



Boulder

# Computer Vision; 3D



[YouTube Playlist](#)

**Maziar Raissi**

**Assistant Professor**

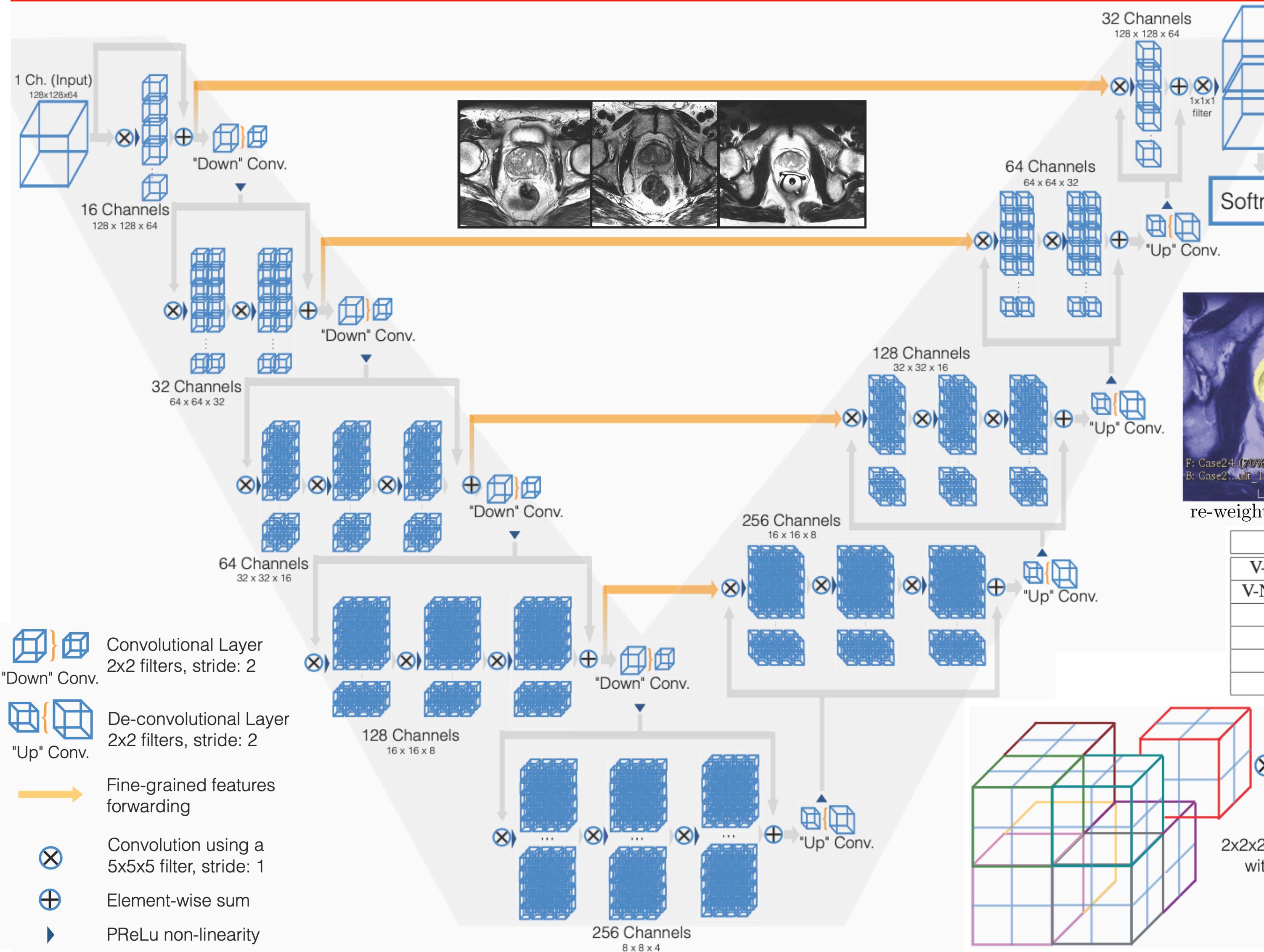
Department of Applied Mathematics

University of Colorado Boulder

[maziar.raissi@colorado.edu](mailto:maziar.raissi@colorado.edu)



# V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation



## Dice Loss

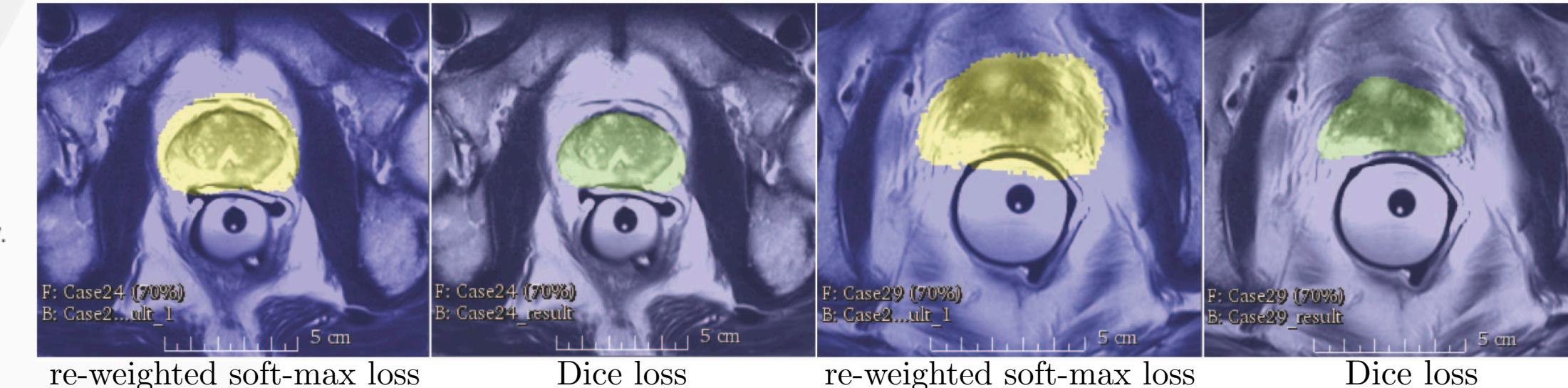
Class Imbalance! (Foreground v.s. Background)

$$D = \frac{2 \sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

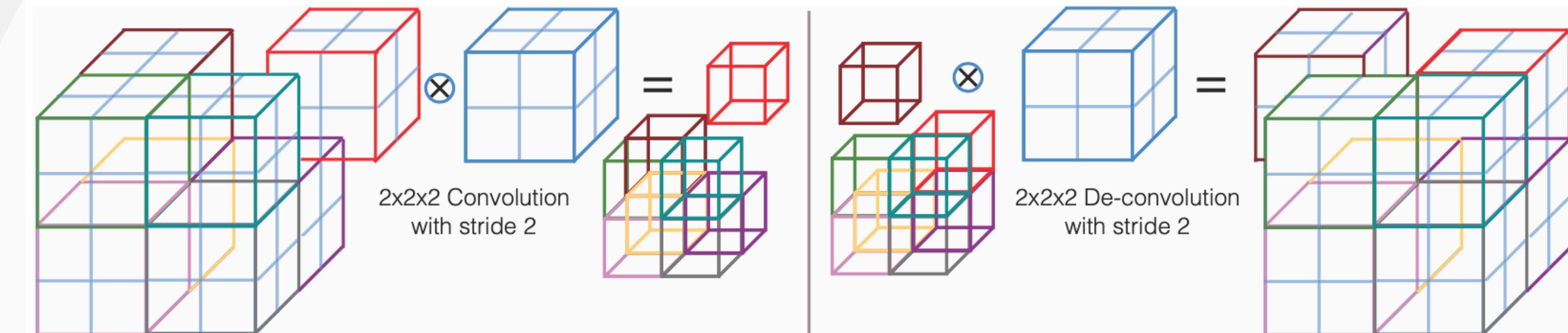
$N \rightarrow$  number of voxels

$p_i \in P \rightarrow$  predicted binary segmentation volume  
 $g_i \in G \rightarrow$  ground truth binary volume

If  $g_i = 0$ , the background does not contribute to the loss!



Algorithm	Avg. Dice	Avg. Hausdorff distance	Score on challenge task	Speed
V-Net + Dice-based loss	$0.869 \pm 0.033$	$5.71 \pm 1.20$ mm	82.39	1 sec.
V-Net + mult. logistic loss	$0.739 \pm 0.088$	$10.55 \pm 5.38$ mm	63.30	1 sec.
Imorphics [22]	$0.879 \pm 0.044$	$5.935 \pm 2.14$ mm	84.36	8 min.
ScrAutoProstate	$0.874 \pm 0.036$	$5.58 \pm 1.49$ mm	83.49	1 sec.
SBIA	$0.835 \pm 0.055$	$7.73 \pm 2.68$ mm	78.33	–
Grislies	$0.834 \pm 0.082$	$7.90 \pm 3.82$ mm	77.55	7 min.

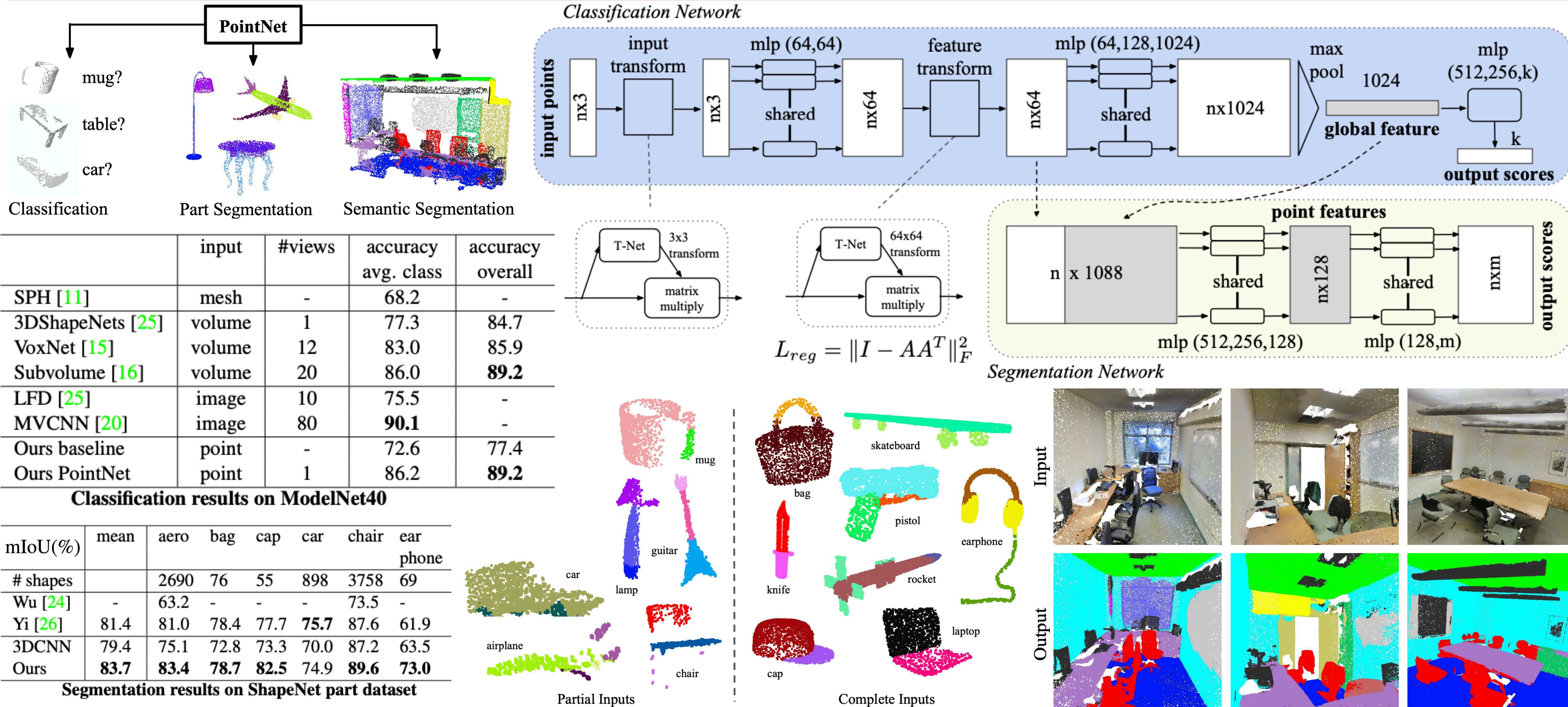




# PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation



[YouTube Video](#)



# PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space


[YouTube Video](#)

Visualization of a scan captured from a Structure Sensor



## PointNet

$\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d \rightarrow$  unordered point set  
 $f(x_1, x_2, \dots, x_n) = \gamma(\max_{i=1, \dots, n}\{h(x_i)\})$   
 $\gamma, h \rightarrow \text{MLP}$

## Hierarchical Point Set Feature Learning

### Local Context

- set abstraction levels
- sampling layer (centroids of local regions)
- grouping layer (finding “neighboring” points around centroids)
- PointNet layer (mini-PointNet)

### Sampling Layer

Iterative Farthest Point Sampling (FPS)

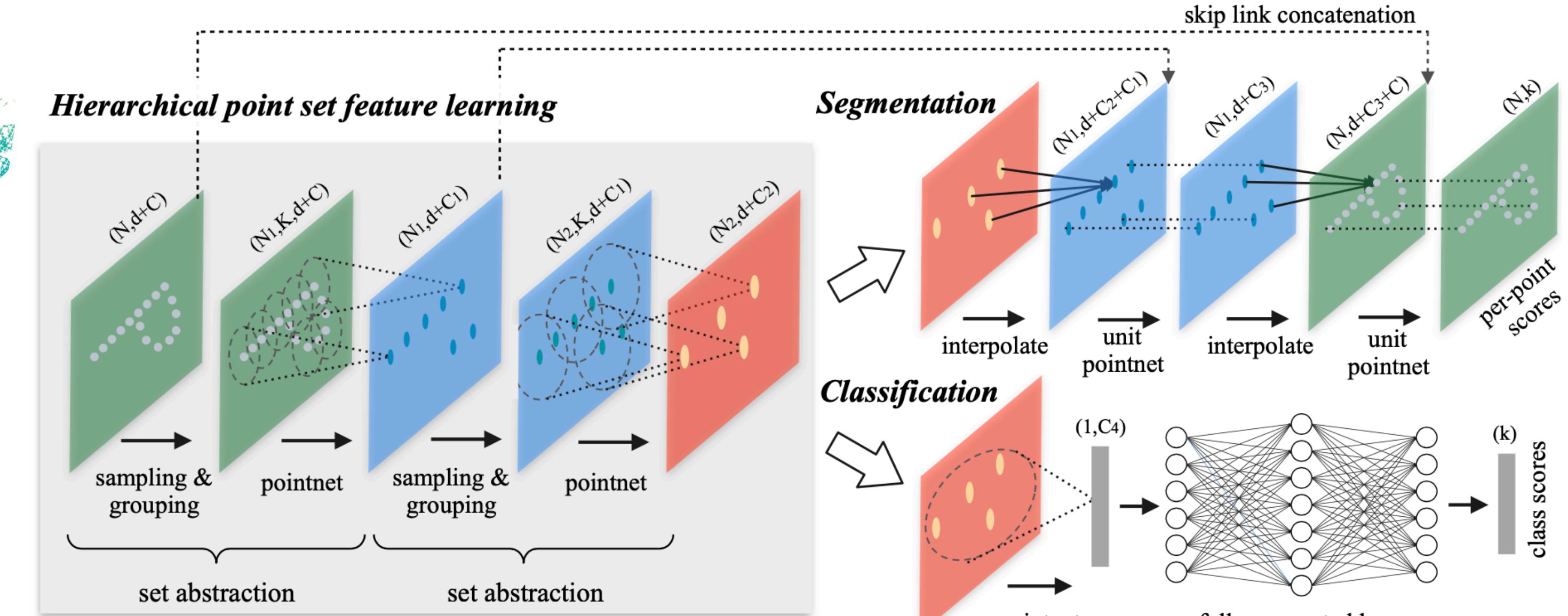
$$\{x_{i_1}, x_{i_2}, \dots, x_{i_m}\} \subset \{x_1, x_2, \dots, x_n\}$$

$x_{i_j} \rightarrow$  the most distant point from the set  $\{x_{i_1}, \dots, x_{i_{j-1}}\}$

### Grouping Layer

$K \rightarrow$  number of points in the neighborhood of centroid points

Ball query works better than kNN!



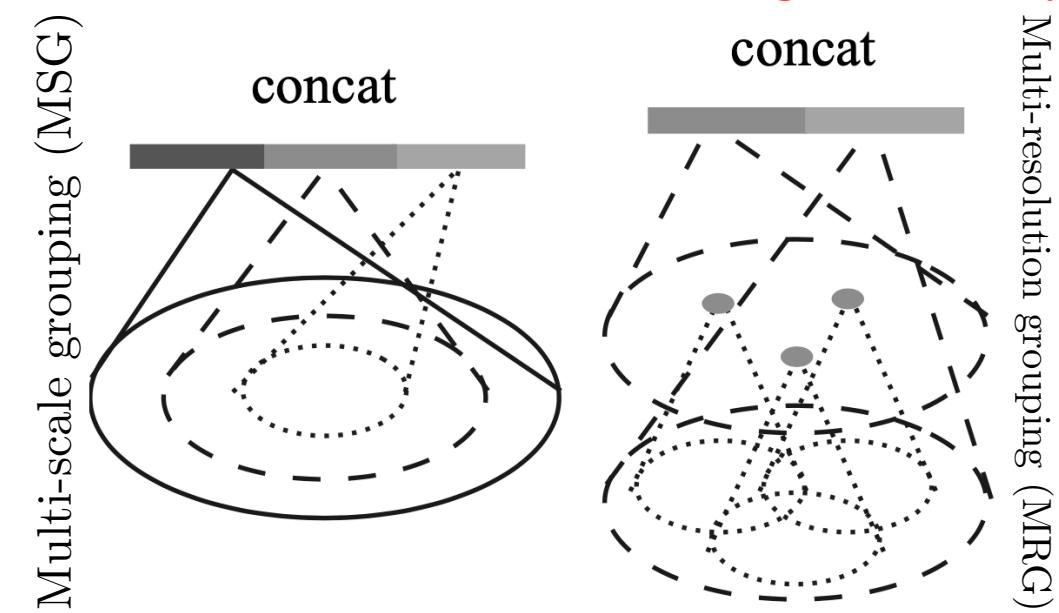
## PointNet Layer

Relative coordinates

$$x_i^{(j)} \leftarrow x_i^{(j)} - \hat{x}^{(j)}, i = 1, \dots, K, j = 1, \dots, C$$

centroid

## Non-Uniform Sampling Density



## Random Input Dropout

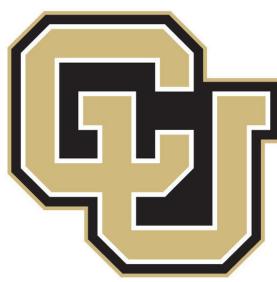
For each training point set, choose a dropout ratio  $\theta \sim U[0, p]$  (e.g.,  $p = 0.95$ ). For each training point, randomly drop the point with probability  $\theta$ .

## Segmentation

$$f^{(j)}(x) = \frac{\sum_{i=1}^k w_i(x) f_i^{(j)}}{\sum_{i=1}^k w_i(x)}$$

$$\text{where } w_i(x) = \frac{1}{d(x, x_i)^p}, j = 1, \dots, C$$

Inverse Distance Weighted Interpolation averaged based on  $k$  nearest neighbors



Boulder

# Dynamic Graph CNN for Learning on Point Clouds



[YouTube Video](#)

EdgeConv → edge convolution

- indoor navigation
- self-driving vehicles
- robotics
- shape synthesis & modeling

LiDAR scanners

- ModelNet40
- ShapeNetPart
- S3DIS

## Edge Convolution

$$X = \{x_1, \dots, x_n\} \subset \mathbb{R}^F$$

$\underbrace{\quad}_{F\text{-dimensional point cloud with } n \text{ points}}$

$$F = 3 \rightarrow x_i = (x_i, y_i, z_i)$$

color, surface, normal, etc.

$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \rightarrow$  directed graph representing local point cloud structure

$$\mathcal{V} = \{1, \dots, n\} \rightarrow \text{vertices}$$

$$\mathcal{E} \subset \mathcal{V} \times \mathcal{V} \rightarrow \text{edges}$$

$k$ -nearest neighbor ( $k$ -NN) graph of  $X$  in  $\mathbb{R}^F$

The graph includes self-loop

$$e_{ij} = h_\theta(x_i, x_j) \text{ where } h_\theta : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$$

$\underbrace{\quad}_{\text{edge feature}}$

$$x'_i = \square_{j:(i,j) \in \mathcal{E}} h_\theta(x_i, x_j) \quad \square = \sum \text{ or } \max$$

## Choice of $h$ & $\square$

$$x'_{im} = \sum_{j:(i,j) \in \mathcal{E}} \theta_m \cdot x_j \rightarrow \text{standard convolution}$$

$\underbrace{\quad}_{m\text{-th filter}}$

$$h_\theta(x_i, x_j) = h_\theta(x_i) \rightarrow \text{PointNet}$$

$$h_\theta(x_i, x_j) = h_\theta(x_j) \rightarrow \text{PointCNN}$$

$$x'_{im} = \sum_{j \in \mathcal{V}} h_\theta(x_j) g(u(x_i, x_j))$$

$\underbrace{\quad}_{\text{pairwise distance}}$

Gaussian kernel in Euclidean space

$$h_\theta(x_i, x_j) = h_\theta(x_i - x_j) \rightarrow \text{encode local information}$$

$$h_\theta(x_i, x_j) = \bar{h}_\theta(x_i, x_i - x_j) \rightarrow \text{encode global \& local information}$$

$$e'_{ijm} = \text{ReLU}(\theta_m \cdot (x_j - x_i) + \phi_m \cdot x_i)$$

$$x'_{im} = \max_{j:(i,j) \in \mathcal{E}} e'_{ijm}$$

## Dynamic Graph CNN (DGCNN)

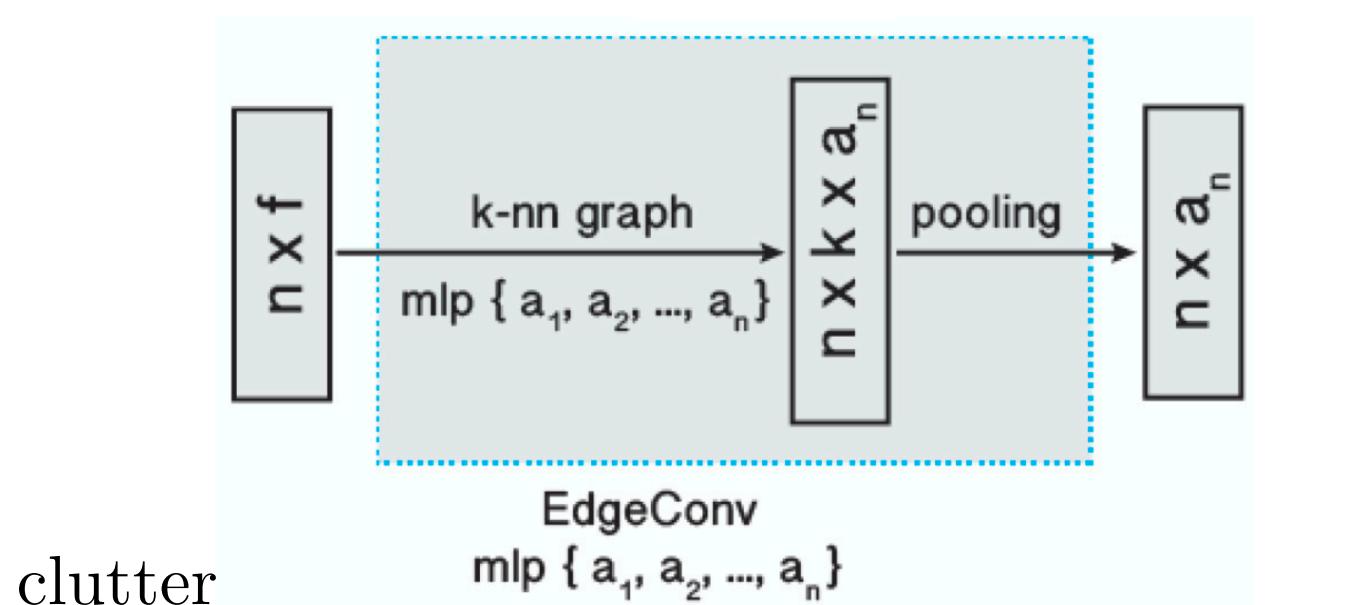
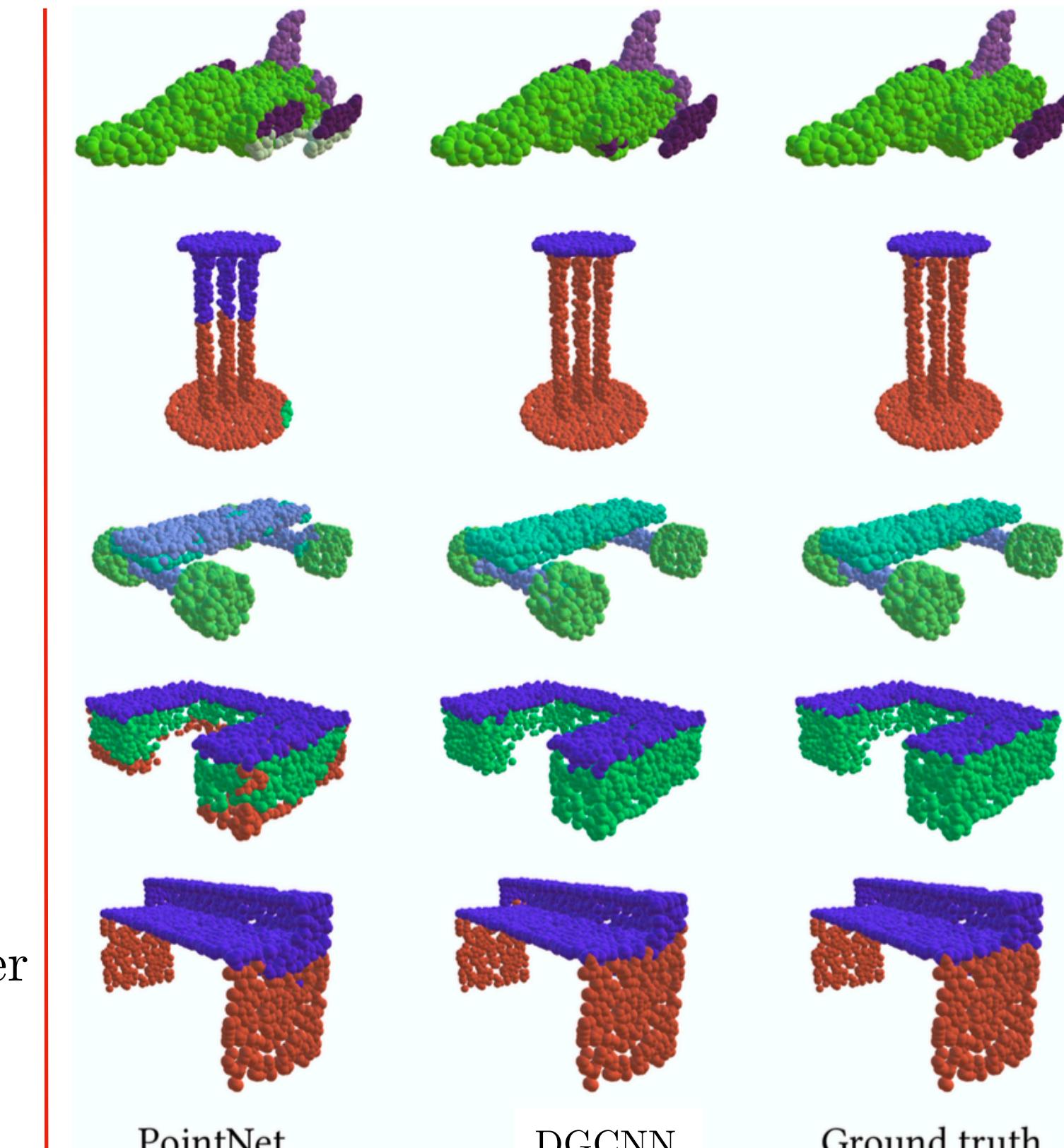
$\mathcal{G}^\ell = (\mathcal{V}^\ell, \mathcal{E}^\ell) \rightarrow$  dynamically constructed per each layer

## Datasets

ModelNet40 contains 12,311 meshed CAD models from 40 categories.

ShapeNet part dataset contains 16,881 3D shapes from 16 object categories, annotated with 50 parts in total.

Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) includes 3D scan point clouds for 6 indoor areas including 272 rooms in total. Each point belongs to one of 13 semantic categories—e.g., board, bookcase, chair, ceiling, and beam—plus clutter





Boulder

# Point Transformer

– autonomous driving – augmented reality – robotics

S3DIS (Stanford large-scale 3D Indoor Spaces): semantic scene segmentation

ShapeNetPart: object part segmentation

ModelNet40: shape classification

## Transformers Background

– scalar attention – vector attention

$\mathcal{X} = \{x_i\}$  → set of feature vectors

The set can be a collection of feature vectors that represent the entire signal (e.g., sentence or image) or a collection of feature vectors from a local patch within the signal (e.g., an image patch).

$$y_i = \sum_{x_j \in \mathcal{X}} \rho(\varphi(x_i)^T \psi(x_j) + \delta) \alpha(x_j) \rightarrow \text{standard scalar dot-product attention layer}$$

$y_i \rightarrow$  output feature

$\varphi, \psi, \alpha \rightarrow$  feature transformations (linear projections or MLPs)

$\delta \rightarrow$  position encoding function

$\rho \rightarrow$  normalization function (e.g., softmax)

$$y_i = \sum_{x_j \in \mathcal{X}} \rho(\gamma(\beta(\varphi(x_i), \psi(x_j)) + \delta)) \odot \alpha(x_j) \rightarrow \text{vector attention} \\ (\text{can modulate individual feature channels})$$

$\beta \rightarrow$  relation function (e.g., subtraction)

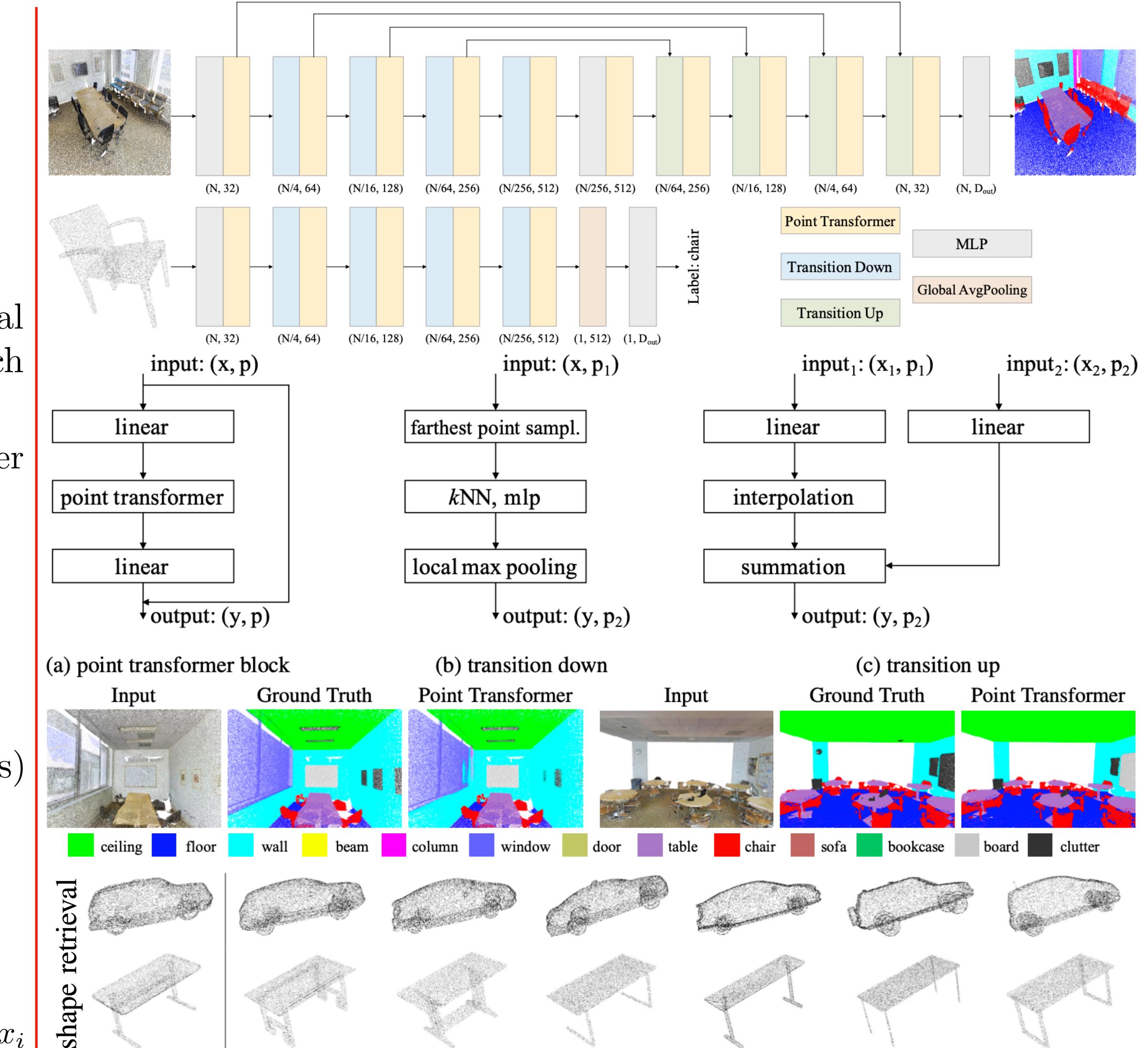
$\gamma \rightarrow$  mapping function (e.g., an MLP)

## Point Transformer Layer

$$y_i = \sum_{x_j \in \mathcal{X}(i)} \rho(\gamma(\varphi(x_i) - \psi(x_j) + \delta)) \odot (\alpha(x_j) + \delta)$$

$\mathcal{X}(i) \subset \mathcal{X} \rightarrow$  set of points in a local neighborhood (specifically,  $k$  nearest neighbors) of  $x_i$

$\delta = \theta(p_i - p_j) \rightarrow$  position encoding     $p_i, p_j \rightarrow$  3D point coordinates for points  $i$  and  $j$

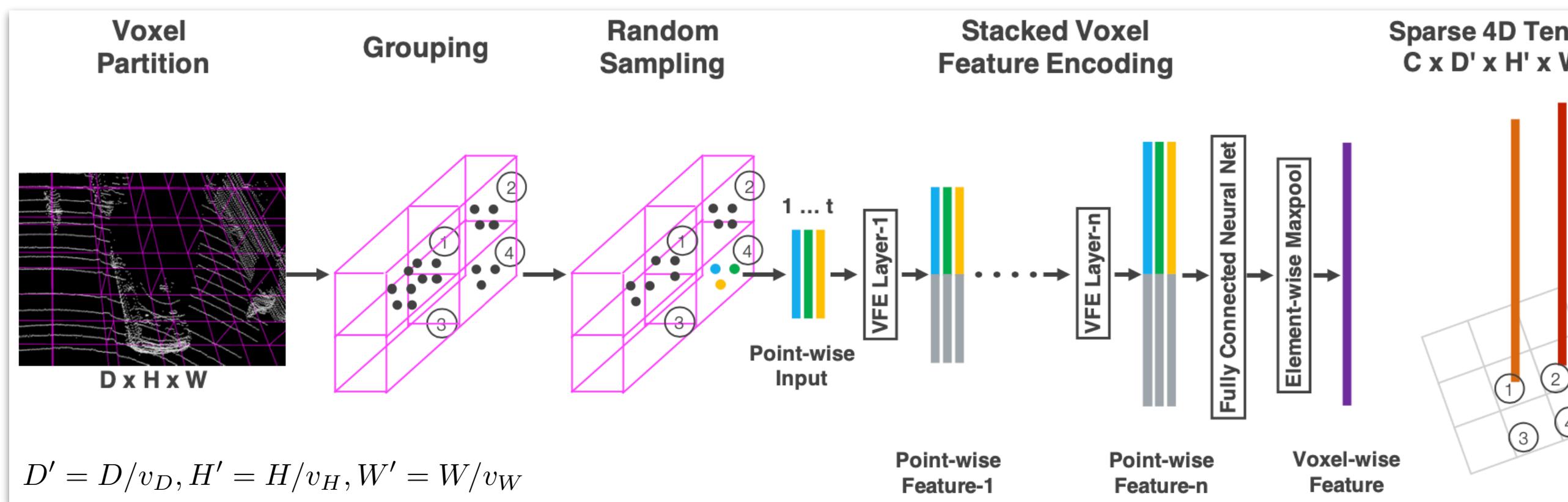
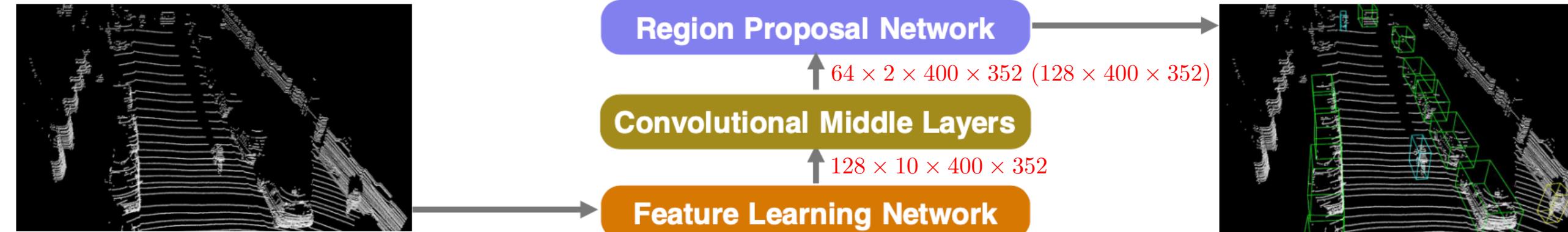




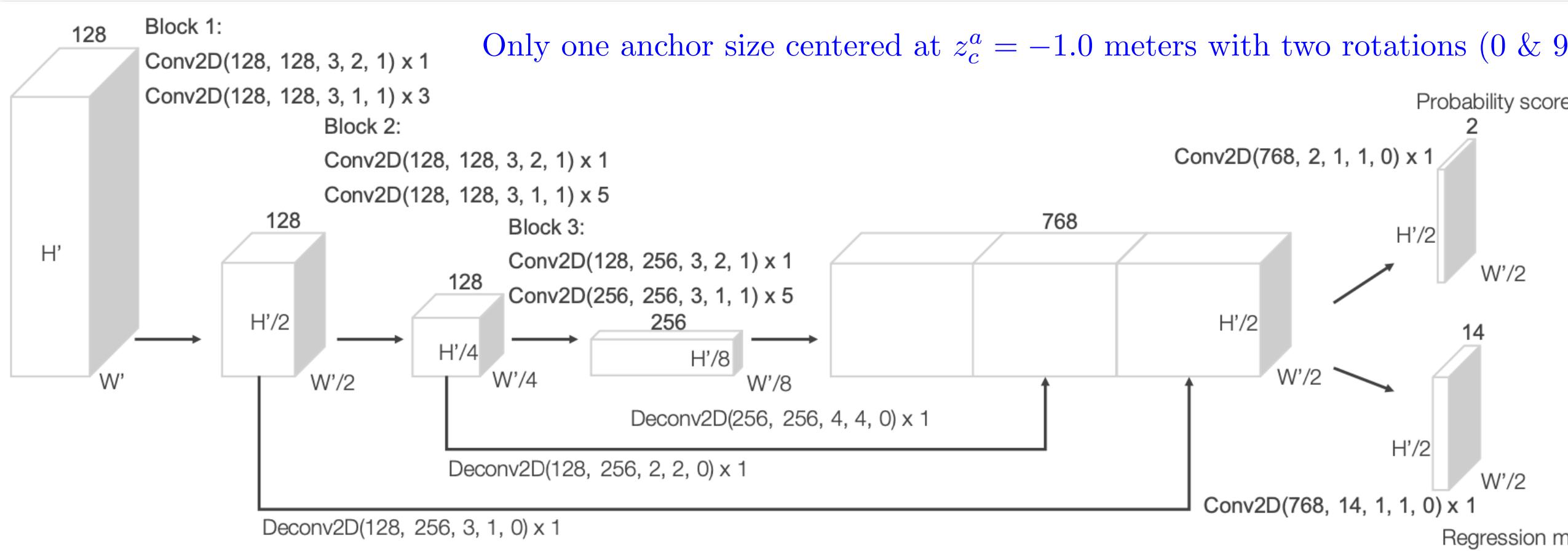
Boulder

# VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection

– autonomous navigation – housekeeping robots – augmented/virtual reality

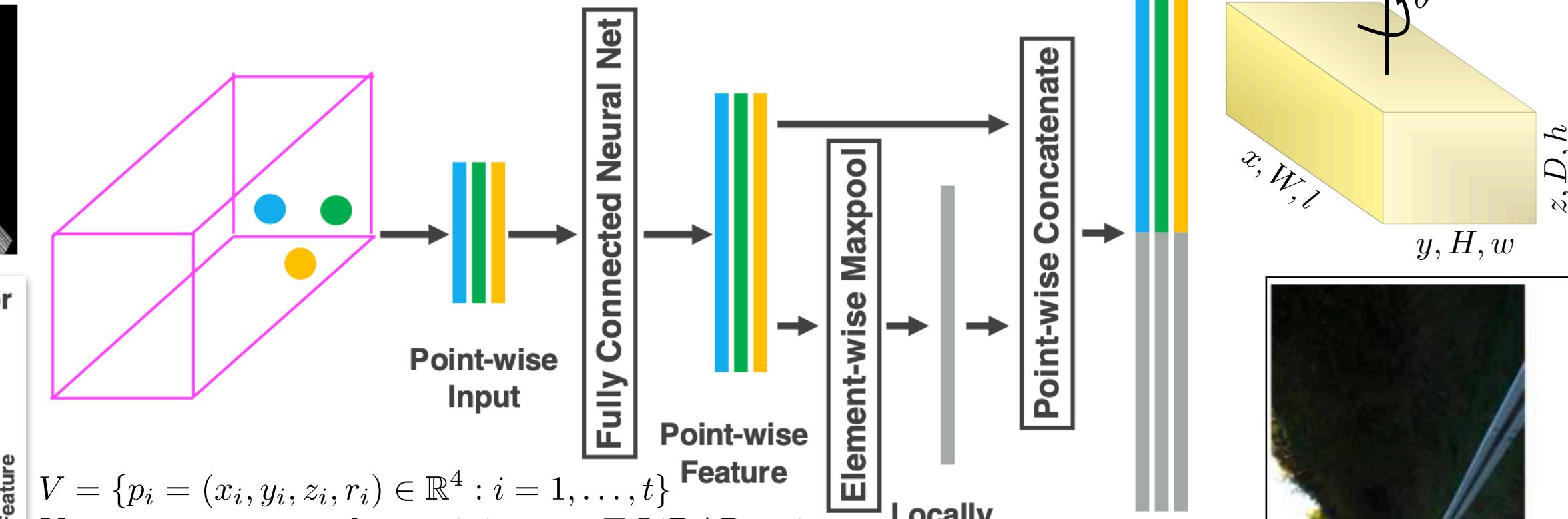


$$D' = D/v_D, H' = H/v_H, W' = W/v_W$$



Zhou, Yin, and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

## Voxel Feature Encoding



$V = \{p_i = (x_i, y_i, z_i, r_i) \in \mathbb{R}^4 : i = 1, \dots, t\}$   
 $V \rightarrow$  nonempty voxel containing  $t \leq T$  LiDAR points  
 $r_i \rightarrow$  received reflectance  
 $(v_x, v_y, v_z) \rightarrow$  local mean as the centroid of all points in  $V$   
 $V_{in} = \{\hat{p}_i = (x_i, y_i, z_i, r_i, x_i - v_x, y_i - v_y, z_i - v_z) \in \mathbb{R}^7 : i = 1, \dots, t\}$   
 $V_{in} \rightarrow$  input feature set

## Loss Function

$\{a_i^{pos} : i = 1, \dots, N_{pos}\} \rightarrow$  set of positive anchors  
 $\{a_j^{neg} : j = 1, \dots, N_{neg}\} \rightarrow$  set of negative anchors  
 $(x_c^g, y_c^g, z_c^g, l^q, w^g, h^g, \theta^g) \rightarrow$  3D ground truth box  
 $(x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a) \rightarrow$  matching positive anchor  
 $d^a = \sqrt{(l^a)^2 + (w^a)^2} \rightarrow$  diagonal of the base of the anchor box  
 $L = \alpha \frac{1}{N_{pos}} \sum_i L_{cls}(p_i^{pos}, 1) + \beta \frac{1}{N_{neg}} \sum_j L_{cls}(p_j^{neg}, 0) + \frac{1}{N_{pos}} \sum_i L_{reg}(\mathbf{u}_i, \mathbf{u}_i^*)$   
 $\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a},$   
 $\Delta l = \log(\frac{l^g}{l^a}), \Delta w = \log(\frac{w^g}{w^a}), \Delta h = \log(\frac{h^g}{h^a}),$   
 $\Delta \theta = \theta^g - \theta^a$   
 $u^* \in \mathbb{R}^7 \rightarrow$  residual vector  
**Data Augmentation & Efficient Implementation of the**  $\rightarrow$  (see the paper!)  
**Feature Learning Network**





Boulder

# NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



$\mathbf{x} = (x, y, z) \rightarrow$  3D location (input)  
 $(\theta, \phi) \rightarrow$  2D viewing direction (input)  
 $\mathbf{d} \rightarrow$  3D viewing direction (unit vector)  
 $\mathbf{c} = (r, g, b) \rightarrow$  emitted color (output)  
 $\sigma \rightarrow$  volume density (output)  
 $F_\Theta : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$   
 ↪ approximate scene representation

Multi-view consistency constraint

$$\mathbf{x} \rightarrow \begin{cases} \sigma \\ \mathbf{h} \\ \mathbf{d} \end{cases} \rightarrow \mathbf{c}$$

Volume Rendering with Radiance Fields  
 $\sigma(\mathbf{x}) \rightarrow$  volume density (differential probability of a ray terminating at an infinitesimal particle at location  $\mathbf{x}$ )

$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d} \rightarrow$  camera ray with near and far bounds  $t_n$  and  $t_f$   
 $C(\mathbf{r}) \rightarrow$  expected color of camera ray  $\mathbf{r}$

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \text{ where } T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s))ds\right)$$

$T(t) \rightarrow$  accumulated transmittance along the ray from  $t_n$  to  $t$ , i.e., the probability that the ray travels from  $t_n$  to  $t$  without hitting any other particle

Numerical Estimate

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right]$$

$$\delta_i = t_{i+1} - t_i$$

Positional Encoding

$$F_\Theta = F'_\Theta \circ \gamma$$

$$\gamma : \mathbb{R} \rightarrow \mathbb{R}^{2L}$$

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p))$$

<https://www.youtube.com/watch?v=JuH79E8rdKc>

Hierarchical Volume Sampling

Two networks: “coarse” and “fine”

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \quad w_i = T_i(1 - \exp(-\sigma_i \delta_i))$$

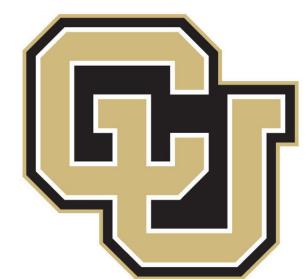
$$\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$$

piecewise-constant PDF along the ray sample a second set of  $N_f$  locations from this distribution

$N_c + N_f \rightarrow$  total number of samples

Loss  $\mathcal{R} \rightarrow$  set of rays in each batch

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$



Boulder



# Questions?

[YouTube Playlist](#)

---