2023 2nd International Conference for Innovation in Technology (INOCON)
Bangalore, India. Mar 3-5, 2023

Network Optimization of Large-Scale CNNs for Applications in Autonomous Machines

Harish Rohan Kambhampaty
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, India
harishrohank@gmail.com

Naga Datha Saikiran Battula
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, India
battulasaikiran2002@gmail.com

Salwesh Rentala
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, India
rsalwesh@gmail.com

Sai Charan Challa
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, India
sricharansai27@gmail.com

Anuradha K.
Computer Science & Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, India
kodalianuradha8@gmail.com

***Abstract*—In the age of automation, a growing number of devices require high dimensional data to increase the reliability of the machines to better predict and classify objects for autonomous decision-making. 3D object detection directly links to environmental interpretation and therefore builds the base for prediction and movement planning for autonomous systems. With greater computing and storage capacities at the edge level, higher-density data can be captured and analyzed, and hence better predictability of natural environments is achieved. Convolutional Neural Networks are becoming increasingly important for object detection and tracking in many autonomous systems due to the improvement of affordable edge-computing devices capable of working with higher-dimensional data, such as LiDAR sensors, in recent years.**

Although the capabilities of edge-computing devices are growing, the computational requirements for real-time applications are also rising. This paper proposes two types of optimizations to the networks: pruning and quantization. The proposed optimizations are applied to a base model trained on the KITTI dataset, which shows an average inference speed-up of up to 20% while minimizing the loss of performance.

***Index Terms*—Network Optimization, CNN, Point-Cloud, 3D Object Detection, Autonomous Systems**

I. Introduction

One of the most significant problems the robotics community still needs to address is object detection in dynamic, real-time situations. A dynamic situation refers to a work setting involving a human operator that only partially controls a physical or technical process independent of the system's dynamics [1]. A real-time situation involves an action by the system with no noticeable delay from its effect (or consequence) [2]. A solution should incorporate segmentation, tracking, and classification elements and enable the insertion of new object classes without requiring a specialist's specification of new models.

The availability of dense 3D sensors in recent years presents the perfect opportunity to investigate an extensive dataset method to track categorization. With these sensors, independent of object type, every item in the environment may be segmented and tracked using data. This works particularly well in the context of driving, as objects actively try to keep well separated from one another. Identifying an item that has been appropriately segmented and tracked can offer hundreds or thousands of training examples with only one key press. Tracking and segmentation considerably simplify the process of labeling massive volumes of data.

In the realm of autonomous driving, object detection technology is also crucial. The area of autonomous driving has long been a focus of automotive research. Driverless and Advanced Driver Assistance Systems (ADAS) are the two categories of autonomous driving systems that currently exist [2]. The latter focuses on helping the driver, lowering the driver's stress while driving, and enhancing vehicle safety. This form focuses on partial autonomous automobile driving to reduce the driver's workload. Both use various onboard sensors to gather information, integrate it with map data for system calculations, and direct the vehicle to arrive at a preset location. Artificial intelligence developments like those in computer vision and deep learning are crucial to the advancement of autonomous driving.

979-8-3503-2092-3/23/$31.00 ©2023 IEEE

A. Point-Cloud Data

Point cloud data is a collection of physical points in space. The most notable feature of point cloud data is depth information, which can be used to perceive the objects' exact location and better understand the surroundings using point cloud data. Due to the recent advancements in low-cost LiDAR (Light Detection and Ranging) sensors, point-cloud processing is becoming increasingly crucial for object detection and tracking in autonomous driving and many other applications.

One benefit of object detection using point clouds is scale consistency in a three-dimensional environment. One of the most well-known sensors for producing 3D data of objects to characterize forms and locate items is LiDAR. Many well-known, cutting-edge 3D object detectors, including YOLO, PointNet, and 3DOP, have recently been proposed for 3D object detection. Further, real-time video stream processing has also been used to classify objects.

Finding all Regions of Interest in the image and determining their locations is the goal of object detection. Approximating an object's size and location in a three-dimensional environment is the goal of 3D object detection. In contrast, 3D object classification entails classifying detected items according to their shape, size, volume, and various other physical features.

Self-driving cars often employ LiDAR (Light Detection and Ranging) alongside many other cameras and sensors placed in various parts of the car to gather perceptual data. The visual data is then analyzed and localized to detect items like lanes, automobiles, and pedestrians. The capacity of computers to interpret visual data has increased dramatically compared to conventional approaches thanks to the rapid development of deep learning and artificial intelligence technologies, and several businesses have started to produce artificial intelligence-specific processors to enable Edge Computing.

LiDAR, cameras, and millimeter-wave RADAR are common sensor types used in intelligent driving systems. The advantages of LiDAR include 3D modeling, a broad detection area, and excellent detection precision. Deep learning target identification systems incorporating LiDAR detectors are a popular area of research. The existing deep learning-based LiDAR detection systems still have underlying issues that need to be resolved for broader acceptance.

The first challenge is in the kind of data generated by a LiDAR. Typical deep learning target identification algorithms cannot be employed directly for LiDAR perception because the data generated by the LiDAR sensor is a sparse point cloud, which is less dense than the picture output by the camera. The second challenge is the three-dimensional data produced by the point cloud by LiDAR, which increases the computational requirements due to the addition of the third axis in the coordinate system.

Autonomous systems need high dimensional data to improve reliability and better forecast environmental change for autonomous decision-making. The foundation for prediction and movement planning for autonomous systems is built by the direct connection between 3D object detection and environmental interpretation. Higher-density data can be collected and analyzed at the "edge" level thanks to recent improvements in processing and storage capacities, improving the predictability in natural settings. Despite some promising work in this field, reliable 3D object detection remains an unresolved challenge.

CNNs are notorious for high computational requirements. Network Optimization is the optimization of the CNN network structure and its associated weights to improve the model's performance. This paper proposes Network Optimizations for Deep-Learning solutions involving large Convolutional Neural Networks (CNNs) and studies the effects of using such techniques.

II. Literature Survey

In this section, some notable prior and existing methods that have been proposed and implemented by various sources will be described briefly.

Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross proposed Complex-YOLO, a 3D object detection CNN that uses only point clouds [3]. It is based on YOLOv2, a fast object detection network for 2D RGB images. The authors propose a specific Euler-Region Proposal Network (E-RPN) that is capable of estimating the pose of the object by the addition of an imaginary and real fraction to the regression network. The experiments were carried out on the KITTI [4] dataset.

Fernando Julio Cendra, Lan Ma, Jiajun Shen, and Xiaojuan Qi present a framework for unsupervised 3D recognition [5]. The proposed SL3D framework solves two objectives: clustering and learning feature representation to generate pseudo-labeled data. It is a wide-ranging framework that may be used for many 3D recognition tasks, including semantic segmentation, object detection, and classification. Its goal is to overcome the scalability limitations in 3D recognition tasks caused by its requirement of using annotations for model training. The efficiency of the SL3D framework is confirmed through extensive experiments.

Y. Shen et al. present a new 3D object detection paradigm called ImLiDAR which seeks to combine data from two cross-sensors, LiDAR and camera sensors [6]. The proposed method uses a cross-sensor dynamic message propagation module to combine the best of multi-scale image and point features. In order to deal with the irregular nature of the classification and localization confidences as well as the sensitivity of hand-tuned hyperparameters, the paper also puts forth a direct set prediction problem that allows the creation of a set-based detector that is effective. The results show clear improvements in ImLiDAR over 23 state-of-the-art 3OD methods on SUN-RGBD and KITTI datasets.

A. Tamajo et al. present a graph convolution-based unit called the Shrinking unit for designing CNN-like 3D point cloud feature extractors [7]. According to the authors, the difference in performance between automated point cloud systems and their picture counterparts is mainly due to the lack of design inspiration from the image domain. Using internal, local, and global correlations between points in the point cloud, the suggested Shrinking unit may be stacked

horizontally and vertically to extract features from the point cloud data. The method's performance was evaluated using the ModelNet-10 benchmark dataset, where it achieved a classification accuracy score of 90.64%.

Y. Yao et al. propose a knowledge distillation method for 3D point cloud pre-trained models to acquire knowledge directly from 2D representation learning models, specifically the image encoder of CLIP [8]. The authors introduce a cross-attention mechanism to extract concept features from the 3D point cloud and compare them with the semantic information from 2D images. The proposed method is evaluated on various downstream tasks, which shows that it outperforms the state-of-the-art 3D pre-training methods on synthetic and real-world datasets.

III. Methodology

This section details the model's architecture, the data sources used, and the optimization techniques utilized. The dataset has a 3D point cloud data with the images and calibration files for data points which are taken using a data loader. The target class contains three categories: Car, Pedestrian, and Cyclist. The preprocessing step includes random rotation (20-degree range), scaling (5% range), and horizontal flipping. The batch size of the data loader is set according to the VRAM in the GPU. The data is divided into training, testing, and validation sets.

YOLOv4 is one of the most popular object detection architectures based on the Darknet Framework [9]. The network architecture uses the mish activation function while applying batch normalization to the YOLO architecture layers. Upsampling and Downsampling have been used to improve the speed and stability of the model. The Adam optimizer has been chosen since it best combines the properties of AdaGrad and RMSProp algorithms that can handle sparse gradients on noisy problems. The training is performed for 150 epochs.

A. Optimization Techniques

Large CNN networks often have suboptimal runtime performance due to high memory and processing power requirements, drastically increasing the inference speed. Pruning and quantization are often applied to improve energy savings and inference speed without significantly increasing accuracy losses in the model. The improvement in efficiency significantly affects the affordability of edge devices capable of autonomous driving and other complex tasks.

1) Pruning: Pruning involves the elimination of certain weights to decrease inference requirements and reduce the model size. There are two main pruning technique types:

1) Structured pruning: involves eliminating weights by actively modifying the network, i.e., the total number of parameters is reduced , as shown in Fig. 1. In this paper, two techniques have been implemented:
   a) Ln-Structured Pruning
   b) Random-Structured Pruning
2) Unstructured pruning: involves eliminating weights by zeroing any weights that do not significantly affect the prediction, as shown in Fig. 2. In this paper, two techniques have been implemented:
   a) L1-Unstructured Pruning
   b) Random-Unstructured Pruning

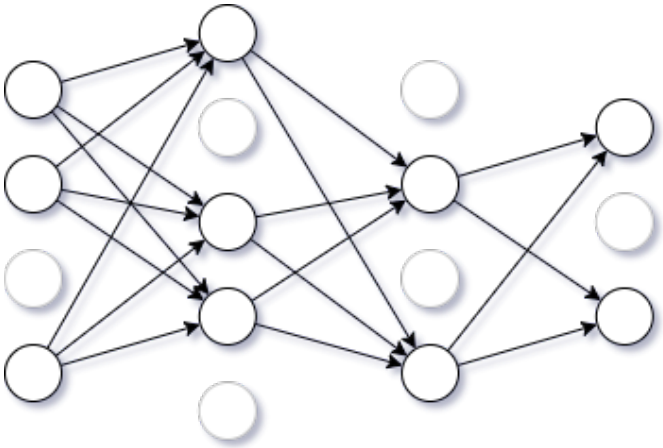

Fig. 1. Structured Pruning

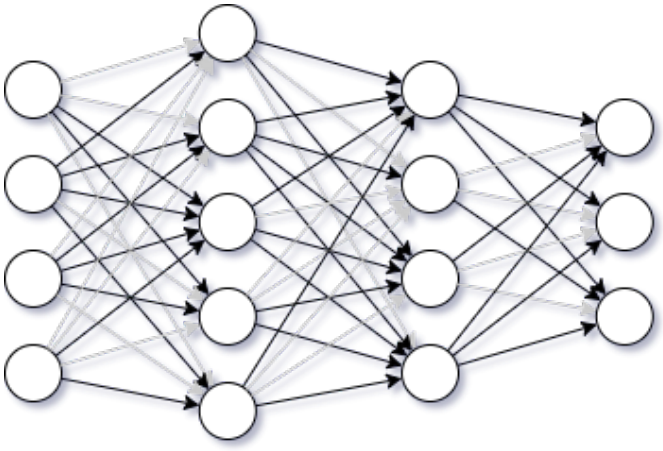

Fig. 2. Unstructured Pruning

Both types of pruning techniques have been studied in this paper. Pruning was performed to eliminate 2, 4, 7, 10, 15, 20, 30, and 40 percent of nodes/weights. Section IV contains more details about the results and analysis.

2) Quantization: Quantization involves lowering bit-widths from floating-point precision (to INT8 and FLOAT16) by approximation, drastically reducing the memory requirements. There are two main quantization technique types:

1) Post-Training Quantization (PTQ): involves conversion of bit-widths after the complete model has undergone training. There are two techniques that are commonly used:
   a) Post-Training Quantization - Static (PTQ-S): involves fusion of activation functions into preceding layers of the model wherever possible. Calibration

dataset is required for determination of optimal quantization parameters. PTQ-S is most effective when compute efficiency needs to be increased and memory bandwidth is low.

b) Post-Training Quantization - Dynamic (PTQ-D): involves conversion of bit-widths after the complete model has undergone training, i.e., weights are quantized ahead of time. PTQ-D is most effective when loading weights from memory dominates the model execution time.

2) Quantization-Aware Training (QAT): involves conversion of bit-widths during the training process, i.e., between epochs. The models influence quantization for achieving higher accuracy. During training, the weights are calculated with floating-point precision while simultaneously applying rounding and clamping to simulate the lower bit-width. After conversion, the activation functions and weights are quantized while the activation functions are fused onto the preceding layers. There are three modes of QAT:

a) Static
b) Dynamic
c) Weight-only

PTQ-D techinique has been studied in this paper. PTQ-D was applized on the Linear layers of the network. Section IV contains more details about the results and analysis.

B. Dataset

The KITTI dataset [] was used in the implementation. The KITTI dataset consists of more than 12919 images. Four datasets were used for training the model which include:

- Velodyne Point-Cloud Data: 3D point-cloud data that serves as training and testing data for the model. Totally 29GB in size.
- Left-Side Color Image Data: 2D images (of only left-side perspective) for prediction visualization. Totally 12GB in size.
- Camera Calibration Matrices: for prediction visualization. Totally 16MB in size.
- Training labels: input labels from training. Totally 5MB in size.

The dataset consists of three categories: Cars, Pedestrians, and Cyclists. In this paper, the prediction precision, recall, and f1-score are averaged across the classes.

C. Testing Environment

Testing was performed on a system with the following specifications:

- CPU: i7 8th Gen (non-overclocked)
- GPU: NVIDIA GTX 1070 Ti (8GB GDDR5 VRAM) (non-overclocked)
- RAM: 16GB
- OS: Ubuntu 18.04 LTS (64-bit)

The tests are performed on a random sample consisting of 1414 iterations and then averaged out.

IV. Results & Analysis

This section shows the results of the tests performed and also analyzes the results obtained. The tests were performed on a system with specifications as mentioned in Section III-C. The tests include results from training a base model, which was pruned at 2, 4, 7, 10, 15, 20, 30, and 40 percent of nodes/weights. The tests are performed on a random sample of 1414 iterations and then averaged out. The confidence threshold has been set to 85%, i.e., after pruning, only predictions with more than 85% confidence will be considered.

The base model and the pruned models were again tested with PTQ-D enabled only for the Linear layers of the network.

A. Comparison of Average Inference Times

The average inference time is the amount of time a model takes to make a prediction (i.e., produce an output) from the time it receives the input. Network Optimization aims to minimize the inference time, i.e., the lower the inference time, the better the optimization. This section gives all inference times in milliseconds (ms).

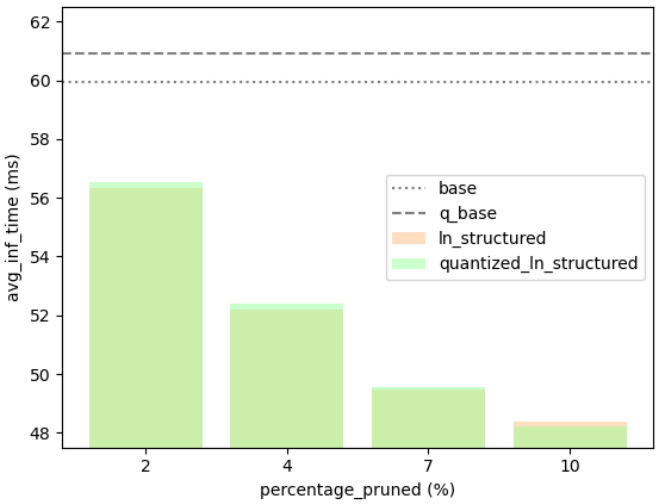

Fig. 3. Average inference times using Ln-Structured Pruning Technique

Fig. 3 shows the average inference times for the given percentage of pruned nodes using the L1-Unstructured Pruning technique. It includes the average inference times with quantization enabled and disabled, however, the difference is not significant. Here, only four pruning results are shown as the rest of the pruned models have low confidence. From this graph, we can observe that the average inference times are significantly lesser and gradually fall with increasing pruning percentage. At 10% pruning, the average inference time dropped by approximately 20%.

Fig. 4 shows the average inference times for the given percentage of pruned nodes using the Random-Structured Pruning technique. Here, due to low confidence five of the eight pruning results have been omitted, i.e., greater than 7%. From this graph, we can observe that the inference times are

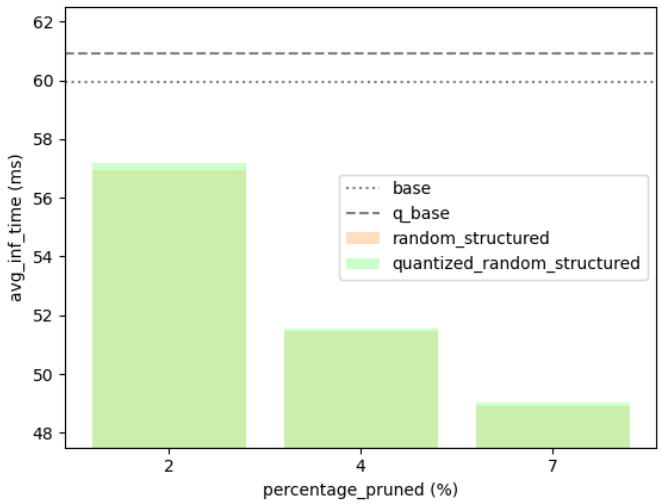

Fig. 4. Average inference times using Random-Structured Pruning Technique

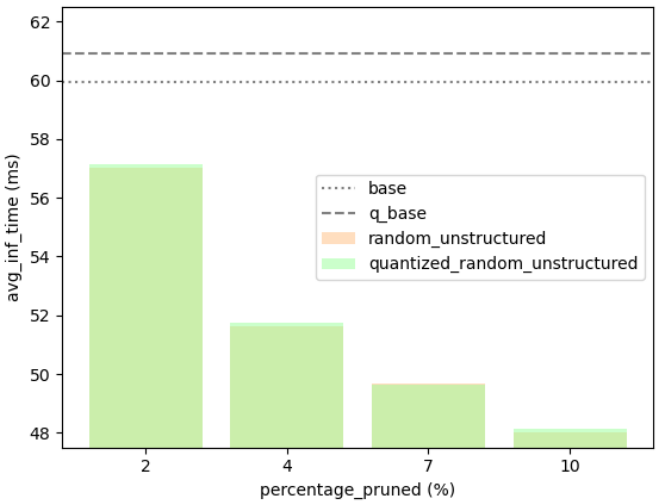

Fig. 6. Average inference times using Random-Unstructured Pruning Technique

gradually falling. At 7% pruning, the average inference time dropped by approximately 20% again.

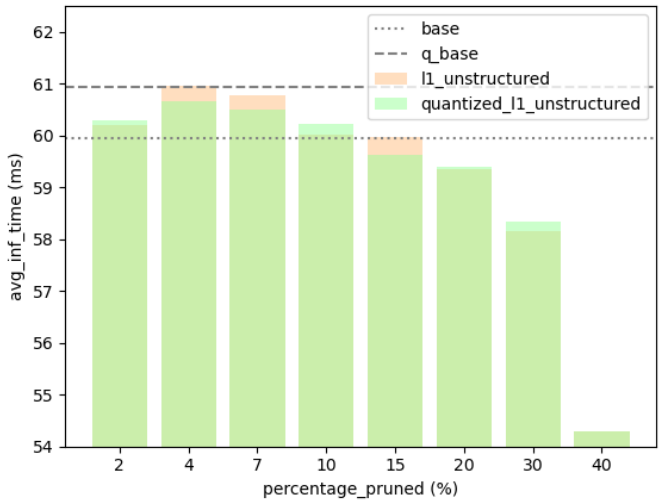

Fig. 5. Average inference times using L1-Unstructured Pruning Technique

Fig. 5 shows the average inference times for the given percentage of pruned nodes using the L1-Unstructured Pruning technique. This is the first pruning technique with all eight pruning levels. From this graph, we can observe that the inference times are significantly lesser for the 2, 10, 15, 20, 30, and 40 percent pruning levels with respect to the base model's inference time. At 40% pruning, the model steeply declines in the inference times. However, this may also be a sign of high loss.

Fig. 6 shows the average inference times for the given percentage of pruned nodes using the Random-Unstructured Pruning technique. It includes the average inference times with quantization enabled and disabled. Due to low confidence, four of the eight pruning results have been omitted, i.e., greater than 10%. From this graph, we can observe that the inference times are significantly lesser for the 2, 10, 15, 20, 30, and 40 percent pruning levels with respect to the base model's inference time. At 40% pruning, the model steeply declines in the inference times. However, this may also be a sign of high loss. At 40% pruning, the average inference time dropped by approximately 12%, which is the lowest compared to other techniques at much lower levels of pruning.

With each of the techniques, it is observable that the average inference time is considerably reduced with an increase in the percentage of nodes/weights pruned. Quantization further reduces the average inference time by a small amount in some cases.

B. Comparison by F1-Scores

The F1-score is defined as the harmonic mean of the precision and recall of a model. It is a metric that is often used to measure the performance of a model.

From Fig. 7, it can be observed that the F1-scores decrease as the percentage of nodes/weights pruned is increased. This also causes a fall in confidence in the prediction of the model, which inadvertently increases the loss in a model. L1 Unstructured has the most gradual decrease in f1-scores among the four techniques. In contrast, the F1-scores of the other three techniques – namely Ln Structured, Random Structured, and Random Unstructured – have a steep decline indicating a dramatic rise in loss.

Among the four techniques, L1-Unstructured performs the best with an F1-score of 0.785, at the highest pruning percentage of 30% while providing a speed-up of 3.1%. However, among structured pruning techniques, Ln-Structured pruning performs the best with an F1-score of 0.745, at the highest pruning percentage of 2% while providing a speed-up of 6.1%.

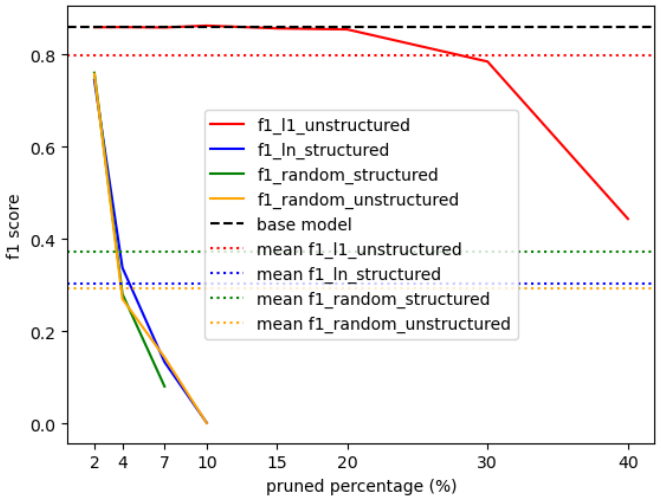

Fig. 7. F1-scores of base model and pruned models at various pruning levels

V. Conclusion

This paper describes two types of network optimizations – pruning and quantization – in detail. Pruning and quantization were then applied to improve the inference time of the model while simultaneously measuring its performance by extensively testing the model. The comparison of four pruning techniques in this paper yields a trade-off criteria for optimizing CNNs, i.e., inference speed vs performance for 3D object detection using point cloud data.

The tests revealed that the L1-Unstructured pruning technique performs the best with an F1-score of 0.785, at the highest pruning percentage of 30% while providing a speed-up of 3.1%. However, among structured pruning techniques, Ln-Structured pruning performs the best with an F1-score of 0.745, at the highest pruning percentage of 2% while providing a speed-up of 6.1%. These results indicate that a significant improvement in inference time can be obtained without drastically sacrificing the model's performance.

This work can be further extended in the following four ways:

1) Implementing other pruning techniques (both structured and unstructured) that can be applied to the model and tested to check for better inference times without loss in performance.
2) Implementing both unstructured and structured pruning techniques on the same model at varying pruning percentages.
3) Implementing other quantization techniques, such as Quantization-Aware Training (QAT).
4) Implementing other network optimization techniques, such as Factorization and Convolution Optimization, to optimize the model further while minimizing loss in performance.

References

[1] Informa Healthcare, , & Karwowski, W. (Eds.). (2006). International Encyclopedia of Ergonomics and Human Factors - 3 Volume Set (2nd ed.). CRC Press. https://doi.org/10.1201/9780849375477

[2] F. Bounini, D. Gingras, V. Lapointe and H. Pollart, "Autonomous Vehicle and Real Time Road Lanes Detection and Tracking," 2015 IEEE Vehicle Power and Propulsion Conference (VPPC), Montreal, QC, Canada, 2015, pp. 1-6, doi: 10.1109/VPPC.2015.7352903.

[3] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross, "Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds", arXiv:1803.06199v2 [cs.CV] 24 Sep 2018.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[5] Fernando Julio Cendra, Lan Ma, Jiajun Shen, and Xiaojuan Qi, "SL3D: Self-supervised-Self-labeled 3D Recognition", arXiv:2210.16810 [cs.CV] 30 Oct 2022.

[6] Yiyang Shen, Rongwei Yu, Peng Wu, Haoran Xie, Lina Gong, Jing Qin, and Mingqiang Wei, "ImLiDAR: Cross-Sensor Dynamic Message Propagation Network for 3D Object Detection", arXiv:2211.09518 [cs.CV] 17 Nov 2022.

[7] Alberto Tamajo, Bastian Plaß, and Thomas Klauer, "Shrinking unit: a Graph Convolution-Based Unit for CNN-like 3D Point Cloud Feature Extractors", arXiv:2209.12770 [cs.CV] 26 Sep 2022.

[8] Yuan Yao, Yuanhan Zhang, Zhenfei Yin, Jiebo Luo, Wanli Ouyang, and Xiaoshui Huang, "3D Point Cloud Pre-training with Knowledge Distillation from 2D Images", arXiv:2212.08974 [cs.CV] 17 Dec 2022.

[9] Redmon, Joseph and Farhadi, and Ali, "YOLOv3: An Incremental Improvement", arXiv, 2018