voice alerts may overlap, lag, or skip signs. This could confuse or distract the driver. | Implement a voice alert queue system that prioritizes important signs and limits the rate of spoken alerts. Use asynchronous processing to prevent voice playback from blocking detection. Display non-critical alerts visually if voice feedback is delayed. |
| App should recover gracefully if the camera feed or model fails | Medium | System crashes or lost video feed during driving could reduce safety and usability. | Implement error handling and automatic recovery routines that restart the camera feed or notify the user to re-enable it. |
| App must provide accurate text-to-speech feedback for recognized signs | Medium | Mispronunciations or delayed voice alerts could confuse drivers and reduce safety. | Integrate a text to speech engine with low latency. Test voice output timing to ensure alerts are synchronized with sign detection. Allow users to adjust voice volume. |
| Drivers can switch between regions | Optional - Medium | Training for different regions requires many different datasets and may require retraining or fine-tuning the model to recognize region-specific traffic signs. | Implement a modular knowledge base that allows loading region-specific models or label sets. Provide downloadable sign packs or model updates for different regions. |
| App should allow customization for alert settings | Low | Different users may prefer different alert types or volumes. | Add user settings for toggling voice alerts and adjusting sensitivity |

# 4. Product Overview

## 4.1. Overview & Scope

The TSR application will recognize traffic signs in real time using CNN-based classification. It will provide visual overlays, audio alerts, and confidence percentages. Optional features may include customizable regional sign sets and customization features for visual and audio output.

## 4.2. Summary of Capabilities

**Functional Requirements**

- Live camera feed input and video/image upload.
- Preprocessing for resolution, lighting, angle, occlusion.
- CNN-based recognition engine for sign classification.
- Support for multiple signs per frame.
- Real-time visual and audio feedback.
- Knowledge base of traffic signs with regional customization.
- Testing/training mode with performance metrics.

**Non-Functional Requirements**

- $\geq$95% accuracy on benchmark dataset.
- Latency under 200 ms per frame.
- Real-time performance (15–30 FPS).
- Secure, local image processing.
- Portable across Android.
- Reliable in varied conditions (day/night, weather).

## 4.3 Requirement Analysis (Scope and Limitations)

**Scope:**

- Prototype mobile app capable of real-time recognition of major traffic sign categories.
- Dataset training with GTSRB.
- Integration with OpenCV for live camera feed.

**Limitations:**

- Mobile deployment may require significant optimizations (model compression, lightweight UI).

## 4.4 Assumptions and Dependencies

- Availability of labeled traffic sign datasets (e.g., GTSRB).
- Stable smartphone hardware with sufficient processing capability.
- Open-source libraries like OpenCV remain actively supported.

## 4.5 Project Priorities

1. Functional prototype with real-time sign recognition.
2. High accuracy and low-latency performance.
3. Usable, mobile-friendly interface.
4. Testing and evaluation mode.
5. Optional features if time permits.