

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition and Intelligence

Contrastive Learning on 3D Scenes Using 3D and RGB Information

Chandan Yeshwanth





DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition and Intelligence

Contrastive Learning on 3D Scenes Using 3D and RGB Information

Kontrastives Lernen auf 3D-Szenen mit 3D- und RGB-Informationen

Author:	Chandan Yeshwanth
Supervisor:	Prof. Dr. Matthias Nießner
Advisor:	Prof. Dr. Matthias Nießner
Submission Date:	15.1.2022



I confirm that this Master's Thesis in Robotics, Cognition and Intelligence is my own work and I have documented all sources and material used.

Munich, 15.1.2022 Chandan Yeshwanth

Acknowledgments

I thank my advisor Prof. Matthias Nießner for his encouragement, advice and guidance during this work. His positive energy, hands-on involvement and willingness to spend time on our discussions enabled me to grow as a researcher.

I thank Christoph Weiler for his assistance with hardware at the Visual Computing Lab. I would also like to thank Prof. Angela Dai for her advice on implementing 3DMV, on which this work is based. Thank you to all the students of the Visual Computing Lab for our engaging conversations.

I thank my friends for their company during this work. Lastly, but most importantly, I thank my parents and family for their constant inspiration and confidence in me.

Abstract

Neural networks for semantic segmentation of 3D reconstructed indoor scenes are effective when trained from scratch on a large number of scenes. However, transfer learning with supervised pretraining cannot be used to further improve performance, especially on smaller datasets due to the lack of large-scale labeled datasets and expensive nature of vertex-level annotation. Contrastive learning is a promising technique to perform *unsupervised pretraining* and improve performance on a downstream task. This task creates a feature space where similar 3D locations have similar representations, and vice versa. Further, *multimodal learning* using information from the original RGB frames in addition to the 3D reconstruction has been shown to be superior to using either source by itself. In this work, we propose a *combination* of these techniques and show their effects to be complementary in improving downstream segmentation performance on the *ScanNet* dataset.

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**

2 Related Work **3**
2.1 2D Semantic Segmentation . . . 3
2.2 3D Semantic Segmentation . . . 3
2.3 Multimodal Learning with 3D and 2D . . . 4
2.4 Contrastive Learning . . . 5
2.4.1 Contrastive Learning in 2D . . . 5
2.4.2 Contrastive and Self-supervised Learning in 3D . . . 5
2.4.3 Contrastive Learning with 3D and 2D Data . . . 6

3 Method **8**
3.1 Dataset and Data Preparation . . . 8
3.1.1 RGB Images, Depth Images and Backprojection . . . 8
3.1.2 Occupancy Grid and Subvolumes . . . 8
3.2 Model . . . 10
3.2.1 2D Models . . . 10
3.2.2 3D Models . . . 11
3.2.3 Contrastive Learning . . . 12
3.3 Setup and Training Details . . . 13

4 Experiments and Results **15**
4.1 Metrics and Evaluation . . . 15
4.2 Quantitative Results . . . 15
4.2.1 2D Semantic Segmentation . . . 15
4.2.2 3D Semantic Segmentation with Full 2D Data . . . 15
4.2.3 3D Semantic Segmentation with Limited 2D Data . . . 18
4.3 Qualitative Results . . . 19
4.4 Training Time . . . 19

5 Limitations and Future Work **23**

6 Conclusion **24**

List of Figures **25**

List of Tables **26**

Acronyms **27**

Bibliography **28**

1 Introduction

Semantic segmentation of indoor scenes has been enabled by datasets such as ScanNet [1] and Matterport3D [2]. These datasets are built by capturing a sequence of 2-dimensional (2D) RGB and depth images of an indoor scene with handheld RGB-D sensors, and then reconstructing the 3-dimensional (3D) scene mesh and camera extrinsics of the sequence. The scene can then be represented as a mesh, voxel grid, or point cloud and the corresponding neural network architectures can be used for semantic understanding of the scene [3, 4, 5, 6]. These methods have proven to be effective in several tasks such as semantic segmentation [6], instance segmentation [7] and object detection [8].

However, these scene datasets are relatively small compared to RGB segmentation datasets [9, 10]. The transfer learning paradigm has been used successfully on 2D image tasks when data is scarce, by first pretraining Convolutional Neural Networks (CNNs) on large datasets such as ImageNet [11] and then finetuning on the target dataset and task. Pretraining is usually done in a supervised manner. The model learns a representation of the input data that is useful on the downstream task as well. However, this has not been possible with 3D data as it is time-consuming to obtain, and expensive to annotate at scale.

Another form of pretraining that does not require labels on the source dataset is self-supervised learning, where an unsupervised pretext task is performed by the network. One variant of this is contrastive learning: given a dataset with only the input samples x_i, a model minimizes the distances between the representations of *similar* pairs of samples (positives) (x_i, x_j), and maximizes the distances between representations of *dissimilar* pairs of samples (negatives) (x_i, x_k). The contrastive learning problem is then defined by the choices of positive and negative samples, choice of distance metric and formulation of the overall loss function combining positives and negatives [12]. Contrastive learning methods have obtained convincing improvements over other self-supervised baselines [13]. Contrastive and self-supervised learning have been applied to 3D data such as point clouds [14, 15] and voxels [16, 17]. Contrastive pretraining has also been shown to be particularly effective when the downstream dataset is small [17, 18, 19]. An example of such a dataset is the the Limited Reconstructions (LR) [17] setting of the ScanNet dataset, which was recently introduced and includes 4 subsets with 1%, 5%, 10% and 20% of the training data respectively.

In an orthogonal direction, previous research has shown that using multiple modalities is beneficial for the segmentation task. Particularly, 3D and 2D (RGB) data provide complementary signals [5] for semantic understanding. Correspondences between 3D locations in the mesh and the 2D locations in the image are established using the camera's intrinsic and extrinsic parameters.

Using additional RGB images to perform semantic segmentation of 3D data works well because RGB images of 3D scenes have a higher resolution than the 3D mesh, point cloud

or voxel grid. For example, when a Velodyne point cloud is projected to an RGB image, only 5.9% of the pixels are occupied while the rest is empty space [20]. Further, a single 3D location usually corresponds to multiple locations in a single 2D *view*, and can be covered by more than one such view. The 2D information from each of these non-overlapping locations is complementary, and can be aggregated for inference on the 3D location.

In this work, we combine the contrastive pretraining method with multimodal understanding, by contrasting representations of 3D locations with those of the corresponding 2D locations. Motivated by the easier availability of 2D labeled images, we use these to train 2D CNNs in a supervised manner. We then pretrain the 3D network using only the contrastive objective while keeping the 2D network fixed, and show that this is an effective initialization for the 3D network, particularly in the LR setting. We explore 2 settings: using full 2D data, and limited 2D data, as well as several design choices for pretraining and show the effect of these on the downstream task of semantic segmentation on ScanNet.

In summary, our contributions are

- we show that the combination of contrastive pretraining and multimodal learning improves upon using either of these individually when applied to semantic segmentation of 3D scenes
- we show the effectiveness of this method when using either full 2D or limited 2D data from ScanNet, along with ablation studies for relevant design choices

2 Related Work

2.1 2D Semantic Segmentation

Semantic segmentation of RGB images has been performed with Fully Convolutional Network (FCN) models such as U-Net [21], ENet [22] and DeepLab [23]. FCNs usually have a contracting path, in which the spatial resolution of the input is reduced through convolutions, while the number of features per location is increased, followed by an up-convolution or upsampling path that increases the spatial resolution of the features and reduces their feature dimensions. They also have skip connections between these two paths, in order to use high resolution input features to localize in the output.

2.2 3D Semantic Segmentation

3D convolutions are the 3-dimensional extension of 2D convolutions, and have been used for medical image segmentation [3, 4]. Similar to 2D FCN networks, 3D convolutions can be applied on 3D patches of the input scene represented as an occupancy grid or Truncated Signed Distance Field (TSDF). 3D convolutions have been used in ScanComplete [24] with an FCN architecture for scene completion, where training is on subvolumes and inference is on the full scene, and in ScanNet [1] and 3DMV [5], where training is on subvolumes and the network outputs only labels for the center column of the subvolume. OccuSeg [25] uses an additional occupancy signal at each voxel for instance segmentation, and optimizes feature and spatial embeddings with different objectives.

A drawback of 3D convolutions on occupancy grids is that grids obtained from real world data such as outdoor urban scenes and indoor scenes are mostly empty [20], and hence large consecutive subvolumes have a constant value. Applying convolutions over these subvolumes of the grid leads to redudant computation, and applying convolutions on the whole occupancy grid is prohibitively expensive. Minkowski networks [6] and Submanifold Sparse Convolution networks [26] eliminate these computations and perform only the necessary ones at active locations in the input, while supporting common operations such as pooling and transposed convolutions. Rather than determining the output locations of the convolution operator based on its kernel size, padding and stride, they allow the input and output to be at arbitrary 3D locations.

Another class of models such as PointNet [27] and PointNet++ [4] operates directly on point clouds. These methods have the advantage that they do not need to represent free space explicitly, and treat point clouds as unordered sets. PointNet transforms point features with a multi-layer perceptron and then aggregates them with max pooling, thus ensuring

permutation invariance to the input. However, no local information is used as is done in convolutions, which is overcome in PointNet++ by selecting centroids of local regions and then aggregating features within these regions.

2.3 Multimodal Learning with 3D and 2D

Our work is based on 3DMV [5], one of the first methods to use features from RGB images for 3D semantic segmentation. They treat the scene as an occupancy grid, and predict labels for the center columns of subvolumes. Features of RGB images are extracted using a pretrained ENet [22] and projected onto the corresponding voxels of the occupancy grid, which leads to a significant increase in segmentation accuracy. Additionally they show that simply training on the backprojected features from the 2D network without 3D information leads to a performance comparable to using only 3D geometry information.

A more recent work, Virtual Multiview Fusion [28] uses a similar approach, but samples 'virtual views' in the scene for better coverage, and with greater field of view, as well as generated additional channels of information from the 2D views. The class probabilities from these views are simply accumulated after backprojecting and used as the predictions on the 3D points.

Several other works have used this backprojection approach to combine 2D and 3D features: bidirectional and hierarchical networks to combine features at multiple levels [29], creation of 3D pseudo labels from projected 2D labels to train a 3D network [30], the Diffuser which accumulates 2D labels onto a 3D volume as a transductive label propagation problem [30], additionally fusing the depth images instead of using a given 3D mesh [31]. HMP3D [32] combines 2D and 3D voxel grid predictions using a Markov Random Field. Multi-view PointNet [20] obtains 2D image features on multiple views, and accumulates their 3D projections onto nearest-neighbors in the point cloud before applying a PointNet++ [4] model. Frey et al. [33] extend this method by projecting the created 3D volume back into 2D views and retraining the 2D segmentation network, enabling continual learning for robots. Chiang et al. [34] extend the backprojection method to point-based networks. PanopticFusion [35] extends this method to both semantic and instance segmentation, by performing these tasks in 2D and then projecting the results to 3D, while 3D-SIS [7] backprojects 2D features to 3D for instance segmentation. In addition, they train on parts of the scene and can perform inference on whole scenes. CoCoNets [36] create a continuous 3D field where 3D features can be interpolated at any point, by contrasting different 3D locations with each other.

Image2Point [37] uses RGB information differently, by first training on 2D images and then *inflating* (tiling) the network weights along a 3rd dimension and then finetuning.

Neural Semantic Fields (NeSF) [38] pretrains neural radiance fields on 2D images and samples them to obtain density grids, then maps the density grid to a semantic label grid, and finally renders this back to the 2D space. In this process they use only 2D supervision, but perform both 2D and 3D segmentation.

Voint Cloud [39] renders a point cloud to 2D images, obtains feature maps on these images and then backprojects the features to 3D space to obtain a set of features for each 3D point,

known as a 'Voint'. These are then transformed by convolution and aggregated by pooling to obtain per-point features.

In contrast to these methods which do not use the 2D features for pretraining, we use the 2D features to create an effective initialization for the 3D network in an unsupervised manner.

2.4 Contrastive Learning

2.4.1 Contrastive Learning in 2D

Contrastive Predictive Coding [40] first proposed the InfoNCE loss, which creates features that encode shared information between different parts of the input, while discarding information that is not relevant.

Contrastive learning on RGB images frequently uses augmentations to create positive pairs of images, while all other pairs are negatives. The contrastive loss aims to produce similar predictions or representations for augmented versions of the same input, and different representations for different inputs, thus making the model invariant to augmentations. SimCLR [13] uses a simple architecture that applies 2 different augmentations on each image to produce positives. SimCLRv2 [41] performs a similar unsupervised pretraining, and then distils the network into a smaller network that is used for task-aware semi-supervised finetuning. Bootstrap Your Own Latent (BYOL) [42] 'bootstraps' this process since the initial representations cannot reliably be treated as positives, by obtaining the target features from a target network, and the features of another augmented input from an online network. The online network is updated at each iteration, and the target network is a moving average of the online network. This approach avoids negative samples completely. YADIM [12] brings multiple contrastive learning methods under a single framework where a choice has to be made for these 5 components: augmentation, encoder, representation extraction or projection, similarity measure and loss function. CLIP [43] is closer to our work, as it contrasts 2 different modalities for pretraining: images and their associated text captions, and show improvements in zero-shot downstream tasks. XMC-GAN [44] uses a similar idea of contrasting images and captions as an additional loss term for image generation.

2.4.2 Contrastive and Self-supervised Learning in 3D

Contrastive learning on 3D data largely follows the methods used in 2D based on augmentation for positives, and sampling different locations for negatives. Additionally, we describe a few self-supervised methods, where supervision is from a pretext task on a part of the input, without any labels. PointContrast [14] generates 2 views of a point cloud in the same coordinate system by backprojecting RGBD sequences to 3D, applies augmentations and contrasts point features using the ground truth correspondences. P4Contrast [15] uses both 3D and 2D features, by concatenating the features of the individual modalities and then contrasting point-pixel pairs. It does not explicitly contrast 3D features with the corresponding 2D features. Contrastive Scene Contexts [17] partitions the 3D space uniformly based on distance and angle from selected anchor points, and performs contrastive

learning within each partition separately. The Multimodal Contrastive Prototype (M2CP) [45] similarly concatenates representations from different modalities, and creates a per-class 'prototype' in order to contrast representations of different classes. 'One Thing One Click' generates pseudo-labels over super-voxels, and contrasts these with per-class prototypes. Self-supervision has been used to learn semantic segmentation such that the network's output is also a spatial partitioning of the point cloud [46]. Self-supervised Geometric Perception (SGP) [47] alternately optimizes a teacher network that obtains pseudo labels on points given features, and a student network that learns features given pseudo labels. DepthContrast [48] constructs single-view point clouds and grids from RGBD images, and contrasts the global representations of these 2 modalities with each other. A similar approach has been used with active learning on super-points [49] while constraining the predicted labels within each super-point. The Semantic Query Network (SQN) [50] enforces consistency in labels of nearby points in a weakly supervised setting, by querying points based on their features. Self-Supervised Photometric Scene Generation (SPSG) [16] performs self-supervision on RGBD images by completing 3D scenes with a partial set of frames, rendering the scene in differentiable manner and applying a perceptual loss on the rendered views.

Other pretext tasks have been used for self-supervised and contrastive learning on point clouds, such as reconstruction from randomly rearranged parts [51], self-reconstruction and normal estimation [52], point cloud completion with occlusions [53]

Unlike these methods which only contrast 3D with 3D and 2D with 2D by augmentation and negative mining strategies, we contrast 3D locations with their corresponding 2D locations and use implicit negatives through the Noise Contrastive Estimation (NCE) loss.

2.4.3 Contrastive Learning with 3D and 2D Data

Several previous works use both 2D and 3D information along with self-supervised or contrastive learning to train 3D networks, but do not always contrast the 2D information with the corresponding 3D information. Pri3D [19] performs the opposite task as us; they use a pretrained 3D network to initialize a 2D network by contrasting corresponding points and pixels. In addition they contrast different positive and negative pixel pairs as well, and show improvements on 2D downstream tasks, particularly in the data-efficient regime.

The idea of pretraining one modality with another has been used earlier as distillation by Z. Liu et al. [54], where the 3D network is pretrained on a large 3D dataset, and then used to inform a 2D segmentation network, such that it can perform inference without the 3D data. Another stream of work contrasts pairs of (2D, 3D) samples by contrasting each modality separately, but with a joint loss. The TupleInfoNCE [55] formulation shows that either modality can easily be ignored while doing this, which necessitates high quality negative samples, while positives can be obtained easily by augmentation. Negatives are obtained by disturbing either modality, or by picking a sample at a different 3D location.

The work that is closest to ours is Learning from 2D [18]; they backproject pixel locations and features to obtain a single-view point cloud for each image, and then contrast the 3D point features with the 2D features. They show improvements in the low-data regime, but not in the ScanNet LR settings, while the full data setting does not benefit from 2D features. The

3D network uses sparse convolutions [6, 26], and features are upsampled to reach the original spatial resolution, since the backprojection. Their point-pixel loss formulation is based on the standard NCE loss. Our work differs from theirs in that we use a 3D occupancy grid obtained from multiple views instead of a single-view point cloud, and we directly project the 2D features to the original spatial resolution. In addition, our experiments focus on the contrastive pretraining phase rather than downstream tasks.

Another similar method is SimIPU [56], which contrasts point cloud and image features. Correspondences are obtained by projecting the point locations on the image plane and sampling 2D features. Additionally, it uses an intra-modal contrastive loss on the point features, and shows marginal improvements on point cloud datasets such as KITTI [57].

3 Method

3.1 Dataset and Data Preparation

We perform all experiments on the ScanNet v2 [1] dataset. It contains 1513 indoor scene meshes reconstructed from RGB-D trajectories, of which 1201 are in the training set and the remaining 312 are in the validation set. It has objects from 40 classes from the NYU40 [58] dataset. The ScanNet benchmark performs evaluation on a subset of 20 classes out of these 40 classes, and hence we report our results on the same subset of 20 classes.

For the data-efficient setting, we use the ScanNet LR dataset [17]. It includes 4 subsets of the training dataset, with 1%, 5%, 10% and 20% of the training scenes respectively, while the validation and test sets remain the same.

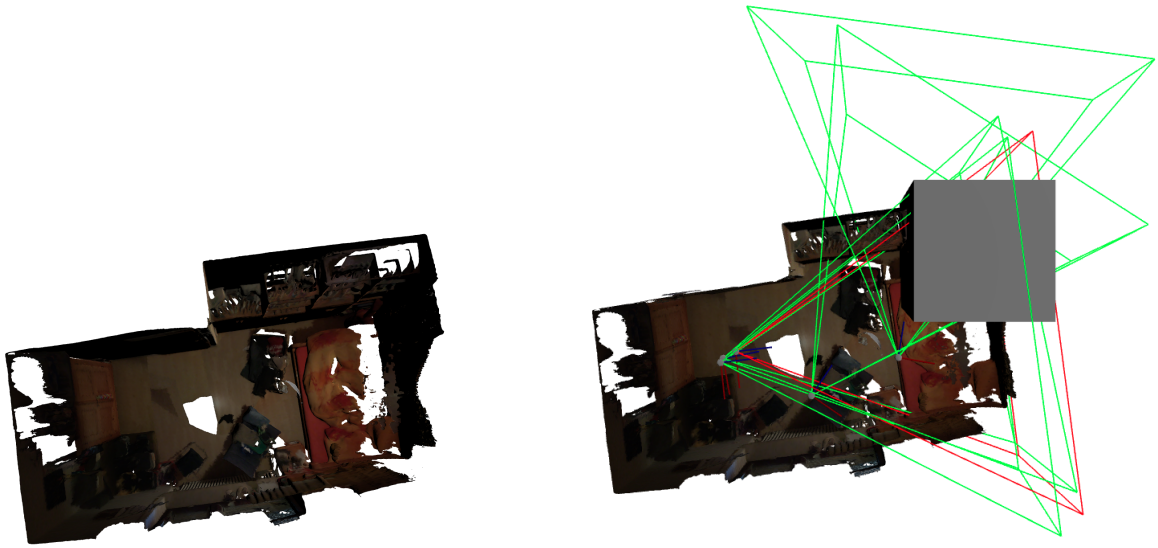
3.1.1 RGB Images, Depth Images and Backprojection

We use RGB images of size $(320, 240)$, and depth images of size $(40, 30)$, as done in 3DMV [5]. We use the camera intrinsics and extrinsics provided in the ScanNet dataset to backproject 2D coordinates to 3D. When training 2D models, we apply augmentations randomly using the Albumentations library [59]: flip left-right; jitter the color, bright, hue and saturation values; apply Gaussian blur and Gaussian noise; and then normalize the images with the statistics of images in ImageNet [11]. When training 2D models on the full ScanNet dataset, we use every 10^{th} frame from the training set of scenes, giving a total of 194,312 frames. We evaluate on the full ScanNet validation set of scenes, while using every 75^{th} frame giving a total of 7,354 frames.

3.1.2 Occupancy Grid and Subvolumes

We largely follow the approach of 3DMV [5] in preparing the dataset for dense convolutional models. We first prepare occupancy grids from the ScanNet 3D reconstructed meshes by voxelizing them using the Trimesh [60] 'subdivide' method: edges in the mesh are subdivided until they lie completely within a grid cell. We use a voxel size of $0.05m$, and prepare a binary occupancy grid where occupied voxels contain the value 1 and free space contains 0. Labels which are originally defined on the vertices of the mesh are assigned to voxels based on their nearest neighbors. This results in an input x and label y volume for each scene.

From these volumes, we sample 100 subvolumes per scene randomly for the training set. We discard subvolumes that have only structural elements: wall and floor, with 95% probability. For each sample, we start with a minimum required subvolume occupancy of 20% and attempt to find a valid subvolume 20 times, and then decrease the minimum



(a) Scene Mesh

(b) Subvolume and top-5 covering camera frustums. Red indicates the camera with maximum coverage

Figure 3.1: A single scene from ScanNet, along with a randomly sampled subvolume and the corresponding camera frustums

occupancy requirement iteratively by 1% till a minimum of 1%. We sample subvolumes of size $32 \times 32 \times 64$ voxels, with the height being twice the width and length such that the full height of the scene is captured in a single subvolume. One such subvolume is shown in Fig. 3.2.

Then, we iterate over all the camera poses in the scene and find the number of voxels in the subvolume that are covered by each pose. For computational reasons and due to poses with overlapping fields of view, we only consider every 10^{th} pose in the trajectory. The scene as seen from the camera is shown in Fig. 3.3.

The coverage of a subvolume by a camera pose is determined as follows. We first determine the X, Y and Z extents of the camera frustum, and pick the voxels that lie within these extents. The frustum is defined by a minimum depth of $0.4m$ and a maximum depth of $4m$. We then create local (x, y, z) grid coordinates for the individual voxel centers and map these to pixel coordinates using the world-to-grid transformation of the subvolume, and the camera extrinsic and intrinsic. A voxel is considered to be covered by a particular pixel, if its grid coordinates when mapped lie within the extents of the depth image, and the voxel's location is within a distance of 1 voxel from the backprojected depth image. This gives us an array of voxel indices that are covered by pixels, and the corresponding pixel indices that cover these voxels. Using this method, we ensure that every covered voxel is associated with a unique image pixel, while a particular pixel may cover 0, 1 or more voxels. An example of correspondences between the image and occupancy grid is shown in Fig. 3.5.

In this way, we compute the coverage of a subvolume by every camera pose, and greedily

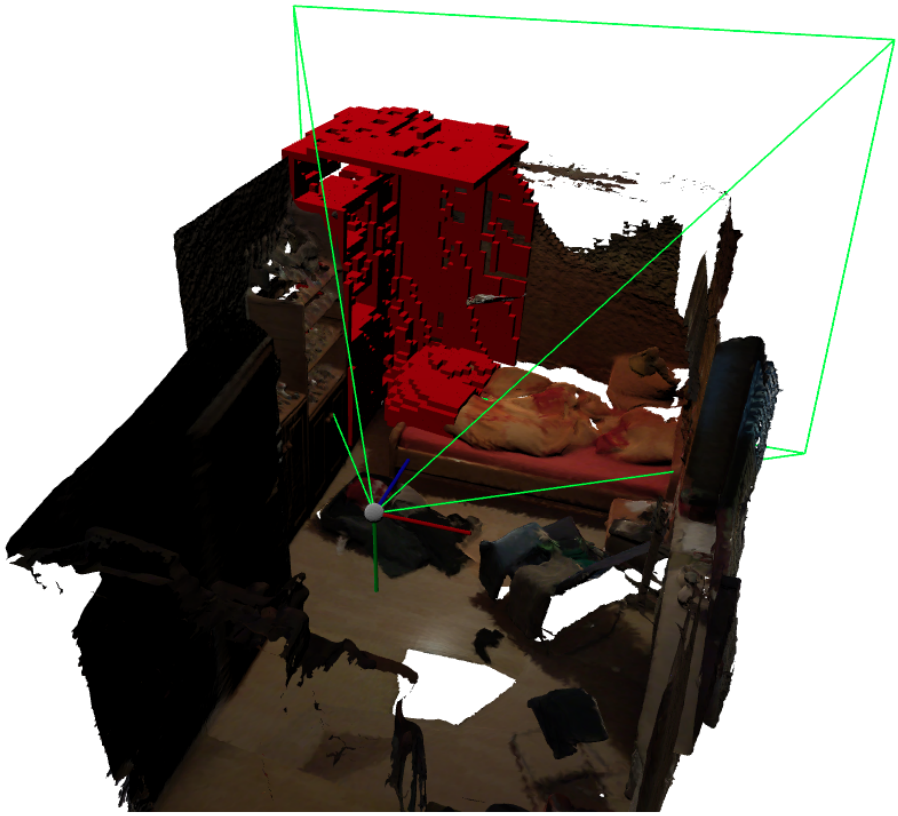


Figure 3.2: Voxels (red) belonging to a single subvolume overlaid on the scene mesh. The camera with the maximum coverage of the subvolume is shown

select the top 5 poses that add additional coverage to the subvolume such that we reduce redundancy in the chosen views. A single scene and subvolume are shown in Fig. 3.1.

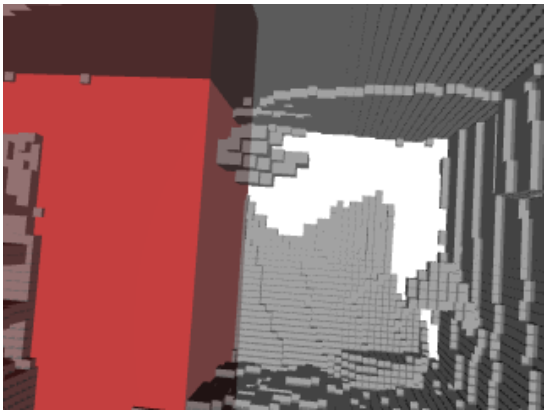
This results in a total of 120,100 subvolumes, which we augment by four 90° rotations each about the Z axis, to give a total of a 480,400 subvolumes. The validation set is prepared similarly, but without random sampling and augmentation. Instead, we pad the scene on the right, back and top ends to multiples of the subvolume size and create non-overlapping subvolumes in a sliding window manner across the XY plane, resulting in 4,302 subvolumes.

3.2 Model

3.2.1 2D Models

We perform experiments with two RGB semantic segmentation models: ENet [22] and DeepLabv3 [23]. The ENet is trained only on the ScanNet dataset with 40 classes. Intermediate features of dimension 128 are extracted from the *dilated3_7* layer.

We use DeepLabv3 with a ResNet50 backbone pretrained on the COCO 2017 dataset [9], and reduce the size of the classifier head from the original model by changing the internal



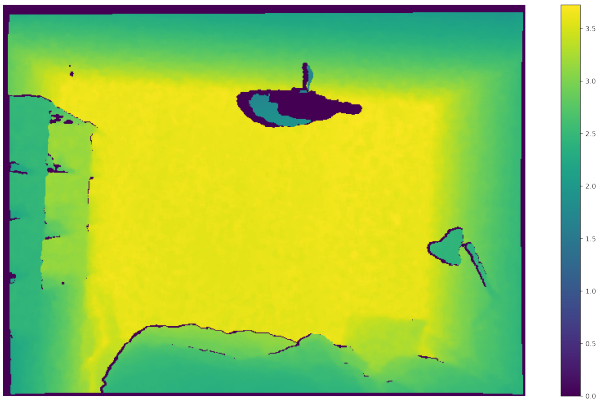
(a) Subvolume (red) as part of the occupancy grid of voxels



(b) Mesh as seen from the camera



(c) RGB image



(d) Depth image

Figure 3.3: Voxel grid, mesh, color and depth images as seen from a single view

feature dimension in the Atrous Spatial Pyramid Pooling and the last 2 convolution layers from 256 to 128. During training, we train only this classifier head and keep the rest of the model fixed. Intermediate features of dimension 128 are extracted after the classifer head, before interpolating to the original image dimension.

3.2.2 3D Models

Our 3D segmentation model uses an architecture similar to that of 3DMV [5]: we use a contracting path that reduces the spatial dimension by a factor of 2 in each residual block, while the expanding path does the opposite. The contracting and expanding paths each have 3 layers with 32, 64 and 128 channels respectively.

When 2D features are backprojected to 3D, they are inserted into a volume of the same dimension as the 3D input; $(32, 32, 64)$ with 128 channels and the spatial dimension is reduced by 2 residual blocks to a dimension of 32 and $1/4$ spatial resolution. 2D features are concatenated to the 3D ones at this $1/4$ spatial resolution after the 4th block in the 3D

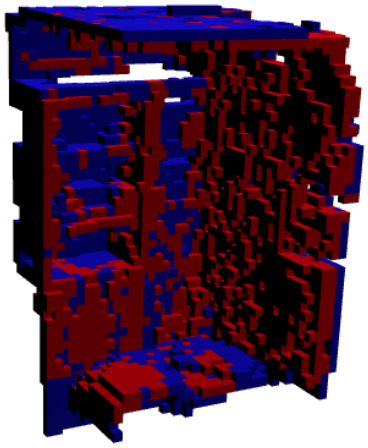


Figure 3.4: Occupied voxels (blue) and backprojected voxels (red)

network, as done in 3DMV, and used as input to the next layer. In the 3D-Only model, only the 3D input is used and 2D features are not concatenated.

The number of parameters in each model is listed in Table 3.1.

Model	ENet	DeepLabv3	3D-only	3D+2D
Parameters	354K	31.3M (22.8M trainable)	8.2M	9M

Table 3.1: 2D and 3D Model Sizes

3.2.3 Contrastive Learning

We apply the contrastive loss between features from the 2D model, and those from the 3D model. The loss is applied only at 3D locations which overlap with the backprojected 2D locations. We use the standard NCE formulation used in Learning from 2D [18]; given 2D features z_i^{2D} and 3D features z_i^{3D}, the contrastive loss is

$$\mathcal{L}_{\text{contrastive}} = -\sum_{i=1}^{N} \log \frac{exp(z_i^{3D} \cdot z_i^{2D}/\tau)}{\sum_{j=1}^{N} exp(z_i^{3D} \cdot z_j^{2D}/\tau)}$$

where τ is the temperature hyperparamter, and N corresponding 2D and 3D points are sampled randomly from each batch. We set $\tau = 0.04$ and $N = 8192$.

3D features are extracted from the last layer, and 2D features are taken from the backprojected volume. Both features are considered at their original spatial resolution. Pretraining is performed by optimizing only the NCE loss, without considering the semantic labels. The Cross Entropy (CE) loss is then used in the finetuning phase with semantic labels. The architecture of this model is shown in Figure 3.6.

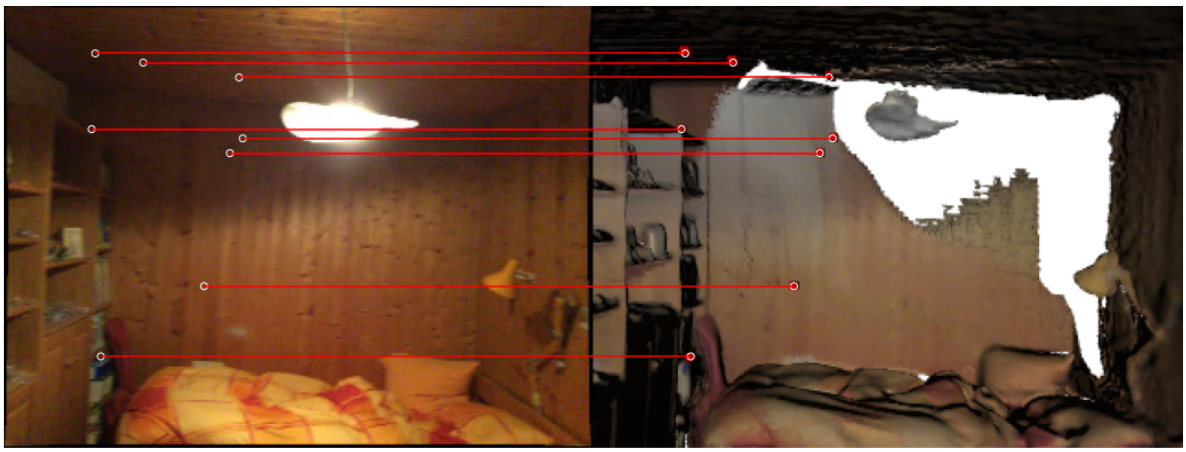


Figure 3.5: Correspondences between pixels in the RGB image and corresponding voxels in the mesh

3.3 Setup and Training Details

For all models, we optimize the CE loss with the Adam [61] optimizer and a constant learning rate of 10^{-4} without L2 regularization. We train with a maximum batch size of 12 (lower when using multiple views) and evaluate on the validation set every 2000 iterations, unless the training set has fewer batches as in the case of the LR1 setting. We perform model selection by early stopping; we stop training when the validation CE loss does not improve by more than 0.005 in 3 validation steps. When training with only contrastive loss, we stop training based on the validation contrastive loss. We train on a NVIDIA GeForce GTX 1070 Ti. We use class weights for the CE loss based on the inverse log frequency of classes as done in 3DMV [5].

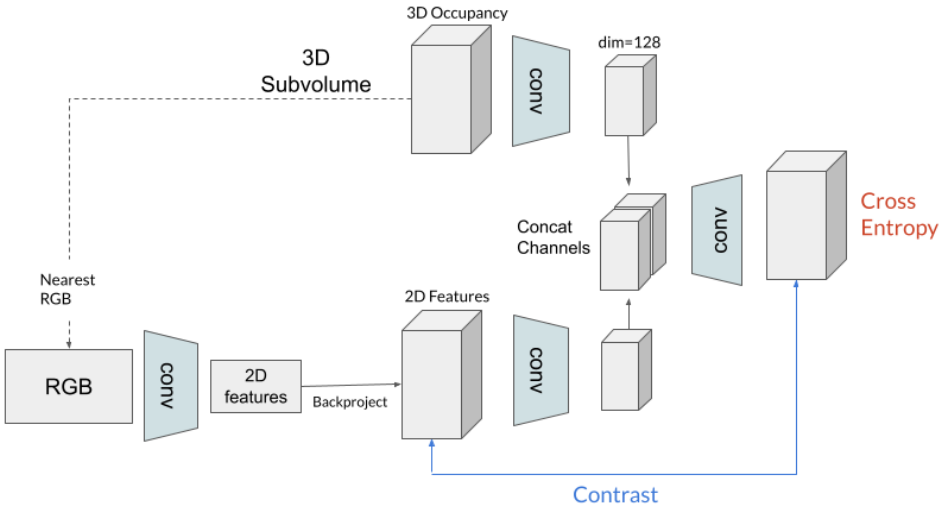


Figure 3.6: Architecture of 3D+2D model with contrastive and cross entropy losses. The 3D network is applied on the 3D subvolume, while the 2D network is applied on the nearest RGB image. Features from these networks are concatenated along the channel dimension in the 3D+2D model. The 3D input and backprojected 2D features have an initial resolution of 32x32x64 voxels, and are concatenated at a $\frac{1}{4}$ resolution. The contrastive loss (blue) is applied only during pretraining, while the cross entropy loss (red) is used only while finetuning. In the 3D-Only model, the concatenation of features is omitted and 2D features are used only for contrasting.

4 Experiments and Results

4.1 Metrics and Evaluation

We report the individual and mean Intersection over Union (IoU), and mean accuracy metrics for semantic segmentation across the 20 classes in the ScanNet benchmark, on the validation set [1]. In the case of 2D segmentation, these metrics are based on pixel-level labels and predictions. For 3D segmentation, these metrics are computed for voxel labeling on the occupancy grid described earlier.

Accuracy is the ratio of correctly classified voxels to the total number of voxels.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The IoU is computed as

$$iou = \frac{TP}{TP + FP + FN}$$

4.2 Quantitative Results

4.2.1 2D Semantic Segmentation

The performances of ENet and DeepLabv3 models trained on ScanNet images are given in Table 4.1. ENet is trained only on the full dataset, while DeepLabv3 is trained on the LR splits as well as the full dataset.

Table 4.1: 2D Semantic Segmentation Baselines on ScanNet

Dataset	Method	bath	bed	bksf	cab	chair	cntr	curt	desk	door	floor	othr	pic	ref	show	sink	sofa	tab	toil	wall	wind	mIoU	mAcc
Full	ENet	0.62	0.45	0.26	0.31	0.37	0.25	0.28	0.33	0.39	0.69	0.16	0.25	0.33	0.36	0.40	0.45	0.47	0.64	0.60	0.29	0.39	54.40
LR1	DeepLabv3	0.00	0.34	0.06	0.08	0 .33	0.02	0.22	0.05	0.20	0.63	0.03	0.12	0.05	0.00	0.00	0.17	0.31	0.00	0.50	0.08	0.16	25.16
LR5		0.37	0.46	0.08	0.34	0.45	0.18	0.25	0.26	0.27	0.72	0.15	0.19	0.30	0.13	0.24	0.39	0.45	0.58	0.59	0.32	0.34	48.32
LR10		0.48	0.47	0.28	0.36	0.49	0.30	0.40	0.31	0.37	0.76	0.16	0.28	0.37	0.44	0.33	0.40	0.50	0.60	0.59	0.31	0.41	55.94
Full		0.73	0.61	0.32	0.45	0.59	0.35	0.49	0.45	0.51	0.82	0.36	0.33	0.55	0.47	0.59	0.57	0.61	0.79	0.69	0.42	0.54	68.44

4.2.2 3D Semantic Segmentation with Full 2D Data

Results with contrastive pretraining on the downstream segmentation task applied to the 3D and 3D+2D models are shown in Table 4.2 as *+NCE*. Here, 3D+2D refers to the model where 2D features are concatenated to the 3D ones. Additionally we report 3 baselines that are obtained by projecting the Ground Truth (GT) labels, predictions of the ENet model and

predictions of the DeepLabv3 model. In all cases, the respective 2D model is trained on the full 2D dataset.

Table 4.2: 3D Segmentation using Full 2D Data and Contrastive Pretraining

Dataset	Method	bath	bed	bksf	cab	chair	cntr	curt	desk	door	floor	othr	pic	ref	show	sink	sofa	tab	toil	wall	wind	mIoU	mAcc
Full	Proj. ENet Preds	0.30	0.37	0.18	0.22	0.13	0.27	0.18	0.29	0.14	0.10	0.15	0.14	0.13	0.21	0.10	0.08	0.17	0.08	0.28	0.09	0.18	23.78
	Proj. DeepLabv3 Preds	0.31	0.39	0.24	0.26	0.20	0.32	0.23	0.34	0.19	0.16	0.16	0.18	0.19	0.33	0.16	0.11	0.24	0.18	0.34	0.18	0.23	28.63
	Proj. GT Labels	0.34	0.41	0.38	0.36	0.25	0.43	0.31	0.46	0.34	0.27	0.53	0.31	0.27	0.46	0.31	0.21	0.26	0.23	0.40	0.32	0.34	37.96
LR1	3D	0.00	0.04	0.06	0.01	0.29	0.01	0.04	0.01	0.01	0.73	0.01	0.00	0.00	0.00	0.00	0.02	0.24	0.00	0.42	0.00	0.10	13.66
	+ NCE (DeepLabv3)	0.00	0.19	0.15	**0.25**	**0.52**	**0.33**	0.14	**0.18**	0.08	0.71	**0.05**	0.04	0.13	0.00	0.07	0.27	**0.46**	0.00	0.54	0.02	0.21	30.88
	3D+2D (ENet)	0.00	0.15	0.15	0.05	0.29	0.12	0.03	0.03	0.02	0.75	0.02	0.00	0.02	0.00	0.00	0.00	0.29	0.00	0.48	0.00	0.12	18.11
	+ NCE (ENet)	0.00	0.31	0.16	0.17	0.46	0.17	0.23	0.09	0.23	**0.83**	0.02	**0.11**	0.06	0.00	0.01	0.37	0.41	0.00	0.58	0.00	0.21	33.90
	3D+2D (DeepLabv3)	0.00	0.30	0.11	0.14	0.40	0.11	0.33	0.06	0.26	0.62	0.02	0.09	0.10	0.00	0.00	0.26	0.25	0.00	0.54	0.06	0.18	27.59
	+ NCE (DeepLabv3)	0.00	**0.34**	**0.16**	0.20	0.49	0.24	**0.40**	0.16	**0.33**	0.81	0.02	0.10	**0.14**	0.00	**0.12**	**0.34**	0.43	0.00	**0.61**	**0.06**	**0.25**	**36.67**
LR5	3D	0.12	0.10	0.05	0.17	0.43	0.23	0.04	0.12	0.07	0.77	0.07	0.00	0.00	0.01	0.06	0.21	0.27	0.18	0.51	0.08	0.18	27.98
	+ NCE (DeepLabv3)	0.34	0.32	0.01	0.26	0.59	**0.30**	0.13	0.23	0.17	0.84	0.08	0.04	0.06	0.12	0.19	0.39	0.36	0.42	0.56	0.17	0.28	40.91
	3D+2D (ENet)	0.32	0.35	0.09	0.20	0.55	0.28	0.24	0.22	0.32	0.86	0.07	0.05	0.07	**0.13**	0.15	0.40	0.39	0.36	0.60	0.25	0.29	43.97
	+ NCE (ENet)	0.39	0.30	**0.20**	0.27	0.54	0.29	0.26	0.29	0.32	0.81	0.10	0.17	0.03	0.12	0.24	0.33	0.39	0.41	0.61	0.27	0.32	48.97
	3D+2D (DeepLabv3)	0.34	0.38	0.16	0.32	0.60	0.29	**0.46**	0.31	**0.38**	**0.88**	0.27	0.16	0.12	0.09	0.16	**0.47**	**0.42**	0.36	0.62	0.29	0.35	51.45
	+ NCE (DeepLabv3)	**0.45**	**0.46**	0.18	**0.32**	**0.62**	0.27	0.40	**0.32**	0.35	0.85	**0.28**	**0.18**	**0.15**	0.12	**0.25**	0.46	0.40	0.46	**0.63**	**0.34**	**0.37**	**52.41**
LR10	3D	0.33	0.13	0.17	0.27	0.58	0.31	0.11	0.17	0.12	0.88	0.06	0.04	0.09	0.16	0.18	0.34	0.38	0.50	0.52	0.10	0.27	40.88
	+ NCE (DeepLabv3)	0.42	0.26	0.30	0.30	0.63	0.39	0.08	0.28	0.15	0.91	0.09	0.03	0.12	0.18	0.25	0.36	0.47	0.52	0.58	0.16	0.32	46.16
	3D+2D (ENet)	0.42	0.37	0.31	0.33	0.63	0.36	0.33	0.30	0.32	0.90	0.11	0.19	0.10	0.19	0.32	0.47	0.44	0.50	0.62	0.27	0.37	53.46
	+ NCE (ENet)	0.49	0.32	0.24	0.30	0.63	0.31	0.35	0.28	0.35	0.88	0.15	**0.21**	0.12	0.20	0.33	0.45	0.44	0.54	0.62	0.27	0.38	53.85
	3D+2D (DeepLabv3)	0.52	0.44	0.34	0.34	0.67	**0.40**	0.49	0.33	**0.402**	0.90	0.30	0.16	**0.19**	**0.28**	0.29	0.51	0.53	**0.60**	0.64	0.34	0.43	60.24
	+ NCE (DeepLabv3)	**0.58**	**0.45**	**0.42**	**0.39**	**0.68**	**0.40**	**0.54**	**0.35**	0.38	**0.91**	**0.33**	0.19	0.17	0.26	**0.38**	**0.54**	**0.53**	0.57	**0.65**	**0.37**	**0.45**	**61.42**
Full	3D	0.51	0.41	0.33	0.34	0.68	0.37	0.24	0.34	0.19	0.91	0.17	0.06	0.26	0.31	0.38	0.50	0.49	0.67	0.61	0.20	0.40	53.65
	+ NCE (DeepLabv3)	0.56	0.46	0.35	0.36	0.69	0.38	0.28	0.38	0.20	0.92	0.17	0.04	0.31	0.27	0.44	0.50	0.54	0.70	0.62	0.25	0.42	57.13
	3D+2D (ENet)	0.59	0.53	0.39	0.40	0.73	0.43	0.42	0.44	0.42	0.93	0.30	0.20	0.32	0.32	0.52	0.56	0.60	**0.74**	0.69	0.36	0.49	65.26
	+ NCE (ENet)	0.63	0.54	0.38	0.39	0.73	0.41	0.41	0.40	0.41	0.93	0.29	0.20	0.26	0.29	0.49	0.58	0.57	0.72	0.66	0.38	0.48	64.83
	3D+2D (DeepLab)	0.62	**0.59**	**0.48**	0.42	0.74	0.41	0.57	0.44	0.43	0.93	**0.40**	**0.23**	0.34	**0.41**	0.50	0.60	0.59	0.70	0.68	**0.42**	0.52	**69.68**
	+ NCE (DeepLabv3)	**0.64**	0.56	0.47	**0.44**	**0.74**	**0.46**	**0.57**	**0.46**	**0.48**	**0.93**	0.39	0.20	**0.35**	0.37	**0.54**	**0.60**	**0.62**	0.68	**0.70**	0.41	**0.53**	69.09

In the LR5 and LR10 settings, both 3D and 3D+2D models outperform the baselines of backprojecting 2D predictions or ground truth labels from the validation set to 3D, showing that the 3D CNN applied on the occupancy grid is complementary to using RGB and depth information alone.

Next, we observe that contrastive pretraining consistently improves downstream performance over the 3D-only or 3D+2D baselines in the limited data settings. This improvement is most evident in the LR1 setting with 1% of 1201 scenes, i.e. 12 scenes available for training. When using ENet trained on full 2D data, this gives a 75% improvement in mean IoU, and 38% in the case of DeepLabv3. The relative improvement is lesser with more data; the LR10 setting with ENet is improved by 2% and with DeepLabv3, by 4%. Even in the full data setting, it improves performance by 1%. This indicates the trend that weaker baselines are improved the most by pretraining.

Importantly, we see that concatenating 2D features and using contrastive pretraining have complementary effects, since 3D+2D pretraining improves upon both concatenating and pretraining a 3D-only model. However, the performance gain from contrasting alone is smaller than from concatenating 2D features. Hence, simply concatenating 2D features in the limited 3D data setting is more effective than pretraining on a large unlabeled 3D dataset.

Further, we see that the performance of the pretrained 2D model impacts the downstream 3D performance: the DeepLabv3 model with higher 2D IoU leads to a higher 3D IoU both when concatenating as well as pretraining. Hence, the discriminative ability of features from

the 2D model is directly correlated with their effectiveness in the 3D setting.

Ablation Studies with ENet

We perform several ablation studies that are reported in Table 4.3. First, we finetune the pretrained ENet to different extents during the contrastive pretraining phase to see if this results in 2D features that are better suited to 3D learning, and compare with the pretrained ENet that is kept fixed during contrastive pretraining (*ENet fixed*). The different extents of ENet training are: training the ENet from *scratch* without any 2D supervision, finetuning the *full* ENet model, or only the *last layer*. Additionally, we report results of jointly training in a single stage with both CE and NCE losses in the LR1 setting. Here, the cross entropy loss is applied only on the LR part of the data, while the contrastive loss is applied on every training sample. The LR samples are upsampled such that every batch on average contains the same number of labeled and unlabeled samples. As an extension, the 3D+2D model is also jointly trained with an ENet from scratch with additional 2D supervision.

Table 4.3: Ablation Studies, 3D Segmentation using ENet and Full 2D Data

Dataset	Method	mIoU	mAcc
LR1	CE + NCE, 2D+3D	0.120	18.21
	+ ENet scratch	0.093	14.42
	3D	0.095	13.66
	3D+2D	0.102	17.73
	+ NCE, ENet from scratch	0.099	15.63
	+ NCE, Finetune full ENet	0.106	16.46
	+ NCE, Finetune ENet last layer	0.159	23.26
	+ NCE, ENet fixed	**0.211**	**33.89**
LR5	3D	0.175	27.97
	3D+2D	0.294	43.96
	+ NCE, ENet from scratch	0.160	25.37
	+ NCE, Finetune full ENet	0.178	29.41
	+ NCE, Finetune ENet last layer	0.260	37.87
	+ NCE, ENet fixed	**0.316**	**48.97**
LR10	3D	0.227	34.67
	3D+2D	0.373	53.46
	+ NCE, ENet from scratch	0.217	35.86
	+ NCE, Finetune full ENet	0.204	31.82
	+ NCE, Finetune ENet last layer	0.363	54.05
	+ NCE, ENet fixed	**0.377**	**53.76**

We observe that performance decreases as the extent of finetuning of the ENet model is increased; the best performance is obtained by keeping the 2D model fixed during pretraining.

When the whole 2D network is finetuned, performance is lower than that of the corresponding 3D+2D and 3D baselines. Joint learning with upsampling of the labeled data proves to be infeasible with an mIoU of 0.120 as compared to pretraining with an mIoU of 0.211.

4.2.3 3D Semantic Segmentation with Limited 2D Data

In the second setting, we assume that 2D data is limited and hence train 2D models on the corresponding LR split of the training set. These models are then used for contrastive pretraining and to concatenate 2D features with 3D features. Results are shown in Table 4.4.

Table 4.4: 3D segmentation with DeepLabv3 trained on Limited 2D Data and Contrastive Pretraining

Dataset	Method	bath	bed	bksf	cab	chair	cntr	curt	desk	door	floor	othr	pic	ref	show	sink	sofa	tab	toil	wall	wind	mIoU	mAcc
LR1	3D	0.00	0.04	0.06	0.01	0.29	0.01	0.04	0.01	0.01	0.73	0.01	0.00	0.00	0.00	0.00	0.02	0.24	0.00	0.42	0.00	0.10	13.66
	+ NCE	0.00	0.10	**0.14**	**0.14**	0.40	**0.18**	0.10	0.05	0.01	0.73	0.02	0.00	**0.08**	0.00	0.00	0.06	**0.38**	0.00	0.50	0.02	0.15	22.10
	3D+2D	0.00	0.11	0.14	0.05	0.35	0.03	0.12	0.04	0.05	0.73	0.01	0.01	0.03	0.00	0.00	0.08	0.29	0.00	0.49	0.00	0.13	21.26
	+ NCE	0.00	**0.27**	**0.15**	0.08	**0.40**	0.10	**0.24**	**0.06**	**0.16**	**0.83**	**0.03**	**0.12**	0.04	0.00	**0.01**	**0.14**	0.34	0.00	**0.53**	**0.03**	**0.18**	**26.93**
LR5	3D	0.12	0.10	0.05	0.17	0.43	0.23	0.04	0.12	0.07	0.77	0.07	0.00	0.00	0.01	0.06	0.21	0.27	0.18	0.51	0.08	0.18	27.98
	+ NCE	0.31	0.27	0.00	0.25	0.57	0.28	0.10	0.18	0.15	0.84	0.04	0.01	0.00	**0.08**	0.15	0.32	0.36	**0.42**	0.51	0.17	0.25	36.93
	3D+2D	0.31	0.29	0.18	0.27	0.48	0.26	0.27	0.20	0.24	0.82	0.08	0.06	0.00	0.06	0.13	0.27	0.34	0.19	0.58	0.27	0.26	40.31
	+ NCE	**0.38**	**0.37**	**0.25**	**0.27**	**0.57**	**0.28**	**0.30**	**0.28**	**0.28**	**0.86**	**0.13**	**0.15**	**0.13**	0.07	**0.18**	**0.35**	**0.44**	0.41	**0.61**	**0.31**	**0.33**	**49.31**
LR10	3D	0.33	0.13	0.17	0.27	0.58	0.31	0.11	0.17	0.12	0.88	0.06	0.04	0.09	0.16	0.18	0.34	0.38	0.50	0.52	0.10	0.27	40.88
	+ NCE	0.41	0.29	0.24	0.28	0.62	0.35	0.16	0.25	0.17	0.90	0.07	0.05	0.18	0.18	0.20	0.38	0.43	0.40	0.54	0.12	0.31	46.87
	3D+2D	0.49	0.39	0.34	0.31	0.63	0.28	**0.40**	0.30	0.31	0.89	0.14	0.14	0.18	0.26	0.25	0.39	0.46	0.51	0.61	0.29	0.38	55.56
	+ NCE	**0.55**	**0.42**	**0.36**	**0.32**	**0.63**	**0.38**	0.35	**0.32**	**0.32**	**0.91**	**0.15**	**0.17**	**0.18**	**0.30**	**0.27**	**0.41**	**0.50**	**0.59**	**0.62**	**0.29**	**0.40**	**57.15**

Contrastive pretraining consistently improves on baselines, with an increase of 38% mIoU and 5% mIoU in LR1 and LR10 3D+2D settings respectively; a smaller improvement in the LR1 setting and a larger improvement in the LR10 setting compared to experiments using full 2D data.

Design Choices

We evaluate these design choices in Table 4.5:

contrasting at 2 levels we use an additional contrastive loss term between 1/2 the input spatial resolution in the 3D network (second layer from the end), and the layer previous to the end of the classifier in the DeepLabv3 network.

additional negatives we use an additional 3D-3D loss term (*+3D-3D*) where the positive pair is as before, and every other 3D location is a negative, i.e

$$\mathcal{L}_{\text{contrastive}} = -\sum_{i=1}^{N} \log \frac{exp(z_i^{3D} \cdot z_i^{2D}/\tau)}{\sum_{j=1}^{N} exp(z_i^{3D} \cdot z_j^{2D}/\tau)} + \sum_{i=1}^{N} \log \frac{exp(z_i^{3D} \cdot z_i^{2D}/\tau)}{\sum_{j \neq i} exp(z_i^{3D} \cdot z_j^{3D}/\tau)}$$

and a combined formulation (*3D-3D, combined*) where the 3D-3D pairs are added as negatives to the existing loss

$$\mathcal{L}_{\text{contrastive}} = -\sum_{i=1}^{N} \log \frac{exp(z_i^{3D} \cdot z_i^{2D}/\tau)}{\sum_{j=1}^{N} exp(z_i^{3D} \cdot z_j^{2D}/\tau) + \sum_{j \neq i} exp(z_i^{3D} \cdot z_j^{3D}/\tau)}$$

using multiple views we add more views of the subvolumes while contrasting features as well as while concatenating. The features from multiple 2D locations corresponding to the same 3D location are pooled channel-wise as done in 3DMV, to obtain a single feature vector.

We can draw the following conclusions from this table:

- our simple method of 2-level contrasting adds extra constraints to the learning process and reduces performance rather than improving it.
- similarly, adding 3D-3D negatives either as a separate loss or in a combined loss enforces additional constraints that are not easily optimized in our current setup and hence reduces performance below the 3D+2D baseline, except in the LR10 split.
- pooling 2D features from 3 or 5 views improves performance slightly beyond the 1-view baseline, as shown in 3DMV. Using contrastive pretraining with these pooled features improves performance compared to the corresponding 3D+2D multiview baseline. However, the performance gain with 5 views is smaller than the gain with 3 views, leading to similar performances. We also experimented with contrasting the 3D features with individual 2D features before pooling, which resulted in a similar decrease in performance. Hence, effectively making use of multiview information for contrasting remains to be explored.

4.3 Qualitative Results

Predictions of our models on a single scene from the ScanNet validation set are shown in Figure 4.1. To obtain labels on the vertices of the original mesh, we quantize the vertices with the same voxel size of 0.05 while maintaing the mapping between original and quantized vertices. Then, for each quantized vertex we assign the label of the nearest voxel center from the prediction, and propagate these back to the original vertices. We see that the quality of segmentation progressively increases as the available data increases, and when contrastive pretraining is used.

4.4 Training Time

Approximate times required for training are provided in Table 4.6.

Table 4.5: Design Choices for 3D Segmentation using DeepLabv3 and Limited 2D Data

Dataset	Method	mIoU	mAcc
LR1	3D+2D	0.127	21.26
	+ NCE, 2 levels	0.165	23.70
	+ NCE + 3D-3D	0.157	25.17
	+ NCE + 3D-3D, combined	0.156	24.72
	+ NCE	**0.176**	**26.92**
	3D+2D, 3 views	0.130	20.80
	+ NCE	**0.180**	**27.84**
	3D+2D, 5 views	0.156	24.35
	+ NCE	**0.181**	**27.27**
LR5	3D+2D	0.264	40.30
	+ NCE, 2 levels	0.308	44.80
	+ NCE + 3D-3D	0.298	45.08
	+ NCE + 3D-3D, combined	0.298	45.32
	+ NCE	**0.330**	**49.30**
	3D+2D, 3 views	0.310	44.92
	+ NCE	**0.336**	**49.54**
	3D+2D, 5 views	0.314	46.64
	+ NCE	**0.333**	**51.62**
LR10	3D+2D	0.378	55.56
	+ NCE, 2 levels	0.357	52.88
	+ NCE + 3D-3D	**0.395**	**56.78**
	+ NCE + 3D-3D, combined	0.360	55.29
	+ NCE	0.379	53.44
	3D+2D, 3 views	0.401	58.95
	+ NCE	**0.408**	**59.60**
	3D+2D, 5 views	**0.420**	58.53
	+ NCE	0.416	**60.49**

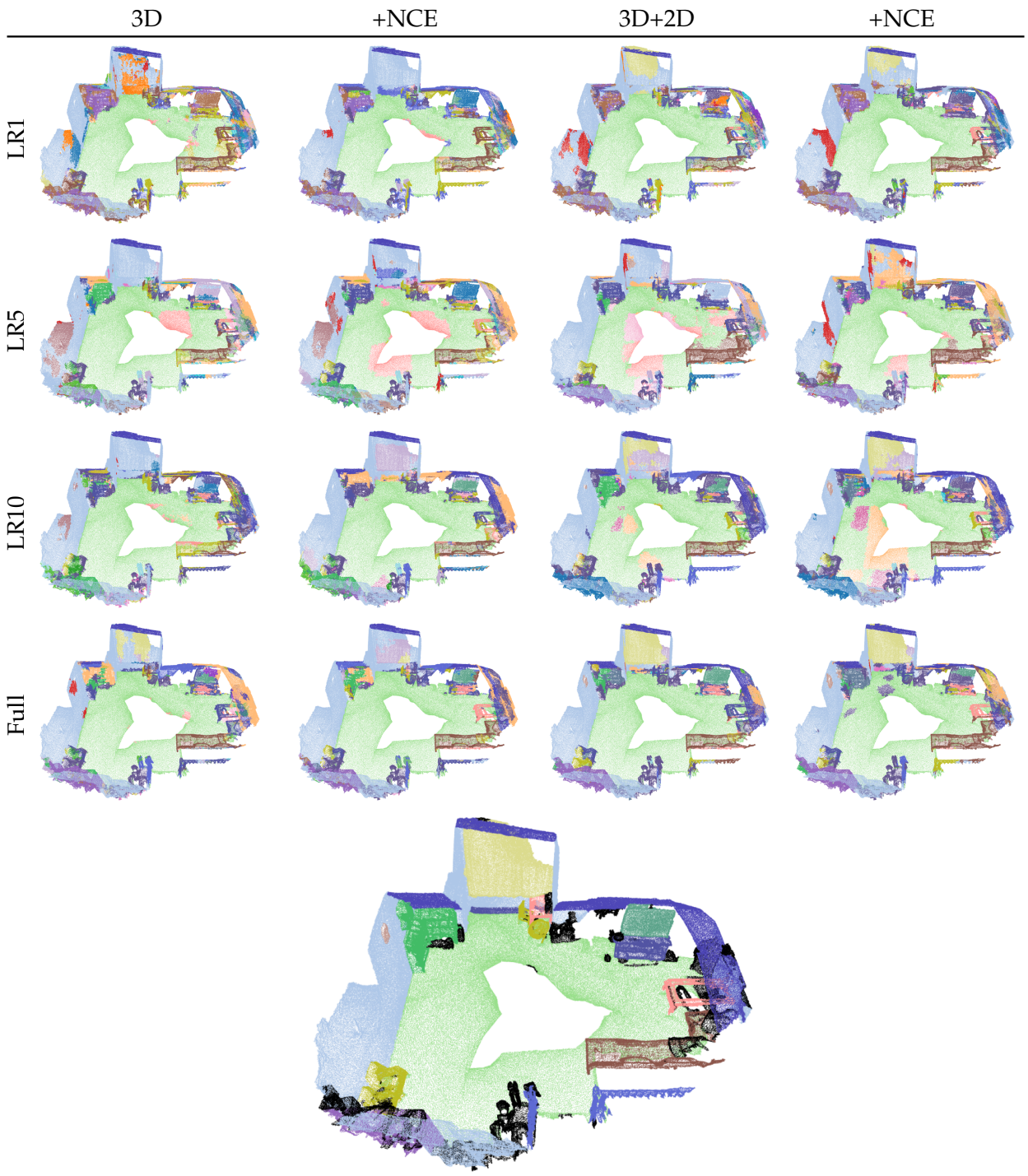


Figure 4.1: Predictions on a single scene using 3D, 3D+2D models and contrastive pretraining (top) trained on different splits of data (left). The ground truth labels are shown in the last row

Table 4.6: Training Times for Contrastive Pretraining, 3D and 3D+2D Models

Dataset	Method	Time (hours)
LR1	3D	0.75
	3D+2D	1
	3D+2D, 3 views	1.5
	3D+2D, 5 views	2
LR5	3D	2.5
	3D+2D	3
	3D+2D, 3 views	3.5
	3D+2D, 5 views	5.5
LR10	3D	5
	3D+2D	5
	3D+2D, 3 views	7.5
	3D+2D, 5 views	16.5
Full	3D, NCE Only	20
	3D+2D, NCE Only	18
	3D	11.5
	3D+2D	18

5 Limitations and Future Work

Our method operates only on subvolumes of the 3D scene. Future work can extend this method to sparse convolutions on full scenes and utilize more context in the scene, which requires a suitable strategy to select camera views for each 3D location, or generate virtual views [28]. Future work can also find better ways to contrast with features from multiple views without pooling them, giving better features in the 3D network.

Another interesting direction is examining the effect of different 2D models on contrastive learning and the downstream task. These 2D models could be pretrained for tasks other than semantic segmentation, or on other datasets.

Since our focus is in initializing the weights of the 3D model, we may not necessarily have 2D data in the downstream task. In this case, 2D features can be concatenated and contrasted during pretraining, and not concatenated during finetuning. This can be achieved by using zero features in the existing model, or by removing the corresponding channels and filters from the model.

In our experiments we see that using the contrastive and cross-entropy losses together for joint training reduces downstream performance, hence we are restricted to training in 3 phases: 2D model with labels, 3D model with contrastive loss, and then the 3D model with labels. Combining these 3 phases into 1 will reduce the complexity of experimentation, but this requires learning effective 2D features before they are used by the 3D network, possibly by bootstrapping [42]. At the same time, the 3D-2D contrastive loss could be combined with the 3D-3D and 2D-2D contrastive losses with other formulations.

6 Conclusion

We showed that contrasting 3D features with their corresponding 2D features is an effective pretraining method for 3D networks. This strategy particularly improves downstream segmentation performance by 5-75% when the training dataset is small, between 12 and 120 scenes. We also showed that pretraining has effects complementary to those of multimodal learning by fusing 2D and 3D features without pretraining, hence proving that the 3D network's initialization is crucial in the low-data regime. Separately, we demonstrated that the effectiveness of this method can be reliably improved by improving the 2D segmentation performance of the CNN, which is a relatively well-established task. Our experiments provide several directions for future work, which should lead to better semantic understanding of indoor scenes.

List of Figures

3.1 A single scene from ScanNet, along with a randomly sampled subvolume and the corresponding camera frustums 9
3.2 Voxels (red) belonging to a single subvolume overlaid on the scene mesh . . . 10
3.3 Voxel grid, mesh, color and depth images as seen from a single view 11
3.4 Occupied voxels (blue) and backprojected voxels (red) 12
3.5 Correspondences between pixels in the RGB image and corresponding voxels in the mesh 13
3.6 Architecture of 3D+2D model with contrastive and cross entropy losses 14

4.1 Predictions on a single scene using different models 21

List of Tables

3.1 2D and 3D Model Sizes 12

4.1 2D Semantic Segmentation Baselines on ScanNet 15
4.2 3D Segmentation using Full 2D Data and Contrastive Pretraining 16
4.3 Ablation Studies, 3D Segmentation using ENet and Full 2D Data 17
4.4 3D segmentation with DeepLabv3 trained on Limited 2D Data and Contrastive Pretraining 18
4.5 Design Choices for 3D Segmentation using DeepLabv3 and Limited 2D Data . 20
4.6 Training Times for Contrastive Pretraining, 3D and 3D+2D Models 22

Acronyms

CE Cross Entropy. 12, 13, 17

CNN Convolutional Neural Network. 1, 2

FCN Fully Convolutional Network. 3

GT Ground Truth. 15

IoU Intersection over Union. 15

LR Limited Reconstructions. 1, 2, 6, 8, 13, 15, 17, 18

NCE Noise Contrastive Estimation. 6, 7, 12, 17

TSDF Truncated Signed Distance Field. 3

Bibliography

[1] 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*. 2017, pp. 5828–5839.

[2] 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". In: *International Conference on 3D Vision (3DV)* (2017).

[3] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

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

[5] A. Dai and M. Nießner. "3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 452–468.

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

[7] J. Hou, A. Dai, and M. Nießner. "3d-sis: 3d semantic instance segmentation of rgb-d scans". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4421–4430.

[8] Z. Yang, Y. Sun, S. Liu, and J. Jia. "3dssd: Point-based 3d single stage object detector". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11040–11048.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[12] W. Falcon and K. Cho. "A framework for contrastive self-supervised learning and designing a new approach". In: *arXiv preprint arXiv:2009.00104* (2020).

[13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[14] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany. "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding". In: *European Conference on Computer Vision*. Springer. 2020, pp. 574–591.

[15] Y. Liu, L. Yi, S. Zhang, Q. Fan, T. Funkhouser, and H. Dong. "P4Contrast: Contrastive Learning with Pairs of Point-Pixel Pairs for RGB-D Scene Understanding". In: *arXiv preprint arXiv:2012.13089* (2020).

[16] A. Dai, Y. Siddiqui, J. Thies, J. Valentin, and M. Nießner. "SPSG: Self-Supervised Photometric Scene Generation from RGB-D Scans". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2021.

[17] J. Hou, B. Graham, M. Nießner, and S. Xie. "Exploring data-efficient 3d scene understanding with contrastive scene contexts". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15587–15597.

[18] Y.-C. Liu, Y.-K. Huang, H.-Y. Chiang, H.-T. Su, Z.-Y. Liu, C.-T. Chen, C.-Y. Tseng, and W. H. Hsu. "Learning from 2D: Pixel-to-Point Knowledge Transfer for 3D Pretraining". In: *arXiv preprint arXiv:2104.04687* (2021).

[19] J. Hou, S. Xie, B. Graham, A. Dai, and M. Nießner. "Pri3D: Can 3D Priors Help 2D Representation Learning?" In: *arXiv preprint arXiv:2104.11225* (2021).

[20] M. Jaritz, J. Gu, and H. Su. "Multi-view pointnet for 3d scene understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019.

[21] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[22] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. "Enet: A deep neural network architecture for real-time semantic segmentation". In: *arXiv preprint arXiv:1606.02147* (2016).

[23] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[24] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4578–4587.

[25] L. Han, T. Zheng, L. Xu, and L. Fang. "Occuseg: Occupancy-aware 3d instance segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2940–2949.

[26] B. Graham and L. van der Maaten. "Submanifold sparse convolutional networks". In: *arXiv preprint arXiv:1706.01307* (2017).

[27] C. R. Qi, H. Su, K. Mo, and L. 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*. 2017, pp. 652–660.

[28] A. Kundu, X. Yin, A. Fathi, D. Ross, B. Brewington, T. Funkhouser, and C. Pantofaru. "Virtual multi-view fusion for 3d semantic segmentation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 518–535.

[29] W. Hu, H. Zhao, L. Jiang, J. Jia, and T.-T. Wong. "Bidirectional Projection Network for Cross Dimension Scene Understanding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14373–14382.

[30] K. Genova, X. Yin, A. Kundu, C. Pantofaru, F. Cole, A. Sud, B. Brewington, B. Shucker, and T. Funkhouser. "Learning 3D Semantic Segmentation with only 2D Image Supervision". In: *arXiv preprint arXiv:2110.11325* (2021).

[31] D. Menini, S. Kumar, M. R. Oswald, E. Sandstrom, C. Sminchisescu, and L. Van Gool. "A real-time online learning framework for joint 3d reconstruction and semantic segmentation of indoor scenes". In: *arXiv preprint arXiv:2108.05246* (2021).

[32] K. Lai, L. Bo, and D. Fox. "Unsupervised feature learning for 3d scene labeling". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3050–3057.

[33] J. Frey, H. Blum, F. Milano, R. Siegwart, and C. Cadena. "Continual Learning of Semantic Segmentation using Complementary 2D-3D Data Representations". In: *arXiv preprint arXiv:2111.02156* (2021).

[34] H.-Y. Chiang, Y.-L. Lin, Y.-C. Liu, and W. H. Hsu. "A unified point-based framework for 3d segmentation". In: *2019 International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 155–163.

[35] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji. "Panopticfusion: Online volumetric semantic mapping at the level of stuff and things". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4205–4212.

[36] S. Lal, M. Prabhudesai, I. Mediratta, A. W. Harley, and K. Fragkiadaki. "CoCoNets: Continuous Contrastive 3D Scene Representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12487–12496.

[37] C. Xu, S. Yang, B. Zhai, B. Wu, X. Yue, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka. "Image2Point: 3D Point-Cloud Understanding with Pretrained 2D ConvNets". In: *arXiv preprint arXiv:2106.04180* (2021).

[38] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth. "NeSF: Neural Semantic Fields for Generalizable Semantic Segmentation of 3D Scenes". In: *arXiv preprint arXiv:2111.13260* (2021).

[39] A. Hamdi, S. Giancola, and B. Ghanem. "Voint Cloud: Multi-View Point Cloud Representation for 3D Understanding". In: *arXiv preprint arXiv:2111.15363* (2021).

[40] A. v. d. Oord, Y. Li, and O. Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[41] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. "Big Self-Supervised Models are Strong Semi-Supervised Learners". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/fcbc95ccdd551da181207c0c1400c655-Abstract.html.

[42] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu koray, R. Munos, and M. Valko. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 21271–21284. URL: https://proceedings.neurips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: http://proceedings.mlr.press/v139/radford21a.html.

[44] H. Zhang, J. Y. Koh, J. Baldridge, H. Lee, and Y. Yang. "Cross-modal contrastive learning for text-to-image generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 833–842.

[45] Z. Chen, L. Jing, Y. Liang, Y. Tian, and B. Li. "Multimodal Semi-Supervised Learning for 3D Objects". In: *arXiv preprint arXiv:2110.11601* (2021).

[46] B. Eckart, W. Yuan, C. Liu, and J. Kautz. "Self-Supervised Learning on 3D Point Clouds by Learning Discrete Generative Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8248–8257.

[47] H. Yang, W. Dong, L. Carlone, and V. Koltun. "Self-supervised Geometric Perception". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14350–14361.

[48] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra. "Self-supervised pretraining of 3d features on any point-cloud". In: *arXiv preprint arXiv:2101.02691* (2021).

[49] X. Shi, X. Xu, K. Chen, L. Cai, C. S. Foo, and K. Jia. "Label-efficient point cloud semantic segmentation: An active learning approach". In: *arXiv preprint arXiv:2101.06931* (2021).

[50] Q. Hu, B. Yang, G. Fang, Y. Guo, A. Leonardis, N. Trigoni, and A. Markham. "SQN: Weakly-Supervised Semantic Segmentation of Large-Scale 3D Point Clouds with 1000x Fewer Labels". In: *arXiv preprint arXiv:2104.04891* (2021).

[51] J. Sauder and B. Sievers. "Self-supervised deep learning on point clouds by reconstructing space". In: *arXiv preprint arXiv:1901.08396* (2019).

[52] Y. Rao, J. Lu, and J. Zhou. "Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5376–5385.

[53] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner. "Unsupervised Point Cloud Pre-Training via Occlusion Completion". In: *International Conference on Computer Vision, ICCV*. 2021.

[54] Z. Liu, X. Qi, and C.-W. Fu. "3d-to-2d distillation for indoor scene parsing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4464–4474.

[55] Y. Liu, Q. Fan, S. Zhang, H. Dong, T. Funkhouser, and L. Yi. "Contrastive Multimodal Fusion with TupleInfoNCE". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 754–763.

[56] Z. Li, Z. Chen, A. Li, L. Fang, Q. Jiang, X. Liu, J. Jiang, B. Zhou, and H. Zhao. "SimIPU: Simple 2D Image and 3D Point Cloud Unsupervised Pre-Training for Spatial-Aware Visual Representations". In: *arXiv preprint arXiv:2112.04680* (2021).

[57] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[58] P. K. Nathan Silberman Derek Hoiem and R. Fergus. "Indoor Segmentation and Support Inference from RGBD Images". In: *ECCV*. 2012.

[59] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[60] Dawson-Haggerty et al. *trimesh*. Version 3.2.0. Dec. 2019. URL: https://trimsh.org/.

[61] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.