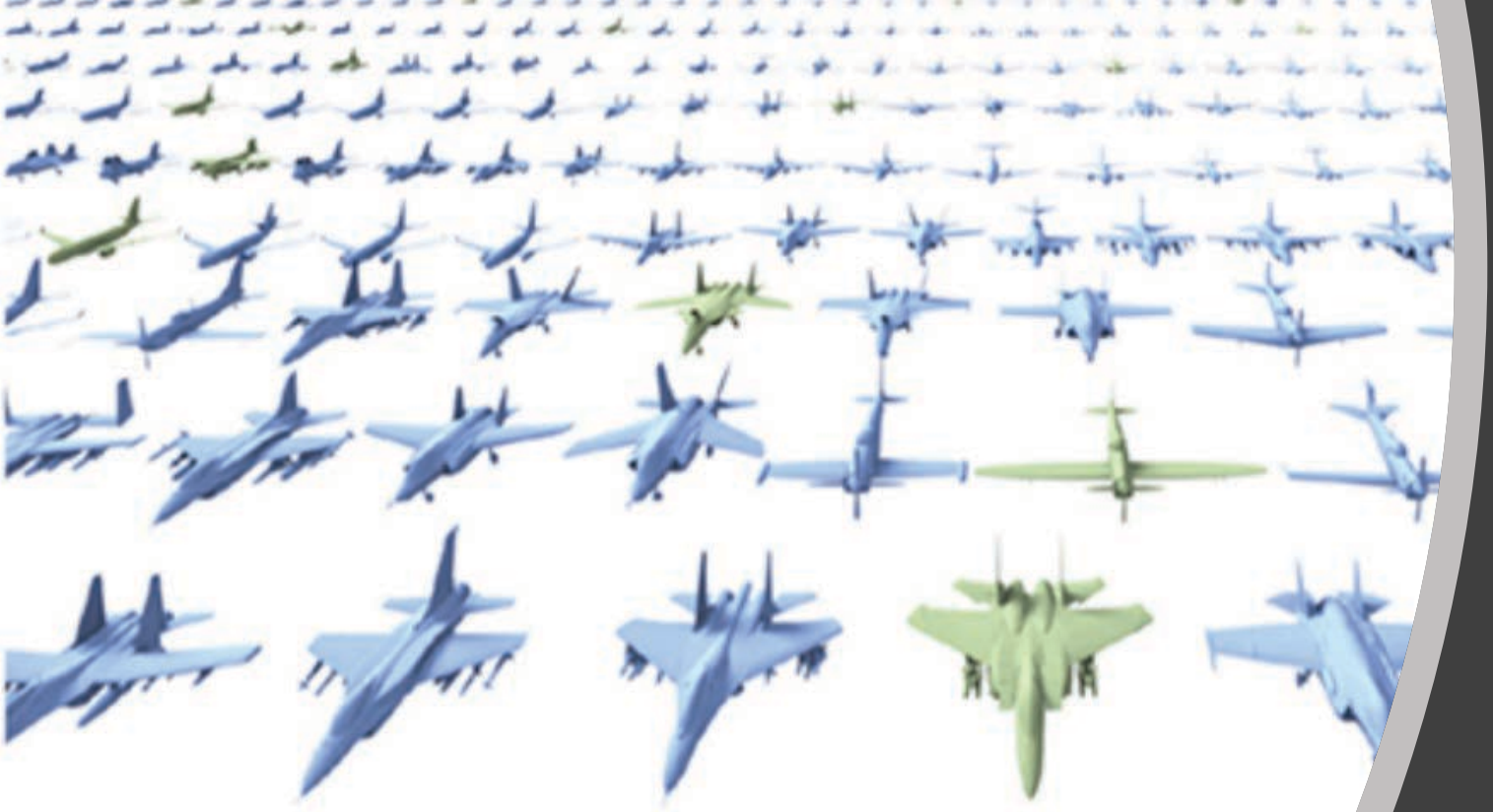
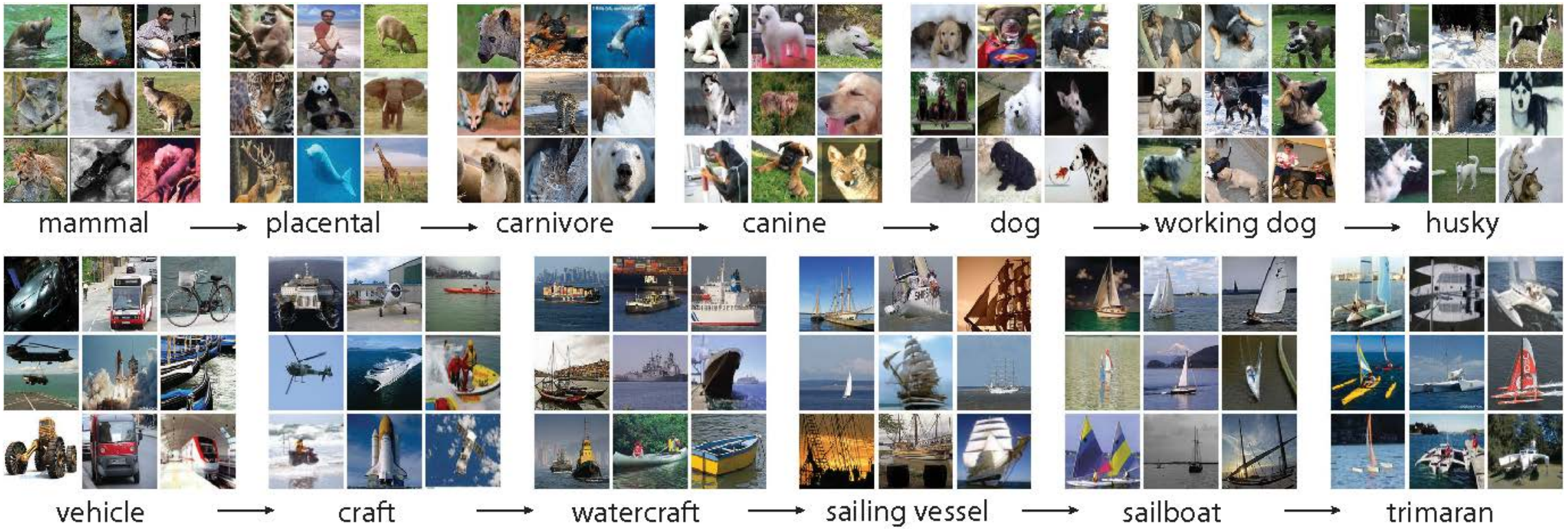


Learning to Generate 3D Structures

Siddhartha Chaudhuri

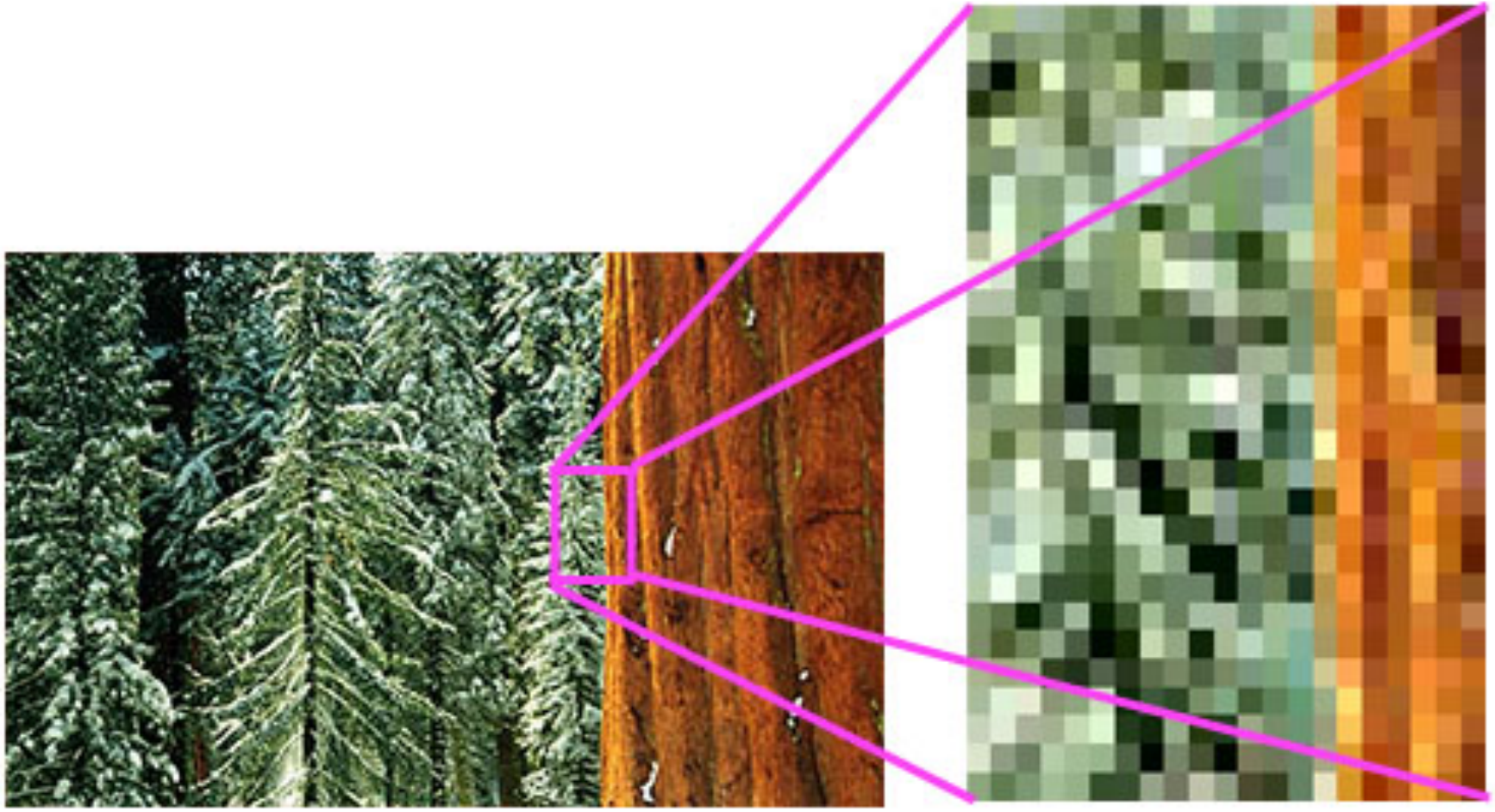
Adobe Research and IIT Bombay

Visual machine learning is traditionally on 2D images

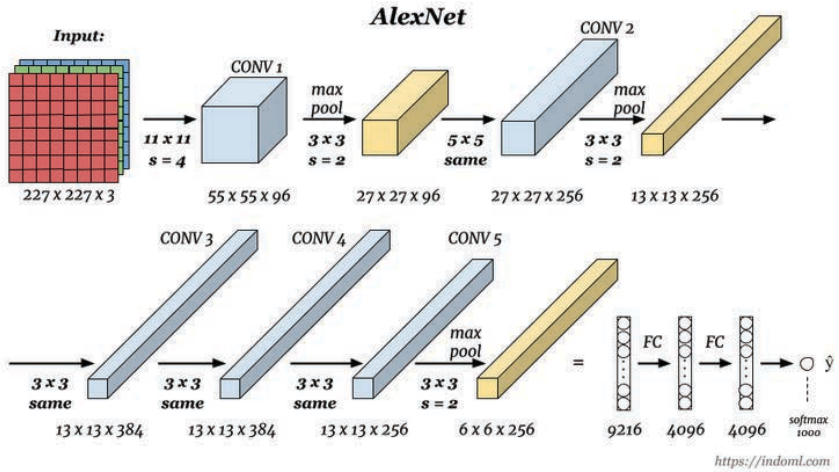


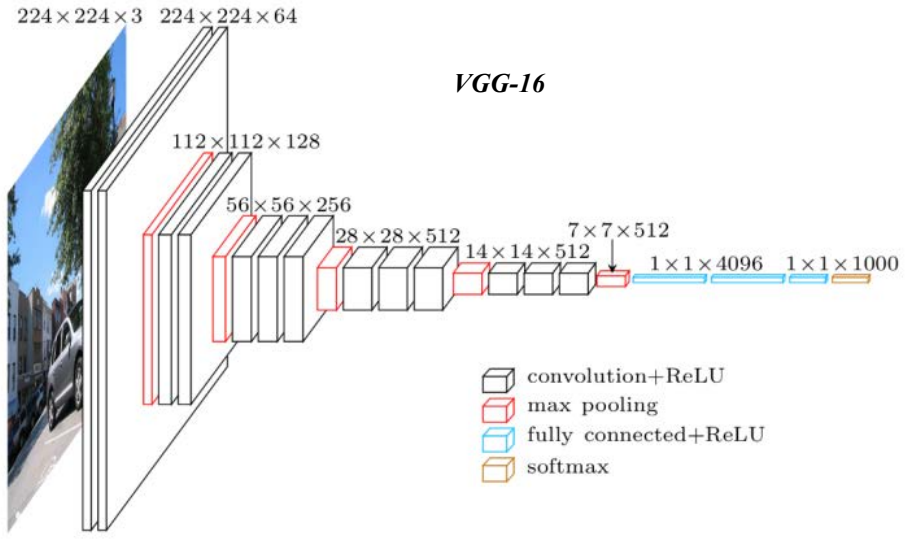
ImageNet, Deng et al. 2009

Images are 2D grids of pixels



2D deep networks do ***convolution*** on grids





VGG-16

http://blog.heuritech.com

- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

2	3	1
0	5	1
1	0	8

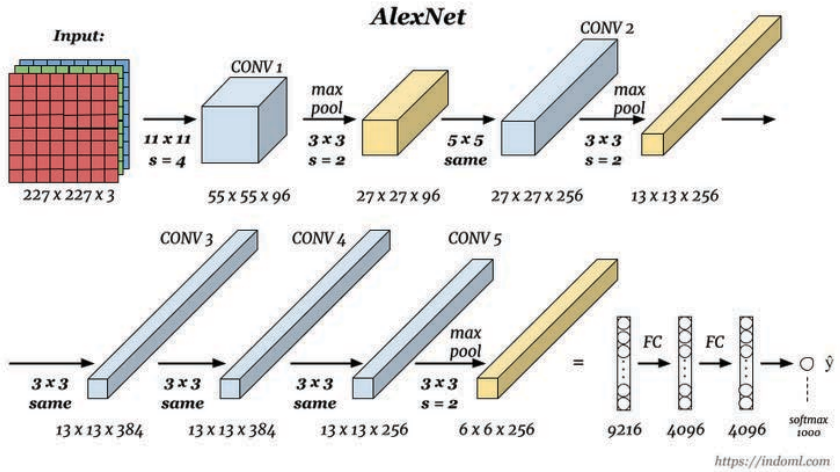
$*$

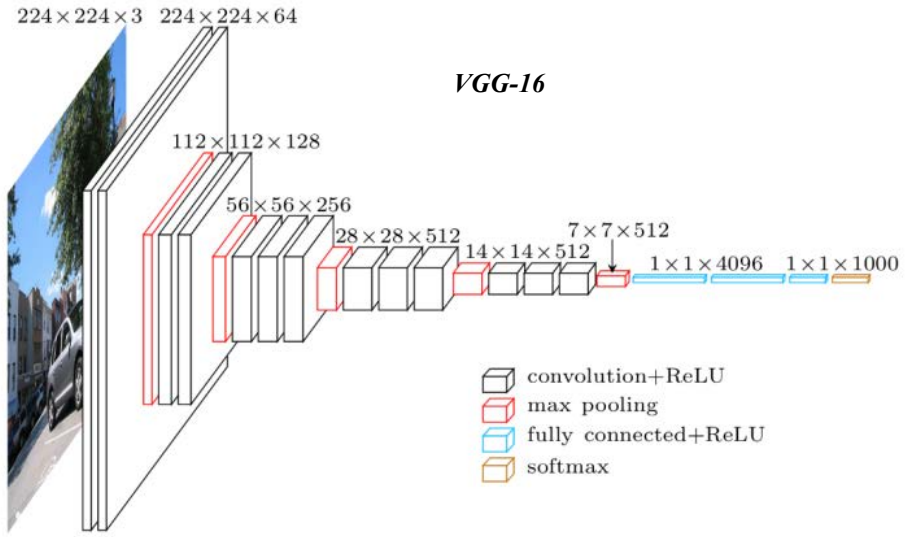
0	-1	0
-1	5	-1
0	-1	0

$= ?$

(we'll assume everything outside the 3x3 is zero)

2D deep networks do ***convolution*** on grids





VGG-16

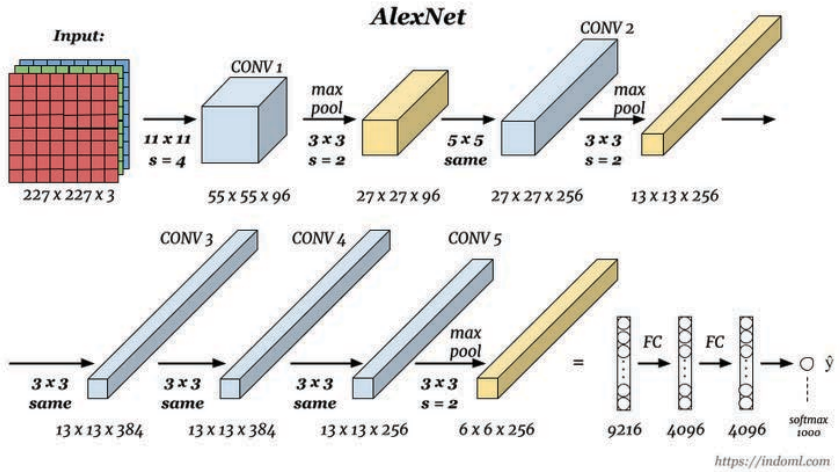
http://blog.heuritech.com

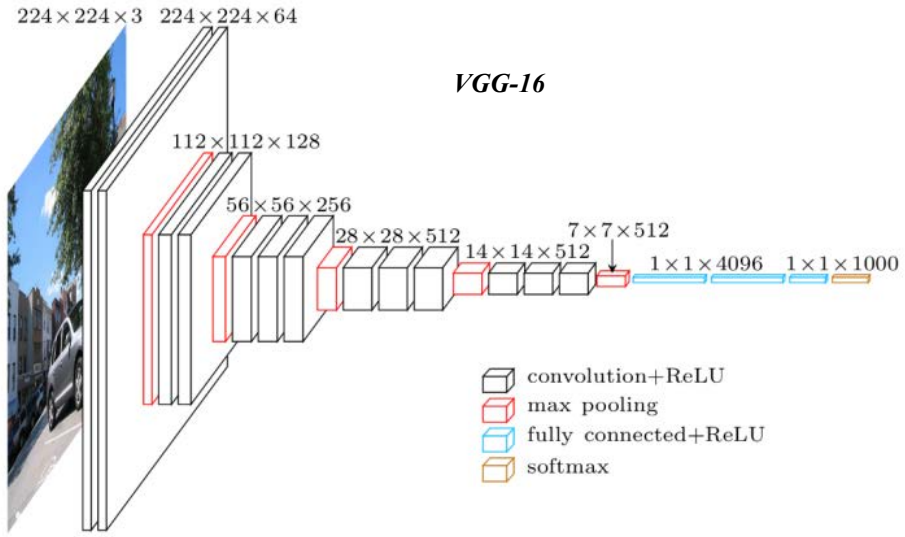
- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

0	-1	0	
-1	25	31	1
0	01	50	1
	1	0	8

7		

2D deep networks do ***convolution*** on grids





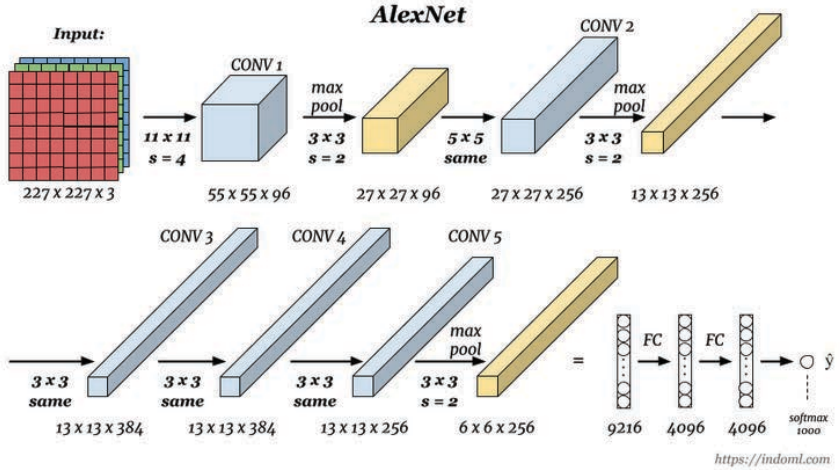
http://blog.heuritech.com

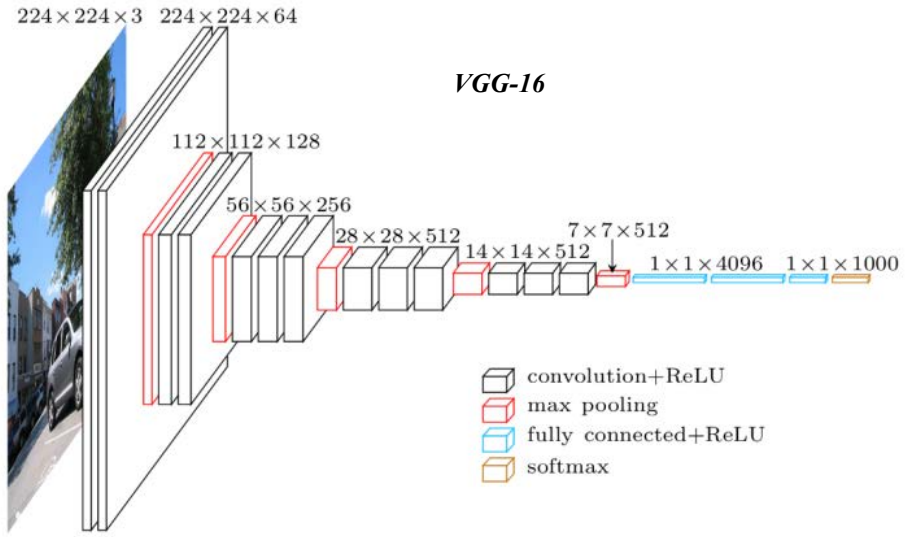
- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

0	-1	0
2 -1	3 5	1 -1
0 0	5 -1	1 0
1	0	8

7	7	

2D deep networks do ***convolution*** on grids





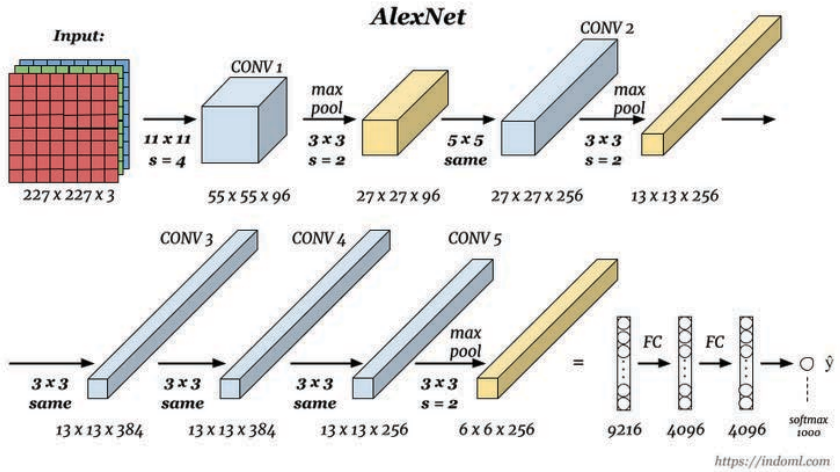
http://blog.heuritech.com

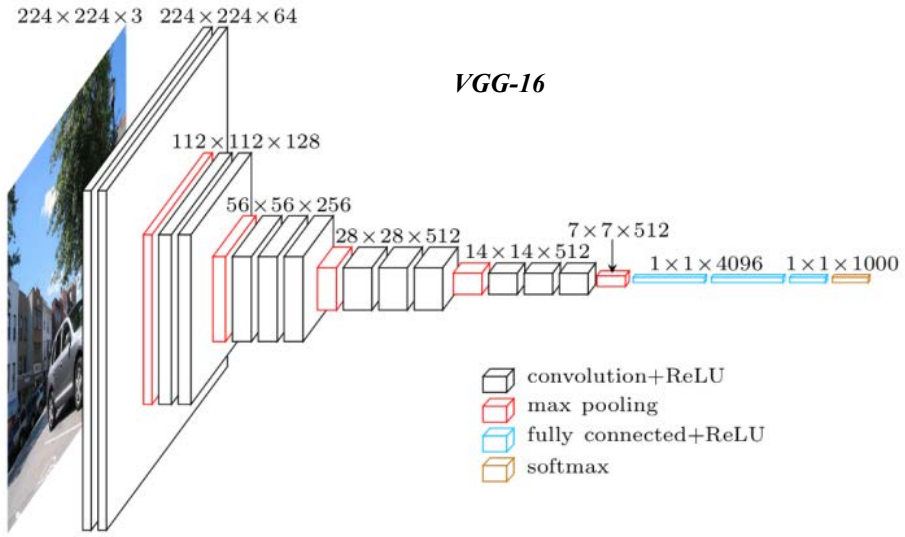
- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

	0	-1	0
2	3 -1	1 5	-1
0	5 0	1 -1	0
1	0	8	

7	7	1

2D deep networks do ***convolution*** on grids



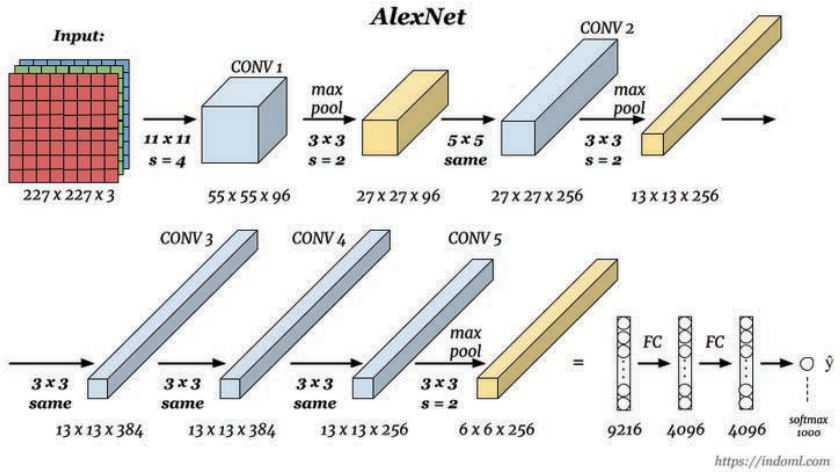


- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

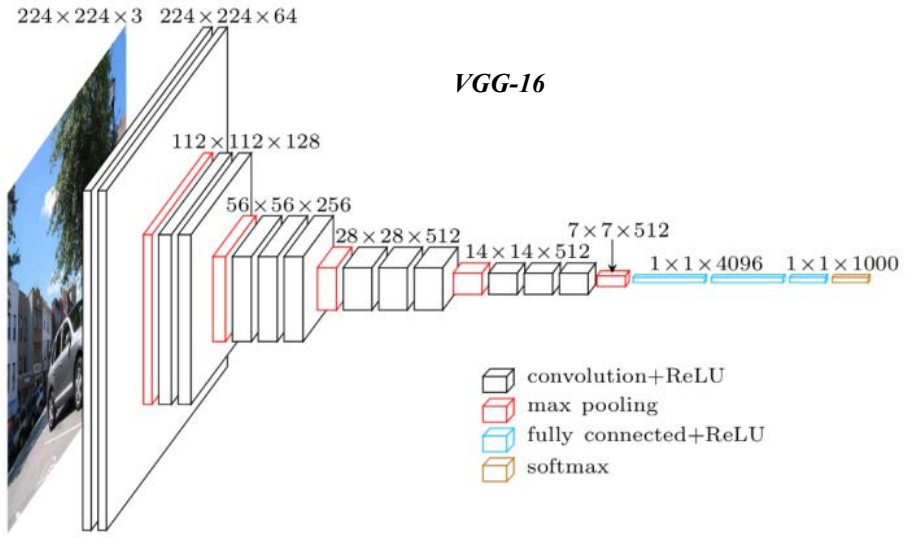
0	2 1	3 0	1
-1	0 5	5 -1	1
0	1 -1	0 0	8

7	7	1
-8		

2D deep networks do ***convolution*** on grids



- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

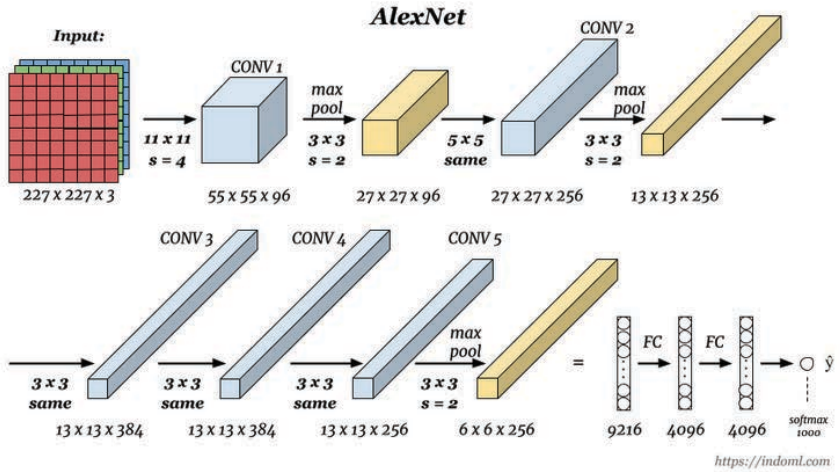


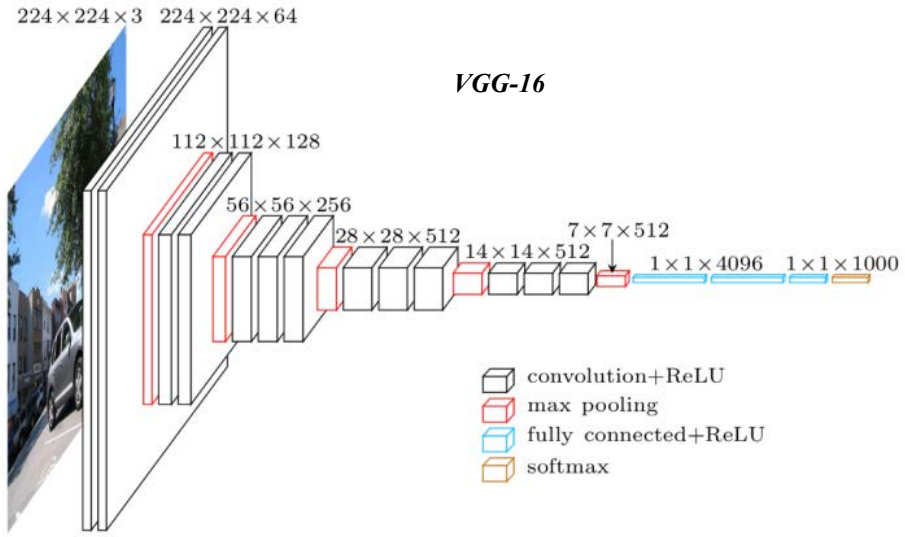
http://blog.heuritech.com

20	31	10
01	55	11
10	01	80

7	7	1
-8	21	

2D deep networks do ***convolution*** on grids





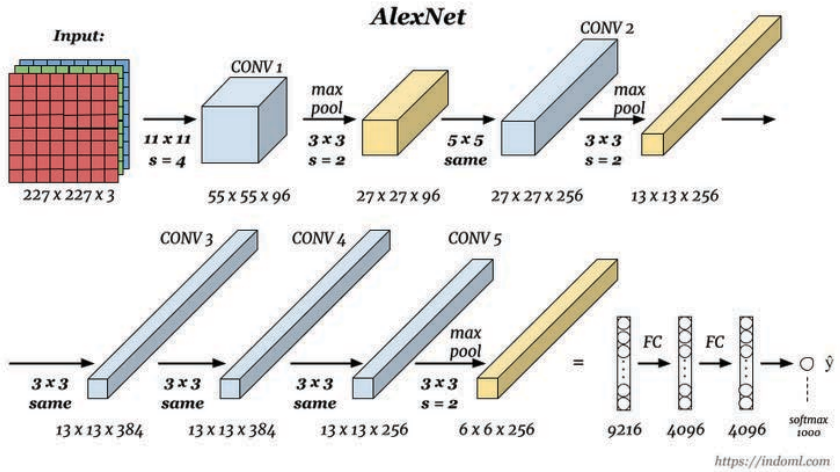
http://blog.heuritech.com

- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

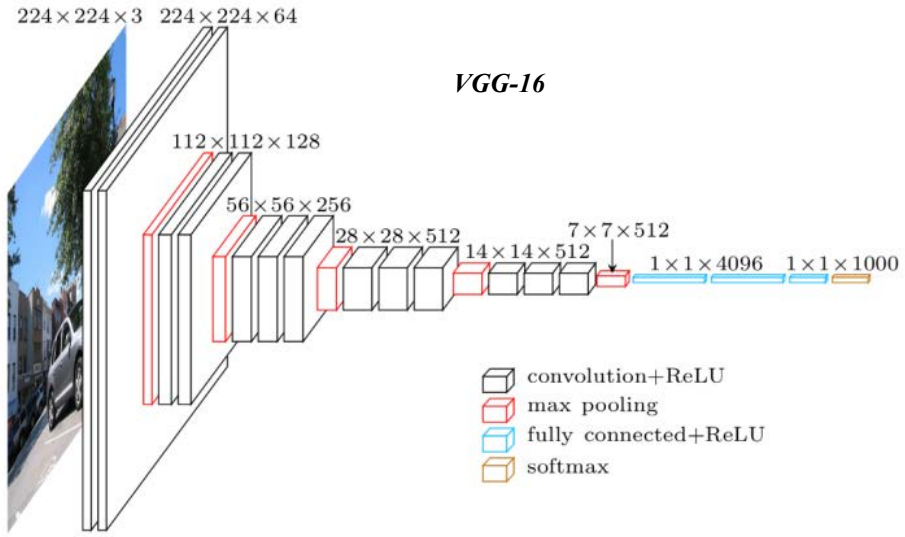
2	3 0	1 -1	0
0	5 -1	1 5	-1
1	0 0	8 -1	0

7	7	1
-8	21	-9

2D deep networks do ***convolution*** on grids



- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

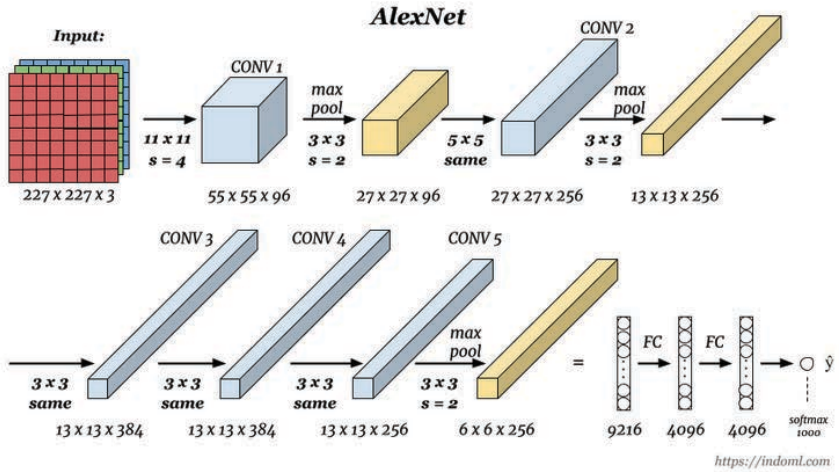


http://blog.heuritech.com

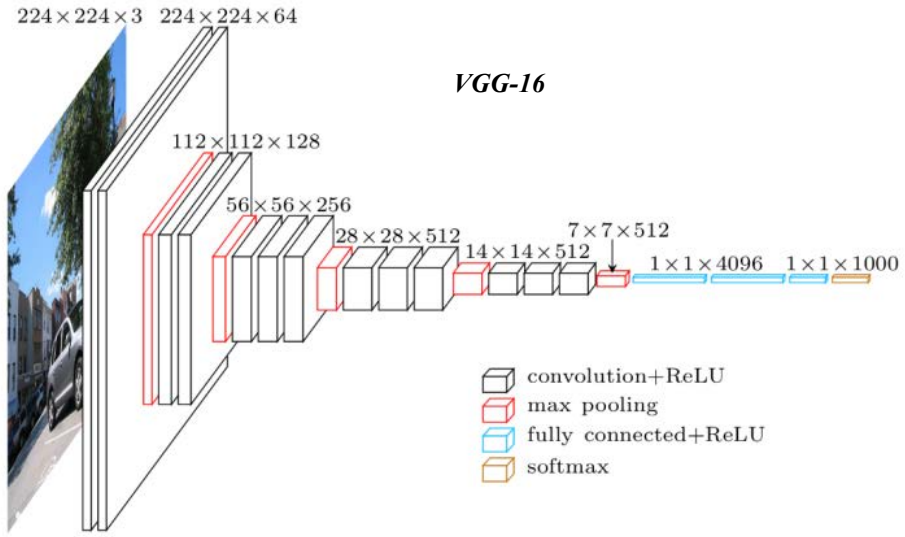
	2	3	1
0	0 1	5 0	1
-1	1 5	0 1	8
0	-1	0	

7	7	1
-8	21	-9
5		

2D deep networks do ***convolution*** on grids



- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

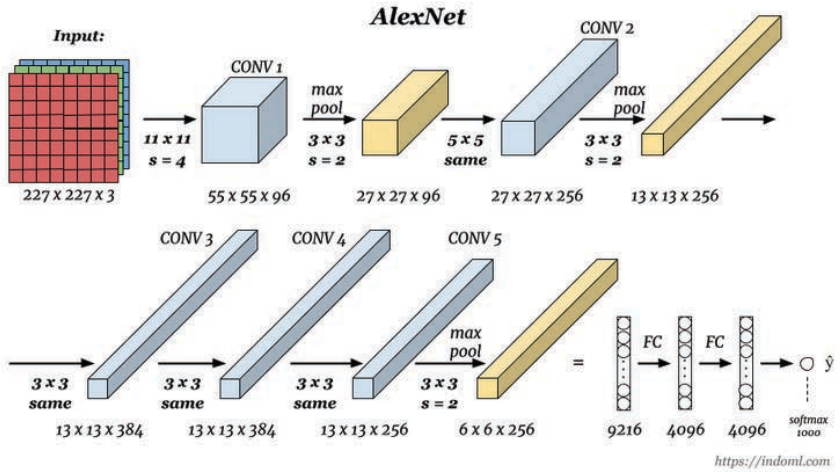


http://blog.heuritech.com

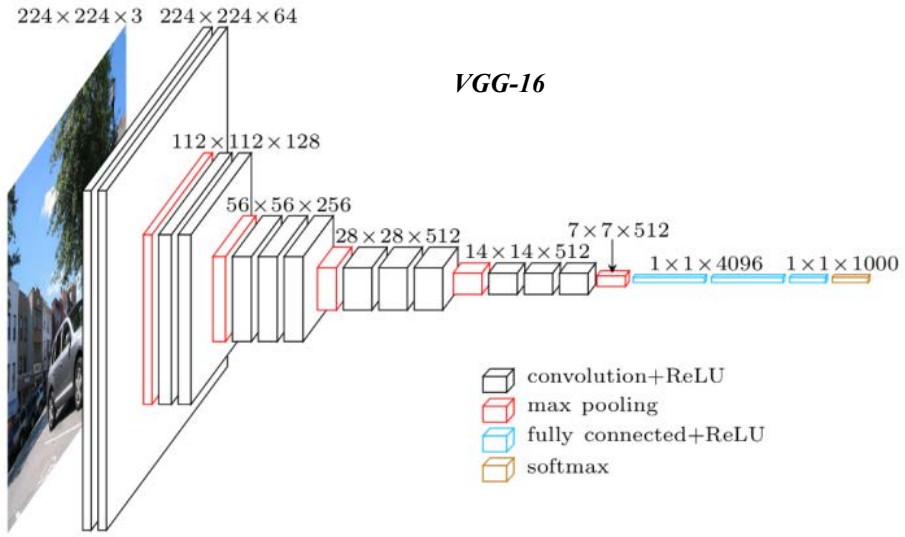
2	3	1
0 0	5 1	1 0
1 -1	0 5	8 -1
0	-1	0

7	7	1
-8	21	-9
5	-14	

2D deep networks do ***convolution*** on grids



- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

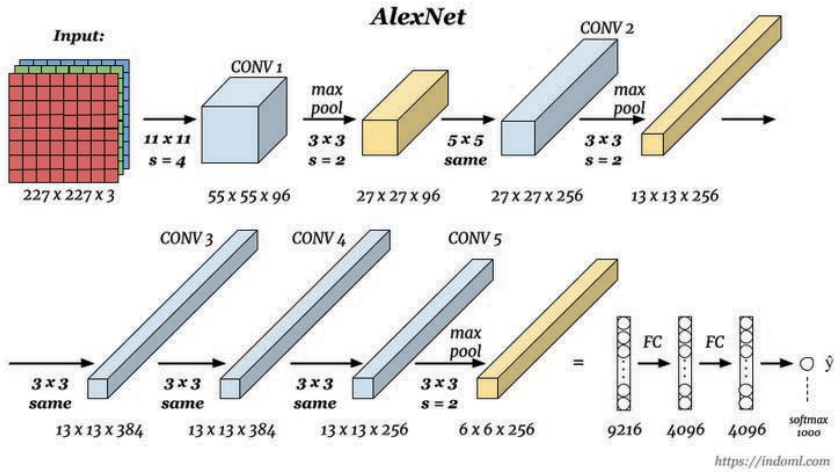


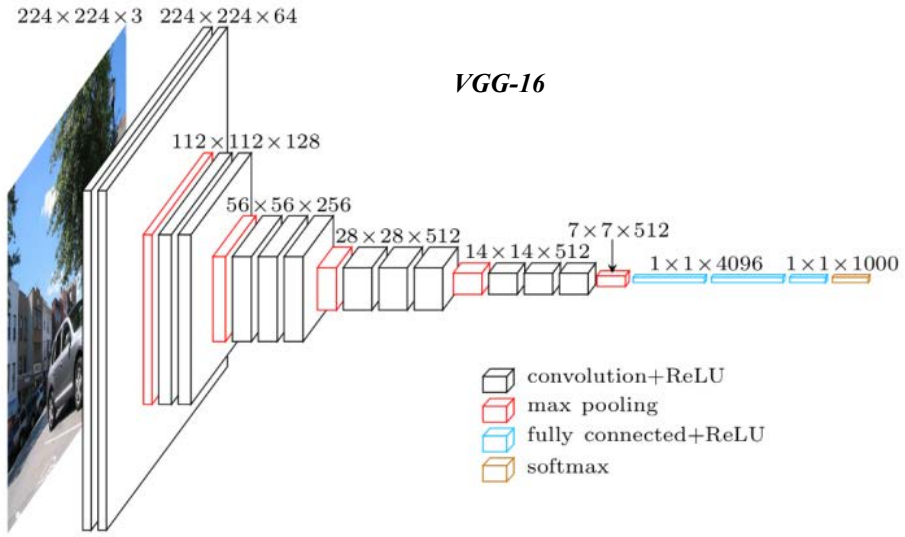
http://blog.heuritech.com

2	3	1	
0	5 0	1 1	0
1	0 1	8 5	-1
	0	-1	0

7	7	1
-8	21	-9
5	-14	39

2D deep networks do ***convolution*** on grids





- Convolution measures "weighted overlap" of *f* with another function *g* as it is (reversed and) shifted over *f*

2	3	1
0	5	1
1	0	8

$*$

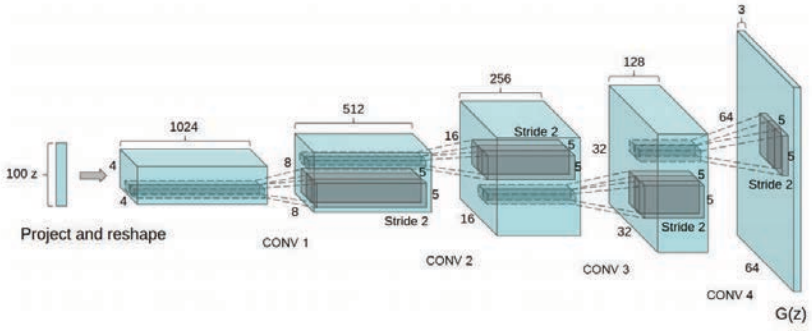
0	-1	0
-1	5	-1
0	-1	0

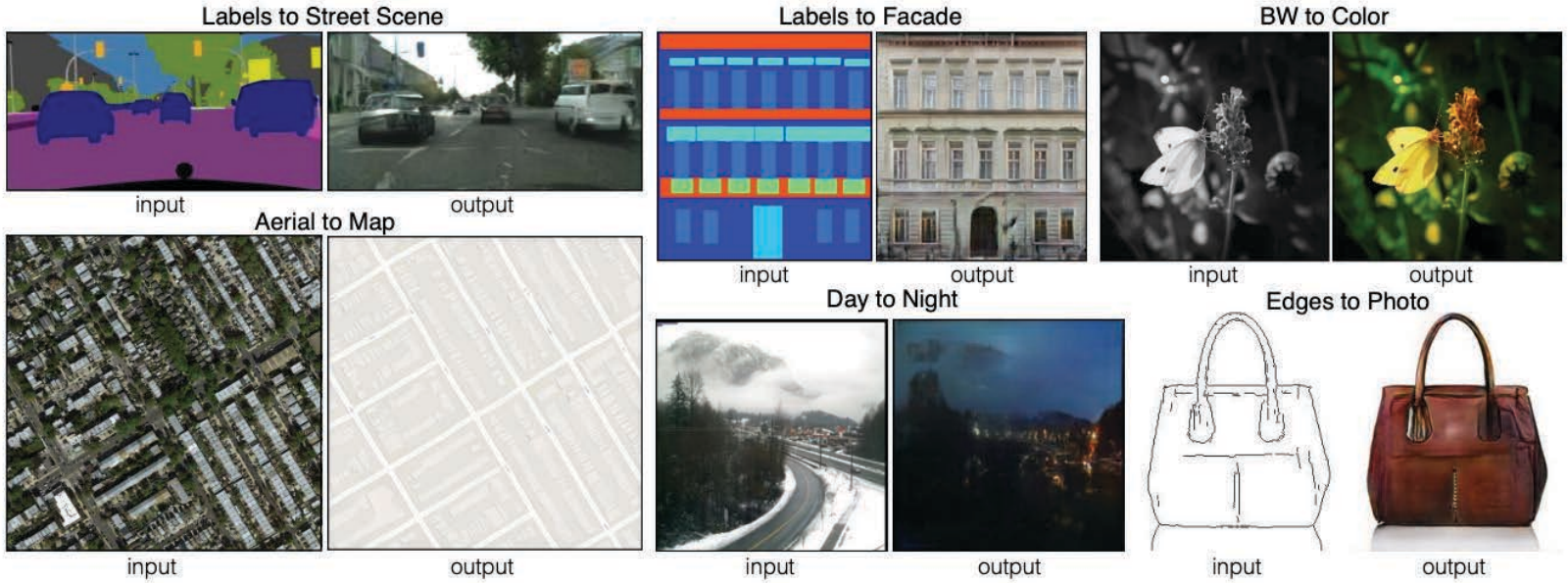
$=$

7	7	1
-8	21	-9
5	-14	39

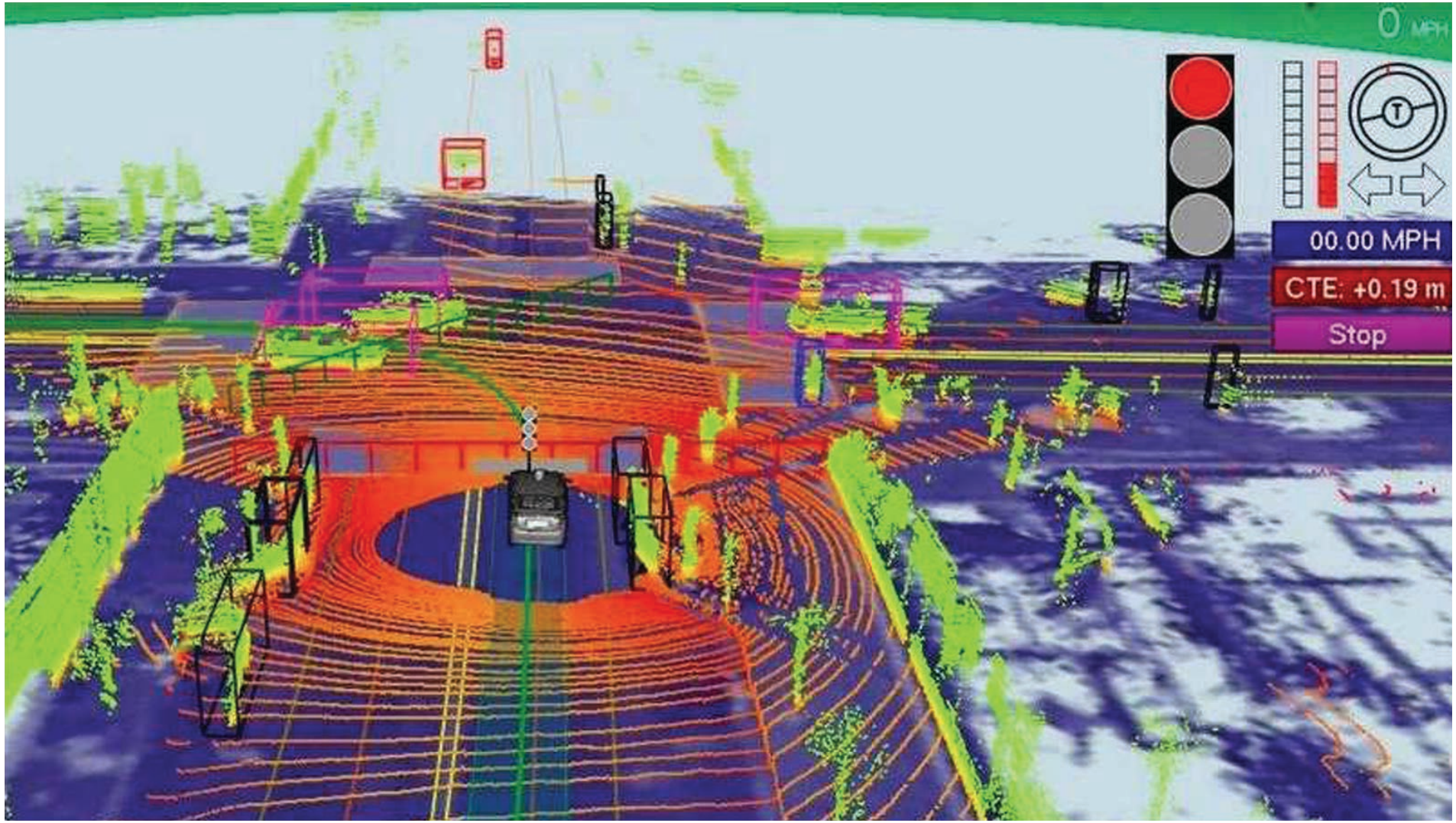
Radford et al. 2015, Mirza et al. 2014, Isola et al. 2016, Karras et al. 2018, Brock et al. 2018

2D generative models use 2D convolution





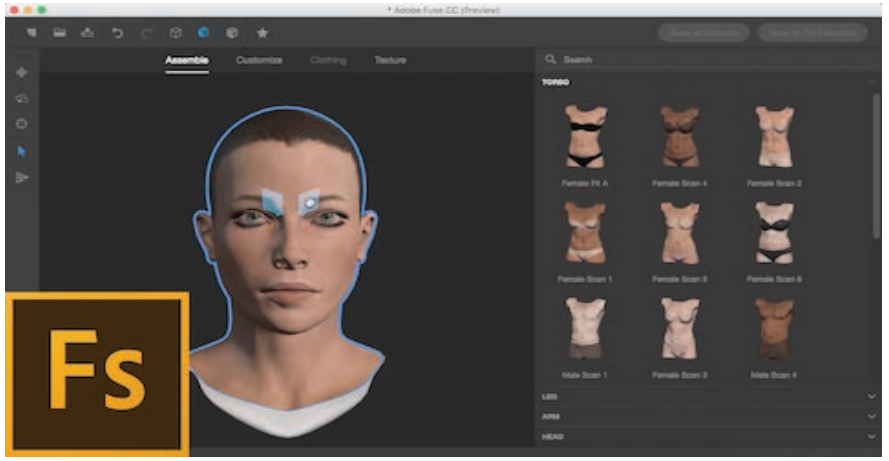
But the real world is (at least) 3D



Microsoft; Adobe; Schumacher et al. 2015; IKEA

Synthesizing 3D shapes is widely important

CAD/CAM



3D modeling for AR/VR/entertainment

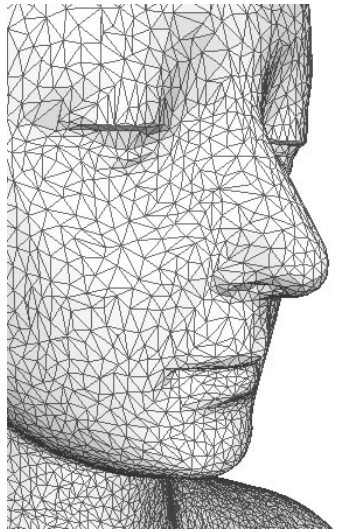
Fabrication

Architecture and interior design

Many ways to represent 3D shapes

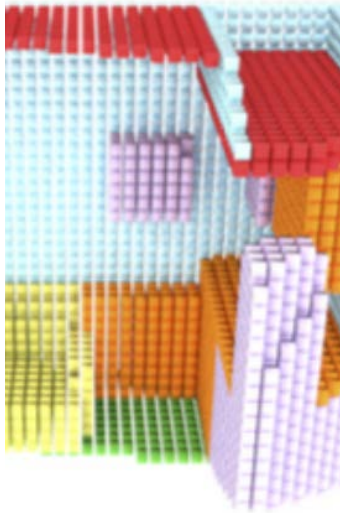
Point cloud

✗



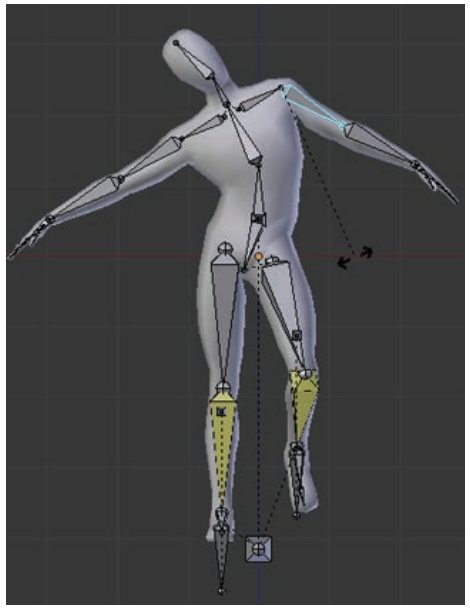
Polygon mesh

✗



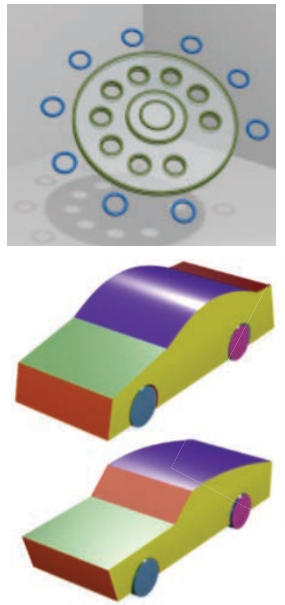
Voxel grid

✓



Rigid skeleton

✗



Wires and patches

✗

Part and object assembly

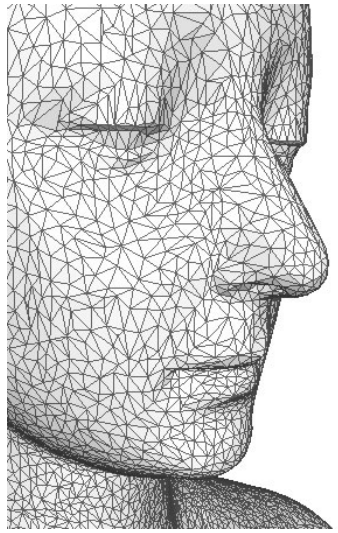
✗

And 2D convolution doesn't extend directly to most of them ☹

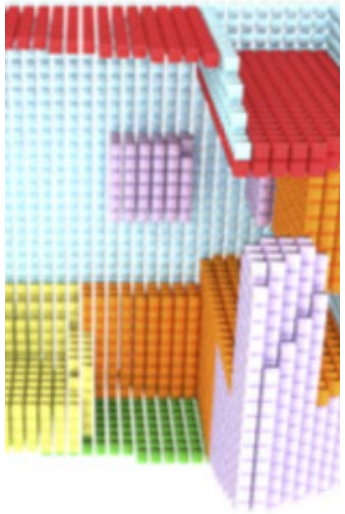
(no grid structure with common parametrization)

Some representations are *structure-aware*

Point cloud



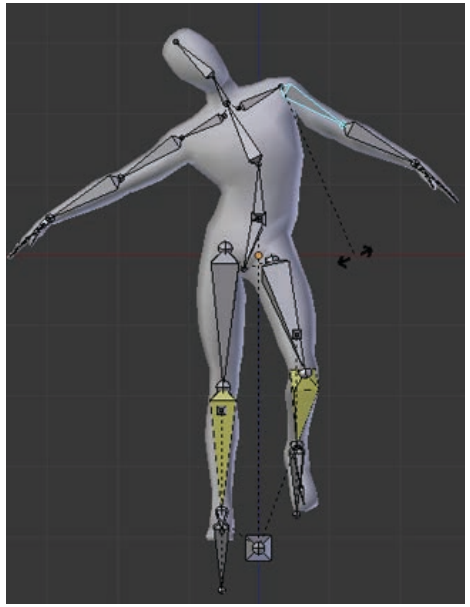
Polygon mesh



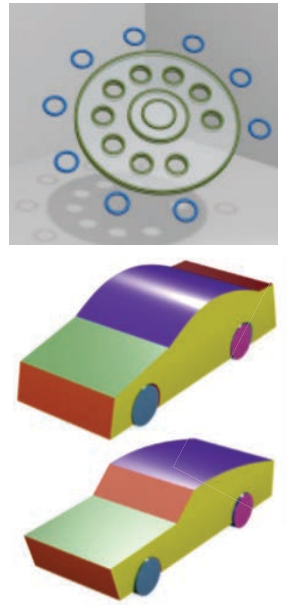
Voxel grid

Not

structure-aware



Rigged skeleton



Wires and patches

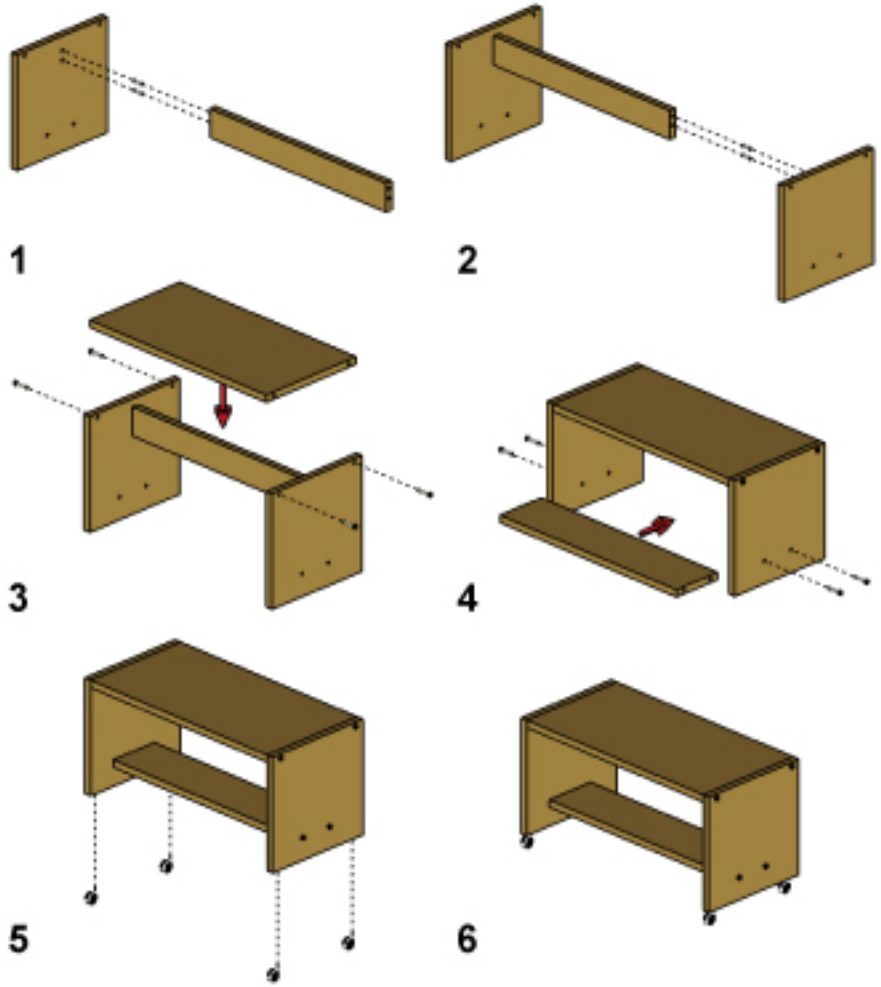
Part and object assembly

Structure-aware

Agrawala et al. 2003

Shape structure: *what* and *why*?

- ***What***: Low-dimensional semantic/functional abstraction (parts, layouts, deformations...)

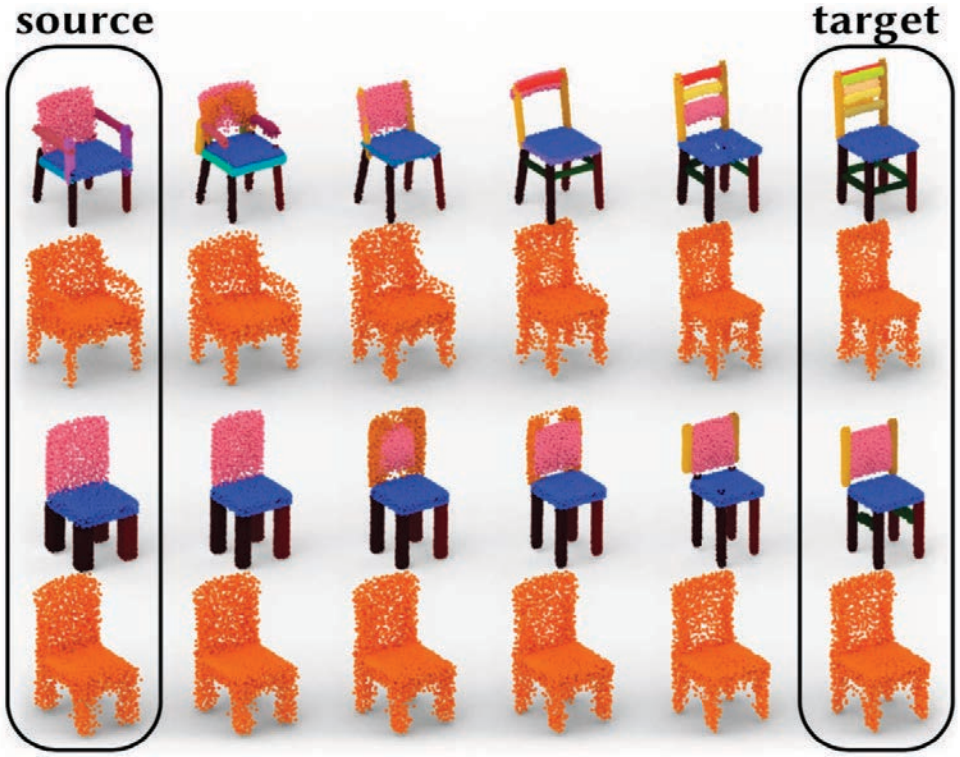


Mo et al. 2019

Shape structure: *what* and *why*?

- ***What***: Low-dimensional semantic/functional abstraction (parts, layouts, deformations...)

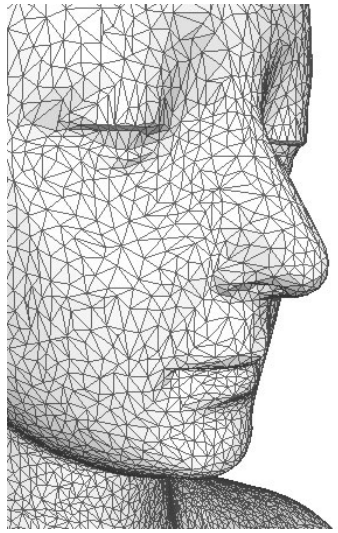
- ***Why*** is structure useful for shape generation?
  - Low-dimensional: less training data, more captured variability
  - Explicit factorization assists learning, model architectures can be decoupled
  - High-quality output: e.g. synthesize novel structure, reuse fine-grained part geometry
  - Output pre-rigged for high-level modeling



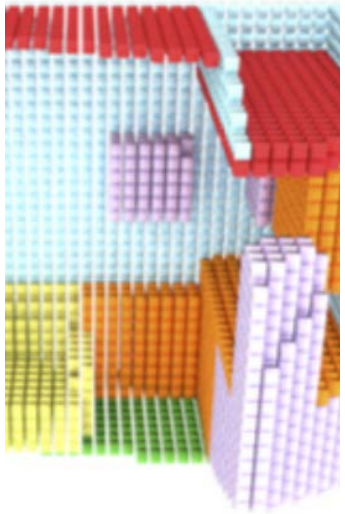
VAE latent space interpolation, **with** and **without** structure-aware model

Many ways to generate 3D shapes

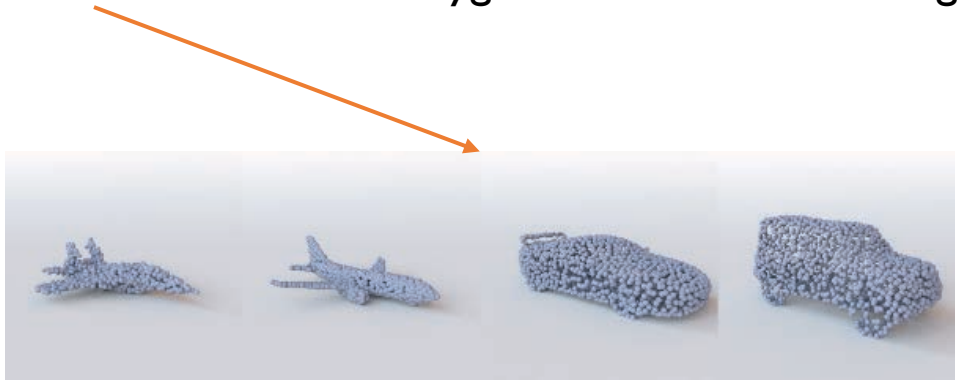
Point cloud



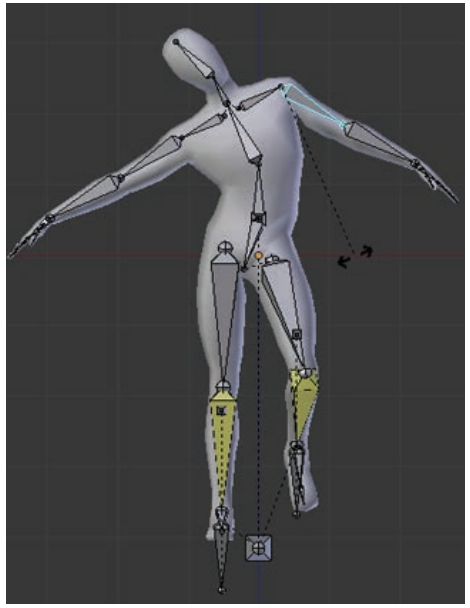
Polygon mesh



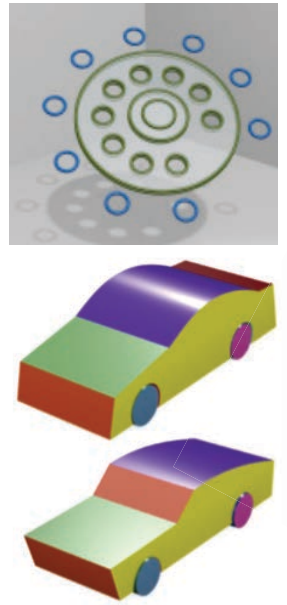
Voxel grid



Achlioptas et al. 2017



Rigged skeleton

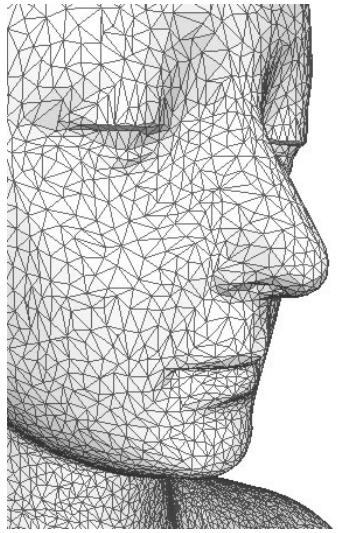


Wires and patches

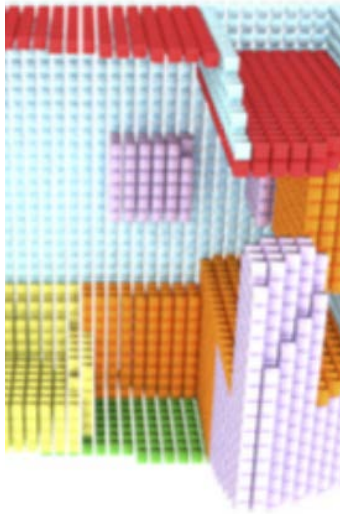
Part and object assembly

Many ways to generate 3D shapes

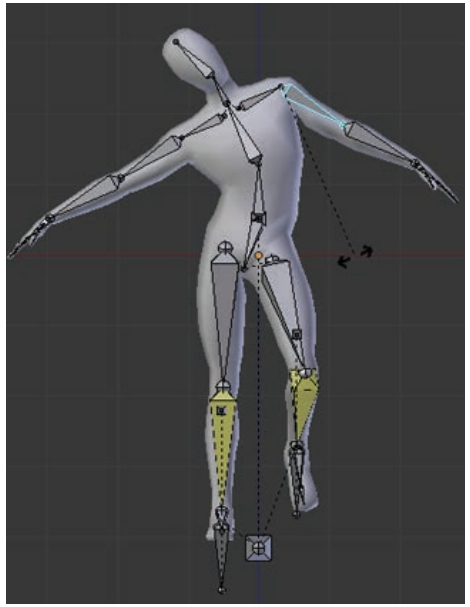
Point cloud



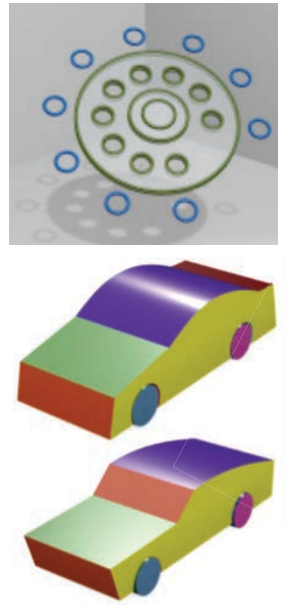
Polygon mesh



Voxel grid

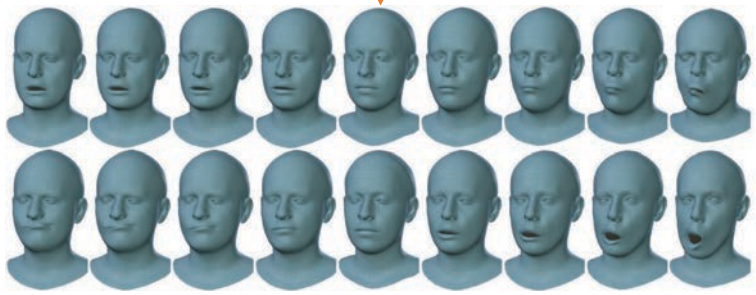


Rigged skeleton



Wires and patches

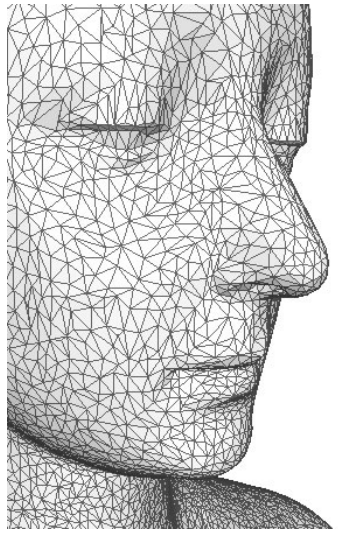
Part and object assembly



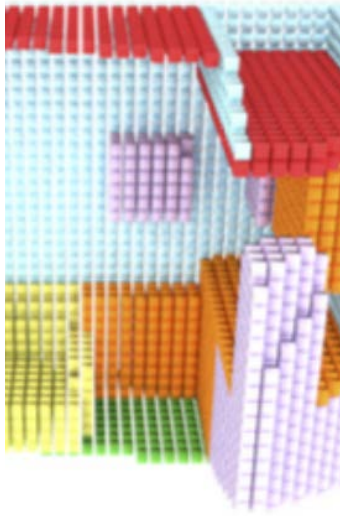
Ranjan et al. 2018

Many ways to generate 3D shapes

Point cloud

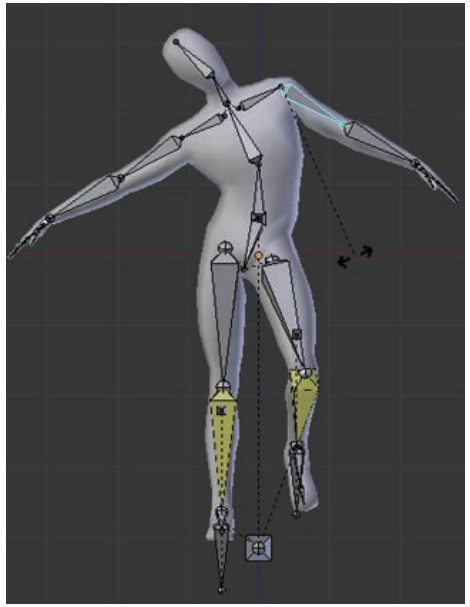


Polygon mesh

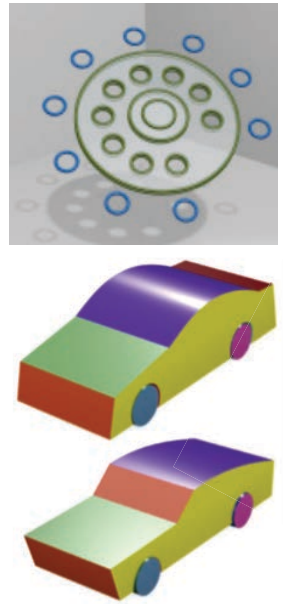


Voxel grid

Wu et al. 2015



Rigged skeleton



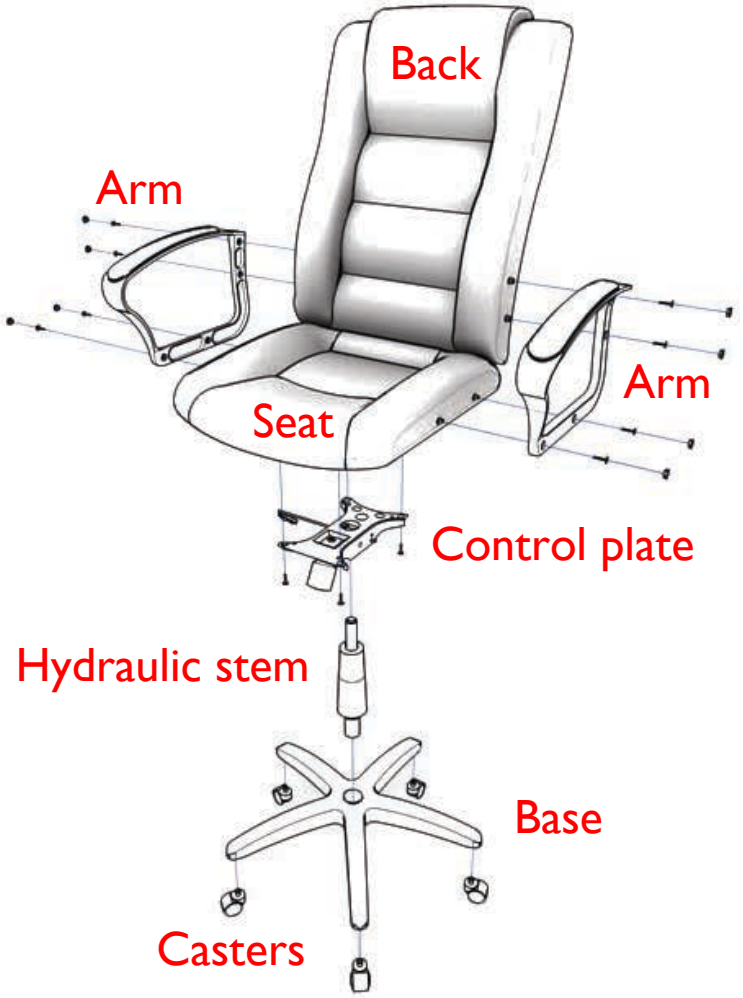
Wires and patches

Part and object assembly

Focus of this talk

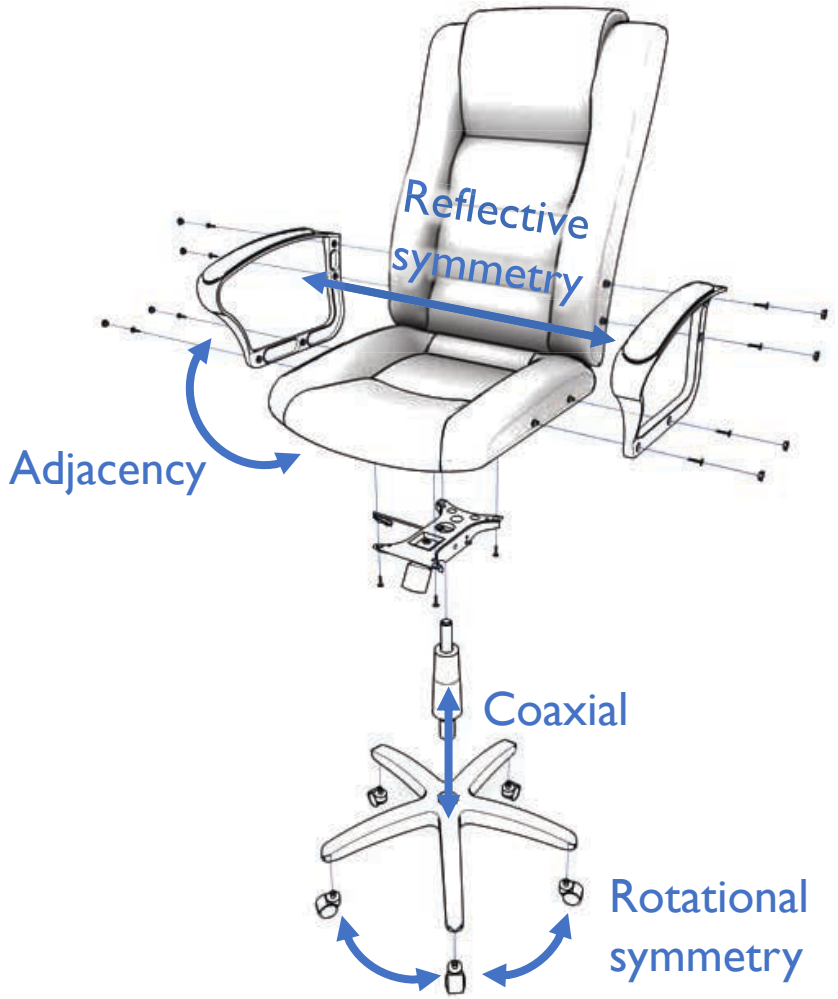
Structure is commonly described by (hyper-)graphs

- Parts *(vertices)*
  - Discrete and continuous variation



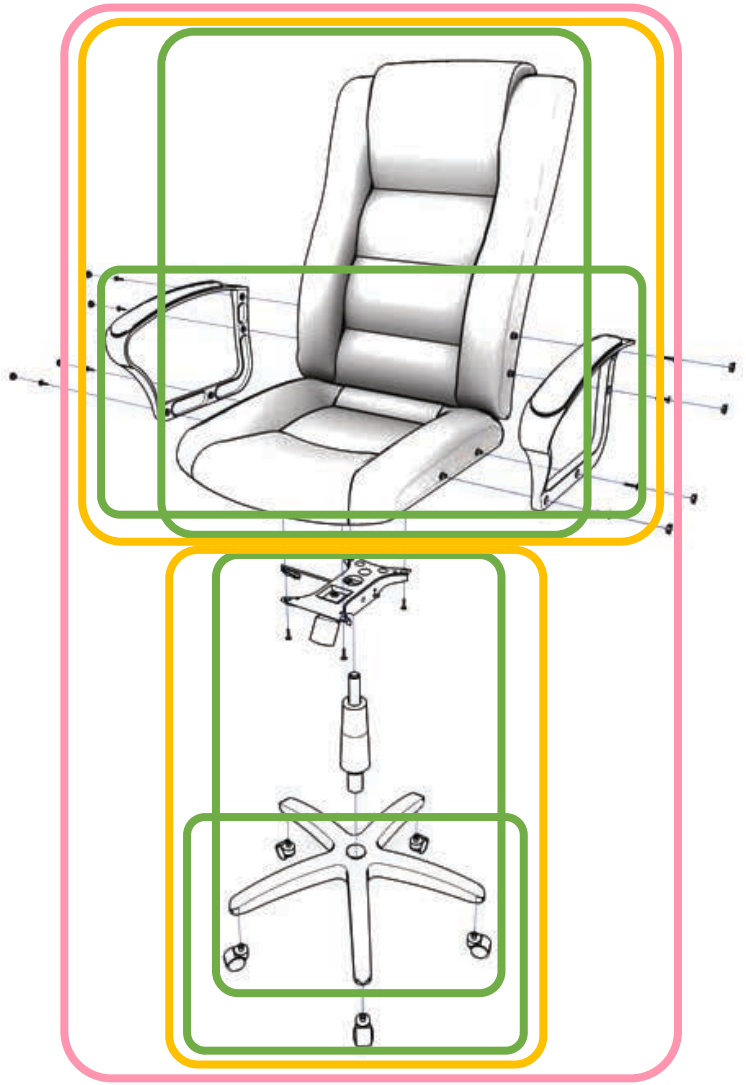
Structure is commonly described by (hyper-)graphs

- Parts *(vertices)*
  - Discrete and continuous variation
- Layout relationships *(edges)*



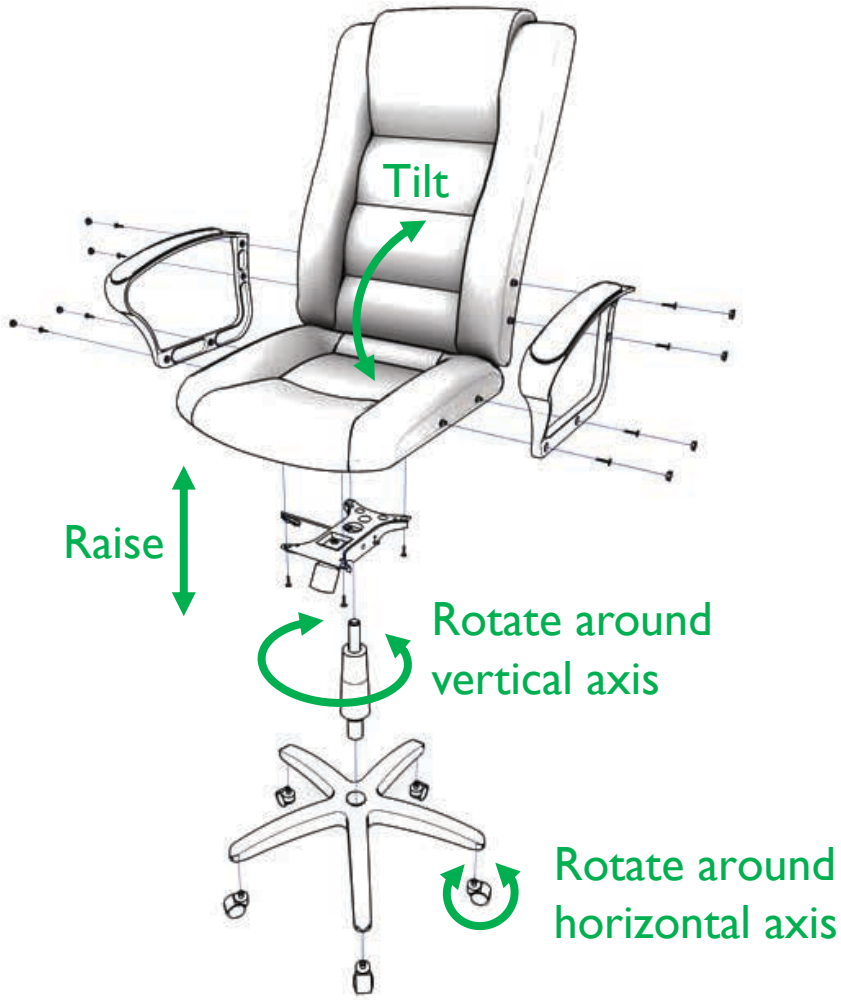
Structure is commonly described by (hyper-)graphs

- Parts *(vertices)*
  - Discrete and continuous variation
- Layout relationships *(edges)*
  - Hierarchical



Structure is commonly described by (hyper-)graphs

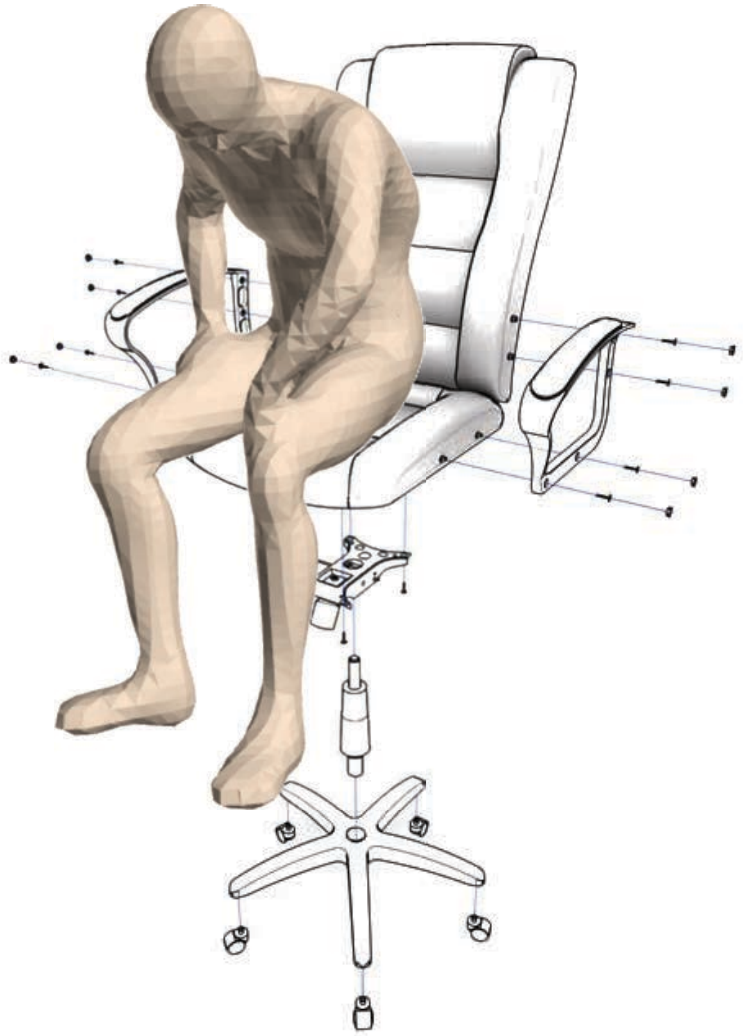
- Parts *(vertices)*
  - Discrete and continuous variation
- Layout relationships *(edges)*
  - Hierarchical
- Motion



Structure is commonly described by (hyper-)graphs

- Parts *(vertices)*
  - Discrete and continuous variation
- Layout relationships *(edges)*
  - Hierarchical
- Motion
- Function

} *attributes*



Structure is commonly described by (hyper-)graphs

- Parts *(vertices)*
  - Discrete and continuous variation
- Layout relationships *(edges)*
  - Hierarchical
- Motion
- Function

} *attributes*

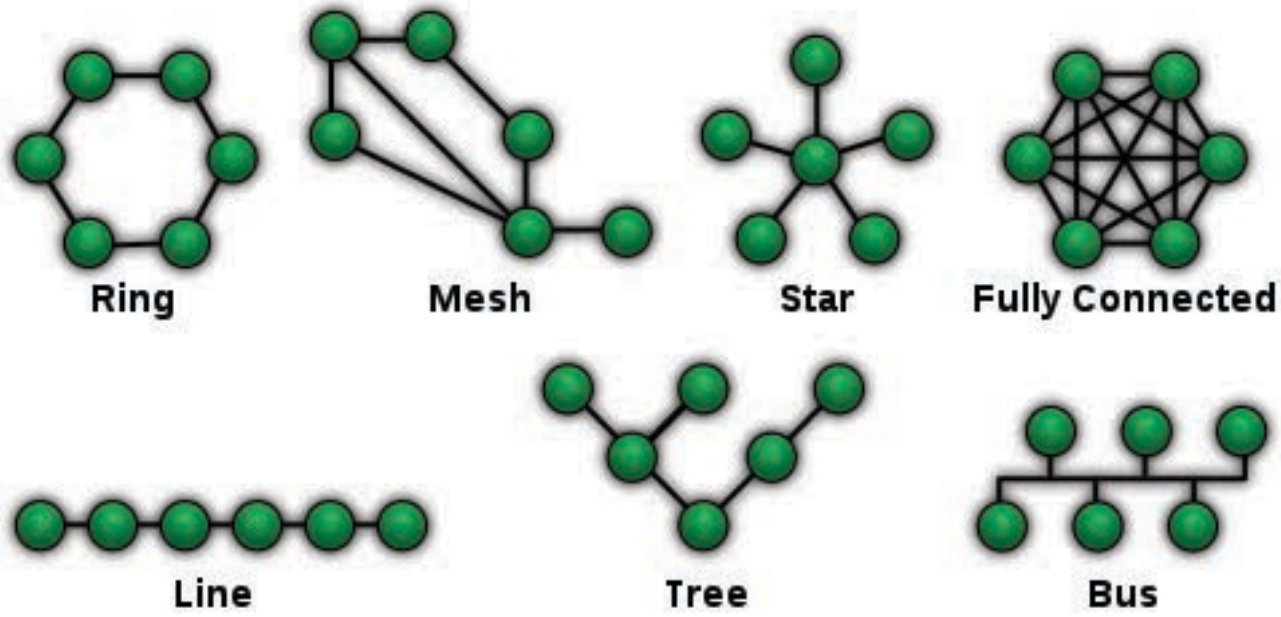
- Text and other annotations

The web of interdependencies is typically too complex to specify by hand. Hence, it is **learnt** from a repository of exemplar structures

Learning generative models of **shape structure**

$\Leftrightarrow$

Learning generative models over **spaces of graphs**



- Each point in the space is a graph
- Can't assume fixed topology

What is a **generative** model?

- Let $\mathbf{x}$ be observed data (e.g. shape) and $\mathbf{y}$ be target data (e.g. label)
- The distribution $P(\mathbf{y} \mid \mathbf{x})$ is a ***discriminative* model**
  - No way to sample $\mathbf{x}$ itself
  - Great for vision applications where we need to predict class etc ($\mathbf{y}$) for an observed object ($\mathbf{x}$)

- The distribution $P(\mathbf{x}, \mathbf{y})$, or just $P(\mathbf{x})$, is a ***generative* model**
  - Samplers like MCMC or GAN/VAE can sample $\mathbf{x}$ according to the distribution
  - Great for graphics applications where we need to **synthesize new objects**!
  - $P(\mathbf{x} \mid \mathbf{y})$ is a *conditional* generative model, e.g. $\mathbf{y}$ is a sketch and $\mathbf{x}$ is the corresponding 3D shape

Focus of this talk!

Probabilistic graphical model

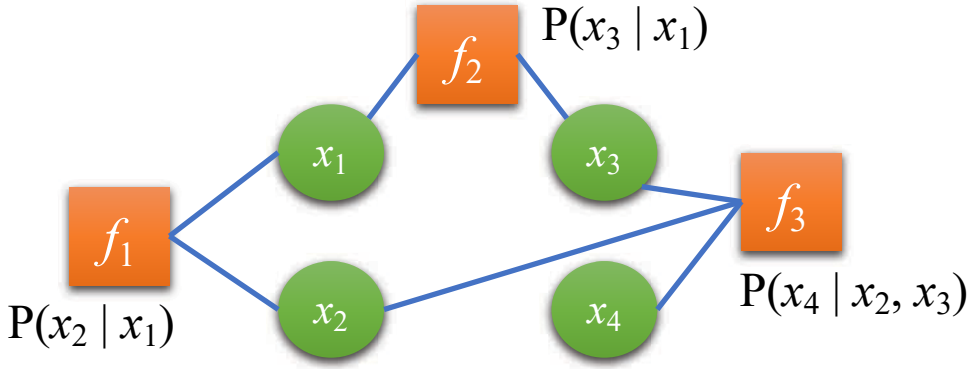
Factor graph, Bayesian network, Markov random field…

- If $\mathbf{x}$ is high-dimensional, $\mathrm{P}(\mathbf{x})$ can be very complicated
- Often, it can be written as a product of smaller, independent **factors**

$$\mathrm{P}(\mathbf{x}) = \mathrm{P}(\mathbf{x}_1 \mid \dot{\mathbf{x}}_1) \times \mathrm{P}(\mathbf{x}_2 \mid \dot{\mathbf{x}}_2) \times \ldots \times \mathrm{P}(\mathbf{x}_k \mid \dot{\mathbf{x}}_k)$$

  where $\mathbf{x}_1 \ldots \mathbf{x}_k$ and (potentially empty) $\dot{\mathbf{x}}_1 \ldots \dot{\mathbf{x}}_k$ are subsets of the variables of $\mathbf{x}$

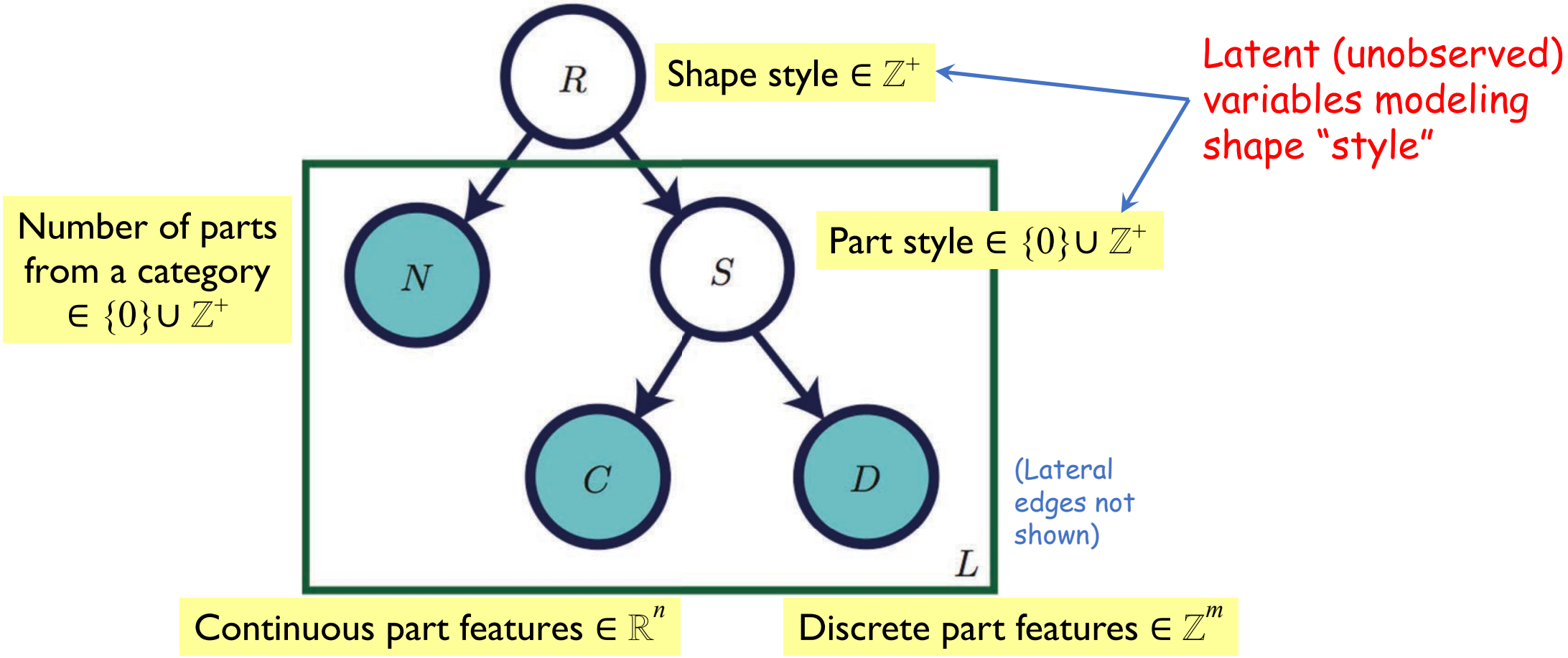
- A **graphical model** visually interprets this factorization as a graph



$$\begin{aligned}\mathrm{P}(\mathbf{x}) = \; & \mathrm{P}(x_4 \mid x_2, x_3) \\ & \times \mathrm{P}(x_3 \mid x_1) \\ & \times \mathrm{P}(x_2 \mid x_1) \\ & \times \mathrm{P}(x_1)\end{aligned}$$

A graphical model for assembling existing parts

Part graph encoded with fixed number of variables, layout is deterministic



$$P(\mathbf{X}) = P(R) \prod_{l \in \mathcal{L}} \left[P(S_l \mid R) P(N_l \mid R, \pi(N_l)) P(\mathbf{C}_l \mid S_l, \pi(\mathbf{C}_l)) P(\mathbf{D}_l \mid S_l, \pi(\mathbf{D}_l)) \right]$$

Sampling the learned model

Kalogerakis, Chaudhuri, Koller and Koltun, 2012

Sampling the learned model

Learned latent shape "styles"

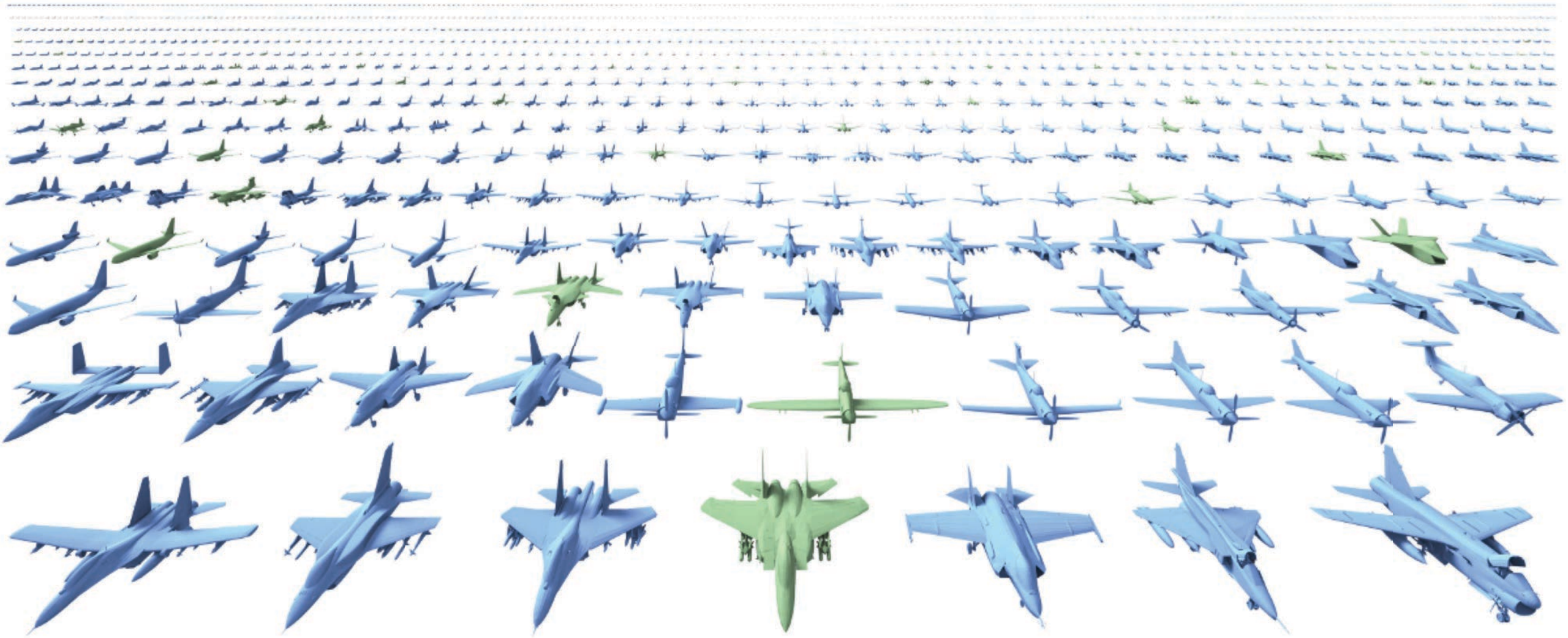
Learned latent component "styles"

Kalogerakis, Chaudhuri, Koller and Koltun, 2012

Sampling the learned model

Kalogerakis, Chaudhuri, Koller and Koltun, 2012

Sampling the learned model

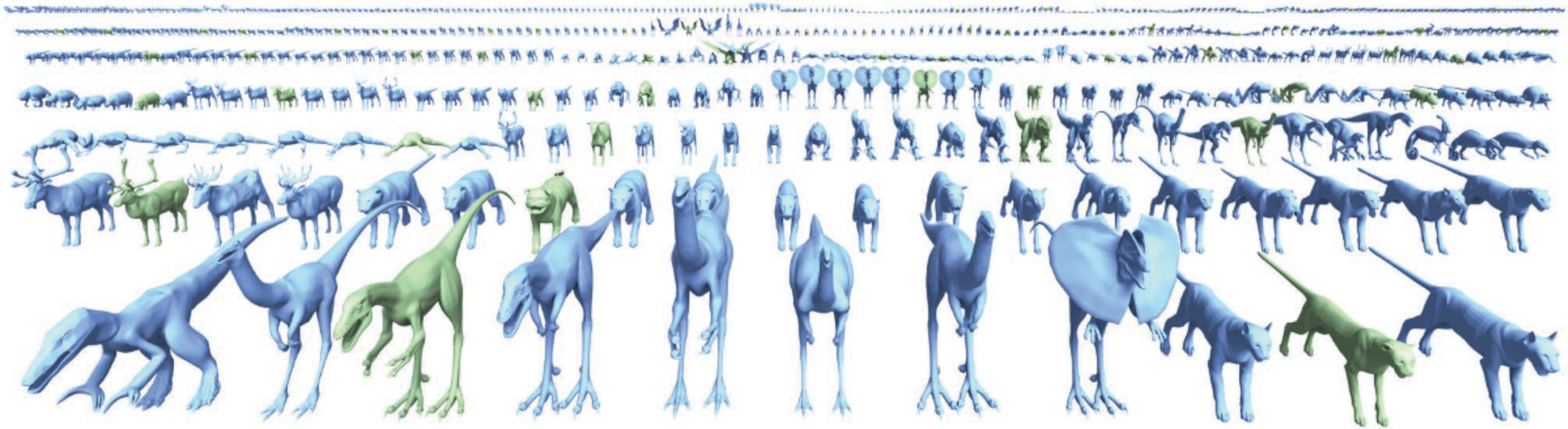


Kalogerakis, Chaudhuri, Koller and Koltun, 2012

Sampling the learned model

Kalogerakis, Chaudhuri, Koller and Koltun, 2012

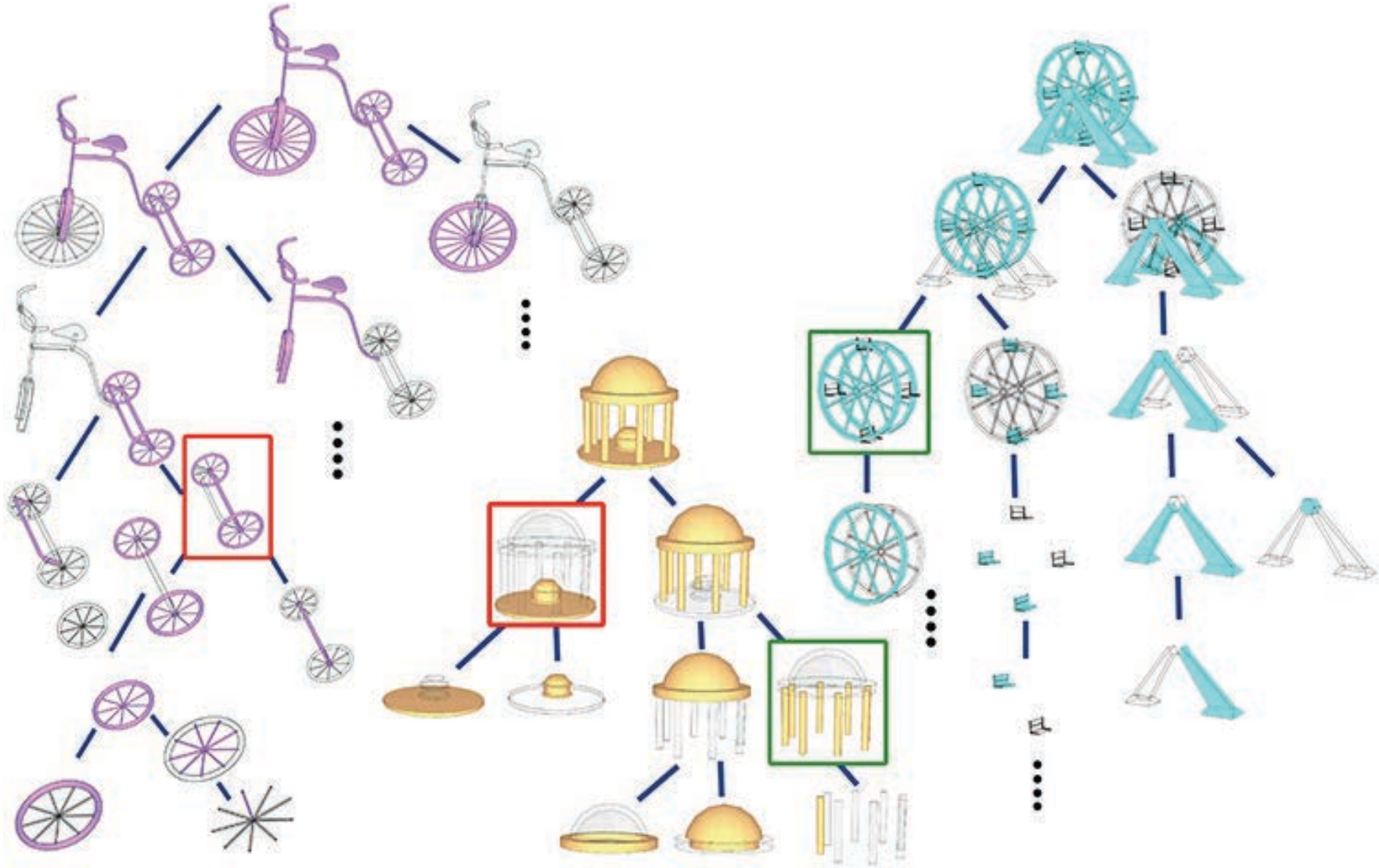
Sampling the learned model



Kalogerakis, Chaudhuri, Koller and Koltun, 2012

Shapes have natural hierarchical structure

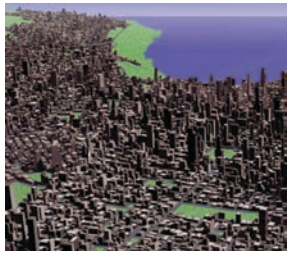
Hierarchical grouping driven by cognitive or functional imperatives



Wang et al., 2011

Probabilistic context-free grammars

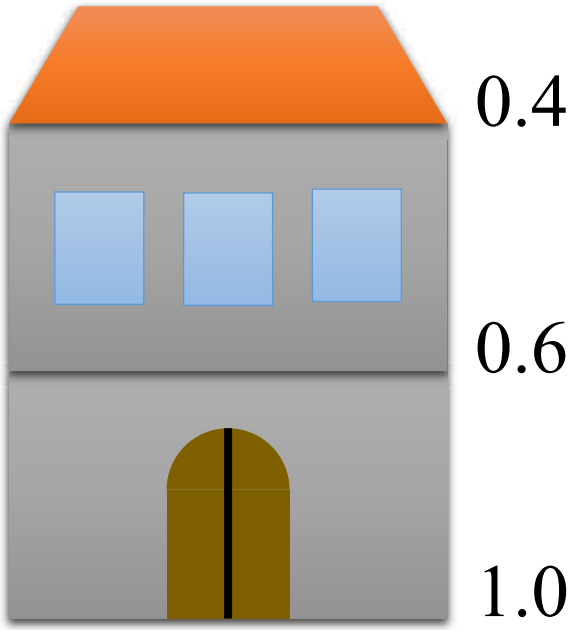
- PCFGs are "dynamic" graphical models
- Set of expansion rules, each with a probability



START : ***G***

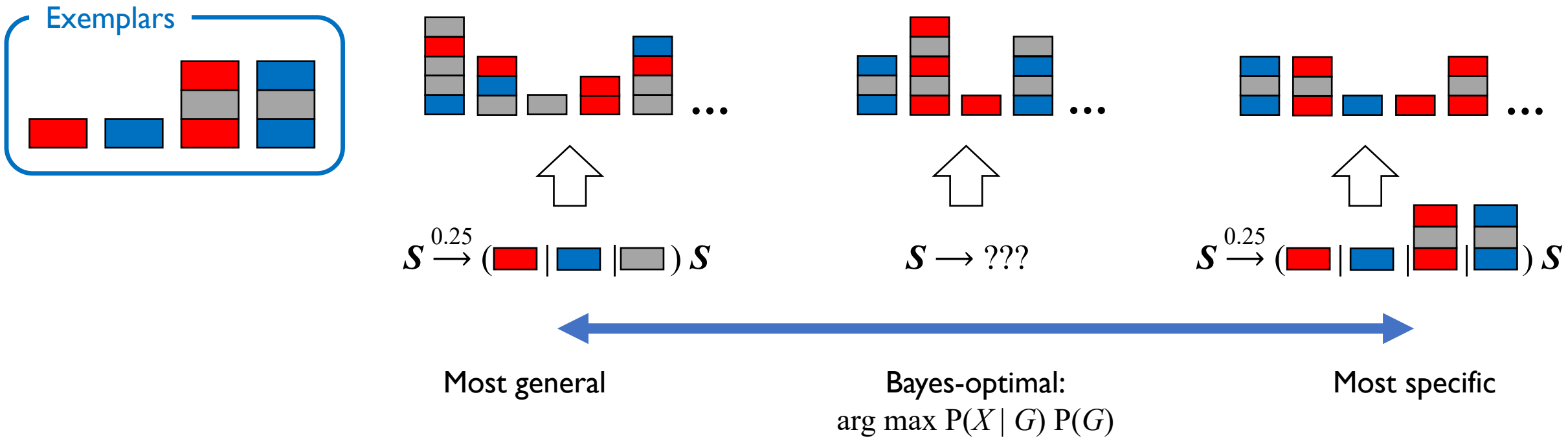
G ⟶ [Floor Door] ***F***	(1.0)
F ⟶ [Floor Windows] ***F***	(0.6)
F ⟶ [Roof]	(0.4)

$$P(\mathbf{x}) = 1.0 \times 0.6 \times 0.4$$



Talton et al., 2012

Learning a PCFG from exemplar hierarchies

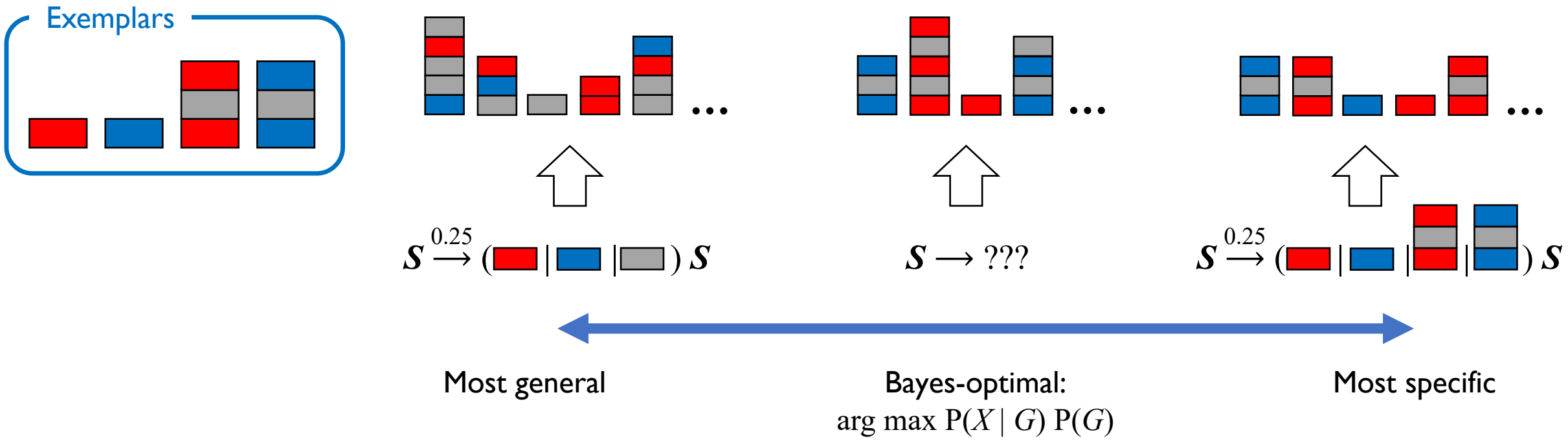


$\mathrm{P}(X \mid G)$ favors grammars which are likely to produce the exemplars

$\mathrm{P}(G)$ favors grammars which are compact (model prior)

Talton et al., 2012

Learning a PCFG from exemplar hierarchies



Optimal grammar G* is learned by **Bayesian model merging**

- Start with most specific grammar
- Explore more general (compact) grammars by merging and splitting nonterminals to improve $\mathrm{P}(X \mid G)\ \mathrm{P}(G)$ via MCMC

Talton et al., 2012

Shapes sampled from learned grammars

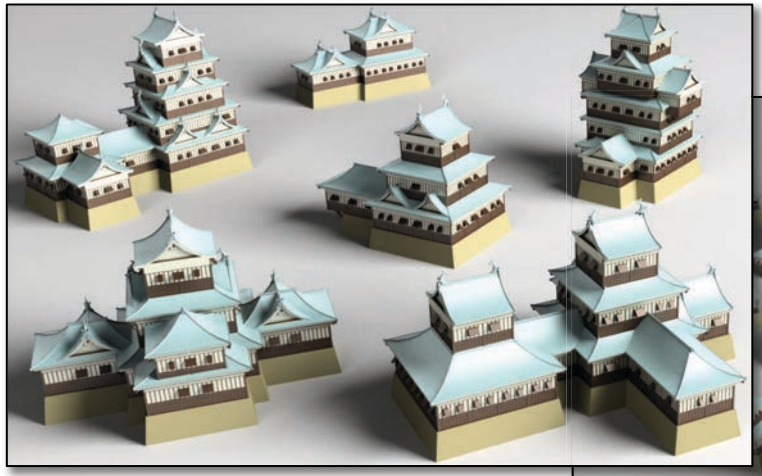
Exemplars

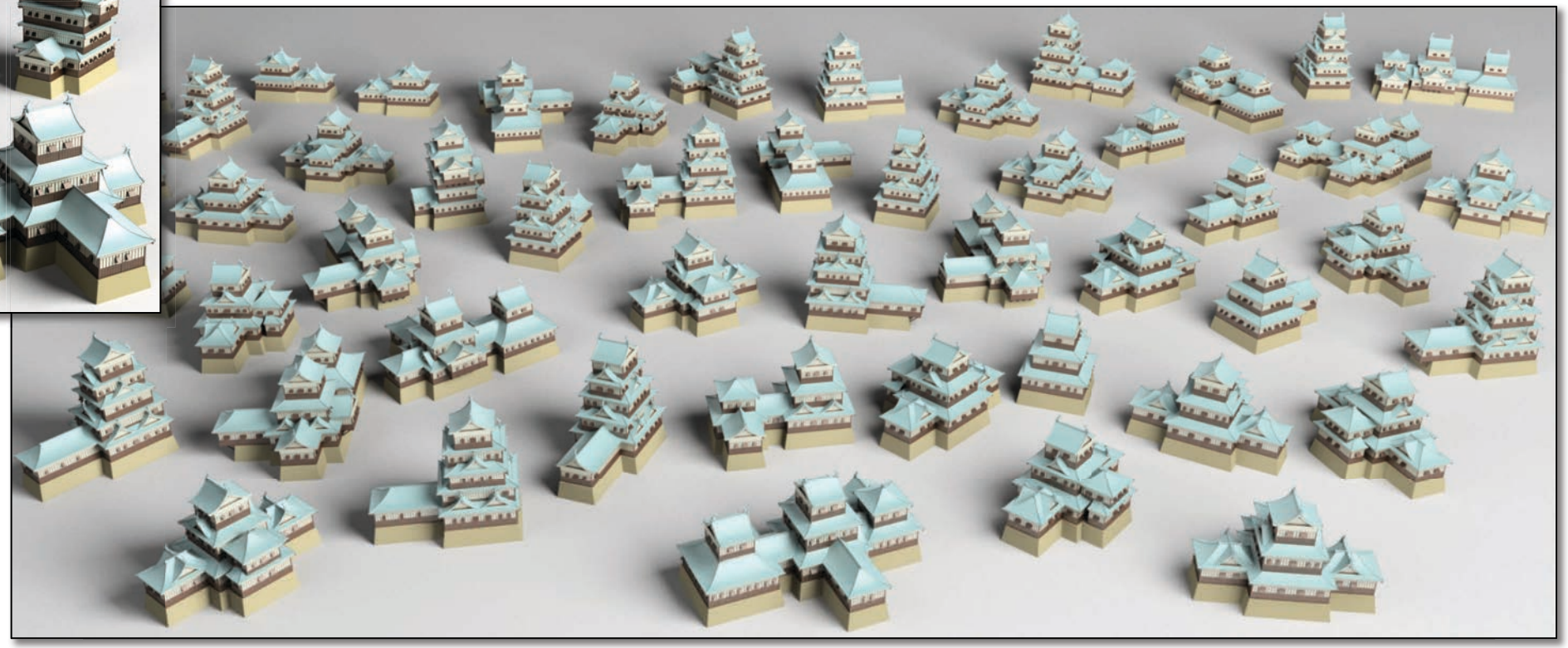
Sampled

Talton et al., 2012

Shapes sampled from learned grammars

Exemplars





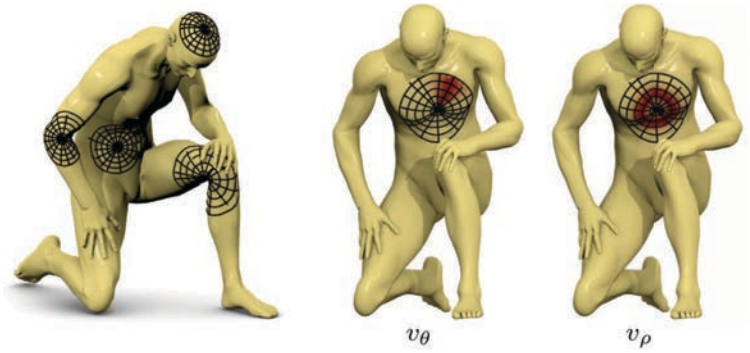
Sampled

Talton et al., 2012

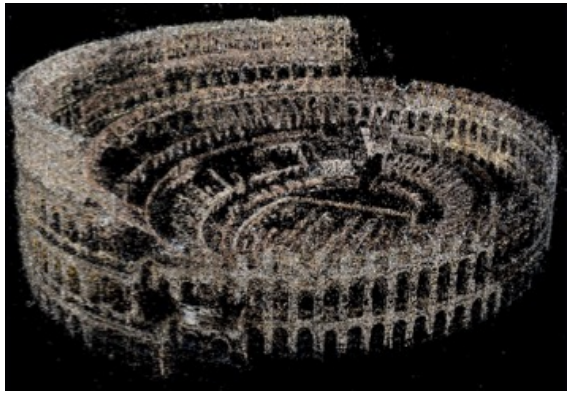
Masci et al. 2015, Qi et al. 2017, Riegler et al. 2017, Wang et al. 2017, 2018, Chen et al. 2019

Deep generative models of 3D structure

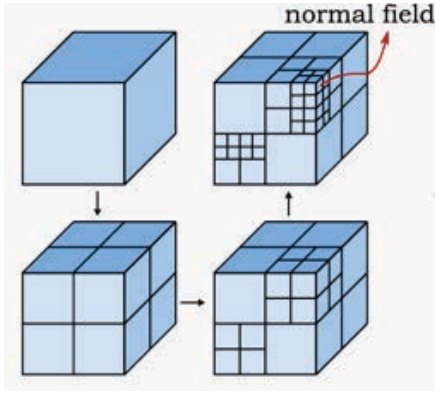
- Classical graphical models are hard to scale to complex, high-dimensional spaces with significant structural variation
- Deep neural networks are powerful high-dimensional models, but do not lend themselves naturally to non-grid-structured ("irregular") domains such as part layouts and scene graphs
  - Elegant adaptations for low-level, non-structure-aware 3D representations have been developed



Convolution on curved manifolds



Permutation-invariant NNs

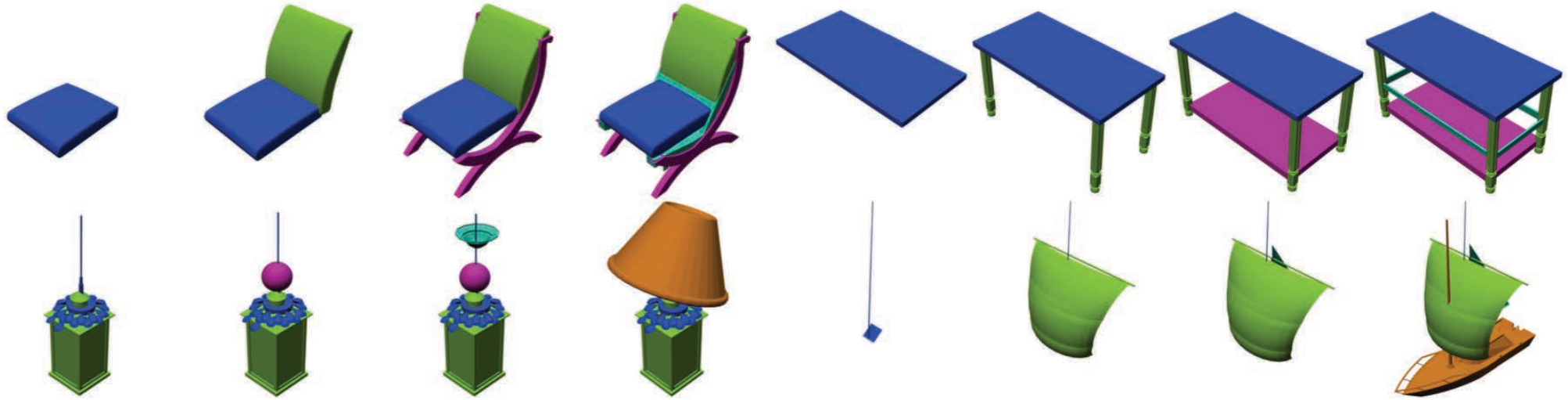


Convolution on octrees

Implicit field NNs

Incremental synthesis from existing parts

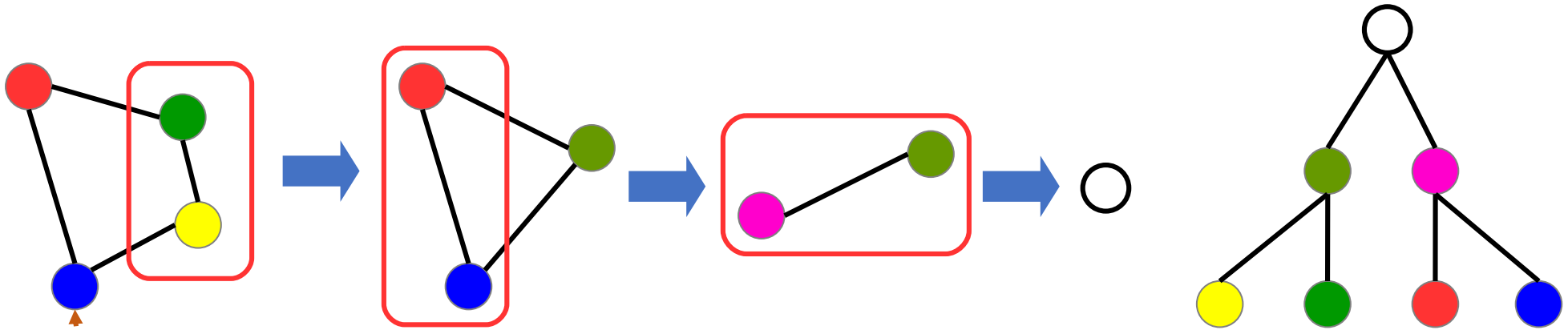
- "Recurrently" construct the shape one part at a time



- Each new part is chosen and placed (by trained networks) to complement the partially constructed shape
- Greedy strategy, avoids joint optimization over the whole structure

Encoding graphs as trees

- Edges of a graph can be collapsed sequentially to form a tree
- Each node of the tree has an associated n-D feature vector, computed recursively
- The process can be reversed to reconstruct the graph (modulo cycles)



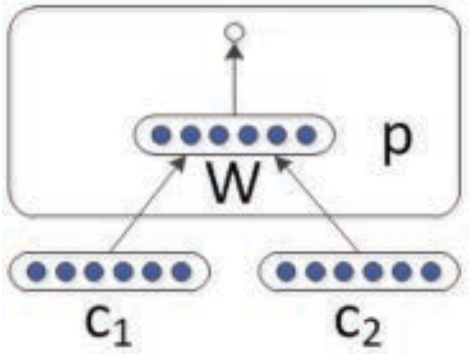
Graph represents layout of parts in 3D shape

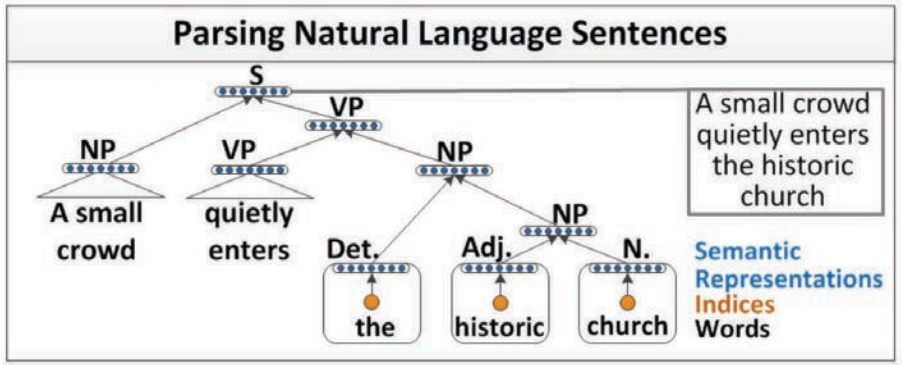
Recursive neural networks (RvNN)

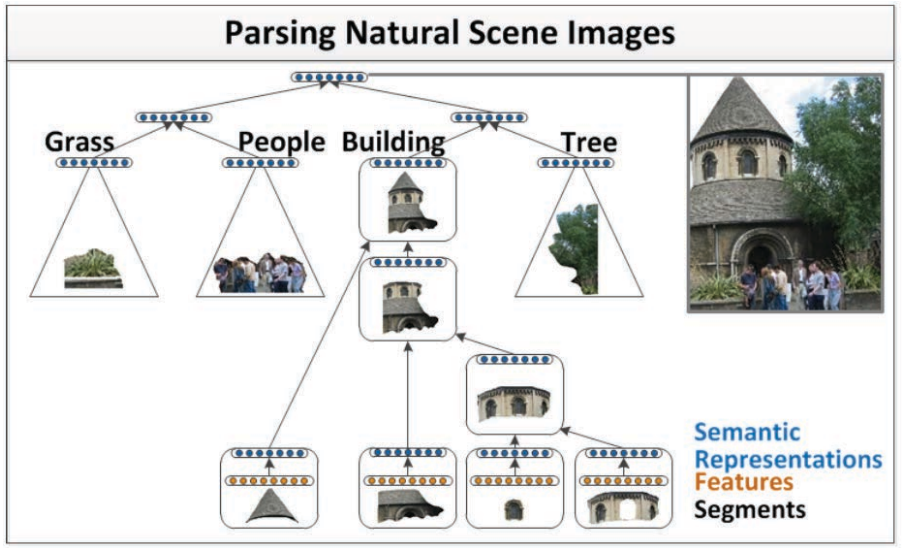
- Repeatedly merge two nodes into one
- Each node has an n-D feature vector, computed recursively

$$p = f(W[c_1; c_2] + b)$$

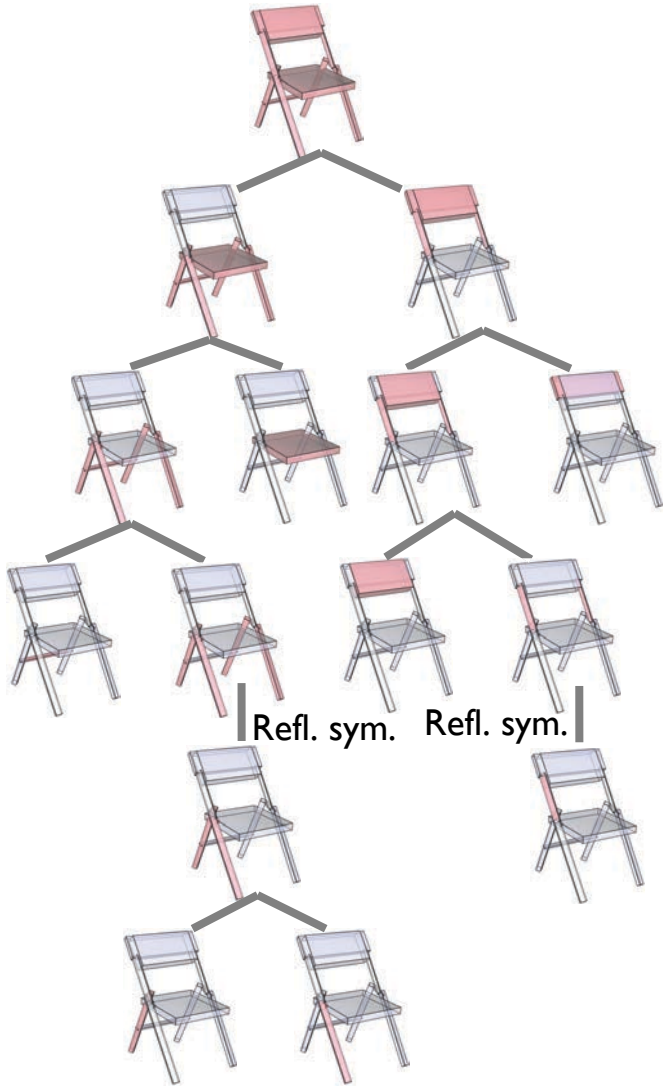
- End up with a fixed n-D vector for the root node, encoding the whole tree

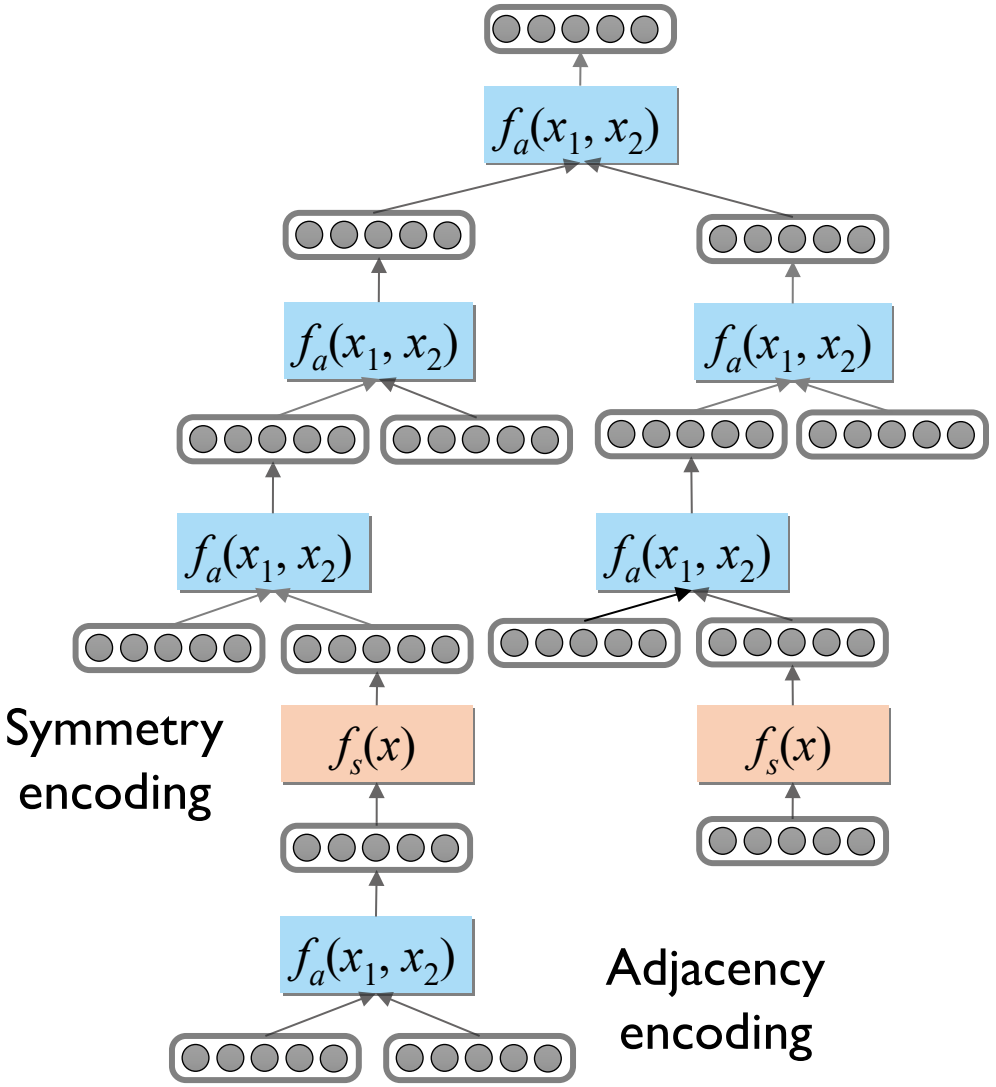






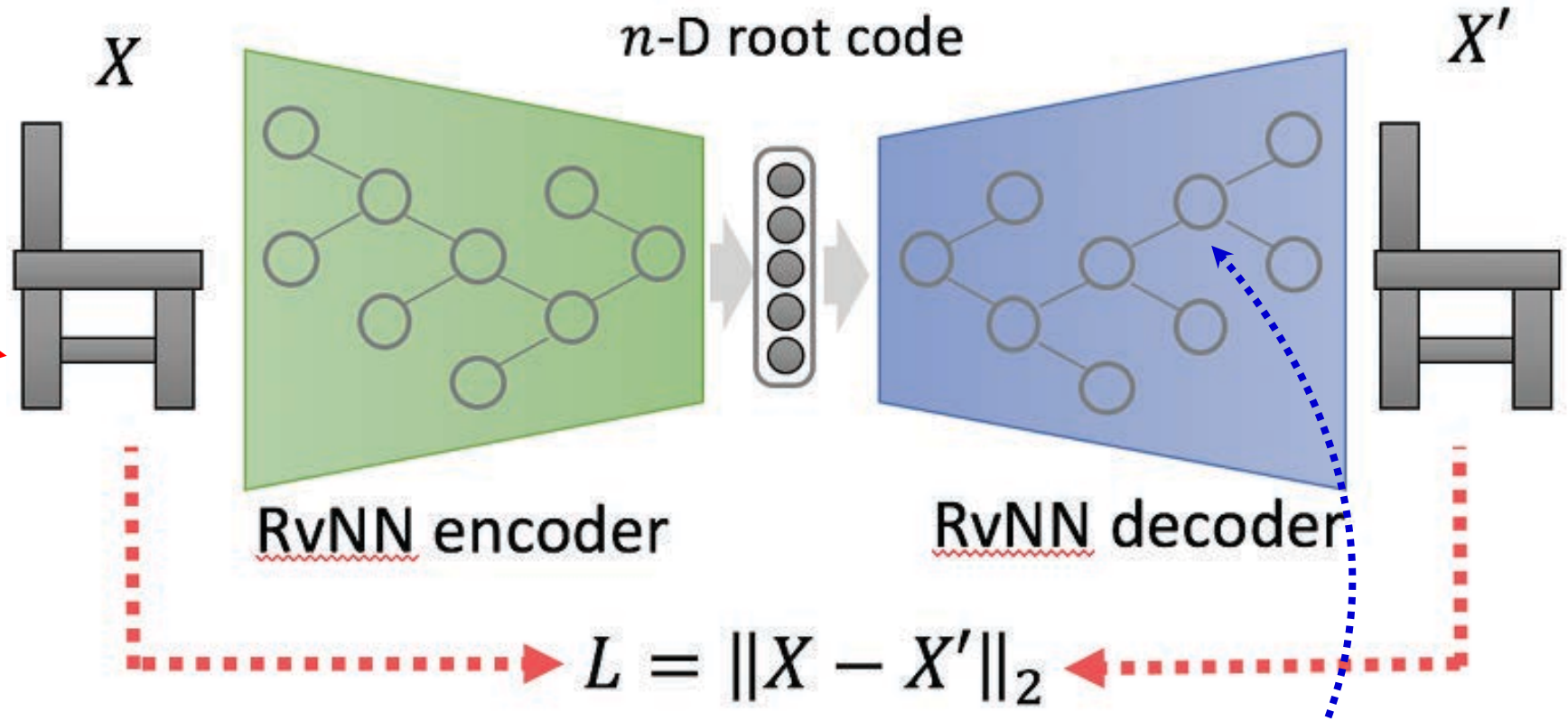
GRASS: Encoding 3D part graphs with RvNNs





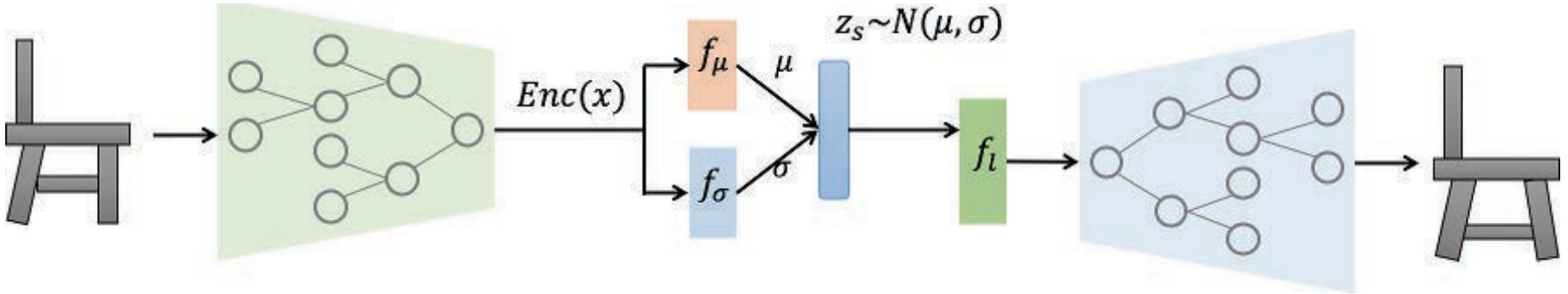
A recursive autoencoder

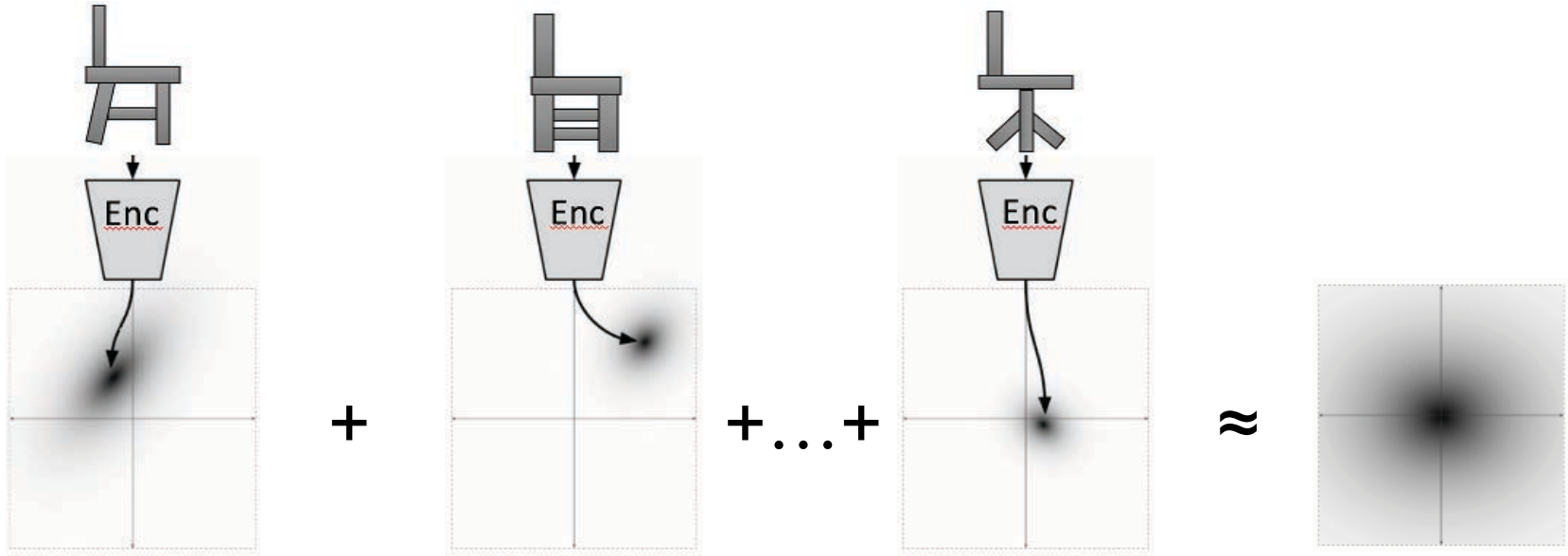
Input: collection of unlabeled parts



A classifier trained in parallel tells us whether to split a node into two adjacent parts, or into a symmetric group

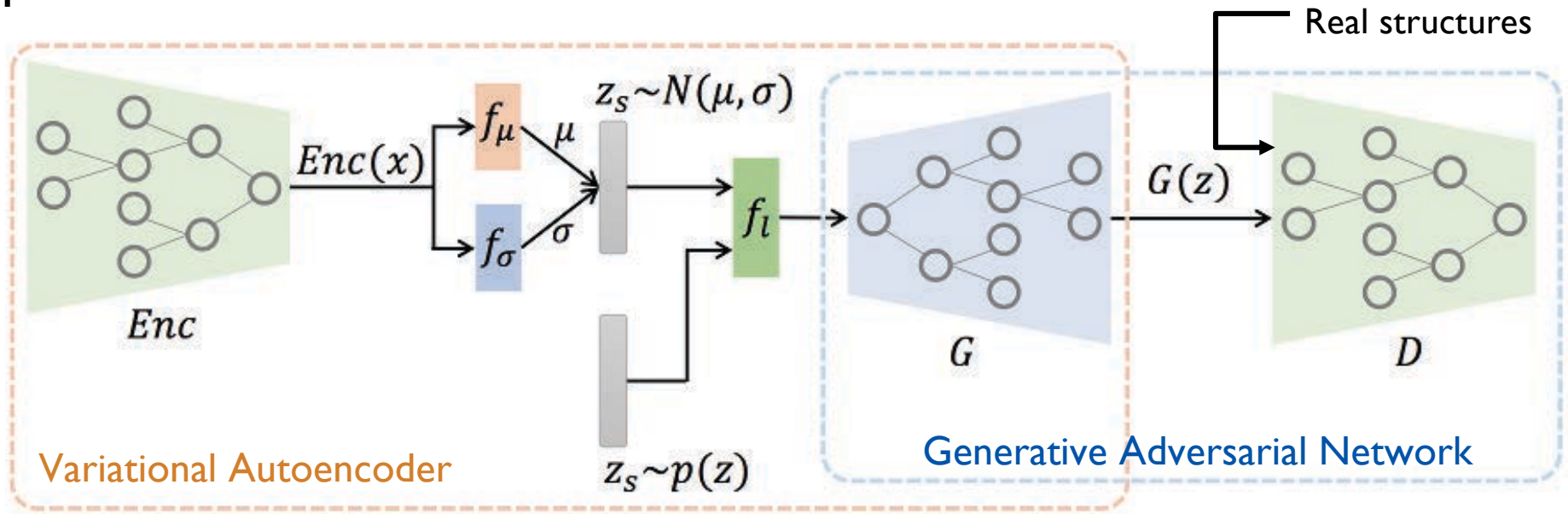
Making the model generative: Variational Autoencoder (VAE)





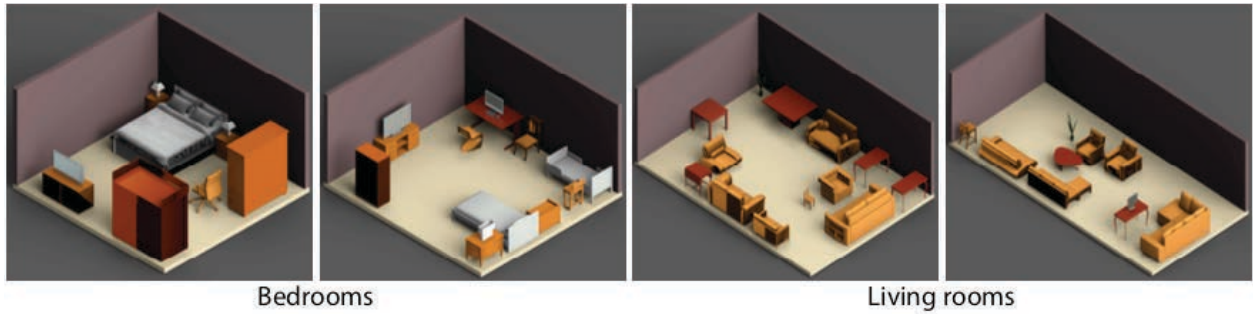
Making the VAE robust

- Adversarial training (VAE-GAN):
  - Discriminator tries to tell plausible structure (training hierarchy) from implausible one
  - Generator tries to fool discriminator by mapping random codes to plausible structures

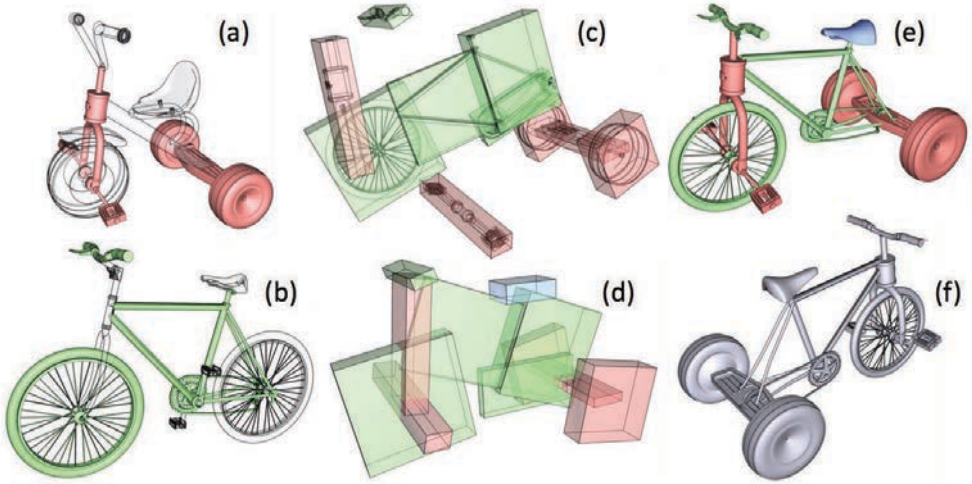


Some applications of shape RvNNs

Shape interpolation and synthesis



Scene synthesis



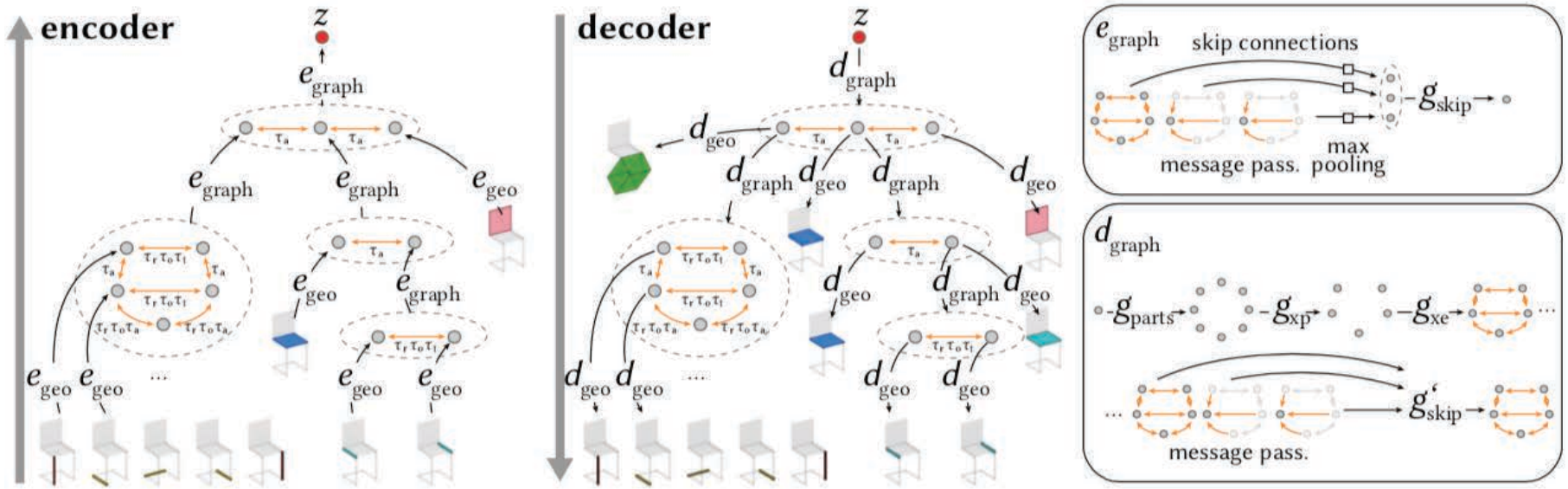
Shape composition

Evolving shape collections

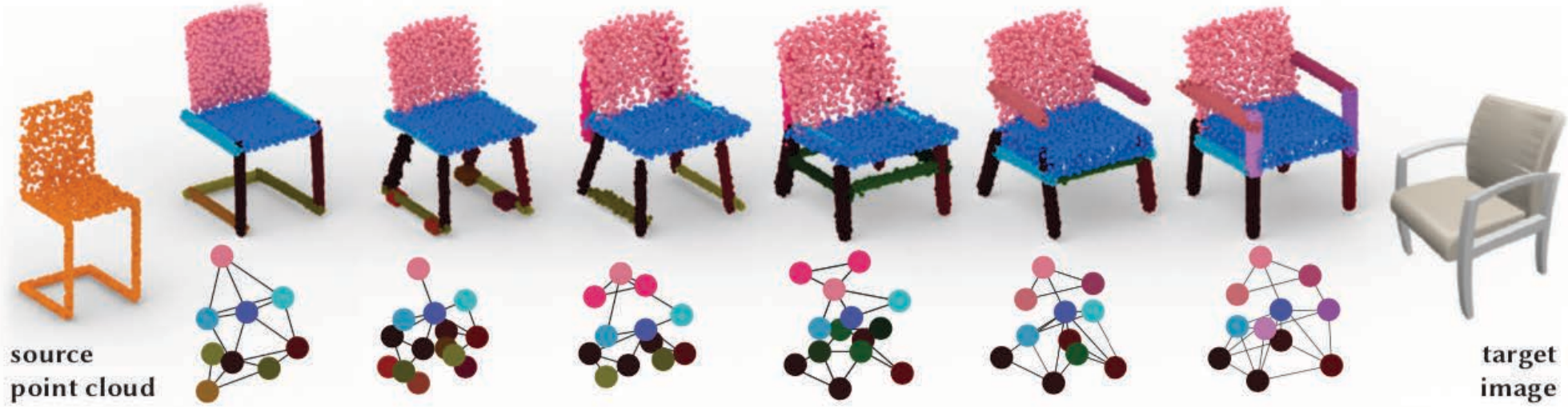
Scan reconstruction

Extending RvNNs with *n*-ary merges

StructureNet

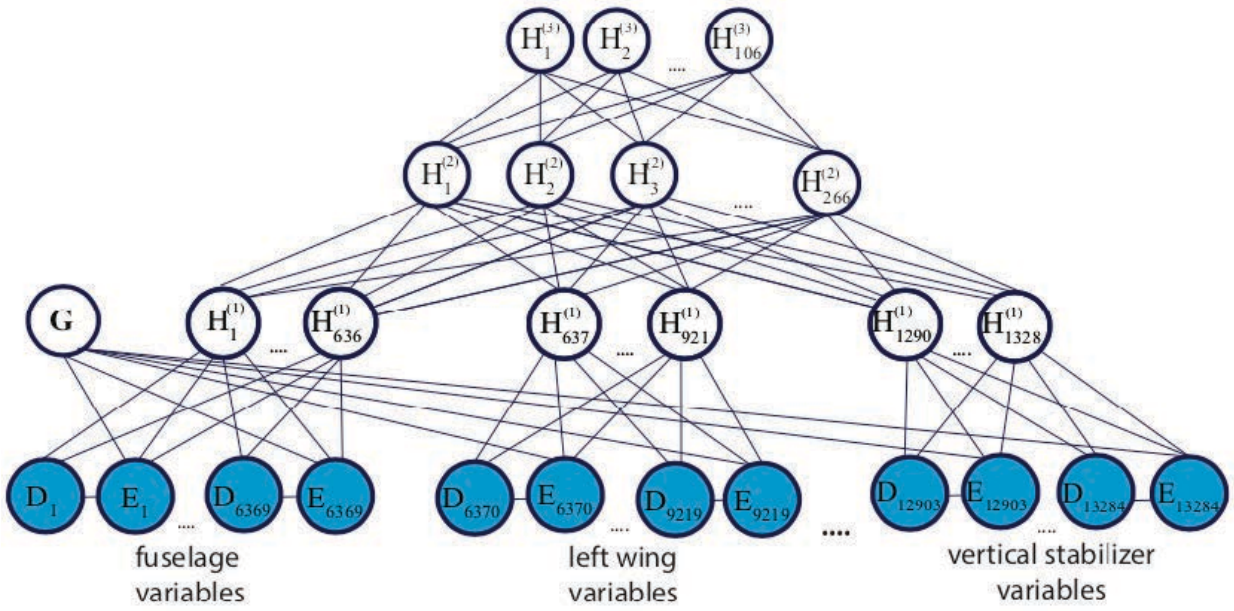


Graph convnets encode/decode variable-degree nodes



Modeling fine-grained geometry

- RvNNs are powerful models of layout variability, but not (yet) of fine-grained local geometry
- Huang et al. [2015] developed a structure-aware surface generator, using a Deep Boltzmann Machine



Modeling fine-grained geometry

- SDM-Net [Gao et al. 2019] models parts as deformed boxes, and chains a part VAE with a simple structure VAE

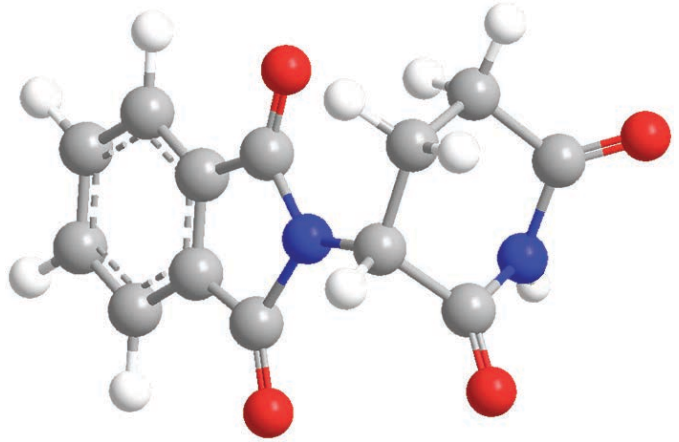
Input PSG AtlasNet **SDM-Net**

Takeaways

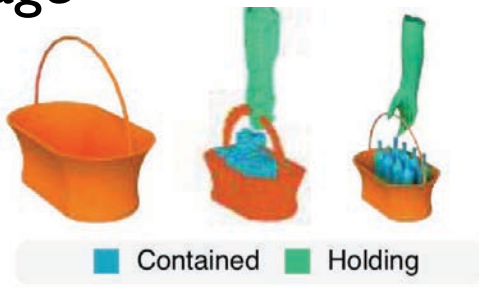
- Structure is an efficient low-dimensional factorization that captures rich shape variation
- Modeling structural variation requires new (or newly borrowed) learning architectures
- Structure-aware generation produces annotated, high-level-editable output "for free"
- Models of semantic and functional intent can layer on top of structural shape priors
  [UIST '13, SIGGRAPH '15]

Lilian van Daal; American Chemical Society; Hu et al. 2018; Xu et al. 2019

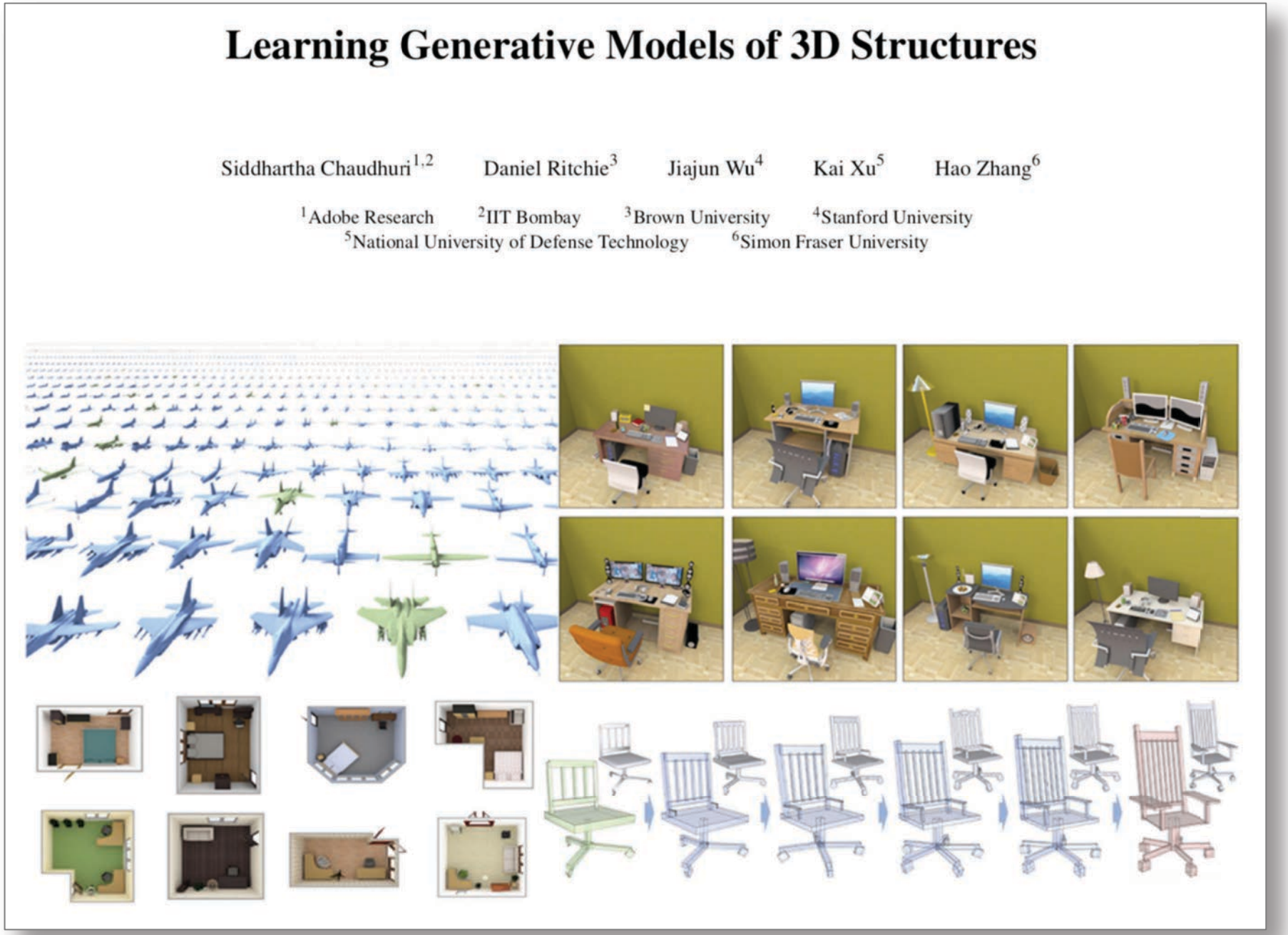
Research Directions



- **Richer structural models**
  - Domain-specific architectures
  - Probabilistic program induction
- **Easier training**
  - Weakly/semi-supervised training (infer structure without training structures)
  - Few-shot generalization with domain knowledge
- **Richer objectives**
  - Linking structure with physical materials and fabrication constraints
  - Function, motion, language
    - furniture
    - mechanical assemblies
    - drug design …



For much more, see our forthcoming Eurographics State-of-the-Art Report (STAR)



http://www.cse.iitb.ac.in/~sidch