Generative Sparse Detection Networks for 3D Single-shot Object Detection

JunYoung Gwak[1], Christopher Choy[2], and Silvio Savarese[1]

[1] Stanford University {jgwak,ssilvio}@stanford.edu
[2] NVIDIA cchoy@nvidia.com

Abstract. 3D object detection has been widely studied due to its potential applicability to many promising areas such as robotics and augmented reality. Yet, the sparse nature of the 3D data poses unique challenges to this task. Most notably, the observable surface of the 3D point clouds is disjoint from the center of the instance to ground the bounding box prediction on. To this end, we propose Generative Sparse Detection Network (GSDN), a fully-convolutional single-shot sparse detection network that efficiently generates the support for object proposals. The key component of our model is a generative sparse tensor decoder, which uses a series of transposed convolutions and pruning layers to expand the support of sparse tensors while discarding unlikely object centers to maintain minimal runtime and memory footprint. GSDN can process unprecedentedly large-scale inputs with a single fully-convolutional feed-forward pass, thus does not require the heuristic post-processing stage that stitches results from sliding windows as other previous methods have. We validate our approach on three 3D indoor datasets including the large-scale 3D indoor reconstruction dataset where our method outperforms the state-of-the-art methods by a relative improvement of 7.14% while being 3.78 times faster than the best prior work.

Keywords: Single shot detection, 3D object detection, generative sparse network, point cloud

1 Introduction

3D reconstructions have become more commonplace as a complete reconstruction pipeline become built into consumer devices, such as mobile phones or head-mounted displays, for applications in robotics and augmented reality. Among these applications, perceptions on 3D reconstructions is the first step allowing users to interact with a virtual world in 3D. For example, indoor navigation applications can aid a user to localize objects, and mixed reality applications need to track objects to give users information relevant to the current status of their surroundings. Many of these virtual-reality and mixed-reality applications require identifying and detecting 3D objects in real-time.

However, unlike 2D images where the input is in a densely packed array, 3D data is scanned or reconstructed as a set of points or a triangular mesh. These

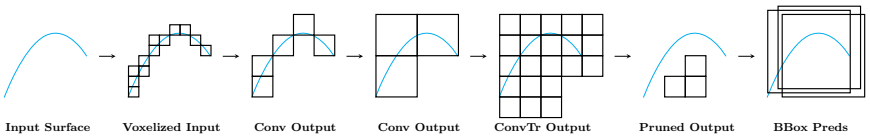

Fig. 1: The top-down view of the cross-section of our simplified 3D sparse anchor generation pipeline: a 3D scanner samples the surface of an object which we convert to a sparse tensor. Then, an encoder extracts hierarchical sparse tensor features with a series of convolutions. During the decoder stage, we apply a transposed convolution to upsample and expand the support of the sparse tensor. Finally, we prune out unnecessary supports that do not contain anchors and make bounding box anchor predictions.

data occupy a small portion of the 3D space and pose unique challenges for 3D object detection. First, the space of interest is three dimensional which requires cubic complexity to save or process data. Second, the data of interest is very sparse, and all information is sampled from the surface of objects.

Many previous 3D object detectors proposed various methods to process cubically growing sparse 3D data, and can be categorized into one of two branches: 3D object detection by converting sparse 3D data into a dense representation [19, 28, 1, 15, 13] or by directly feeding a set of points into multi-layer perceptrons [24, 35]. First, dense 3D representation for indoor object detection [28, 1, 13] uses volumetric features which have memory and computational complexity of $O(N^3)$ where N is the resolution of the space. This representation requires large memory, which prevents the utilization of deep networks and requires cropping the scenes and stitching the results to process large or high-resolution scenes. Second, multi-layer perceptrons that process a scene as a set of points limit the number of points a network can process. Thus, as the size of the point cloud increases, the method suffers from either low-resolution input which makes it difficult to scale the method up for larger scenes (see Section 5.2) or apply sliding-window style cropping and stitching which prevents the network to see a larger context [35].

We instead propose to resolve the cubic complexity with our hierarchical sparse tensor encoder, adopting a sparse tensor network [8] to efficiently process a large scene fully-convolutionally. As we use a sparse representation, our network is fast and memory-efficient compared with a single-shot method that uses dense tensors [13]. It allows our network to adopt extremely deep architectures while requiring a fraction of the memory and computation. Also, compared with multi-layer perceptrons, our method scales to large scenes without sacrificing point density or the receptive field size of a network by cropping a scene into smaller windows [24, 35].

Another key challenge of a 3D object detector is that the support of the input 3D scans and the support of the object bounding box anchors are disjoint. In other words, we have samples of 3D points on the surface of the objects, but

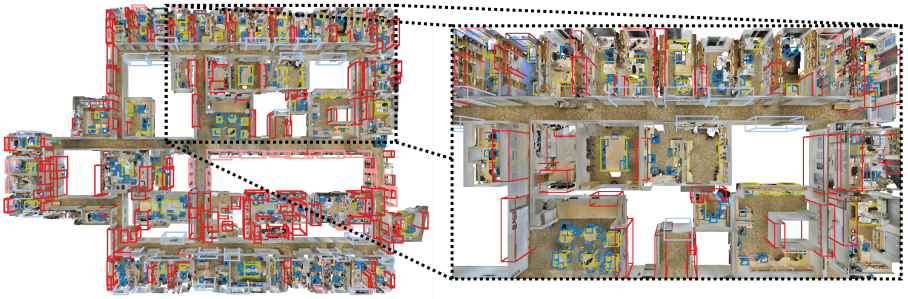

Fig. 2: Detection results on the entire S3DIS building 5: Our proposed method can process 78M points, 13984m^3, 53 room building as a whole in a *single fully-convolutional feed-forward pass*, only using 5G of GPU memory. Left: bird-eye-view of the entire building 5, Right: partial view of the same building.

not on the center of the object where a bounding box anchor is located. This is due to the fact that many objects are convex and we cannot directly observe the object center. For this, we propose a generative sparse tensor decoder that repeatedly upsamples the support of input to expand and cover the support of anchors while discarding unlikely object centers to maintain minimal runtime and memory footprint (Fig. 1).

To sum, we propose Generative Sparse Detector Network (GSDN), a deep *fully-convolutional* single-shot 3D object detection algorithm with a sparse tensor network. Our single-shot 3D object detection network consists of two components: an hierarchical sparse tensor encoder which efficiently extracts deep hierarchical features, and a generative sparse tensor decoder which expands the support of the sparse input to ground object proposals on. Experimentally, GSDN outperforms the state-of-the-art methods on two large-scale indoor datasets while being faster than the best prior work. We also analyze the speed and memory footprint of the model and demonstrate the extreme scalability of our method on orders of magnitudes larger 3D scenes (Fig. 2).

2 Related Work

3D Indoor Object Detection. In a 3D indoor setting or 3D indoor datasets [5, 1], the distribution of object placement creates unique challenges: objects such as lamps and ceiling lights can be placed on a wall or a ceiling, or objects can be placed on top of another object such as a desk or a bed. However, such challenging setup does not exist in outdoor datasets and most 3D outdoor object detectors simply project the 3D problem into a 2D ground plane [19, 15, 38].

Thus, in this section, we cover 3D indoor object detection specifically. The indoor 3D object detection using neural networks can be classified into one of the following categories: sliding-window with classification, clustering-based methods, bounding-box proposal, or combinations of the above methods. First, the sliding

window with classification extracts a 3D patch for object classification which is used as a simple object detector [28, 1].

Second, clustering-based methods learn features or vectors in a metric space where clustering results in instance segmentation. Lahoud *et al.* [14] uses metric learning to train the feature space. Liu *et al.* [17], Yi *et al.* [36], Wang *et al.* [33], and Qi *et al.* [24] predict object centers per 3D point and cluster the center votes.

Third, the bounding box proposal methods adopt 2D rectangular bounding box proposal methods to 3D. Wang *et al.* [32] proposed Vote3D, which predicts 3D bounding boxes on a sparse grid for object detection. Yang *et al.* [35] directly predicts bounding boxes from MLP of global point cloud features. Hou *et al.* [13] makes a straight-forward 3D extension of region proposal networks on dense voxels. GSDN is a bounding box proposal method with a crucial difference in maintaining the sparsity of the input point cloud and target anchor space, enabling much faster inference on many orders of magnitude larger scene with better performance than state-of-the-art methods.

3D Generative Networks. Generating 3D shapes from a neural network can be classified into two broad categories: continuous 3D point representations [20, 23, 37, 31] and discrete grid representations [4, 30, 2, 7, 6]. Specifically, within the discrete representations, some use sparse representations for 3D reconstruction which allow a high-resolution voxel or signed-distance-function (SDF) reconstruction [30, 2, 7, 6]. Unlike previous works that focus on the shapes of objects, we use the generative process to predict the bounding box anchors. Also, compared with some sparse generative processes that subdivide voxels [30, 6], our method extends the support with transposed convolutions to cover bounding box anchors which are located behind 3D surface observations.

Sparse Tensor Networks. A conventional neural network processes a dense tensor such as temporal data, images, or videos using a series of linear operations and non-linear operations. Most of the linear operations also use dense tensors for parametrization. In mobile and embedded systems, a sparse parametrization of neural networks [10, 22, 21] has been widely studied to compress a neural network. Graham *et al.* [9] instead proposes to take *spatially* sparse tensors as inputs and generate *spatially* sparse feature maps. Using a sparse tensor as an input has gained more popularity since its success on 3D data processing [8, 9, 2, 3]. We adopt these spatially sparse networks, or sparse tensor networks to scale detection networks to an unprecedented depth and to handle extremely large scenes. Additionally, we propose to *dynamically* generate new coordinates to efficiently support bounding box center coordinates that are often missing in surface-scanned inputs.

3 Preliminaries

In this section, we briefly go over the basic 3D representation, a sparse tensor, and introduce basic operations that are critical for the generative sparse tensor network. Throughout the paper, we will use lowercase letters for variable scalars, t; uppercase letters for constants, N; lowercase bold letters for vectors, $\mathbf{v}$; uppercase

bold letters for matrices, $\mathbf{R}$; Euler scripts for tensors, $\mathscr{T}$; and calligraphic symbols for sets, $\mathcal{C}$.

3.1 Sparse Tensor

A tensor is a multi-dimensional array that can represent high-dimensional data. A sparse tensor of order-D, $\mathscr{T} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_D}$, is a D-dimensional array where majority of its elements are 0. Adopting the conventional sparse matrix representation, a sparse matrix can be represented as a set of non-zero coordinates $\mathcal{C} = \text{supp}(\mathscr{T})$ where supp is the set-theoretic support operator as in standard mathematical terminology, and corresponding features $\mathcal{F}$.

$$\mathscr{T}[x_i^1, x_i^2, \cdots, x_i^D] = \begin{cases} \mathbf{f}_i & \text{if } (x_i^1, x_i^2, \cdots, x_i^D) \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where x_d^i denotes d-th axis coordinate of the i-th non-zero element and $\mathbf{f}_i$ is the feature associated to the i-th non-zero element. These non-zero elements contain information that are equivalent to a sparse tensor $\mathcal{T} \Leftrightarrow (\mathcal{C}, \mathcal{F})$. These sets can also be converted to matrices of COOrdinate representation (COO) $\mathbf{C}, \mathbf{F}$ where each row is an element of the corresponding coordinate and feature sets $(\mathcal{C}, \mathcal{F})$.

3.2 Sparse Tensor for 3D Data Representation

The 3D data of interest in this work uses point clouds or meshes to represent 3D surfaces. We can represent a mesh or a point cloud as a sparse tensor by discretizing the coordinates of vertices or points. This process requires defining the discretization step size (voxel size) which is a hyperparameter that affects the performance of a neural network [3, 2].

4 Generative Sparse Detection Networks

In this section, we propose the generative sparse detection networks for 3D object detection. Unlike the 2D object detection networks [16, 27], we use a sparse tensor as the 3D representation throughout the network including the intermediate features. Thus, all layers such as convolution and batch normalization are well defined for sparse tensors [8, 2]. Throughout the paper, we will implicitly refer to all tensors as sparse tensors and layers as sparse tensor counterparts.

The network consists mainly of two parts: a hierarchical sparse tensor encoder and a generative sparse tensor decoder. The first part of the network generates sparse tensor feature maps that can sufficiently capture geometry and identity of objects and the second part proposes new supports based on the feature maps.

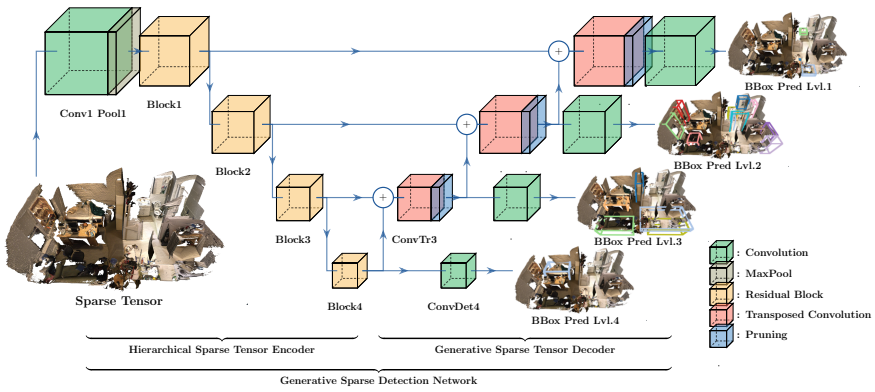

Fig. 3: Network overview: generative sparse detection networks process a sparse tensor input first with a series of strided convolutions followed by a few residual network blocks to generates hierarchical sparse tensor feature maps (Sec. 4.1). The second stage upsamples the sparse tensor feature maps using transposed convolution and pruning (Sec. 4.2). Note that all feature maps are sparse tensors and all layers process sparse tensors fully-convolutionally.

4.1 Hierarchical Sparse Tensor Encoder

We use residual networks [12], specifically high-dimensional variants proposed in Choy *et al.* [2], as the backbone of our model. Note that the backbone network can be replaced with more modern and recent variants. The network consists of residual blocks and strided convolutions that reduce the resolution of the space and increase the receptive field size exponentially. First, the network takes a high-resolution sparse tensor as an input $\mathscr{T}_0$ and generate hierarchical feature maps $\mathscr{T}_l$ with a series of downsampling and residual blocks $f_l(\cdot; \mathbf{W}_l)$ for $l \in [1, ..., L]$. The encoder can be represented succinctly as

$$\mathscr{T}_l \leftarrow f_l(\mathscr{T}_{l-1}; \mathbf{W}_l) \text{ for } l \in [1, ..., L]$$

We cache all of the hierarchical sparse tensor feature maps $\mathscr{T}_l$ for $l \in [1, ..., L]$ which will be fed into the generative sparse tensor decoder.

4.2 Generative Sparse Tensor Decoder

The second half of the network expands the support of the hierarchical sparse tensors feature maps $\mathscr{T}_l$ to cover the support for bounding box anchors. We approximate this process with transposed convolutions (also known as upconvolution, deconvolution). Given an input sparse tensor $\mathscr{T}$, we create an output sparse tensor $\mathscr{T}'$ that $\text{supp}(\mathscr{T}) \subset \text{supp}(\mathscr{T}')$. Yet, not all voxels generated from this process contain bounding box anchors and can be dynamically removed to save the memory and computation cost. Thus, we propose *sparsity pruning*,

where we *dynamically* determine which coordinates to prune based on learned parameters. By applying a transposed convolution followed by sparsity pruning, we increase the resolution of the space while limiting the memory and computation cost. Without pruning, the number of coordinates grows cubically after every transposed convolution and our training pipeline fails from lack of memory. Additionally, we make skip connections between the hierarchical sparse tensor feature maps and the upsampled sparse tensors to recover the fine details of the input.

4.2.1 Transposed Convolution and Sparsity Pruning We use transposed convolutions with the kernel size greater than 2 to not just upsample, but expand the support of a sparse tensor. This process affects the sparsity pattern of a sparse tensor and the support of the output sparse tensor is the stencil or outer-product of the convolution kernel shape on the input sparsity pattern $\text{supp}(\mathscr{T}') = \mathcal{C} \otimes [-K, ..., K]^3$. Mathematically, a transposed convolution on a 3D sparse tensor $\mathscr{T}$ with $\text{supp}(\mathscr{T}) = \mathcal{C}$ can be defined as follows:

$$\mathscr{T}'[x, y, z] = \sum_{i,j,k \in \mathcal{N}(x,y,z)} \mathbf{W}[x - i, y - j, z - k]\mathscr{T}[i, j, k] \text{ for } (x, y, z) \in \mathcal{C}' \quad (2)$$

where $\mathcal{C}' = \mathcal{C} \otimes [-K, ..., K]^3, \mathcal{N}(x, y, z) = \{(i, j, k) || x - i| \leq K, |y - j| \leq K, |z - k| < K, (i, j, k) \in \mathcal{C}\}$, W is the 3D convolution kernel weights and $2K + 1$ is the convolution kernel size. This results in denser sparsity pattern on the output tensor $\mathscr{T}'$ with $\text{supp}(\mathscr{T}') = \mathcal{C} \otimes [-K, ..., K]^3$. Note that unlike the subdivision, the transposed convolution expands a sparse point into an arbitrarily large dense region and multiple regions could overlap with each other (Fig. 4).

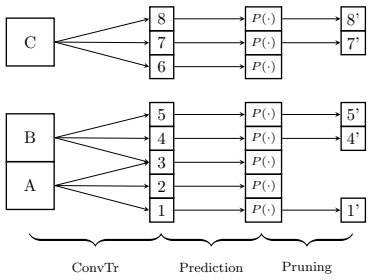

Fig. 4: Expansion and pruning: transposed convolution upsamples a low-resolution sparse tensor into a high-resolution sparse tensor. Then, we prune out some of the upsampled coordinates with sparsity predictions $P_{\text{s}}(\cdot)$.

After a transposed convolution, not all the newly created coordinates contain object bounding box anchors. Thus, we remove some of these voxels that have a small probability of containing bounding box anchors. We denote a function that returns the probability given features at each voxel as $P_{\text{s}}(\cdot)$ and remove all voxels $P_{\text{s}}(\cdot) < \tau$, where τ is the sparsity pruning confidence threshold.

$$\mathbf{p} = P_{\text{s}}(\mathscr{T}; \mathbf{W}_P) \quad (3)$$

$$\mathscr{T}' = \text{SparsityPruning}(\mathscr{T}, \mathbf{p} < \tau) \quad (4)$$

4.2.2 Skip Connection and Sparse Tensor Addition The upsampled sparse tensor feature maps from the generative process have gone through extreme spatial

compression that allows neurons to see larger context, but have lost spatial resolution. To recover the fine details of the input, we create the skip connections to the cached feature map from the encoder [2, 3]. Since both the upsampled feature map and the lower layer feature map are all sparse tensors, we use sparse tensor addition. This process also expands the support to be the union of the supports of both sparse tensors.

4.3 Multi-scale Bounding Box Anchor Prediction

Every voxel after the sparsity pruning potentially contains bounding box anchors. Therefore, we make a direct prediction of the bounding box parameters for every layer of the pruned sparse tensors. Specifically, for each k anchor box, the network predicts 1 object anchor likelihood score, 6 offsets relative to the anchor box, and c semantic class scores. This results in $(c + 7)k$ outputs per voxel.

To capture as many shape variations, we use bounding box anchors with different aspect ratios. Specifically, for each anchor ratio seed a_r, we use all unique permutations of $\left[\sqrt{a_r}, \sqrt{a_r}, \frac{1}{\sqrt{a_r}}\right]$ as the aspect ratios of an anchor. In total, we use $k = 13$ anchors with $a_r \in \{1, 2, 4, \frac{1}{2}, \frac{1}{4}\}$ including the identity ratio.

However, even with these various anchor ratios, it is difficult to capture the extreme scale variation among 3D objects. Thus, we predict anchors at various stages of the decoder to capture the scale variation of 3D objects similar to Liu *et al.* [18]. We construct the anchors at each level to double the size of the anchors at the previous level.

4.4 Summary of GSDN Feed Forward

We summarize the feed forward pass of the generative sparse detection networks in Alg. 1. The algorithm generates L levels of hierarchical sparse tensor feature maps from the previous level feature maps on Line 3. Then, during the generative phase, we extract anchors and associated bounding box information (Line 8), predict sparsity and prune out voxels (Line 10), and apply transposed convolution (Line 12). We add the upsampled sparse tensor to the corresponding sparse tensor feature map from the encoder (Line 7).

4.5 Losses

The generative sparse detection network has to predict four types of outputs: sparsity prediction, anchor prediction, semantic class, and bounding box regression. First, the sparsity and anchor prediction are binary classification problems. However, the majority of the predictions are negative as many voxels does not contain positive anchors. Thus, we use balanced cross entropy loss:

$$L_{\mathrm{b}}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{2|\mathcal{P}|} \sum_{i \in \mathcal{P}} \log(P(\hat{\mathbf{y}}_i)) - \frac{1}{2|\mathcal{N}|} \sum_{i \in \mathcal{N}} \log(1 - P(\hat{\mathbf{y}}_i))$$

Algorithm 1: Generative Sparse Detection Networks

Input: $\mathscr{T}$, $f_l(\cdot;\mathbf{W}_l)$, $f_l^{\mathrm{Tr}}(\cdot;\mathbf{W}_l^{\mathrm{Tr}})$, $g_l^b(\cdot;\mathbf{W}_l^b)$, $P_{\mathrm{s}}(\cdot;\mathbf{G}_l^s)$ for $l \in [1,...,L]$, τ_s
Output: $\{\mathbf{B}_l\}_l$ for $l \in [1,...,L]$

$\mathscr{T}_0 \leftarrow \mathscr{T}$
/* Hierarchical Sparse Tensor Encoder § 4.1 */
for $l \leftarrow 1,...L$ **do**
  $\mathscr{T}_l \leftarrow f_l(\mathscr{T}_{l-1})$ // Hierarchical feature tensors
/* Generative Sparse Tensor Decoder § 4.2 */
$\mathscr{T}_L^{\mathrm{Tr}} \leftarrow \mathscr{T}_L$
for $l \leftarrow L,...,1$ **do**
  if $l < L$ **then**
    $\mathscr{T}_l^{\mathrm{Tr}} \leftarrow \mathscr{T}_l^{\mathrm{Tr}} + \mathscr{T}_l$ // Skip connection §4.2.2
  $\mathbf{B}_l \leftarrow g_l^b(\mathscr{T}_l^{\mathrm{Tr}})$ // Anchor predictions §4.3
  $\mathbf{p}_l \leftarrow P_l^{\mathrm{s}}(\mathscr{T}_l^{\mathrm{Tr}})$ // Sparsity predictions
  $\mathscr{T}_l^{\mathrm{Tr}} \leftarrow \mathrm{SparsityPruning}(\mathscr{T}_l^{\mathrm{Tr}}, \mathbf{p}_l < \tau)$ // Pruning §4.2.1
  if $l > 1$ **then**
    $\mathscr{T}_{l+1}^{\mathrm{Tr}} \leftarrow f_l^{\mathrm{Tr}}(\mathscr{T}_l^{\mathrm{Tr}})$ // Transposed convolution §4.2.1
return $\{\mathbf{B}_l\}_l$

where $\mathcal{P} = \{i|y_i = 1\}$ and $\mathcal{N} = \{i|y_i = 0\}$ are the set of indices with positive and negative labels respectively. We define an anchor to be positive if any of the anchors in a voxel overlaps with any ground-truth bounding boxes for 3D IoU > 0.35 and negative if 3D IoU < 0.2. As the sparsity prediction must contain all anchors in subsequent levels, we define a sparsity to be positive if any of the subsequent positive anchor associated to the current voxel is positive. We do not enforce loss on anchors that have $0.2 <$3D IoU < 0.35.

Finally, for positive anchors, we train semantic class prediction of the highest overlapping ground-truth bounding box class with the standard cross entropy, L_{class}, and bounding box center and size regression parameterized by difference of the center location relative to the size of the anchor and the log difference of the size of the bounding box with the Huber loss [27], L_{reg}. The final loss is the weighted sum of all losses:

$$L = \lambda_{\mathrm{s}} L_{\mathrm{s}} + \lambda_{\mathrm{anc}} L_{\mathrm{anc}} + \lambda_{\mathrm{class}} L_{\mathrm{class}} + \lambda_{\mathrm{reg}} L_{\mathrm{reg}}$$

where we use $\lambda_{\mathrm{s}} = 1$, $\lambda_{\mathrm{anc}} = 1$, $\lambda_{\mathrm{class}} = 1$, $\lambda_{\mathrm{reg}} = 0.1$ for all of our experiments.

4.6 Prediction post-processing

We train the network to overestimate the number of bounding box anchors as we label all anchors with 3D IoU >0.35 as positives. We filter out overlapping predictions with non-maximum suppression and merge them by computing score-weighted average of all removed bounding boxes to fine tune the final predictions similar to Redmon *et al.* [26].

Method	Single Shot	mAP@0.25	mAP@0.5
DSS [28, 13]	✗	15.2	6.8
MRCNN 2D-3D [11, 13]	✗	17.3	10.5
F-PointNet [25]	✗	19.8	10.8
GSPN [36, 24]	✗	30.6	17.7
3D-SIS [13]	✓	25.4	14.6
3D-SIS [13] + 5 views	✓	40.2	22.5
VoteNet [24]	✗	58.6	33.5
GSDN (Ours)	✓	**62.8**	**34.8**

Table 1: Object detection mAP on the ScanNet v2 validation set. DSS, MRCNN 2D-3D, FPointNet are from [13]. GSPN from [24]. Our method, despite being single-shot, outperforms all previous state-of-the-art methods.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	mAP
Hou *et al.* [13]	12.75	63.14	65.98	46.33	26.91	7.95	2.79	2.30	0.00	6.92	33.34	2.47	10.42	12.17	74.51	22.87	58.66	7.05	25.36
Hou *et al.* [13] + 5 views	19.76	69.71	66.15	71.81	36.06	30.64	10.88	27.34	0.00	10.00	46.93	14.06	**53.76**	35.96	87.60	42.98	84.30	16.20	40.23
Qi *et al.* [24]	36.27	**87.92**	88.71	**89.62**	58.77	**47.32**	38.10	44.62	7.83	56.13	**71.69**	47.23	45.37	57.13	94.94	54.70	92.11	37.20	58.65
GSDN (Ours)	**41.58**	82.50	**92.14**	86.95	**61.05**	42.41	**40.66**	**51.14**	**10.23**	**64.18**	71.06	**54.92**	40.00	**70.54**	**99.97**	**75.50**	**93.23**	**53.07**	**62.84**

Table 2: Class-wise mAP@0.25 object detection result on the ScanNet v2 validation set. Our method outperforms previous state-of-the-art on majority of the semantic classes.

5 Experiments

We evaluate our method on three 3D indoor datasets and compare with state-of-the-art object detection methods (5.1). We also make a detailed analysis of the speed and memory footprint of our method (5.2). Finally, we demonstrate the scalability of our proposed method on extremely large scenes (5.3).

Datasets. We evaluate our method on the ScanNet dataset [5], annotated 3D reconstructions of 1500 indoor scenes with instance labels of 18 semantic classes. We follow the experiment protocol of Qi *et al.* [24] to define axis-aligned bounding boxes that encloses all points of an instance without any margin as the ground truth bounding boxes.

The second dataset is the Stanford Large-Scale 3D Indoor Spaces (S3DIS) dataset [1]. It contains 3D scans of 6 buildings with 272 rooms, each with instance and semantic labels of 7 structural elements such as floor and ceiling, and five furniture classes. We train and evaluate our method on the official furniture split and use the most-widely used *Area 5* for our test split. We follow the same procedure as above to generate ground-truth bounding boxes from instance labels.

Finally, we demonstrate the scalability of GSDN on the Gibson environment [34] as it contains high-quality reconstructions of 575 multi-story buildings.

Metrics. We adopt the average precision (AP) and class-wise mean AP (mAP) to evaluate the performance of object detectors following the widely used convention of 2D object detection. We consider a detection as a positive match when a 3D intersection-over-union(IoU) between the prediction and the ground-truth bounding box is above a certain threshold.

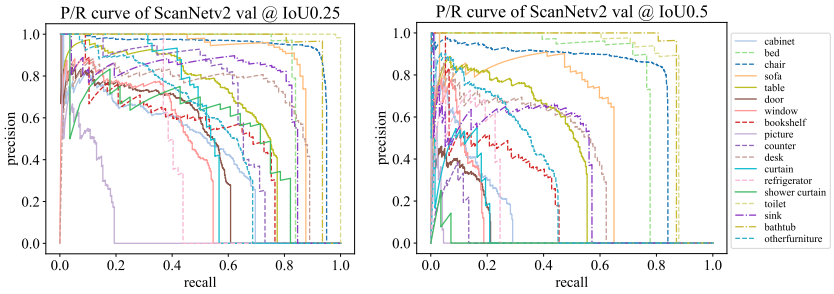

Fig. 5: Per-class precision/recall curve of ScanNetV2 validation object detection.

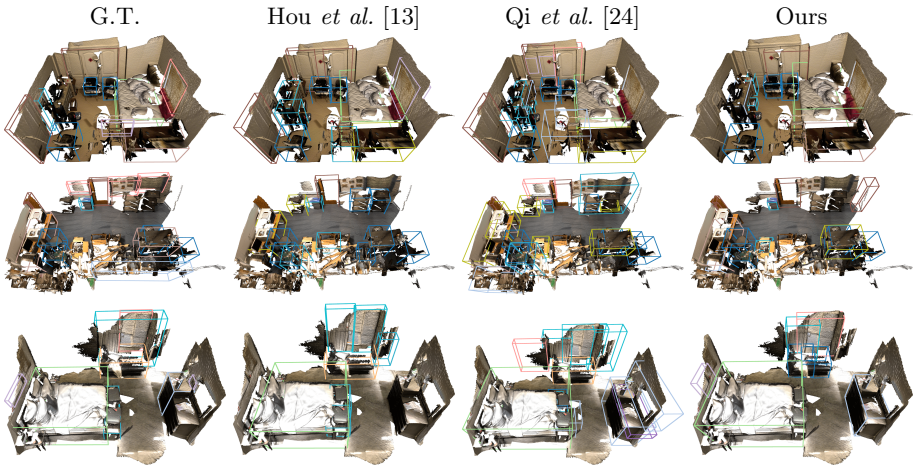

Fig. 6: Qualitative object detection results on the ScanNet dataset.

Training hyper-parameters. We train our models using SGD optimizer with exponential decay of learning rate from 0.1 to 1e-3 for 120k iterations with the batch size 16. As our model can process an entire scene fully-convolutionally, we do not make smaller crops of a scene. We use high-dimensional ResNet34 [2, 12] for the encoder. For all experiments, we use voxel size of 5cm, transpose kernel size of 3, with $L = 4$ scale hierarchy, sparsity pruning confidence $\tau = 0.3$, and 3D NMS threshold 0.2.

5.1 Object detection performance analysis

We compare the object detection performance of our proposed method with the previous state-of-the-art methods on Table 1 and Table 2. Our method, despite being a single-shot detector, outperforms all two-stage baselines with 4.2% mAP@0.25 and 1.3% mAP@0.5 performance gain and outperforms the state-of-the-art on the majority of semantic classes.

Fig. 7: Qualitative object detection results on the S3DIS dataset.

IoU Thres.	Metric	Method	table	chair	sofa	bookcase	board	avg
0.25	AP	Yang *et al.* [35]*	27.33	53.41	9.09	14.76	29.17	26.75
		GSDN (ours)	73.69	98.11	20.78	33.38	12.91	47.77
	Recall	Yang *et al.* [35]*	40.91	68.22	9.09	29.03	50.00	39.45
		GSDN (ours)	85.71	98.84	36.36	61.57	26.19	61.74
0.5	AP	Yang *et al.* [35]*	4.02	17.36	0.0	2.60	13.57	7.51
		GSDN (ours)	36.57	75.29	6.06	6.46	1.19	25.11
	Recall	Yang *et al.* [35]*	16.23	38.37	0.0	12.44	33.33	20.08
		GSDN (ours)	50.00	82.56	18.18	18.52	2.38	34.33

Table 3: Object detection result on furniture subclass of S3DIS dataset building 5. *: Converted the instance segmentation results to bounding boxes for reference

We also report the S3DIS detection results on Table 3. We compare the performance of our method against Yang *et al.* [35], a detection-based instance segmentation method, where we use the scene-level instance segmentation result as a proxy of the object detection the network learned at 1m×1m blocks. Our method in contrast to Yang *et al.* [35] takes the whole scene as an input, thus does not require slow pre-processing and post-processing, and is not limited by the cropped receptive field.

We plot class-wise precision-recall curves of ScanNet validation set on Figure 5. We found that some of the PR curves drop sharply, which indicates that the simple aspect-ratio anchors have a low recall.

Finally, we visualize qualitative results of our method on Figure 6 and Figure 7. In general, we found that our method suffers from detecting thin structures such as bookcase and board, which may be resolved by adding more extreme-shaped anchors. Please refer to the supplementary materials for the class-wise breakdown of mAP@0.5 on the ScanNet dataset and class-wise precision-recall curves for the S3DIS dataset.

5.2 Speed and Memory Analysis

We analyze the memory footprint and runtime in Figure 9 and Figure 8. For the memory analysis, we compare our method with the dense object detector [13] and measured the peak memory usage on ScanNetV2 validation set. As expected, our

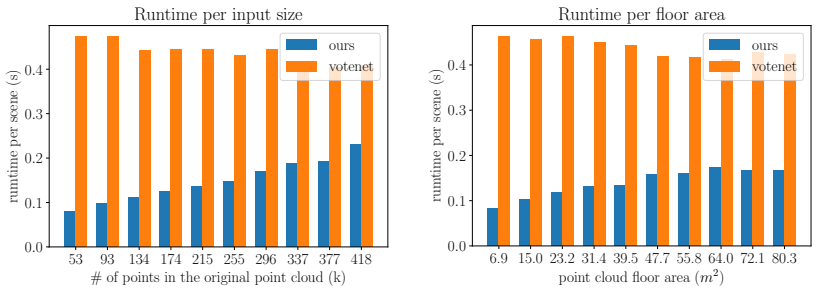

Fig. 8: Runtime comparison on ScanNet v2 validation set: Qi *et al.* [24] samples a constant number of points from a scene and their post-processing is inversely proportional to the density, whereas our method scales linearly to the number of points, and sublinearly to the floor area while being significantly faster.

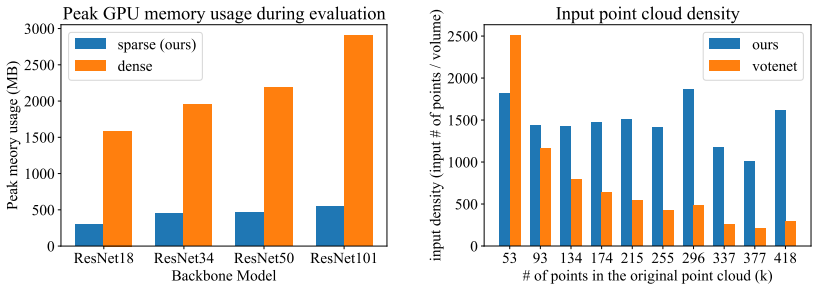

Fig. 9: Left: Memory usage comparison on ScanNet dataset evaluation: Our proposed sparse encoder and decoder maintains low memory usage compared to the dense counterparts. Right: Point cloud density on ScanNet dataset. Our model maintains constant input point cloud density compared to Qi *et al.* [24], which samples constant number of points regardless of the size of the input.

proposed network maintains extremely low memory consumption regardless of the depth of the network while that of the dense counterparts grows noticeably.

For runtime analysis, we compare the network feed forward and post-processing time of our method with Qi *et al.* [24] in Figure 8. On average, our method takes 0.12 seconds while Qi *et al.* [24] takes 0.45 seconds to process a scene of ScanNetV2 validation set. Moreover, the runtime of our method grows linearly to the number of points and sublinearly to the floor area of the point cloud, due to the sparsity of our point representation. Note that Qi *et al.* [24] subsamples a constant number of points from input point clouds regardless of the size of the input point clouds. Thus, the point density of Qi *et al.* [24] changes significantly as the point cloud gets larger. However, our method maintains the constant density as shown in Figure 9, which allows our method to scale to extremely

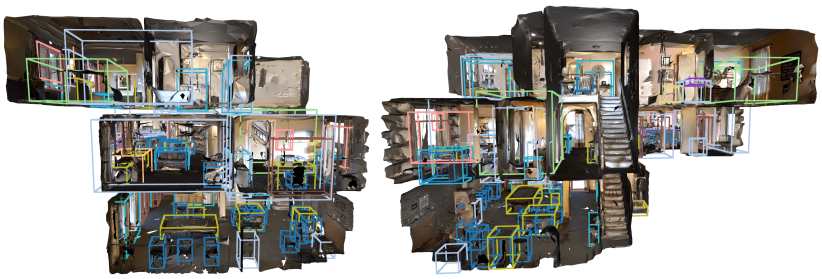

Fig. 10: Detection on a Gibson environment scene *Uvalda* [34]: GSDN can process a 17-room building with 1.4M points in a single *fully-convolutional* feed-forward.

large scenes as shown in Section 5.3. In sum, we achieve 3.78× speed up and 4.2% mAP@0.25 performance gain compared to Qi *et al.* [24] while maintaining the same point density from small to large scenes.

5.3 Scalability and generalization of GSDN on extremely large inputs

We qualitatively demonstrate the scalability and generalization ability of our method on large scenes from the S3DIS dataset [1] and the Gibson environment [34]. First, we process the entire building 5 of S3DIS which consists of 78M points, 13984m^3 volume, and 53 rooms. GSDN takes 20 seconds for a single feed-forward of the entire scene including data pre-processing and post-processing. The model uses 5G GPU memory to detect 573 instances of 3D objects, which we visualized on Figure 2.

Similarly, we train our network on ScanNet dataset [5] which only contain single-floor 3D scans. However, we tested the network on multi-story buildings. On Figure 10, we visualize our detection results on the scene named *Uvalda* from Gibson, which is a 3-story building with 173m^2 floor area. Note that our fully-convolutional network, which was only trained on single-story 3D scans, generalizes to multi-story buildings without any ad-hoc pre-processing or post-processing. GSDN takes 2.2 seconds to process the building from the raw point cloud and takes up 1.8G GPU memory to detect 129 instances of 3D objects.

6 Conclusion

In this work, we present the Generative Sparse Detection Network (GSDN) for single-shot fully-convolutional 3D object detection. GSDN maintains sparsity throughout the network by generating object centers using the proposed generative sparse tensor decoder. GSDN can efficiently process large-scale point clouds without cropping the scene into smaller windows to take advantage of the full receptive field. Thus, GSDN outperforms the previous state-of-the-art method by 4.2 mAP@0.25 while being 3.78× faster. In the follow-up work, we will examine and adopt various image detection techniques to boost the accuracy of GSDN.

References

1. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (2016)
2. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3075–3084 (2019)
3. Choy, C., Park, J., Koltun, V.: Fully convolutional geometric features. In: ICCV (2019)
4. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)
5. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5828–5839 (2017)
6. Dai, A., Diller, C., Nießner, M.: Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. arXiv preprint arXiv:1912.00036 (2019)
7. Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., Nießner, M.: Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2018)
8. Graham, B., Engelcke, M., van der Maaten, L.: 3D semantic segmentation with submanifold sparse convolutional networks. CVPR (2018)
9. Graham, B., van der Maaten, L.: Submanifold sparse convolutional networks. arXiv preprint arXiv:1706.01307 (2017)
10. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Hou, J., Dai, A., Nießner, M.: 3d-sis: 3d semantic instance segmentation of rgb-d scans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4421–4430 (2019)
14. Lahoud, J., Ghanem, B., Pollefeys, M., Oswald, M.R.: 3d instance segmentation via multi-task metric learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9256–9266 (2019)
15. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3d lidar using fully convolutional network. arXiv preprint arXiv:1608.07916 (2016)
16. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
17. Liu, C., Furukawa, Y.: Masc: multi-scale affinity with sparse convolution for 3d instance segmentation. arXiv preprint arXiv:1902.04478 (2019)
18. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)

19. Maturana, D., Scherer, S.: VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In: IROS (2015)
20. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2019)
21. Narang, S., Elsen, E., Diamos, G., Sengupta, S.: Exploring sparsity in recurrent neural networks. arXiv preprint arXiv:1704.05119 (2017)
22. Parashar, A., Rhu, M., Mukkara, A., Puglielli, A., Venkatesan, R., Khailany, B., Emer, J., Keckler, S.W., Dally, W.J.: Scnn: An accelerator for compressed-sparse convolutional neural networks. ACM SIGARCH Computer Architecture News **45**(2), 27–40 (2017)
23. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
24. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9277–9286 (2019)
25. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 918–927 (2018)
26. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7263–7271 (2017)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
28. Song, S., Xiao, J.: Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In: CVPR (2016)
29. Tange, O., et al.: Gnu parallel-the command-line power tool. The USENIX Magazine **36**(1), 42–47 (2011)
30. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In: IEEE International Conference on Computer Vision (ICCV) (2017), http://lmb.informatik.uni-freiburg.de/Publications/2017/TDB17b
31. Tchapmi, L.P., Kosaraju, V., Rezatofighi, S.H., Reid, I., Savarese, S.: Topnet: Structural point cloud decoder. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
32. Wang, D.Z., Posner, I.: Voting for voting in online point cloud object detection. In: Robotics: Science and Systems. vol. 1, pp. 10–15607 (2015)
33. Wang, W., Yu, R., Huang, Q., Neumann, U.: Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2569–2578 (2018)
34. Xia, F., R. Zamir, A., He, Z.Y., Sax, A., Malik, J., Savarese, S.: Gibson env: real-world perception for embodied agents. In: Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on. IEEE (2018)
35. Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N.: Learning object bounding boxes for 3d instance segmentation on point clouds. In: Advances in Neural Information Processing Systems. pp. 6737–6746 (2019)
36. Yi, L., Zhao, W., Wang, H., Sung, M., Guibas, L.J.: Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3947–3956 (2019)

37. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 3D Vision (3DV), 2018 International Conference on (2018)
38. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)