UC San Diego

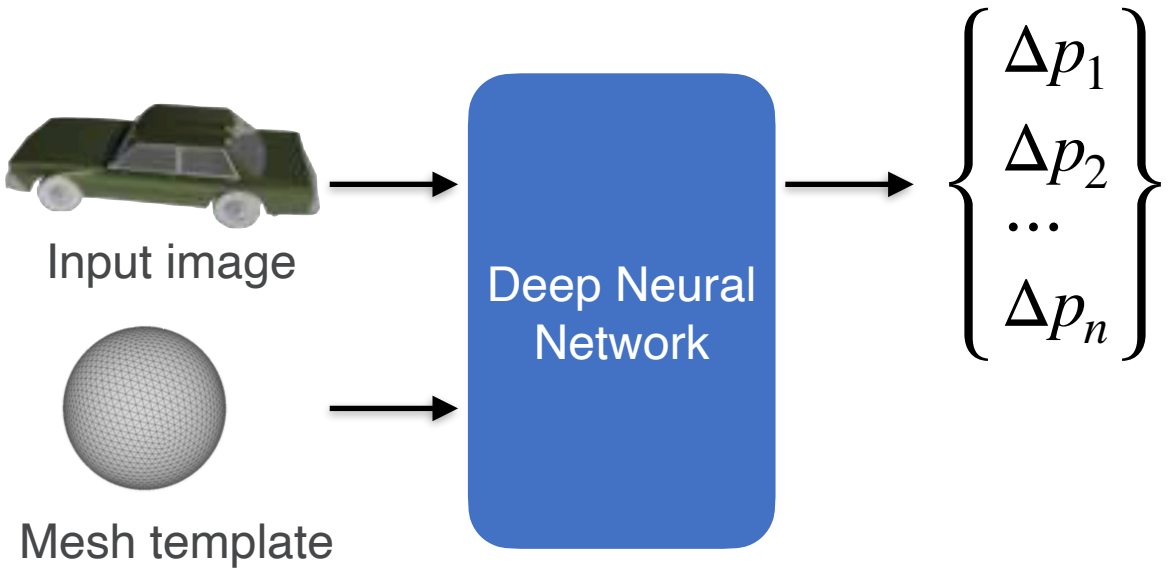
Cont. of Single-Image to 3D

Hao Su

Image to Surfaces

Editing-based Mesh Modeling

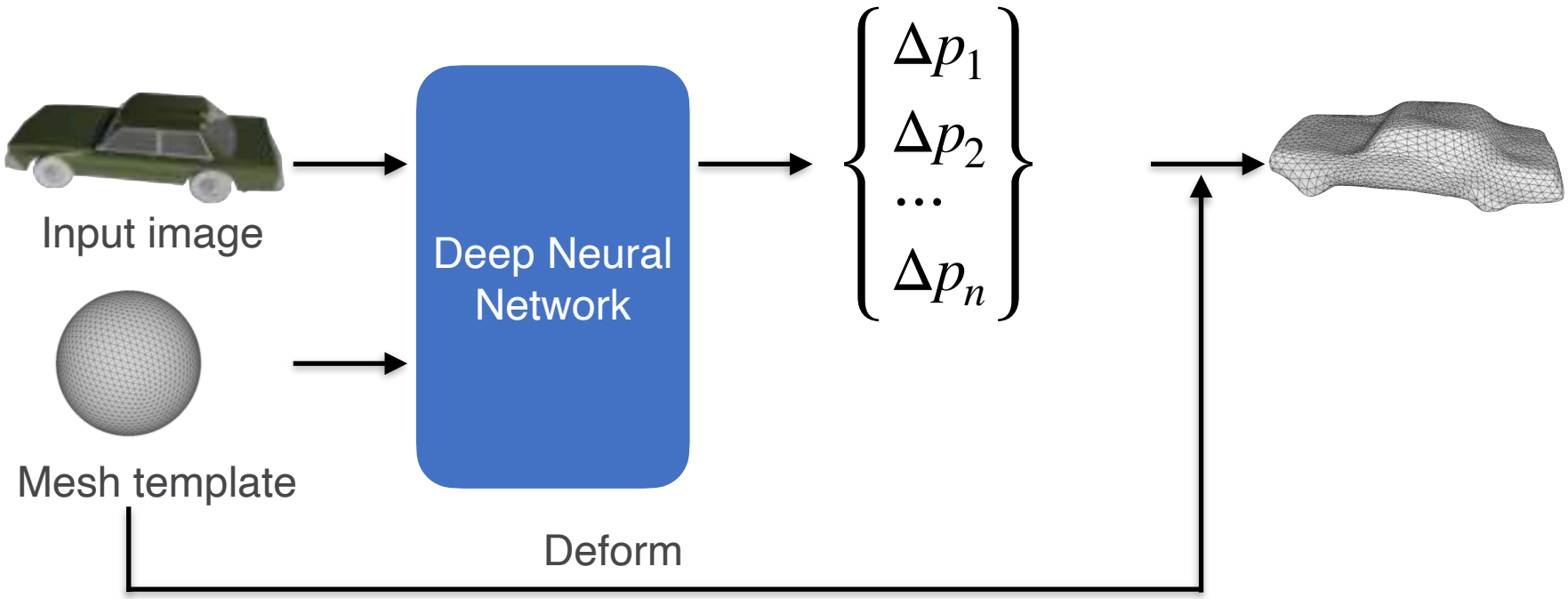
- Key idea: starting from an established mesh and modify it to become the target shape



Editing-based Mesh Modeling

- Key idea: starting from an established mesh and modify it to become the target shape

For example, deformation-based modeling:



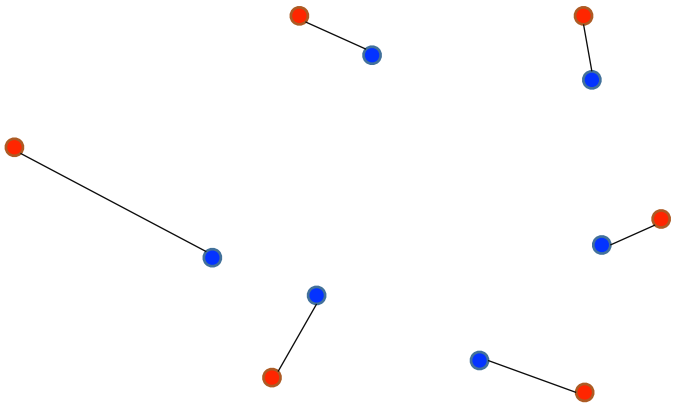
Losses for Mesh Editing

Some Example Losses

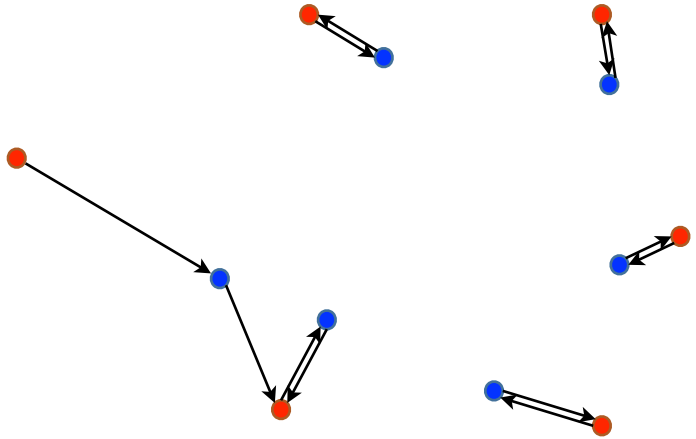
- Vertices distance.
  - Vertices point set distance.
- Uniform vertices distribution.
  - Edge length regularizer.
- Mesh surface smoothness.
- Normal Loss.

Loss I: Set Distance between Vertices

- Vertices are a set of points
- Recall the metrics for point clouds



Earth Mover's distance



Chamfer distance

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in s_1} \|x - y\|_2^2$$

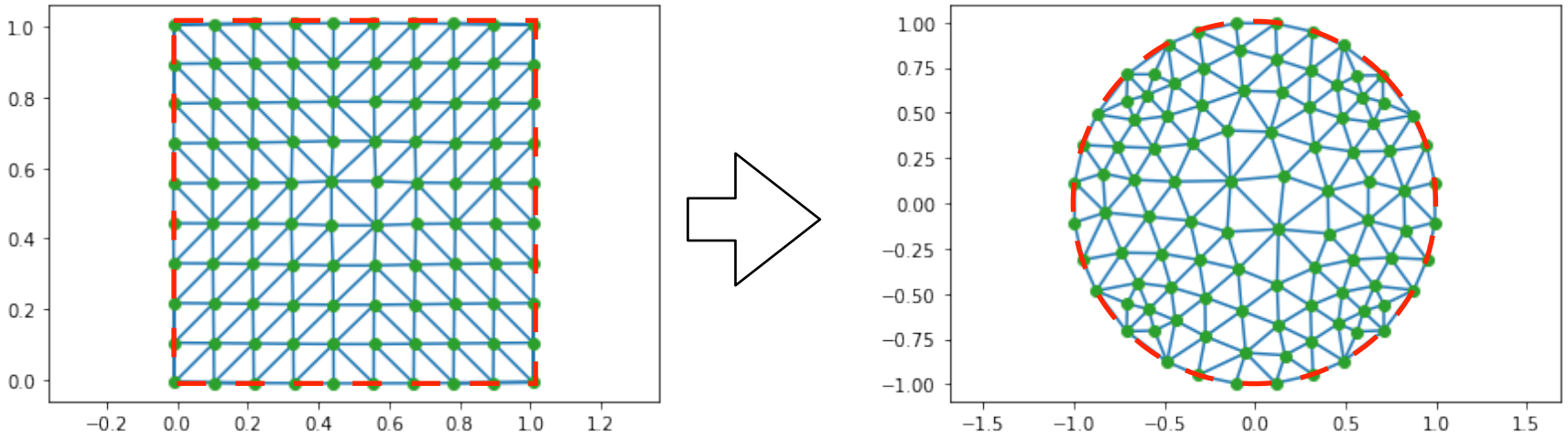
Loss II: Uniform Vertices Distribution

- Penalizes the flying vertices and overlong edges to guarantee the high quality of recovered 3D geometry
- Encourage equal edge length between vertices

$$L_{\text{unif}} = \sum_{p} \sum_{k \in N(p)} \|p - k\|_2^2$$

$$L_{\text{unif}} = \sum_{p} \sum_{k \in N(p)} \|p - k\|_2^2$$

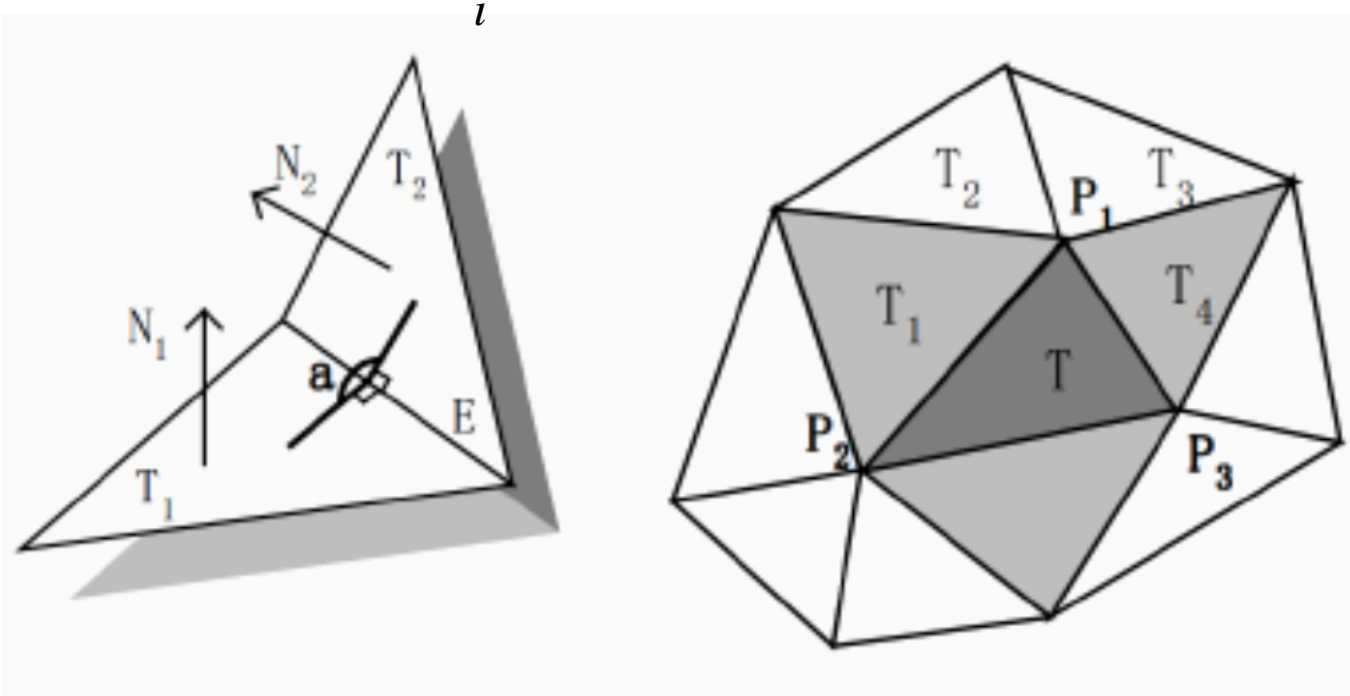
Effect of minimizing l when fixing topology and setting boundary points to the new positions



Loss III: Mesh Smoothness

- L_{smooth} encourages that intersection angles of faces are close to 180 degrees.

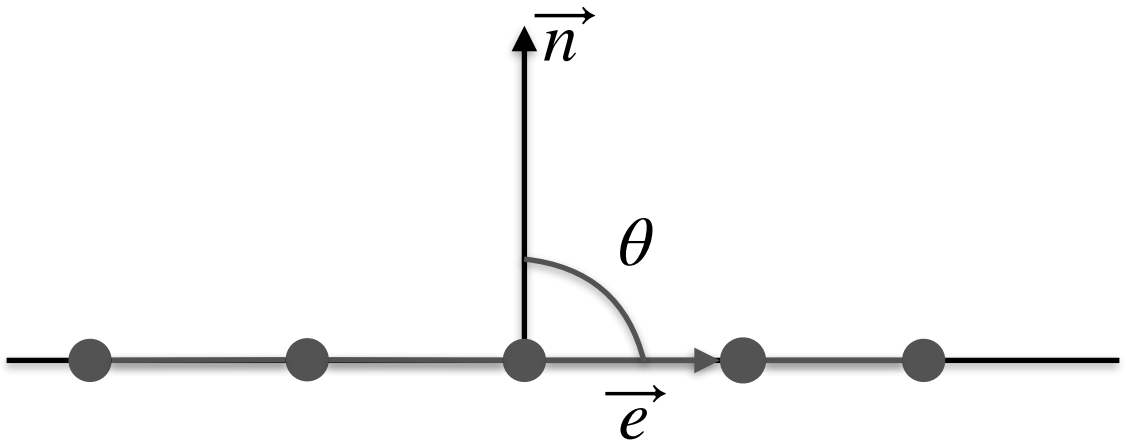
$$L_{smooth} = \sum_i (\cos\theta_i + 1)^2$$



Loss IV: Normal Loss

- **Key assumption**: vertices within a local neighborhood lie on the same tangent plane.

- Regularize edge to be perpendicular to the underlying groundtruth surface normal



Loss IV: Normal Loss

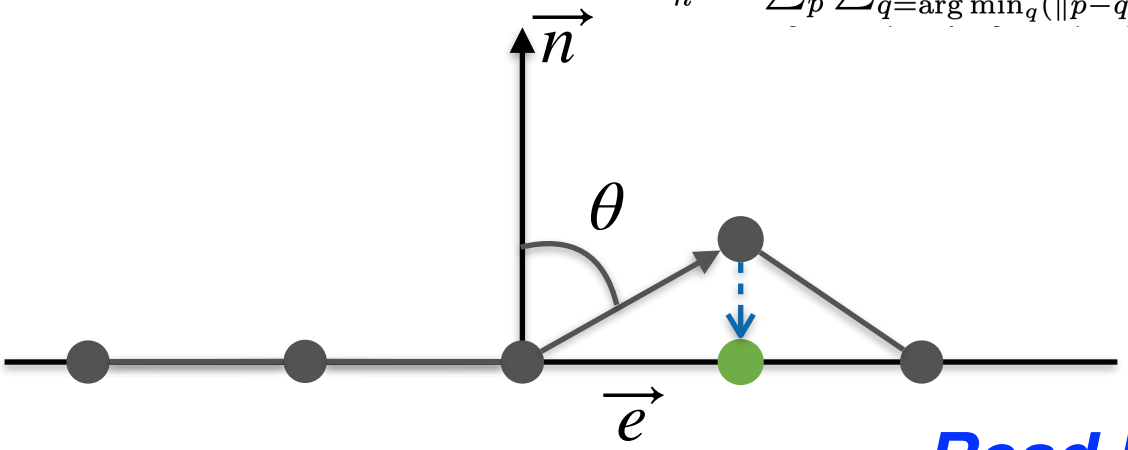
- But how to find the vertices normal?

- One approach: use the nearest ground truth point normal as current vertex normal.

Loss IV: Normal Loss

- But how to find the vertices normal?
- One approach: use the nearest ground truth point normal as current vertex normal.
- Penalize the edge direction to perpendicular to vertex normal.

$$l_n = \sum_p \sum_{q=\arg\min_q(\|p-q\|_2^2)} \|\langle p-k, \mathbf{n}_q \rangle\|_2^2$$

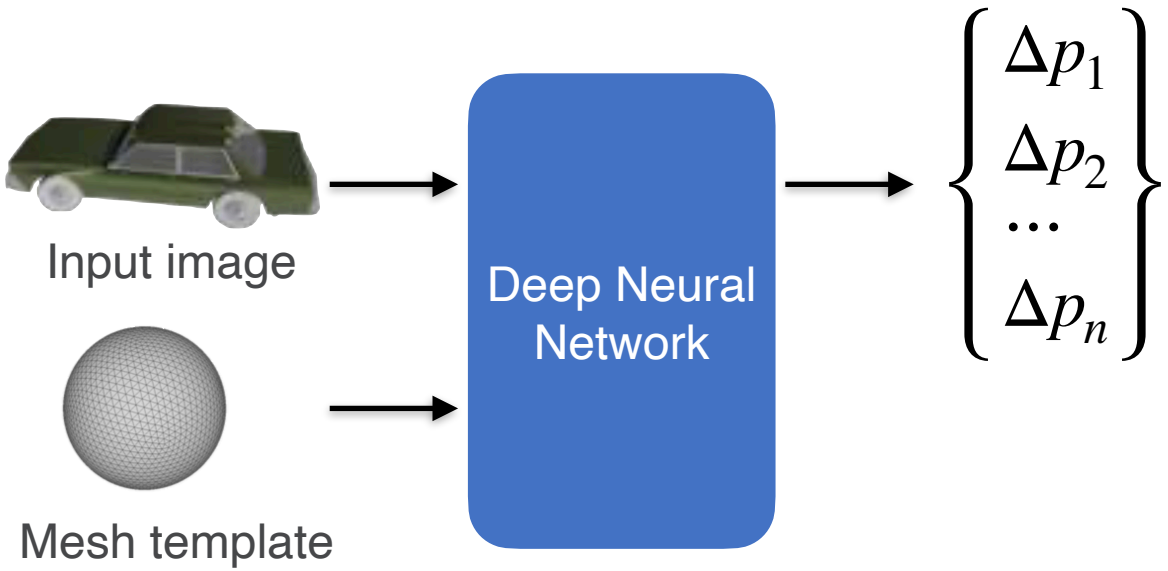


Read by Yourself

Controlling Mesh Editing

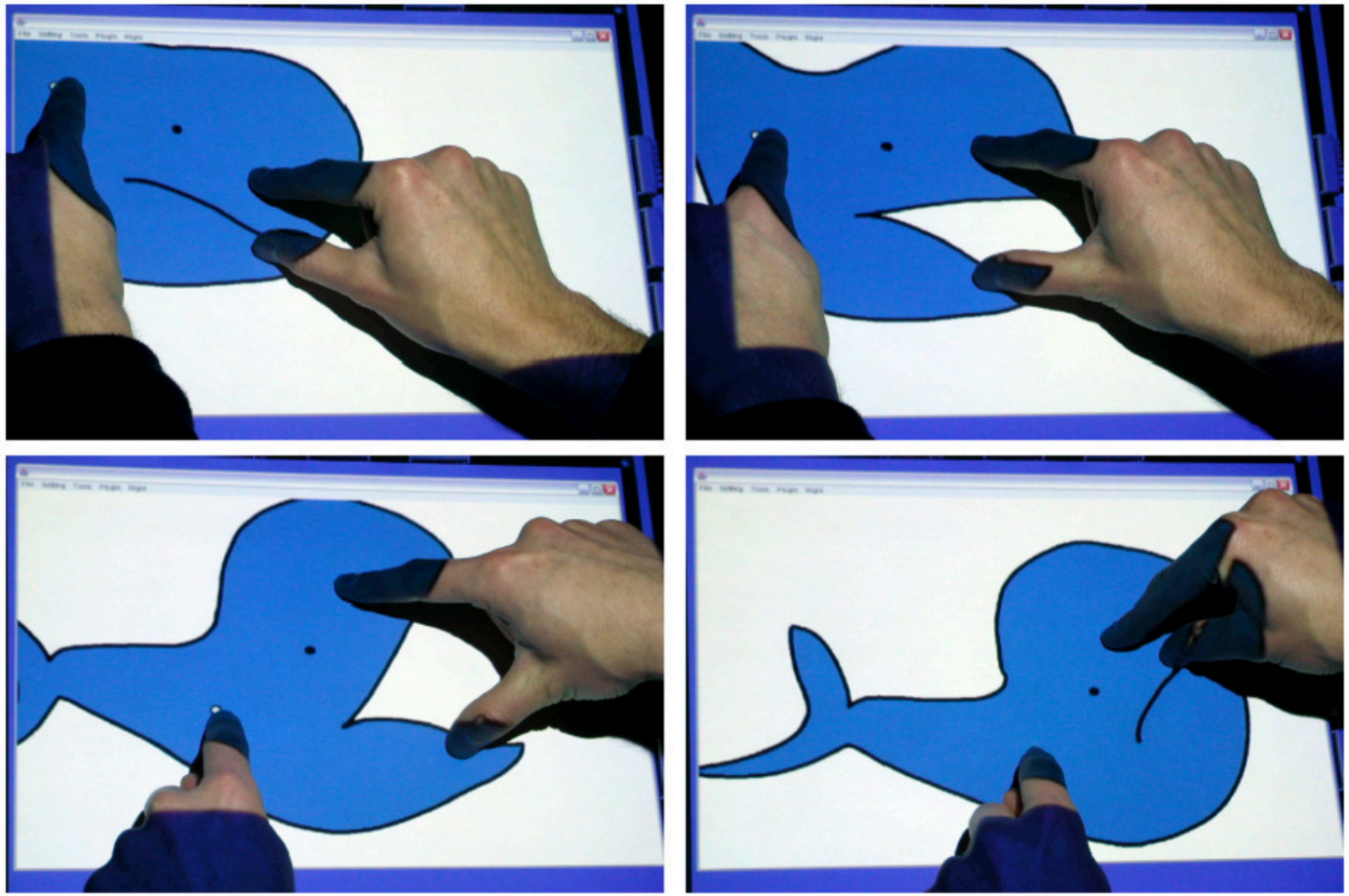
Deformation-based Mesh Modeling

- How do we parameterize movement of vertices of a template mesh?



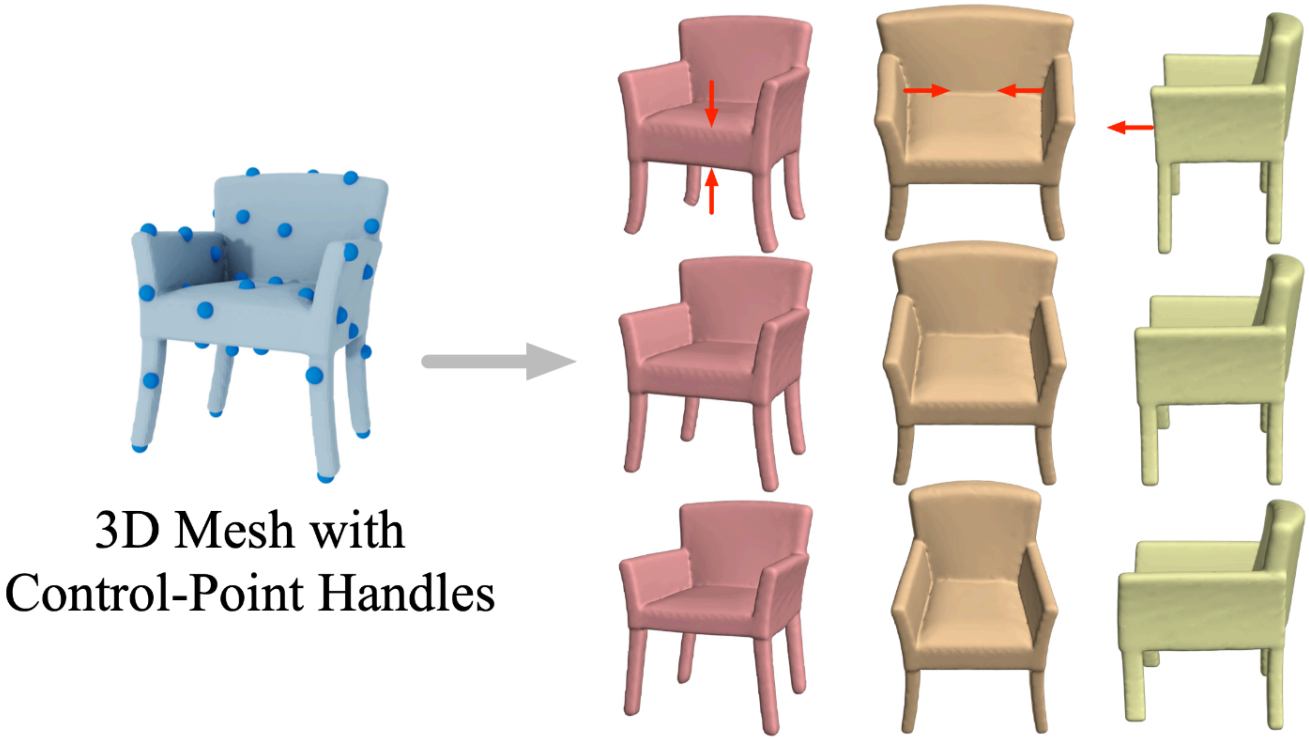
Challenges of Mesh Editing (I)

- While one can control at vertices/edges level, we may expect to find low-dimensional control handles



Possible Solution

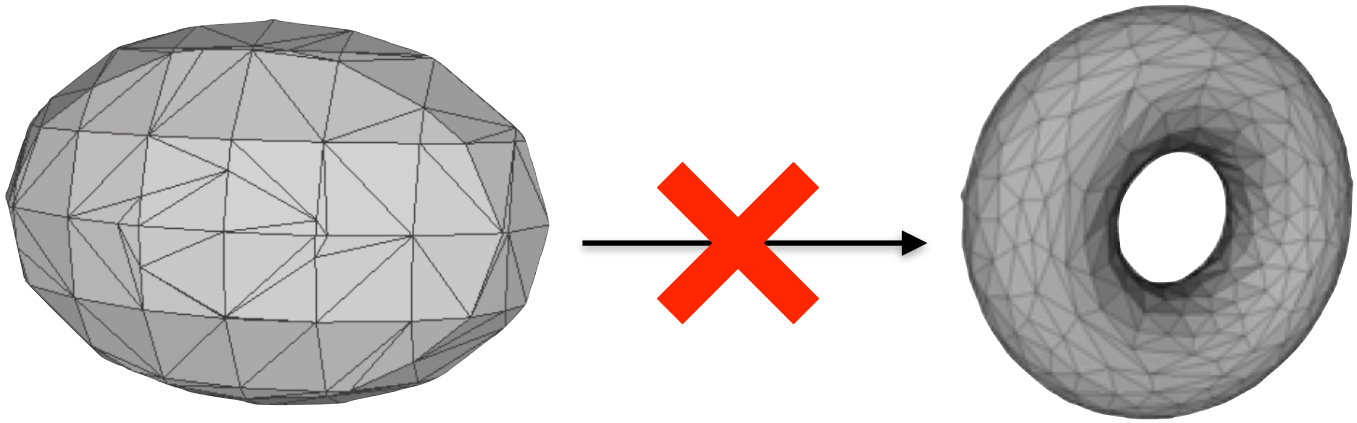
- Deformation field



Defer to later lecture

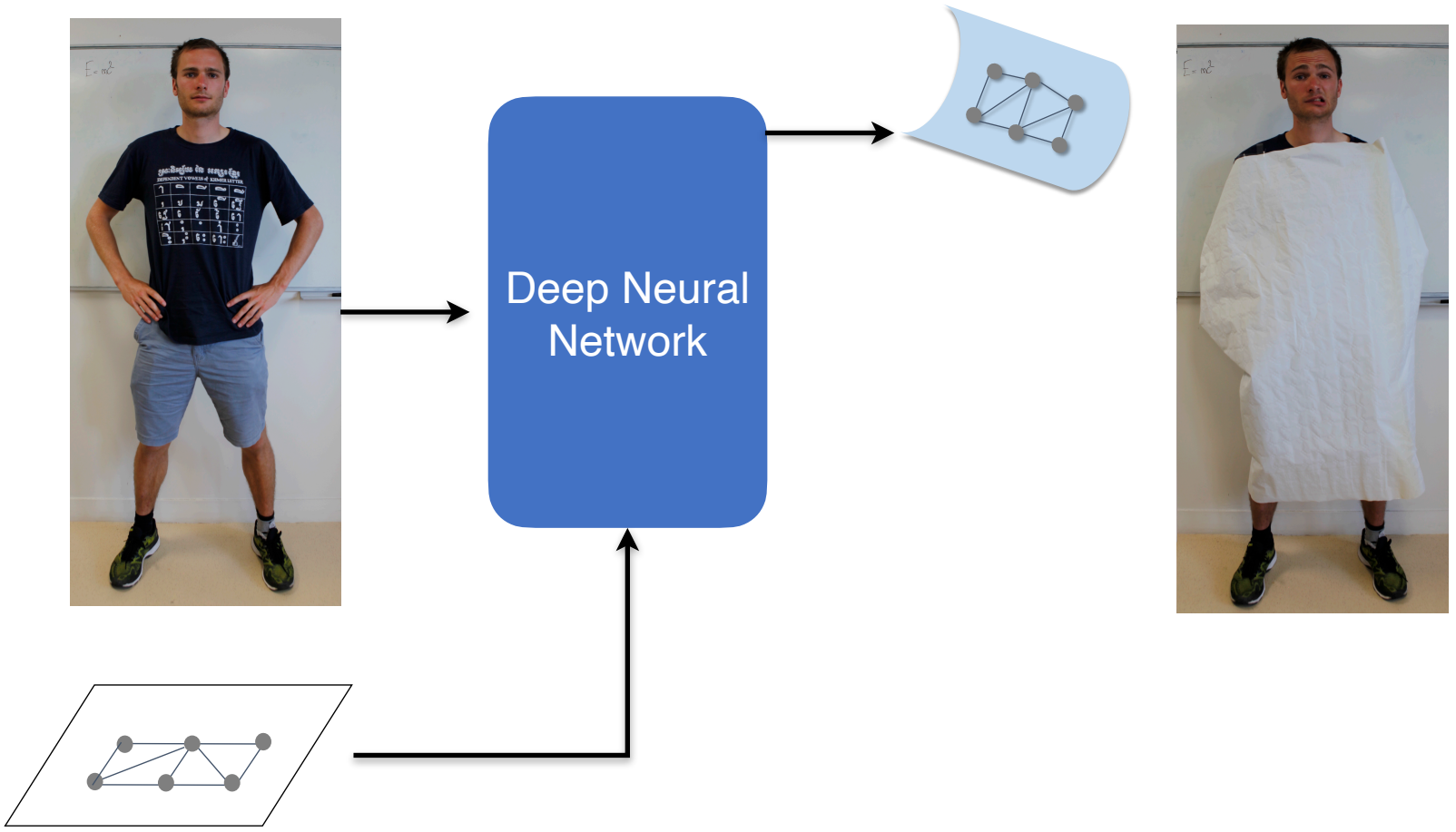
Challenges of Mesh Editing (II)

- Deformation alone is NOT able to change the topology of template mesh.



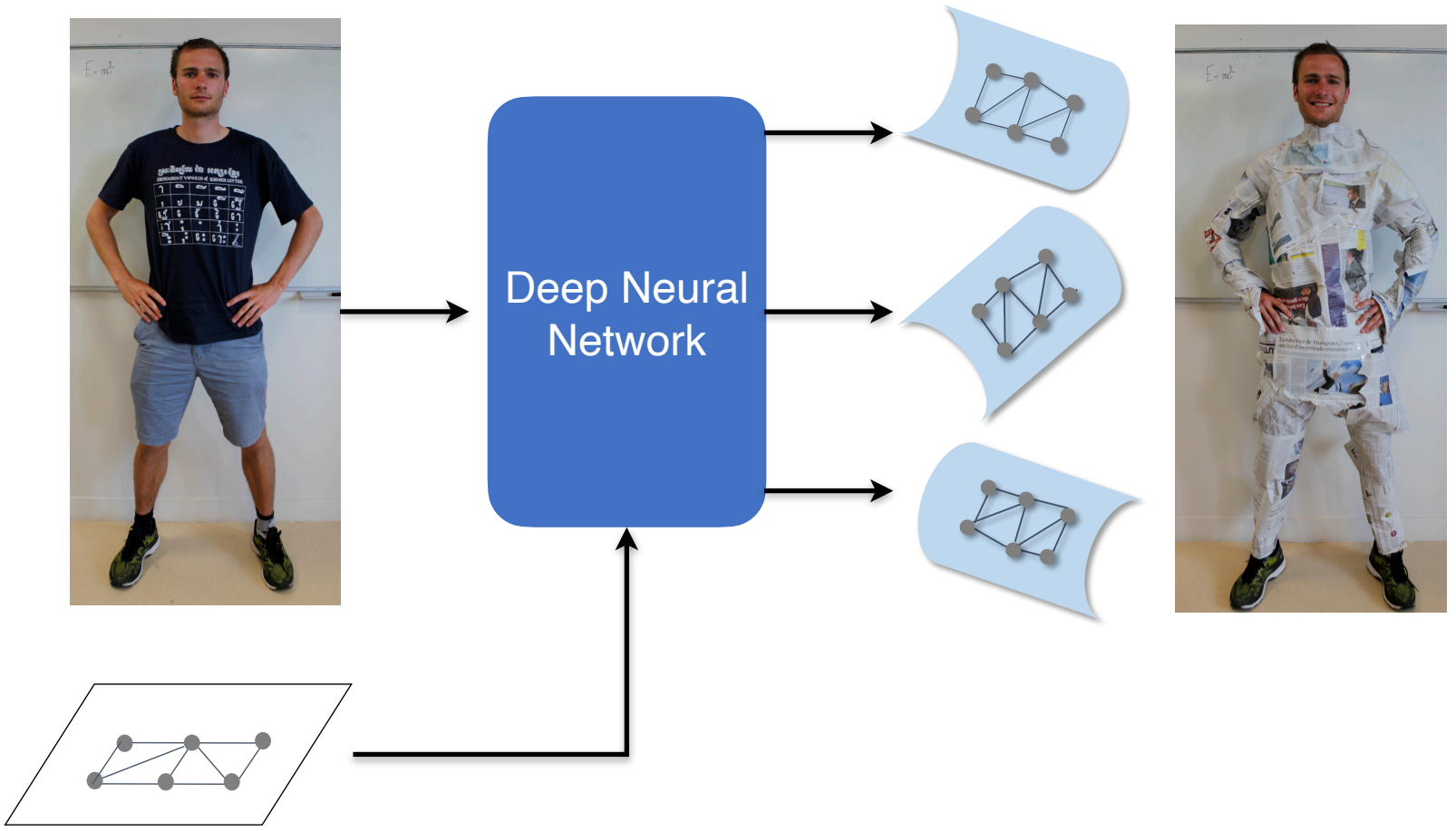
Possible Solutions

- Idea 1: Multiple template mesh.



Possible Solutions

- Idea 1: Multiple template mesh.



Possible Solutions

- Issues of multiple templates.

  - Self-intersections and overlaps caused by multiple disconnected patches.

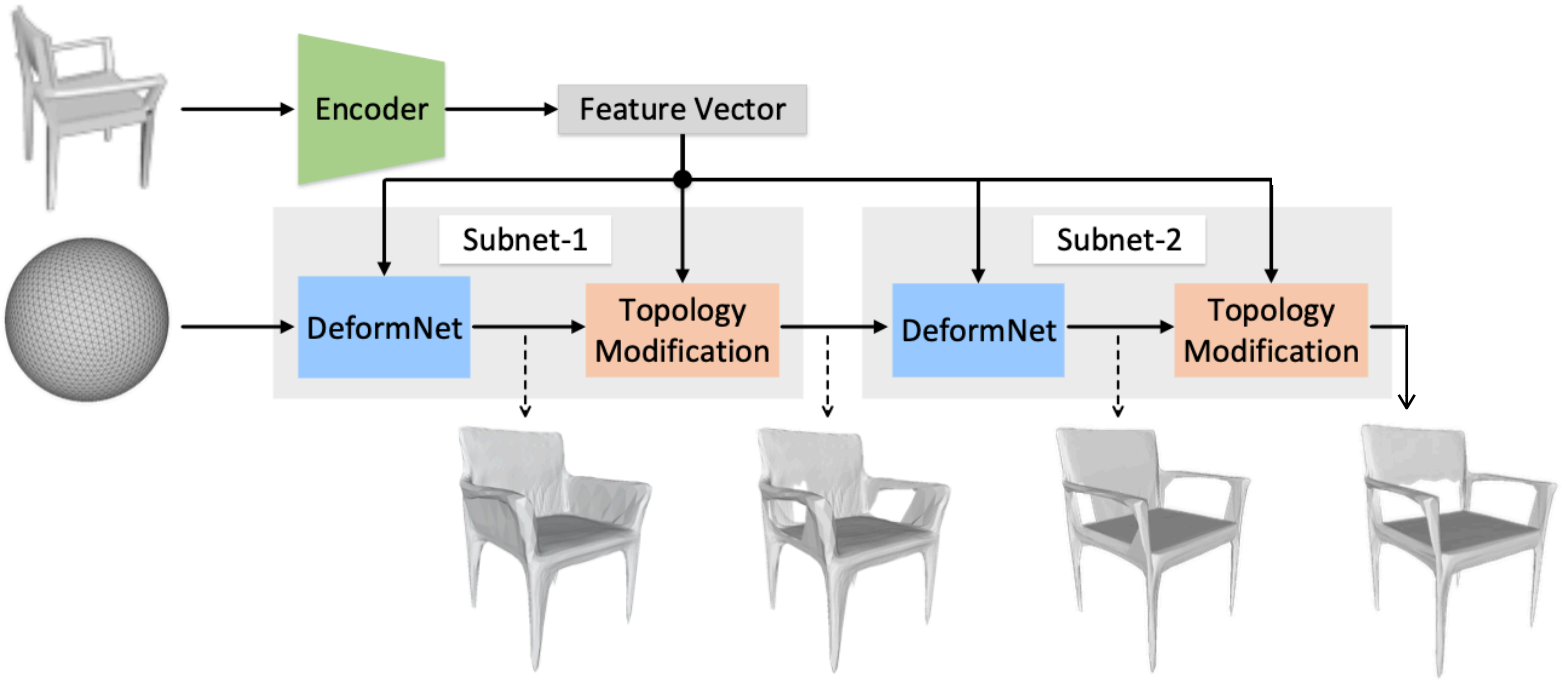
  - Hard to generate a proper deformation that can cover the surface with low distortion.



Possible Solutions

- Idea 1: Multiple template mesh.

- Idea 2: Modify template topology by remove mesh faces.



Complex Mesh Topology

- Problems of modifying topology by removing faces.
  - Nontrivial to determine a proper pruning threshold.
  - Open boundaries introduced by the face pruning.
  - Hard to generate a proper face pruning for complex shapes.

Summary

- Usually many losses have to be added
- Often have issues with local minimums in optimization
- Adding topology constraints is hard

Implicit Neural Representations

3D representations

- Voxel, point, and mesh representations ⇒ Discrete
- Implicit Neural Representation ⇒ Continuous

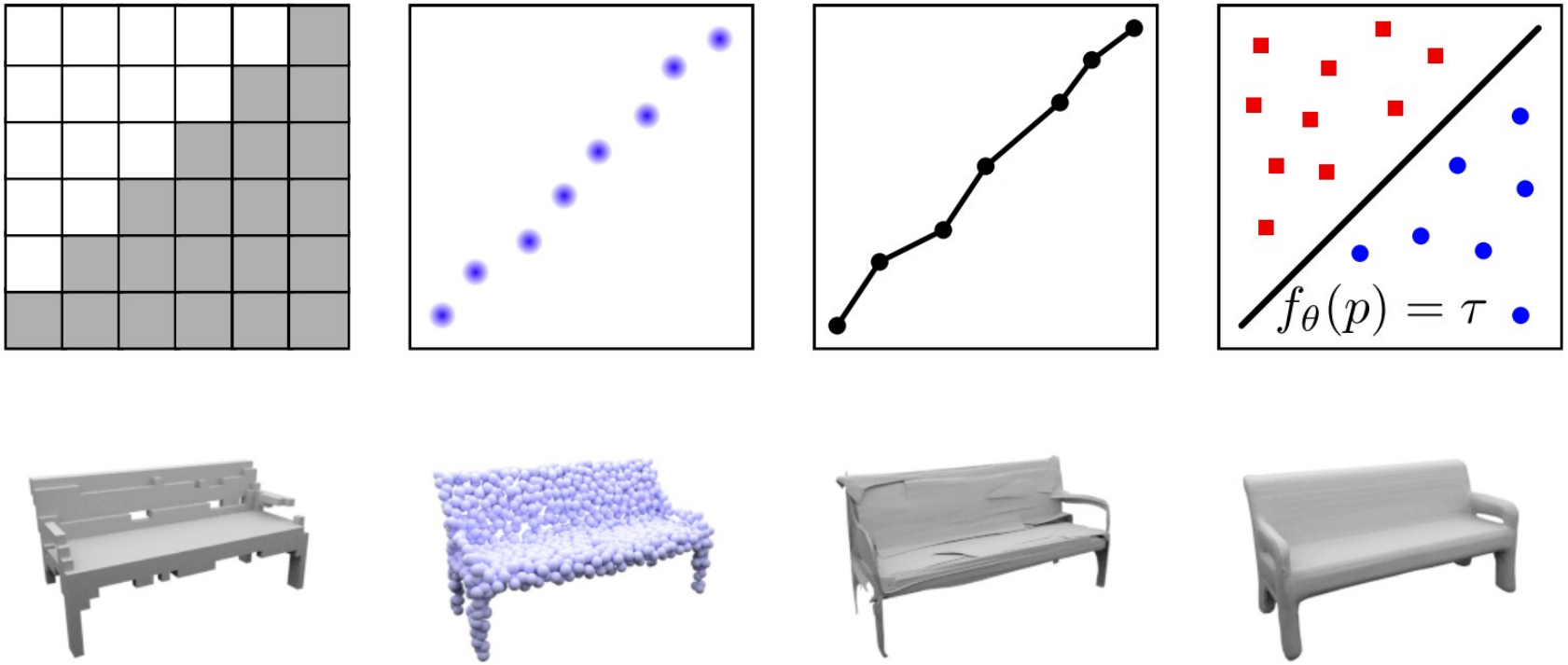


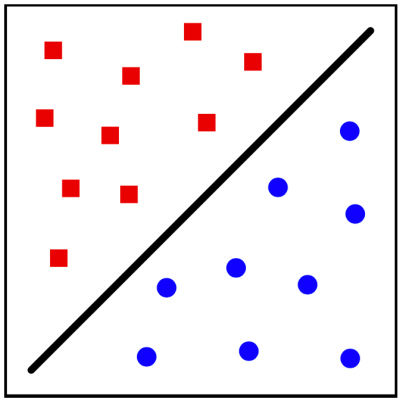
Figure from occupancy network.

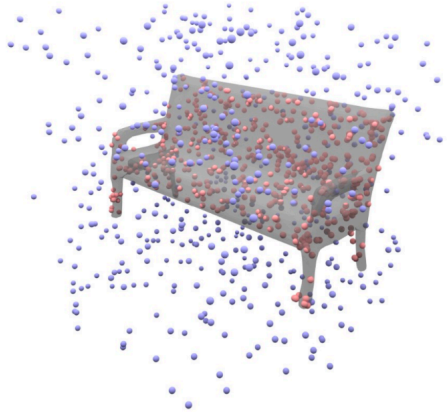
Implicit Field Representation

- **Key idea**: consider surface implicitly as decision boundary of a non-linear classifier:

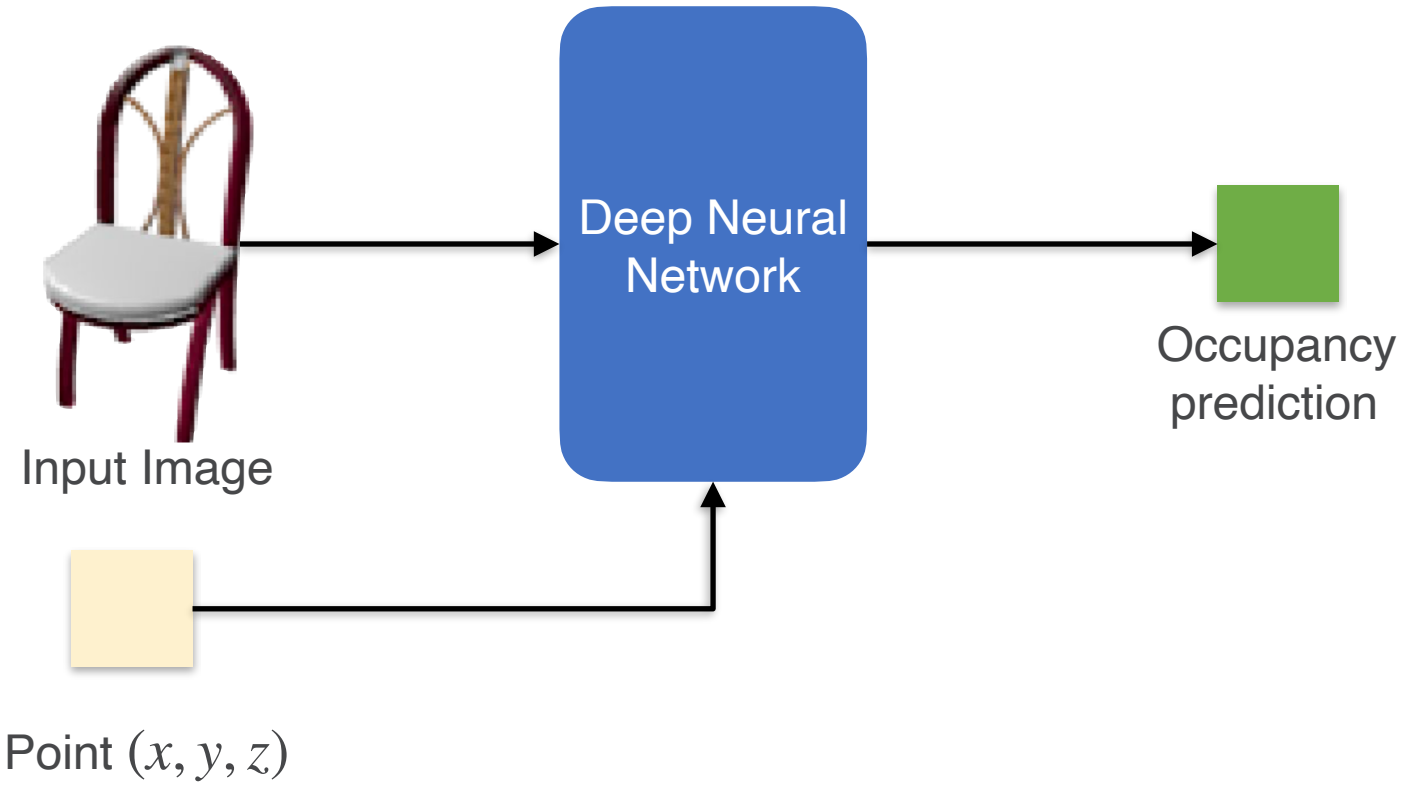
$$f_\theta : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0, 1]$$

3D Location | Condition (eg, Image) | Occupancy Probability





Implicit Field Network Architecture



Limitation and Future Work

- Few-shot?
- How to model the distribution of multiple options of invisible area?
- How to add structure constraint? Example: model a shape with several holes.

L8: 3D Networks

Hao Su

Slides prepared by Dr. Songfang Han

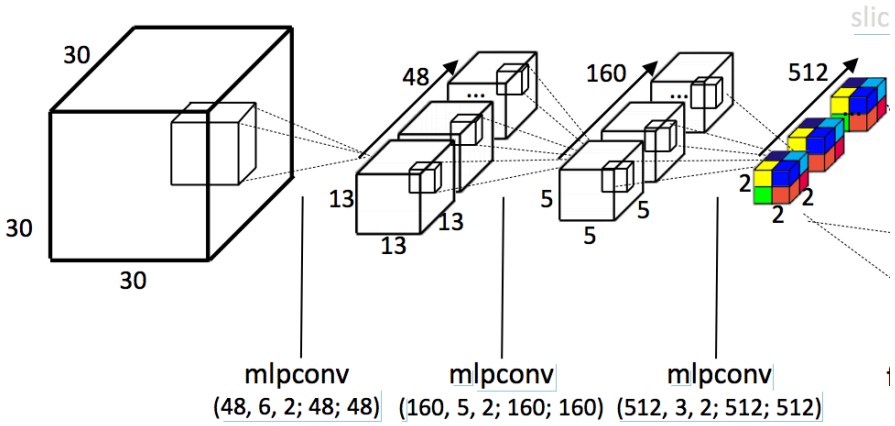
Volumetric CNN

Voxelization

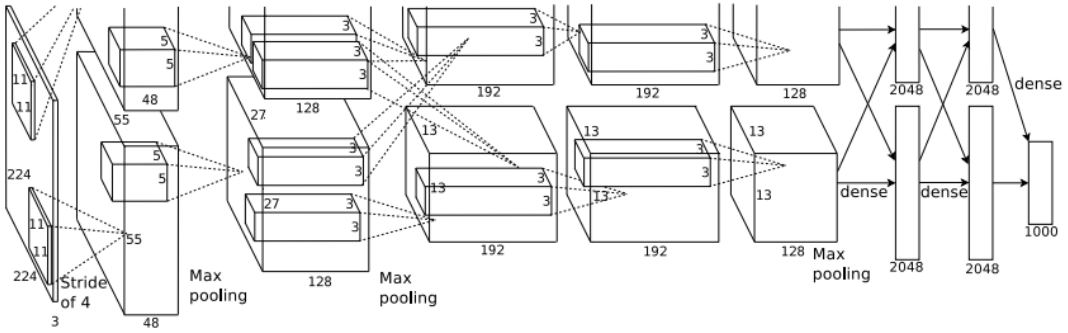
Represent the occupancy of regular 3D grids

3D CNN on Volumetric Data

3D convolution uses 4D kernels



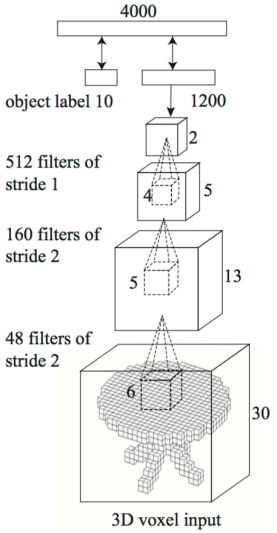
Complexity Issue



AlexNet, 2012

Input resolution: 224x224

224x224=50176

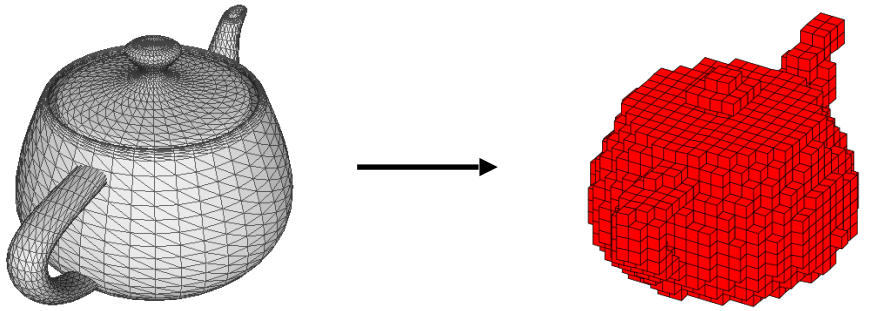


3DShapeNets, 2015

Input resolution: 30x30x30

224x224=27000

Complexity Issue



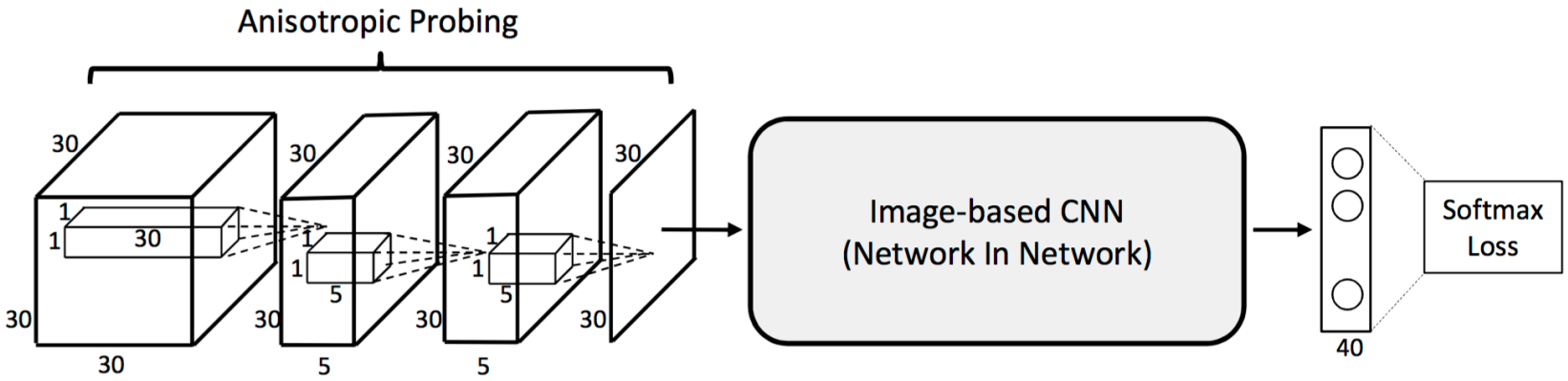
Polygon Mesh

Occupancy Grid
30×30×30

Information loss in voxelization

Idea 1: Learn to Project

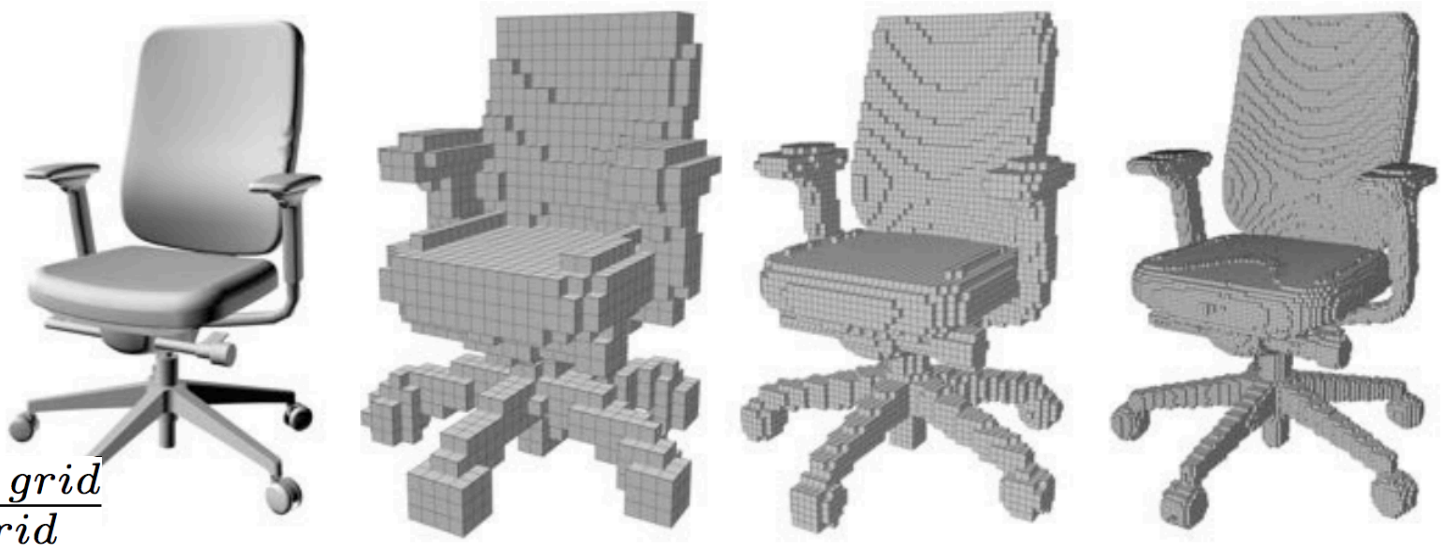
Idea: "X-ray" rendering + Image (2D) CNNs
very low #param, very low computation



Su et al., "**Volumetric and Multi-View CNNs for Object Classification on 3D Data**", *CVPR 2016*

Many other works in autonomous driving that uses **bird's eye view** for object detection

More Principled: Sparsity of 3D Shapes

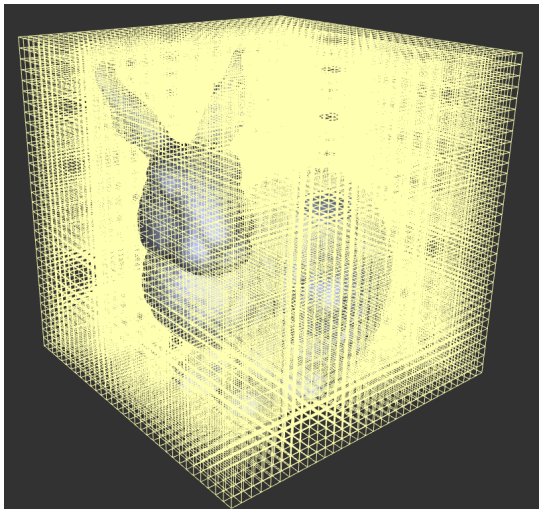


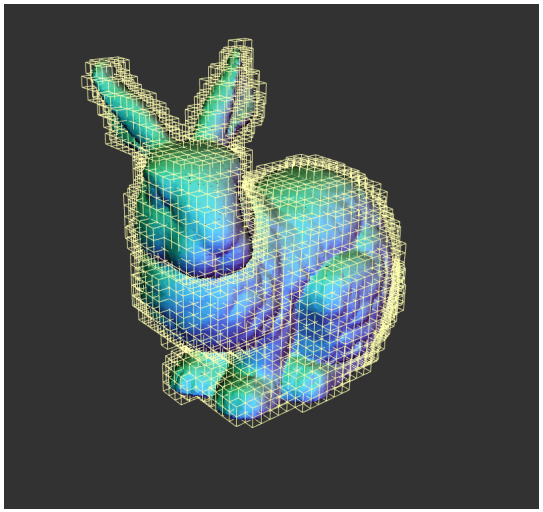
$$\frac{\#occupied\ grid}{\#total\ grid}$$

Occupancy:	10.41%	5.09%	2.41%
Resolution:	32	64	128

Store only the Occupied Grids

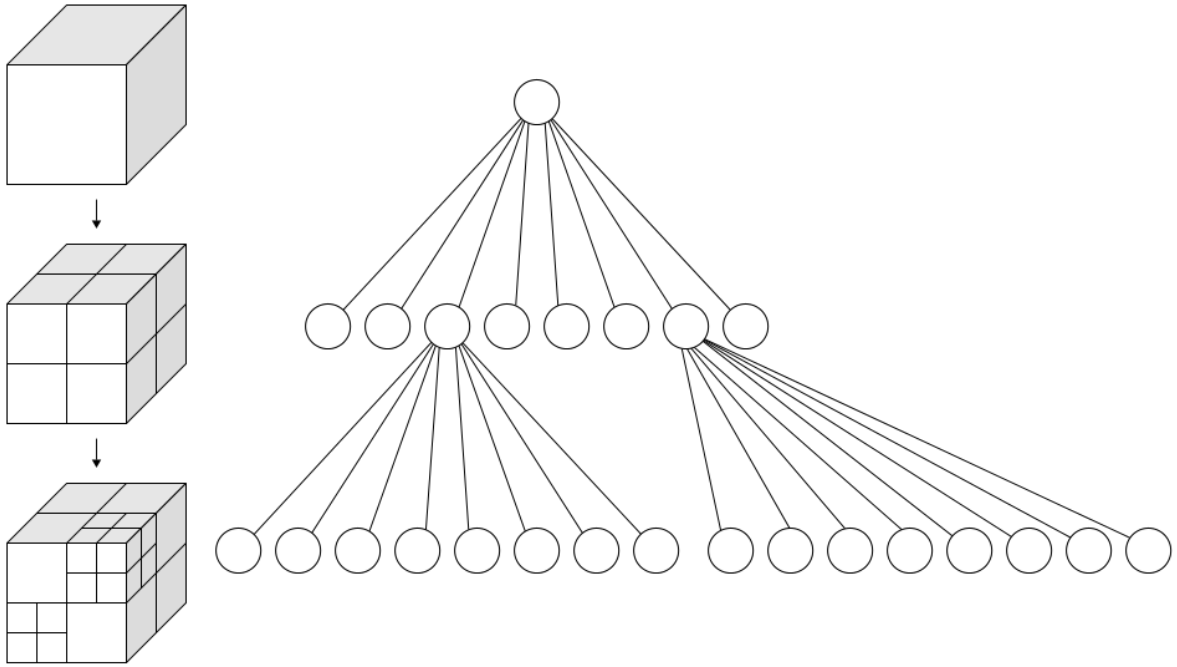
- Store the sparse surface signals
- Constrain the computation near the surface



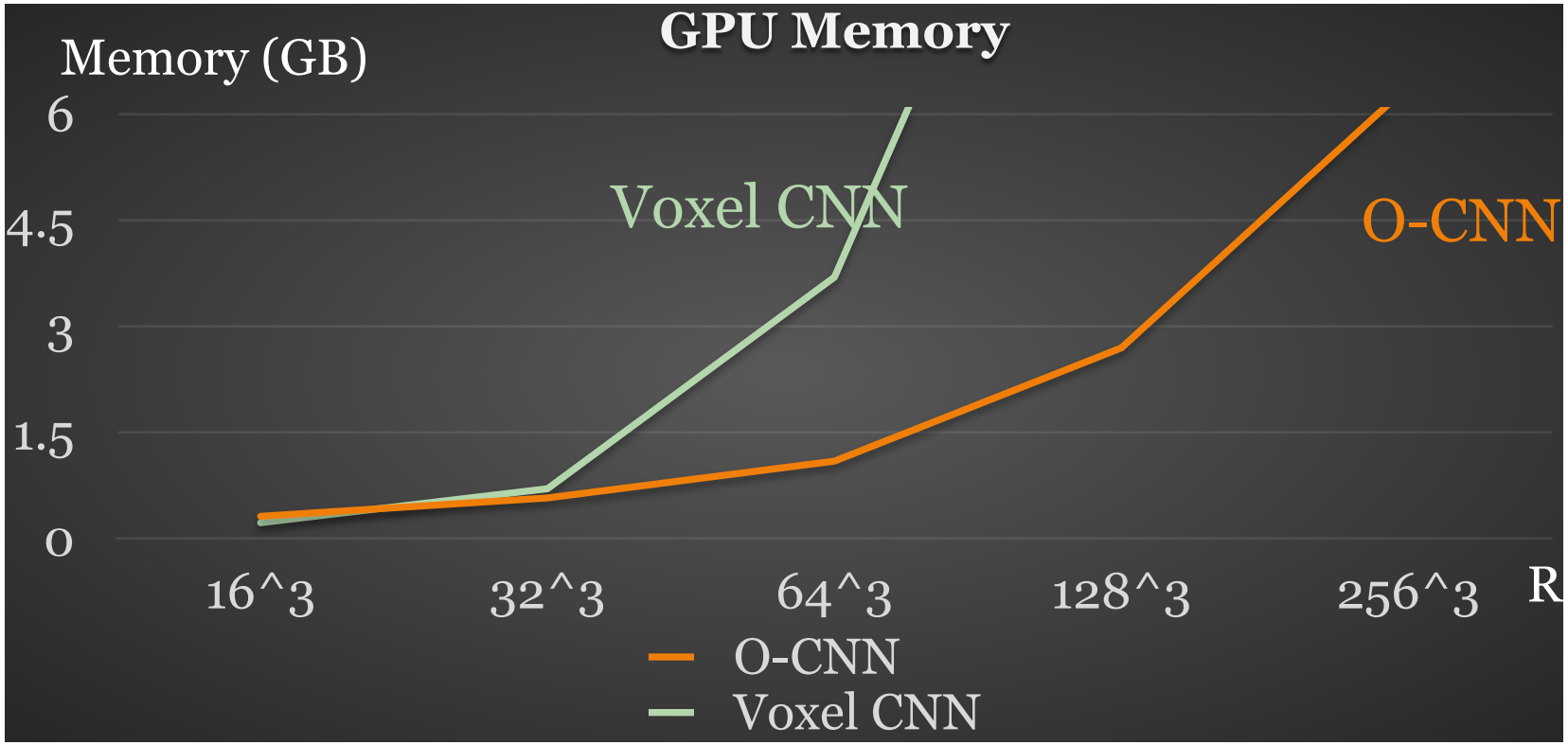


Octree: Recursively Partition the Space

- Each internal node has exactly eight children
- Neighborhood searching: Hash table



Memory Efficiency

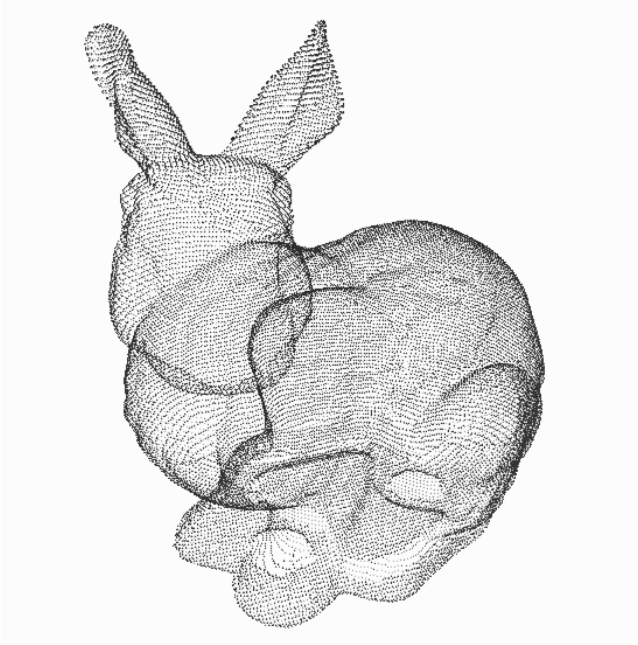


Implementation

- SparseConvNet
  - https://github.com/facebookresearch/SparseConvNet
  - Uses ResNet architecture
  - State-of-the-art for 3D analysis
  - Takes time to train

Graham et al., "**Submanifold Sparse Convolutional Networks**", *arxiv*

Point Networks

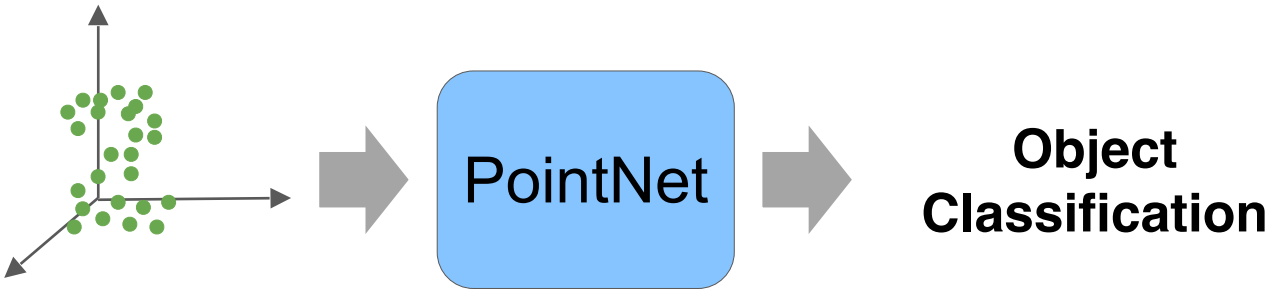


Point cloud

(The most common 3D sensor data)

Directly Process Point Cloud Data

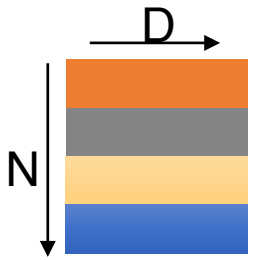
End-to-end learning for **unstructured, unordered** point data



Qi, Charles R., et al. "**Pointnet: Deep learning on point sets for 3d classification and segmentation**", CVPR 2017
Zaheer, Manzil, et al. "**Deep sets**", NeurIPS 2017

Properties of a Desired Point Network

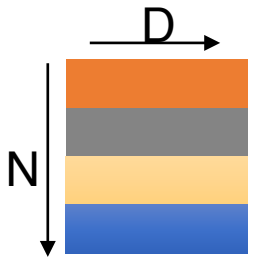
Point cloud: N **orderless** points, each represented by a D dim coordinate



2D array representation

Properties of a Desired Point Network

Point cloud: N **orderless** points, each represented by a D dim coordinate



2D array representation

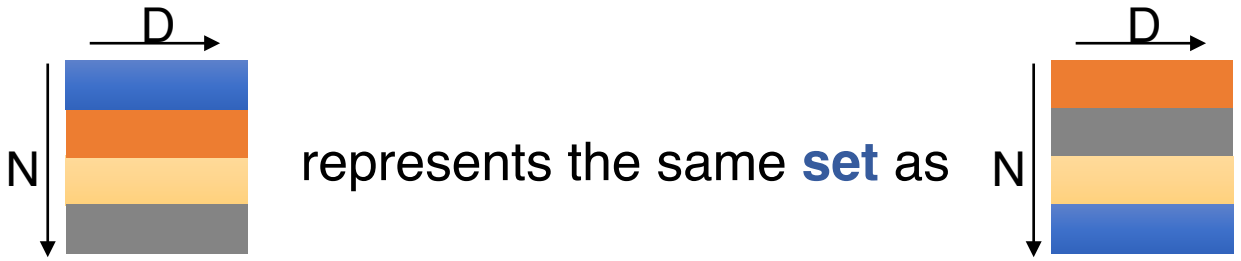
Permutation invariance

Transformation invariance

Permutation Invariance of PointNet

Permutation Invariance

Point cloud: N **orderless** points, each represented by a D dim coordinate



2D array representation

Permutation Invariance: Symmetric Function

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

Examples:

$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$

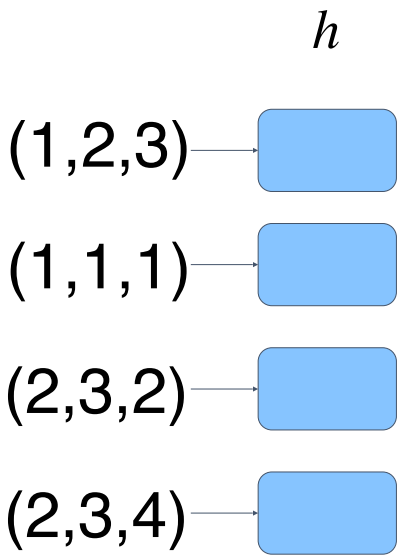
$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

…

Construct a Symmetric Function

Observe:

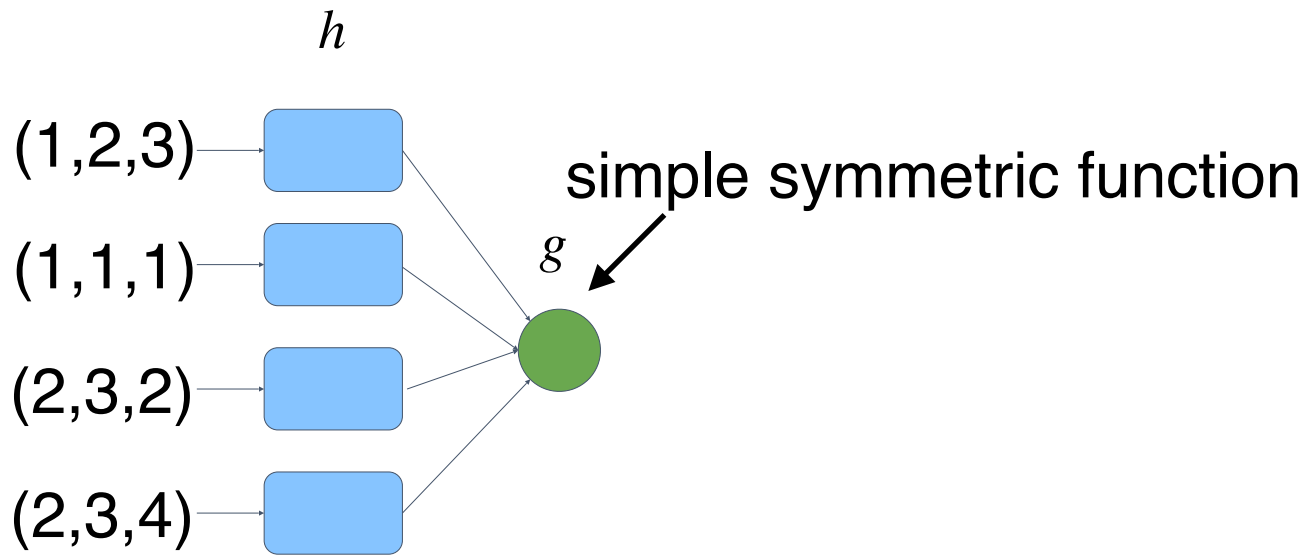
$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric



Construct a Symmetric Function

Observe:

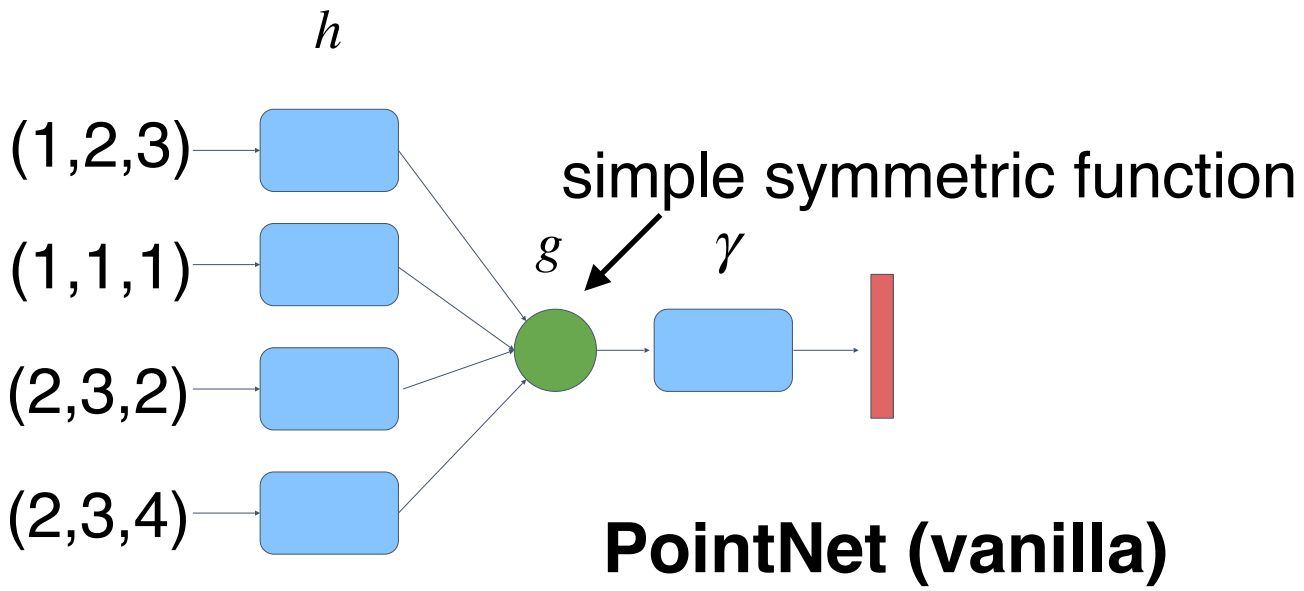
$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric



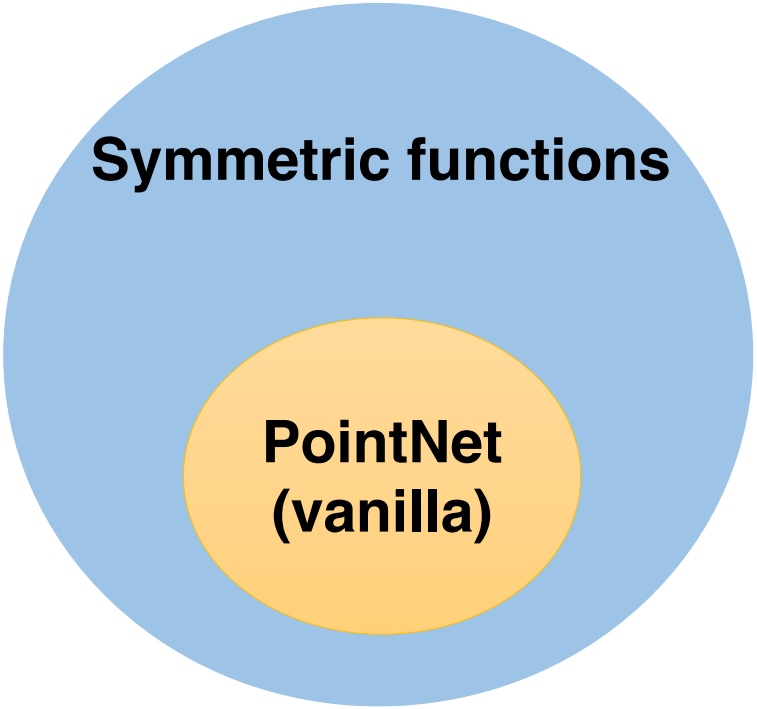
Construct a Symmetric Function

Observe:

$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric



Q: What Symmetric Functions Can Be Constructed by PointNet?



Universal Approximation Theorem

- Can approximate any “continuous” functions over sets
- “Continuous”: A function value would change by little if the point positions vary by little

$$\left| f(S) - \gamma\left(\underset{x_i \in S}{\mathrm{MAX}} \{h(x_i)\} \right) \right| < \epsilon$$

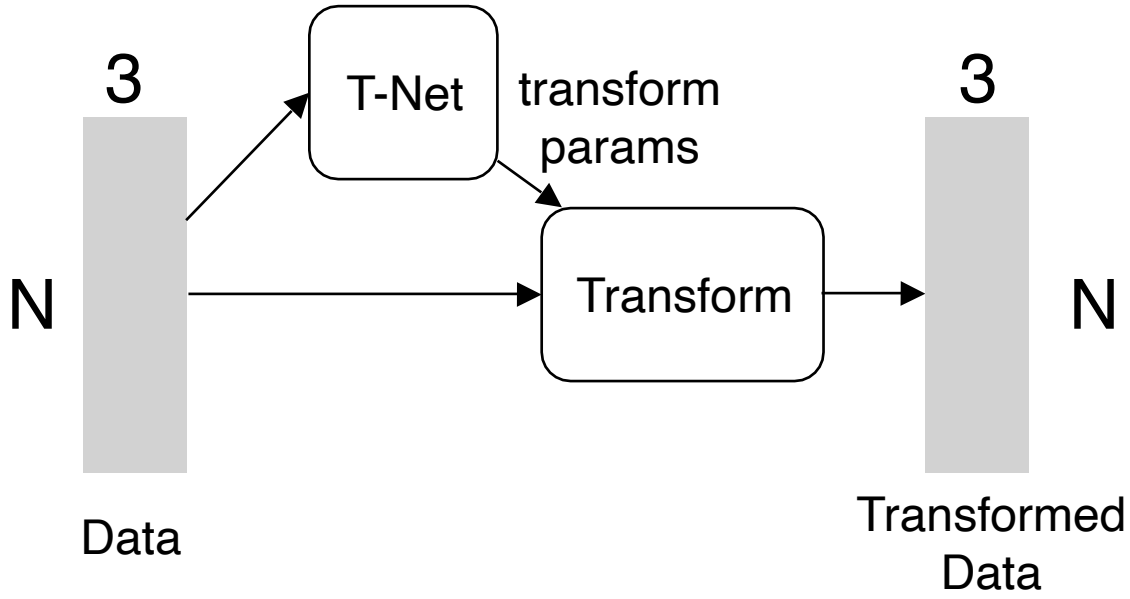
$S \subseteq \mathbb{R}^d$,

PointNet (vanilla)

Transformation Invariance of PointNet

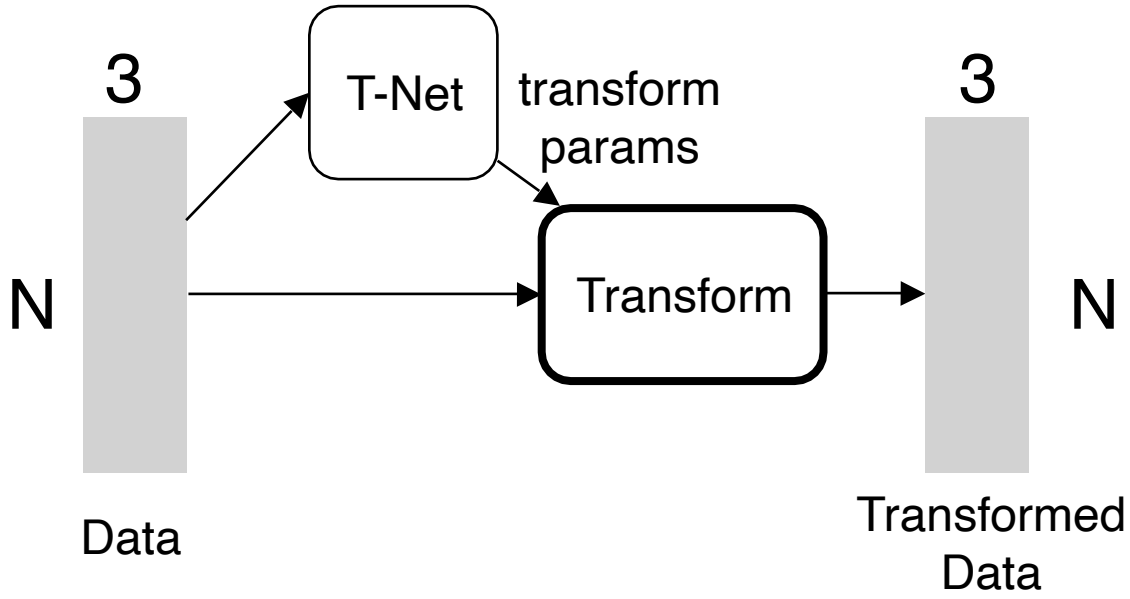
Input Alignment by Transformer Network

Idea: Data dependent transformation for automatic alignment



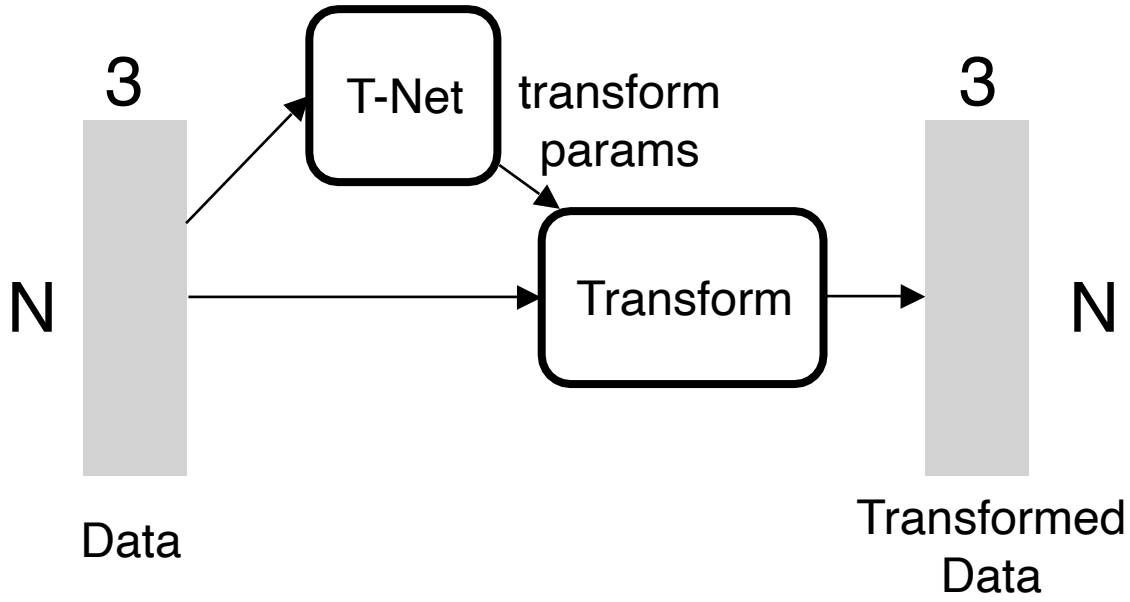
Input Alignment by Transformer Network

Idea: Data dependent transformation for automatic alignment



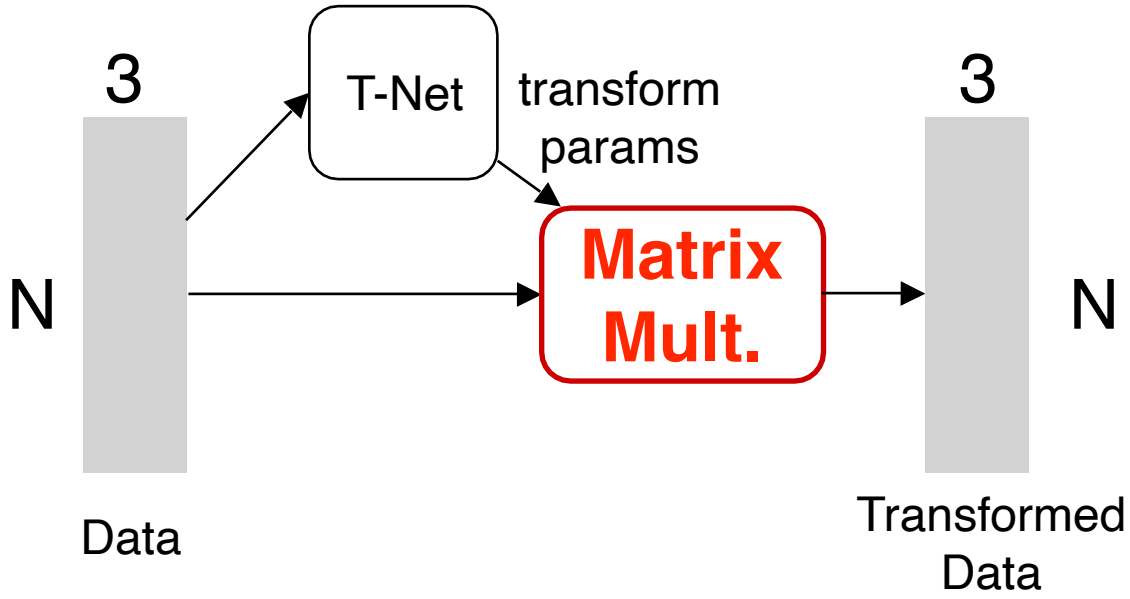
Input Alignment by Transformer Network

Idea: Data dependent transformation for automatic alignment

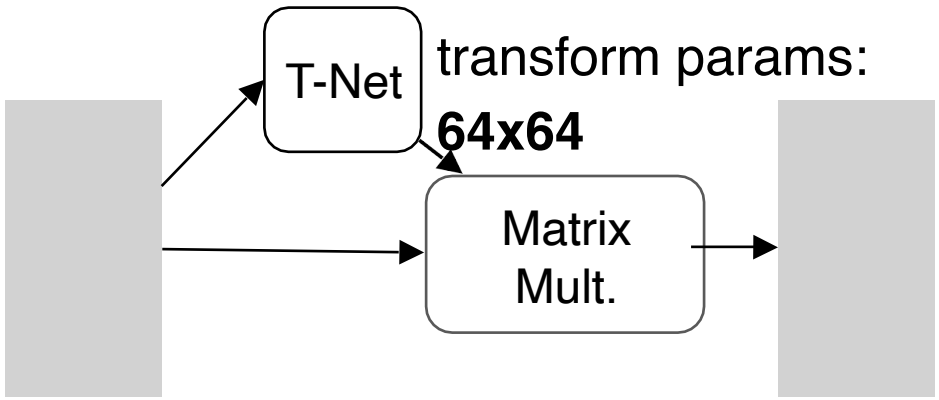


Input Alignment by Transformer Network

The transformation is just matrix multiplication!



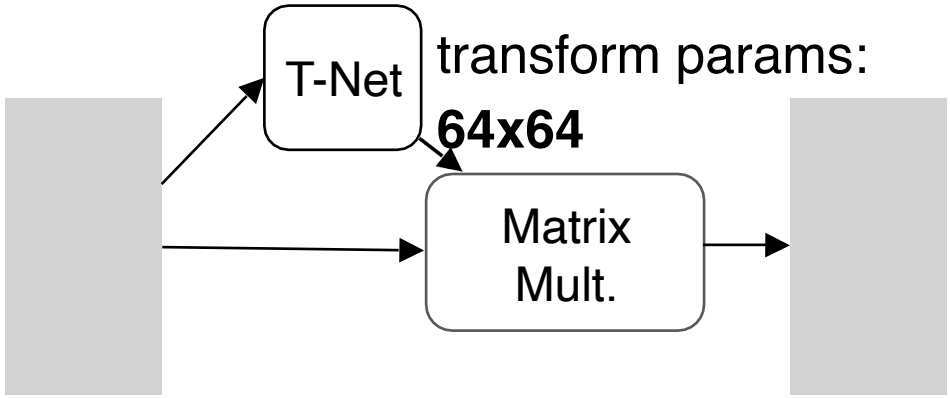
Embedding Space Alignment



Input embeddings: **Nx64**

Transformed embeddings: **Nx64**

Embedding Space Alignment



Input embeddings: **Nx64**

Transformed embeddings: **Nx64**

Regularization:

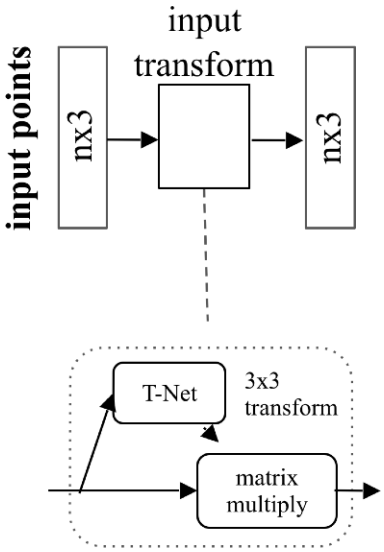
Transform matrix A64x64 close to orthogonal:

$$L_{reg} = \|I - AA^T\|_F^2$$

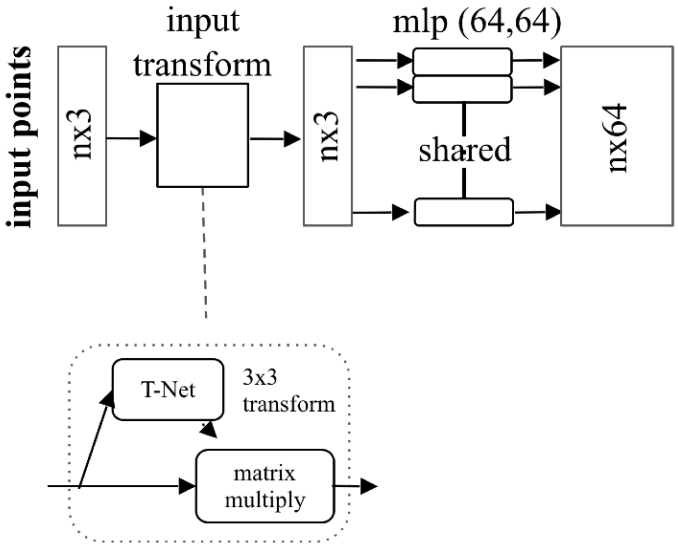
Detailed Implementation of PointNet

PointNet Classification Network

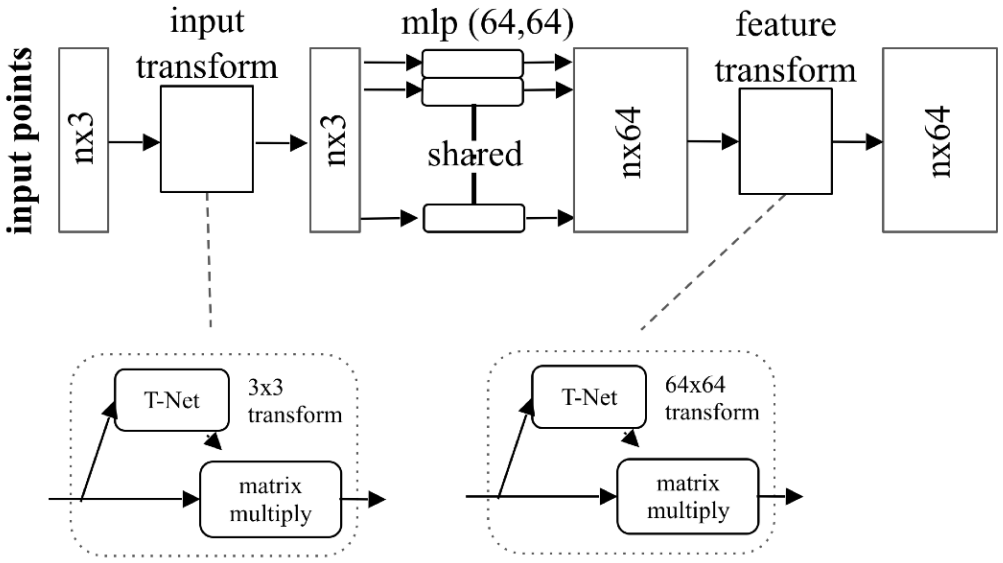
PointNet Classification Network



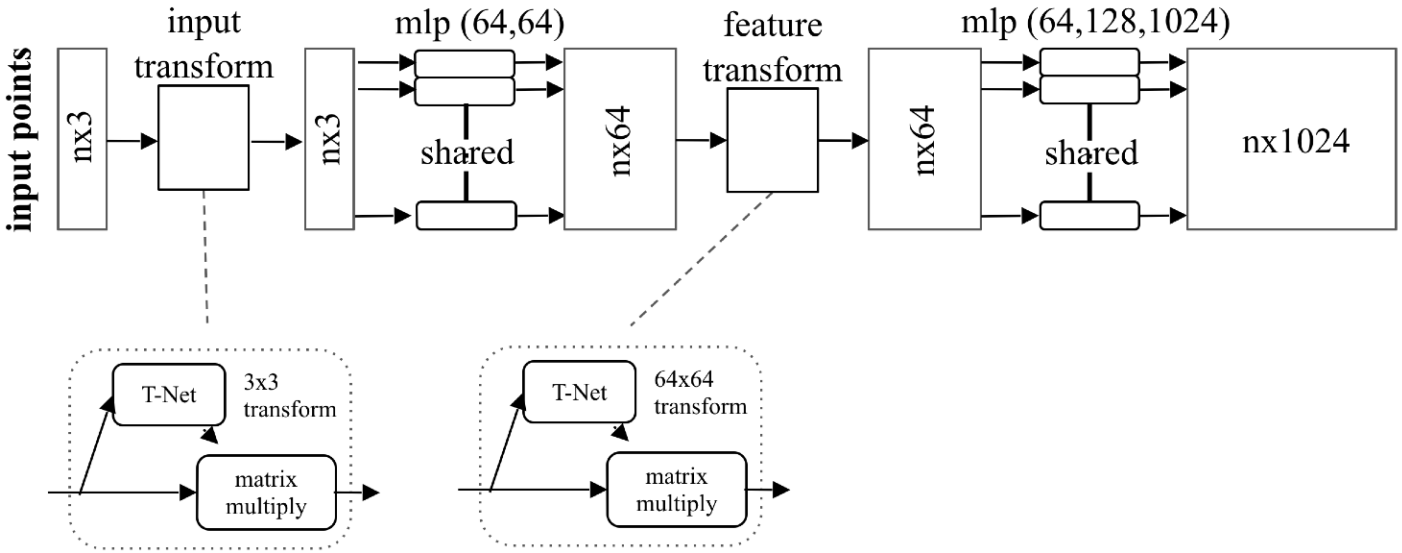
PointNet Classification Network



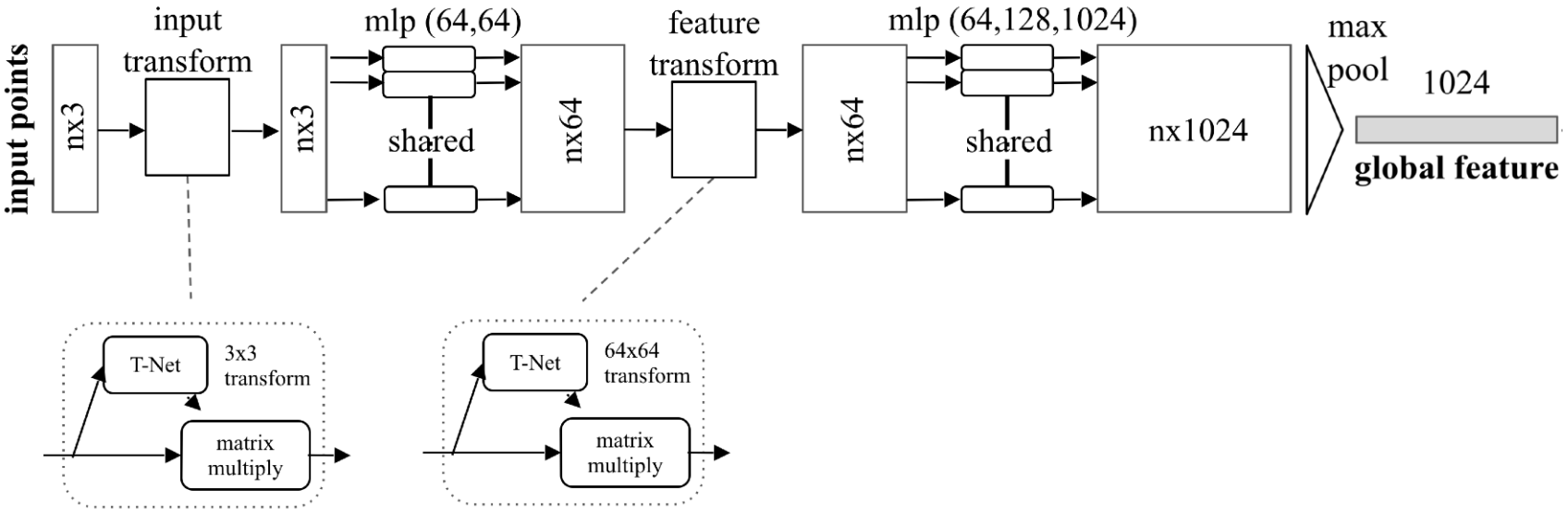
PointNet Classification Network



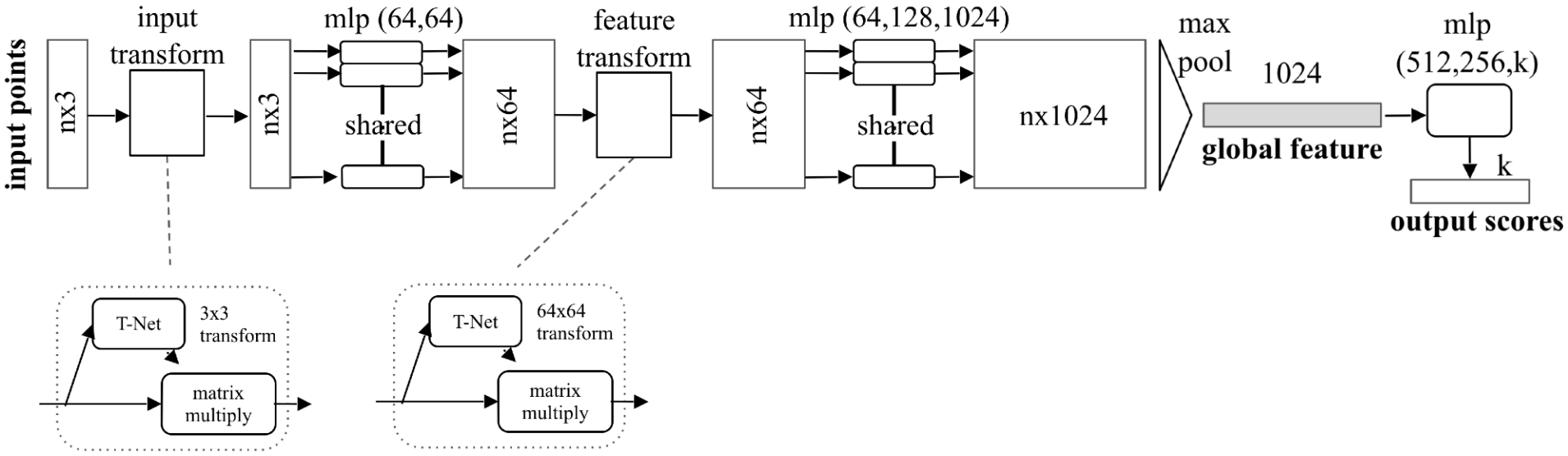
PointNet Classification Network



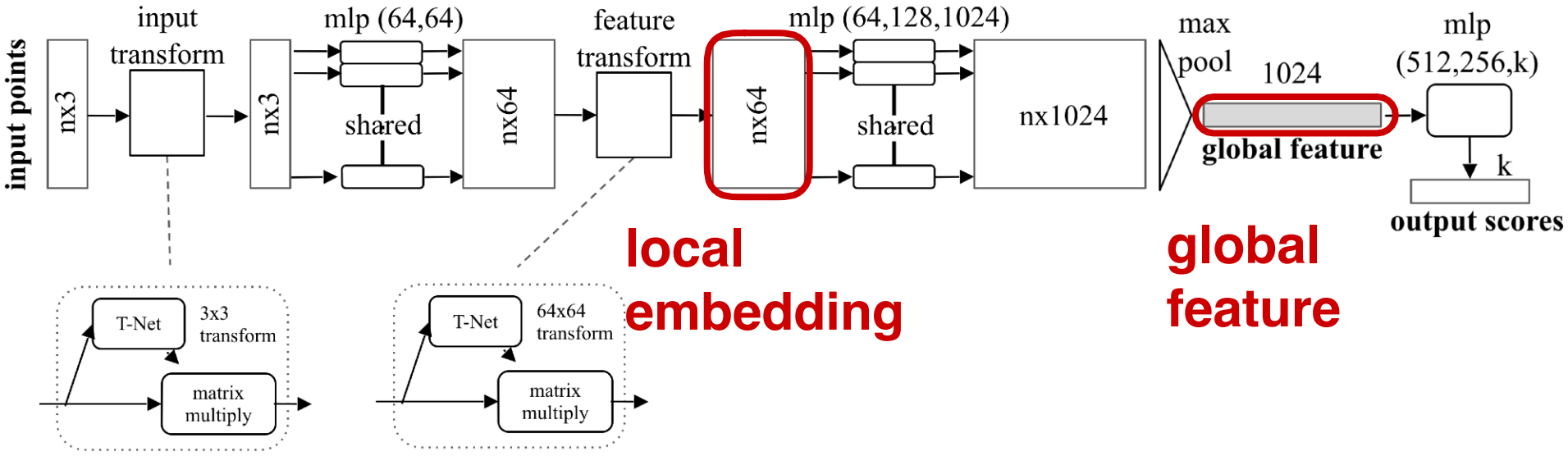
PointNet Classification Network



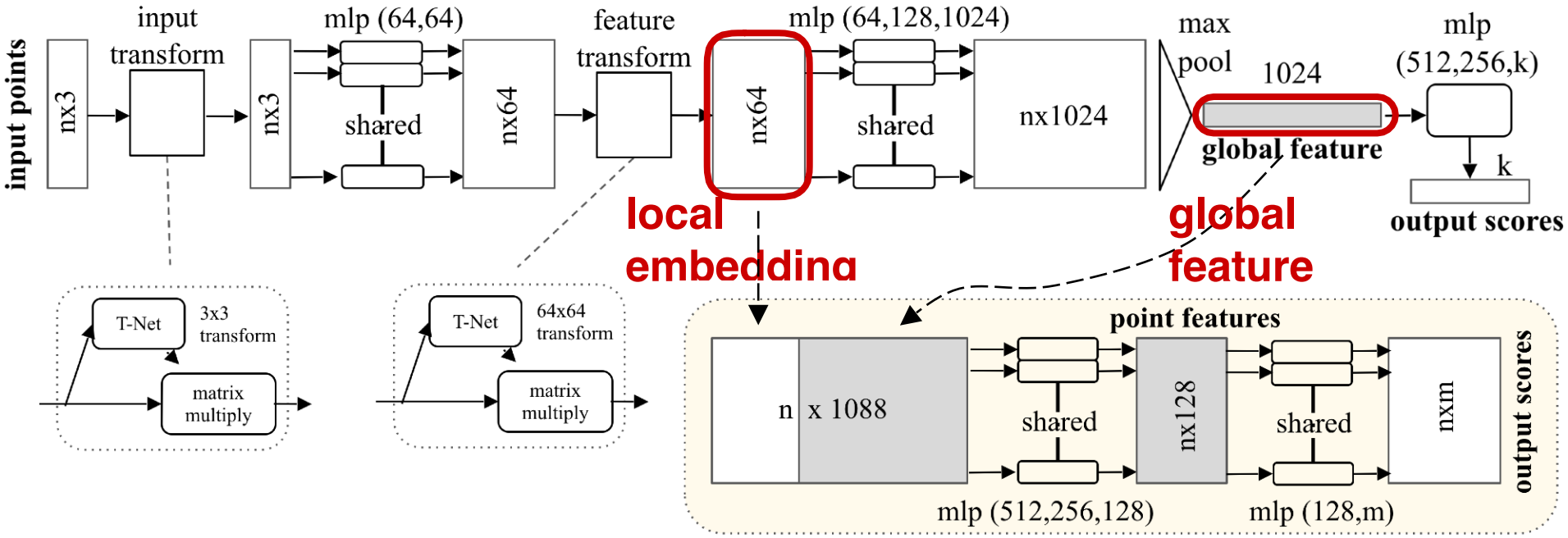
PointNet Classification Network



Extension to Segmentation Network



Extension to Segmentation Network



Robustness to Data Corruption

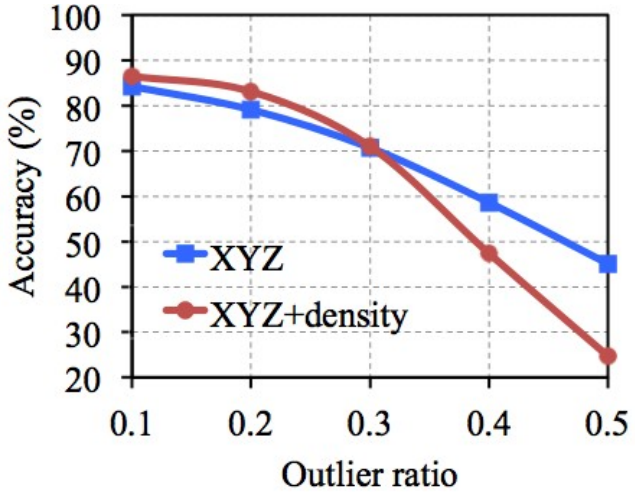
dataset: ModelNet40; metric: 40-class classification accuracy (%)

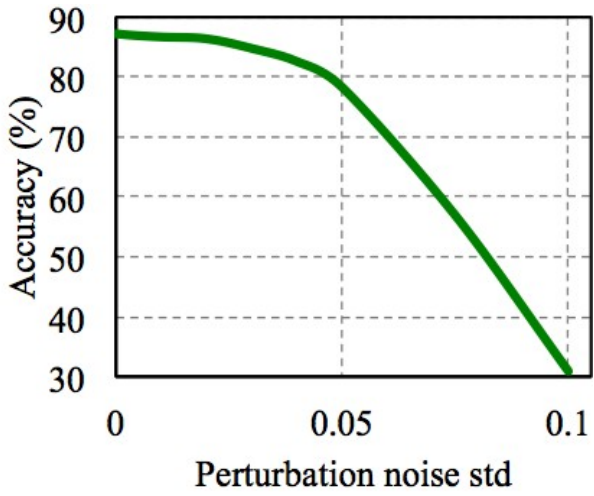
Robustness to Data Corruption

Less than 2% accuracy drop with 50% missing data

dataset: ModelNet40; metric: 40-class classification accuracy (%)

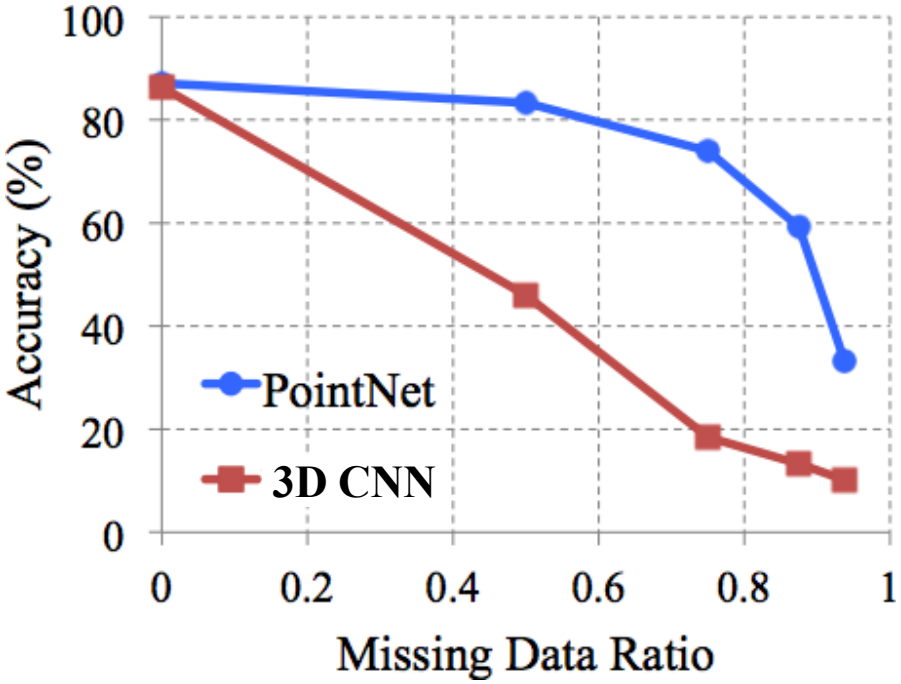
Robustness to Data Corruption





dataset: ModelNet40; metric: 40-class classification accuracy (%)

Robustness to Data Corruption

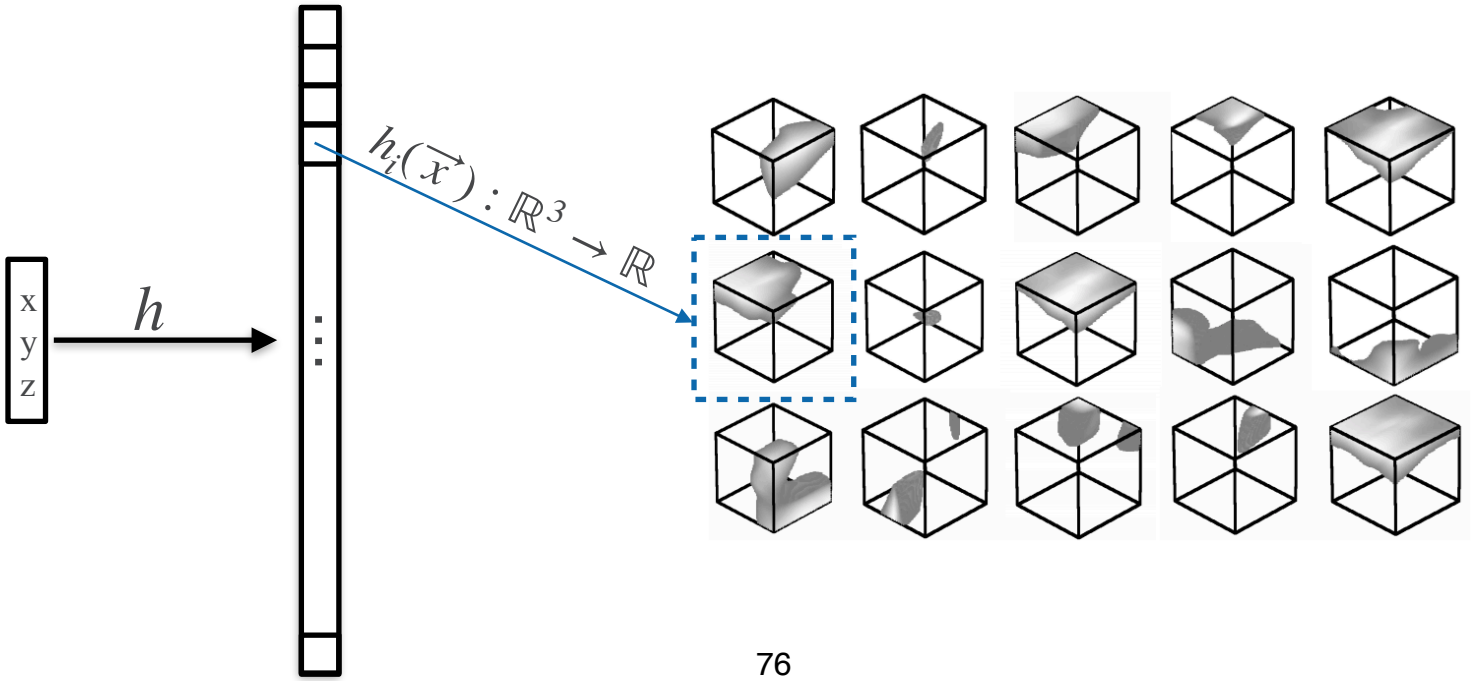


Why is PointNet so robust to missing data?

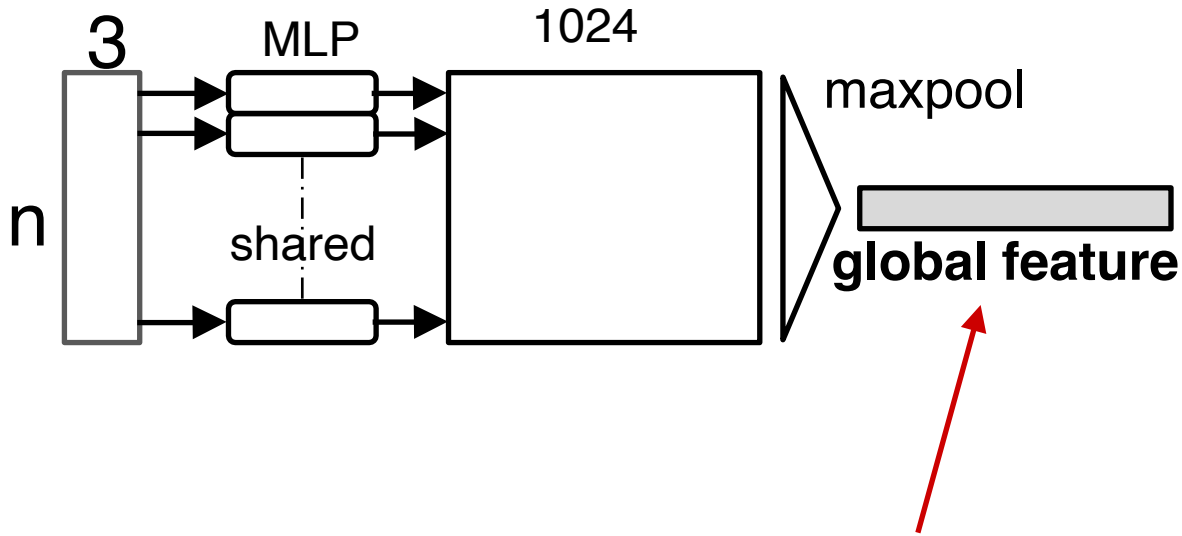
Interpretation to "First Layer" Output

- Think of each dimension as a "binary" variable (the truth is a soft version)
- It encodes whether the point is in a certain spatial region
- The shape of the spatial region is learned

3D voxels of irregular boundaries!



Visualizing Global Point Cloud Features



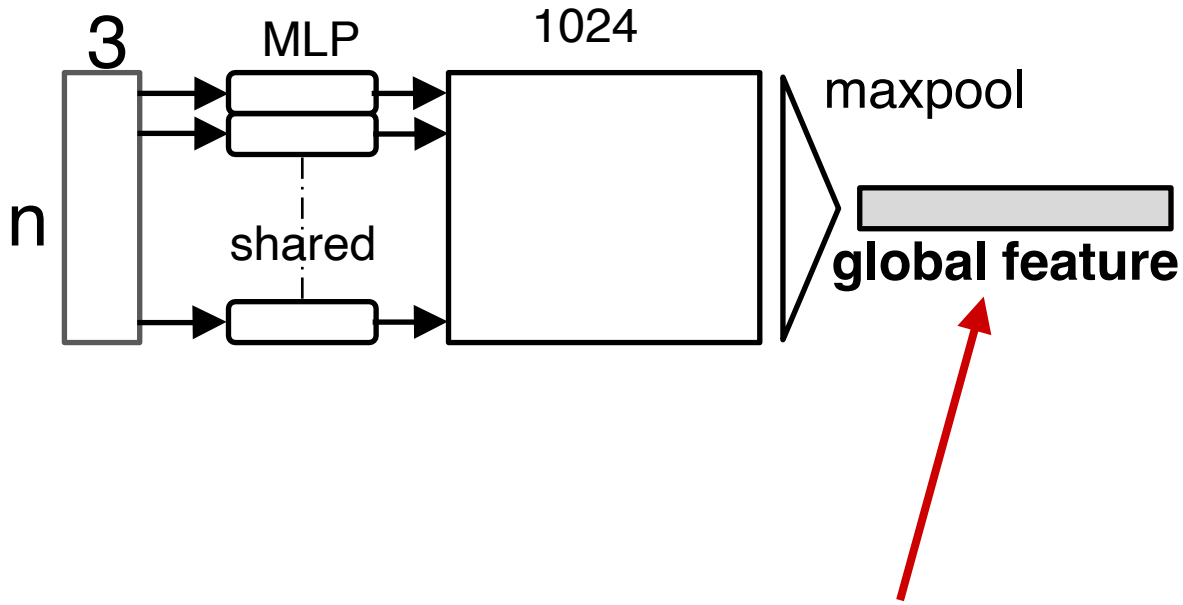
Which input points are contributing to the global feature?
(critical points)

Visualizing Global Point Cloud Features

Original Shape:

Critical Point Set:

Visualizing Global Point Cloud Features



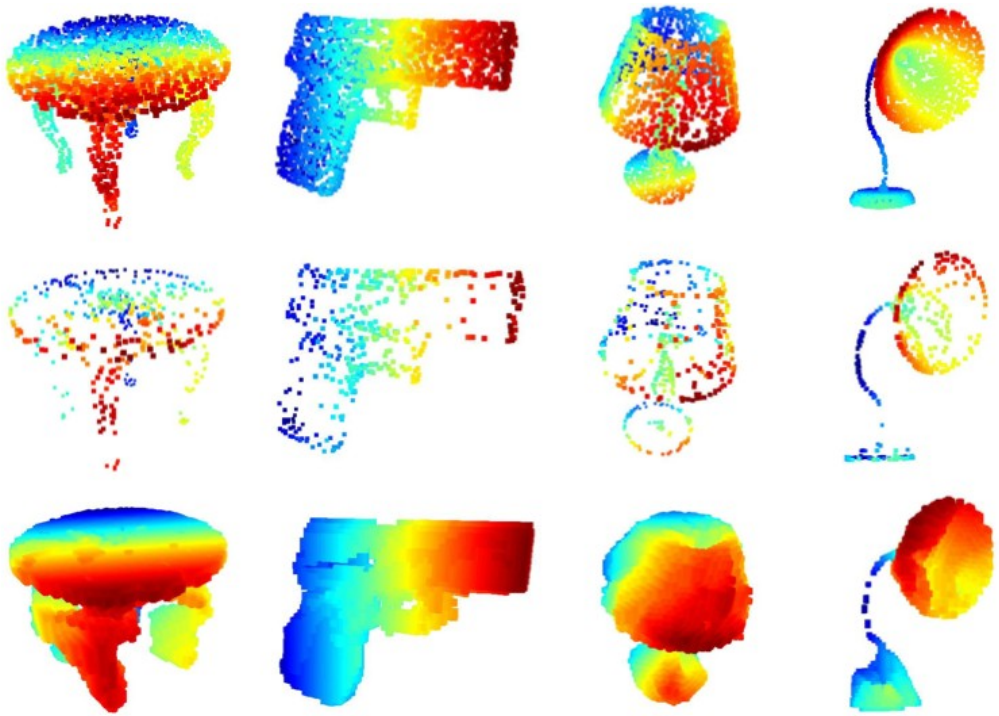
Which points won't affect the global feature?

Visualizing Global Point Cloud Features

Original Shape:

Critical Point Set:

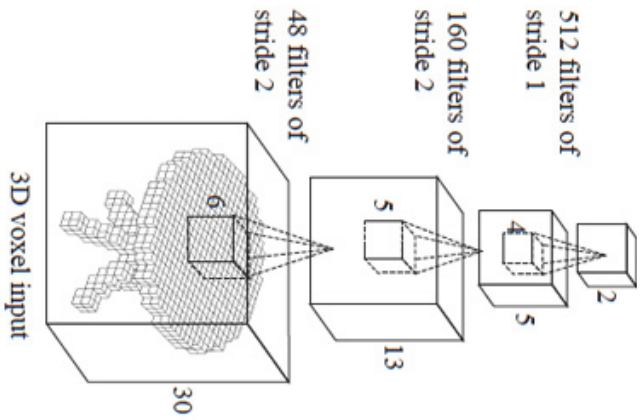
Upper bound set:



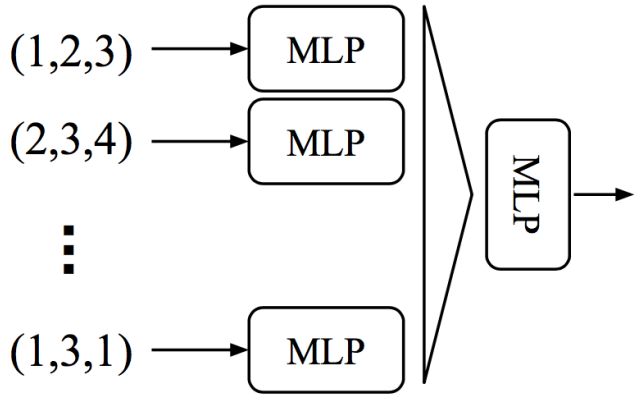
Limitations of PointNet

Hierarchical feature learning
Multiple levels of abstraction

Global feature learning
Either one point or all points



3D CNN (Wu et al.)



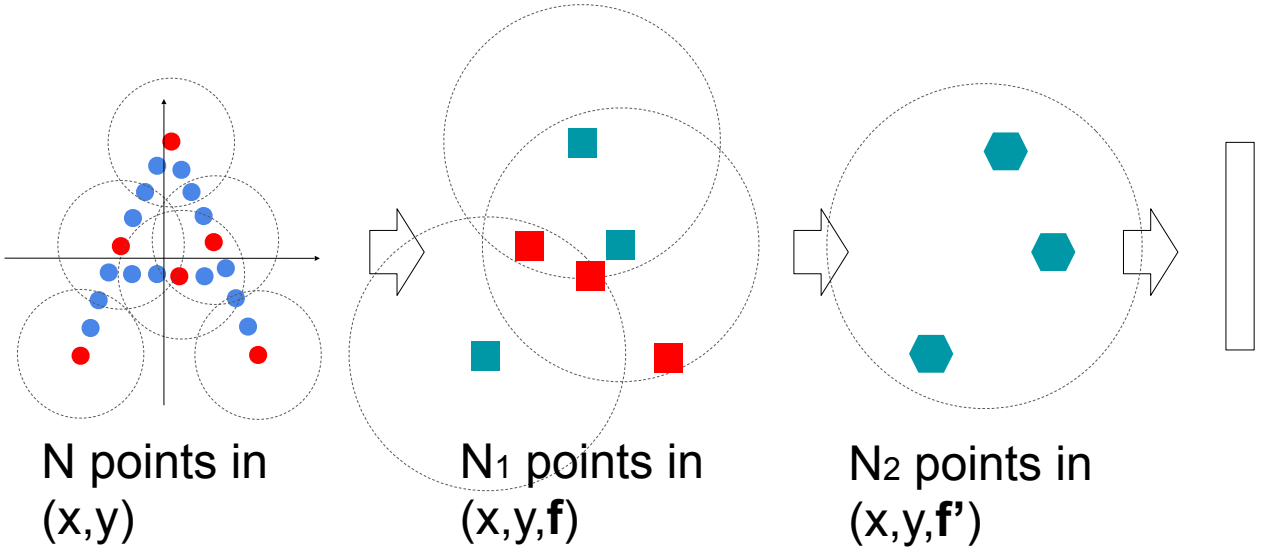
PointNet (vanilla) (Qi et al.)

- No local context for each point!
- Global feature depends on absolute coordinate. Hard to generalize to unseen scene configurations!

Points in Metric Space

- Learn “kernels” in 3D space and conduct convolution
- Kernels have compact spatial support
- For convolution, we need to find neighboring points
- Possible strategies for range query
  - Ball query (results in more stable features)
  - k-NN query (faster)

PointNet v2.0: Multi-Scale PointNet



Repeat

- Sample anchor points by FPS
- Find neighborhood of anchor points
- Apply PointNet in each neighborhood to mimic convolution