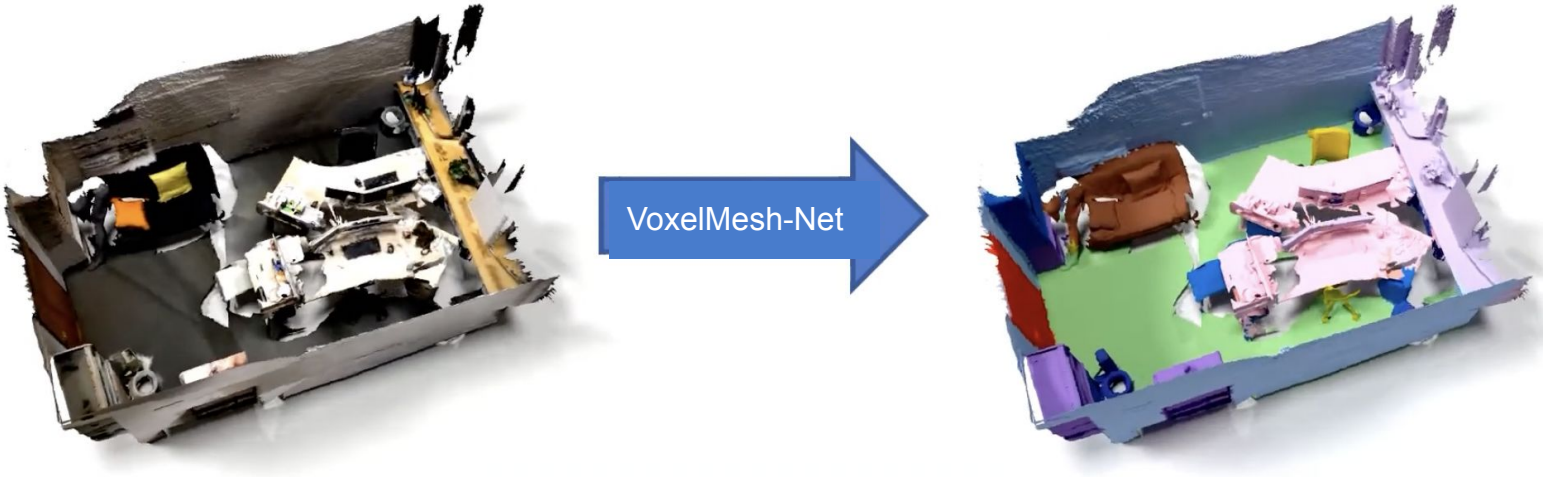
VMNet: Voxel-Mesh Network for Geodesic-Aware 3D Semantic Segmentation

https://arxiv.org/pdf/2107.13824.pdf

Abstract

VoxelMesh-ConvNet over 3d meshes which combine geodesic and euclidean information from convolutions performed on actual 3d meshe and its corresponding voxel representation respectively.

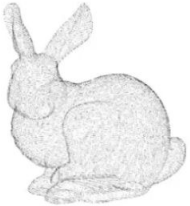
They perform semantic segmentation over 3d scene from scannet dataset.


VoxelMesh-Net

3D data representation

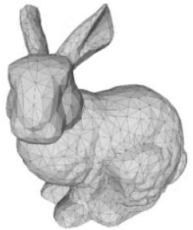
Voxelized

discrete Euclidean neighborhood



Point Cloud

continuous Euclidean neighborhood



Mesh

adjacent mesh neighbors

voxelization process omits all mesh edges and only retains Euclidean positions of mesh vertices.

Point cloud in essence are only points embedded in 3d euclidean space.

Mesh are those point cloud (extracted using lidar, etc) and then connected to form several triangles to represent the surfaces.

Euclidean Information

voxel- based methods suffer from ambiguous features on spatially close objects and struggle with handling complex and irregular geometries due to the lack of geodesic information.

Compared to Point Cloud-

1. they lose resolution compared to point clouds, since several distinct points representing intricate structures will be binned into one voxel if they're close together.
2. voxel grids can lead to unnecessarily high memory usage compared to point clouds in sparse environments, since they actively consume memory to represent free and unknown space whereas point clouds contain only known points.

Voxels

Imperfect approximation of an underlying 3D structure, hence useful in mutli-view problems where earlier 2d images where used. Can’t make enough sense to use it.

Intention of using Voxels, as I understood it

Being different from DCM-Net. No obvious reason was stated in the paper.

Though they show better performance by using voxels but that can be attributed to the use some attention modules which work on both domains of input (voxels and mesh).

Why both voxels and meshes

1. CNN generate similar features for voxels that are close in the Euclidean domain, even though these voxels may belong to different objects and are distant in the geodesic domain.

2. Without the geodesic information about shape surfaces, these Euclidean convolutions may struggle with learning specific object shapes. As

Novel Idea

1. Aggregate intra-domain features
2. Fuse inter-domain features.

Hence two key components of VMNet:

- Intra- domain Attentive Aggregation Module (work on only on mesh)
- Inter-domain Attentive Fusion Module. (work on fusing voxel and mesh features)

Also adding attention to the network was also novel, earlier work like DCM-Net didn’t use that.

VM-Net Arch - Like U-Net

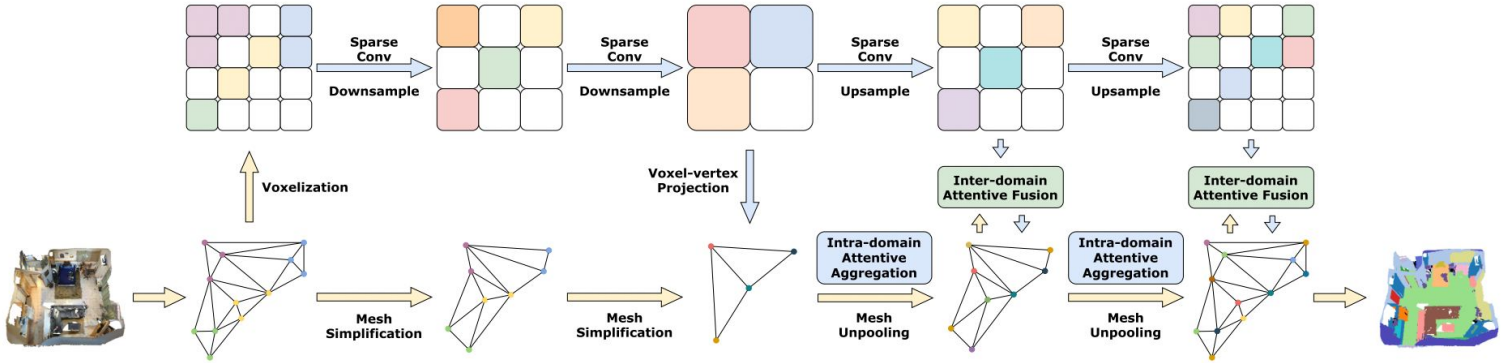


Figure 3. **Overview of Voxel-Mesh Network (VMNet).** Taking a colored mesh as input, we first rasterize it and apply voxel-based sparse convolutions to extract contextual information in the Euclidean domain. These features are then projected from voxels to vertices, and are further aggregated and fused in the geodesic domain producing distinctive per-vertex features. For simplicity, skip connections between the encoder and decoder are neglected here and only three levels of hierarchical voxel downsampling and mesh simplification are shown. The detailed network structure can be found in **Supplementary Section A**.

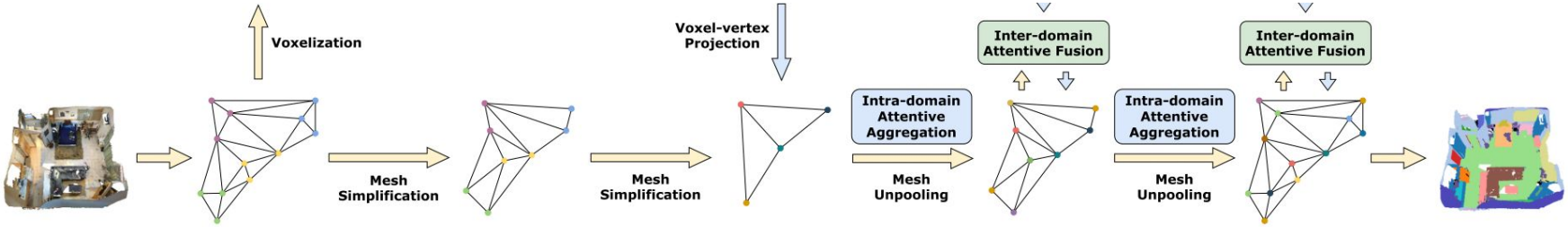
Input to the model

Taking a mesh as input, the colored vertices are first voxelized and then fed to the Euclidean branch.

Euclidean Branch uses sparse voxel-based convolutions.

MLP in Attention layers are Graph Convs.

Geodesic branch



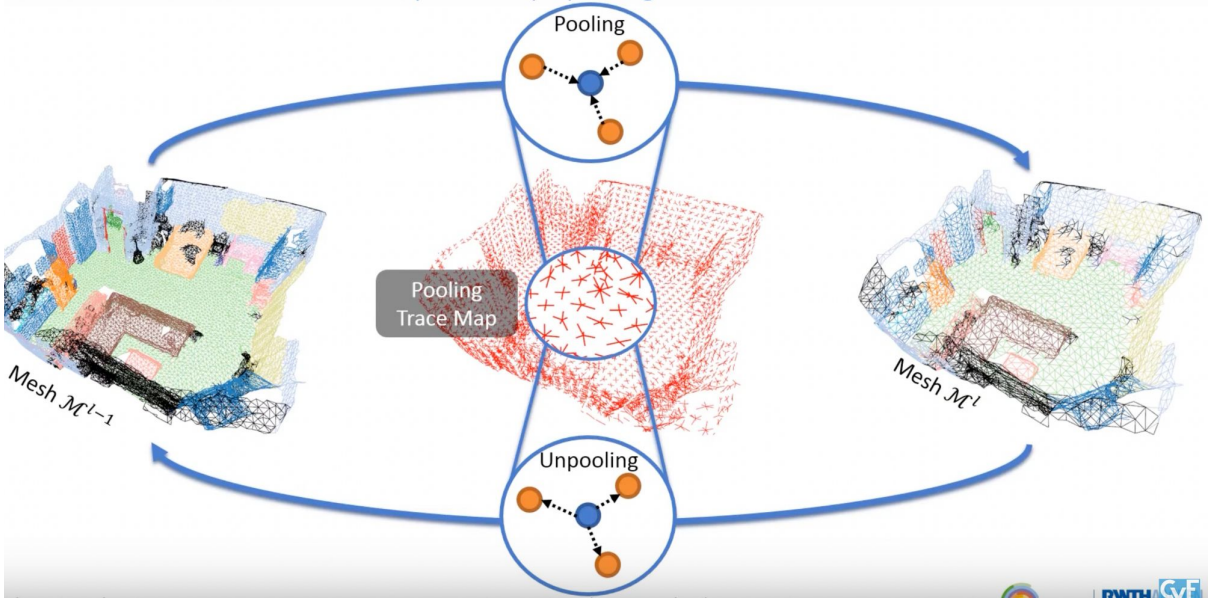
prepare hierarchy of simplified meshes (M0, ...,Ml, ...,ML)

each level of simplified mesh M_l corresponds to a the simplified mesh. They are saved for unpooling downsampling level of sparse voxels Sl. Trace maps of the simplified mesh are saved for unpooling operations between mesh levels.

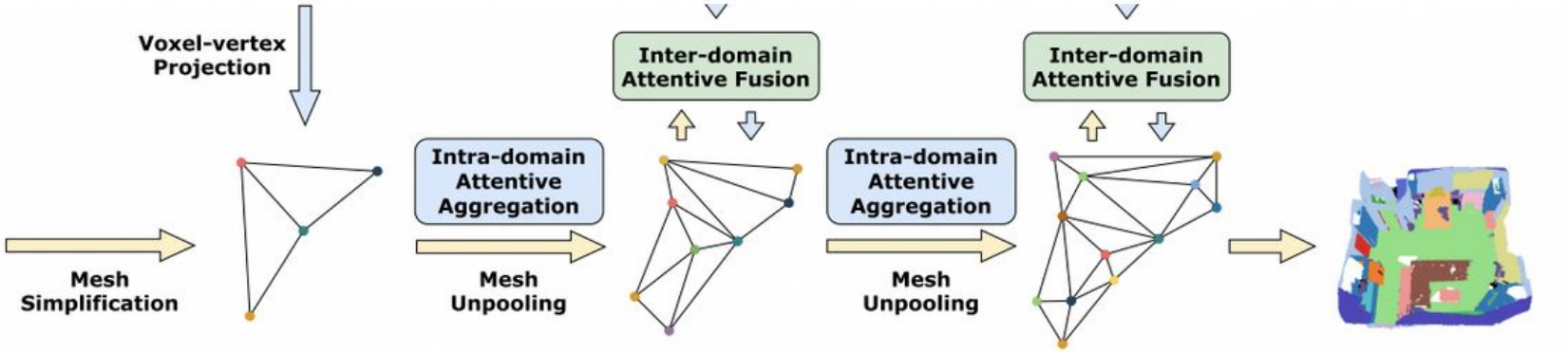
Mesh Simplification methods used: Vertex Clustering (VC) and Quadric Error Metrics (QEM)

Trace maps

It defines which vertices are collapsed together. It is a permutation invariant function. Mesh structure is preserved in all pooling levels using these maps. And when unpooling these maps helps in developing accurate feature maps over mesh.



At the first level of the decoding process (level L), the features are projected from intra-domain attentive aggregation. The resulting geodesic features of ML are unpooled to the next level ML−1.

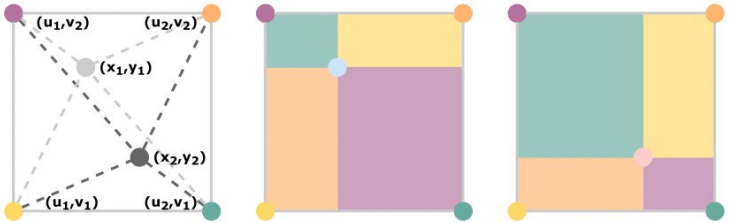


Voxel-vertex projection

Aggregation of euclidean features present in layer l. Now transform the features of voxels Sl back to vertices Ml for further processing in the geodesic domain.

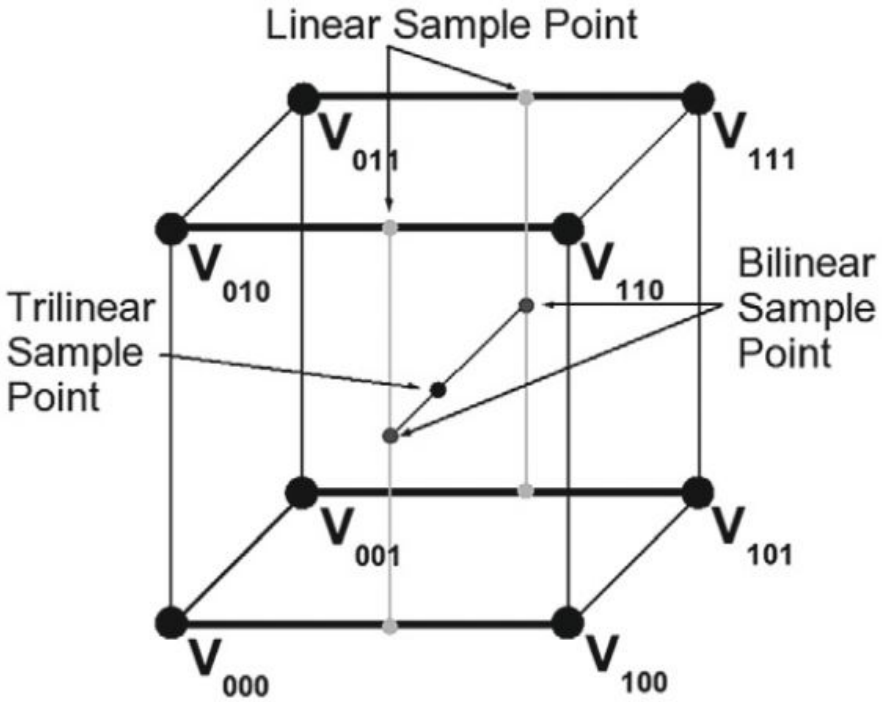
Trilinear projection is used. Trilinear interpolation involves only the immediate voxel neighborhood of a sample point, which are the voxels of the enclosing volume cell.

In fig. Voxels (u,v), points (x,y)



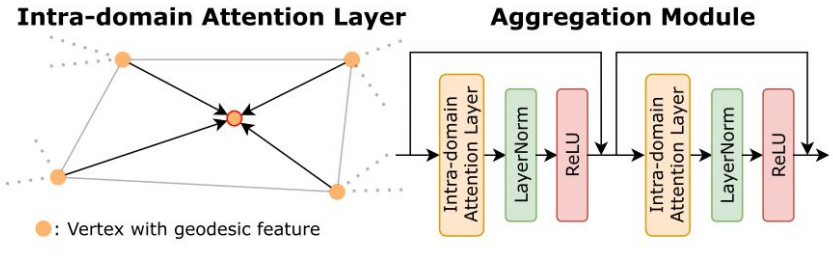
$$
\begin{aligned}
L(\alpha) &= V_0 * (1-\alpha) + V_1 * \alpha | 0 \le \alpha \le 1 \\
B(\alpha_1, \alpha_2) &= L_0(\alpha_1) * (1-\alpha_2) + L_1(\alpha_1) * \alpha_2 \\
&= (V_{00} * (1-\alpha) + V_{10} * \alpha)) * (1-\alpha_2) \\
&\quad + (V_{01} * (1-\alpha) + V_{11} * \alpha)) * \alpha_2 \\
T(x, y, z) &= B_0(x, y) * (1-z) + B_1(x, y) * z \\
&= ((L_0(x) * (1-y) + L_1(x) * y)) * (1-z) \\
&\quad + ((L_2(x) * (1-y) + L_3(x) * y)) * z \\
&= (((V_{000} * (1-x) + V_{100} * x)) * (1-y) \\
&\quad + ((V_{010} * (1-x) + V_{110} * x)) * y) * (1-z) \\
&\quad + (((V_{001} * (1-x) + V_{101} * x)) * (1-y) \\
&\quad + ((V_{011} * (1-x) + V_{111} * x)) * y) * z
\end{aligned}
$$

https://sci-hub.se/https://doi.org/10.1016/B978-0-12-415873-3.00002-X



Intra-domain Attention Layer

Ni is the one-ring neighborhood of vertex Vi. The functions ρintra, αintra , ϕintra , and ψintra are vertex-wise feature transformations implemented by MLP, ωij is the attention coefficient, and d is the size of output feature channels.

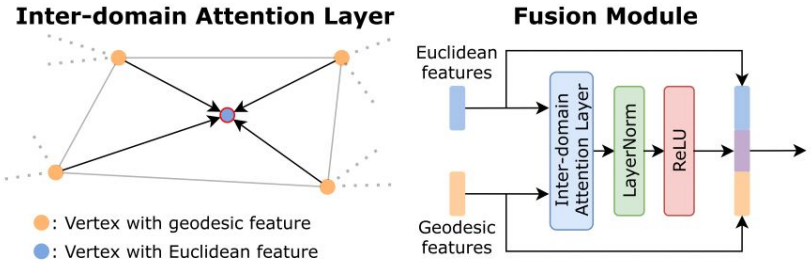


$$f_i^{'geo} = \rho_k^{intra}(f_i^{geo}) + \sum_{j \in N_i} \omega_{ij} \alpha_k^{intra}(f_j^{geo}),$$

$$\omega_{ij} = softmax(\frac{\varphi_k^{intra}(f_i^{geo})^T \psi_k^{intra}(f_j^{geo})}{\sqrt{d}}),$$

Inter-domain Attentive Fusion Module

where Ni is the same one-ring neighborhood of vertex Vi as the one used for intra-domain aggregation



$$f_i^{fuse} = \rho_k^{inter}(f_i^{euc}) + \sum_{j \in N_i} \omega_{ij} \alpha_k^{inter}(f_j^{geo}),$$

$$\omega_{ij} = softmax(\frac{\varphi_k^{inter}(f_i^{euc})^T \psi_k^{inter}(f_j^{geo})}{\sqrt{d}}),$$

Mesh Simplification

Each level of simplified mesh corresponds to a level of downsampled 3D sparse voxels.

Vertex Clustering

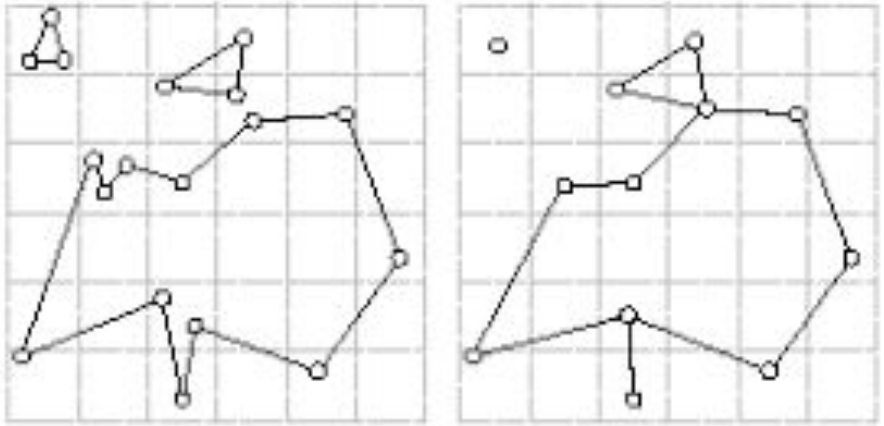
(1) construct a bounding box for all vertices on the triangle mesh;

(2) regularly subdivide the bounding box into smaller cells (cell size depends on the approximation error);

(3) associate each vertex with a single cell which enclose it and form all the vertices inside a specific cell a cluster;

(4) collapse all vertices in all clusters into a representative vertices;

(5) remove triangles having more than one vertex in a cluster and return the new mesh;



Quadric Error Metrics

determines how far a vertex is from an ideal spot after collapse, thus has explicit control over mesh topology.

The plane equation is $ax + by + cz + d = 0$ where $a^2 + b^2 + c^2 = 1$. Kp is quadric. It approximate error around vertex p

$$\mathbf{K_p} = \mathbf{pp}^\mathsf{T} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

QEM Algorithm

1. Calculate this K for all vertices V. K is loss

Now For a given contraction (v1 ,v2)→ v`, loss(v`) should be minimum wrt v1, v2.

Since the loss function is quadratic, finding its minimum is a linear problem. Thus, we find v` by solving ∂1/∂x = ∂1/∂y = ∂1/∂z = 0.

2. Select all valid pairs.
3. Compute the optimal contraction target v̄ for each valid pair (v1 ,v2). The error v`T (Q1 + Q2 ) v` of this target vertex becomes the cost of contracting that pair.
4. Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.
5. Iteratively remove the pair (v1 ,v2)of least cost from the heap, contract this pair, and update the costs of all valid pairs involving v1 .

Note

Directly applying the QEM method on the original meshes results in high-frequency signals in noisy areas.

VM-Net apply the VC method on the original mesh for the first two mesh levels and then apply the QEM method for the remaining mesh levels.

Result

Metric: mean IoU

Method	mIoU(%)	Conv Category
TangentConv [57]	43.8	2D-3D
SurfaceConvPF [68]	44.2	
3DMV [9]	48.3	
TextureNet [23]	56.6	
JPBNet [6]	63.4	
MVPNet [25]	64.1	
V-MVFusion [29]	74.6	
BPNet* [20]	**74.9**	
PointNet++ [45]	33.9	PointConv
FCPN [46]	44.7	
PointCNN [33]	45.8	
DPC [13]	59.2	
MCCN [19]	63.3	
PointConv [65]	66.6	
KPConv [58]	68.4	
JSENet [21]	69.9	
SparseConvNet [17]	72.5	SparseConv
MinkowskiNet [7]	73.6	
SPH3D-GCN [32]	61.0	GraphConv
HPEIN [26]	61.8	
DCM-Net [51]	65.8	
VMNet (**Ours**)	**74.6**	Sparse+Graph Conv

Table 1. **Mean intersection over union scores on ScanNet Test [8].** Detailed results can be found on the ScanNet benchmarking website[2]. * indicates a concurrent work.

Result with and without attention modules

Information	mIoU(%)
Geo Only	58.1
Euc Only	71.0
VMNet(Geo+Euc)	**73.3**

Baseline	Intra	Inter	mIoU(%)
✓			70.2
✓	✓		72.1
✓	✓	✓	**73.3**

Table 4. **Ablation study**: **(Left)** Euclidean and geodesic information; **(Right)** Network components.

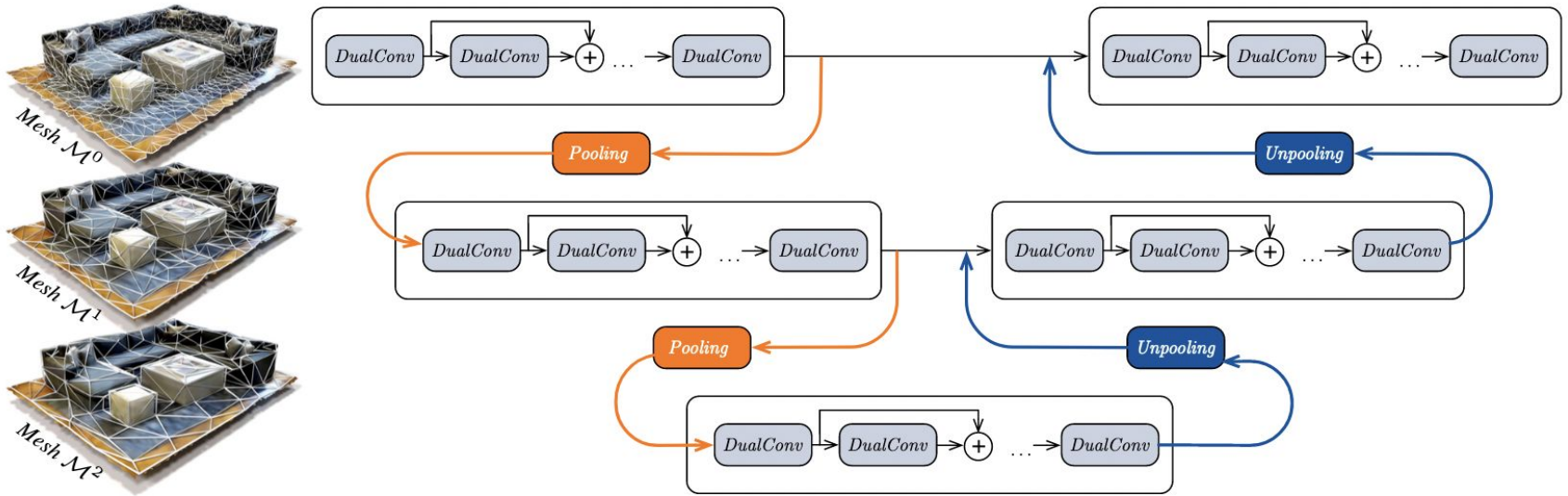
Mesh Simplification and Attention operators

Operator	mIoU(%)
Vector Attention	72.3
EdgeConv	72.6
Scalar Attention	**73.3**

Method	mIoU(%)
VC only	72.3
QEM only	72.9
VC + QEM	**73.3**

Table 5. **Ablation study**: **(Left)** Attentive operators; **(Right)** Mesh simplification.

DCM-Net Arch


Mesh $\mathcal{M}^0$
Mesh $\mathcal{M}^1$
Mesh $\mathcal{M}^2$
DualConv
Pooling
Unpooling

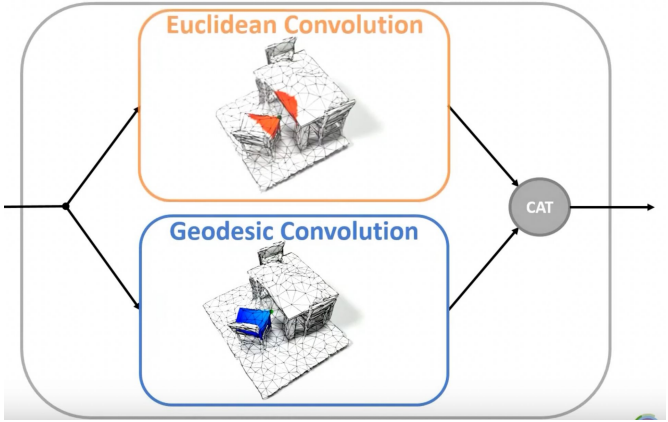
Difference from DCM-Net

Only difference between VM-Net and DCN-Net, is the type of conv operators.

DCM-Net - GraphConv

VM-Net - SparseConv + GraphConv

Then concat of 2 features is done DCM-Net they are not agg as in VM-Net using attention modules.

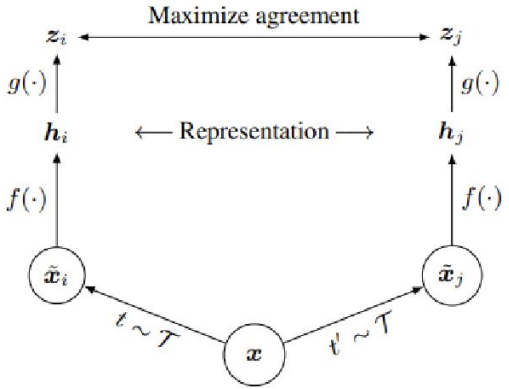


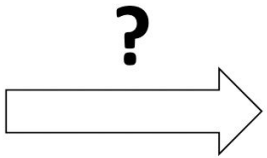
Self-Damaging Contrastive Learning

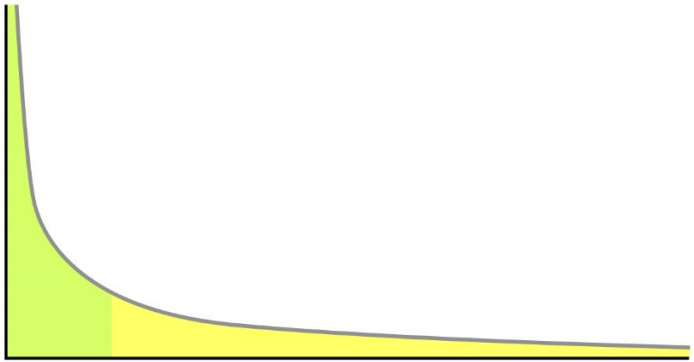
https://arxiv.org/pdf/2106.02990.pdf

Abstract

- Real world data is imbalance and spare in some classes.
- When large networks are pruned, it forgets difficult-to-memorize samples. (generally they are long-tail samples)







Novel idea

create a dynamic **self-competitor model** to contrast with the **target model**, which is a pruned version of the latter.

During training, contrasting the two models will lead to adaptive online mining of the most easily forgotten samples for the current target model, and implicitly emphasize them more in the contrastive loss.

Extensive experiments across multiple datasets and imbalance settings show that SDCLR significantly improves not only overall accuracies but also balancedness, in terms of linear evaluation on the full-shot and few-shot settings.

What do compressed deep neural networks forget?

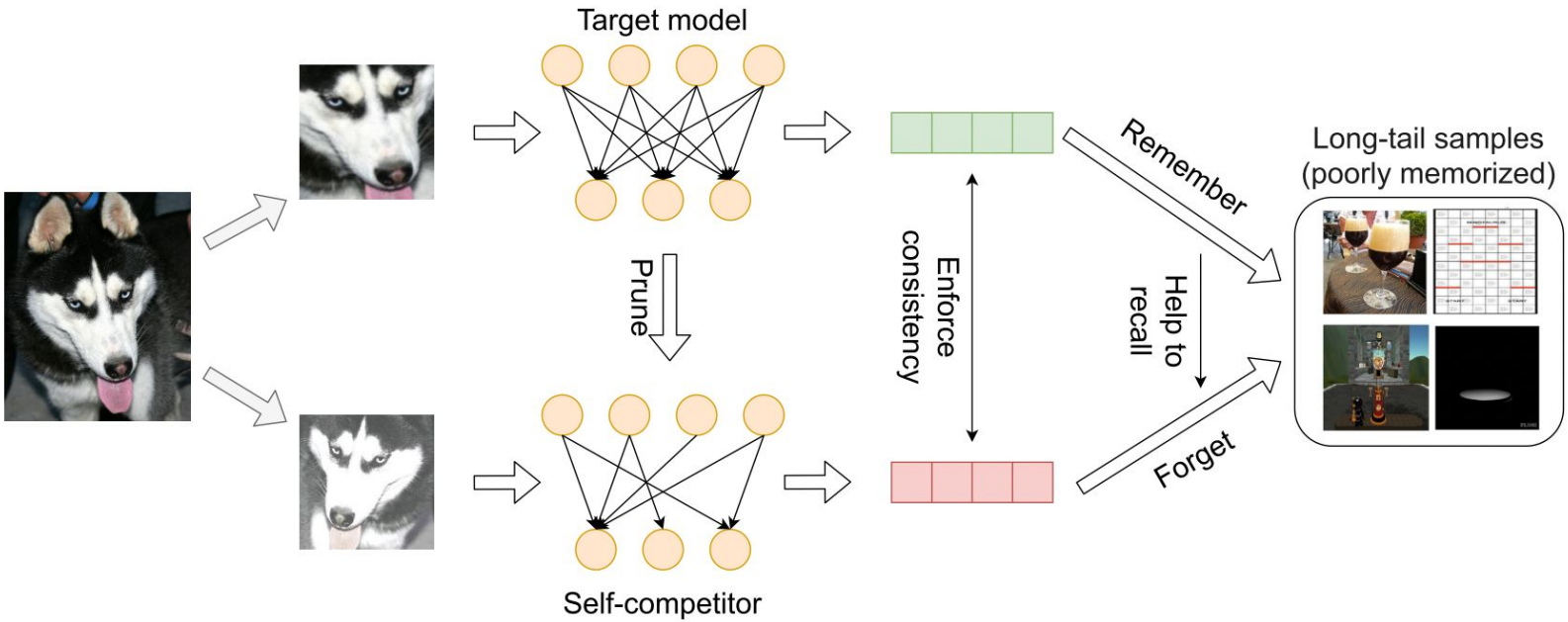
https://arxiv.org/pdf/1911.05248.pdf

Findings:

1. top-1 or top-5 test-set accuracy hide critical details in the ways that pruning impacts model generalization.
2. The examples most impacted by pruning, Pruning Identified Exemplars (PIEs), are more challenging for both models and humans to classify (Human study done here). Compression impairs the model's ability to predict accurately on the long-tail of less frequent instances.
3. Pruned networks are more sensitive to natural adversarial images and corruptions. This sensitivity is amplified at higher levels of compression.

PIE Samples

Pruning Identified Exemplars (PIEs) are images where there is a high level of disagreement between the predictions of pruned and non-pruned models. PIEs can be thought as images that are forgotten after pruning.


Target model
Prune
Self-competitor
Enforce consistency
Remember
Help to recall
Forget
Long-tail samples
(poorly memorized)

Self-Competitor Network

Self competitor network is pruned version of target network. **It happens at training step.**

Since the self-competitor is always obtained and updated from the latest target model, the two branches will co-evolve during training. Their contrasting will implicitly give more weights on long-tail samples.

This also makes it different from SimCLR.

How to Measure Representation Balancedness

The balancedness of a feature space can be reflected by the linear separability w.r.t. all classes.

1. learn the visual representation fv on the training dataset with LCL.
2. training a linear classifier layer L on the top of fv with a labeled balanced dataset (by default, the full dataset where the imbalanced subset is sampled from).
3. evaluating the accuracy of the linear classifier L on the testing set.

Here such accuracy measure is called **linear separability performance.**

Few Shot performance measure

In place of step 2.

we use only 1% samples of the full dataset from which the pre-training imbalanced dataset is sampled.

Such a metric is few shot performance measure.

Experiment

Divide each dataset to three disjoint groups: {Many, Medium, Few}.

In subsets of CIFAR10/CIFAR100, Many and Few each include the largest and smallest 1/3 classes, respectively.

For instance

- CIFAR-100: the classes with [500-106, 105-20, 19-5] samples belong to [Many (34 classes), Medium (33 classes), Few (33 classes)] categories, respectively.

- ImageNet, we follow OLTR (Liu et al., 2019) to define Many as classes each with over training 100 samples, Medium as classes each with 20-100 training samples and Few as classes under 20 training samples.

Loss Function

Normalized Temperature-scaled Cross Entropy Loss as in SimCLR.

$$\mathcal{L}_{\mathrm{CL}} = \frac{1}{N} \sum_{i=1}^{N} -\log \frac{s\left(v_i, v_i^+, \tau\right)}{s\left(v_i, v_i^+, \tau\right) + \sum_{v_i^- \in V^-} s\left(v_i, v_i^-, \tau\right)}$$

$$s\left(v_i, v_i^+, \tau\right) = \exp\left(v_i \cdot v_i^+ / \tau\right)$$

Result

Table 1. Comparing the *linear separability performance* for models learned on balanced subset D_b and long-tail subset D_i of CIFAR10 and CIFAR100. *Many*, *Medium* and *Few* are split based on class distribution of the corresponding D_i.

Dataset	Subset	*Many*	*Medium*	*Few*	All
CIFAR10	D_b	82.93 ± 2.71	81.53 ± 5.13	77.49 ± 5.09	80.88 ± 0.16
	D_i	78.18 ± 4.18	76.23 ± 5.33	71.37 ± 7.07	75.55 ± 0.66
CIFAR100	D_b	46.83 ± 2.31	46.92 ± 1.82	46.32 ± 1.22	46.69 ± 0.63
	D_i	50.10 ± 1.70	47.78 ± 1.46	43.36 ± 1.64	47.11 ± 0.34

Result

Table 2. Comparing the *few-shot performance* for models learned on balanced subset D_b and long-tail subset D_i of CIFAR10 and CIFAR100. *Many*, *Medium* and *Few* are split according to class distribution of the corresponding D_i.

Dataset	Subset	*Many*	*Medium*	*Few*	All
CIFAR10	D_b	77.14 ± 4.64	74.25 ± 6.54	71.47 ± 7.55	74.57 ± 0.65
	D_i	76.07 ± 3.88	67.97 ± 5.84	54.21 ± 10.24	67.08 ± 2.15
CIFAR100	D_b	25.48 ± 1.74	25.16 ± 3.07	24.01 ± 1.23	24.89 ± 0.99
	D_i	30.72 ± 2.01	21.93 ± 2.61	15.99 ± 1.51	22.96 ± 0.43

Result

Table 3. Comparing the *linear separability performance* and *few-shot performance* for models learned on balanced subset D_b and long-tail subset D_i of ImageNet and ImageNet-100. We consider two long tail distributions for ImageNet: Pareto and Exp, which corresponds to ImageNet-LT and Imagenet-LT-exp, respectively. *Many*, *Medium* and *Few* are split according to class distribution of the corresponding D_i.

Dataset	Long tail type	Split type	*linear separability*				*few-shot*			
			Many	*Medium*	*Few*	All	*Many*	*Medium*	*Few*	All
ImageNet	*Pareto*	D_b	58.03	56.02	56.71	56.89	29.26	26.97	27.82	27.97
		D_i	58.56	55.71	56.66	56.93	31.36	26.21	27.21	28.33
ImageNet	*Exp*	D_b	57.46	57.70	57.02	57.42	32.31	32.91	32.17	32.45
		D_i	58.37	56.97	56.27	57.43	35.98	29.56	28.02	32.12
ImageNet-100	*Pareto*	D_b	68.87	66.33	61.85	66.74	48.82	44.71	41.08	45.84
		D_i	69.54	63.71	59.69	65.46	48.36	39.00	35.23	42.16

Comparison with SimCLR

Table 4. Compare the proposed SDCLR with SimCLR in terms of the *linear separability performance*. ↑ means the metric the higher the better and ↓ means the metric is the lower the better.

Dataset	Framework	*Many* ↑	*Medium* ↑	*Few* ↑	*Std* ↓	All ↑
CIFAR10-LT	SimCLR	78.18 ± 4.18	76.23 ± 5.33	71.37 ± 7.07	5.13 ± 3.66	75.55 ± 0.66
	SDCLR	86.44 ± 3.12	81.84 ± 4.78	76.23 ± 6.29	5.06 ± 3.91	82.00 ± 0.68
CIFAR100-LT	SimCLR	50.10 ± 1.70	47.78 ± 1.46	43.36 ± 1.64	3.09 ± 0.85	47.11 ± 0.34
	SDCLR	58.54 ± 0.82	55.70 ± 1.44	52.10 ± 1.72	2.86 ± 0.69	55.48 ± 0.62
ImageNet-100-LT	SimCLR	69.54	63.71	59.69	4.04	65.46
	SDCLR	70.10	65.04	60.92	3.75	66.48

Comparison with SimCLR

Table 5. Compare the proposed SDCLR with SimCLR in terms of the *few-shot performance*. ↑ means the metric the higher the better and ↓ means the metric is the lower the better.

Dataset	Framework	*Many* ↑	*Medium* ↑	*Few* ↑	*Std* ↓	All ↑
CIFAR10	SimCLR	76.07 ± 3.88	67.97 ± 5.84	54.21 ± 10.24	9.80 ± 5.45	67.08 ± 2.15
	SDCLR	76.57 ± 4.90	70.01 ± 7.88	62.79 ± 7.37	6.99 ± 5.20	70.47 ± 1.38
CIFAR100	SimCLR	30.72 ± 2.01	21.93 ± 2.61	15.99 ± 1.51	6.27 ± 1.20	22.96 ± 0.43
	SDCLR	29.72 ± 1.52	25.41 ± 1.91	20.55 ± 2.10	3.98 ± 0.98	25.27 ± 0.83
Imagenet-100-LT	SimCLR	48.36	39.00	35.23	5.52	42.16
	SDCLR	48.31	39.17	36.46	5.07	42.38

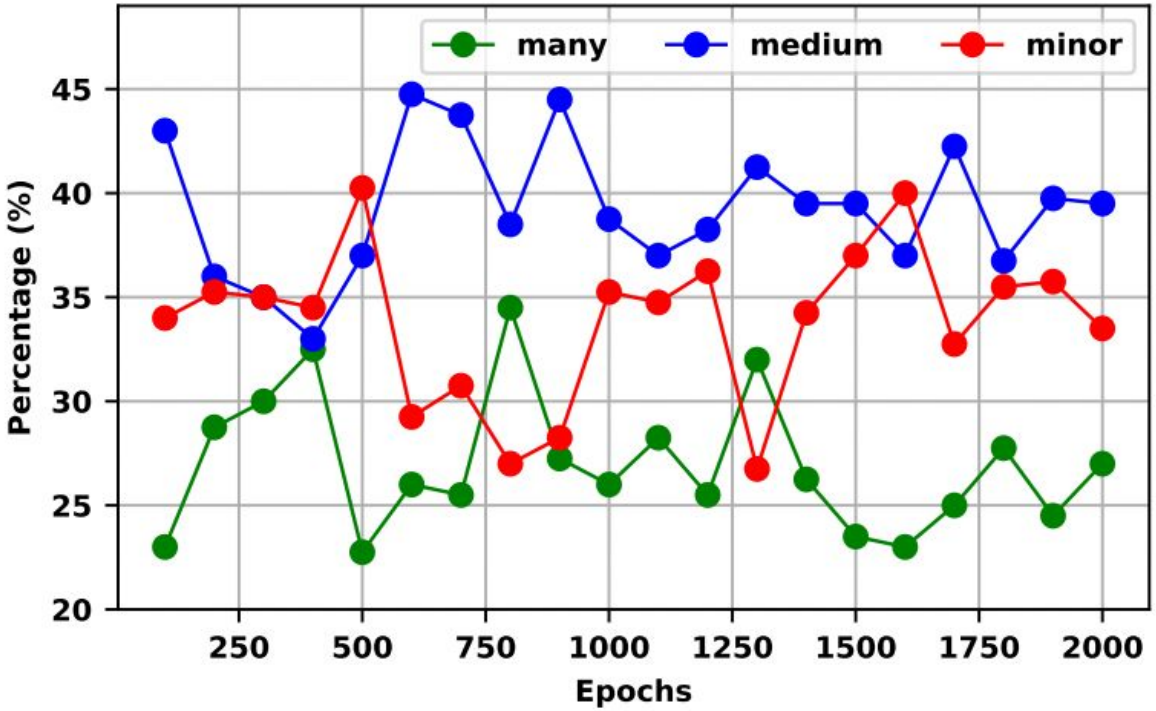


Figure 2. Pre-training on imbalance splits of CIFAR100, The percentage of *many* (●), *medium* (●) and *few* (●) in 1% most easily forgotten data under different training epochs.

Visualization

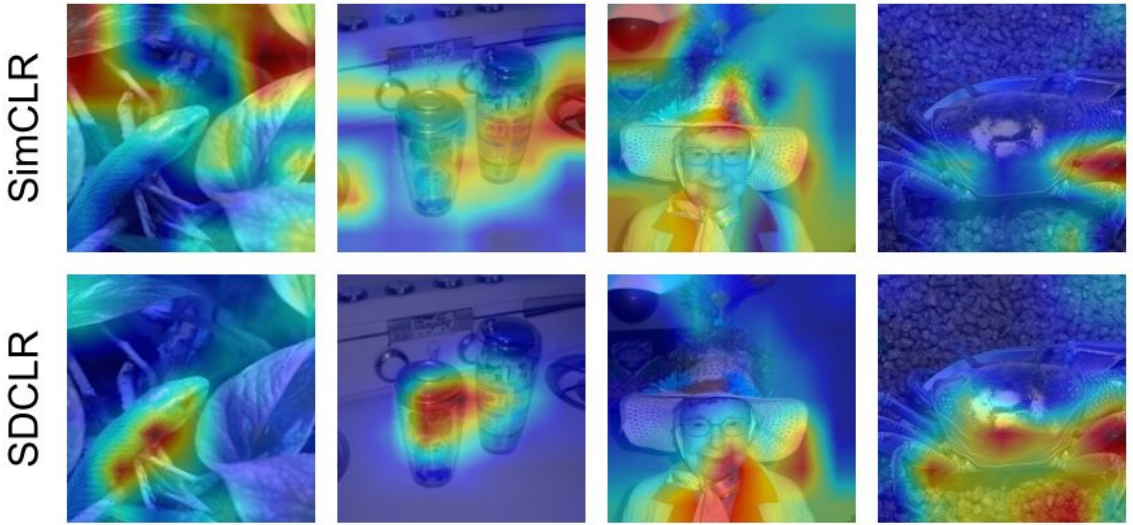


Figure 5. Visualization of attention on tail class images with Grad-CAM (Selvaraju et al., 2017). The first and second row corresponds to SimCLR and SDCLR, respectively.