Dense Captioning for 3D Scenes with SparseConv

Alex Khakhlyuk
Technical University of Munich

Anurag Singh
Technical University of Munich

1. Introduction

There has been a lot of previous work in the area of cross-modal scene understanding, especially for the tasks of image captioning [28] and the related task of dense captioning [16]. However, the research has been focused on dense captioning for 2D data only, until recently, Scan2Cap has proposed a method for dense captioning of RGB-D scans [5]. The task of 3D dense captioning consists of taking a 3D point cloud as input and predicting the bounding boxes of objects in the scene as well as their corresponding natural language descriptions. The biggest challenges in dense captioning are successfully modeling the relations between objects and leveraging the 3D information of objects, such as actual object size or object location, and thus achieving more accurate descriptions.

To deal with these challenges, Scan2Cap has coupled a 3D detection network with a graph module and an attention-based captioning module to create a powerful method that is trained end-to-end. Scan2Cap has leveraged the 3D representation of the scenes and has significantly outperformed state-of-the-art 2D captioning methods. In this work, we wanted to explore several modifications and improvements to Scan2Cap architecture. Our main contribution is replacing the old feature extraction backbone with a new, more powerful one. We explain our proposed modification in Section 4.

2. Related Work

3D Object Detection: The availability of large 3D datasets [8, 1] has helped in research on 3D object detection and instance segmentation tasks. Several methods have been proposed for task of object detection [26, 3, 21, 6] and also for instance segmentation [14].

3D vision and Language: The idea of grounding visual stimulus in natural language and vice-versa has gained lot of attention with introduction of tasks such as image captioning, dense captioning and text-to-image generation [24, 25] or visual grounding [13, 18]. However, vision and language in 3D still remains relatively unexplored [5].

Image Captioning: Johnson et al. in their work [15], introduced a dense captioning task. Their work relates significantly to Dense captioning in 3D. The task generates image captions for all objects in an image. However, the limitations of the method are that it doesn't consider regions outside salient regions. In [29] Yang et al, propose a method to overcome this issue by having global features as input to the captioning module. However, most dense image captioning works suffer from constraints of a single view and are sub-optimal in capturing larger context available in a 3D scene[5].

3. Point-set vs SparseConv

The two main types of architectures used for deep learning on point clouds have been point-set networks [22, 23] and sparse convolutional networks [9, 10, 11, 7].

Point-set networks treat input scenes as unstructured sets of points. The main building blocks of point-set networks are fully-connected layers and pooling operations. The advantages of point-set networks are their lower memory requirements, faster training, and inference. The downside is that they don't use the spatial structure of the point cloud very well. This results in lower accuracy, dependence on the sampling quality, and worse performance on incomplete objects.

Sparse convolutional networks, represent pointclouds as sparse 3D voxel grids and the main layer is a sparse convolution. The spatial structure introduced by voxel grids and convolutions is very beneficial for processing point clouds. Sparse convolutional networks are consistently outperforming other architectures in terms of accuracy in most 3D understanding benchmarks [26, 8, 3].

4. Method

Our project explores how sparse convolutional backbone affects the accuracy of object detection and dense captioning. Scan2Cap uses VoteNet for object detection, which is a point-set network. We aimed to replace it with a SparseConv detector hoping to improve both detection and captioning quality. Scan2Cap architecture (shown in Fig.1) can be summed up as follows. First, a feature extraction backbone is used to process the point cloud. Then, the enriched representation of the point cloud is passed into the detection

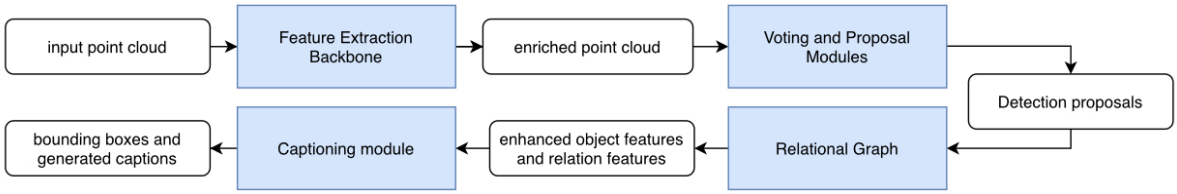


Figure 1: An overview of the Scan2Cap architecture.[5]

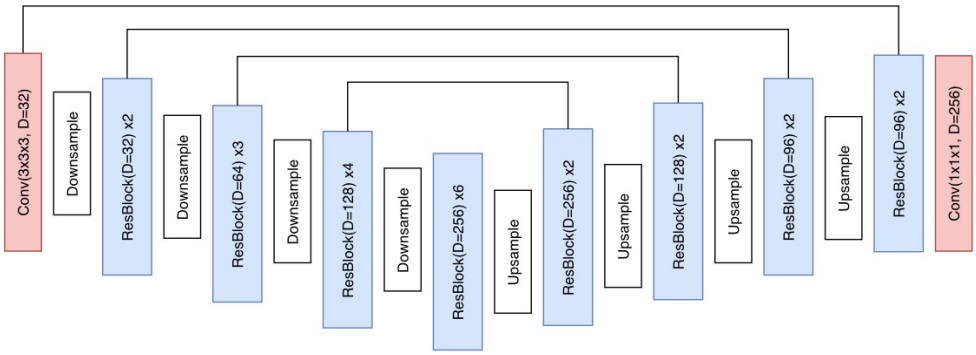


Figure 2: Architecture of our SparseUnet backbone. D specifies the number of channels in the layer. Downsampling blocks are sparse convolutions with stride 2, upsampling blocks are sparse transposed convolutions with stride 2. [7]

modules, which locate the objects in the scene and predict bounding boxes and classes for each of the detection proposals. Detected objects are then passed into the graph and captioning modules to generate a text description for each of the detections. The quality of captions depends both on the accuracy of the detector and the quality of representation extracted by the feature extraction backbone.

In this project, we change the first part of the pipeline, the feature extraction backbone. Original Scan2Cap uses PointNet++ as the backbone. PointNet++ is a point-set network and we replace it with a SparseConv feature learning backbone. In particular, we propose SparseUnet architecture shown in Fig.2. Our SparseUnet backbone is inspired by the semantic segmentation network proposed in [7].

Using SparseConv instead of a point-set network comes at a cost of slower training and inference. This trade-off is acceptable, as Scan2Cap is an offline method. Moreover, as we will see in the experiments section, the difference in speed between the two backbones becomes less significant when they are used as part of Scan2Cap. This is due to the fact that graph and captioning modules require a substantial amount of processing time themselves.

4.1. Using other sparse convolutional detectors

It is possible to replace the whole VoteNet with the latest state-of-the-art SparseConv detector. Good candidates would be GSDN [12] or a modification of PointGroup [14]. Although these networks achieve high accuracy, they introduce several other novelties compared to VoteNet, which would make a direct comparison between a point-set network and a SparseConv network not possible. For this reason, we are keeping the voting and proposal modules from VoteNet while only replacing the PointNet++ backbone. In future work, VoteNet can be replaced completely by the latest state-of-the-art SparseConv detection method.

5. Implementation Details

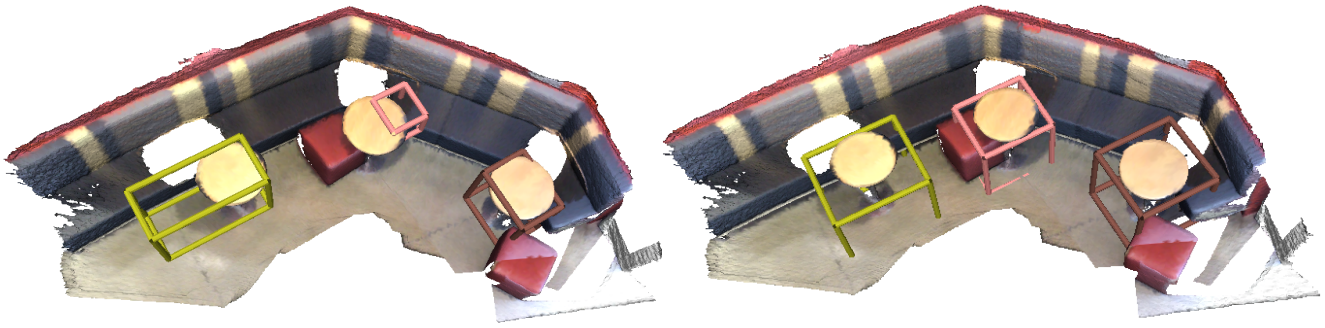
We base our implementation on the original Scan2Cap code written in PyTorch [5, 20]. We use MinkowskiEngine to implement our sparse convolutional backbone[7]. We adopt the training schedule and hyper-parameters used in Scan2Cap. We use a single Nvidia 1080 TI GPU in all of our experiments.

6. Experiments

Dataset: In our experiments we follow Scan2Cap and use ScanRefer [4] dataset which contains 800 ScanNet scenes with 11,046 objects and 51,583 corresponding text descriptions. The descriptions contain information related to object's appearance and its spatial relationship with other annotated objects in the scene.

Data Representation: Points contain color, normals and height from the ground as features. When using the SparseUnet backbone, the input point cloud is voxelized at resolution 2cm. In contrast to Scan2Cap, we don't use extracted Enet features for the sake of simplicity.

Train&val splits: We use standard ScanRefer [4] bench-



(a) PointNet++ (b) SparseUnet

Figure 3: Object detection performance on table class over a scene for PointNet++ and SparseUnet backbones

mark split. Similar to Scan2Cap experiment settings, in our experiments the 36,665 samples are taken for training and 9,508 for validation and also ensure that there is no scene in common for val and train. Our results are reported on val data as test set is hidden officially.

Evaluation Metrics: We follow Scan2Cap in choice of evaluation metrics. For object detection, we use the standard mAP (mean average precision) metric. To measure the quality of generated text descriptions we use image captioning metrics CiDEr [27], BLEU-4[19], METEOR [2] and ROUGE [17]. We evaluate both mAP and captioning metrics at two different IoU thresholds: 0.25 and 0.5. We use C, B-4, M, R as abbreviations for the CiDEr, BLEU-4, METEOR and ROUGE metrics.

6.1. Object Detection:

We do a comparative study of 3D object detection performance of SparseUnet and PointNet++ as described in Table 1. We observe that SparseUnet reaches significantly higher mAP scores, specially with a more challenging 0.5 IoU threshold.

backbone	mAP@0.25IoU	mAP@0.5IoU
PointNet++	51.64	28.80
SparseUnet	**52.05**	**33.59**

Table 1: Comparison of PointNet++ and SparseUnet on 3D object detection task

6.2. Dense Captioning

Comparison of dense captioning performance can be seen in Table 2 and Table 3. We see that SparseUnet outperforms PointNet++ on most metrics. The highest gain is observed for CiDer and BLEU-4.

backbone	B-4	C	M	R
PointNet++	31.54	44.59	25.06	**53.67**
SparseUnet	**32.30**	**49.52**	**25.52**	53.53

Table 2: Comparison of PointNet++ and SparseUnet on dense captioning task with IoU threshold of 0.25

backbone	B-4	C	M	R
PointNet++	21.67	31.58	21.10	43.87
SparseUnet	**23.37**	**35.59**	**21.66**	**44.34**

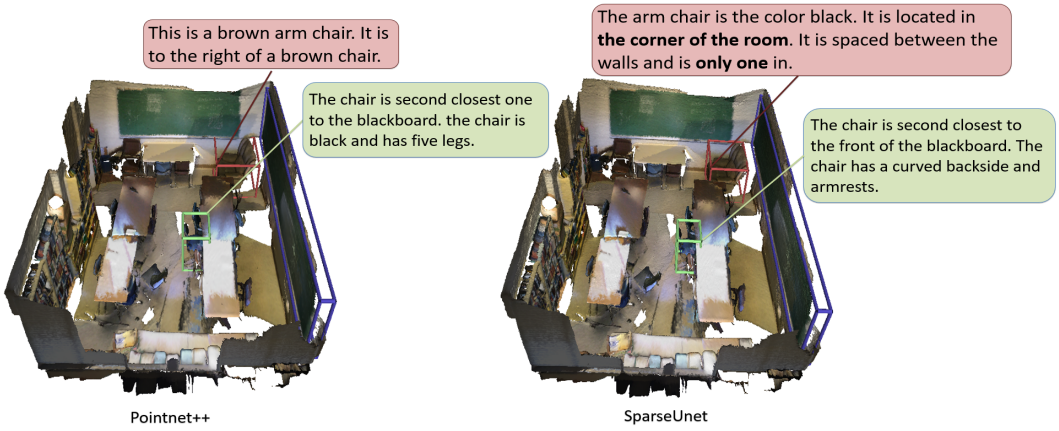
Table 3: Comparison of PointNet++ and SparseUnet on dense captioning task with IoU threshold of 0.5

6.3. Time & Space Complexity

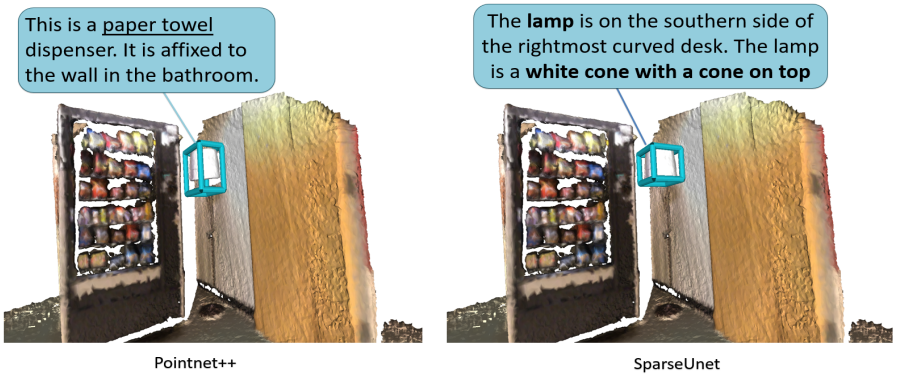
We compare the training time, inference time and memory requirements of the two architectures in Table 4. We observe that memory requirements of SparseUnet architecture are marginally higher than PointNet++. Both networks can still be trained on a single gpu. As expected, training and inference times are significantly higher for SparseUnet. For detection, PointNet++ is 3-4x faster at detection. However, the difference is smaller for the task of captioning. SparseUnet is only 1.5-2x slower than PointNet++ in this setting.

6.4. Qualitative Results

We also perform a qualitative comparison between SparseUnet and PointNet++ for tasks of object detection and dense captioning. Fig. 3 compares the quality of bounding boxes for one of the scenes. We can see that SparseUnet predicts bounding boxes a lot more accurately. In Fig. 4 we compare captioning performance using two scenes as examples. Looking at scene 1, we see that captions generated by SparseUnet are better at capturing the global semantics of the scene and relations between objects. In case of the armchair, SparseUnet network describes the absolute position of the object in the room, while PointNet++ only grounds



(a) Scene 1



(b) Scene 2

Figure 4: Captioning visualization of SparseUnet (left) and PointNet++ (right) in each scene. In scene 1, Armchair and chair bounding-box predictions have been highlighted in red and green respectively. Ground truth bounding box of blackboard being referenced in both captions is highlighted using blue. In scene2, lamp predictions are highlighted using blue.

Task	Backbone	Memory	Forward	Forward+ Backward	Training time	Parameters
Detection	PointNet++	6.7GB	0.22s	0.9s	7h	1.0M
	SparseUnet	7.5 GB	0.82s	2.5s	23h	38M
Captioning	PointNet++	8.0 GB	0.82s	1.4s	39h	2.7M
	SparseUnet	8.8GB	1.15s	3s	70h	40M

Table 4: Memory, inference and training time requirements for SparseUnet and PointNet++ on detection and dense captioning tasks

the armchair relative to another nearby chair which is less descriptive. In scene 2, we see that PointNet++ network misclassifies the lamp, while SparseUnet predicts the class correctly and generates a meaningful caption.

7. Conclusion

Our experiments show that our proposed SparseUnet backbone gives an overall performance improvement over the PointNet++ in tasks of object detection and dense captioning. SparseUnet backbone leads to more accurate bounding boxes. It also seems to be better at capturing the global semantics of the scene, which helps Scan2Cap generating captions that are more meaningful and insightful. This gain in accuracy comes at a cost of slower training and inference, but the trade-off is worth it for offline applications.

References

[1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[2] S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[3] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[4] D. Z. Chen, A. X. Chang, and M. Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 202–221. Springer, 2020.

[5] D. Z. Chen, A. Gholami, M. Nießner, and A. X. Chang. Scan2cap: Context-aware dense captioning in rgb-d scans. *arXiv preprint arXiv:2012.02206*, 2020.

[6] B. Cheng, L. Sheng, S. Shi, M. Yang, and D. Xu. Back-tracing representative points for voting-based 3d object detection in point clouds. *arXiv preprint arXiv:2104.06114*, 2021.

[7] C. Choy, J. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[9] B. Graham. Spatially-sparse convolutional neural networks, 2014.

[10] B. Graham. Sparse 3d convolutional neural networks, 2015.

[11] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks, 2017.

[12] J. Gwak, C. B. Choy, and S. Savarese. Generative sparse detection networks for 3d single-shot object detection. In *European conference on computer vision*, 2020.

[13] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564, 2016.

[14] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020.

[15] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4565–4574, 2016.

[16] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[17] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[18] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20, 2016.

[19] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[21] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds, 2019.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[24] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

[25] S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio. Chatpainter: Improving text to image generation using dialogue. *arXiv preprint arXiv:1802.08216*, 2018.

[26] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[27] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[28] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[29] L. Yang, K. Tang, J. Yang, and L.-J. Li. Dense captioning with joint inference and visual context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2193–2202, 2017.