GRASS: Generative Recursive Autoencoders for Shape Structures

Jun Li
NUDT

Kai Xu
NUDT, Shenzen University, Shandong University

Siddhartha Chaudhuri
IIT Bombay

Ersin Yumer
Adobe Research

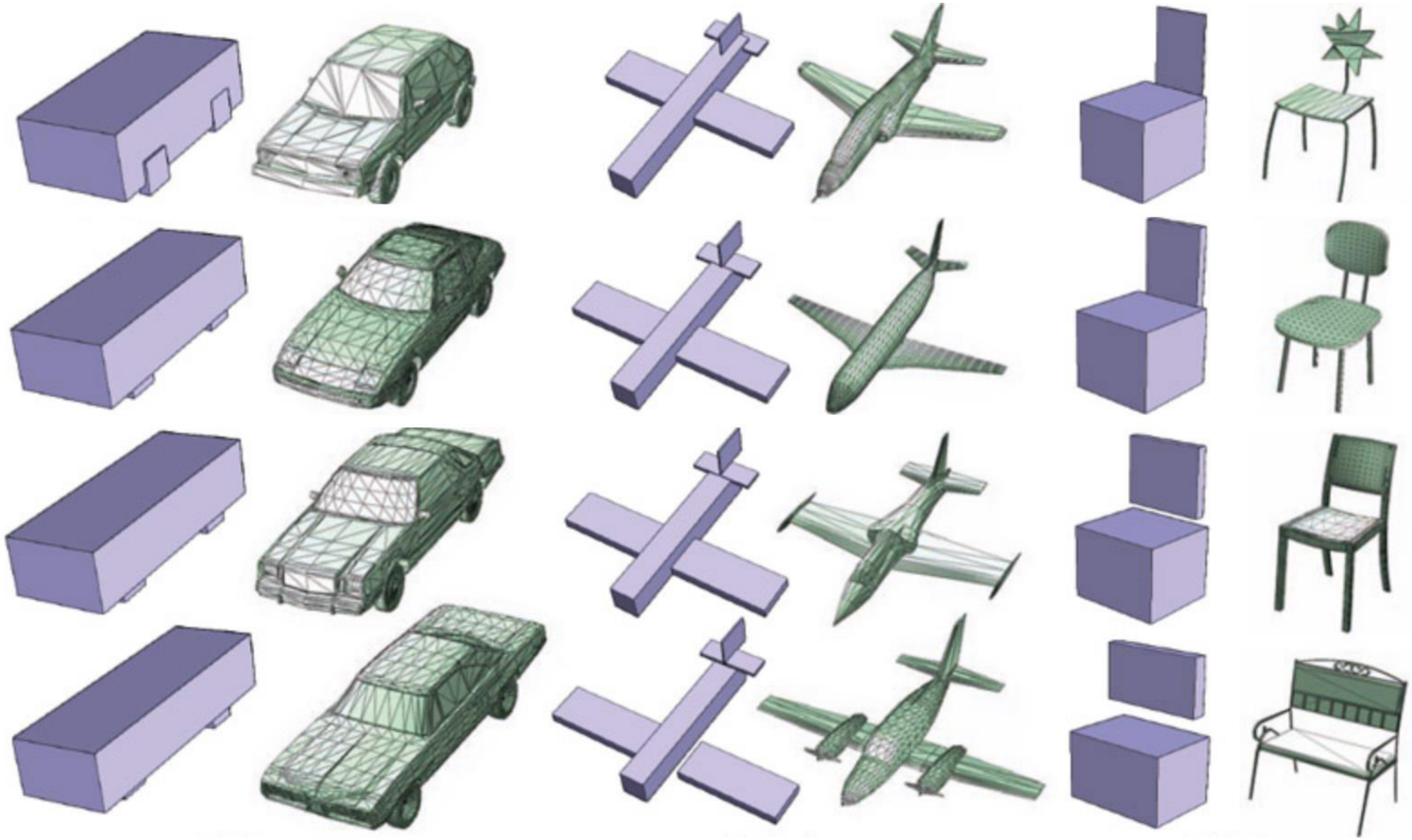
Hao (Richard) Zhang
Simon Fraser University

Leonidas Guibas
Stanford University

Shapes have different **topologies**

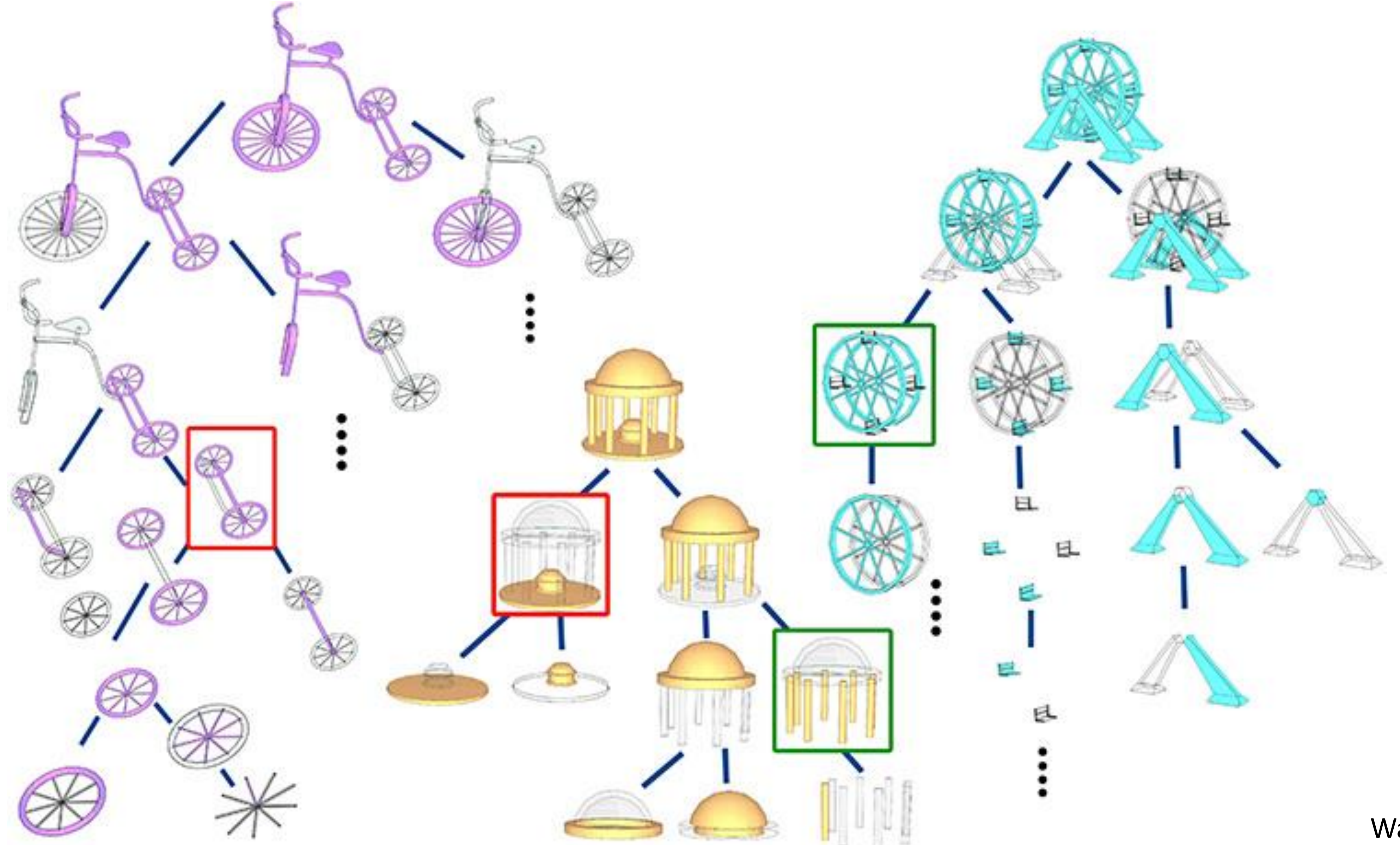
?

Shapes have different **geometries**



Ovsjanikov et al. 2011

Shapes have **hierarchical** compositionality



Wang et al. 2011

Motivating Question

How can we capture

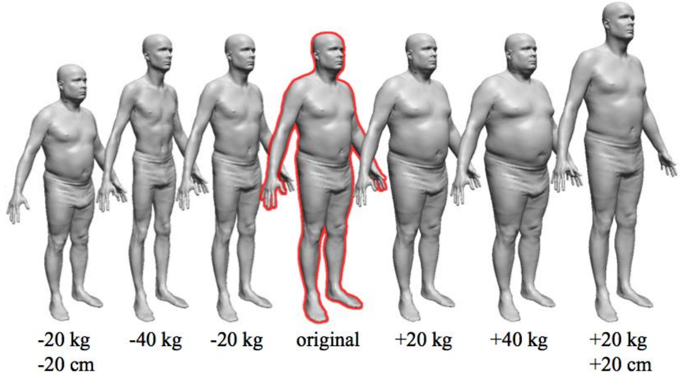
- topological variation
- geometric variation
- hierarchical composition

in a

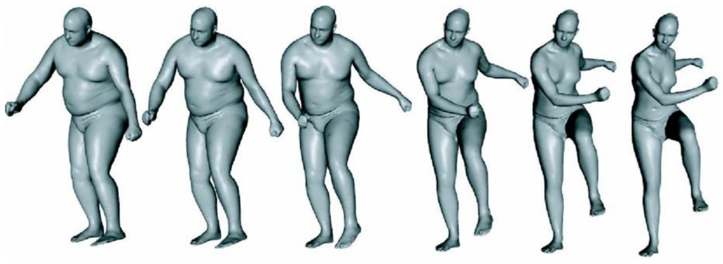
single, generative, fixed-dimensional representation?

Encode ← "Shape DNA" → Generate

Sequences of commands to Maya/AutoCAD

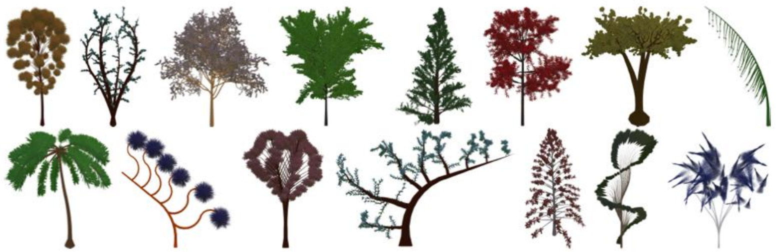


Deformable template [Allen03]

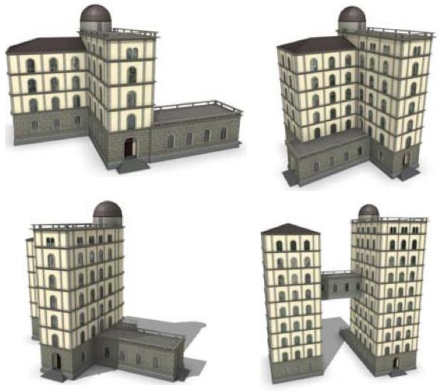


Posed template [Anguelov05]

Parametrized procedure [Weber95]

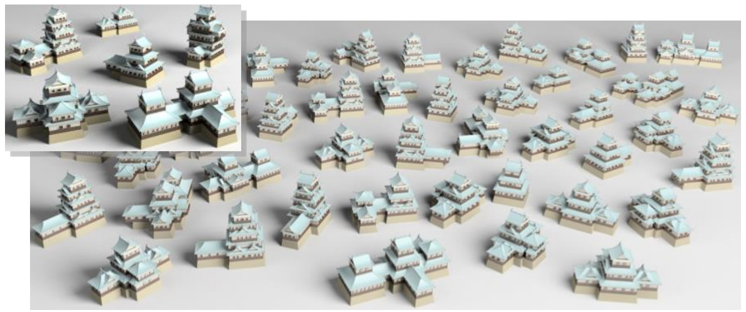


Probabilistic procedure [Talton09]



```
PRIORITY 1:
1:  footprint ⤳ S(1r,building_height,1r) facades
      T(0,building_height,0) Roof("hipped",roof_angle){ roof }
PRIORITY 2:
2:  facades ⤳ Comp("sidefaces"){ facade }
3:  facade : Shape.visible("street")
      ⤳ Subdiv("X",1r,door_width*1.5){ tiles | entrance } : 0.5
      ⤳ Subdiv("X",door_width*1.5,1r){ entrance | tiles } : 0.5
4:  facade ⤳ tiles
5:  tiles ⤳ Repeat("X",window_spacing){ tile }
6:  tile ⤳ Subdiv("X",1r,window_width,1r){ wall |
      Subdiv("Y",2r,window_height,1r){ wall | window | wall } | wall }
7:  window : Scope.occ("noparent") != "none" ⤳ wall
8:  window ⤳ S(1r,1r,window_depth) I("win.obj")
9:  entrance ⤳ Subdiv("X",1r,door_width,1r){ wall |
      Subdiv("Y",door_height,1r){ door | wall } | wall }
10: door ⤳ S(1r,1r,door_depth) I("door.obj")
11: wall ⤳ I("wall.obj")
```

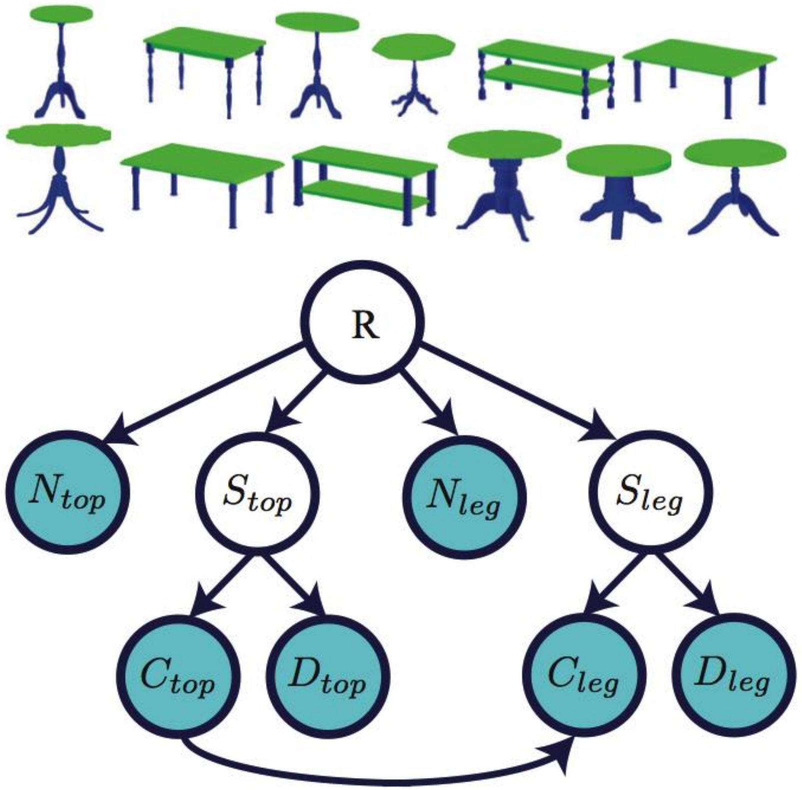
Learned grammar (single exemplar) [Bokeloh10]



Learned grammar (multi-exemplar) [Talton12]

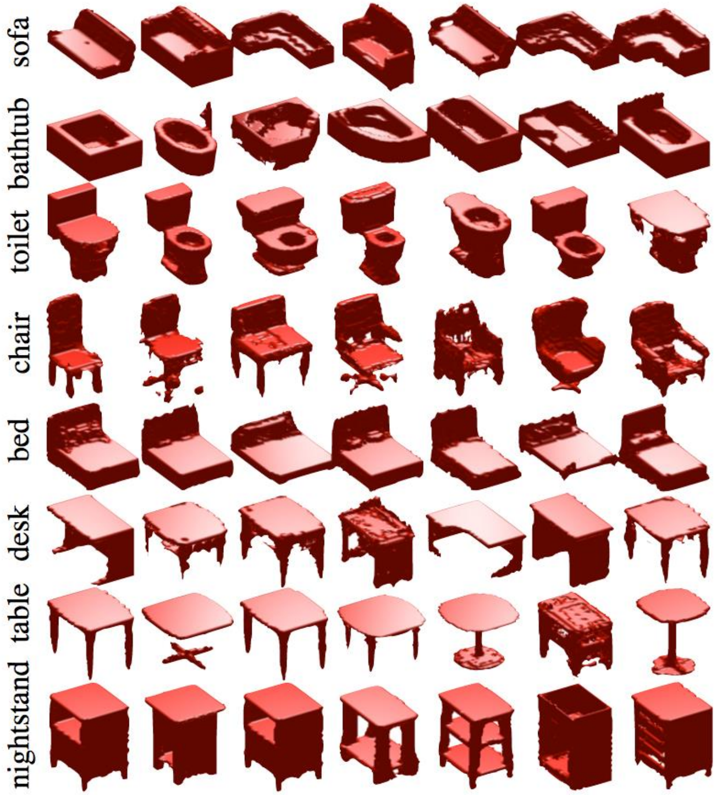
Probabilistic grammar [Müller06]

Structural PGM vs Volumetric DNN



Strongly supervised [Kalogerakis et al. ’12]

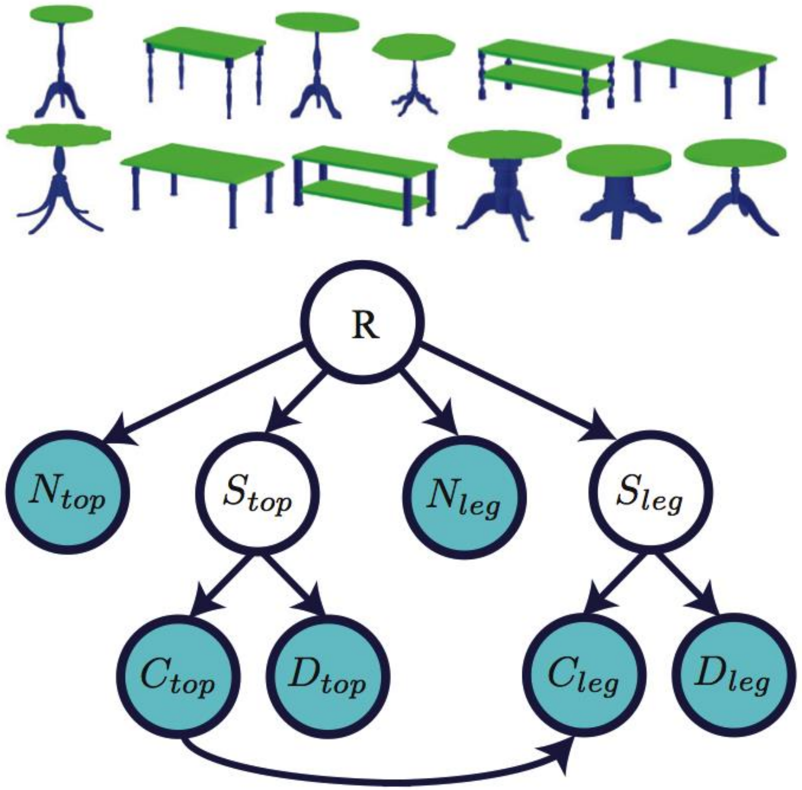
Pros: direct model of compositional structure, (relatively) low-dimensional, high quality output
Cons: limited topological variation, no continuous geometric variation (for generation), no hierarchy, huge effort to segment & label training data



Unsupervised [Wu et al. ’15]

Pros: arbitrary geometry/topology, unsupervised
Cons: low-resolution, no explicit separation of structure vs fine geometry, no guarantee of symmetry/adjacency, no hierarchy, lots of parameters, lots of training data

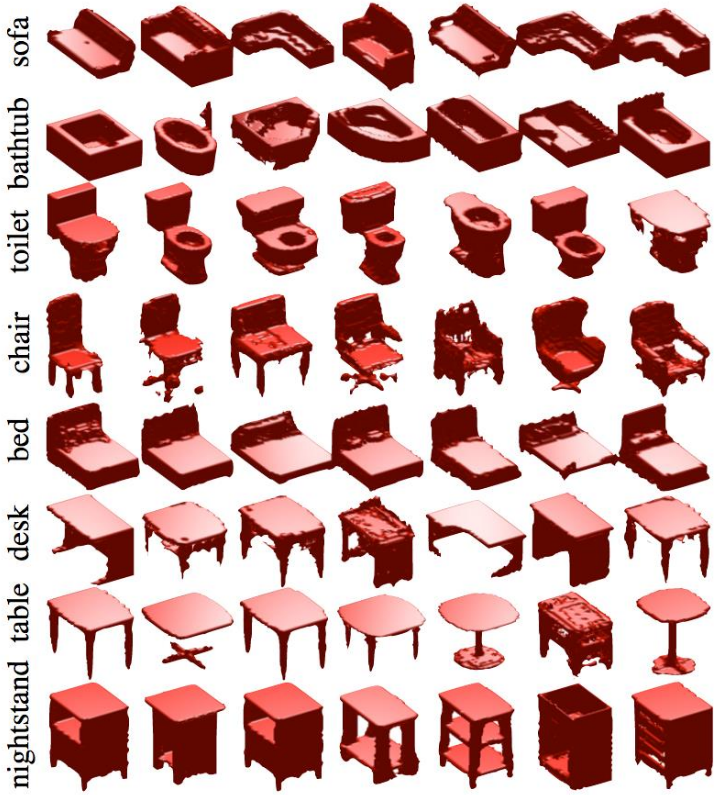
Structural PGM vs Volumetric DNN



Strongly supervised [Kalogerakis et al. ’12]

Pros: direct model of compositional structure, (relatively) low-dimensional, high quality output
Cons: limited topological variation, no continuous geometric variation (for generation), no hierarchy, huge effort to segment & label training data

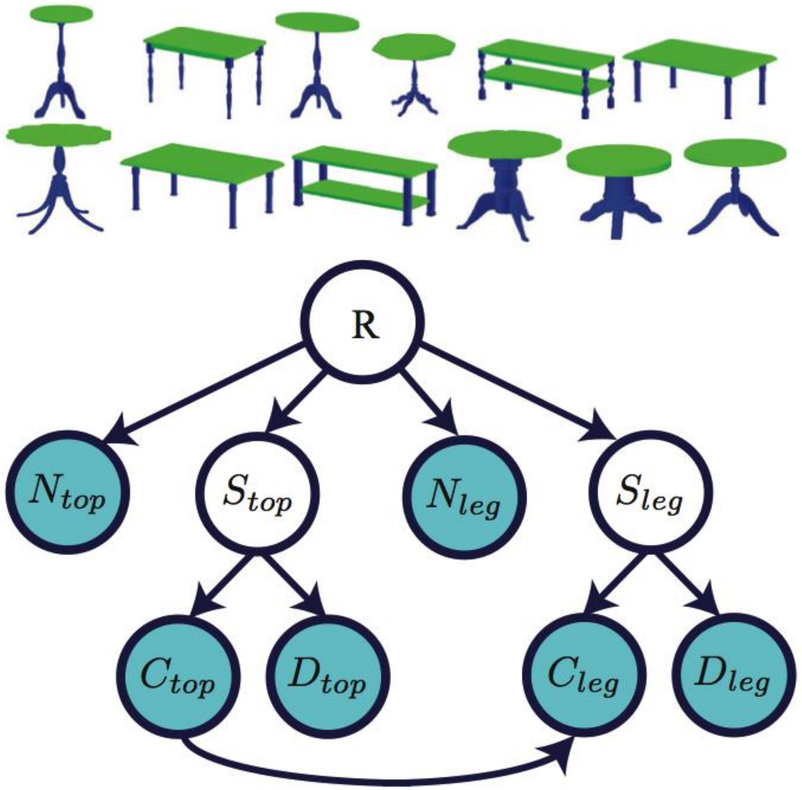
?



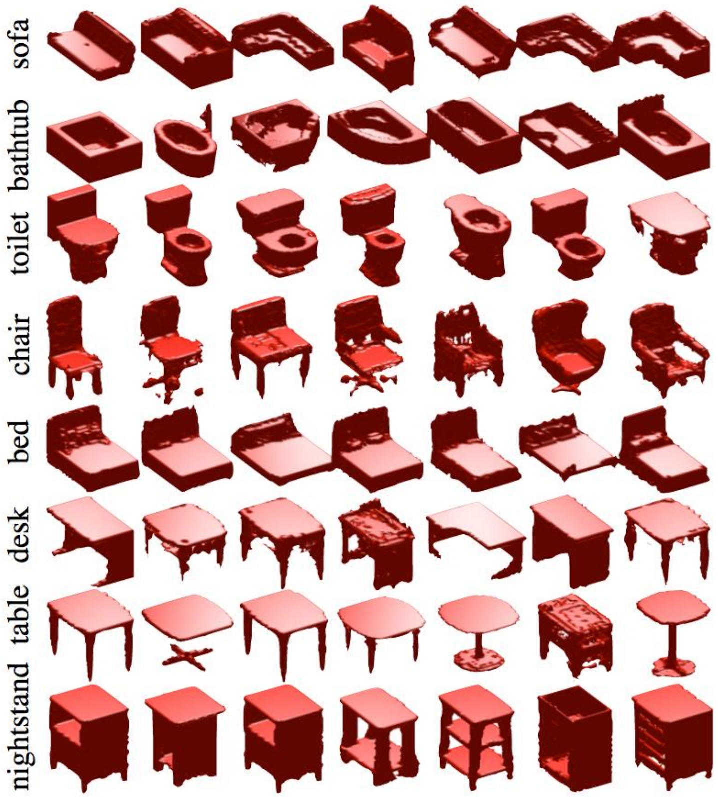
Unsupervised [Wu et al. ’15]

Pros: arbitrary geometry/topology, unsupervised
Cons: low-resolution, no explicit separation of structure vs fine geometry, no guarantee of symmetry/adjacency, no hierarchy, lots of parameters, lots of training data

Structural PGM vs Volumetric DNN



Strongly supervised [Kalogerakis et al. ’12]



Unsupervised [Wu et al. ’15]

Pros: direct model of compositional structure, (relatively) low-dimensional, high quality output
Cons: limited topological variation, no continuous geometric variation (for generation), no hierarchy, huge effort to segment & label training data

Pros: arbitrary geometry/topology, unsupervised
Cons: low-resolution, no explicit separation of structure vs fine geometry, no guarantee of symmetry/adjacency, no hierarchy, lots of parameters, lots of training data

GRASS: Generative neural networks over unlabeled part layouts

- GRASS factorizes a shape into a hierarchical layout of simplified parts, plus fine-grained part geometries
- Weakly supervised:
  - ✓ segments
  - ✗ labels
  - ✗ manually-specified “ground truth” hierarchies
- Structure-aware: learns a generative distribution over richly informative structures

Three Challenges

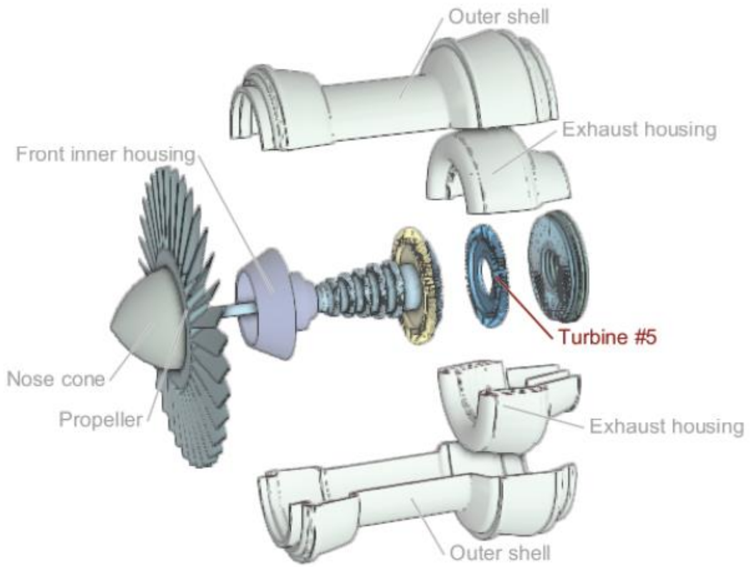
- **Challenge 1:** Ingest and generate arbitrary part layouts with a fixed-dimensional network
  - Convolution doesn't work over arbitrary graphs

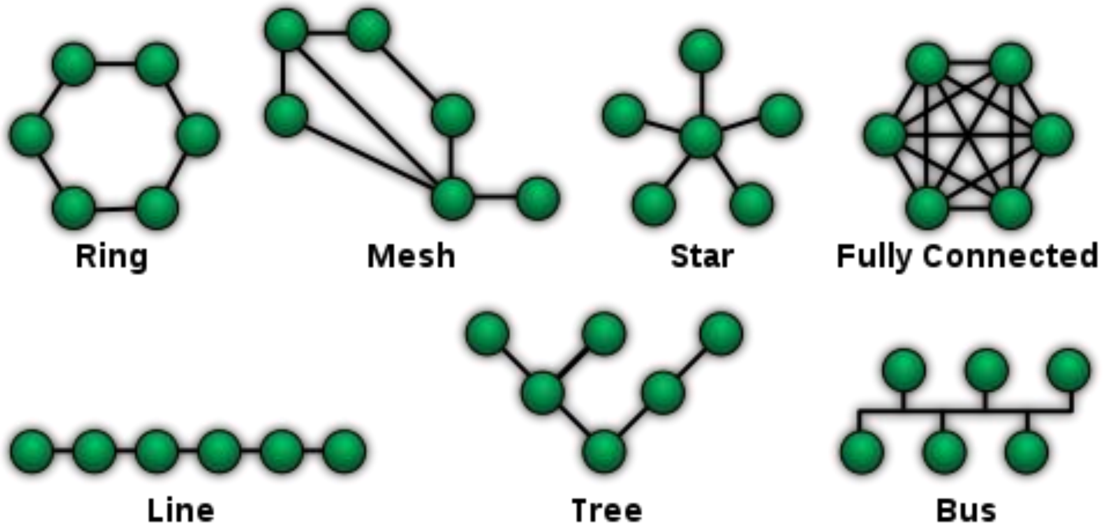
- **Challenge 2:** Map a layout invertibly to a fixed-D code ("Shape DNA") that implicitly captures adjacency, symmetry and hierarchy

- **Challenge 3:** Map layout features to fine geometry

Huge variety of (attributed) graphs

- Arbitrary numbers/types of vertices (parts), arbitrary numbers of connections (adjacencies/symmetries)



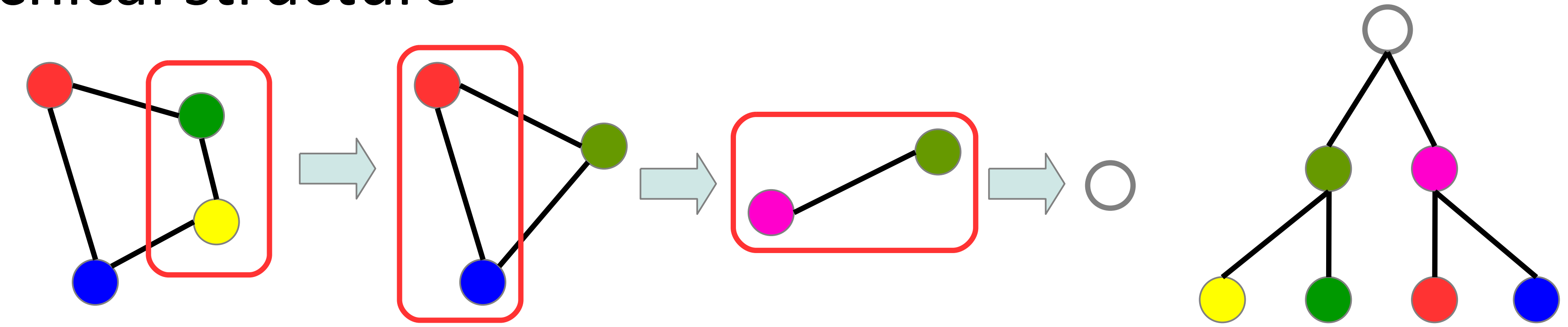


- For linear graphs (chains) of arbitrary length, we can use a recurrent neural network (RNN/LSTM)

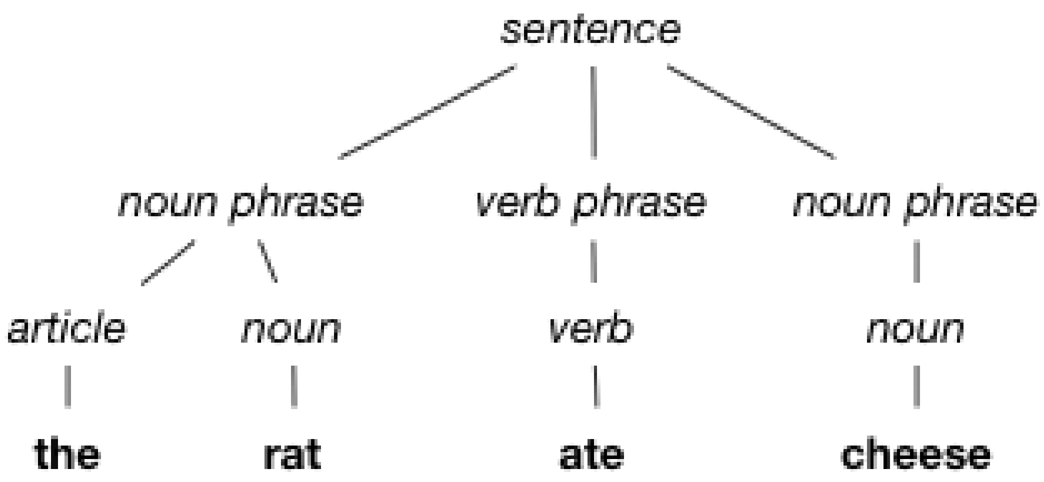
Li et al. 2008, Wikipedia

Key Insight

- Edges of a graph can be collapsed sequentially to yield a hierarchical structure

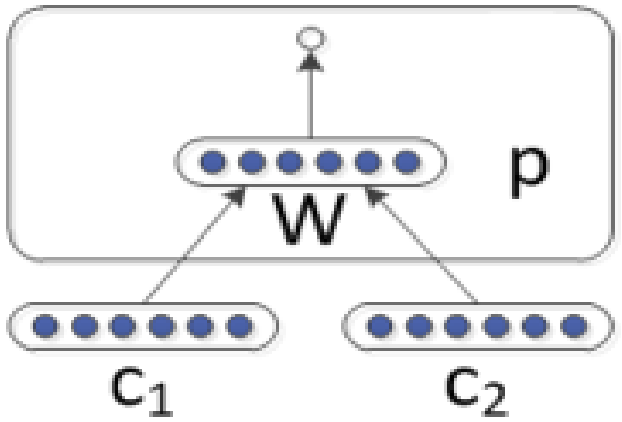


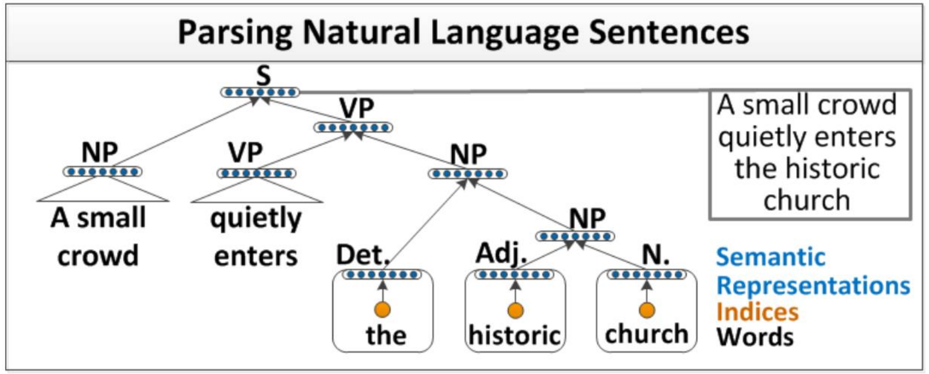
- Looks like a parse tree for a sentence!
- … and there are unsupervised sentence parsers

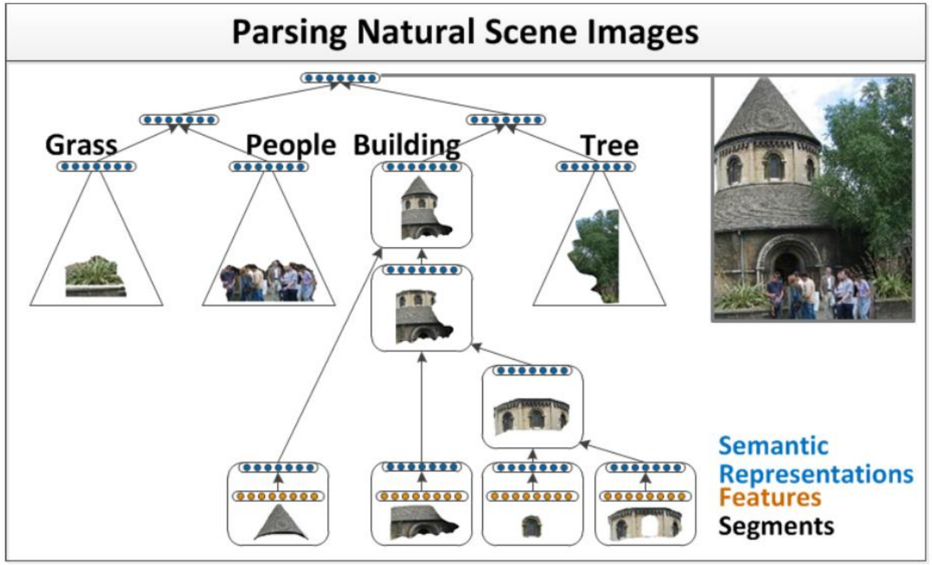


Recursive Neural Network (RvNN)

- Repeatedly merge two nodes into one
- Each node has an n-D feature vector, computed recursively
- $p = f(W[c_1; c_2] + b)$

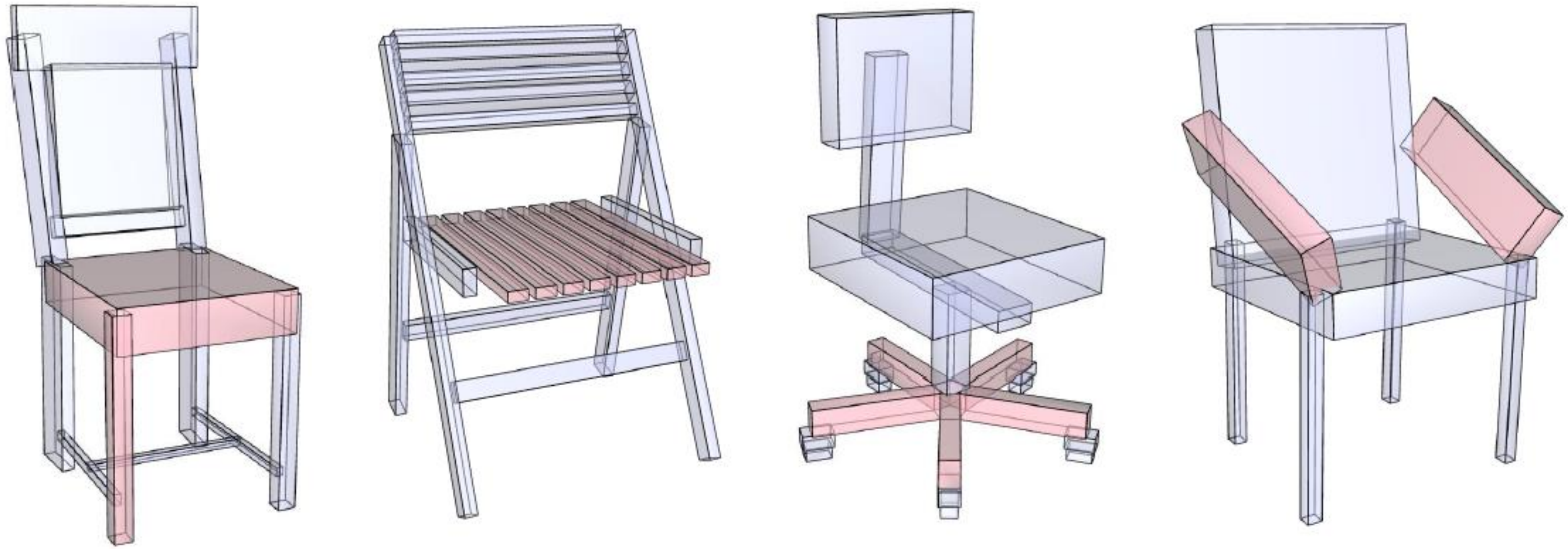






Socher et al. 2011

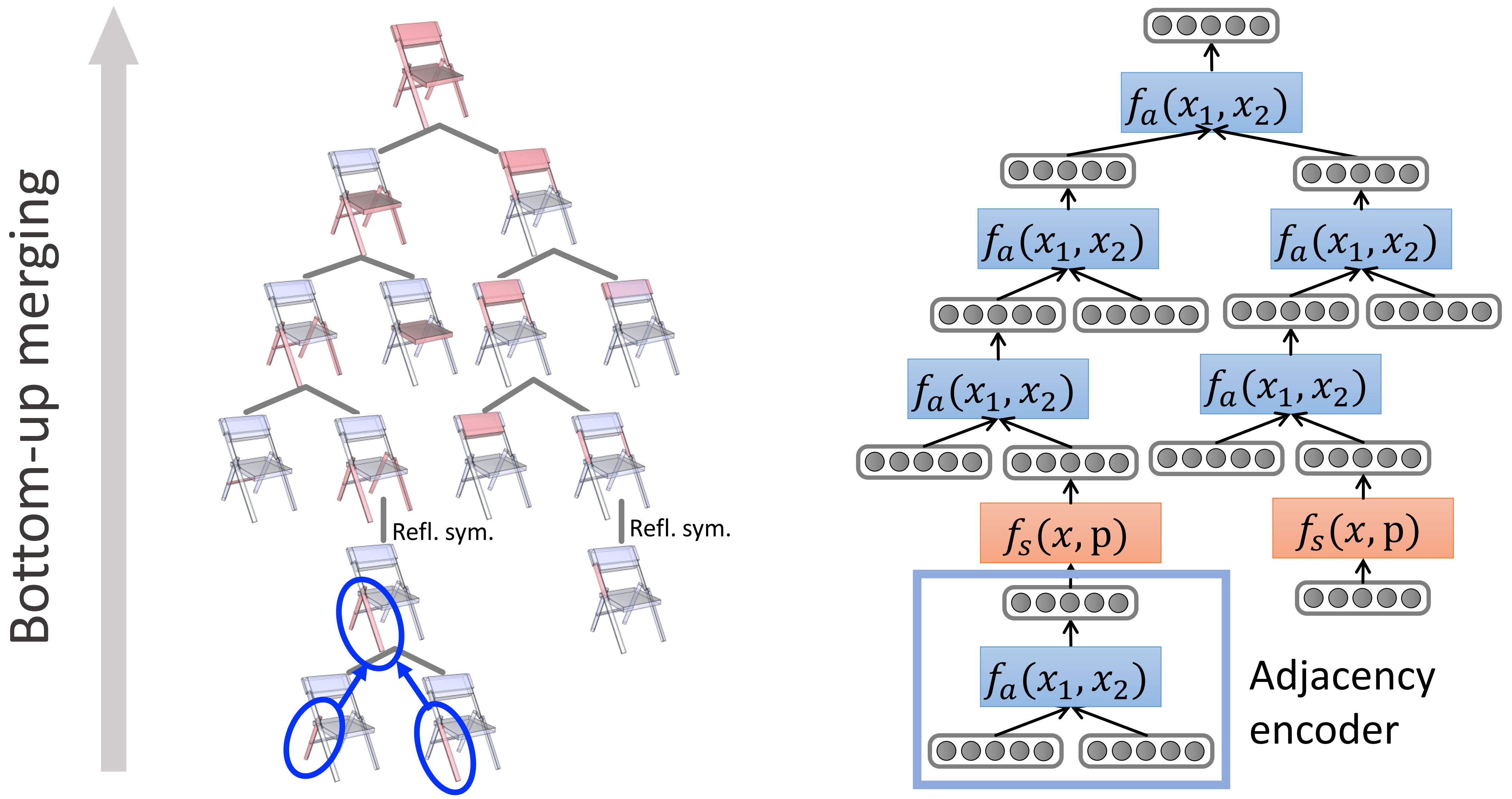
Different types of merges, varying cardinalities!



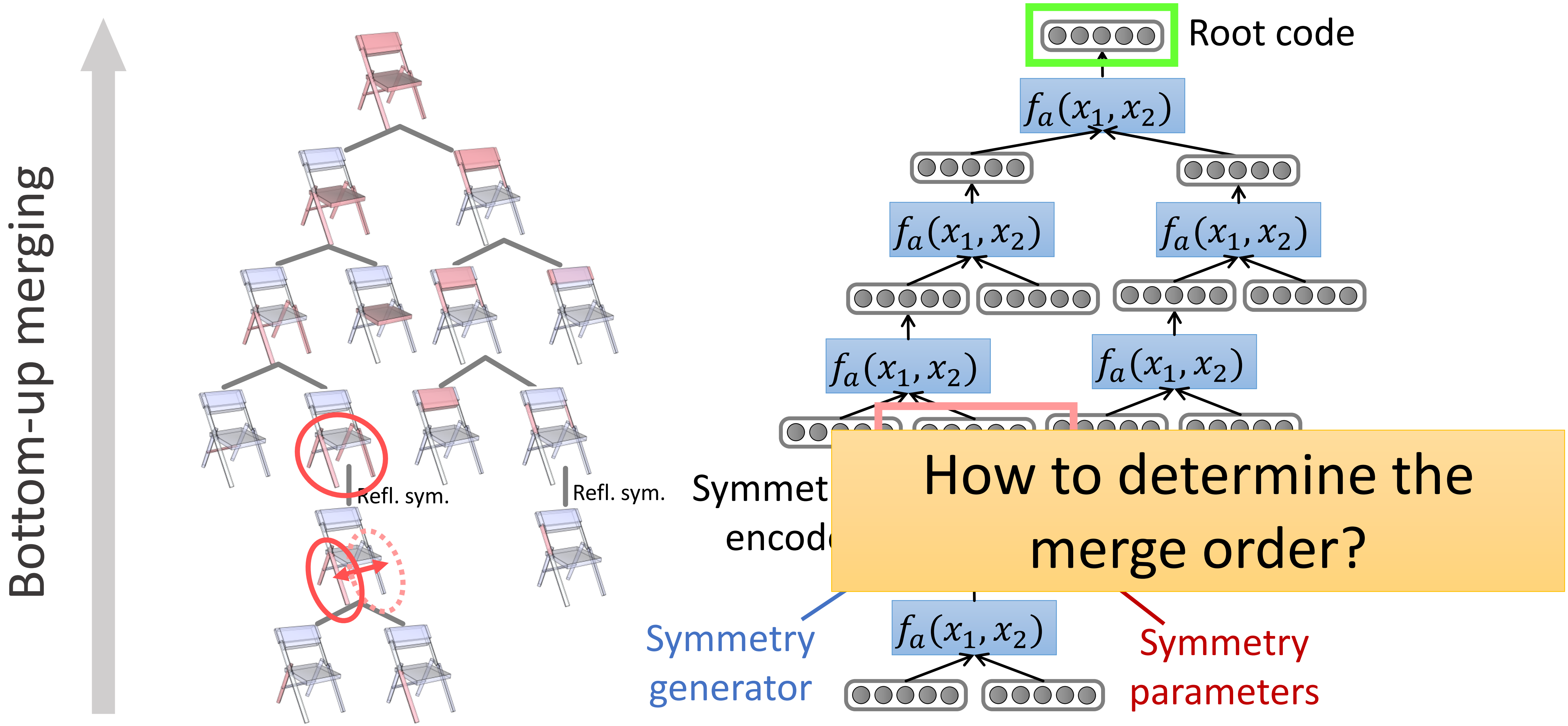
Adjacency | Translational symmetry | Rotational symmetry | Reflectional symmetry

- How to encode them to the same code space?
- How to decode them appropriately, given just a code?

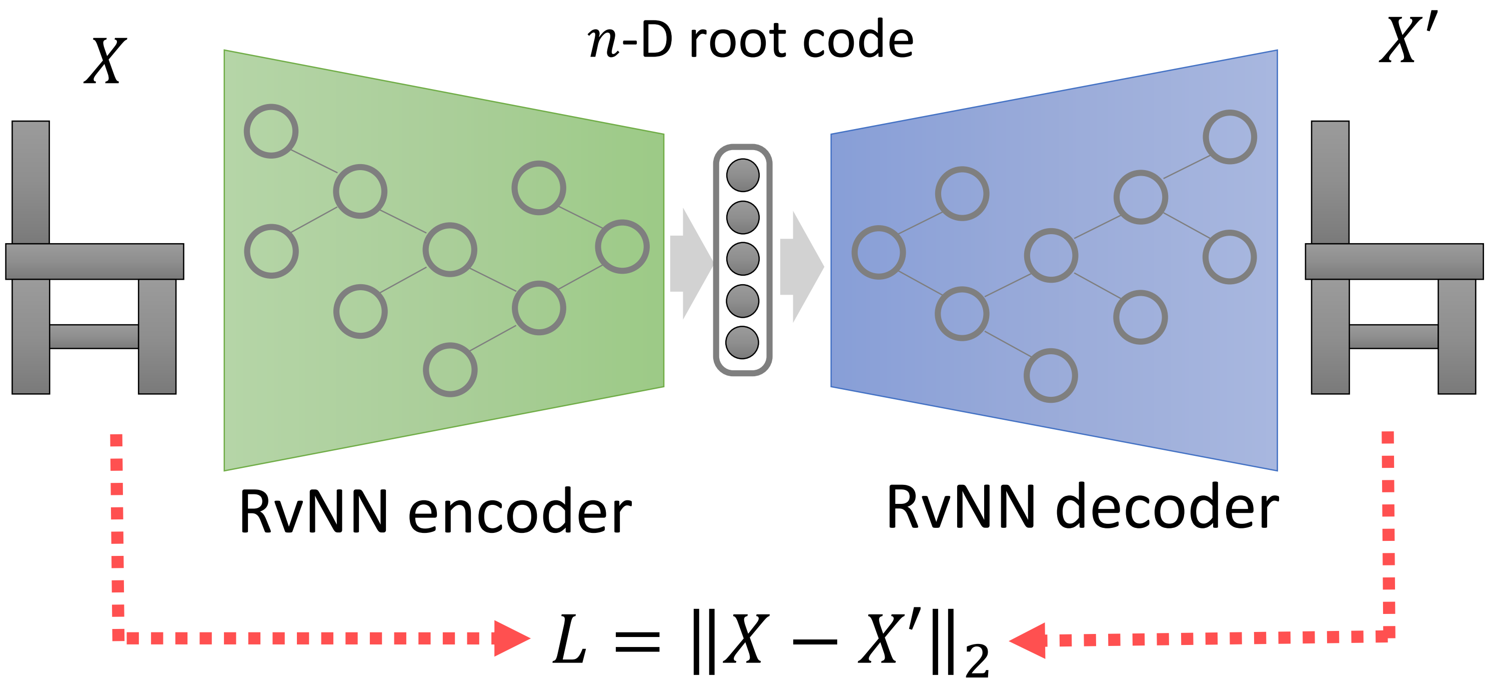
Recursively merging parts



Recursively merging parts



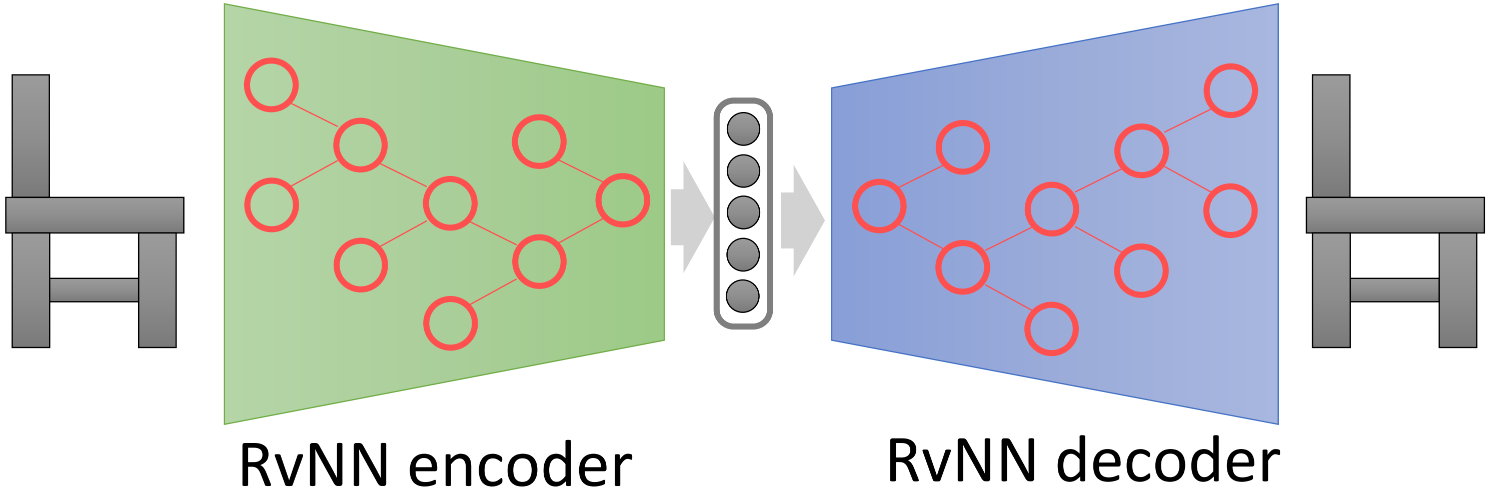
Training with reconstruction loss



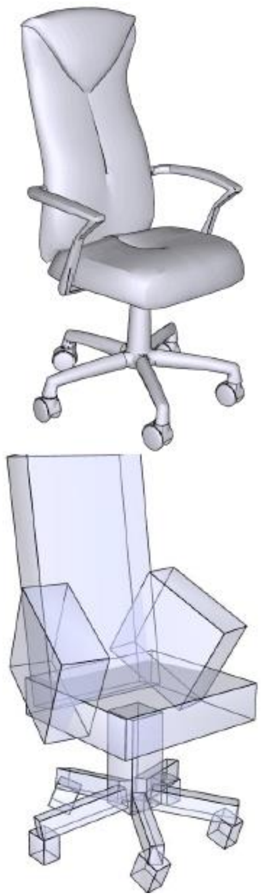
- Learn weights from a variety of randomly sampled merge orders for each box structure

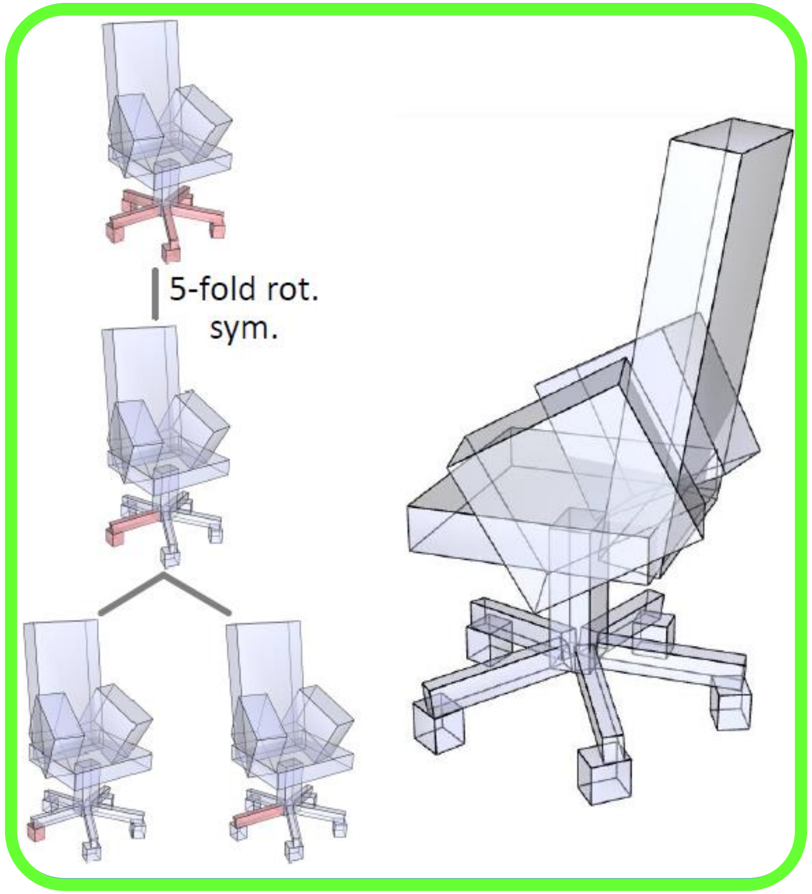
In **testing**

- Encoding: Given a box structure, determine the merge order as:
  - The hierarchy that gives the lowest reconstruction error

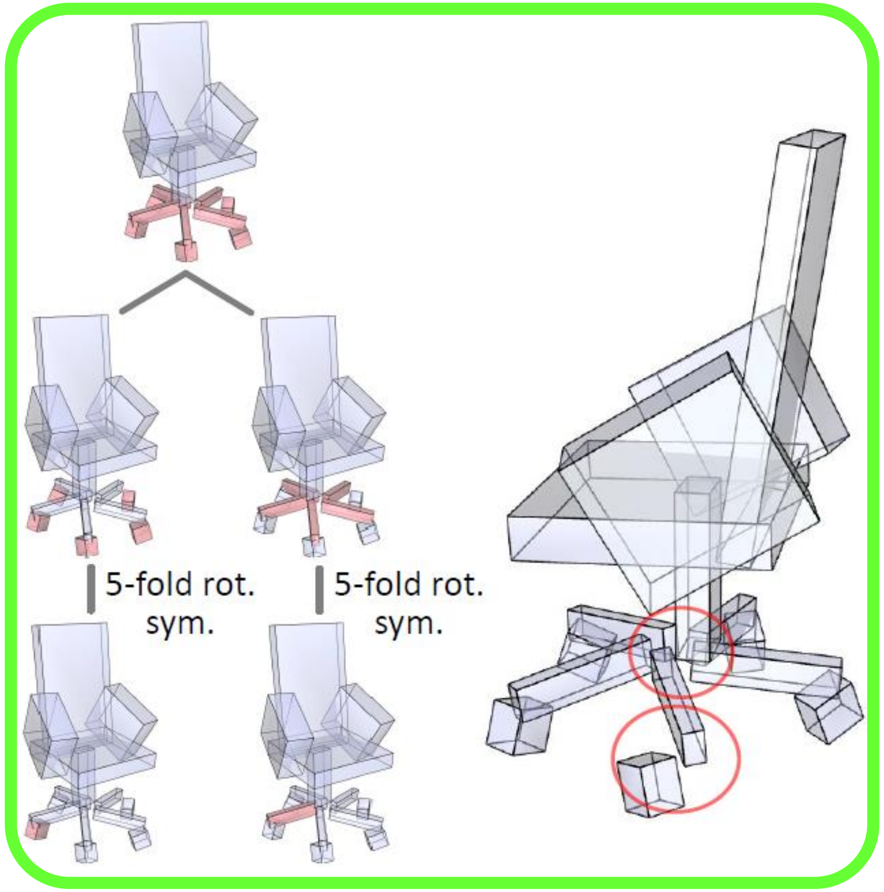


Inferring symmetry hierarchical reconstruction loss





Low reconstruction loss

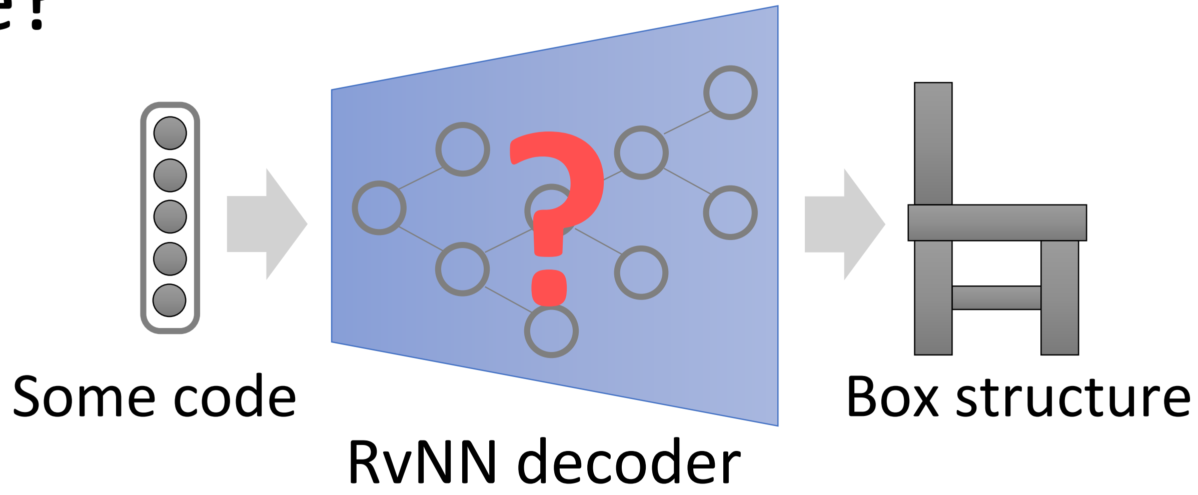


High reconstruction loss

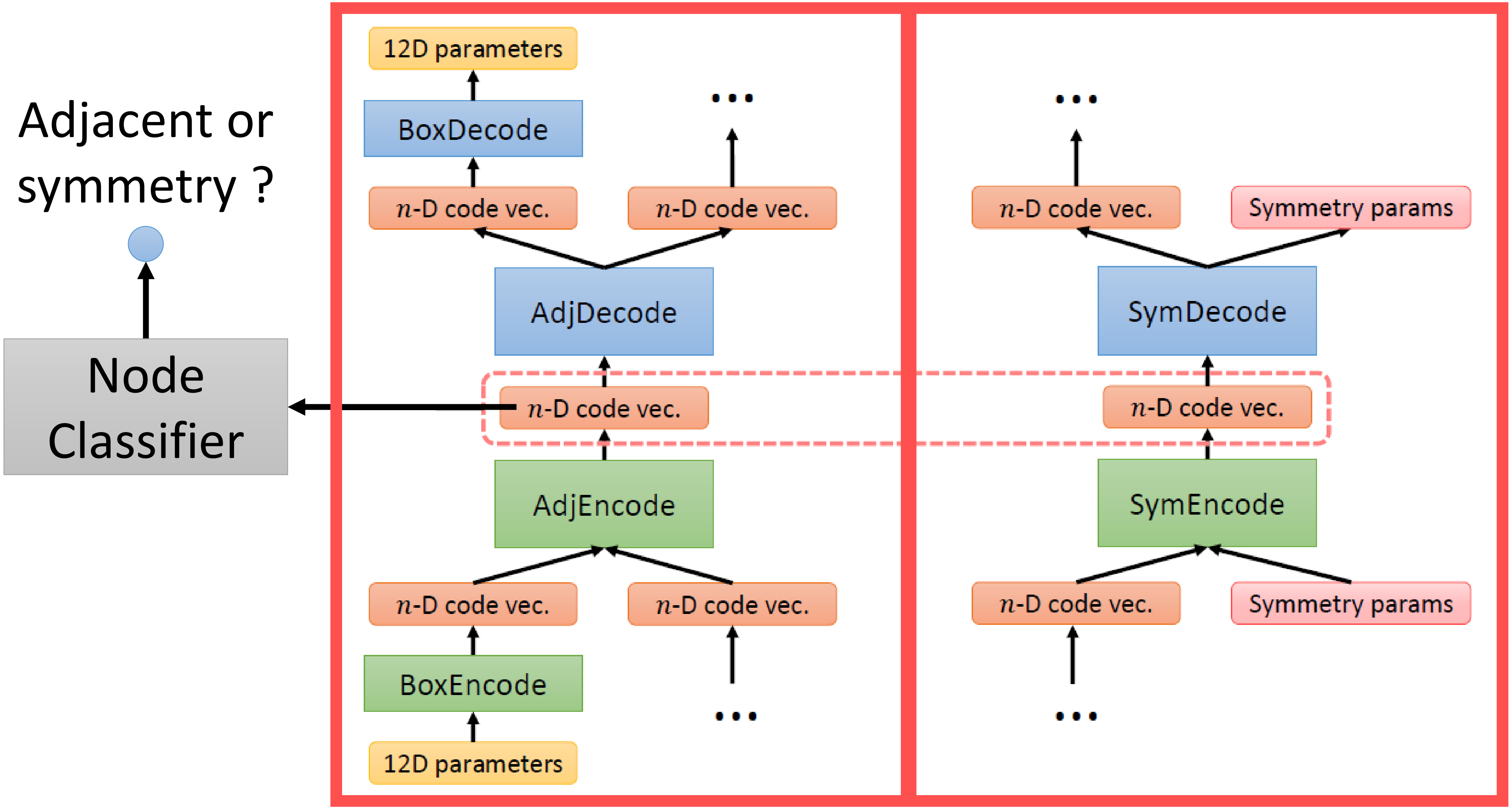
In **testing**

- Encoding: Given a box structure, determine the merge order as:
  - The hierarchy that gives the lowest reconstruction error

- Decoding: Given an arbitrary code, how to generate the corresponding structure?



How to know what type of encoder to use?



Making the network generative

- Variational Auto-Encoder (VAE): Learn a distribution that approximates the data distribution of true 3D structures

$$P(X) \approx P_{gt}(X)$$

- Marginalize over a latent “DNA” code

$$\text{maximize } P(X) = \int \underbrace{P(X|z;\underbrace{\theta}_{\text{Parameters}})}_{\text{Likelihood}} P(z)\,dz$$

Variational Bayes formulation

maximize $P(X) = \int P(X|z;\theta)P(z)dz$

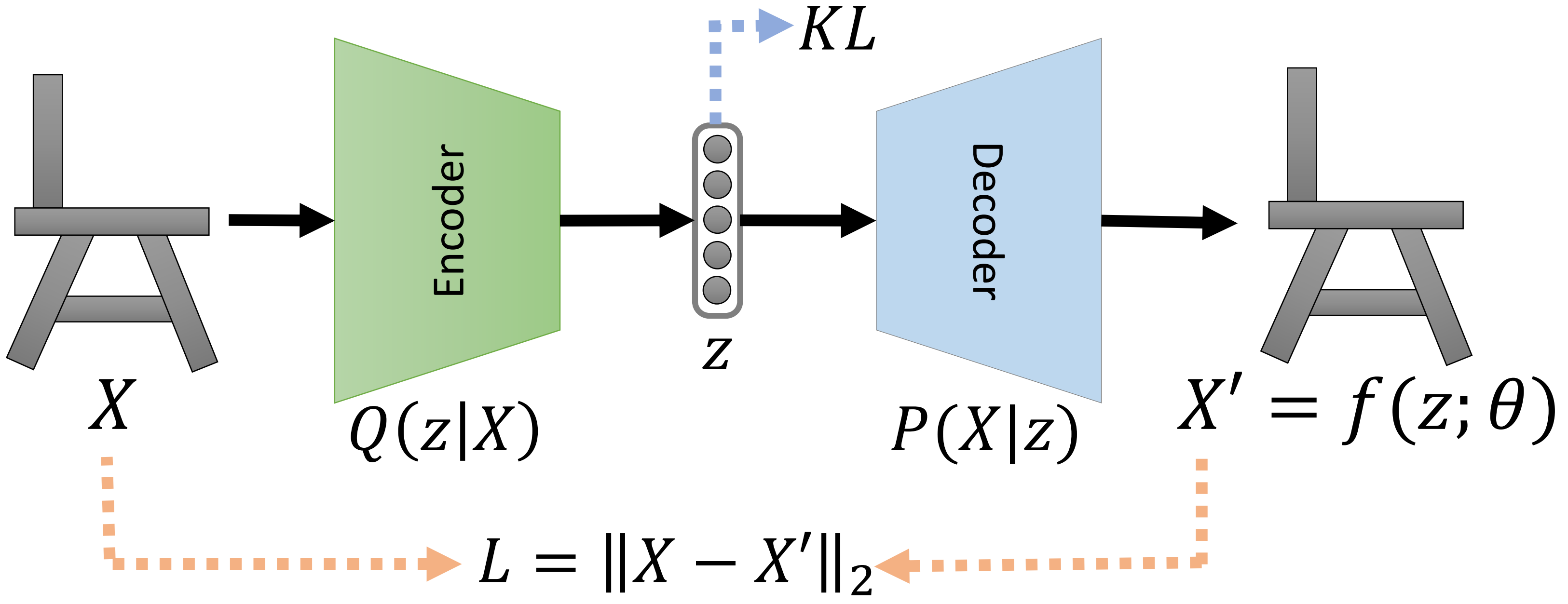
maximize $\boxed{E_{z\sim Q}[\log P(X|z)]} - \boxed{\mathcal{D}[Q(z|X)\|P(z)]}$

z should reconstruct X, given that it was drawn from $Q(z|X)$

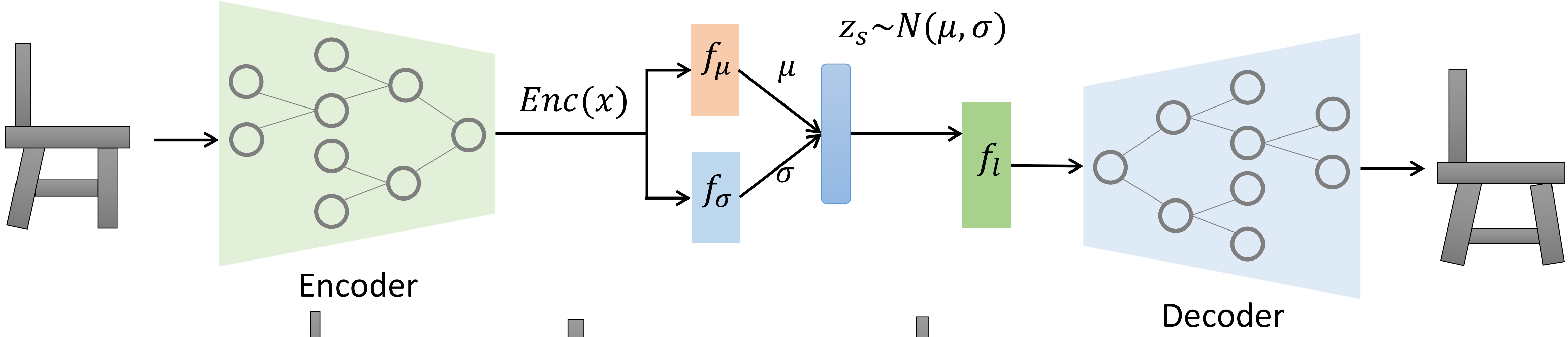
Assuming z's follow a normal distribution

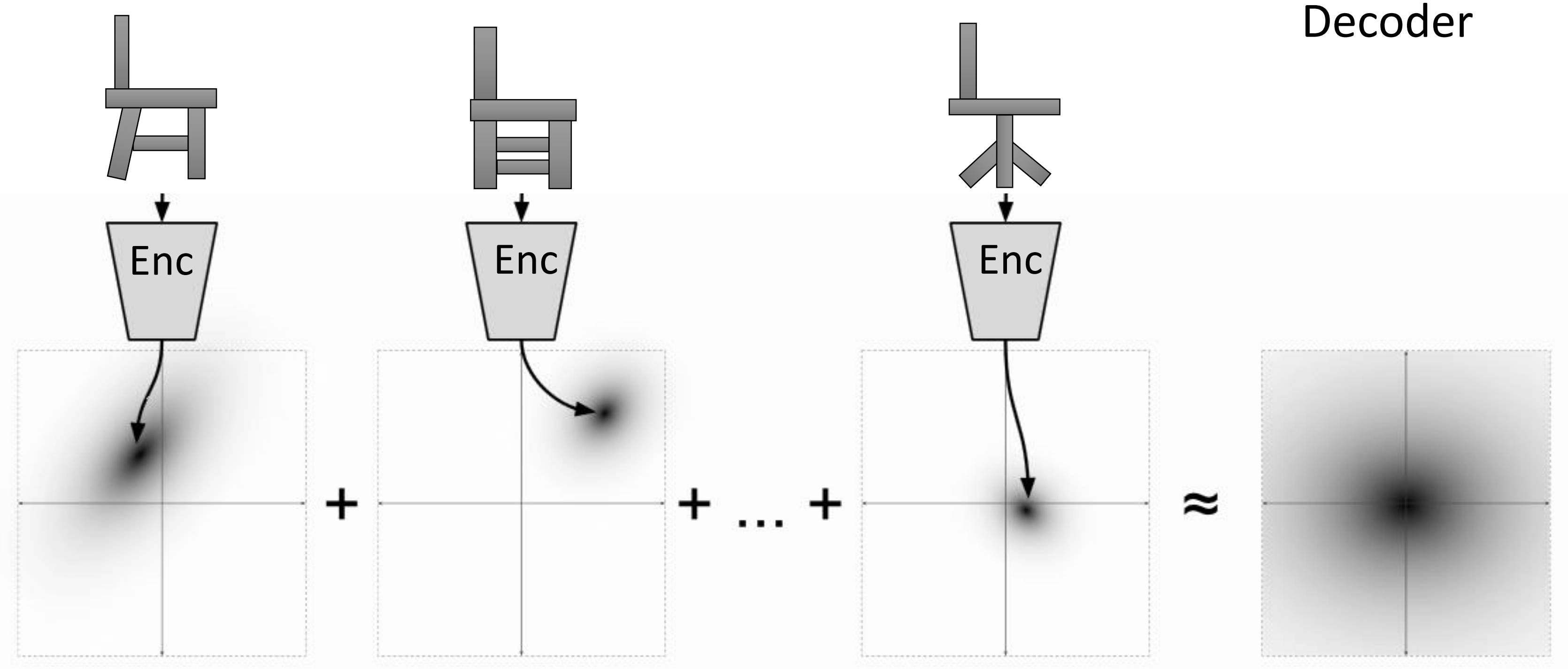
Variational Autoencoder (VAE)

maximize $\underbrace{\boxed{E_{z\sim Q}\left[\log P(X|z)\right]}}_{\text{Reconstruction loss}} - \underbrace{\boxed{\mathcal{D}\left[Q(z|X)\|P(z)\right]}}_{\text{KL divergence loss}}$

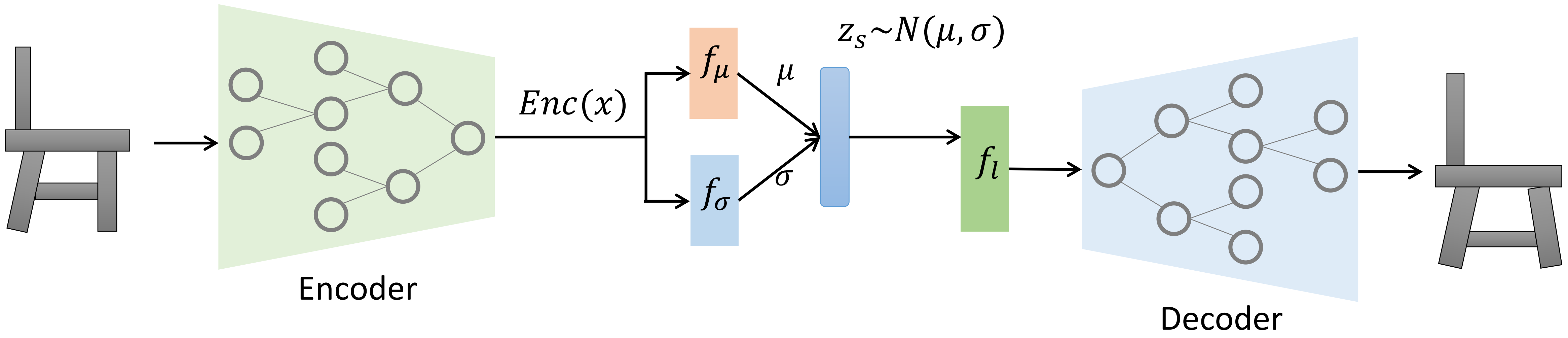


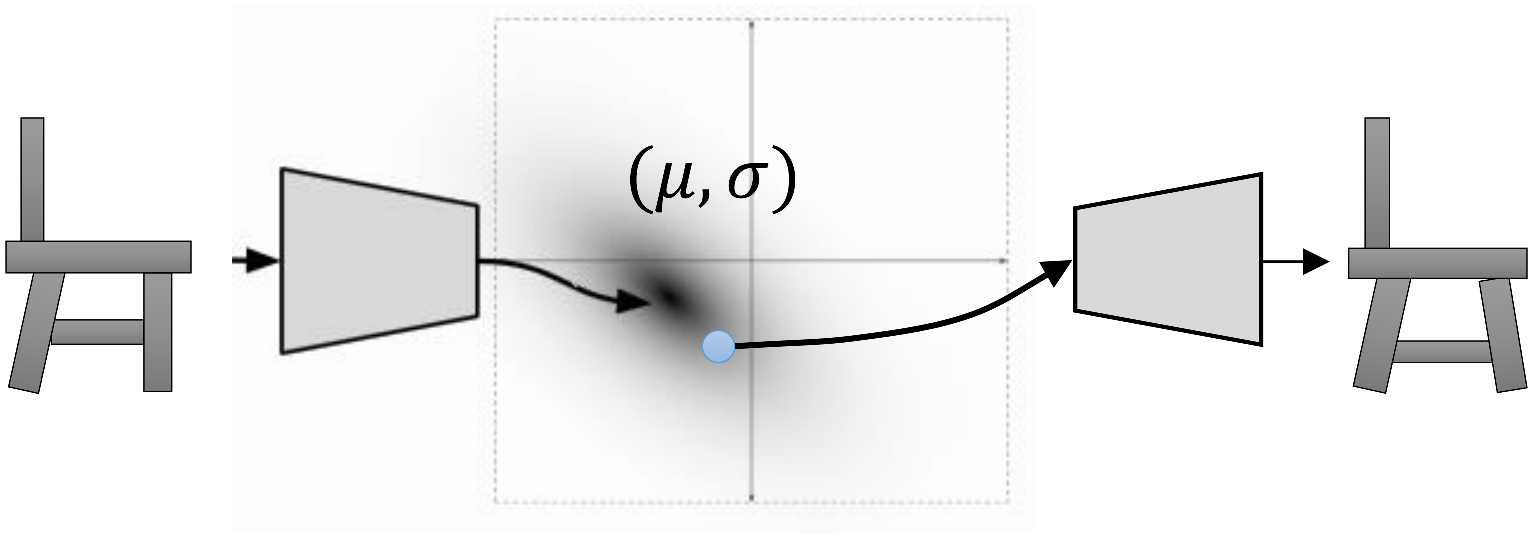
Variational Autoencoder (VAE)



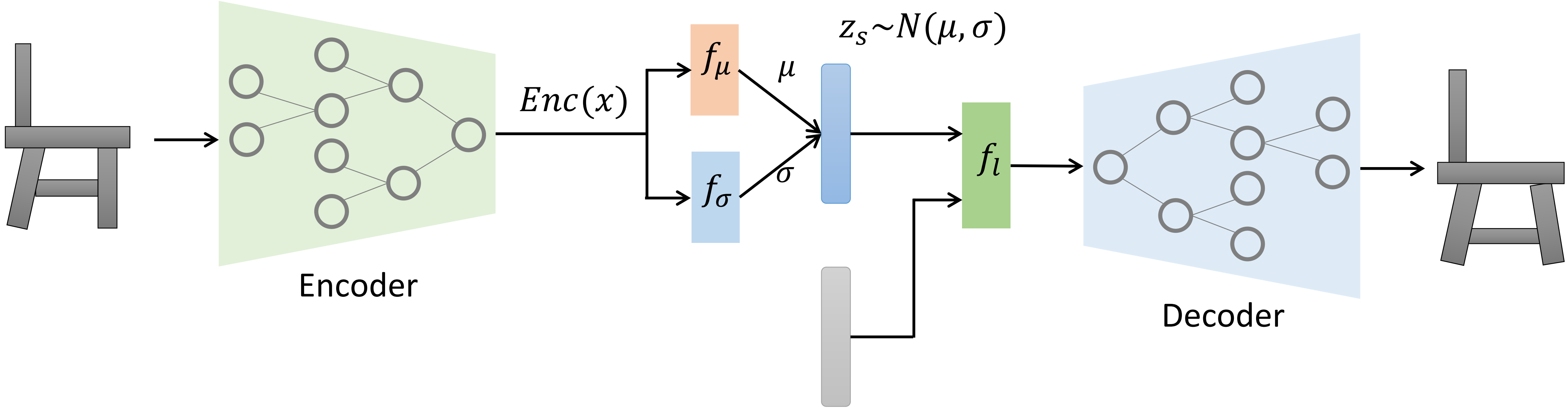


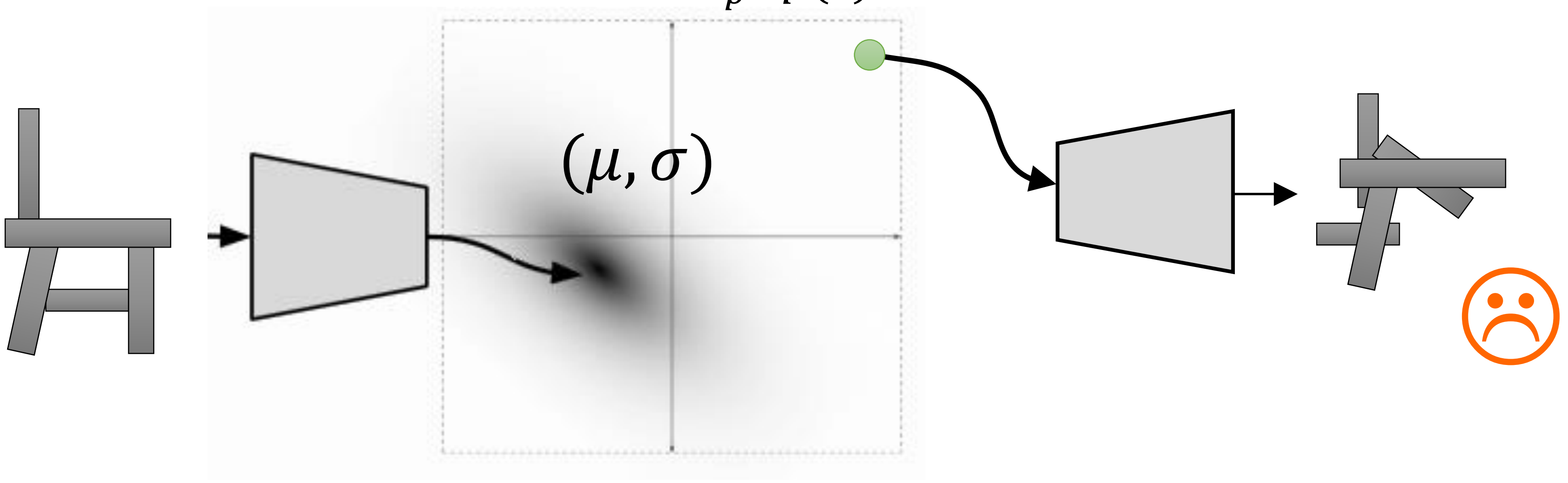
Sampling near μ is robust



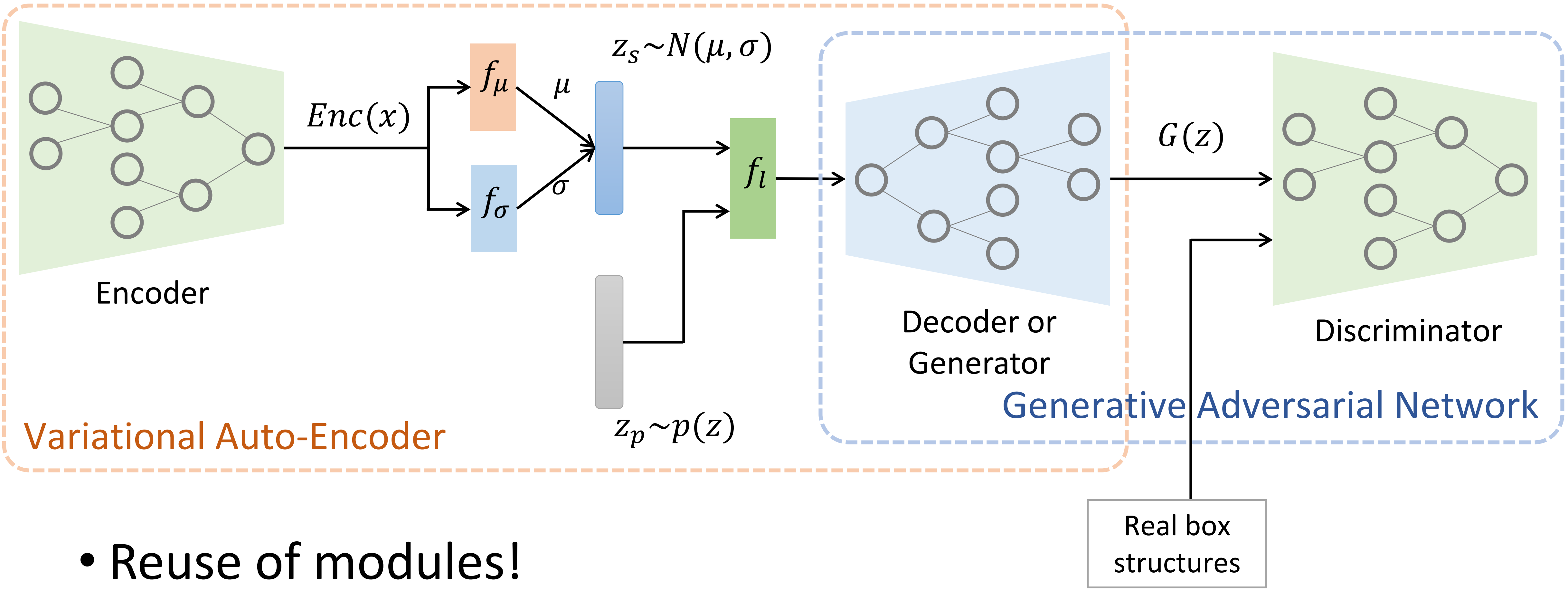


Sampling far away from μ?



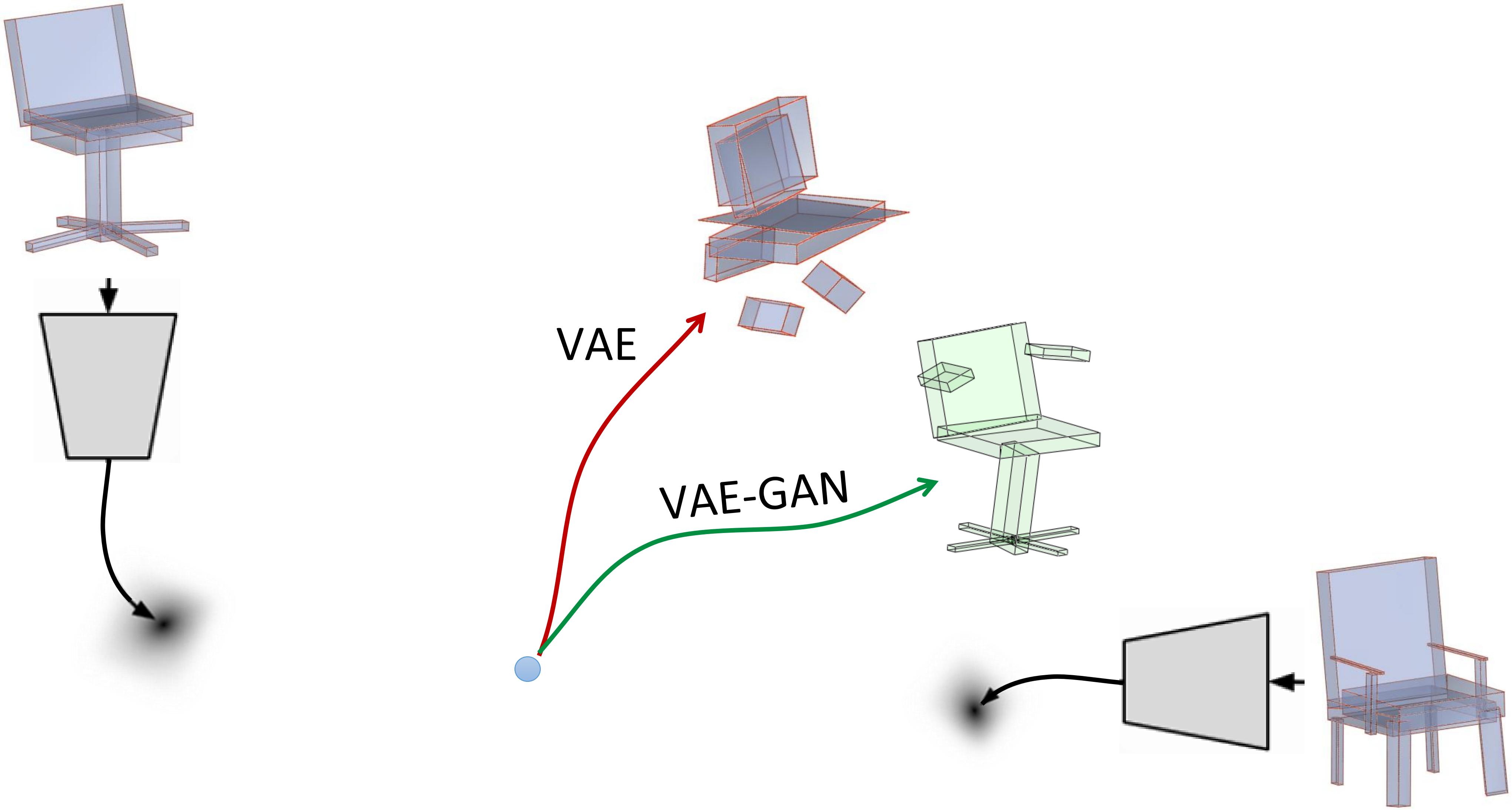


Adversarial training: VAE-GAN

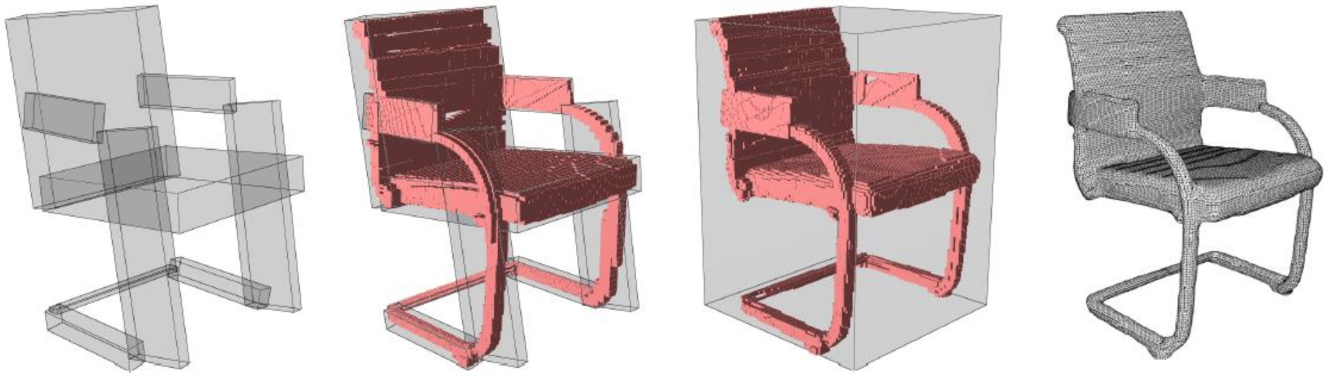


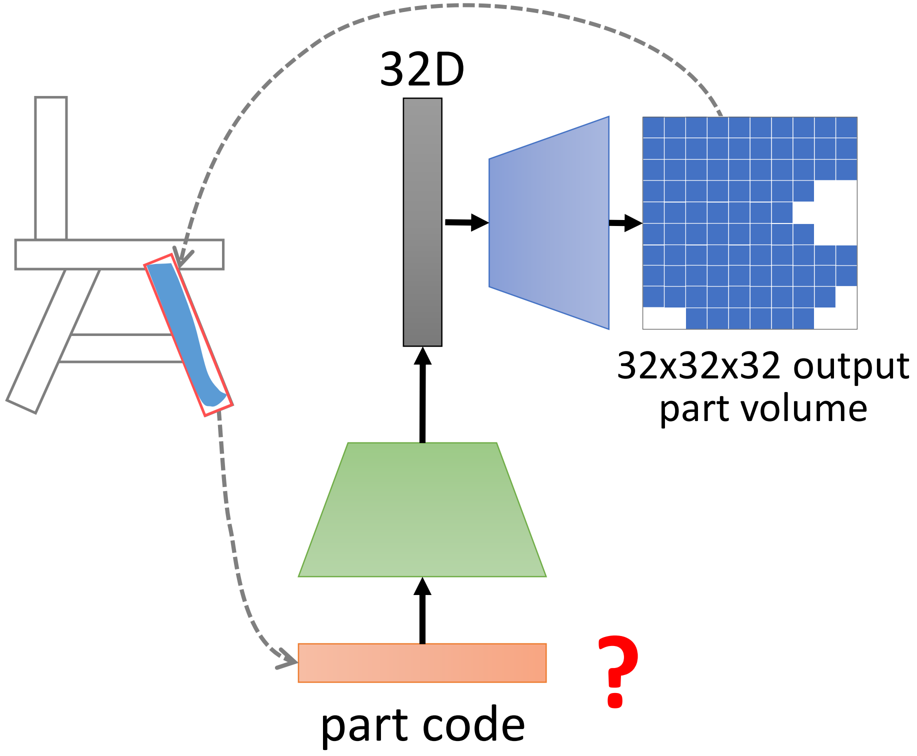
- Reuse of modules!
  - VAE decoder → GAN generator
  - VAE encoder → GAN discriminator

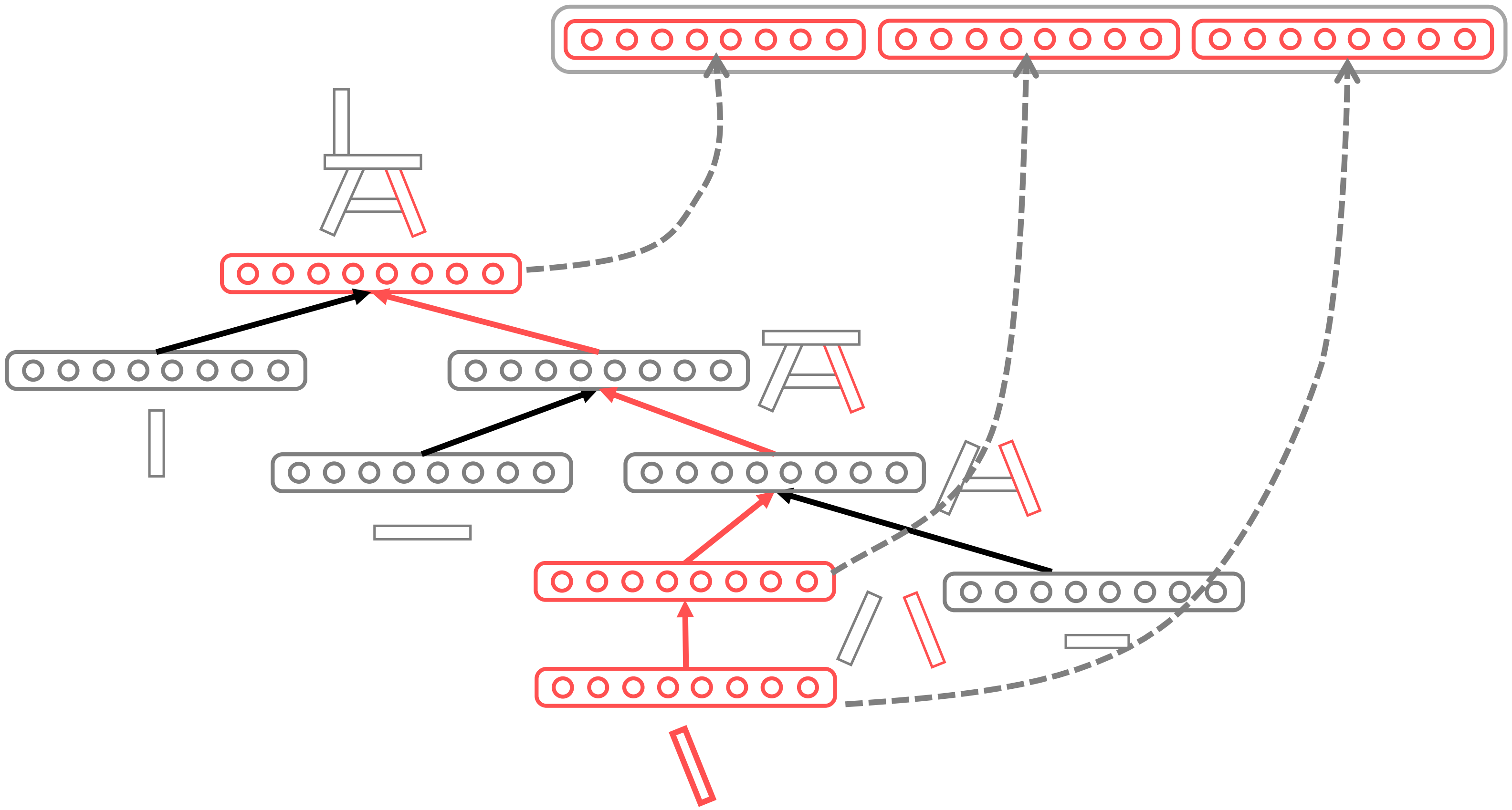
Benefit of adversarial training



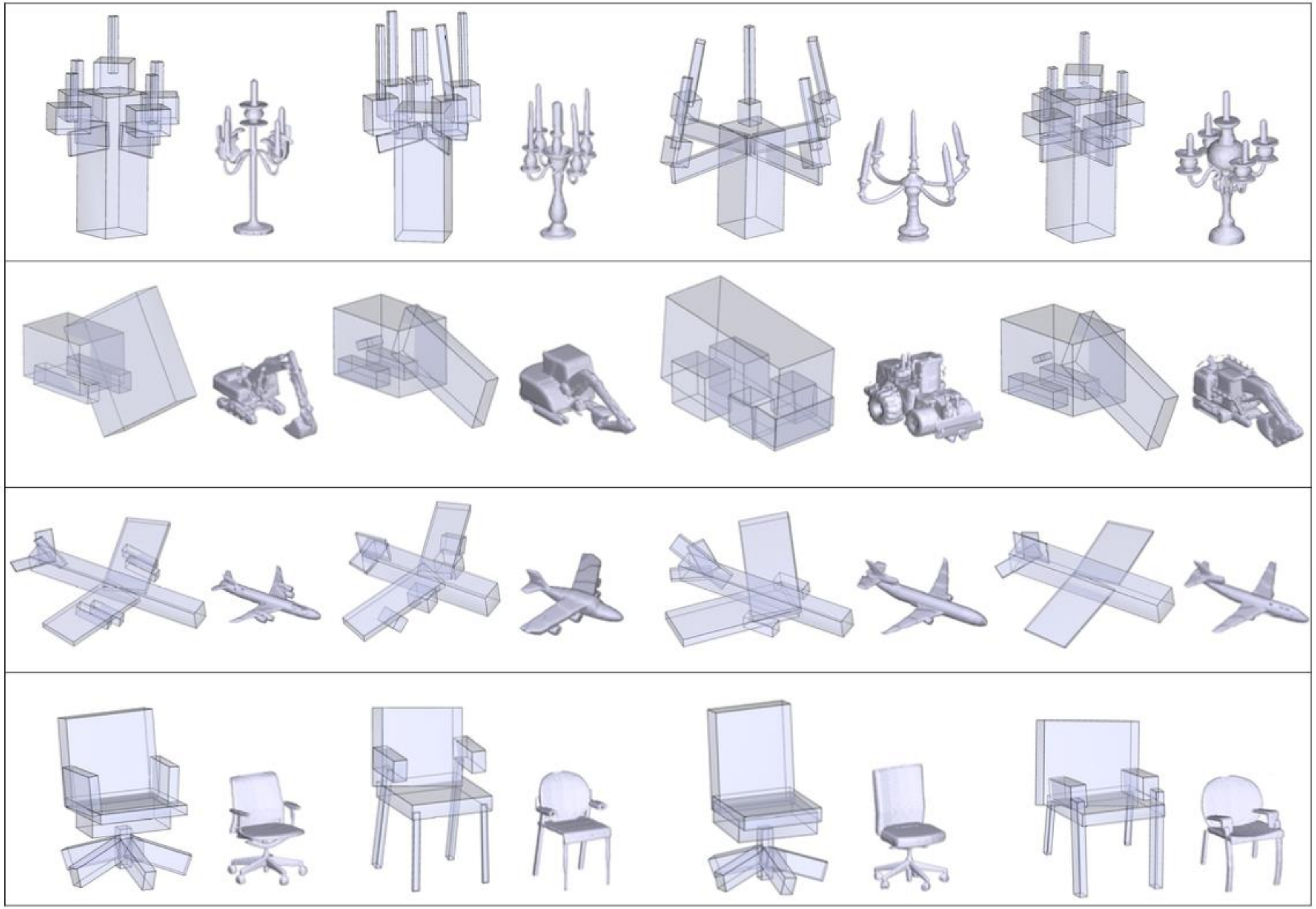
Part geometry synthesis



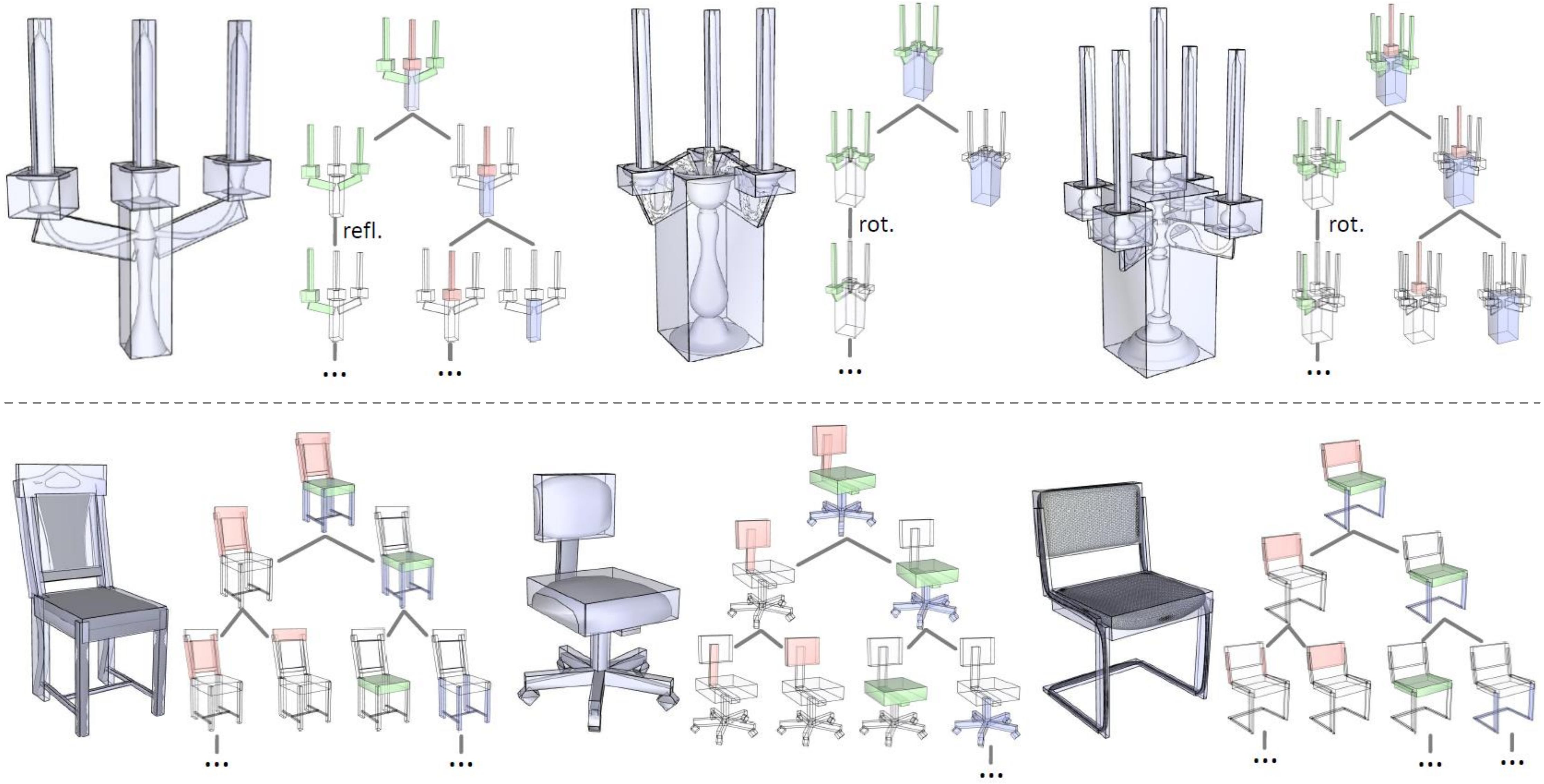




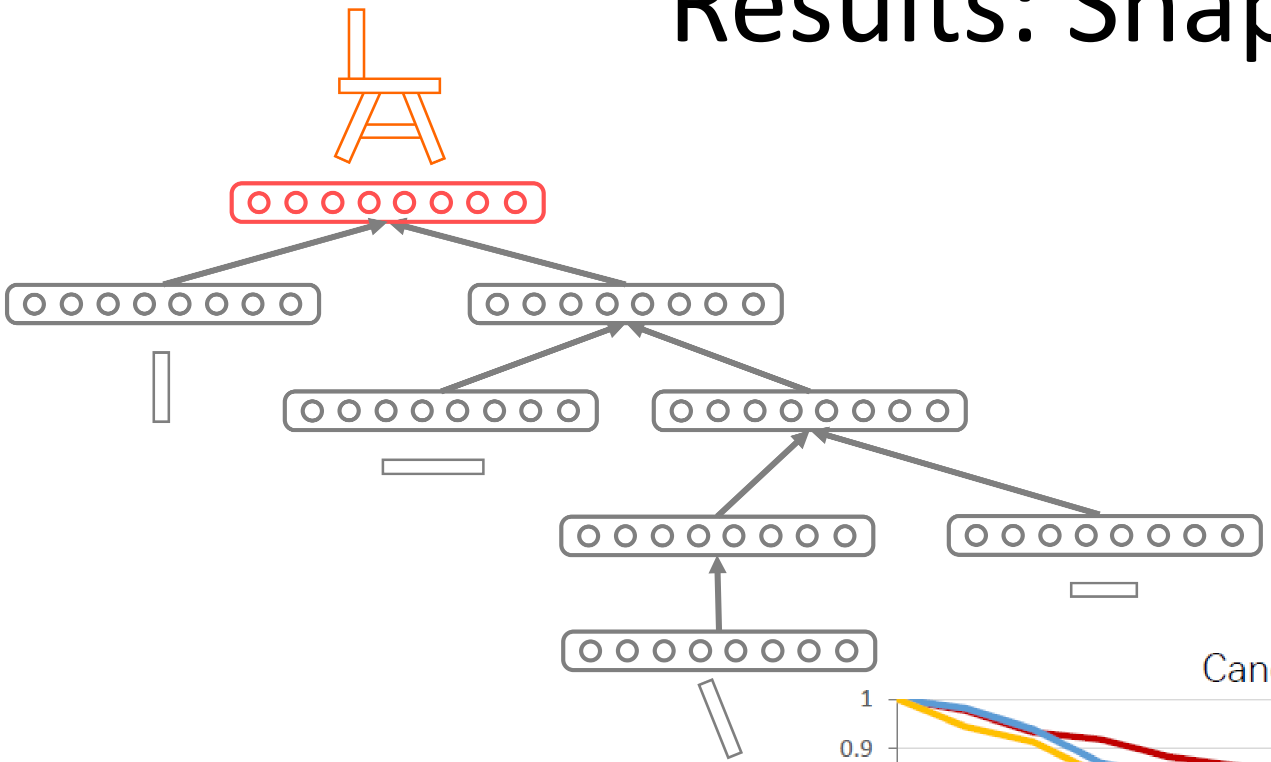
Results: Shape synthesis

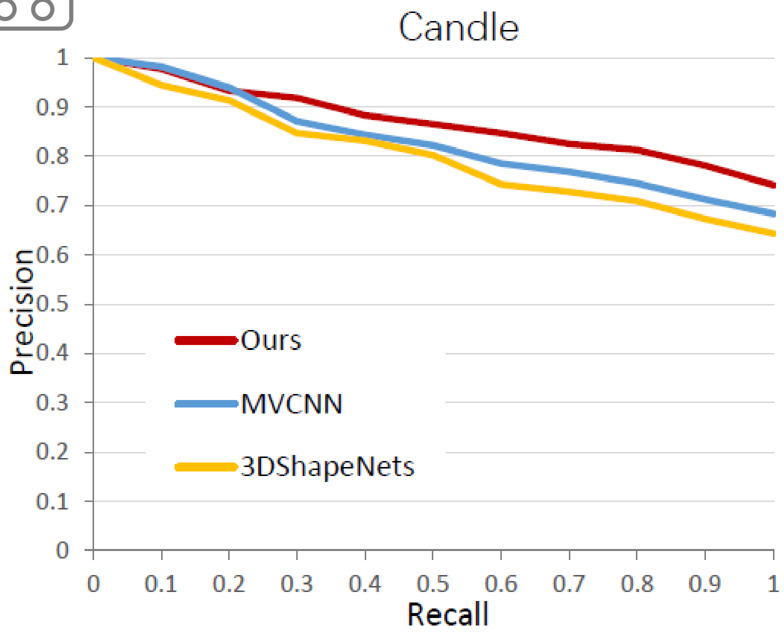


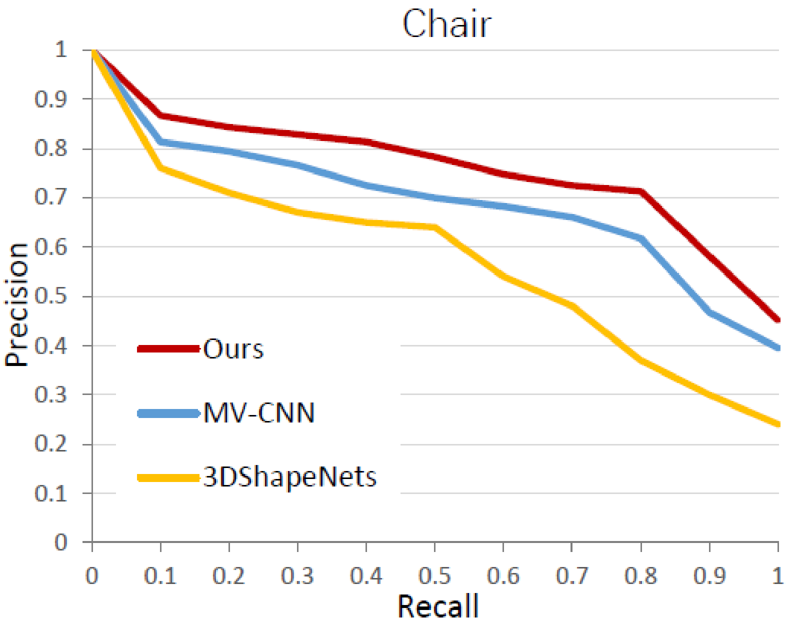
Results: Inferring consistent hierarchies



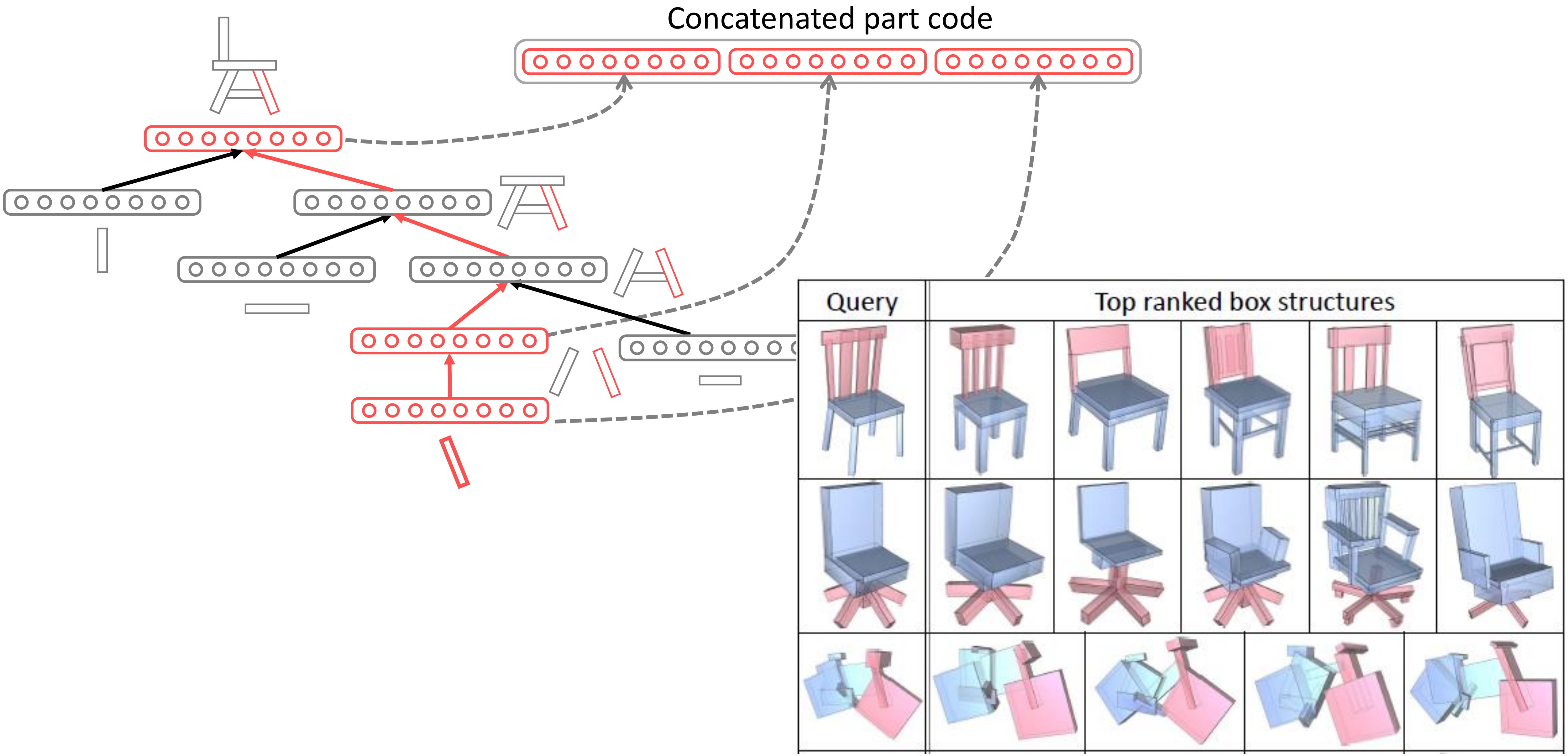
Results: Shape retrieval



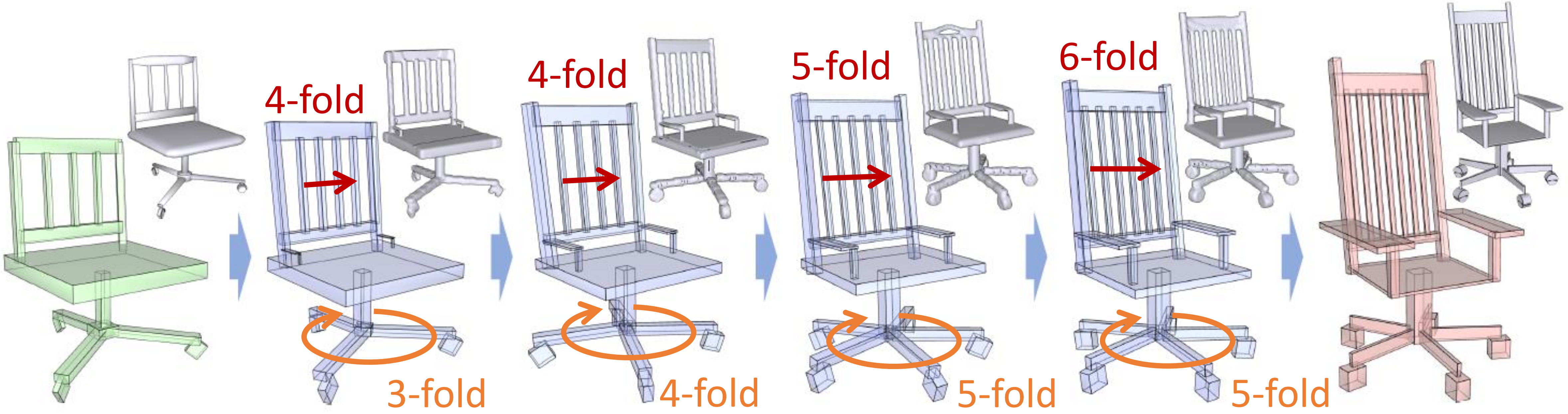




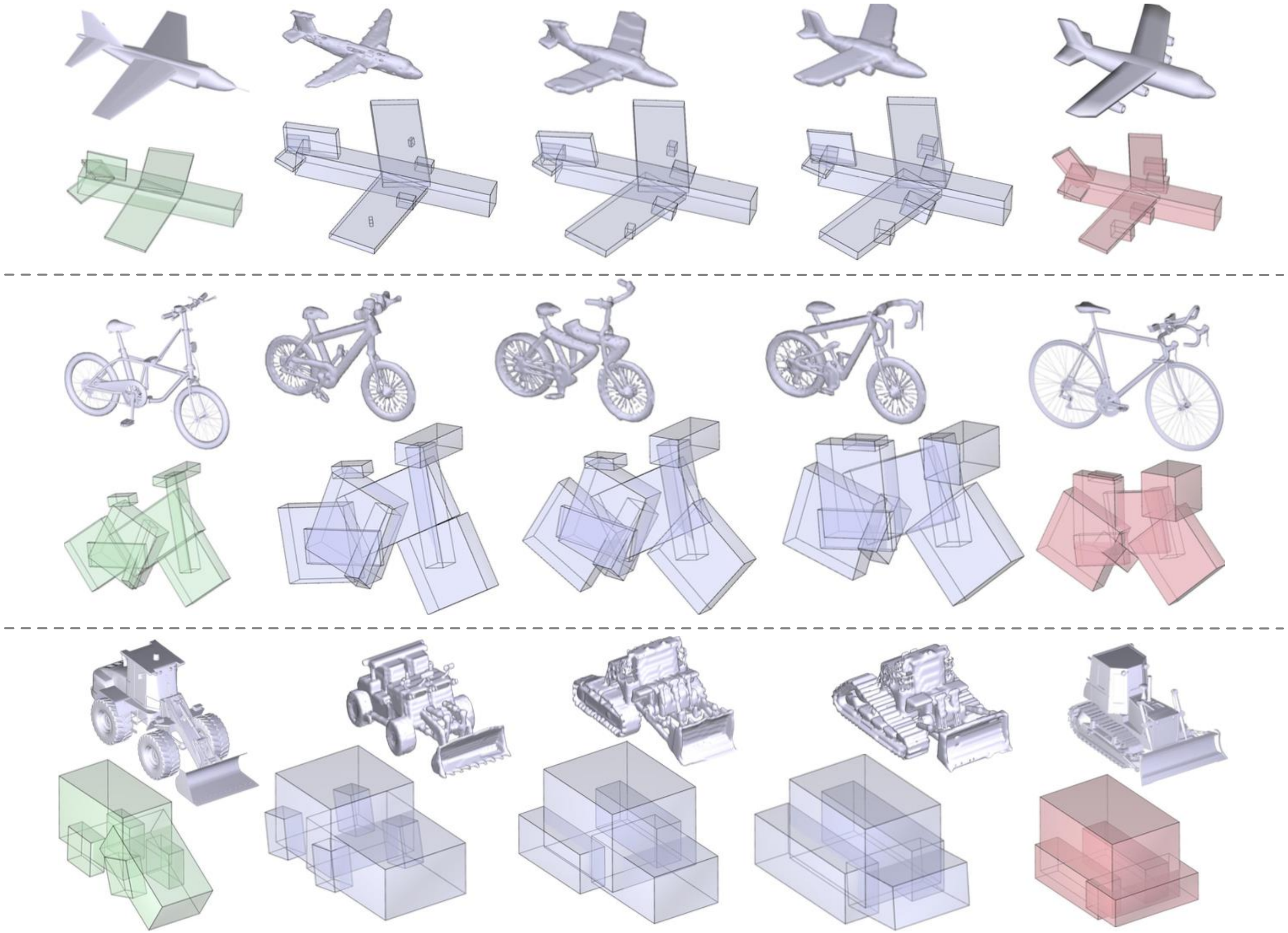
Results: Shape retrieval



Results: Shape interpolation

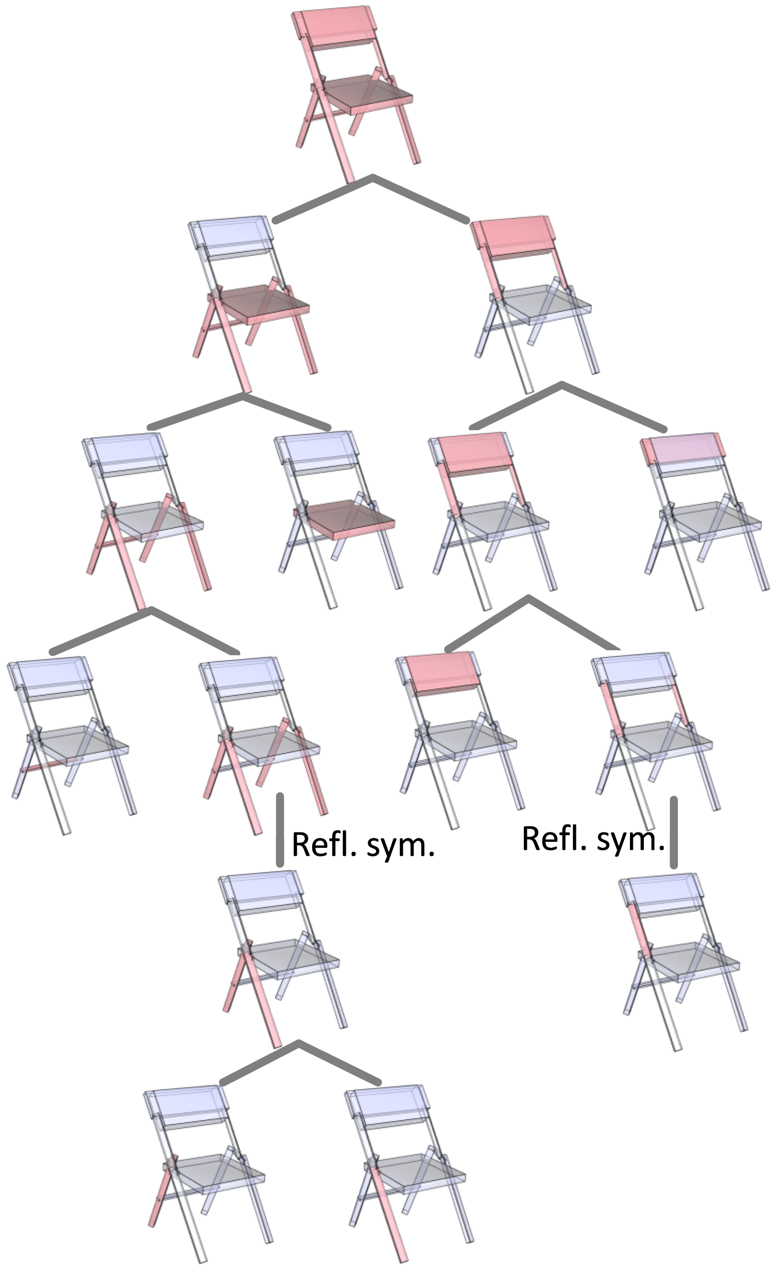


Results: Shape interpolation



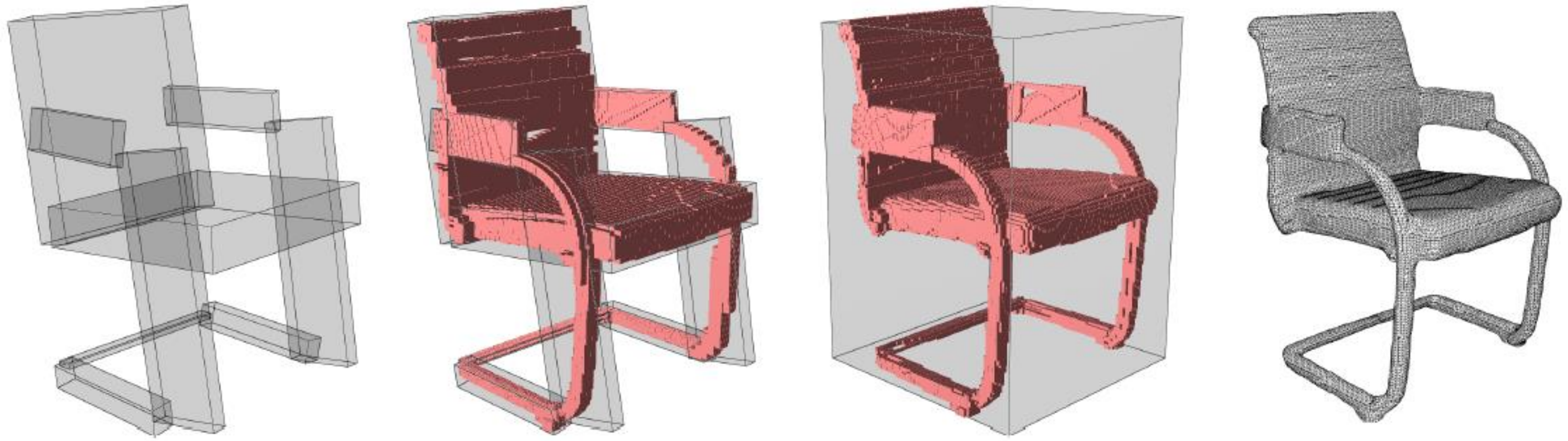
Discussion

- What does our model learn?
  - Hierarchical organization of part structures
  - A reasonable way to generate 3D structure
    - Part by part
    - Bottom-up
    - Hierarchical organization
  - This is the usual way how a human modeler creates a 3D model
    - Hierarchical scene graph



Discussion

- A general guideline for 3D shape generation
- Coarse-to-fine:
  - First generate coarse structure
  - Then generate fine details
  - May employ different representations and models



Acknowledgement

- Anonymous reviewers
- Help on data preparation
  - Yifei Shi, Min Liu, Chengjie Niu and Yizhi Wang
- Research grants from
  - NSFC, NSERC, NSF
  - Google Focused Research Award
  - Gifts from the Adobe, Qualcomm and Vicarious corporations.
  - Jun Li is a visiting PhD student of University of Bonn, supported by the CSC

Thank you!

Code & data available at

www.kevinkaixu.net