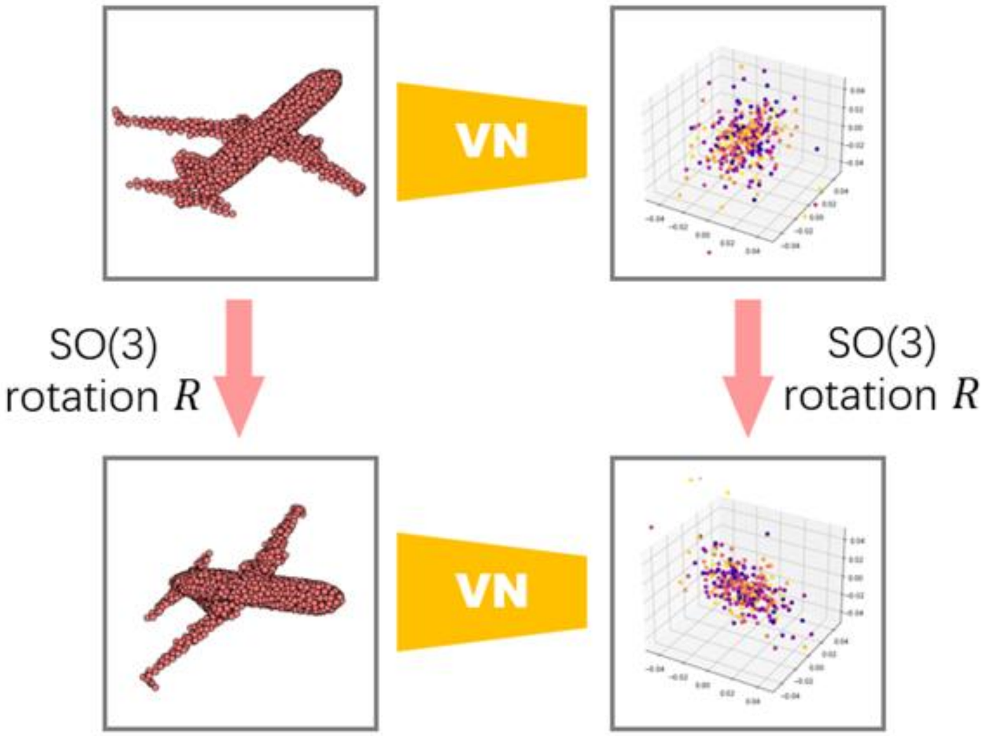
Injecting 3D Priors Into Neural Nets



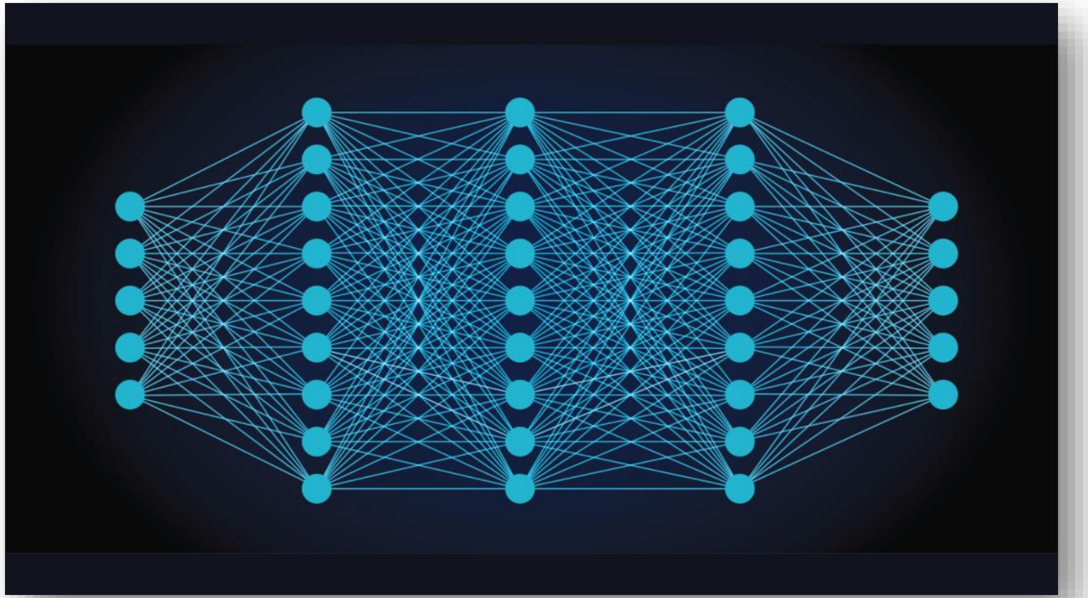
Leonidas Guibas
Stanford University

Leonidas Guibas Laboratory

Geometric Computing

Ind3D: Enforcing Inductive Bias in 3D Generation

Things Neural Nets Should Know



Things Neural Nets Should Know

- Representation shaping: Distill knowledge into the neural representation

  Point Cloud 3D Rotation Equivariance

- Network shaping: Distill knowledge into the neural architectures that use the representation

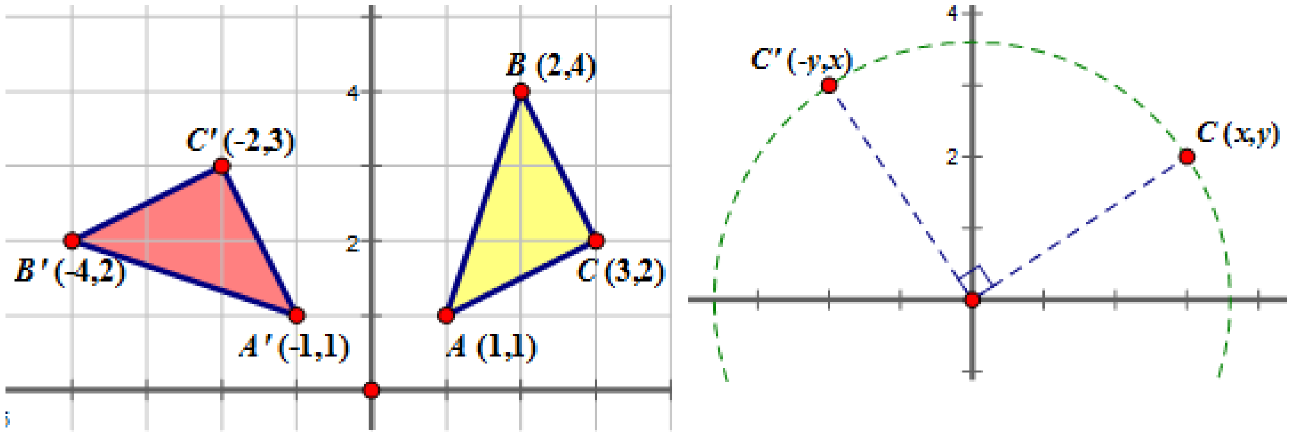
  Mutual Information Grouping in NeRFs and GSs

Enforcing Classic Invariance and Equivariance for 3D Objects and Scenes

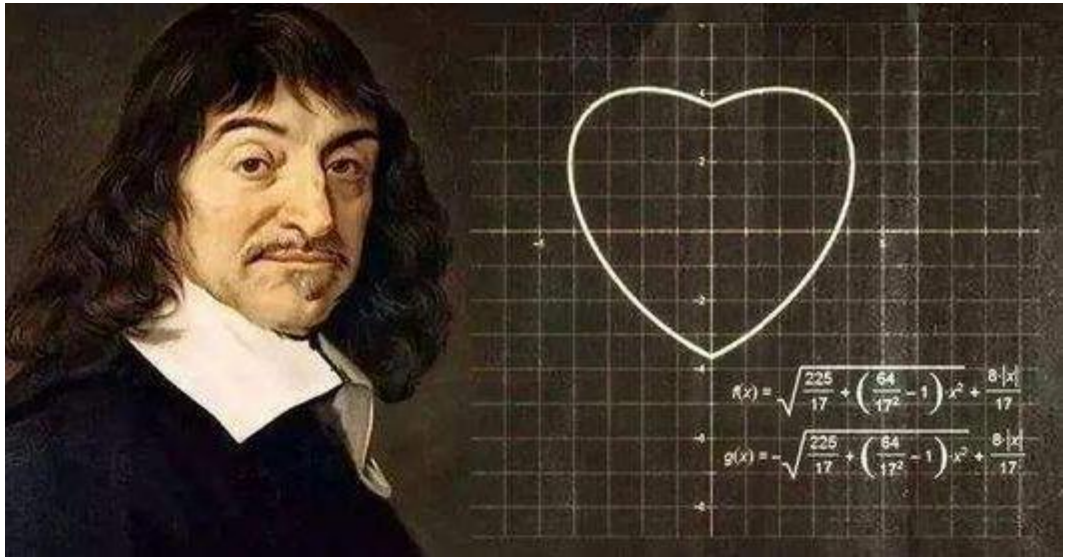
Geometry with Coordinates

Rotation of 90° about the Origin

When we rotate 90° about the origin, we see that the x and y coordinates are reversed and the new x coordinate is negated.

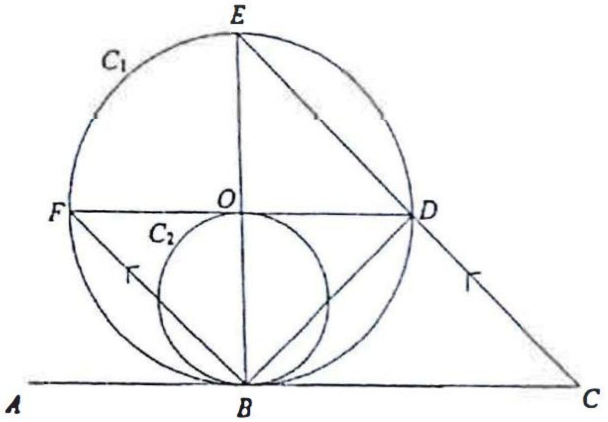


RULE FOR ROTATION BY 90° ABOUT THE ORIGIN $R_{o,90°}(x,y)=(-y,x)$



René Descartes

Geometry without Coordinates



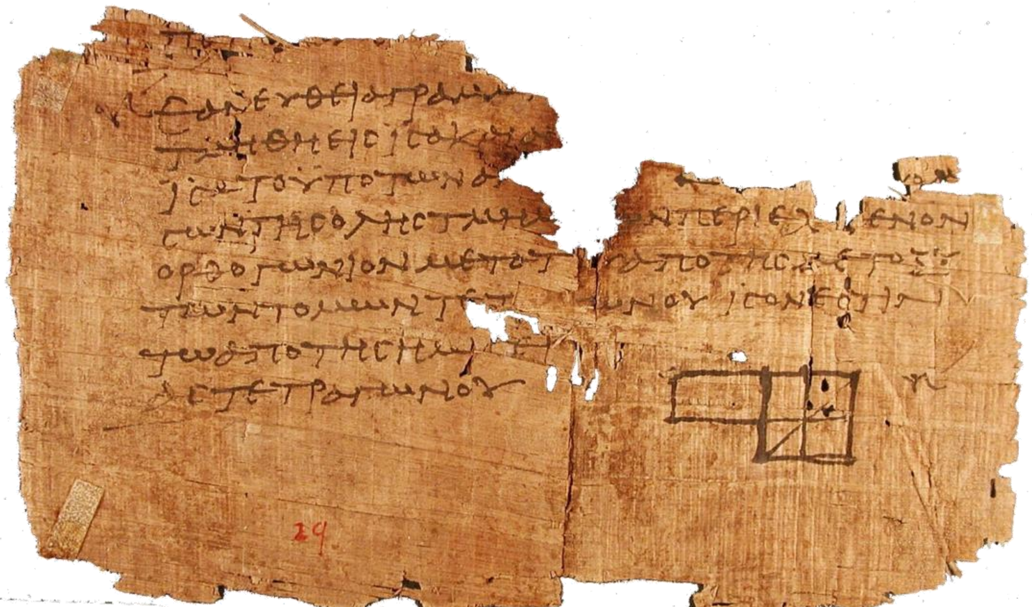
In the above diagram, C_1 is a circle with centre O, and C_2 is another circle passing through the point O and touching the circumference of C_1 at the point B. AC is the tangent to C_1 at B and meets the line ED produced at the point C. DF is the tangent to C_2 at O. BOE is a straight line.

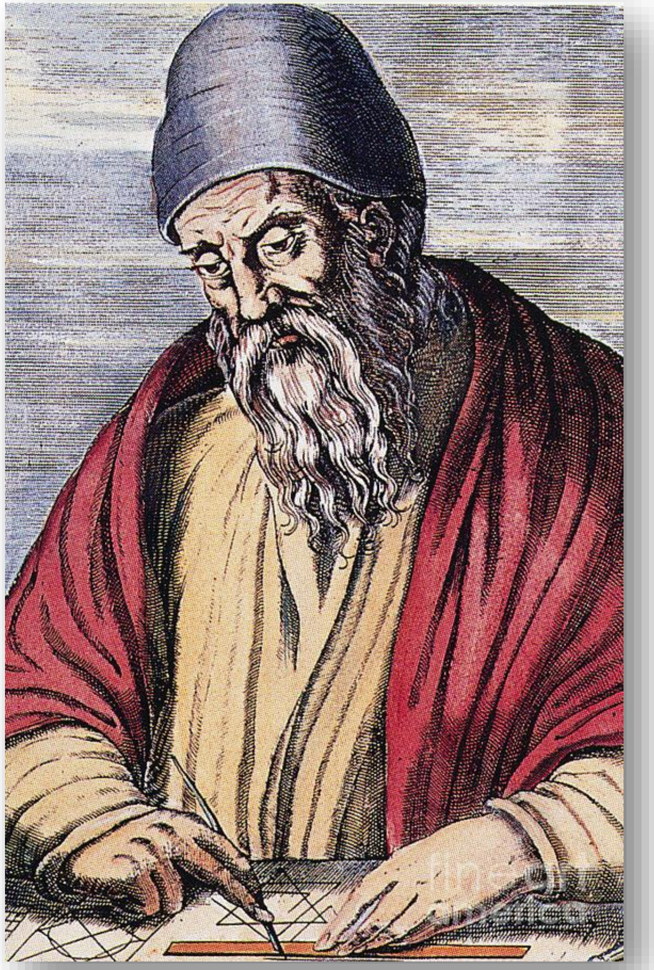
Given that BF is parallel to CE, show that

(i) $\angle BDC$ is a right angle,

(ii) $\dfrac{OE}{BC} = \dfrac{1}{2}$, and

(iii) $CD \times CE = BC^2$

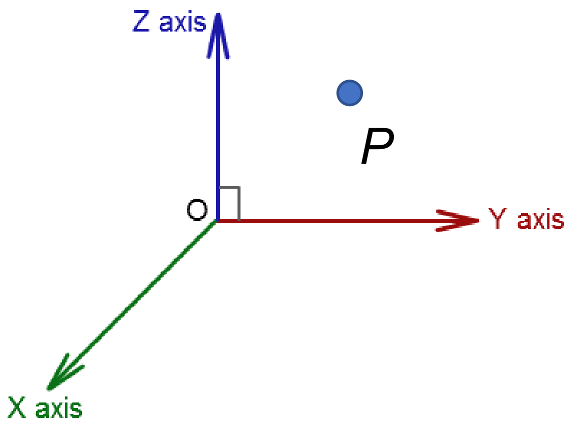


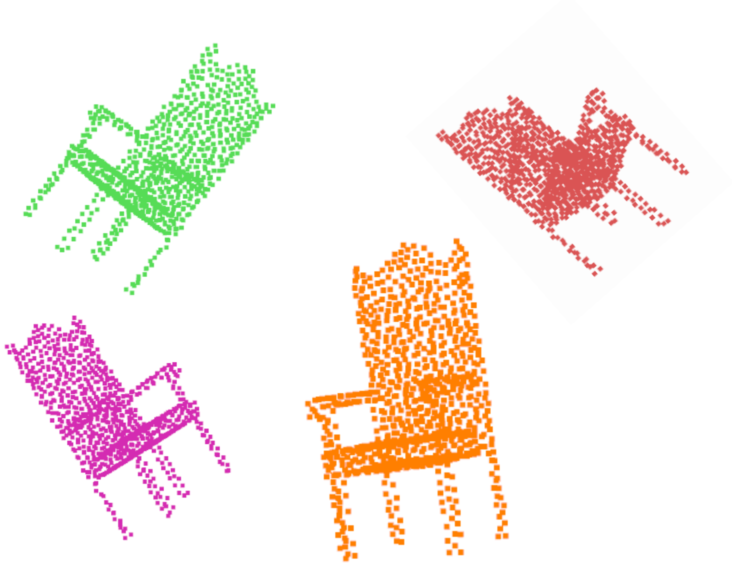


Euclid

Factors of 3D Variation

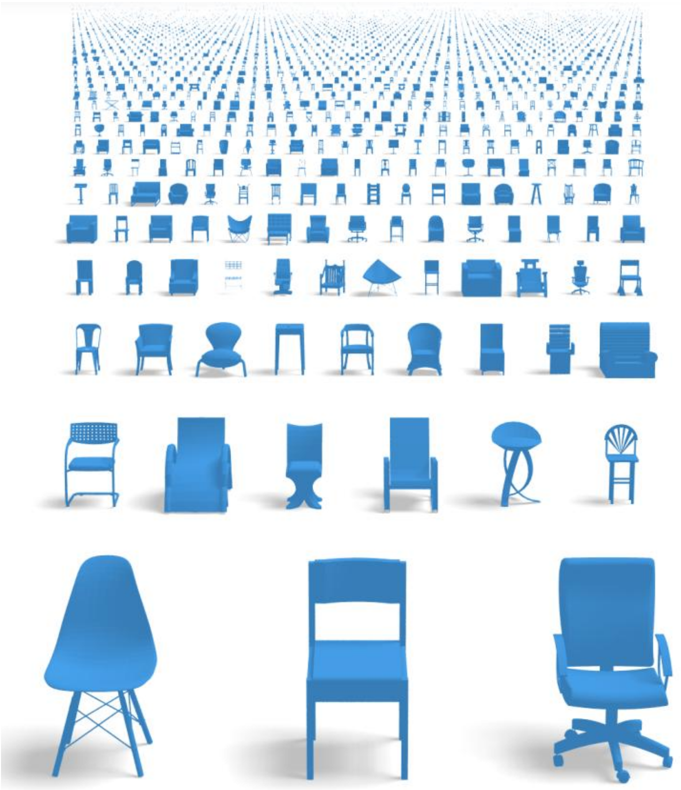
- Geometric data is almost always given to us in a particular coordinate system.
- In many settings this frame or pose aspect of the presentation needs to be disentangled from the intrinsic data geometry, as it may be a nuisance factor.
- Depending on the application, this variation is captured by a transformation group that may include translation, rotation, scale, etc.





Learning on Geometric Shapes

- Typical neural networks are trained on co-aligned collections of shapes



- Such networks do not generalize to objects in arbitrary poses

Neural Network

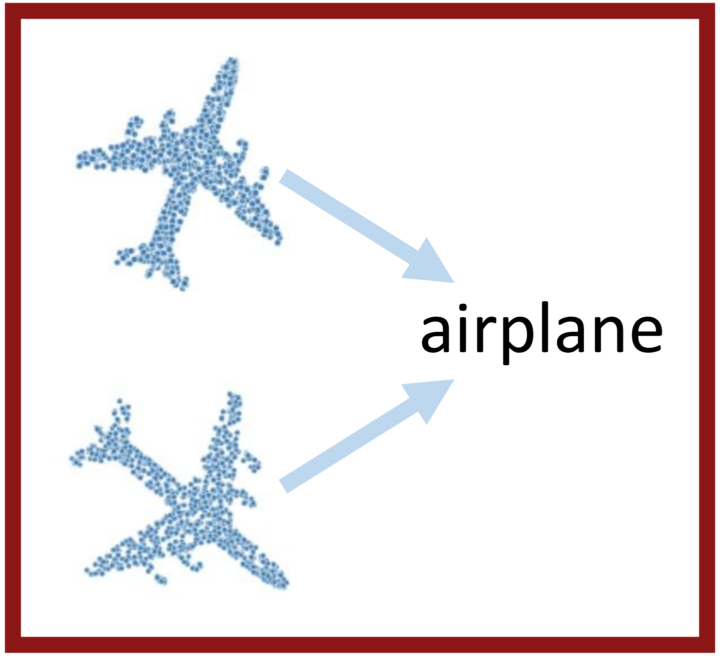
“chair”

Neural Network

“???”

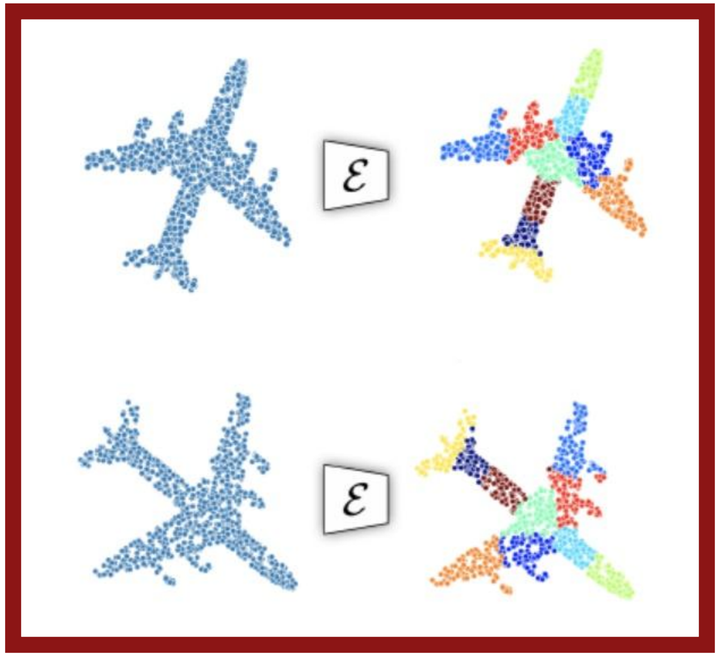
Key Requirement: Invariance and Equivarience

classification



invariant

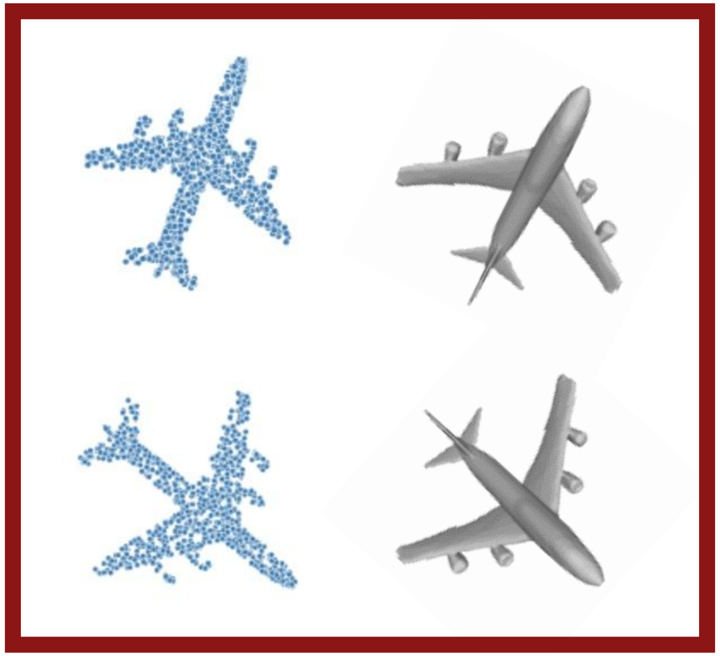
segmentation



equivariant

(or **invariant**, depending on data representation)

reconstruction



equivariant encoder

invariant decoder

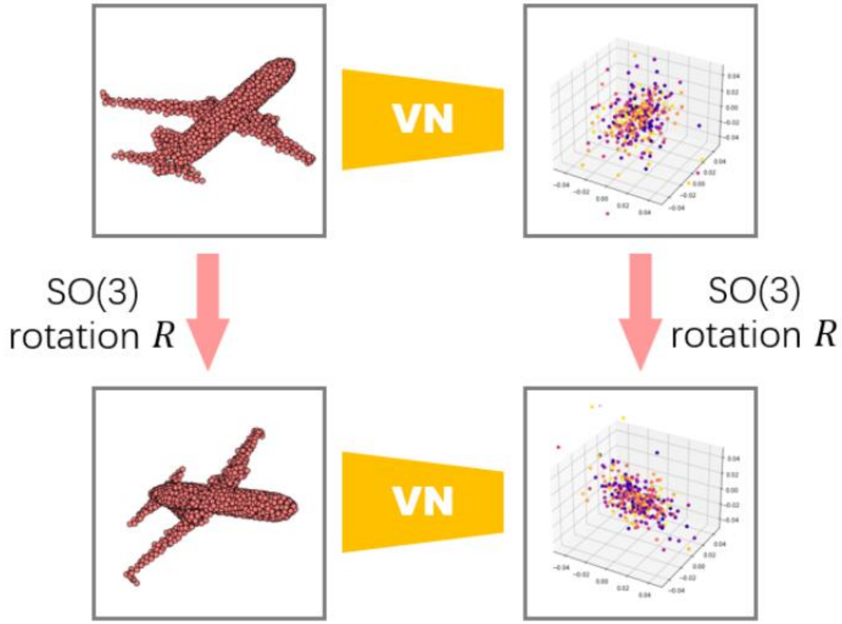
[W. Sun, A. Tagliasacchi, B. Deng, S. Sabour, S. Yazdani, G. Hinton, K. M. Yi, arXiv:2012.04718 (2020)]
[J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, CVPR 2019]

In/Equivariance

We say a neural network $f(\cdot\,;\theta)$ is rotation equivariant, if for any 3D rotation $R \in \mathrm{SO}(3)$ applied to its input $\mathbf{x}$, it is explicitly related to a transformation $D(R)$ on the network output satisfying

$$f(\mathbf{x}R;\theta) = f(\mathbf{x};\theta)D(R)$$

- $D(R)$ should be independent of $\mathbf{x}$
- **Special case:** when $D(R) = R$ is the identity mapping, it is the common-sense “equivariance”
- **Special case:** when $D(R) = \mathrm{I}$ is the identity mapping, it is invariance

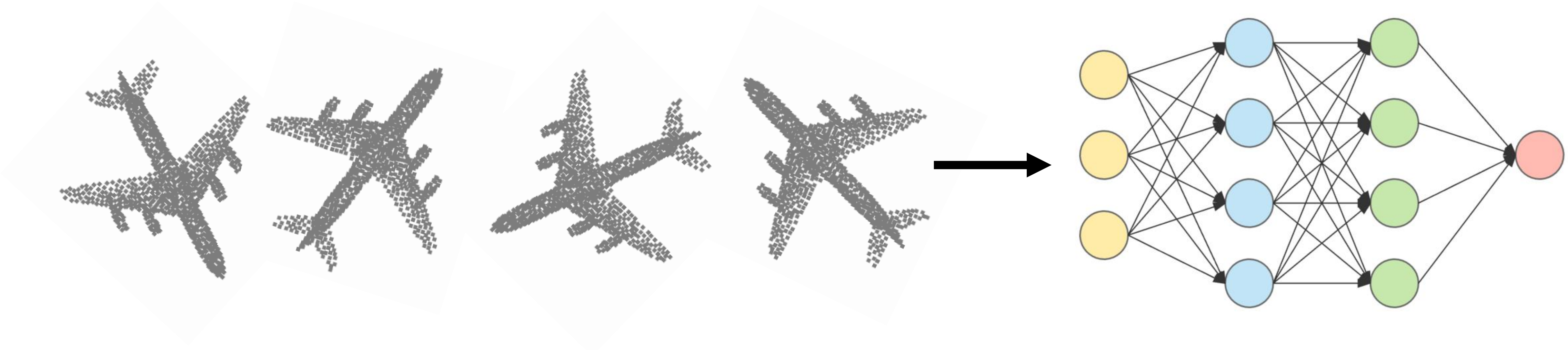


A Naïve Solution: Data Augmentation

Apply random rotations to the training data

So we let the network "see" and learn from all possible poses

- Reducing the generalization gap – but not eliminating it
- Sacrificing data-efficiency – longer training time
- Statistically equivariant/invariant – not guaranteed



Vector Neurons for SO(3) Equivariance

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, Leonidas Guibas,
ICCV'21

Scalar vs Vector Neurons

Classical (scalar) feature $\boldsymbol{z} = [z_1, z_2, \cdots, z_C]^\top \in \mathbb{R}^C$, with $z_i \in \mathbb{R}$

Vector-list feature $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_C]^\top \in \mathbb{R}^{C \times 3}$, with $\boldsymbol{v}_i \in \mathbb{R}^3$

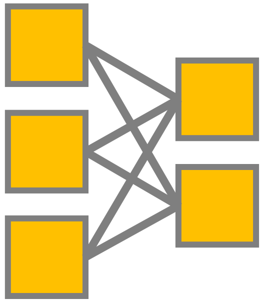
- For pointcloud with N points $\mathcal{V} = \{\boldsymbol{V}_1, \boldsymbol{V}_2, \cdots, \boldsymbol{V}_N\} \in \mathbb{R}^{N \times C \times 3}$

Mapping between network layers:

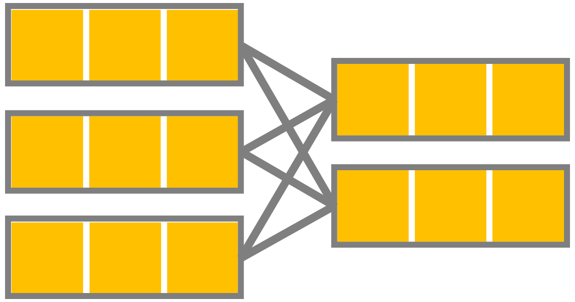
$$f(\cdot; \theta) : \mathbb{R}^{N \times C^{(d)} \times 3} \rightarrow \mathbb{R}^{N \times C^{(d+1)} \times 3}$$

Equivariance to rotation $R \in \mathrm{SO}(3)$

$$f(\mathcal{V}R; \theta) = f(\mathcal{V}; \theta)R$$



(classical) scalar neurons



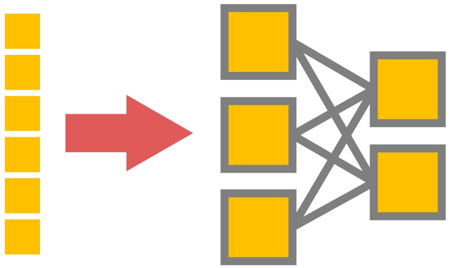
vector neurons

Expressing Transformations in the Latent Space

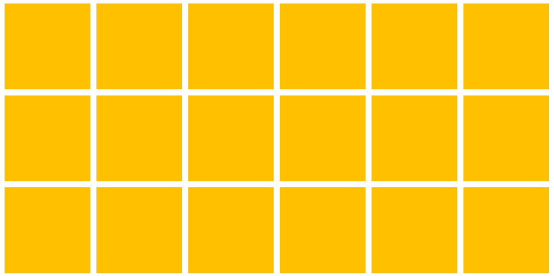
A network whose latent space understands rigid transformations

Classical Neuron: scalar channels

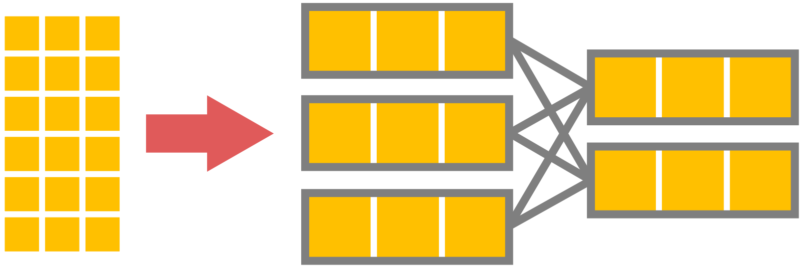
$C \times 1$ feature



Vector Neuron: 3D vector channels

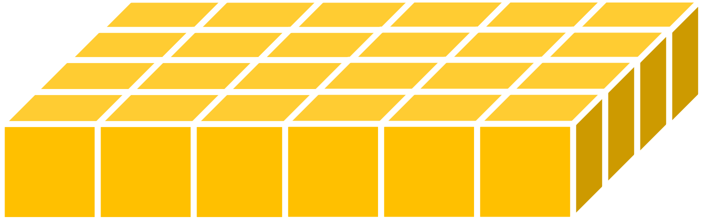


$C \times 3$ feature



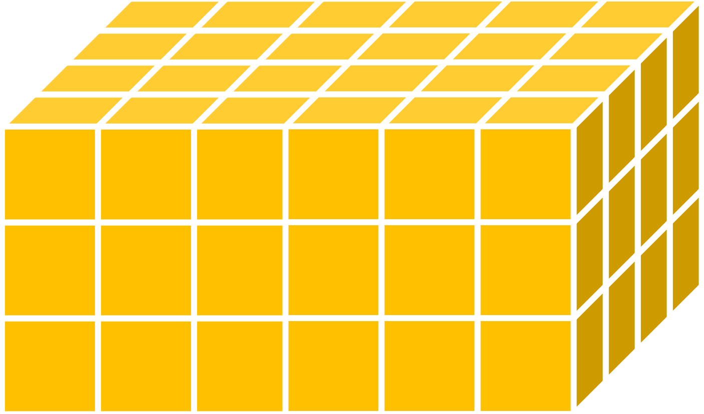
Vector Neuron Features for Point Cloud

Classical:

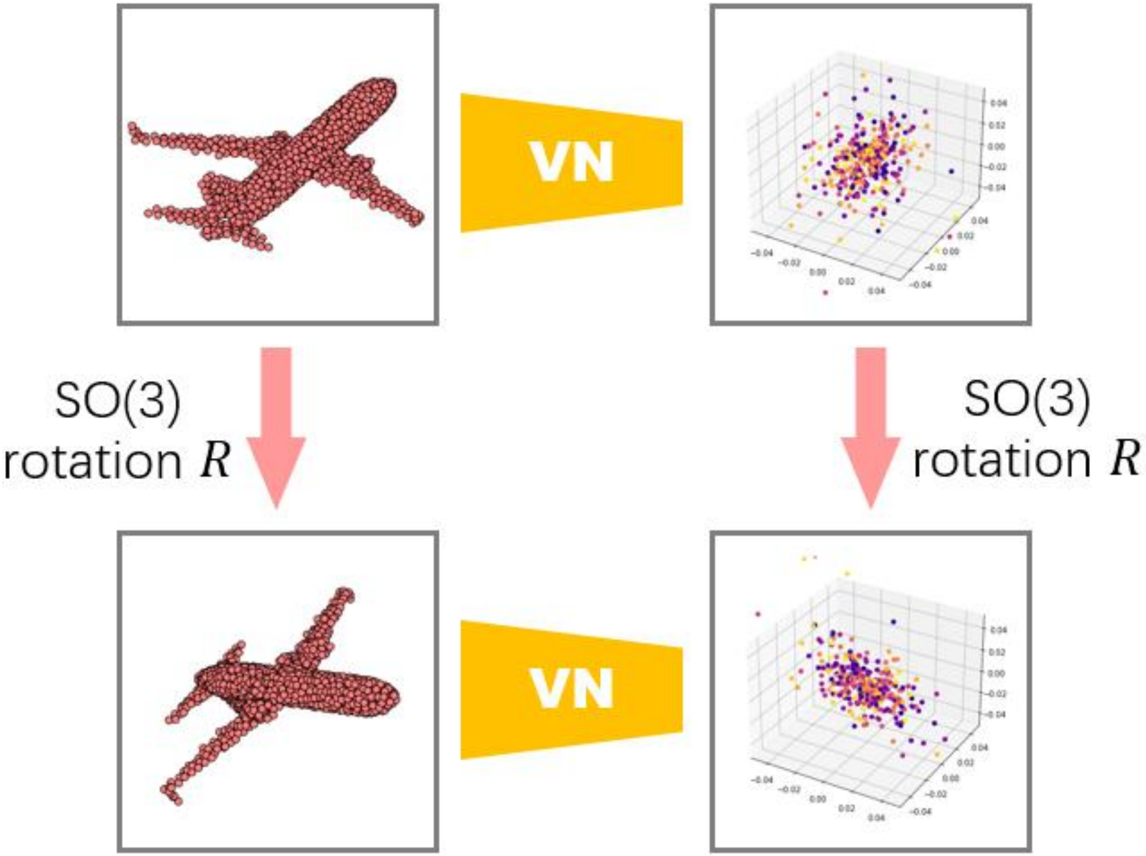


$N \times C \times 1$ feature

VN:

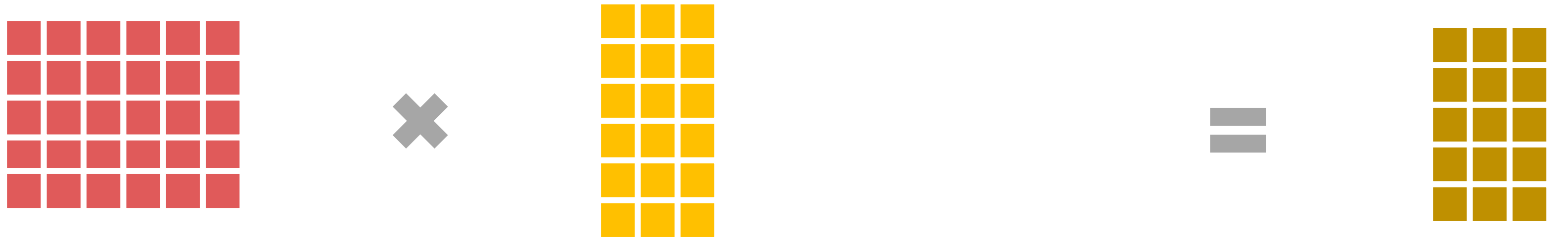


$N \times C \times 3$ feature



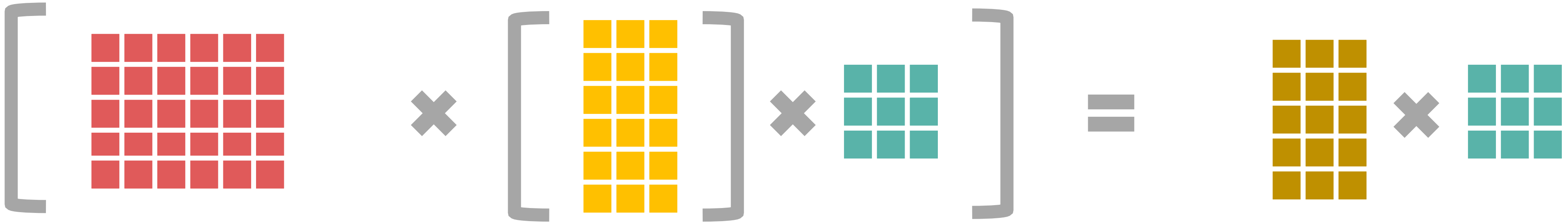
Vector Neuron Linear Operations

Linear operator: left multiply by the learnable weight matrix



$C' \times C$ weight $\quad$ $C \times 3$ feature $\quad$ $C' \times 3$ feature

Equivariance: right multiply by the SO(3) rotation matrix



$C' \times C$ weight $\quad$ $C \times 3$ feature $\quad$ $C' \times 3$ feature

VN Linear Layer

Vector-list feature $\boldsymbol{V} \in \mathbb{R}^{C \times 3}$

Linear operator $f_{\text{lin}}(\,\cdot\,; \mathbf{W})$ with learnable weights $\mathbf{W} \in \mathbb{R}^{C' \times C}$:

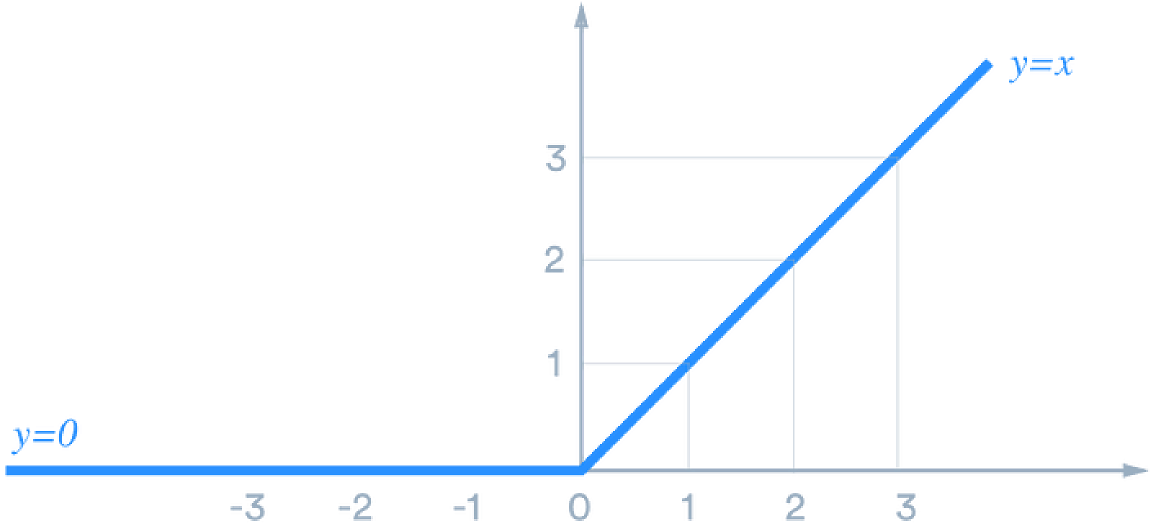
$$\boldsymbol{V}' = f_{\text{lin}}(\boldsymbol{V}; \mathbf{W}) = \mathbf{W}\boldsymbol{V} \in \mathbb{R}^{C' \times 3}$$

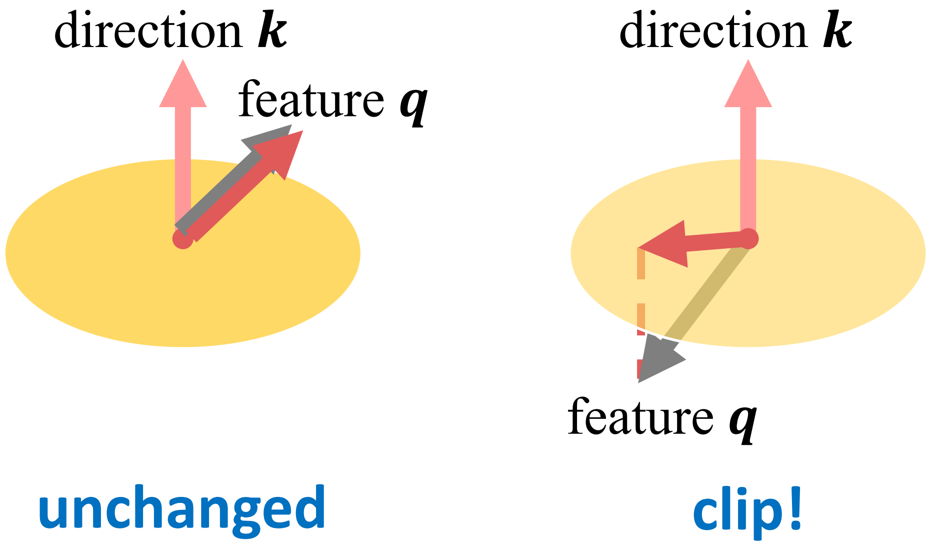
Equivariance to rotation $R \in \mathrm{SO}(3)$:

$$f_{\text{lin}}(\boldsymbol{V}R; \mathbf{W}) = \mathbf{W}\boldsymbol{V}R = f_{\text{lin}}(\boldsymbol{V}; \mathbf{W})R = \boldsymbol{V}'R$$

- $\mathbf{W}$ - left multiplication, R - right multiplication
- Note the absence of a bias term

Vector Neuron ReLU Non-Linearity





Rectified Linear Unit: VN Non-Linearity

ReLU Non-Linearity

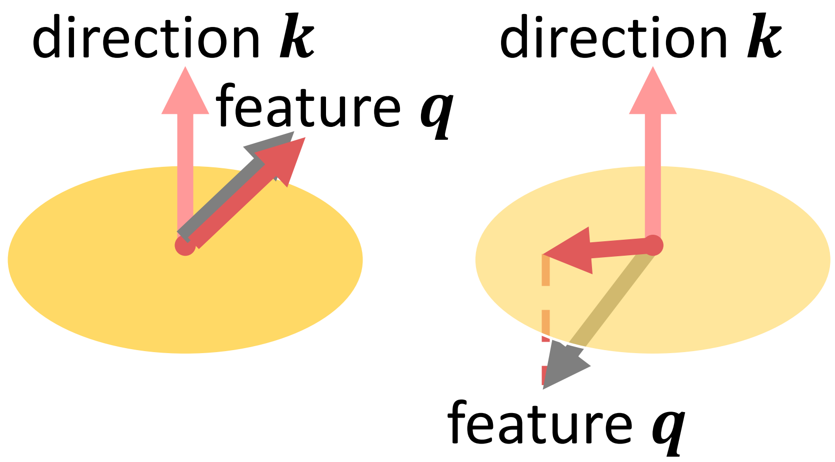
Weights $\mathbf{W} \in \mathbb{R}^{1\times C}$ and $\mathbf{U} \in \mathbb{R}^{1\times C}$

Learn a feature $\boldsymbol{q} = \mathbf{W}\boldsymbol{V} \in \mathbb{R}^{1\times 3}$

Learn a direction $\boldsymbol{k} = \mathbf{U}\boldsymbol{V} \in \mathbb{R}^{1\times 3}$

For each output vector neuron $\boldsymbol{v}' \in \boldsymbol{V}'$

$$\boldsymbol{v}' = \begin{cases} \boldsymbol{q} & \text{if } \langle \boldsymbol{q}, \boldsymbol{k} \rangle \geqslant 0 \\ \boldsymbol{q} - \langle \boldsymbol{q}, \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} \rangle \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} & \text{otherwise} \end{cases}$$



unchanged **clip!**

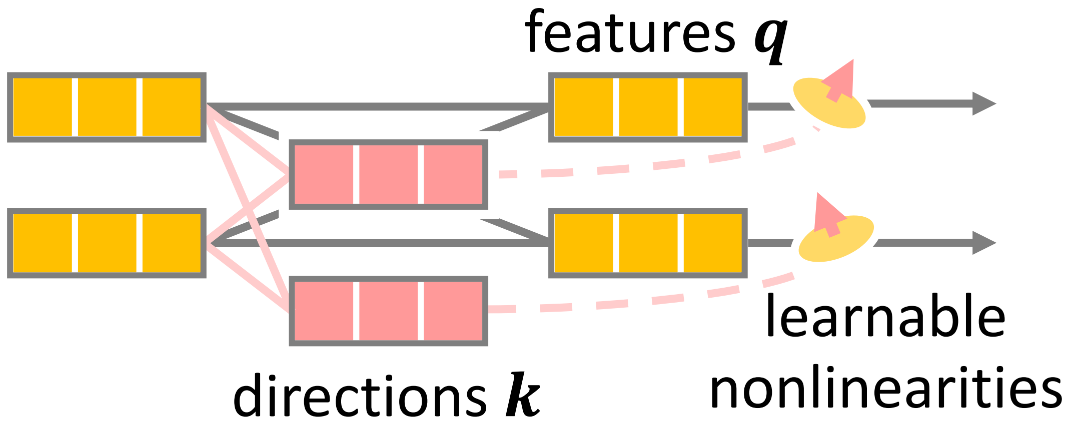
[Also works for uniform scaling]

VN Non-Linearity: A High-d ReLU

Learnable ReLU Non-Linearity

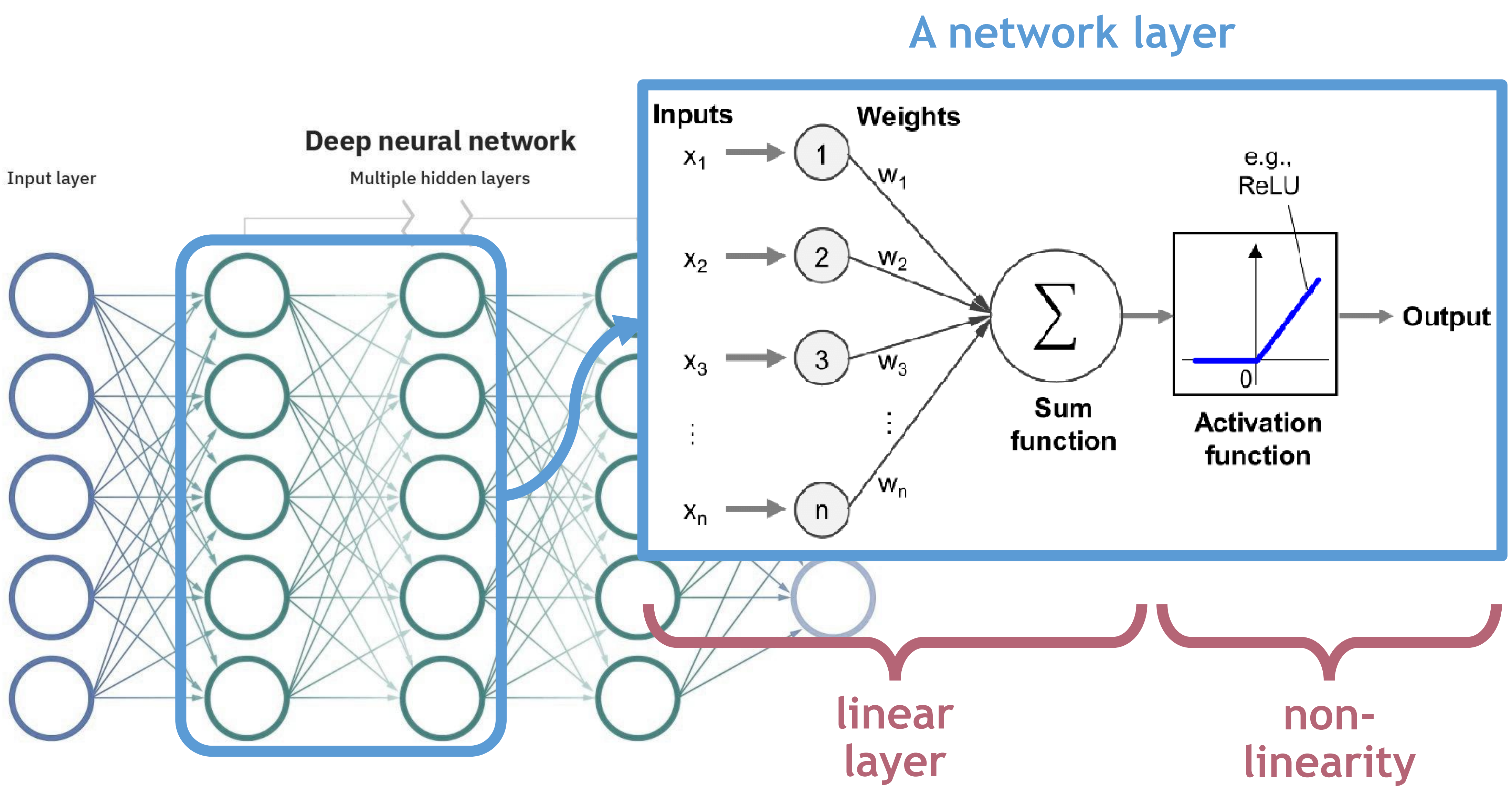
- **Non-linear layer** (with built-in linear layer)
  = **input linear transformation q + non-linearity k**

- Other non-linearities

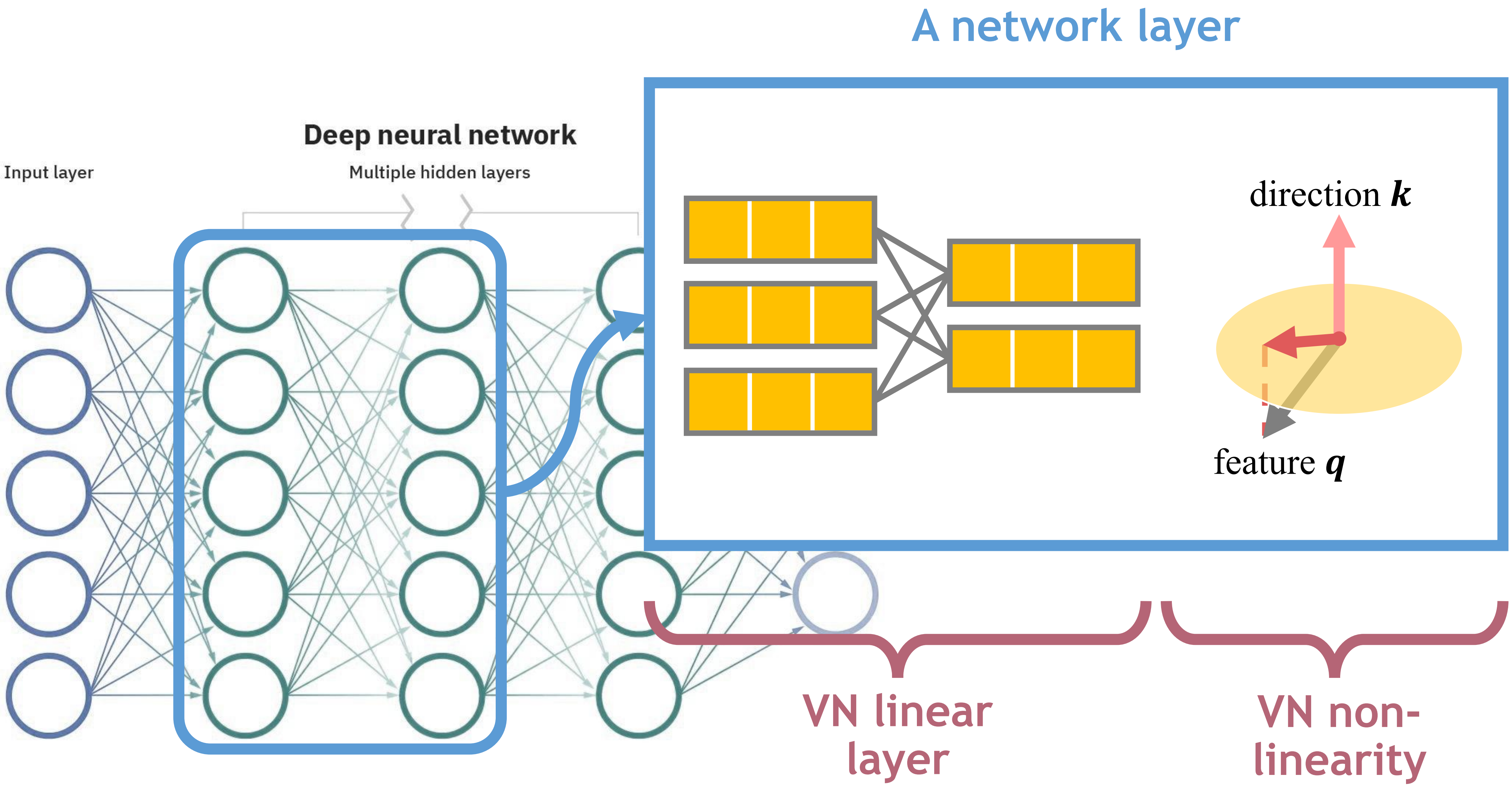


overall structure

Network Layer: Scalar Network

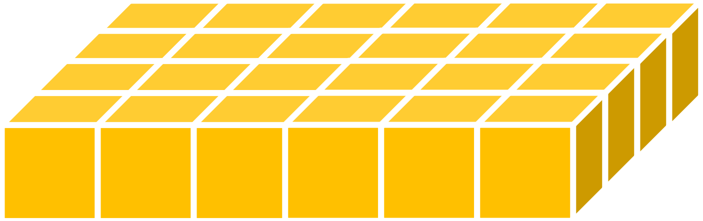


Network Layer: Vector Network



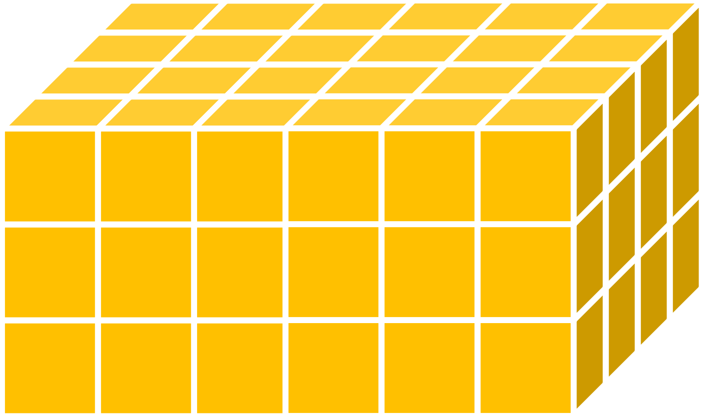
Vector Neuron Features for Point Cloud

Classical:

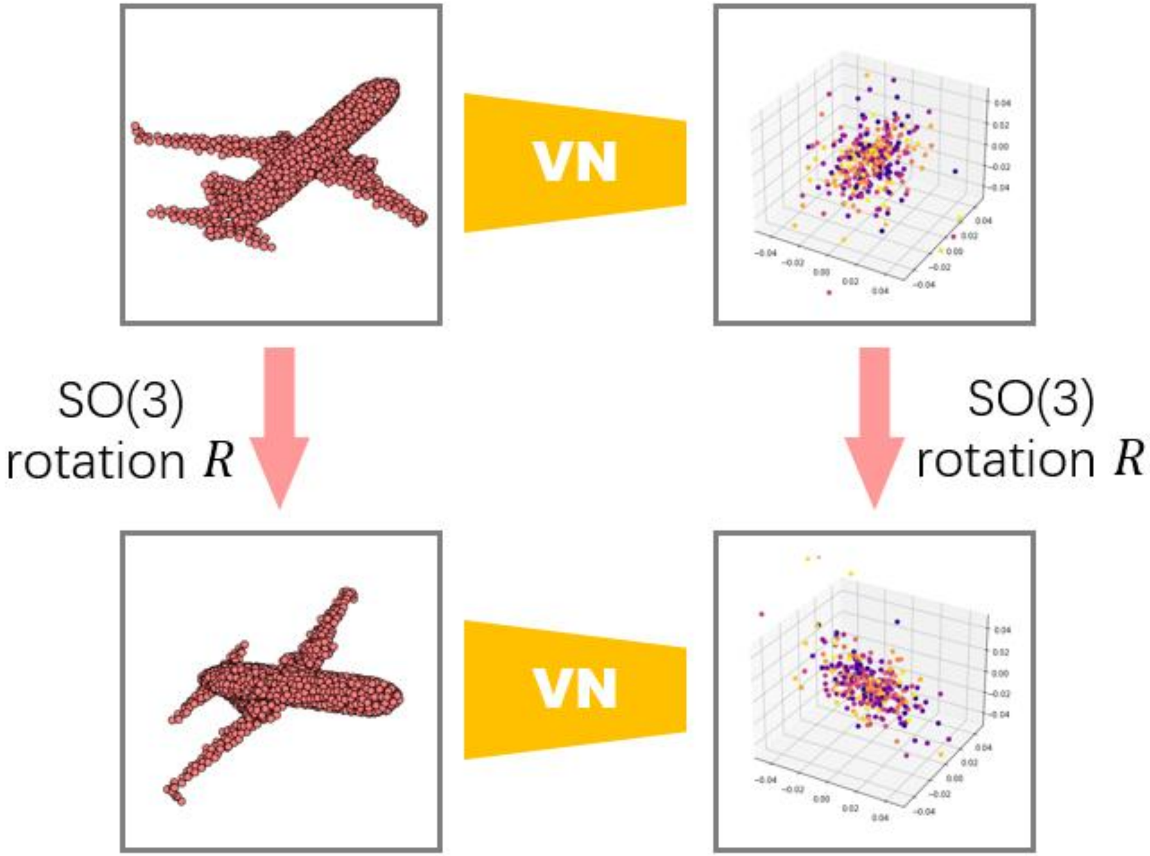


$N \times C \times 1$ feature

VN:



$N \times C \times 3$ feature

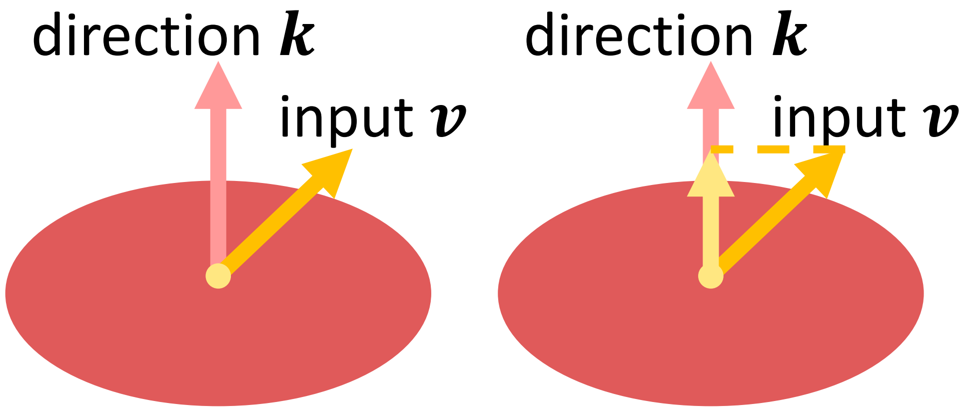


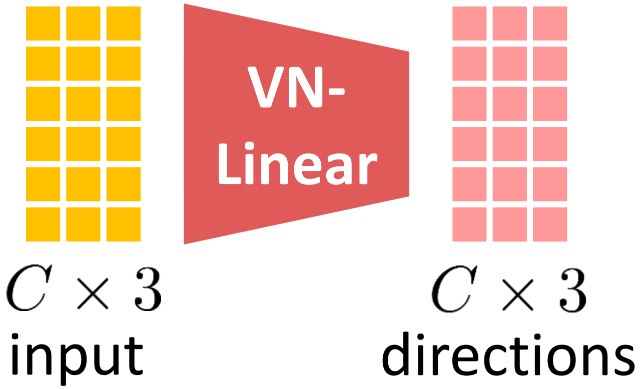
Vector Neuron Pooling

✓ **Mean pooling**

? **Max pooling**

- (Similar to non-linearity)
- argmax alone learned directions



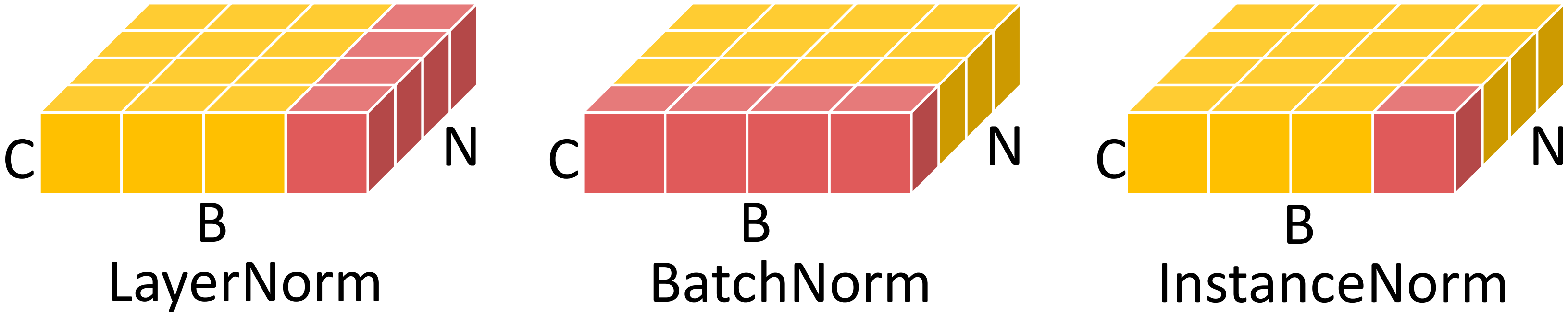


Vector Neuron Normalizations

✓**LayerNorm**

✓**InstanceNorm**

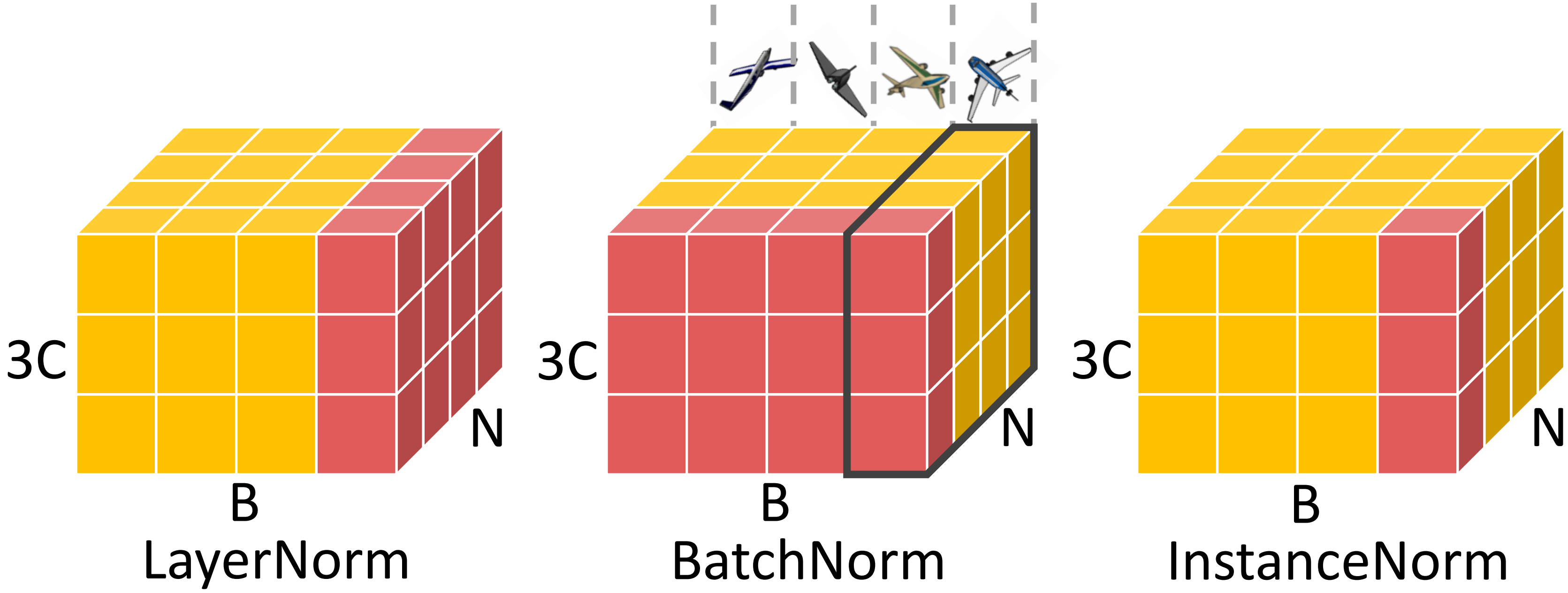
✓**Dropout**



(classical) scalar neurons

? BatchNorm

averaging across arbitrarily rotated inputs would not necessarily be meaningful



Vector neurons

Vector Neuron Normalizations

BatchNorm

- Normalize the **2-norm (invariant component)** of the vector-list feature

$$\boldsymbol{N}_b = \mathrm{ElementWiseNorm}(\boldsymbol{V}_b) \in \mathbb{R}^{N\times 1}$$

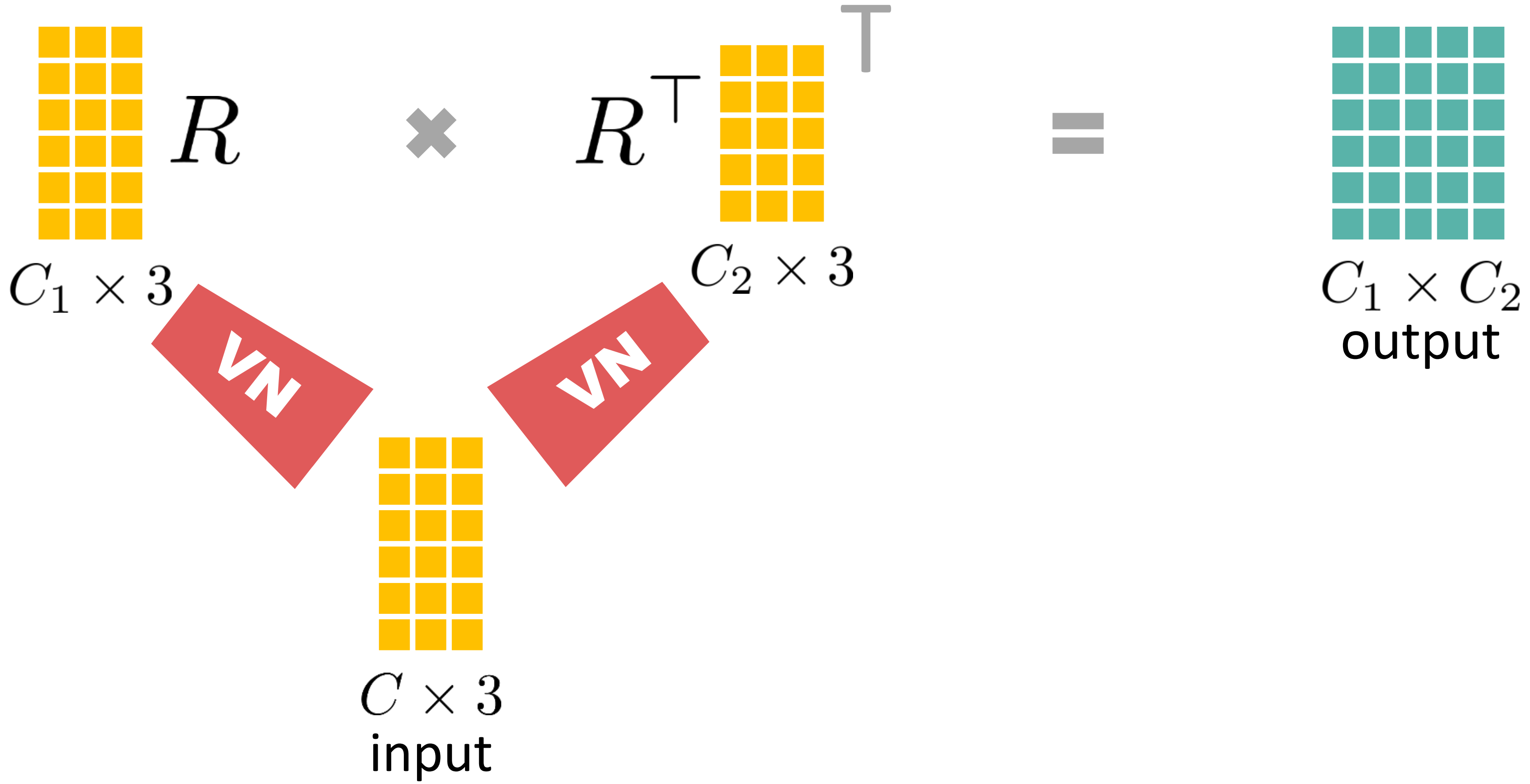
$$\{\boldsymbol{N}'_b\}_{b=1}^{B} = \mathrm{BatchNorm}\left(\{\boldsymbol{N}_b\}_{b=1}^{B}\right)$$

$$\boldsymbol{V}'_b = \boldsymbol{V}_b[c]\frac{\boldsymbol{N}'_b[c]}{\boldsymbol{N}_b[c]} \quad \forall c \in [C]$$

- Element-wise norm: 2-norm for each vector $\boldsymbol{v}_c \in \boldsymbol{V}_b$

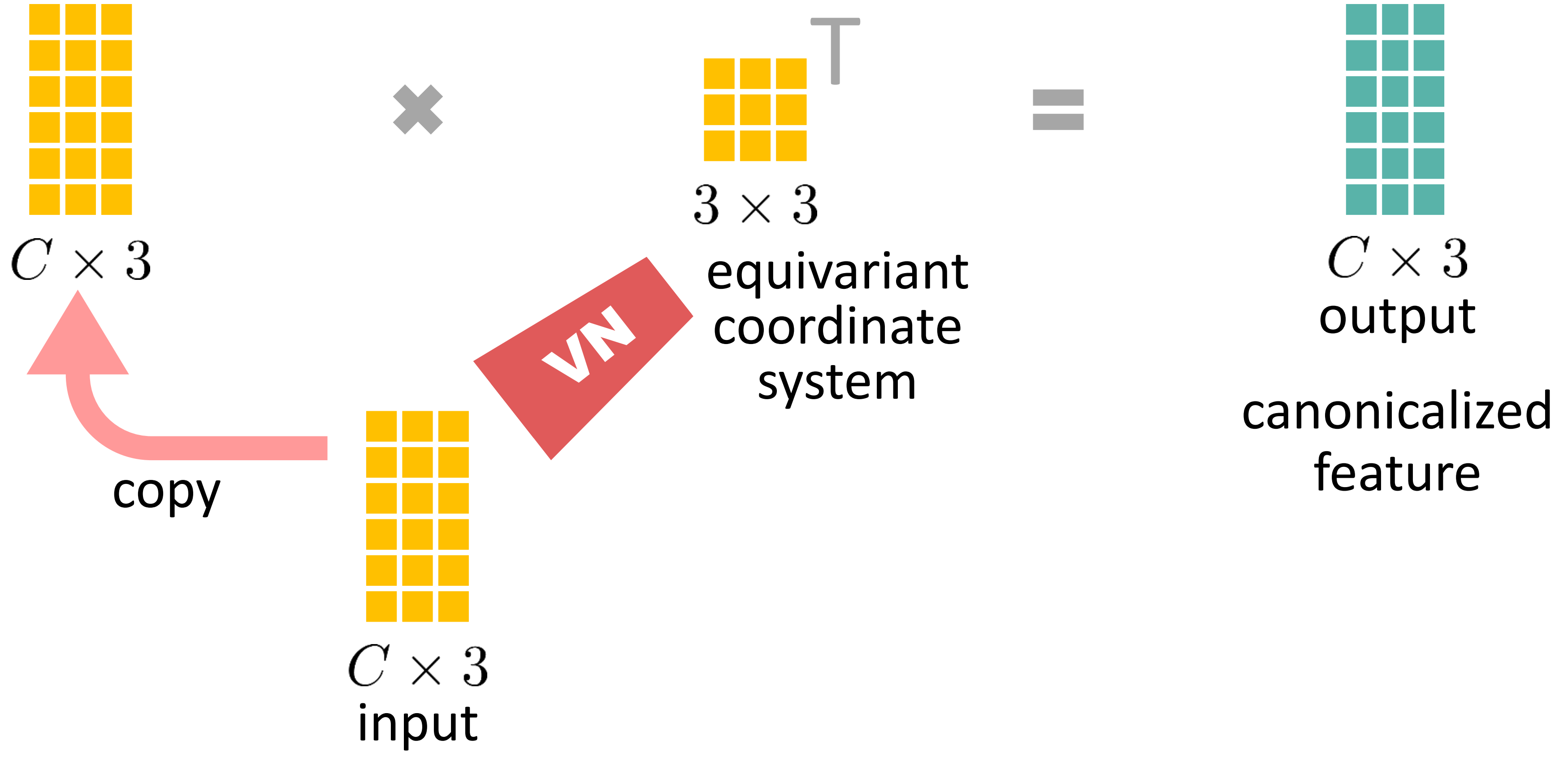
From Equivariance to Invariance

(equivariant feature) × (equivariant feature)$^{\top}$ = (invariant feature)



Vector Neuron Invariant Layer

Specifically…



Vector Neuron Invariant Layer

- Product of an equivariant signal $\boldsymbol{V} \in \mathbb{R}^{C\times 3}$ by the transpose of another equivariant signal $\boldsymbol{T} \in \mathbb{R}^{C'\times 3}$ → invariant signal

- **Special case:** $\boldsymbol{T} \in \mathbb{R}^{3\times 3}$ - an equivariant coordinate system
- For pointcloud, concatenate local feature $\boldsymbol{V} \in \mathbb{R}^{C\times 3}$ with global mean $\overline{\boldsymbol{V}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{V}_n \in \mathbb{R}^{C\times 3}$

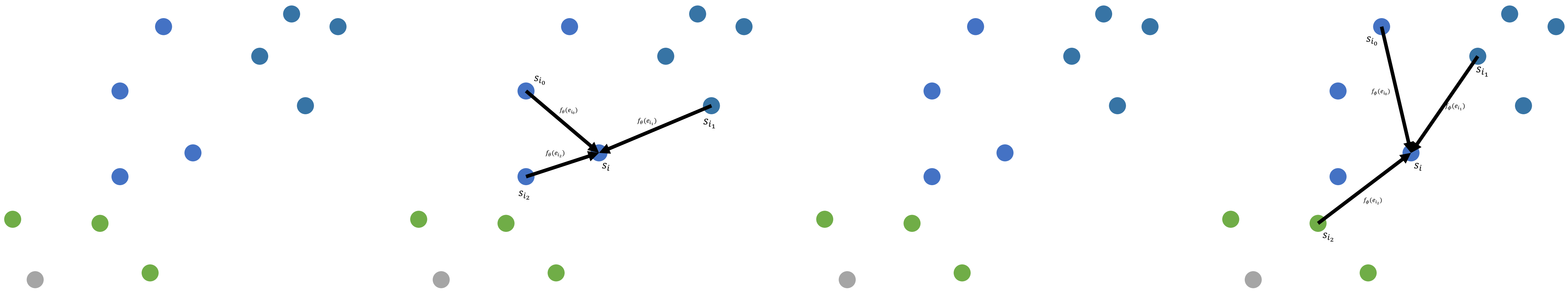
Invariant layer:

$$\boldsymbol{T}_n = \text{VN-MLP}([\boldsymbol{V}_n, \overline{\boldsymbol{V}}])$$

$$\text{VN-In}(\boldsymbol{V}_n) = \boldsymbol{V}_n \boldsymbol{T}_n^\top$$

Vectorize Classical 3D Networks: DGCNN, PointNet

Dynamic Graph CNN (DGCNN)

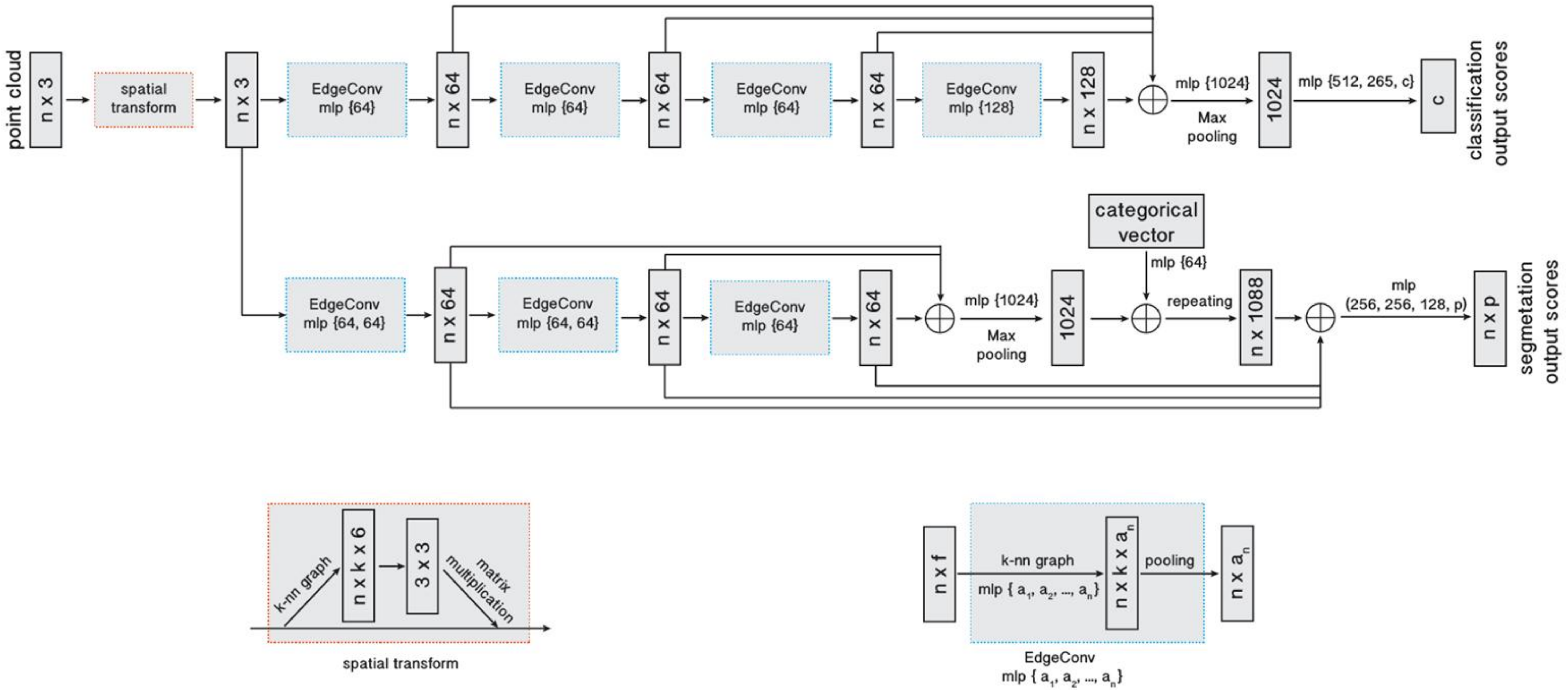


DGCNN alternates feature learning (EdgeConvs) and graph NN reconstruction

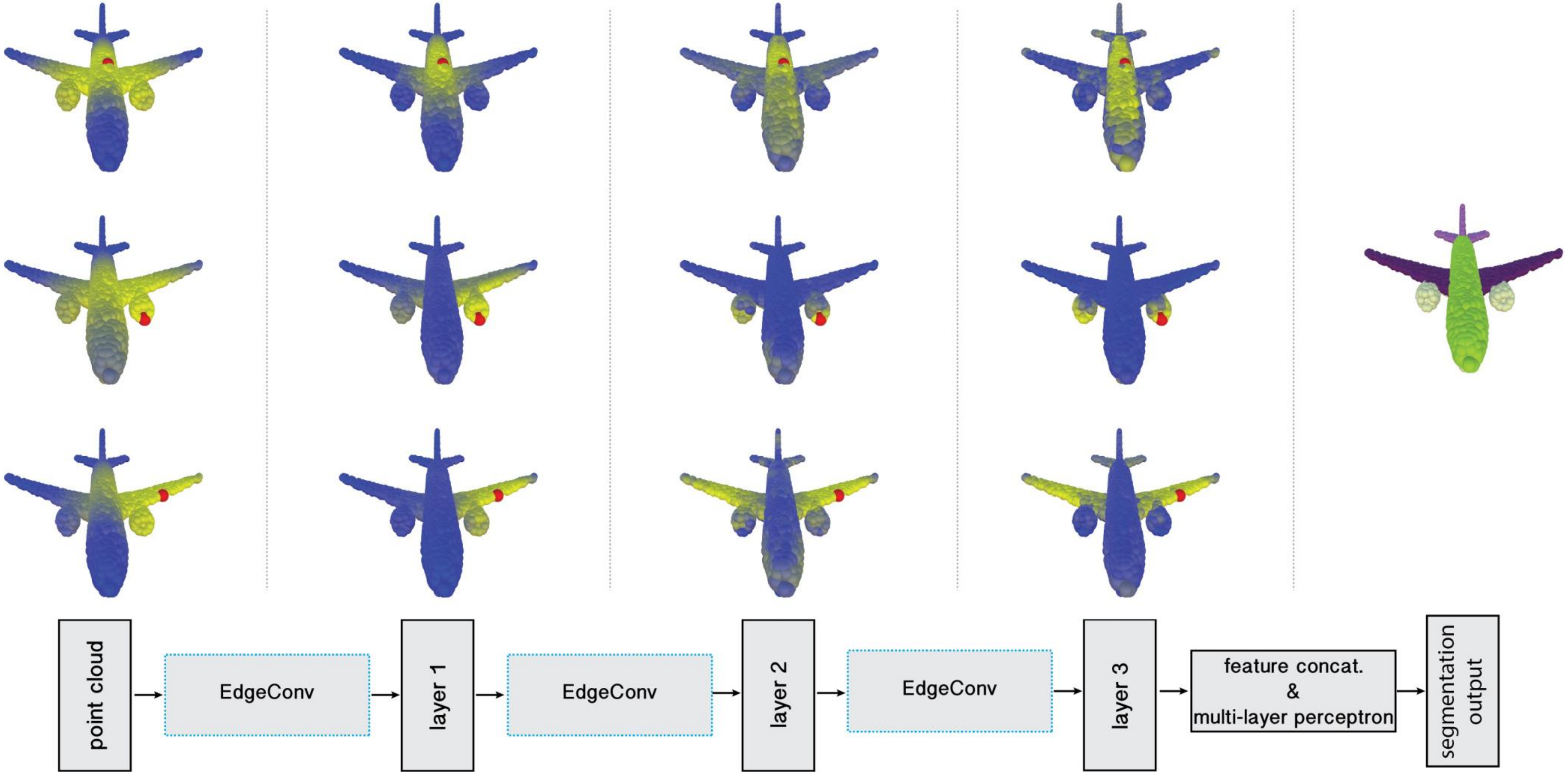
[Wang et al., TOG 2019]

DGCNN Architecture: Alternating Processing

EdgeConv: Edge Convolutions



Dynamic Graph CNN (DGCNN)



Build VN Networks: VN-DGCNN

DGCNN

Edge feature: $'_{nm} = \mathrm{ReLU}(\Theta(\;_m - \;_n) + \Phi\;_n)$

Aggregation: $'_{n} = \mathrm{Pool}_{m:(n,m)\in\mathcal{E}}(\;'_{nm})$

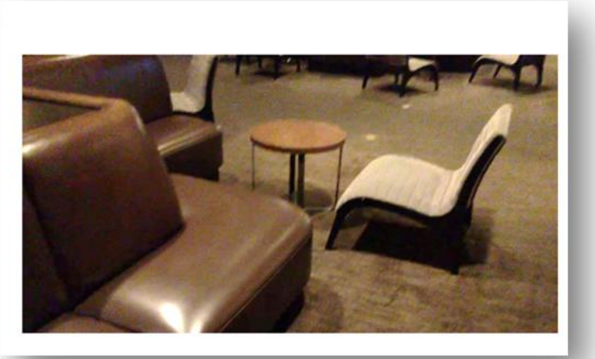
VN-DGCNN

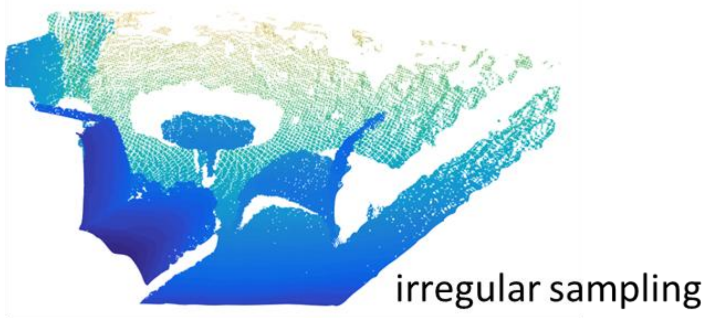
Edge feature: $'_{nm} = \text{VN-ReLU}(\Theta(\;_m - \;_n) + \Phi\;_n)$

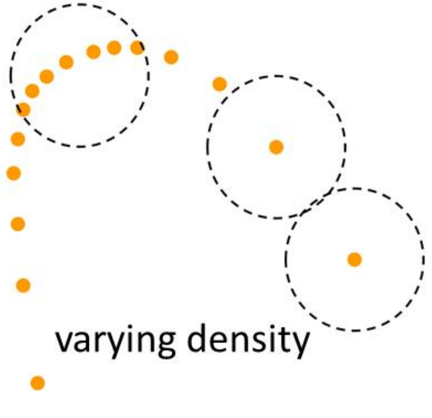
Aggregation $'_{n} = \text{VN-Pool}_{m:(n,m)\in\mathcal{E}}(\;'_{nm})$

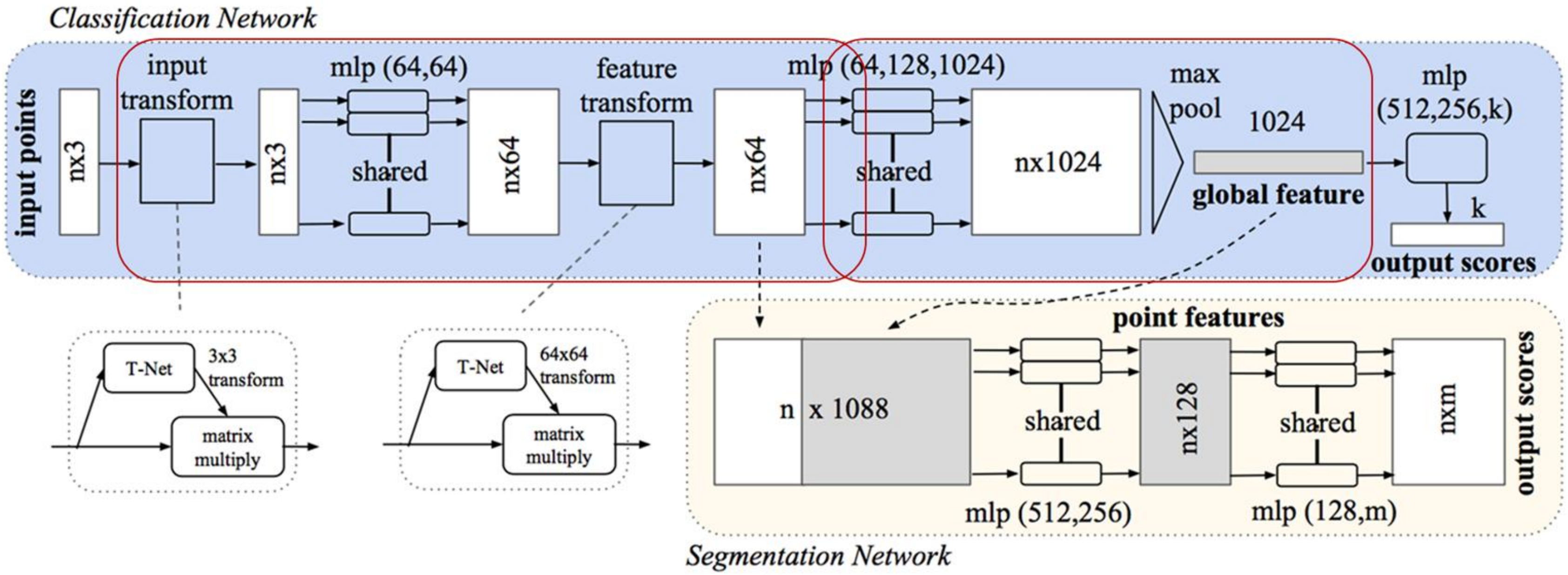
[Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, TOG 2019]

Deep Architectures: PointNet and PointNet++









Vanilla PointNet

Build VN Networks: VN-PointNet

PointNet

$$'\ = \mathrm{Pool}_{\boldsymbol{x}_n \in \mathcal{X}}\left(h(\ _1), h(\ _2) \cdots, h(\ _N)\right)$$

VN-PointNet

$$'\ = \text{VN-Pool}_{\boldsymbol{V}_n \in \mathcal{V}}\left(f(\ _1), f(\ _2), \cdots, f(\ _N)\right)$$

[C. R. Qi, H. Su, K. Mo, L. J. Guibas, CVPR 2017]

Experiments of VN Use

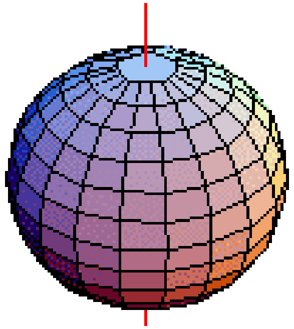
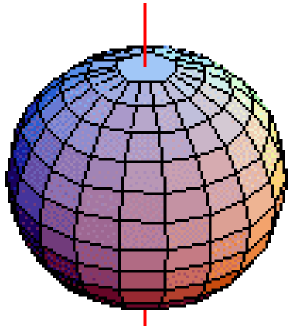
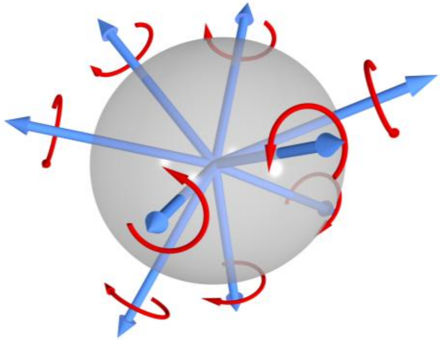
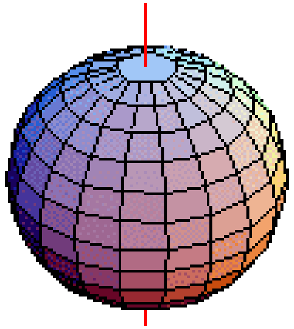
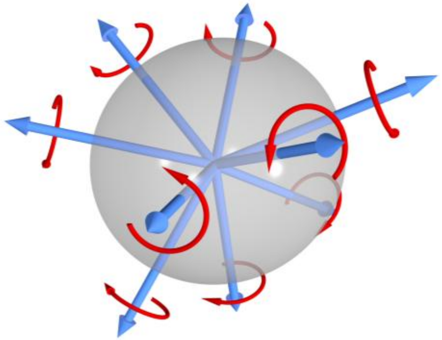
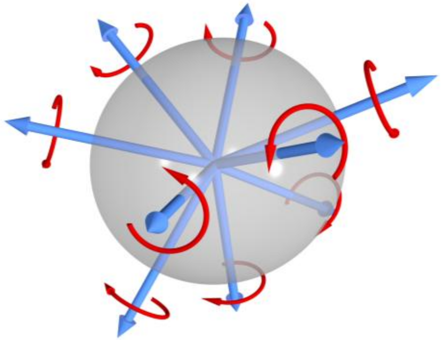
	z/z	z/SO(3)	SO(3)/SO(3)
Train			
Test			

Figure: https://quantum-journal.org/papers/q-2020-06-22-285/

Classification

Results on ModelNet40

Methods	z/z	$z/SO(3)$	$SO(3)/SO(3)$
Point / mesh inputs			
PointNet [25]	85.9	19.6	74.7
DGCNN [35]	90.3	33.8	88.6
VN Networks			
VN-PointNet	77.5	77.5	77.2
VN-DGCNN	89.5	**89.5**	**90.2**
Rotation sensitive methods			
PCNN [2]	92.3	11.9	85.1
ShellNet [40]	**93.1**	19.9	87.8
PointNet++ [26]	91.8	28.4	85.0
PointCNN [20]	92.5	41.2	84.5
Spherical-CNN [11]	88.9	76.7	86.9
a^3S-CNN [21]	89.6	87.9	88.7
Rotation robust methods			
SFCNN [27]	91.4	84.8	90.1
TFN [32]	88.5	85.3	87.6
RI-Conv [39]	86.5	86.4	86.4
SPHNet [24]	87.7	86.6	87.6
ClusterNet [6]	87.1	87.1	87.1
GC-Conv [41]	89.0	89.1	89.2
RI-Framework [18]	89.4	89.4	89.3

Classification

Results on ModelNet40 (%)

- VN networks are **robust to (seen & unseen) rotations**
- Excellent performance compared with other methods
- **SO(3)/SO(3):** equivariance by construction is better than rotation augmentation

Methods	z/z	z/SO(3)	SO(3)/SO(3)
Point / mesh inputs			
PointNet [25]	85.9	19.6	74.7
DGCNN [35]	90.3	33.8	88.6
VN-PointNet	77.5	77.5	77.2
VN-DGCNN	89.5	**89.5**	**90.2**
PCNN [2]	92.3	11.9	85.1
ShellNet [40]	**93.1**	19.9	87.8
PointNet++ [26]	91.8	28.4	85.0
PointCNN [20]	92.5	41.2	84.5
Spherical-CNN [11]	88.9	76.7	86.9
a^3S-CNN [21]	89.6	87.9	88.7
SFCNN [27]	91.4	84.8	90.1
TFN [32]	88.5	85.3	87.6
RI-Conv [39]	86.5	86.4	86.4
SPHNet [24]	87.7	86.6	87.6
ClusterNet [6]	87.1	87.1	87.1
GC-Conv [41]	89.0	89.1	89.2
RI-Framework [18]	89.4	89.4	89.3

Part Segmentation

Results on ShapeNet (mIoU)

- Similarly…

VN Networks

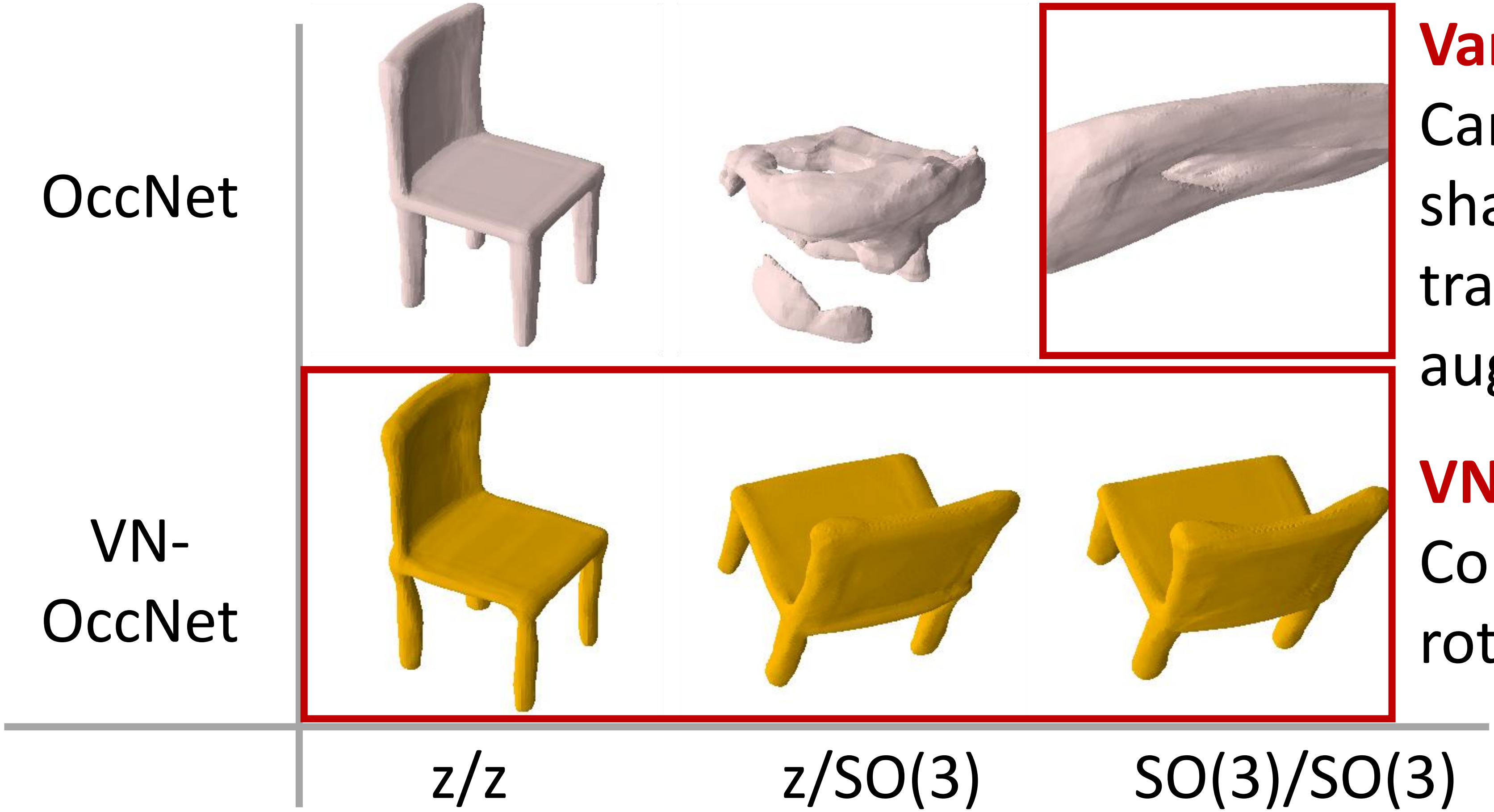
Rotation sensitive methods

Rotation robust methods

Methods	z/SO(3)	SO(3)/SO(3)
Point / mesh inputs		
PointNet [24]	38.0	62.3
DGCNN [34]	49.3	78.6
VN-PointNet	72.4	72.8
VN-DGCNN	**81.4**	**81.4**
PointCNN [19]	34.7	71.4
PointNet++ [25]	48.3	76.7
ShellNet [39]	47.2	77.1
RI-Conv [38]	75.3	75.3
TFN [31]	76.8	76.2
GC-Conv [40]	77.2	77.3
RI-Framework [17]	79.2	79.4

Neural Implicit Reconstruction

Results on ShapeNet (Examples)

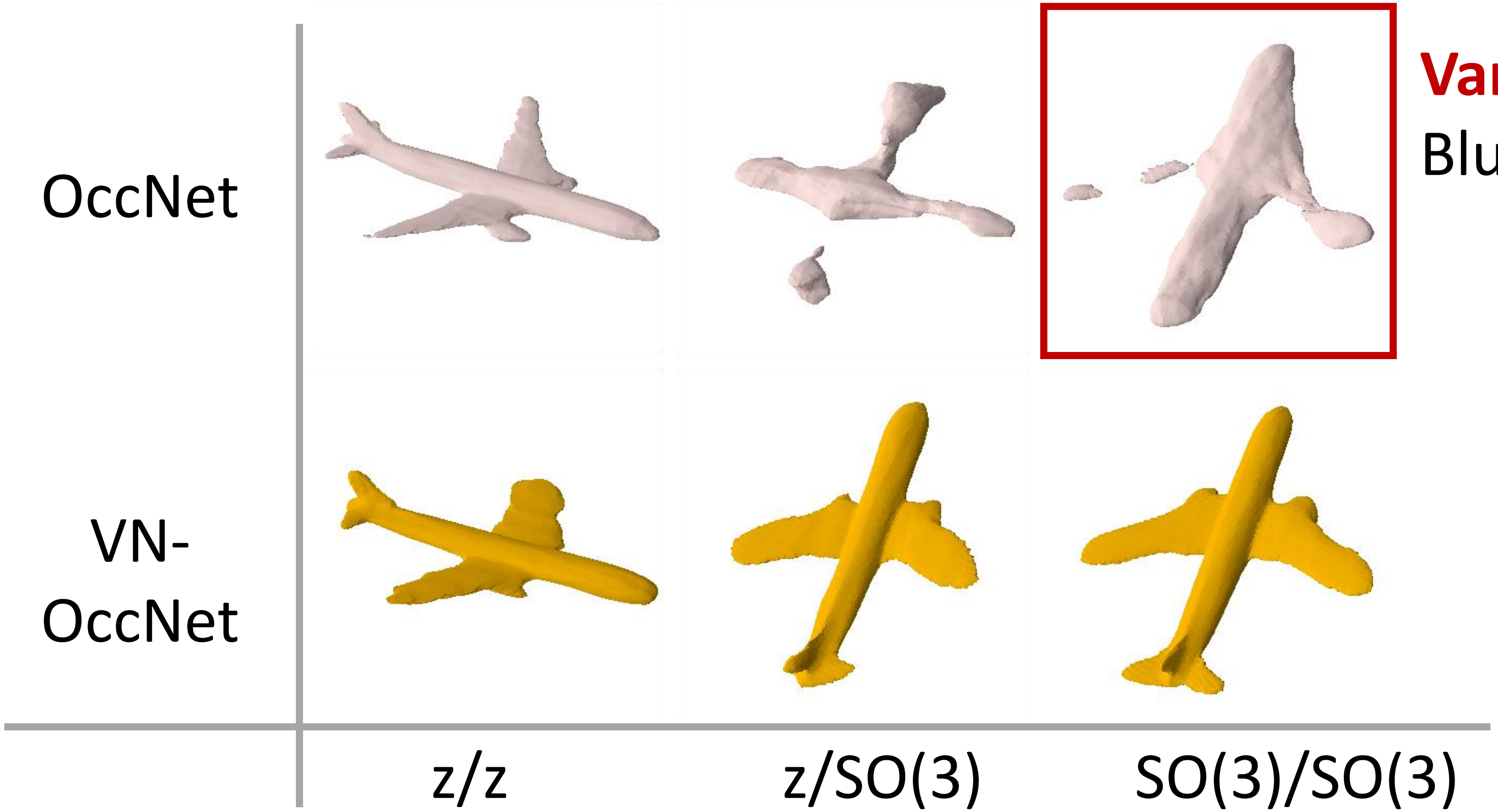


Vanilla OccNet:
Can't learn rotated shapes even when trained with augmentation

VN-OccNet:
Consistent across rotations

Neural Implicit Reconstruction

Results on ShapeNet (Examples)



Vanilla OccNet:
Blurry shapes

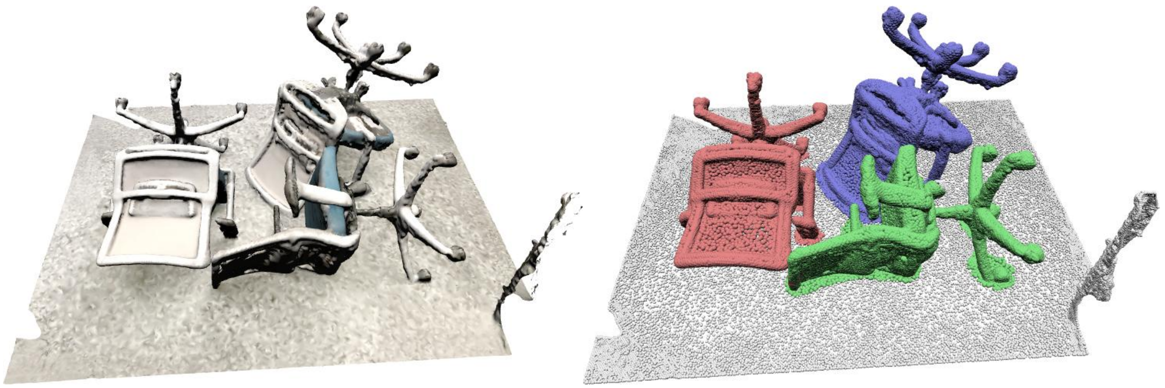
Applications of Vector Neurons

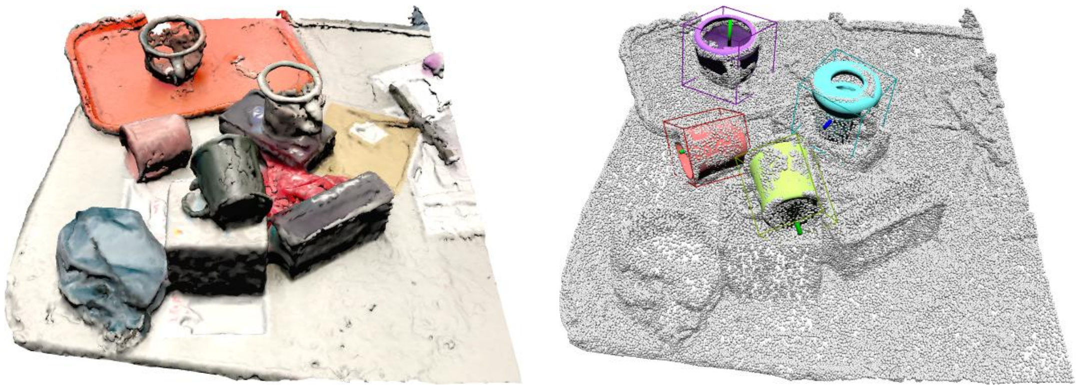
CVPR 2023

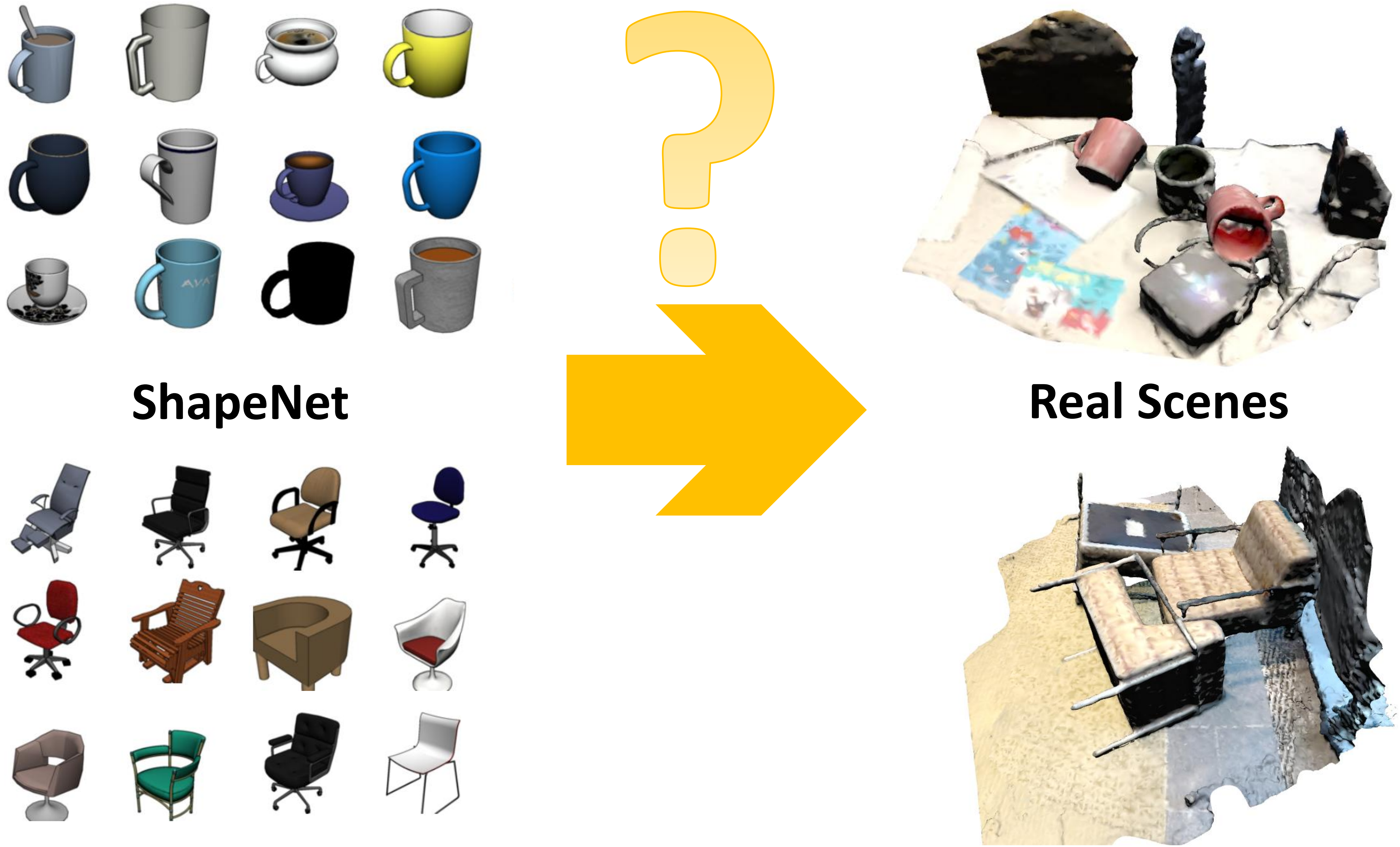
EFEM

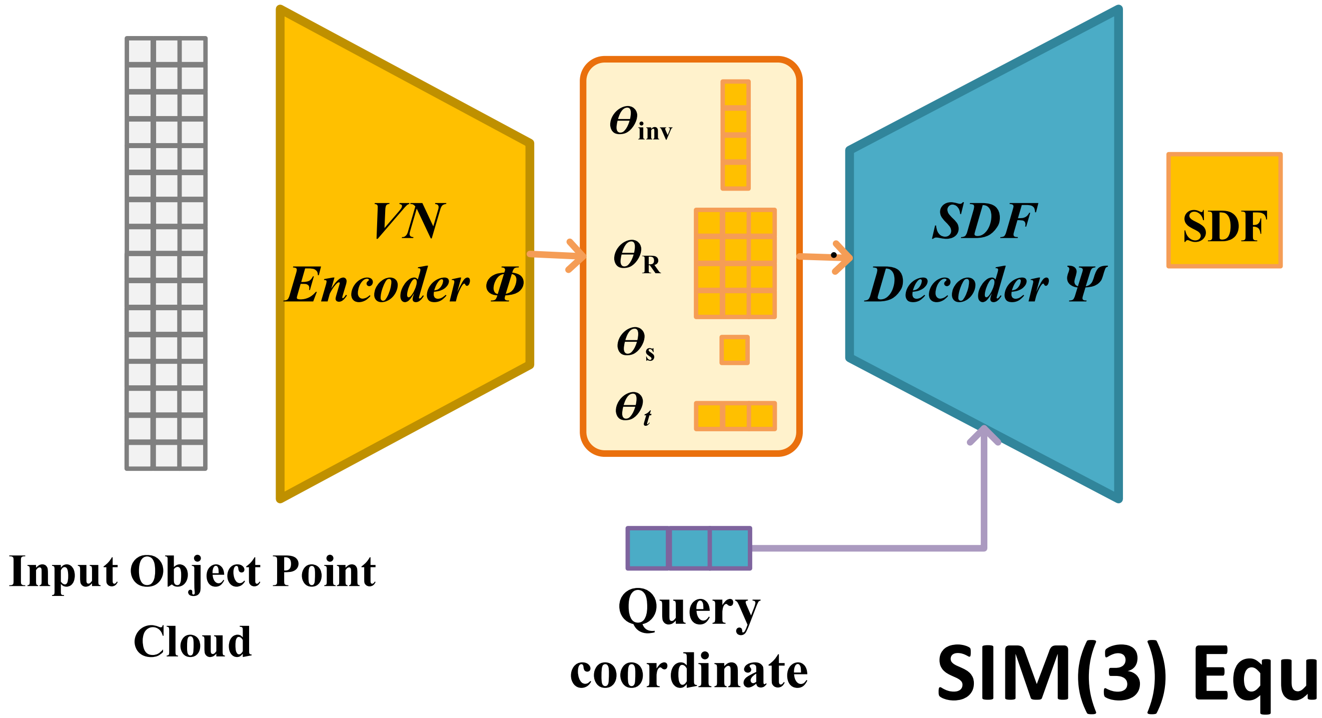
Equivariant neural ***F***ield ***E***xpectation ***M***aximization for 3D Object Segmentation **Without** Scene Supervision

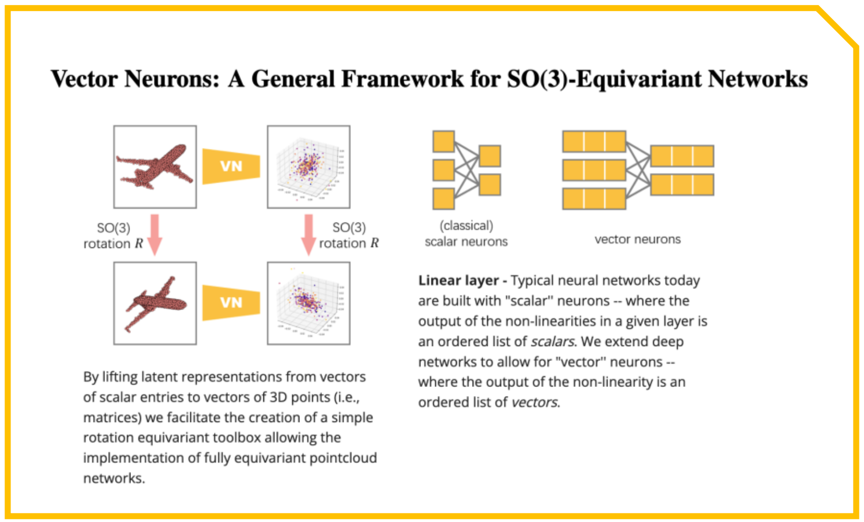
Jiahui Lei[1] Congyue Deng[2] Karl Schmeckpeper[1] Leonidas Guibas[2] Kostas Daniilidis[1]
[1] University of Pennsylvania [2] Stanford University
{leijh, karls, kostas}@cis.upenn.edu, {congyue, guibas}@cs.stanford.edu






ShapeNet
Real Scenes



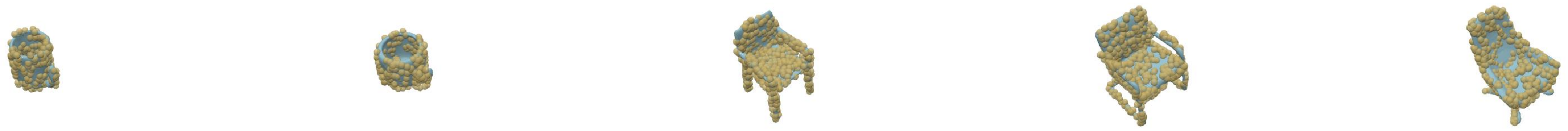


Vector Neurons: A General Framework for SO(3)-Equivariant Networks

By lifting latent representations from vectors of scalar entries to vectors of 3D points (i.e., matrices) we facilitate the creation of a simple rotation equivariant toolbox allowing the implementation of fully equivariant pointcloud networks.

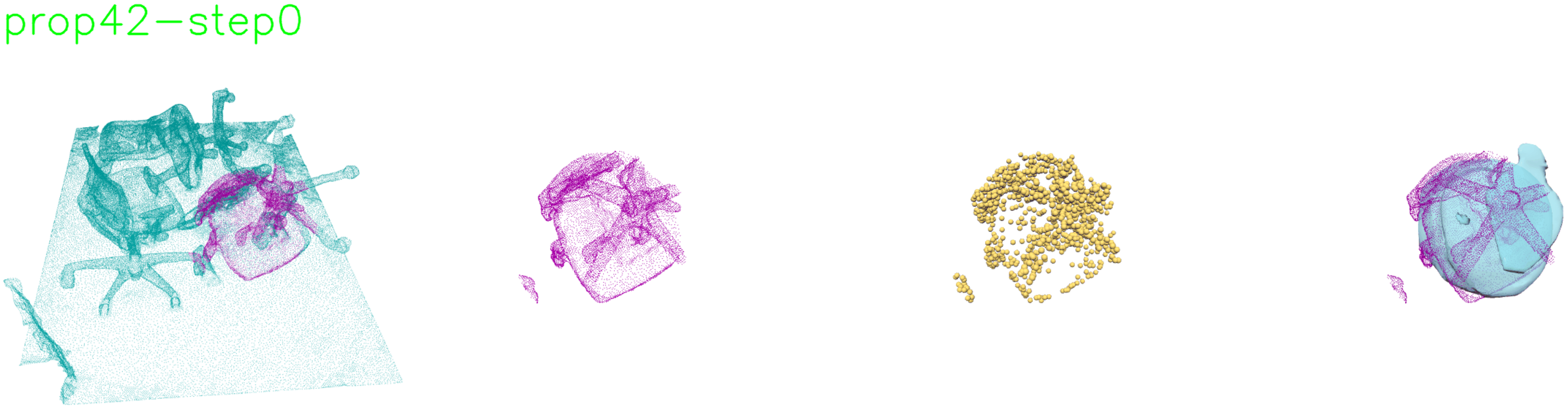
Linear layer - Typical neural networks today are built with "scalar" neurons -- where the output of the non-linearities in a given layer is an ordered list of *scalars*. We extend deep networks to allow for "vector" neurons -- where the output of the non-linearity is an ordered list of *vectors*.

SIM(3) Equivariant Shape Prior

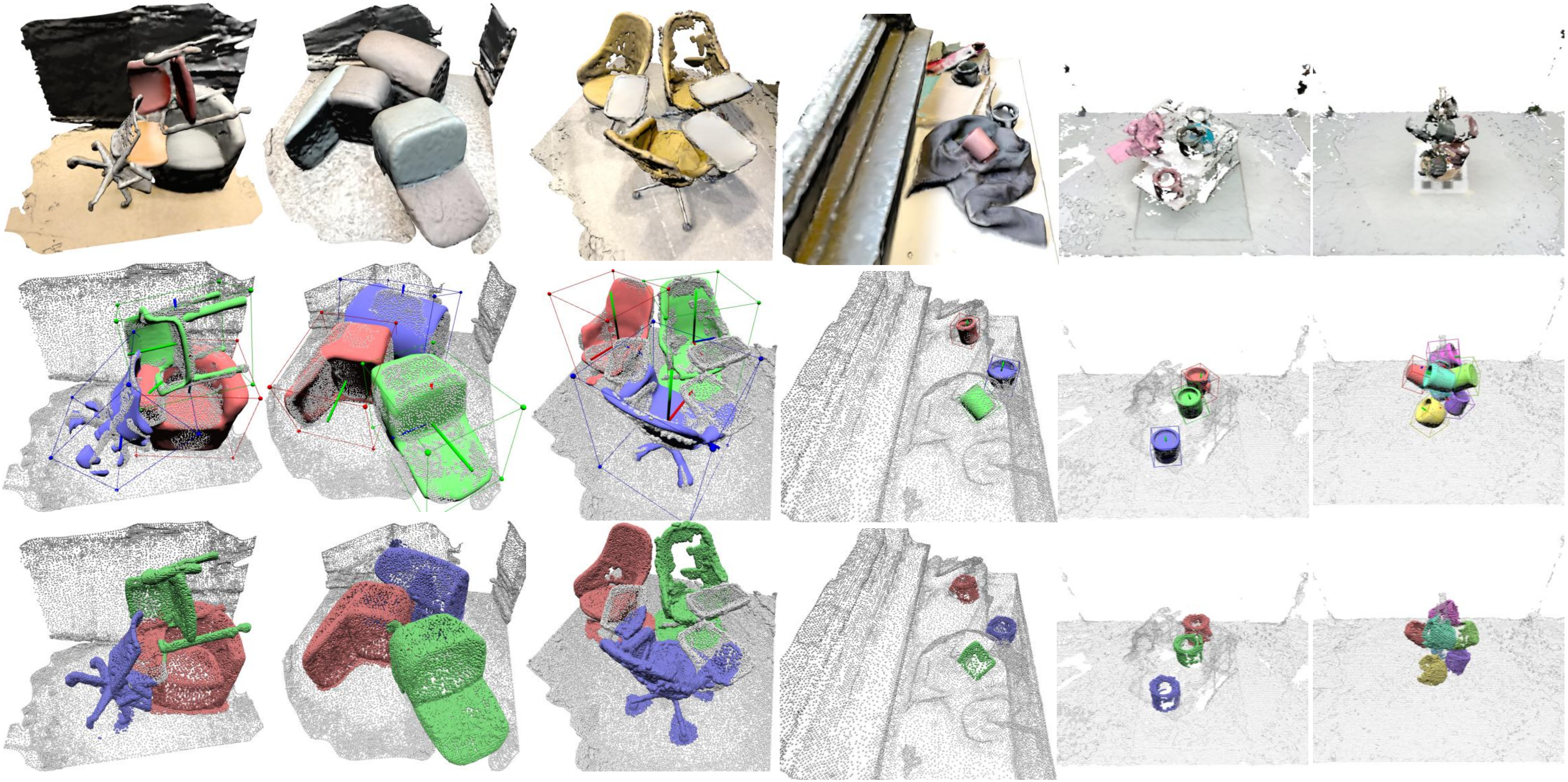


Output meshes follow input transformations

E-M iterative refinement



Chairs and Mugs

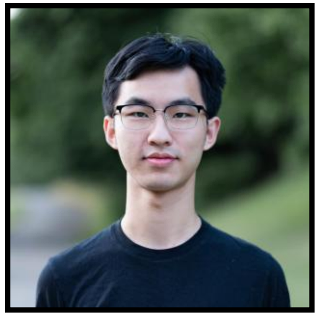
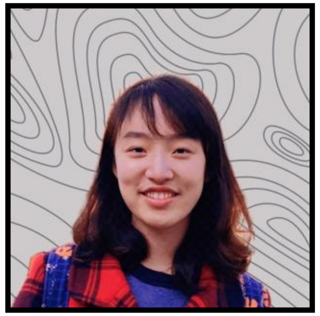
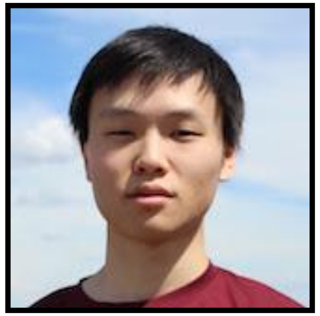


NeurIPS 2023

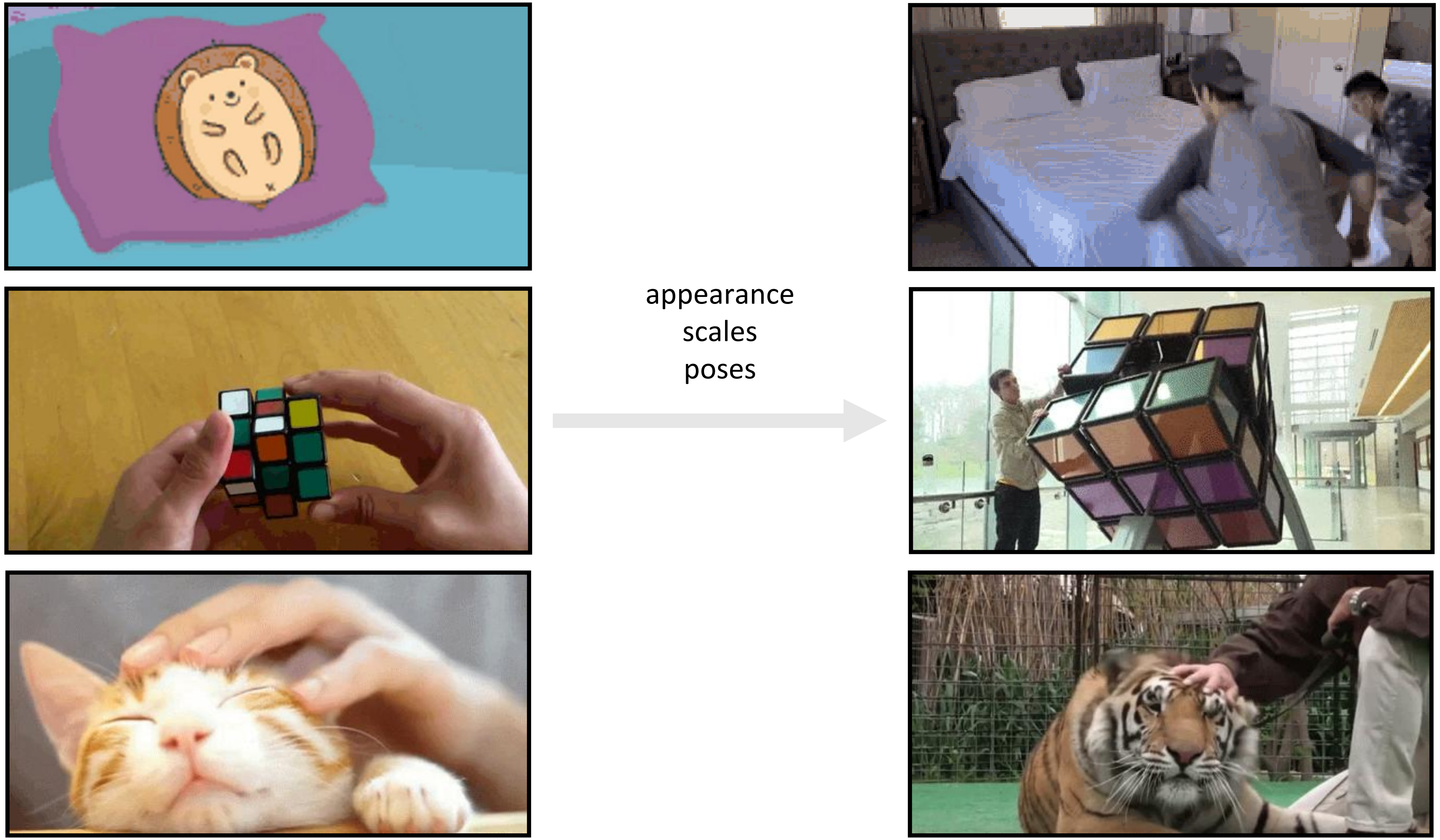
EquivAct: SIM(3)-Equivariant Visuomotor Policies beyond Rigid Object Manipulation

Jingyun Yang[1*] Congyue Deng[1*] Jimmy Wu[2] Rika Antonova[1] Leonidas Guibas[1] Jeannette Bohg[1]

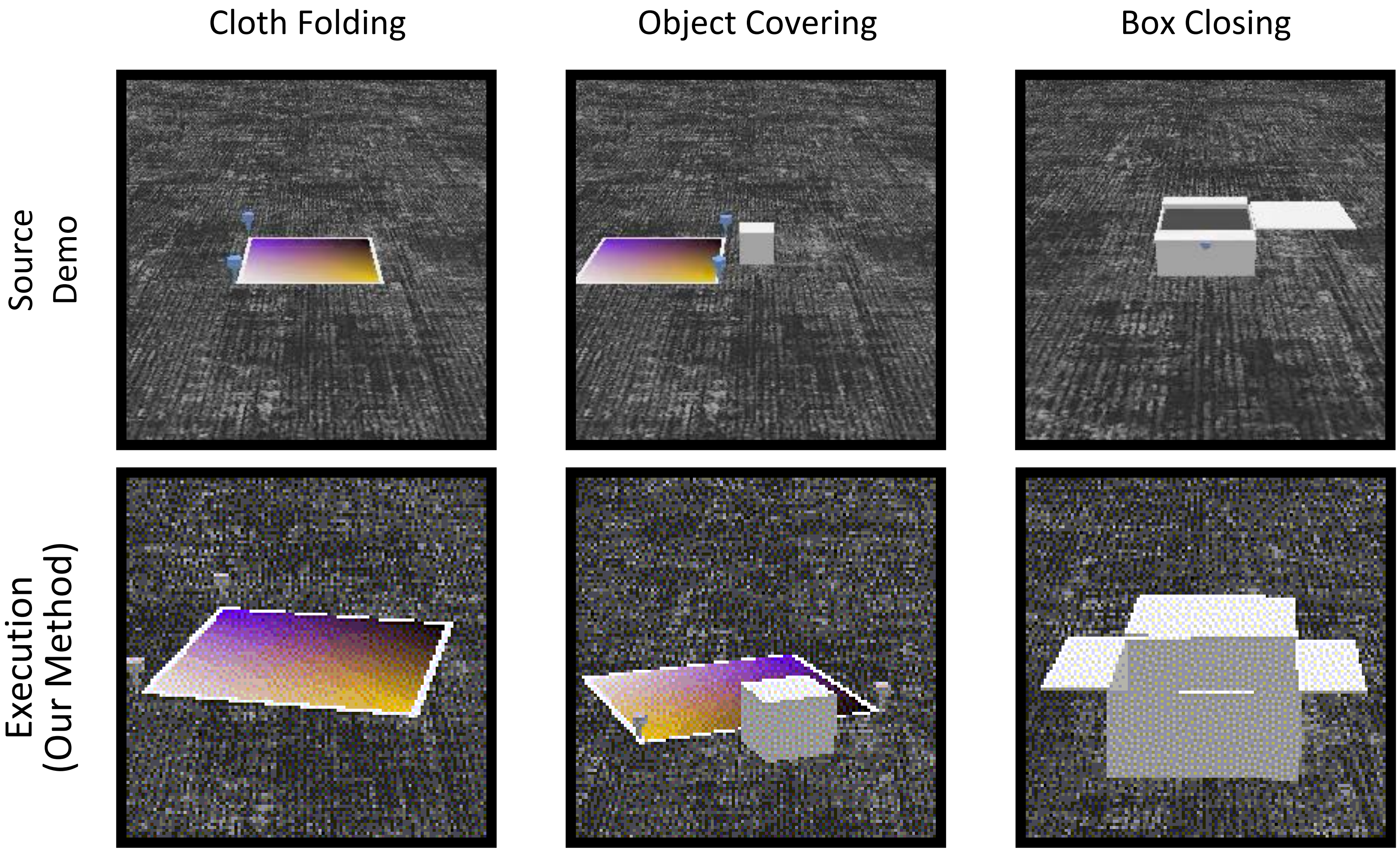
Stanford University[1] Princeton University[2]

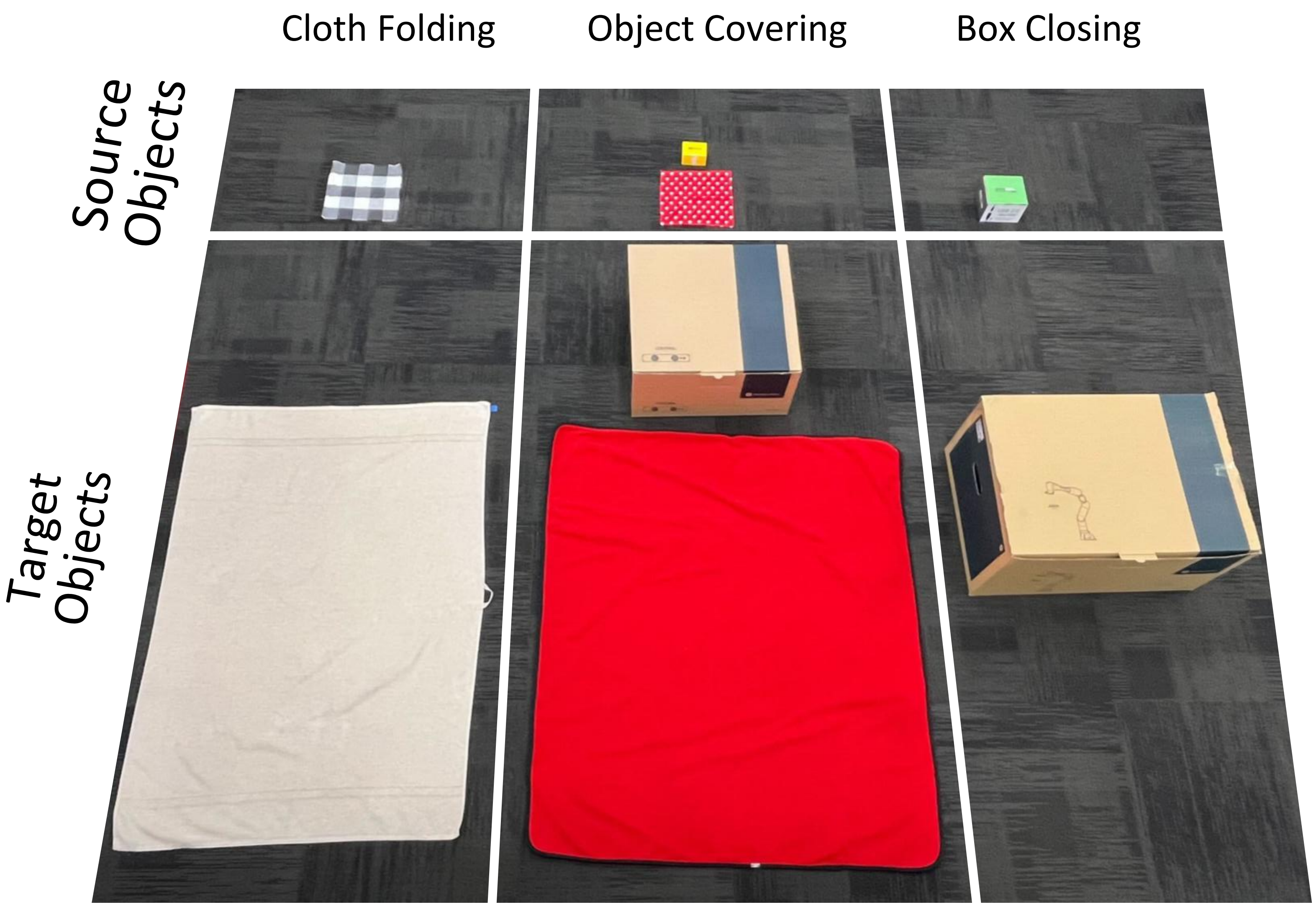





QUESTION ADDRESSED IN THIS WORK

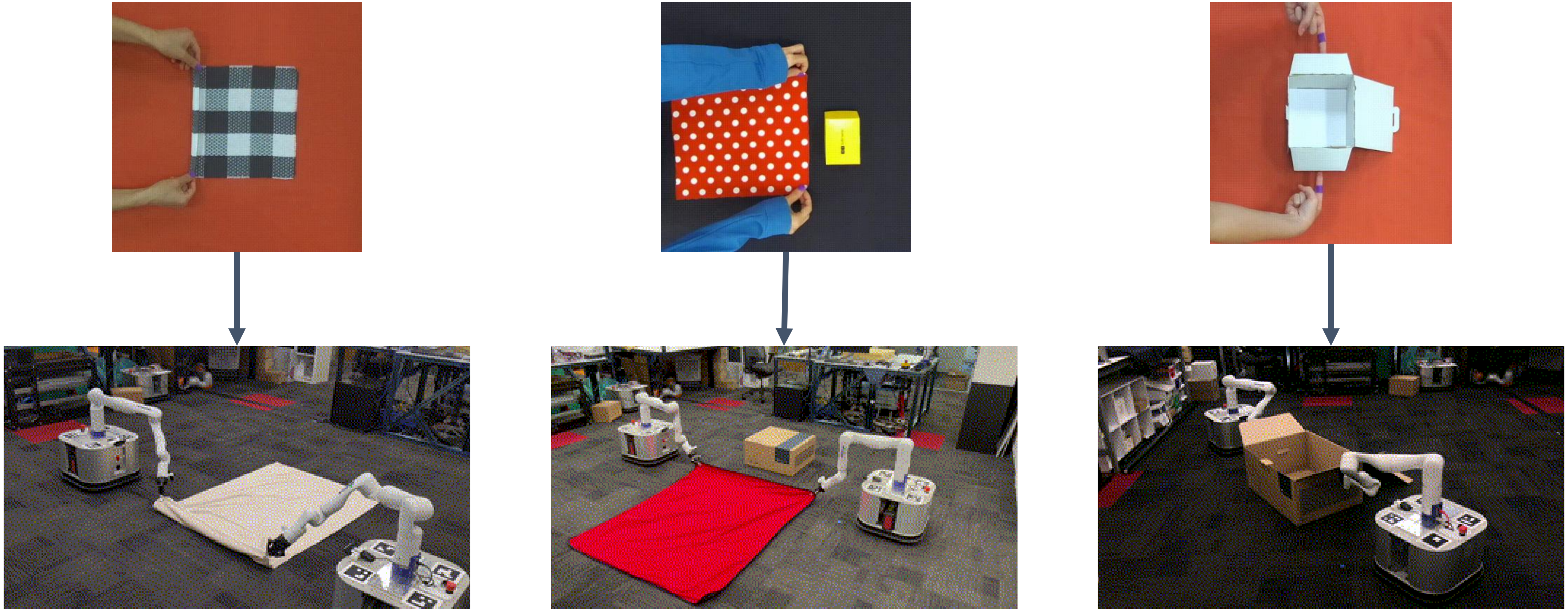


How can robots learn from a few example trajectories and generalize to scenarios with unseen visuals, scales, and poses?


Cloth Folding
Object Covering
Box Closing
Source Demo
Execution (Our Method)

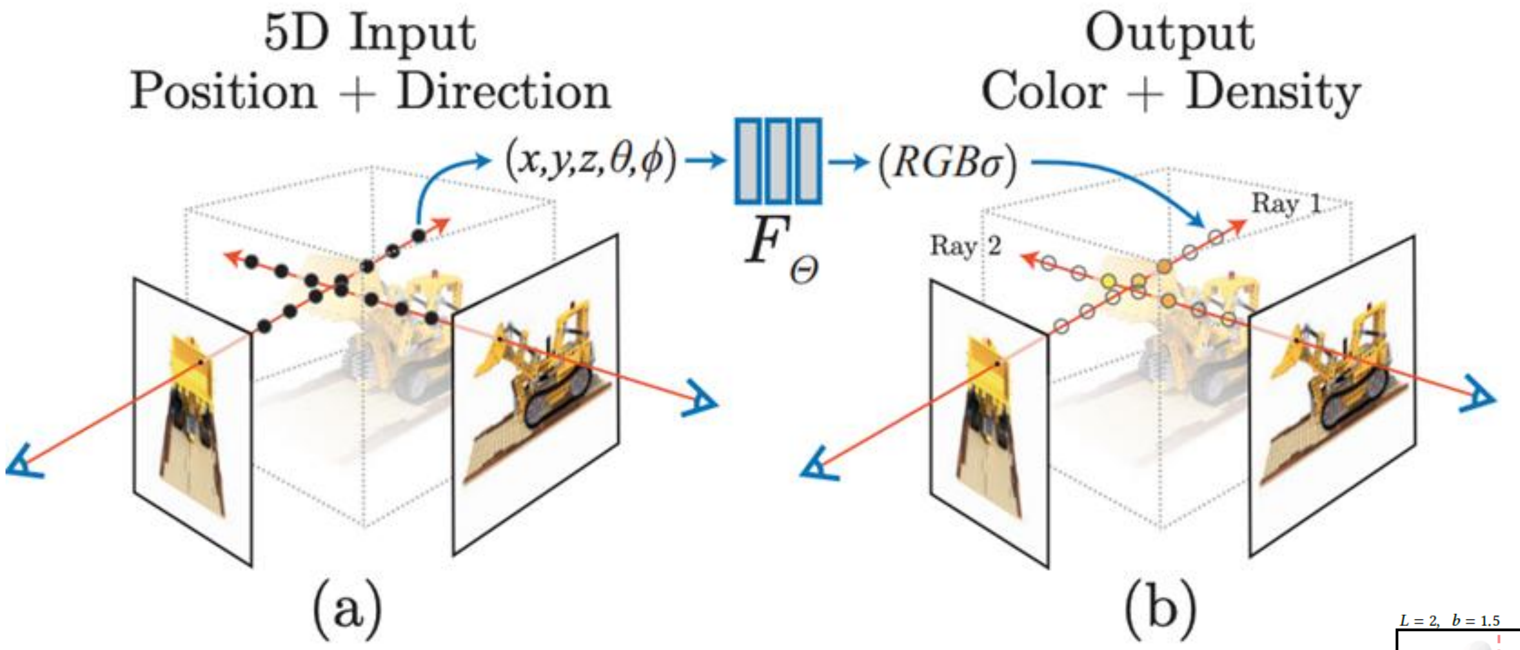

Cloth Folding
Object Covering
Box Closing
Source Objects
Target Objects

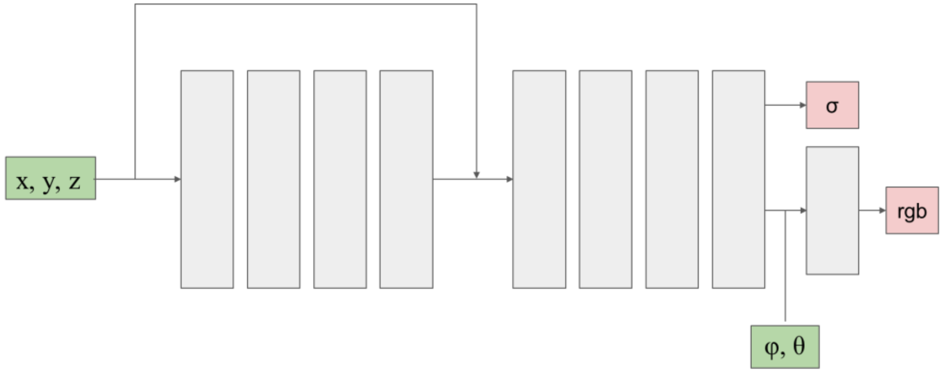
Applications in Robot Manipulation



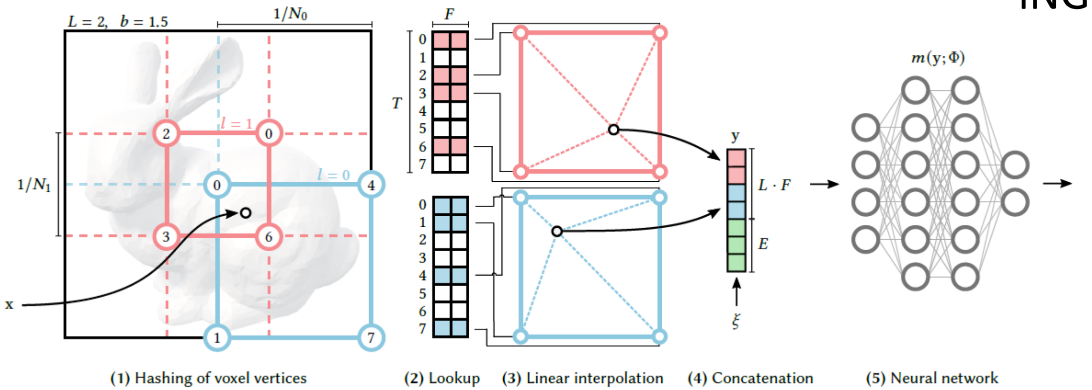
Learned Priors: Semantic Structure and Compositionality in 3D Scenes

Neural Radiance Fields (NeRFs) for 3D Scenes





iNGP

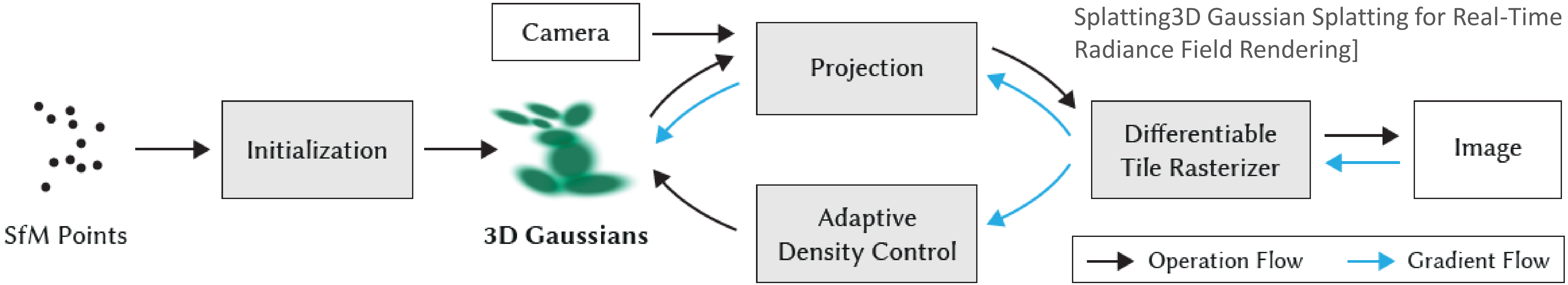


Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2021). NeRF: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1), 99-106.

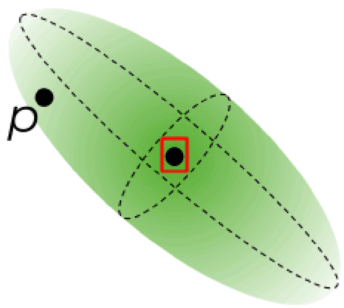
Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG), 41(4), 1-15.

Gaussian Scene Representations: Gaussian Splatting

[From: A Comprehensive Overview of Gaussian Splatting3D Gaussian Splatting for Real-Time Radiance Field Rendering]



$$f_i(p) = \sigma(\alpha_i) \exp(-\frac{1}{2}(p - \boxed{\mu_i})\Sigma_i^{-1}(p - \boxed{\mu_i}))$$



GS Reps (and NeRFs, too) are Unstructured

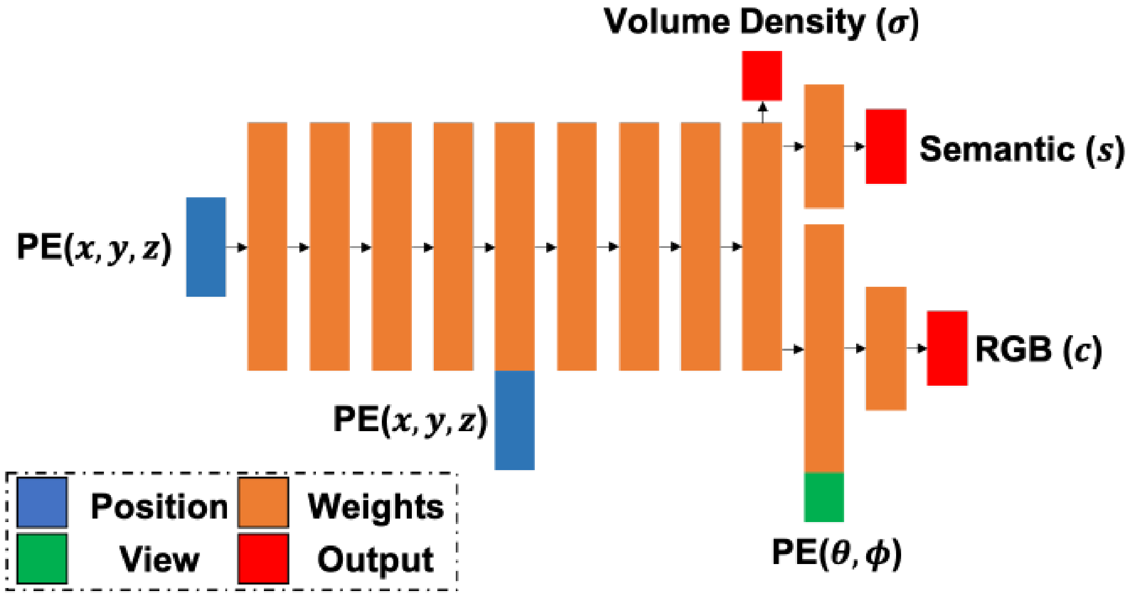
Just millions of individual Gaussians…

How can structure them?

How can we manipulate the scene at the object / entity level?

NeRFs / GSs with Semantic Channels

Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, Andrew J. Davison.
In-Place Scene Labelling and Understanding with Implicit Scene Representation.
ICCV 2021.



Semantic NeRF

Sosuke Kobayashi, Eiichi Matsumoto, Vincent Sitzmann.
Decomposing NeRF for Editing via Feature Field Distillation.
NeurIPS 2022

NeRF Shaping with Sparse Semantic Supervision

Xiamen Xu, Yanchao Yang, Kaichun Mo, Boxiao Pan, Li Yi, Leonidas Guibas.
JacobiNeRF: NeRF Shaping with Mutual Information Gradients
(CVPR 2023).

Supervision for NeRFs: 1^{st} Order vs 2^{nd} Order

- In typical NeRF training, we supervise via pixel values in views (1^{st} order info):
  - this pixel's color is red …
  - this pixel's semantics is "car" …

- But we can also supervise with value relationships (2^{nd} order info)
  - these two pixels should have the same entity ID …

- Semantics / composition can be implicitly encoded in the correlations, or <u>mutual information</u>, between pixels

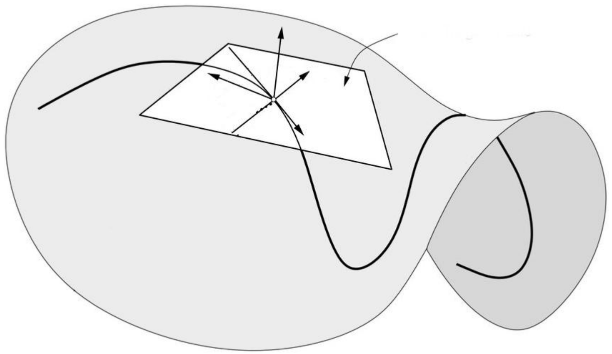
- How can we directly encode into the neural representation semantic correlations? – the NeRF variation space

An Indoor Scene

Understand how the scene is, *and how it could be …*

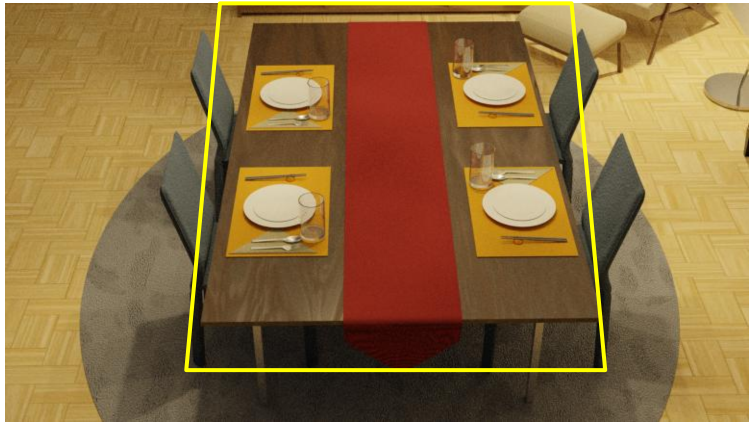
The Compositional Structure of a Scene is Reflected in its Variations

The scene semantic variations:
tangent space



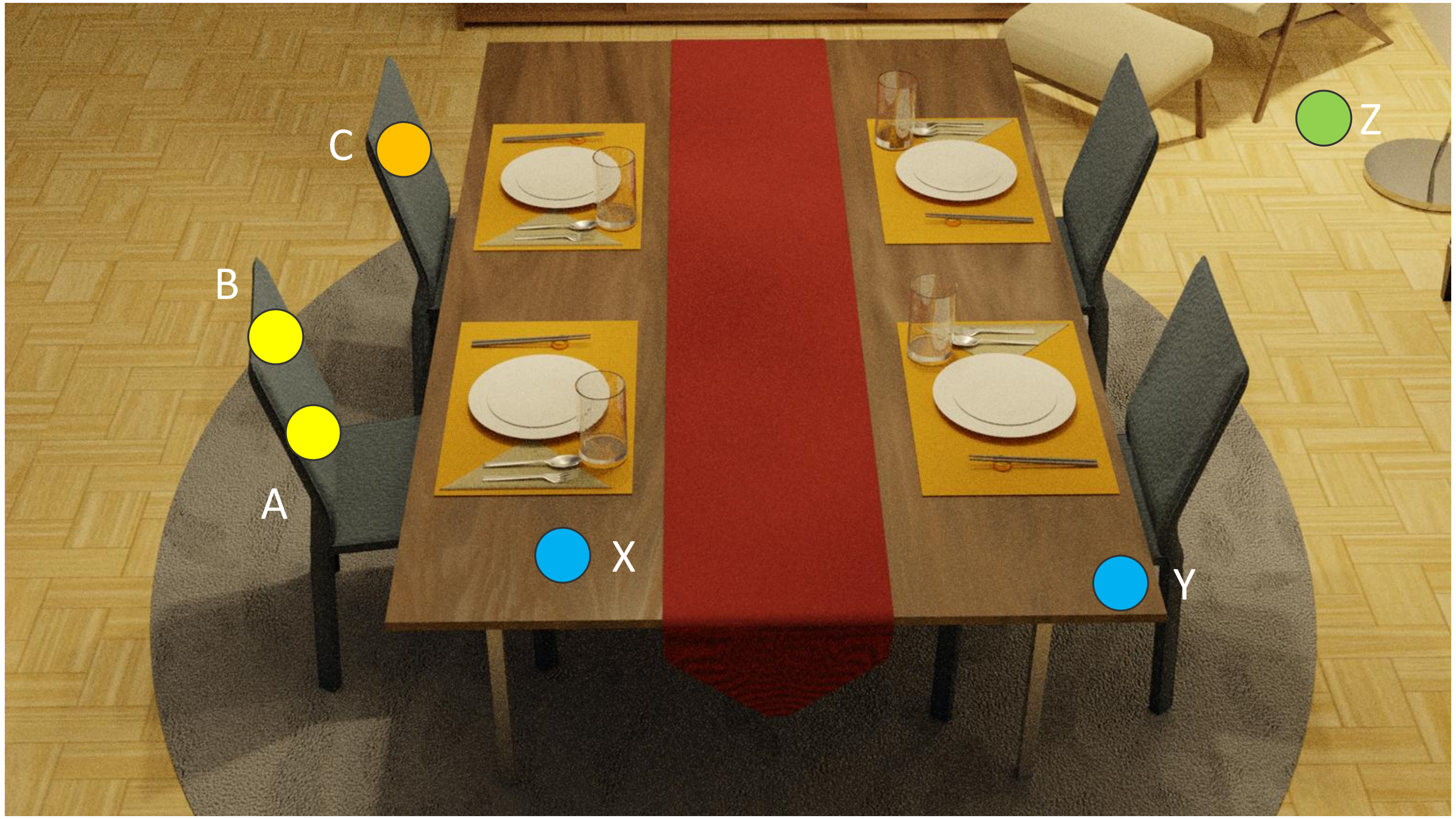
this chair has moved

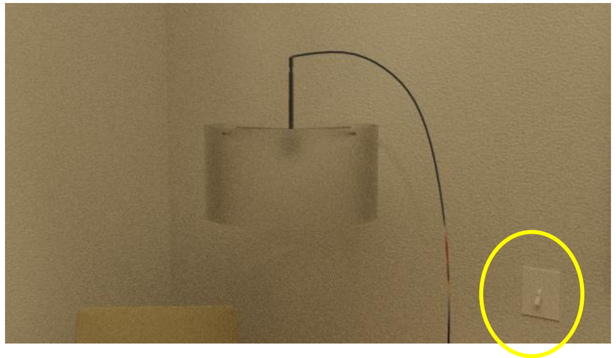
the table became longer

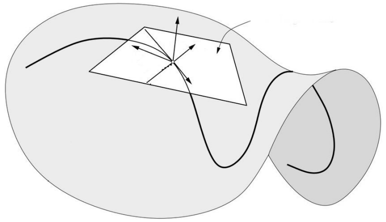


the table became darker

Mutual Information and 2nd Order Relationships



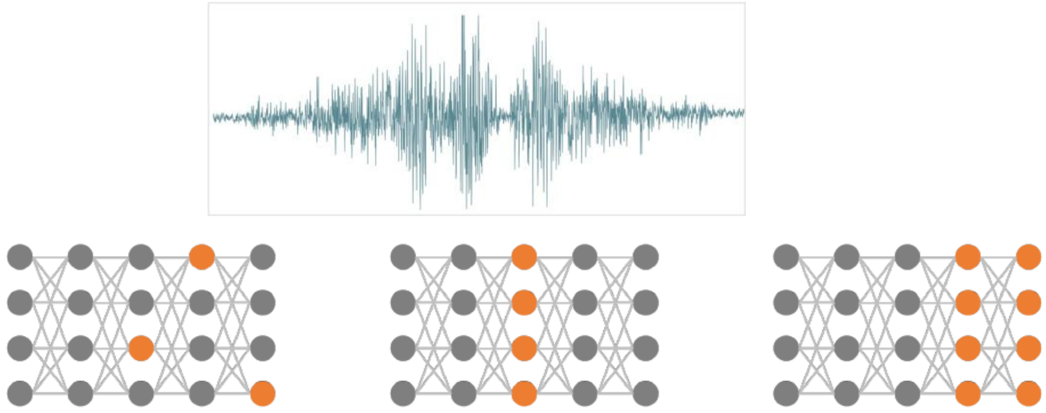




A is more correlated with B than with C

X is more correlated with Y than with Z

NeRF Variation Space Through its Parameters



(a) random neurons (b) a single layer (c) a block of layers

Unfortunately, these variations are not semantically meaningful

"Jiggle" the neuron weights of the NeRF

images (a) (b) (c)

Key Idea: Operate on the NeRF to modify its weights, so as to align the scene semantic variation space with the NeRF parametric variations

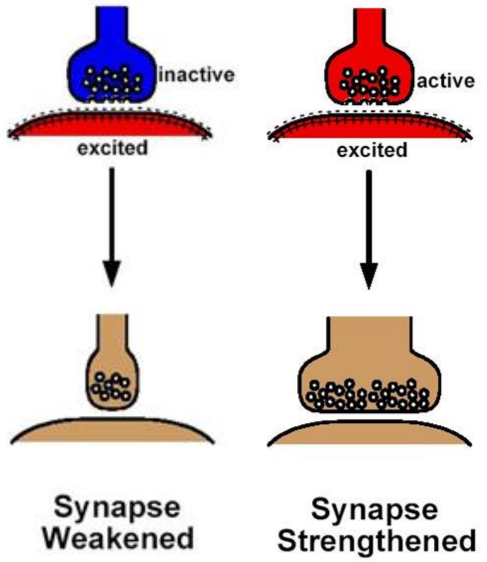
"NeRF Shaping"

NeRF Operators

Shaping Neural Representations

The Hebbian hypothesis:

Neurons that fire together, wire together

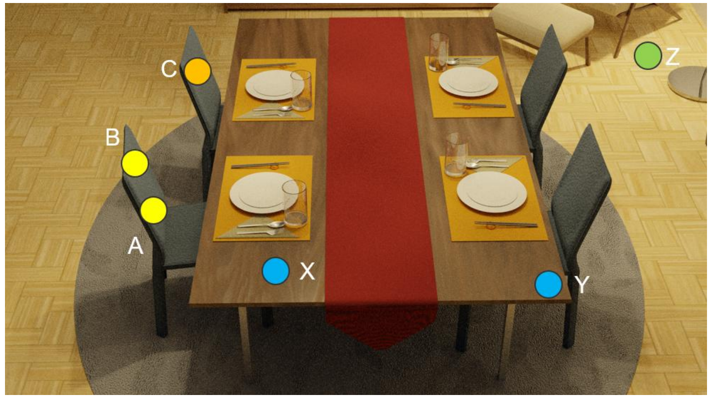


Donald O. Hebb
1904-1985

Can we build neural scene representations that better reflect mutual information correlations in the scene?

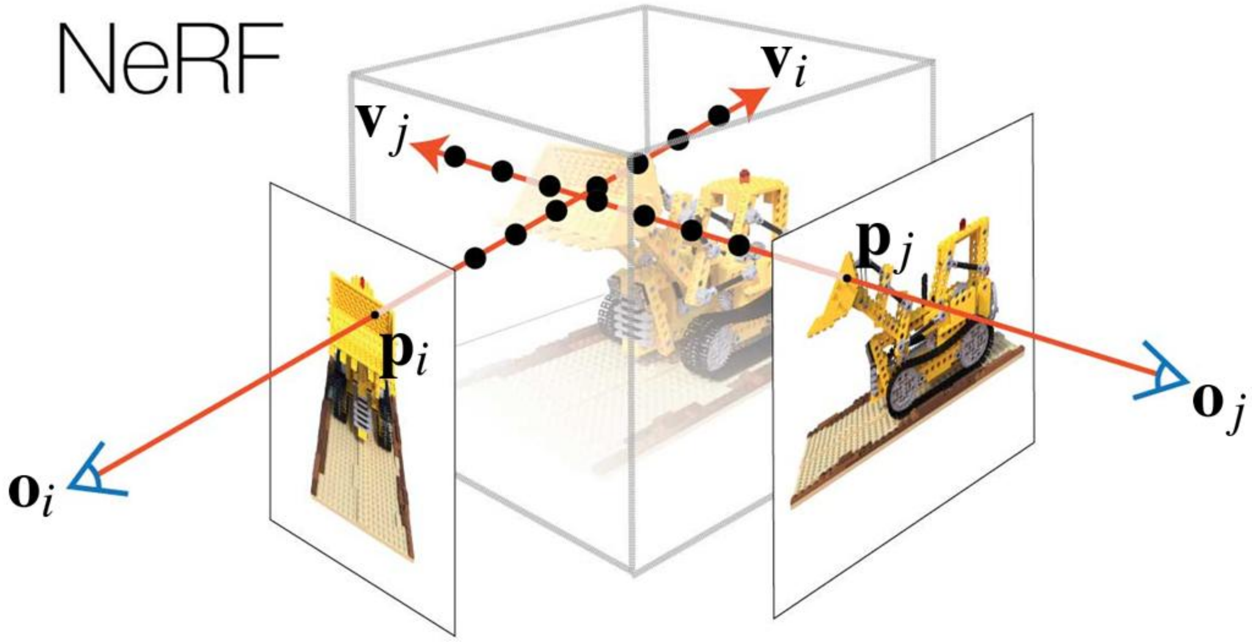
Create “neuronal resonances” through contrastive supervision

$\mathbb{I}(A,B) > \mathbb{I}(A,C)$
$\mathbb{I}(X,Y) > \mathbb{I}(X,Z)$



Mutual information $\mathbb{I}$

Mutual Information via NeRF Gradients



$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{v} \mid t \geq 0$$

$$I(\mathrm{p}) = \Phi(\mathbf{o}, \mathbf{v}; \theta) = \int_0^{+\infty} w(t;\theta) c(\mathbf{p}(t), \mathbf{v}; \theta) \mathrm{d}t$$

$$I\left(\mathrm{p}_i\right) = \Phi\left(\mathbf{o}_i, \mathbf{v}_i; \theta\right)$$
$$I\left(\mathrm{p}_j\right) = \Phi\left(\mathbf{o}_j, \mathbf{v}_j; \theta\right)$$

Mutual information $\mathbb{I}$

$$\hat{I}\left(\mathrm{p}_i\right) = \Phi\left(\mathbf{o}_i, \mathbf{v}_i; \theta^D + \mathbf{n}\right)$$
$$\hat{I}\left(\mathrm{p}_j\right) = \Phi\left(\mathbf{o}_j, \mathbf{v}_j; \theta^D + \mathbf{n}\right)$$

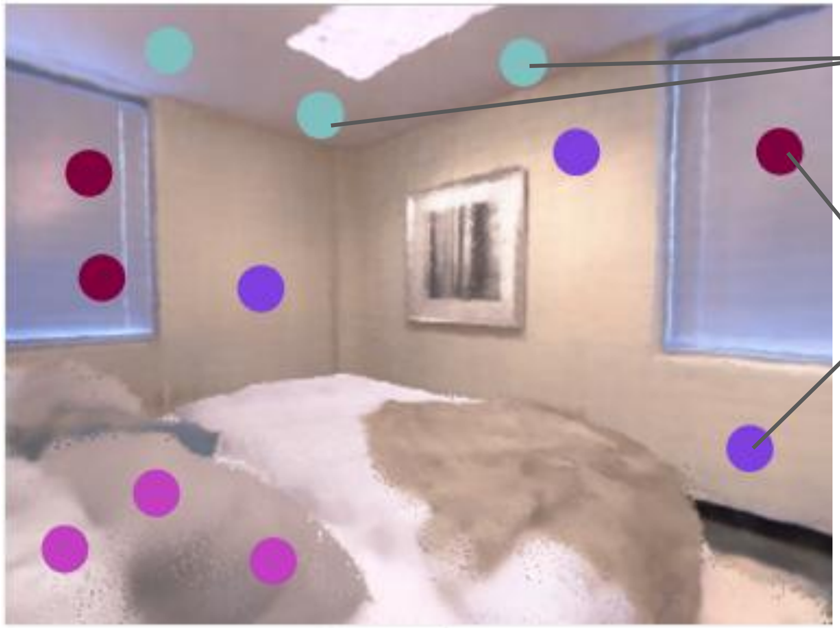
$$\boxed{\mathbb{I}\left(\hat{I}\left(\mathrm{p}_i\right), \hat{I}\left(\mathrm{p}_j\right)\right) \approx \cos\left(\frac{\partial \Phi_i}{\partial \theta^D}, \frac{\partial \Phi_j}{\partial \theta^D}\right)}$$

D = set of parameters selected for shaping
n = noise added

Inter-pixel correlations are captured by cosine similarity of the NeRF Jacobians

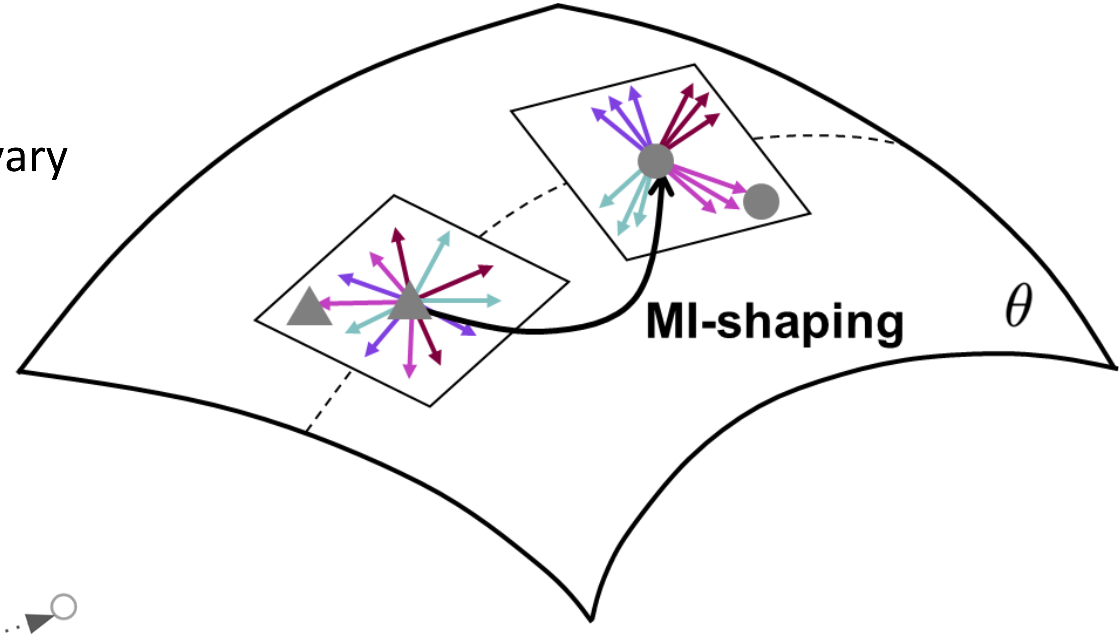
NeRF Mutual Information Shaping

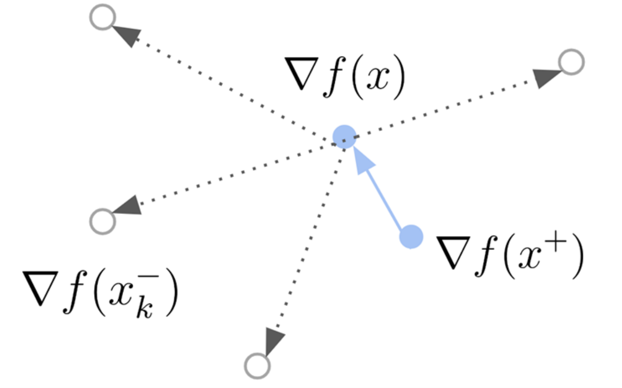
Setting up Semantic "Neuronal Resonances" for Correlated Pixels through Contrastive Learning



These pixels should co-vary

But these should not

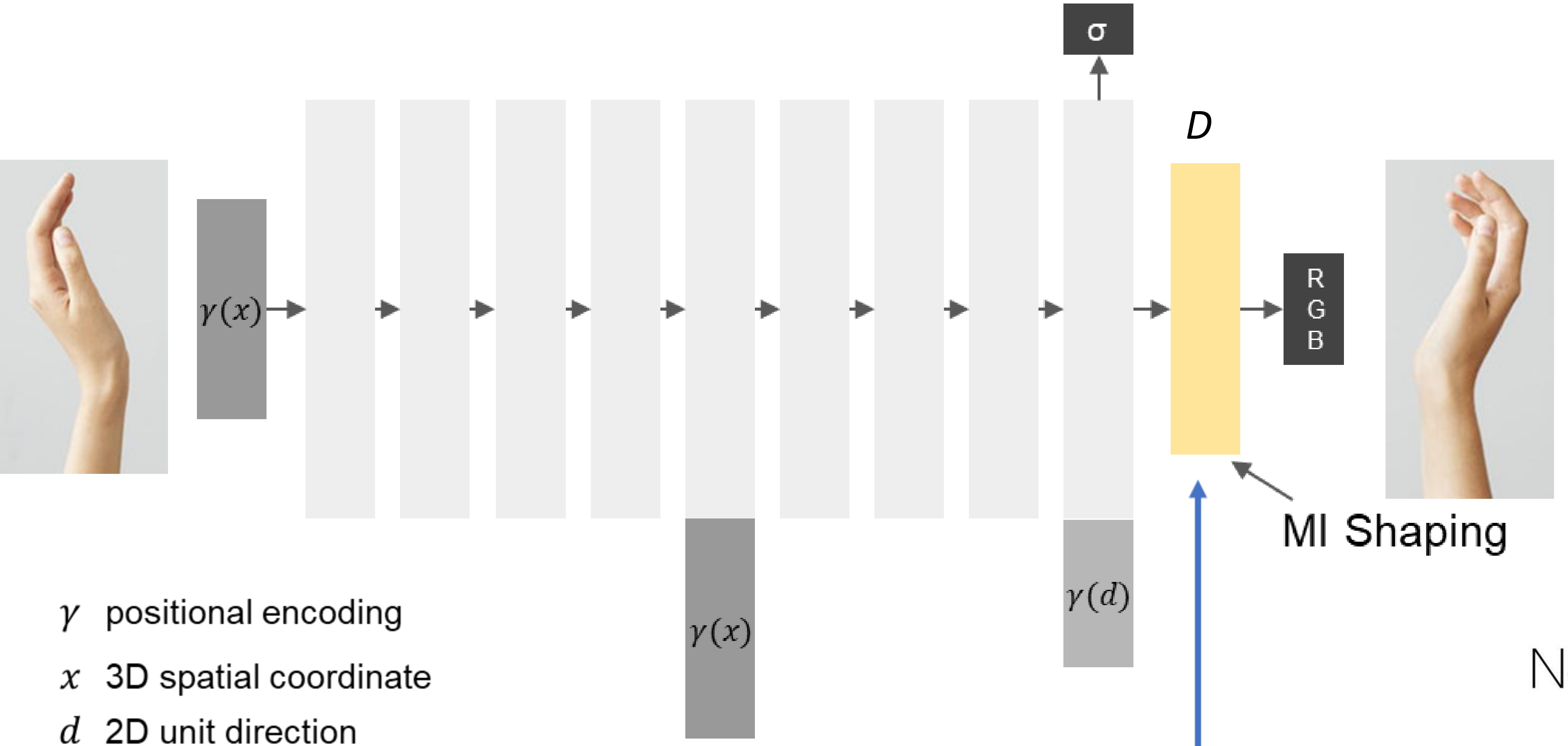




Use DINO features (alternatively, LSeg-CLIP)

Shaping co-aligns gradients of correlated pixels (here points of the same semantic class)

JacobiNeRF (J-NeRF): Shaping via Mutual Information Jacobians



γ positional encoding

x 3D spatial coordinate

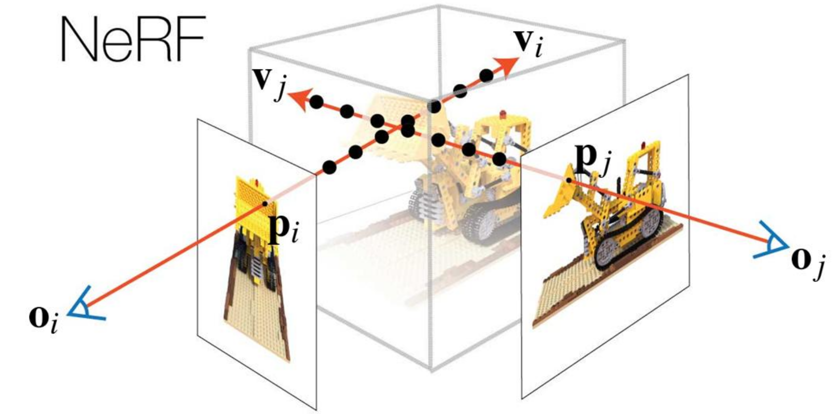
d 2D unit direction

σ point density

$$\cos\left(\frac{\partial \Phi_i}{\partial \theta^D}, \frac{\partial \Phi_j}{\partial \theta^D}\right)$$

Apply gradient shaping on pixel gray scale values
(last three layers of RGB branch)

Carl Gustav Jacob Jacobi
1804-1851



Lightweight Contrastive Training Regimen (After Photometric)

Use DINO features for contrastive learning

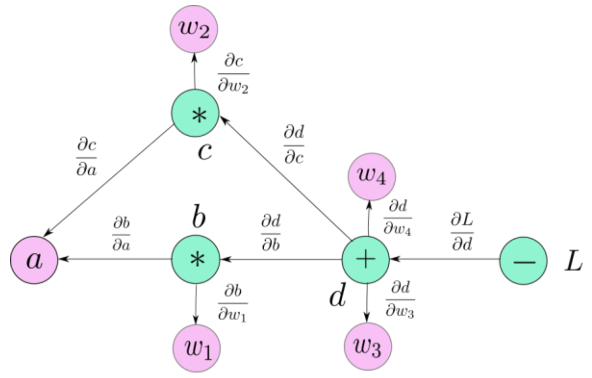
64 batches of 64 rays/pixels across all views, or

64 batches of 64 rays/pixels all in one view

Gradients are on pixel gray level

InfoNCE loss

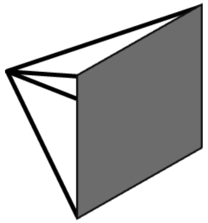
Exploit Autograd for gradients

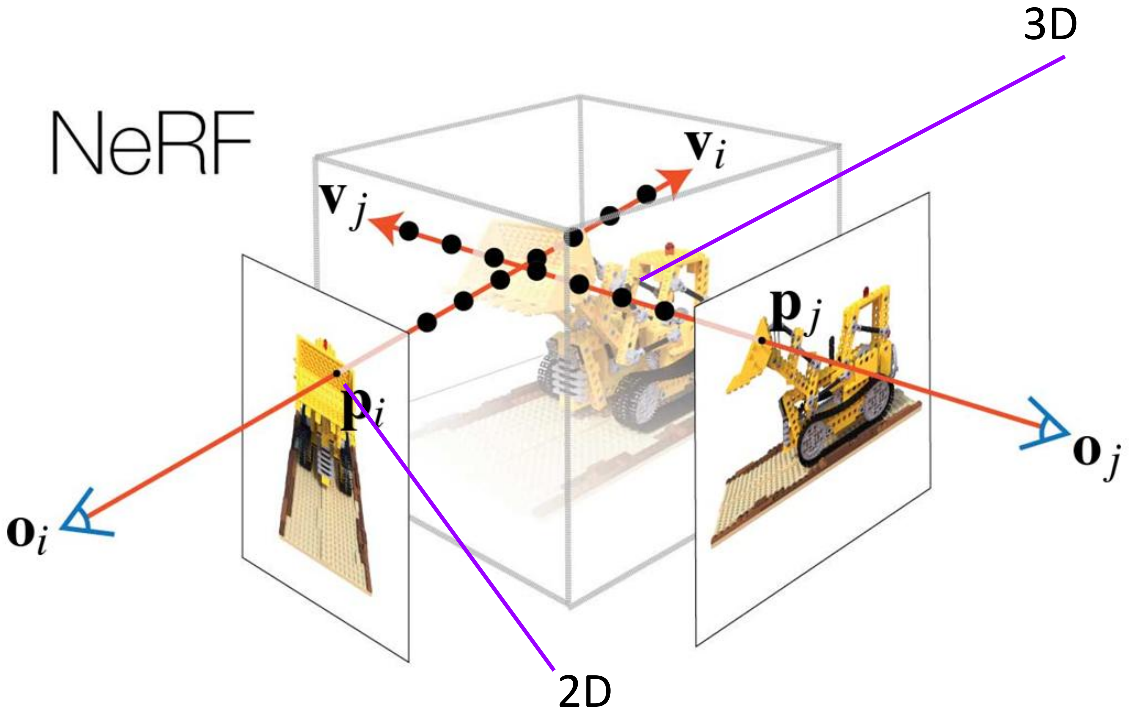


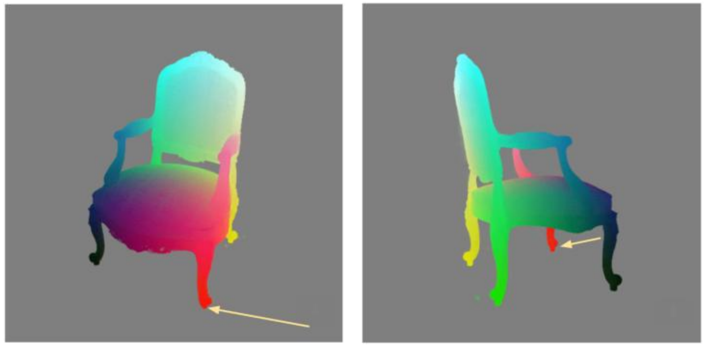
Resonances are transitive ….

2D vs 3D Shaping

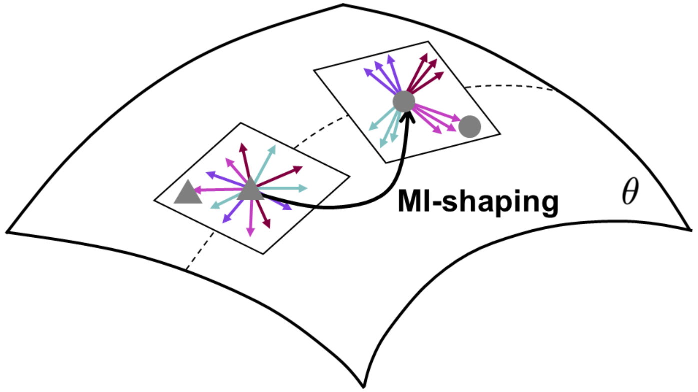
Use NOCS to lift 2D pixels to 3D points.



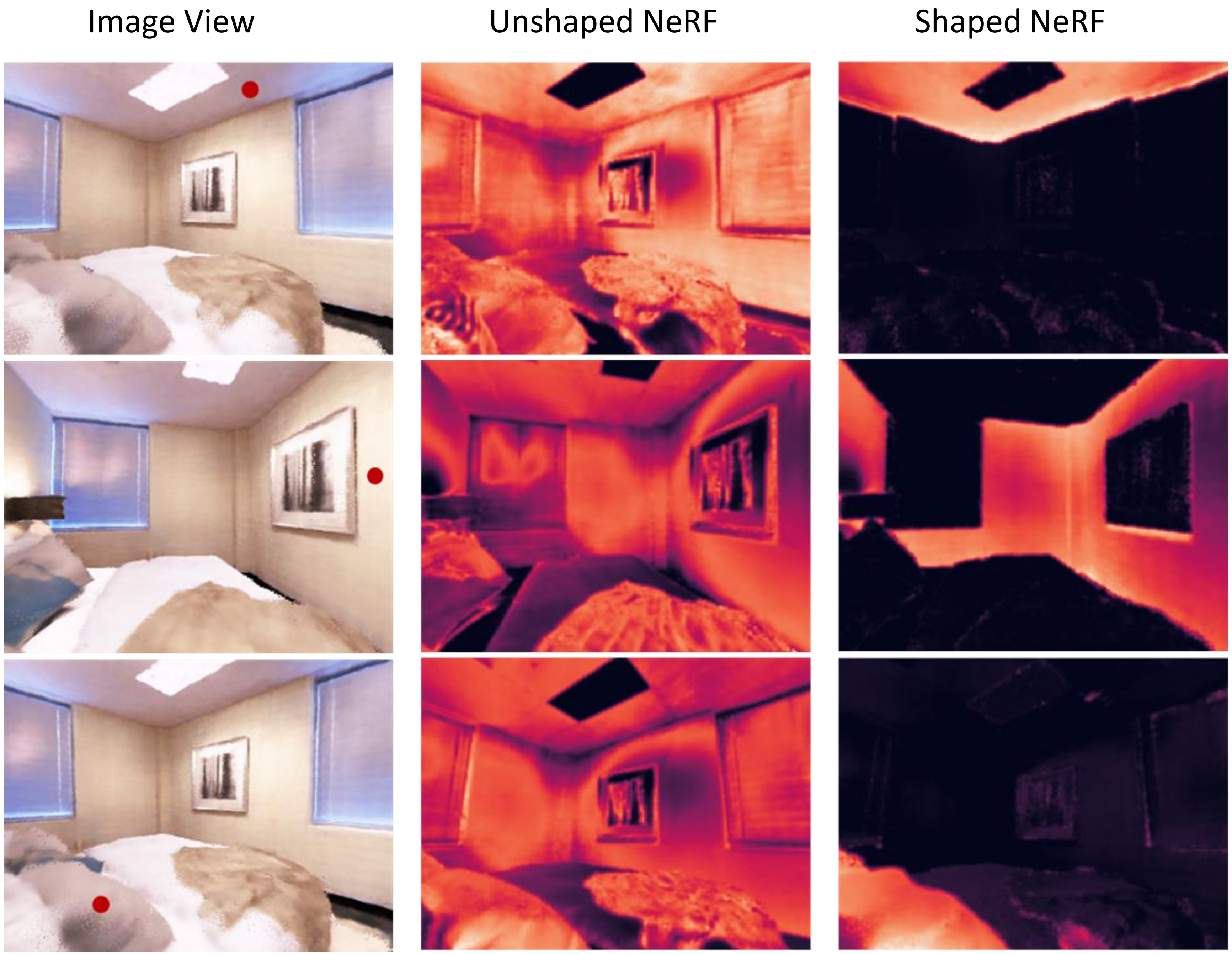




NOCS (x, y, z) coordinates rendered as RGB



Shaping can be applied to either 2D (J-NeRF 2D) or 3D (J-NeRF 3D) network gradients



After shaping:

From a single pixel we can select an entire semantic entity.

Re-coloring via Resonances

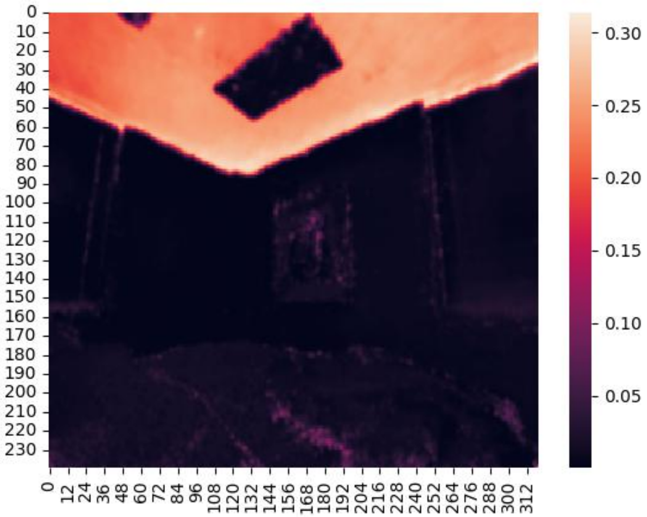
NeRF re-coloring after shaping

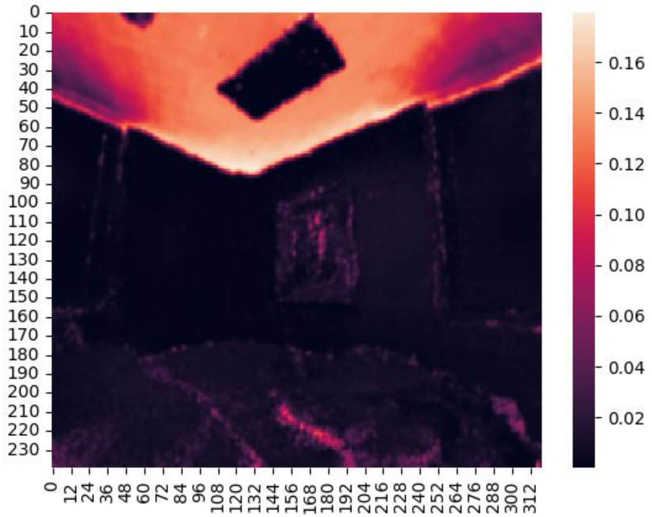
Shape NeRF by aligning grey scale gradients of correlated pixels/points [same as before].

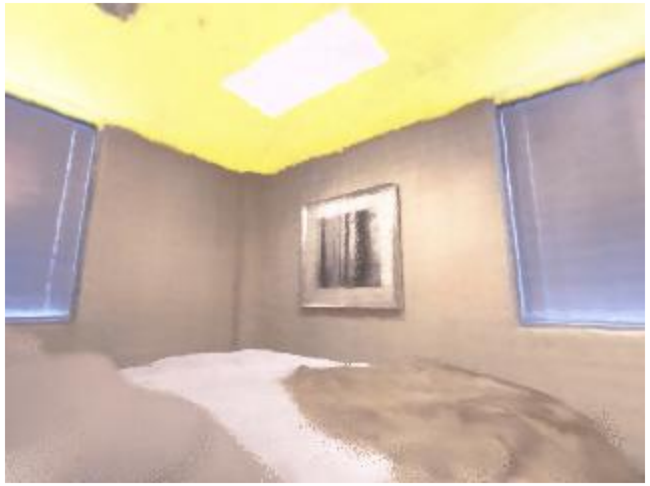
Calculate separate R, G, B gradients; select **one pixel in one view** and push the network parameters along these gradients to reach a desired color value at that pixel.

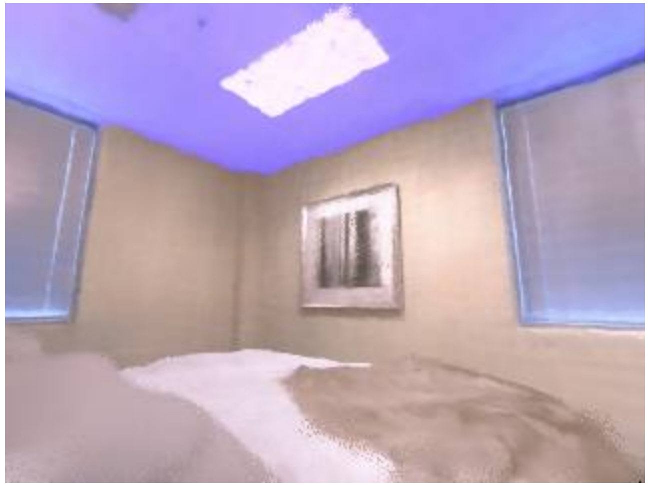
All “resonating” pixels in this and other NeRF views get also automatically recolored …

Ceiling re-colored
(yellow, blue)

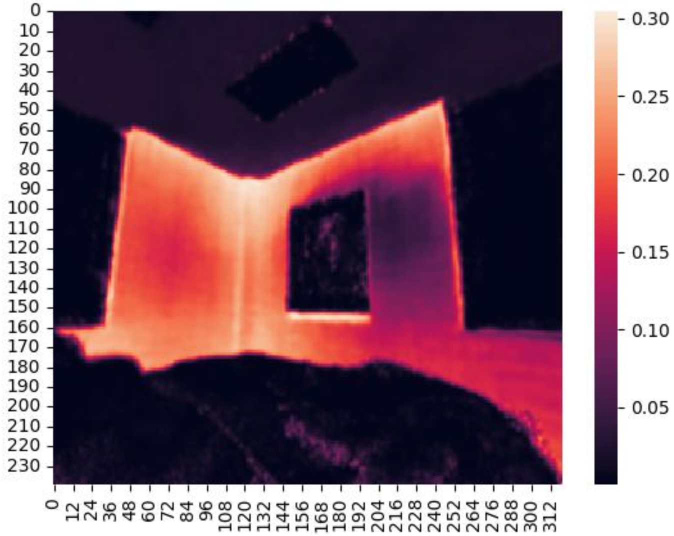


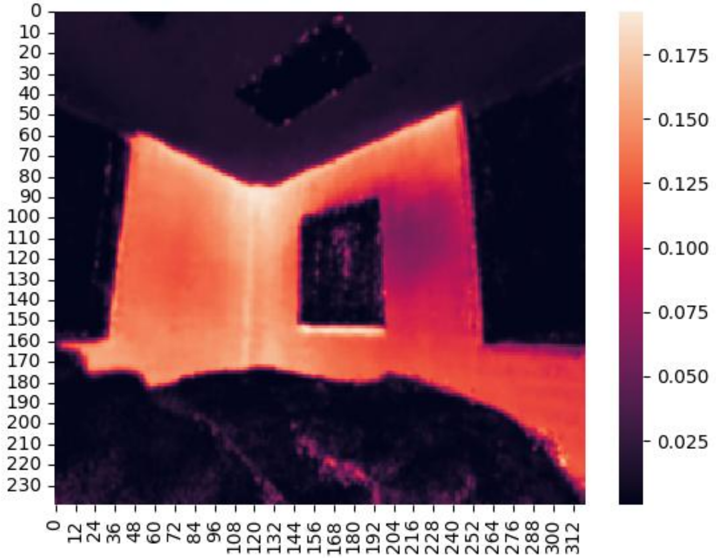


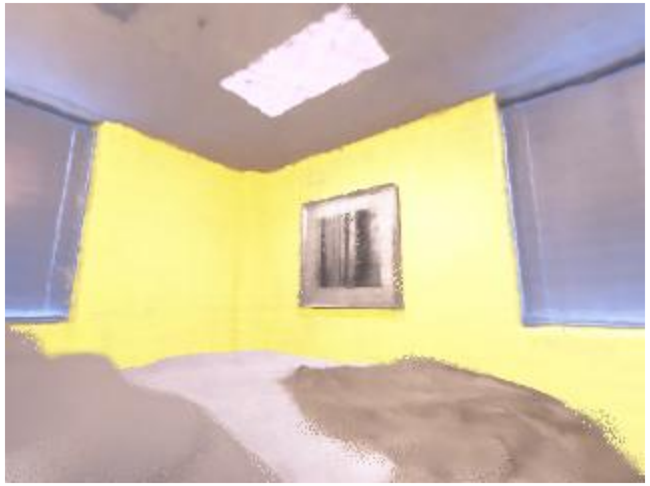


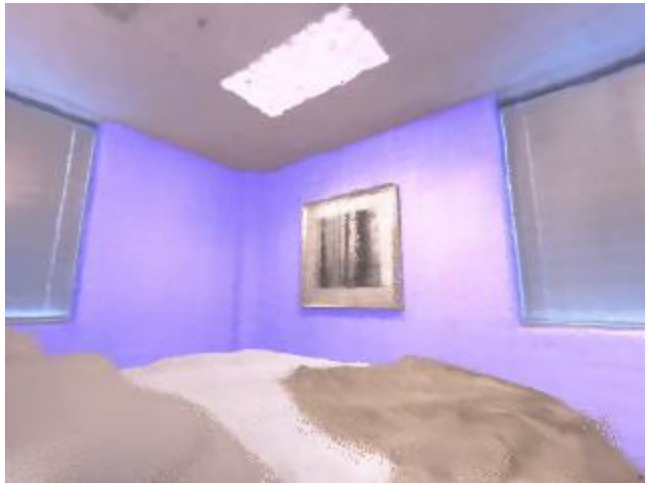


Walls re-colored
(yellow, blue)

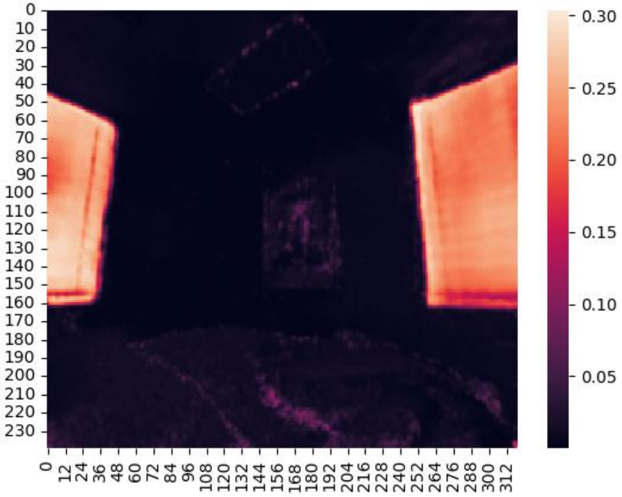


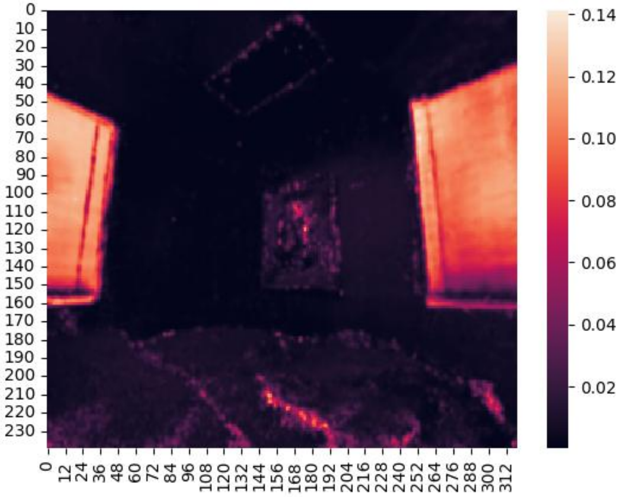


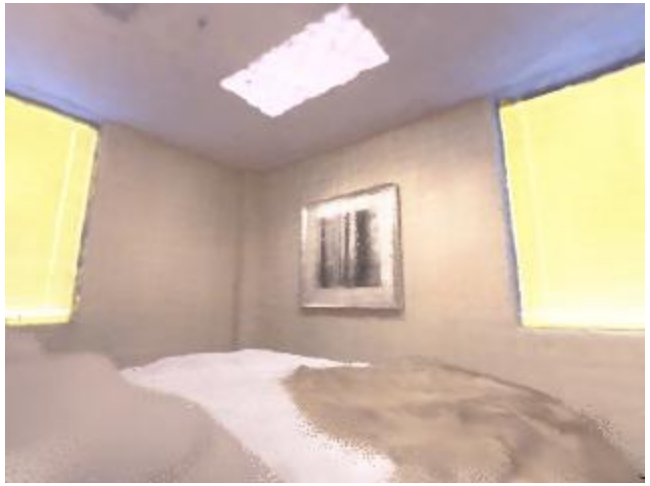




Windows re-colored
(yellow, blue)







Info Propagation Through Resonances in Views

2D version (JacobiNeRF-2D):

for each labeled pixel

- perturb the NeRF along the gradient of the gray value of that pixel (e.g., change the network parameters)
- synthesize the target view from the perturbed NeRF
- calculate the perturbation response at each pixel for every source

- assign in target view pixels to the class generating the maximal response (argmax)

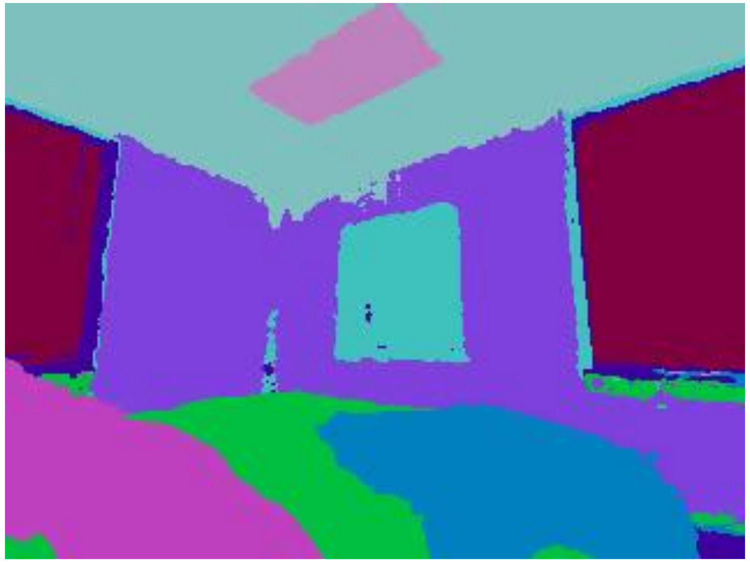
Make a move

See who follows

Semantic Segmentation (sparse, Replica)

Given label

Regenerated view



J-NeRF 3D Semantics
Propagation

Semantic Segmentation (sparse, ScanNet)

Given label

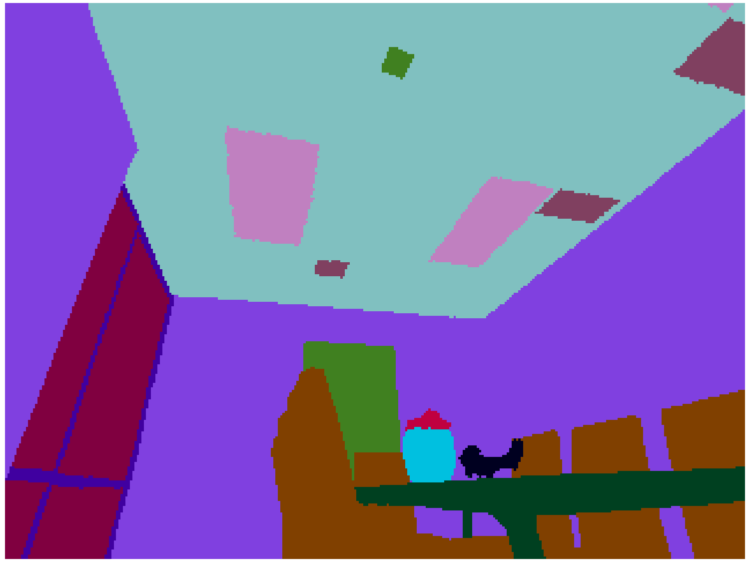
J-NeRF 3D

Semantic Segmentation (dense, Replica)

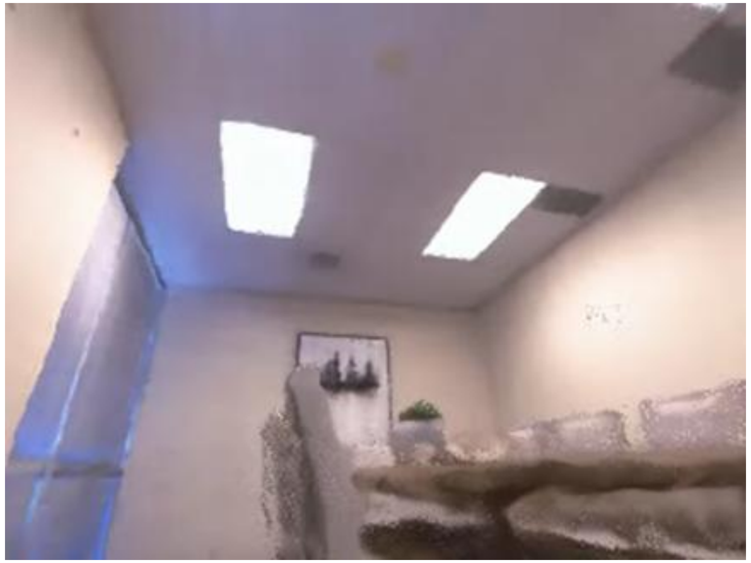
Given label

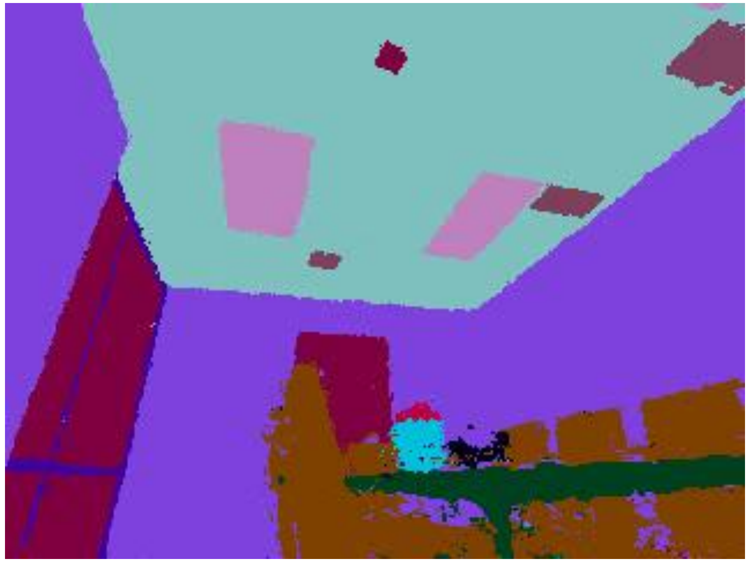
J-NeRF 3D

Semantic Segmentation (dense, Replica)



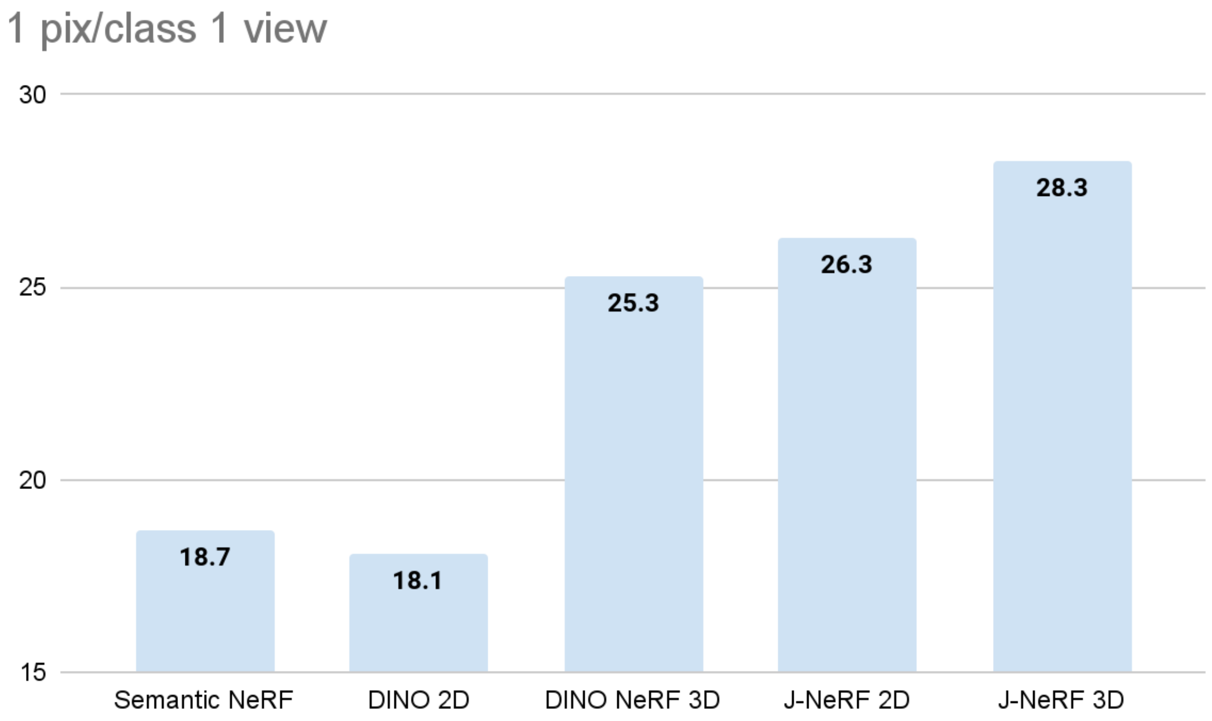
Given label



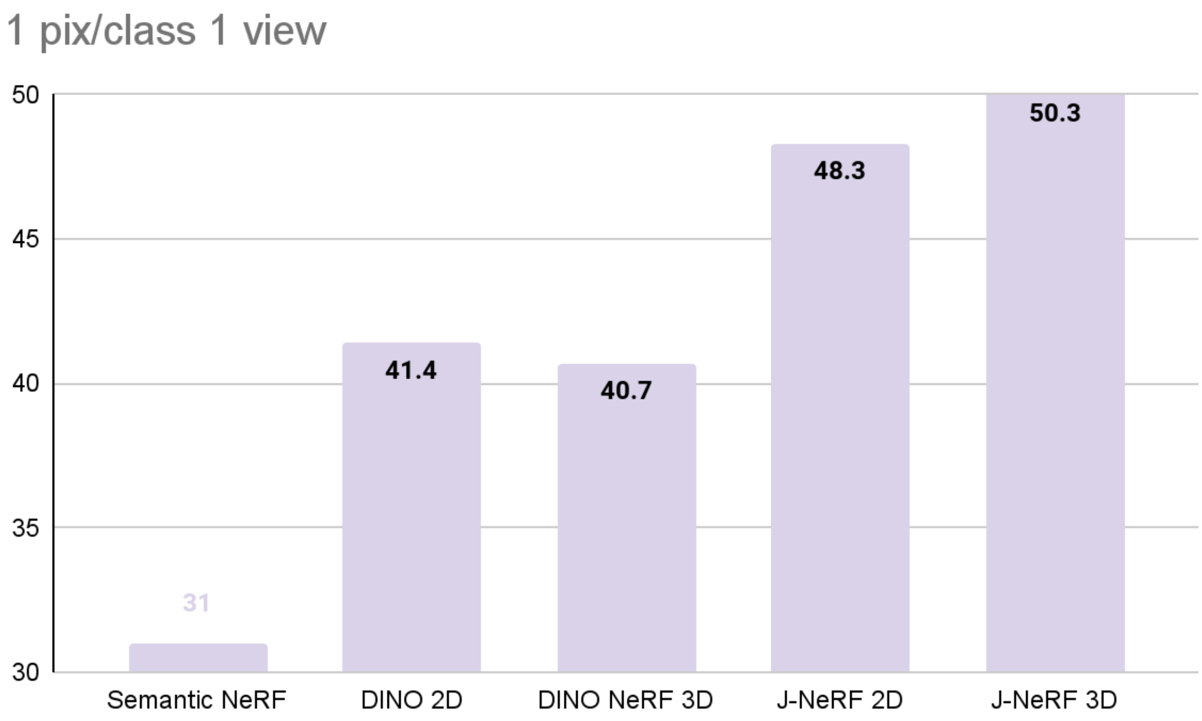


J-NeRF 3D

Semantic Segmentation (sparse 1pix/class, Replica)

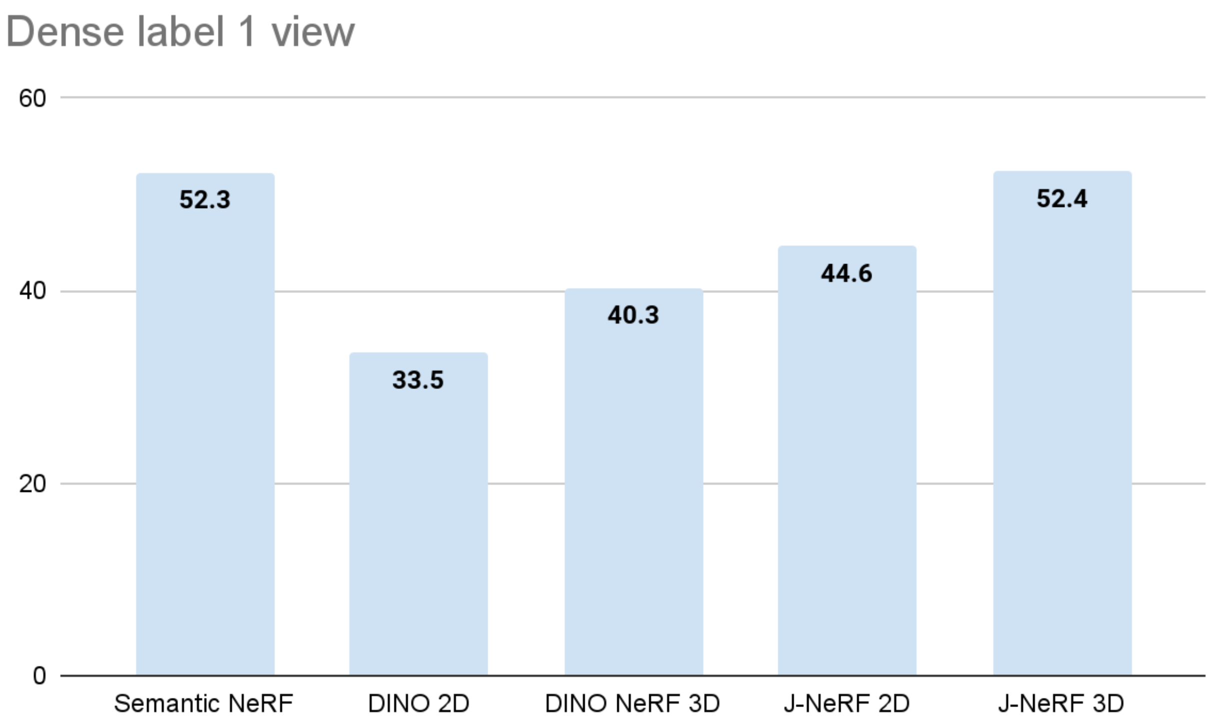


mIoU

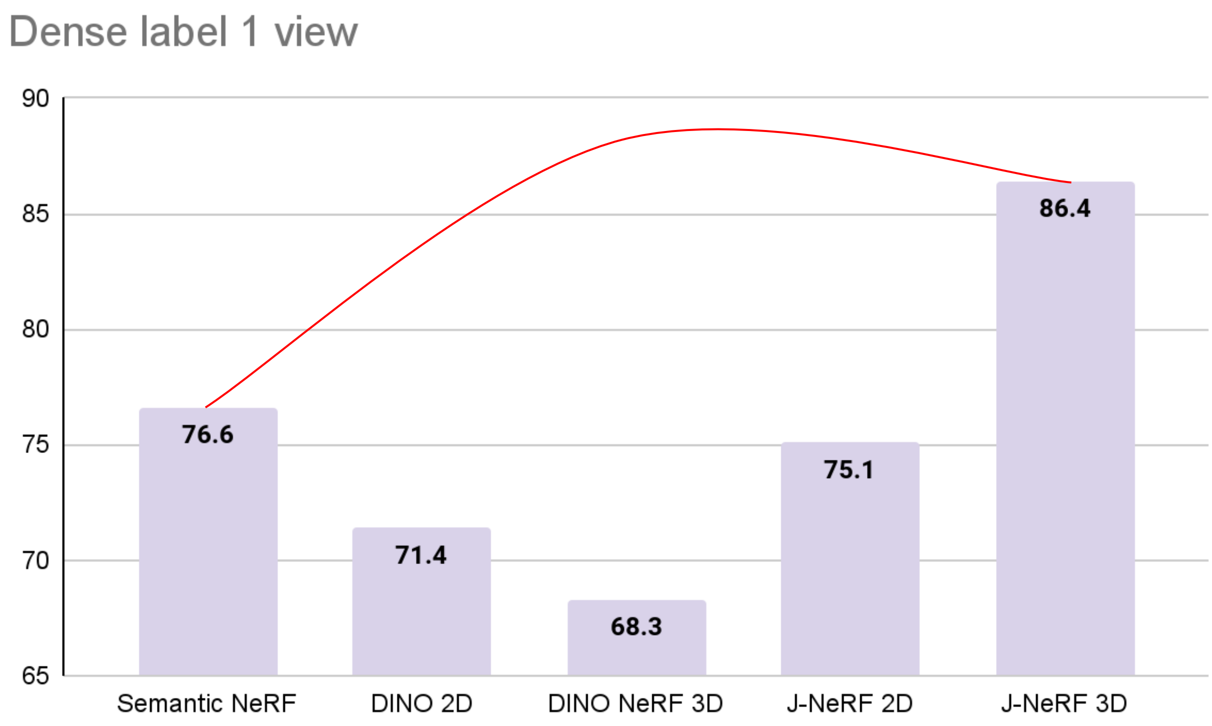


Acc

Semantic Segmentation (1 view, dense labels, Replica)



mIoU



Acc

Light Supervision for Structure Emergence in Gaussian Fields

InfoGaussian: Structure-Aware Dynamic Gaussians through Lightweight Information Shaping. Yunchao Zhang, Guandao Yang, Leonidas Guibas, Yanchao Yang. ICRL 2025.

Static Scene Reconstruction with Gaussian Splatting

SAM "Segment Anything" 2D Dense Instance Supervision

Gaussian Grouping: Segment and Edit Anything in 3D Scenes. Mingqiao Ye, Martin Danelljan, Fisher Yu, Lei Ke.

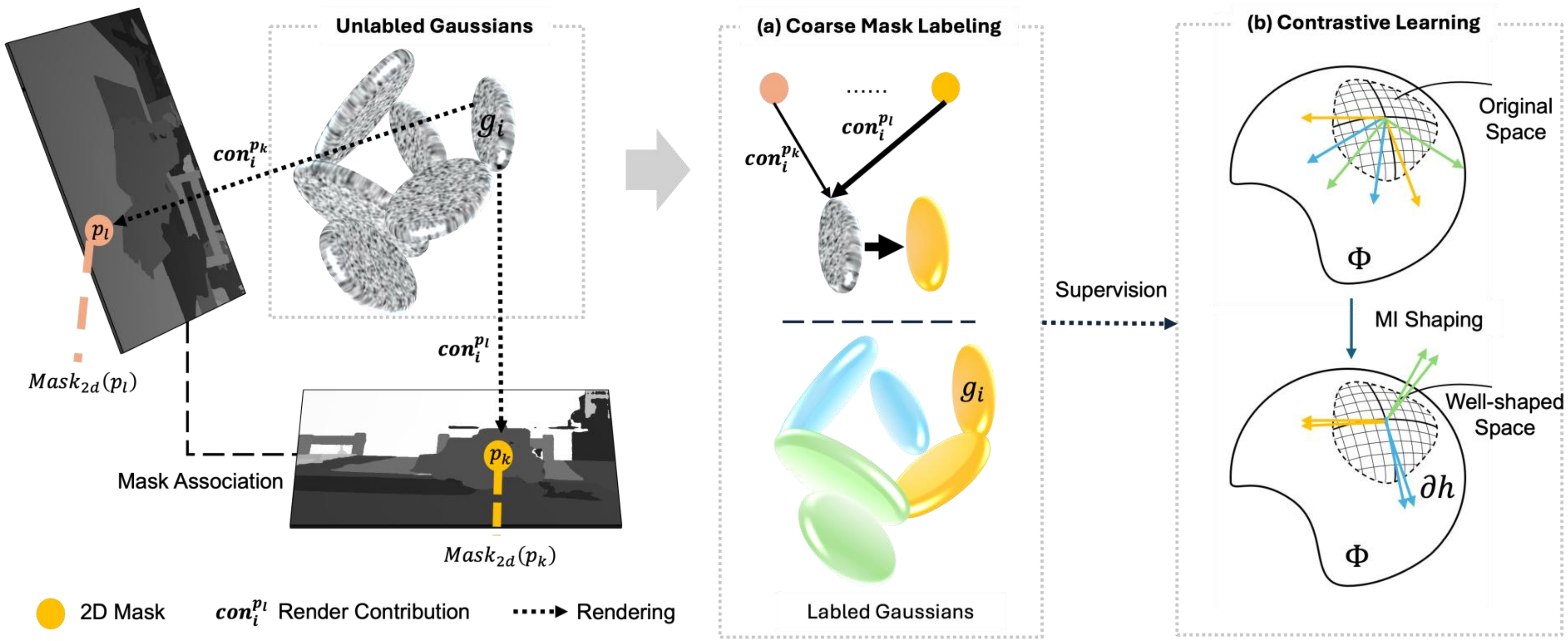
GS Network Shaping for Object Motion

Pixels in the Same Mask Have High Mutual Information (MI)

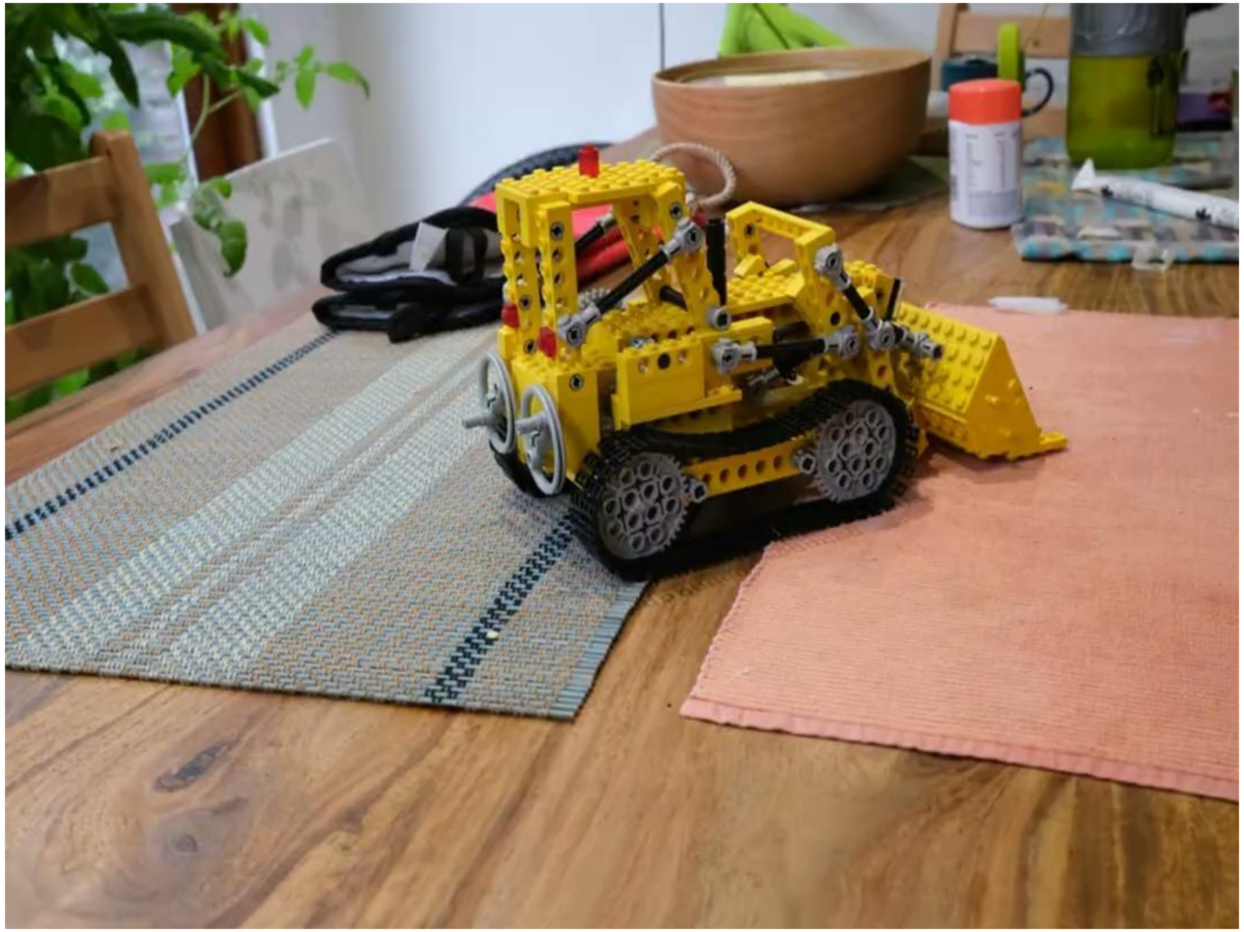
They are likely to change in correlated ways, as objects move.

Correlation Shaping on Attribute Decoding

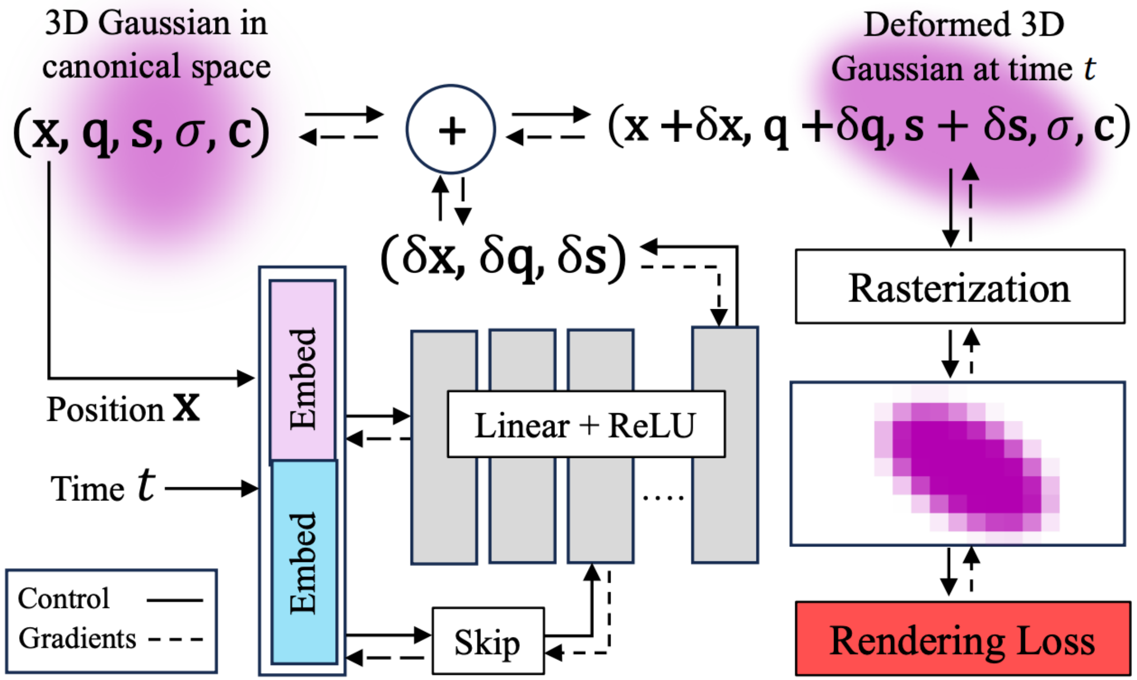
1. Use pretrained vision model (SAM) to generate 2D masks.
2. Label the 3D masks of Gaussians by 2D masks of the pixel.
3. Conduct contrastive learning for mutual information shaping.



Object Motion Without Explicit Grouping



MotionMLP



Motion MLP Gradient Alignments

Aligning MLP gradients via contrastive learning, to force high MI correlations

$$\mathbb{I}\left(\hat{f}\left(g_i\right), \hat{f}\left(g_j\right)\right) = \log\left(\frac{1}{\sqrt{1-\cos^2(\gamma)}}\right) + \text{ const.}$$

$$\hat{f}\left(g_i\right) = \Phi\left(x_i; \theta_D + n\right)$$
$$\hat{f}\left(g_j\right) = \Phi\left(x_j; \theta_D + n\right)$$

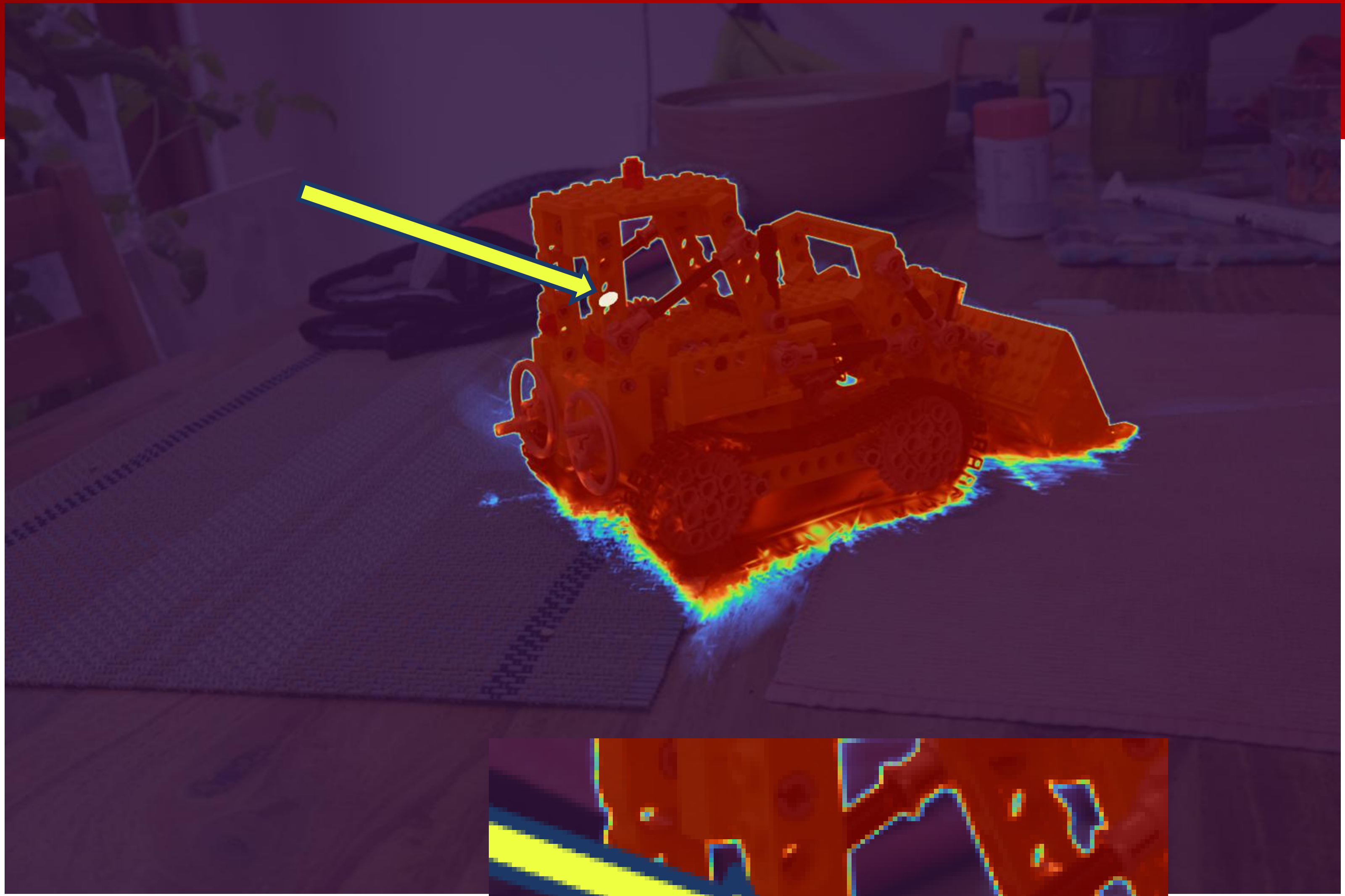
$$\gamma = \cos\left(\frac{\partial \mathbf{\Phi}\left(\mathbf{x}_i; \theta\right)}{\partial \theta_D}, \frac{\partial \mathbf{\Phi}\left(\mathbf{x}_j; \theta\right)}{\partial \theta_D}\right)$$

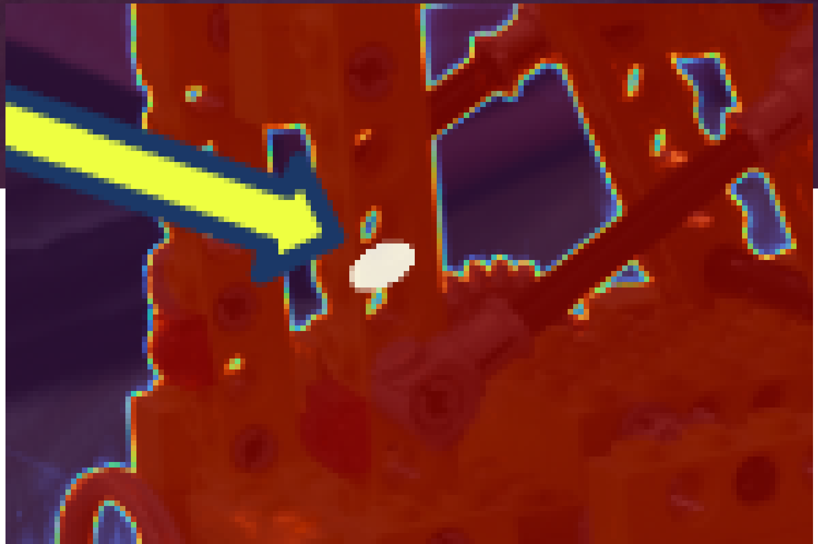
Experiment (Mips-NeRF 360 Scene)

Resonances

~7,900 Gaussians in the bulldozer

~460 participated in the contrastive training





Experiment (Mips-NeRF 360 Scene)

Post-Shaping

Experiment (Mips-NeRF 360 Scene)

Post-Shaping

Experiment (LeRF Scene)

Resonances

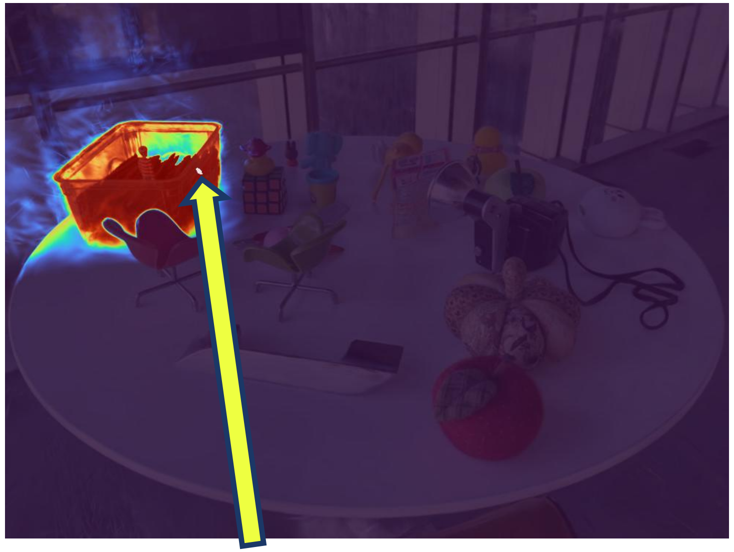
Experiment (LeRF Scene)

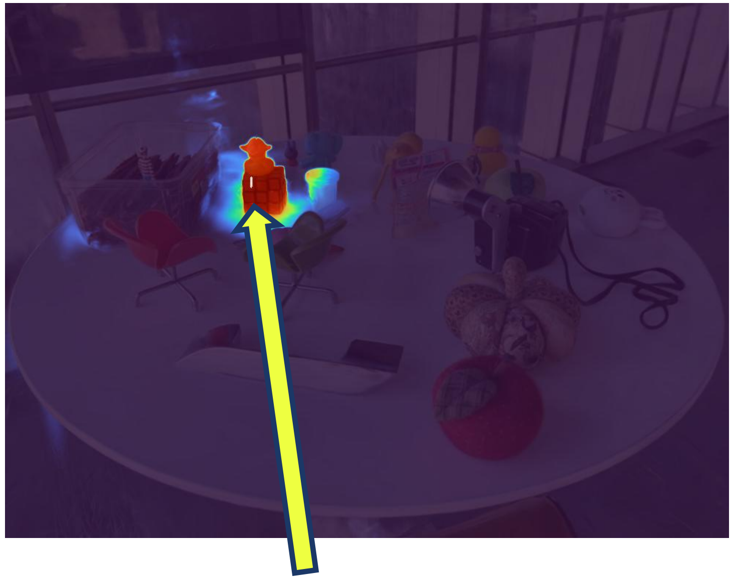
Post-Shaping

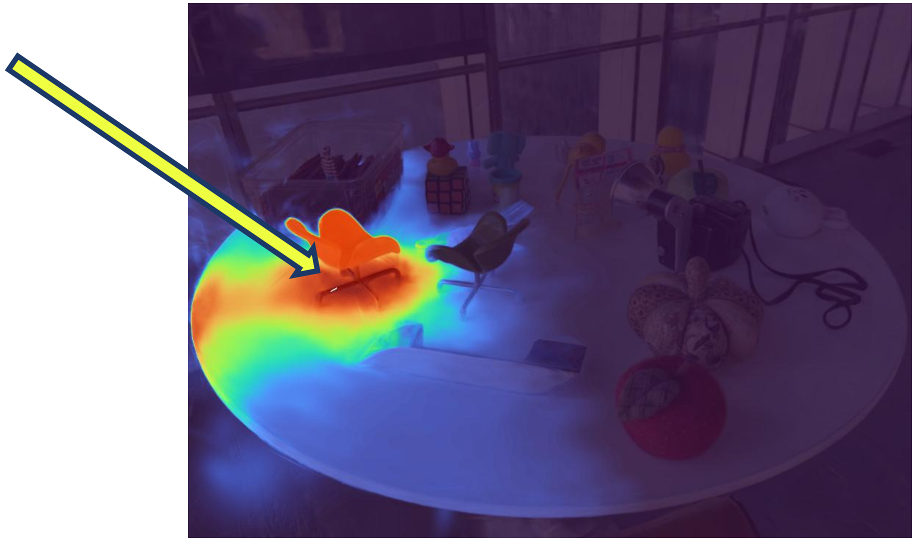
Experiment (LeRF Scene)

Post-Shaping

Resonances

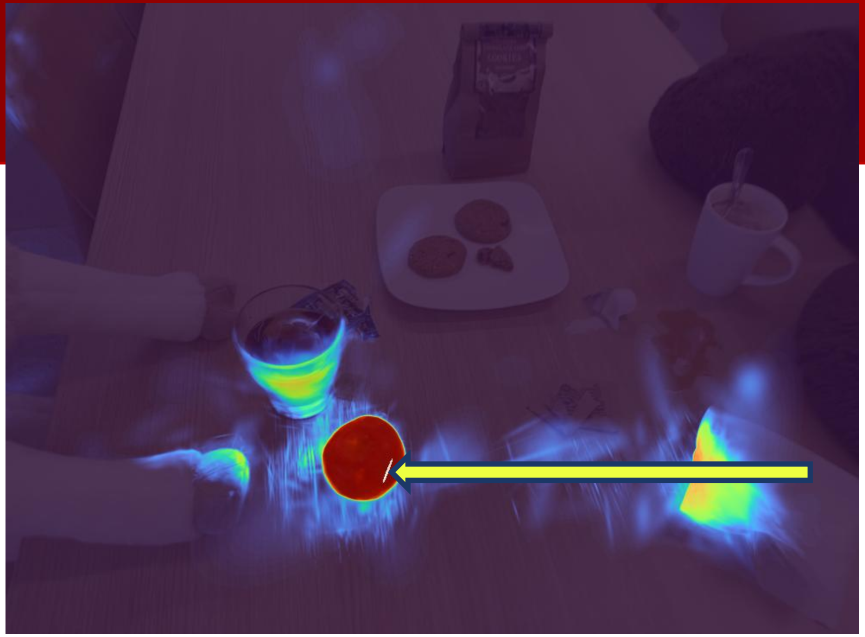


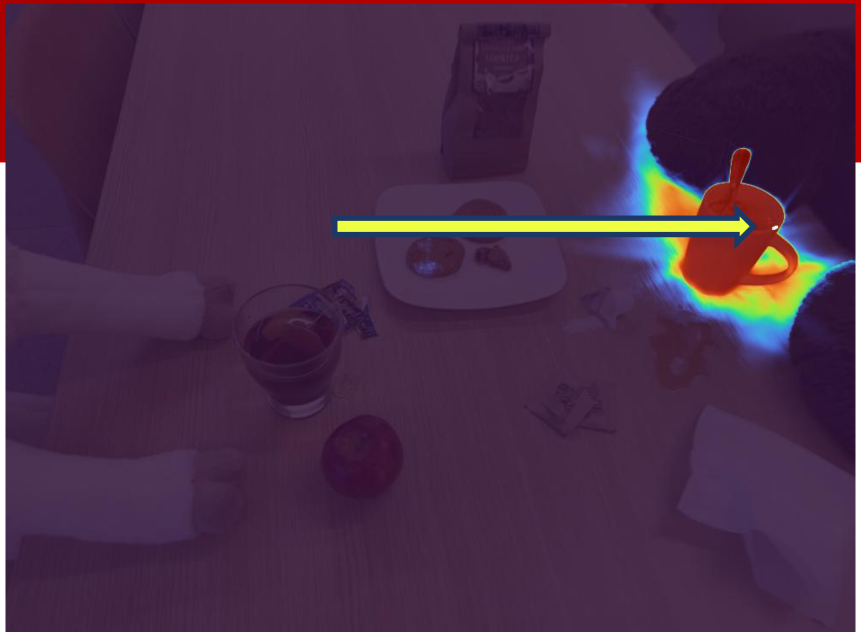


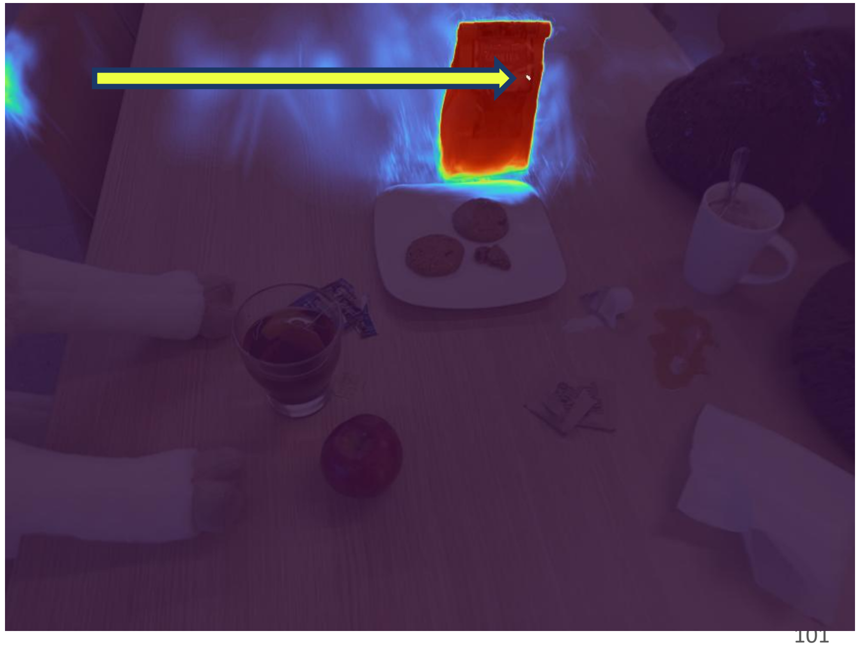


Experiment (LeRF Scene)

Resonances







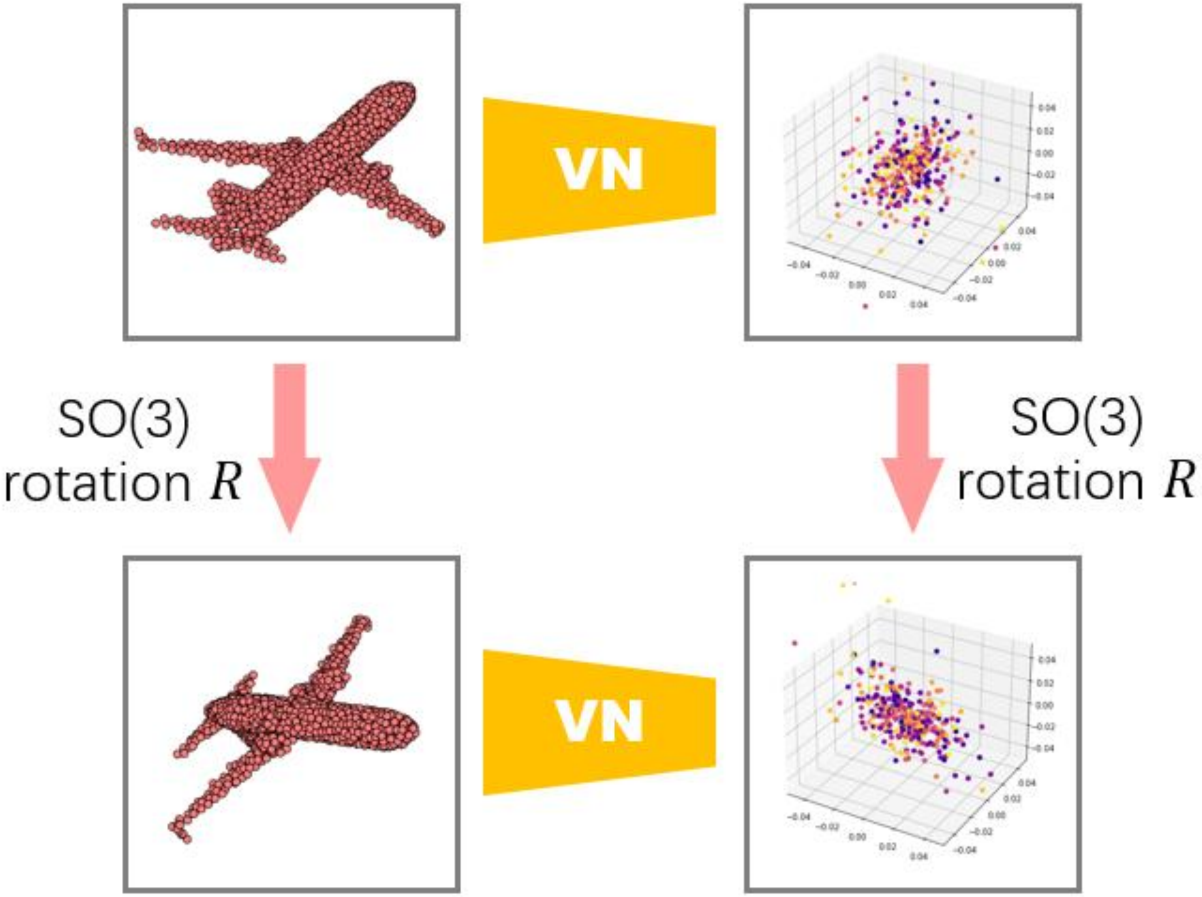
Experiment (LeRF Scene)

Post-Shaping

Summary: Enforcing a Prior (Equivariance)

- **Vector Neurons:**

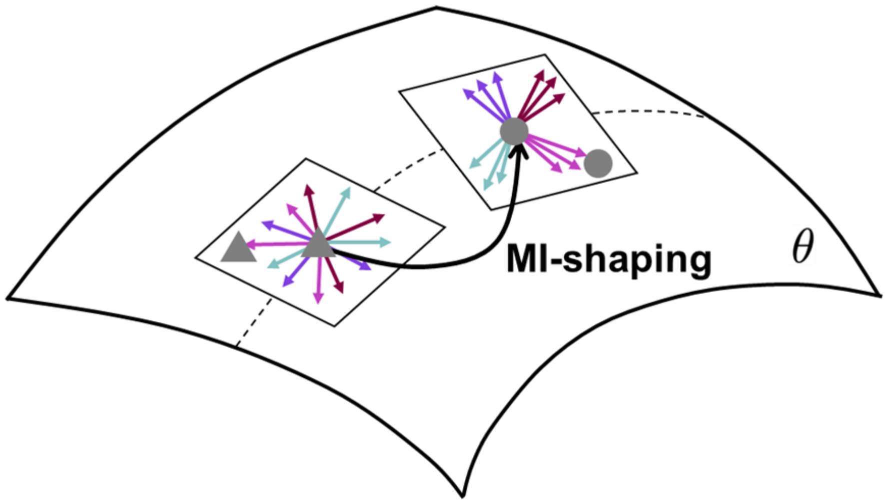
  Lift latent features to 3D vector lists
- **Building blocks:**
  - Linear layer
  - Non-linearity (ReLU)
  - Pooling (MaxPool)
  - Normalizations (BatchNorm)
  - Invariance
- **Network examples:**
  - VN-DGCNN
  - VN-PointNet

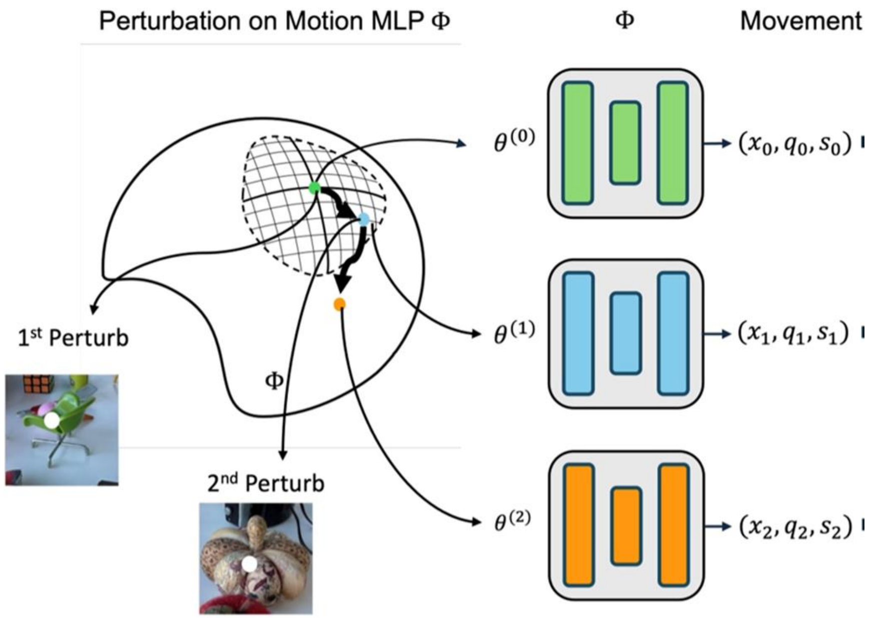


Summary: Learning a Prior (Grouping, Equivariance)

- **Scene Structure from Network Shaping:**

  Gradient alignments according to mutual information
- **Semantic resonances:**
  - Learned from DINO features
  - Allow coherent edit propagation of
    - semantics
    - appearance
- **Motion resonances:**
  - Learned from SAM instance masks
  - Allow coherent entity motions





Thanks


WILLIAM GATES
COMPUTER SCIENCE
353
STANFORD CARDINAL
STANFORD

That's All