Inverting 3D Deep Learning Architectures

Abhinav Agarwalla
aa4@andrew.cmu.edu

Paritosh Mittal
paritosm@andrew.cmu.edu

Shubham Gupta
shubham2@andrew.cmu.edu

Abstract

Several applications of computer vision e.g. *autonomous driving, warehouse management* etc. *are nearing deployment in real-world. However these approaches critically depend on 'black-box' 3D neural networks. Interpreting and understanding these networks is an important and challenging problem for vision community. This project attempts to interpret the learning of 3D neural networks through the lens of model inversion. Specifically, we investigate what a 3D model learns by trying to re-create an optimal input based on a perceived output. Recent methods present solutions for inverting classification and detection networks, but only for 2D inputs. This project extend these approaches to 3D which is significantly more complex and ill-posed. We showcase results on inversion of 3D deep learning architectures for classification and detection and further analyse our findings.*

1. Introduction

The ability of an AI agent to operate and interact in real world is critically dependent on it's understanding of 3D. Hence there is considerable effort by vision community to develop neural models which can solve primary 3D vision tasks of detection [6, 29], classification [15, 21, 30], reconstruction [4, 10, 11, 25] etc. The success of these models in existing benchmarks [1, 5, 20] together with their real world applications in autonomous driving, virtual/ augmented reality, robotics and online marketplaces is testament to their prowess. However, the core component of these approaches are deep neural black-box networks which directly learn the optimal parameters to solve a particular problem given data.

In real world, it is often possible that (1) our data is not sufficient or entirely representative of all possible scenarios; (2) our loss functions are sub-optimal based on the given task; (3) our evaluations do not quantify model generalization effectively etc. Many such problems can significantly impact the performance and applicability of deep neural models particularly for in the wild examples. Model interpretability, a way to understand what the model learns, is hence a critical problem solving which is pivotal towards development of safe and trustworthy AI.

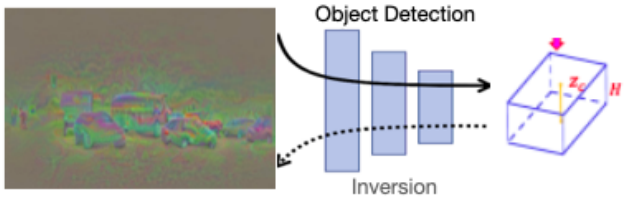


Figure 1. A random image (left) is fed as input to object detector. A fixed bounding box (right) is given as the ground truth, we perform back-propagation on the input by freezing model weights to get the optimum input that can generate the desired output

Recently, model inversion [2, 3, 9] was introduces as a new approach for achieving model interpretability. Inversion techniques deal with re-creating an optimal *input* which maximizes the probability of a given *output*, given a set of frozen model *parameters*. Fig.1 showcases the inversion approach wherein for a given layout of a scene with multiple 3D bounding boxes, the task is to produce an optimal *input* image consistent with given layout. Visualizing the generated *input* image gives key insights into what visual features are relevant and being used by model to make predictions. It can also highlight if the trained model misses or ignores certain 'important' features from original input thereby facilitating considerable insights into the learnings of a neural model.

While there has been some effort [3, 9] towards interpreting models for 2D inputs using inversion. However, to the best of our knowledge there is not significant effort towards inverting and interpreting deep neural networks for 3D vision. The task of 3D model inversion is also considerably more challenging when compared with 2D because of (1) diverse and complex 3D input representations (Mesh, voxels, point-clouds, SDFs etc.) as opposed to images; (2) significant task-specific non-differentiable processing stages; (3) lack of proper 3D shape priors or regularizers etc. This work aims to interpret 3D models of detection and classification using the technique of model inversion.

2. Related Work

3D Object Detection The aim of 3D object detection is to predict three-dimensional bounding boxes (cuboids).

There is considerable amount of work on 3D object detection using Lidar point clouds [6, 8, 22, 29] or RGB images [23, 24]. A vast majority of recent works [7, 14, 19, 28] also explore various fusion techniques to combine inputs from multiple modalities. These models perform exceedingly well on Autonomous Driving benchmarks [1, 5, 20]. CenterNet [32], is a monocular RGB based 3D object detection model with SOTA results on KITTI [5] benchmark. However deep neural models are exceedingly difficult to interpret. Hence to some extent, we know they work well but don't know what they learn. In this work, we aim to invert monocular RGB based 3D object detection models and further visualize their learning.

3D object classification Qi *et al*. [15] explored a novel memory efficient way approach to perform 3D perception tasks like segmentation, classification etc. directly on sparse point-clouds. Qi *et al*. [16] further extended this work by using deep hierarchial feature learning. There have been several works [21, 26, 30] in recent years aimed at using point clouds for 3D object classification. This project however explores to interpret the learning of these models through the lens of inversion.

Model interpretability Miller [12] defined interpretability as, *'the degree to which a human can understand the cause of a decision'*. Simonyan *et al*. [18] proposed a simple approach of visualizing the gradients of each pixel for the 'interest' class. This work however suffered from gradient saturation problems. Zhou *et al*. [31] proposed Class Activation Mapping (CAM) for identifying discriminative regions and trades off performance for more transparency. Selvaraju *et al*. [17] extended this approach by using gradients flowing through final convolution layer to produce coarse localization maps. However, even with these approaches it is difficult to know if even the explanation is correct or not. Molnar [13] highlights some of these concerns *e.g*. these approaches could act as edge-detectors and be insensitive to model and data. Model inversion a recent approach to attempt model interpretability and is aimed at re-creating the optimal input based on the perceived output and model parameters. By re-creating the input 'data', this approach explicitly uses the model parameters and can be a good measure of generalizability.

Inversion There are numerous works on inverting deep learning-based classification architectures that act on 2D images [3, 9]. Recent works [2] extend the scope of inversion to complicated object detection models with non-differential operations, and stage-wise training. However, there are no works on inverting deep learning architecture for classification or detection that operate in the 3D world. We aim to do just that.

3. Methodology

In model inversion, we aim to generate inputs that perfectly match the representations in an intermediate or output layer of neural network. Mathematically, if y is the representation at a particular layer and the model is denoted by f, we obtain:

$$x' = \arg\min_{x} \mathcal{L}(y, f(x)) \tag{1}$$

We try to predict the input which maximizes the probability of a particular output. This is in direct contrast to standard computer vision tasks where the optimization is over the network f rather than x. Figure 1 illustrates the problem where left most image maximizes the probability of output.

We perform model inversion for 3D models in 2 settings - RGB-based 3D object detection and point cloud-based 3D classification.

3.1. Inverting RGB-based 3D Object Detection

We adapt [2] to our case where the output bounding boxes are 3D compared to 2D. We invert the CenterNet [32] 3D model that has been trained on KITTI [5] dataset for the task of 3D object detection. The CenterNet is a heatmap based model where the model outputs heatmaps and not the bounding boxes. The process involves non-differentiable pre-processing steps to generate the input for the model and post-processing to convert the output heatmaps into the detection boxes. In inversion, our aim to understand the model and not the pre-processing and post-processing components that are hand-designed and completely interpretable. Therefore, we suggest 2 modifications in [2] - a) instead of inverting the end-to-end pipeline, we only invert the model portion in the pipeline since it is the only block that is black-box. b) we use the ground-truth labels as the starting point for the inversion since our goal is to understand what the model has learnt and not how good its detection accuracy is.

Concretely, we start with a gaussian image (using ImageNet mean and variance) and pass it though the model. The model is set in eval mode and its weights are frozen. We perform forward pass to calculate the loss using the standard losses of the CenterNet model. The training loss for CenterNet is given by

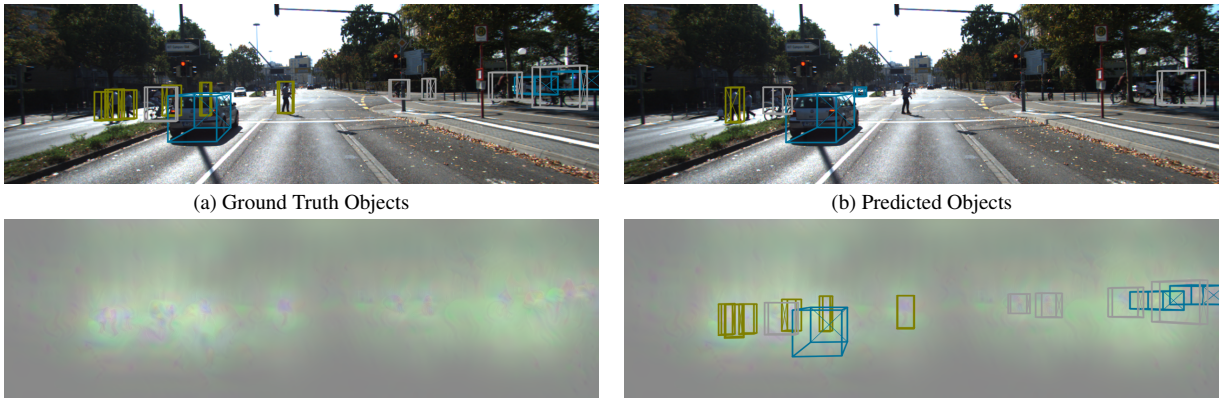
$$\mathcal{L}_{model} = \mathcal{L}_{heatmap} + \mathcal{L}_{depth} + \mathcal{L}_{dim} + \mathcal{L}_{rot} + \mathcal{L}_{wh} + \mathcal{L}_{offset} \tag{2}$$

Additionally, as suggested in [2], we also introduce a total variance (TV) loss in order to reduce the variance of the generated input.

$$\mathcal{L}_{reg} = \mathcal{L}_{TV} \tag{3}$$

The overall loss equation is given by

$$\mathcal{L}_{tot} = \mathcal{L}_{model} + \mathcal{L}_{reg} \tag{4}$$



(a) Ground Truth Objects

(b) Predicted Objects

(c) Inverted image

(d) Predicted Objects for inverted image

Figure 2. Figure shows the a) ground truth boxes, b) predicted boxes for original image, c) inverted image using ground truth objects d) inverted image along with detected objects. We can observe in b) that original image misses certain nearby objects - person in center, left side but is able to detect the distant car (blue left side) even though its not annotated in the ground truth. However, the inverted image d) is able to detect all the objects same as ground truth.

This gradient is backpropagated through the model to the input and only the input is updated keeping the model weights frozen. We perform these steps for 2000 iterations to get the optimized input for the given ground-truth labels. We perform clipping on each iteration to ensure that all the input pixel values are within the variance range of the mean in order to ensure stability. On every 5 iterations, we perform gaussian blur of the image to reduce the variance of the input as it gets updated on each iteration.

3.2. Inverting Point Cloud-based 3D Classification

In this experiment, we wish to invert a 3D object classification network $\mathcal{F}$ to obtain the input point cloud that triggers the prediction of a particular target class c. The input point cloud $\mathcal{P}$ is essentially a set of 3D coordinates. The inversion process starts with a randomly initialized point cloud and updates it through gradients obtained while maximizing the final class probability. Mathematically,

$$\mathcal{L} = \arg\min_{\mathcal{P}} \text{CrossEntropy}(\mathcal{F}(\mathcal{P}), \mathbb{1}_{i==c}) + \Phi(\mathcal{P}) \quad (5)$$

where $\mathcal{L}$ denotes the loss function to be minimized over the point cloud $\mathcal{P}$, which is cross entropy loss in the case of classification. $\Phi(\mathcal{P})$ is a regularizer defined on the point cloud. $\mathbb{1}_{i==c}$ denotes a one-hot-encoded vector with the target class index set as 1.

We utilize PointNet++ [15,16] for our study. PointNet++ is a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. At each level in the hierarchy, an MLP layer is applied on the inputs point which is followed by a max-pooling layer to retain the informative points reducing the number of points at each level. The model has been trained on point clouds obtained from 3D CAD models dataset- ModelNet40 [27]. ModelNet40 is a collection of internet-obtained 3D CAD models of 40 everyday objects.

3.2.1 Initialization

We explore multiple ways to initialize a point cloud. i) Uniformly at random: The points are randomly initialized in the range [-0.2, 0.2]. ii) Zero initialization: All points are initialized to 0. iii) Uniform Spherical Initialization: The points are randomly initialize in a sphere around 0. The inversion process would start with a point cloud initialized through one of the methods mentioned above, and then optimized with back-propagation.

3.2.2 Regularization

Just as Gaussian blurring is utilized to regularize RGB images, we regularize the input point cloud using a neighborhood based *smoothing* or averaging operation. However, we need to define what consists neighborhood of a point in a point cloud to be able to apply the smoothing operation. To this end, we create a *neighborhood-mesh* by linking a point with its nearest K points in the euclidean space. Once the neighbors set $\mathcal{N} = \{\dots\}$ is obtained, the regularization can be simply written as:

$$p_i = p_i + \frac{\lambda}{K} \sum_{j=\mathcal{N}_0}^{\mathcal{N}_K} (p_j - p_i) \quad (6)$$

where p_i denotes the 3D coordinates of point i and λ the degree of regularization. Setting $\lambda = 0$ amounts to no regularization, whereas setting $\lambda = 1$ amounts to replacing the position of the current point with the average of its neighbor's positions.

(a) Ground Truth Objects

(b) Predicted Objects

(c) Inverted image

(d) Predicted Objects for inverted image

Figure 3. In continuation to fig 2, we analyze a more difficult scenario- cluttered cars on a highway. We can observe that the inverted image c) has some artifacts appearing but still the detection on inverted image is identical to the ground truth. Similar to trend observed in fig 2, b) misses some cars while prediction.

The above regularizer is not applied as a loss but as a post-processing step after each gradient update, in a manner similar to weight-clipping. In the current implementation, we apply the regularization operation every 10 iterations.

4. Results

In this section we analyze the performance of our inversion based approach for the tasks of object detection and classification. The goal of our project is to leverage inversion for enabling transparency in the learning of a neural network. Since *interpretability* in itself is abstract and ill-defined, it is difficult to perform quantitative comparisons. Hence for this section we qualitatively analyze the inversion based approach over 3D perception tasks of detection and classification.

4.1. Inverting RGB-based 3D Object Detection

We perform our analysis for randomly sampled images in the validation set. We perform inversion using given set of detections as described in 3.1 to obtain the inverted image. Fig 2 shows the inverted image for a given sample. On close examination, we can observe that the inverted image has similar texture around the objects as the original image.

Next, we perform forward pass on both - original image and inverted image and compare it with the ground truth annotations. Fig 2 (a) shows the ground truth, (b) model output of original image and (d) model output of the inverted image for a given sample. We observe that when we give the original image as input to the model, there are several missed detections for *e.g.*- the person in the center, the persons in the left and cycles in the right. However, the inverted image is able to generate very high quality detections which are identical to the ground truth. The model weights are frozen during the entire process of inversion and we also observe that objects created by inversion have similar structure as those of natural objects (*e.g.* inverted image of a person in fig. 2 is similar to a standing human). We can hence say that the inversion process has some learnt meaningful representation of objects in the image. At the same time, this also shows us that the training of the original model [32] has certain issues as it is not able to detect visually simple person in the center.

In fig 3 we pick a more difficult sample (multiple cars cluttered in left). We observe that the inverted image has some weird artefacts with no clear object boundaries in the cluttered region. Since, CenterNet outputs heatmaps, we believe that the cluttered objects cause the output heatmaps to dissolve for the cluttered environment creating similar effect on the inverted image as well. Interestingly, we still observe that although the model output for original image misses several detections, the model output for inverted image has identical detections as the ground truth. Since, there are no clear object boundaries in the inverted image, it's hard to say that the model has learnt object representations in the cluttered environments.

4.2. Inverting Point Cloud-based 3D Classification

The inversion is successful for all our runs i.e. the input point cloud converged to a solution that was classified as the target class by the network. However, in terms of being interpreted by humans, the results varied a lot depending on the inversion settings.

We visualize the obtained inverted point clouds in Figure 4. It is interesting to see that the PointNet++ [16] model captures global shape of the object and ignores finer details. For instance, the inverted point cloud for the person class looks like a vertical pole. For car class, the four corners are identified and for the airplane class, the end points of the wings and the front is identified.

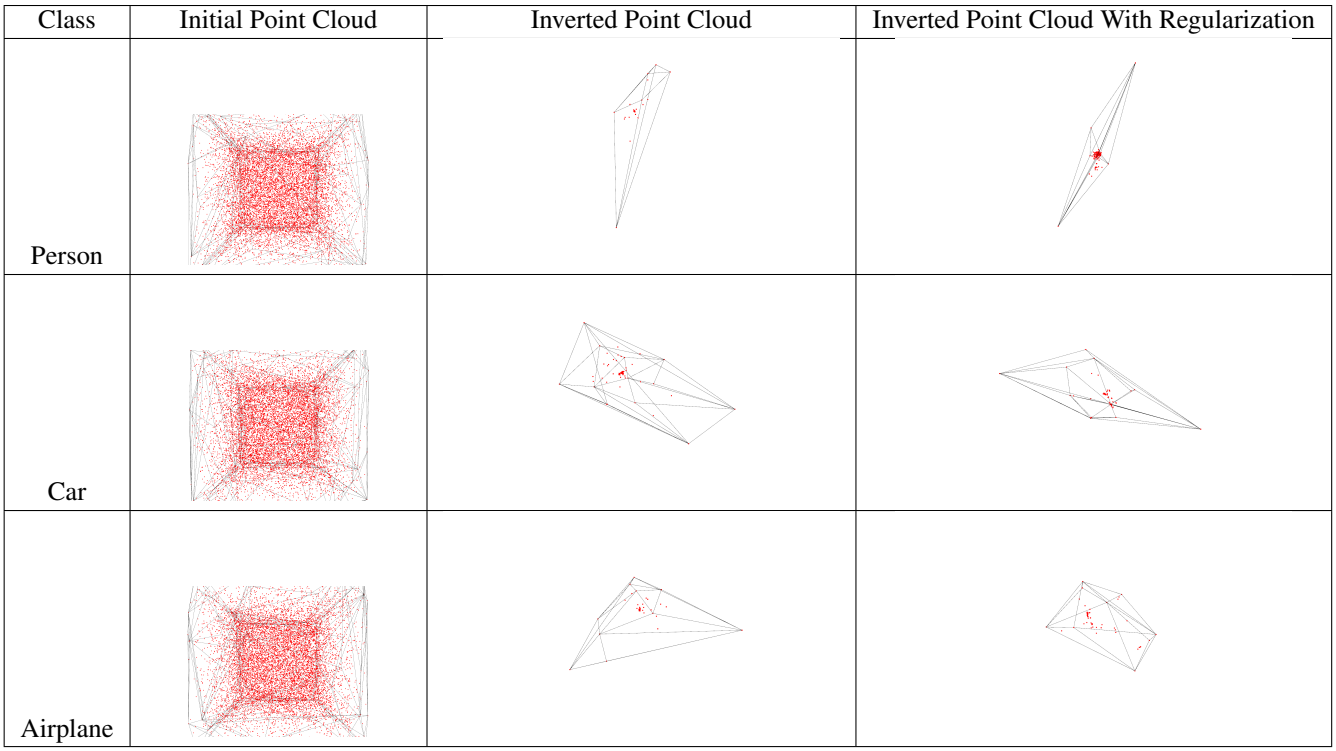


Figure 4. Inverted examples for point cloud-based 3D classification. We observe that we get visually similar inverted point cloud for with/without regularization implying that regularization has little impact on inversion for point clouds.

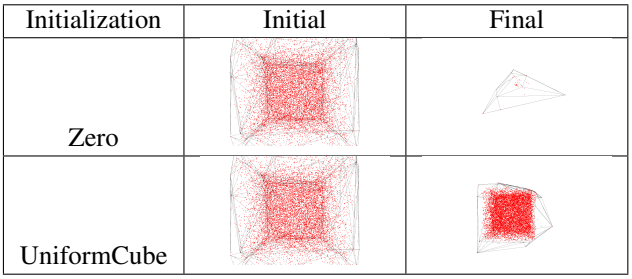


Figure 5. Comparison of zero (nearly zero) and uniform different initialization schemes for point cloud inversion. We observe that zero initialization works better than the uniform cube initialization.

Observation 1: The success of inversion depends on the initialization of point cloud as shown in Figure 5. Contrary to models trained on images, inverting 3D models trained on point clouds is heavily dependent on the initialization. From our experiments we observe that if the point cloud is initialized randomly, the inversion process doesn't drastically change the initial positions leading to an inverted output that visually looks random. Surprisingly, initializing the points to be either zero or uniformly spread out really close to zero results in better looking outputs.

Observation 2: Regularization has little effect on the obtained inverted point cloud as depicted in Figure 4. For our experiments, we fix the regularization strength λ to be 1. We observe that the resulting inverted point cloud looked more *spherical* ie. points were spread out around the center. However, visually the results do not look better than the non-regularized version. In fact, for objects like car, the regularization hurts a bit.

5. Conclusion

We invert the RGB-based 3D object detector and LiDAR-based 3D image classification model. For RGB-based 3D object detector, we observe that with proper hyperparameter tuning along with various optimization tricks, model inversion works well and we are able to mimic the ground truth for multiple images. The inverted images are also contain meaningful boundaries around the objects in the images if the scenes are not too cluttered.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 2

[2] Ang Cao and Justin Johnson. Inverting and understanding object detectors, 2021. 1, 2

[3] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837, 2016. 1, 2

[4] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 1

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2

[6] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020. 1, 2

[7] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 2

[8] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[9] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 1, 2

[10] Priyanka Mandikal, Navaneet K. L., Mayank Agarwal, and Venkatesh Babu Radhakrishnan. 3d-lmnet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image. In *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*, page 55. BMVA Press, 2018. 1

[11] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1

[12] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019. 2

[13] Christoph Molnar. *Interpretable Machine Learning*. 2019. 2

[14] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 2

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2, 3

[16] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017. 2, 3, 4

[17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 2

[18] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2

[19] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019. 2

[20] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 1, 2

[21] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–66, 2018. 1, 2

[22] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 482–490, New York, NY, USA, 2019. Association for Computing Machinery. 2

[23] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. *arXiv preprint arXiv:2104.10956*, 2021. 2

[24] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019. 2

[25] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020. 1

[26] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 2

[27] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 3

[28] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018. 2

[29] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, June 2021. 1, 2

[30] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 1, 2

[31] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. 2

[32] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 2, 4