3D Lidar-Based Object Conditions in Adverse Weather Conditions

Rishi Murumkar

Received September 15, 2023
Accepted February 07, 2024
Electronic access February 29, 2024

In the world of autonomous driving, lidar based 3D object detection is a necessary task in order to assess the environment around the car and ensure passenger safety. In this field, convolutional neural networks, a facet of deep learning, dominate as the conventional approach. To ascertain the functionality of these networks for autonomous driving, it becomes necessary to evaluate neural networks when faced with challenges these cars will confront in the real world, such as object detection under adverse weather conditions. This paper confronts this challenge by evaluating PointPillar and SECOND, models that were pre-trained and tested on the KITTI dataset with clear weather conditions. These models are voxel based convolutional networks that convert 3D data into a 2D format to perform object detection. On the KITTI dataset, PointPillar and SECOND exhibited average precision values of 51.8625% and 70.0212% by the APR40 metric. The performance of both models was then tested on the Canadian Adverse Driving Dataset (CADC), which contains images with snowfall. The models reveal considerable accuracy degradation, dropping close to 0% in precision score on average. In an attempt to alleviate this issue, the networks are then retrained on CADC, returning to standard levels of precision on CADC. PointPillar and SECOND maintain an average precision of 48.4302% and 49.5063% on CADC, yet drop to 0% precision when reevaluated on KITTI.

Introduction

From a dream to a reality, the progress towards completely autonomous vehicles has hastened in the recent past[1]. In order to assure passenger safety, the systems employed in these vehicles must be able to execute multiple tasks to identify the optimal commands given to the vehicle. Of these functionalities, 3D object detection is of particular importance, as it plays a major role in determining system outputs such as steering angle, throttle usage, and braking pressure[2]. Additionally, 36,000 fatalities occurred on US roadways in 2019[3], of which 94% could potentially be eliminated by autonomous vehicles that rely on object detection[4]. Consequently, the performance of state-of-the-art object detection models must be evaluated in all environments in order to motivate robust object detection and improve the performance of autonomous vehicles universally to guarantee passenger safety.

In the field of computer vision, convolutional neural networks have made major breakthroughs, beginning with a top 5 performance of a CNN proposed by Hinton and his student Krizhevsky in the ImageNet Large Scale Visual Recognition Challenge of 2012. In the following year, Ross Girschik was able to utilize a variant of a traditional CNN to perform object detection with outstanding results, proving that deep learning could successfully complete this task. Since then, CNNs have remained the standard method in this field[5].

However, as we approach the possibility of fully functioning, driverless vehicles, it is imperative to ascertain the obstacles these automobiles will encounter in the real world[1]. As autonomous vehicles spread throughout heterogeneous regions, they will encounter practical variations in input data such as varied road signals, scale variability, and blurring that may affect the output commands of the entire system, posing a threat to passenger safety[6]. Of these variations, adverse weather conditions remain a common issue. Conditions such as snowfall may introduce factors such as noise, loss of contrast, and object obscuration that may affect a CNN's ability to accurately recognize objects. As a result, the performance of CNNs with regards to object detection must be evaluated amongst different conditions.

In this paper, the accuracy of PointPillar and SECOND, two convolutional models trained in a specific region, is surveyed when evaluated under weather conditions different from the ones in which they were trained. Following, the models are further trained in adverse weather conditions and reevaluated in both regions. Both PointPillar and SECOND originate from the OpenPCDet repo[7] and are pretrained on the 3D Object Detection Evaluation dataset, which is a subset of the KITTI Vision Benchmark Suite dataset[8]. Experiments evaluating the performance of PointPillar and SECOND were run on the Canadian Adverse Driving Conditions dataset (CADC)[9].

Related Work

Due to a breakthrough in computer processing power followed by novel ideas in the deep learning field, engineers have innovated upon a traditional neural network to develop convolutional neural networks (CNN)[10]. This product of machine learning

has a multitude of applications, including image segmentation, image classification, and object detection [11].

Structure of a CNN

Convolutional neural networks consist of a series of convolutional layers and fully connected layers found in a traditional neural network. When an image is fed to a convolutional layer, a kernel, a matrix of a predetermined size, is moved across the image in specified stride lengths in a process called convolution. As it moves, the kernel calculates the dot product using the pixel values of the image and the values of the kernel itself. The resulting value is input to a new, smaller matrix called a feature map. As kernels convolve, they emphasize and capture features such as edges and textures within all regions of the image. As a result, CNNs are particularly suited for 3D object detection as they can recognize patterns throughout an image and localize an object, regardless of its position in an image. Next, a nonlinear function such as a rectified linear unit (relu), is applied to each value in the feature map [12]. This decreases the computational power needed in the network by eliminating negative values and thus unnecessary neuron activations, which leads to faster training and forward propagation. The resulting feature map is put through a pooling layer, which shrinks the map further for computational purposes [11]. There are several pooling methods, including max pooling, which takes the greatest value out of each square section of the feature map and passes it onto the smaller map. Furthermore, average pooling takes the average of each square section and passes it on [13]. This convolutional process can be repeated as many times as specified. Figure 1 demonstrates the process of convolution.

For image classification, which is required in object detection, the feature map produced by the convolutional layers is flattened out into a vector and fed to the fully connected layers. Each element of the aforementioned vector becomes the value of a neuron. Each neuron is connected to every neuron in the next layer through a weight, which is a coefficient. Information is passed along to a neuron in the next layer through the following process: each neuron is multiplied by their respective weight, a bias is added to it, and this value is input into a nonlinear function for computational purposes. The resultant value is passed to the following neuron. This process continues until the last layer, which is composed of a set of neurons that represent classes. Each neuron in the previous layer is input into a function such as SoftMax that computes a probability for each class as a percentage. Figure two illustrates the process of forwarding a feature map to fully connected layers, resulting in a final activation for image classification purposes in the last layer of a convolutional neural network (CNN).

CNNs for Object Detection

Object detection is a computer vision task that includes identifying the location of an object in an image along with the category it belongs to. Before the deep learning revolution, this task was completed by training a classifier that could identify objects of two classes based on handcrafted features. However, this method was not favorable because the classifier's algorithm was designed to detect features extracted by humans, meaning its functionality was limited. Overtime, researchers have improved these archaic methods by innovating upon them and creating CNNs, which perform feature extraction and classification autonomously. Compared to other deep learning methods, CNNs outperform other methods in 3D object detection due to their parameter sharing and translation invariance, which contribute to robust detection across various object positions and orientations. Several deep learning methods have been proposed to tackle this problem, including faster region based convolutional neural networks (RCNNs), mask RCNNS, the single-shot method, and the single shot multibox detector [11].

Faster RCNN is capable of simultaneously detecting object candidate regions and performing object classification. First, it performs convolution on the entire image to obtain a feature map. It then raster scans the map, applying a multitude of detection windows centered around a spot on the image called an "anchor." Refer to figure 3 for a visual example of detection windows. The score of object likeness is calculated for each detection window, and if the object likeness is past a certain confidence threshold, the detection window becomes a bounding box. Next, RoI pooling resizes the features within the bounding box so they are suitable for a fully connected network. The coordinates of the bounding box are then input into a fully connected network. This network performs object detection by extracting a feature vector of a predetermined size and classifying the object [15].

Mask RCNN is an extension of faster RCNN. It involves the same two stage process, beginning with a region proposal network that detects areas of interest. After RoI pooling, mask RCNN performs object detection with one fully connected network while simultaneously performing image segmentation with a different fully convolutional network. This produces object "masks" which differentiate between a pixel that makes up an object and a pixel that makes up the foreground. However, both faster RCNN and mask RCNN are still not capable of real time object detection as they require time to extract features through multiple go rounds of convolution [16].

Due to this issue, the single-shot method has been gaining popularity, as it is capable of real time object detection. Otherwise known as YOLO (you only look once), this method solves object detection through regression by dividing the input image into a grid of squares. First, feature maps are generated through convolution. Next, each square predicts a bounding box, bounding box confidence score, and a class probability. The model

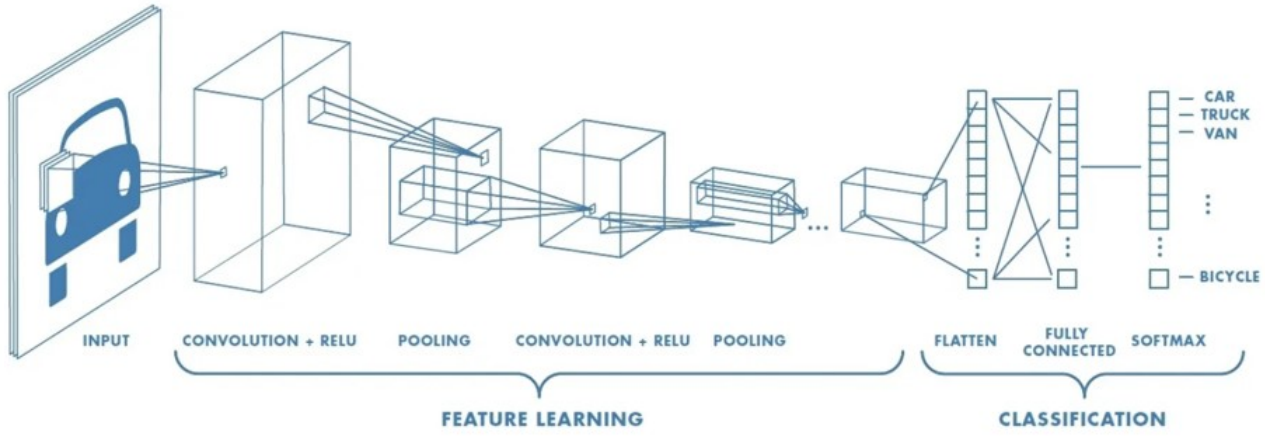


Fig. 1 Process of convolution [14]

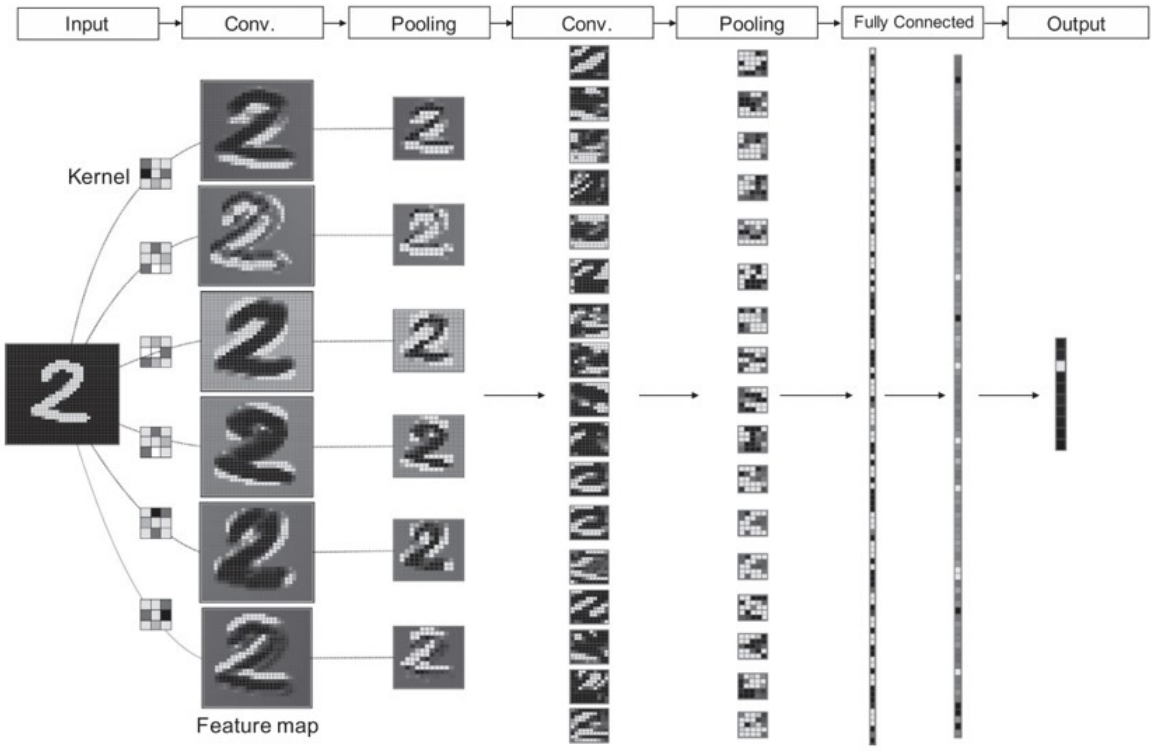
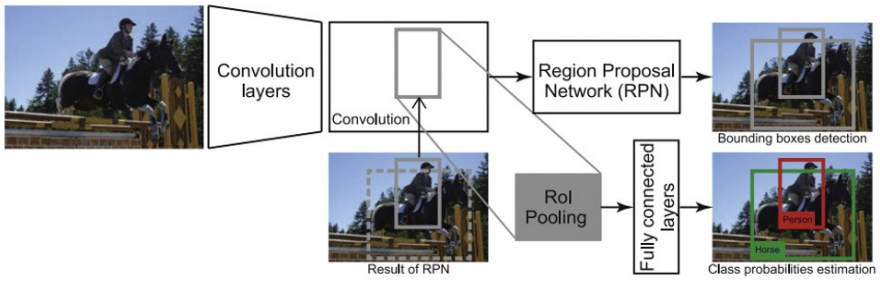


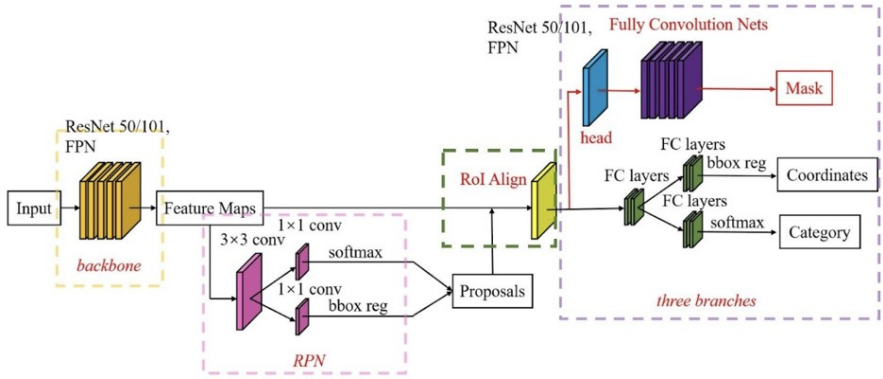
Fig. 2 Process of forwarding a feature map to fully connected layers [11]

takes the highest confidence score and outputs the corresponding bounding box with the object's class as a label, as figure 4 portrays. While YOLO is less computationally expensive because it processes the entire image at once, it is less accurate than faster RCNNs and not proficient at detecting smaller objects [11].

Similar to YOLO, single shot multibox detectors (SSD), perform object detection in real time by processing an image in one forward pass through a network. This method begins with a truncated base convolutional network, which extracts features through convolution. Notably, SSD skips the final classification layers, enabling simultaneous object localization and classification. Following, convolutional feature layers are added that progressively diminish in size, producing multi scale feature maps. This is done so objects at different scales can be detected.



Faster RCNN [11]



Mask RCNN [17]

Fig. 3 Visual example of detection windows

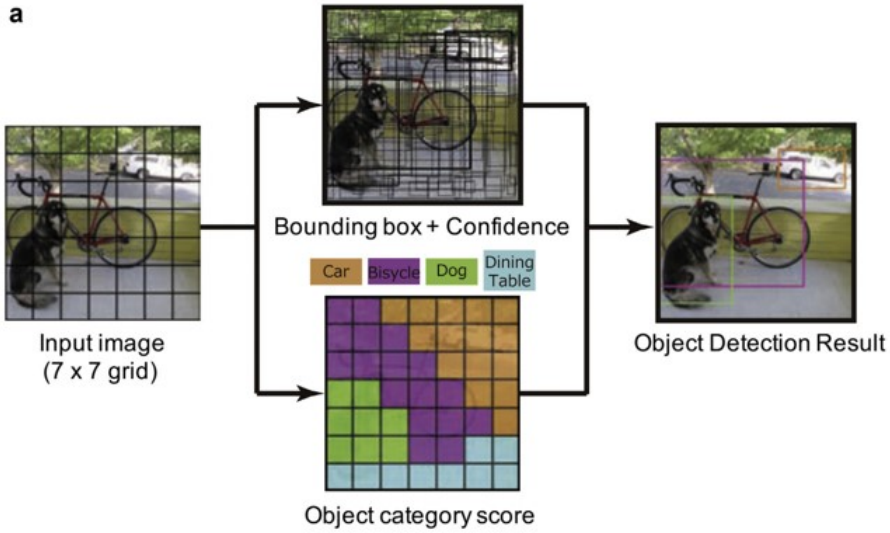


Fig. 4 YOLO [11]

Within each layer, each cell is associated with a set of default bounding boxes. As kernels convolve in a layer, each cell produces an offset that moves a default bounding box to better localize an object and a class score that predicts what object is in a bounding box. By repeating this process over several different feature maps with different sizes and resolutions, the

bounding box better predicts an object's location. Finally, the output consists of the bounding boxes with the highest class scores, representing the detected objects and their positions[18]. Below is an example of an SSD model architecture.

CNNs for 3D Object Detection

3D object detection requires data that is representative of a three-dimensional space. As follows, lidar data is commonly used in this field. Lidar is a remote sensing technology that utilizes laser lights that hit objects at certain points and return back to a sensor. Each reflection point provides valuable three dimensional spatial information based on the time it takes for the light to return. This includes reflectivity, color, position, and intensity, which are all features that contribute to object detection. The culmination of these reflection points produce lidar point clouds (refer to figure 5)[20].

Recent 3D object detection methods begin with voxelizing these point clouds, which divides the 3d space into a grid. All points in a specific location on the grid become a voxel, a volume element, as illustrated by figure 6. Next, all voxels in each vertical column are organized into a fixed length feature vector and projected onto a ground or image plane. This transforms a three-dimensional problem into a two-dimensional problem. From this step, standard convolutional neural networks can perform object detection[22].

However, Lidar comes with its limitations, most importantly, its cost. Lidar technology is expensive, preventing its widespread adoption within 3D object detection systems. Furthermore, Lidar is limited by both its range and resolution. The farther an object is to the sensors, the less accurate the lidar points are, making it challenging to accurately detect distant objects. Moreover, Lidar data is collected through laser beams that reflect off surfaces. If a surface is highly reflective or even transparent, such as mirrors and glasses, Lidar technology will not be able to pick up reflection points accurately, leading to poor data for object detection. Adding on to the issue, occluded objects can not be accurately represented by point clouds as the lasers can not reflect off the entire object, leading to misrepresentations of scenes for object detection[24].

How a CNN Trains

When an untrained CNN is given an input image, the predictions it outputs are likely going to be incorrect as its weighted parameters and biases will not be optimal. In order to tune these factors so the network is motivated to make the correct classifications, a loss function is required. A loss function computes the difference of the prediction of each class probability from the actual value. Common loss functions include mean squared error, cross entropy, and the hinge loss functions. For bounding boxes utilized in object detection, loss functions such as bounding box regression are implemented. Essentially, the goal of training is to reach a local minimum in the loss function, making the model's predictions as close to the ground truth as possible. This is done through back-propagation, which adjusts the weighted parameters of the model in accordance with the loss function, reaching the minimum through gradient descent.

The optimization parameters in a CNN include the kernel weights and the weights and biases found in fully connected layers. In the training phase, the gradient of each parameter with regards to the neural network is calculated by reversing the chain rule, and each weight is adjusted in steps to reach a local minimum. The following formulas are used to update each weight and bias in the model. The learning rate η is used to control the step size. This process encapsulates the function of backpropagation[5].

$$W_i = W_i - \eta \frac{\partial E(W,b)}{\partial W_i}$$
$$b_i = b_i - \eta \frac{\partial E(W,b)}{\partial b_i}$$

A higher learning rate ensures faster training but risks the possibility of overshooting the minimum of the loss function during gradient descent. A lower learning rate reduces the chance of overshooting but extends the training time. It also risks over-fitting, which occurs when a model memorizes the training data to an extent at which it is unable to perform well on new, unseen data.

Furthermore, proper weight initialization is crucial in the training process. There are several methods to initialize these weights, including zero initialization and random initialization[25]. Initializing weights at smaller values can lead to vanishing gradients, which occurs when the gradients produced by back-propagation are too small to reach the minimum of the loss function. On the other hand, initializing weights with large values can lead to exploding gradients, which occur when back-propagation produces large gradients that overshoot the minimum of the loss function.

As a result, the choices of learning rate and weight initialization technique are integral to training a neural network.

Object Detection in Adverse Weather Conditions

In actual practice, many possible factors resulting from weather can affect a convolutional neural network's ability to predict accurate bounding boxes and perform 3d object detection. Factors such as different lighting, partial occlusion of objects, and reduced image quality can bring about unforeseen variations that extend beyond the scope of the training data and resulting in a degradation in performance. Standard 3D object detection models function well on inputs that are similar to the data they have been trained on. This is because they are large functions

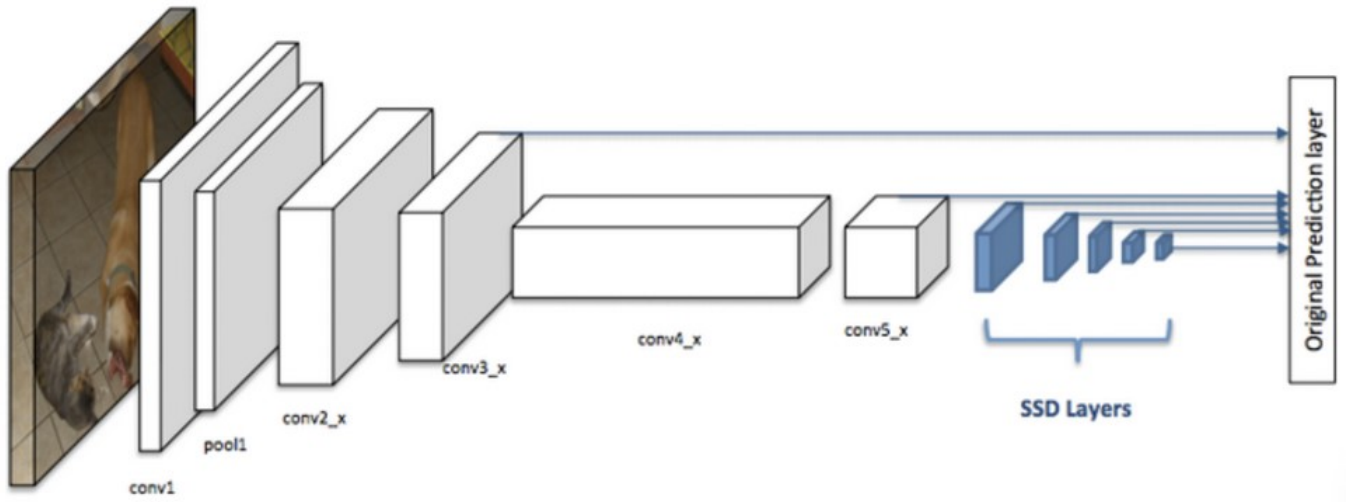


Fig. 5 Single shot multibox detectors (SSD)[19]

Fig. 6 LIDAR Data[21]

that have been trained to produce an output based on a large input of pixel values. When adverse weather conditions are present, the original input can be drastically altered due to the aforementioned factors, affecting the pixel values inputted to the network. This can cause the network to produce a drastically different output.

To combat this issue, several methods have been proposed to improve the generalization ability of autonomous driving systems in adverse weather conditions. Generalization is a neural network's ability to extend its functionality from a trained data set to any set of unforeseen test cases. The aforementioned methods include training with robust data sets containing images under different weather and lighting conditions or removing the visual factors that lead to misclassifications. Large and varied train sets have a higher chance of containing closely related examples for any specific test case. The problem with these methods is that it is expensive to collect and label vast amounts of data. Furthermore, they focus on training the networks for varied scenarios, which means they still might fail when encountering new corner cases. To combat these issues, researchers have proposed the REIN training method, which has no need for extra labeled data, and the ST algorithm, which utilizes unlabeled data[26].

The REIN method is a loss function that regulates first order gradients through double backpropagation. Neural networks are in essence large, nonlinear functions composed of a multitude

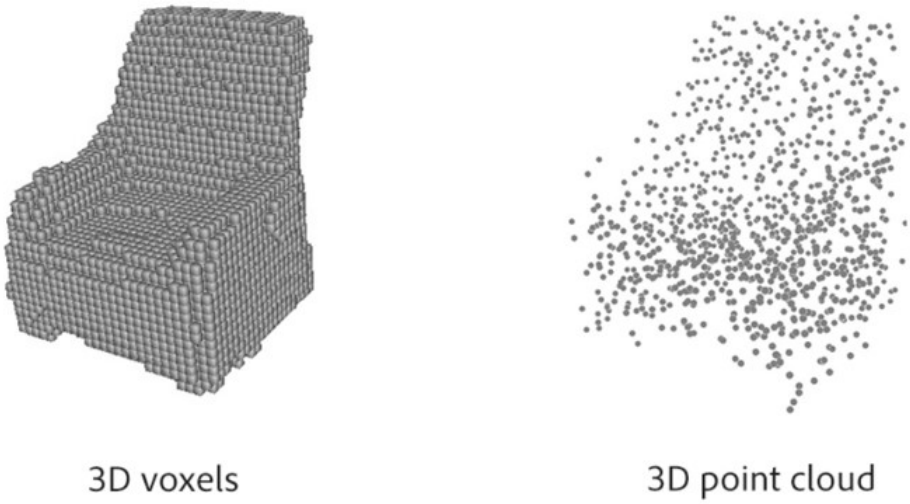


Fig. 7 3D object detection methods [23]

of neurons, weights, and biases. As a result, by performing the first order taylor series expansion on this function, it can be proven that the first order gradient of the neural network is an amplification coefficient to the variation in the input image. Since variations result in wrong outputs, it is true that regulating the first order gradient of the function will lessen the strength of the variation on the output. Thus, neural networks with high generalization abilities should have as small first-order gradients as possible. Figure 7 illustrates this concept.

The REIN method accomplishes this by adding a gradient loss penalty to the loss function, essentially training the neural network to possess smaller gradients. Normal gradient descent adjusts the parameters of the neural network using only first order gradients to find the minimum of the loss function; however, the first order gradients must be minimized as well. Therefore, a second backpropagation is needed to calculate the minimum value of the first order gradients.

On the other hand, the ST algorithm is a semi-supervised training method that utilizes partially labeled data sets, taking advantage of large amounts of unlabeled data. The process begins by training the neural network with clean, labeled data. In order to improve its generalization ability, the neural network is then exposed to large amounts of unlabeled data with variations present in the input images. The network creates pseudo labels for this data through forward propagation, and then retrains itself using the input images and corresponding pseudo labels. However, this can cause problems because if the wrong label is used in training, the model will accumulate wrong weights and biases by mislabeling similar images. To solve this issue, only pseudo labels past a certain confidence threshold are used in training. This ensures that the labels have a higher chance of being correct, creating a nuanced network that is motivated to make accurate predictions on input images with variations. However, this method still does not guarantee high quality pseudo labels for training. As a result, the ST algorithm incorporates several techniques to reduce the number of substandard pseudo labels, including curriculum learning, imbalance sampling, and progressive confidence-based thresholding.

In curriculum learning, the model is trained using images with smaller variations because the labels the NN creates for these images with high confidence are more likely to be accurate. Following, we train the model using these pseudo labels and then make the NN label images with higher variations, expanding its generalization ability in progressive steps. The entire process is iteratively conducted until convergence, yielding a more proficient and adaptable neural network.

In imbalanced sampling, the neural network is trained in batches of data that is randomly taken from both the real labels and the pseudo labels. In each batch, the images with real labels are oversampled while the images with pseudo labels are under sampled during the training phase. This ensures that the backpropagation produces close to the correct gradients as the wrong pseudo labels will only cause minor errors that won't affect the gradient descent drastically.

In the ST algorithm, we have a confidence threshold that selects pseudo labels that the NN is confident in to give the labels the best chance at accuracy (if the model is confident in its prediction, the image is more likely to be correctly labeled). However, a high confidence threshold means less pseudo labels are used in training, which means the neural network will never be fully trained with images with high levels of variation. To solve this, we can implement progressive confidence-based thresholding. This process starts off with a high confidence threshold to ensure that the model is training on quality pseudo

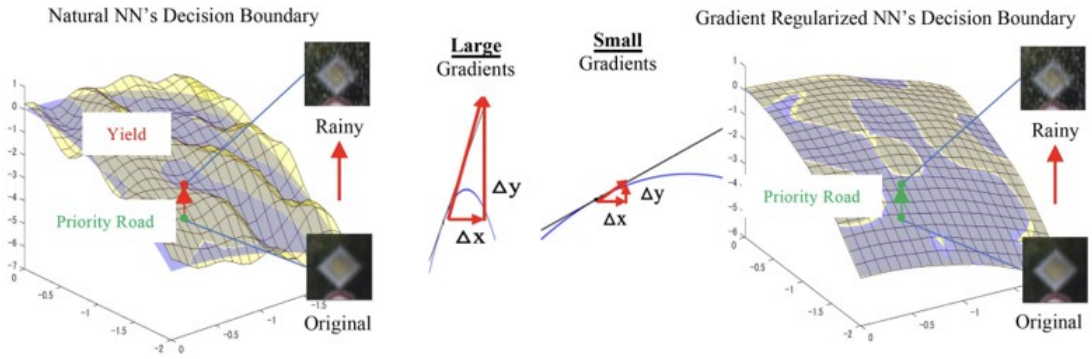


Fig. 8 The REIN method[4,26]

labels. As time goes on and the model is retrained, we can lower the confidence threshold because the NN is more likely to label the unlabeled data correctly with better training.

Methodology

In this section, we first introduce both the KITTI and CADC datasets and proceed to elaborate on the models used for 3d object detection on the data. Furthermore, we describe the average precision metric used to evaluate the results along with detailing the structure of the experiment. The KITTI data is first tested using the models to set baseline results. Then the CADC data is tested to assess the results of weather variations in data inputs on 3d object detection performance. Finally, the models are retrained on CADC to improve its generalization ability and retested on both KITTI and CADC to evaluate the results of retraining.

Dataset Description

Both the KITTI Dataset and the Canadian adverse driving conditions (CADC) dataset were utilized for this experiment. The 3D Object Detection Evaluation dataset, a subset of the KITTI Vision Benchmark Suite was used to train and test the CNN models for 3D object detection. This dataset comprises 7481 training images and 7518 test images, including corresponding point clouds, culminating in 80,256 labeled objects. The image data was collected in Karlsruhe, Germany in clear weather conditions on highways and in rural areas. When downloaded, the KITTI data is separated into two folders, training and testing, that contain corresponding label, image, calibration, and lidar files. The images were collected both in rural areas and highways with upwards of 15 cars and 30 pedestrians per image. This promotes a diversity of objects and lends itself to robust evaluation of 3d object detection models on the data[8].

The CADC dataset is a collection of 75 scenes of 50-100 frames each, resulting in a total of 56,000 images with corresponding point clouds collected by lidar. The data was collected in the Waterloo Regional Municipality, Ontario, Canada during the winter while snowfall was present. The data is separated into 3 folders organized by dates of data collection, and the label, image, calibration, and lidar files can be found in subfolders through each file path. Both the KITTI and CADC data were imported and run in a google colab environment. Moreover, the data includes images from several different routes, all with varying levels of traffic and a variety of vehicles, ensuring a diversity of data and robust experimentation on this data[9].

Model Architecture

In this study, two models pre-trained on the KITTI dataset from the OpenPCDet repo found on github are utilized[7]. Both PointPillar and SECOND are optimal for 3D object detection because they convert 3D inputs into 2D pseudo images, upon which basic and computationally inexpensive object detection can be performed. OpenPCDet is an open source project containing several pretrained models for lidar based 3d object detection. It supports a large number of datasets such as the Waymo and Argoverse 2 Dataset. In order to test the CADC dataset, a branch of the OpenPCDet repo was used, cadc_support[27]. This branch contains a data loader for the CADC dataset along with corresponding scripts for testing with all the KITTI pretrained models. The CADC data was downloaded through a cadc_devkit[28].

a. PointPillar: PointPillar is a state of the art 3D object detection model proposed by Alex H. Lang in 2019. [29] It begins by organizing the point clouds into pillars in order to convert the unstructured data into a 3D grid to format the data for a standard 2D CNN. After data augmentation, each point then becomes a 9D vector. Furthemore, the pillars are subsampled for computational efficiency based on their density. Following this, a linear, batch normalization, and relu layer are applied consecutively to each 9D vector from points in selected pillars to generate a tensor of size (C, P, N), where C is the number of channels (features), P is the number of pillars, and N is the

number of points per pillar. A max operation is performed over this tensor to output a tensor of size (C, P) for each pillar and these encoded features are scattered back to create a pseudo image of size (C, H, W) where H is the height and W is the width of the image. This 2D representation is known as bird's eye view. In this pseudo image, each pillar represents a pixel. Finally, a 2D convolutional model is applied over the pseudo image to predict bounding boxes and classification. PointPillar is suited for 3D object detection due to its ability to process point clouds and consolidate sparse data into dense, 2D data, allowing it to be both accurate and computationally efficient.

b. SECOND: SECOND is a voxel based convolutional network. [30] It begins by voxelizing the input point cloud to organize the sparse data into a structure for neural network applicability. A fully connected layer consisting of a linear layer, batch normalization layer, and a relu layer is applied to each voxel to produce a 3-dimensional feature map and increase computational efficiency. Unlike PointPillar, sparse convolutional layers are applied to the 3d map to reduce its z dimensionality. Sparse convolutional layers take an input tensor of all nonzero values in a voxel, meaning only a fraction of the data in a voxel are input into these layers for computational efficiency. Then, a max pooling layer is applied over the output tensors to reduce its size and vertical dimension. Once the z-dimension has been reduced to one or two, the sparse data are converted into dense feature maps. Then, the data are simply reshaped into a 2d pseudo image. Finally, a 2D convolutional model is applied over the pseudo image to predict bounding boxes and classification. SECOND is well-suited for 3D object detection due to its computational efficiency. It is able to handle sparse, 3D data, converting a 3D image into a 2D pseudo image, allowing conventional CNNs to effectively detect objects in the transformed representation.

Metrics

In this experiment, the effectiveness of the models will be assessed based on their ability to accurately predict bounding boxes for the pedestrian class. This particular class serves as a common factor between both the CADC and KITTI datasets. The metric used to evaluate the model's performance is APR40, a 40 point interpolated average precision metric that calculates the average precision of a model's bounding boxes using a precision-recall curve[31]. When evaluated in the OpenPCDet repo, the results are defined in the APR40 metric. This metric was chosen because it combines both precision and recall over 40 ranges. Precision and recall are fundamental metrics in 3D object detection tasks, as they show a model's capacity to correctly identify positive instances (precision) and its ability to capture all positive instances (recall). APR40 is optimal because it shows the tradeoff between precision and recall. We now define the metric formally.

Since the models implement bounding boxes, it is necessary to utilize a metric that captures the quality of these boxes. A bounding box is considered to match the ground truth if it has an IoU (intersection over union) of 0.5 and above for the pedestrian class. IoU is a measure of the overlap from a predicted bounding box to a ground truth bounding box. The closer the IoU value is to 1, the better the predicted bounding box is[32].

Precision is a measure of the ratio of true positives to the sum of true positives and false positives. Recall is the measure of the ratio of true positives to the sum of true positives and false negatives.

$$Precision = \frac{(TruePositive)}{(TruePositive + FalsePositive)}$$

$$Recall = \frac{(TruePositive)}{(TruePositive + FalseNegative)}$$

Within the confines of this experiment, a precision recall curve is a graphical representation that illustrates the tradeoff between both metrics as the number of allowed predicted bounding boxes increases. When a limited number of bounding boxes are permitted, a model's precision tends to be higher due to a decrease in the false positives predicted. However, the recall tends to be lower as there is a lower chance that the model correctly detected all the bounding boxes that are present in the ground truth. The same logic applies in reverse as the allowance of bounding boxes expands[22,31]. APR40 uses 40 different bounding box thresholds, generating 40 different values of precision and recall for a graph.

The metric ranges through each recall value and sums the maximum precision value corresponding to the recall value greater than the current recall value. Once it obtains a final sum, it finds the average precision value. This process is done using the following equation where N is the number of bounding box thresholds, p is precision, r is the current recall value, and $\hat{r}$ is the recall value greater than the current value.

$$AP_{R_N} = \frac{1}{N} \sum_{r \in R_N} \max_{(\hat{r} \geq r)} p(\hat{r})$$

The results are calculated and categorized into 3 different difficulty levels of object detection: easy, moderate, and hard. In terms of autonomous driving, a high rate of precision implies that of the pedestrians detected, few are falsely detected. This is especially crucial in certain driving situations in which the likelihood of encountering a pedestrian is low. For instance, highways typically have no pedestrians. However, if a vehicle detects one, it could lead to harsh braking, which can be dangerous in a high speed situation.

On the other hand, recall might be prioritized in dense urban environments in which pedestrians are plentiful, as a high recall value would reduce the chance of not detecting real pedestrians and causing an accident.

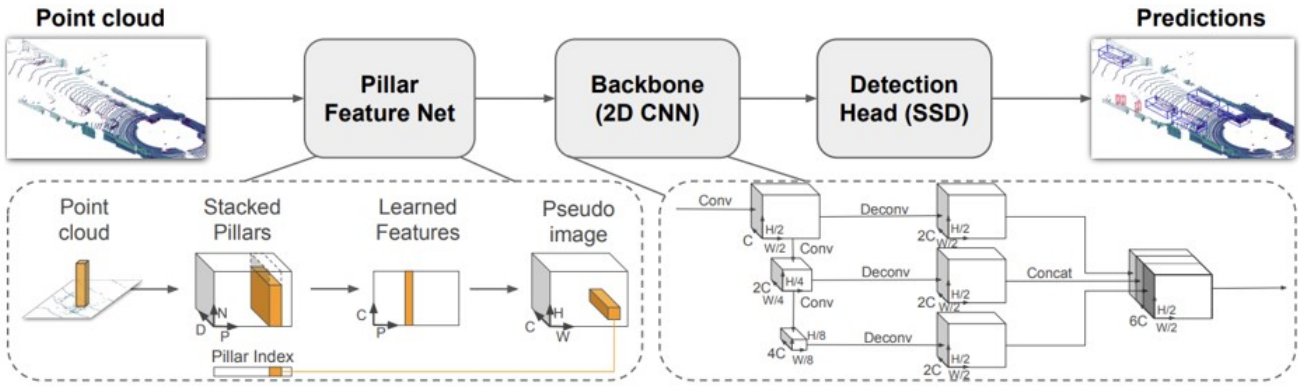


Fig. 9 PointPillar[29]

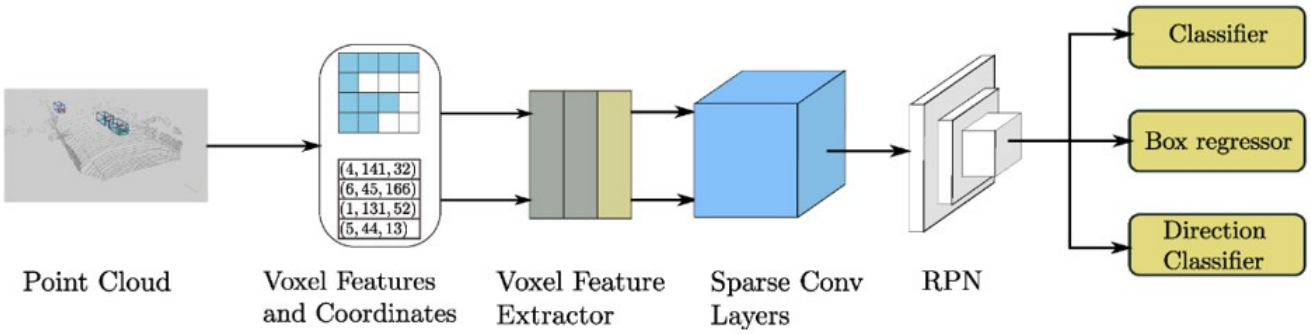


Fig. 10 SECOND[30]

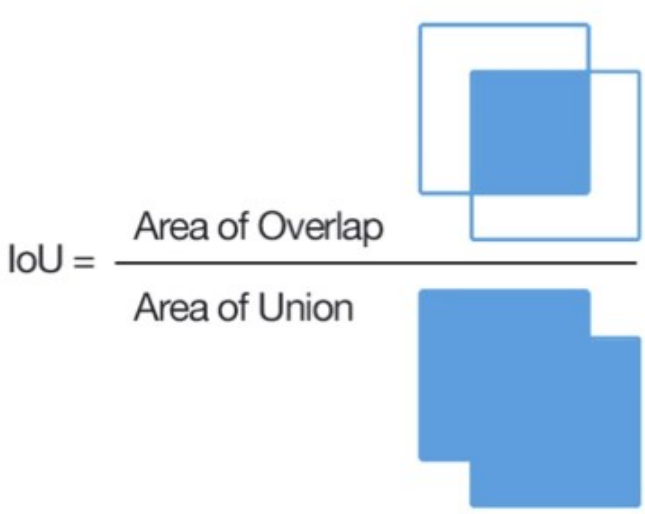


Fig. 11 [32]

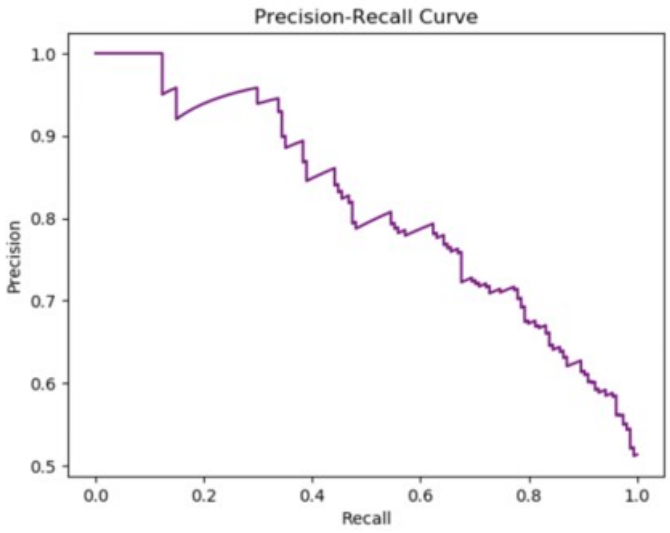


Fig. 12 Precision-Recall Curve[33]

Training and Testing

The pretrained models were evaluated on the KITTI dataset as a baseline for the APR40 scores in the pedestrian class for 3D object detection and object detection in bird's eye view. Following this, the same models were tested on the CADC

dataset using APR40 to determine the drop-off in precision for the same categories. After observing the results, the models were further trained for 50 epochs with a batch size of 8 on the train split from the CADC dataset and re-evaluated on KITTI and CADC to determine the effects of training on the average precision and generalization ability of the models.

In order to maintain consistency between the networks on KITTI and CADC, the default learning rate used to pretrain the models on KITTI was kept the same. Therefore, both PointPillar and SECOND maintained a learning rate of 0.003. This means that the model's weights will be updated by a step size of 0.003 times the gradient of the loss function during training.

During pretraining, PointPillar's weights were initialized using a zero mean Gaussian distribution. Each weight was assigned a randomly generated number from the aforementioned Gaussian. This initialization strategy aims to maintain the stability of weights throughout pretraining, preventing significant adjustments. The intention is to produce weights where no one weight disproportionately impacts the output of the network due to exponential increases or decreases in signal magnitude[34]. SECOND implemented transfer learning by leveraging the pretrained PointPillar weights.

Results and Analysis

Results for Pretrained KITTI model

The APR40 precision (in percentage) of PointPillar for both 3D object detection and object detection in bird's eye view (bev) are given below. These are the results for the pedestrian class of both datasets.

Table 1 KITTI

		Easy	Moderate	Hard	Average
PointPillar	BEV	61.5971	56.0143	52.0457	56.5523
	3D	57.3015	51.4145	46.8715	51.8625
SECOND	BEV	73.8776	70.4969	66.6493	70.3413
	3D	73.7943	70.2258	66.0435	70.0212

Table 2 CADC

		Easy	Moderate	Hard	Average
PointPillar	BEV	0.0000	0.0000	0.0000	0.0000
	3D	0.0000	0.0000	0.0000	0.0000
SECOND	BEV	0.0042	0.0107	0.0116	0.0088
	3D	0.0000	0.0000	0.0000	0.0000

Reasons for the Model's Failure

As per the results, it can be shown that CNNs trained with clean data for the task of lidar based 3d object detection have poor generalization ability. On the KITTI dataset, pointpillar and second performed fairly well, as over half of the pedestrians detected matched the ground truth for all difficulty levels except for the hard category in bird's eye view. However, when tested on CADC, the models were unable to even remotely accurately predict a bounding box with an IoU above or equal to 0.5 for the pedestrian class, which reveals a massive failure in modern CNNs.

The CADC dataset contains numerous variations that could signal the models' overall failures. To begin with, the most obvious variation is the presence of snowfall. Heavy snowfall can provide a multitude of challenges to CNNs. It can drastically alter the visual appearance of the image, affecting the color, texture, and appearance of objects. Furthermore, the accumulation of snow can partially block or occlude objects, making object localization difficult. Adding to the issue, snow has a high albedo, meaning it reflects light easily, which can lead to glare in cameras and a reduction in image quality. Moreover, snow can blend the foreground into the background, making it hard for a CNN to correctly detect a bounding box that separates an object from the environment. Finally, snow leaves water streaks on the camera, further reducing image quality. Since the train data set did not contain images with snowfall, the model was unable to recognize these foreign patterns and thus unable to make accurate predictions.

Another challenge presented by the CADC dataset includes pedestrian appearance. The pedestrians present in heavy Canadian snowfall versus the pedestrians present on a sunny day in Germany are likely to be clothed differently. The Canadian pedestrians are more likely to be wearing heavy clothing that differentiates them in size and form from the German pedestrians, which is another pattern that pointpillar and second weren't trained to recognize.

Efforts to Improve Performance on CADC

In an effort to improve the performance of PointPillar and SECOND on CADC, the models were trained further on the robust CADC dataset and evaluated again on the test set of CADC. The results are given below.

		Easy	Moderate	Hard	Average
PointPillar	BEV	62.7322	61.3602	53.6483	59.2469
	3D	52.2476	49.8547	43.1883	48.4302
SECOND	BEV	62.7399	61.0078	54.2155	59.3211
	3D	53.6463	50.7538	44.1188	49.5063

The robust dataset proved to have a dramatic effect on the

ability of the models to detect pedestrians in the presence of adverse weather conditions. The average precision for each difficulty level jumped up significantly. The overall improvement is due to the fact that the train dataset motivated the model to better handle the variations present during snowfall such as heavy clothing, glare, watermarks, etc.. Extensive and diverse training datasets are more likely to contain closely related instances relevant to a particular test scenario. This means that the model has seen a large variety of patterns during training and has understood how to correctly predict and label bounding boxes for the objects present in the image in the testing phase[5].

KITTI Performance After Retraining

To gauge if the models retained their accuracy on the KITTI dataset, the models were reevaluated on the test split of the KITTI dataset. The results are shown below.

		Easy	Moderate	Hard	Average
PointPillar	BEV	0.0000	0.0000	0.0000	0.0000
	3D	0.0000	0.0000	0.0000	0.0000
SECOND	BEV	0.0028	0.0025	0.0024	0.0026
	3D	0.0001	0.0001	0.0001	0.0001

Conclusion & Limitations

The aim of this research was to test the performance of state of the art CNNs in lidar based 3d object detection in adverse weather conditions which they weren't trained for. The results obtained reveal a lacking ability in object detection models for self-driving vehicles, specifically with regards to detecting pedestrians in harsh weather. When the pretrained models were evaluated in the region it was trained on, it produced significant results. However, when evaluated under adverse weather conditions, the models utterly crashed. The aim of autonomous driving is to produce vehicles that are able to operate without a driver in any global region, particularly under one object detection model. Developing a new model for every region would cost a significant and unreasonable amount of human resources.

In an effort to improve the generalization ability of the models, both networks were retrained in adverse weather conditions for object detection. While they produced substantially better results in snowy conditions, there was a considerable dropoff in accuracy in clear conditions. This occurred because the adverse conditions dataset motivated the models to correctly detect pedestrians only in the presence of snow. As a result, it can be shown that CNNs trained primarily in adverse conditions will fail in clear conditions. In terms of autonomous driving, this training method would result in a poor 3D object detection model as it would influence a model to perform well in only a particular type of region.

However, there are several training methods that may have improved the performance of the models in both regions. For instance, the pretrained KITTI models could have been trained iteratively on CADC, with periodic evaluations conducted on both CADC and KITTI datasets at specified training intervals. Theoretically, the accuracy on CADC would rise while the accuracy on KITTI fell as the model retrains with weights that are progressively more biased towards CADC. Once the model reaches a point at which the performance on both datasets is substantial, the training process would stop. This would produce a model with weights that could handle patterns in both clear and adverse weather conditions. However, it is unclear whether or not such a point exists. Furthermore, another potential training method would be to merge the train sets of the KITTI and CADC datasets together. By this method, the models would be trained on both datasets, leading to unbiased weights. However, this process would increase the training time significantly and require a notable increase in computational power needed. Furthermore, it would eliminate the advantage of having a pretrained model.

Limitations of this experiment include the resource intensive process of data loading and training. On average, each time the CADC dataset was loaded into google colab, the process took about an hour and forty minutes, followed by an hour of generating the data infos that were essential to preprocess the data. Furthermore, the training processes demanded approximately 17 hours on average, adding another layer of time intensive commitment needed to conduct the research. The fundamental problem of these models is that they are difficult to do research on individually. Having a continuous environment going that has the data loaders running and in the memory is computationally expensive and currently, there is no method of avoiding this issue. Moreover, due to a time constraint and lack of models supported by the CADC dataset, there was a restriction on the number of models tested on CADC and number of training methods implemented. A variety of network architectures would provide a more comprehensive view of the generalization ability of modern CNNs through comparative analysis between an array of models.

In a broader scope, the results produced by this research indicate that modern day CNNs aren't ready to be implemented in autonomous driving systems. In order to assure passenger safety, neural networks for 3D object detection must attain near perfect results during testing for both precision and recall. Furthermore, they also must possess a capable generalization ability, allowing them to perform successfully on an infinite number of never before seen corner cases.

If CNNs are ultimately adopted into real world driving systems, they have the potential to form the backbone of advanced driving technology such as cruise control, automatic lane changing, automatic emergency braking, and adaptive cruise control.

This is because CNNs are inherently advanced at recognizing the environment around them. They are proficient in detecting features and producing correct 3D object detection outputs. As a result, they can be used to ensure ultimate driver safety, creating a scenario in which a driver no longer has to pay attention to the road.

CNNs can promise to not only transform the driving experience but also save countless lives. By minimizing human error, these advanced neural networks may contribute to a future where road accidents are mitigated, and the driving environment becomes safer. The potential ripple effects of this technology extend far beyond societal safety, promising advancements in global transportation efficiency. However, this is only the case if CNNs can operate in all scenarios, including situations involving adverse weather conditions. To combat the inherent poor generalization ability of CNNs, techniques like the REIN loss function, which controls first-order gradients via double backpropagation, and the ST semi-supervised training method, which utilizes large amounts of unlabelled data, can be employed. These methods have the potential to minimize the effect of variations in prediction outputs and to better train neural networks to handle variations in input data.

References

1 D. Parekh, *Electronics*, **11**, 2162,.

2 F. Munir, S. Azam, M. Hussain, A. Sheri and M. Jeon, Proceedings of the 2018 International Conference on Sensors, Signal and Image Processing.

3 Nhtsagov, https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-05/CA_FY2021_AR.pdf., Available:.

4 V. Kukkala, *IEEE Consumer Electronics*.

5 W. Zhiqiang and L. Jun, 2017 36th Chinese Control Conference (CCC, Dalian, China, pp. 11104–11109,.

6 J. Horwitz and H. Timmons, *Quartz*.

7 O. D. Team, *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*.

8 A. Geiger, P. Lenz, C. Stiller and R. Urtasun, *Cvlibs.net*.

9 M. Pitropov, *Canadian Adverse Driving Conditions Dataset*, http://arxiv.org/abs/2001.10117., Arxiv.org. [Online]. Available:.

10 R. Bhatia, *Analytics India Magazine*.

11 H. Fujiyoshi, T. Hirakawa and T. Yamashita, *IATSS Research*, **43**, 244–252,.

12 B. Krishnamurthy, *Built In*.

13 A. Zafar, *Applied Sciences*, **12**, year.

14 Prabhu, *Medium*.

15 *Object Detection and Classification using R-CNNs*, https://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/., Online]. Available:.

16 K. He, G. Gkioxari, P. Dollár and R. Girshick, *CoRR*, **1703**, year.

17 *Mask R-CNN*, https://zhuanlan.zhihu.com/p/62492064?ref=blog.roboflow.com., . [Online]. Available:.

18 W. Liu, *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(*, Springer, Cham, vol. 9905.

19 *ArcGIS API for Python. [Online*.

20 Abdishakur, *Spatial Data Science*.

21 V. L. Authors, *Velodyne Lidar*.

22 W. Zimmer, E. Ercelik, X. Zhou, X. Ortiz and A. Knoll, *A Survey of Robust 3D Object Detection Methods in Point Clouds'*, arXiv [cs. CV].

23 *A papier-mâché approach to learning 3D surface generation*, https://research.adobe.com/news/a-papier-mache-approach-to-learning-3d-surface-generation/., Online]. Available:.

24 S. Aerial, *Scout Aerial Australia*, p. 16– –2021.

25 S. Yadav, *Towards Data Science*.

26 F. Yu, Z. Qin, C. Liu, D. Wang and X. Chen, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **40**, 1258–1271,.

27 M. Pitropov, *OpenPCDet: OpenPCDet Toolbox for LiDAR-based 3D Object Detection*.

28 M., *Pitropov, cadc_devkit: A devkit for the Canadian Adverse Driving Conditions (CADC) dataset*.

29 A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom, *PointPillars: Fast Encoders for Object Detection from Point Clouds'*, arXiv [cs. LG]. 2019.

30 Y. Yan, Y. Mao and B. Li, *Sensors*, **18**, year.

31 J. Vianney, S. Aich and B. Liu, *RefinedMPL: Refined Monocular PseudoLiDAR for 3D Object Detection in Autonomous Driving'*, arXiv [cs. CV].

32 *Intersection over Union (IoU*, https://hasty.ai/docs/mp-wiki/metrics/iou-intersection-over-union., Available:.

33 Zach, *How to create a precision-recall curve in Python*, https://www.statology.org/precision-recall-curve-python/., Statology, 09-Sep-2021. [Online]. Available:.

34 K. He, X. Zhang, S. Ren and J. Sun, *CoRR*, **1502**, year.