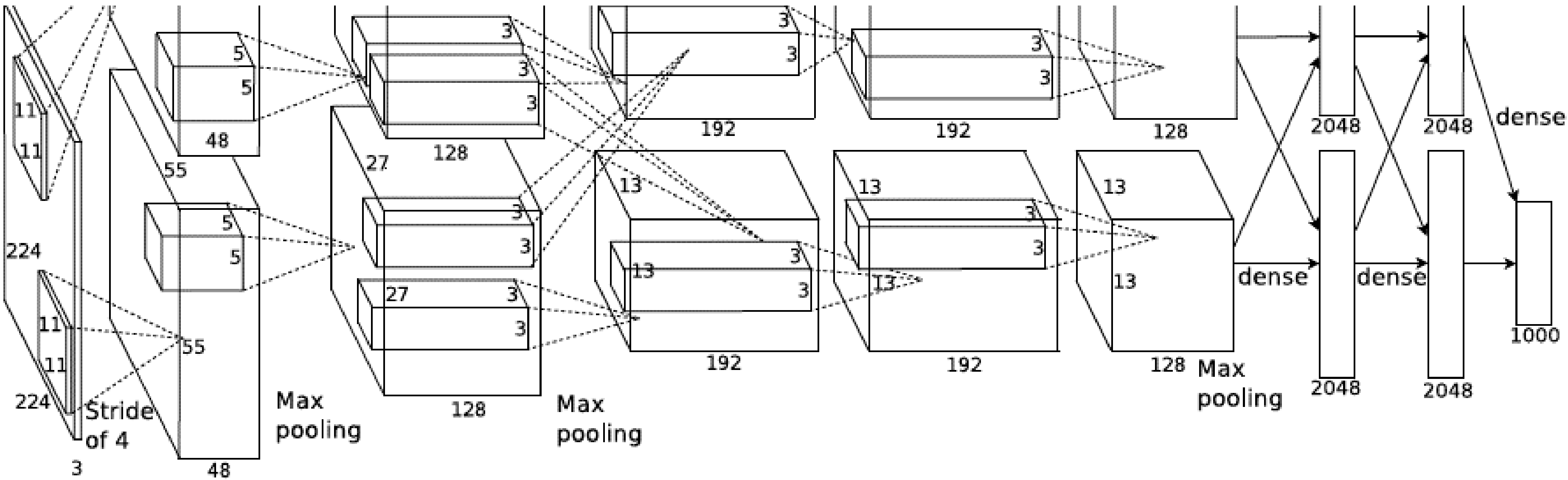
Convolutional Neural Networks



Computer Vision

Jia-Bin Huang, Virginia Tech

Today’s class

- Overview
- Convolutional Neural Network (CNN)
- Understanding and Visualizing CNN
- Training CNN

Image Categorization: Training phase

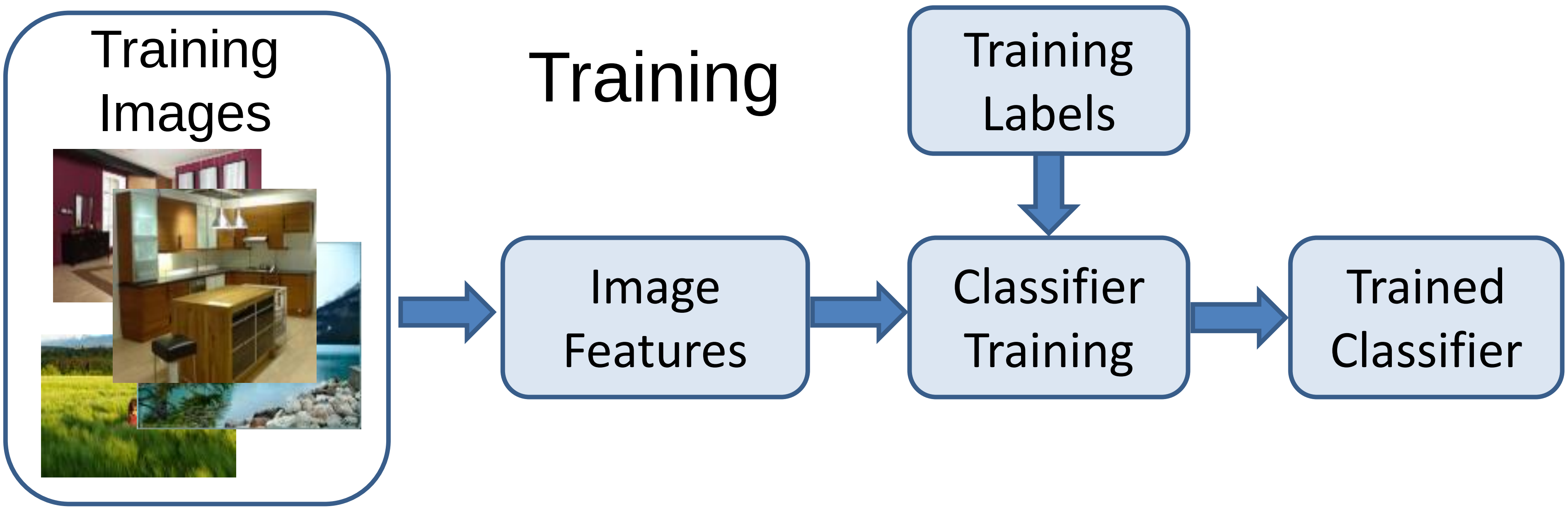
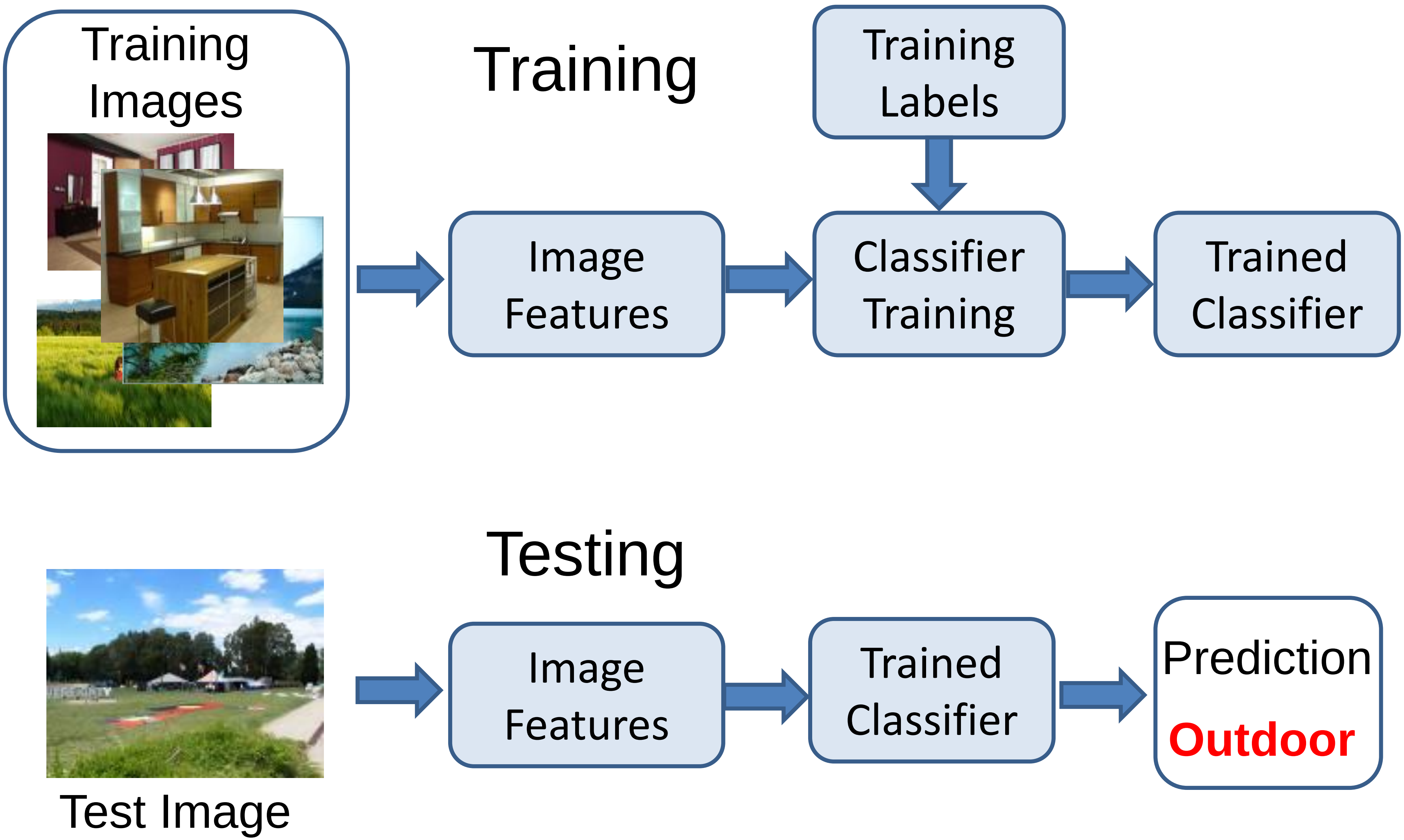
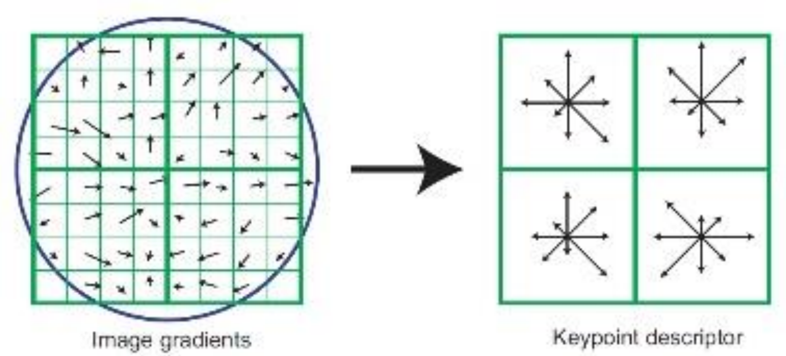


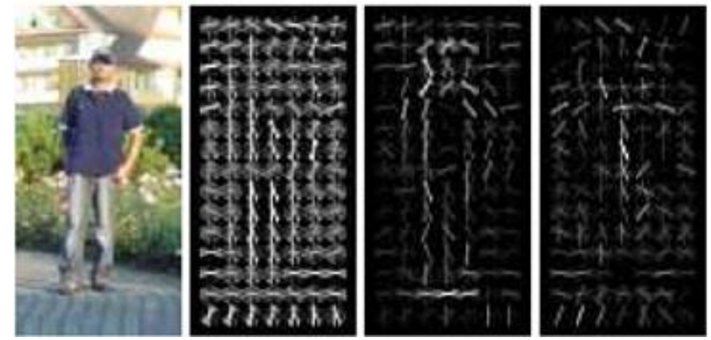
Image Categorization: Testing phase



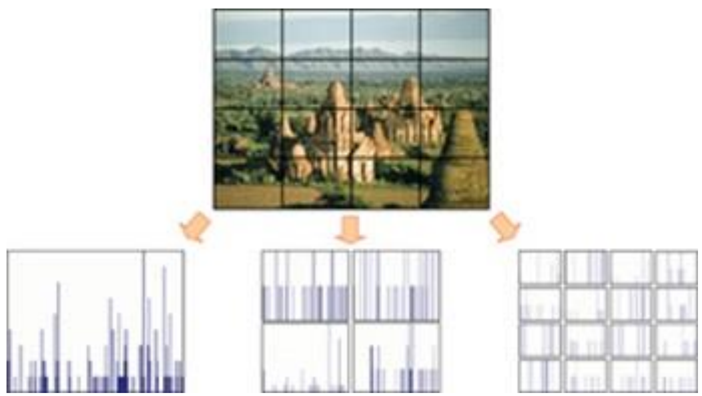
Features are the Keys



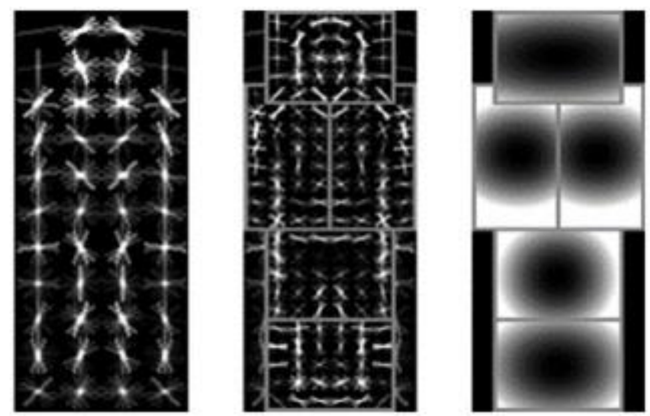
SIFT [Loewe IJCV 04]



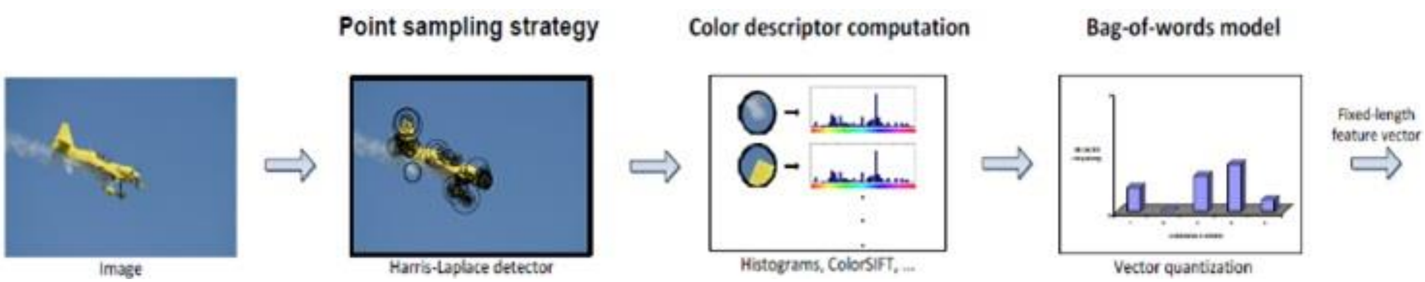
HOG [Dalal and Triggs CVPR 05]



SPM [Lazebnik et al. CVPR 06]



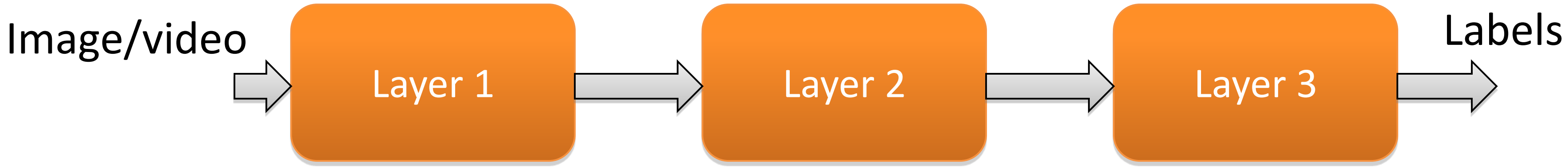
DPM [Felzenszwalb et al. PAMI 10]



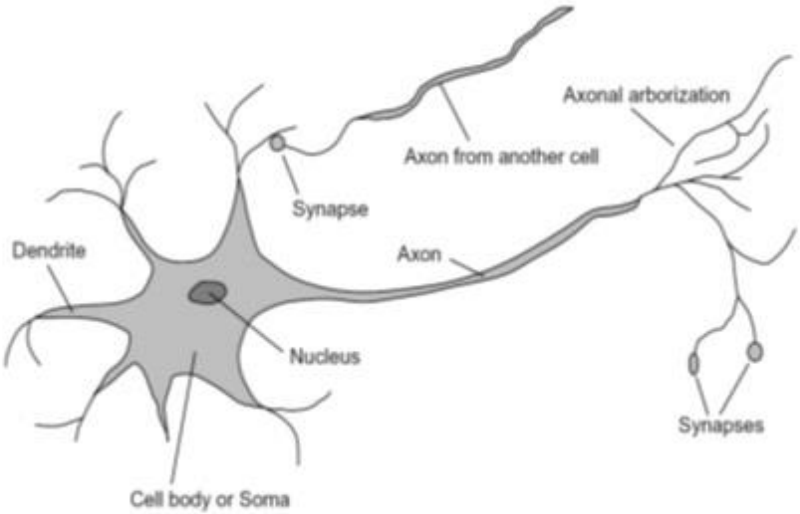
Color Descriptor [Van De Sande et al. PAMI 10]

Learning a Hierarchy of Feature Extractors

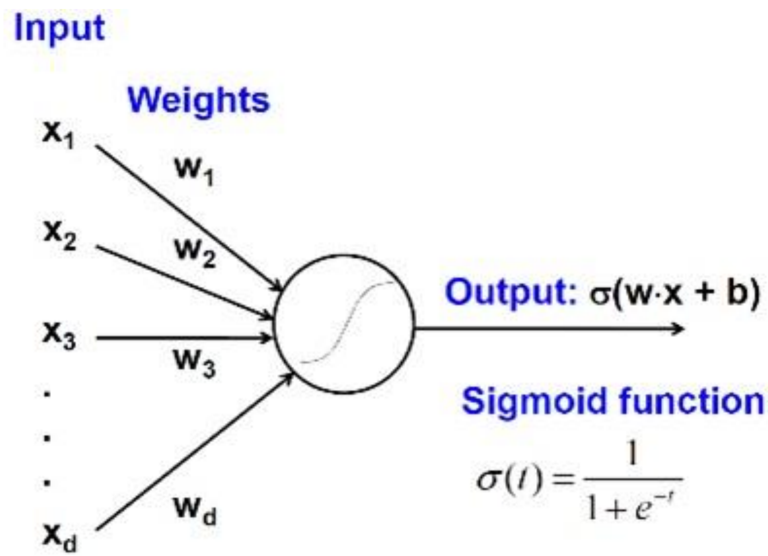
- Each layer of hierarchy extracts features from output of previous layer
- All the way from pixels → classifier
- Layers have the (nearly) same structure



Biological neuron and Perceptrons



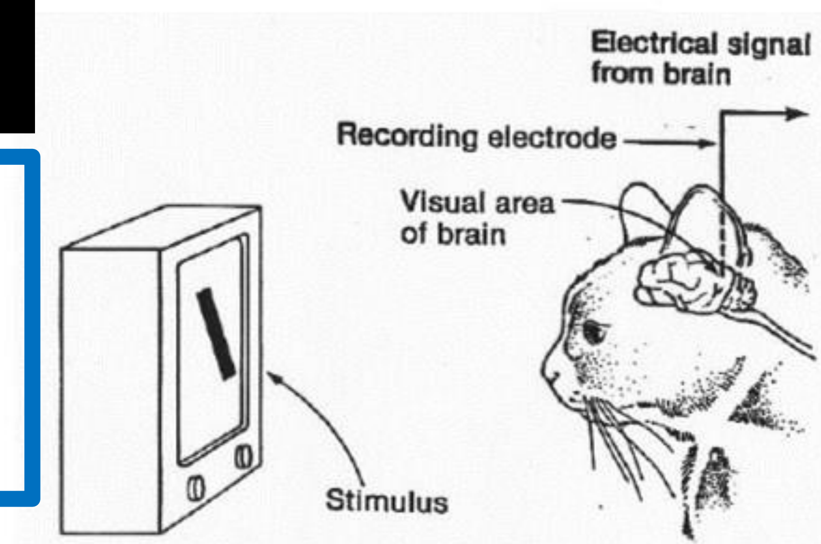
A biological neuron



An artificial neuron (Perceptron)
- a linear classifier

Simple, Complex and Hypercomplex cells

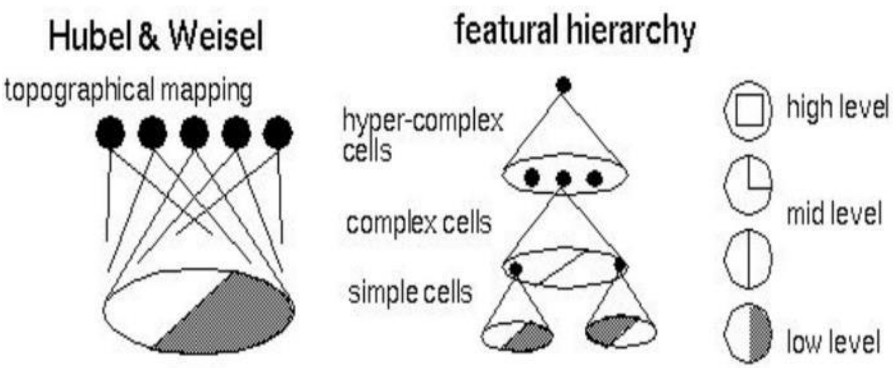
David H. Hubel and Torsten Wiesel



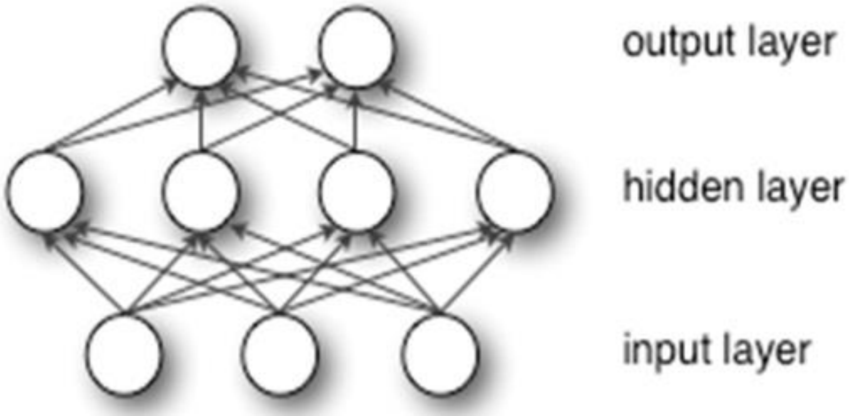
Suggested a **hierarchy** of **feature detectors** in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells.

David Hubel's Eye, Brain, and Vision

Hubel/Wiesel Architecture and Multi-layer Neural Network



Hubel and Weisel's architecture



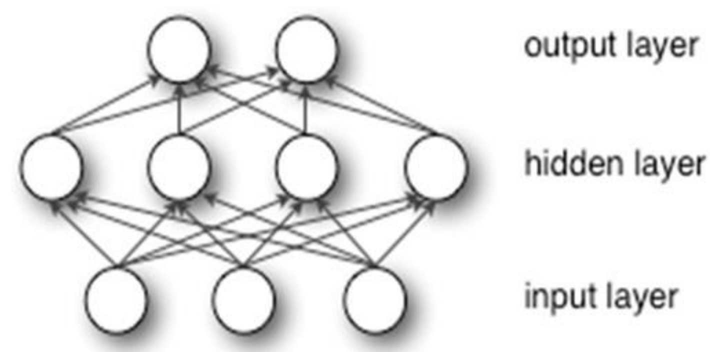
Multi-layer Neural Network
- A *non-linear* classifier

Multi-layer Neural Network

- A non-linear classifier
- **Training:** find network weights **w** to minimize the error between true training labels y_i and estimated labels $f_{\boldsymbol{w}}(\boldsymbol{x_i})$

$$E(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - f_{\mathbf{w}}(\mathbf{x}_i)\right)^2$$

- Minimization can be done by gradient descent provided f is differentiable
- This training method is called **back-propagation**

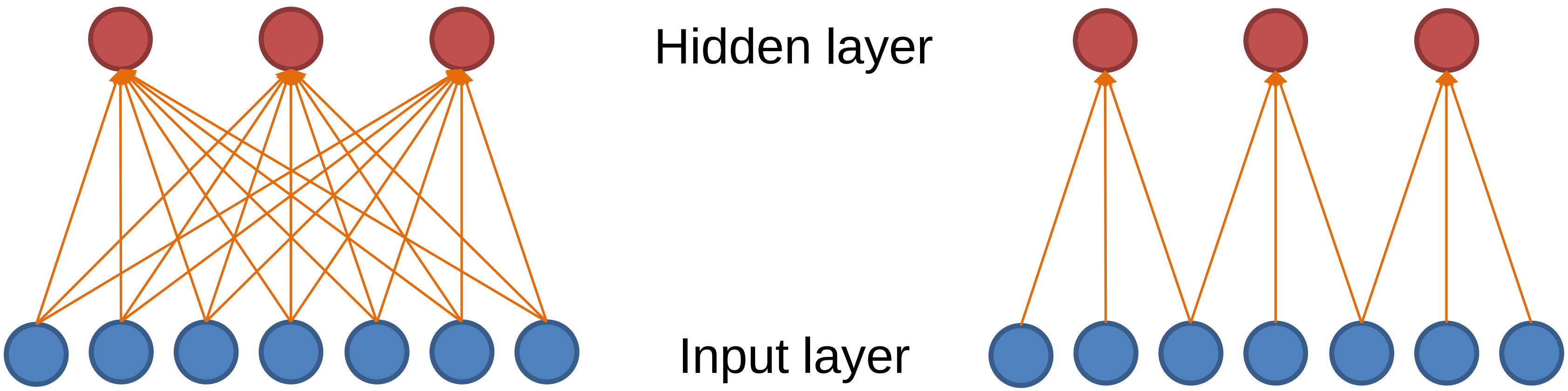


Convolutional Neural Networks

- Also known as CNN, ConvNet, DCN

- CNN = a multi-layer neural network with
  1. Local connectivity
  2. Weight sharing

CNN: Local Connectivity

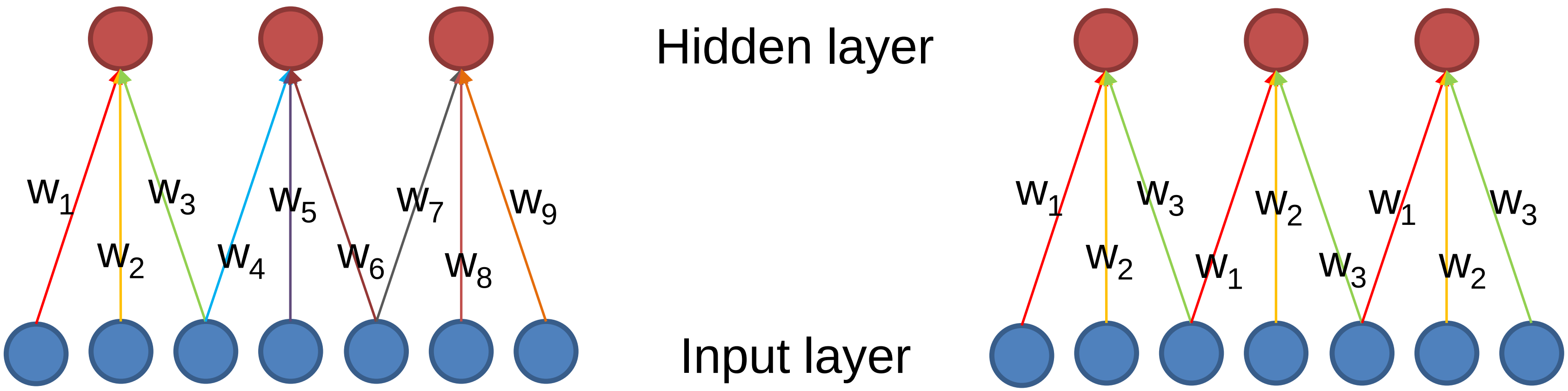


Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Global connectivity: 3 x 7 = 21
  - Local connectivity: 3 x 3 = 9

CNN: Weight Sharing

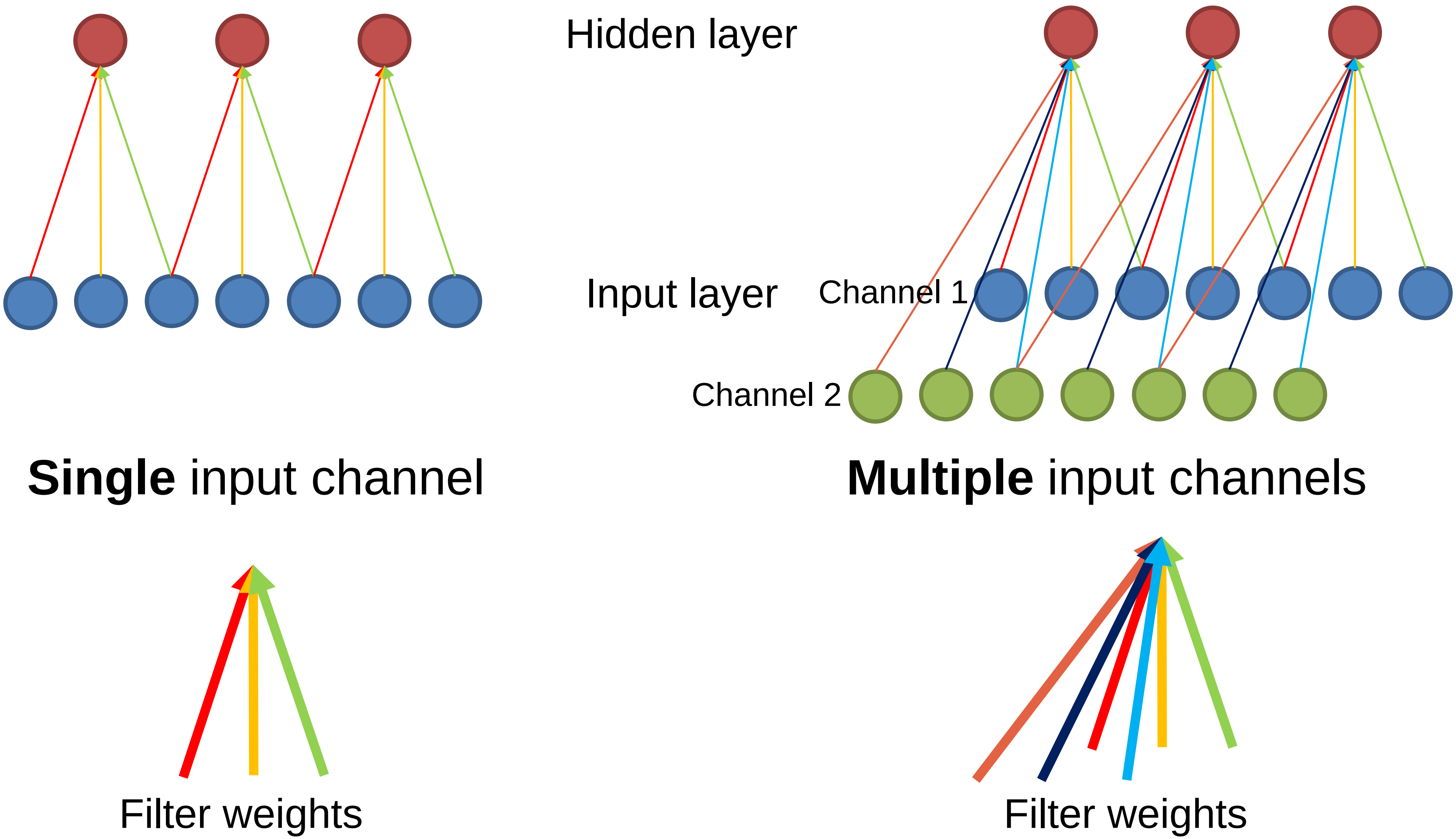


Without weight sharing

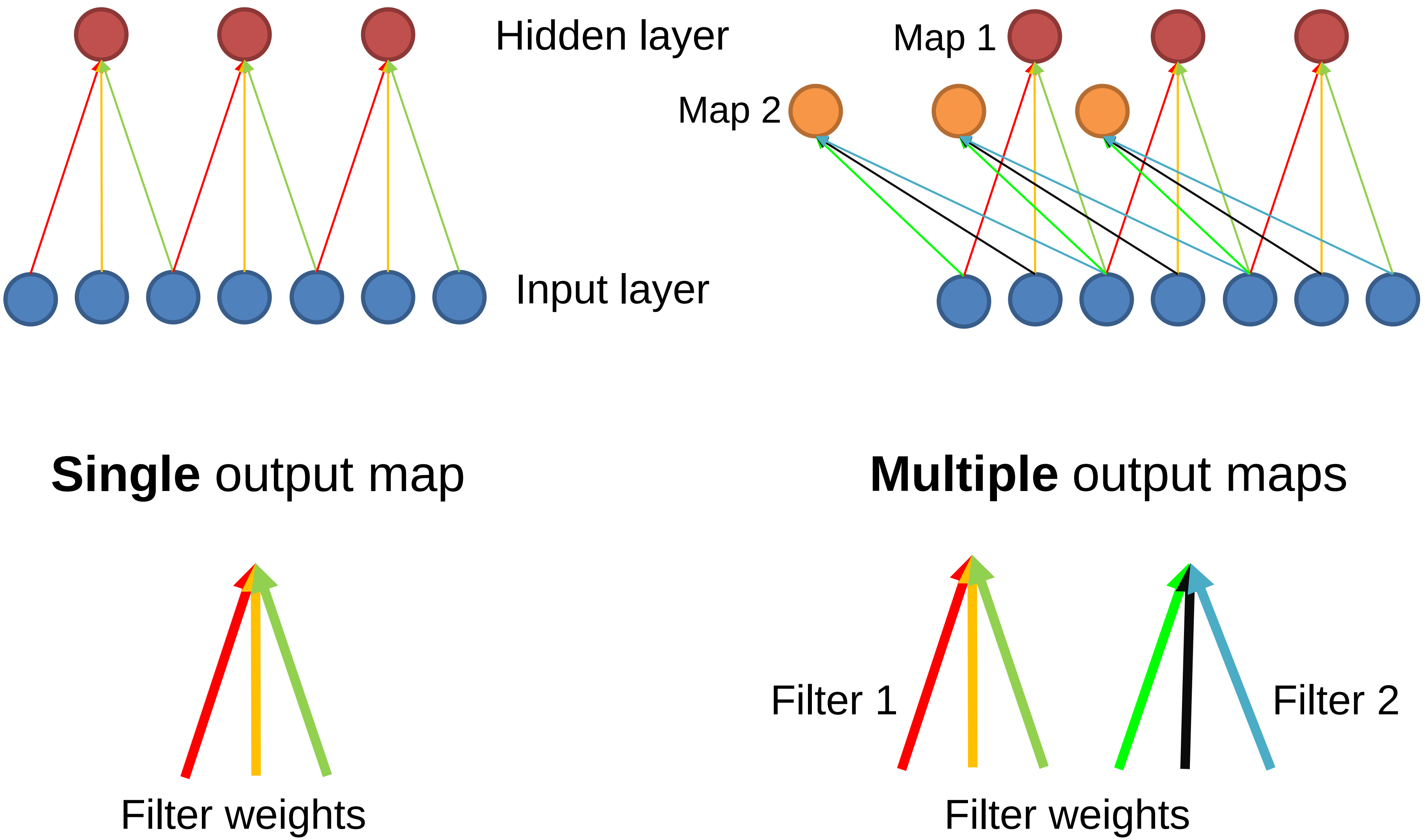
With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing: 3 x 3 = 9
  - With weight sharing : 3 x 1 = 3

CNN with multiple input channels



CNN with multiple output maps



Putting them together

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps

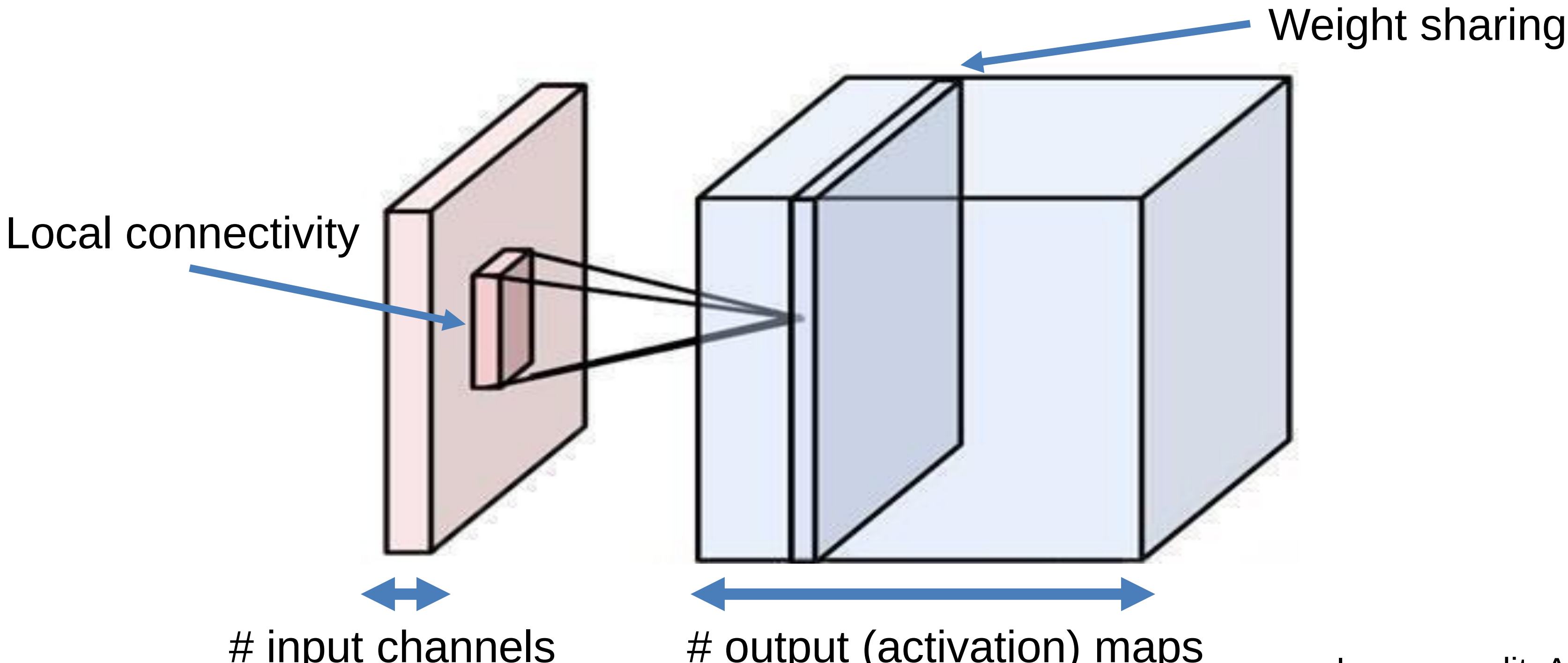


Image credit: A. Karpathy

Neocognitron [Fukushima, Biological Cybernetics 1980]

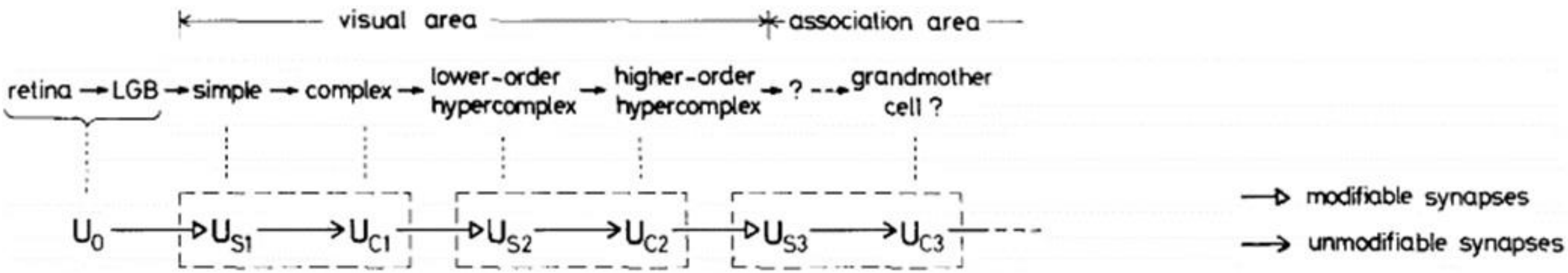
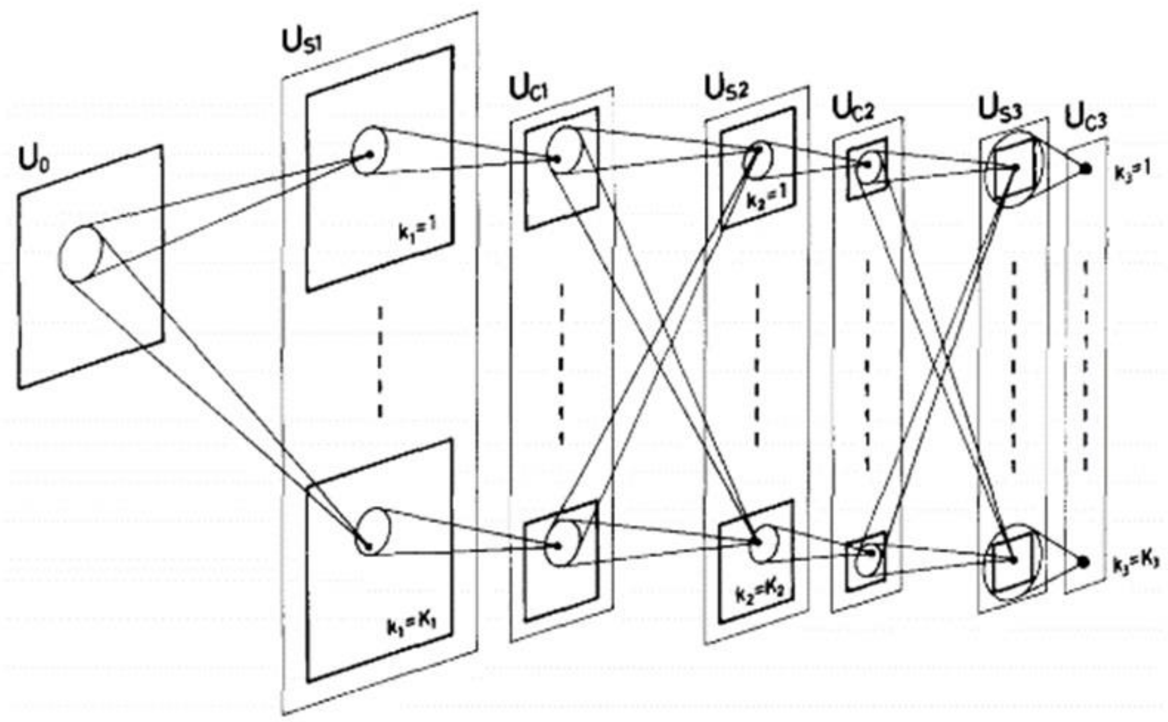


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

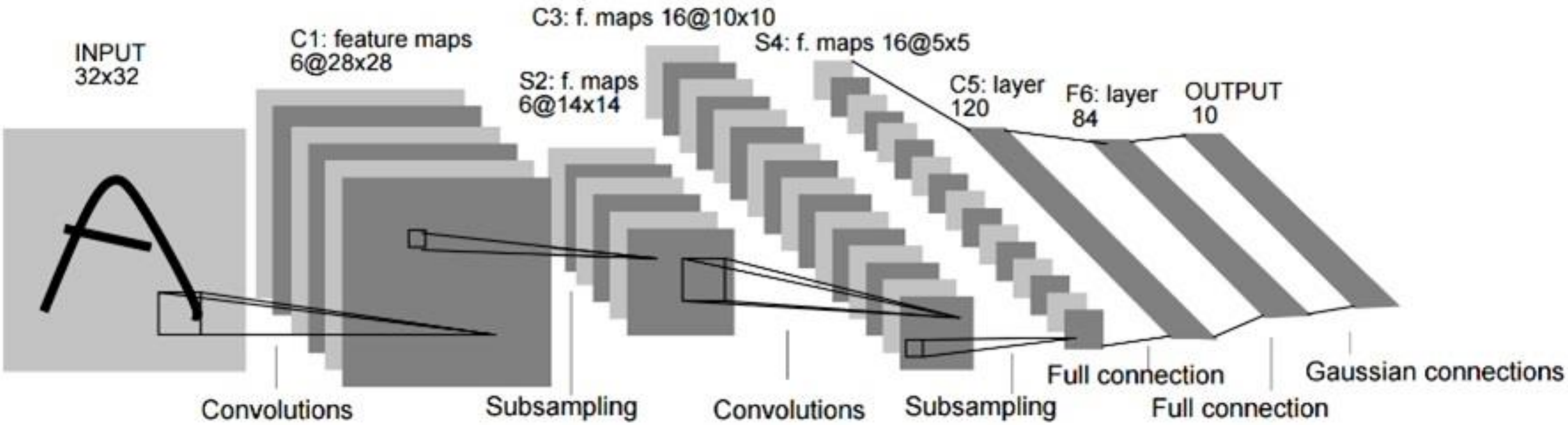


Deformation-Resistant Recognition

S-cells: (simple)
- extract local features

C-cells: (complex)
- allow for positional errors

LeNet [LeCun et al. 1998]

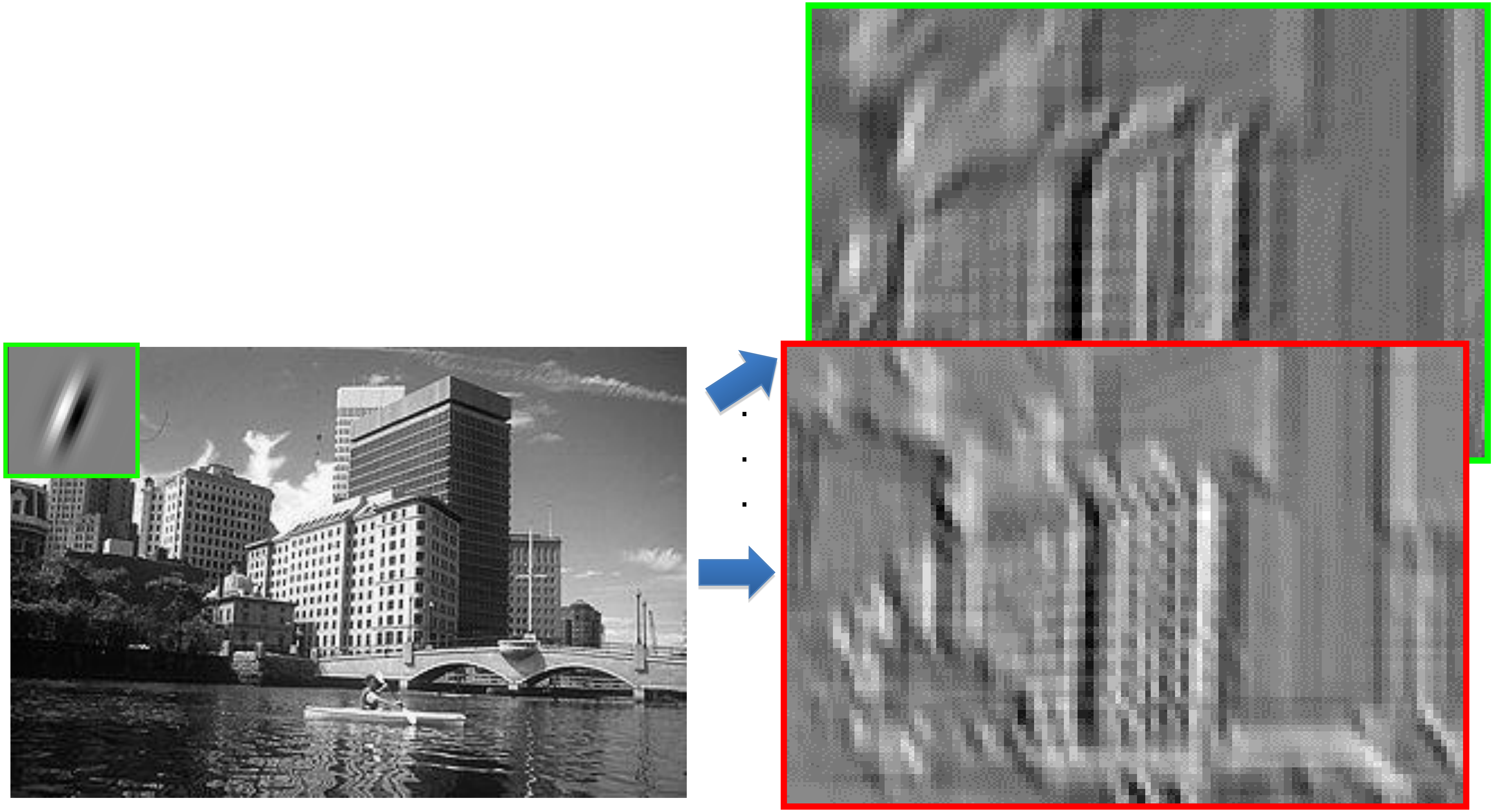


LeNet-1 from 1993

Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998]

What is a Convolution?

- Weighted moving sum

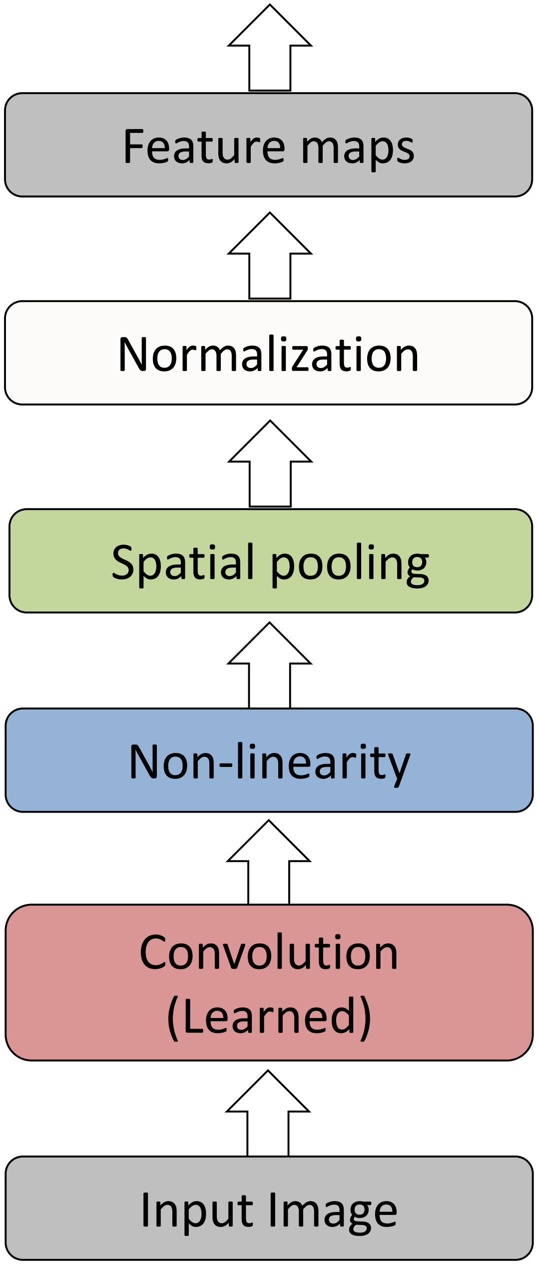


Input

Feature Activation Map

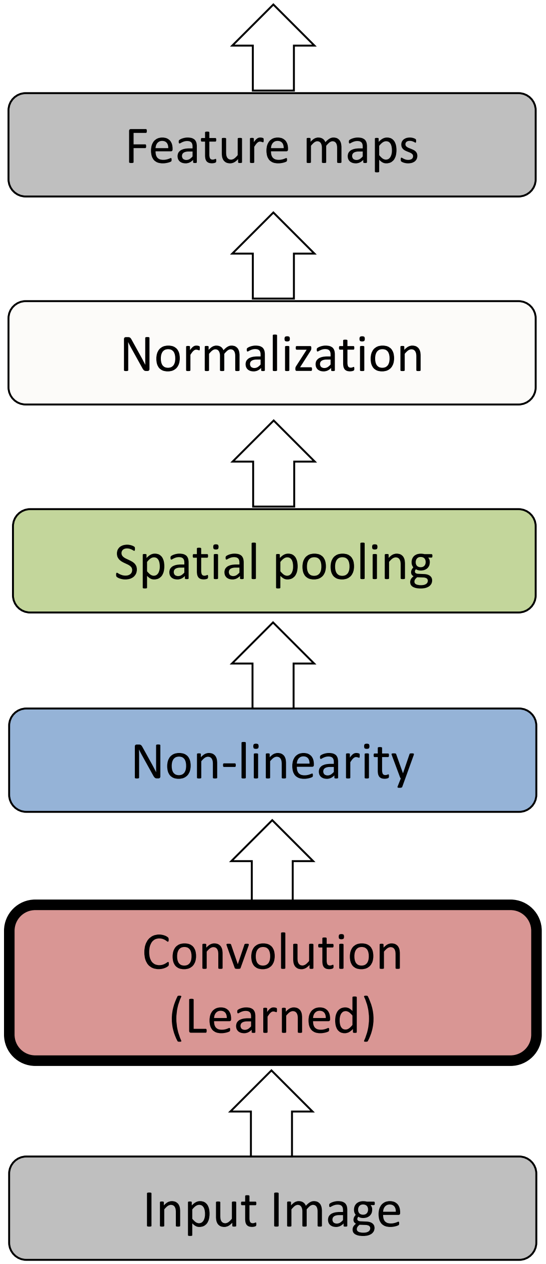
slide credit: S. Lazebnik

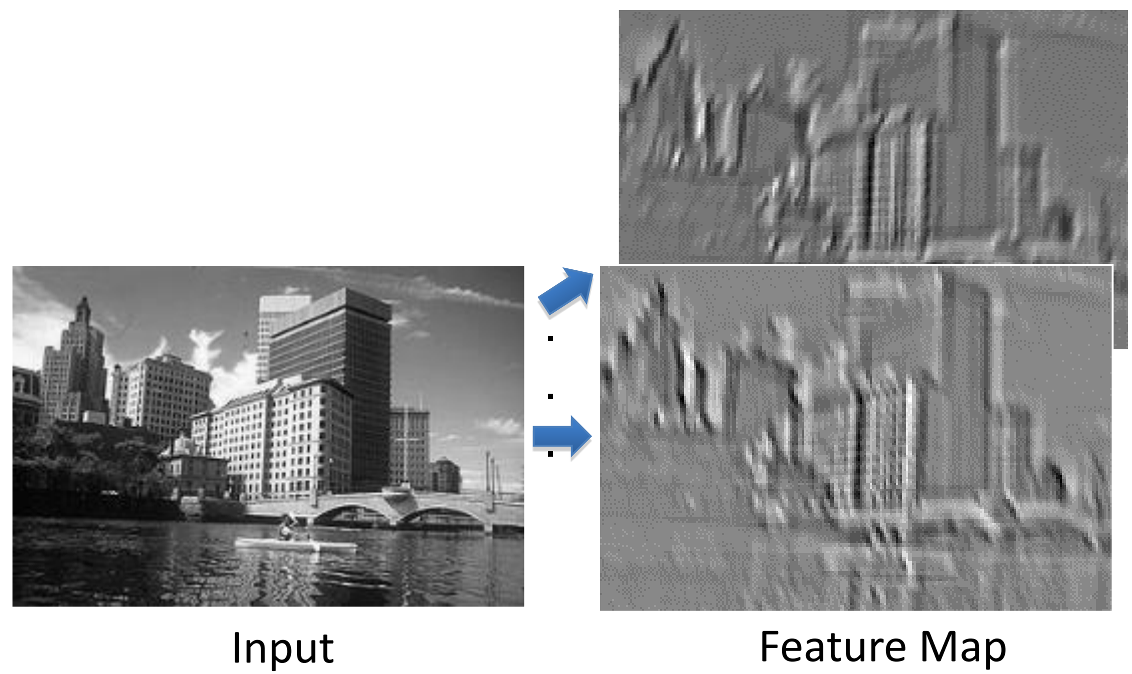
Convolutional Neural Networks



slide credit: S. Lazebnik

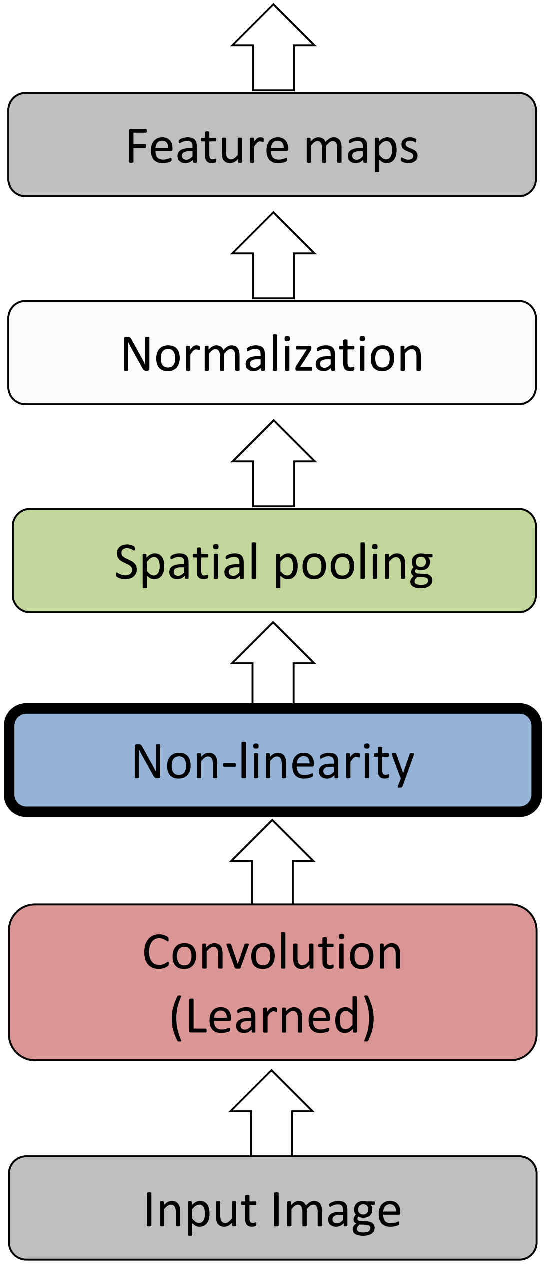
Convolutional Neural Networks



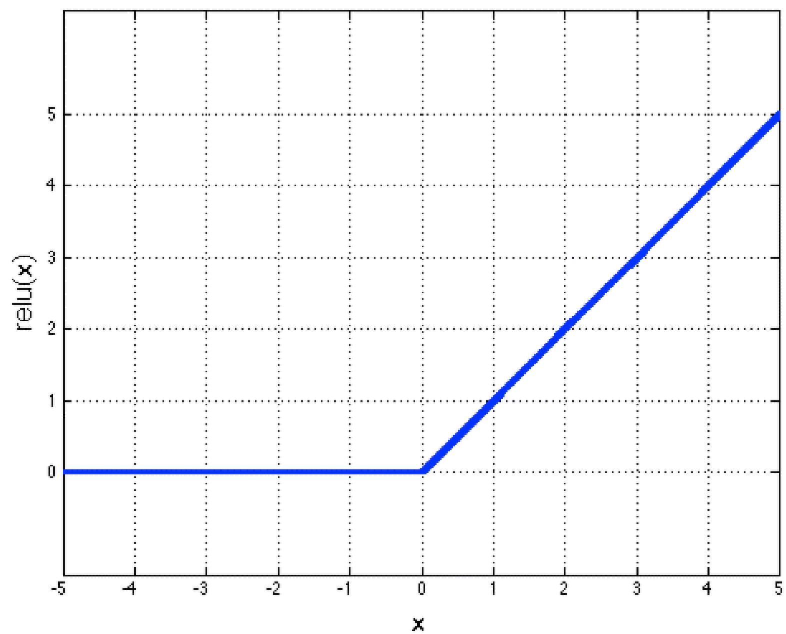


slide credit: S. Lazebnik

Convolutional Neural Networks

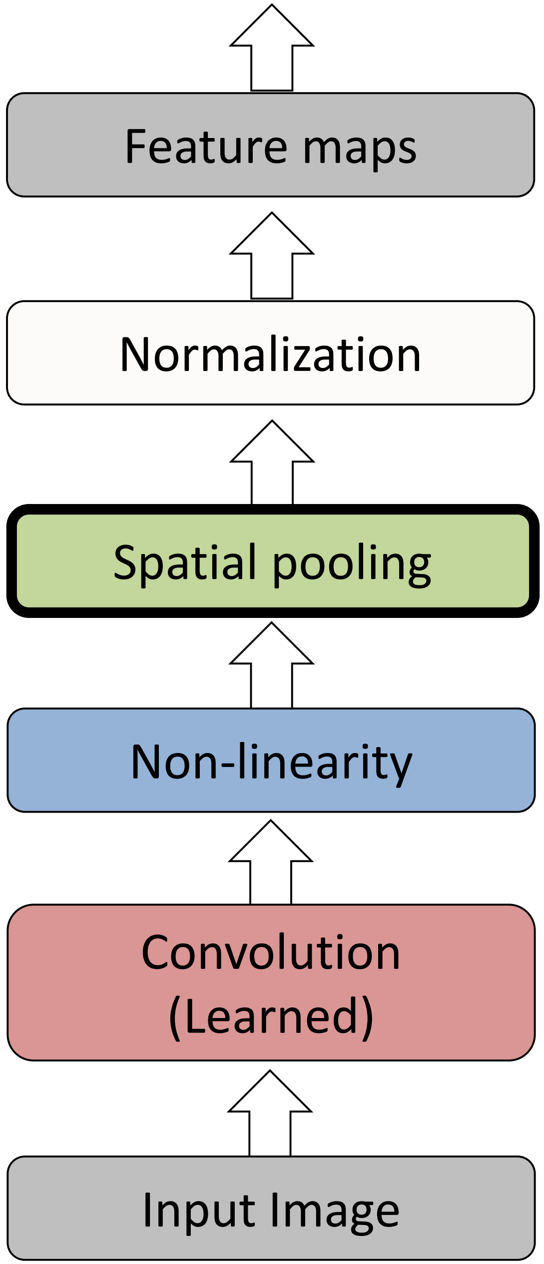


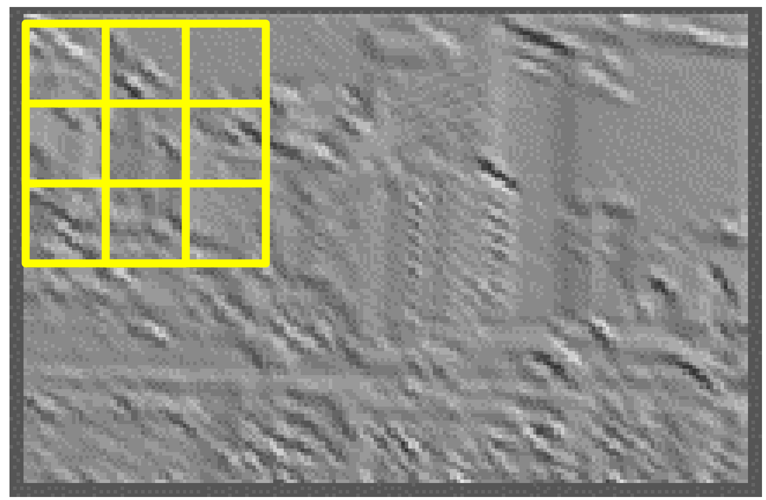
Rectified Linear Unit (ReLU)



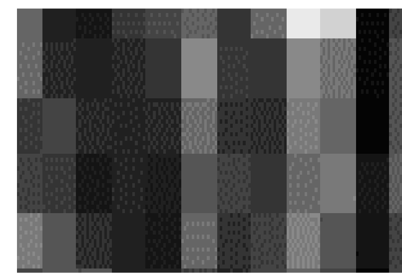
slide credit: S. Lazebnik

Convolutional Neural Networks





Max pooling

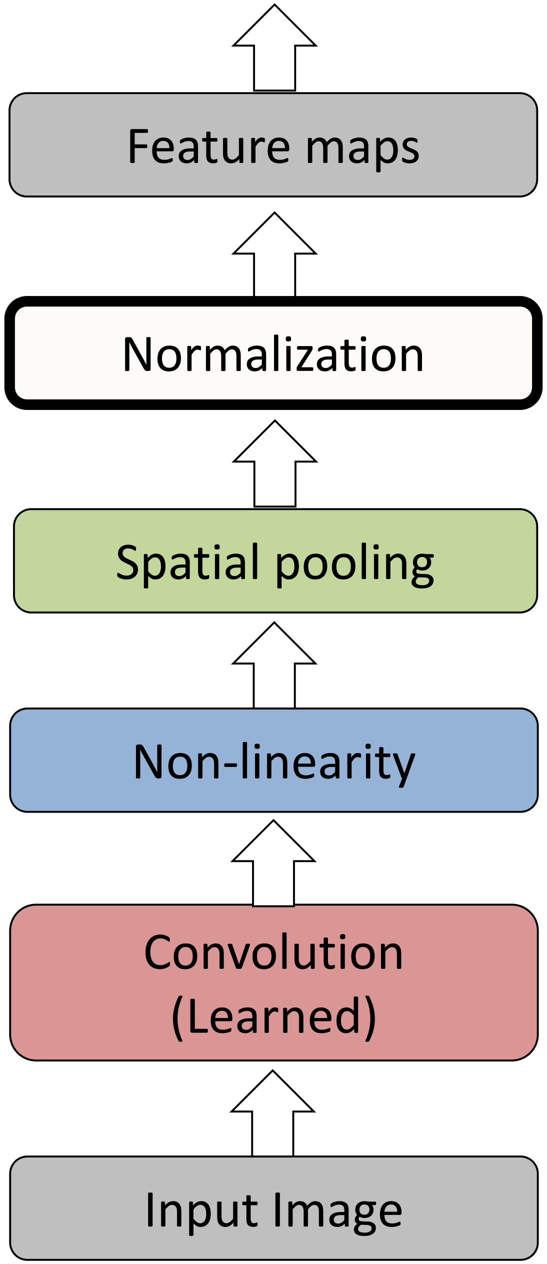


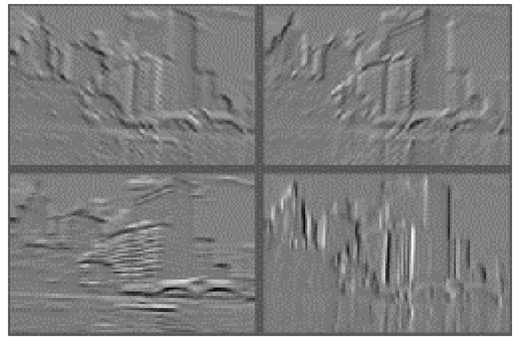
Max-pooling: a non-linear down-sampling

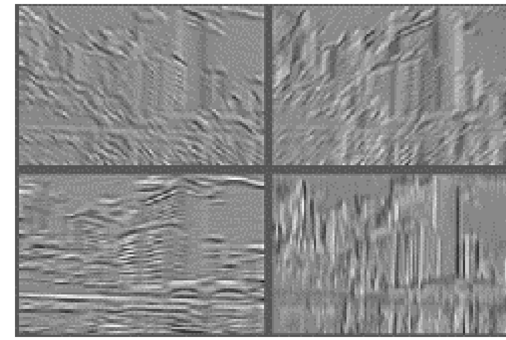
Provide *translation invariance*

slide credit: S. Lazebnik

Convolutional Neural Networks

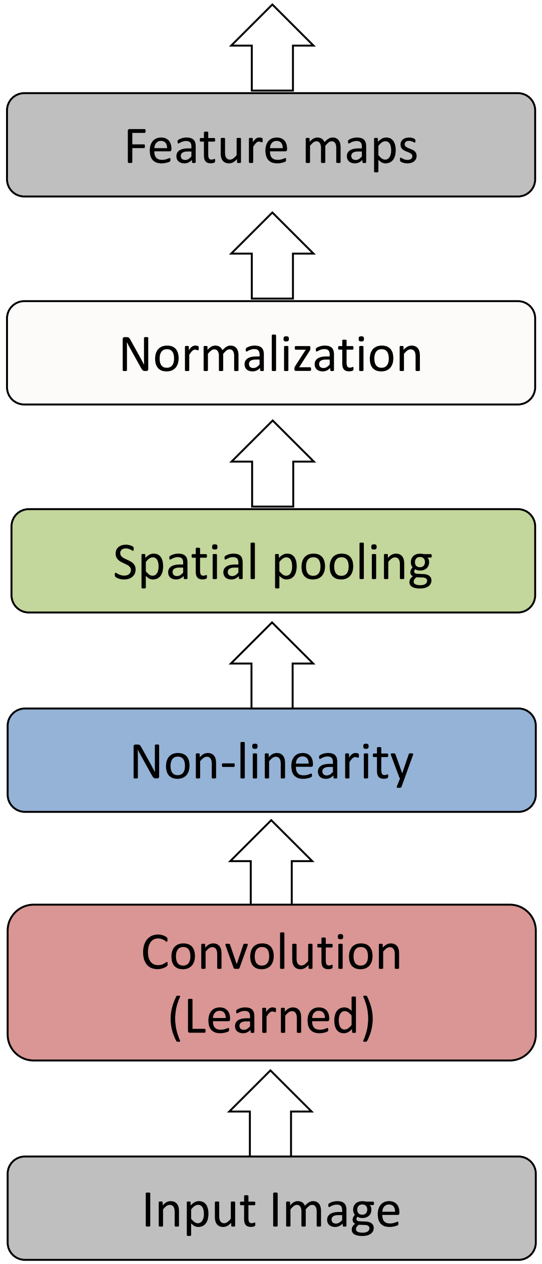



Feature Maps


Feature Maps
After Contrast
Normalization

slide credit: S. Lazebnik

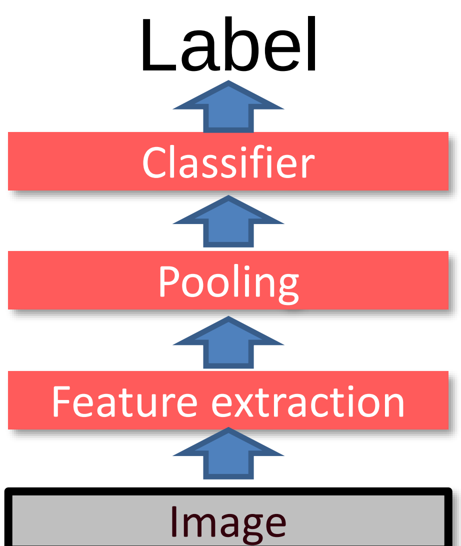
Convolutional Neural Networks

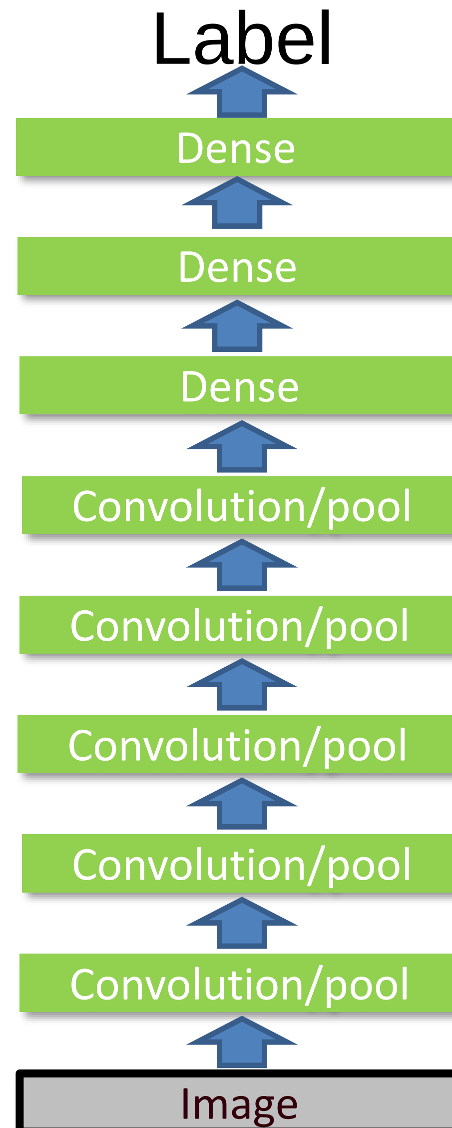


slide credit: S. Lazebnik

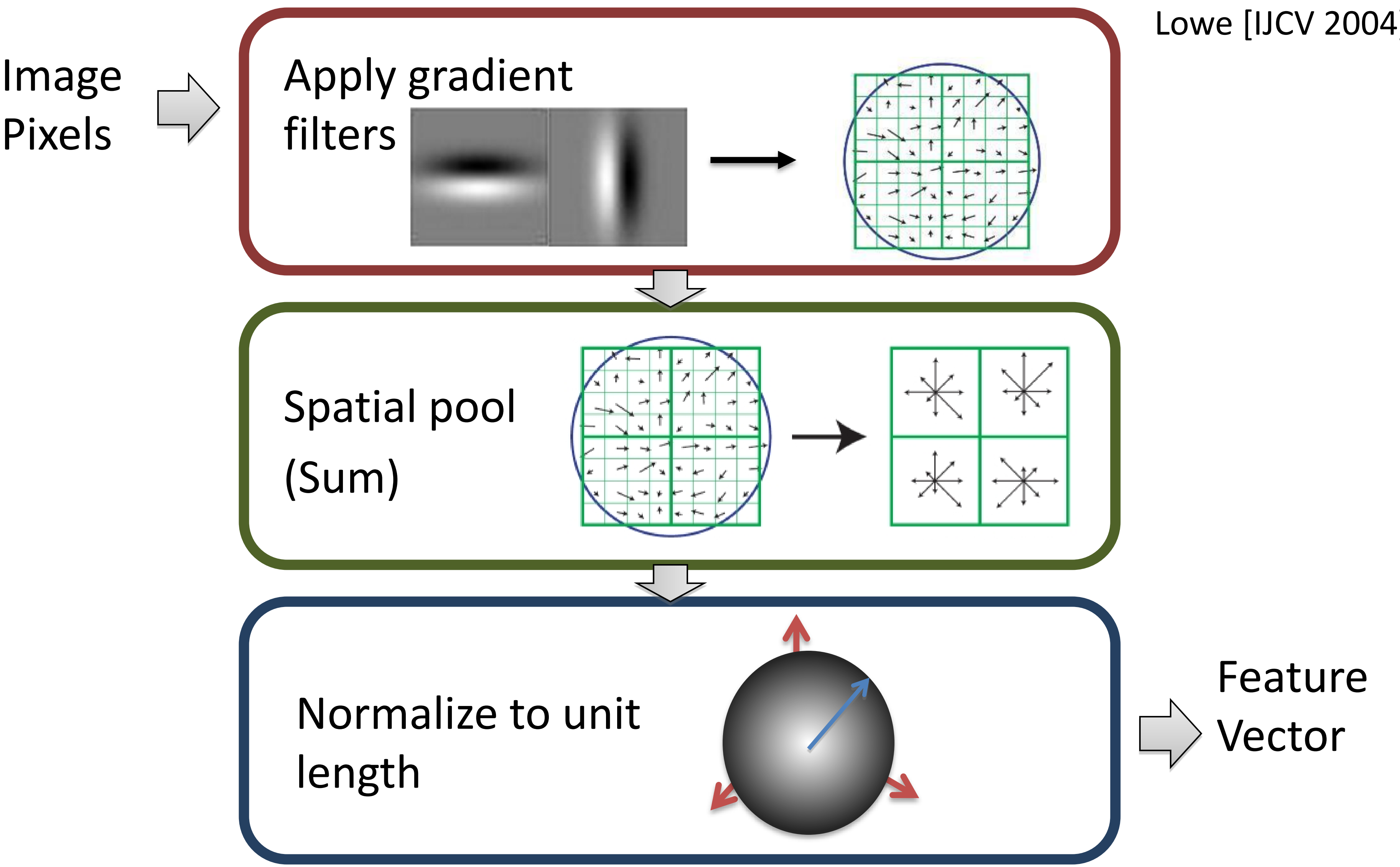
Engineered vs. learned features

Convolutional filters are trained in a supervised manner by back-propagating classification error





SIFT Descriptor

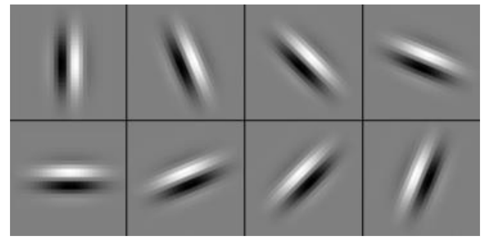


SIFT Descriptor

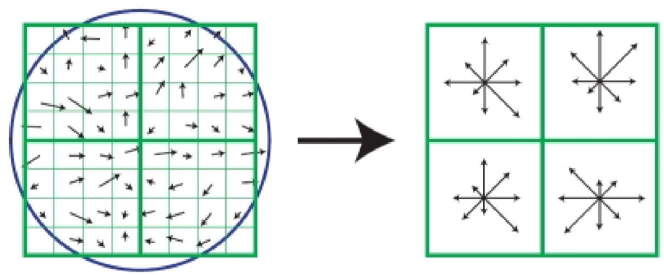
Lowe [IJCV 2004]

Image Pixels

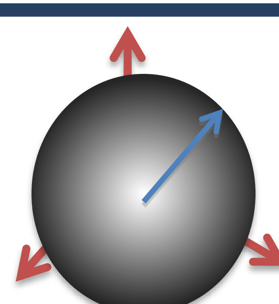
Apply oriented filters



Spatial pool (Sum)



Normalize to unit length



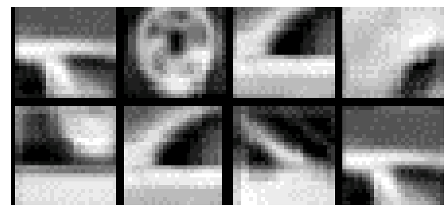
Feature Vector

slide credit: R. Fergus

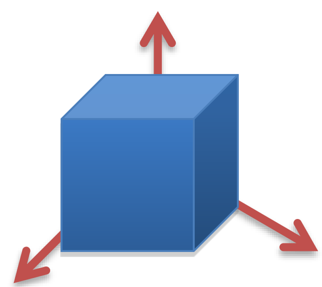
Spatial Pyramid Matching

SIFT Features

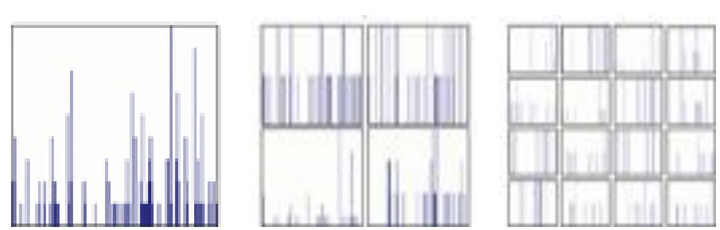
Filter with Visual Words



Max



Multi-scale spatial pool (Sum)

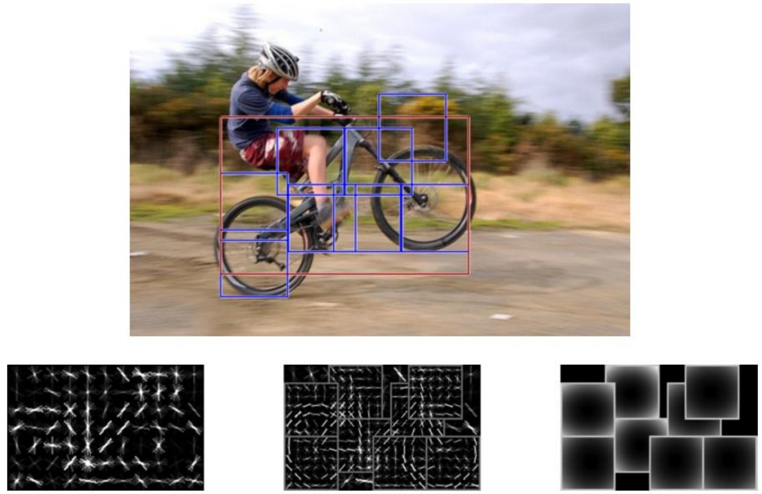


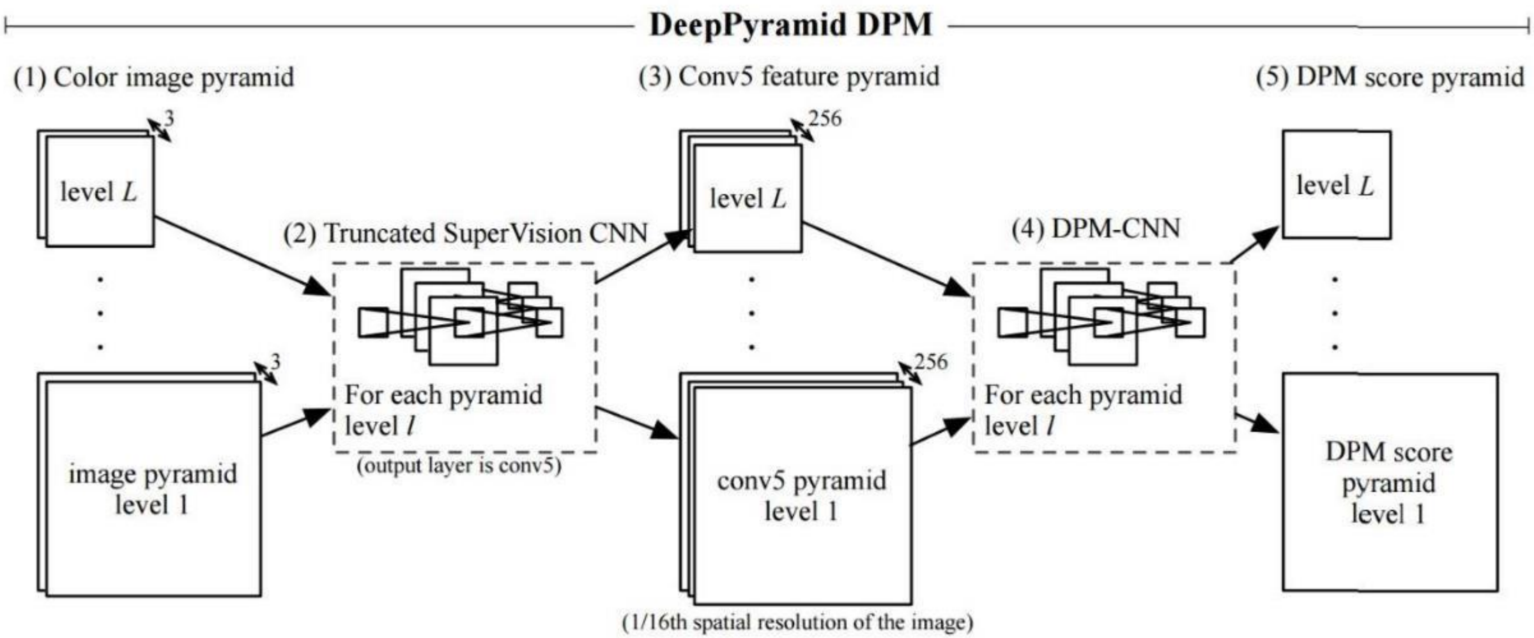
Classifier

Lazebnik, Schmid, Ponce [CVPR 2006]

slide credit: R. Fergus

Deformable Part Model

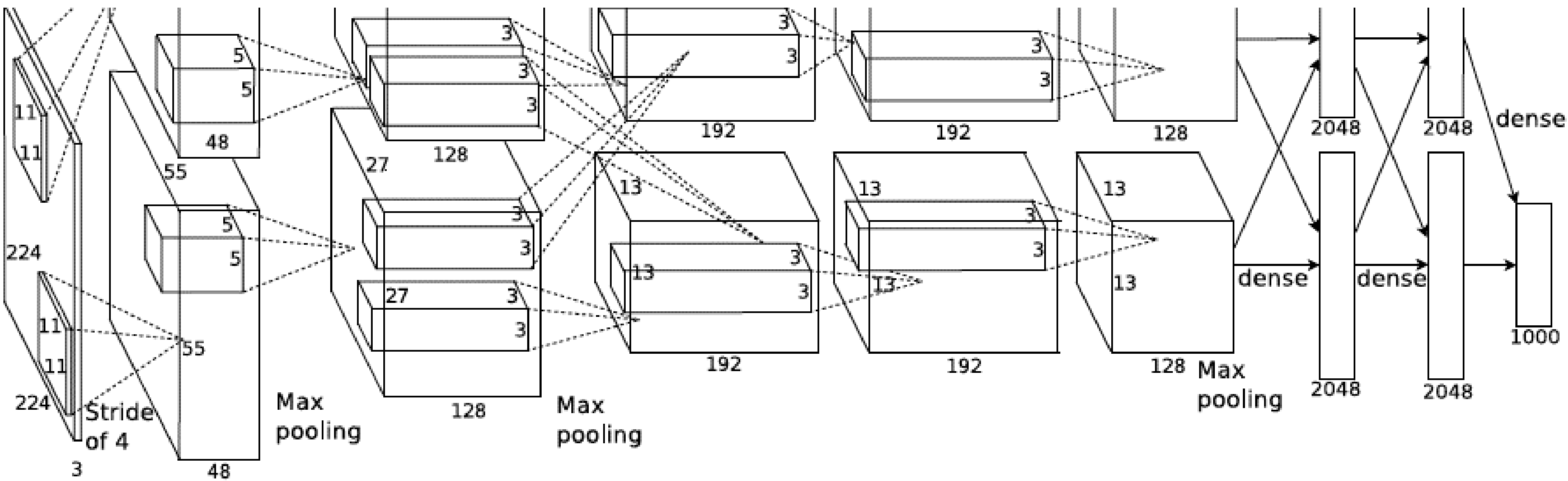




Deformable Part Models are Convolutional Neural Networks [Girshick et al. CVPR 15]

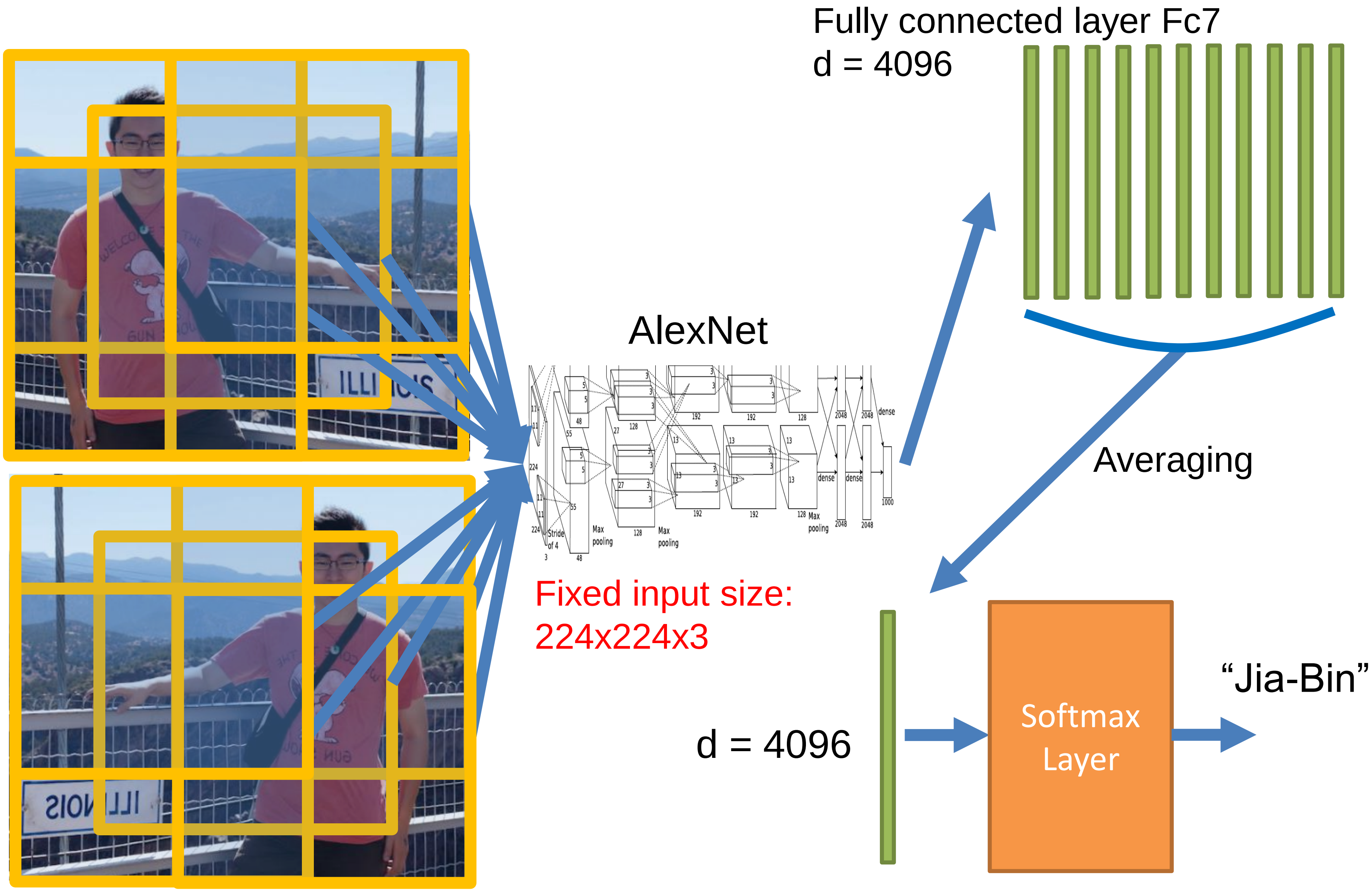
AlexNet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



A. Krizhevsky, I. Sutskever, and G. Hinton,
ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Using CNN for Image Classification



ImageNet Challenge 2012-2014

Best non-convnet in 2012: 26.2%

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

Team	Method	Error (top-5)
DeepImage - Baidu	Data augmentation + multi GPU	5.33%
PReLU-nets - MSRA	Parametric ReLU + smart initialization	4.94%
BN-Inception ensemble - Google	Reducing internal covariate shift	4.82%

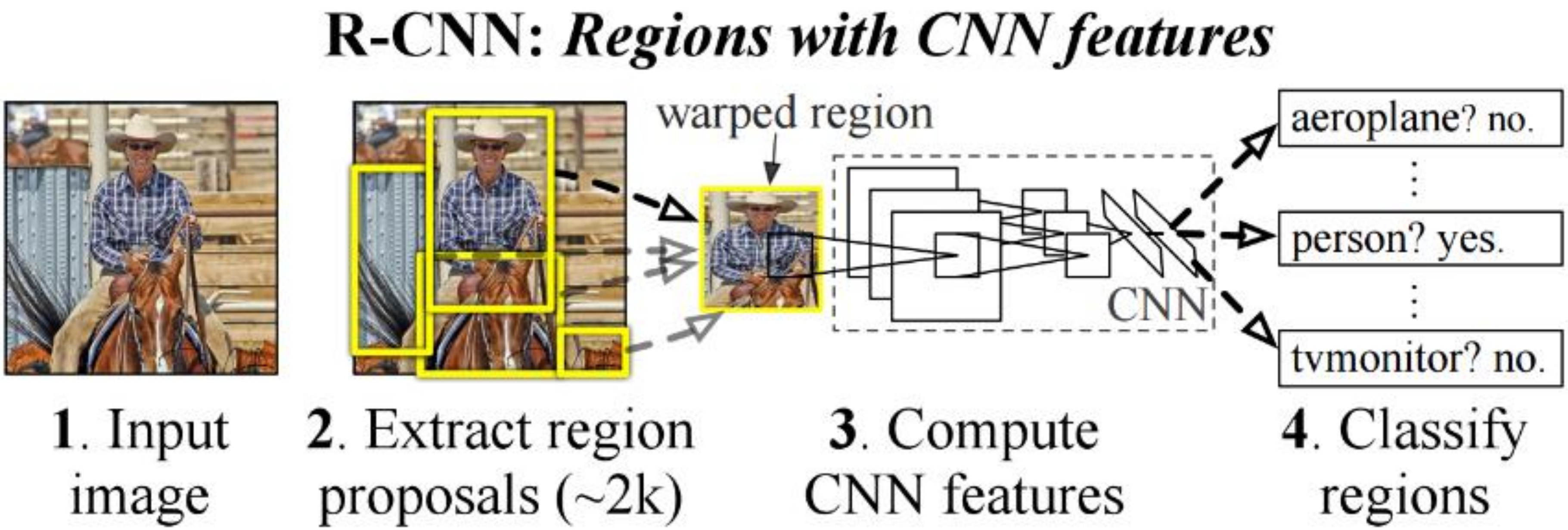
Beyond classification

- Detection
- Segmentation
- Regression
- Pose estimation
- Matching patches
- Synthesis
- Style transfer

and many more…

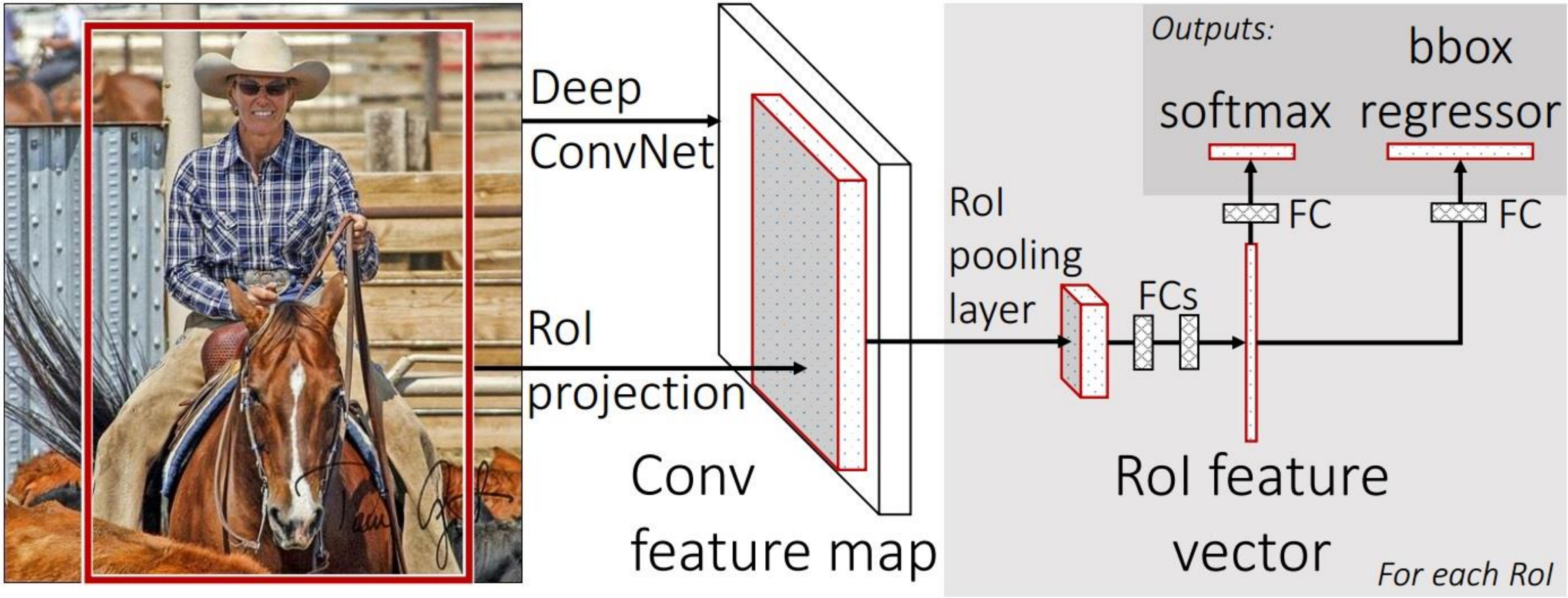
R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL



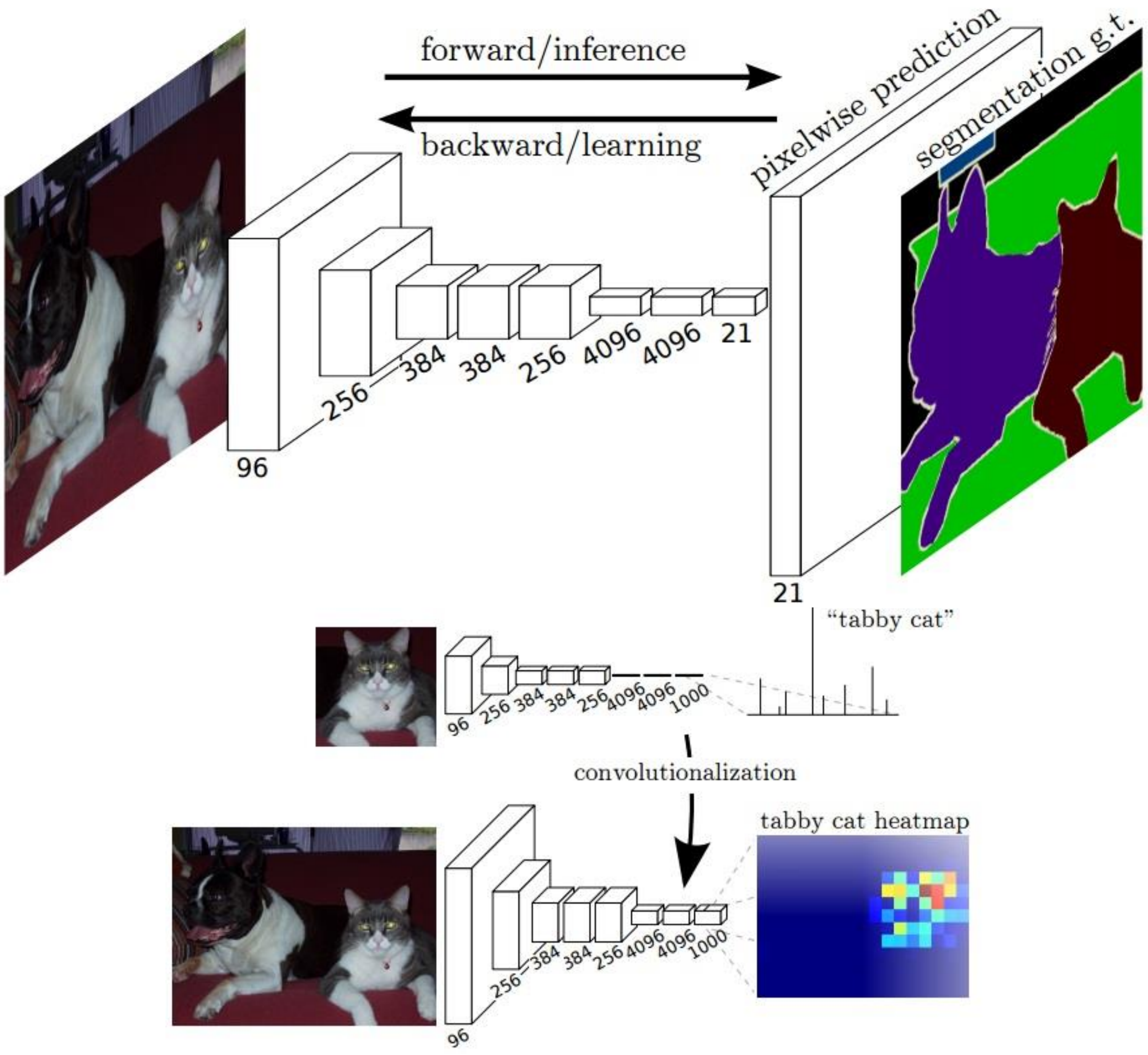
RCNN [Girshick et al. CVPR 2014]

Fast R-CNN



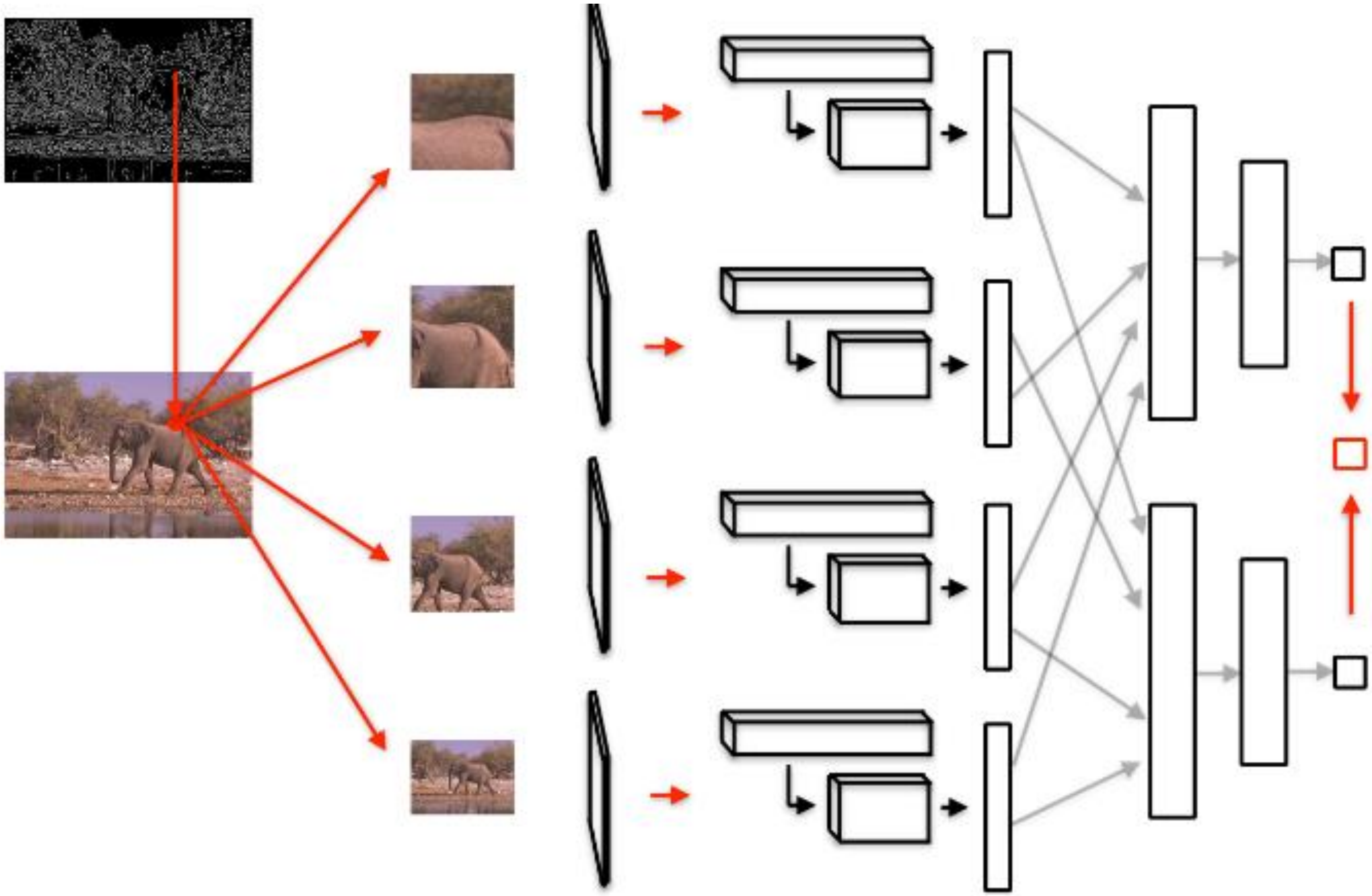
Fast RCNN [Girshick, R 2015]
https://github.com/rbgirshick/fast-rcnn

Labeling Pixels: Semantic Labels



Fully Convolutional Networks for Semantic Segmentation [Long et al. CVPR 2015]

Labeling Pixels: Edge Detection

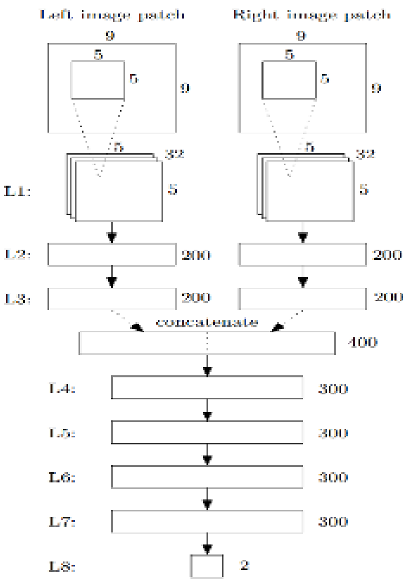


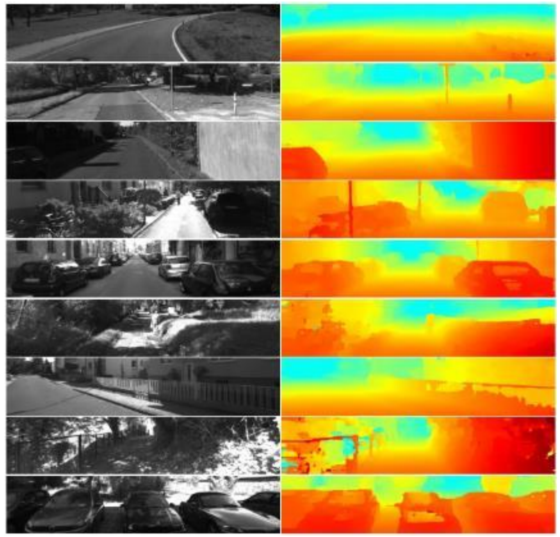
DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection
[Bertasius et al. CVPR 2015]

CNN for Regression

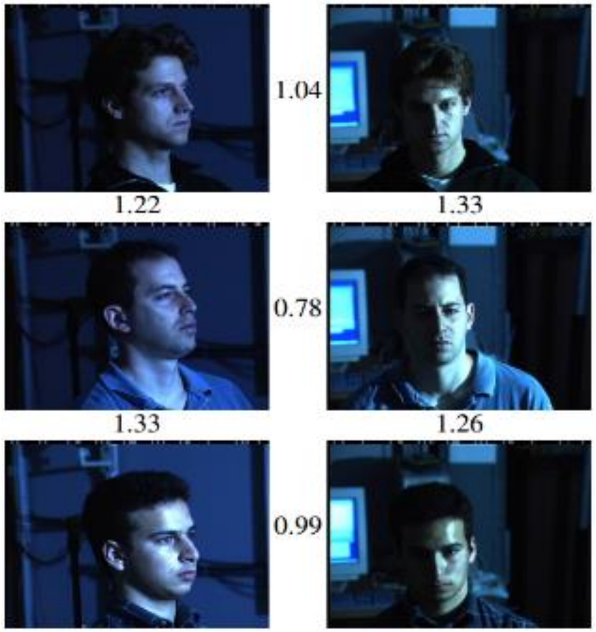
DeepPose [Toshev and Szegedy CVPR 2014]

CNN as a Similarity Measure for Matching

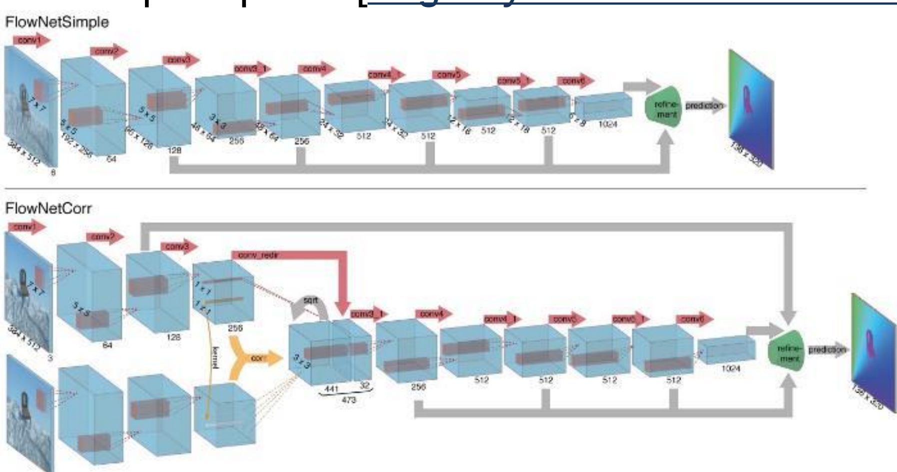




Stereo matching [Zbontar and LeCun CVPR 2015]
Compare patch [Zagoruyko and Komodakis 2015]



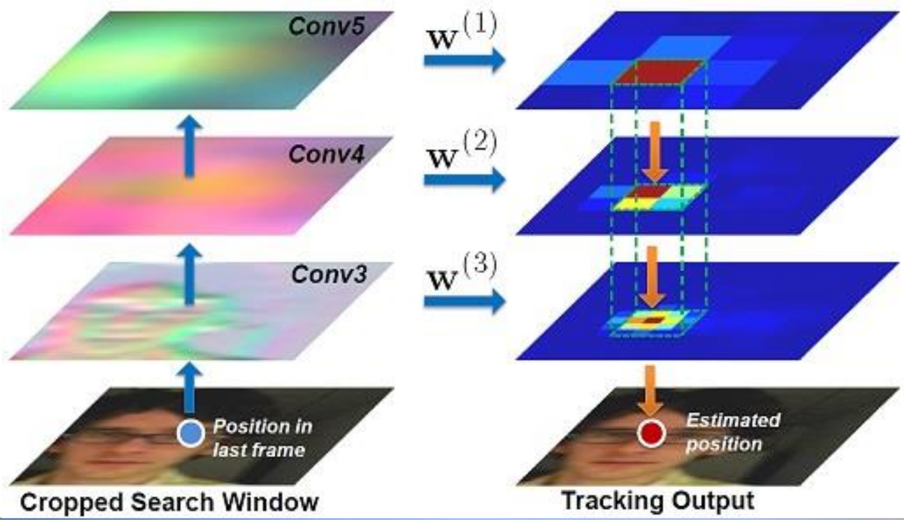
FaceNet [Schroff et al. 2015]



FlowNet [Fischer et al 2015]

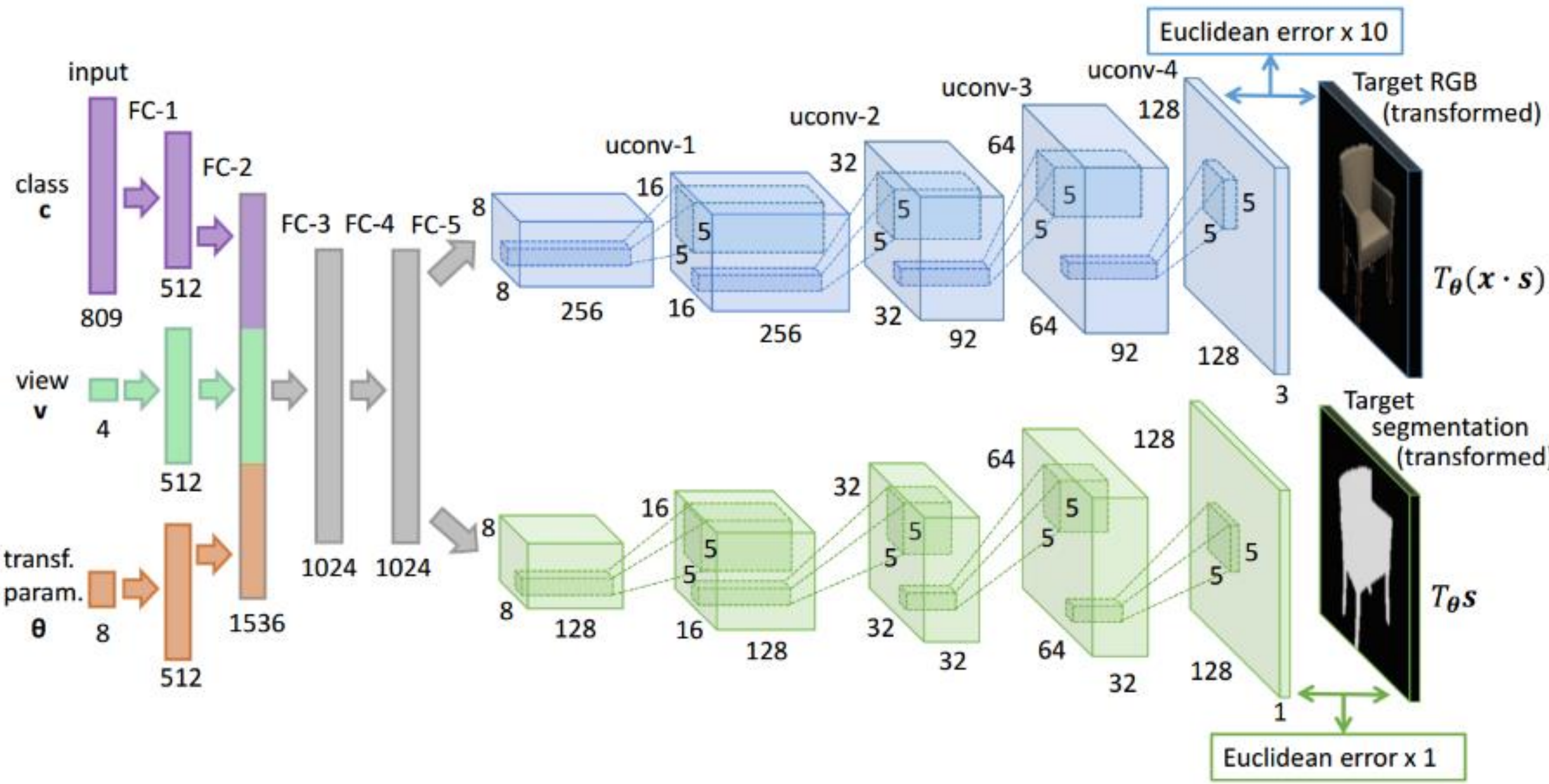
Match ground and aerial images
[Lin et al. CVPR 2015]

CNN for Online Visual Tracking



Hierarchical Convolutional Features for Visual Tracking [Ma et al. ICCV 2015]

CNN for Image Generation



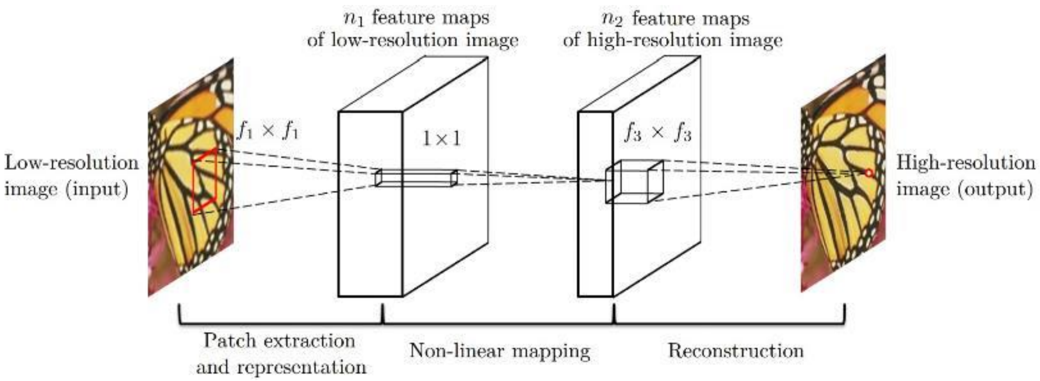
Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

Chair Morphing

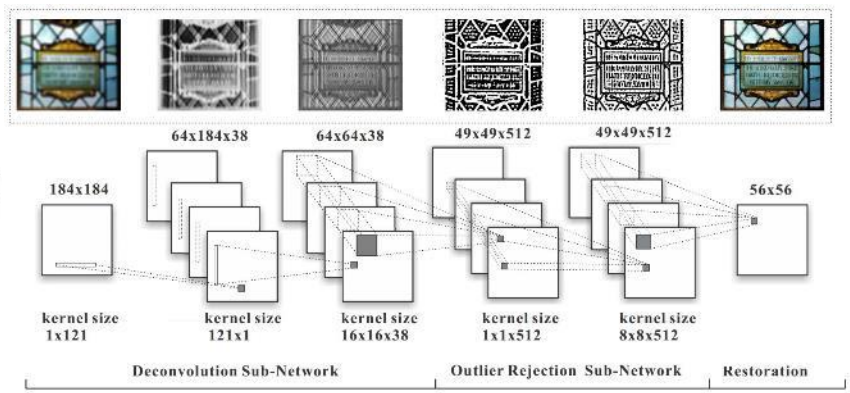
1

Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

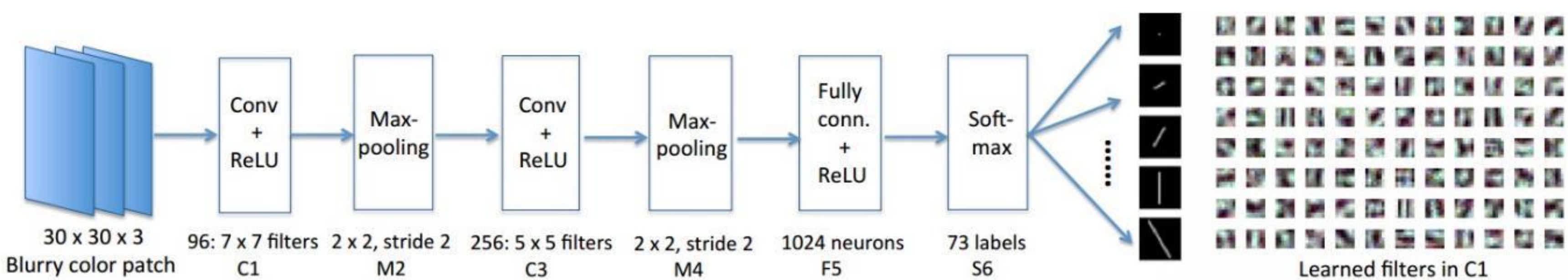
CNN for Image Restoration/Enhancement



Super-resolution
[Dong et al. ECCV 2014]



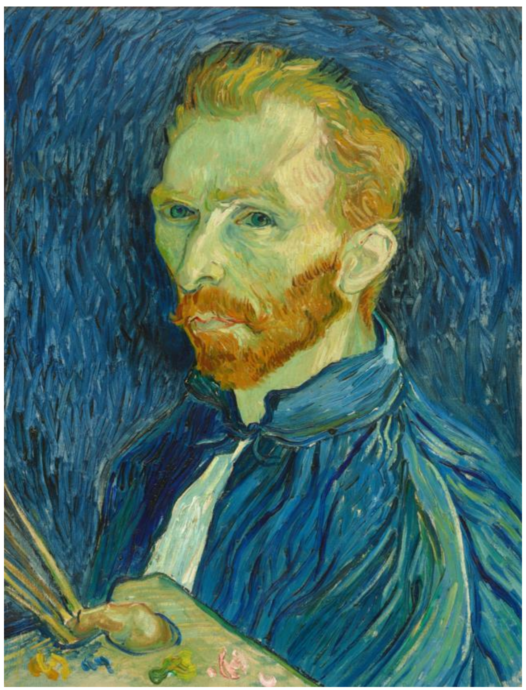
Non-blind deconvolution
[Xu et al. NIPS 2014]



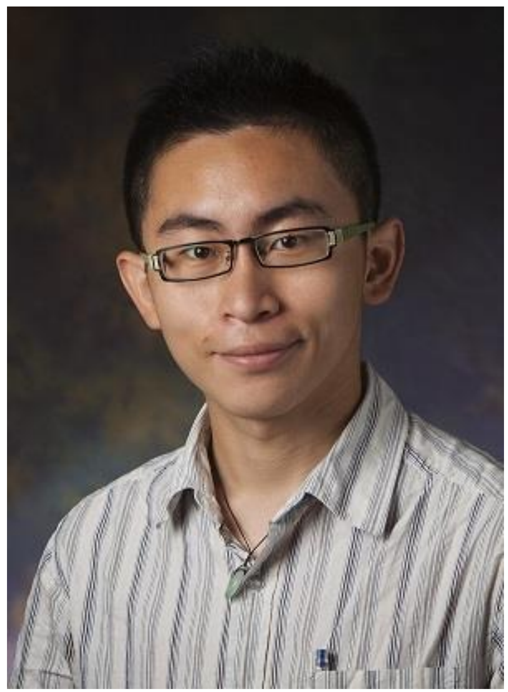
Non-uniform blur estimation
[Sun et al. CVPR 2015]

Style Transfer

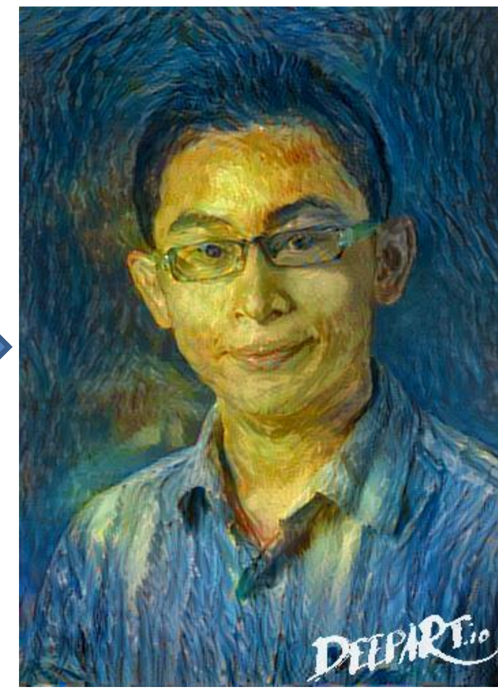
- Find an output image with
  - similar activations of early layers (low-level) of source image
  - similar activations of later layers (high-level) of target image



Source image



Target image



Output (deepart)

A Neural Algorithm of Artistic Style [Gatys et al. 2015]

Image Captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

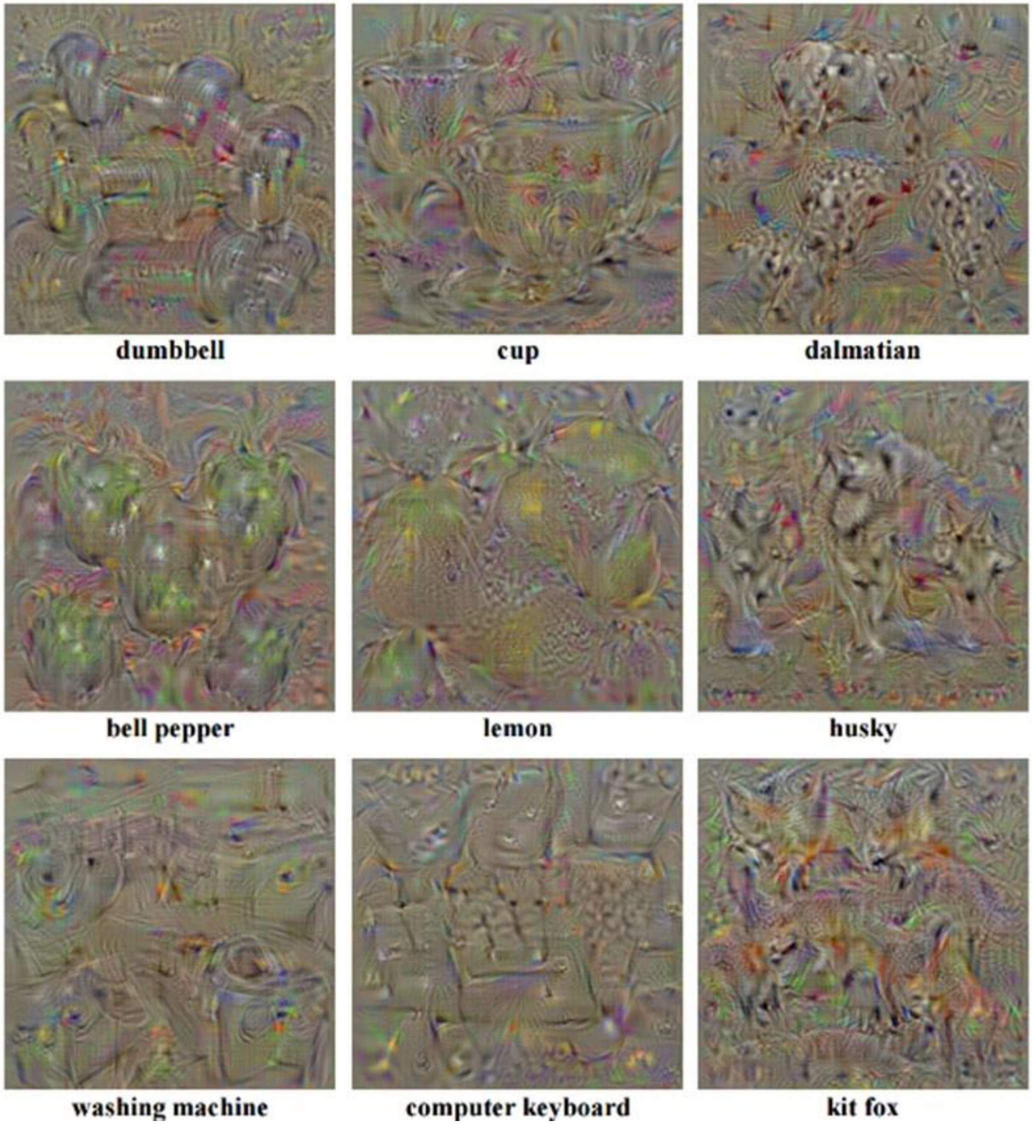
"man in blue wetsuit is surfing on wave."

http://cs.stanford.edu/people/karpathy/deepimagesent/

Understanding and Visualizing CNN

- Find images that maximize some class scores
- Individual neuron activation
- Visualize input pattern using deconvnet
- Invert CNN features
- Breaking CNNs

Find images that maximize some class scores



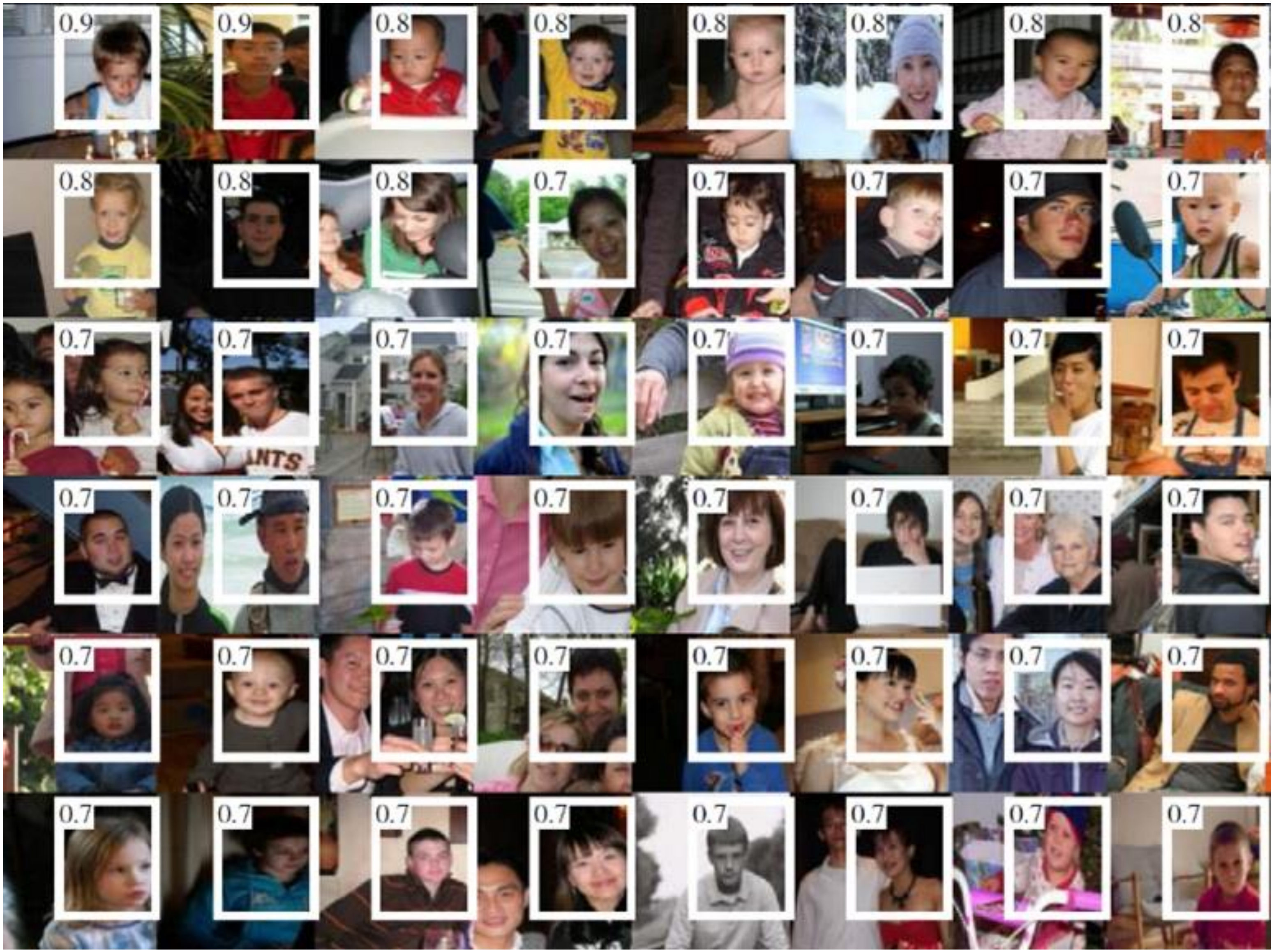
person: HOG template

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
[Simonyan et al. ICLR Workshop 2014]

Individual Neuron Activation

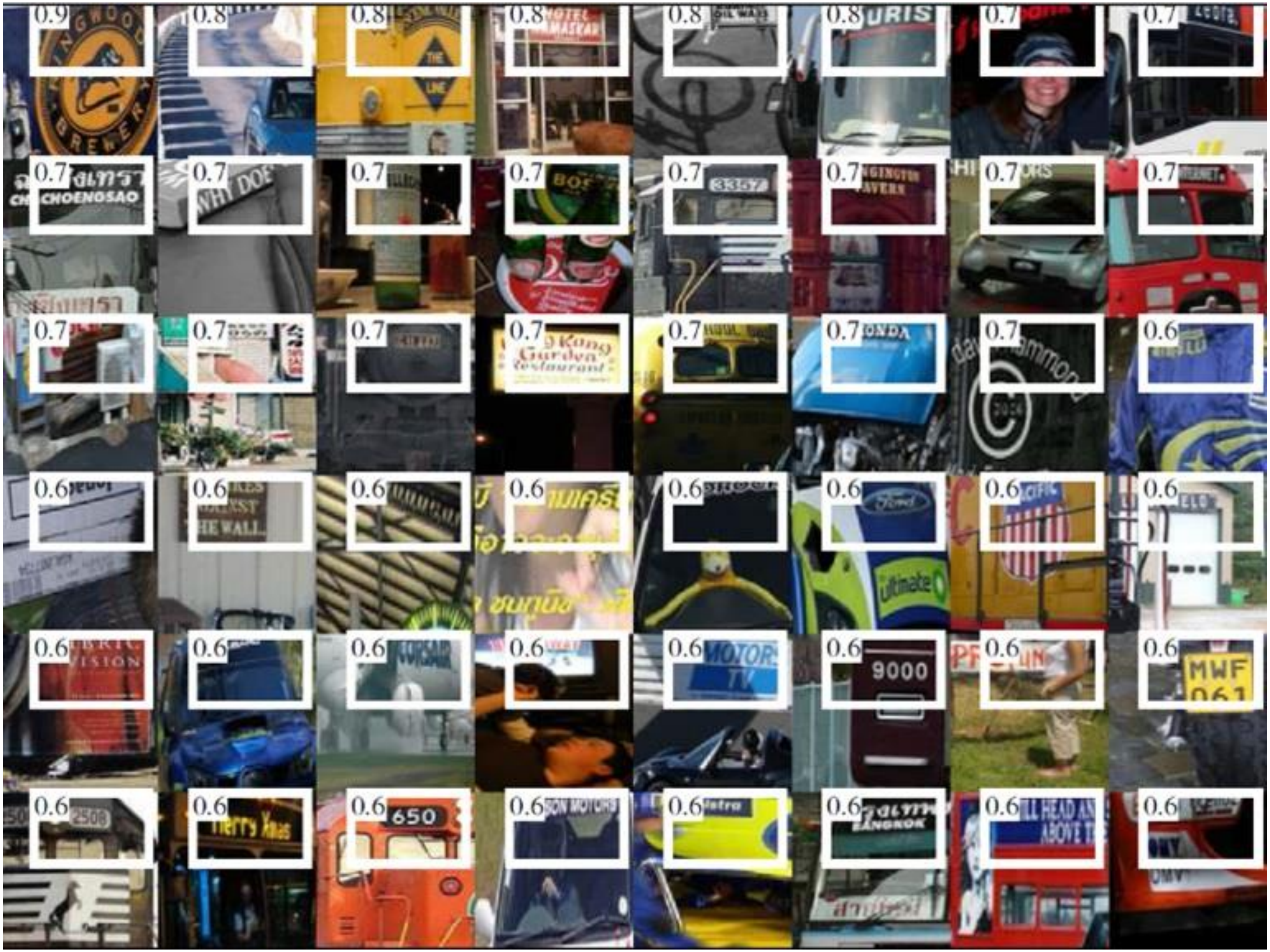
RCNN [Girshick et al. CVPR 2014]

Individual Neuron Activation



RCNN [Girshick et al. CVPR 2014]

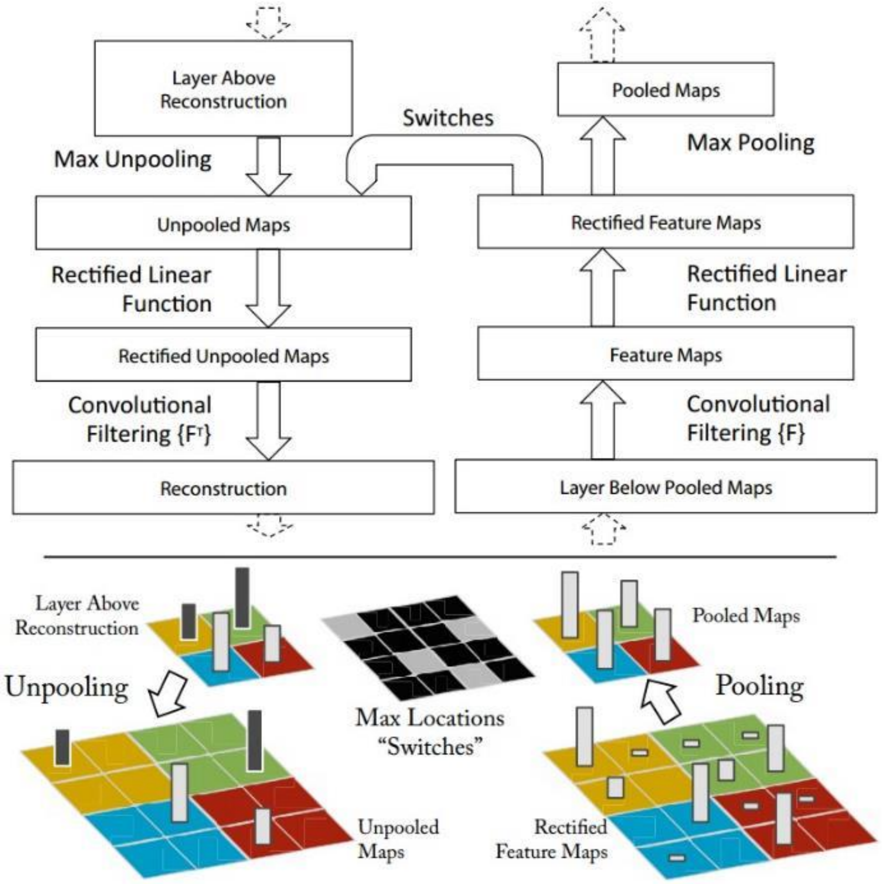
Individual Neuron Activation



RCNN [Girshick et al. CVPR 2014]

Map activation back to the input pixel space

- What input pattern originally caused a given activation in the feature maps?

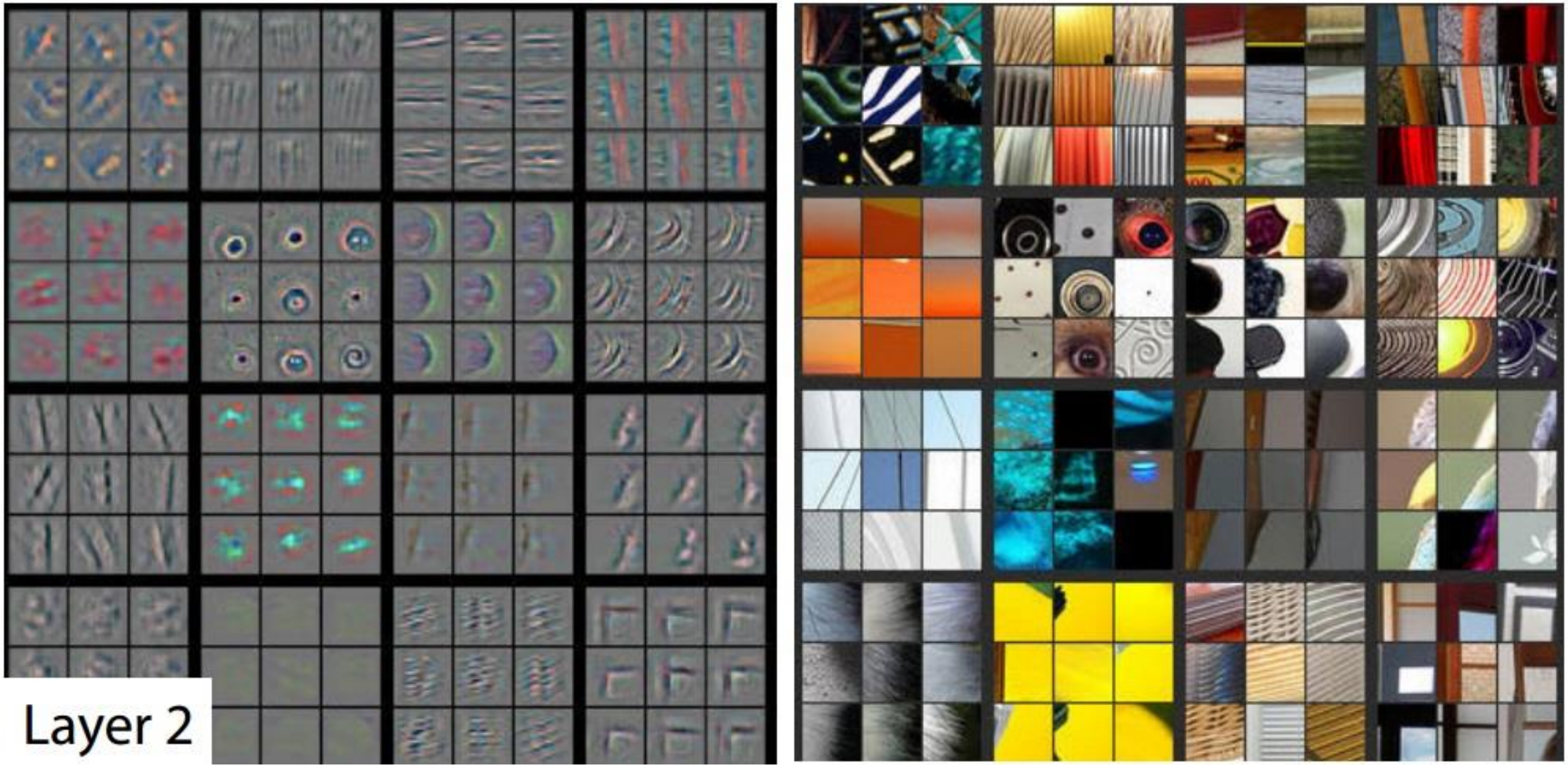


Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 1

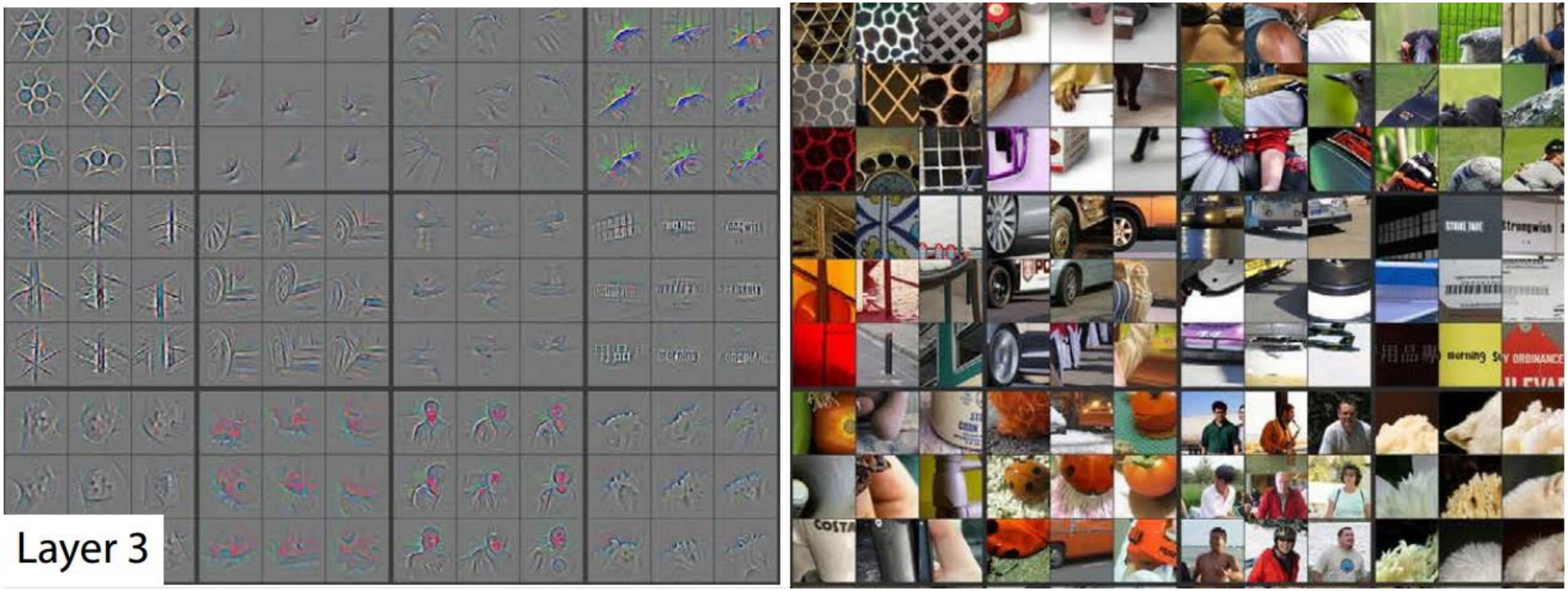
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 2



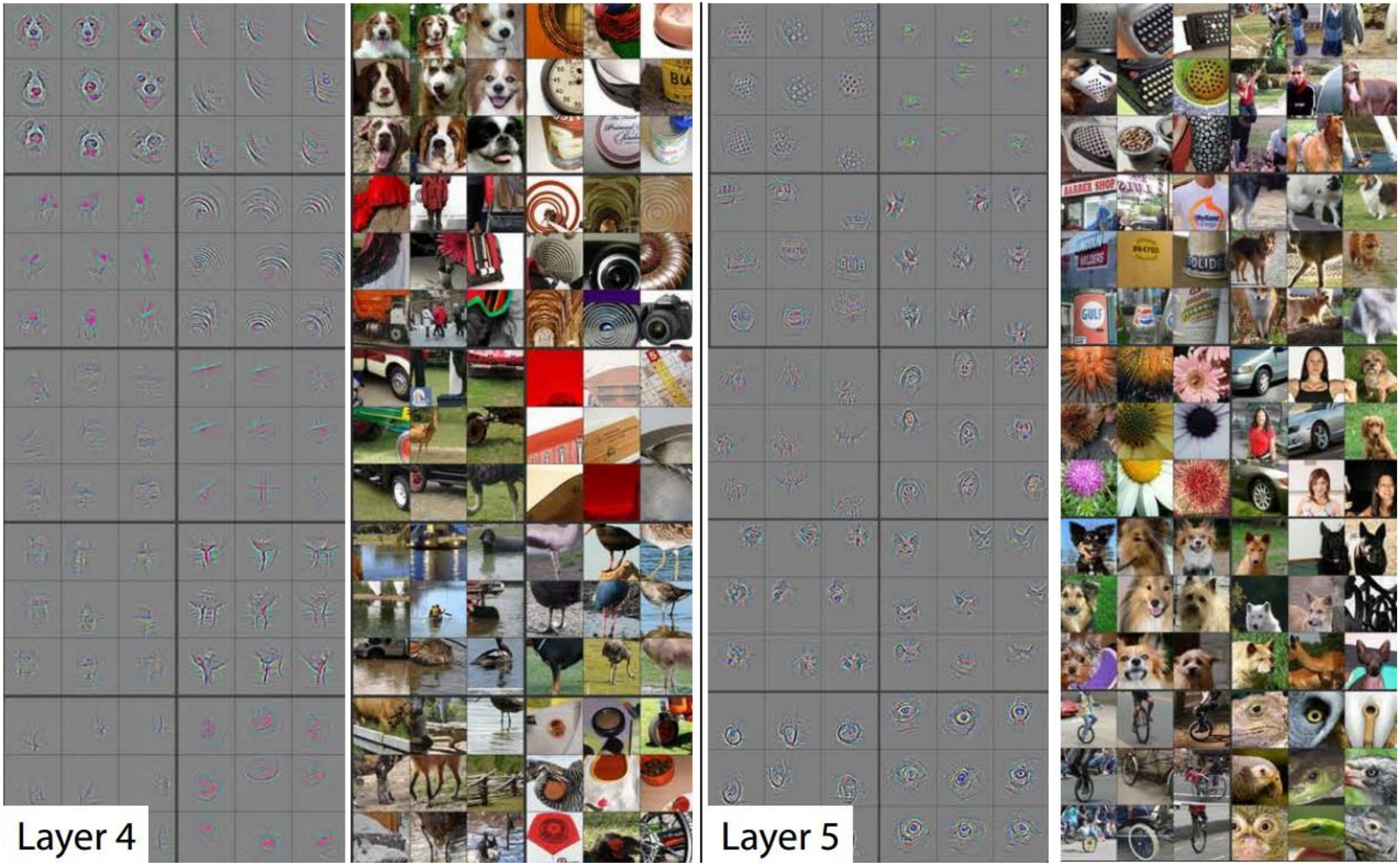
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 3



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

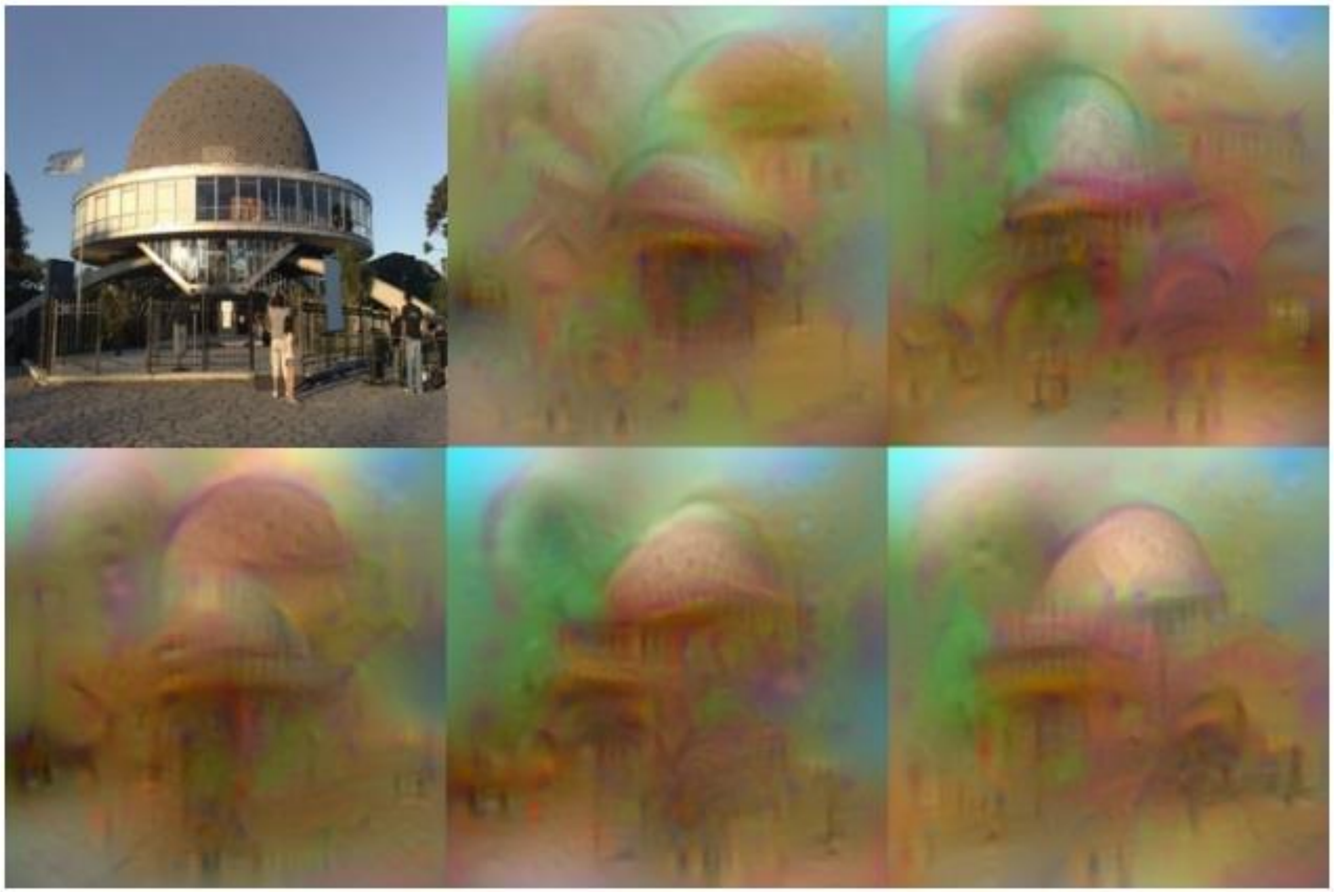
Layer 4 and 5



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Invert CNN features

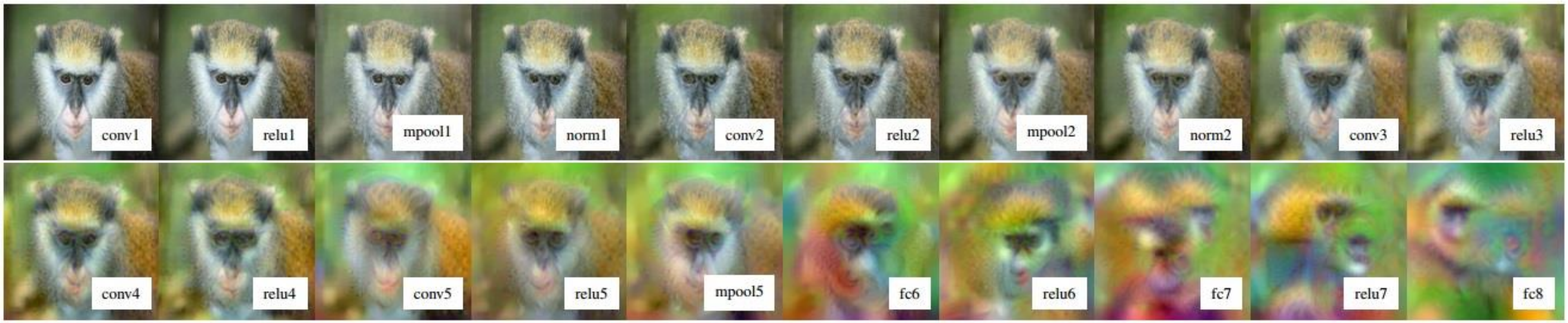
- Reconstruct an image from CNN features



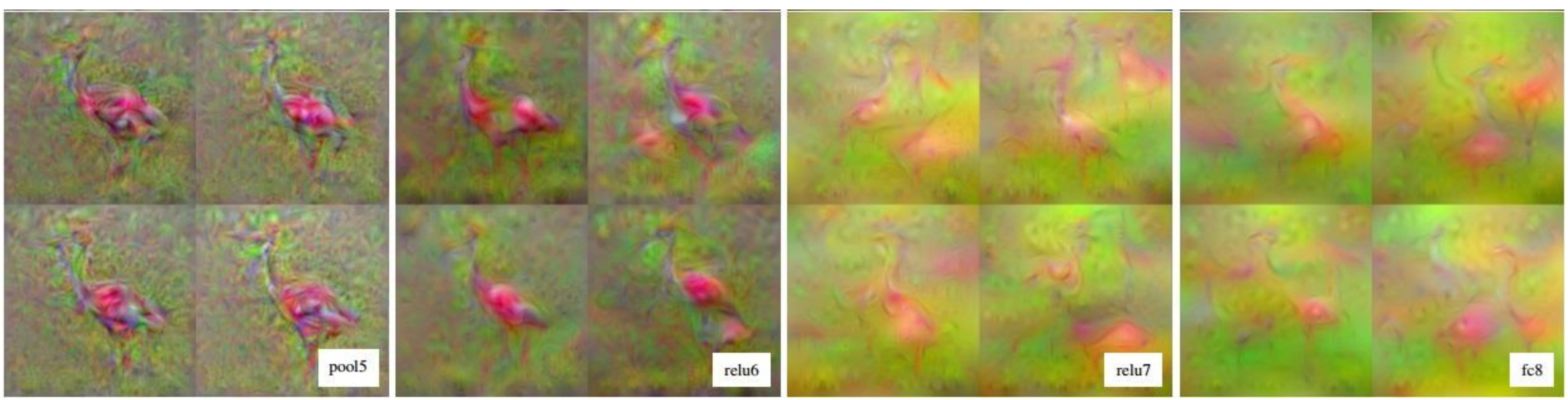
Understanding deep image representations by inverting them
[Mahendran and Vedaldi CVPR 2015]

CNN Reconstruction

Reconstruction from different layers

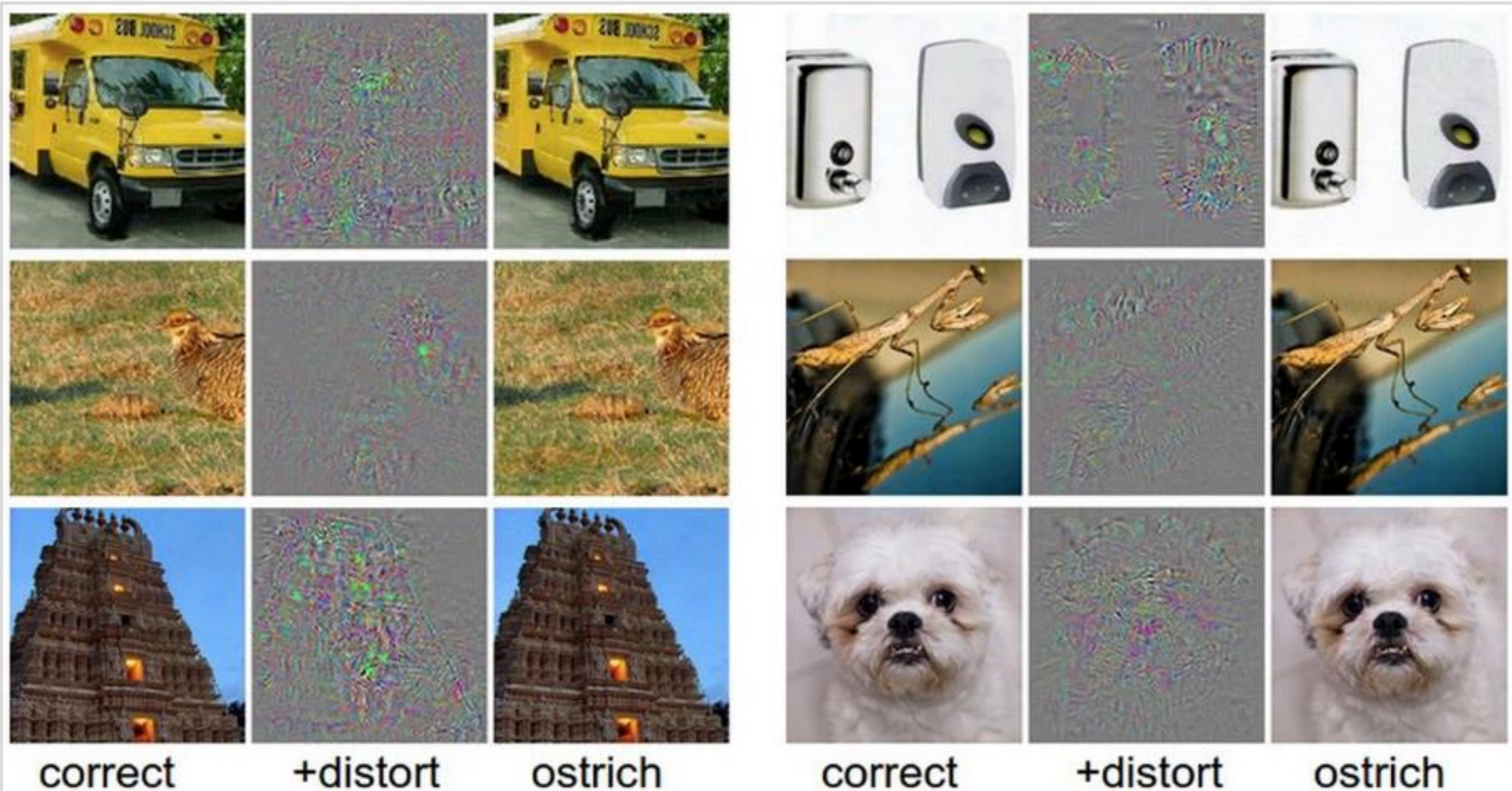


Multiple reconstructions



Understanding deep image representations by inverting them
[Mahendran and Vedaldi CVPR 2015]

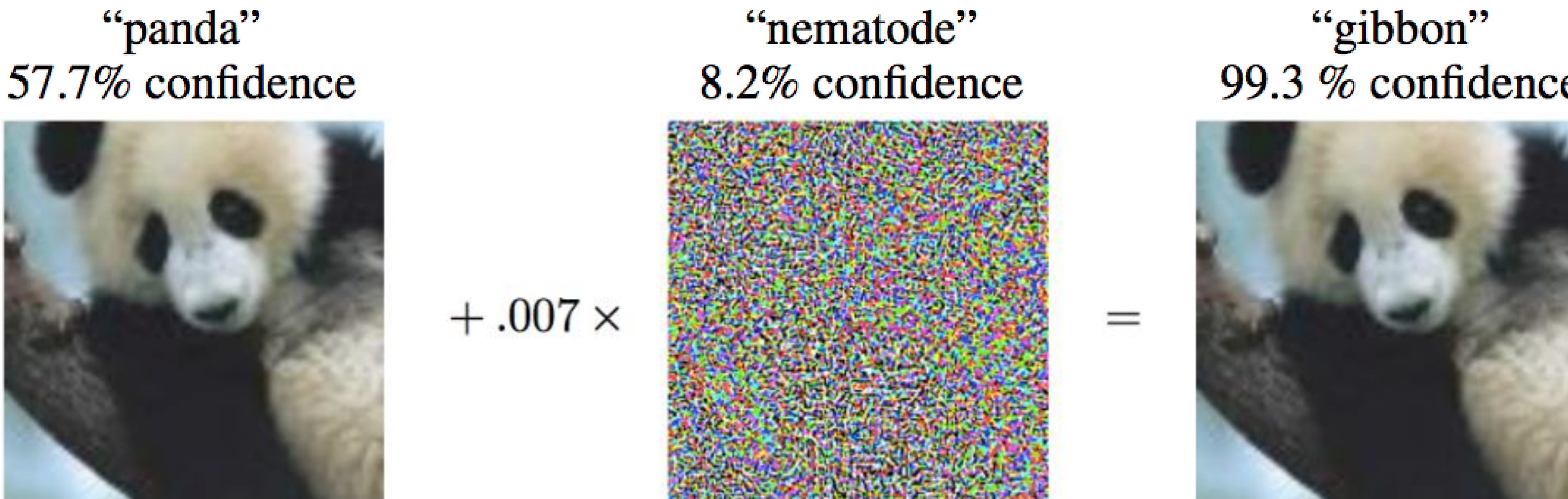
Breaking CNNs



Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

Intriguing properties of neural networks [Szegedy ICLR 2014]

What is going on?



x

$$\frac{\partial E}{\partial \mathbf{x}}$$

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \frac{\partial E}{\partial \mathbf{x}}$$

Explaining and Harnessing Adversarial Examples [[Goodfellow ICLR 2015]]

http://karpathy.github.io/2015/03/30/breaking-convnets/

What is going on?

- Recall gradient descent training: modify the weights to reduce classifier error

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$

- Adversarial examples: modify the *image* to *increase* classifier error

$$\mathbf{x} \neg \ \mathbf{x} + a \frac{\P E}{\P \mathbf{x}}$$

Explaining and Harnessing Adversarial Examples [Goodfellow ICLR 2015]

http://karpathy.github.io/2015/03/30/breaking-convnets/

Fooling a linear classifier

- Perceptron weight update: add a small multiple of the example to the weight vector:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \mathbf{x}$$

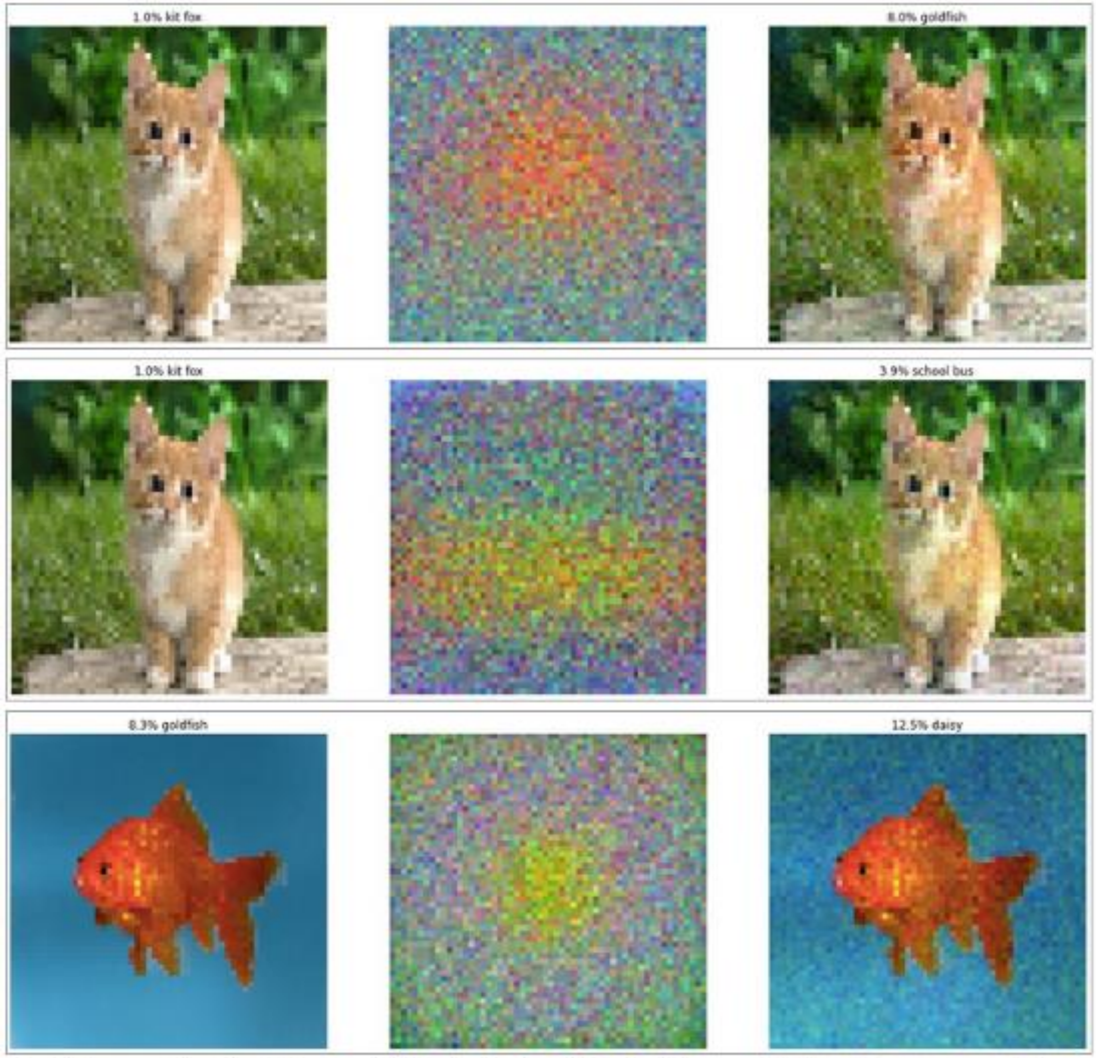
- To fool a linear classifier, add a small multiple of the weight vector to the training example:

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{w}$$

Explaining and Harnessing Adversarial Examples [Goodfellow ICLR 2015]

http://karpathy.github.io/2015/03/30/breaking-convnets/

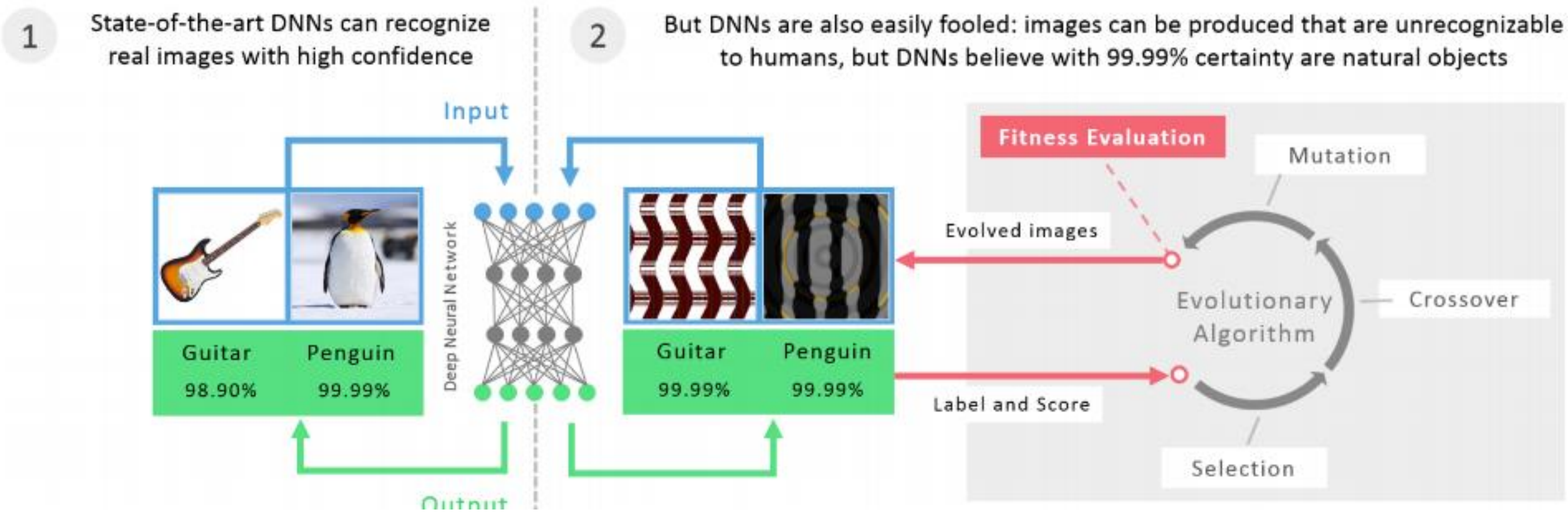
Fooling a linear classifier



Fooled linear classifier: The starting image (left) is classified as a kit fox. That's incorrect, but then what can you expect from a linear classifier? However, if we add a small amount "goldfish" weights to the image (top row, middle), suddenly the classifier is convinced that it's looking at one with high confidence. We can distort it with the school bus template instead if we wanted to.

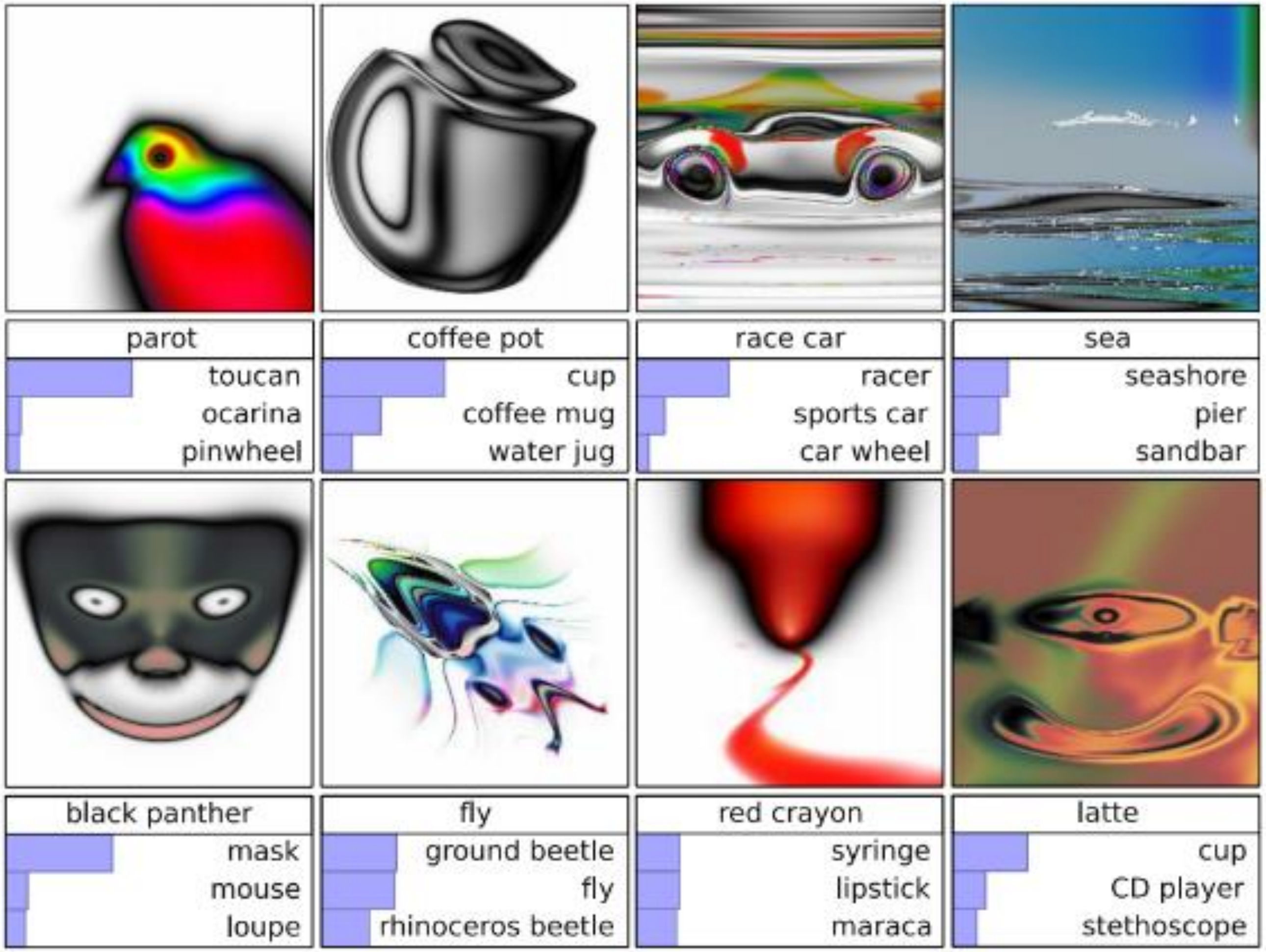
http://karpathy.github.io/2015/03/30/breaking-convnets/

Breaking CNNs



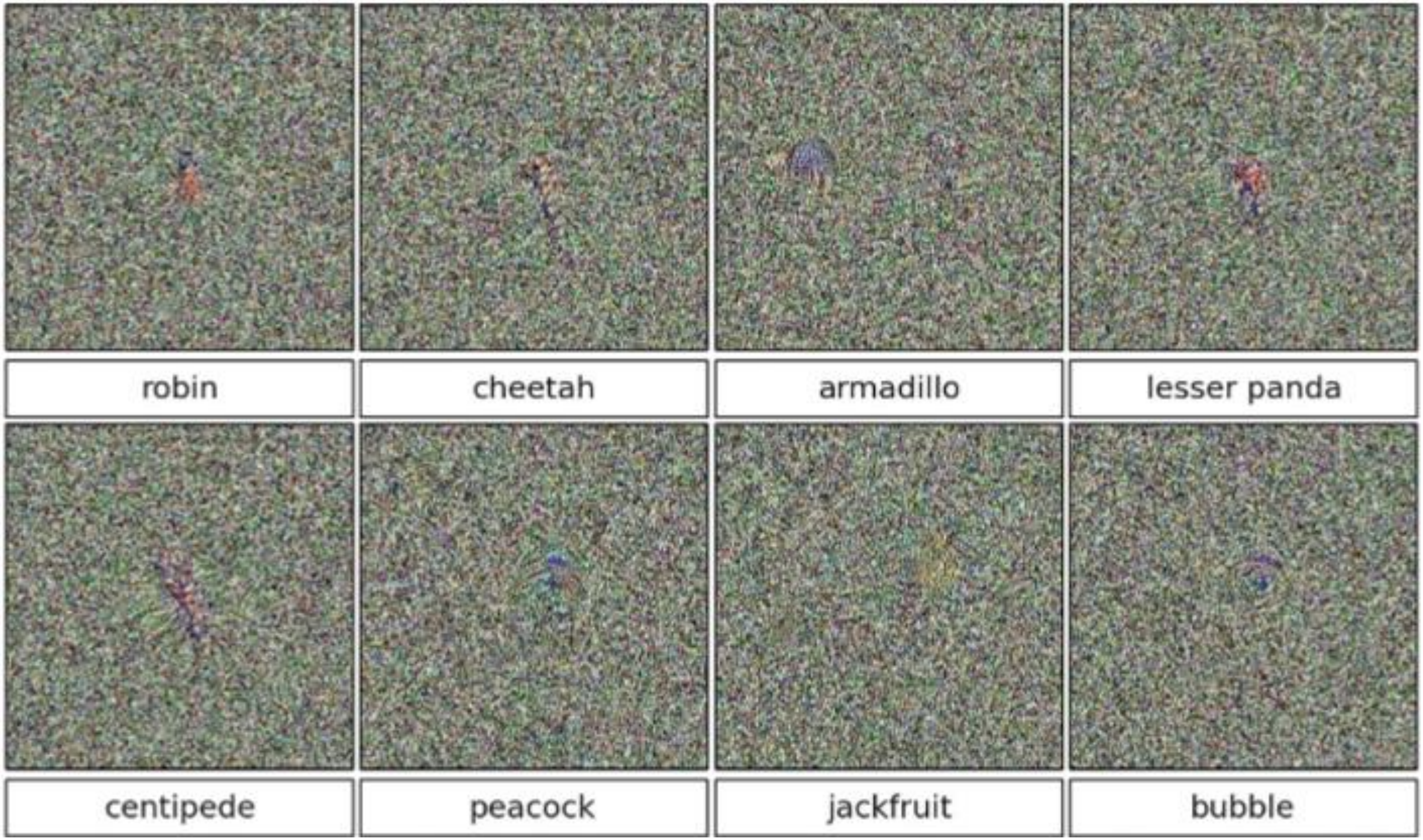
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]

Images that both CNN and Human can recognize



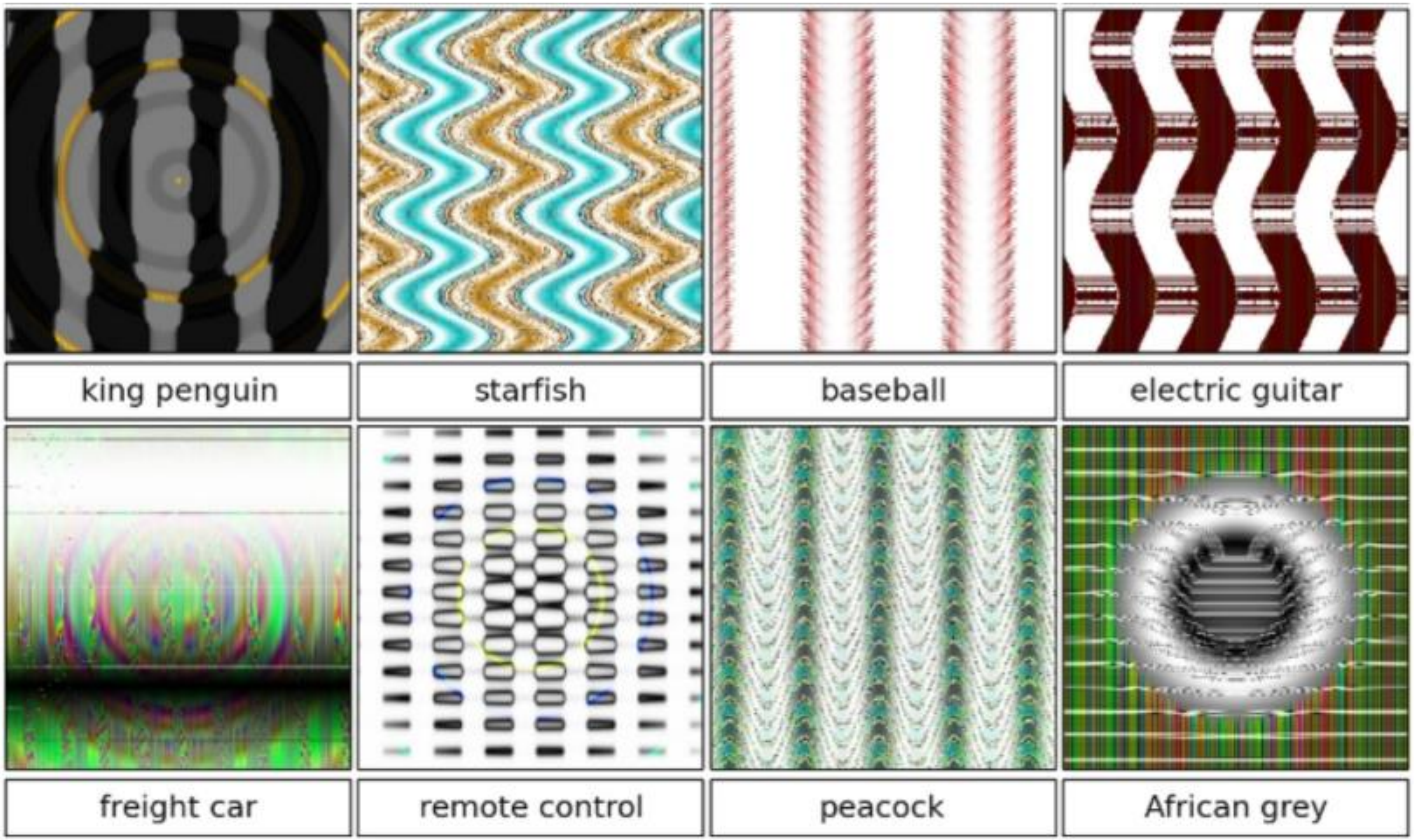
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]

Direct Encoding



Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]

Indirect Encoding



Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]

Training Convolutional Neural Networks

- Backpropagation + stochastic gradient descent with momentum
  – Neural Networks: Tricks of the Trade
- Dropout
- Data augmentation
- Batch normalization
- Initialization
  – Transfer learning

Training CNN with gradient descent

- A CNN as composition of functions

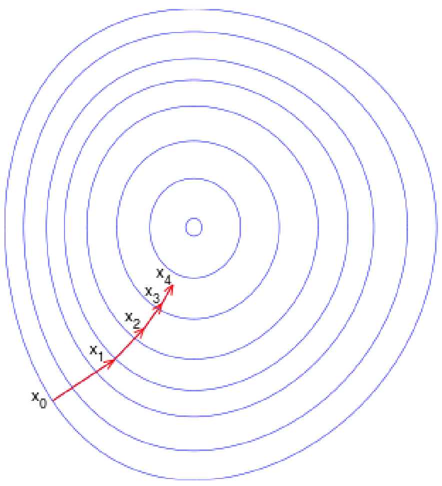
$$f_{\boldsymbol{w}}(\boldsymbol{x}) = f_L(\ldots\ (f_2(f_1(\boldsymbol{x}; \boldsymbol{w}_1); \boldsymbol{w}_2) \ldots ; \boldsymbol{w}_L)$$

- Parameters

$$\boldsymbol{w} = (\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots \boldsymbol{w_L})$$

- Empirical loss function

$$L(\boldsymbol{w}) = \frac{1}{n}\sum_i l(z_i, f_{\boldsymbol{w}}(\boldsymbol{x_i}))$$



- Gradient descent

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta_t \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}(\boldsymbol{w}^{t})$$

New weight

Old weight

Learning rate

Gradient

An Illustrative example

$$f(x,y) = xy, \qquad \frac{\partial f}{\partial x} = y, \frac{\partial f}{\partial y} = x$$

Example: $x = 4, y = -3 \Rightarrow f(x,y) = -12$

Partial derivatives

$$\frac{\partial f}{\partial x} = -3, \qquad \frac{\partial f}{\partial y} = 4$$

Gradient

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$$

Example credit: Andrej Karpathy

$$f(x, y, z) = (x + y)z = qz$$

$$q = x + y$$
$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz$$
$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$

Goal: compute the gradient

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}]$$

Example credit: Andrej Karpathy

$$f(x, y, z) = (x + y)z = qz$$

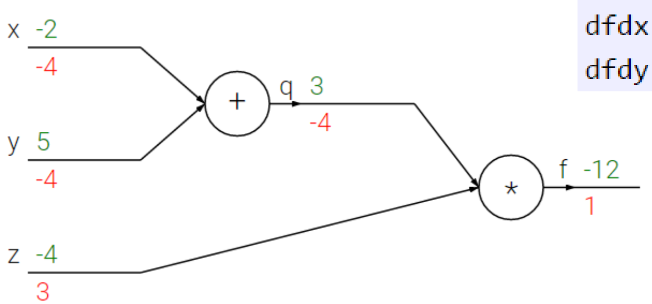
$$q = x + y$$
$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$f = qz$$
$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$

Chain rule:

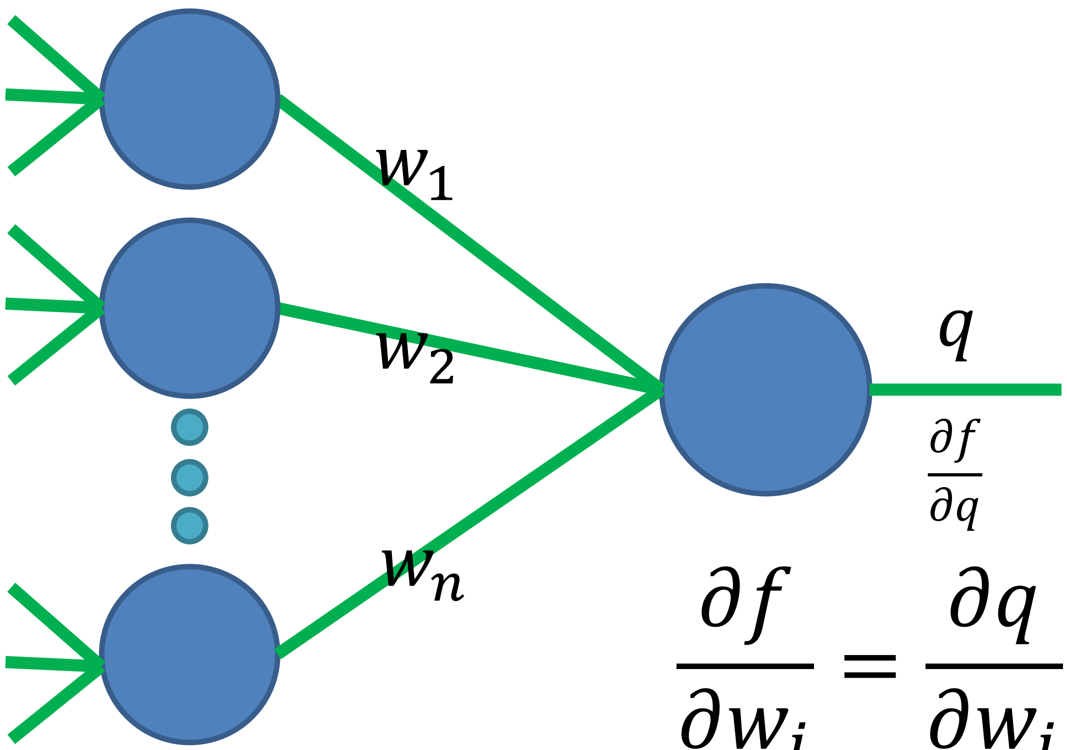
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

```
# set some inputs
x = -2; y = 5; z = -4

# perform the forward pass
q = x + y # q becomes 3
f = q * z # f becomes -12

# perform the backward pass (backpropagation) in reverse order:
# first backprop through f = q * z
dfdz = q # df/dz = q, so gradient on z becomes 3
dfdq = z # df/dq = z, so gradient on q becomes -4
# now backprop through q = x + y
dfdx = 1.0 * dfdq # dq/dx = 1. And the multiplication here is the chain rule!
dfdy = 1.0 * dfdq # dq/dy = 1
```



Example credit: Andrej Karpathy

Backpropagation (recursive chain rule)

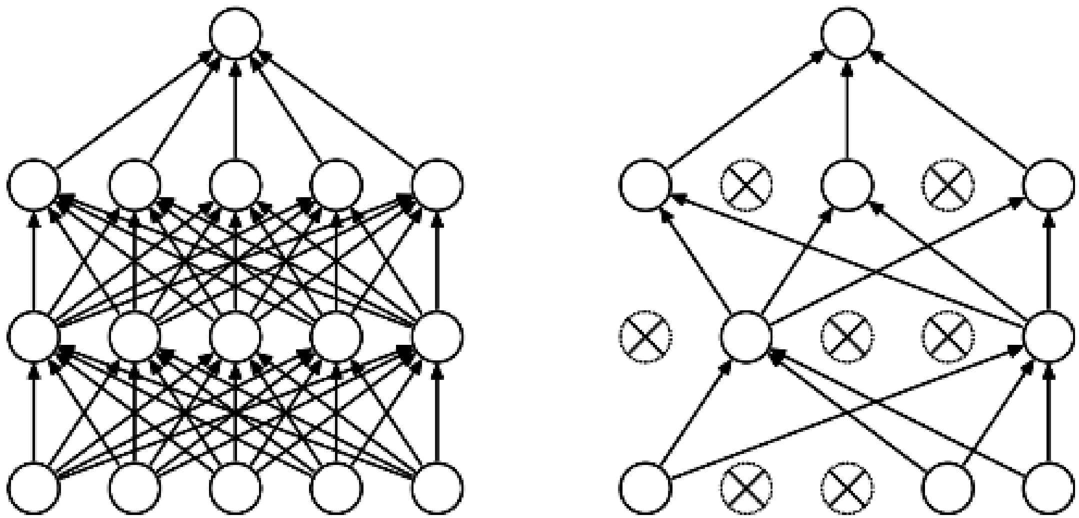


$$\frac{\partial f}{\partial w_i} = \frac{\partial q}{\partial w_i} \frac{\partial f}{\partial q}$$

Local gradient — Can be computed during forward pass

Gate gradient — The gate receives this during backprop

Dropout



(a) Standard Neural Net (b) After applying dropout.

Intuition: successful conspiracies

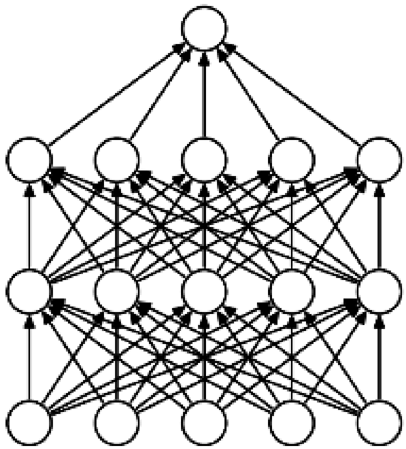
- 50 people planning a conspiracy

- Strategy A: plan a big conspiracy involving 50 people
  - Likely to fail. 50 people need to play their parts correctly.

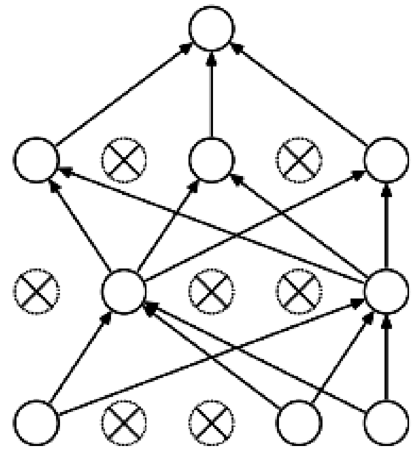
- Strategy B: plan 10 conspiracies each involving 5 people
  - Likely to succeed!

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

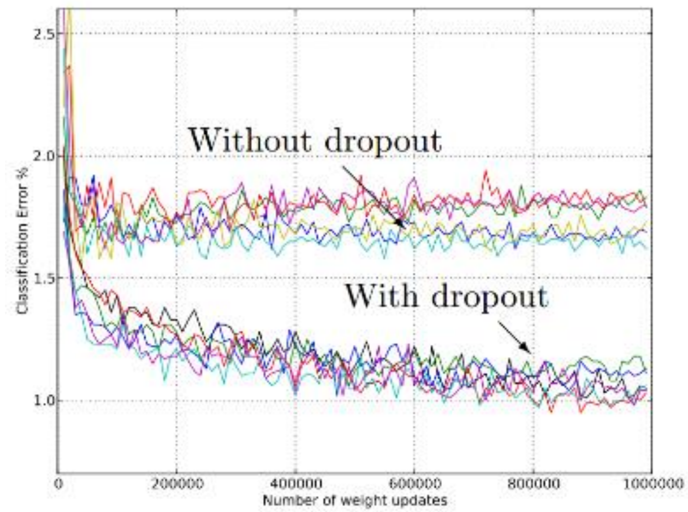
Dropout

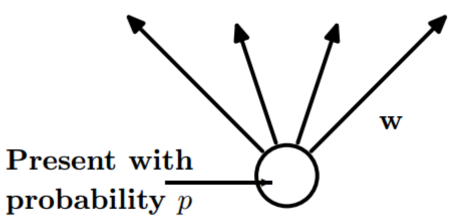


(a) Standard Neural Net

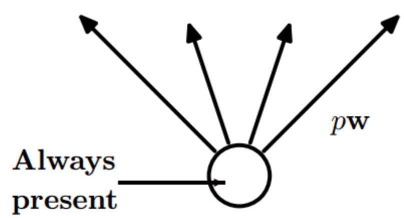


(b) After applying dropout.





(a) At training time



(b) At test time

Main Idea: approximately combining exponentially many different neural network architectures efficiently

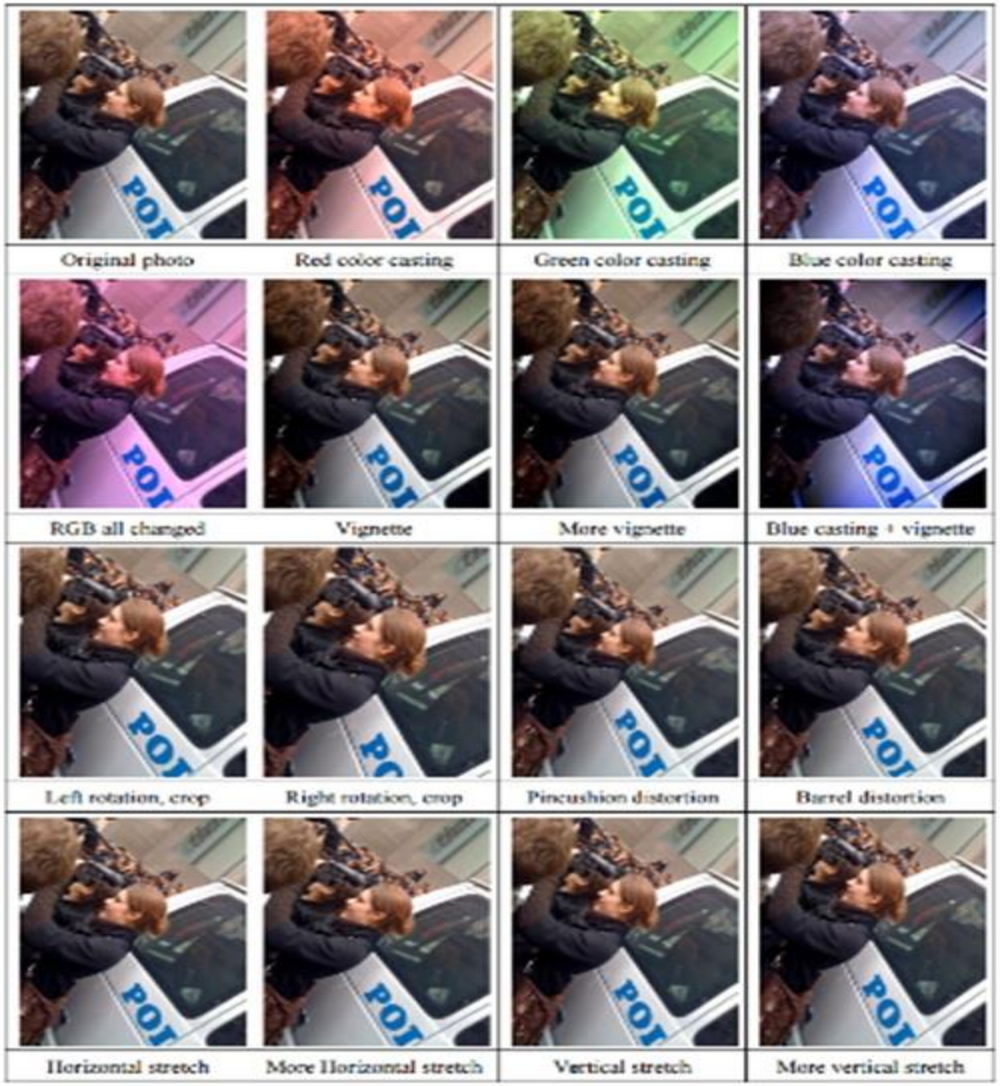
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

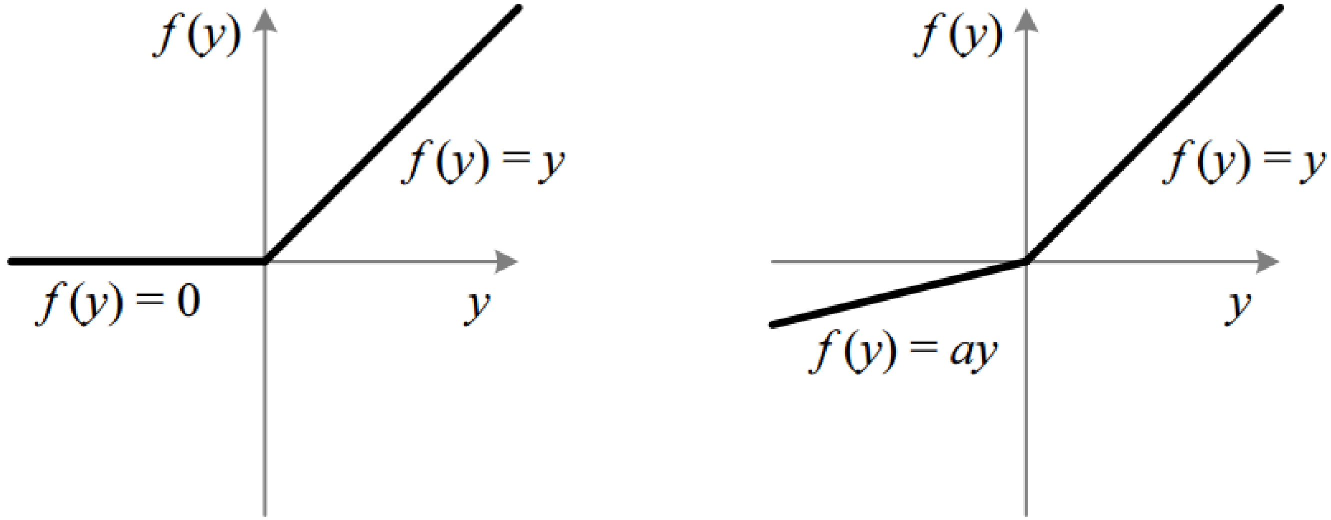
Data Augmentation (Jittering)

- Create *virtual* training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion



Deep Image [Wu et al. 2015]

Parametric Rectified Linear Unit



	team	top-5 (**test**)
in competition ILSVRC 14	MSRA, SPP-nets [11]	8.06
	VGG [25]	7.32
	GoogLeNet [29]	6.66
post-competition	VGG [25] (arXiv v5)	6.8
	Baidu [32]	5.98
	MSRA, PReLU-nets	**4.94**

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [He et al. 2015]

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

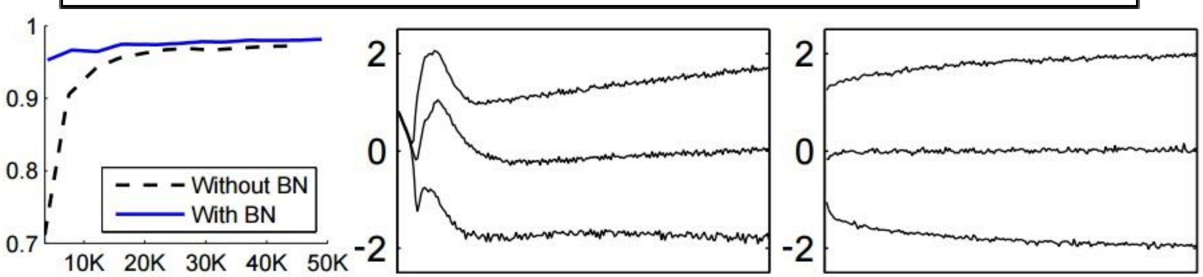
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

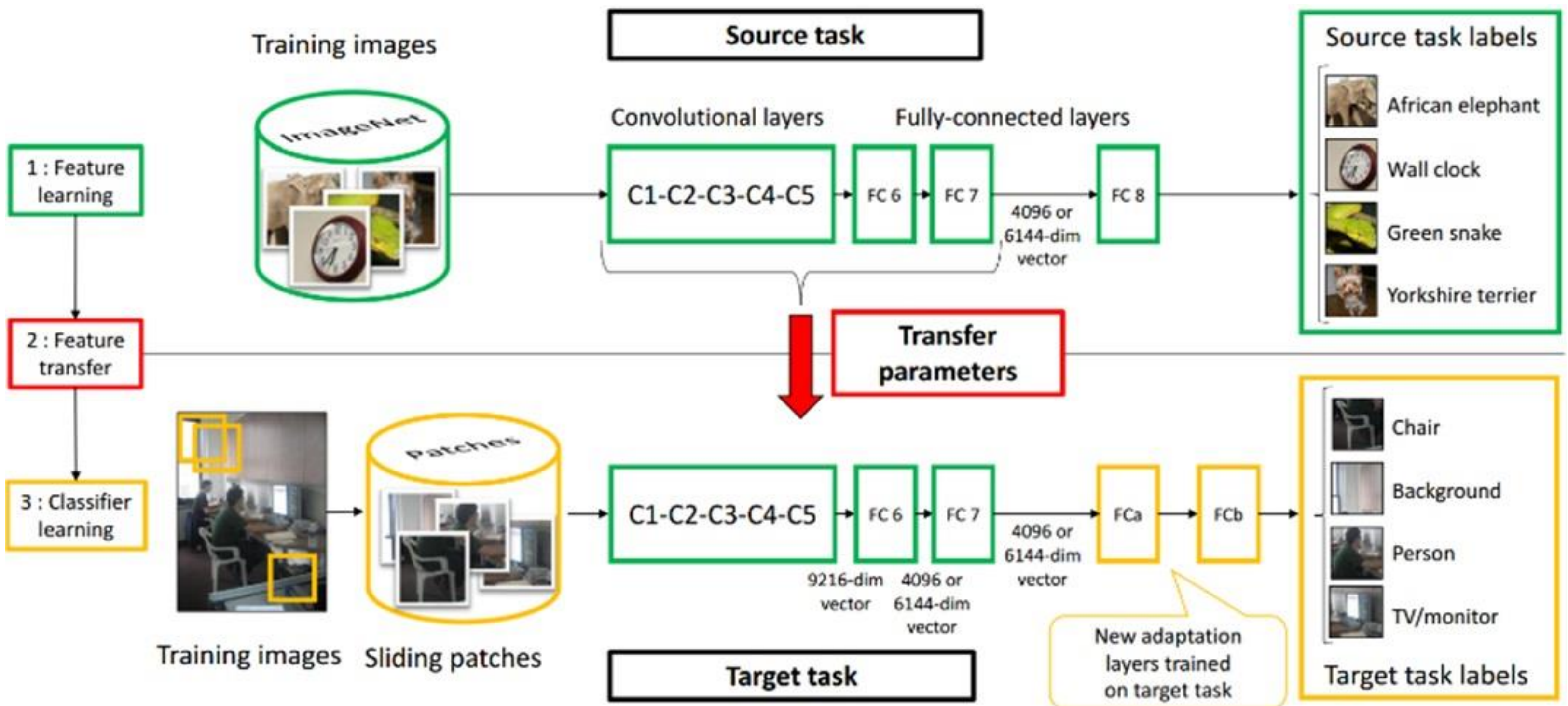


(a) (b) Without BN (c) With BN

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

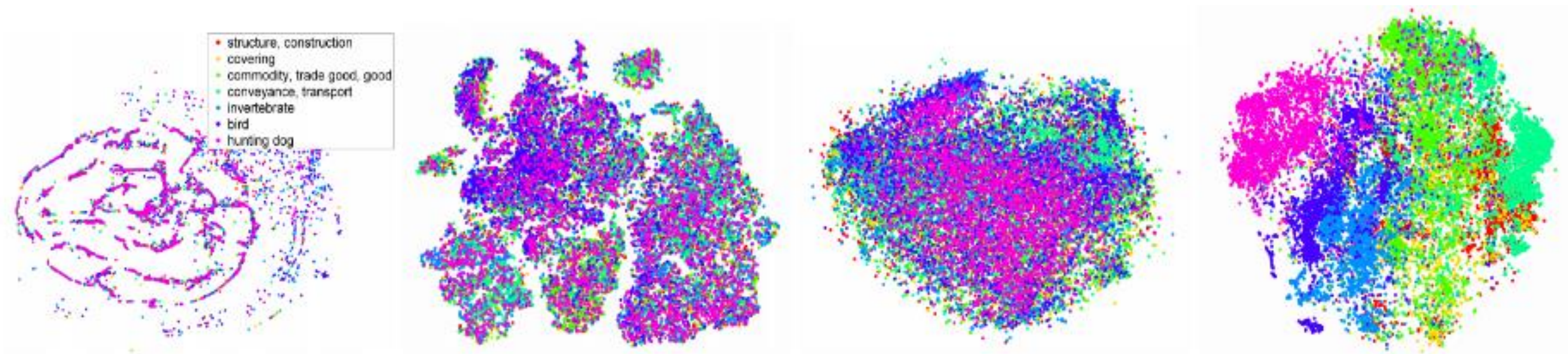
Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Weight initialization for CNN



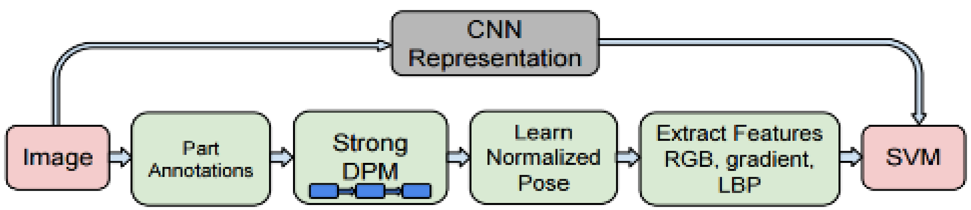
Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014]

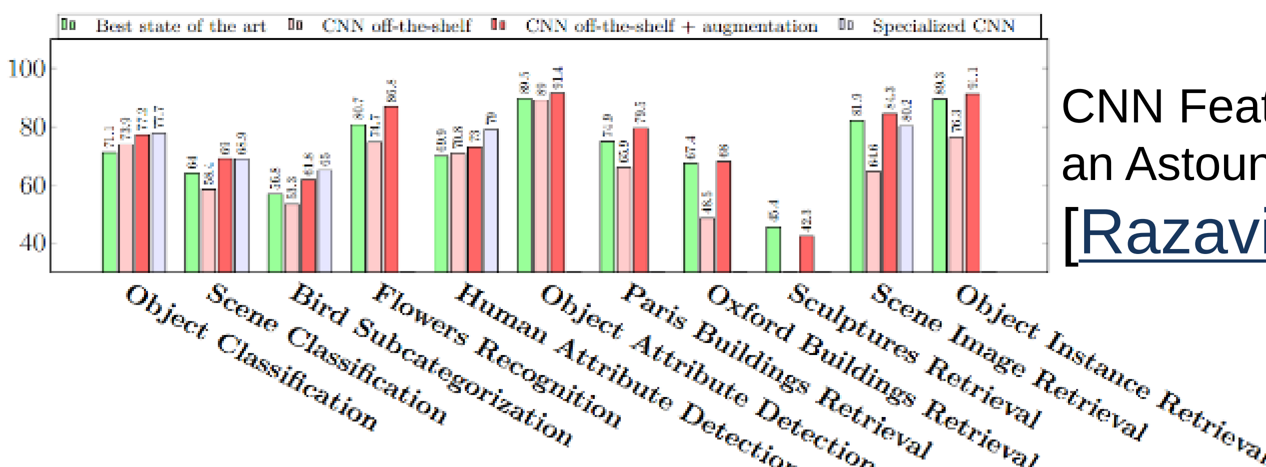
Convolutional activation features



(a) LLC (b) GIST (c) DeCAF_1 (d) DeCAF_6

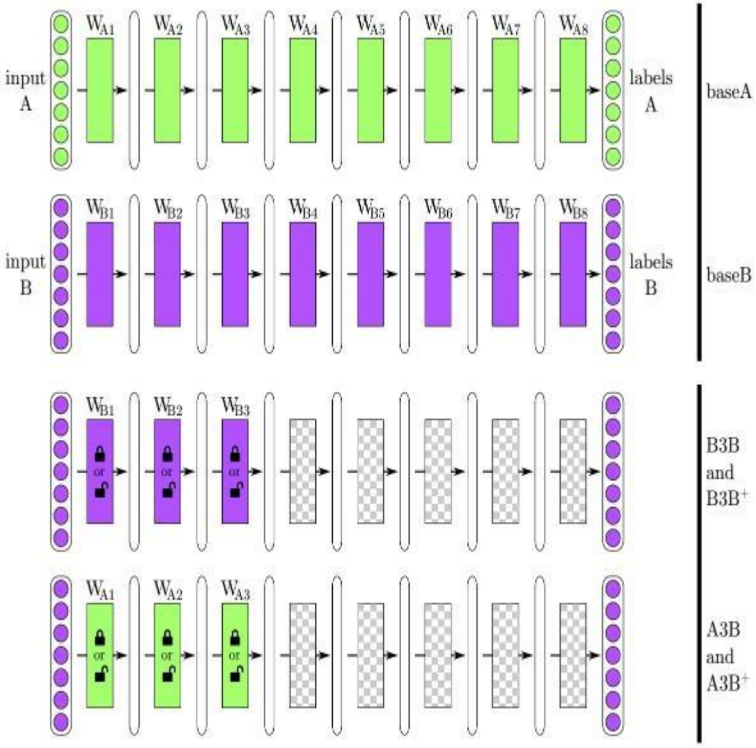
[Donahue et al. ICML 2013]

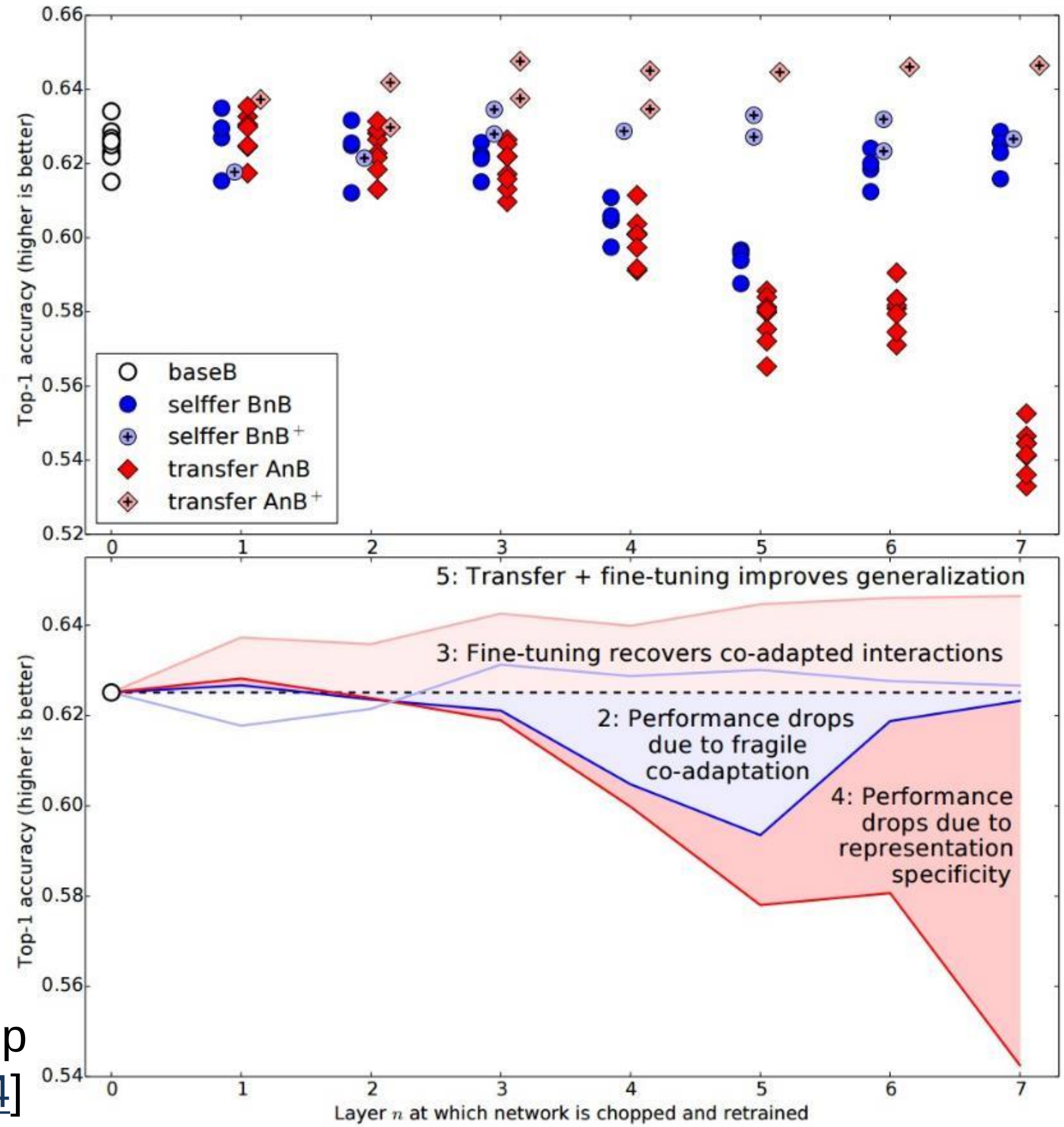




CNN Features off-the-shelf:
an Astounding Baseline for Recognition
[Razavian et al. 2014]

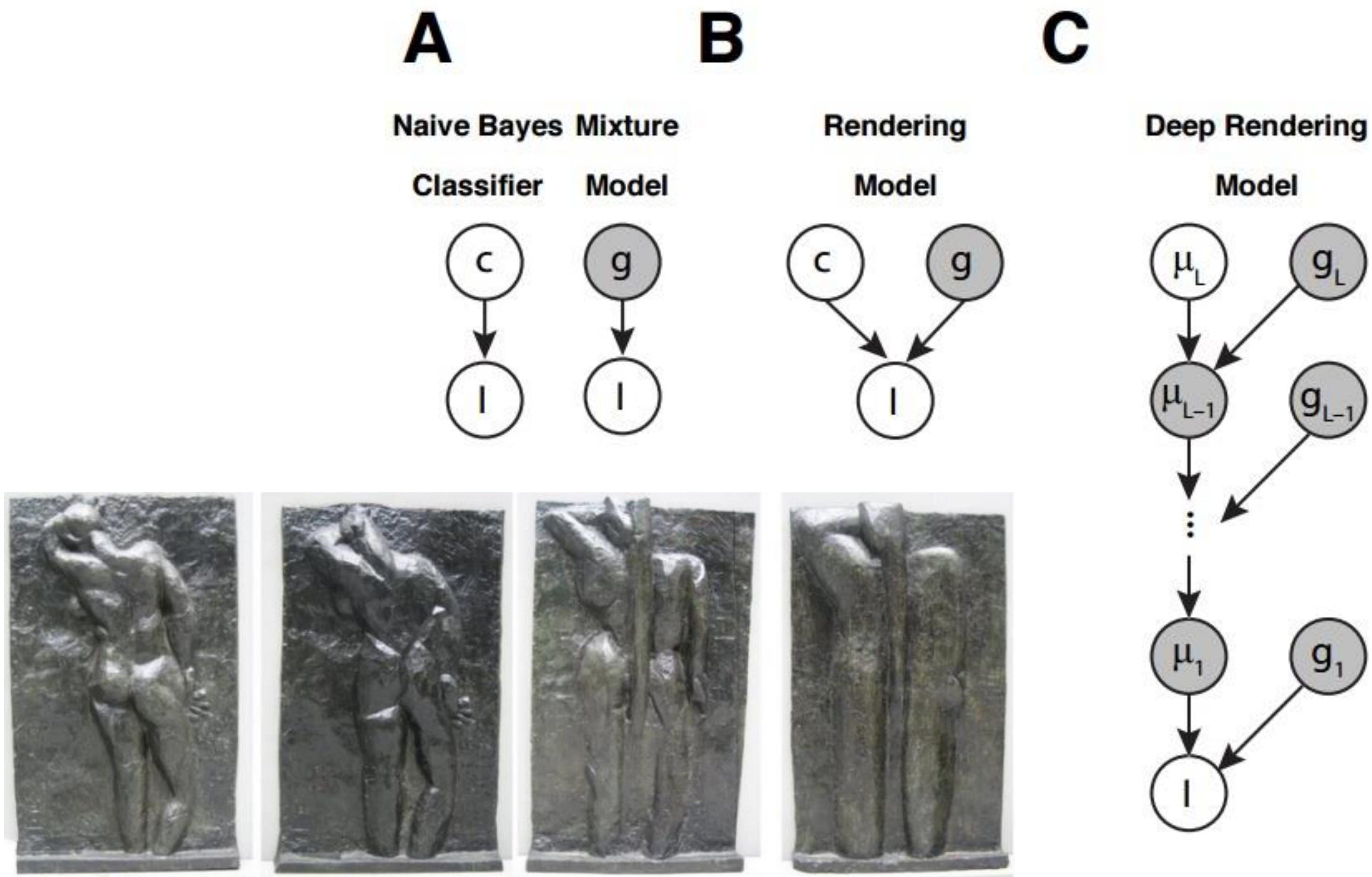
How transferable are features in CNN?





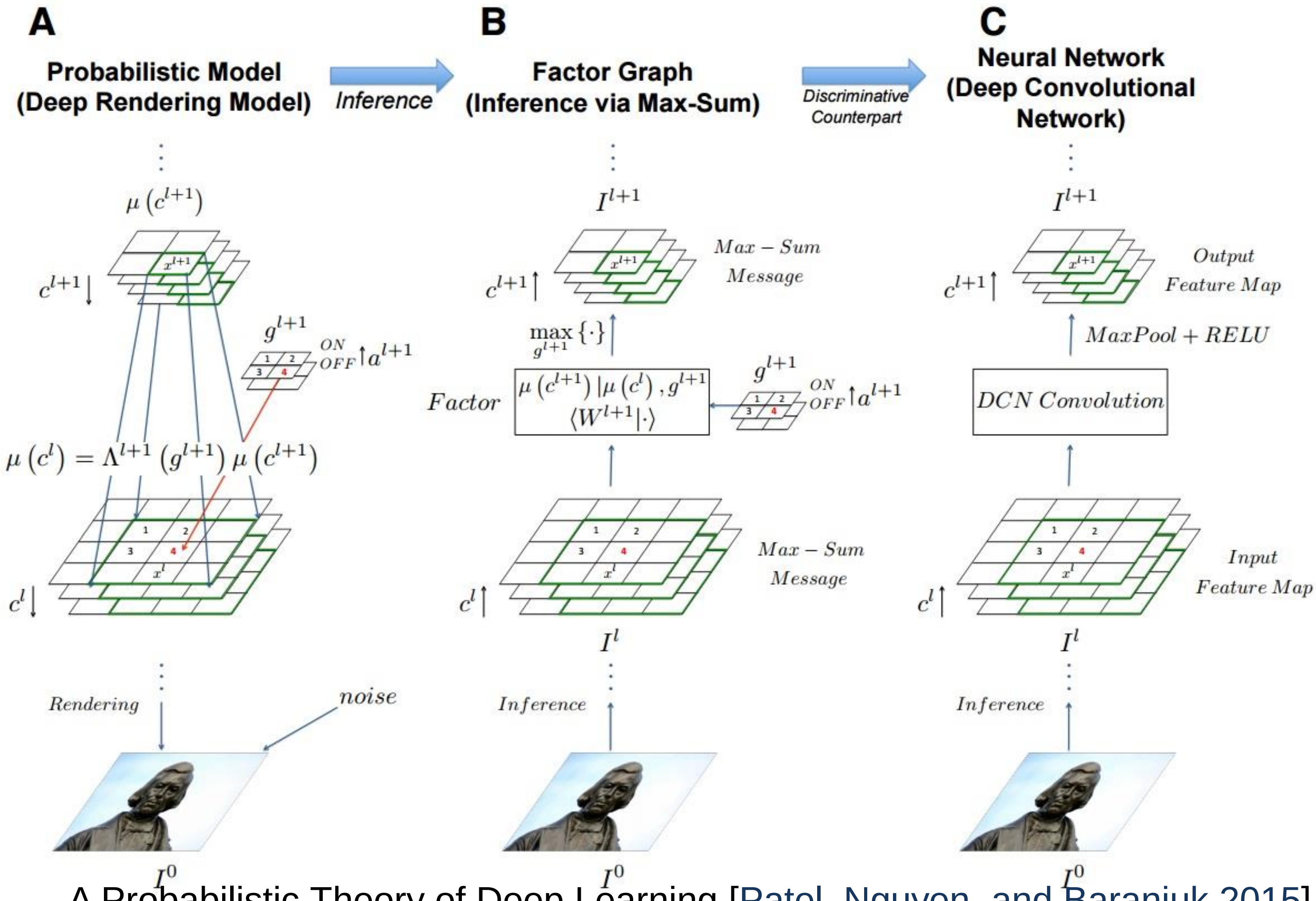
How transferable are features in deep neural networks [Yosinski NIPS 2014]

Deep Rendering Model (DRM)



A Probabilistic Theory of Deep Learning [Patel, Nguyen, and Baraniuk 2015]

CNN as a Max-Sum Inference



A Probabilistic Theory of Deep Learning [Patel, Nguyen, and Baraniuk 2015]

	Aspect	Neural Nets Perspective *Deep Convnets (DCNs)*	Probabilistic Perspective *Deep Rendering Model (DRM)*
	Model	Weights and biases of filters at a given layer	Partial Rendering at a given abstraction level/scale
		Number of Layers	Number of Abstraction Levels
Model		Number of Filters in a layer	Number of Clusters/Classes at a given abstraction level
		Implicit in network weights; can be computed by product of weights over all layers or by activity maximization	Category prototypes are finely detailed versions of coarser-scale super-category prototypes. Fine details are modeled with affine nuisance transformations.
	Inference	Forward propagation thru DCN	Exact bottom-up inference via Max-Sum Message Passing (with Max-Product for Nuisance Factorization).
		Input and Output Feature Maps	Probabilistic Max-Sum Messages (real-valued functions of variables nodes)
Inference		Template matching at a given layer (convolutional, locally or fully connected)	Local computation at factor node (log-likelihood of measurements)
		Max-Pooling over local pooling region	Max-Marginalization over Latent Translational Nuisance transformations
		Rectified Linear Unit (ReLU). Sparsifies output activations.	Max-Marginalization over Latent Switching state of Renderer. Low prior probability of being ON.
	Learning	Stochastic Gradient Descent	Batch Discriminative EM Algorithm with Fine-to-Coarse E-step + Gradient M-step. *No coarse-to-fine pass in E-step.*
Learning		N/A	Full EM Algorithm
		Batch-Normalized SGD (Google state-of-the-art [BN])	Discriminative Approximation to Full EM (assumes Diagonal Pixel Covariance)

Tools

- Caffe
- cuda-convnet2
- Torch
- MatConvNet
- Pylearn2
- TensorFlow

Resources

- http://deeplearning.net/
  - Hub to many other deep learning resources
- https://github.com/ChristosChristofidis/awesome-deep-learning
  - A resource collection deep learning
- https://github.com/kjw0612/awesome-deep-vision
  - A resource collection deep learning for computer vision
- http://cs231n.stanford.edu/syllabus.html
  - Nice course on CNN for visual recognition

Things to remember

- Overview
  - Neuroscience, Perceptron, multi-layer neural networks
- Convolutional neural network (CNN)
  - Convolution, nonlinearity, max pooling
  - CNN for classification and beyond
- Understanding and visualizing CNN
  - Find images that maximize some class scores; visualize individual neuron activation, input pattern and images; breaking CNNs
- Training CNN
  - Dropout; data augmentation; batch normalization; transfer learning
- Probabilistic interpretation
  - Deep rendering model; CNN forward-propagation as max-sum inference; training as an EM algorithm


xtranormal