

# Simulating Self-driving Cars: Traffic Sign Recognition

*Francis Leung, Matthew McElhaney, Swati Akella*

Keywords: Deep Learning, Tensorflow, Jetson TX2, Image Classification, Transfer Learning

## Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Abstract</b>	<b>2</b>
<b>Problem Statement</b>	<b>2</b>
<b>Data</b>	<b>2</b>
Sources	2
German Traffic Sign Recognition Benchmark	2
Mapillary Traffic Signs Dataset	3
Preprocessing	3
Train/Validation/Test	4
<b>Solution</b>	<b>4</b>
Model Selection and Training	4
Validation Results	5
Implementation	6
Inference on the Edge	6
Test Results	8
<b>Conclusion</b>	<b>9</b>
<b>References</b>	<b>9</b>
<b>Appendix</b>	<b>10</b>
Appendix A - Overview of The Image Dataset	10

# Abstract

Self driving cars are subject to intense research by both academia and technology companies. Part of the required tasks for a self driving car is the recognizing and actioning of street signs. This paper explores the use of a deep learning model that can classify a subset of signs accurately and deploys the trained model for inference on an edge device.

## Problem Statement

Self driving cars encounter many technical challenges during normal operation. Several examples are:

- Obtaining accurate navigation and location information
- Operating safely in a variety of environmental conditions
- Maintaining situational awareness as the environment around the car constantly

The focus of this paper is on the final challenge, specifically the identification of street signs using an onboard vehicle camera. The scope of these signs include speed limit signs, stop, yield, and animal crossing. This paper explains the process for training a model in the cloud, moving it to the edge, and executing inference and action based on the sign observed.

## Data

### Sources

The primary data source for training and testing were obtained from two repositories of traffic signs:

1. German Traffic Sign Recognition Benchmark provided by the German Federal Ministry of Education and Research<sup>2</sup>
2. Mapillary Traffic Signs Dataset<sup>1</sup>

### German Traffic Sign Recognition Benchmark

	Dataset
Absolute Size	422 mb
Image Count	39,210
Total Classes	42

An overview of the image dataset can be found in Appendix A.

## Mapillary Traffic Signs Dataset

	Dataset
Absolute Size	47.1 gb
Image Count	52,453
Total Classes	312

Both datasets contained images of signs in different lighting, weather conditions and clarity, which is important for our inference model as we expect our car to be able to operate at all times of day and during all seasons. Examples include:



***Figure 1. Image that is taken from a distance from the camera***



***Figure 2. Image that is taken in poor lighting***

## Preprocessing

After review of the dataset, we reduced the number of classes from 42 to 10 for the self-driving car simulation. These were chosen as specific actions could be programmed following the recognition of the sign. The final classes selected were:

Sign	Potential Action by Car Program
30 km/h	slow down or speed up
50 km/h	slow down or speed up
60 km/h	slow down or speed up
70 km/h	slow down or speed up
80 km/h	slow down or speed up
100 km/h	slow down or speed up
120 km/h	speed up
Deer Crossing	slow down
Stop	slow down & stop
Yield	slow down

We combined the two datasets by cropping images from the Mapillary dataset and adding them to the German dataset to increase the amount of training data available. We also used an augmentor on the cropped images, which transformed the images with translations (by 10 pixels to the left or right), rotations (0-10 degrees clockwise/counterclockwise), adding noise (0.005-0.15) and adding blur (0.5 and 1.0).

## Train/Validation/Test

We applied a 80/20 training and validation set, resulting in 17,250 training images and 4,310 validation images. For testing, we created “routes” for a car comprising a set of 18 signs. These images were set aside prior to the augmentation steps above so that there was no chance that the model had “seen” them prior to testing.

## Solution

### Model Selection and Training

We attempted two different approaches to solve the problem of identifying traffic signs: 1) An object detection model that would recognise the presence of street signs from images taken from a moving vehicle and classify them, and 2) a simpler classification model that would classify images of signs without other objects in the background.

We reached out to Mapillary and received an education license to their research dataset. This dataset consisted of images and fully annotated json files. When we started working with this data, we ran into multiple roadblocks such as image files not having annotated files and vice versa, certain bounding boxes were outside image height and width. We decided to annotate a set of 4,500 images with LabelImg and use this data for training. While training, we realized that the object detection API was not updated for TF2 and although we used a container with TF1, we faced multiple deprecation issues.

Given the difficulties we encountered with the object detection approach, we focused on the image classification model. Optimizing for time spent in model training, we decided to use transfer learning via feature\_vectors from pre-trained models from Google’s Tensorflow Hub selecting *Inception v3*, *Inception-ResNet V2*, and *ResNet V2 with 101 layers*. These models were chosen as they were the latest state of the art models.

We trained our model both in the cloud with Tesla V100s and onboard the Jetson TX2, and found that while it was faster to train in the cloud, it was also faster to iterate through different training regimes on the TX2. We also tested different optimisers and loss functions, but found best results with the *Adam* optimizer and the *BinaryCrossEntropy* (with from\_logits = True) loss function.

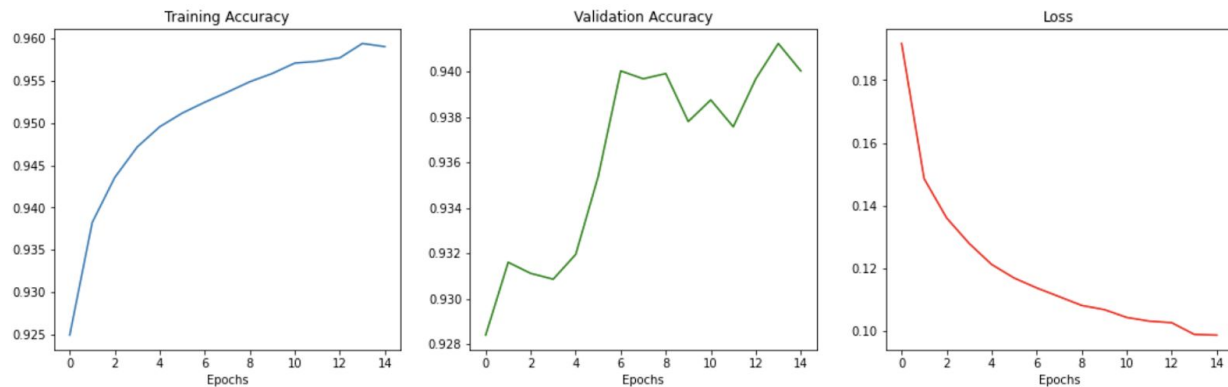
## Validation Results

We trained each model for 15 epochs and tracked both the training accuracy and validation accuracy of each model. In our initial testing, we found that the models had difficulty differentiating between the different speed limits. As discussed above, we increased the number of training images but this however, did this not lead to a major improvement in accuracy. Our final results were as follows:

	Inception v3	Inception-ResNet V2	ResNet V2 101
Epochs	15	15	15
Max Validation Accuracy	0.9412	0.9354	0.9439
Final Validation Accuracy	0.9400	0.9354	0.9439

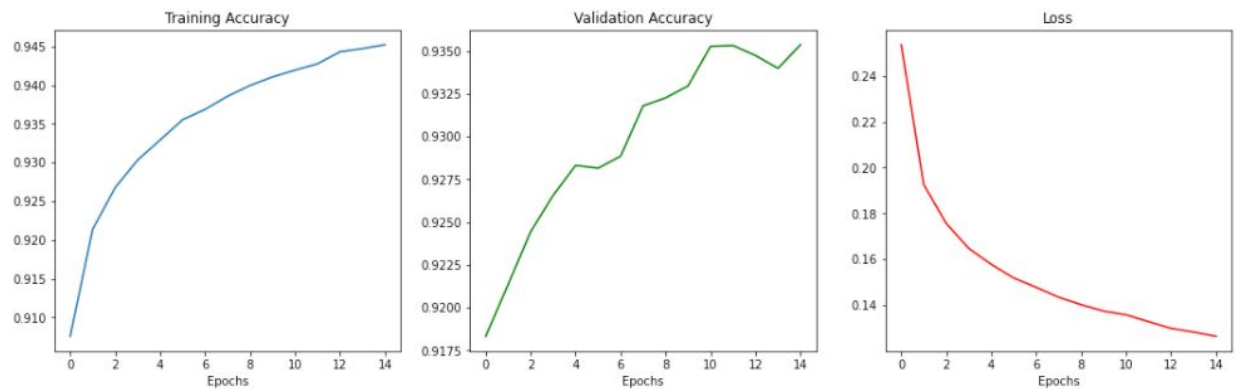
We chose ResNet V2 101 as our final model for inference as it displayed the best accuracy in training and during validation. Given that accuracy also did not significantly improve, we chose not to train the model for any longer than 15 epochs.

### Inception V3



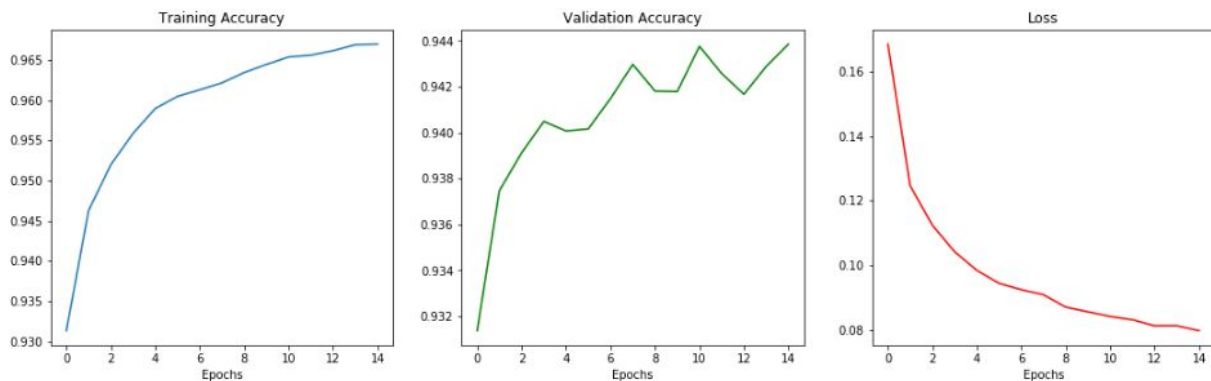
**Figure 3. Training progress for Inception V3**

## Inception-ResNet



**Figure 4. Training progress for Inception-Resnet**

## ResNet V2 101 Layers



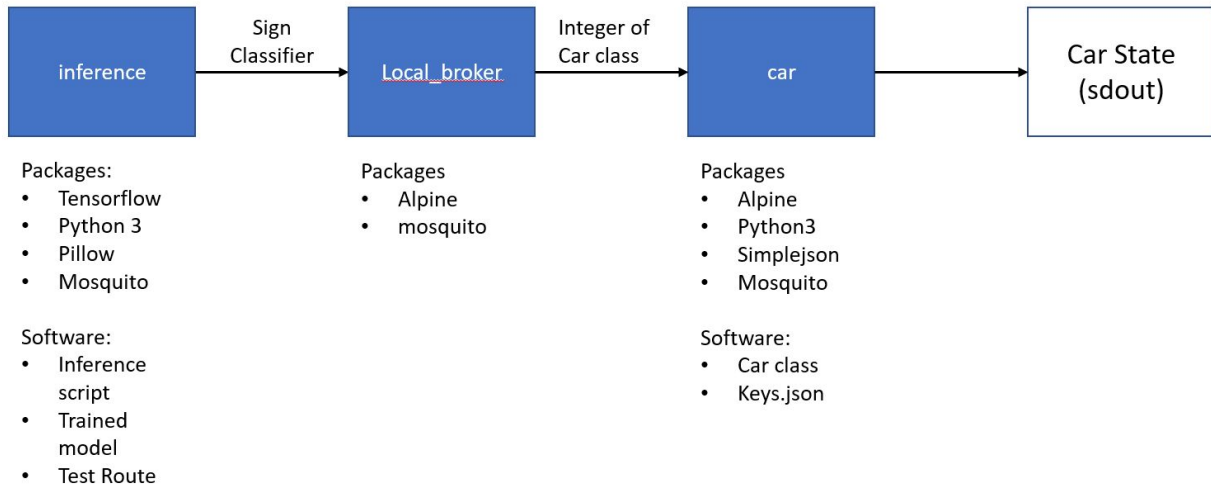
**Figure 5. Training progress for ResNet-101**

## Implementation

### Inference on the Edge

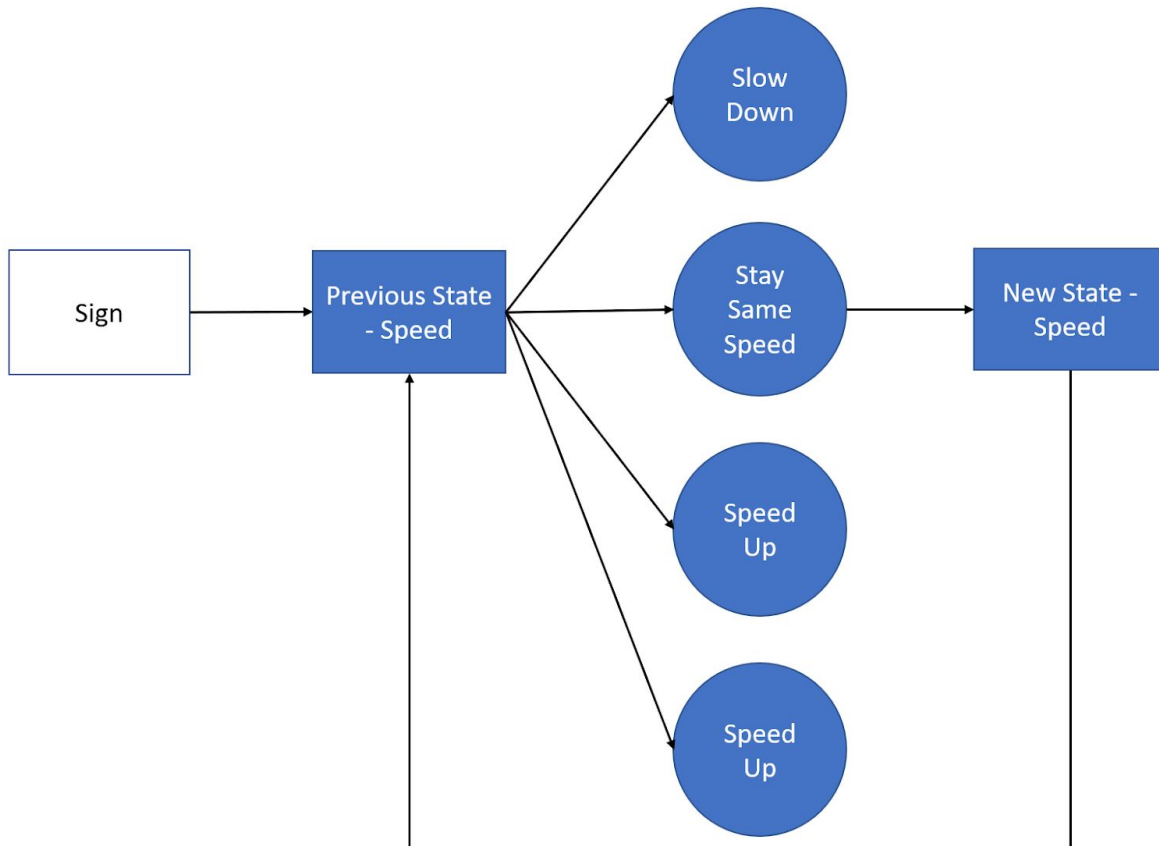
To implement our solution on the edge, we created 3 docker containers:

1. Inference contains the trained model, an executable python file and the “test route” folder of images. This container acts as the “sensor” of our self-driving car, taking visual input, running a prediction and publishing it onto the MQTT message queue.
2. Broker acted as the MQTT broker
3. Car contains an executable python file that decodes the prediction from the MQTT message queue and controls the speed of the simulated car.



**Figure 6. Diagram of containers on the Jetson TX2 for inference and action**

The car container has four states-slowing down, staying the same speed, speeding up, and stopping. The states are changed based on the previous state and the sign recognized by the inference container.



**Figure 7. Diagram of containers on the Jetson TX2 for inference and action**

## Test Results

A recorded demonstration of the inference and car modules in action can be found here:

[Recording - Inference On Jetson](#)

Sign - Actual	Prediction	Success/Failure
30	60	X
100	30	X
30	60	X
Stop	Stop	✓
Stop	Stop	✓
Yield	Yield	✓
Stop	Stop	✓
30	60	X
50	100	X
30	60	X
Yield	Yield	✓
30	60	X
50	70	X
50	60	X
Yield	Yield	✓
Stop	Stop	✓
50	100	X
100	30	X
Testing Accuracy	7/18	39%



# Conclusion

Our model had mixed performance with an accuracy of 39% on the test route. Notably, it was able to pick out the stop and yield signs with 100% accuracy but was completely unsuccessful in differentiating between the speed limits. As a result, the car accelerated when it shouldn't have and slowed down when it was unnecessary to do so. In the real world, this would undoubtedly have resulted in traffic citations and possibly an accident. Hence, since our model is not yet production ready and requires additional training and validation. We also highlight the importance of sufficient and well-labelled training data as key to training and performance.

We recommend the additional steps to further improve our self-driving car simulation:

1. Increase the amount of training data, particularly for the speed signs so that the model can better learn to differentiate between them.
2. Implement an object detection model rather than simply image classification so the input data does not need to be pre-cropped.
3. Continue to optimize hyperparameters to achieve better performance.






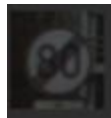

# References

1. Mapillary Traffic Sign Research Edition Dataset.  
<https://www.mapillary.com/dataset/traffic-sign>
2. Stallkamp, Johannes, et al. "German Traffic Sign Recognition Benchmark GTSRB." Electronic Research Data Archive, University of Copenhagen, [sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html](http://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html).

# Appendix



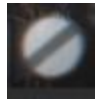
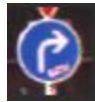
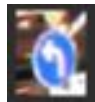
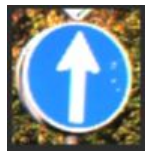

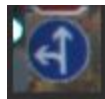

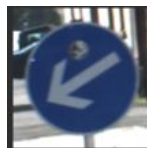
## Appendix A - Overview of The Image Dataset

**Yellow indicates the sign class was used in the final model.**

Classification Number	Sign	Example Image	Number of Images
0	20 KPH Speed Limit		210
1	30 KPH Speed Limit		2,220
2	50 KPH Speed Limit		2,250
3	60 KPH Speed Limit		1,410
4	70 KPH Speed Limit		1,980
5	80 KPH Speed Limit		1,860
6	Not 80 KPH Speed Limit (Autobahn use only)		420
7	100 KPH Speed Limit		1,440

8	120 KPH Speed Limit		1,410
9	Car Passing		1,470
10	Truck Passing		2,010
11	Priority over traffic from both side streets		1,320
12	Priority Road Start		2,100
13	Yield		2,160
14	Stop		780
15	Closed to all vehicles		630
16	Truck		420
17	No Entry		1,110
18	Caution		1,200

19	Left Turn		210
20	Right Turn		360
21	Quick Left Turn		330
22	Bumpy Road		390
23	Slippery Road		510
24	Left Merge		270
25	Men At Work		1,500
26	Light Ahead		600
27	Pedestrian Alert		240
28	Children Crossing		540
29	Bicycle Alert		270

30	Snow Alert		450
31	Deer Alert		780
32	No Symbol		240
33	Blue Right Turn		689
34	Blue Left Turn		420
35	Blue Straight Ahead		1,200
36	Blue Straight and Right Turn		390
37	Blue Straight and Left Turn		210
38	Blue Turn Around Right		2,070
39	Blue Turn Around Left		300

40	Roundabout		360
41	No Car Passing		240
42	No Truck Passing		240