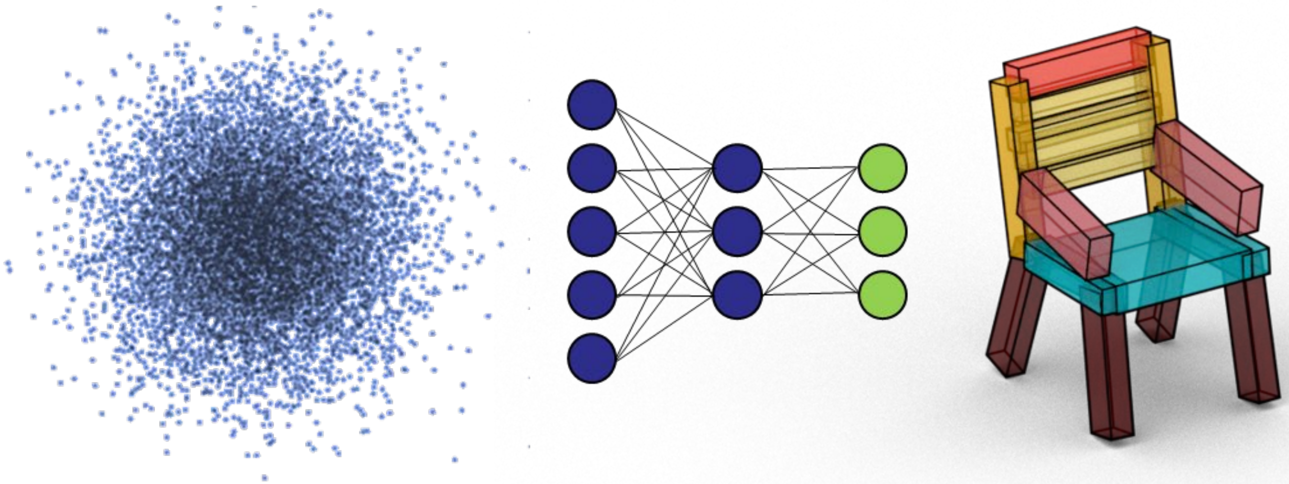
CS348n: Neural Representations and Generative Models for 3D Geometry



Leonidas Guibas
Computer Science Department
Stanford University

Leonidas Guibas Laboratory

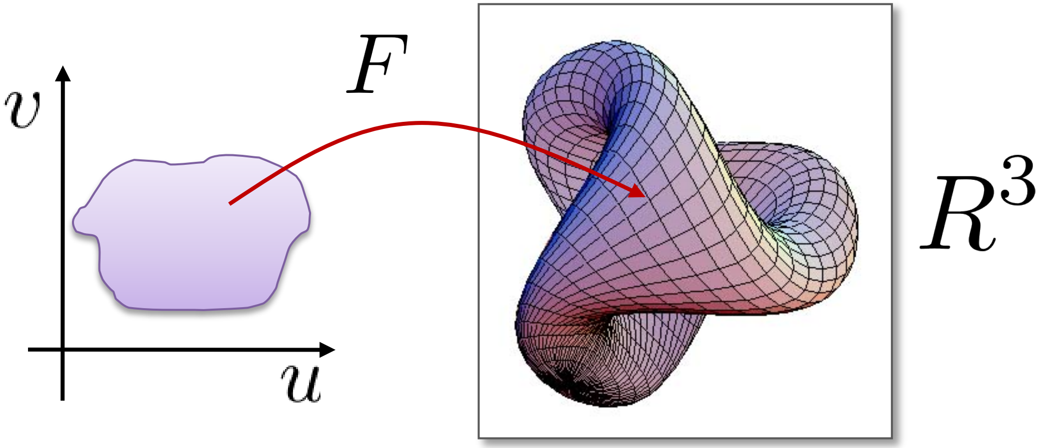
Geometric Computing

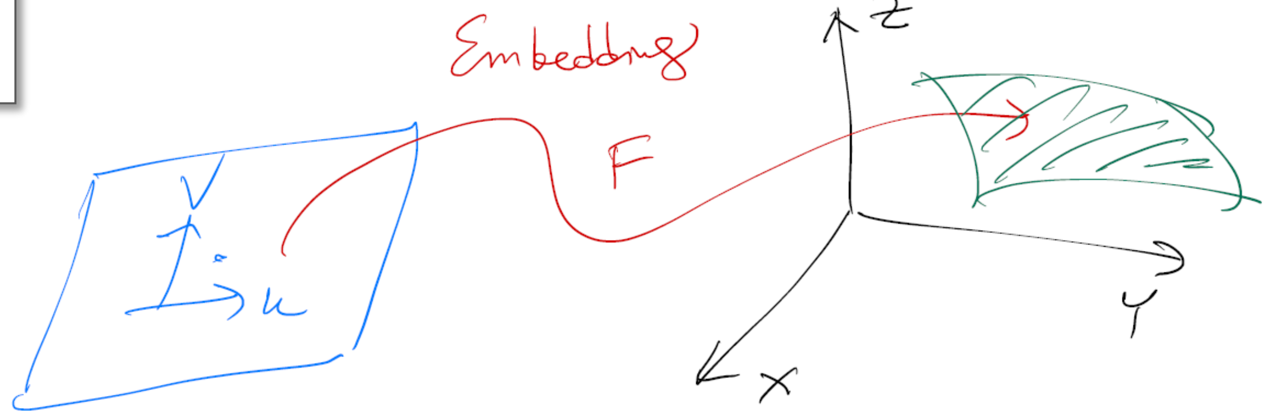
Class Logistics

- Homework 1 (VAE) is due today, 11:59 pm
- Homework 2 (deepSDF) is out, due Wed, Feb 9, 2022 (two weeks)
- Class will continue on Zoom until Clark S361 becomes available (~Feb 15)

Last Time: Neural Parametrics, GANs

Parametric Curves and Surfaces via Functions





$$F(u,v) \equiv (X(u,v), Y(u,v), Z(u,v))$$

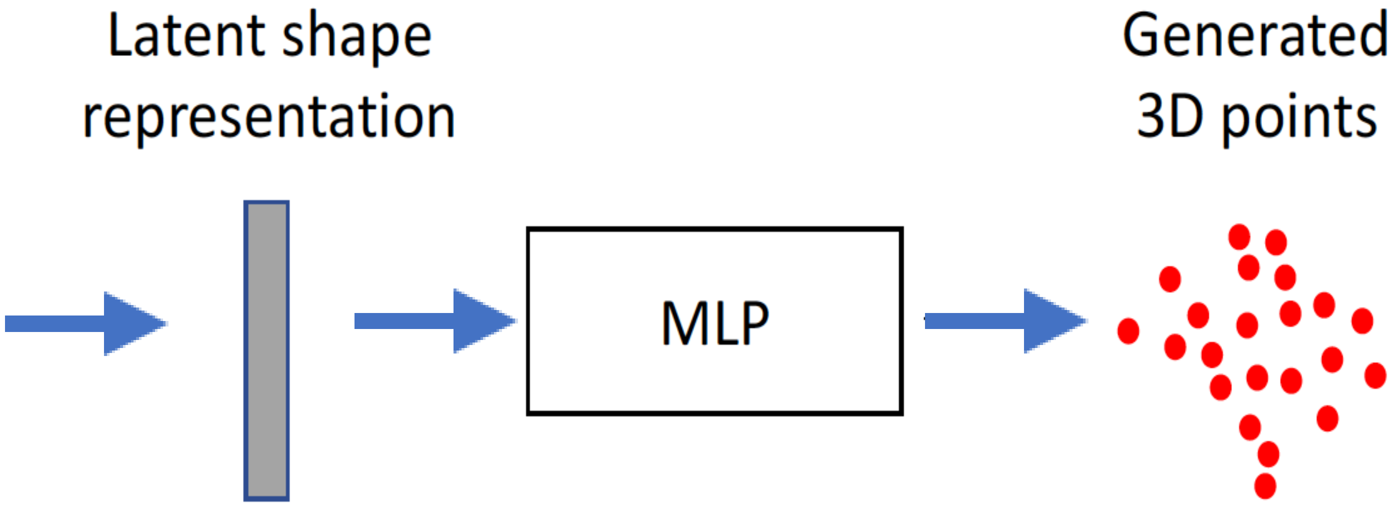
$$F: \underset{\uparrow \text{ parameter space}}{R^2} \longrightarrow R^3 \leftarrow \text{ambient space}$$

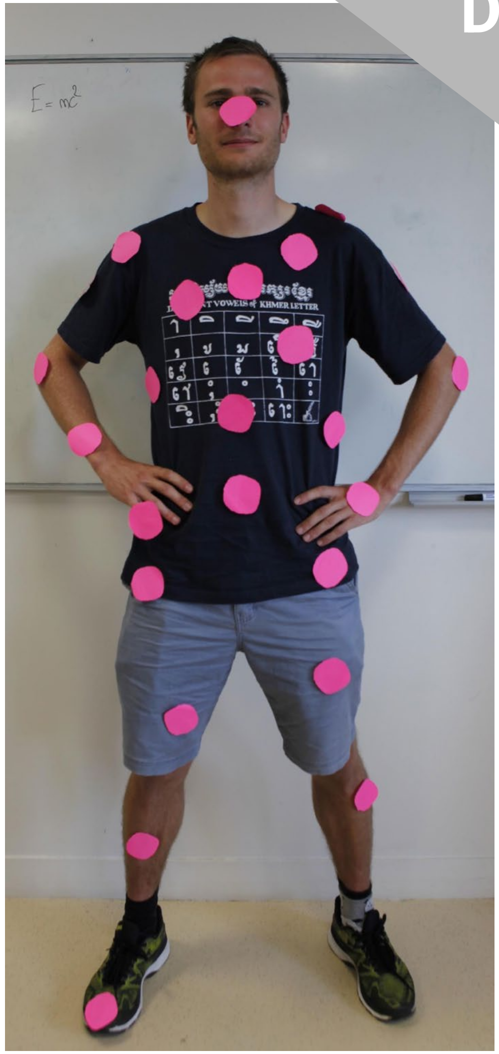
These parametric mappings can be explicit functions, but can also be neural networks

Generating points

Encoder
E

Decoder
D

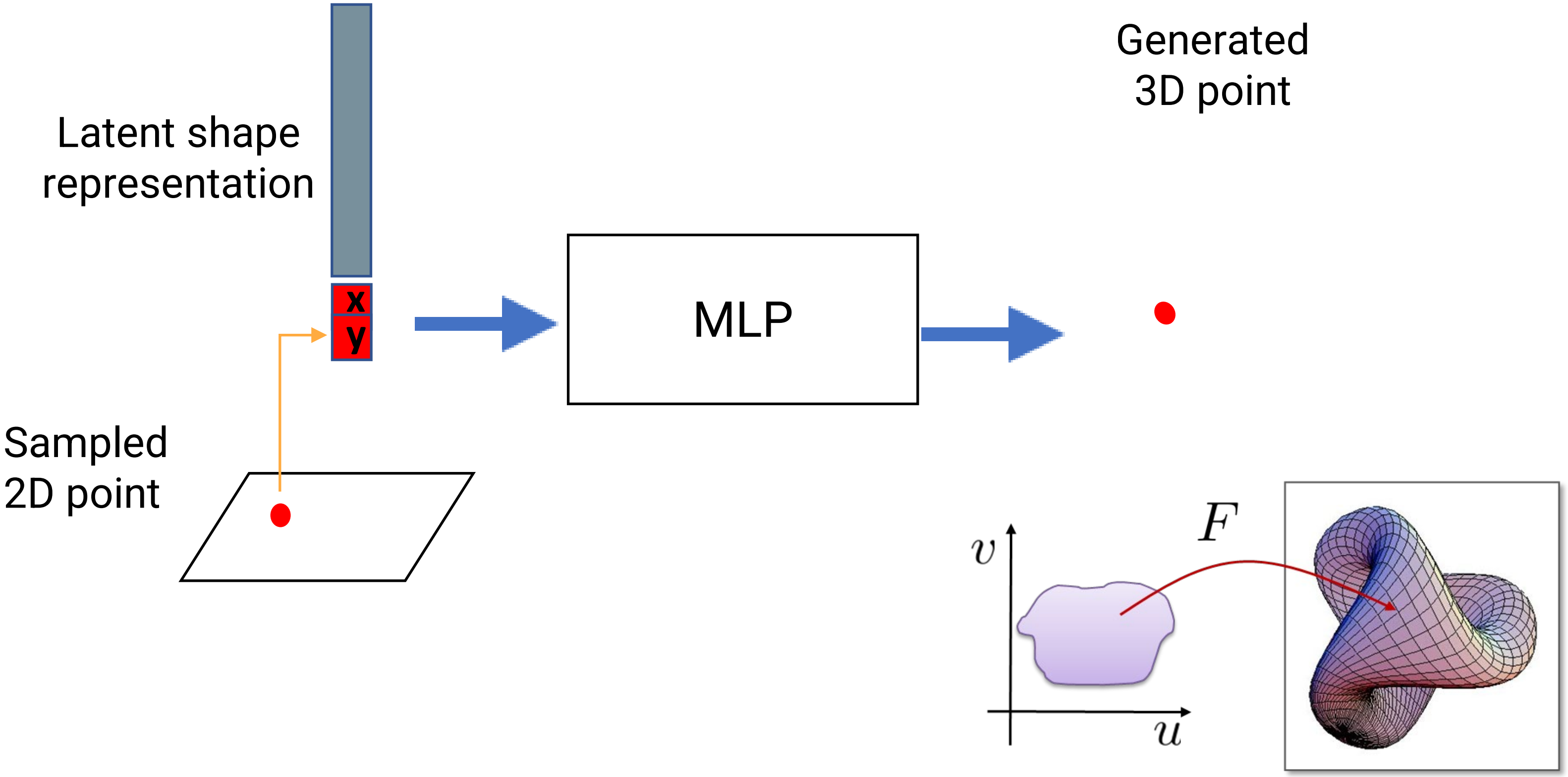




Issue: no idea of surfaces

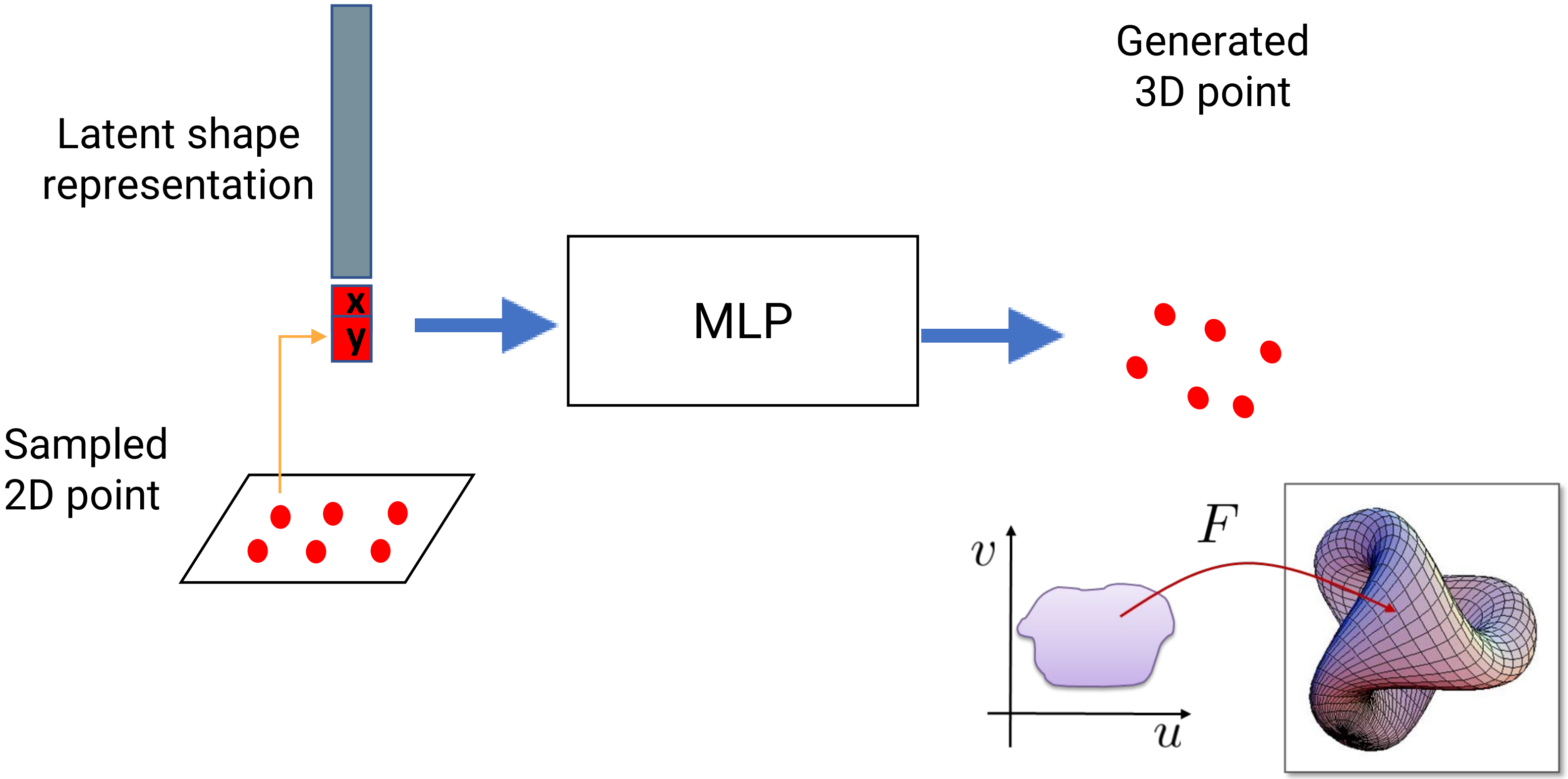
Deform a surface : space mapping trick [Groueix2018]

Decoder
D

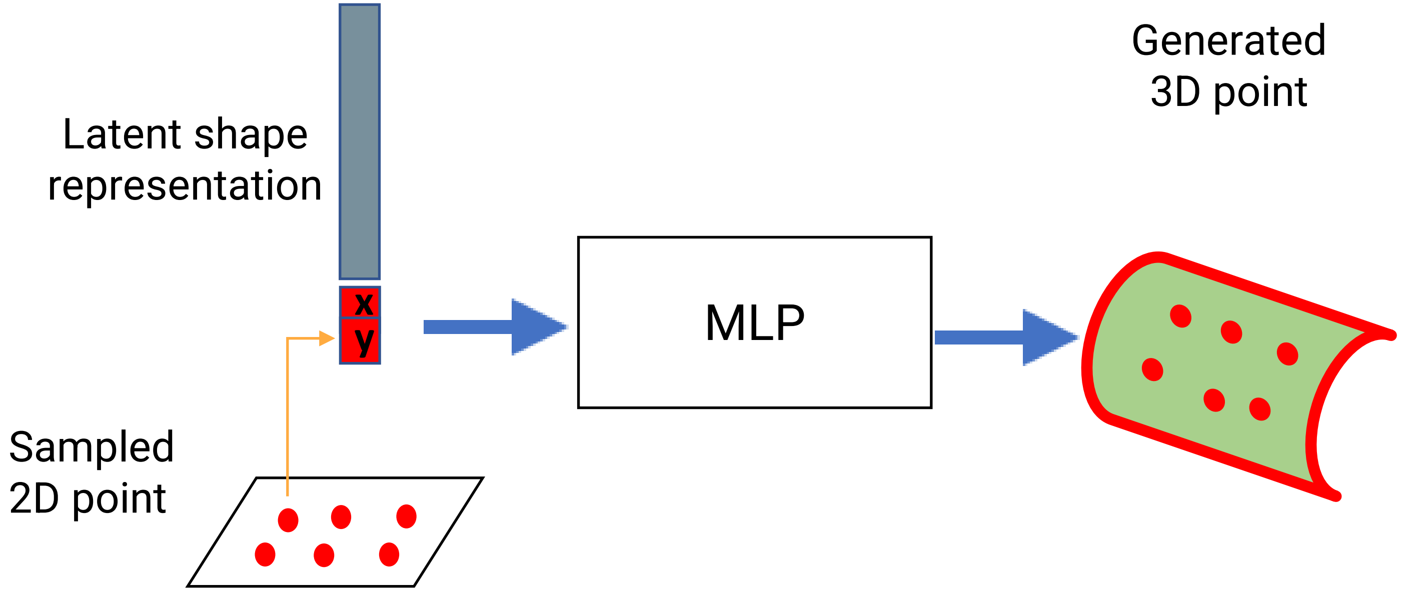


Deform a surface [Groueix2018]

Decoder
D

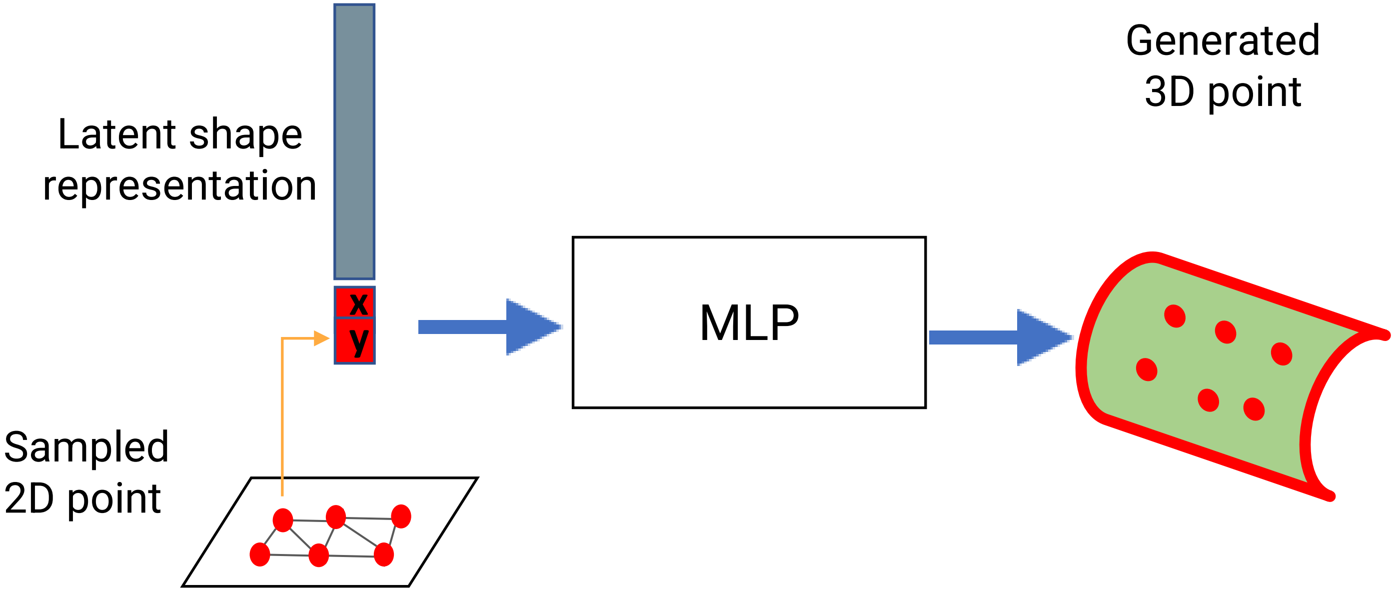


Deform a surface [Groueix2018]

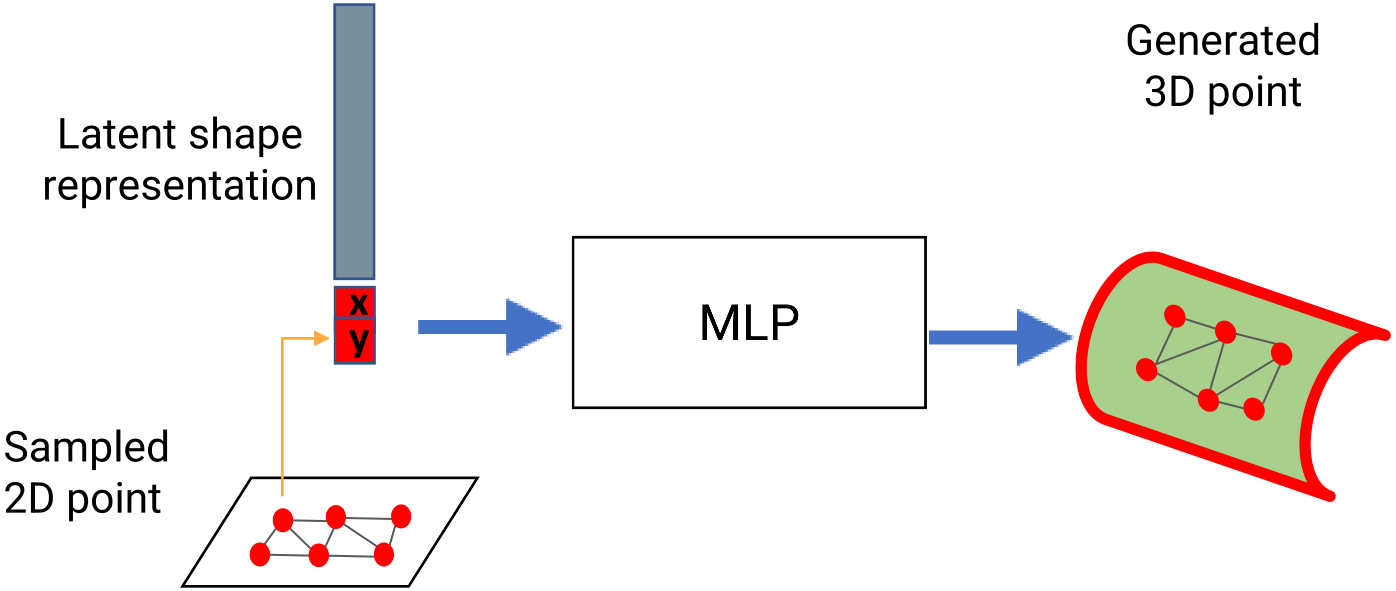


Deform a surface [Groueix2018]

Decoder
D



Deform a surface [Groueix2018]

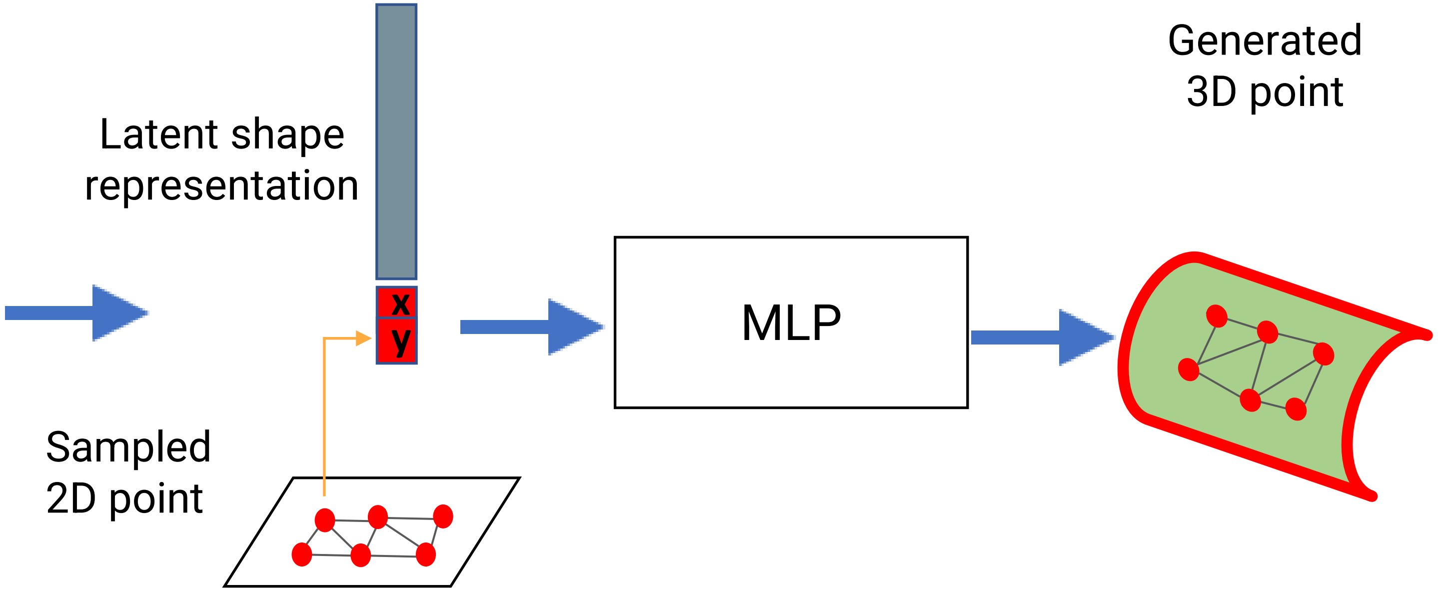


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

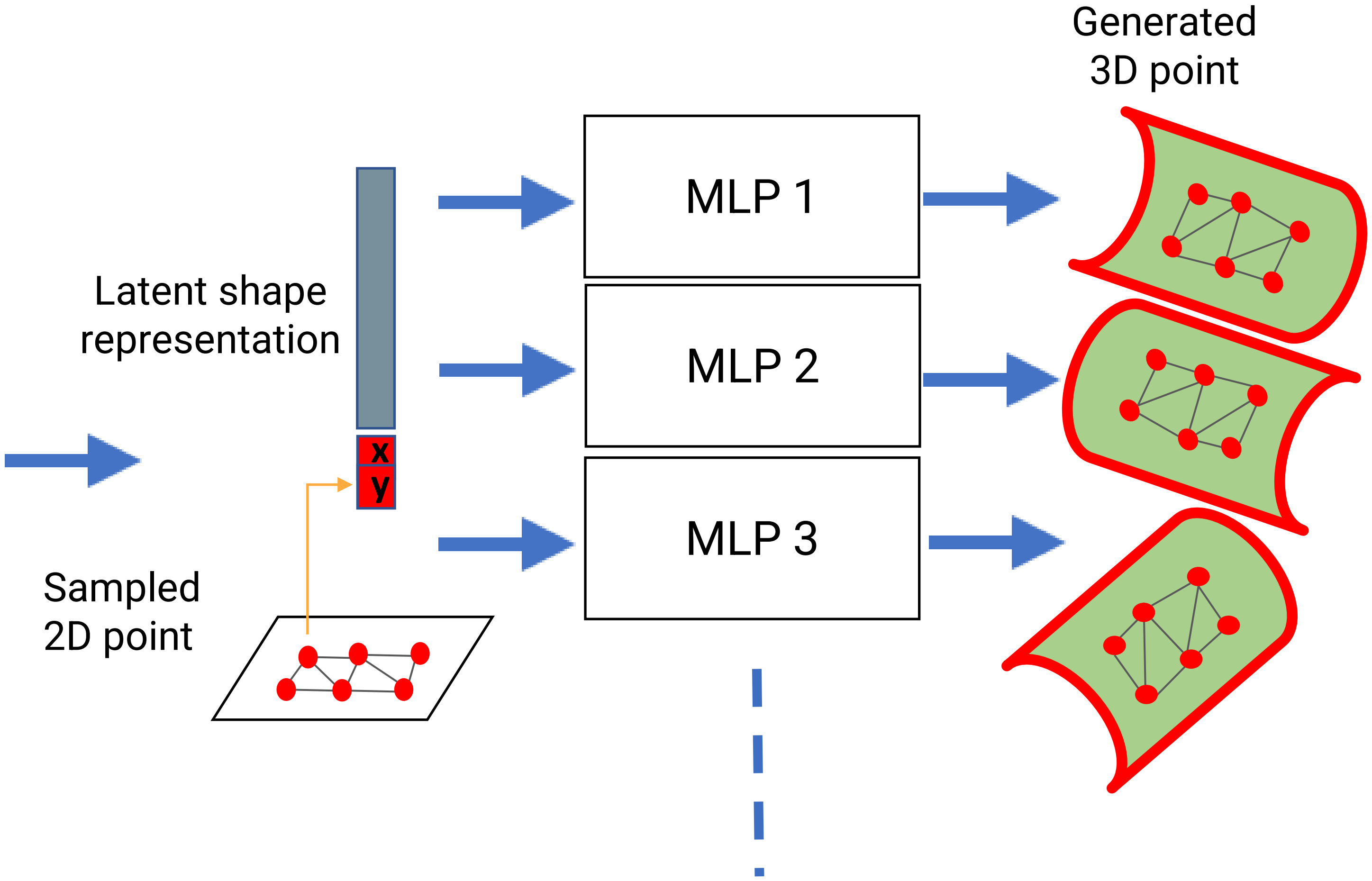


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape



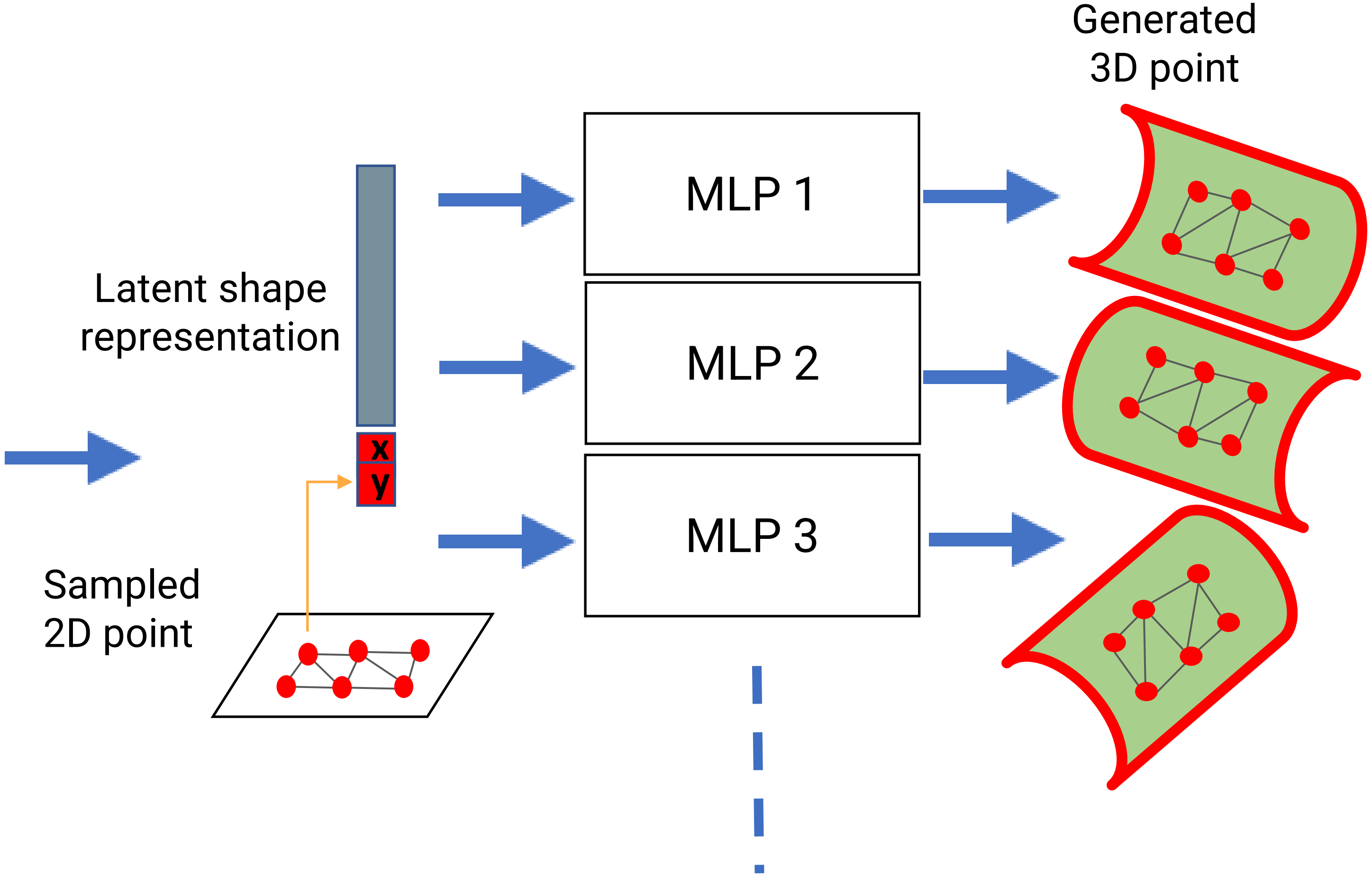
Piecewise Parametric Surfaces

Thibault Groueix, Pierre-Alain Langlois, 2019

Deform a surface [Groueix2018]

Encoder
E

Test Shape



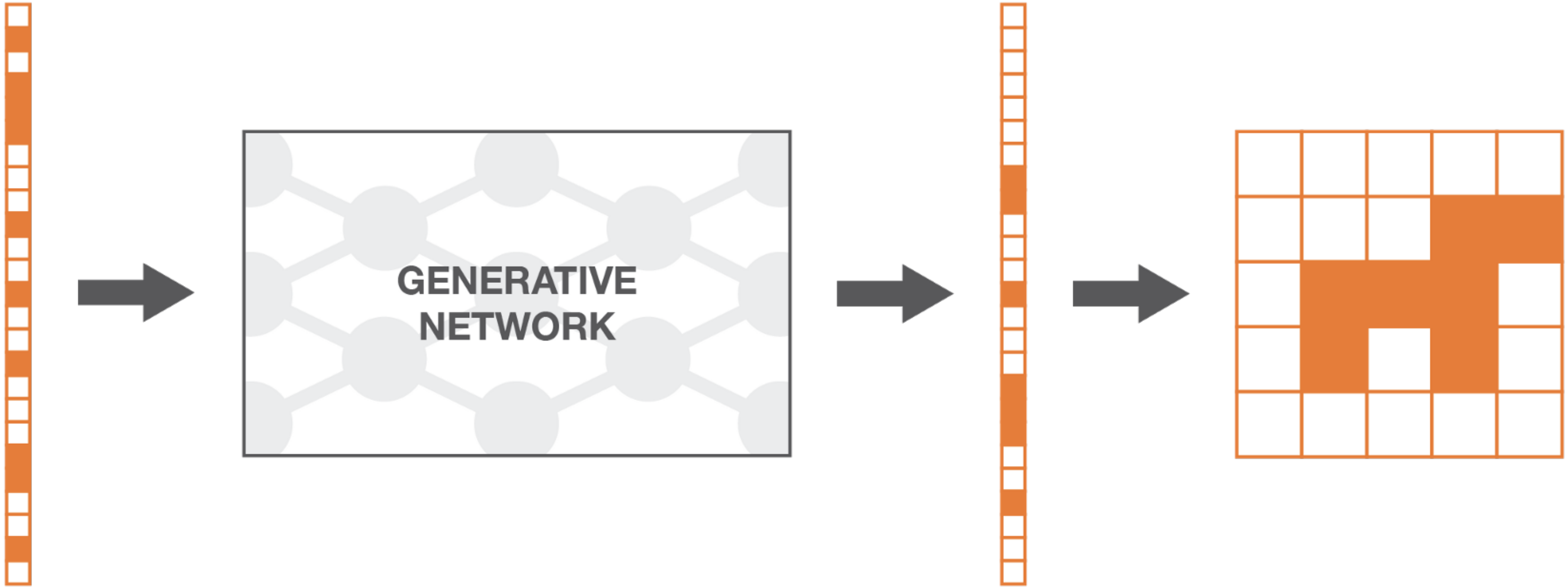
Decoder
D

Thibault Groueix, Pierre-Alain Langlois, 2019

Direct application : mesh parametrization

Thibault Groueix, Pierre-Alain Langlois, 2019

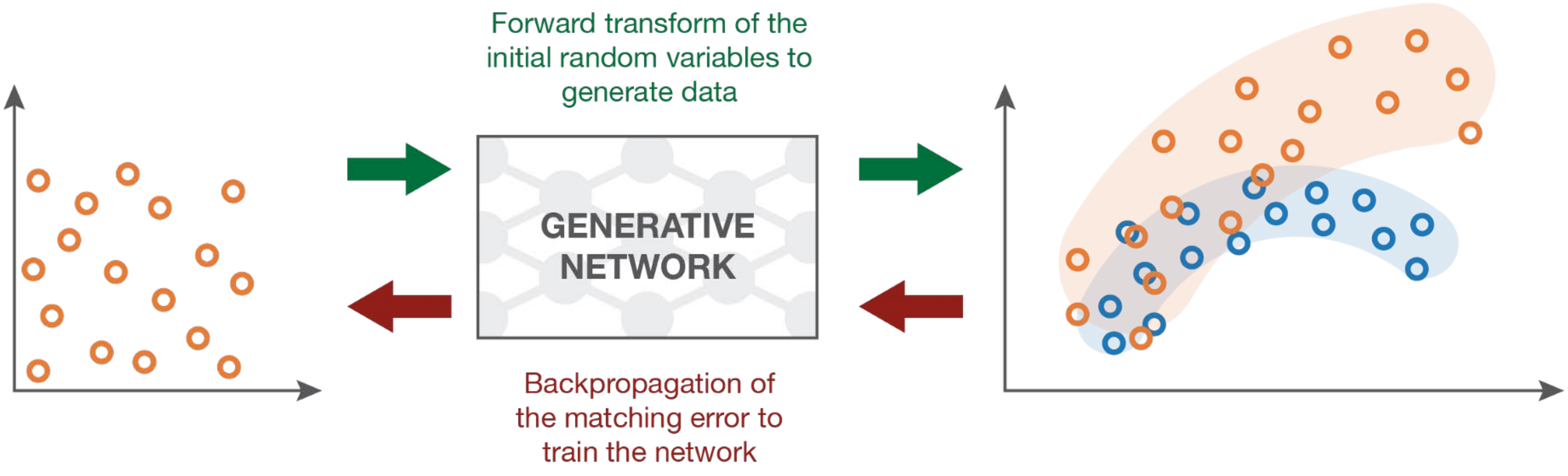
Generative Models



Input random variable (drawn from a simple distribution, for example uniform).	The generative network transforms the simple random variable into a more complex one.	Output random variable (should follow the targeted distribution, after training the generative network).	The output of the generative network once reshaped.

Thibault Groueix, Pierre-Alain Langlois, 2019

Gradient Descent Based on this Loss for Training



Input random variables (drawn from a uniform).

Generative network to be trained.

The generated distribution is compared to the true distribution and the “matching error” is backpropagated to train the network.

Extremely expensive!

An Alternative: Compare on a Downstream Task

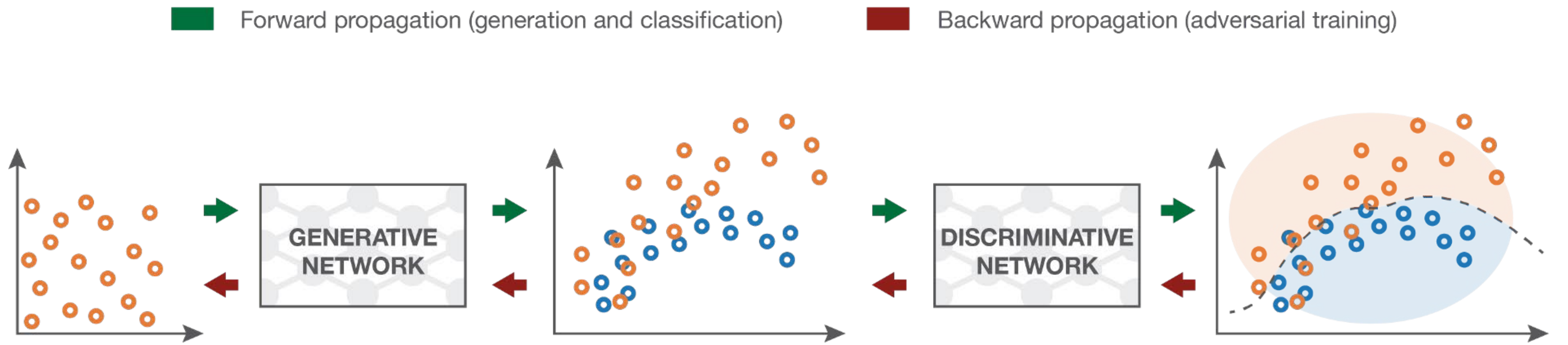
- An indirect loss
- **Generative Adversarial Networks (GANs)**: compare distributions through a downstream task
- Use the loss of that task to improve the generator
- Make that task itself be a trainable neural network

Use a Discriminator

- Learn a discriminator through another neural network
- the goal of the generator is to fool the discriminator, so the generative neural network is trained to maximize the final classification error (between true and generated data)
- the goal of the discriminator is to detect fake generated data, so the discriminative neural network is trained to minimize the final classification error

At each iteration of the training process, the weights of the generative network are updated in order to increase the classification error (error gradient ascent over the generator’s parameters) whereas the weights of the discriminative network are updated so that to decrease this error (error gradient descent over the discriminator’s parameters).

An Adversarial Discriminator



Input random variables.	The generative network is trained to **maximise** the final classification error.	The generated distribution and the true distribution are not compared directly.	The discriminative network is trained to **minimise** the final classification error.	The classification error is the basis metric for the training of both networks.

A Mathematical Formulation

- a generative network $G(.)$ that takes a random input z with density p_z (the "noise vector") and returns an output x_g = G(z) that should follow (after training) the targeted probability distribution
- a discriminative network $D(.)$ that takes an input x that can be a "true" one (x_t, whose density is denoted p_t) or a "generated" one (x_g, whose density p_g is the density induced by the density p_z going through G) and that returns the probability $D(x)$ of to be a "true" data

Error $$\begin{aligned} E(G,D) &= \frac{1}{2}\mathbb{E}_{x\sim p_t}[1-D(x)] + \frac{1}{2}\mathbb{E}_{z\sim p_z}[D(G(z))] \\ &= \frac{1}{2}\left(\mathbb{E}_{x\sim p_t}[1-D(x)] + \mathbb{E}_{x\sim p_g}[D(x)]\right) \end{aligned}$$

$$\max_G \left(\min_D E(G,D)\right)$$

A minimax Nash equilibrium

GAN Advantages

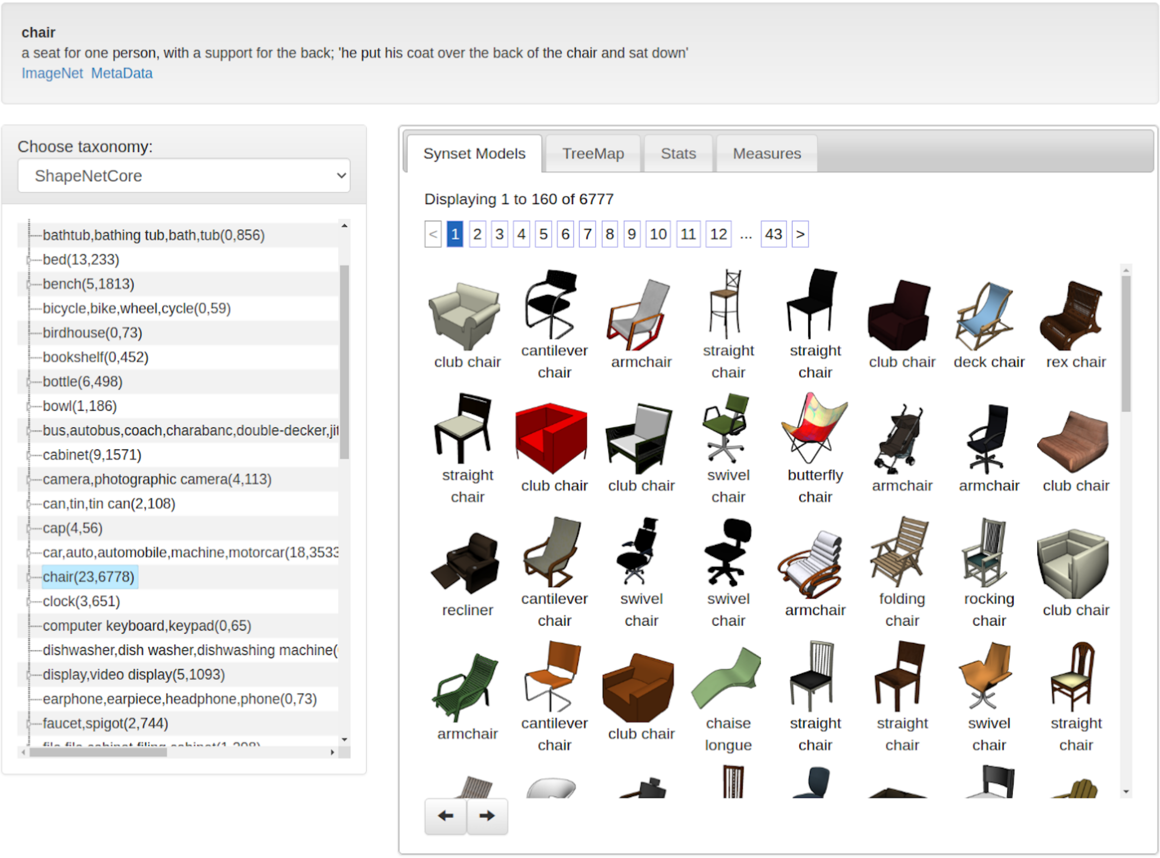
- Generation is straightforward
- Mode detail is captured
- Training does not require MLE estimation
- Robust to overfitting (generator never sees the training data)
- Impressive empirical results

GAN Issues

- Learned probability distribution is implicit
  - Vanilla GANS only good for sampling/generation
- Training is difficult and often unstable
  - Non-convergence
  - Vanishing gradients
  - Mode collapse

Public 3D Data Sets

ShapeNet: Large-scale 3D Shape CAD Models

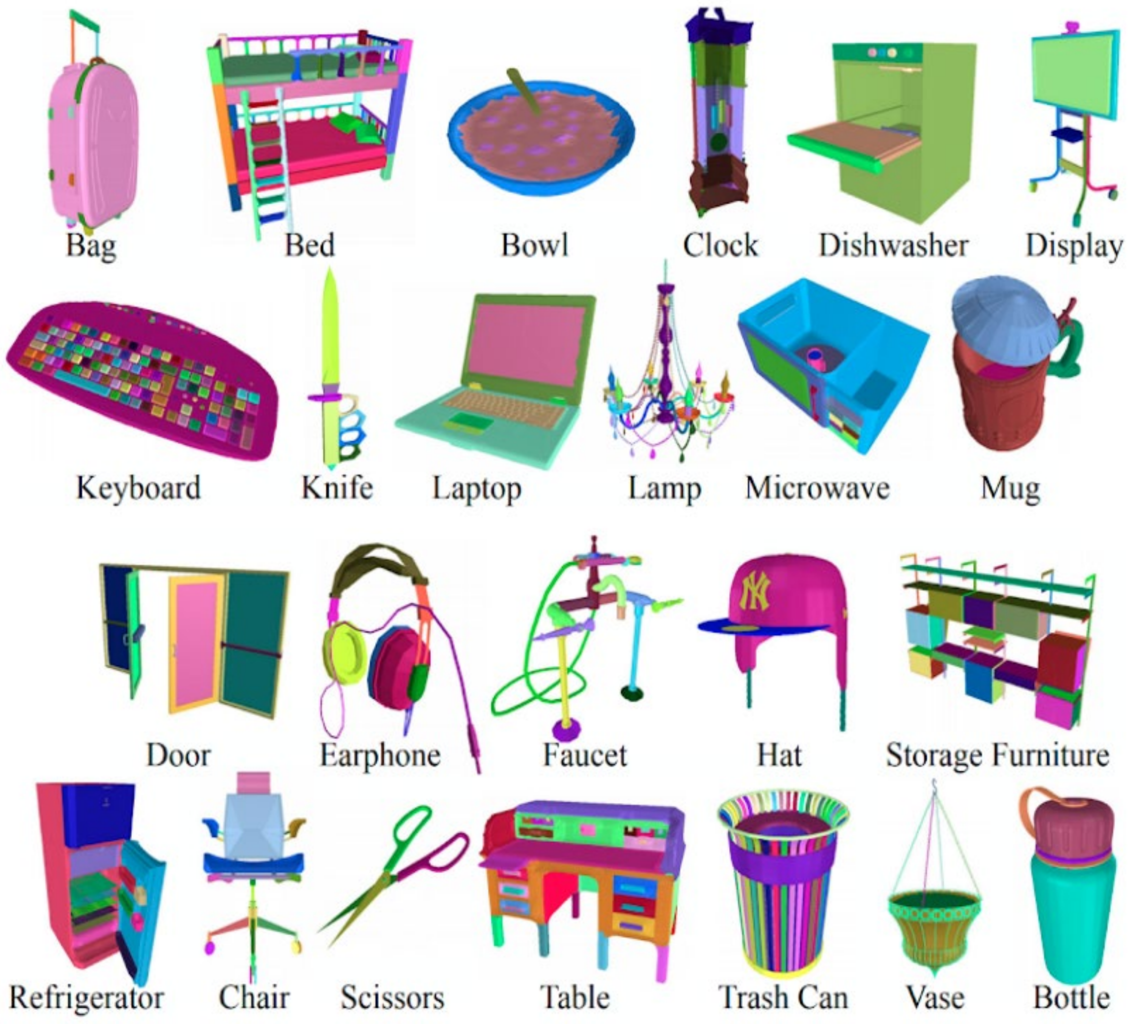


- Synthetic, 3D Shapes
- > 3M Models, 3K Categories
- ShapeNetCore:
  - 51,300 unique 3D models
  - 55 common categories
  - Canonical Poses/Sizes
  - Shape/Images/Text/etc.
- Advance fields in 3DDL

https://www.shapenet.org

Chang et al., "**Shapenet: An information-rich 3d model repository**"

PartNet: Part Segmentation Annotation



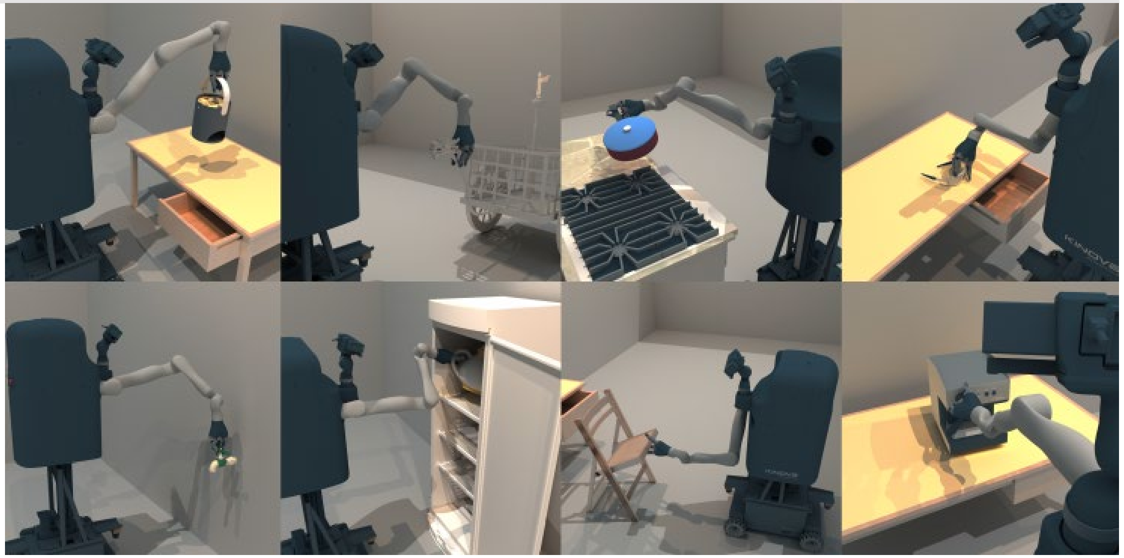
https://partnet.cs.stanford.edu/

- Synthetic, 3D Shapes
- Based upon ShapeNet
- 573,585 Part Instances
- 26,671 Objects, 24 Categories

- Part Segmentations
  - Fine-grained
  - Hierarchical
  - Instance-level

Mo et al., "**PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding**", CVPR 2019

PartNet-Mobility and SAPIEN: Part Articulation



- Synthetic, 3D Shapes
- Based upon ShapeNet/PartNet
- 14,068 Articulated Parts
- 2,346 Objects, 46 Categories

- Part Articulation
- Physical Simulation

- Support the study of various robotic manipulation tasks

https://sapien.ucsd.edu/

Xiang et al., "**SAPIEN: A SimulAted Part-based Interactive ENvironment**", CVPR 2020

ABC: with Parametrized Curves and Surfaces

- Synthetic, 3D CAD Models
- > 1M Models
- mostly, Mechanical Parts

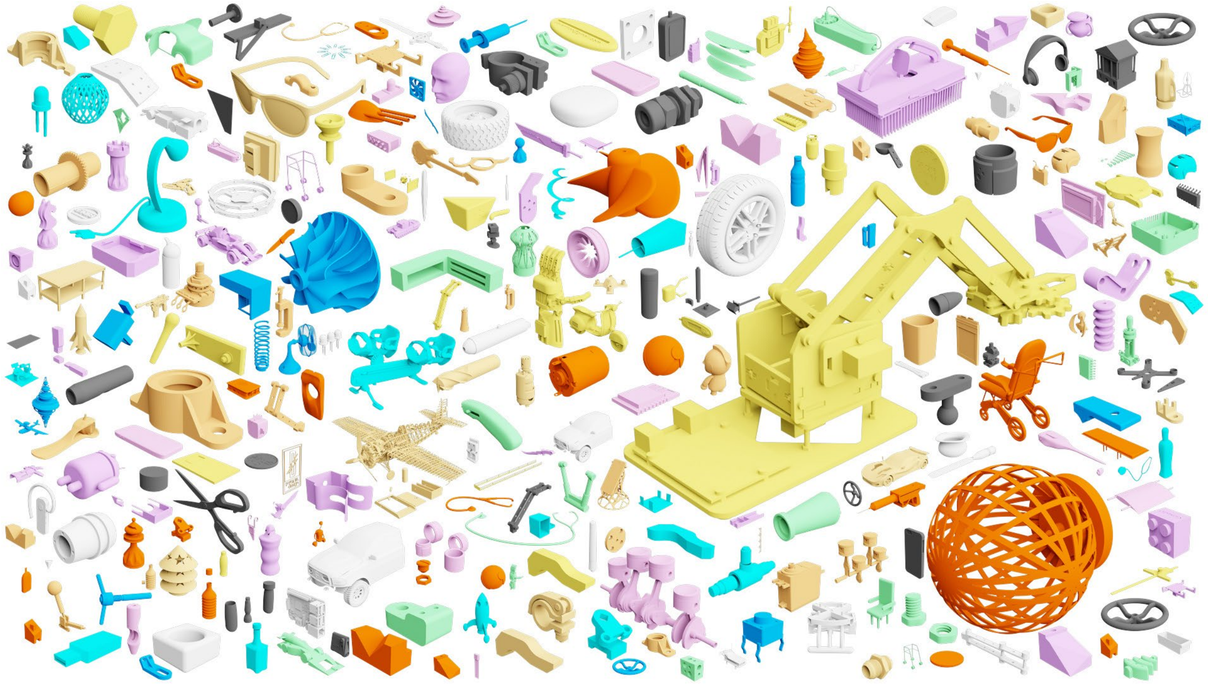
- with Explicitly Parametrized Curves and Surfaces

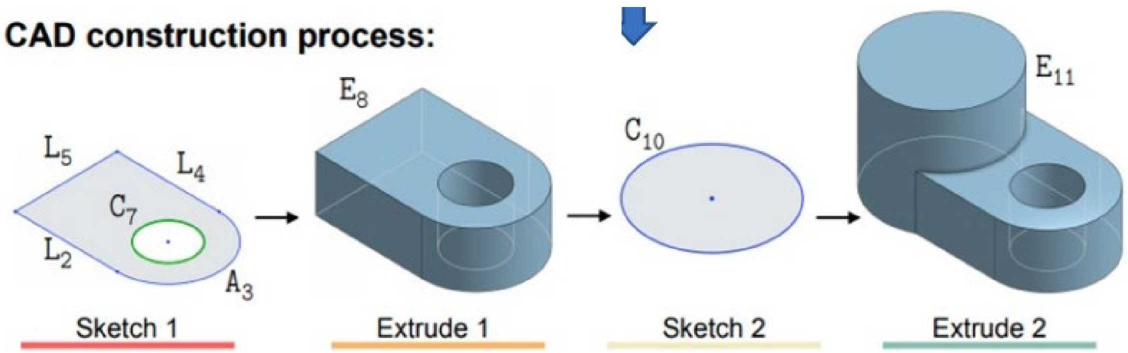
- Normal Estimation
- Surface Parametrization

https://deep-geometry.github.io/abc-dataset/

Sebastian et al., "**ABC: A Big CAD Model Dataset For Geometric Deep Learning**", CVPR 2019

AutoDesk Fusion 360 Gallery Dataset





- Synthetic, 3D CAD Models
- ~ 20K Designs

- Sketch + Extrude
- B-representation

- Reconstruction Dataset
- Segmentation Dataset
- Assembly Dataset

https://github.com/AutodeskAILab/Fusion360GalleryDataset

Willis et al., "**Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences**", ACM Transactions on Graphics (TOG)

CO3D: Common Objects in 3D

CO3D

- Real-world 3D Shape Scans
- 1.5M Multi-view Images
- 19K Objects, 50 Categories

- Novel View Synthesis
- 3D Reconstruction

https://github.com/facebookresearch/co3d

Reizenstein et al., "**Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction**", ICCV 2021

3D-Front: Large-scale Synthetic 3D Scenes

- Synthetic, 3D CAD Models
- from Professional Designers

- 18,797 Rooms
- 7,302 Furniture Objects

- Indoor Scene Synthesis
- Texture Synthesis

https://tianchi.aliyun.com/specials/promotion/alibaba-3d-scene-dataset

Fu et al., "**3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics**", ICCV 2021

ScanNet: Large-scale Real-world 3D Scene Scans

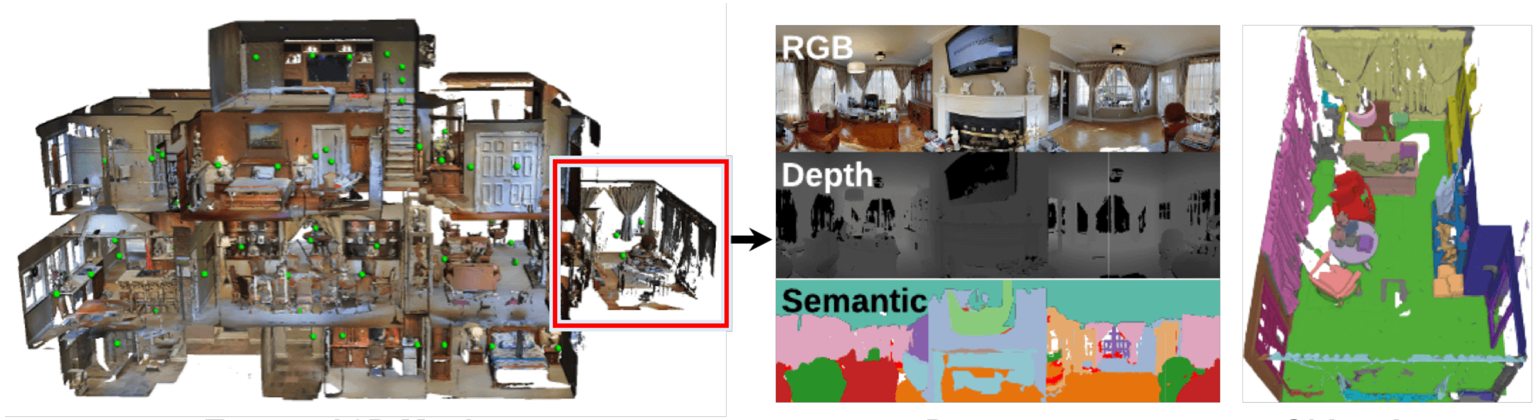
http://www.scan-net.org/

- Real 3D Scene Scans
- RGB-D Videos with 2.5M Views
- 1,500 3D Real Scene Scans

- Semantic/Instance Segmentation
- Object Detection/Classification
- Scene Completion/
  Reconstruction / Generation

Dai et al., "**ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes**", CVPR 2017

Matterport3D and HM-3D: Larger and Largest



Textured 3D Mesh Panoramas Object Instances

- 194,400 RGB-D Images
- 90 Building-scale Real Scans

https://niessner.github.io/Matterport/

Chang et al., "**Matterport3D: Learning from RGB-D Data in Indoor Environments**", 3DV 2017

- 1,000 Building-scale Real Scans (the Largest until now)

https://aihabitat.org/datasets/hm3d/

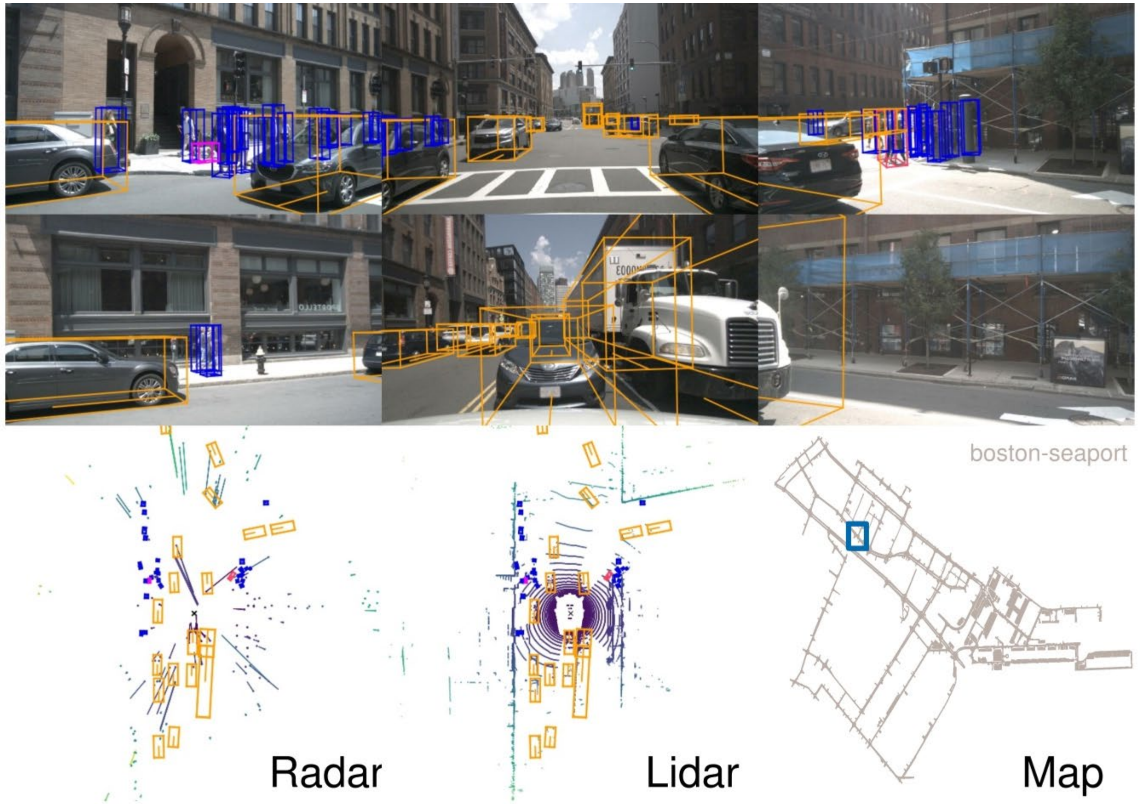
Ramakrishnan et al., "**Habitat-Matterport 3D Dataset HM3D: 1000 Large-scale 3D Environments for Embodied AI**", NeurIPS (dataset) 2021

Kitti and nuScenes: Outdoor Road Scenes for AV

- 389 Images
- 200K Object Annotations

http://www.cvlibs.net/datasets/kitti/

Geiger et al., "**Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite**", CVPR 2012



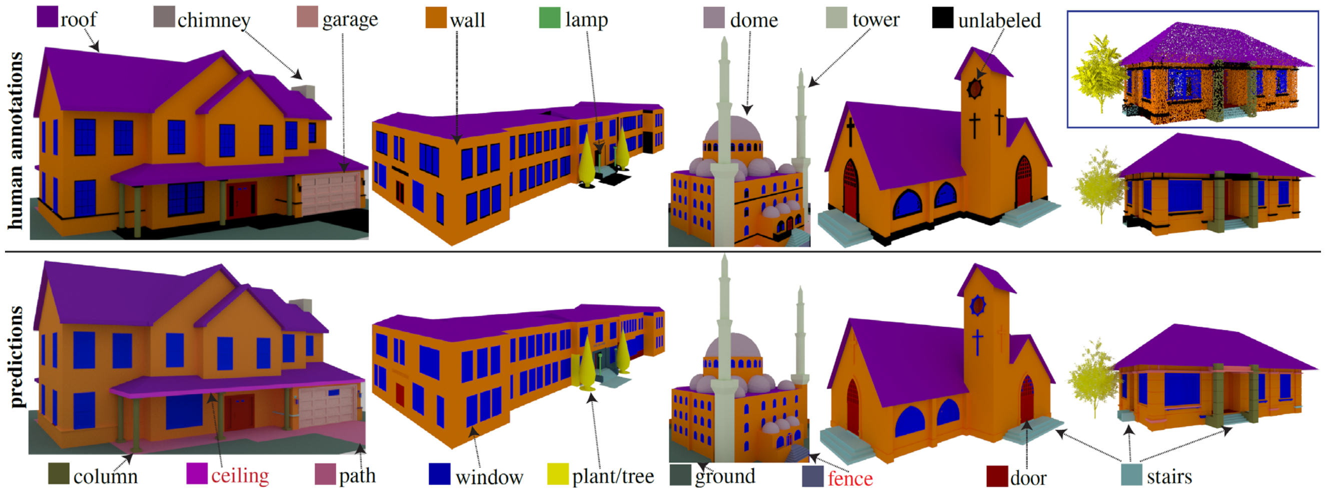
"Ped with pet, bicycle, car makes a u-turn, lane change, peds crossing crosswalk"

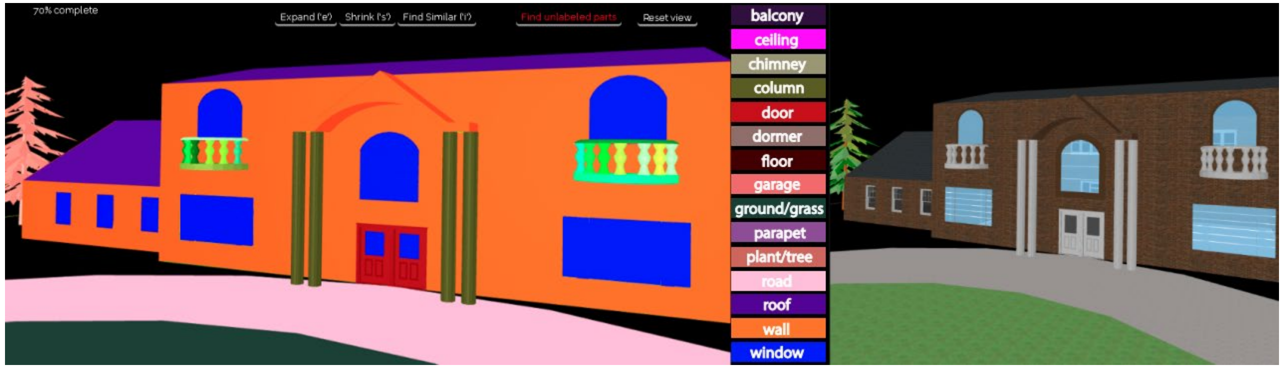
- 1,000 Scenes
- 23 Classes and 8 Attributes

https://www.nuscenes.org/

Caesar et al., "**nuScenes: A multimodal dataset for autonomous driving**", CVPR 2020

BuildingNet: Large-scale 3D Building Models





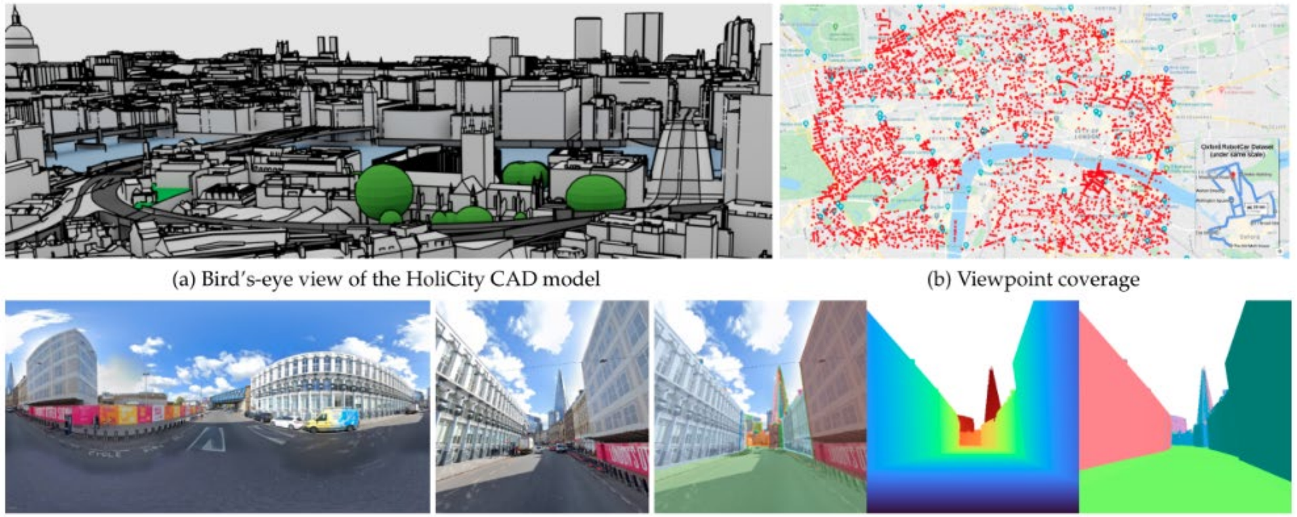
- Synthetic, 3D CAD Models
- 292K Parts, 2K Buildings
- E.g. houses, churches, skyscrapers, town halls, libraries, and castles

- Part Annotations
  - e.g. roof, chimney, wall, lamp
- Edge Annotations
  - e.g. proximity, support, containment

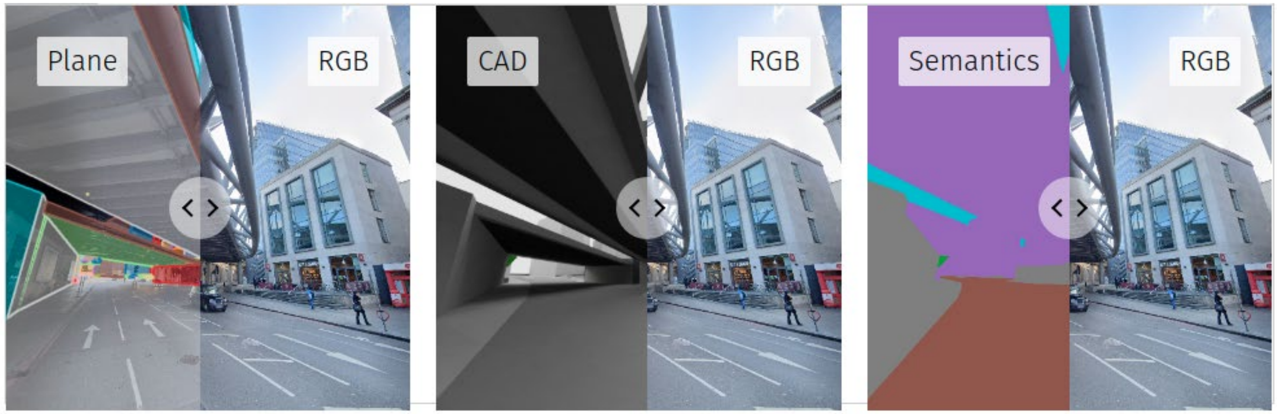
https://buildingnet.org/

Selvaraju et al., “**BuildingNet: Learning to Label 3D Buildings**”, ICCV 2021

HoliCity: A City-scale 3D Dataset



(a) Bird's-eye view of the HoliCity CAD model
(b) Viewpoint coverage
(c) Panorama
(d) RGB
(e) Renderings (surface segments, depth, normal)



https://holicity.io/

- Real-world Scenes in London
- Aligned with 3D CAD Models
- 13312 x 6656 m^2

- 3D Structural Annotations
  - e.g. planes, corners, lines
- Semantic Segmentation

- Support the study of city-scale 3D tasks

Zhou et al., "**HoliCity: A City-Scale Data Platform for Learning Holistic 3D Structures**"

SensatUrban: An Urban-Scale Dataset

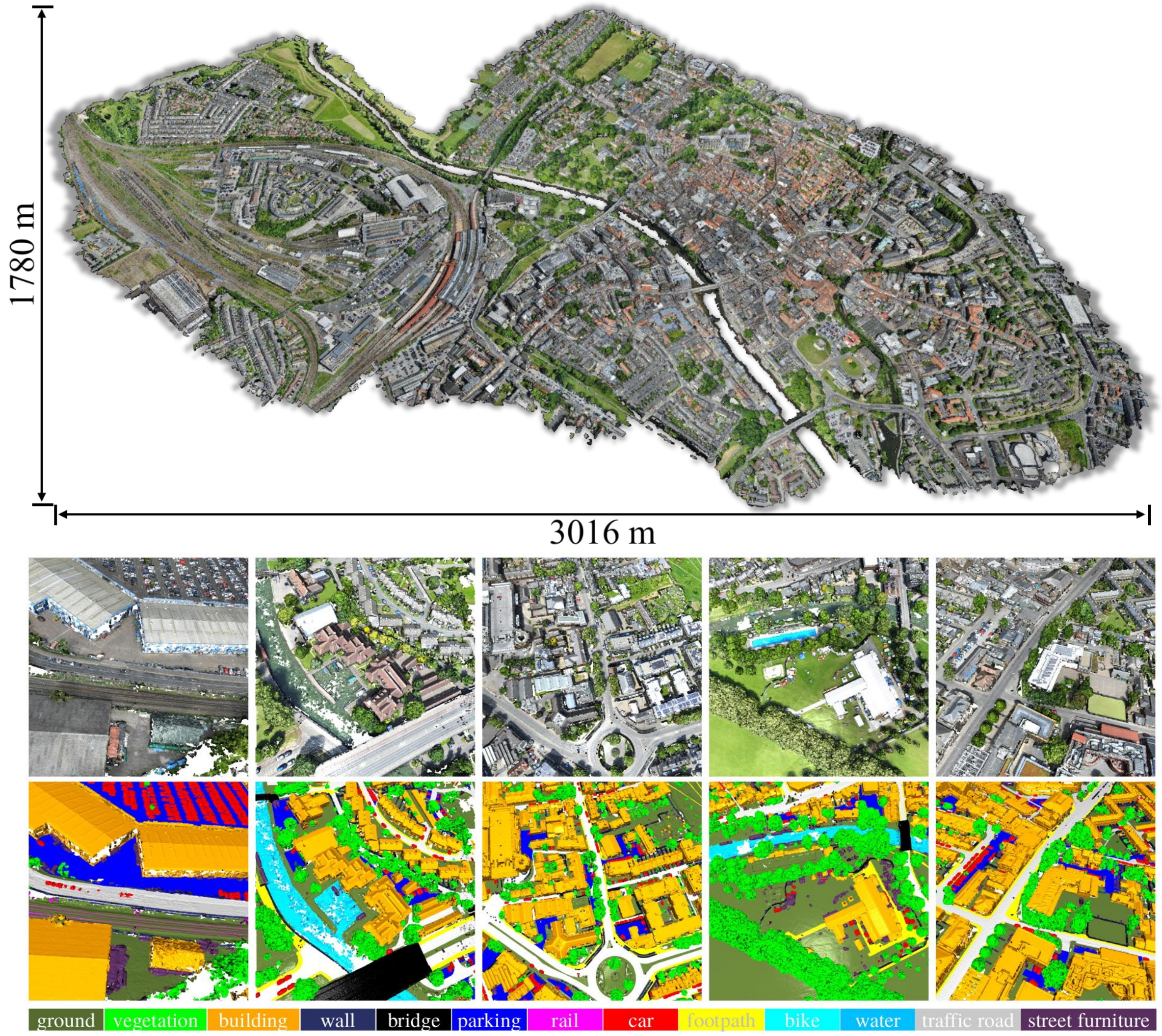


Figure 3: Examples of our SensatUrban dataset. Different semantic classes are labeled by different colors.

https://github.com/QingyongHu/SensatUrban

- 3D Real-world Scans
- 6 km ^ 2 City Landscape
- 13 Semantic Classes
  - e.g. ground, vegetation, car
- Support the study of urban-scale 3D tasks

Hu and Yang et al., "**Towards Semantic Segmentation of Urban-Scale 3D Point Clouds: A Dataset, Benchmarks and Challenges**", CVPR 2021

3D Generation as a Multi-Step Process

Autoregressive Models, PolyGen

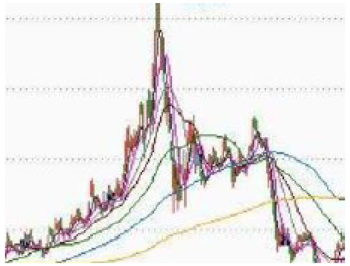
Autoregressive Models

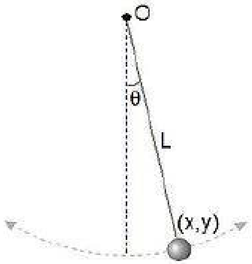
The term ***autoregressive*** originates from the literature on time-series models where observations from the previous time-steps are used to predict the value at the current time step.

Put simply, an autoregressive model is merely a feed-forward model which predicts future values from past values:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p\, y_{t-p} + \varepsilon_t \quad , \quad \varepsilon_t \sim \mathrm{N}(0, \sigma^2)$$

y_i could be:
The specific stock price of day i...
The amplitude of a simple pendulum at period i...
Or any variable that depends on its preceding values!





Autoregressive Models: Factorization

Main challenge: distributions over high dimensional objects is actually very sparse!!

Too many possibilities! ⟶ Main idea: **write as a product of simpler terms**

Definition of conditional probability:

$$P(x_1, x_2) = P(x_1)\,P(x_2|x_1)$$

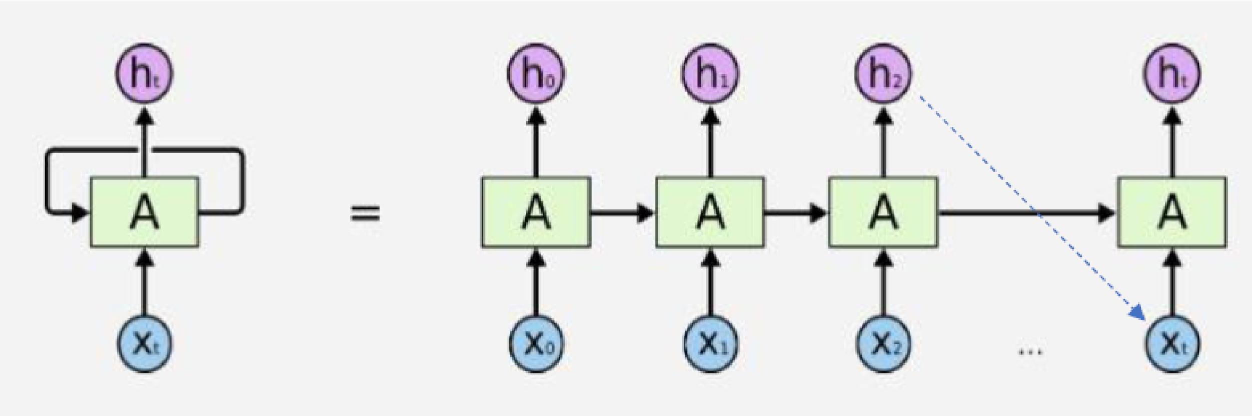
Product rule:

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} p_\theta(x_i|x_{<i})$$

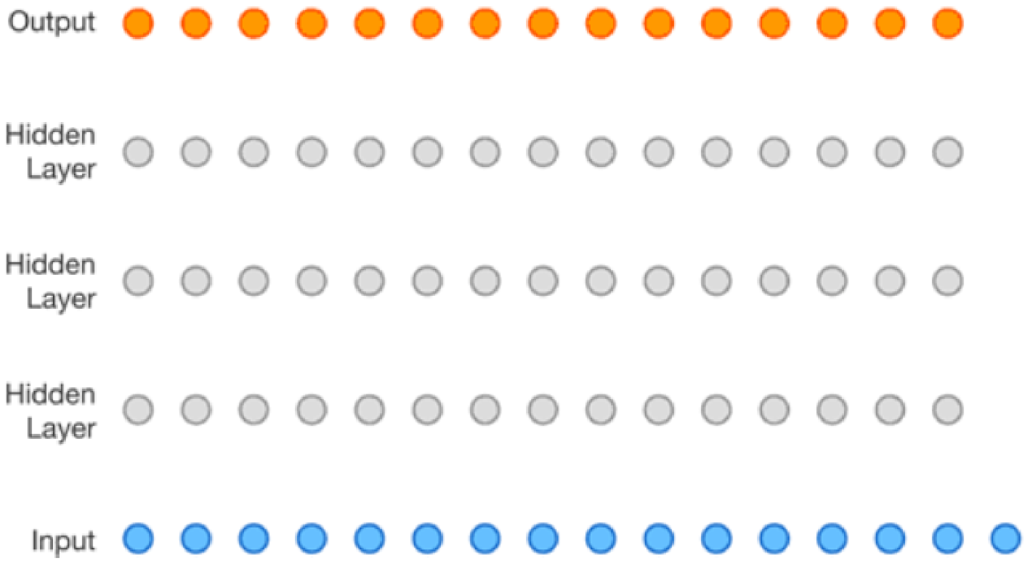
Divide and conquer ! We can solve the joint distribution $P(\boldsymbol{x})$ by solving simpler conditional distributions $p_\theta(x_i|x_{<i})$ one by one

Can you tell the exact likelihood of the next pixel (noted as a red point) conditioned on the given pixels?

Autoregressive models and RNNs



Obligatory RNN diagram. Source: Chris Olah.



WaveNet animation. Source: Google DeepMind.

Relationship with RNN:

Like an RNN, an autoregressive model's output h_t,at time t depends on not just x_t, but also $x_1, x_2, \ldots, x_{i-1}$ from previous time steps.

However, unlike an RNN, the previous $x_1, x_2, \ldots, x_{i-1}$ are not provided via some hidden state: they are given just as an input to the model.

Comparison

Differences between **Autoregressive models (AR), VAE and GAN**:

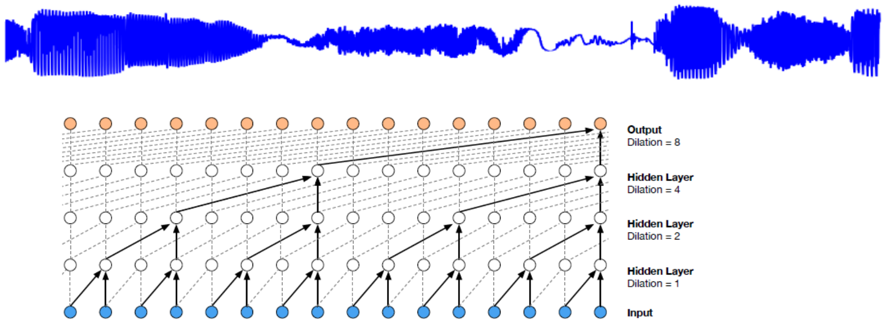
GAN model doesn't define any distribution, it adapts discriminator to learn the data distribution implicitly. $P(X, Z) = P(X|Z)P(Z)$

VAE model believes the data distribution is too complex to model directly, thus it tries to learn the distribution by defining an intermediate distribution and learning the map between the defined simple distribution to the complex data distribution. $P(X, Z) = P(X|Z)P(Z)$

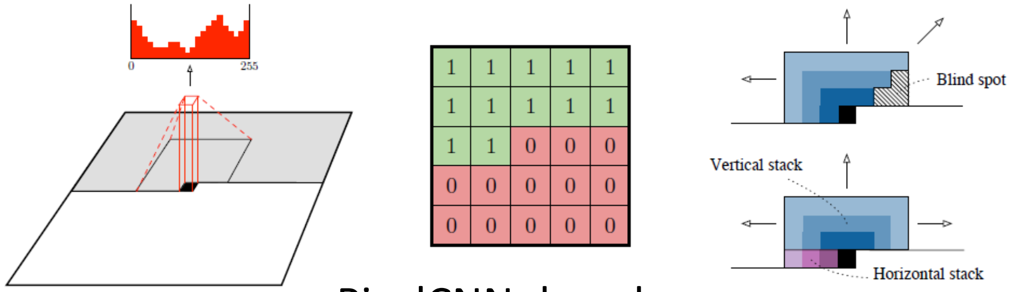
AR model on the one hand assumes that the data distribution can be learned directly (tractable), then it define its outputs as conditional distributions to solve the generation problem by directly modeling each conditional distribution.

Examples

- WaveNet (Deep Mind)



- Conditional Pixel Networks (Deep Mind)



PixelCNN decoders

- Pixel Recurrent Networks (Deep Mind)

Figure 1. Image completions sampled from a PixelRNN.

DeepMind

PolyGen: An Autoregressive Generative Model of 3D Meshes

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, Peter Battaglia

ICML 2020

Generating Virtual Worlds

3D objects populate virtual worlds

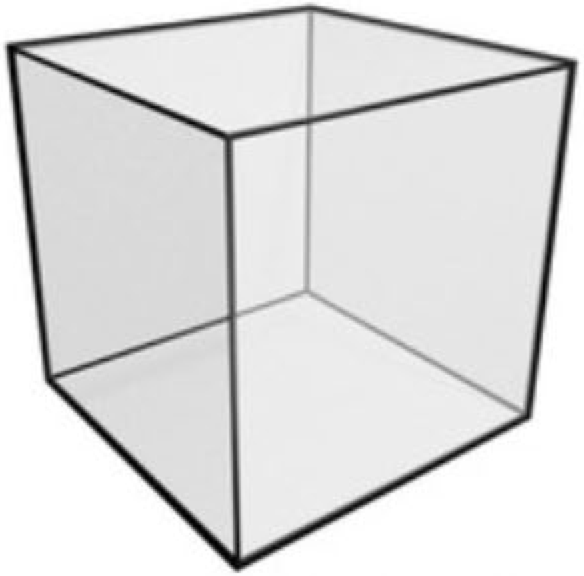
- VR / AR
- Games
- Film / Television
- AI environments

Objects are made out of **meshes**

Generating Virtual Objects

Mesh Representations: OBJ Format

```
# cube.obj
v 1.000000 1.000000 -1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 1.000000 -1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 1.000000
f 1 5 7 3
f 4 3 7 8
f 8 7 5 6
f 6 2 4 8
f 2 1 3 4
f 6 5 1 2
```

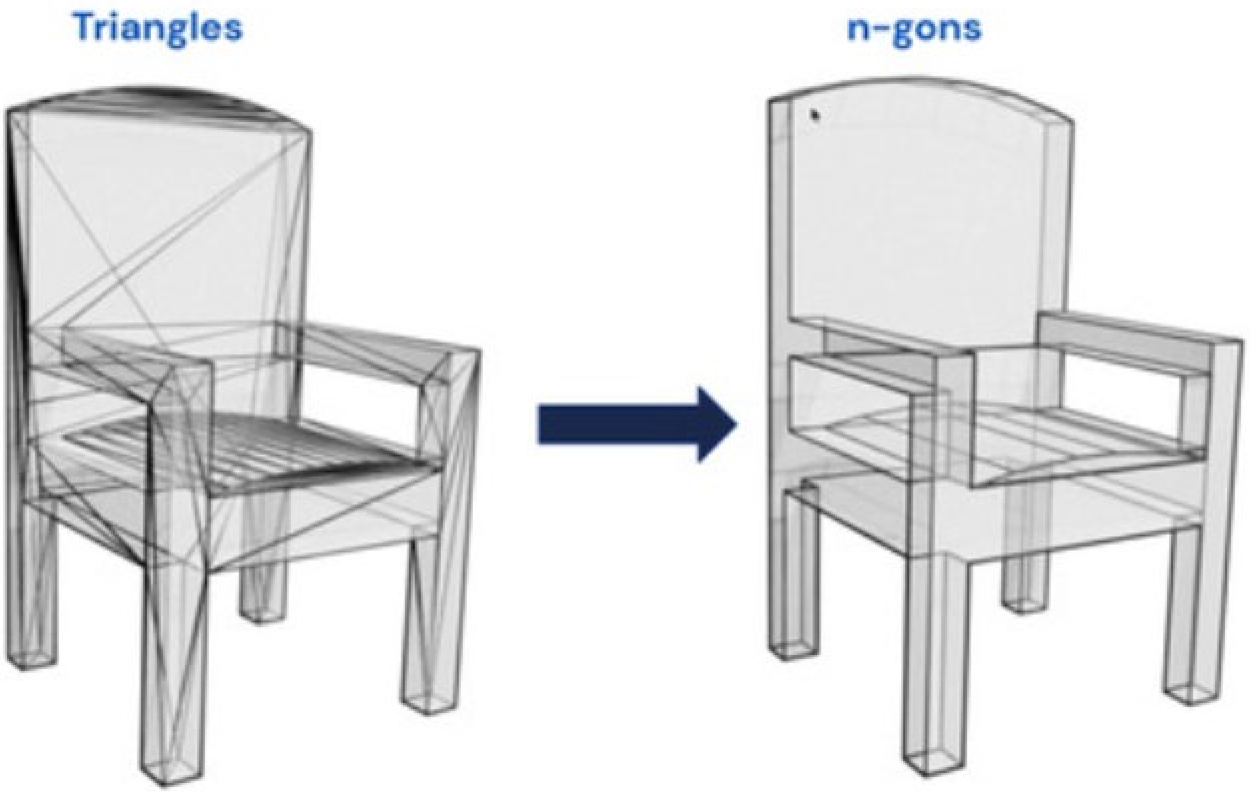


$$\mathcal{M} = (\mathcal{V}, \mathcal{F})$$

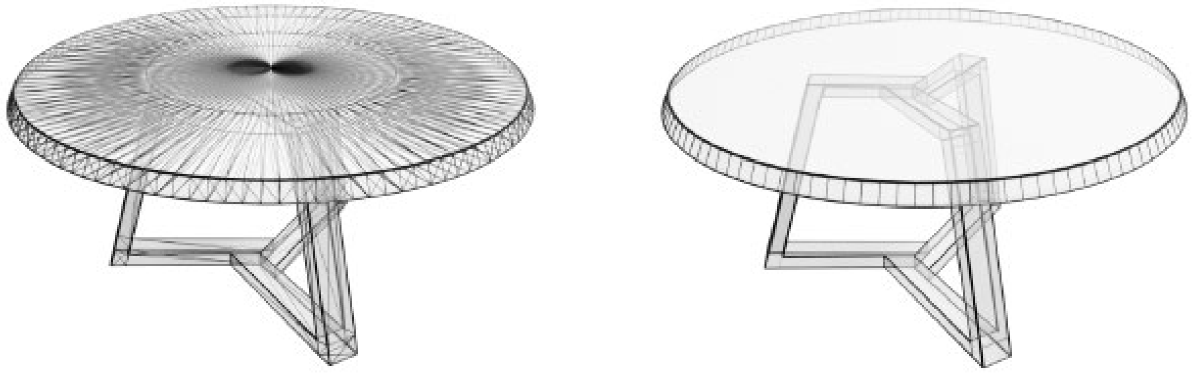
ShapeNet Core DataSet: ~55K Meshes

Higher-Order Elements: N-Gon Meshes

- Meshes can be triangulated in many ways
- N-gons simplify the modeling problem



N-Gons Allow More Efficient Representations



(a) Triangle mesh (b) n-gon mesh

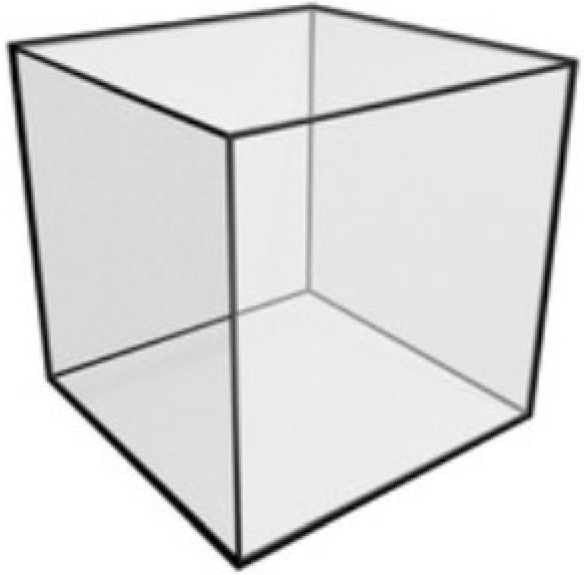
Allows
- fewer elements
- more canonical meshes (easier to learn)
- but, polygons need to be planar

Modeling Strategy

$$p(\mathcal{V}, \mathcal{F}) = p(\mathcal{V})p(\mathcal{F}|\mathcal{V})$$

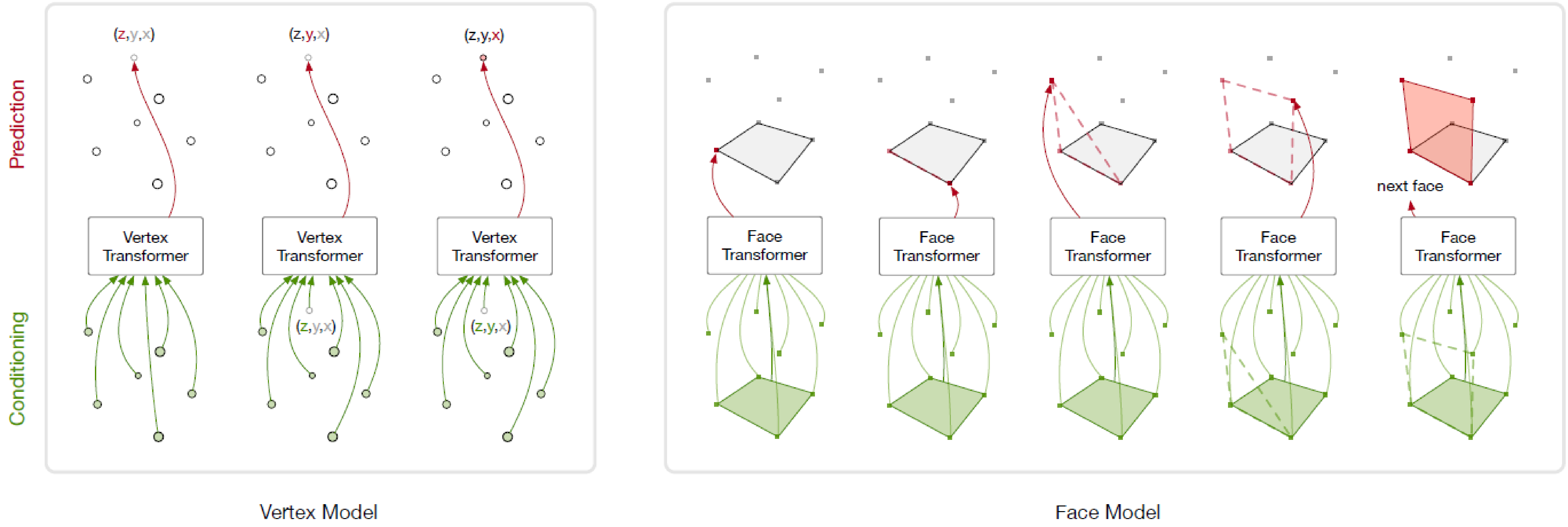
1. Model vertices
2. Model faces given vertices

$p(\mathcal{V})$	Vertex model
$p(\mathcal{F}\|\mathcal{V})$	Face model



$$\mathcal{M} = (\mathcal{V}, \mathcal{F})$$

Modeling Strategy, in More Detail



Must mask invalid predictions

Auto-Regressive Vertex Model

Vertices

$$\mathcal{V} = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_V]$$

Treat as long sequence. **Order** by z-value, then y-value, then x-value

Stopping token

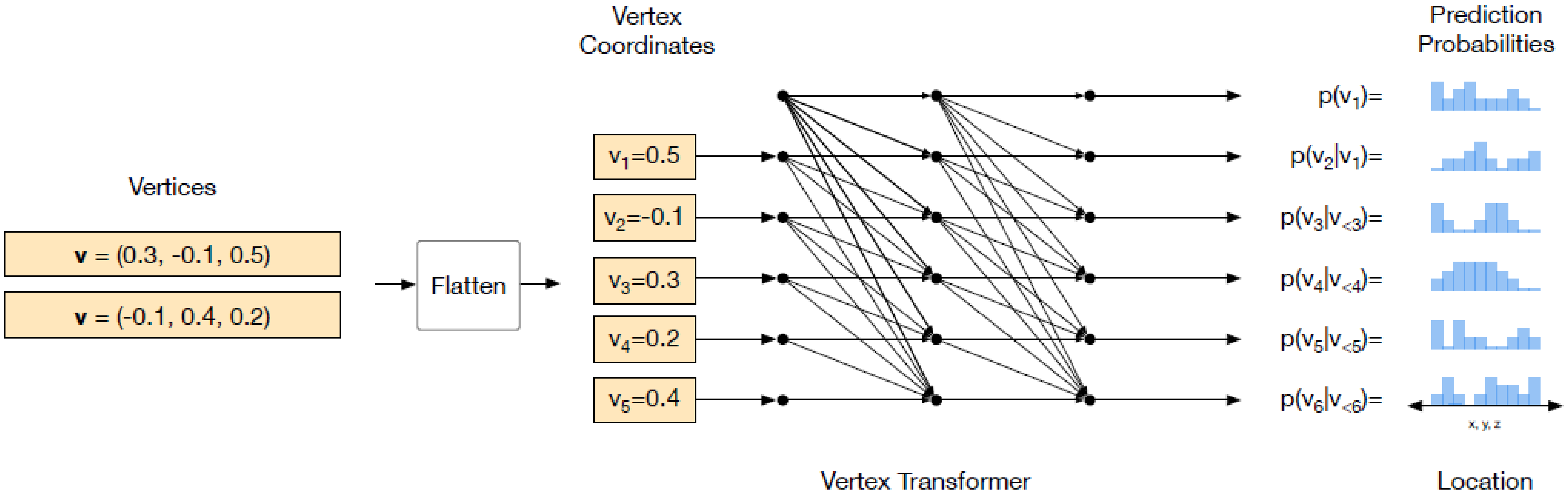
$$\mathcal{V} = [z_0, y_0, x_0, z_1, y_1, x_1, \ldots, z_V, y_V, x_V, s]$$
$$= [v_0, v_1, v_2, \ldots, v_{3V+1}]$$

8-bits

Quantize vertices and predict **softmax** distributions (like WaveNet / PixelCNN) with **autoregressive** architecture

$$p_\theta(\mathcal{V}) = \prod_n p_\theta(v_n | v_{<n})$$

Auto-Regressive Vertex Model



- Outputs **predictive distribution** for sequence of vertex coordinates.
- Train to maximize summed log-probability of sequence (aka cross-entropy loss)

Transformer can learn long-range dependencies: e.g., symmetries

Auto-Regressive Face Model

$$\mathcal{F} = [\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_F]$$

Treat as long sequence with new-face indicators. **Order** by lowest vertex index, then second lowest, etc

New face token

Stopping token

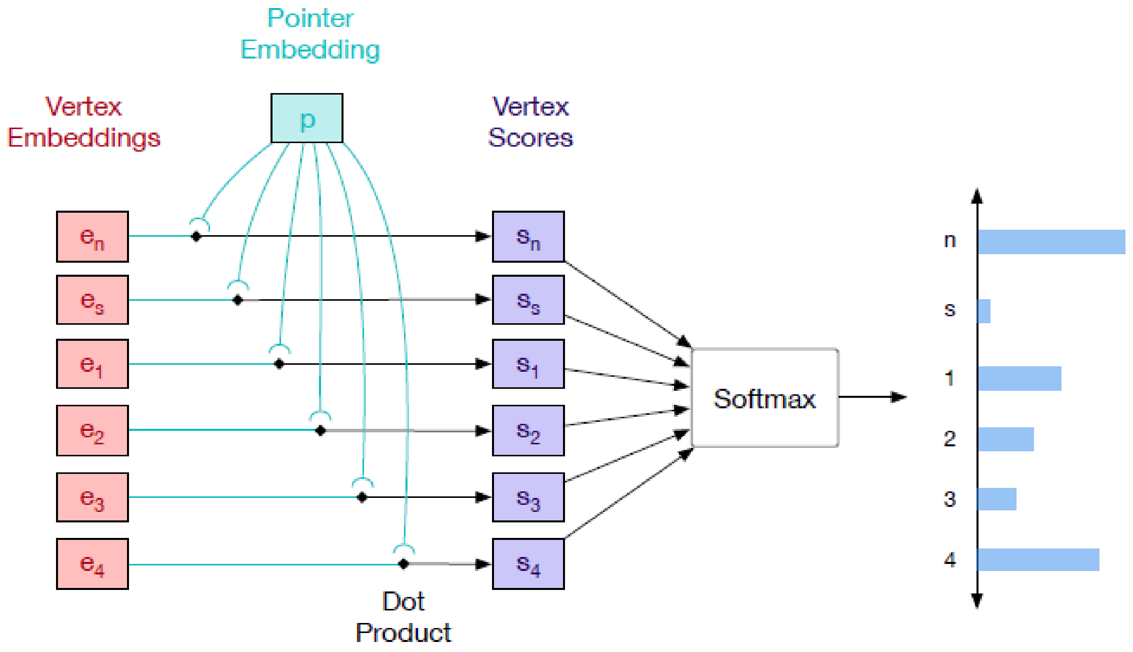
$$\begin{aligned}\mathcal{F} &= [f_0^0, f_1^0, f_2^0, n, f_1^0, f_1^1, f_1^2, \ldots, f_F^0, f_F^1, f_F^2, s] \\ &= [s_0, s_1, s_2, \ldots, s_{N_f}]\end{aligned}$$

Predict **softmax** distribution over vertex indices

$$s \in \{1, 2, \ldots, N_v, n, s\}$$

$$p_\theta(\mathcal{F}|\mathcal{V}) = \prod_n p_\theta(s_n | s_{<n}, \mathcal{V})$$

Pointer Networks



Pointer Networks

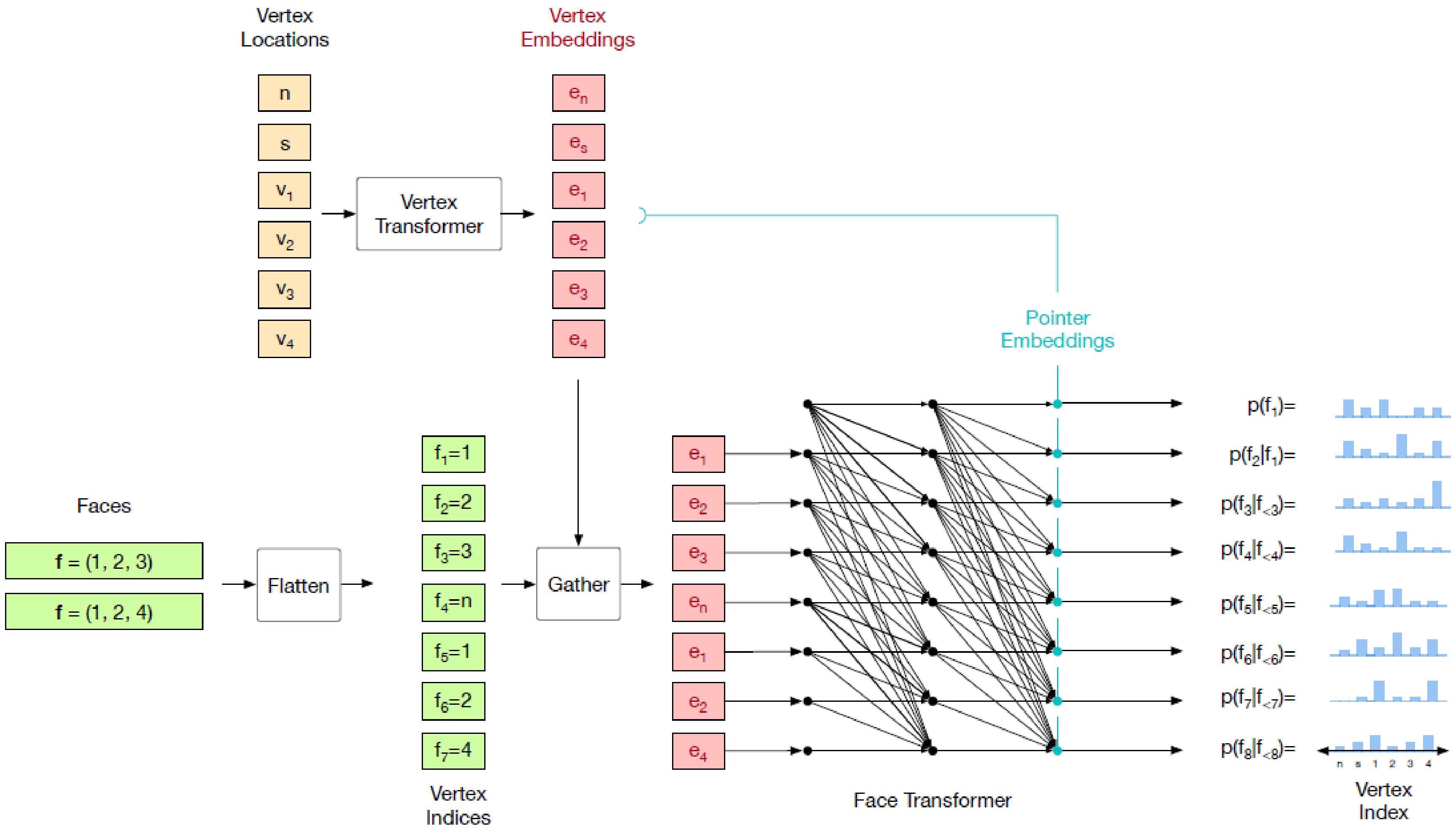
Oriol Vinyals* Google Brain

Meire Fortunato* Department of Mathematics, UC Berkeley

Navdeep Jaitly Google Brain

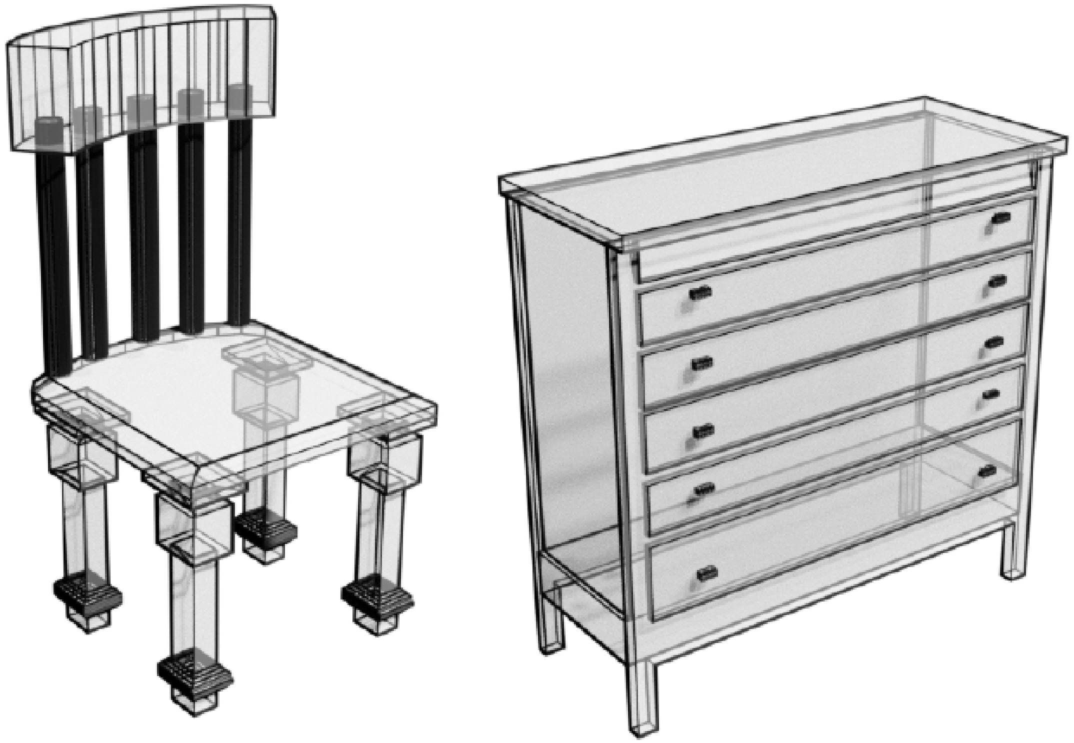
Pointers are compared to vertex embeddings and normalized to obtain a predictive distribution

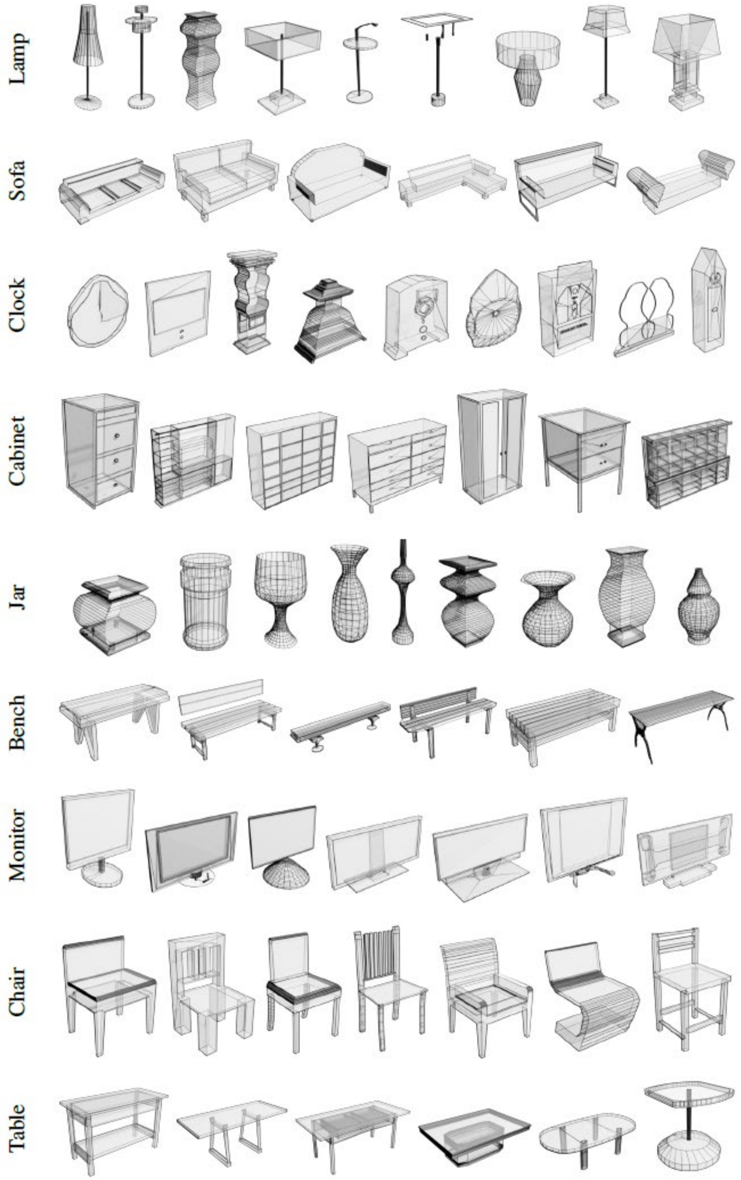
Auto-Regressive Face Model



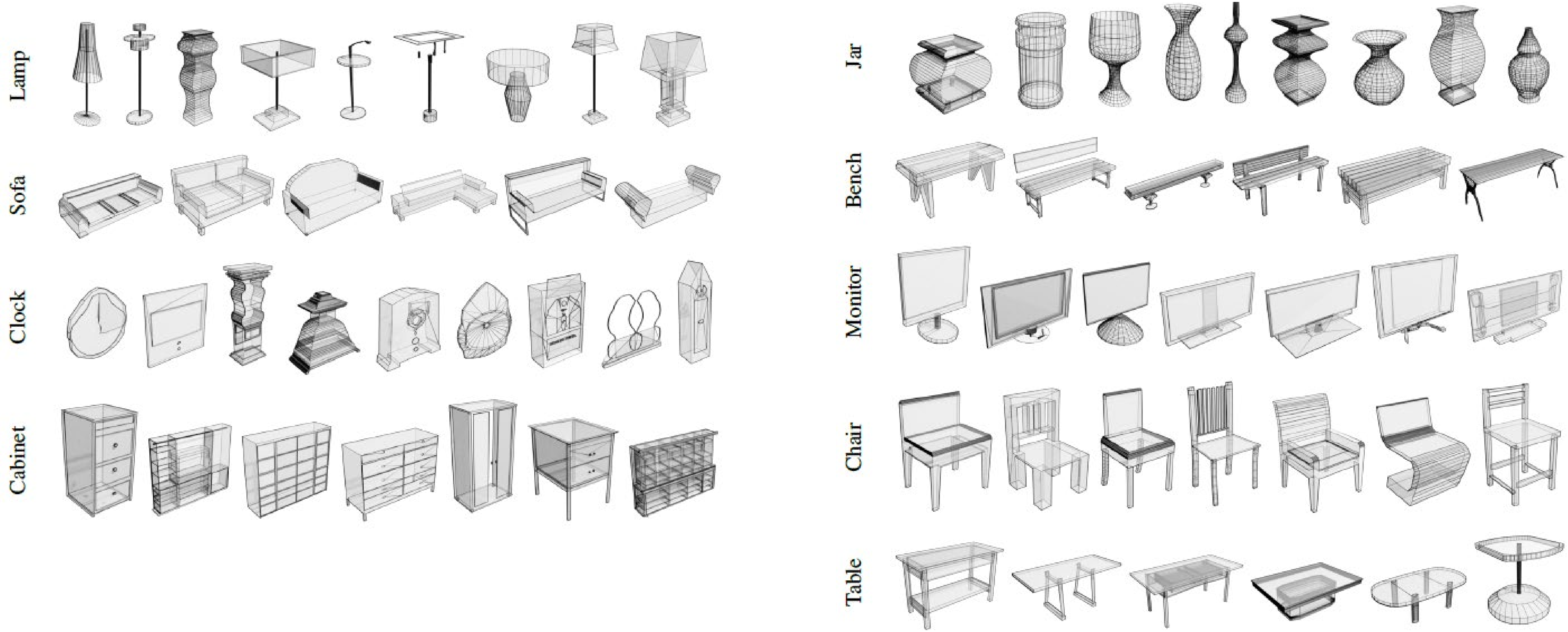
- Vertex encoder produces **contextual vertex embeddings**
- Face model uses vertex embeddings as input -> **outputs pointers**

Results: Conditioning on Class Labels





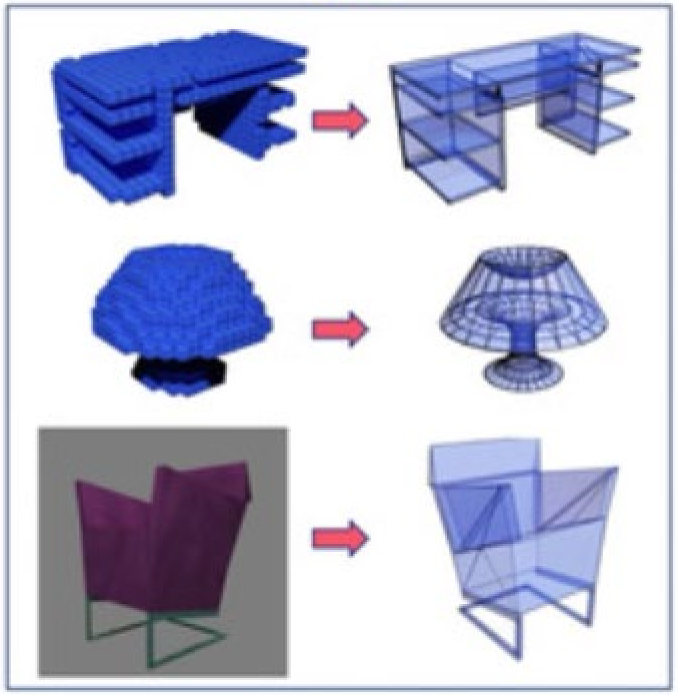
More Examples of Generations



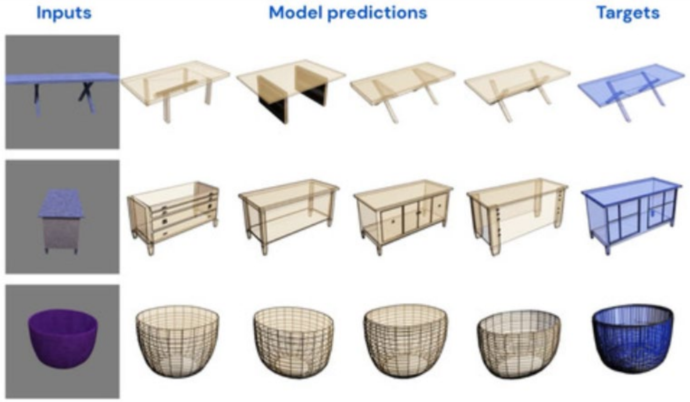
Conditioning on Images and Voxels

- Generate meshes given **voxel** or **Image** inputs
- Enables object design for non-experts (like MineCraft)
- Use **conv-net encoder** and pass embeddings to vertex / face model

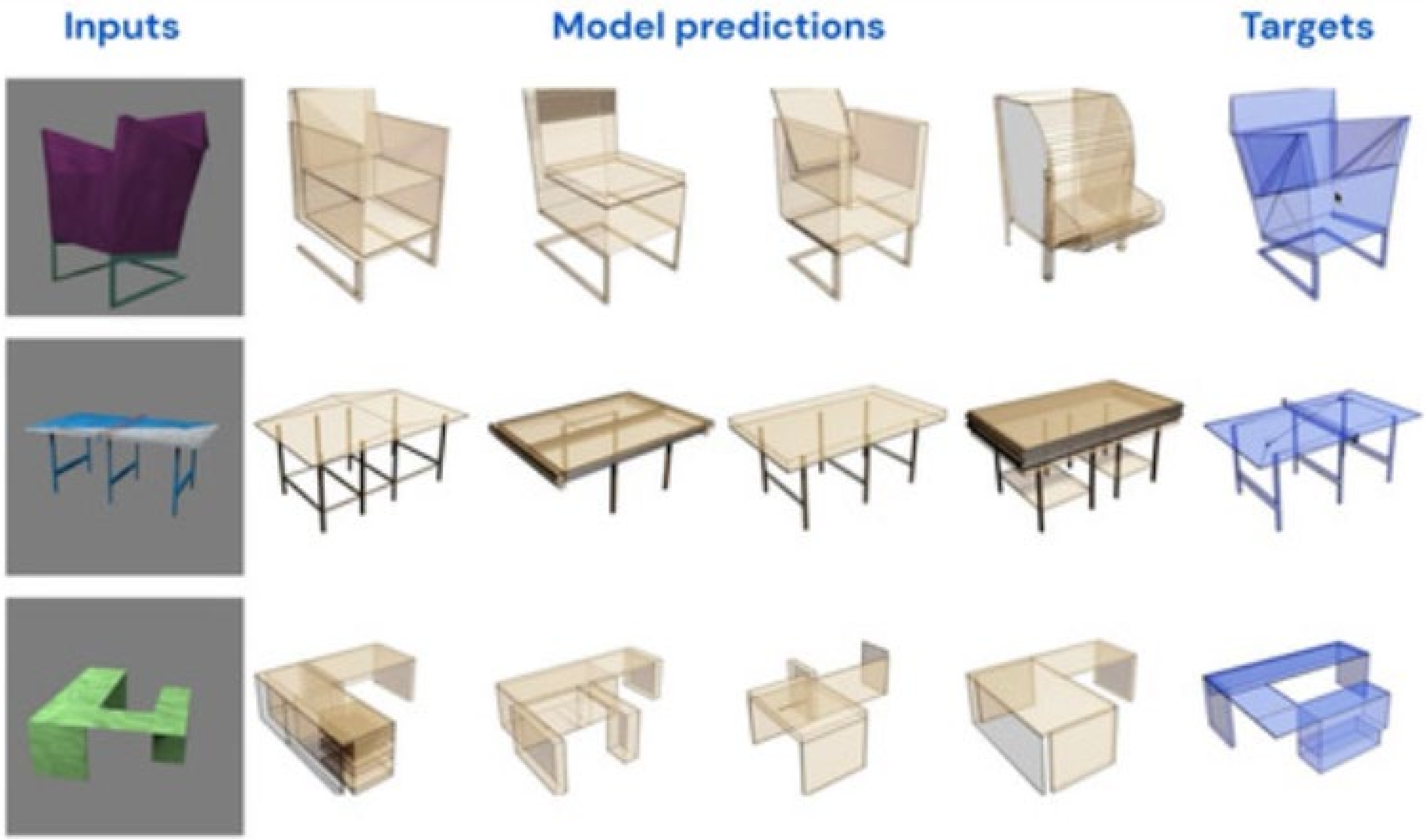
Goal



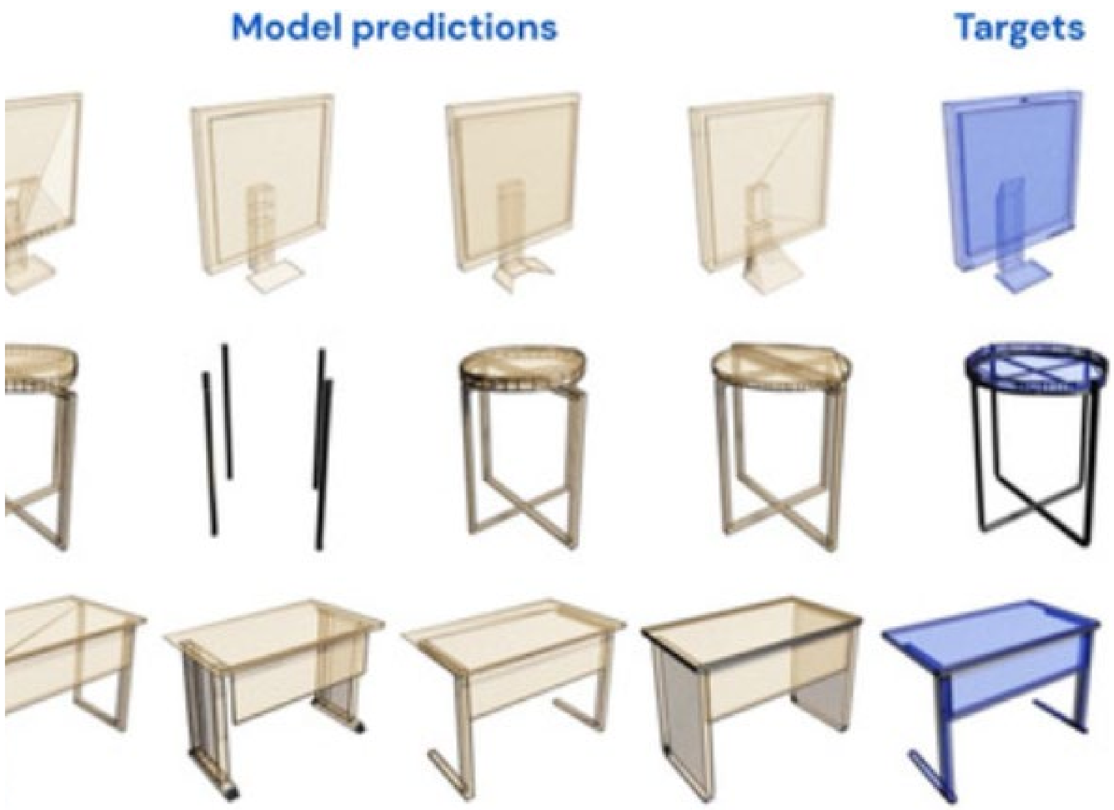
Conditioning on Images



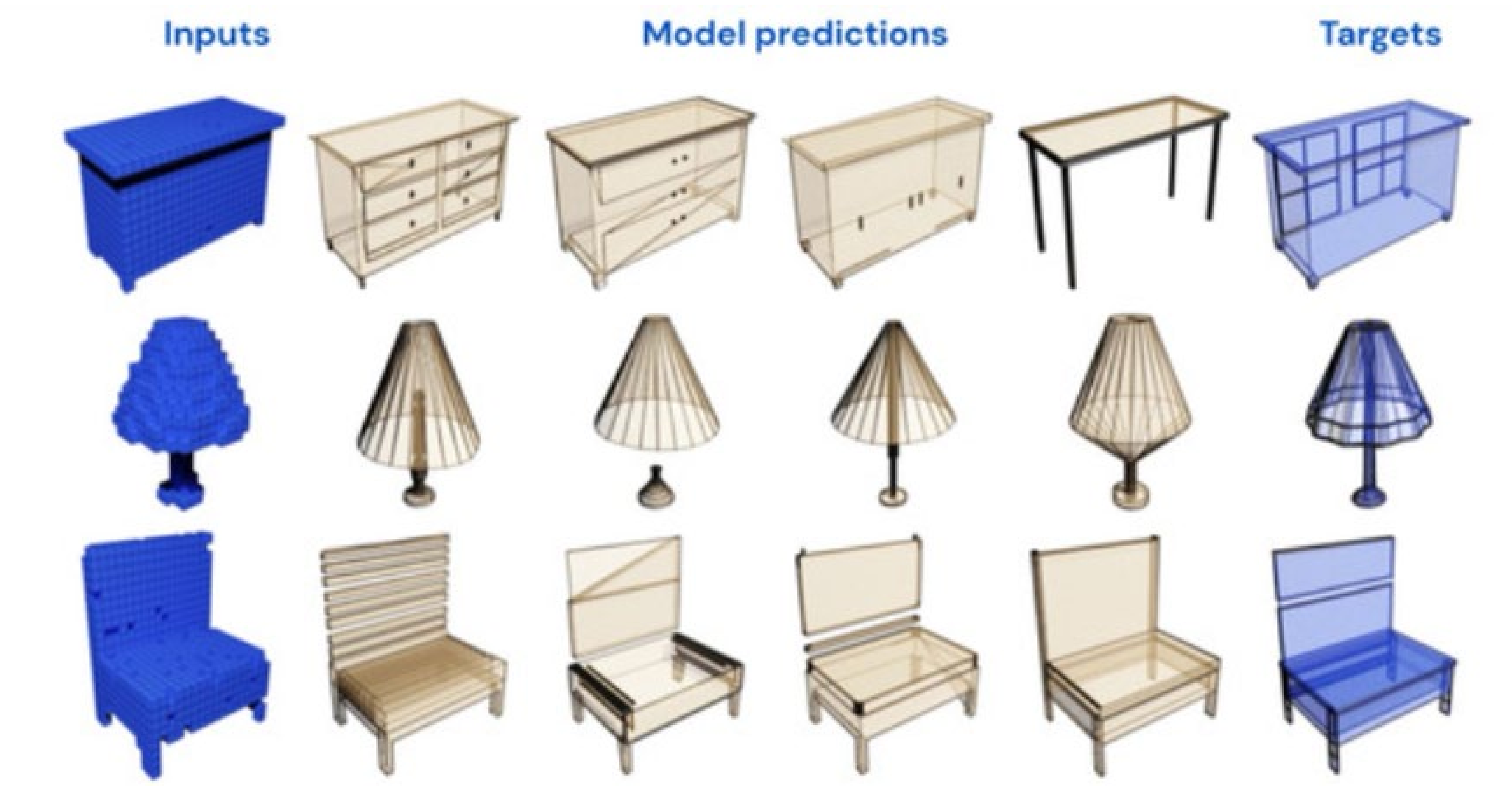
Conditioning on Images



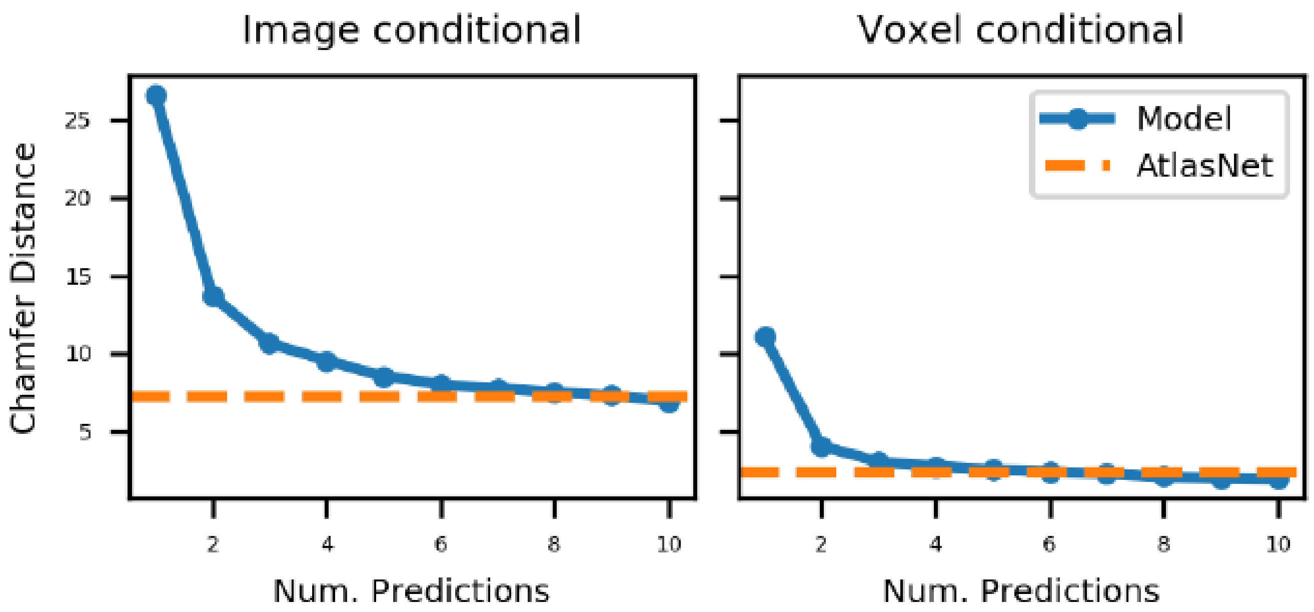
Conditioning on Voxels



Conditioning on Voxels

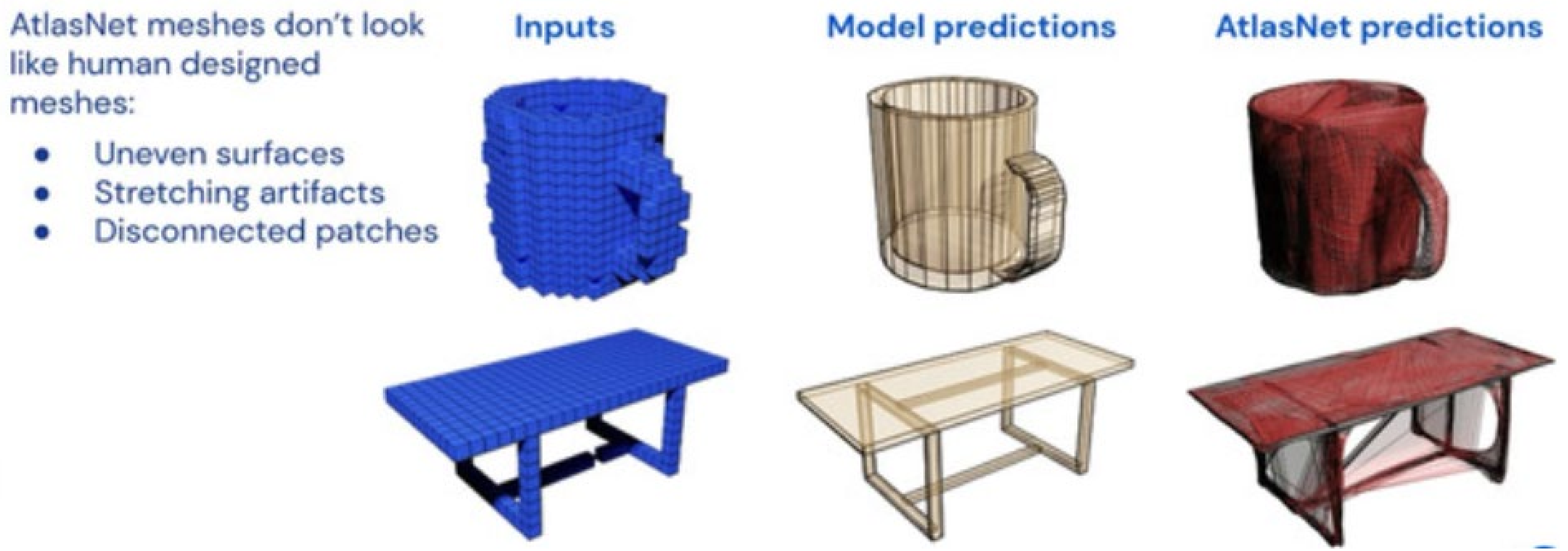


Conditional Generation Evaluation: Chamfer Dist



- AtlasNet wins on a single prediction
- But PolyGen catches up with more predictions

Is Chamfer the Best Metric?



Optimizing Chamfer Can Lead to Noisy Meshes

(a) PolyGen (b) Occupancy Networks

PolyGen Summary

- Mesh generation using autoregressive models is **feasible**
- Directly modeling human-designed meshes means we can output diverse and realistic meshes
- Challenges remain in **scaling** to larger meshes, and due to availability of **large datasets**

Flow Models, PointFlow

PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows. Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, Bharath Hariharan. ICCV'19

That's All