# Geometry Processing and Geometric Deep Learning

MVA Course, Lecture 3, part 1

16/ 10 / 2024

Maks Ovsjanikov

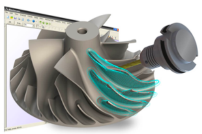
# My Background

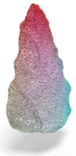
## Positions:

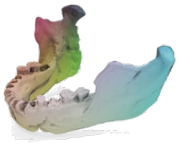
2018 – Full Professor, GeomeriX team, Ecole Polytechnique
2024 – Visiting Researcher, Google DeepMind, Paris

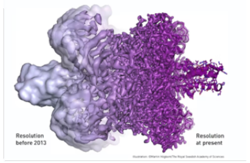
## Research Profile:

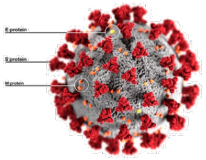
Analysis and Deep Learning on 3D data, *classification*, *segmentation*, *correspondence*, *reconstruction*, *alignment*, etc.







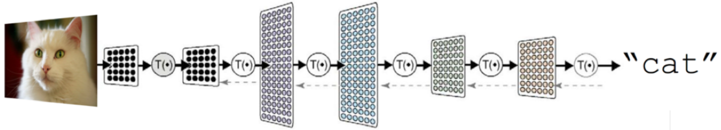


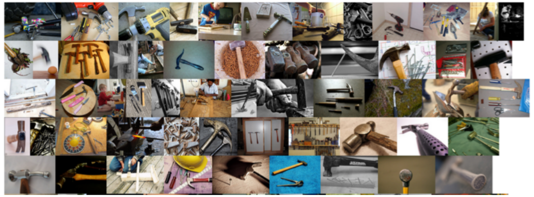
# Today: Deep Learning on 3D shapes

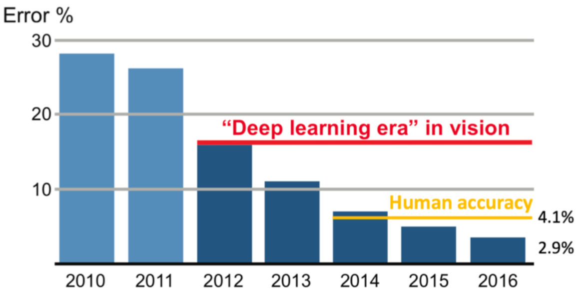
- Recap of CNNs and their properties
- Multi-view, extrinsic, projection-based approaches
- Spectral methods, pros and cons
- Intrinsic approaches
- Learning via diffusion

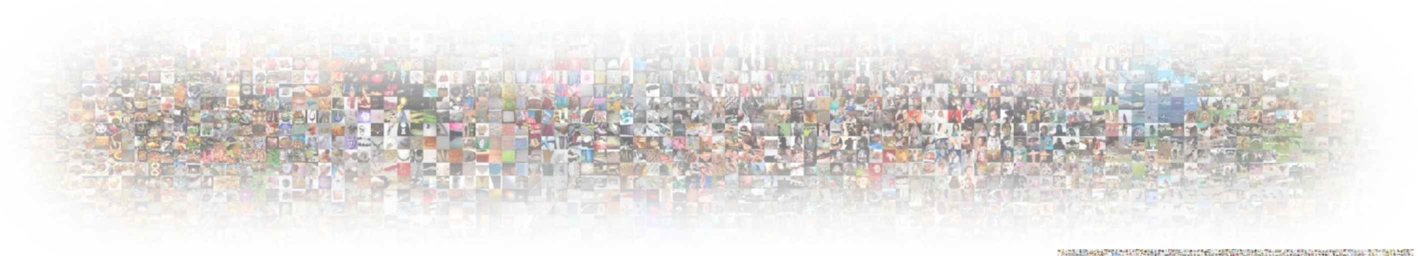
# Learning on images




Test images for "Hammer"





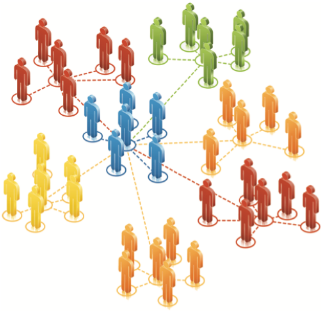



# Learning on different data

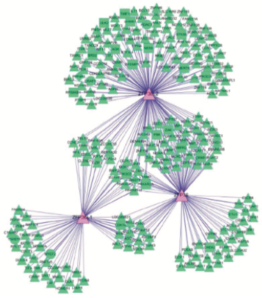
Doubt thou the stars are fire;
Doubt that the sun doth move;
Doubt truth to be a liar;
But never doubt I love...

Text

Audio signals

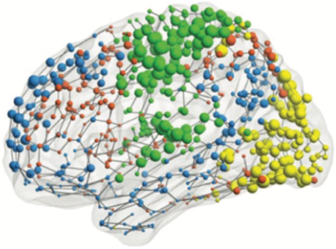


Social networks

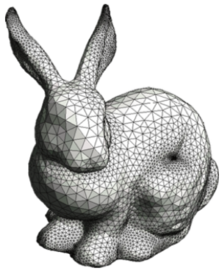


Regulatory networks

Images



Functional networks

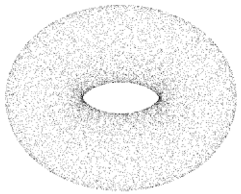


3D shapes

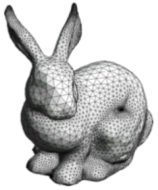
# Deep Learning for 3D shapes

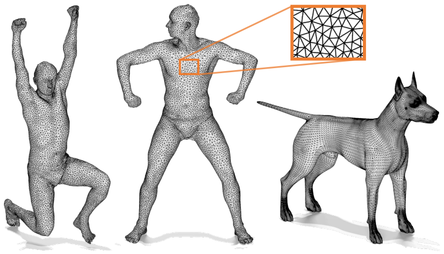
- Main Challenge

3D shapes (typically) do not have a canonical (grid-like) representation!



**3D point cloud:** an *unorganized collection of 3D coordinates*





**3D mesh:** a *collection of points* and *triangles* connecting them.

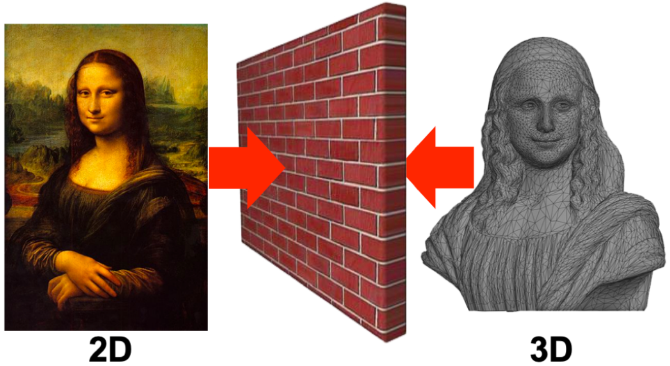
**Main question:**

How to design neural networks on *unstructured domains* such as 3D shapes?

# Domain vs data on a domain

Images are grids of pixels!

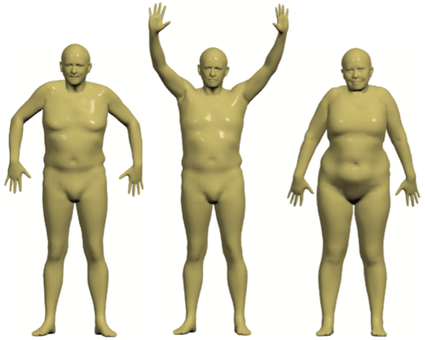
# Domain vs data on a domain




In computer vision images are *data* on a fixed domain (grid of pixels).
In 3D, $h$ (the shape) is what we're learning on.

# Fixed vs different domains

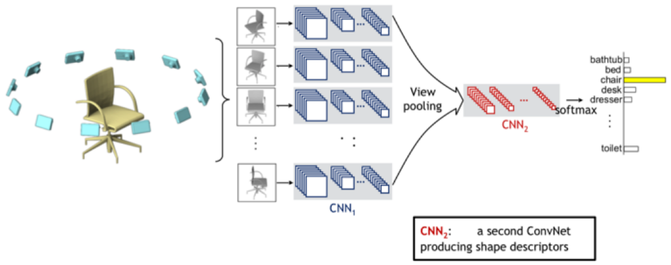
Social network
(fixed graph)



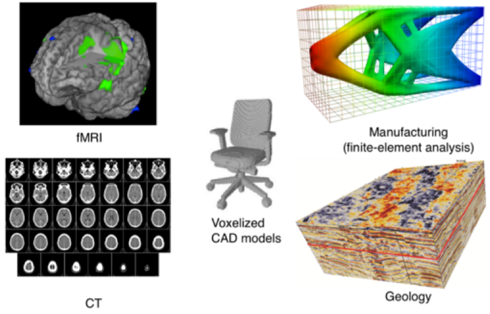
3D shapes
(different manifolds)

- In graph neural networks, we often deal with problems on a fixed domain (e.g., predicting communities).
- In 3D (and other fields), we typically want the solutions to *generalize to unseen domains/shapes.*

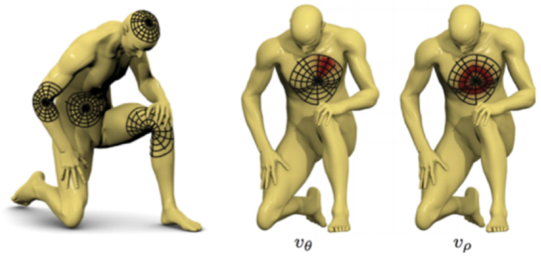
# Approaches for 3D Deep-Learning



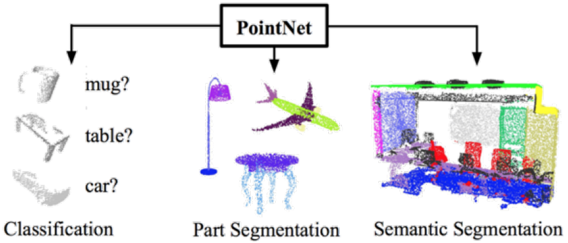

Multi-view based




Volumetric




Intrinsic (surface-based)




Point-based

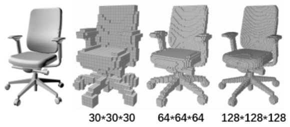
# 3D Shape Analysis and Learning

**Main questions:**

- How to *represent* the 3D shapes to enable learning?
- How to design *robust* and *principled* data analysis approaches?

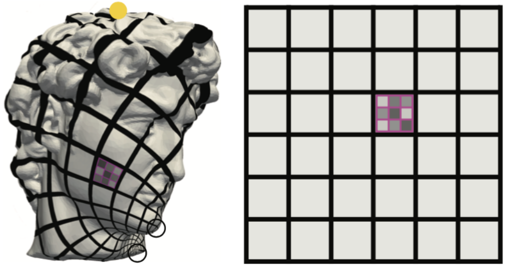
# Some approaches covered today

Image-based



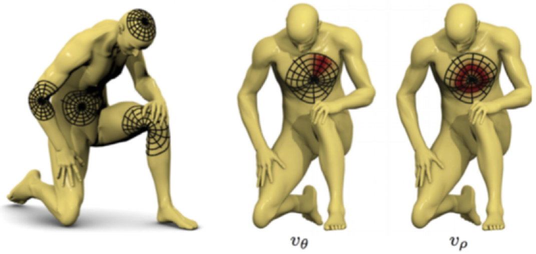

Voxel-based

Extrinsic



Embedding domain

Spectral domain




Intrinsic (surface-based)

# Background

# Supervised learning

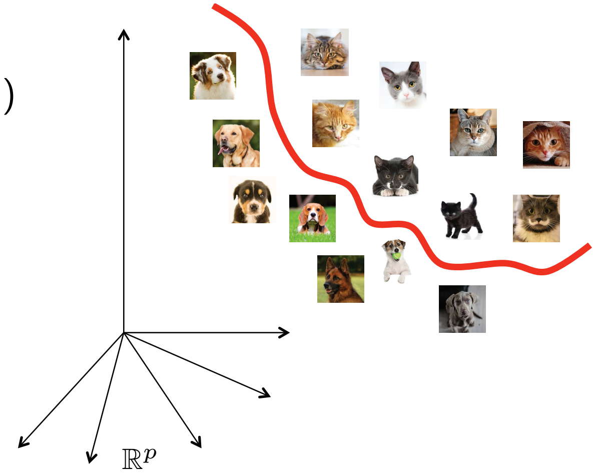
- Data vectors $\mathbf{f} \in \mathbb{R}^p$
  (e.g. for 512×512 images $p \approx 10^6$)

- Unknown classification functional
  $y : \mathbb{R}^p \to \{1, \ldots, L\}$ in $L$ classes

- Training set

$$S = \{(\mathbf{f}_i \in \mathbb{R}^p,\, y_i = y(\mathbf{f}_i))\}_{i=1}^{T}$$

- Parametric model $y_{\boldsymbol{\Theta}}$ of $y$

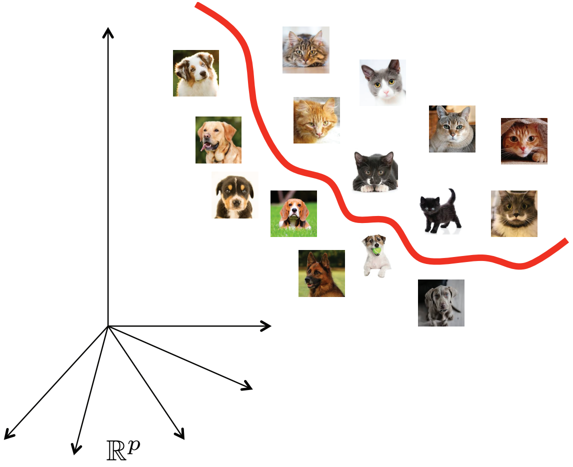


# Supervised learning

- Data vectors $\mathbf{f} \in \mathbb{R}^p$
  (e.g. for 512×512 images $p \approx 10^6$)
- Unknown classification functional
  $y : \mathbb{R}^p \to \{1, \ldots, L\}$ in $L$ classes
- Training set

$$S = \{(\mathbf{f}_i \in \mathbb{R}^p,\, y_i = y(\mathbf{f}_i))\}_{i=1}^{T}$$

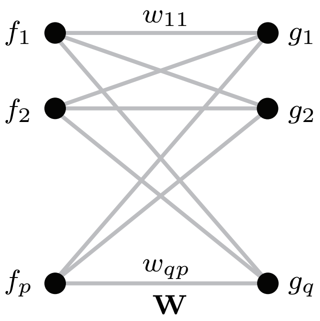
- Parametric model $y_{\boldsymbol{\Theta}}$ of $y$




Supervised learning: find optimal model parameters by minimizing the loss $\ell$ on the training set

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{i=1}^{T} \ell(y_{\boldsymbol{\Theta}}(\mathbf{f}_i), y_i)$$

# Basic Neural Network (NN)



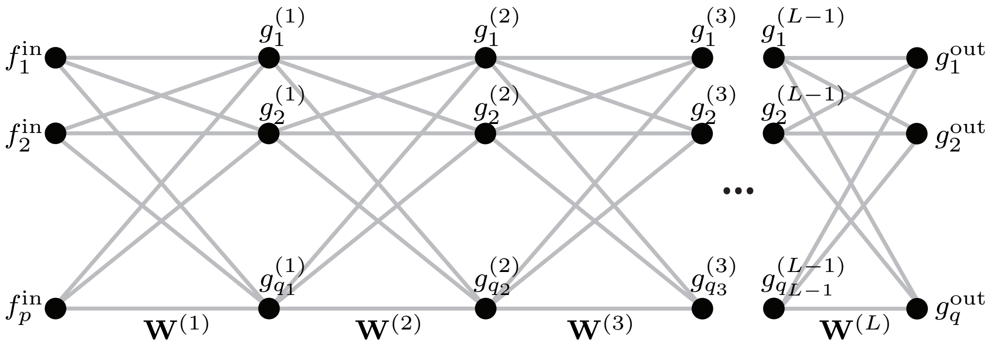

Single linear layer

Linear layer $$g_l = \xi\left(\sum_{l'=1}^{p} f_{l'} w_{l,l'}\right) \qquad \begin{array}{l} l = 1, \dots, q \\ l' = 1, \dots, p \end{array}$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters layer weights $\mathbf{W}$ (including bias)

# NN = Multi-Layer Perceptron (MLP)




Deep neural network consisting of $L$ layers

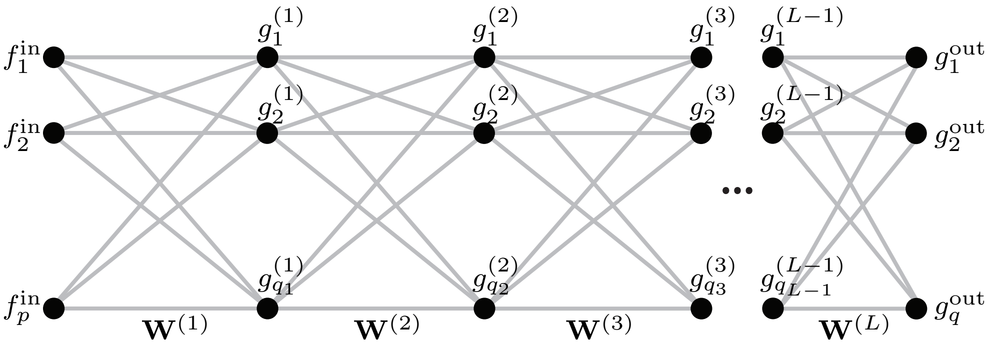
Multi-Layer Perceptron (**MLP**)

Linear layer $\quad \mathbf{g}^{(k)} = \xi\left(\mathbf{W}^{(k)}\mathbf{g}^{(k-1)}\right)$

Activation, e.g. $\quad \xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Parameters $\quad$ weights of all layers $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}$ (including biases)

# NN = Multi-Layer Perceptron (MLP)




Deep neural network consisting of $L$ layers

Net output $\quad \mathbf{g}^{\text{out}} = \xi\left(\dots \mathbf{W}^{(2)}\xi\left(\mathbf{W}^{(1)}\mathbf{f}^{\text{in}}\right)\right) = y_{(\mathbf{W}^{(1)},\dots,\mathbf{W}^{(L)})}(\mathbf{f}^{\text{in}})$

Activation, e.g. $\quad \xi(x) = \max\{x, 0\}$ $\quad$ rectified linear unit (ReLU)

Parameters $\quad$ weights of all layers $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}$ (including biases)

# Neural Network Expressive Power

**Universal Approximation Theorem** Let $\xi$ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$

Cybenko 1989; Hornik 1991

# Neural Network Expressive Power

**Universal Approximation Theorem** Let $\xi$ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0, 1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ (including bias) s.t.

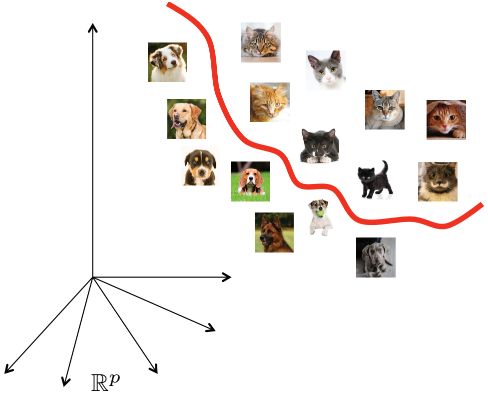
$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f}) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0, 1]^p$$

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How to find the parameters?
- ☹ Does it generalize well / overfit?
- ☹ Shallow nets work poorly for high-dimensional data s.a. images

Cybenko 1989; Hornik 1991

# Neural Network Expressive Power

- Applying MLPs directly on the input data is usually too inefficient !
  - An RBG image of size 512x512 leads to input size: $f_{in}$ = 512x512x3 ≈ **$10^6$** nodes.
  - If $f_{out} = f_{in}$ then a *single-layer* MLP would have ≈ **$10^{12}$** trainable parameters!




- Need to exploit *structure in the data*!

# Translation invariance for images


$$y(\mathcal{T}_v f) = y(f) \quad \forall f, v$$

where

- image is modeled as a function $f \in L^2([0,1]^2)$
- $\mathcal{T}_v f(x) = f(x - v)$ is a translation operator
- $v \in [0,1]^2$ is a translation vector
- $y : L^2([0,1]^2) \to \{1, \ldots, L\}$ is classification functional

# Key Properties of Convolution

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

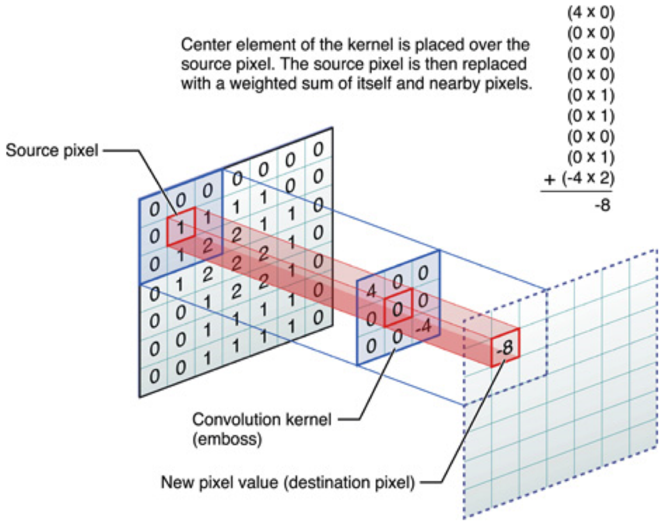
Often we call $f$ the signal and $g$ a kernel (or filter).

Key properties of convolution:

- **Linearity:** $(f \star (\alpha_1 g_1 + \alpha_2 g_2)) = \alpha_1 (f \star g_1) + \alpha_2 (f \star g_2)$
- **Shift equivariance:** $\tau_y (f \star g) = (\tau_y f) \star g = f \star (\tau_y g)$ where $(\tau_y f)(x) = f(x - y)$
- **Theoretical result:** any linear shift-invariant operator in Euclidean space can be represented as a convolution[1].

[1]L. Hörmander, *Estimates for translation invariant operators in $L_p$ spaces*, Acta Math., 1960.

# Convolutional kernel




Replace general MLP weights with convolutional kernels

**Key idea: use a *learned* kernel**

# Convolutional layer

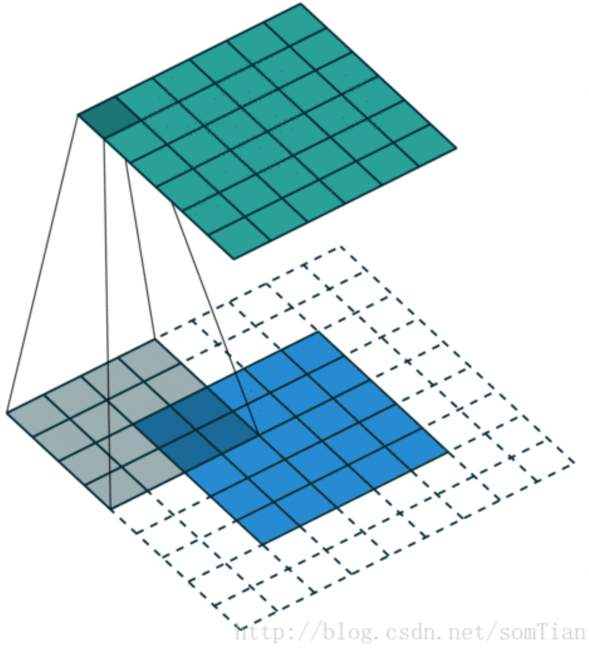
$$(f * g)[m, n] =$$

$$= \sum_{k,l} g[k, l] f[m + k, n + l]$$

Gray: Learned kernel

Blue: Input image (layer)

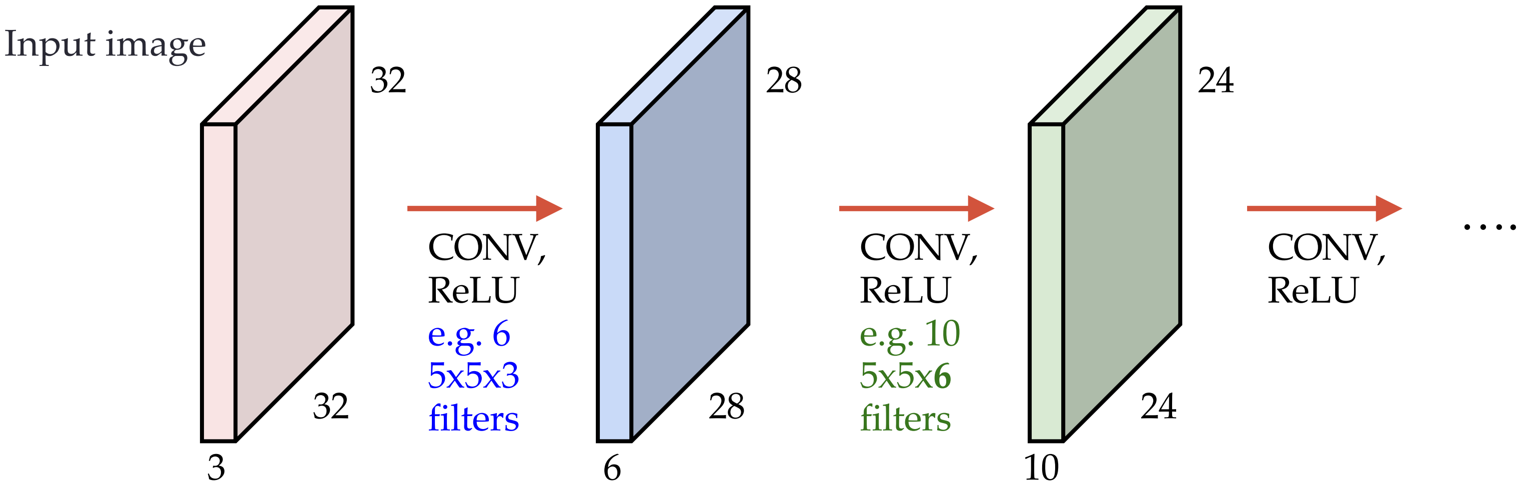
Green: Output layer



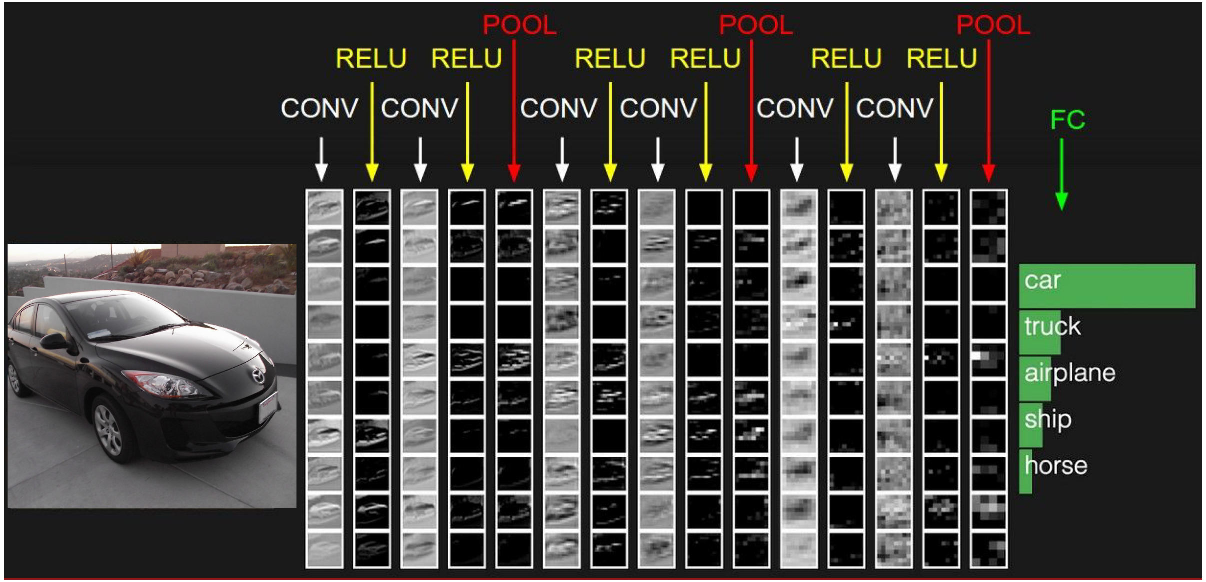

**Output:** convolve (slide) the kernel over all spatial locations

# Convolutional Neural Networks

ConvNet is a sequence of Convolutional Layers with *non-linear activation* functions



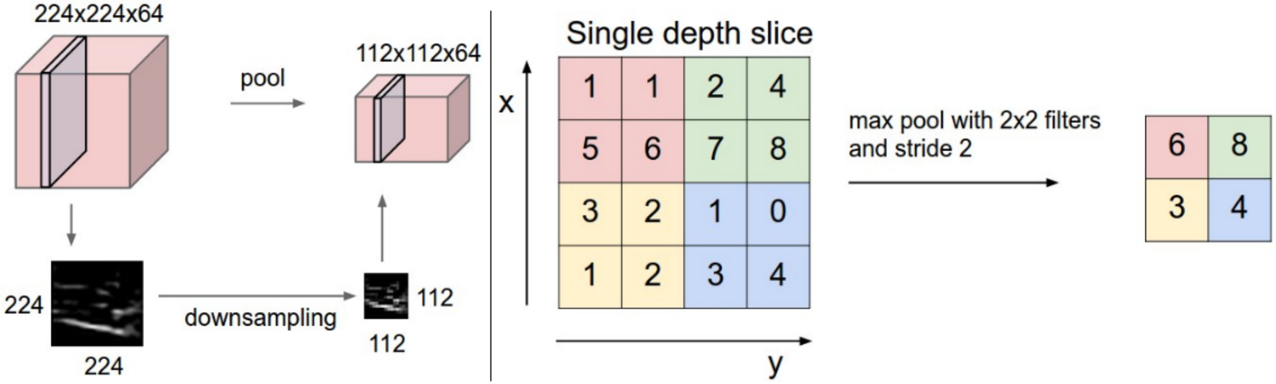
# ConvNets: Typical Architecture




Slide Credit: Marc'Aurelio Ranzato

# Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently:




Slide credit: Fei-Fei & J. Johnson & S. Yeung, 2017

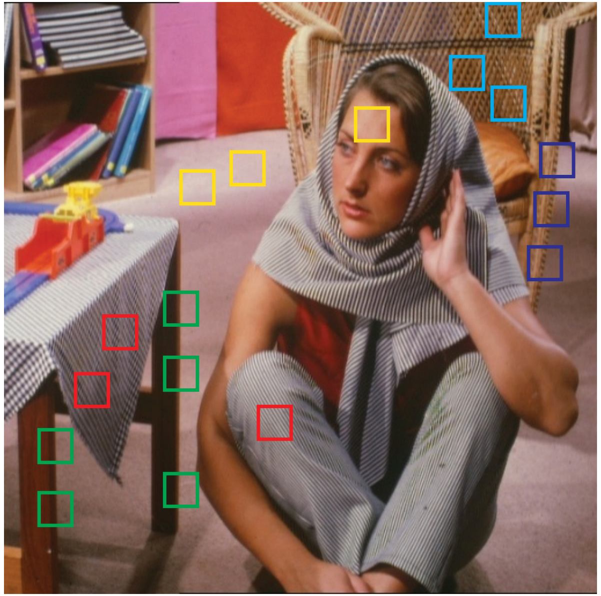
# Stationarity and Self-similarity

Properties of “natural” signals:

**Stationarity:** Certain *motifs* repeat throughout a signal.

**Locality:** Nearby points are more correlated than points far away.

**Compositionality**: Everything in nature is composed of parts that are composed of sub-parts and so on

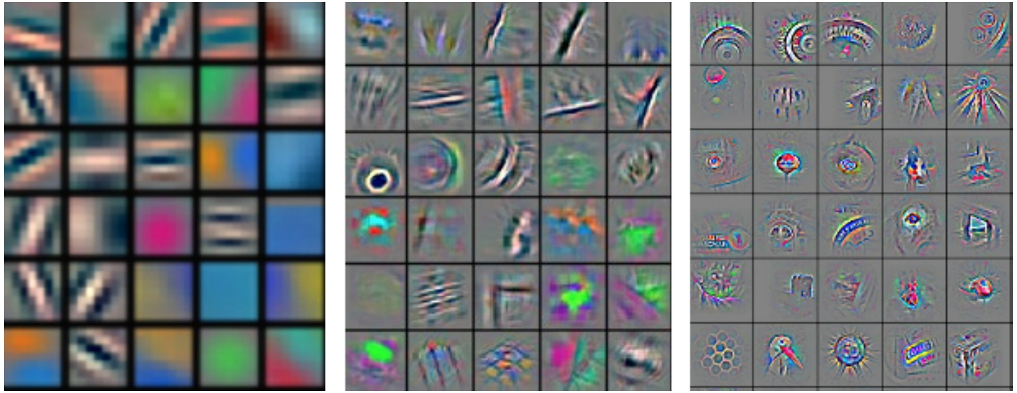


Data is self-similar across the domain

CNNs help to exploit these properties!

# Hierarchy and Compositionality

Data is compositional: images, video, sound are formed of hierarchical local stationary patterns.

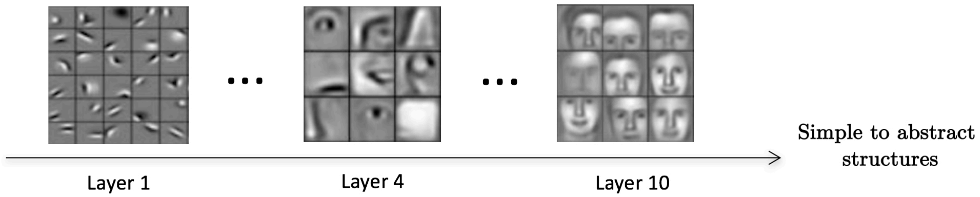


Typical features learned by a CNN becoming increasingly complex at deeper layers

Zeiler, Fergus 2013

# Hierarchy and Compositionality

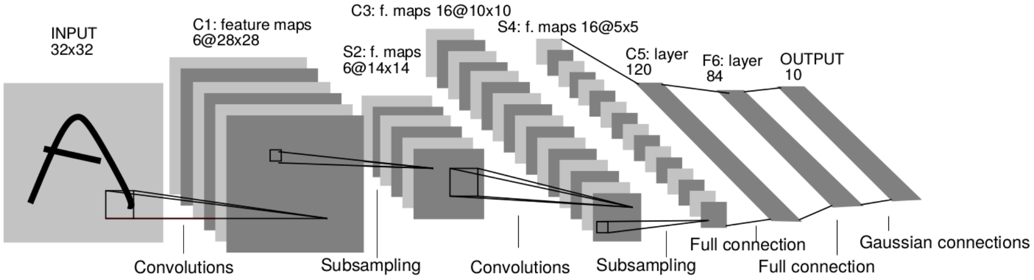
Data is compositional: images, video, sound are formed of hierarchical local stationary patterns.




Typical features learned by a CNN becoming increasingly complex at deeper layers

Zeiler, Fergus 2013

# Key Properties of CNNs

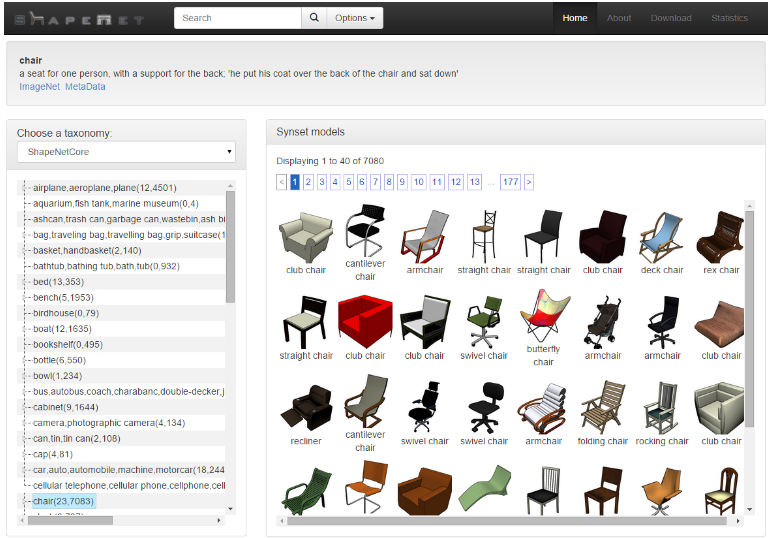



- ☺ Convolutional filters **(Translation invariance)**
- ☺ Multiple layers **(Compositionality)**
- ☺ Filters localized in space **(Locality)**
- ☺ Weight sharing **(Self-similarity)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size $n$)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)
- ☺ $\mathcal{O}(\log n)$ layers in classification tasks

LeCun et al. 1989

# CNNs for 3D shapes

# Datasets: ShapeNet




>220k 3D models


**ShapeNetCore** is used in most papers: 51k shapes in 55 object categories. The training set, validation set, and test set are composed of 35,764, 5,133, and 10,265 shapes, respectively.

Chang, Angel X., Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese et al. "Shapenet: An information-rich 3d model repository." arXiv preprint arXiv:1512.03012 (2015).

# Datasets

ABC: A Big CAD Model Dataset For Geometric Deep Learning, CVPR 2019

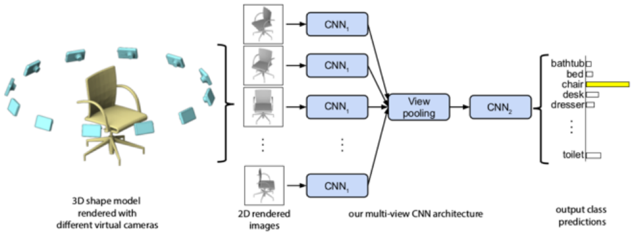

Wu, Zhirong, et al. "3d shapenets: A deep representation for volumetric shapes." CVPR 2015

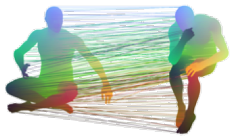

ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes, CVPR 2017

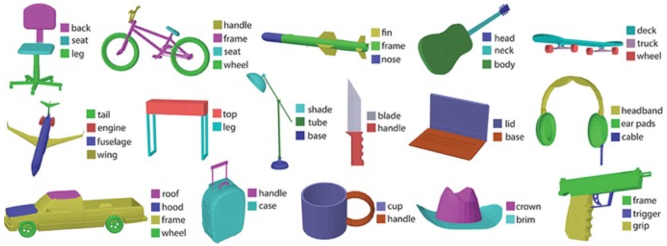

Deitke, Matt, et al. "Objaverse-xl: **A universe of 10M+ 3d objects**." arXiv preprint arXiv:2307.05663 (2023).

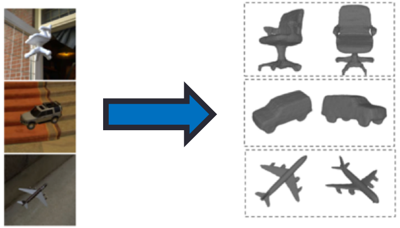
# 3D shape analysis tasks

**Fundamental tasks:** *classification, segmentation, correspondence, reconstruction, alignment,* etc. on 3D data.

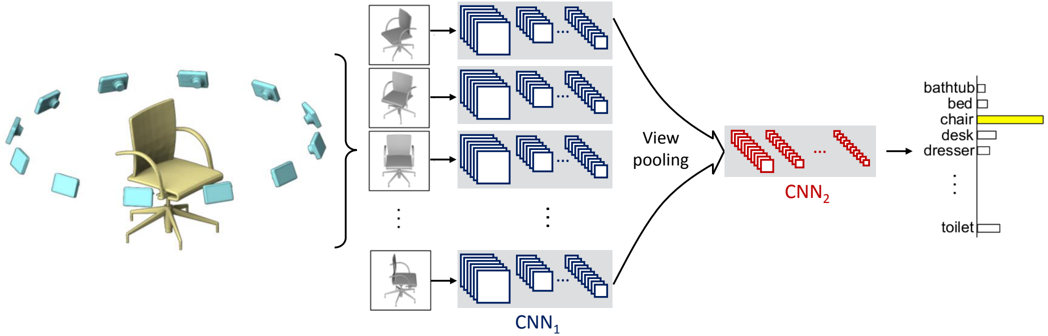




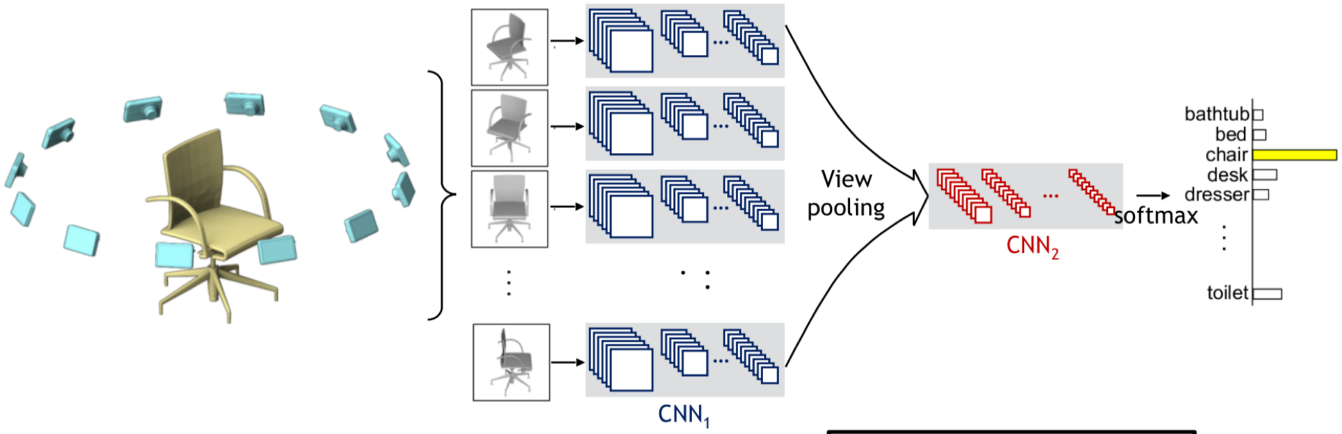
# View-Based 3D Deep Learning




- Represent 3D object as a collection of range images from different views
- $CNN_1$: Extract image features (parameters are shared across views)
- Element-wise max pooling across all views
- $CNN_2$: Produce shape descriptors + final prediction

Hang Su et al., "**Multi-view Convolutional Neural Networks for 3D Shape Recognition**", *ICCV* 2015.

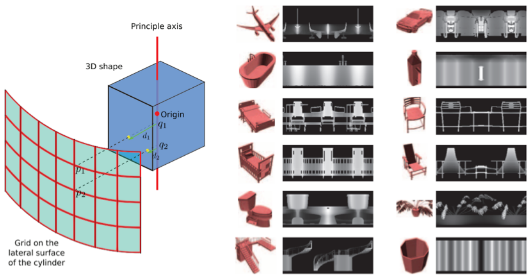
# View-Based 3D Deep Learning

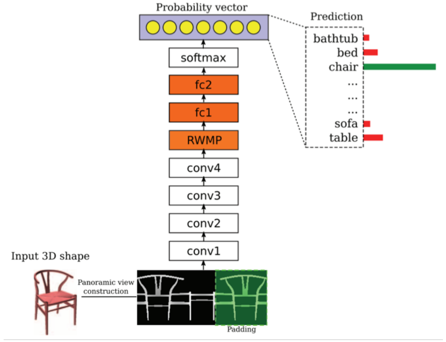


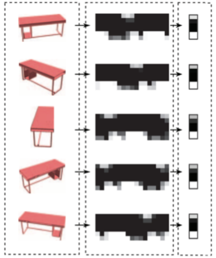

**CNN$_2$**: a second ConvNet producing shape descriptors

Hang Su et al., "**Multi-view Convolutional Neural Networks for 3D Shape Recognition**", *ICCV* 2015.

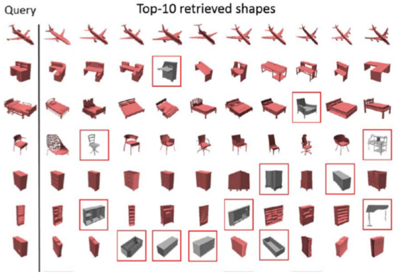
# Other View-based Methods.



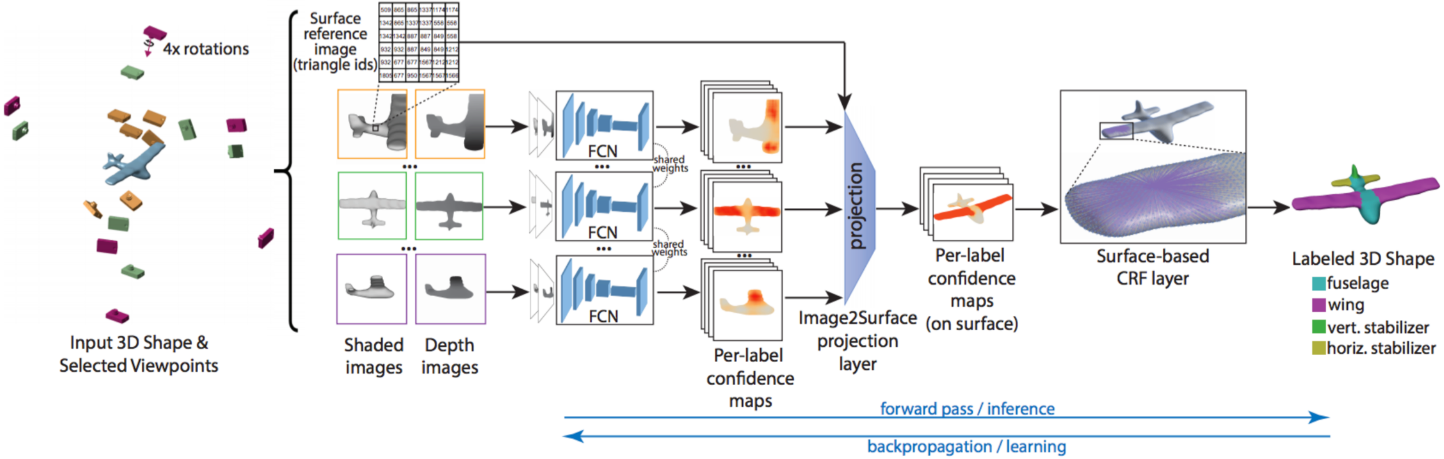



Rotation equivariance




Shi et al. *DeepPano: Deep Panoramic Representation for 3-D Shape Recognition*, IEEE Signal Processing Letters, 2015

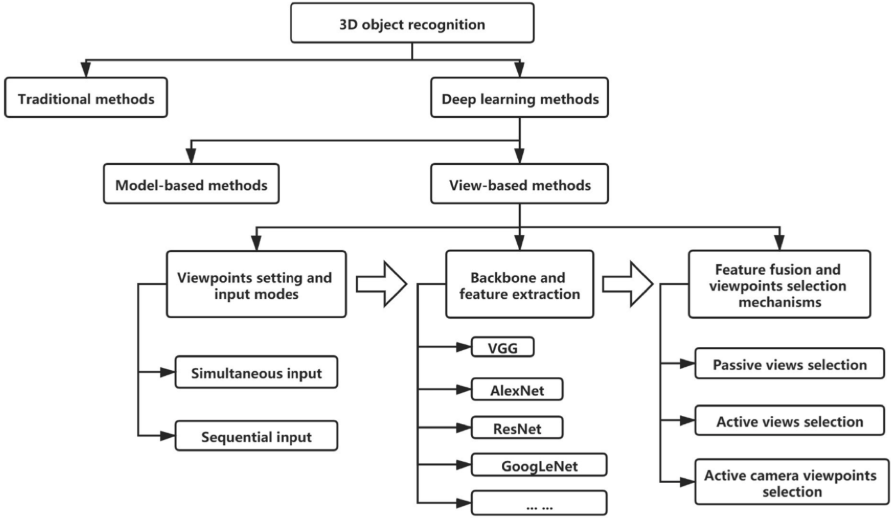
# View-Based Deep Learning Methods




Kalogerakis et al. **"3D Shape Segmentation with Projective Convolutional Networks"**, *CVPR2017.*

# Other View-based Methods.

**Many** other multi-view approaches!



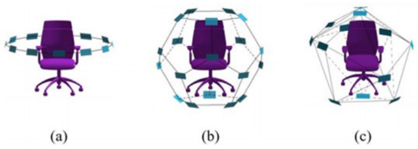




**Fig. 3.** Different view configurations in View-GCN [33].

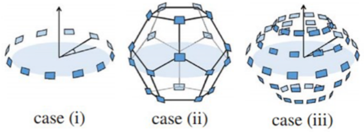


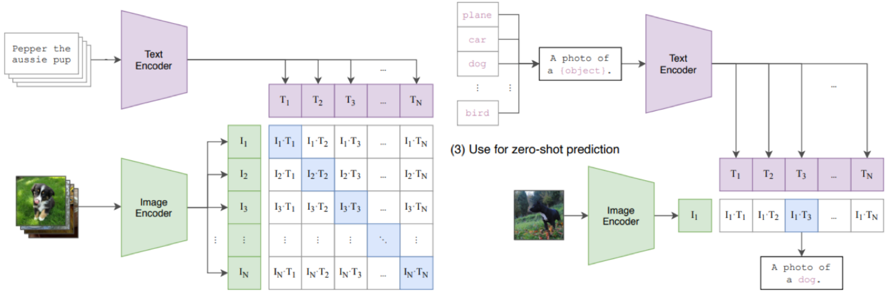

**Fig. 4.** Illustration of the three viewpoints setups considered in RotationNet [30].

Qi, Shaohua, et al. "**Review of multi-view 3D object recognition methods based on deep learning.**" Displays, 2021

# Other View-based Methods.

## Using powerful 2D models for text-based analysis

CLIP:



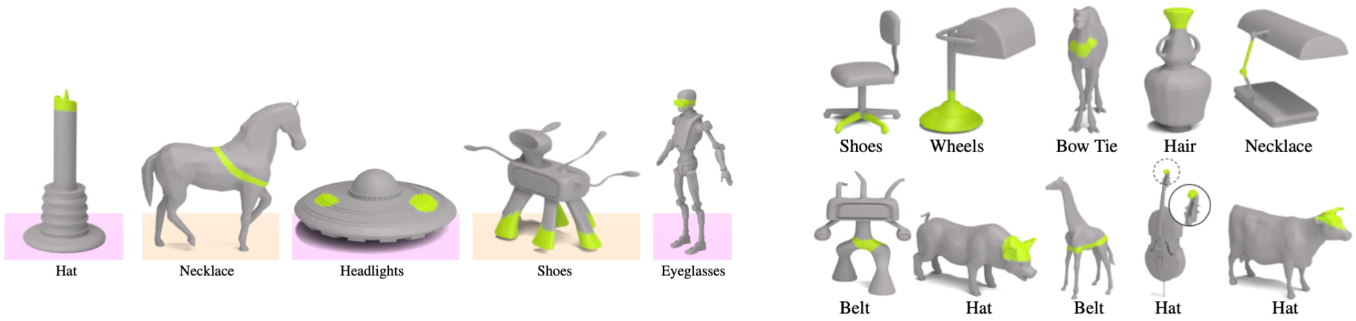

Text2Mesh:

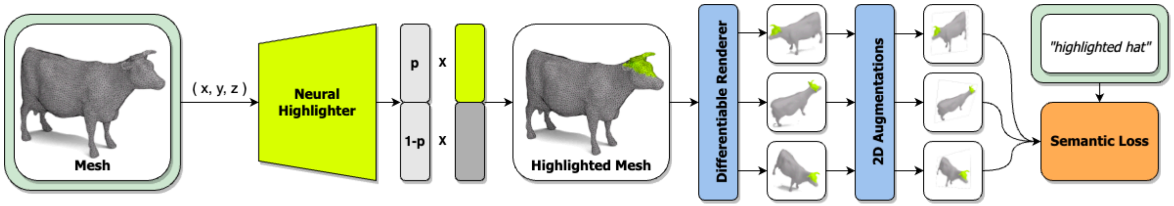

**CLIP:** Radford, Alec et al. "**Learning transferable visual models from natural language supervision**." In ICML, 2021.

Michel, Oscar, et al. "Text2mesh: Text-driven neural stylization for meshes." CVPR 2022

# Other View-based Methods.

Using a pre-trained CLIP model to discover semantic regions on a 3D shape



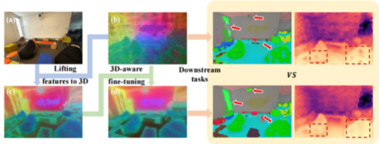

D. Decatur et al. "3d highlighter: Localizing regions on 3d shapes via text descriptions"  CVPR 2023.

# Other View-based Methods.

## Improving 2D Feature Representations by 3D-Aware Fine-Tuning

Yuanwen Yue[1], Anurag Das[2], Francis Engelmann[1,3], Siyu Tang[1], Jan Eric Lenssen[2]
[1]ETH Zurich [2]Max Planck Institute for Informatics [3]Google
ECCV 2024

Paper | Code | Demo | Colab | Poster

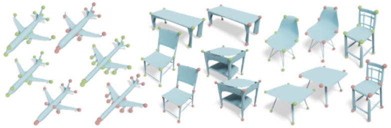



TL;DR: We propose 3D-aware fine-tuning to improve 2D foundation features. Our method starts with lifting 2D image features (e.g. DINOv2) (b) to a 3D representation. Then we finetune the 2D foundation model using the 3D-aware features (c). We demonstrate that incorporating the fine-tuned features (d) results in improved performance on downstream tasks such as semantic segmentation and depth estimation on a variety of datasets with simple linear probing (right). Feature maps are visualized using principal component analysis (PCA).

## Back to 3D: Few-Shot 3D Keypoint Detection with Back-Projected 2D Features

Thomas Wimmer[1,2], Peter Wonka[3], Maks Ovsjanikov[1]
[1]École Polytechnique, [2]Technical University of Munich, [3]KAUST
CVPR 2024

Paper | arXiv | Code | Poster

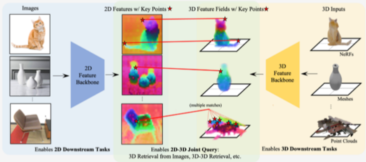


Qualitative results of our proposed method B2-3D for few-shot keypoint detection using back-projected features (red) with ground truth keypoint annotations (green).

## ConDense: Consistent 2D/3D Pre-training for Dense and Sparse Features from Multi-View Images

Xiaoshuai Zhang[1,5*], Zhicheng Wang[5], Howard Zhou[5], Soham Ghosh[5], Danushen Gnanapragasam[5], Varun Jampani[3,5*], Hao Su[1,4], Leonidas Guibas[2,5],
[1]UC San Diego, [2]Stanford University, [3]Stability AI, [4]Hillbot, [5]Google Research
* Work conducted while at Google Research.

arXiv (coming) | Paper | Video (coming) | Code | Email Contact




ConDense extract co-embedded feature for 2D or 3D inputs. ConDense not only has improved performance over previous 2D or 3D pre-training pipelines but also enables efficient cross-modality, cross-scale queries such as 3D retrieval and duplicate detection.

## Diffusion 3D Features (Diff3F)

Decorating Untextured Shapes with Distilled Semantic Features

[CVPR 2024]

Niladri Shekhar Dutt[1,2], Sanjeev Muralikrishnan[1], Niloy J. Mitra[1,3]
[1] University College London, [2] Ready Player Me, [3] Adobe Research

Arxiv | Paper | Code | Video | Poster

Diff3F is a a novel feature distiller that harnesses the expressive power of in-painting diffusion features and distills them to points on 3D surfaces. Here, the proposed features are employed for point-to-point shape correspondence between assets varying in shape, pose, species and topology. We achieve this without any fine-tuning of the underlying diffusion models, and demonstrate results on untextured meshes, point clouds, and raw scans. Note that we show raw point-to-point correspondence, without any regularization or smoothing. Inputs are **point clouds, non-manifold meshes, or 2-manifold meshes**. The left most mesh is the source and all remaining 3D shapes are targets. Corresponding points are similarly colored.

...

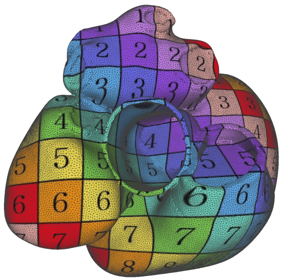
# View-based Methods.

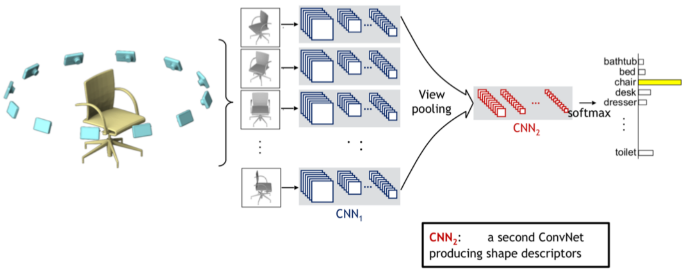
**Advantages**

- Efficiency & simplicity
- Can use (pre-trained) CNNs!
- Can be used to *optimize* 3D shapes via *differentiable rendering*.

**Limitations**

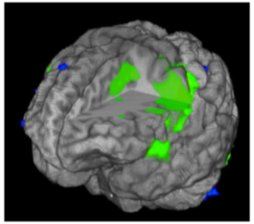
- Are cumbersome for *local* analysis (e.g., segmentation)
- Are not adapted to *deformable* shapes
- Not great for topologically complex shapes



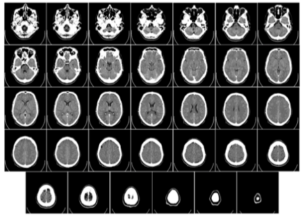


# Volumetric Representations

Represent data as a signal on *voxel grid:* $\mathbb{R}^{d \times d \times d}$

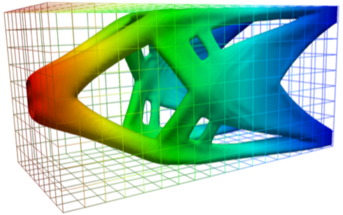


fMRI

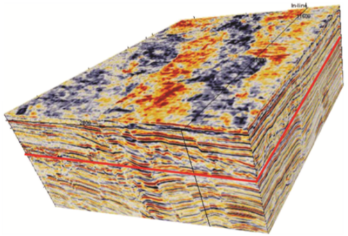


CT

Voxelized
CAD models



Manufacturing
(finite-element analysis)



Geology

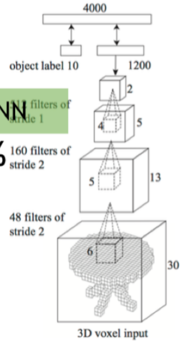
Can directly use convolutions (with 3D kernels)!

Credit: E. Kalogerakis

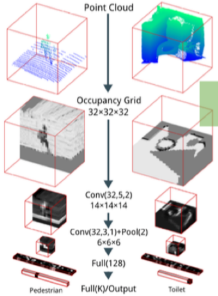
# Volumetric Representations

3DShapeNets from Princeton
CVPR 2015

3D CNN
77.3%

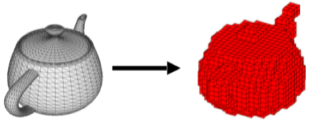



VoxNet from CMU Robotics
IEEE/RSJ 2015




3D CNN
83.0%

**Information loss in voxelization**

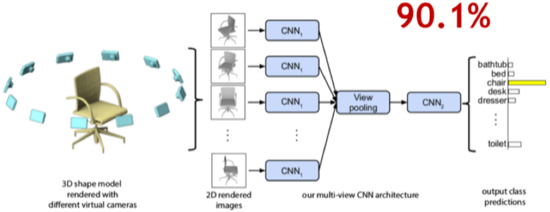


CAD model

Occupancy Grid
30x30x30

Rendering +
2D CNN
90.1%

MVCNN from UMass
ICCV 2015

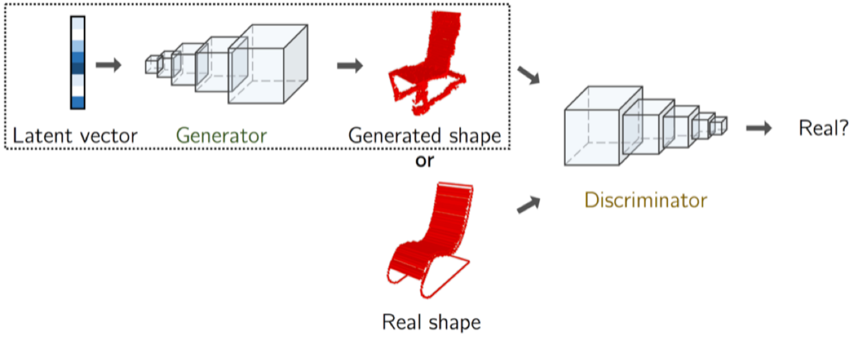



Credit: E. Kalogerakis

*Volumetric and multi-view CNNS for object classification on 3d data*

# Volumetric Representations

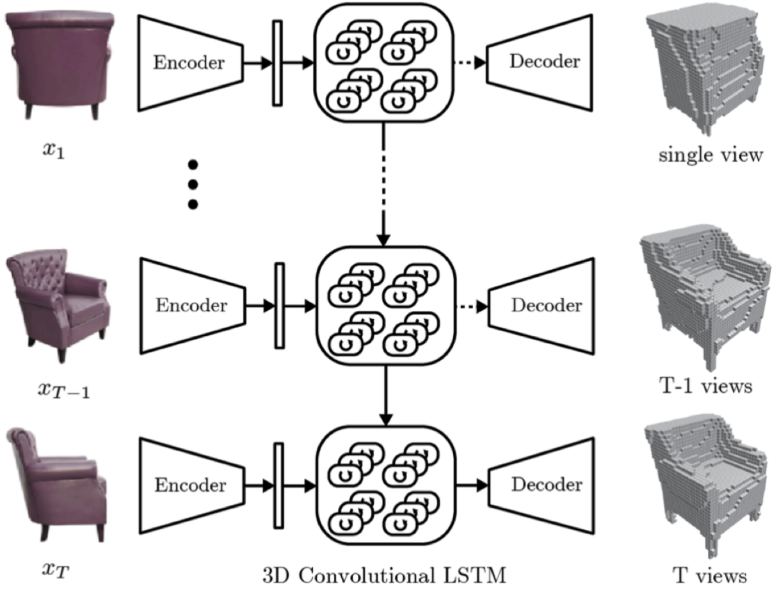
**Volumetric Adversarial Generative Networks:**




Wu et al. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*, NIPS 2016

# Volumetric Representations

## Volumetric Adversarial Generative Networks:




Choy et al. **3D-R^2N^2: A unified approach for single and multi-view 3D object reconstruction** *ECCV 2016*

# Other Volumetric Methods.

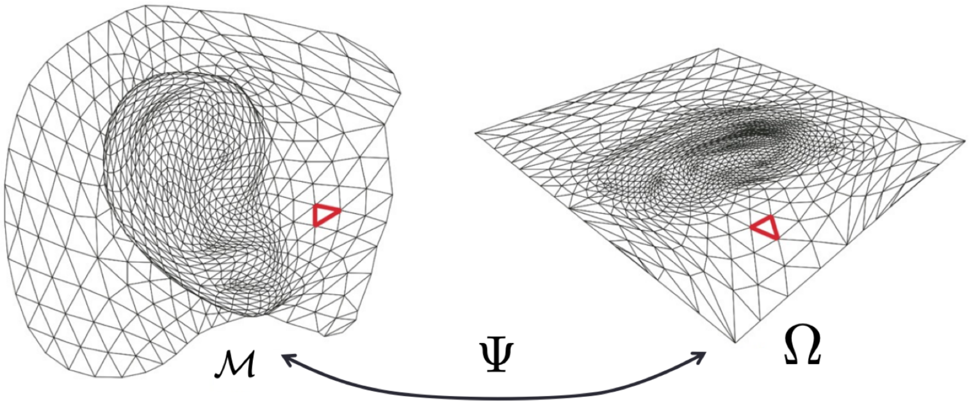
1. Yan et al., *Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision*, NIPS 2016
2. Klokov et al., *Escape from cells: Deep kd-networks for the recognition of 3d point cloud models*, ICCV 2017
3. Wang et al., *O-cnn: Octree-based convolutional neural networks for 3d shape analysis*, TOG 2017
4. Dai et al., *Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis*, CVPR 2017

…

Will get back to this next (and the following) weeks.

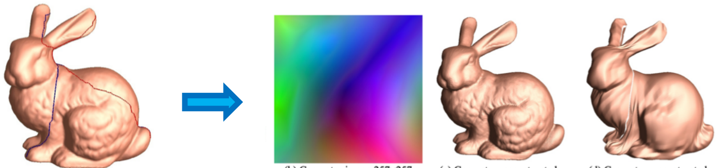
# Global parametrization methods

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.

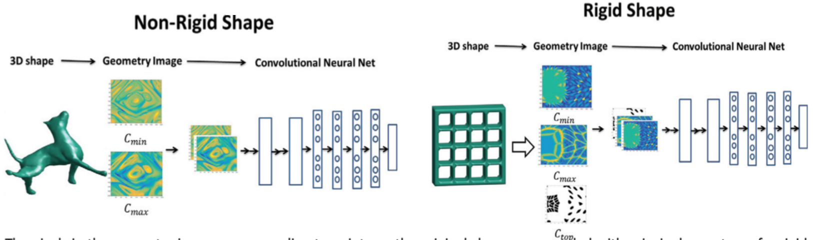


# Global parametrization methods

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



(a) Original mesh with cut 70K faces; genus 0 (b) Geometry image 257×257 (b∗) Compr. to 1.5KB (not shown) (c) Geometry reconstructed entirely from b (d) Geometry reconstructed entirely from b∗




The pixels in the geometry image corresponding to points on the original shape are encoded with principal curvatures for rigid shapes and HKS for non-rigid shapes. Then a standard CNN architecture can be modeled to learn the 3D shape.

***Shape Classification and Retrieval***

Non-rigid Shapes

| | McGill1 | | McGill2 | | SHREC1 | | SHREC2 | |
|---|---|---|---|---|---|---|---|---|
| | *Classify* | *Retrieve* | *Classify* | *Retrieve* | *Classify* | *Retrieve* | *Classify* | *Retrieve* |
| **ShpGoogle** | NA | NA | NA | NA | 62.6 | **0.65** | 70.8 | 0.74 |
| **Zerkine** | 63 | 0.64 | 57.5 | 0.69 | 43.3 | 0.47 | 50.8 | 0.64 |
| **LFD** | 75 | 0.67 | 72.5 | 0.68 | 56.7 | 0.5 | 65.8 | 0.65 |
| **ShapeNets** | 65 | 0.29 | 57.2 | 0.28 | 52.7 | 0.1 | 48.4 | 0.13 |
| **Conformal** | 55 | 0.36 | 80 | 0.58 | 60.6 | 0.45 | 85 | 0.65 |
| **SPHARM** | 62 | 0.35 | 82.5 | 0.58 | 59 | 0.45 | 82.5 | 0.66 |
| **Ours** | 83 | 0.75 | 92.5 | 0.72 | 88.6 | 0.65 | 96.6 | 0.72 |

Rigid Shapes

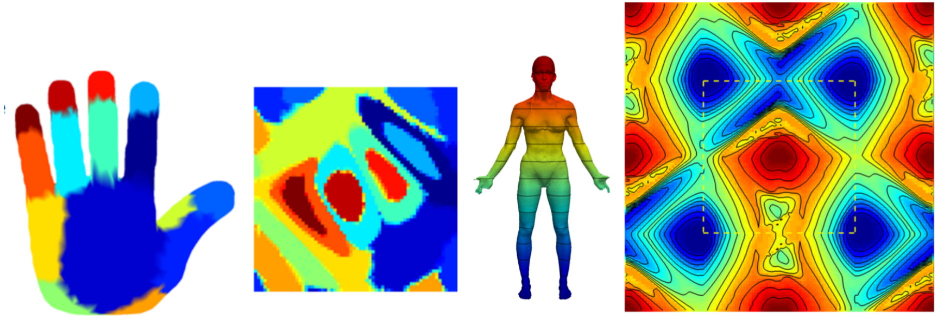
| | | VoxNet | DeepPano | LFD | ShpNets | SphHarm | Conf | SPHARM | Ours |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | Classify | 92 | 85.5 | 79.8 | 83.5 | 79.9 | 78.2 | 79.9 | 88.4 |
| **Net10** | Retrieve | NA | 84.1 | 49.8 | 69.2 | 45.9 | 67.4 | 65.2 | 74.9 |
| **Model** | Classify | 83 | 77.6 | 75.4 | 77.3 | 68.2 | 75.6 | 75.9 | 83.9 |
| **Net40** | Retrieve | NA | 76.8 | 40.9 | 49.9 | 34.4 | 46.2 | 44.8 | 51.3 |


Gu, Xianfeng, Steven J. Gortler, and Hugues Hoppe. "Geometry images." SIGGRAPH 2002.

Sinha, Ayan et al. "Deep learning 3D shape surfaces using geometry images." ECCV 2016

# Global parametrization methods

Key idea: map the input surface to some parametric domain (e.g. 2D plane) where operations can be defined more easily.



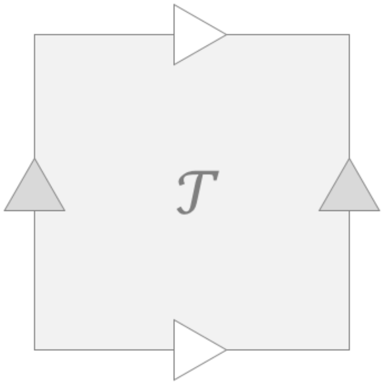
- Enables adoption of Euclidean techniques in the embedding space
- Provides invariance to certain operations
- Parametrization may be non-unique
- The map can introduce distortions

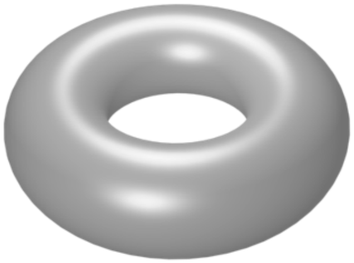
Sinha et al. 2016; Maron et al. 2017

# Convolution on Surfaces

Is translation-invariant convolution on surfaces possible?

Yes! The torus is the only closed orientable surface admitting a translational group.

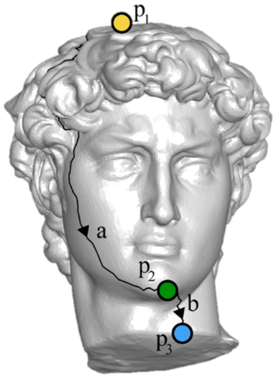


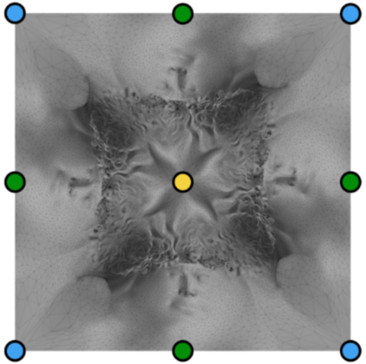
CNNs can be well-defined over the flat-torus!

# Global parametrization methods

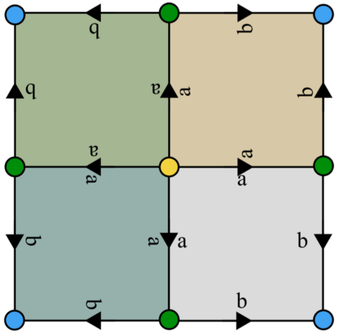
Torus 4-cover




Surface $\mathcal{S}$ with sphere topology



Flat-torus $\mathcal{T}$ with 4 replicas of $\mathcal{S}$



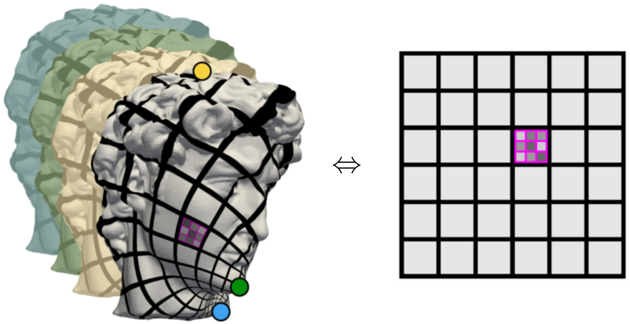

Standard Euclidean 2D CNN architectures can now be used on $\mathcal{T}$.

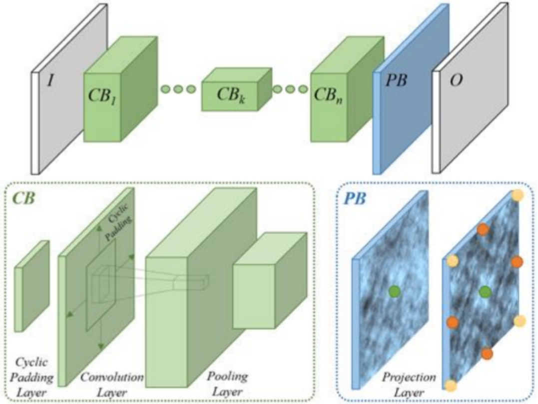
Maron, Haggai, et al. "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.* 36.4, 2017

# Projection-based methods

## Torus 4-cover

For each triplet $\{p_1, p_2, p_3\} \in \mathcal{S}$, use orbifold-Tutte to map $\mathcal{S}^4$ to $\mathcal{T}$.





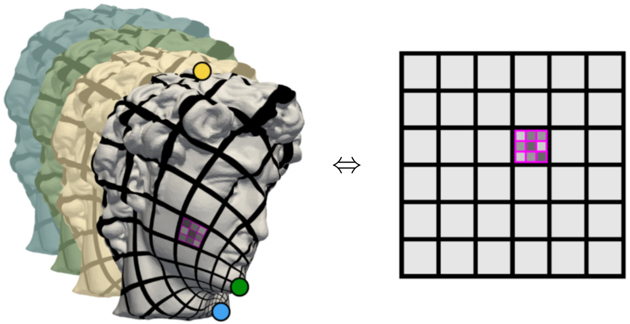

Tha mapping from $\mathcal{S}^4$ to $\mathcal{T}$ is a conformal homeomorphism.


Maron, Haggai, et al. "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.* 36.4, 2017

# Projection-based methods

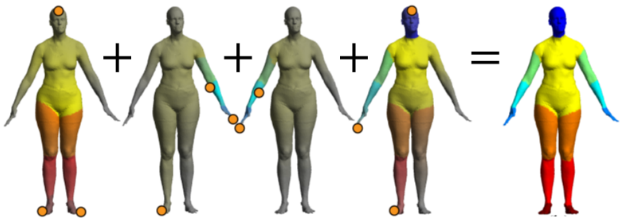
## Torus 4-cover

For each triplet $\{p_1, p_2, p_3\} \in \mathcal{S}$, use orbifold-Tutte to map $\mathcal{S}^4$ to $\mathcal{T}$.



Conformality introduces scale changes.

- "Magnifying glass" effect
- Prediction aggregation from different triplets at test time



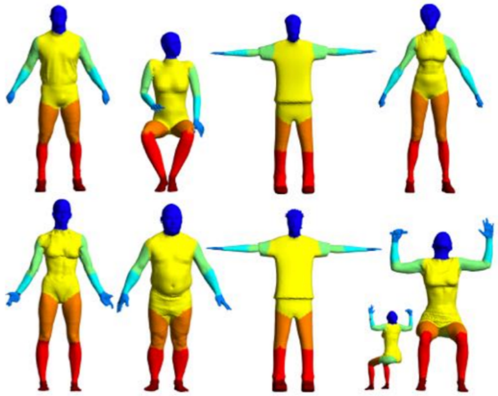
Tha mapping from $\mathcal{S}^4$ to $\mathcal{T}$ is a conformal homeomorphism.

Maron, Haggai, et al. "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.* 36.4, 2017

# Projection-based methods

Segmentation results

Training data



Predictions on the test set

Maron, Haggai, et al. "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.* 36.4, 2017

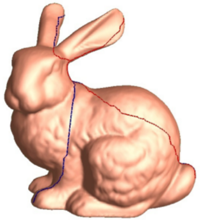
# Projection-based Methods.

**Advantages**

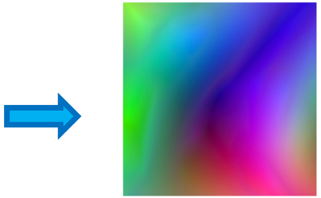
- Represent the shape as a whole (rather than *partial* views)
- Can reuse shape parametrization methods
- Enable the use of CNNs

**Limitations**

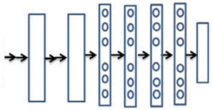
- Parametrizations are not unique
- Typically induce (often heavy) *distortion*
- Rarely used in practice anymore



(a) Original mesh with cut
70K faces; genus 0



(b) Geometry image 257×257
(b∗) Compr. to 1.5KB (not shown)



**Main question (for the rest of the lecture):**

How to enable neural networks to operate *directly on deformable 3D surfaces*?