The Road Ahead:

Creating a CNN for Traffic Sign Detection and Classification

Argishti Ovsepyan

Problem Statement Outline

- **Objective:** Implementing a Convolutional Neural Network (CNN) to detect and interpret traffic signs, a critical component in automated driving technology.
- **Significance:** Traffic sign detection is vital for safety and regulation compliance in autonomous vehicles.
- **Impact on Industry:** The successful implementation of traffic sign detection and classification can play a transformative role in the automotive industry, enhancing the user experience in modern driving.

Agenda

01

Finding Data

Preprocessing the tiny LISA dataset for modeling

02

EDA

Investigating traffic sign class distribution

03

Modeling

Creating cold start CNN models and exploring VGG19 transfer learning

04

Visualization

Observing all CNN layers and implementing Grad-CAM overlay for final layer

05

Future Work

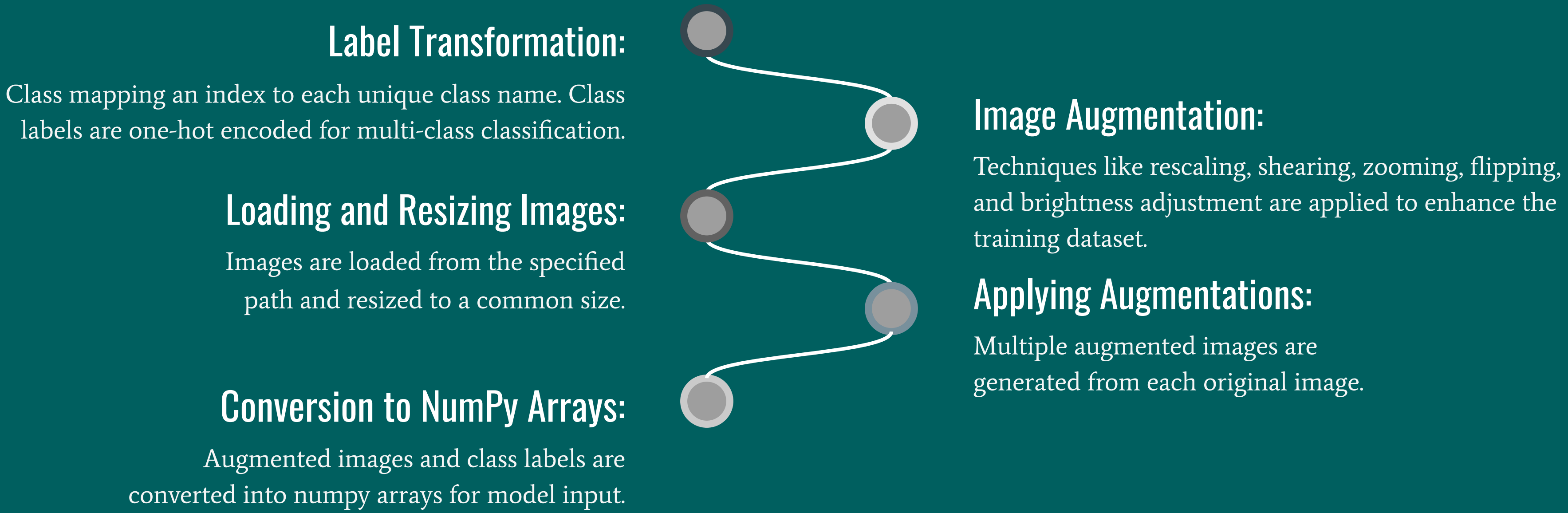
Investigating limitations resulting in incorrect predictions

01

Finding Data

Preprocessing the tiny LISA dataset for modeling

Preprocessing **data**



Label Transformation:

Class mapping an index to each unique class name. Class labels are one-hot encoded for multi-class classification.

Image Augmentation:

Techniques like rescaling, shearing, zooming, flipping, and brightness adjustment are applied to enhance the training dataset.

Loading and Resizing Images:

Images are loaded from the specified path and resized to a common size.

Applying Augmentations:

Multiple augmented images are generated from each original image.

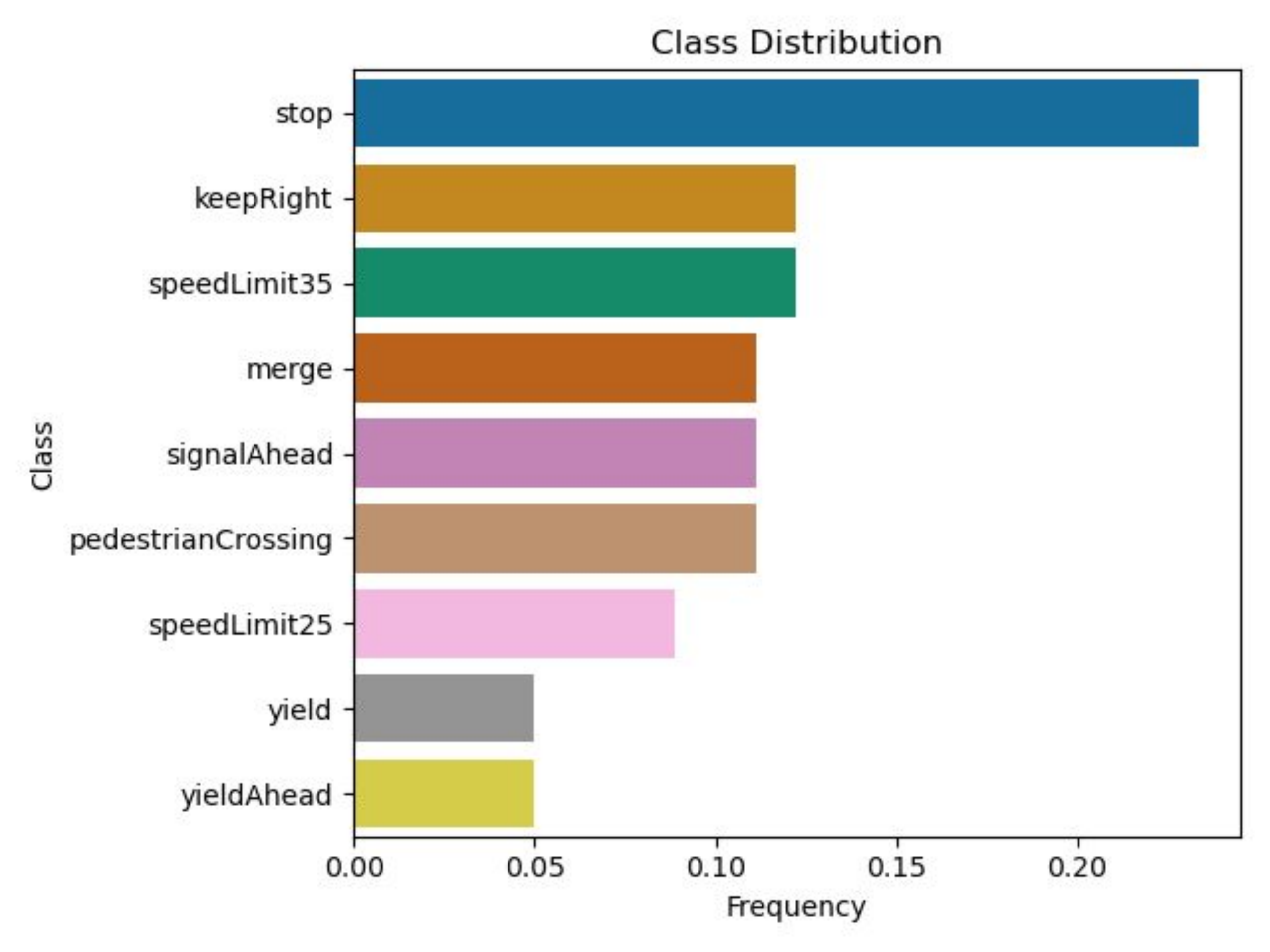
Conversion to NumPy Arrays:

Augmented images and class labels are converted into numpy arrays for model input.

02

EDA

Investigating traffic sign class distribution

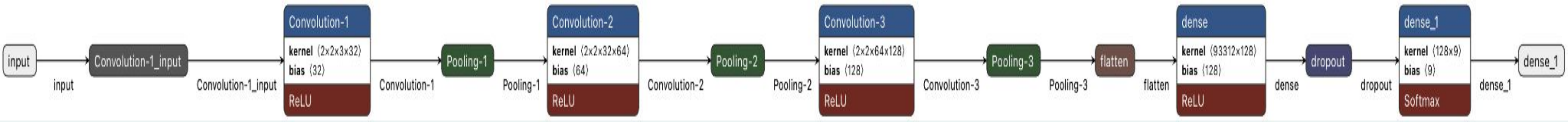

Class Distribution
Class
stop
keepRight
speedLimit35
merge
signalAhead
pedestrianCrossing
speedLimit25
yield
yieldAhead
0.00
0.05
0.10
0.15
0.20
Frequency

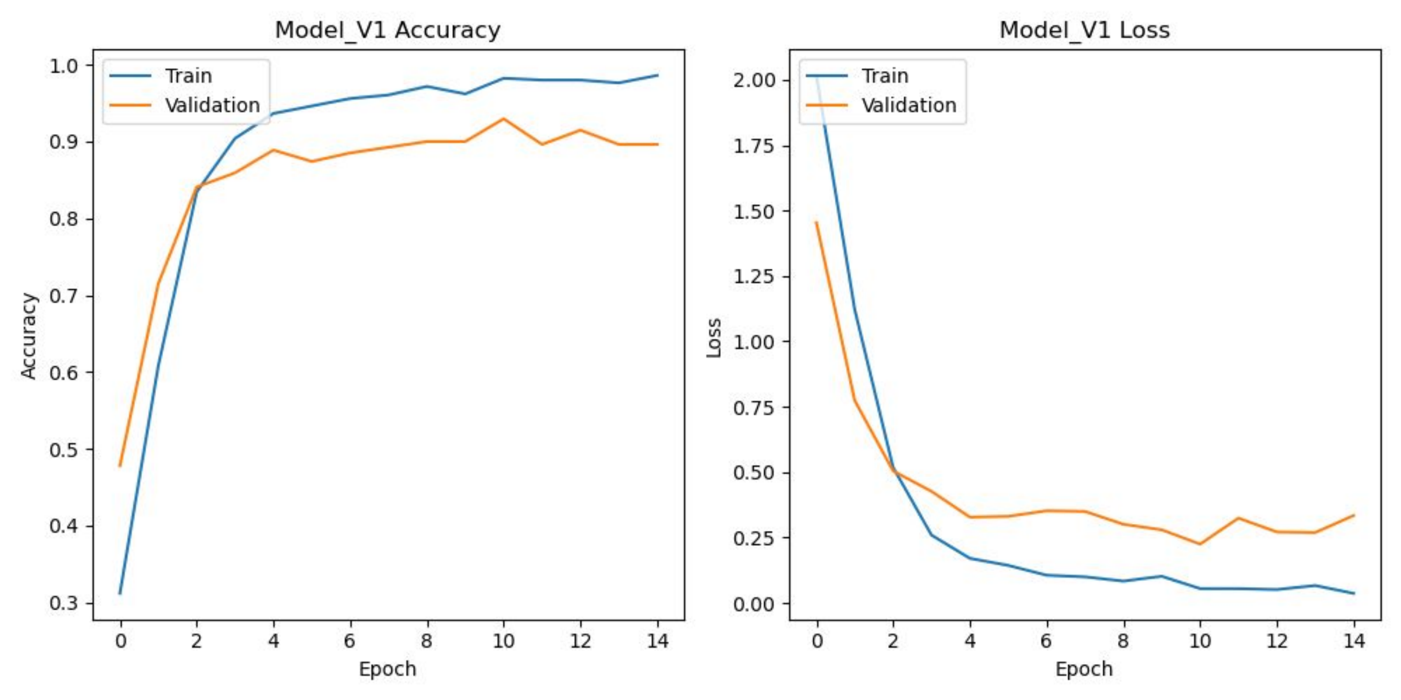
03

Modeling

Creating cold start CNN models and exploring VGG19 transfer learning

Model Version 1



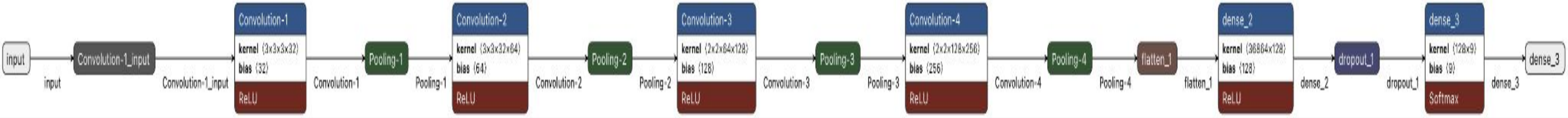


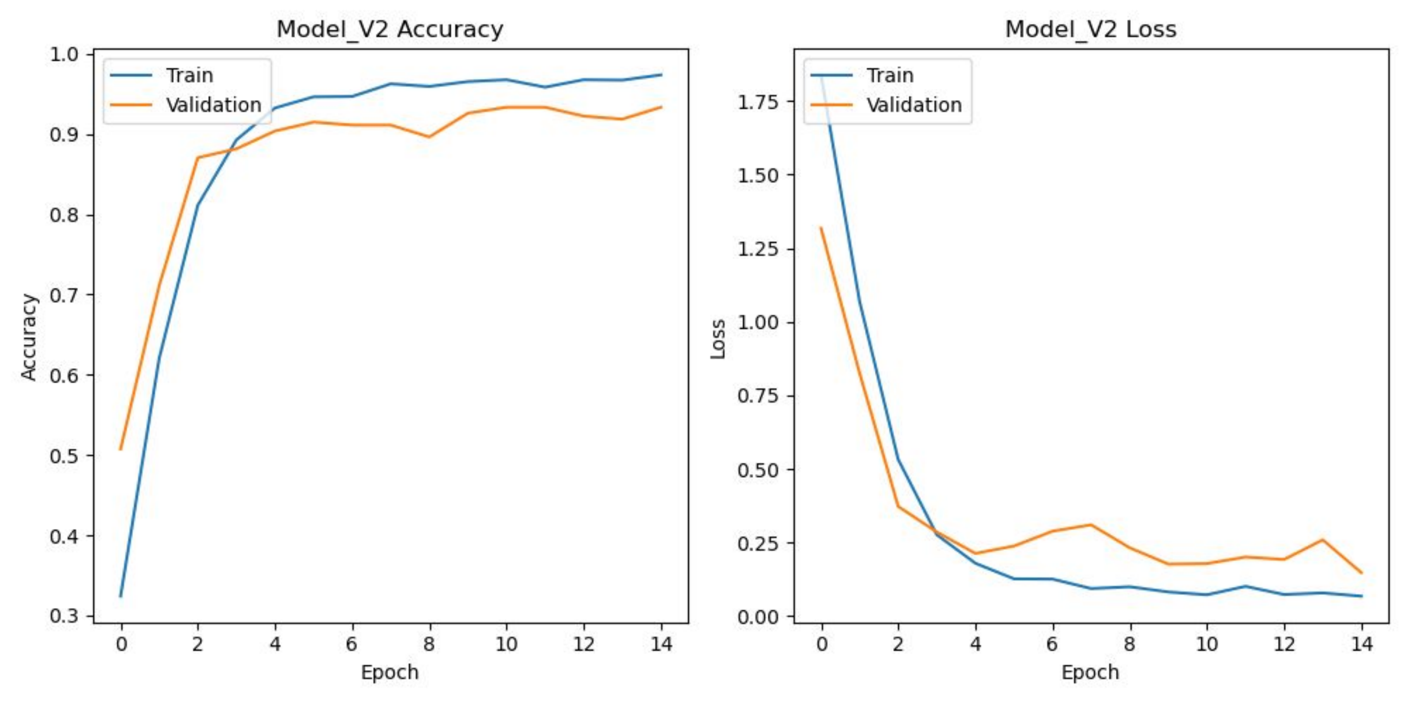
Holdout Results:

0.8963 Accuracy

0.3243 Loss

Model Version 2





Holdout Results:

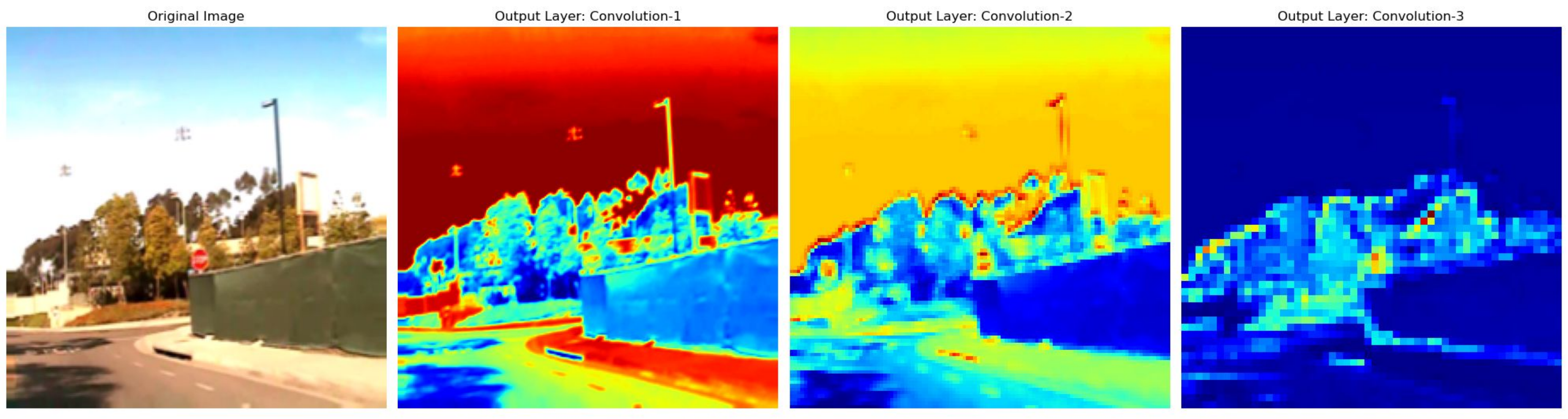
0.9296 Accuracy

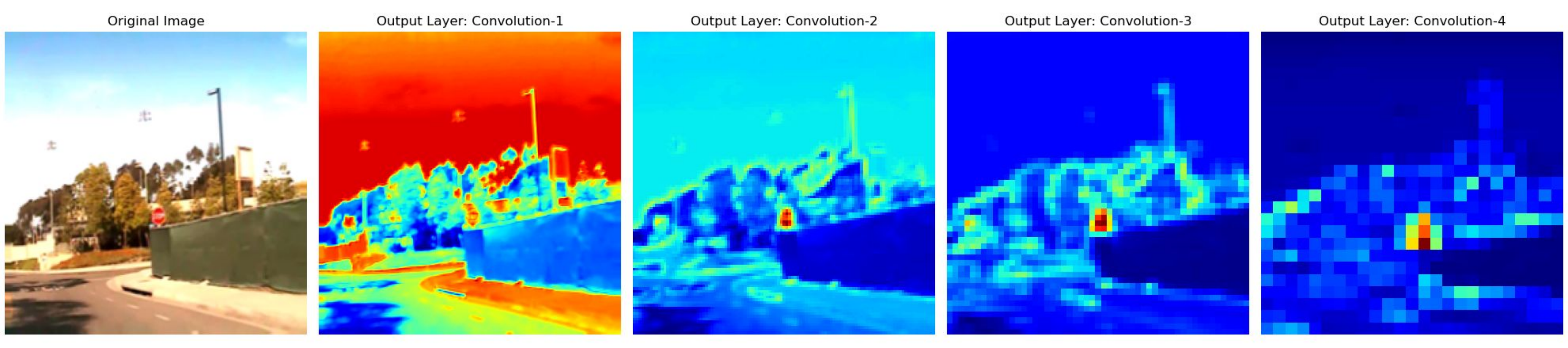
0.2318 Loss

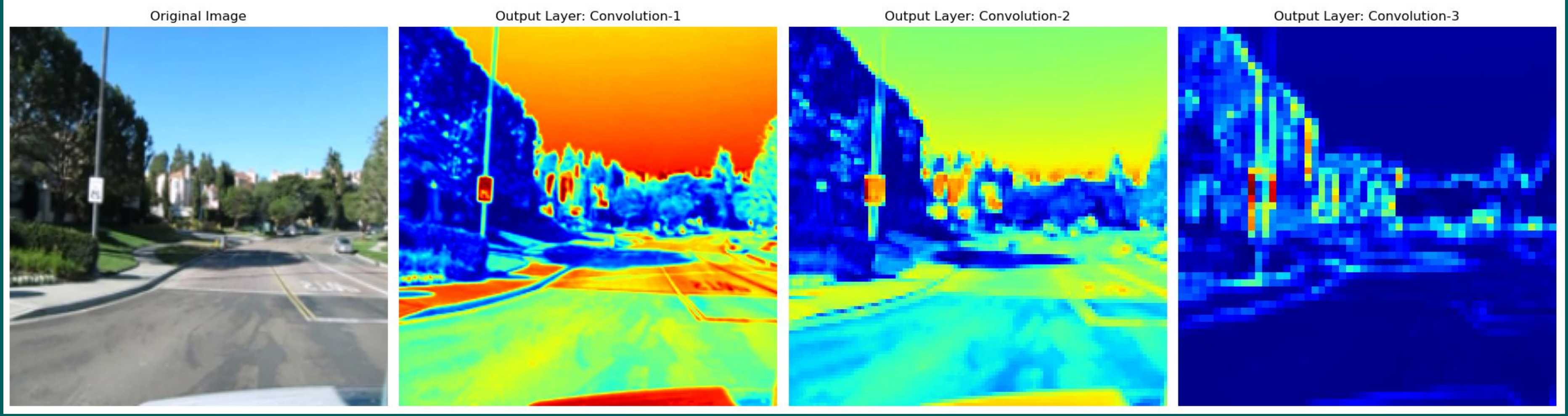
04

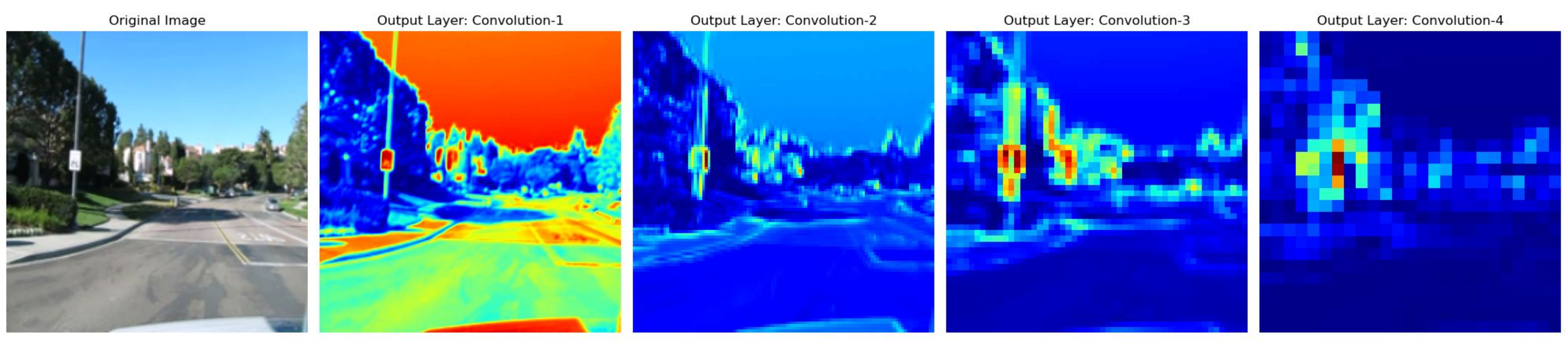
Visualization

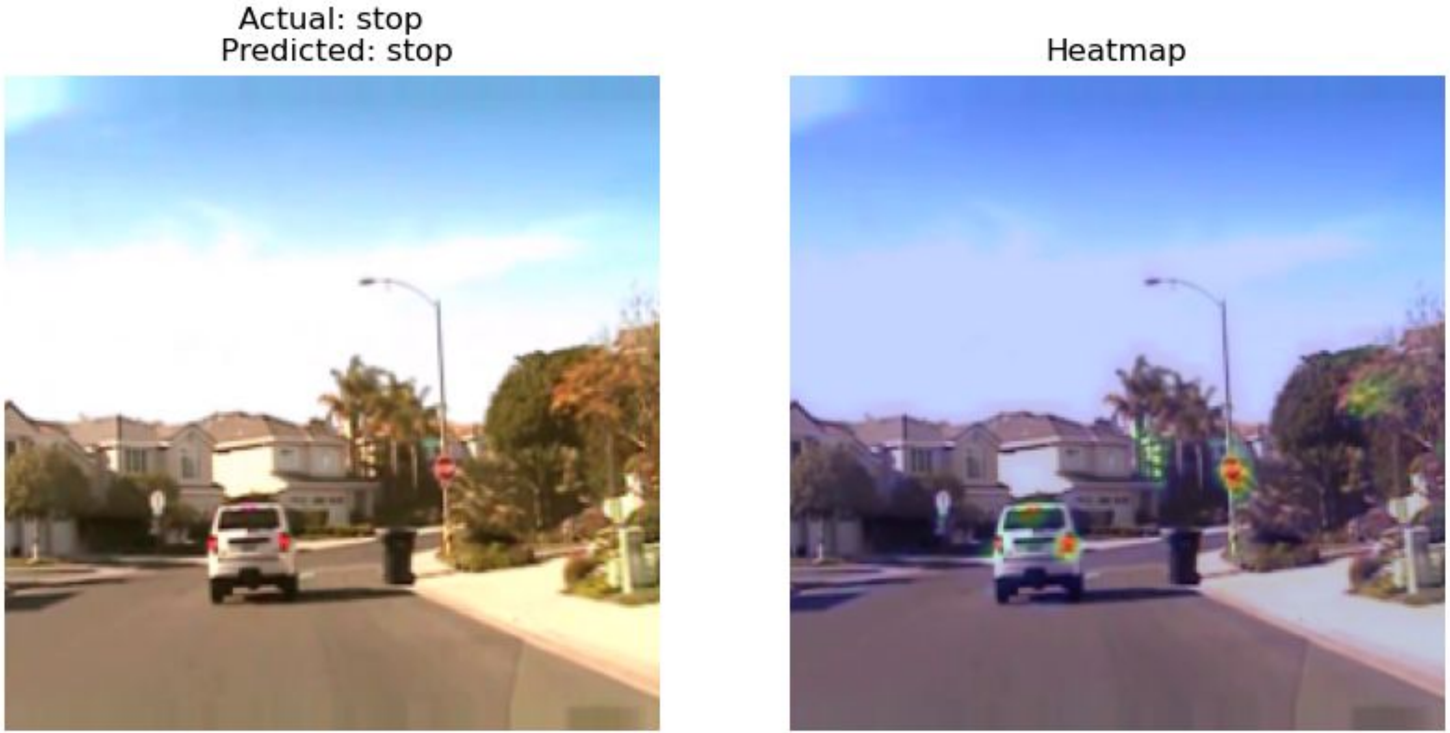
Observing all CNN layers and implementing Grad-CAM overlay for final layer


Original Image
Output Layer: Convolution-1
Output Layer: Convolution-2
Output Layer: Convolution-3


Original Image
Output Layer: Convolution-1
Output Layer: Convolution-2
Output Layer: Convolution-3
Output Layer: Convolution-4

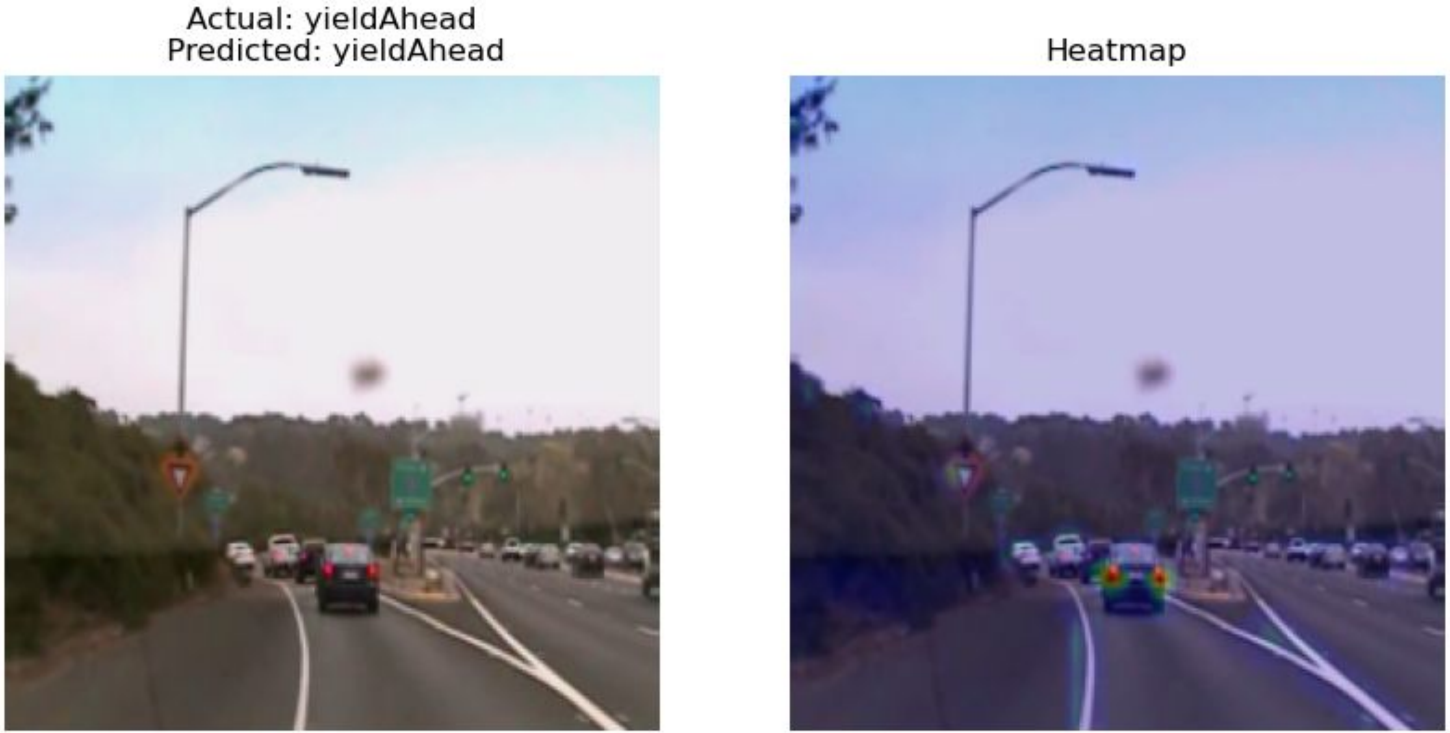

Original Image
Output Layer: Convolution-1
Output Layer: Convolution-2
Output Layer: Convolution-3


Original Image
Output Layer: Convolution-1
Output Layer: Convolution-2
Output Layer: Convolution-3
Output Layer: Convolution-4



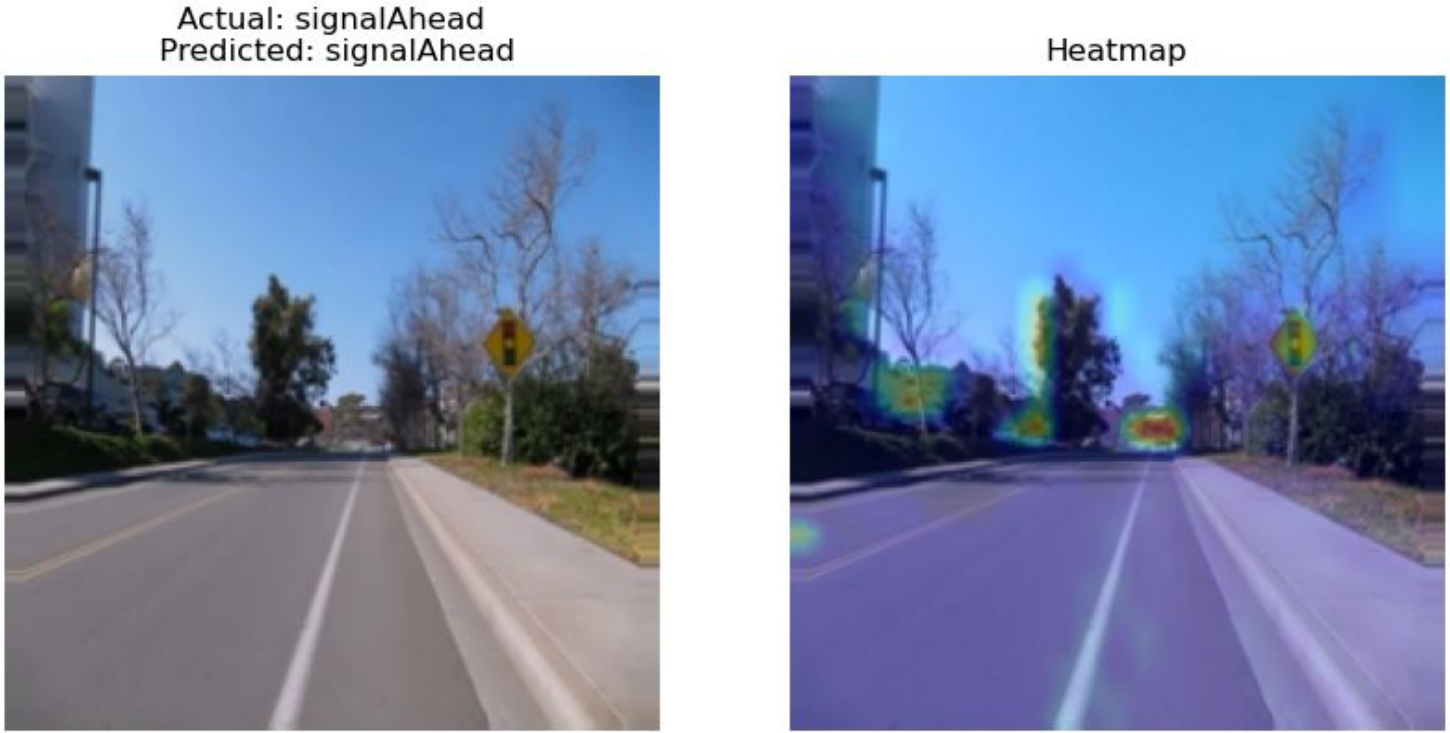
Actual: stop
Predicted: stop

Heatmap



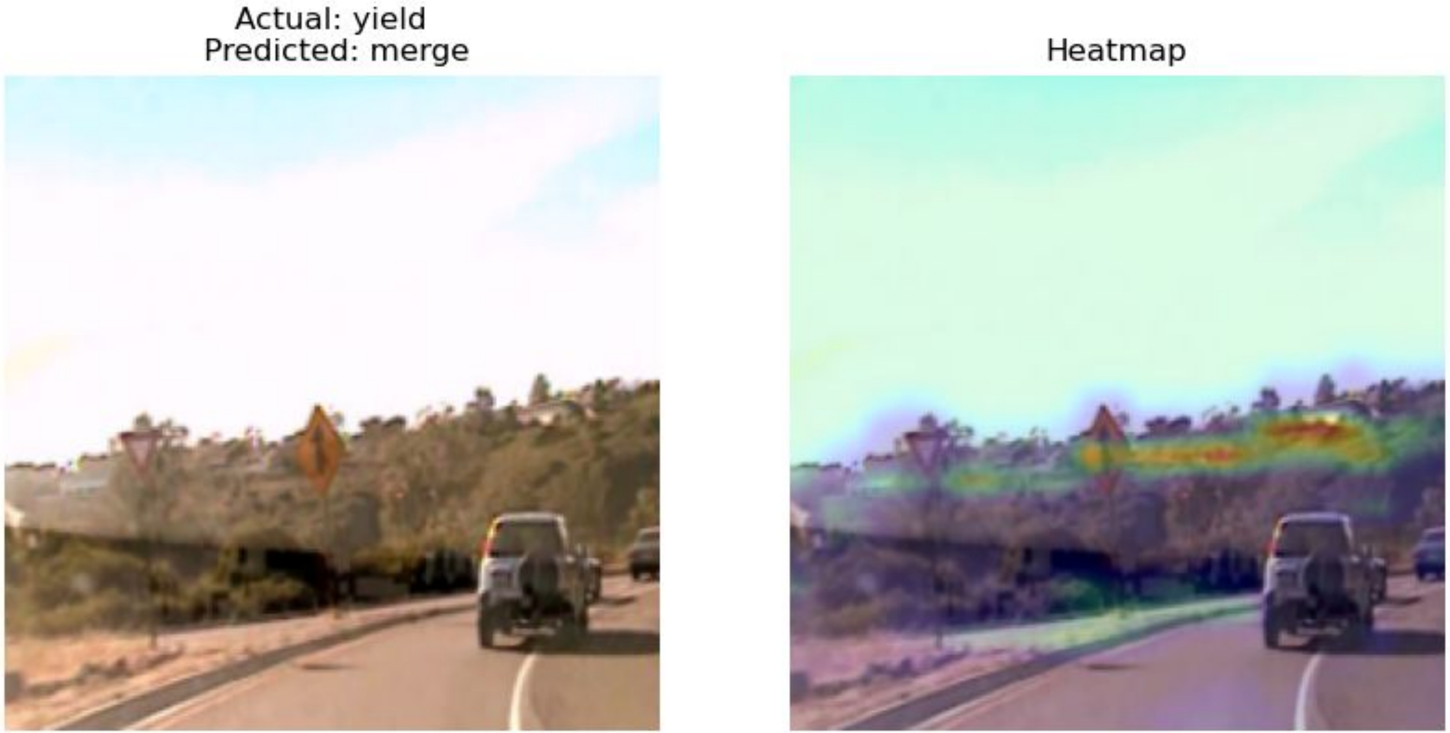
Actual: yieldAhead
Predicted: yieldAhead

Heatmap



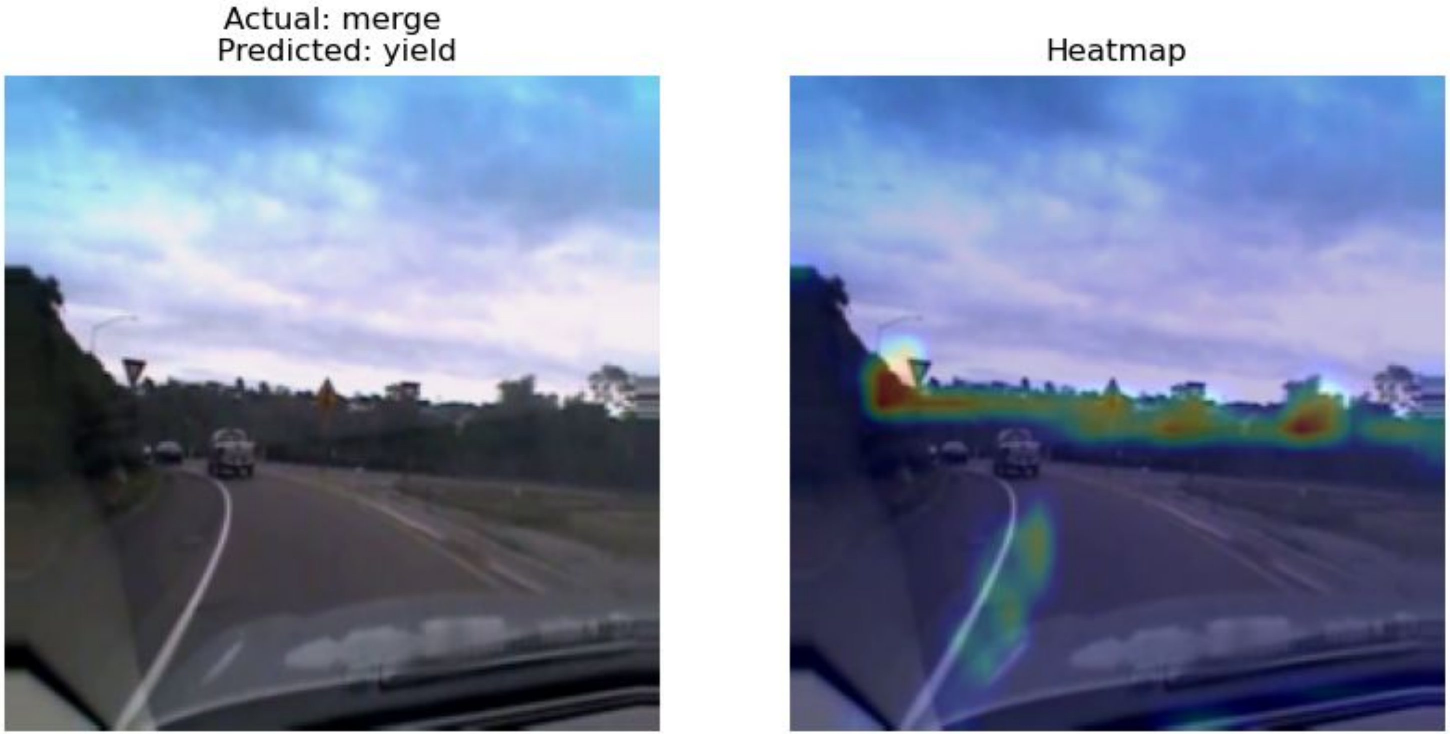
Actual: signalAhead
Predicted: signalAhead

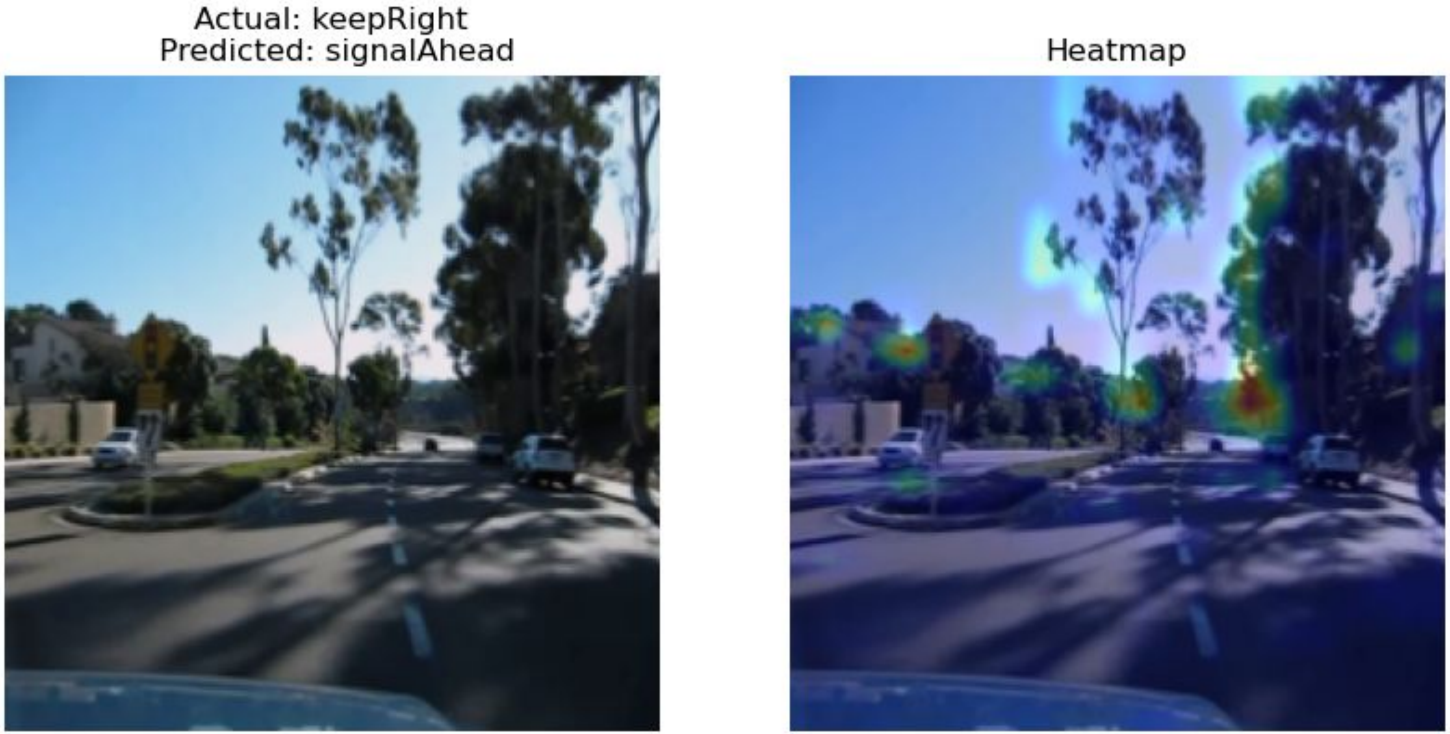
Heatmap



Actual: yield
Predicted: merge

Heatmap


Actual: merge
Predicted: yield
Heatmap



Actual: keepRight
Predicted: signalAhead

Heatmap

05

Future Work

Investigating limitations resulting in incorrect predictions

- Investigate issues with "Keep Right" predictions and handling of images with multiple signs.
- Adjust the model, exploring variations in batch size and layer composition.
- Examine the impact of different image presentations (e.g., black and white, grayscale) on predictions.
- Assess how color (especially red and yellow) affects the model's decisions.
- Expand the diversity of training images, including different distances from the camera, for a more robust solution.
- Refine the model to reduce misclassifications and enhance its performance in real-world scenarios.

Thanks!

Do you have any questions?