NRP Traffic Sign Detection for Autonomous Driving

Student Project in the Lecture of Cognitive Systems SS18

Mariyam Fedoseeva
Hang Li
Zhen Han
Sebastian Bachem
Kai Wu
Andong Tan

Abstract

In this project, we intend to implement the visual-based traffic sign detection for autonomous driving on Neurorobotics Platform(NRP). For the traffic sign detection part, a street model with traffic signs is setup in NRP and a small dataset of 3 traffic signs is used to train a pre-trained CNN. We use the Husky robot as the car model, which has a camera mounted on it. Traffic signs in the images obtained from the robot's camera will be classified and localized. Finally, the car will adjust its motion according to the classification result of the closest traffic sign.

I. Introduction and motivation

Traffic sign recognition is one of the features of Advanced driver-assistance systems (ADAS) and also an inevitable problem for autonomous driving.

Traffic signs are usually placed by the roadside or above the road, providing information or giving guidance to road users for a safer driving. Traffics signs contains important information required for driving, such as directing drivers to drive in the correct lane at proper speed or warning drivers of obstacles and potential risks.

Forward-facing cameras mounted in vehicles is usually required to detect and recognize road signs. There are various algorithms for traffic sign recognition, such as detection based on the shape of the sign board, character recognition and deep learning methods. Nowadays, convolutional neural networks (CNN) have been used to locate and classify a large variety of traffic signs, mainly driven by the requirements of autonomous cars. The neural network will first be trained with large amount of images of the to be detected traffic signs and their corresponding "tags". The trained network can then work in real-time to detect traffic signs captured by the camera in vehicles.

In this project, a small dataset of three traffic signs (two speed limit signs and a stop sign) is used to train a pre-trained convolutional neural network. An analysis of the recognition accuracy of the neural network will be given in this report. Apart from that, a street scene with the three different traffic signs is built in Neurorobotics Platform (NRP [1]) and the Husky robot with a forward-facing camera is used as the car model. The real-time camera view of this robot will be fed in our trained CNN and the robot will response to the recognition result with a look-up-table. Thanks to NRP, where simulations are calculated by the physics engine, the dynamic performance of the car incorporating the traffic-sign recognition network will be analyzed in this work.

II. Cognitive Functions and Models Involved

A. Cognitive Functions

Several cognitive functions are implemented in our virtual car. The following subsections describe those cognitive functions and their role in the NRP car in more detail.

1) Perception and Attention: The car can sense its environment visually through the camera, and process that visual information in the tensorflow [2] neural network.

Therefore the perception of the car is comprised of the early exteroceptive perception in the camera (modality: visual), as well as the "higher level" processing implemented by the neural network, which extracts meaningful information about the traffic signs, and ignores irrelevant information about, for example, the background.

This perception process makes use of *attention* on multiple layers:

- in the convolutional neural network layers, the learned filters detect relevant features of the raw input image, while less important details are mostly ignored.
- the output of the neural network is further filtered: the car selects the sign that has the largest bounding box and ignores the other detected signs. This allows it to focus on the sign that is closest ahead, and not react to other signs that are still further ahead and therefore not relevant yet.

The above forms of attention can be classified as *selective* attention, because they allow the car to filter for the information that is relevant for its task.

2) Memory and Knowledge: The NRP car model has both short-term and long-term memory, which allows it to store knowledge.

Short term memory is used as the working memory that supports the perception-action loop: it stores the camera image until it is processed by the neural network, as well as the neuron activations during neural network inference, and the variables in the python code that post-processes the neural-network output and controls the motors.

Long term memory is implemented by the neural network weights, which can store information during training (although training is currently not done closed-loop, but open-loop). That information is stored in the saved model file, and can be retrieved during operation of the car, when the stored model is loaded and used for inference on the camera images. This long term memory stores knowledge about our NRP world, and about how traffic signs look, and how they differ from roads and background, and from each other.

Further knowledge is stored (in code) for the programmed aspects of the robot behavior, like the required motor control signals and voltages that result in velocities that are appropriate for each traffic sign.

3) Learning and Development: For our model car, learning happens in open-loop mode, when the neural network is trained. This is a form of supervised learning, and implemented by the code that feeds the training inputs and training targets to the neural network, and by the optimizer that updates the neural network weights in each training step.

Development is *not* implemented, as the car body, including sensors and actuators, does not evolve, but is fixed.

4) Autonomy: The virtual NRP car has only a rather limited form of autonomy: In a virtual world like ours, it can safely drive down a road and comply with traffic regulations. It can safely handle changes in the order, orientation and exact size of the traffic signs (behavioral autonomy). However it cannot handle environments that differ more, like have roads with curves, because lane following along road turns is not implemented.

5) Social Cognition: No social cognition is implemented in the NRP car, as the virtual world it operates in is single-actor.

6) Consciousness: We do not believe that consciousness has emerged in the NRP car.

B. Cognition Model

The NRP car implements a combination of both cognitivist and emergent principles:

The image recognition in the neural network follows the emergent paradigm, since only the architecture of the network is specified, and the image recognition capabilities emerge during training.

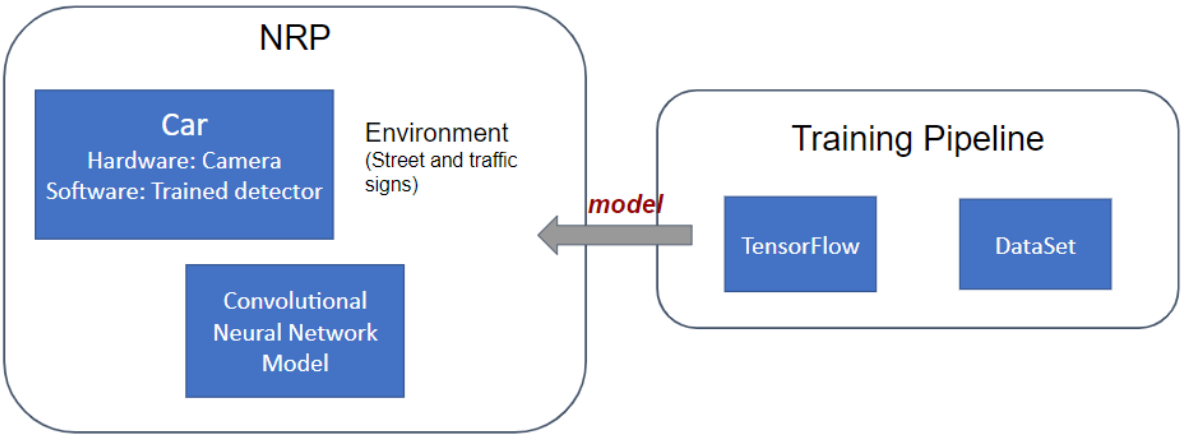

Fig. 1. Architecture of the NRP traffic sign detection for autonomous driving

Other aspects of the car, however, fit better into the cognitivist paradigm: The reactions to the detected traffic sign are encoded in rules, which provide a mapping of sign types to target speeds, and define how to control the motors. (These aspects of the behavior are implemented in the python code, as opposed to implicitly given by the neural network weights)

III. Architecture and Implementation

This chapter describes the architecture and the implementation of the traffic sign detection of an autonomous car in the neurorobotics platform.

The architecture involves NRP and TensorFlow. In NRP, a street environment is established to carry out the simulation. A car equipped with camera is set up in the environment. After starting the simulation, the car logic controls its velocity. It subscribes to rostopic ”sign”. The object detector node will publish sign value and the car responses to the value. The object detector node use TensorFlow object detection module to recognize images from video. The convolutional neural network model is trained in TensorFlow.

The implementation part will be described as follows:

- Traffic signs detection
- Car control
- Traffic signs
- Environment setup: street and car model

A. Detection of Traffic Signs

To implement the detection of traffic signs a neural network model was trained. Training was performed on pre-trained model from the tensorflow object detection models zoo, namely ssdlite mobilenet v2 model pre-trained on COCO dataset [3]. MobileNets family of computer vision models is designed for resource constraint environments while maintaining high accuracy. They target mobile and embedded devices, their careful use of resources allows us to achieve satisfactory inference speed when running the detection in NRP, which does not support the use of graphics processing units at the moment. The model also implements SSDLite framework, which is a mobile friendly version of the regular SSD framework [4]. This use of multibox detector method allows the model to perform localization of objects, which allows us to get the bounding boxes of the traffic signs.

The dataset for training was fully generated without using existing traffic signs datasets. One image per each traffic sign used in the experiment was pasted on backgrounds, which were images randomly drawn

Fig. 2. Example of an image generated to train the neural network model

from the COCO dataset. By providing sufficiently many and diverse images (not semantically related to the traffic scenes) we attempt to make our model generalize well in any environment. 3000 (three thousand) images per each traffic sign was generated as well as automatically produced annotation, which contained the class of the image and location of the traffic sign in the image by providing x coordinate minimum and maximum and y coordinate minimum and maximum. To make the detection more robust several random modifications were applied to signs in each image, such as perspective deformation, brightness change, blur and noise. In 3 we can see an example of one image from a total of 9000 images dataset.

After training was performed several model checkpoints were tested in NRP and the best performing model was selected by simple observation of the performance. The current version of the model was achieved at step 18999 with the batch of 16 images, which is less than 37 epochs. The batch size was constrained by the GPU's memory on the personal computer used for training. Hyperparameters such as learning rate were used from the default training configuration from the tensorflow object detection api. Training script and config file, as well as dataset generation script will be submitted as a project deliverable. 8 percent of images were used as validation set, breaking down the 9000 images into 720 images validation and 8280 images training set.

Inference graph was saved in the .opt/graph_def directory of the NRP container as well as accompanying label map. The training was performed using tensorflow and the transfer function of our experiment makes use of tensorflow api. Object detector transfer function runs a tensorflow session, which returns the number of detections in an image it gets from the car's camera, scores for these detections, classes and boxes, where boxes are maximum and minimum x and y coordinates of the detected traffic signs. Afterwards the square of the bounding box for each detection is calculated to determine, which of the signs has the biggest bounding box, assuming that this sign is the closest one to the car. After such sign is determined it is passed to the car's transfer function, where the car's logic is implemented.

B. Car Control

The car is set to drive straightly forward at a constant speed at its original state. When traffic signs come into the camera view, the closest sign determined by our neural network will be passed to the

transfer function by a global map variable. According to the output of the detection process, the transfer function will return a ”Twist” message, which contains a vector describing the three-dimensional linear velocity and a vector describing the three-dimensional angular velocity. This message will then published to the ROS topic ”/husky/cmd_vel” and the desired velocity will be applied by the car. The car has a base speed coefficient, which is modified in response to the detected sign. The car can accelerate/decelerate and stop. The speed level is maintained until the car detects another sign.

C. Speed Limit Signs

The traffic sign model in the experiment is the most important part of the environment as it directly influences the cars decision making. For the simplicity, only speed limit signs are to be used because this sign can already lead to a visible effect (acceleration or deceleration) on the car. The traffic sign cognition function of the car can therefore be obviously proved.

The technique used in the car to recognize the traffic sign is neural network. And the training dataset of the neural network is a fully generated dataset. As this dataset doesnt include all types of speed limit signs, the model should include the right speed limit model (rather than a random one) so that the trained model in the car can detect and recognize the speed limit sign in the right way. In the experiment, speed limit signs for 100 km/h and 20 km/h are used.

A compatible model in NRP is defined mainly by three files: “limit20.dae”, “model.config”, and “model.sdf”. The “model.config” file configures the author, model name, model version and description. The “model.sdf defines” the physical property of the model like geometry and collision property. The “limit20.dae” specifies the property of the speed limit sign in detail. For example, texture, material, and relative position of the traffic signs are defined in this file.

To speed up the model building, a reference model in “.skp” format is downloaded from the internet. To change it into a “.dae” format which is compatible with NRP, the 3D modeling software “Sketchup” in operating system Windows 10 (this software is not supported in Ubuntu) is used to convert the file to “.dae” format. The following operations are implemented in Ubuntu 16.04 operating system.

After having the above files, the scripts are edited according to the model physical property, model name, model directory, and reference pictures. A new “.sdf” file is created to configure the geometry, link, collision property with the use of the “.dae” file.

After implementing the above steps, the model should be available in the NRP platform.

D. Straight Road Section

Currently no urban road models are available in the NRP model library. To make a vivid street model and thus improve the simulation visual quality, a road model is needed. The main requirement on the road model is that it should be able to support objects on it. Since the car calculates it’s motion in the simulation world, the road need to be flat enough to avoid any unpredictable behaviors of the car. NRP models like large concrete ground and carpet are referenced to make the road model.

The road model consists of four files: model.config, model.sdf, road.dae, road.png. Since NRP is partly based on Gazebo, the model structure is quite similar to the gazebo model. A gazebo model has three levels: 3D object, model, world. NRP model is quite similar to this structure.

The road model starts from a 3D object, which is defined in ”road.dae”. Important information such as geometry and texture of the road is defined in this file. The image road.png is referenced in road.dae as texture of road surface. The geometry section has three main sources: mesh position, mesh normals, mesh map. To build our road model based on the large concrete ground, we have adjusted some values in these sources. For example, the large concrete ground has 9 sections, while we only need 3 sections to make a road. So mesh positions need to be changed. As for the texture, the dae file has a complex structure. The image is imported as library_images in road.dae. It is then used in library_effects. In the next step, it makes library_materials. Finally materials is apply to the model.

Fig. 3. A sreenshot of the environment in the traffic sign detection experiment

”model.sdf” is the bridge between a 3D model and a gazebo model. This file imports the 3D model ”road.dae”, defines its initial position, collision and visual property. The road model is now a complete module and can be used in NRP platform. This is the level of Gazebo model. A street environment corresponds to world level in Gazebo, which will be explained in the next section.

”model.config” defines descriptions of the model and author. The information is not relevant for running simulations.

E. Street Environment Setup Section

The street scene consists of three parts. A long straight road, traffic signs at the side of the road, background.

After opening an experiment, we cleared the environment and built our street scene. We imported three roads into the environment. Limit100, limit20, stop signs are placed consequently along the road. The car starts in front of limit100 as a high speed, when it percepts the limit100 sign, is slows down to follow the traffic rule. Afterwards it percepts limit20 and slows down again. It will stop when it sees the stop sign. So three signs are arranged in this order.

We put three large concrete ground models below road and traffic signs. This is only for the visualization. Because the road has collision property and can support the car. A sun light model is also added into the environment. Otherwise the street would be dark everywhere.

We also encountered a problem when we run our car on the road. The car wouldn’t follow straight lines. The reason was the friction between road and wheels of the car. They both used ode engine to calculate friction. Parameters mu and mu1 control the contact force. So parameters in the model are modified to keep the car on a straight line.

After setting up all the environment, we downloaded the world model so that we can use it later.

F. Robot Setup Section

We used husky model in our experiment. To set up the hardware part of the car, three steps are performed.

Firstly, modify parameters of the husky model in the NRP. As mentioned in the last section, the friction plays an important role in the robot motion. Since the friction parameter of this car is already large enough, we increased the mass of the car so that it has a tight contact with the road.

Secondly, import the robot model in TrafficSignsProject.bibi in the experiment directory. That's where we define which robot to use.

Finally, define the initial position of the robot in TrafficSignsProbject.exc. The car should be placed above the road and fall down to the road. Otherwise, it has the risk of being stuck in the road.

IV. Experiments and Evaluation

The objective of this project was to build an experimental implementation of traffic sign detection for a car inside the NRP. A street environment and car should be modeled in the Neurorobotics Platform. A traffic sign detection Model should be implemented in tensorflow, and the videofeed from the car's camera be processed by that model to detect traffic signs.

As a potential future step, we envisaged that the perception - cognition - action loop could be closed, and the sign detections could be fed back in real-time to the car's motor contorller and make it react to the traffic signs as it drives along the road.

In the following paragraphs, we evaluate the results of this experiment, if the project objectives could be met, if the envisaged implementation worked in practice, and also evaluate the performance caracteristics of the detection model, the core of the car's "brain".

A. Feasability of Traffic Sign Detection in a Simulated Car in NRP

The implementation that we present demonstrates that traffic sign detection in NRP is feasable. The car can drive down the road and detects the traffic signs on the roadside in real-time, without getting distracted by other objects in the scene. The car can also focus on the traffic sign that is immediately in front of it (and currently relevant), while not getting mislead by signs that are further down the road. Additionally, it remembers the most recent sign it passed, afer it is out of sight, and therefore know the corresponding speed limit, until it is orverriden by the next sing on the road.

B. A Closed Perception - Cognition - Action Loop

In addition to pure detetction, our implementation demonstrates a closed perception - cognition - action loop: the car reacts to the traffic signs in real-time, influencing its own future perception and therefore closing the loop. It drives at medium speed initially, accelerates to high speed as it passes the "100" speed limit, decelerates to low speed as it passes the "20" limit sign, and observes the "stop" sign by coming to a halt. While the sequence of the signs is fixed in our particular experiment (for greater simplicity of the NRP modeling), the car's can in principle react to any such sequence of signs appropriately.

C. Detection Performance

Apart from the overall Experiment, we also specifically evaluate the performance of the traffic sign detection model.

The detection model is based on deep learning. As deep learning is generally susceptible to overfitting due to high model complexities, it is important to consider the separation of the training and test data sets. Training was done on generated images consisting of background sceneries and images of trafifc signs that were pasted into those backgrounds. This method provides a large number of different training samples, however they all share certain caracteristis such as ligting conditions. Therefore we believe that evaluating on such a generated dataset (even if the individual samples are all distinct from those in the training set) is not ideal, as it might not detect generalization problems. Since the actual objective of the model is to detect traffic signs in NRP, we chose to evaluate the model directly on the real NRP data, which is most relevant for the task and more independent from the training dataset.

We performed multiple test runs of the NRP experiment, and checked the correctness of the detections. In order test the robustness of the system, we slightly changed the run parameters for each test run, altering the car's starting position, orientation and speed. Table I shows the parameter modifications and results for the test runs.

TABLE I

TEST RUNS OF THE NRP DETECTION CAR AND DETECTION SYSTEM: THE RESULTS SHOW THAT THE DETECTION ITSELF IS VERY RELIABLE, ERRORS OCCUR IS THE CAR STARTS OR DIRVES FARER TO THE SIDE, WHICH HAS THE EFFECT THAT THE TRAFFIC SIGNS LEAVE THE CAMERA'S FIELD OF VIEW BEFORE THEY ARE CLOSE ENOUGHT TO BE CONSIDERED RELEVANT. A CHANGE OF SOFTWARE CONFIGURATION COULD IN PRINCIPLE ALLEVIATE THIS PROBLEM, IF DESIRED.

run #	configuration	100 detected	20 detected	stop detected	note
1	x = 1.7 (default)	✓	✓	✓	-
2	(default)	✓	✓	✓	-
3	speed * 5/6	✓	✓	✓	-
4	x = 2, + 0.5 °	✓	✓	✓	-
5	x = 1	no	no	✓	to far to side, sign left field of view when approached
6	6m back	✓	✓	✓	-
7	-1 °	✓	✓	✓	-
8	13m back	✓	✓	✓	-
9	+1 °	✓	✓	no	stop sign missed because car drove to far to the side
10	x = 1.8	✓	✓	✓	-
total		90 % success	90 % success	90 % success	percentatges depend on start configuration distribution

D. Inference Speed

Since deep learning methods tend to require high computational power and real-time simulations in the NRP are desired, inference speed is an important criterion. We measured the total time spend in the object detection code per time step. In production systems, neural networks inference can be greatly accellerated when performed on GPUs or specialised neural network inference hardware, for simplicity of the NRP setup, and due to hardware availability, however, our performance benchmark was done on a general purpose CPU. Higher speeds are therefore possible in principle, with apprpriate hardware. Table II shows the results of measurements on a 2.6 GHz Intel Core i7-6600U CPU euqipped laptop. In the current configuration, this results in an overal realtime-factor of 2.1 for the complete simulation (including physics simulation, scene rendering, car control and object detection).

TABLE II

PER-STEP COMPUTATION TIME OF OBJECT DETECTION

step	time [s]
0	0.14
1	0.144
2	0.107
3	0.126
4	0.156
5	0.134
6	0.211
7	0.158
8	0.183
9	0.205
10	0.217
11	0.155
12	0.167
avg	0.162

V. Task Distribution Across Team Members

The distribution of the subtasks of this project on our team members is shown in table III

TABLE III
TASK DISTRIBUTION ACROSS TEAM MEMBERS

Mariyam Fedoseeva	NN training, inference, integration of parts, documentation
Andong Tan	asset preparation in NRP, documentation
Kai Wu	car logic, reaction to detected signs, documentation
Sebastian Bachem	evaluation, documentation, early tensorflow experiments
Hang Li	environment setup in NRP, car model preparation, documentation
Zhen Han	environment setup in NRP, documentation

VI. Conclusion

A. Summary

In this project we developed and implemented a demonstration of closed-loop traffic sign detection for the control of a virtual car on the Neurorobotics Platform.

A road scene was modeled and equipped with traffic signs of different speed limits and a stop sign, and a virtual car was set up to drive down the road. The speed of the car was controlled by the transfer function and adapted in accordance with the detected traffic signs.

Images from a virtual camera on the car were passed to a neural network model through NRP transfer functions, processed by a transfer-learned tensorflow neural network model, and the detection results were passed back to the motor control of the car.

Both correctness and speed of the neural network inference were evaluated in the NRP environment. The evaluation shows that the inference works robustly and is able to successfully detect the traffic sings.

B. Discussion

As we describe in the evaluation section, the performance of the detection model was measured on real experiment runs in our NRP scene, with some variety of starting configurations and driving speeds. Other evaluation setups are possible, like an external evaluation on some test dataset, or an evaluation within NRP, but with a different distribution of experiment configurations.

With other evaluation setups, results would not be identical: Evaluating on an external dataset could result in different performance, if the model genralises better or worse from it's training data to this test set, than it generalizes from the training data to the NRP world data.

While an evaluation on a larger dataset could show the detection behavior for a wider set of situations, this approach has the disadvantage that no such suitable datasets are available that use NRP-world data and the required set of traffic signs. Since the principle objective for this project is to implement traffic sign detection in NRP, we decided that an evaluation on a test set of NRP data was to be preferred, due to its better similarity to the relevant task.

C. Limitations

The scope of this project was to implement a demonstration scenario. The road scene is therefore relatively simple, and only a small subset of real-world traffic signs occur and need to be detected. Examples of elements that do not occur in our scenario, but do exist in real-world traffic, are, among others:

- other speed-related sings than 100km/h, 20km/h and stop
- traffic signs that are not speed related, such as informational signs, warnings, parking sings, etc
- curved roads, requiring steering and lane-following

- intersections
- other cars
- road-works
- pedestrians, cyclists, animals, miscellaneous objects
- etc.

Extending the scenario to include other traffic signs would be rather straight-forward and is very likely feasible, as long as pre-programmed reactions to the detection events are appropriate. Other limitations, such as reactions to other traffic participants, are, however, much more difficult to overcome. Therefore, and in accordance with the scope of the project, this project is to be understood as an example and demonstration of some capabilities of the Neurorobotics Platform in combination with deep learning methods, as opposed to a demonstration of advanced autonomous driving.

D. Outlook

There are several potential improvements which could be added to our current implementation in future work. Creating bigger, and more diverse NRP scenes would allow to test-run the experiment in a wider set of conditions. Implementing lane detection and -following would be the basis for running the car in a more natural road scene that includes curves, multiple roads, and intersections.

A larger-scale extension would be to enable closed-loop training, so that the car can improve it's detection capabilities and driving skills *while driving*, and potentially use learned control through reinforcement learning.

Some improvements on the usability side would be a streamlined deployment mechanism, that allows to easily and automatically install and update the experiment in an NRP installation. Also, support for external (e.g. cloud) GPUs might be helpful to enable fluent simulations independent of the locally available hardware, especially in combination with closed-loop training.

REFERENCES

[1] F. Roehrbein, M. Gewaltig, C. Laschi, G. Klinker, P. Levi, and A. Knoll, "The neurorobotic platform: A simulation environment for brain-inspired robotics," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, June 2016, pp. 1–6.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[3] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[4] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381