Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme:
Leveraging AI and Machine Learning for Societal Challenges, CAS 2019

Deep Learning Feature/Parameter Insights for Waymo's Open Dataset

Ryan Patton[a]

[a]*Missouri University of Science and Technology's Engineering Management and Systems Engineering Department, 223 Engineering Management, 600 W. 14th St., Rolla, MO, 65409-0370, United States of America*

Abstract

Improving on state-of-the-art Deep Learning applications for autonomous vehicles (AVs) necessitates a systematic breakdown of the AV general software architecture into automated and assisted functional groupings: strategic advisory (route planning), tactical assist (maneuver planning), control and assist (collision avoidance), and damage mitigation (minimum risk actions). Each functional group can be further divided into four sub-tasks: object detection, object identification and recognition, object localization, and movement prediction [11]. This paper examines object detection and labelling techniques used with Waymo's Open Datasets to compare how deep models can improve their performance and accuracy. The datasets read frames from collected camera images, create a visualization of range images, convert range images to point clouds for visualization, examine the points in each light detection and ranging (LIDAR) sensor, produce a 3-dimensional (3D) point cloud, and visualize the final outputted camera projection accounting for different lighting contrasts. By testing deep models on this foundation, transformations and their respective parameters such as translation, scale, shear, rotation, contrast, brightness, averaging, Gaussian, blur median, and bilateral filter can experience incremental improvements in their range accuracies. The opportunity to use Waymo's Open Datasets comprised of 1,950 segments each containing 200,000 frames of image data collected from diverse geographies and conditions gives credit and confidence that the deep models tested support applicable real-world driving scenarios [21]. Finally, the deep models touched on in this paper are examined for their related correlations to other applications in the autonomous driving (AD) industry.

Keywords: autonomous driving; autonomoous vehicles; object detection; object labeling; R-CNN; Fast R-CNN

1. Introduction

In a race to establish Levels 4 and 5 autonomous driving standards established by the Society of Automotive Engineers International (SAE) and seen in Table 1, deep learning, machine learning, and computer vision researchers rush to analyze newer, more comprehensive datasets collected with better hardware (SONAR, LiDAR, radars, cameras, etc.…). Although demonstrable AVs have been around since the 1980s, the field saw major gains in the 2000s when Defense Advanced Research Projects Agency (DARPA) hosted multiple autonomous vehicle challenges where the driving systems completed obstacle courses without any human intervention [4, 20]. Since then, interest in the industry has grown and as a result, datasets have transformed from a few hundred annotated examples to thousands of labeled examples capable of reproducing benchmarks for reconstruction, motion estimation, perception, recognition tasks, tracking, and closing the gap between laboratory and real-life settings. Collecting large amounts of annotated data for supervised deep models, however, presents its own challenges such as producing high quality data that can analyze at the pixel-level for optical flow, environment changes, and/or semantic segmentation [3]. Datasets traditionally split into computer vision datasets, autonomous driving datasets, and more recently, synthetic datasets for game engines, with all focuses needed for Levels 4 and 5 autonomous driving. Computer vision datasets examine object recognition, object tracking, stereo and 3-dimensional (3D) reconstruction, and optical flow. Autonomous driving datasets examine object detection and semantic segmentation, tracking, traffic sign detection, road and lane detection, flow and stereo, and long-term autonomy. Synthetic game engine datasets examine simulation effects that can translate to real driving scenarios and inverse consequences [19]. At the intersection of computer vision and autonomous driving sits Waymo's Open Dataset, used for 3D object detection, 2D object detection, 3D tracking, 2D tracking, and domain adaptation.

Table 1. Levels of Vehicle Automation

SAE Automation Category	Vehicle Function
Level 0	Human driver does everything.
Level 1	An automated system in the vehicle can sometimes assist the human driver conduct some parts of driving.
Level 2	An automated system can conduct some parts of driving, while the human driver continues to monitor the driving environment and performs most of the driving.
Level 3	An automated system can conduct some of the driving and monitor the driving environment in some instances, but the human driver must be ready to take back control if necessary.
Level 4	An automated system conducts the driving and monitors the driving environment, without human interference, but this level operates only in certain environments and conditions.
Level 5	The automated system performs all driving tasks, under all conditions that a human driver could.

Described as one of the "largest, richest, and most diverse self-driving sets ever released for research", the dataset collected from Waymo's actual autonomous vehicles operating in Phoenix, AZ, Kirkland, WA, Mountain View, CA, and San Francisco, CA, shows what driving conditions may look like at day and night and in various weather conditions. The segments of data contain synchronized data from five lidars and five front-and-side-facing cameras, sensor calibrations and poses, object labels (vehicles, pedestrians, cyclists, and signage) with 3D bounding boxes for all LiDAR frames, and object labels with 2D bounding boxes for camera data in segments. In addition to the datasets, Waymo released a Google Colab notebook and GitHub repository for tutorials and support for building deep learning models [18]. Since Waymo's Open Dataset releases in 2019, neural networks (NNs), convolutional neural networks (CNNs), region based convolutional neural networks (R-CNNs), mask R-CNNs, behavioral cloning models, recurrent neural networks (RNNs), residual neural networks (ResNets), long short-term memory (LSTM) models, StarNet

models, trajectory models, and many more models have all contributed to more complete autonomous software architectures [1].

This paper examines Waymo's Open Dataset to evaluate its deep learning relevance, proposes existing deep learning techniques and frameworks that may be applied to the datasets, and asserts the effects certain parameters and hyper-parameters may have on more fully developed models.

1.1. Problem Statement

Given the breadth and depth of the Waymo's Open Dataset, the largest LiDAR point cloud dataset to date, what deep learning models work best for effectively capturing the data and making useful autonomous driving inferences from those models for future development? LiDAR point cloud modeling for 2D/3D detection and domain adaptation struggle with sparsity and irregular formatting [2]. Past work shows it is easiest to transform point clouds to regular voxels for processing with regular convolutions, or directly estimating 3D bounding boxes with PointNet from raw cloud point data. Sensor input AV datasets present not only a challenge to come up with better algorithms for more accurate results, but also better algorithms for integrating the datasets within a 2D/3D/domain adaptation framework, with a stronger emphasis currently given on integration efforts.

1.2. Research Questions

This paper aims to bridge the gap between ever-complicated datasets used in deep learning AV applications to surmise approaches for innovative advancements. Questions that aid this discovery process include:

(1) What common parameter and hyper-parameter trends can be observed between AV datasets and other domains?
(2) What domain knowledge is required specifically for Waymo's Open Dataset and other autonomous vehicle datasets to establish trends between related work, and from general frameworks from this knowledge?
(3) What are common pre-processing techniques used for AV datasets, and for LiDAR sensors specifically?
(4) What are limiting factors for deriving the commonly used algorithms and deep learning models, and what is being done to overcome limiting factors?
(5) How does Waymo's Open Dataset advance future work in the AV industry both long-term and short-term?

Research questions (1) – (3) aim to set a transitory framework for exploring deep learning models in the AV industry. Questions (4) – (5) build on the knowledge gained from questions (1) – (3) to determine how best new models may be applied to specialized problems such as perception or mapping, and how new, seemingly non-overlapping deep learning research areas in autonomous driving can drive systematic advancements.

1.3. Research Objectives and Contributions

To the best of the author's current knowledge, this paper claims to be the first of its kind to investigate deep learning models used with Waymo's Open Dataset and develop a parameter/hyper-parameter framework that can be applied to future AV datasets. While work done by [12, 15] develops a driving model for existing level-5 autonomous cars using Waymo's sensor data, the work proposed in this paper uses the same sensor data to show what work has been done in proposing deep learning models, and show a mutually beneficial relationship between enriched sensor datasets in the future and the effects certain parameters and hyper-parameters have on the various models, both present and future. Due to organizational privacy and proprietary restrictions, data is limited for how the applications of this research may look in real-life driving scenarios, at both an individual and fleet level [13]. To overcome these limitations, the effectiveness of parameters and hyper-parameters are evaluated based on the training results from R-CNNs and variations of R-CNNs specific in the AV domain [9]. The relationship explored in this paper from the input features and model's parameters show how future AV deep learning models may benefit from quicker pre-processing techniques by grasping the fundamental trends between the AV's sensor inputs.

2. Review of Related Literature – Deep Learning Models

To gather proficient knowledge and understand the intricacies of the Problem Statement, three related literature topics need to be addressed: (1) autonomous vehicles (AVs) and relevant hardware/software architectures; (2) Waymo's Open Dataset Challenge past progression; and (3) deep learning models used in both the AV industry and in particular, for Waymo's Open Dataset.

To understand the software architecture, the ways AV hardware interacts with the software to gather sensor input is just as important. Figure 1 shows a typical AV architecture based on a modified 2011 Ford Escape that was the first to travel 74 km on urban roads and highways in Brazil [16]. Further studies of AV architecture reveal 15 general filter modules for receiving sensor data and transferring data into actionable output: (1) GNSS module – transforms latitude and longitude data into x, y, z coordinates; (2) Map Server module – provides offline map to give AV's current position; (3) Localizer module – localizes AV on offline map and generates point clouds from LiDAR to place AV in 8 dimensions that are x, y, z, roll, pitch, yaw, v, and phi; (4) Moving Objects Detector module – detects moving obstacles and their speed; (5) 2D Mapper module – creates online map of path and static/moving obstacles around AV; (6) ODGM Generator module – generates ODGM from a fused 2D map; (7) Lane Detector module – identifies path markings; (8) Traffic Sign Detector module – detects and recognizes traffic signs; (9) Traffic Light State Detector module – detects traffic lights along path and identifies status of lights; (10) Path Planner module – computes a path from AV's current position to goal position; (11) Behavior Selection module – establishes goal for AV and suggests path(s) to goal; (12) Motion Planner module – constructs trajectory from current position to goal; (13) Obstacle Avoider module – evaluates current path to change velocity and direction if necessary; (14) AV module – driver of car; and (15) Health Monitor module – checks for functioning of all modules [16].

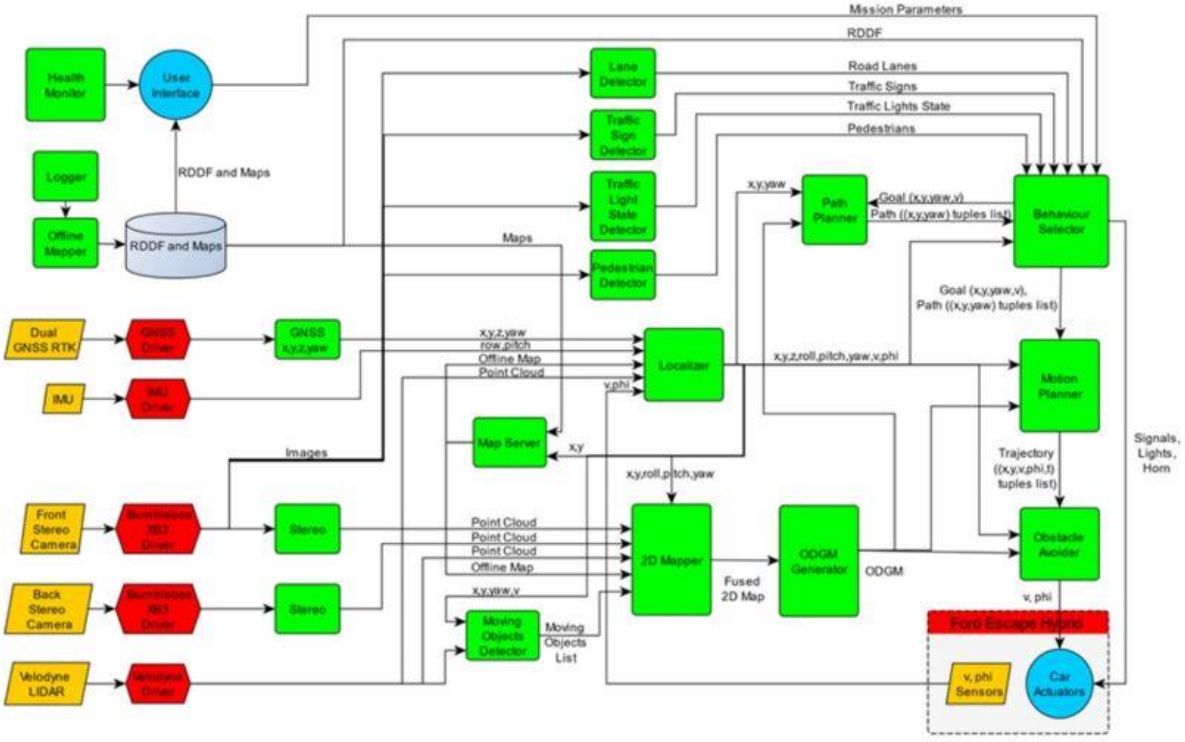

Fig. 1. Intelligent Autonomous Robotic Automobile's (IARA) Architecture

Studying the progress on Waymo's Open Datasets, reveals the 3D detection and domain adaptation winning team's architecture heavily relied on anchor-free one-stage 3D detection (ADFet), and consisted of using a point

cloud encoder, 3D feature extractor, region proposal network (RPN), and an anchor-free detector. The point cloud encoder is the stage where they make use of their CNN framework, and some parameters along the way are: Conv2-128 and t = 4, Conv3-256 and t = 1, Conv3-128 and t = 1, Conv3-128 and t = 1, batch normalization, ReLU, AdamW optimizer with one-cycle policy, learning rate set to 3×10^{-3}, division factor to 2, momentum ranges from 0.95 to 0.85, weight decay to 0.01, sampling at every 20 frames, validating at every 10 frames, weights between 0.3 and 1.5 for size, z, off, and orientation, training on 1/20 training data, testing on 1/10 validation data, and epochs at 60 for initial training and 10 epochs for whole the whole dataset's evaluation [10].

For 2D object detection and tracking, the winning team's architecture utilized Cascade R-CNN for setting different threshold object detections, and use very deep CNNs with minimal difference, namely ResNet152, ResNet50, and ResNet101. The images are scaled and augmented but little is revealed about additional parameters on their model past training it at 20 epochs and randomly sampling 120,000 images for training and validation [14].

Many AV models make use of R-CNNs, first proposed in 2014, for their object detection work because they break images down by regions in each image before feeding into the CNN for more accurate results. ReLU is another common activation function used because it allows for stochastic gradient descent with backpropagation of errors that many deep neural networks train with due to its computational simplicity, representational sparsity, linear behavior, and its ability to exploit improvements in hardware [5]. Adam optimizer is commonly used in AV data applications because it has little memory requirements, it is well suited for problems that are large in data and/or parameters, it is appropriate for noisy and/or sparse gradients, it is appropriate for non-stationary objectives (like in AVs), and most importantly, hyper-parameters have intuitive interpretation and typically require little tuning [6]. When training on full datasets, epochs are typically kept under 100 max and that scales down quickly with larger datasets. Most other parameters for AV deep learning models depend more on the dataset and objectives from the dataset.

3. Research Methodology – Deep Learning Models Tested

3.1. Waymo's Open Dataset

The work presented makes use of Waymo's official repository which contains definition of the dataset format, evaluation metrics, and helper functions in TensorFlow to aid building models. The helper functions provide sufficient guides for installing waymo_open_dataset package needed to work with the dataset (terabytes of data needed for downloading otherwise), reading the frames and examining the frames' contexts, visualizing camera images and camera labels, visualizing range images, converting to point clouds for visualization, examining the number of points in each LiDAR sensor, showing the generated point clouds, visualizing camera projections, and additional code for installing dependencies for a better API that includes metrics computation [17]. The input features seen in Table 2 provide the analysis baseline for testing the model in this paper.

Table 2. 12 Input Features from Waymo's Open Dataset

Features	Feature Definitions
$V_X, V_Y, V_Z \in \mathbb{R}$	The velocity of AV in x, y, z directions.
$D_X, D_Y \in \mathbb{R}$	The distance from AV to the front car in x, y directions.
$V_{FX}, V_{FY}, V_{FZ} \in \mathbb{R}$	The velocity of front car in x, y, z directions.
$A_{FX}, A_{FY}, A_{FZ} \in \mathbb{R}$	The acceleration of front car in x, y, z directions.
num_v_labels $\in \{0, 1\}$	The number (existence) of front vehicles, which describes the complexity of the scenario.

3.2. Configuration & Pre-Processing

Using Waymo's helper functions eliminates the need for most pre-processing efforts. When the files are downloaded, transformation code is needed because the files download as .tfrecord files. Other existing methods use

a KITTI adapter (note: KITTI is another earlier dated object detection dataset), frameworks that are not reliant on TensorFlow and Bazel and do not pull many dependencies, additional python scripts on top of Bazel to format data to the user's liking, and dataset size converters (so user would only pull 1/10 the size of the dataset compared to the whole dataset). The author's work explored some alternative/add-on methods listed above but used Waymo's official methods for pre-processing. Data augmentation was also used during pre-processing to avoid overfitting.

Starting with given .jpeg images, the architecture went through the following pipelines before intrinsic evaluation of the 12 features: .jpeg images to convolutional layer, to batch normalization, to max pooling, to convolutional layer again, to batch normalization again, to max pooling again, and finally to dense layers that are outputting image representations. The .jpeg image representations after going through the pipeline meet up with the 12 features fed into dense layers to be trained together through dense layers for final output. The single input channel used for the 12 features are useful as the encoder finds key information from input features to generate a latent representation of features. The final output seen after decoding outputs relative acceleration and positioning to compare with validation data.

3.3. R-CNN & Variations of R-CNN

Convolutional Neural Networks (CNNs), widely used within the computer vision field for their relatively lower error rates, provide a jumping off point for Region Based Convolutional Neural Networks (R-CNNs). R-CNN, Fast R-CNN, Faster R-CNN, YOLO, Mask R-CNN, and many CNN derivations. This experiment makes use of R-CNN with various cropped image sizes, max pooling, strides of (5, 5) and (3, 3), filters of 24, 36, 48, and 64, a loss in terms of mean absolute error, 8 epochs, dropout regularization, ReLU and sigmoid for activation, and Adam optimizers.

The R-CNN model is constantly compared to its global frame, vehicle frame, sensor frame, image frame, and LiDAR spherical coordinate system (Cartesian coordinate system). The important equations for comparing the AV's positioning, velocity, and acceleration against the Cartesian coordinate system are:

$$range = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

$$azimuth = atan2(y, x) \tag{2}$$

$$inclination = atan2(z, \sqrt{x^2 + y^2}) \tag{3}$$

4. Implementation & Results of Parameter Model

The R-CNN model noted the hyper-parameters most important to success: the length of the previous frames, length of future frames, and training epochs. If the frames are too long, the model struggles with complexity and is costly. If the model is too short, it cannot predict accurately and reflects poor performance. For future predictions, the model is again too complex if it is too long. The previous frames' length was tried at 5, 10, 15 and 20. The future frames' length was tried at 2, 3, 4, and 5. These balances show the related distortion between common frameworks and related perceptions.

The performance of the model is based on comparing training and validation datasets at $1/30^{th}$ of the whole using mean absolute error. Mean absolute error provides a better method of capturing the results compared to mean squared error because the units of data needed for positioning were crucial and large errors in prediction were not as much of a concern with validation data readily available. The results for all sorts of scenarios were captured but some of the more meaningful results can be seen by consulting the datasets and comparing those images for different positionings and maneuverments seen in Figures 2, 4, and 6. The MAE for each maneuver can be seen in Figures 3, 5, and 7.

Finally, Figure 8 shows what camera synchronization looks like between frames, using ground truth as a reference for validation.

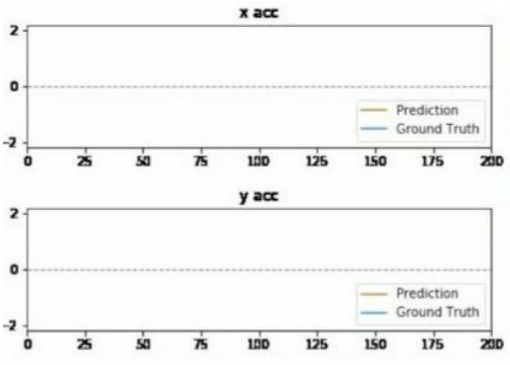

Fig. 2. Scenario: Following Vehicle from Behind

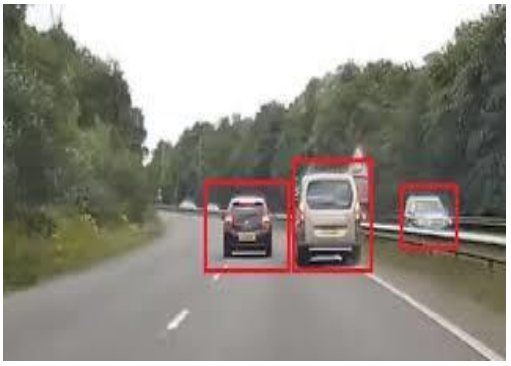

Fig. 3. Visualization of Following Vehicle from Behind

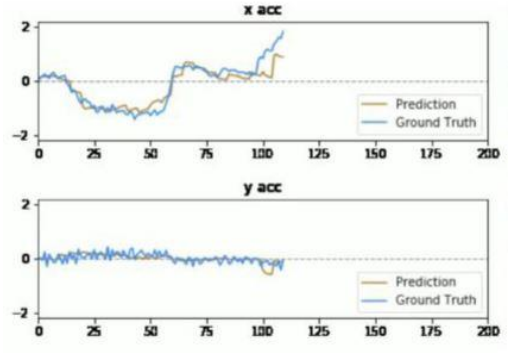

Fig. 4. Scenario: Vehicle in Front is Turning

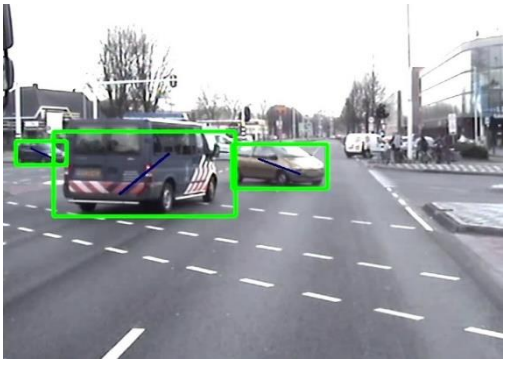

Fig. 5. Visualization of Car in Front is Turning

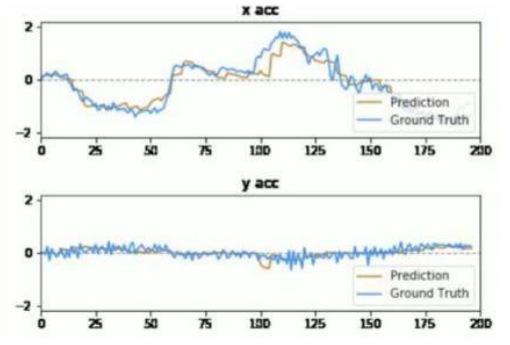

Fig. 6. Scenario: Applying Brakes at a Stop Light

Fig. 7. Visualization of Applying Brakes at a Stop Light

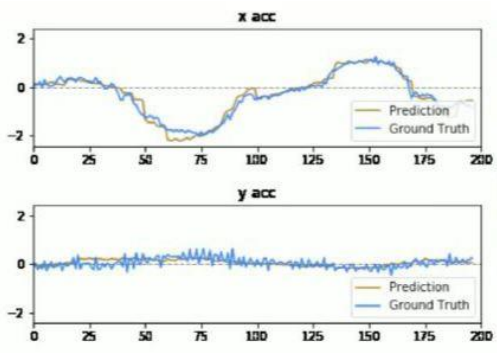

Fig. 8. Synchronized Framing between Camera Angles

Figures 2 – 7 show crucial driving scenarios and what acceleration (as a derivative of position and velocity against time) looks like with respective images compared to validation ground truth data. To recap, Figure 8 shows how each of the 6 camera angles in each frame compare to one another as an extension of the work conducted to capture Figures 2-8. Figures 2-8 are not comprehensive, their purpose is to illustrate the shifting dynamics of object detection in motion, and how object detection works relative to the AVs frame of reference. Table 3 seen below shows the loss for each input feature except complexity (velocities in x, y, z direction; distance to front of car; velocity of front of car; acceleration; and number of front vehicles for mimicking complexity) between training and validation.

Table 3. Input Features Loss

Features	Feature Definitions	Training Loss	Validation Loss
$V_X, V_Y, V_Z \in \mathbb{R}$	The velocity of AV in x, y, z directions.	0.11	0.03
$D_X, D_Y \in \mathbb{R}$	The distance from AV to the front car in x, y directions.	0.09	0.04
$V_{FX}, V_{FY}, V_{FZ} \in \mathbb{R}$	The velocity of front car in x, y, z directions.	0.18	0.15
$A_{FX}, A_{FY}, A_{FZ} \in \mathbb{R}$	The acceleration of front car in x, y, z directions.	0.19	0.15
num_v_labels $\in \{0, 1\}$	The number (existence) of front vehicles	0.13	0.08

5. Conclusions

The results indicate close interaction between the AV's pose, coordinates, velocity, and acceleration. This relationship extends out to 2D/3D object detection and tracking and domain adaptation because the input features fed into R-CNN must take into account object detection. The results within provide a framework for applying the principles into more robust and complicated CNN-architectures similar to the results obtained by the winners of Waymo's Open Dataset competition. Referring back to Research Questions, the scope of this work provides introductory performance that can utilize the parameter and hyper-parameter relationships to gain a better understanding of the way the AV interacts with objects surrounding it. The results of the experimentation were not limited by the amount of available data as only 1/30 was sampled, however, implied limitations might include higher quality images generated, images generated in more environments, and images generated with more objects pre-labeled. Waymo's Open Dataset could be expanded by including the AV's coordinate measurements as well as mapping localization metrics, and how the AV performs at a fleet level. The directions, velocity, and acceleration of the AV as it relates to real-life driving scenarios can achieve imperfect but gaugable metrics for evaluating the AVs performance in various driving conditions.

6. Future Work

The natural progression from the work in this paper is to apply the lessons learned from the vehicles' relative pose, velocity, and acceleration to the driving scenarios within the dataset for improved 2D/3D object detection and labeling. Future work may also correlate simulated datasets on game engines to real-life performance as seen here. Simulated datasets can train from results seen here to improve the effects of an AV's pose on object detection and handling of the AV in anomalous circumstances [7, 8]. While Waymo's Open Dataset targets 2D/3D object detection and tracking and domain adaptation, pose is related to a number of other AV deep learning applications such as long-term route planning, map localization, and extended range perception.

References

[1] Autonomous Vehicle Implementation Predictions. (n.d.). Retrieved from https://vtpi.org/avip.pdf

[2] Badue, C., Guidolini, R., Carneiro, R., Azevedo, P., Cardoso, V., Forechi, A., … De Souza, A. (2019, Octobe 02). Self-Driving Cars: A Survey. Retrieved from https://arxiv.org/abs/1901.04407

[3] Bresson, G., Alsayed, Z., Yu, L., & Glaser, S. (1970, January 01). [PDF] Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving: Semantic Scholar. Retrieved from https://www.semanticscholar.org/paper/Simultaneous-Localization-and-Mapping:-A-Survey-of_Bresson-Alsayed/db31ea6597cb76b4294767982709e17a6744e2

[4] A Brief Survey of Autonomous Vehicle Possible Attacks, Exploits, and Vulnerabilities. (2018, October 03). Retrieved from https://deepai.org/publication/a-brief-survey-on-autonomous-vehicle-possible-attacks-exploits-and-vulnerabilities

[5] Brownlee, J. (2020, August 20). A Gentle Introduction to the Rectified Linear Unit (ReLU). Retrieved from https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks

[6] Brownlee, J. (2020, August 20). Gentle Introduction to the Rectified Linear Unit (ReLU). Retrieved from https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning

[7] Chen, X., Cho, H., Ciregan, D., Cubuk, E., T. DeVries and G. W. Taylor, Dwibedi, D., … Zoph, B. (2020, April 02). Improving 3D Object Detection through Progressive Population Based Augmentation. Retrieved from https://deepai.org/publication/improving-3d-object-detection-through-progressive-population-based-augmentation

[8] Cheng, S., Leng, Z., Cubuk, E., Zoph, B., Bai, C., Ngiam, J., … Anguelov, D. (2020, July 16). Improving 3D Object Detection through Progressive Population Based Augmentation. Retrieved from https://arxiv.org/abs/2004.00831/

[9] Deep Multi-modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. (n.d.). Retrieved fromhttps://www.researchgate.net/publication/331198137_Deep_Multi-modal_Object_Detection_and_Semantic_Segmentation_for_Autonomous_Driving_Datasets_Methods_and_Challenges

[10] Ding, Z., Hu, Y., Ge, R., Huang, L., Chen, S., Wang, Y., & Liao, J. (2020, June 28). 1st Place Solution for Waymo Open Dataset Challenge – 3D Detection and Domain Adaptatio. Retrieved from https://arxiv.org/abs/2006.15505

[11] Huang, Y., & Chen, Y. (2020, July 04). Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies. Retrieved from https://arxiv.org/abs/2006.06091

[12] Issues in Autonomous Vehicle Testing and Deploymet. (n.d.). Retrieved from https://crsreports.congress.gov/product/pdf/R/R45985

[13] Janai, J., Guney, F., Behl, A., & Geiger, A. (2019, December 17). Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. Retrieved from https://arxiv.org/abs/1704.05519v2

[14] Latt, A. (2020, June 15). 2D Detection on Waymo Open Dataset. Retrieved from https://medium.com/@lattandreas/2d-detection-on-waymo-open-dataset-f111e760d15b

[15] An LSTM-Based Autonomous Driving Model Using Waymo Open … (n.d.). Retrieved from https://arxiv.org/pdf/2002.05878

[16] Ngiam, J., Caine, B., Han, W., Yang, B., Chai, Y., Sun, P., … Vasudevan, V. (2019, December 02). StarNet: Targeted Computation for Object Detection in Point Clouds. Retrieved from https://arxiv.org/abs/1908.11069

[17] Open Dataset. (n.d.). Retrieved from https://waymo.com/open

[18] Scalability in Perception for Autonomous Driving: Waymo … (n.d.). Retrieved from https://openaccess.thecvf.com/content_CVPR_2020/papers/Sun_Scalability_in_Perception_for_Autonomous_Driving_Waymo_Open_Dataset_CVPR_2020_paper.pdf

[19] Sinha, G. (2020, August 10). Deep Learning method for object detection: R-CNN explained. Retrieved from https://towardsdatascience.com/deep-learning-method-for-object-detection-r-cnn-explained-ecdadd751d22

[20] A Survey of Autonomous Driving: Common Practices and … (n.d.). Retrieved from https://arxiv.org/pdf/1906.05113v2.pdf

[21] Waymo-Research (n.d.). Waymo-research/waymo-open-dataset. Retrieved from https://github.com/waymo-research/waymo-open-dataset