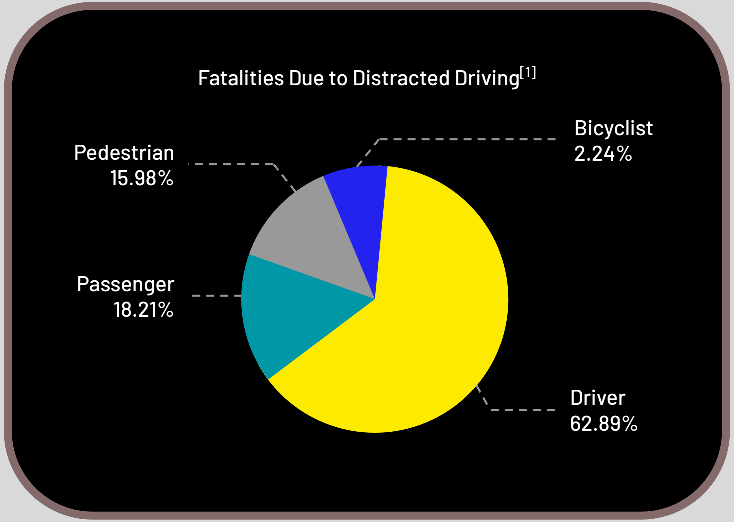
Automotive Retrofit Vision System (ARVIS)

Devin Reichenbach
Ryan Januszko
Kyle Smith
Joseph Wenger

Distracted Driving

- Each day in the U.S., 9 fatalities and over 1,000 injuries are caused by accidents in which distracted driving was reported to be involved.
- The majority of such fatal accidents result in the death of the driver.
- Younger drivers are at a higher risk, specifically individuals aged 15-20 years old.[2]

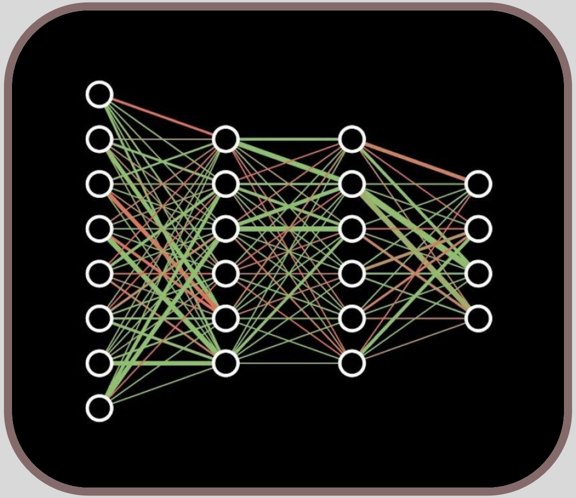


Potential Solutions

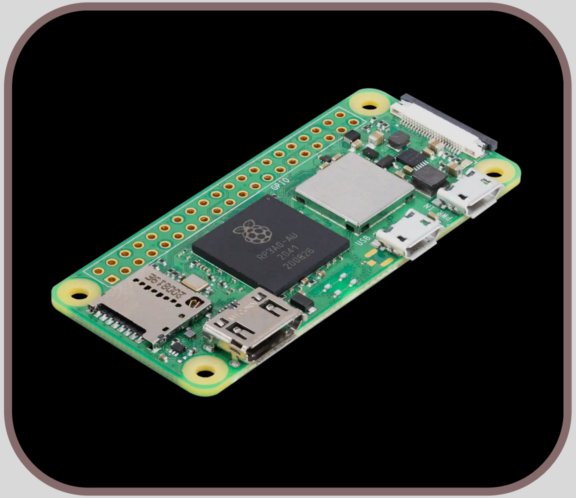
- Government Response
  - 2009 – President Obama issued executive order, banning all federal employees from texting and driving.
  - 2011 – State of Pennsylvania also banned texting and driving.
- Manufacturers' Response
  - Automotive safety system improvements
  - Newer vehicles equipped with ADAS*
- Origin of Our Solution
  - Average age of vehicles on the road in 2024 was 14 years.[3]
  - Vehicles manufactured earlier than 2015 possess significantly less advanced safety features.[4]
- Our solution is designed for installation in vehicles not equipped with ADAS to implement similar features without the cost of a new vehicle.

** Automated Driver Assistance System*

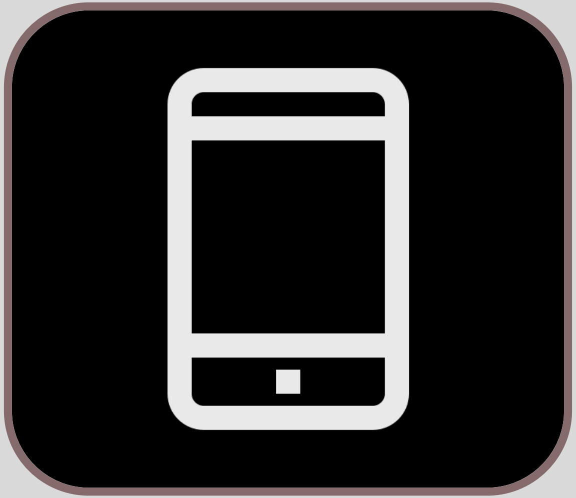
Three Key Components



Computer Vision Model

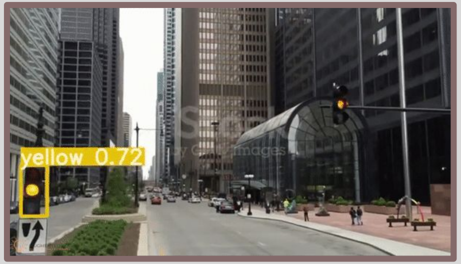


Embedded Hardware

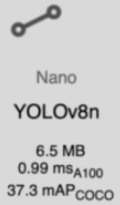


User Interface

What is Object Detection?

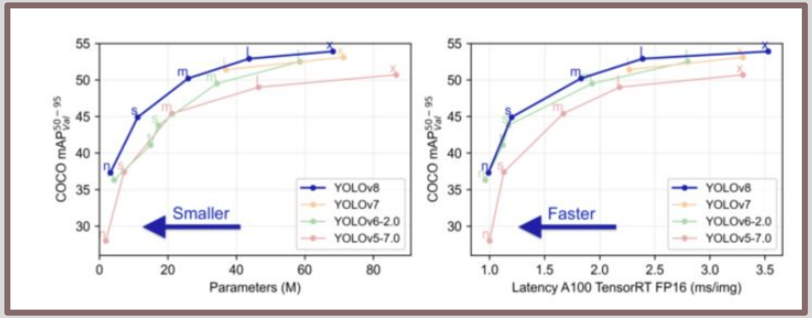


A computer vision task involves **detecting**, **localizing**, and **classifying** desired objects in an image.



Architecture - YOLOv8n

- Single-Stage Convoluted Neural Network
  - Detects and Classifies objects in entire image in one pass
  - Ideal for real-time video processing
- Smallest and fastest model of the YOLOv8 family
  - Sacrifice accuracy for size and speed
  - Perfect for edge devices



Detection Model Dataset

- Approximately 24,000 annotated images
  - Red, Yellow, Green Lights
  - Speed Limits: 15, 25, 30, 35, 40, 45, 50, 55, 65
  - Stop Signs
  - Pedestrian Crossings
- Annotated Images?
  - Informs model where and what object(s) is in the corresponding image
  - Label File Contents
    - <class> <x_center> <y_center> <width> <height>
- Used Annotated Datasets archived online
  - Mostly LISA Dataset

```
— dataset
    ├── test
    │   ├── images
    │   │   └── image1.jpg
    │   └── labels
    │       └── image1.txt
    ├── train
    │   ├── images
    │   │   └── image2.jpg
    │   └── labels
    │       └── image2.txt
    └── val
        ├── images
        │   └── image3.jpg
        └── labels
            └── image3.txt
```

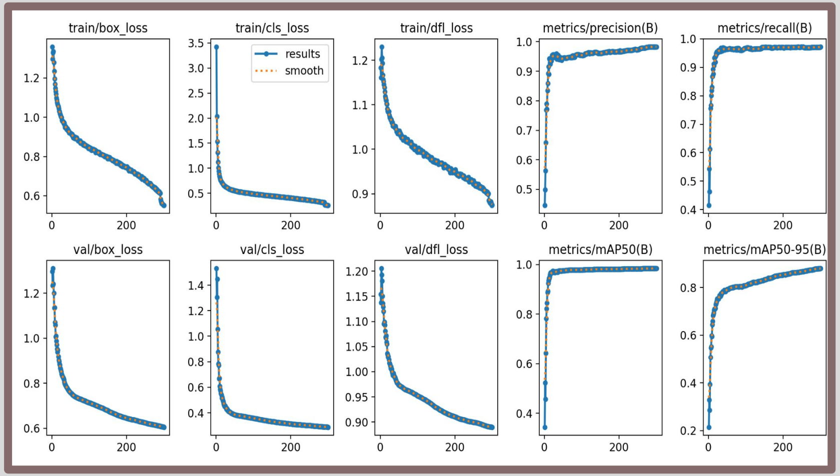
Training the Model

How is a model trained?

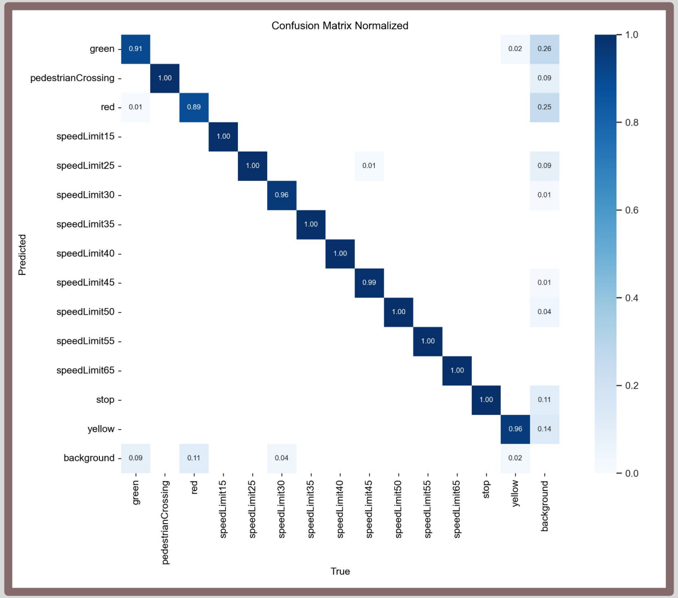
- Show the model a labeled example and let it make a prediction
- Calculate the error between the guess and the actual location of the object
- Adjust the internal parameters to reduce error
  - YOLOv8n contains 3 million parameters
- Repeat over many iterations (epochs) to help model learn patterns

Training our Model

- 300 epochs over the course of approximately 2 days
- After each epoch, model runs through a validation set to evaluate performance of model during training



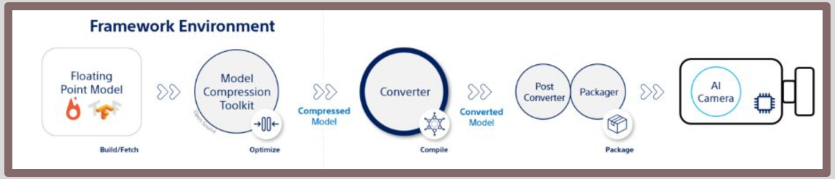
Training the Model



Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	2373	2720	0.983	0.972	0.984	0.881
green	264	337	0.966	0.878	0.924	0.701
pedestrianCrossing	125	138	0.971	0.986	0.993	0.869
red	284	356	0.974	0.857	0.927	0.68
speedLimit15	17	17	0.98	1	0.995	0.96
speedLimit25	201	205	0.981	1	0.995	0.929
speedLimit30	70	70	0.985	0.946	0.992	0.893
speedLimit35	230	245	0.999	1	0.995	0.895
speedLimit40	36	36	0.993	1	0.995	0.895
speedLimit45	88	100	1	0.991	0.995	0.897
speedLimit50	27	27	0.97	1	0.995	0.953
speedLimit55	8	8	0.96	1	0.995	0.988
speedLimit65	32	32	0.991	1	0.995	0.938
stop	733	855	0.992	1	0.995	0.931
yellow	266	294	0.996	0.946	0.982	0.801

Speed: 0.3ms preprocess, 2.9ms inference, 0.0ms loss, 0.8ms postprocess per image

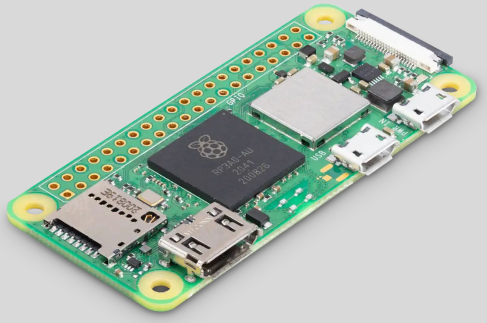
Model Implementation



- Post Training Quantization (PTQ)
  - Convert 32-bit floating point to 8-bit integer
    - decreases accuracy, makes compatible with IMX500 AI Camera
- C++ Video Object Detection
  - Speed
- Python Wrapper
  - C++ executable wrapped in Python for data manipulation
  - Integrity
    - Count frames an object is on screen to reduce false positives
  - Simplify data transmission to companion app

Raspberry Pi Zero 2W

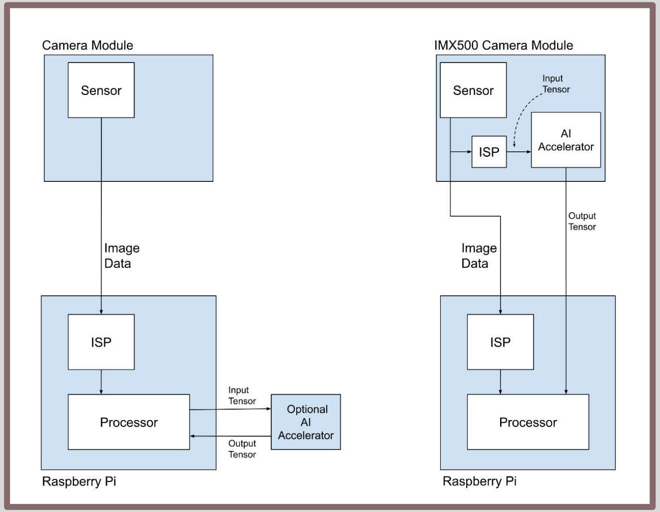
- General Features
  - Inexpensive
  - Small Form Factor
- Technical Features
  - ARM Cortex-A53 (64-bit) Processor
    - 1 GHz
  - 512 MB SDRAM
  - Bluetooth and WiFi Capabilities
  - Low Power Consumption
    - 0.6–1.2W
- Operating System
  - Pi OS Lite (32-bit)
  - Headless operation



Raspberry Pi AI Camera

- Augments captured video with tensor metadata increasing machine learning compute power.
- 12.3 megapixel image sensor
- 10 fps at full resolution (4056x3040)
  30 fps at “binned mode” (2028x1520)

\+ Supplemental processing power allowed us to achieve higher performance without a more powerful SBC.



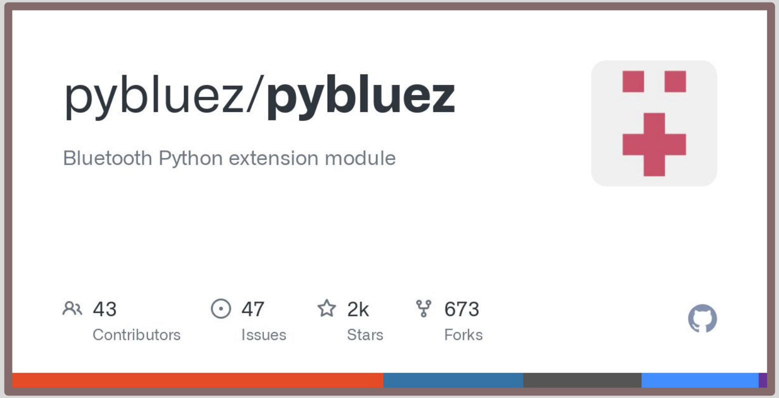
Comparison between systems with and without an AI camera

Bluetooth Communication

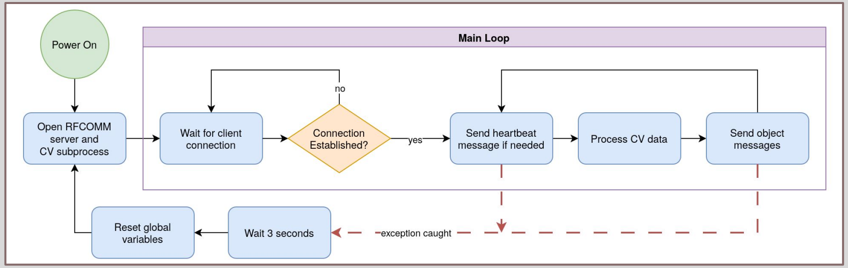
Achieved using the PyBluez Module for Python
- Python 3.11

Client-Server Model
- Server: Raspberry Pi Zero 2W
- Client: Smartphone App

Server sends data to the Client



Bluetooth Implementation



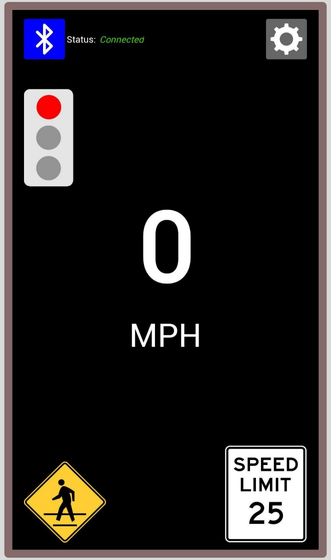
PyBluez
- Server establishes connection with an RFCOMM socket
- Code modified from MIT example RFCOMM server

Error Handling
- Watchdog sends a heartbeat message to client
- Client requires message to be sent every two seconds
- Server is refreshed after three seconds following an error

Improvements
- Simplified pairing
    - Button-Pairing
- Run Bluetooth on a Separate Thread
    - Improves Performance and Robustness
    - Report Errors with CV Program's client

Companion App

- Android mobile application for UI implementation
- Created with MIT App Inventor 2
  - \+ Minimal experience required
  - \+ Rapid development/prototyping
  - \- No custom classes
  - \- Difficult to design freely
  - \- Inaccurate preview resolution
- Utilitarian design
  - Intended to minimize distraction to driver
- Key Features
  - Current speed vs. speed limit display
  - Road sign display
  - Traffic light display
  - Optional, customizable audio alerts
  - Bluetooth connection status and error reporting



SWOT Analysis

Strengths:

- \+ Cost Effective
- \+ Repairable
- \+ Community Supported
- \+ Small Form Factor

Weaknesses:

- – Performance Limitations
- – Rudimentary Hardware
- – Lack of Brand Trust
- – Small Feature Set
- – Zero Security Measures

Opportunities:

- \+ Regulatory Compliance Features
- \+ Sign Violations
- \+ Insurance Implementation
- \+ System Expansion

Threats:

- – Established Competition
- – Frequent Changes to Available Hardware
- – Firmware Support Reaching End of Life

Cost Analysis

ARVIS costs around $150 to produce:

- Raspberry Pi Zero 2: $15
- Raspberry Pi AI Camera: $70
- Miscellaneous Parts: ~$5
- Tools and Labor: ~$50*

Commercial Viability

- Closest competing products cost $190-$300
- We can either undercut the competition or price competitively
- $200: Profitable and cheaper than most competition
- $230-$260: Competitive and highly profitable

Type S brand AI Dash Cam System: $249.99

** Estimated cost per unit covering software development manufacturing resources*

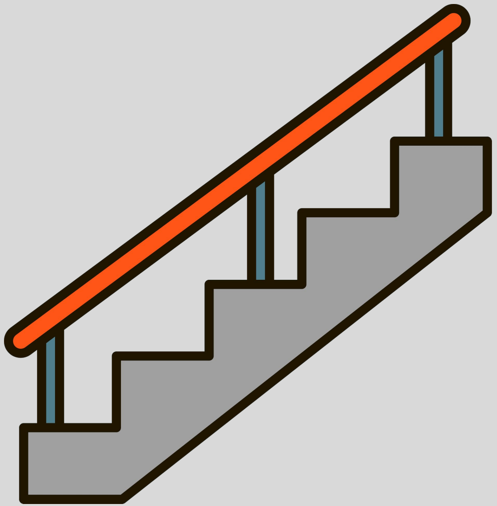
Test ID	Description	Expected Results	Actual Results	Pass/Fail
1	Collect data with correctly formatted bounding boxes	Coordinates and labels are recorded accurately for each object	Coordinates and labels were accurate	Pass
2	Model detects object and draws bounding box	Bounding box and label are correctly displayed around the object	Label displayed, but bounding box was inaccurate	Fail
3	Model detects multiple objects and draws bounding boxes	Bounding boxes and labels are correctly drawn and displayed	Bounding boxes and labels displayed correctly with predictions	Pass
4	Establish Bluetooth connection to phone	Successful pairing confirmation from Raspberry Pi	Pairing message received successfully	Pass
5	Message transmission from Pi to mobile app	Encoded message appears in app terminal	Message displayed correctly in terminal	Pass
6	Display signs and traffic lights on app from Pi	Signs and traffic lights display correctly on app	Signs appeared, but screen formatting was incorrect	Pass

Demonstration

This video was recorded from a team member's own vehicle, driving to the University. The screen of the mobile app was recorded simultaneously.

Next Steps

- Improve Model Accuracy
  - Speed limit sign classification
- Implement additional features on embedded hardware
  - Lane departure detection
  - Collision detection
- Implement additional features on user interface
  - Track user activity
- Migrate mobile application to Flutter
  - Currently only supported by Android
  - Better design capability



[1] Buckfire Law. “Distracted Driving Accidents,” Buckfire & Buckfire, P.C. [Online]. Available: https://buckfirelaw.com/case-types/car-accidents/distracted-driving/.

[2] National Highway Traffic Safety Administration, Distracted Driving 2022, Report No. DOT HS 813 703, 2024. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813703.

[3] Hedges & Company, “Average Age of Cars & Trucks in the U.S.,” Hedges Blog, Feb. 2024. [Online]. Available: https://hedgescompany.com/blog/2024/02/average-age-of-cars-trucks/.

[4] MITRE, Market Penetration of Advanced Driver Assistance Systems (ADAS), PR-24-2614, Sept. 2024. [Online]. Available: https://www.mitre.org/sites/default/files/2024-09/PR-24-2614-PARTS-Market-Penetration-Advanced-Driver-Assistance-Systems.pdf.

[5] A. Sharma, “Training the YOLOv8 Object Detector for OAK-D,” PyImageSearch, May 01, 2023. https://pyimagesearch.com/2023/05/01/training-the-yolov8-object
-detector-for-oak-d/

[6] Raspberrypi, “AI Camera - Raspberry Pi Documentation,” Raspberrypi.com, 2024. https://www.raspberrypi.com/documentation/accessories/ai-camera.html

[7] Ultralytics, “YOLO Performance Metrics,” Ultralytics.com, 2023. https://docs.ultralytics.com/guides/yolo-performance-metrics/#interpretation-of-results

[8] A. Nanda, “Quantizing Neural Network Models - TDS Archive - Medium,” Medium, Sep. 07, 2024. https://medium.com/data-science/quantizing-neural-network-models-8ce49332f1d3