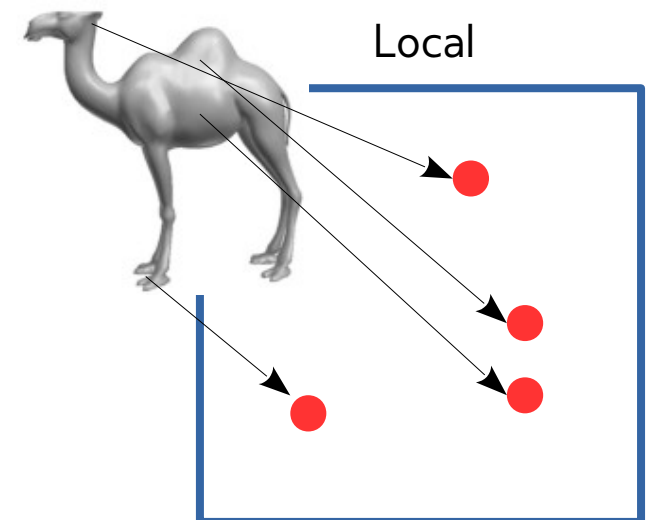
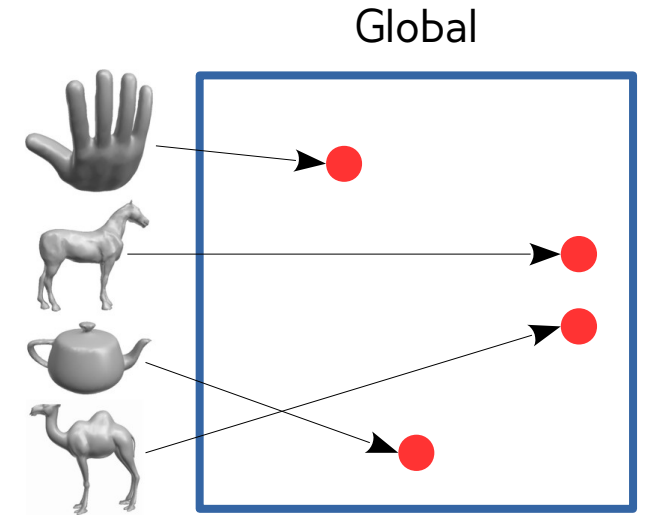


Shape Descriptors - III

Recap

- A **shape descriptor** is a set of numbers that describes a shape in a way that is
 - Concise
 - Quick to compute
 - Efficient to compare
 - Discriminative
- **Local descriptors** describe (neighborhoods around) points
- **Global descriptors** describe whole objects
- Typically, the descriptors form a **vector space** with a **meaningful distance metric**

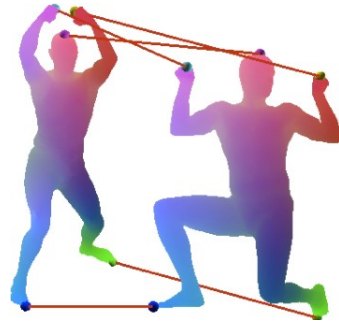


Local

Global



Feature detection



Correspondences



Registration



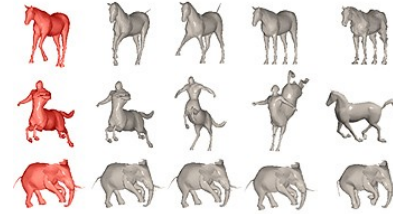
Symmetry detection



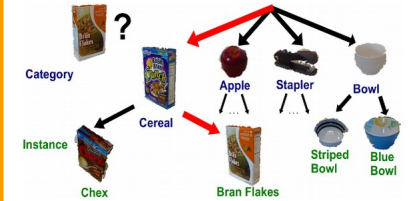
- head
- torso
- upper
- lower
- hand
- upper
- lower
- foot

Segmentation

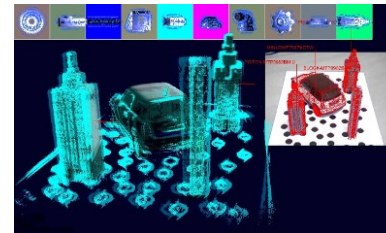
Labeling



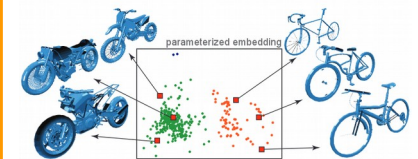
Retrieval



Classification



Recognition



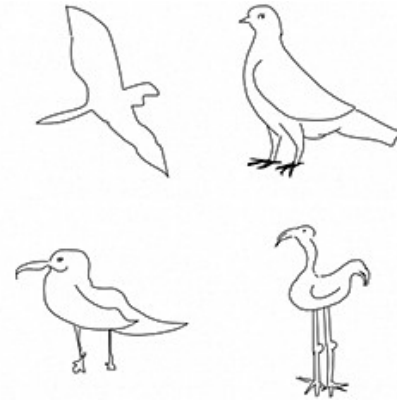
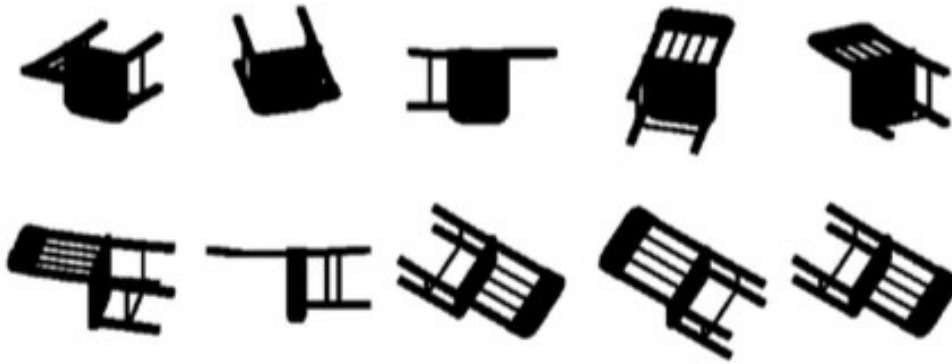
Clustering

Today

- 2D global descriptors for 3D shapes
 - Light Field Descriptor (LFD)
 - Multi-View Convolutional Neural Network (MVCNN)

Why 2D?

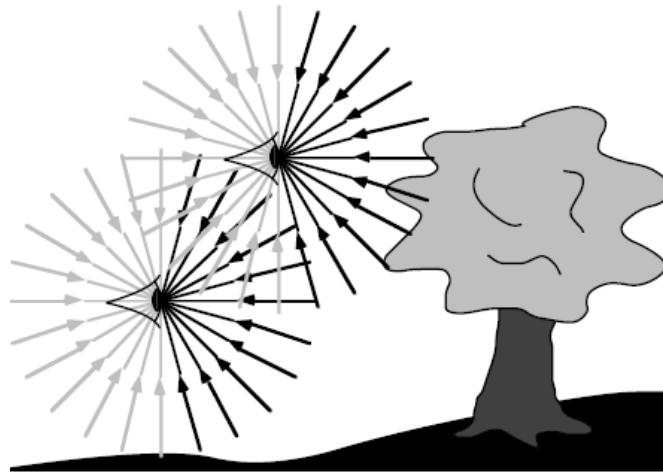
- 2D views contain a **lot of information** about a shape
 - That's how humans see stuff, and we do quite well



- For many applications, the additional information in 3D data quickly reaches **diminishing returns** and can even hurt performance since statistical models need to be more complex
- We have huge amounts of **prior information and models** for processing 2D data

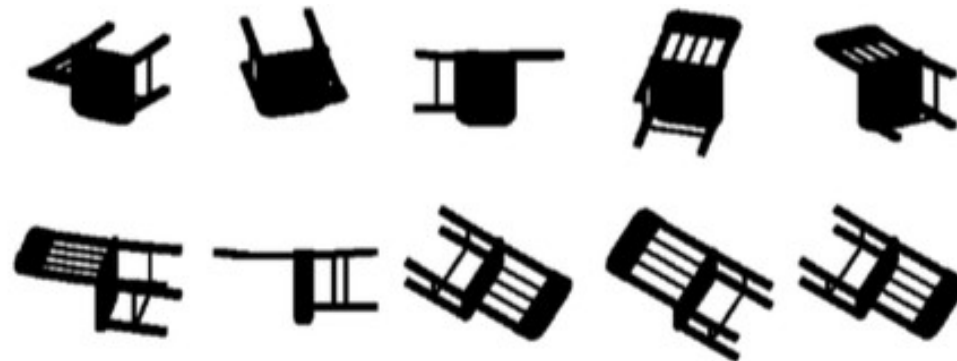
Light Field

- A **light field** (or **plenoptic function**) captures the radiance at a (3D) point along a (2D) direction
 - It is a 5D function
 - In free space, all points on a straight line have the same light field value in that direction, so reduces to a 4D function
 - With the free space assumption, a set of perspective images of an object from all possible directions constitutes its light field



Light Field Descriptor

- The **Light Field Descriptor** (LFD) of a 3D shape is a set of 2D images of it, taken from a 2D array of cameras
 - 20 cameras positioned at the vertices of a regular dodecahedron
 - Images rendered as silhouettes, so 10 unique views (say from a hemisphere)



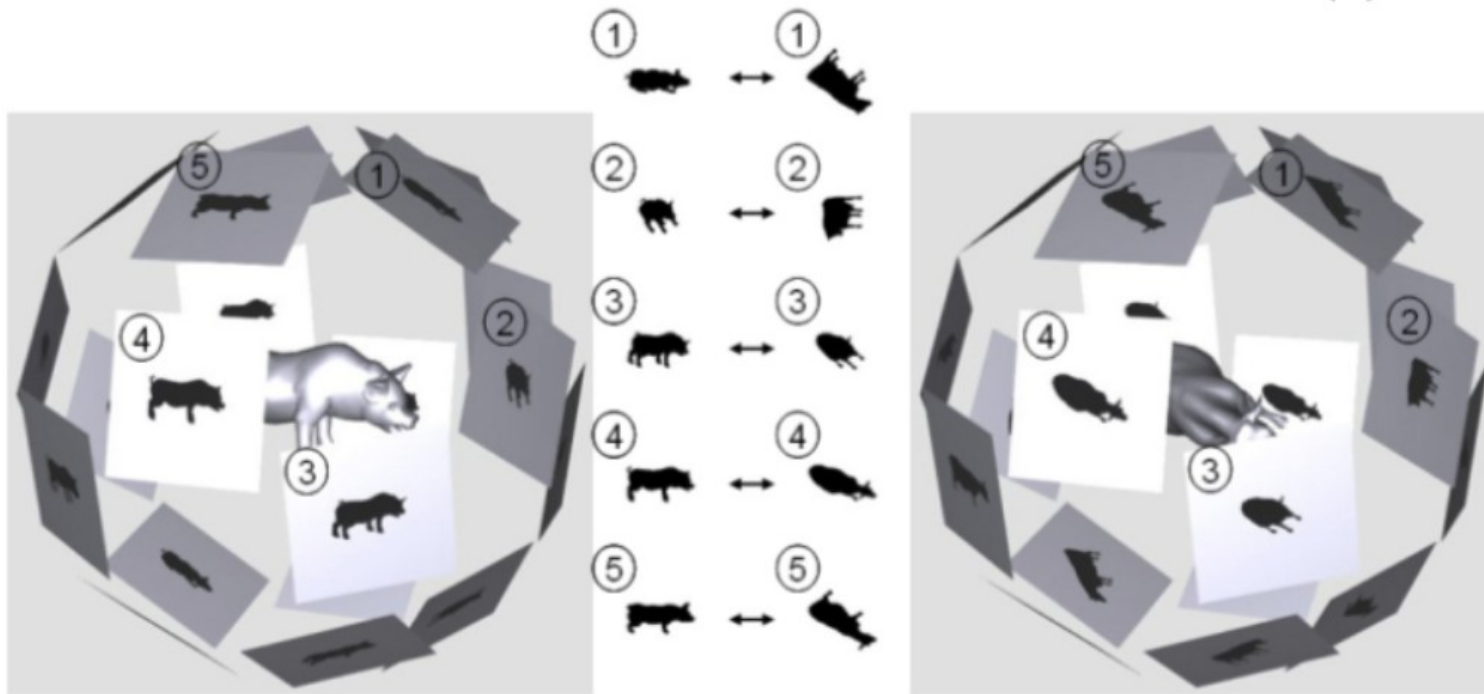
Comparing Shapes with LFD

- Consider two shapes



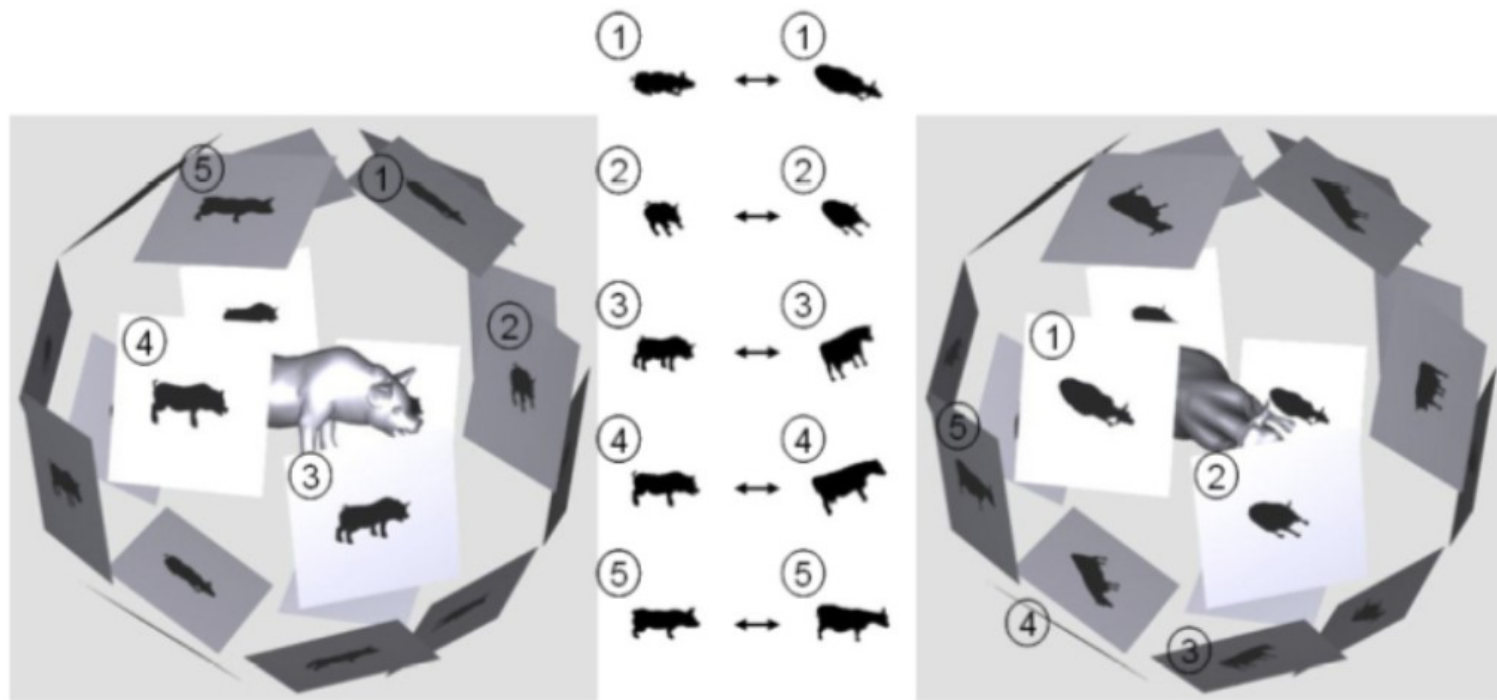
Comparing Shapes with LFD

- A candidate rotation aligns the two sets of images
 - Comparing aligned image pairs gives a similarity metric



Comparing Shapes with LFD

- And another...



Comparing Shapes with LFD

- 60 different ways of aligning the dodecahedra
- The distance between two shapes A and B , with image sets $\{A_i\}$, $\{B_i\}$ is

$$D(A, B) = \min_{r=1}^{60} \sum_{i=1}^{10} d_{\text{image}}(A_i, B_{\text{rot}(r,i)})$$

where $B_{\text{rot}(r,i)}$ is the image aligned to A_i by the r 'th rotation

More views for more accuracy

- To increase chances of finding the right alignment, store image sets $\{A^j\}$ from N different dodecahedra ($N = 10$ in original paper)

$$D_{\text{LFD}}(A, B) = \min_{j,k=1}^N D(A^j, B^k)$$

- $(N(N - 1) + 1) \times 60$ image comparisons (= 5460 in this case)

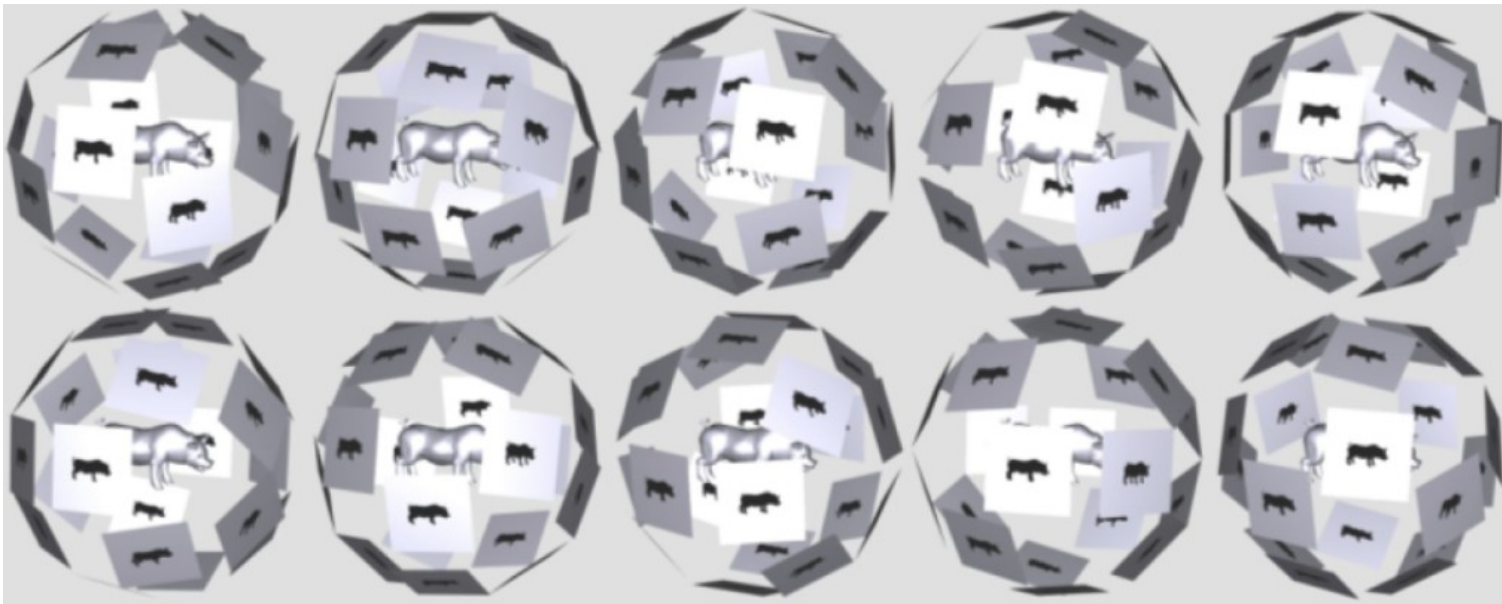


Image Comparison Metric

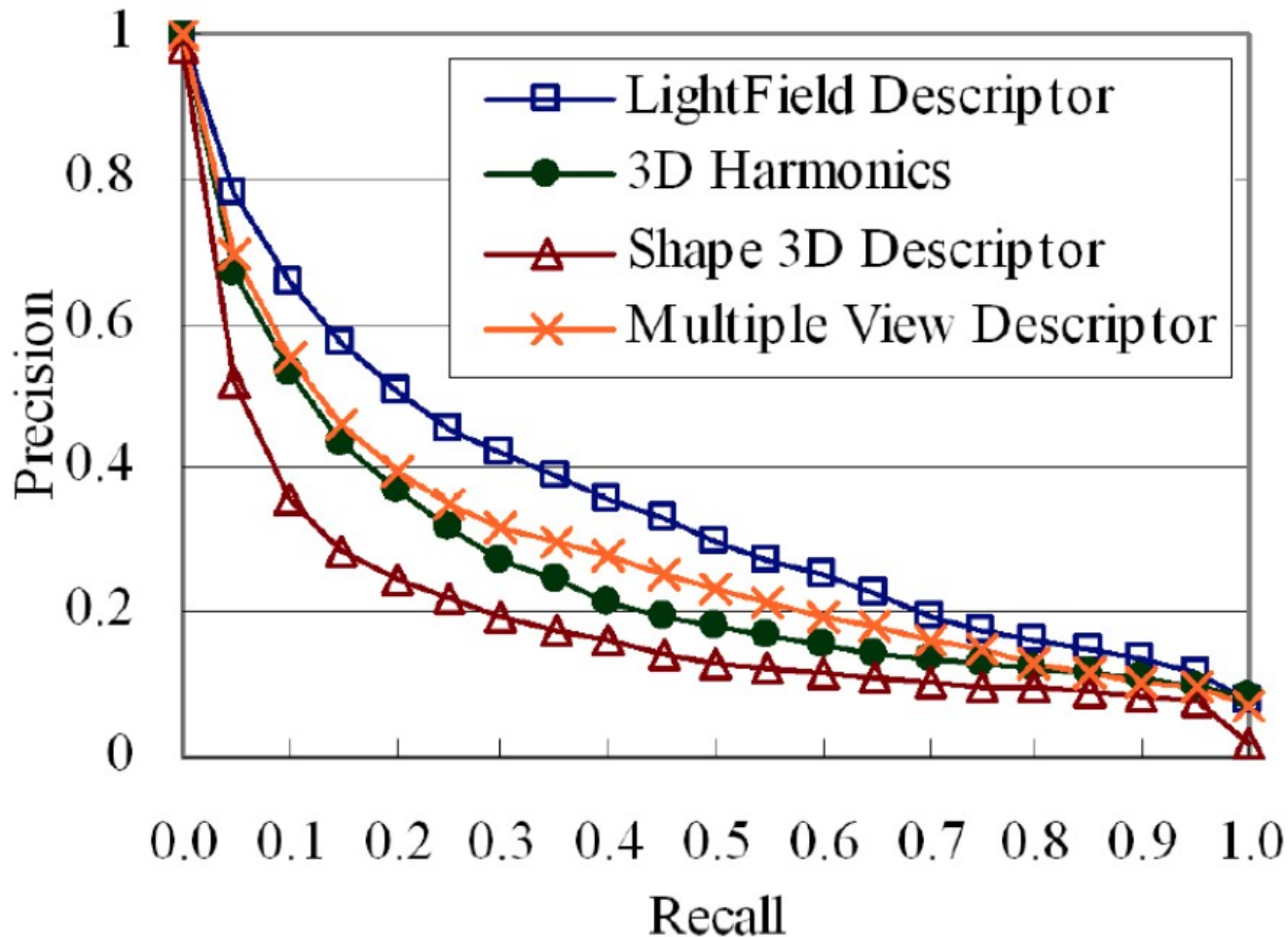
- Combine a “region-based” and a “contour-based” 2D descriptor
- Region-based descriptor
 - Combine information from all pixels in region
 - Do not emphasize boundary features
 - **Zernike Moment Descriptors** (ZMD) [35 8-bit coefficients]
- Contour-based descriptor
 - Captures only boundary information, ignoring interior
 - **Fourier Descriptors** (FD) [10 8-bit coefficients]

$$d_{\text{image}}(\text{Img}_1, \text{Img}_2) = \sum_{k=1}^{45} |C_{1,k} - C_{2,k}|$$

Querying Large Databases

- LFD is **not a natural vector space** (need to search over rotations), so can't apply traditional methods to accelerate nearest neighbor search
- **Progressively refine** descriptors for faster search
 - Use a few image sets, and a few highly quantized coefficients, to prune database and identify likely alignments
 - Progressively redo the search in the pruned database with more descriptors and more coefficients, using candidate alignments from the previous step

Results



3D Harmonics:
discussed last class

Shape 3D Descriptor:
curvature histograms

**Multiple View
Descriptor:** align
shapes using PCA,
compare views along
principal axes

Test database: 1833 shapes, with 549 shapes classified into 47 functional categories, the remaining shapes classified as "miscellaneous"

Properties of LFD

- Not very concise (100×45 coefficients)
- Reasonably quick to compute
- Not very efficient to match
- Good discrimination
- Invariant to rigid transformations
- Invariant to small deformations
- Insensitive to noise
- Insensitive to mesh topology
- Robust to degeneracies

What if we use better image descriptors?

- ZMD/FD are ok, but hardly the state of the art in modern computer vision (circa 2016)
- Convolutional Neural Nets (CNNs) have revolutionized image recognition tasks

Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
CNN	37.5%	17.0%

In 2012, the error rate in the ImageNet visual recognition challenge was halved by a deep CNN (gains are typically incremental). There are 1000 categories: the baseline of random guessing would have a 99.9% error.

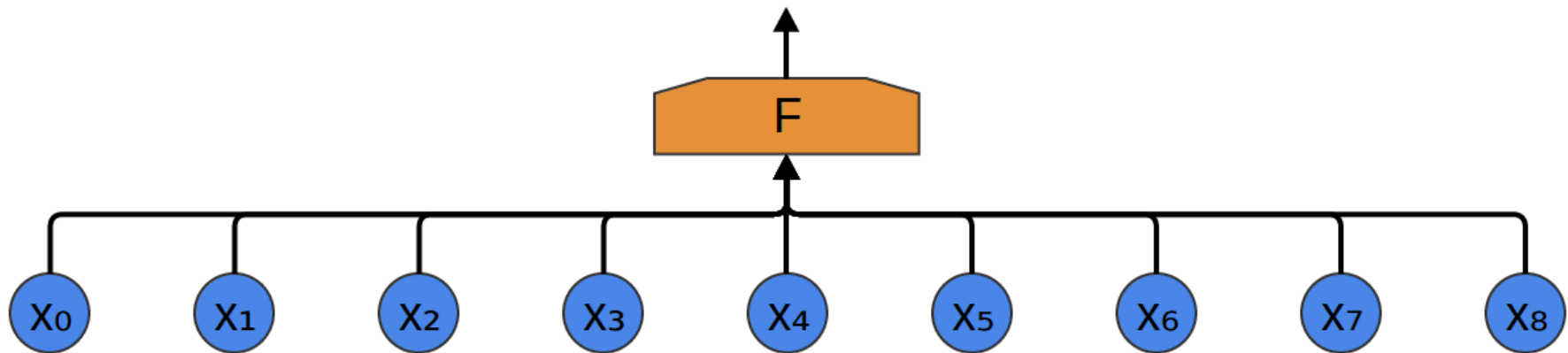
What is a Convolutional Neural Network?

- Imagine we have a set of N samples from some signal
- We want to produce a prediction, e.g. whether the signal represents a human voice, or a picture of a cat, or a depth image of a building



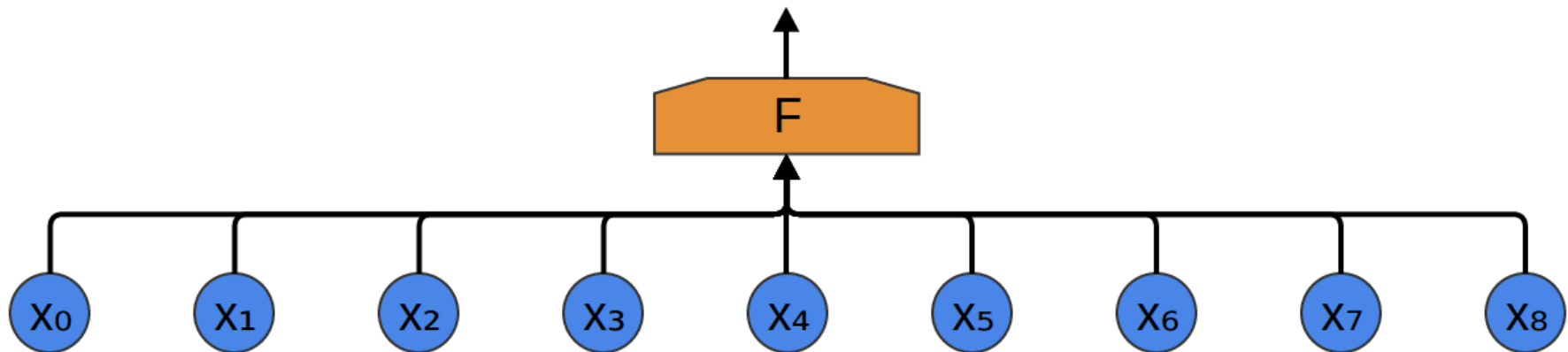
What is a Convolutional Neural Network?

- We can compute the probability as a function F of these values
 - In a **fully-connected** network, the function takes in all the inputs at once, e.g. as $g(\mathbf{w} \cdot \mathbf{x})$, where \mathbf{w} is a weight vector and g is some nonlinear transformation such as a sigmoid function



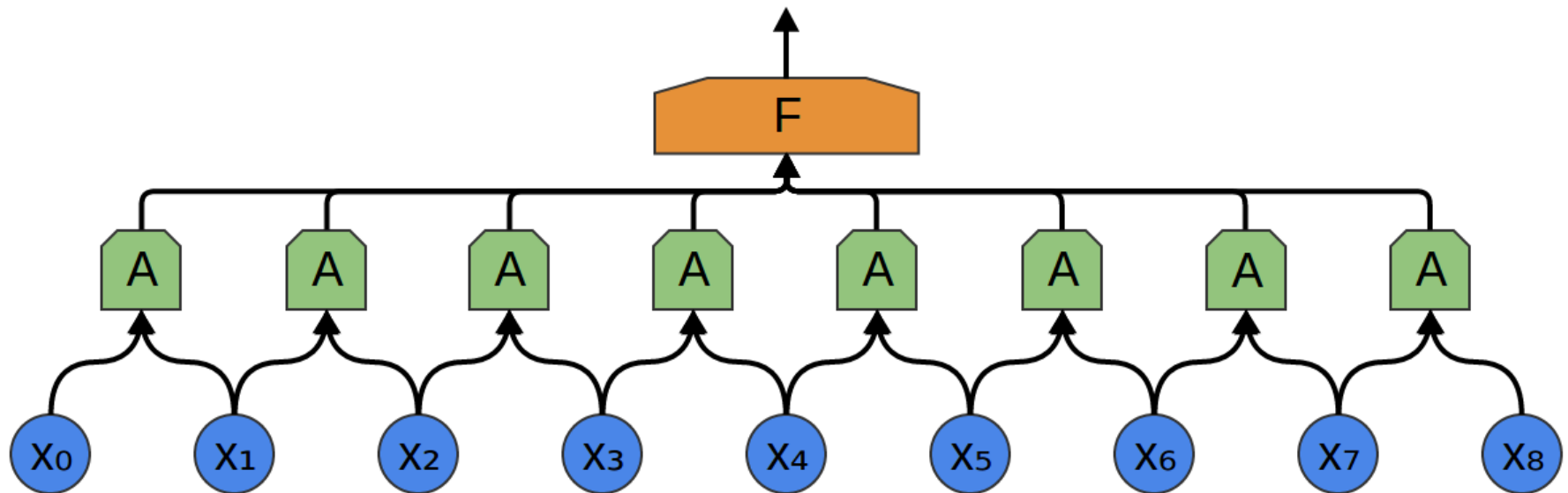
What is a Convolutional Neural Network?

- Fully-connected networks have some drawbacks
 - The function is **very high-dimensional** (all inputs processed at once)
 - **No complex relationships** between inputs is modeled (just a dot product)
 - Local information is **not captured in a “translation-invariant” way** (a feature of the signal at the left end of the sequence must be learned independently of the same feature occurring at the right end)



What is a Convolutional Neural Network?

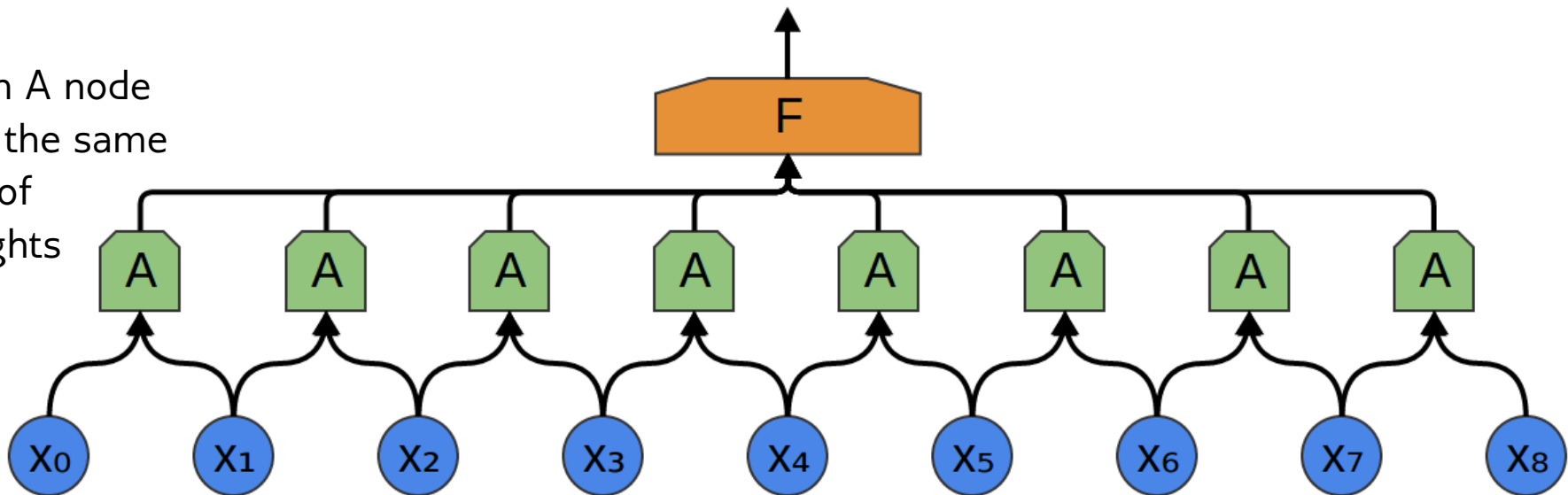
- **Solution:** a **convolutional layer**
- A filter (again, a dot product followed by a nonlinear transformation) is applied on local neighborhoods of the signal



What is a Convolutional Neural Network?

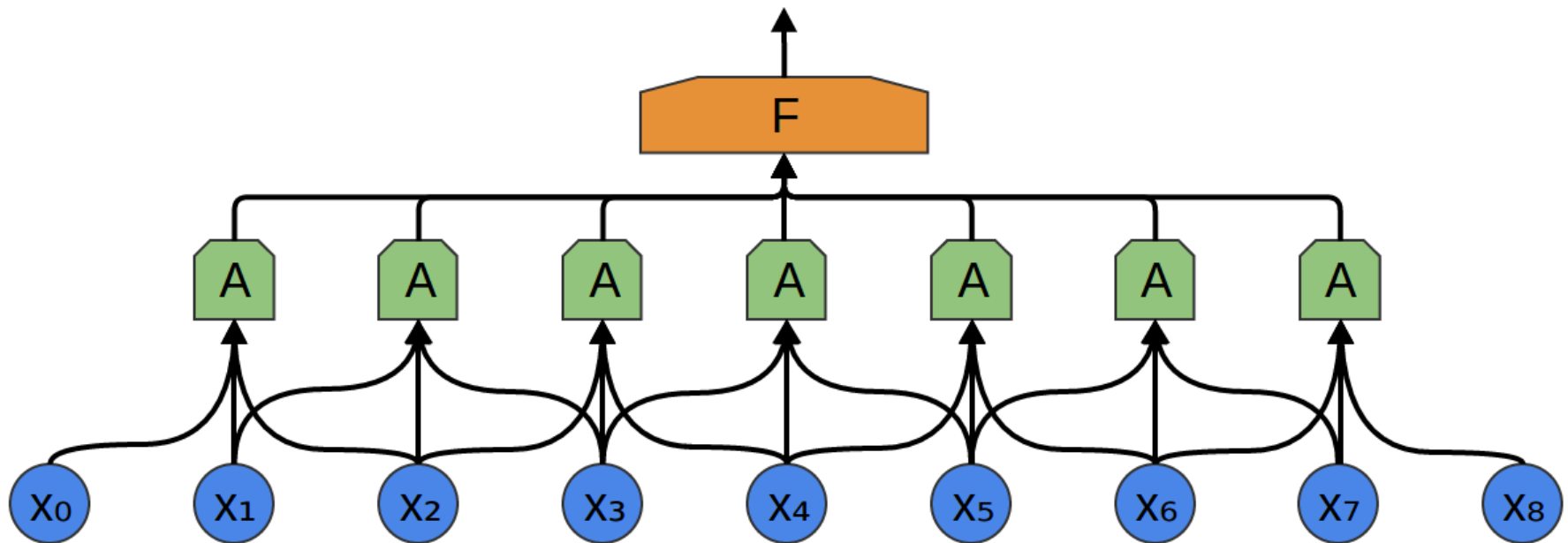
- All filters **share the same weights!**
 - Dramatically reduces number of parameters of the network
- The final output is a function of the filter responses

Each A node
has the same
set of
weights



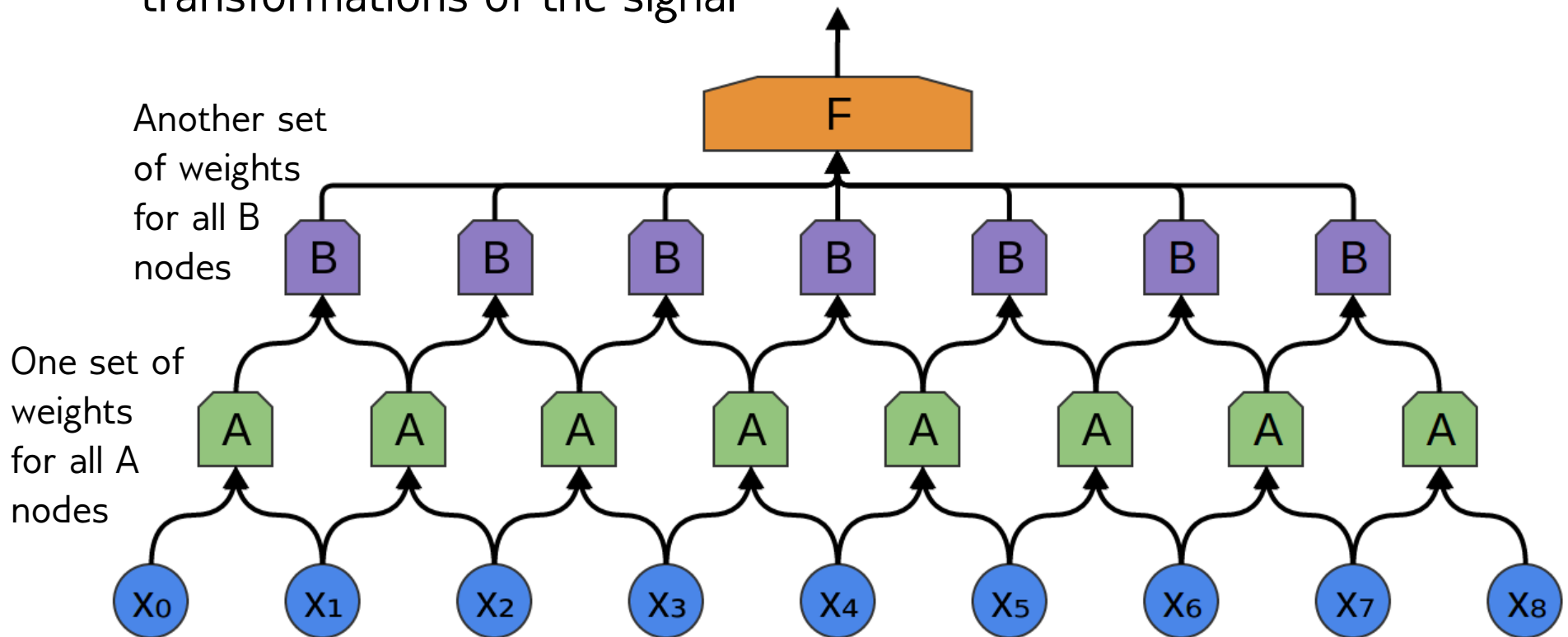
What is a Convolutional Neural Network?

- We can make the neighborhoods larger, to capture broader local features



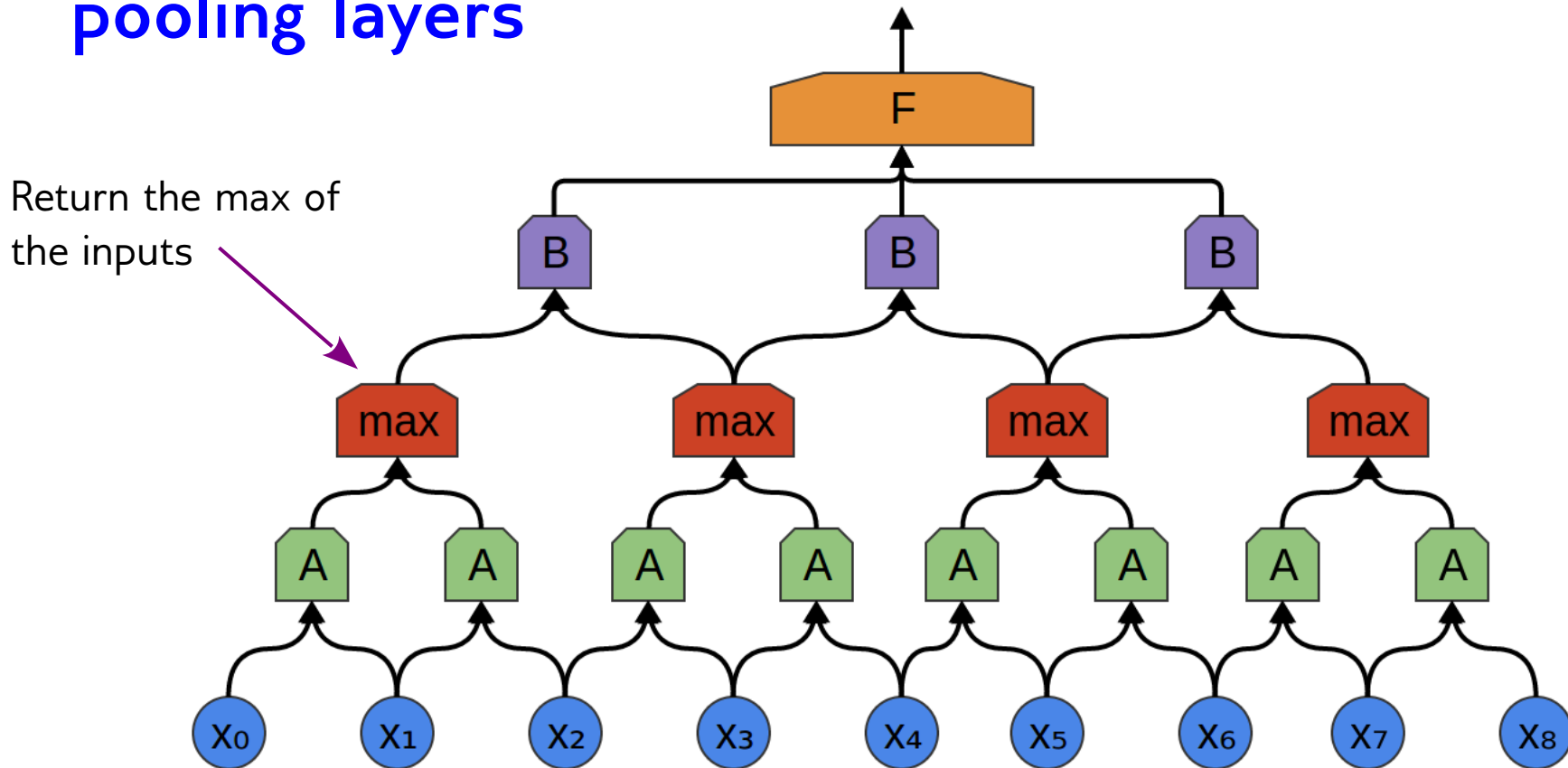
What is a Convolutional Neural Network?

- Convolutional layers are **composable**: they can be stacked with each layer providing inputs for the next layer
 - Higher layers can capture more abstract features since they effectively cover larger neighborhoods, and combine multiple different nonlinear transformations of the signal



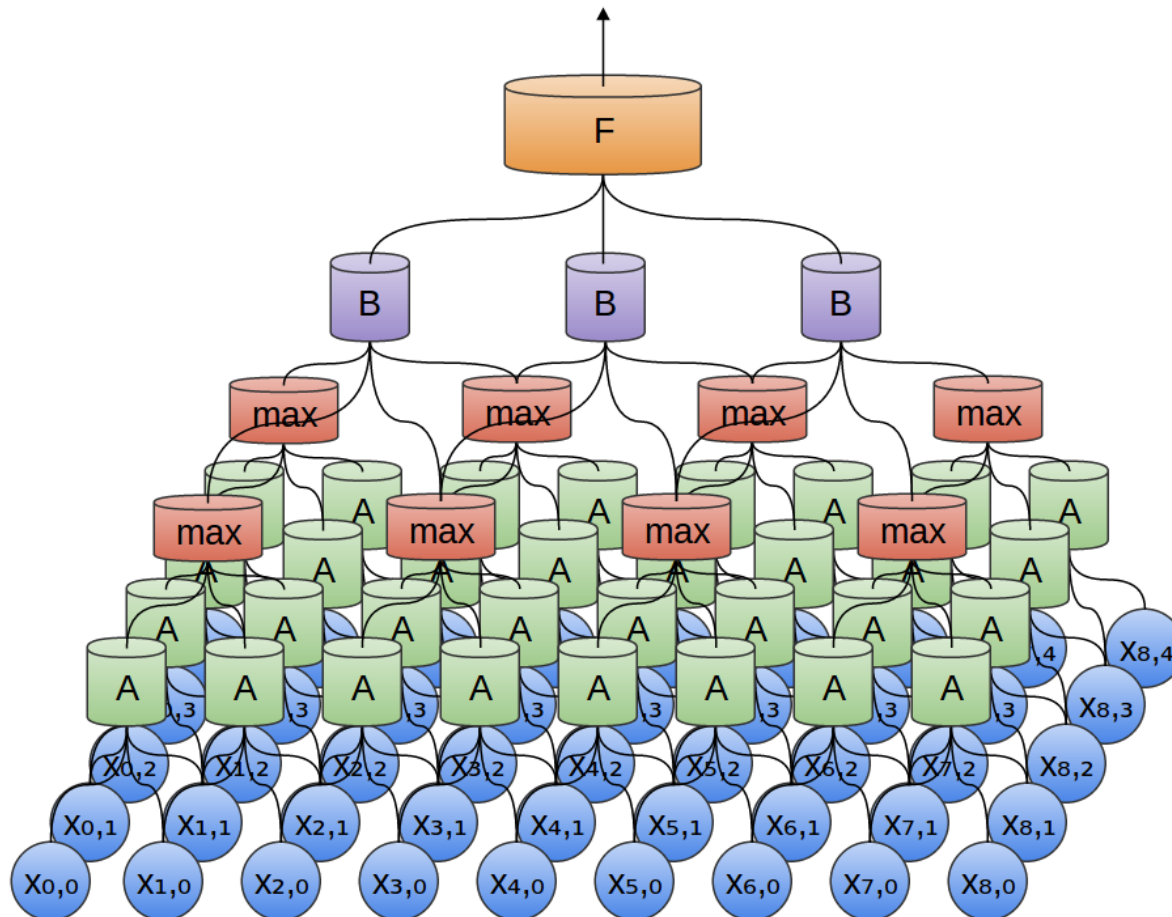
What is a Convolutional Neural Network?

- To make the network robust to small translations in detected features, and to reduce the amount of redundant data fed into higher layers, we introduce **pooling layers**



What is a Convolutional Neural Network?

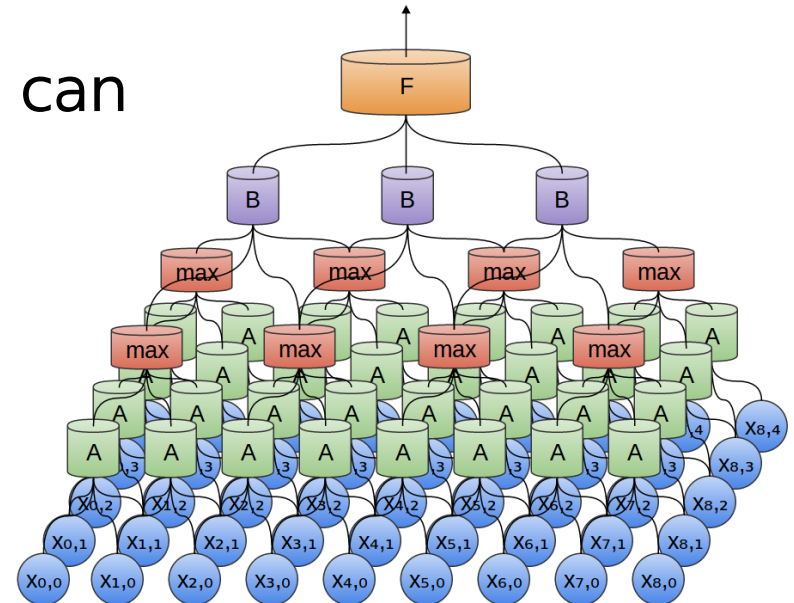
- The signal can be 2D: the filters are now also 2D, but it's all essentially the same



What is a Convolutional Neural Network?

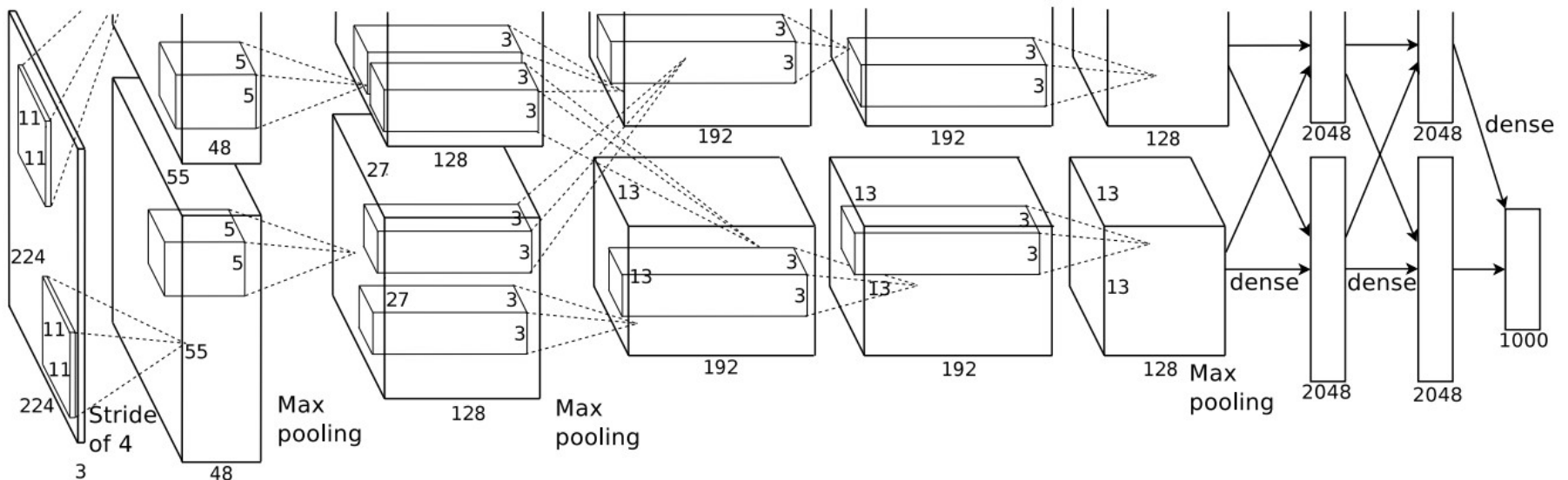
- The function computed by this gigantic model is **differentiable*** w.r.t. the weights
 - Given training data and a **loss function** measuring the deviation from predicted and actual values, we can optimize the weights by gradient descent
 - The gradient of the loss function can be found efficiently by a method called **back-propagation**

* nearly everywhere

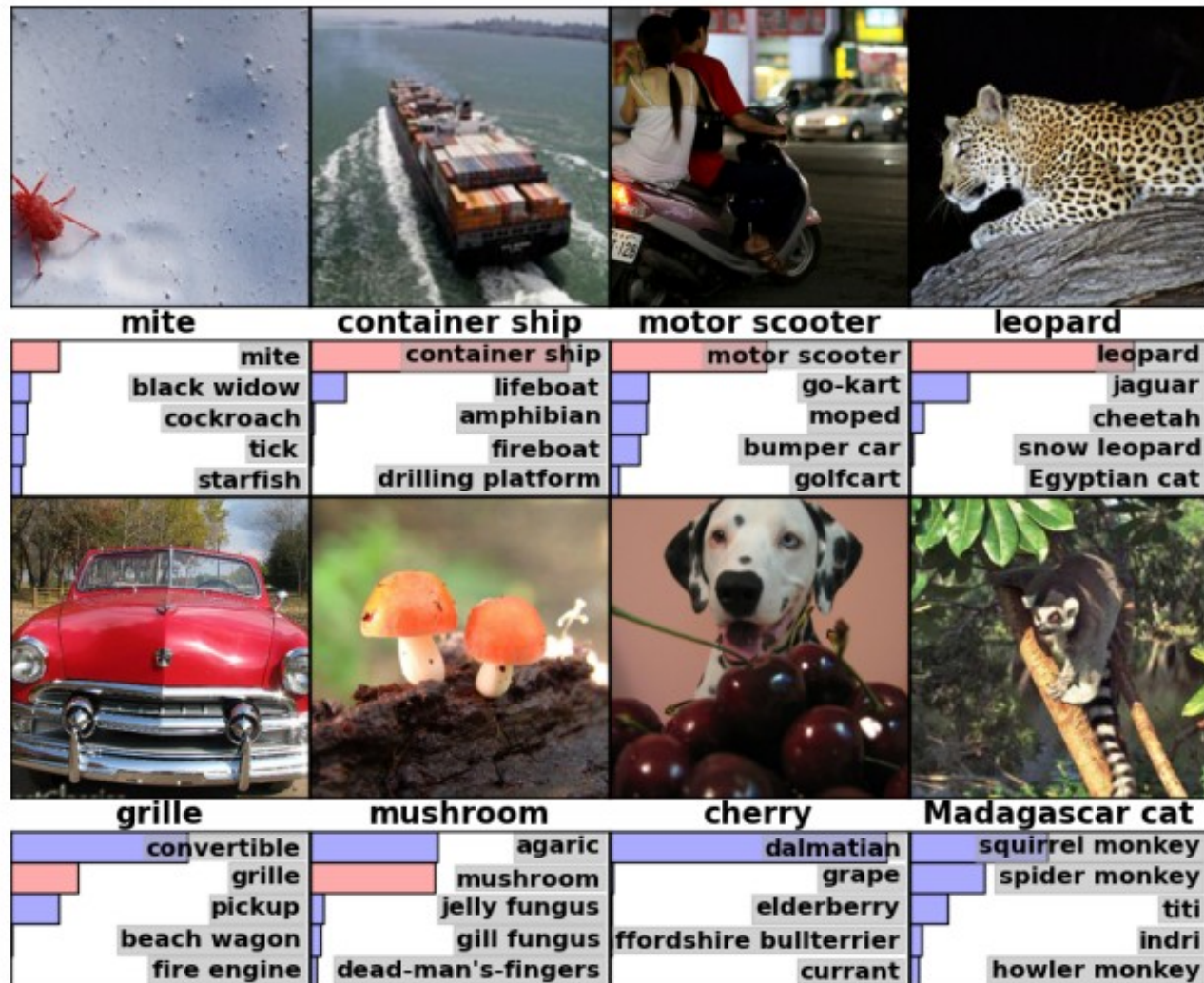


A real-world CNN

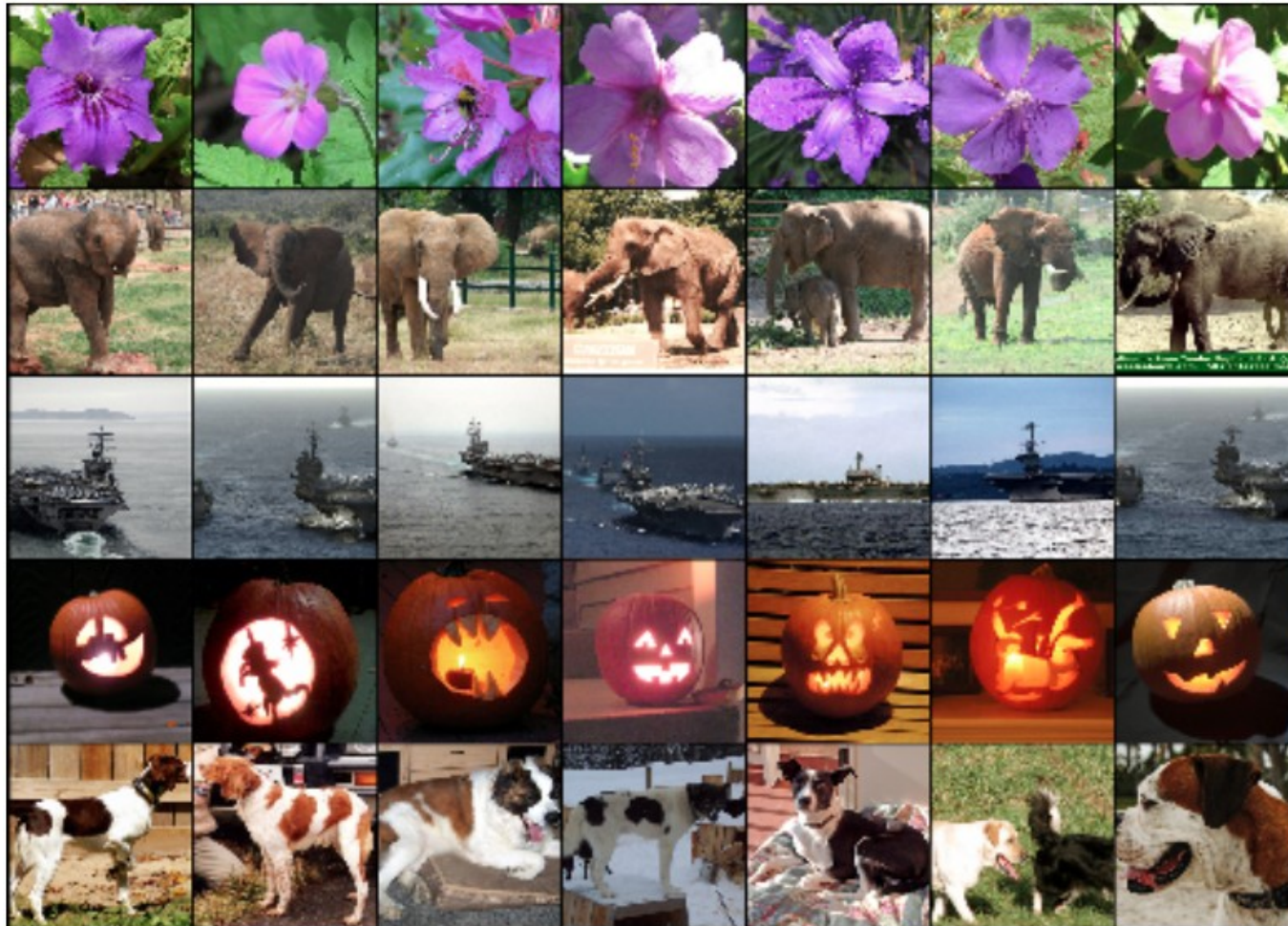
- 5 convolutional layers, 3 max-pooling layers, 3 fully-connected layers
- ~60 million parameters (despite the weight sharing!)



Using the CNN for classification



Using the CNN for retrieval



The descriptor is the vector of neuron activations in the second last layer

Query

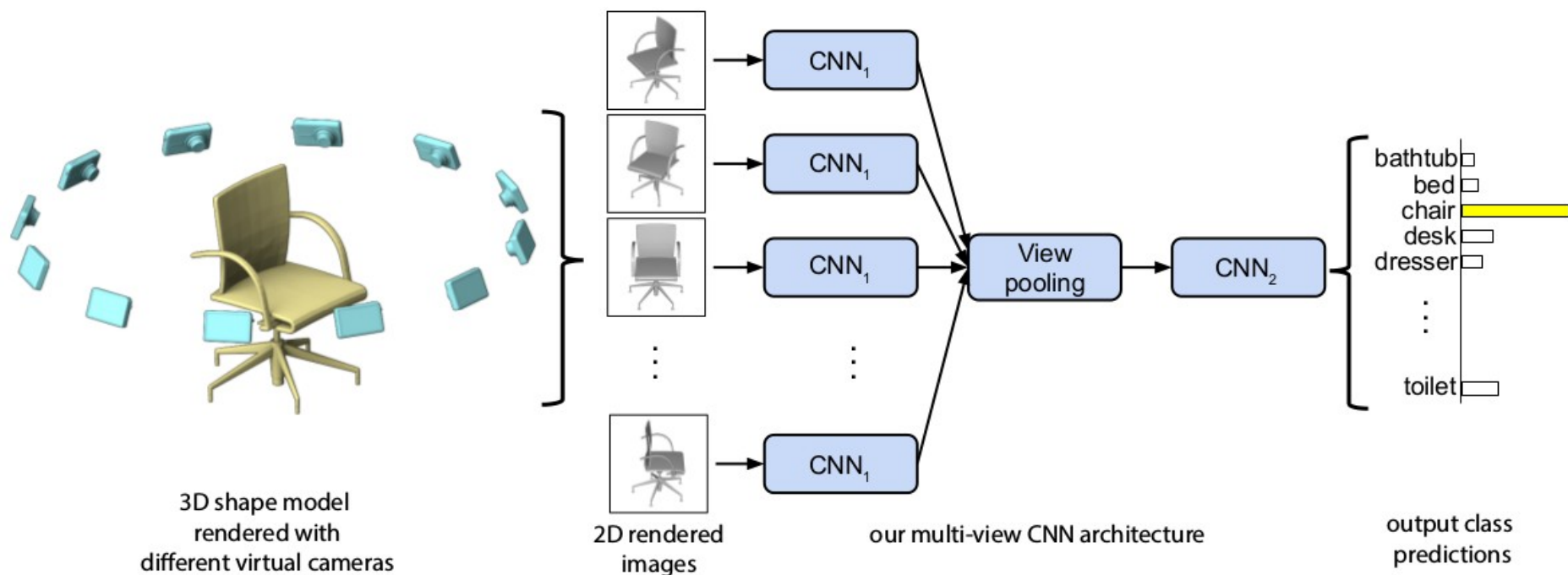
Top 6 results

Image CNN for 3D shapes

- Let's take a CNN trained on a (huge) image database, and use it to analyze views of 3D shapes
 - **Render** a 3D shape from an arbitrary viewpoint
 - Pass it through the **pre-trained CNN** and take the neuron activations in the second-last layer as the descriptor
 - For more accuracy, **fine-tune** the network on a training set of rendered shapes before testing
- Just this alone, with a single view (from an unknown direction) of the shape, bumps up the mAP retrieval accuracy (area under PR curve) on a 40-class, 12K-shape collection from 40.9% (LFD) to **61.7%**.
 - An LFD-like approach with 12 views/shape further improves to **62.8%**

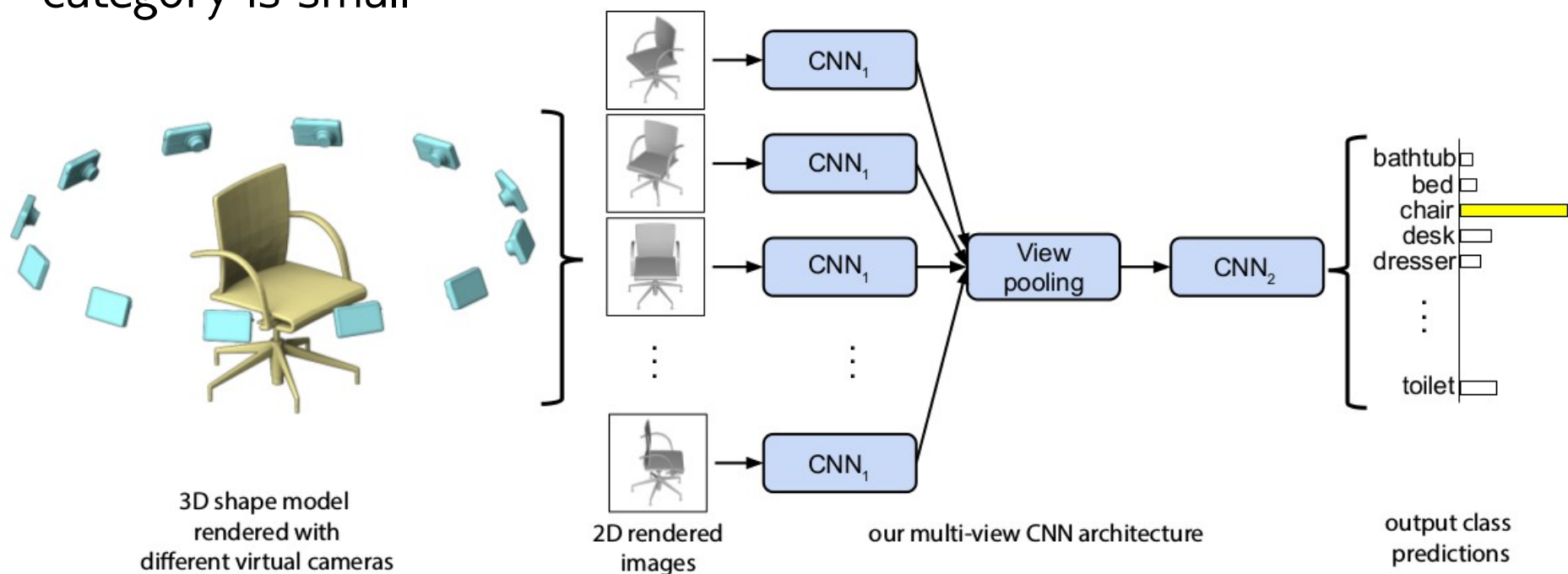
Combining Views

- A smarter way to aggregate information from multiple views
 - Take the output signal of the last convolutional layer of the base network (CNN_1) from each view, and combine them, element-by-element, using a max-pooling operation
 - Pass this **view-pooled** signal through the rest of the network (CNN_2)

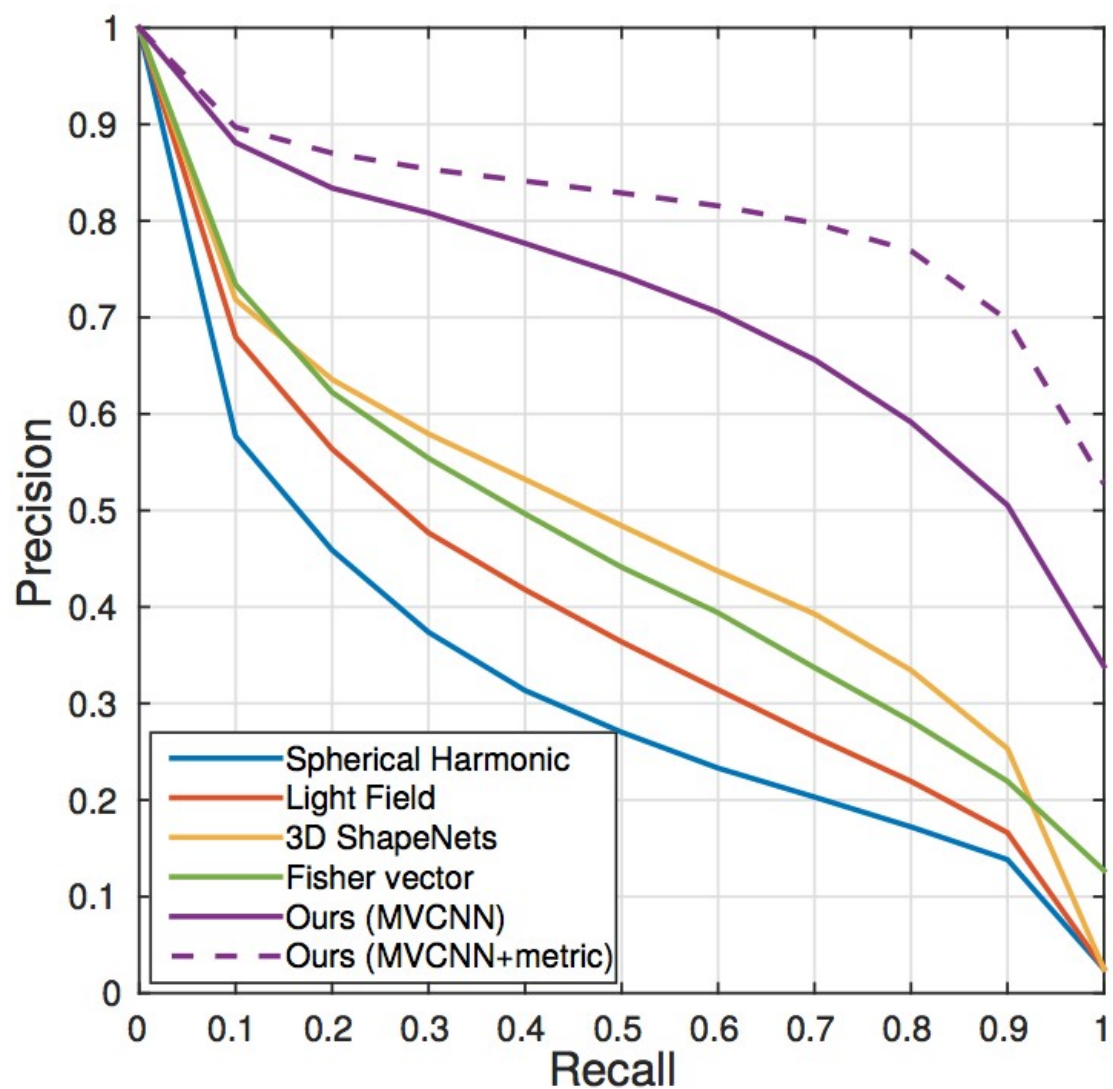


Combining Views

- The view-pooled CNN can still be trained (in exactly the same way) using back-propagation and gradient descent
- For retrieval, the descriptor from the second-last layer can be further tuned by learning a Mahalanobis metric (a projection of the descriptors) where the distance between shapes of the same training category is small



How well does this work?



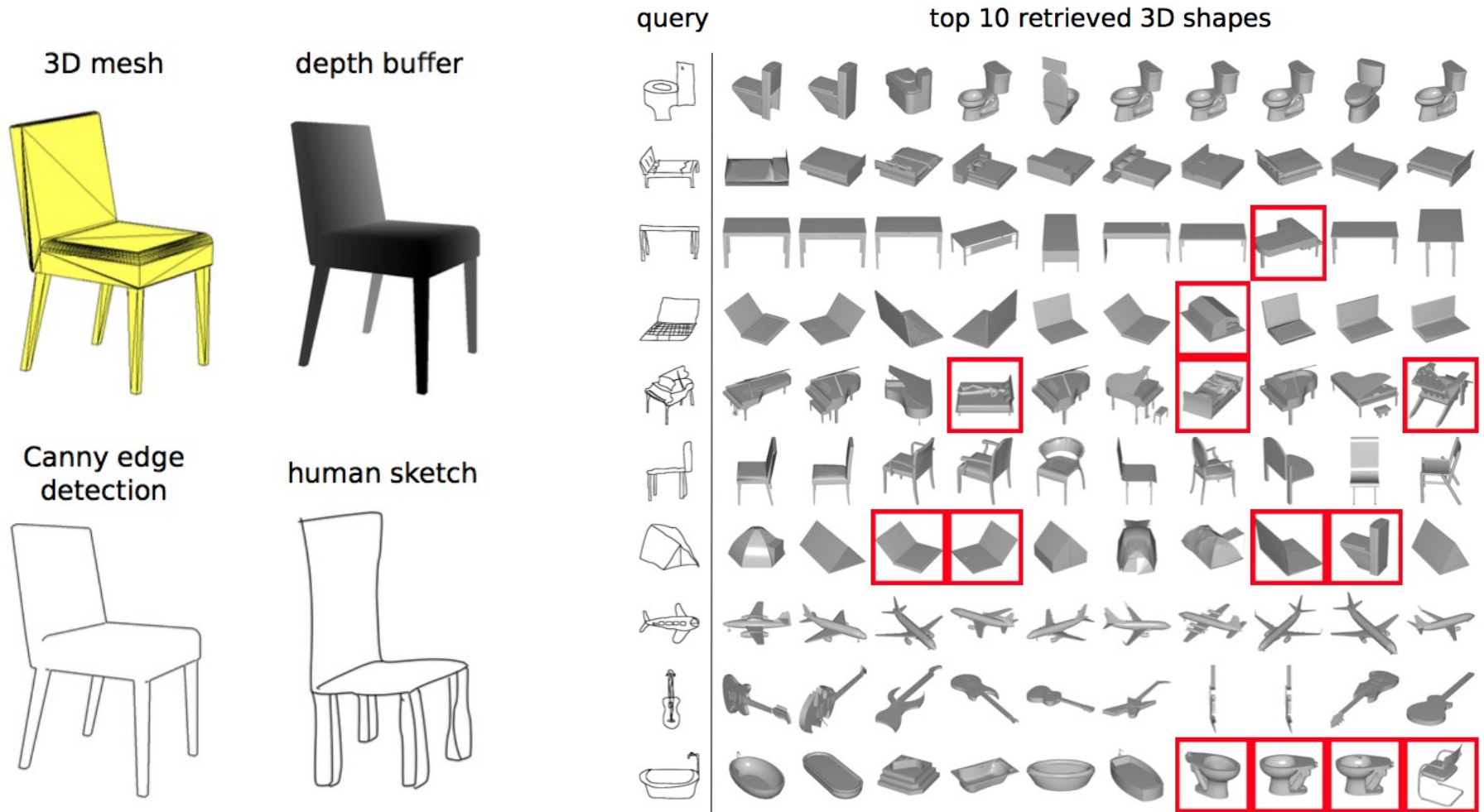
How well does this work?

Method	Training Config.			Test Config.	Classification (Accuracy)	Retrieval (mAP)
	Pre-train	Fine-tune	#Views	#Views		
(1) SPH [16]	-	-	-	-	68.2%	33.3%
(2) LFD [5]	-	-	-	-	75.5%	40.9%
(3) 3D ShapeNets [37]	ModelNet40	ModelNet40	-	-	77.3%	49.2%
(4) FV	-	ModelNet40	12	1	78.8%	37.5%
(5) FV, 12×	-	ModelNet40	12	12	84.8%	43.9%
(6) CNN	ImageNet1K	-	-	1	83.0%	44.1%
(7) CNN, f.t.	ImageNet1K	ModelNet40	12	1	85.1%	61.7%
(8) CNN, 12×	ImageNet1K	-	-	12	87.5%	49.6%
(9) CNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	88.6%	62.8%
(10) MVCNN, 12×	ImageNet1K	-	-	12	88.1%	49.4%
(11) MVCNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	89.9%	70.1%
(12) MVCNN, f.t.+metric, 12×	ImageNet1K	ModelNet40	12	12	89.5%	80.2%
(13) MVCNN, 80×	ImageNet1K	-	80	80	84.3%	36.8%
(14) MVCNN, f.t., 80×	ImageNet1K	ModelNet40	80	80	90.1%	70.4%
(15) MVCNN, f.t.+metric, 80×	ImageNet1K	ModelNet40	80	80	90.1%	79.5%

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

A side benefit of view-based representations

- The MVCNN can be fine-tuned to retrieve 3D models based on hand-drawn 2D sketches



Properties of MVCNN

- Not very concise (4096 second-last layer neurons)
- Reasonably quick to compute (render and pass through CNN)
- Efficient to compare (natural vector space)
- Good discrimination
- Invariant to rigid transformations
- Invariant to small deformations
- Insensitive to noise
- Insensitive to mesh topology
- Robust to degeneracies