GAMES
3D Deep Learning

Qixing Huang
September 2\textsuperscript{nd} 2021
AlexNet
Image Generation

The diagram illustrates the process of generating images using a series of deconvolution steps. Starting with a code vector, the process involves projecting and reshaping it to a higher-dimensional space. Each deconvolution step increases the dimensionality and adds more detail until the final image is generated. The network architecture is designed to handle different strides and feature maps, allowing for the creation of high-resolution images. The real images from ImageNet and the generated images are shown side by side for comparison.
3D Surface Representations

- Triangular mesh
- Implicit surface
- Light Field Representation
- Point cloud
- Part-based models

- Scene Graph
3D Voxel Grids
3D Deep Learning

3D Shape as Volumetric Representation

mesh

binary voxel

[Wu et al. 15]
3D ShapeNets

A Deep Belief Network is a generative graphical model that describes the distribution of input $x$ over class $y$.

- Convolution to enable compositionality
- No pooling to reduce reconstruction error

<table>
<thead>
<tr>
<th>Layer</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 1-3</td>
<td>convolutional RBM</td>
</tr>
<tr>
<td>Layer 4</td>
<td>fully connected RBM</td>
</tr>
<tr>
<td>Layer 5</td>
<td>multinomial label + Bernoulli feature form an associate memory</td>
</tr>
</tbody>
</table>
3D ShapeNets ≠ CNNs

\[ p(x, y) \quad p(y|x) \]

\[ p(y|x) \quad \text{discriminative process} \]

\[ p(x|y) \quad \text{generative process} \]

* 3D ShapeNets can be converted into a CNN, and discriminatively trained with back-propagation.

Convolutional Deep Belief Network \( p(x, y) \)

[Wu et al. 15]
As a 3D Shape Prior

Convolutional Deep Belief Network $p(x, y)$

Sampled Models

[Wu et al. 15]
3D Generative Adversarial Network [Wu et al. 16]
Objects generated by Wu et al. [2015] (30 × 30 × 30)

Objects generated by a volumetric autoencoder (64 × 64 × 64)
Sparse 3D Convolutional Networks
[Ben Graham 2016]

40x40x40 Grid
Sparsity for lower layers
Low resolution for upper layers
Octree classification networks

[Wang et al. 18]

The responses of some convolutional filters at different levels on two models are rendered. Red represents a large response and blue a low response.
Discussion

+ Easy to implement
+ Hardware friendly

- Low resolution
- No structural information
- Cannot utilize 2D training data
Light Field Representation
3D shape model rendered with different virtual cameras
<table>
<thead>
<tr>
<th>Method</th>
<th>Training Config.</th>
<th>Test Config.</th>
<th>Classification (Accuracy)</th>
<th>Retrieval (mAP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre-train</td>
<td>Fine-tune</td>
<td>#Views</td>
<td>#Views</td>
</tr>
<tr>
<td>(1) SPH [16]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(2) LFD [5]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(3) 3D ShapeNets [37]</td>
<td>ModelNet40</td>
<td>ModelNet40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(4) FV</td>
<td>-</td>
<td>ModelNet40</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>(5) FV, 12×</td>
<td>-</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(6) CNN</td>
<td>ImageNet1K</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>(7) CNN, f.t.</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>(8) CNN, 12×</td>
<td>ImageNet1K</td>
<td>-</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>(9) CNN, f.t.,12×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(10) MVCNN, 12×</td>
<td>ImageNet1K</td>
<td>-</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>(11) MVCNN, f.t., 12×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(12) MVCNN, f.t.+metric, 12×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>(13) MVCNN, 80×</td>
<td>ImageNet1K</td>
<td>-</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>(14) MVCNN, f.t., 80×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>(15) MVCNN, f.t.+metric, 80×</td>
<td>ImageNet1K</td>
<td>ModelNet40</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

* f.t.=fine-tuning, metric=low-rank Mahalanobis metric learning
Discussion

+ Can utilize 2D training data
+ Efficient since using 2D convolutions
+ Top-performing algorithms

-- Redundancy
-- Loss of information per view
-- How to pick views?

Questions:

? Convolutions on Spheres
Point cloud Representation
[Su et al. 17a, Su et al. 17b]
Object Classification on Partial Scans

Input:
Partial scan (XYZ)

Output:
Category classification
Car
Car
Airplane
Table
Mug
Object Part Segmentation

Input:
Point cloud (XYZ)

Output:
Per point label
Semantic Segmentation for Indoor Scenes

Input:
Point cloud (XYZRGB) of a room

Output (current performance):
Semantic segmentation of the room

- wall
- table
- floor
Uniform Framework: PointNet

Point Set → PointNet → Object Classification

PointNet → Part Segmentation

PointNet → Semantic Segmentation in scenes

PointNet → Point Feature Learning

...
Theorem 1. Suppose $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot, \cdot)$. $\forall \epsilon > 0$, $\exists$ a continuous function $h$ and a symmetric function $g(x_1, \ldots, x_n) = \gamma \circ \text{MAX}$, such that for any $S \in \mathcal{X}$,

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

where $x_1, \ldots, x_n$ is the full list of elements in $S$ ordered arbitrarily, $\gamma$ is a continuous function, and max is a vector max operator that takes $n$ vectors as input and returns a new vector of the element-wise maximum.
Figure 2. **PointNet Architecture.** The classification network takes $n$ points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification score for $k$ classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, the numbers in brackets are its layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.
## ModelNet shape 40-class classification

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>40%</td>
</tr>
<tr>
<td>LSTM</td>
<td>75%</td>
</tr>
<tr>
<td>Conv-Max-FC (1 max)</td>
<td>84%</td>
</tr>
<tr>
<td>Conv-Max-FC (2 max)</td>
<td>86%</td>
</tr>
<tr>
<td>Conv-Max-FC (2 max) + Input Transform</td>
<td>87.8%</td>
</tr>
<tr>
<td>Conv-Max-FC (2 max) + Feature Transform</td>
<td>86.8%</td>
</tr>
<tr>
<td>Conv-Max-FC (2 max) + Feature Transform + orthogonal regularization</td>
<td>87.4%</td>
</tr>
<tr>
<td><strong>Conv-Max-FC (2 max) + Input Transform + Feature Transform + orthogonal regularization</strong></td>
<td><strong>88.9%</strong></td>
</tr>
</tbody>
</table>

*Best Volumetric CNN: 89.1%*
*However, PointNet is around 5x - 10x faster than Volumetric CNN*
3D Surface Representations

- Triangular mesh
- Implicit surface
- Part-based models
- Light Field Representation
- Point cloud
Matching in Embedding Spaces
[Wei, Huang, Ceylan, Vouga, Li 16]
Existing methods usually follow a two-step approach (e.g., SIFT flow)

• Local descriptor computation

• Dense pixel labeling via MRF inference
  – Preserve descriptors
  – Preserve smoothness
Issues of such two-step approach

Partial similarity

Inefficient when matching multiple objects
Embedding --- establishing correspondences in the embedding space

Spectral embedding [Liu et al. 06]

Sensitive to 1) partial similarity, and 2) geometric and topological changes
Properties of the desired embedding space

Corresponding points are matched in the embedding space

Embedding preserves continuity
The benefits of object embedding

• Correspondences become nearest neighbor query
  – Efficiency for multiple object matching
    $O(n)$ embeddings + $O(n^2)$ queries
  – Partial similarity

– Fuzzy correspondences
The biggest message of deep neural networks

- Approximate any function given sufficient data
Focus on depth images

- Scanning devices generate depth images

- Complete shape embedding are aggregated from depth image embeddings
  - 3D convolution is not ready yet
Architecture

The input is a depth image

The output is a per-pixel descriptor (dim 16)

Convolution + Deconvolution
Training data

- 4 animation sequences (dense correspondences)
- 2500 shapes from Yobi3D (33 feature points)
Direct versus Indirect

- Descriptor learning (e.g., triplet loss [Schroff et al. 15])

- Classification loss (e.g., the second last layer of AlexNet)
We employ a classification loss

Classes are defined in terms of super-patches

We use multiple segmentations --- so the probability of two points belong to the same segment is related to their distance
We employ the classification loss

\[
\{w_i^*\}, w^* = \arg\min_{\{w_i\}, w} \sum_{i=1}^{M} l(w_i, w)
\]
Evaluation on the FAUST dataset

Cumulative error distribution, intra-subject
Evaluation on the FAUST dataset

Cumulative error distribution, inter-subject
Multi-view 3D Models from Single Images
With a Convolutional Network [ECCV’ 16]
Fig. 5. Depth map predictions (top row) and the corresponding ground truth (bottom row). The network correctly estimates the shape.
Multi-view 3D Models from Single Images with a Convolutional Network

Maxim Tatarchenko, Alexey Dosovitskiy, Thomas Brox

Department of Computer Science
University of Freiburg
{tatachm, dosovits, brox}@cs.uni-freiburg.de

ECCV 2016
Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision [Yan et al. 16]
Figure 1: (a) Understanding 3D object from learning agent’s perspective; (b) Single-view 3D volume reconstruction with perspective transformation. (c) Illustration of perspective projection. The minimum and maximum disparity in the screen coordinates are denoted as \(d_{\text{min}}\) and \(d_{\text{max}}\).

\[
\mathcal{L}_{\text{vol}}(I^{(k)}) = \| f(I^{(k)}) - V \|_2^2
\]

\[
\mathcal{L}_{\text{proj}}(I^{(k)}) = \sum_{j=1}^{n} \mathcal{L}_{\text{proj}}^{(j)}(I^{(k)}; S^{(j)}, \alpha^{(j)}) = \frac{1}{n} \sum_{j=1}^{n} \| P(f(I^{(k)}); \alpha^{(j)}) - S^{(j)} \|_2^2
\]

\[
\mathcal{L}_{\text{comb}}(I^{(k)}) = \lambda_{\text{proj}} \mathcal{L}_{\text{proj}}(I^{(k)}) + \lambda_{\text{vol}} \mathcal{L}_{\text{vol}}(I^{(k)})
\]
Learning Semantic Deformation Flows with 3D Convolutional Networks [Yumer and Mitra 2016]
Semantic Scene Completion from a Single Depth Image [Song et al. 17]
A Point Set Generation Network for 3D Object Reconstruction from a Single Image [Fan, Su, Guibas, 2017]
Network Architecture

vanilla version

two prediction branch version
Distance Metrics

• Chamfer distance

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

• Earth Mover’s distance

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

\[ \phi : S_1 \to S_2 \text{ is a bijection} \]
Visual results

Synthetic Data

Real World Data
CD (Left) versus EMD (Right)
GRASS: Generative Recursive Autoencoders for Shape Structures
[Li, Xu, Chaudhuri, Yumer, Zhang, Guibas, SIGGRAPH’ 17]
Huge Variety of (Attributed) Graphs

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

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

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

- Looks like a parse tree for a sentence!
Recursive Neural Network (RvNN)

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

\[ p = f(W [c_1; c_2] + b) \]
Recursively Merging Parts
Different types of merges, varying cardinalities!
Training with Reconstruction Loss

- Learn weights from a variety of randomly sampled merge orders for each box structure
Results: Shape interpolation
Discussion