CMPSCI 670: Computer Vision

Deep learning

University of Massachusetts, Amherst
November 17, 2014

Instructor: Subhransu Maji

Administrivia

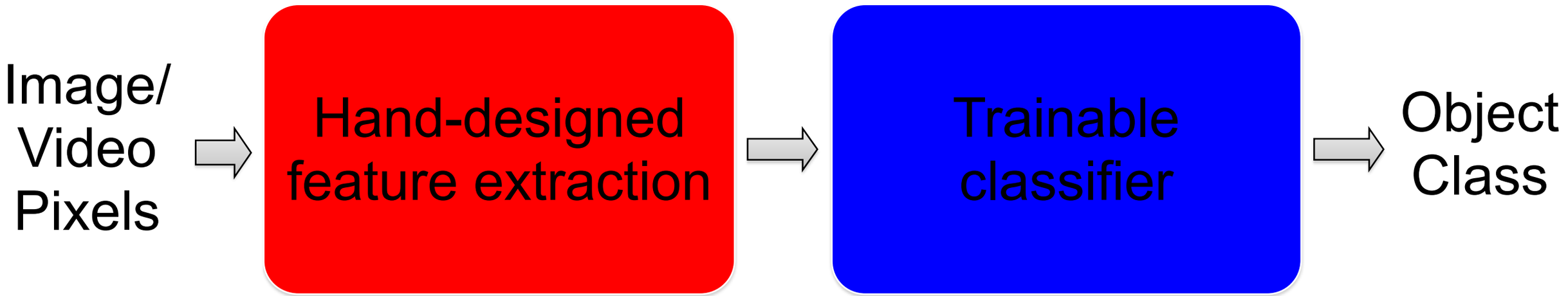
- Homework 4 - grades posted
- Homework 5 - due on Wednesday
- Project
  - Presentations on Dec. 1 and 3
  - Each person (or team) will get 7 (or 10) mins to present
    - Preliminary results, data analysis, etc
  - Final report due on Dec. 13 (hard deadline)
- Next lecture is a guest lecture by:
  - “Crafting the Perfect Selfie using Computer Vision” Aditya Khosla, MIT

Overview

- Shallow vs. deep architectures
- Background
  - Traditional neural networks
  - Inspiration from neuroscience
- Stages of CNN architecture
- Visualizing CNNs
- State-of-the-art results
- Packages

Many slides are by Rob Fergus and S. Lazebnik

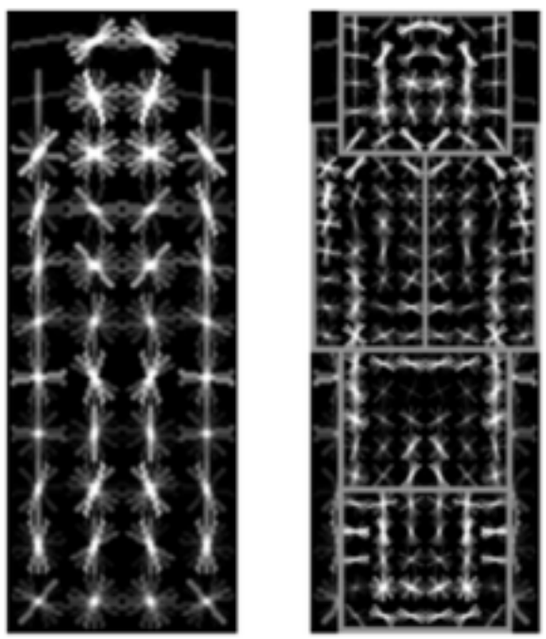
Traditional Recognition Approach



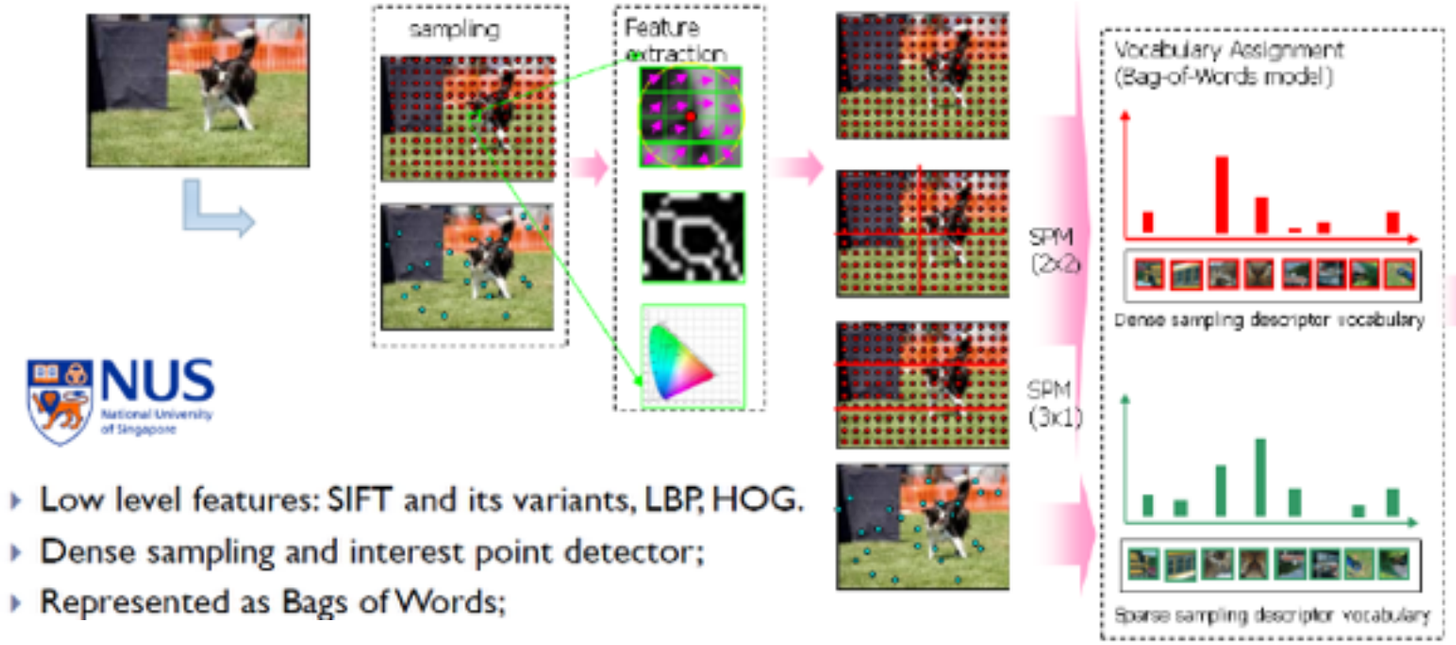
- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

Traditional Recognition Approach

- Features are key to recent progress in recognition
- Multitude of hand-designed features currently in use
  - SIFT, HOG, ……
- Where next? Better classifiers? Or keep building more features?



Felzenszwalb, Girshick,
McAllester and Ramanan, PAMI 2007



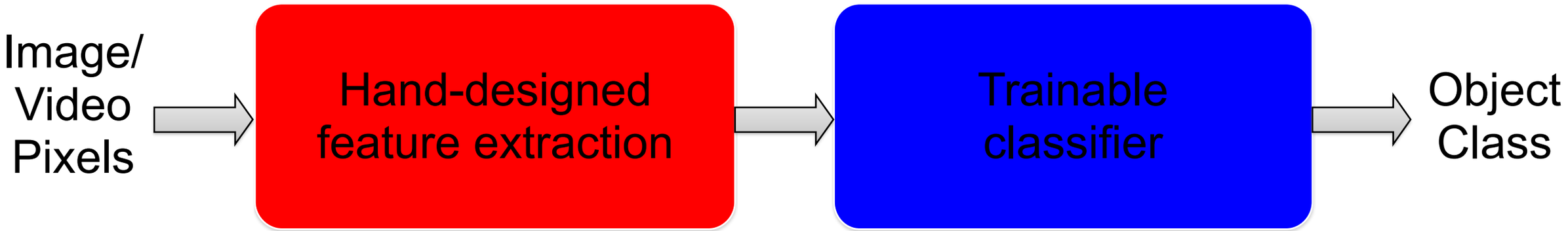
Yan & Huang
(Winner of PASCAL 2010 classification competition)

What about learning the features?

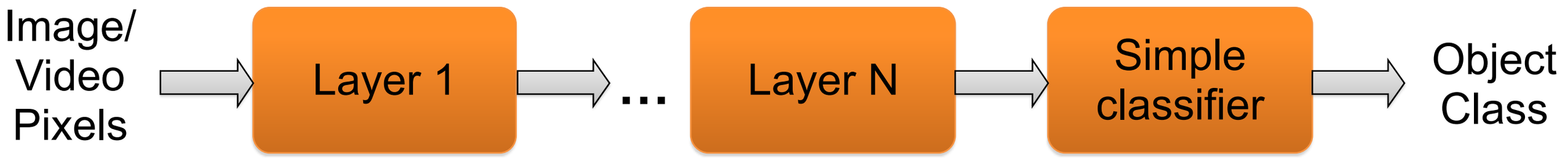
- Learn a ***feature hierarchy*** all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

“Shallow” vs. “deep” architectures

Traditional recognition: “Shallow” architecture



Deep learning: “Deep” architecture



Artificial neural networks

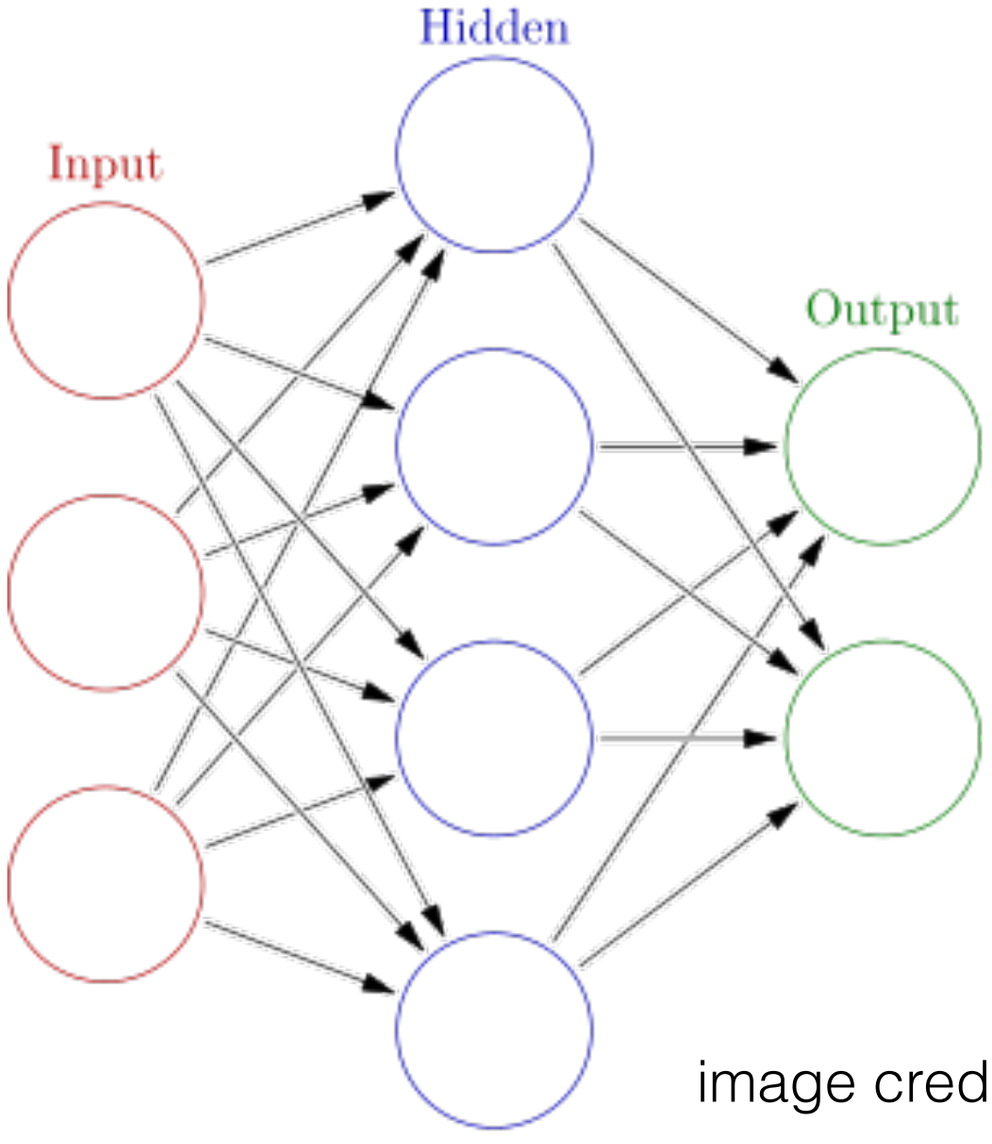
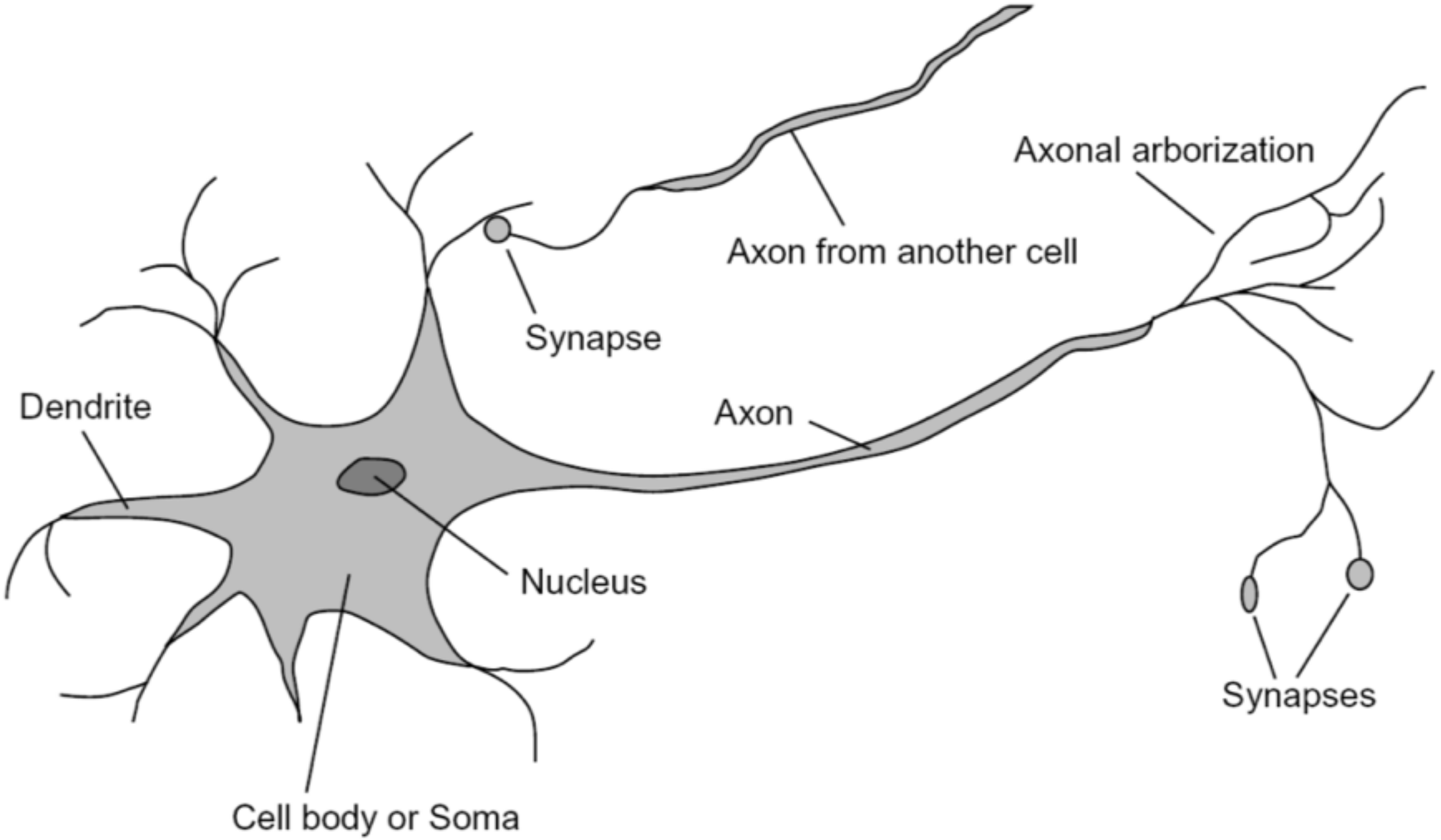


image credit wikipedia

- Artificial neural network is a group of interconnected nodes
- Circles here represent artificial “neurons”
- Note the directed arrows (denoting the flow of information)

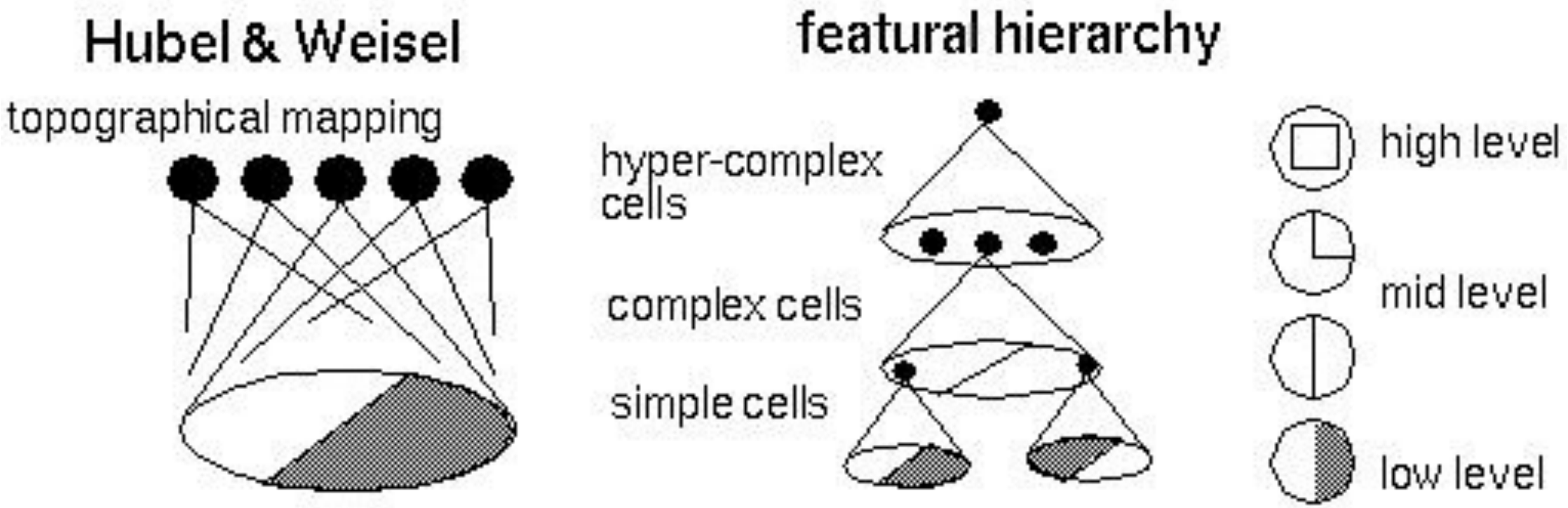
Inspiration: Neuron cells



http://en.wikipedia.org/wiki/Neuron

Hubel/Wiesel Architecture

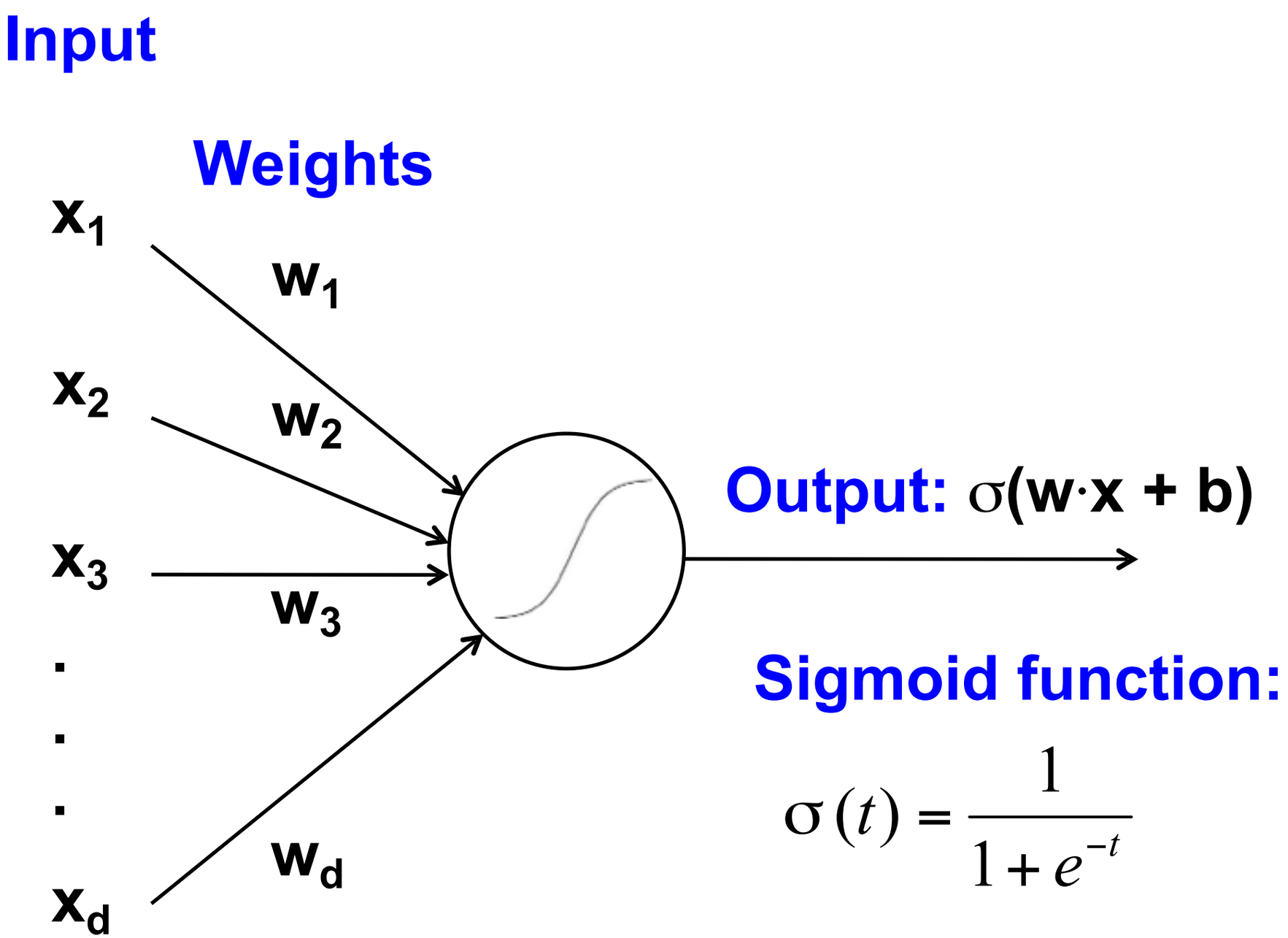
- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
- Visual cortex consists of a hierarchy of simple, complex, and hyper-complex cells



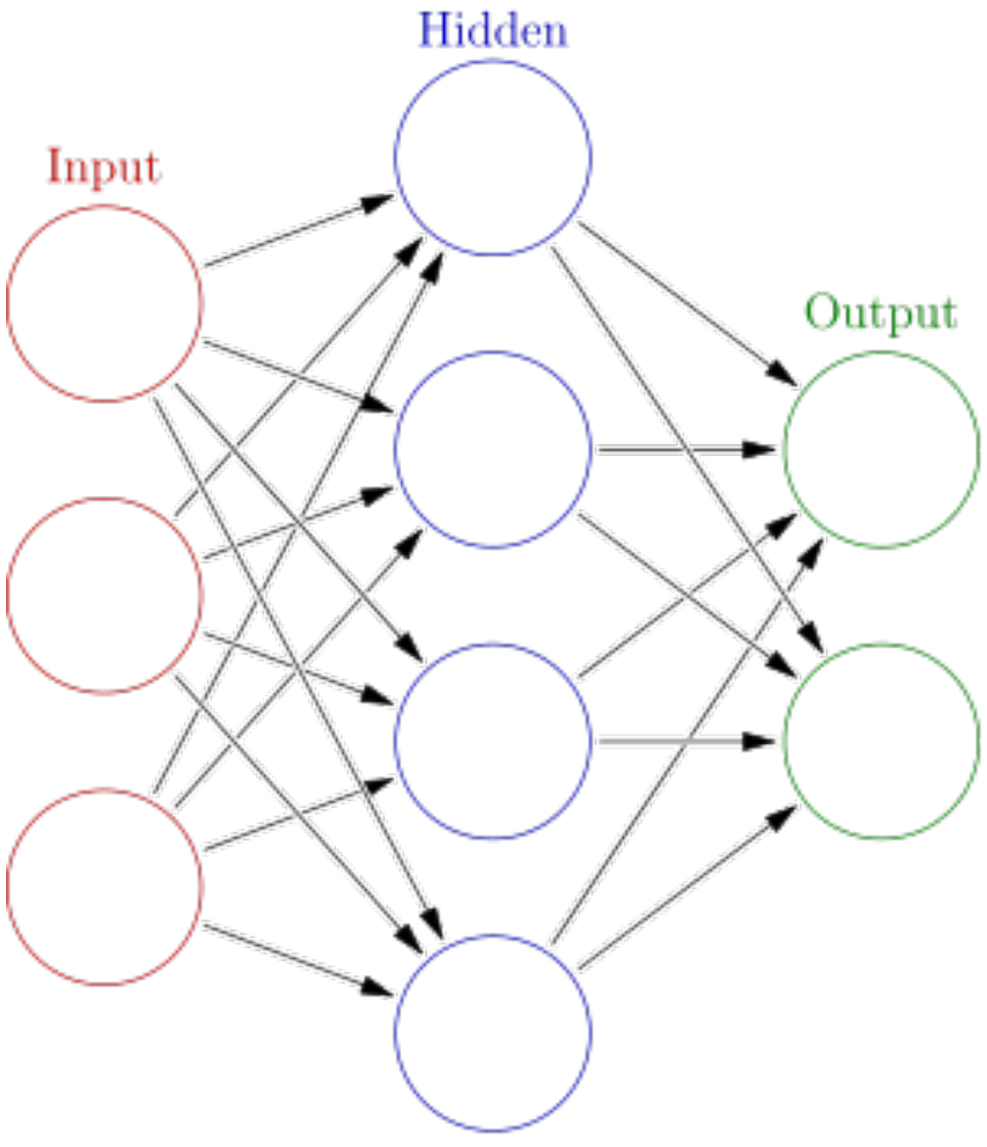
Source

The basic unit of computation

"Peceptron", Frank Rosenblatt 1957

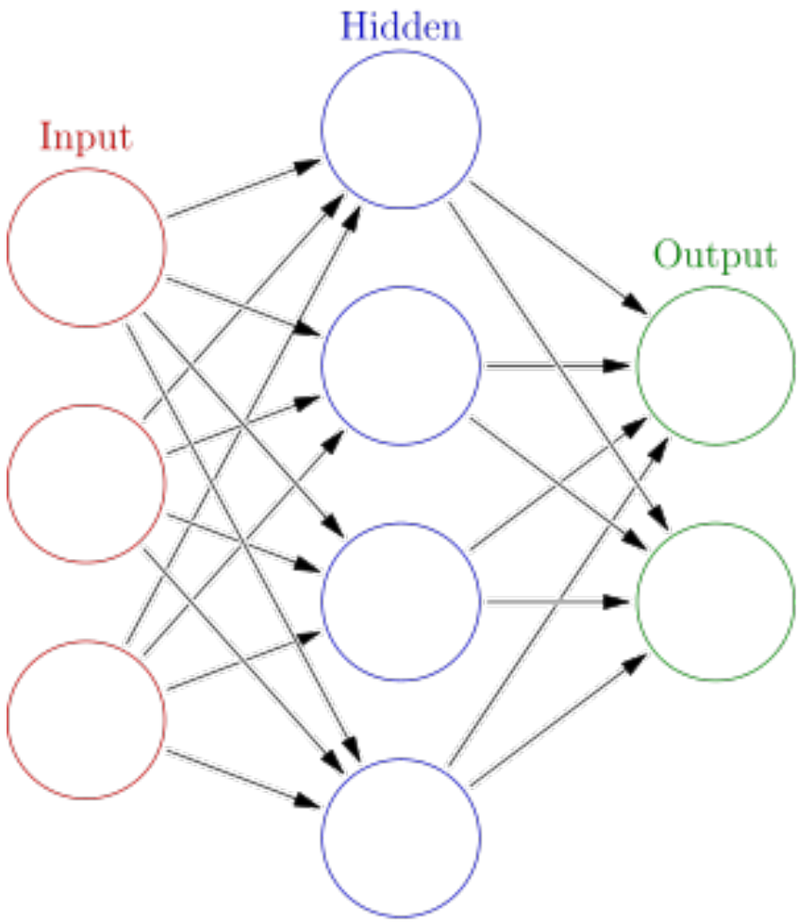


Non-linearity is important



- Without non-linearity, the whole system is linear
- Unfortunately, neural network research stagnated for decades after the publication by Minsky and Papert, 1969, who showed that a perceptron cannot represent the “xor” function

Training ANNs



we know the
desired output

“Chain rule” of gradient

$$d\mathbf{f}(\mathbf{g}(x))/dx = (d\mathbf{f}/d\mathbf{g})(d\mathbf{g}/dx)$$

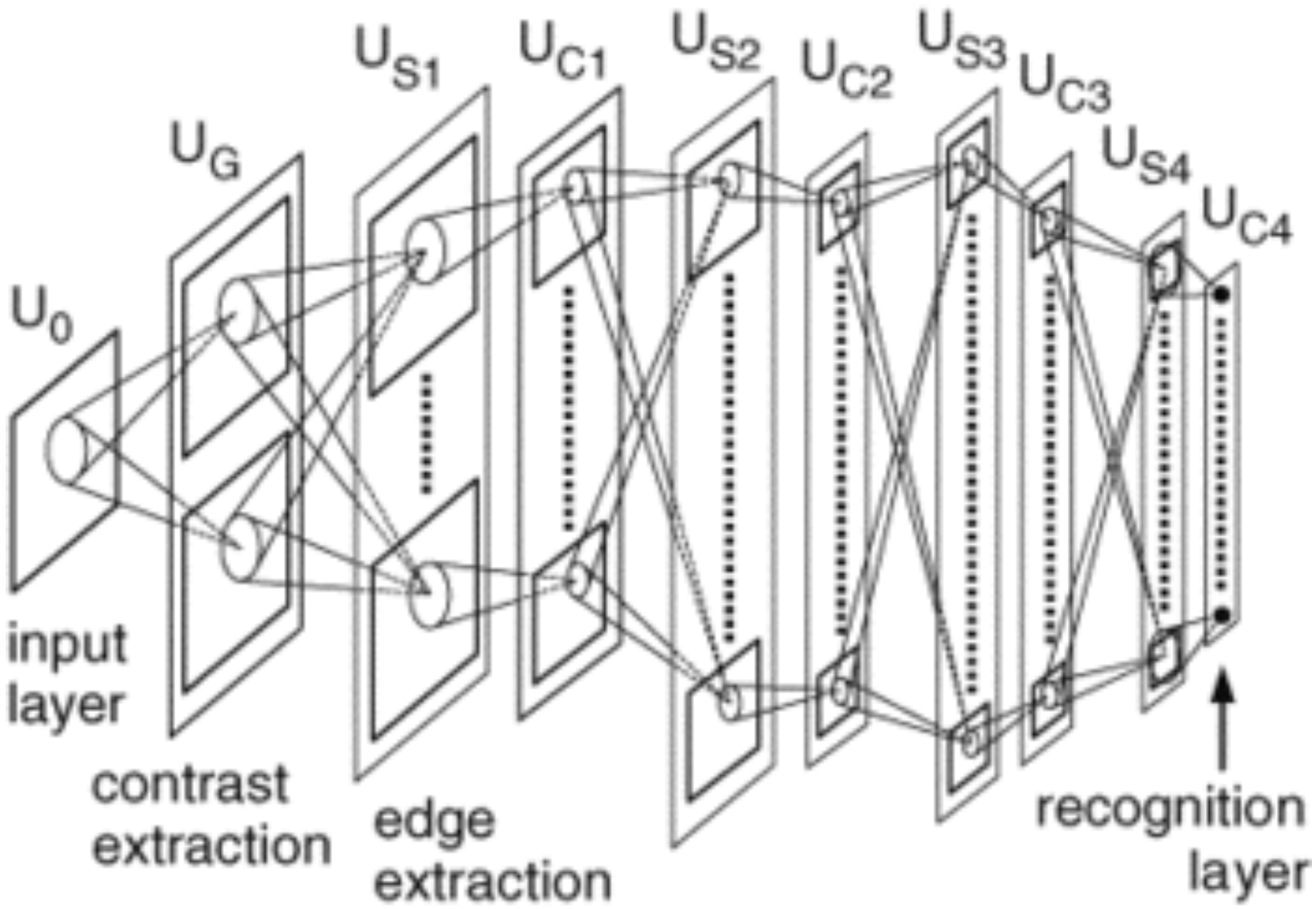
- **Back-propagate** the gradients to match the outputs
- Were too impractical till computers became faster

http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf

Issues with ANNs

- In the 1990s, simpler and faster learning methods such as SVMs and boosting were favored over ANNs.
- Why?
  - Need many layers to learn good features — many parameters need to be learned
  - Needs vast amounts of training data (related to the earlier point)
  - Convergence is slow, get stuck in local minima
  - Vanishing gradients for deep models

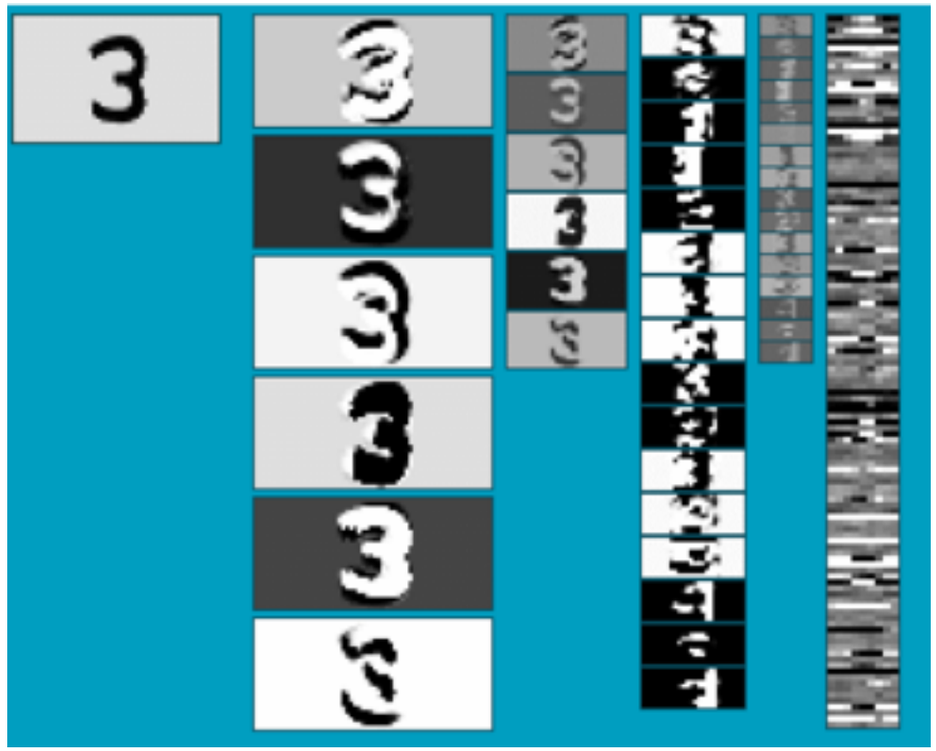
ANNs for vision

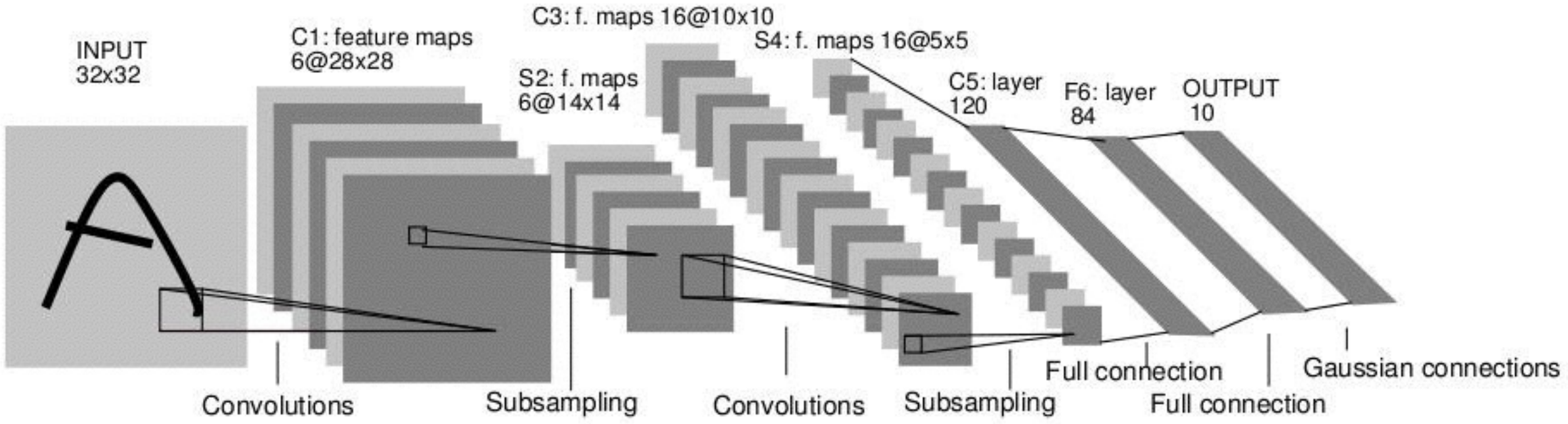


The **neocognitron**, by Fukushima (1980)
(But he didn't propose a way to learn these models)

Convolutional Neural Networks

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end

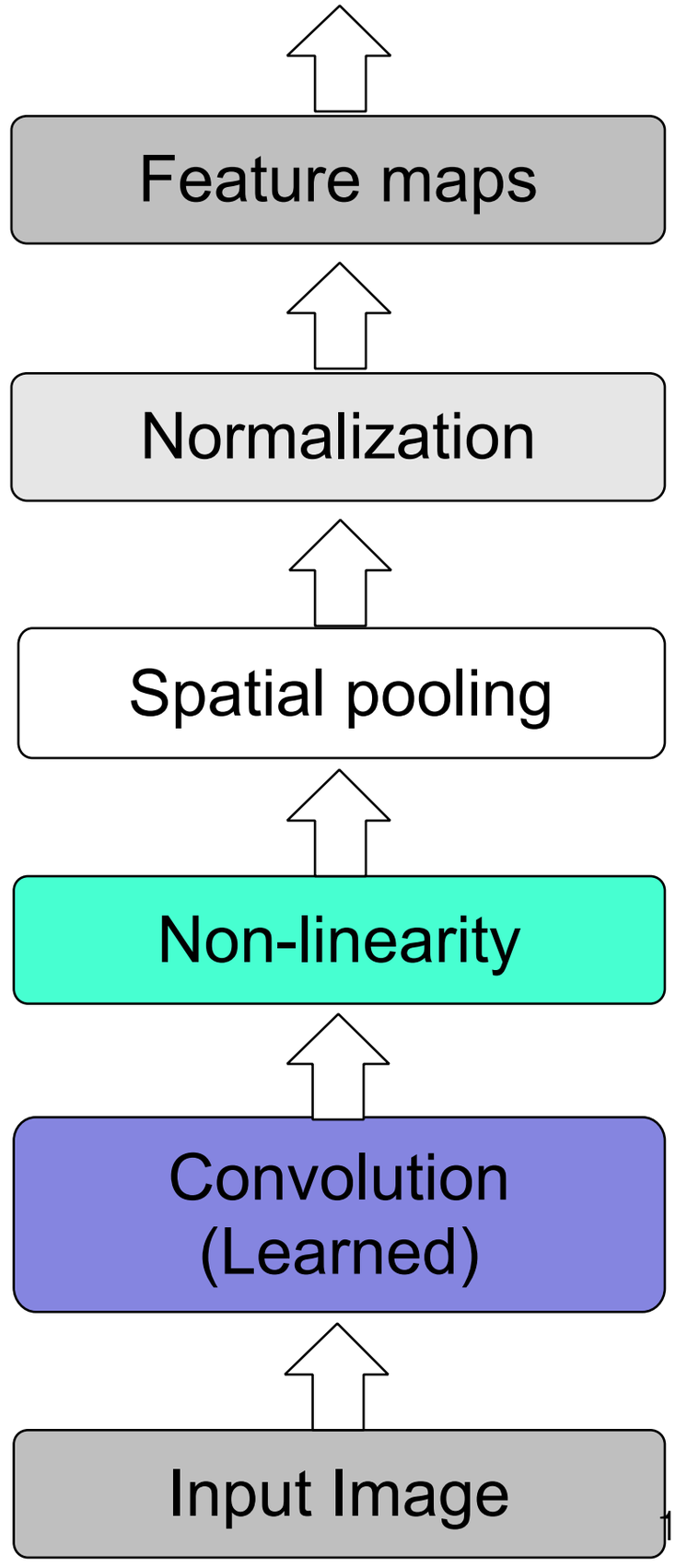




Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

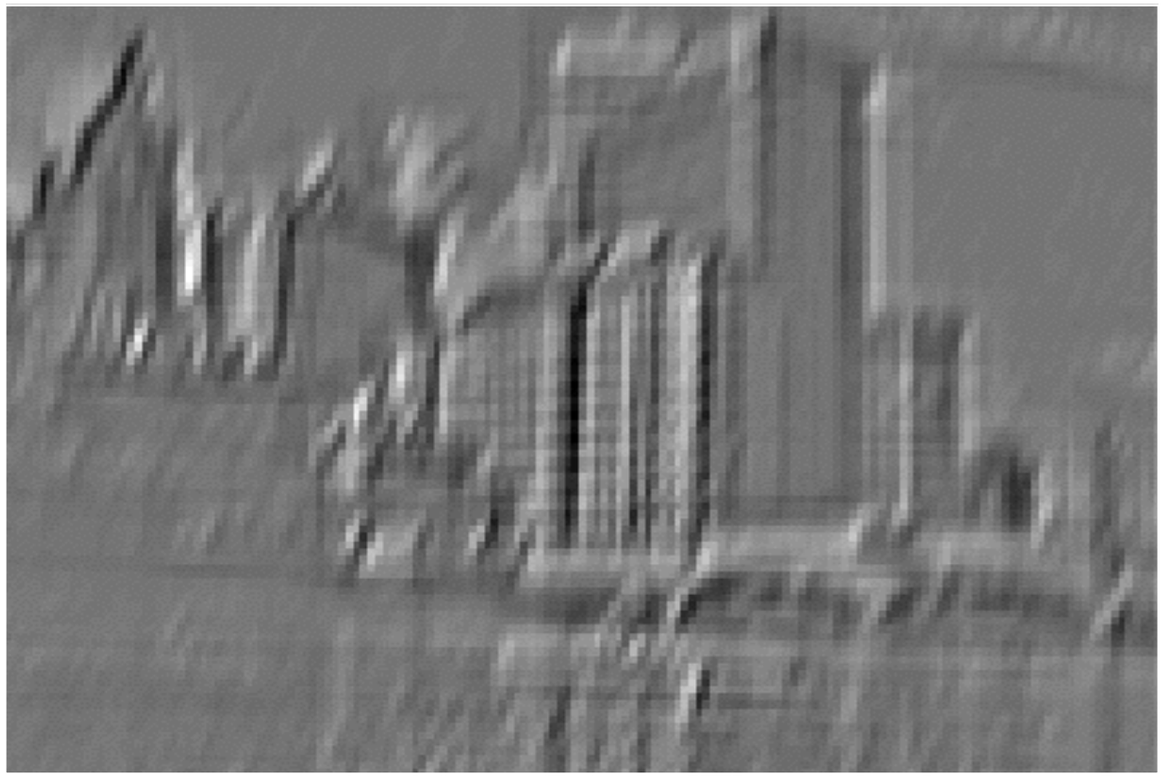
Convolutional Neural Networks

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. Normalization
- Supervised training of convolutional filters by back-propagating classification error

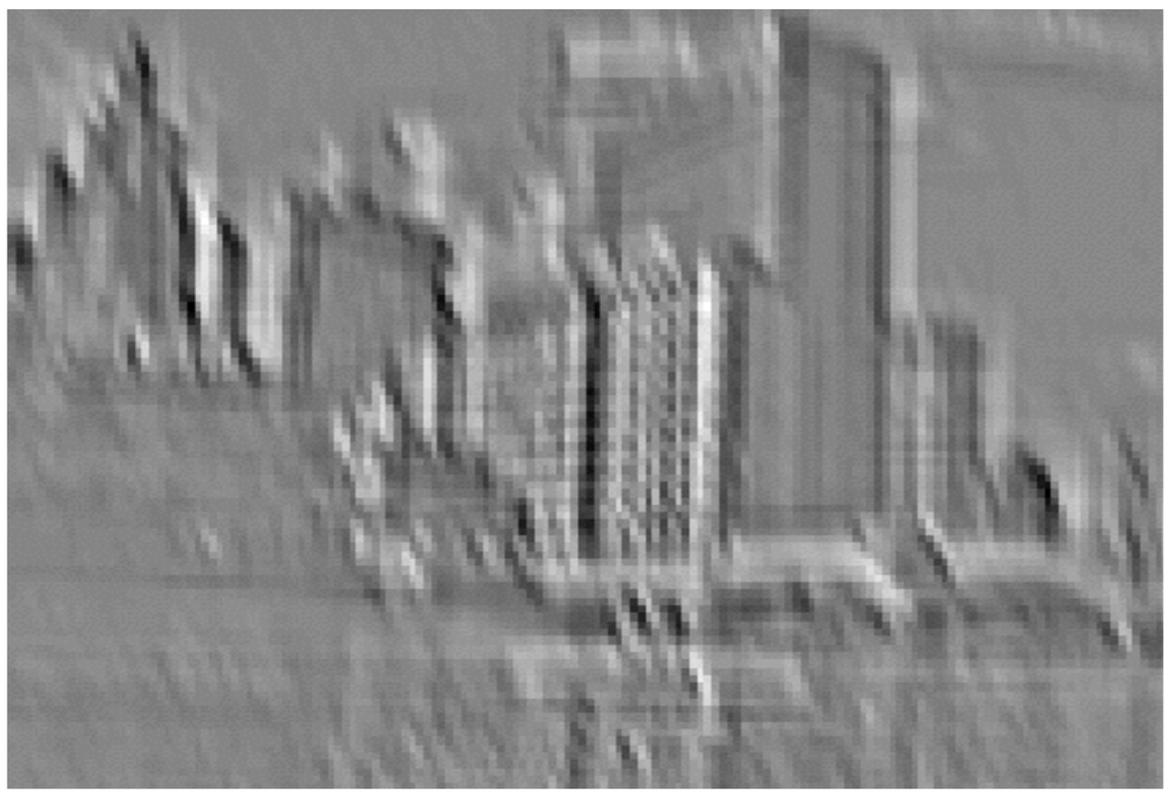


1. Convolution

- Dependencies are local
- Translation invariance
- Few parameters (filter weights)
- Stride can be greater than 1 (faster, less memory)



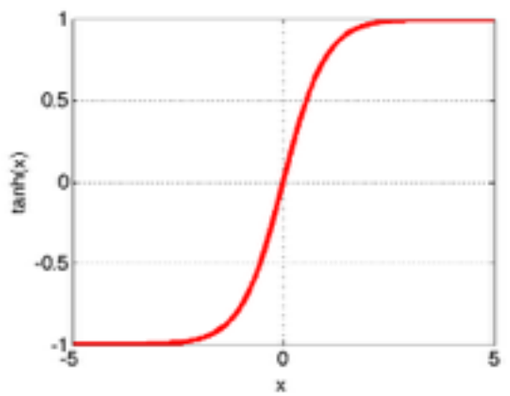
Input

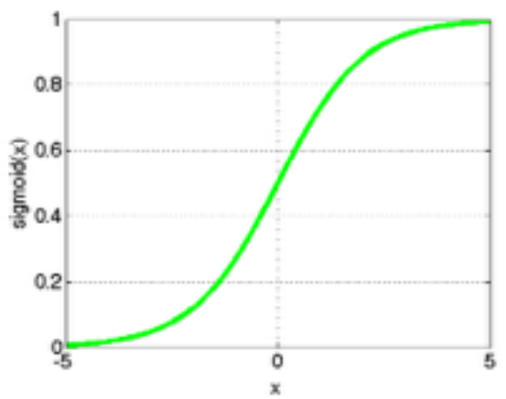


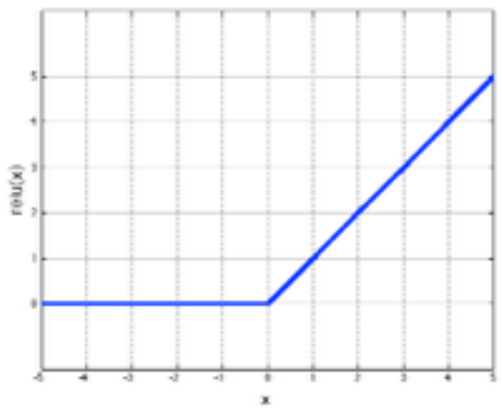
Feature Map

2. Non-Linearity

- Per-element (independent)
- Options:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear unit (ReLU)
    - Simplifies backpropagation
    - Makes learning faster
    - Avoids saturation issues
    → Preferred option

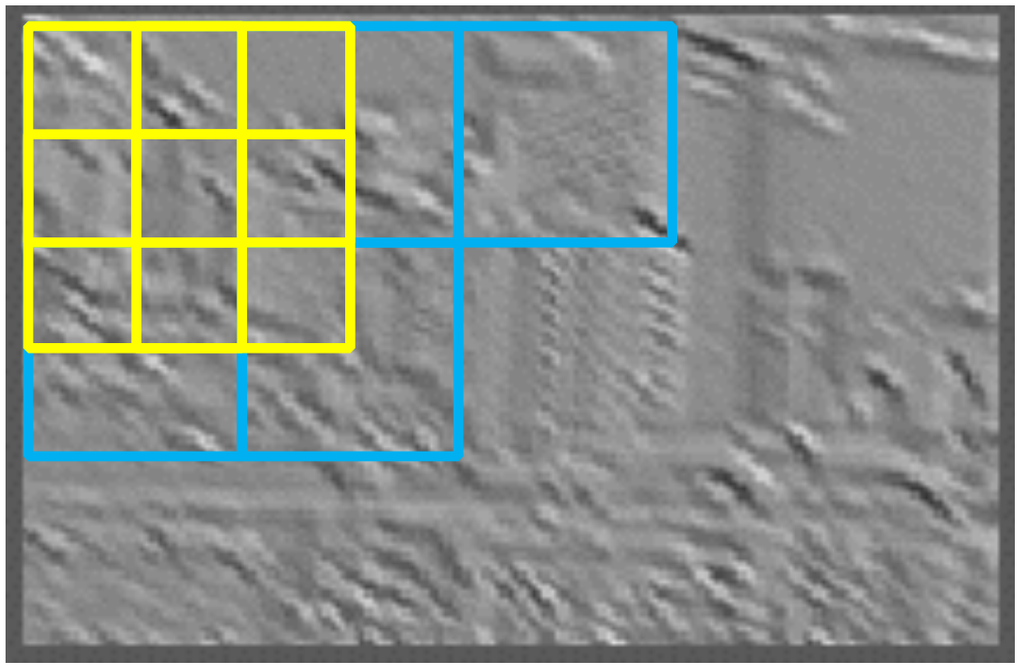




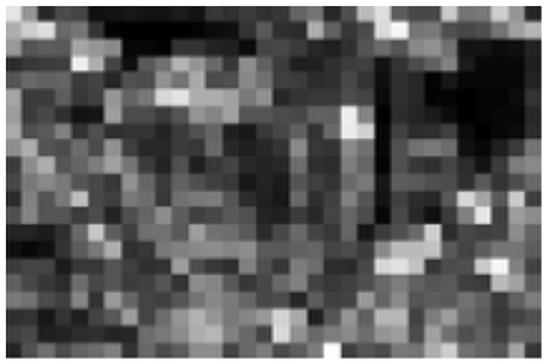


3. Spatial Pooling

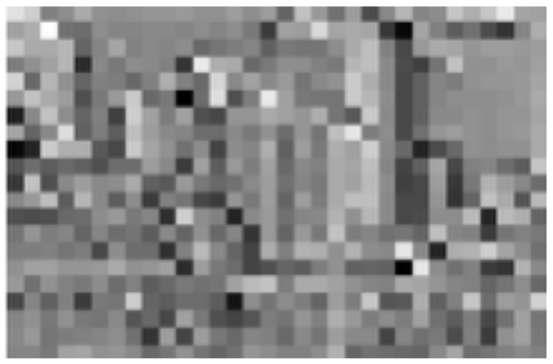
- Sum or max
- Non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (see more of input)



Max

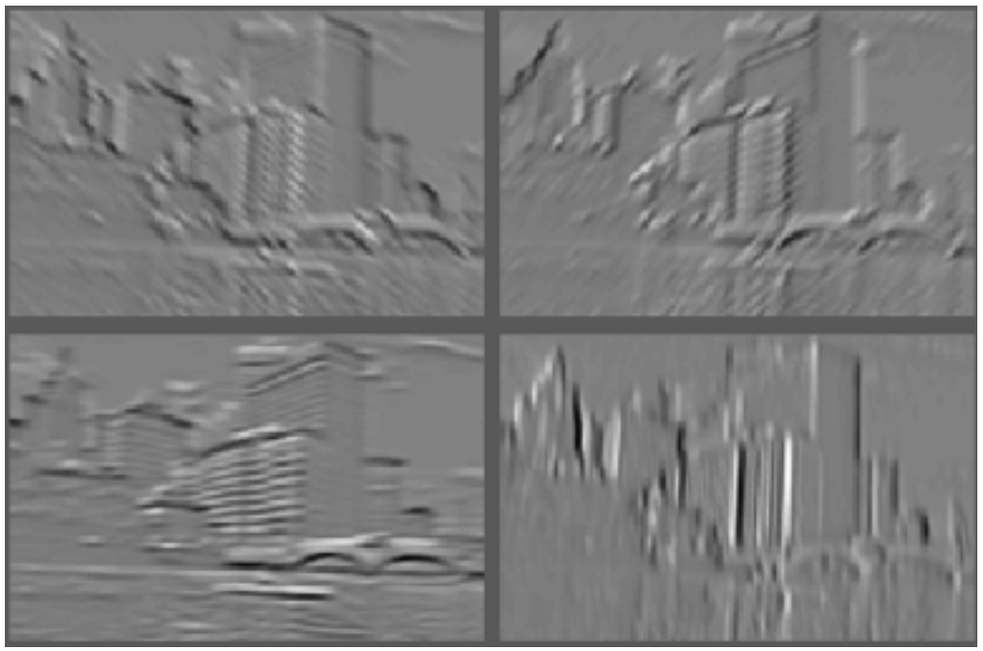


Sum

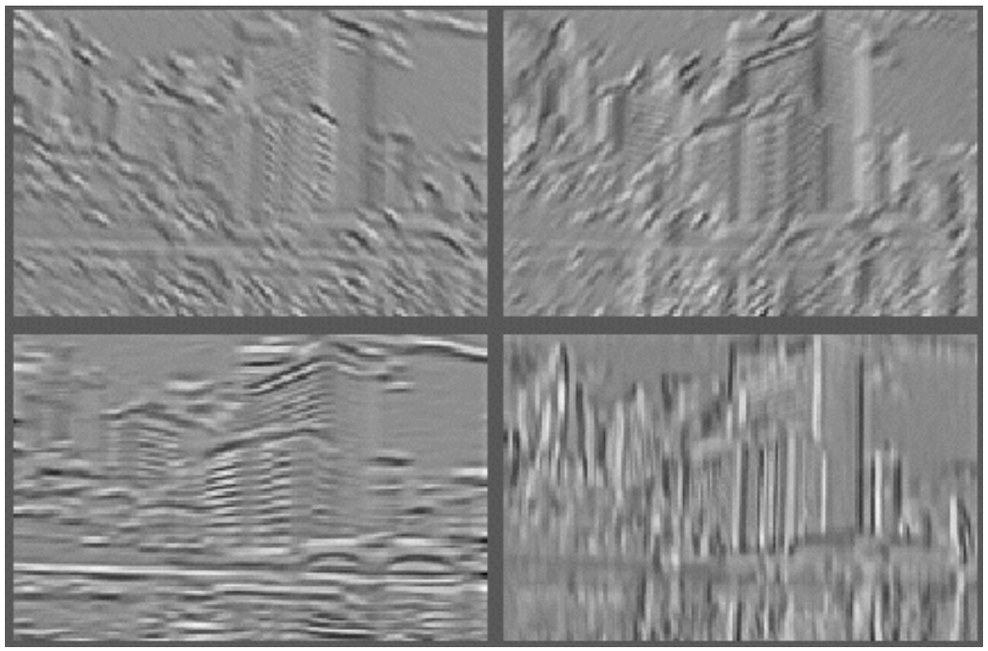


4. Normalization

- Within or across feature maps
- Before or after spatial pooling



Feature Maps



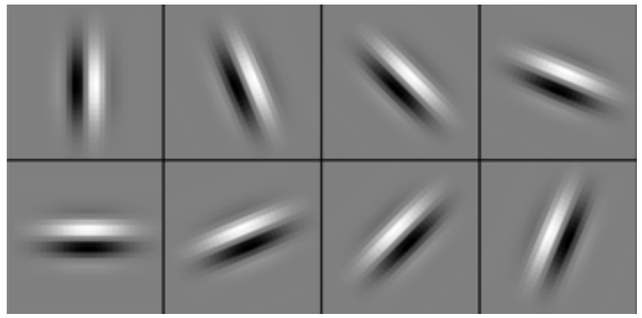
Feature Maps
After Contrast Normalization

Compare: SIFT Descriptor

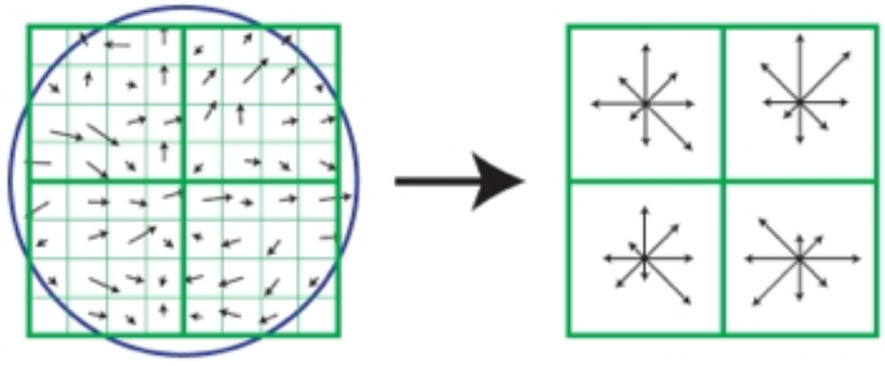
Lowe
[IJCV 2004]

Image Pixels

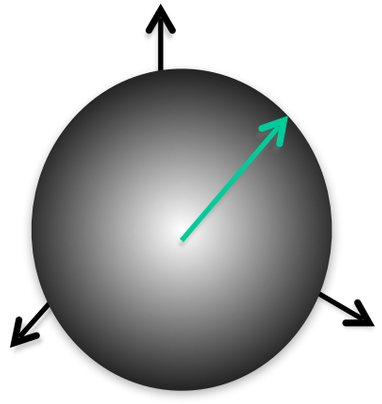
Apply oriented filters



Spatial pool (Sum)



Normalize to unit length

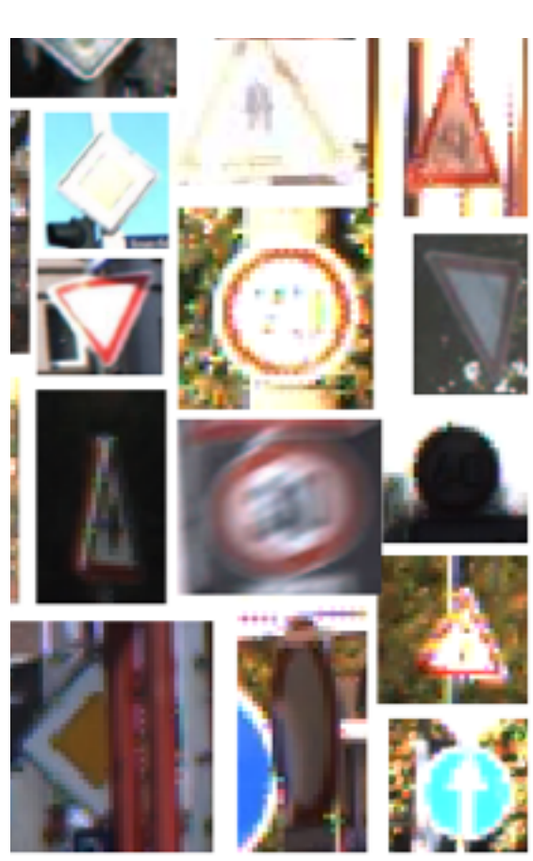


Feature Vector

CNN successes

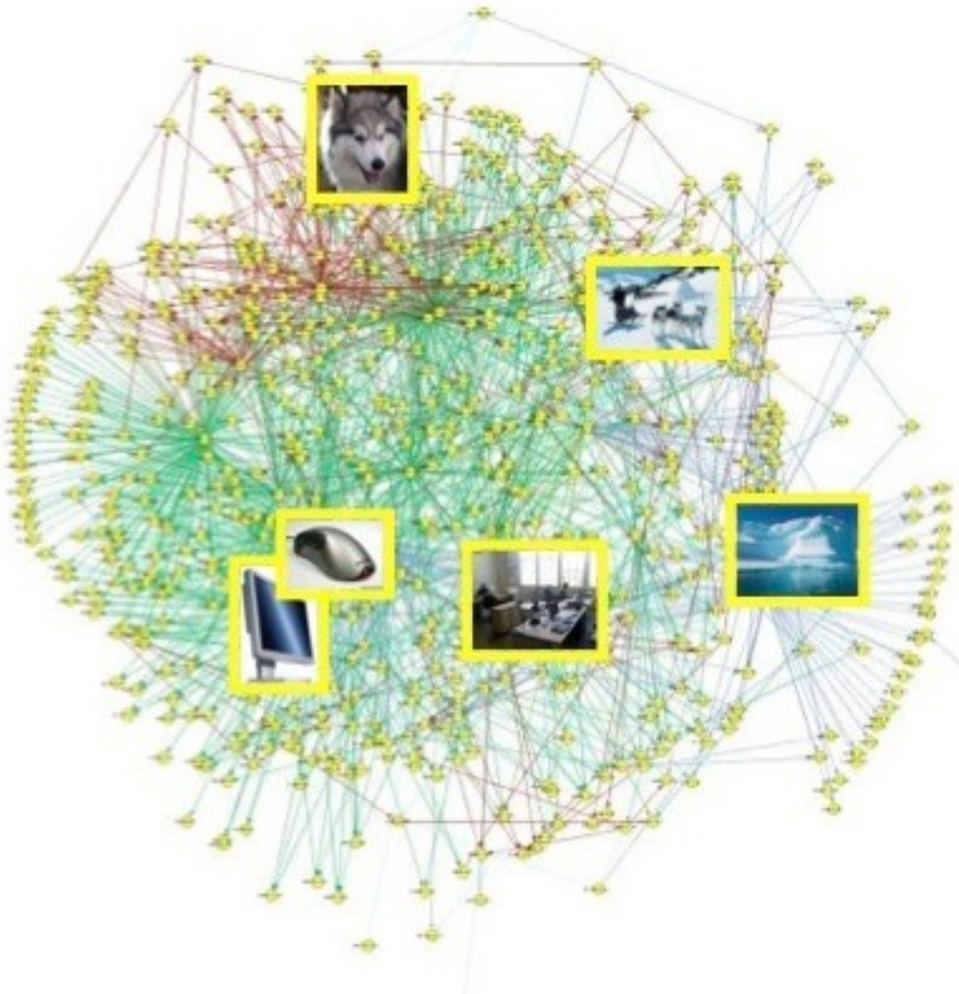
- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]

- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]



- But until recently, less good at more complex datasets
  - Caltech-101/256 (few training examples)

ImageNet Challenge 2012



[Deng et al. CVPR 2009]

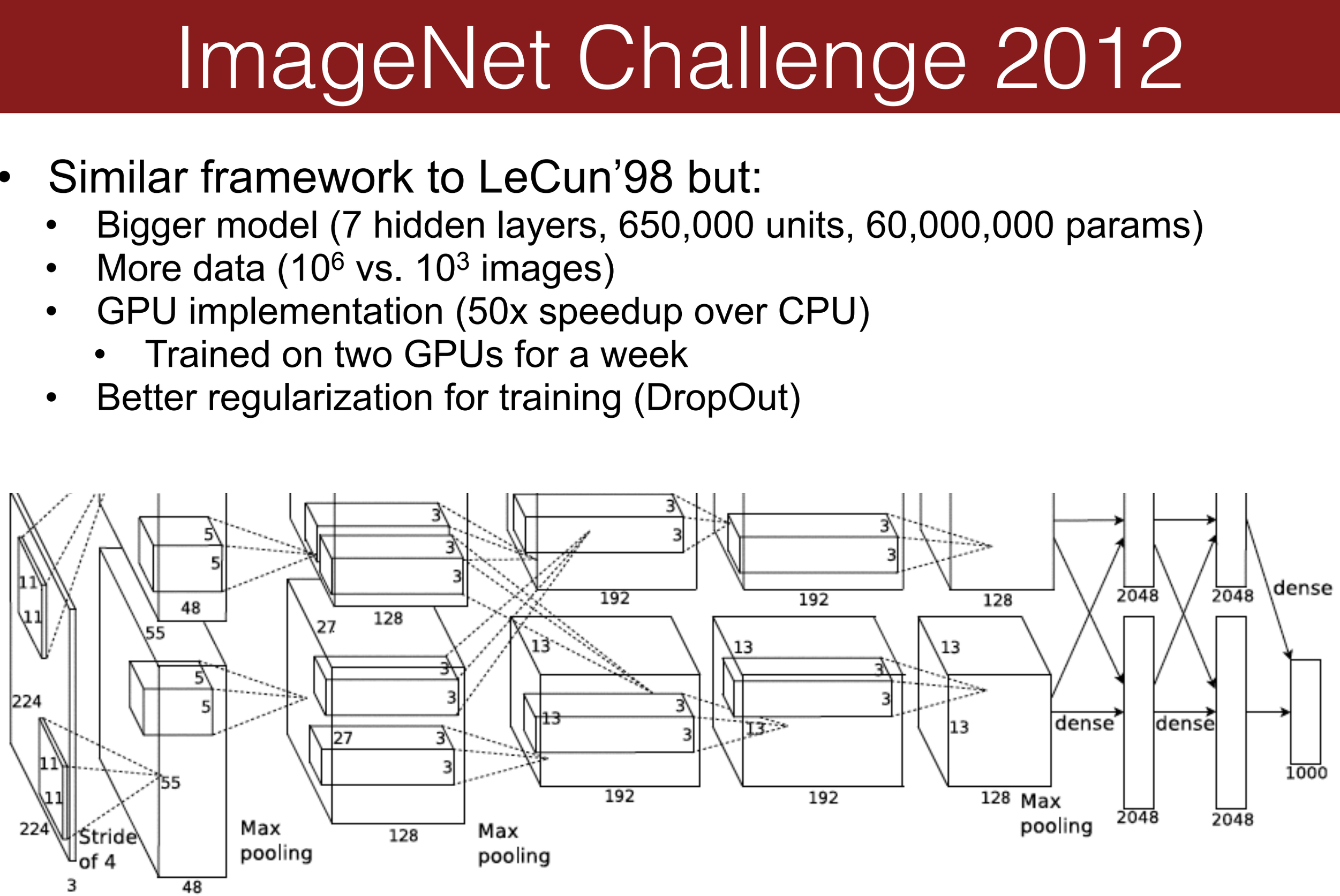
IMAGENET

- 14+ million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- The challenge: 1.2 million training images, 1000 classes

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

ImageNet Challenge 2012

- 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
  - Better regularization for training (DropOut)

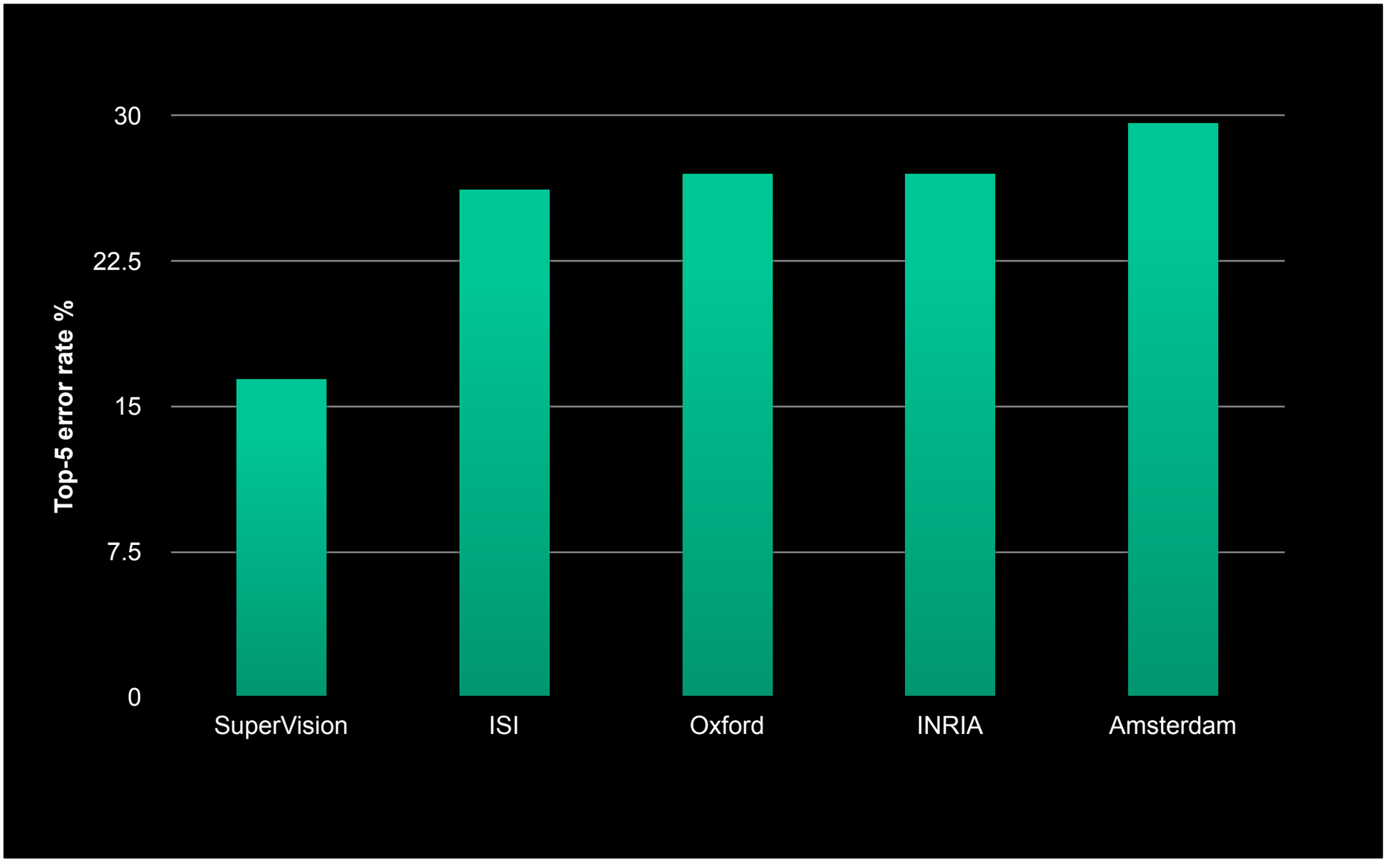


A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

ImageNet Challenge 2012

Krizhevsky et al. -- **16.4% error** (top-5)

Next best (SIFT + Fisher vectors) – **26.2% error**



Visualizing CNNs

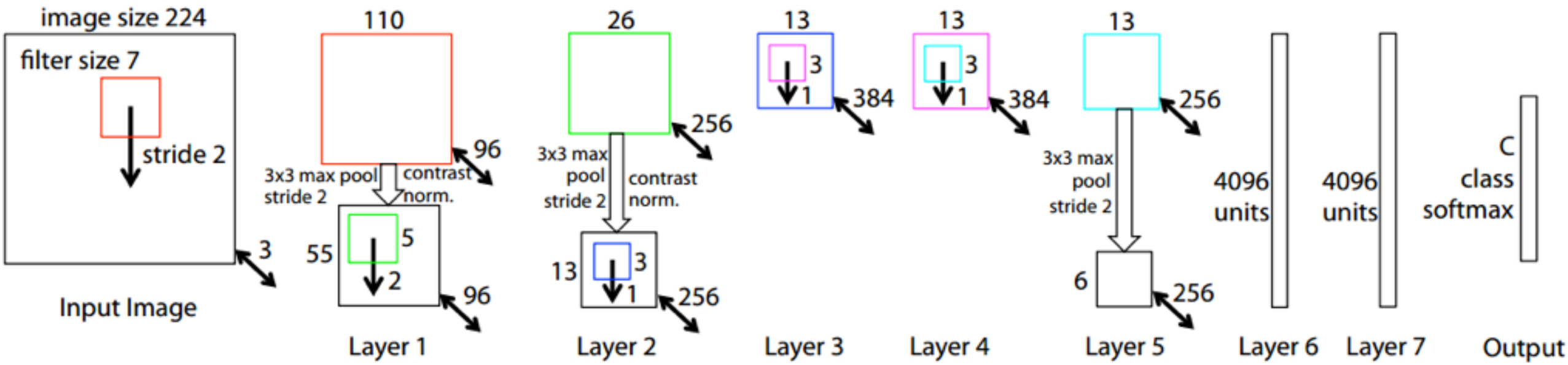
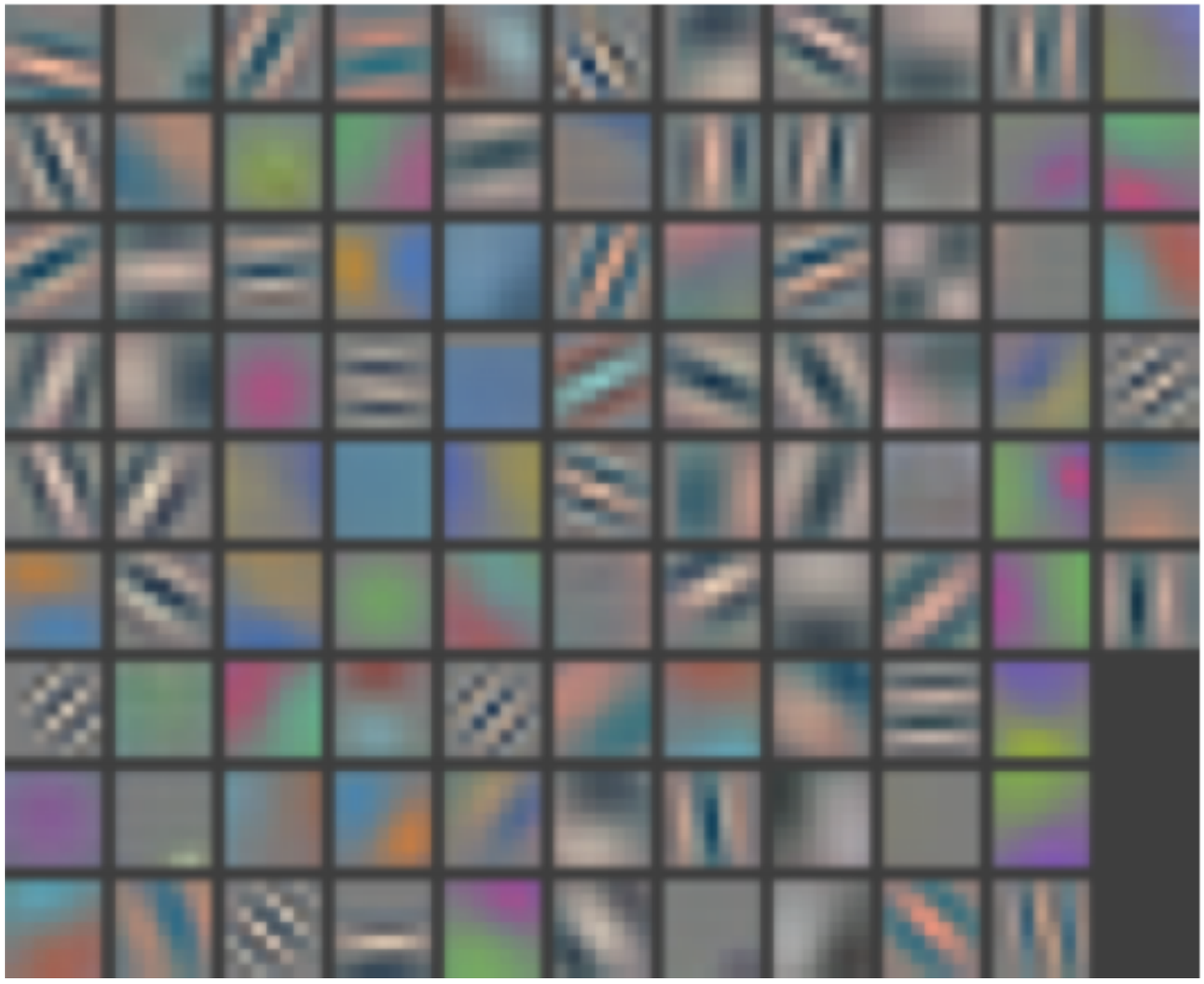


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

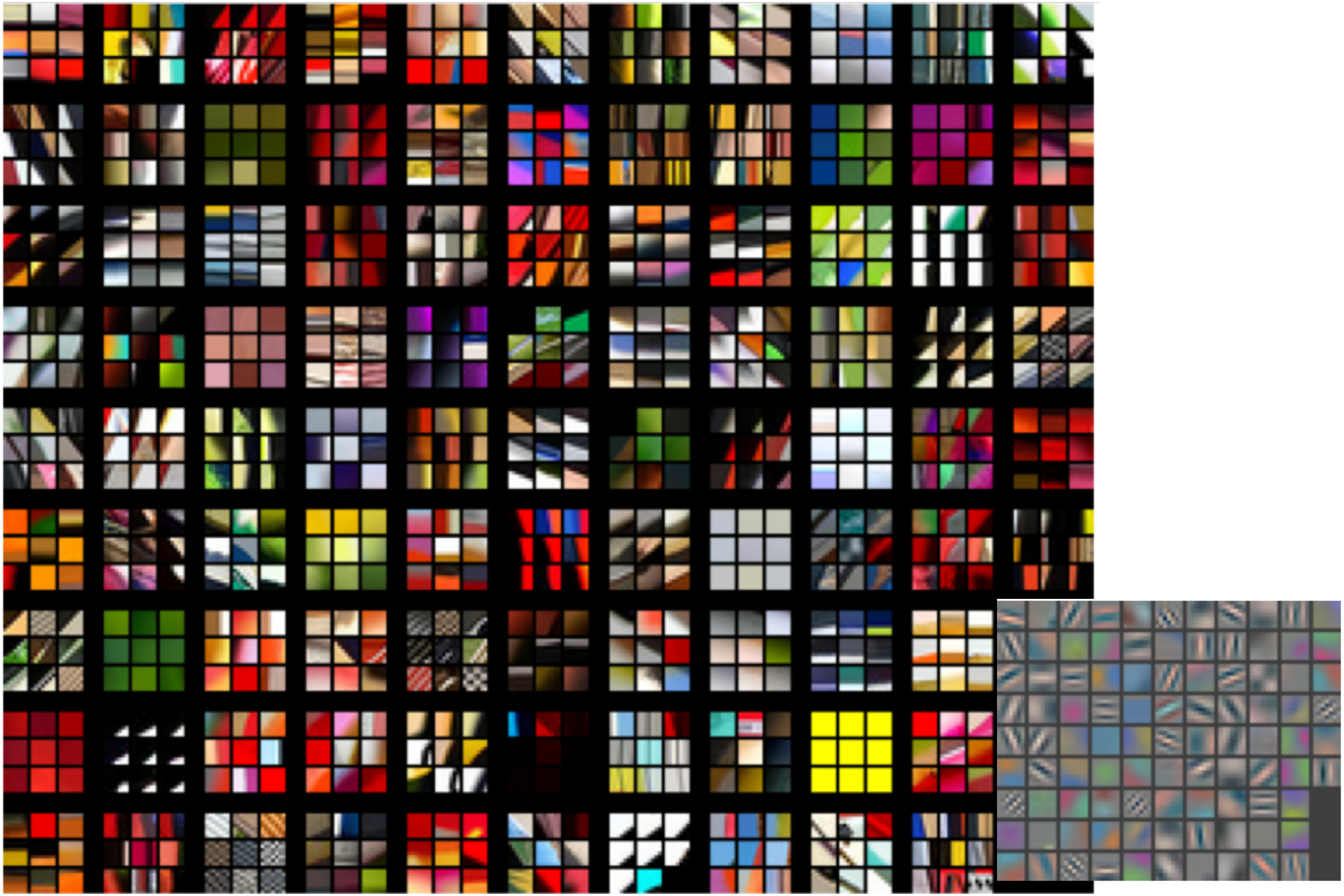
M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, arXiv preprint, 2013

Layer 1 Filters



Similar to the filter banks used for texture recognition

Layer 1: Top-9 Patches



Layer 2: Top-9 Patches

- **Patches from validation images that give maximal activation of a given feature map**

Layer 2: Top-9 Patches

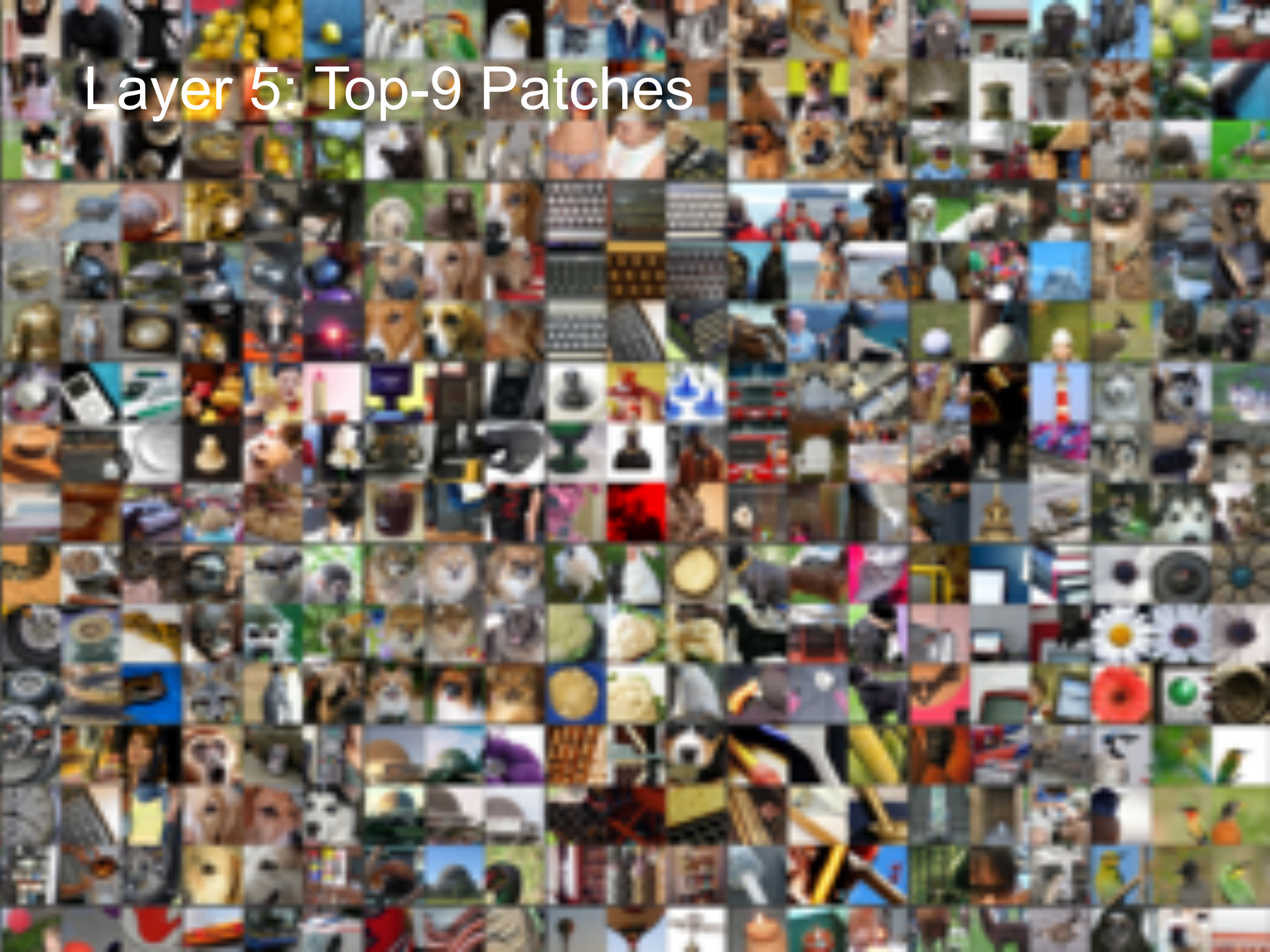
Layer 3: Top-9 Patches

Layer 3: Top-9 Patches

Layer 4: Top-9 Patches

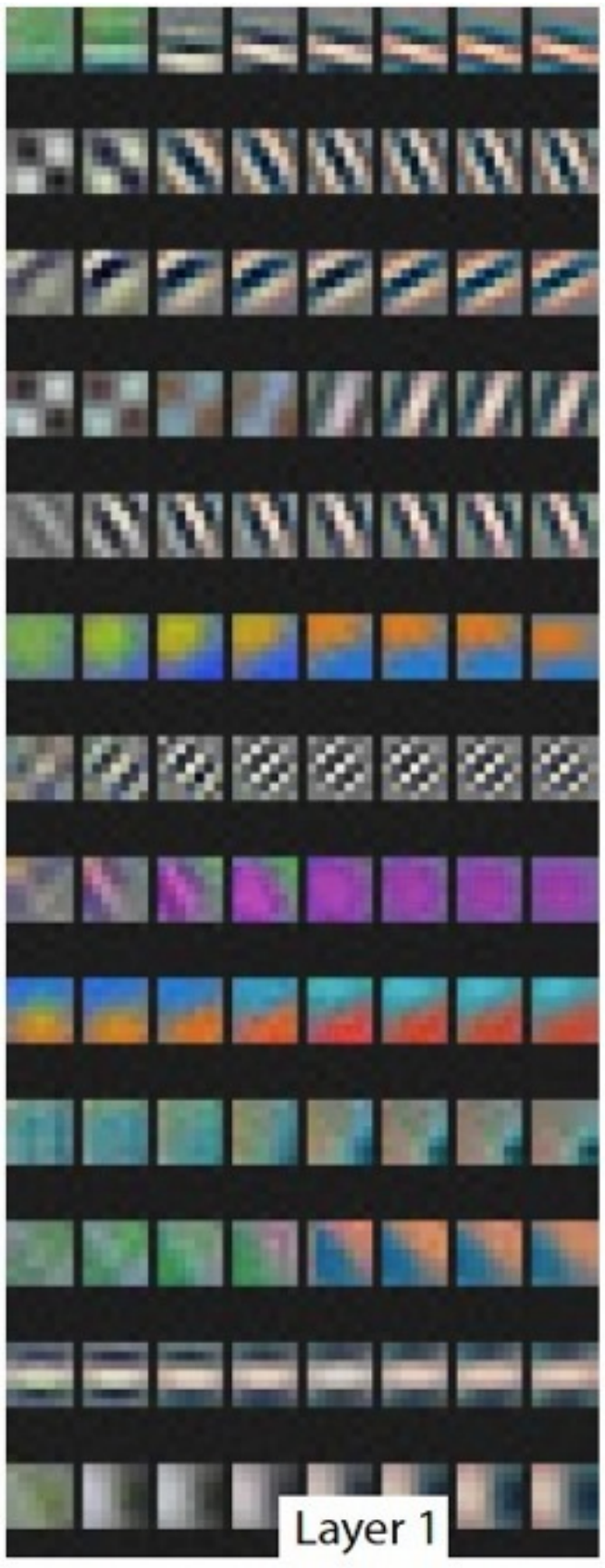
Layer 4: Top-9 Patches

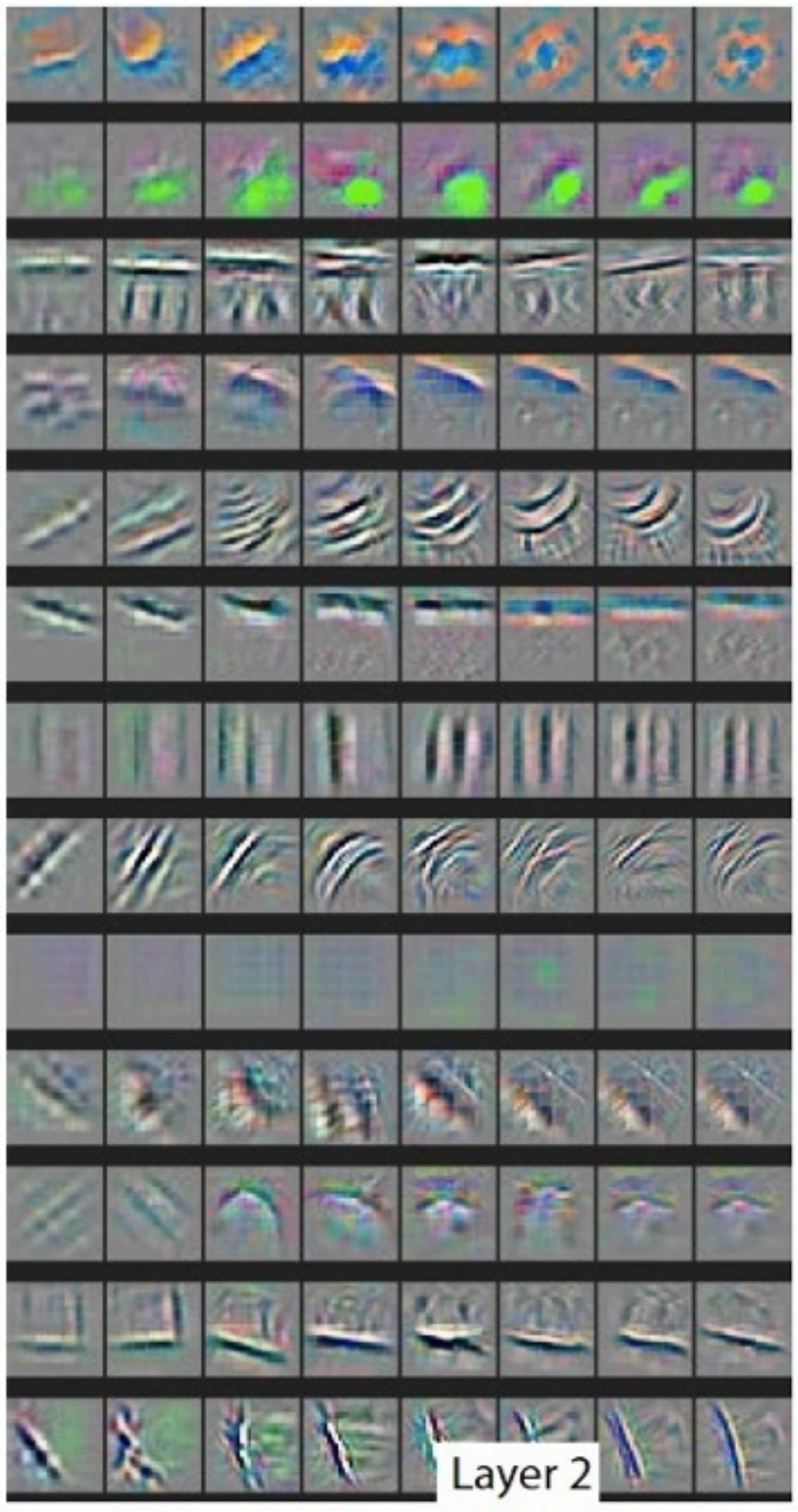
Layer 5: Top-9 Patches

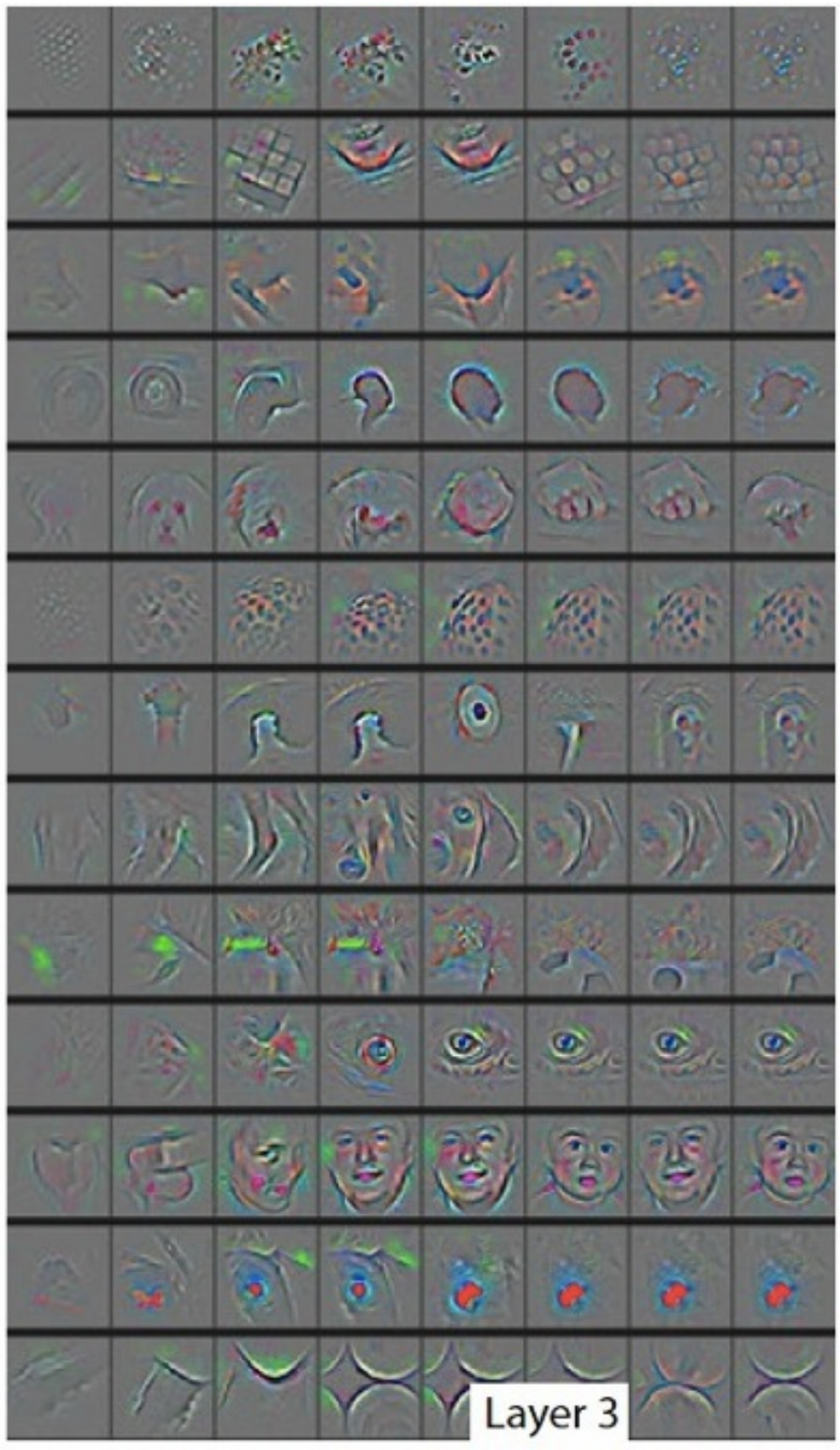


Layer 5: Top-9 Patches

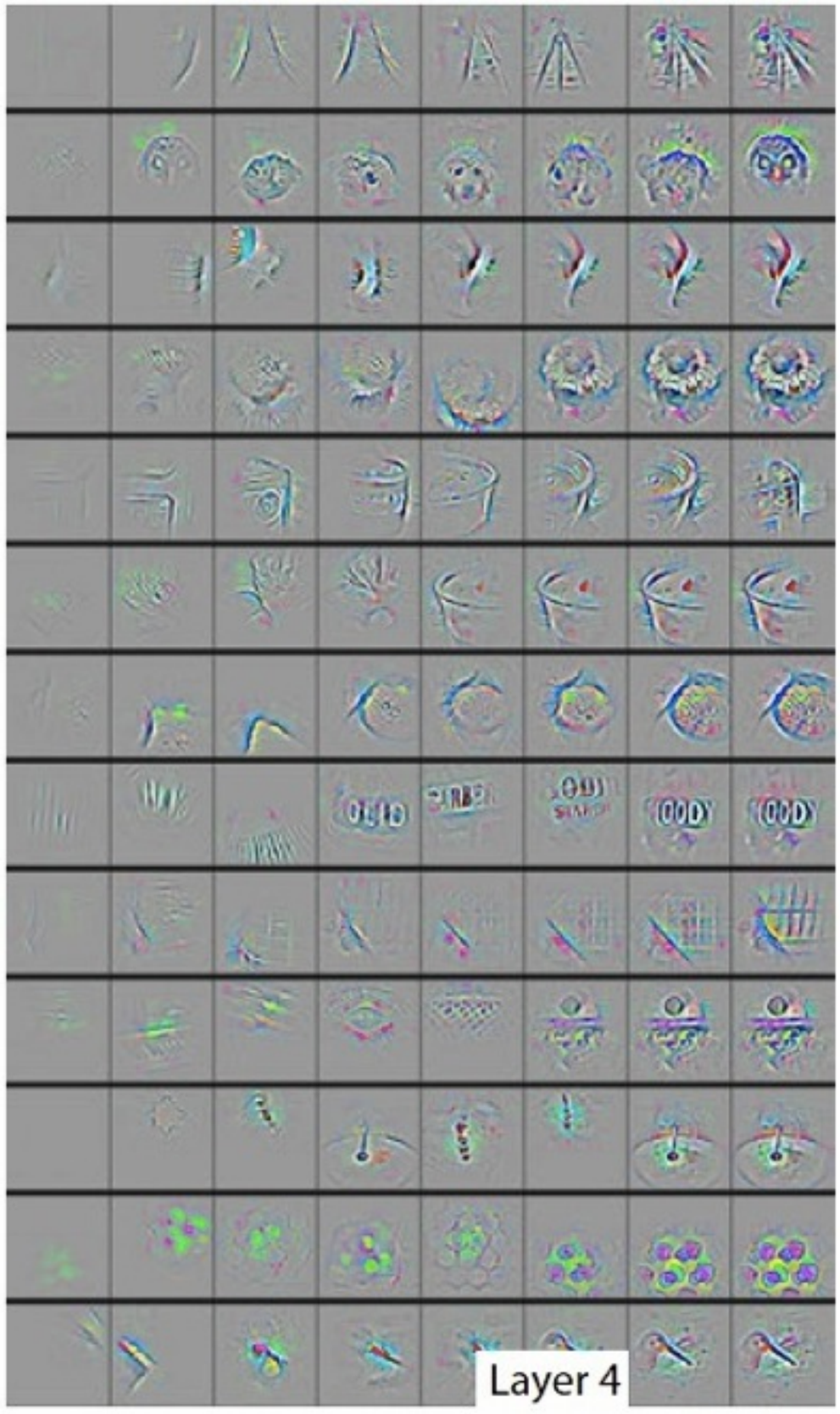
Evolution of Features During Training

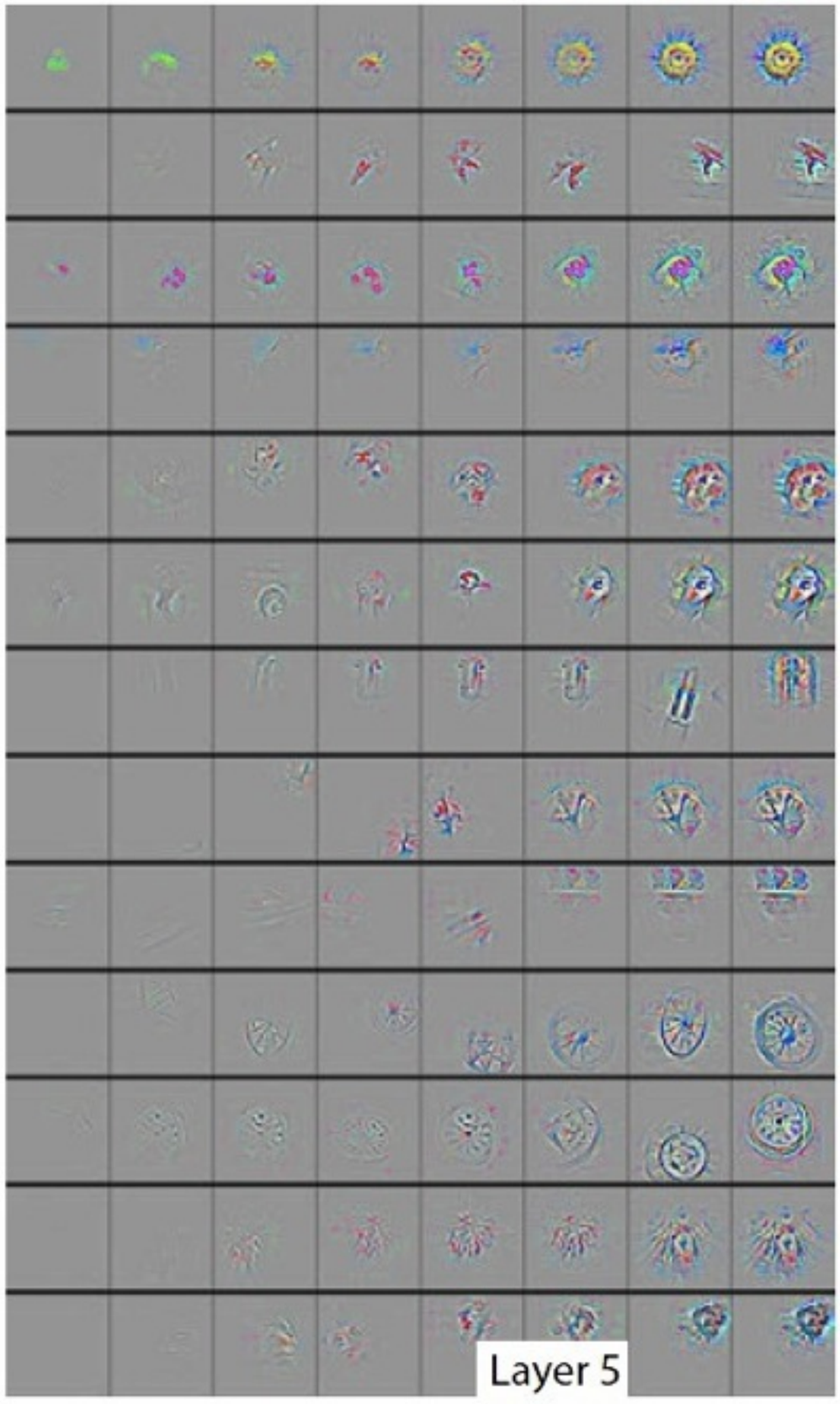






Evolution of Features During Training



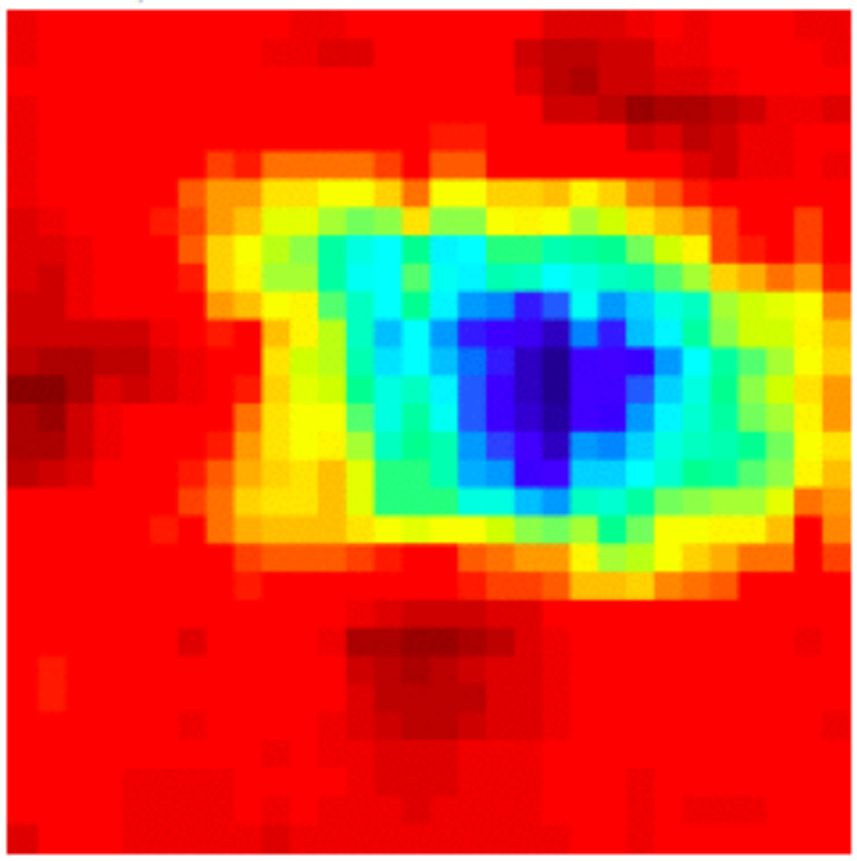


Occlusion Experiment

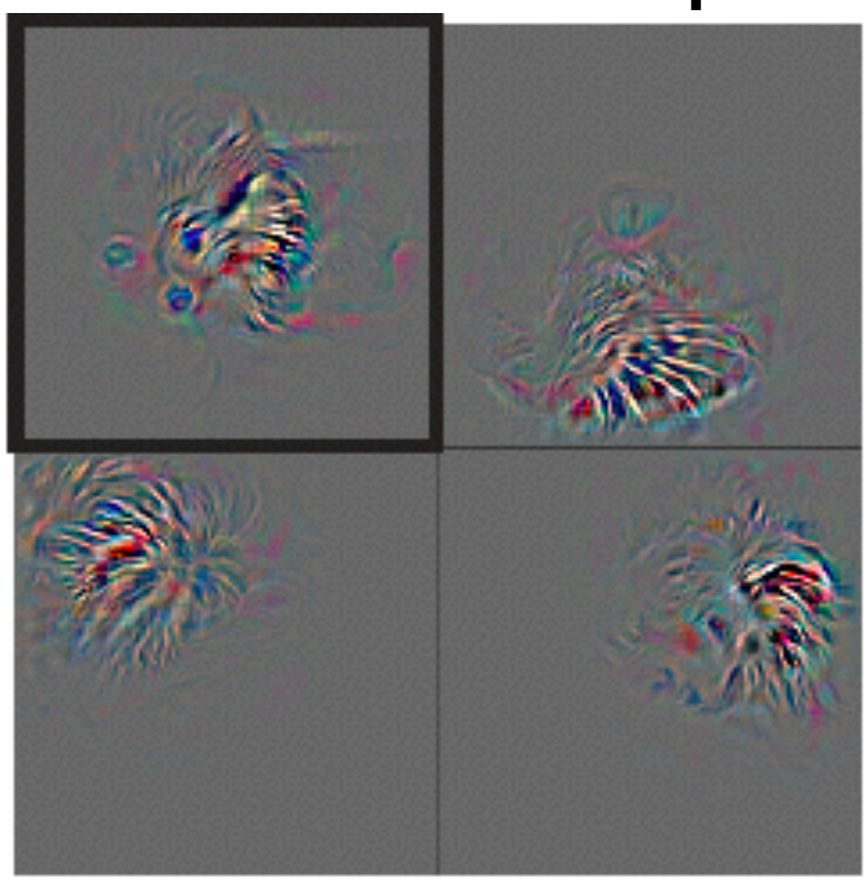
- Mask parts of input with occluding square

- Monitor output (class probability)

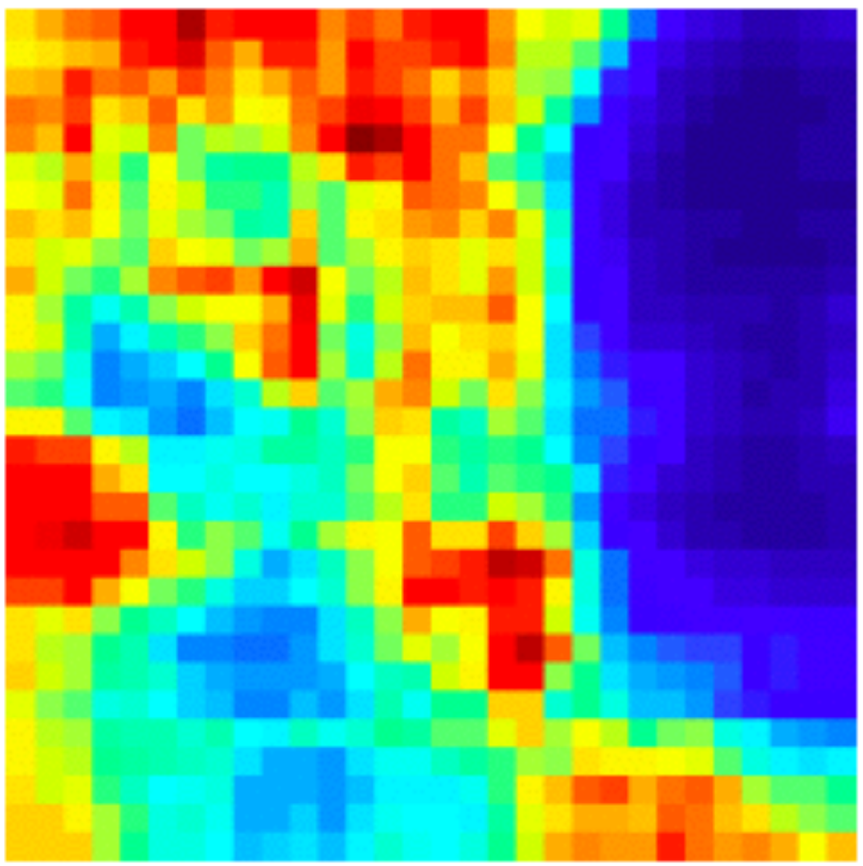
Total activation in most active 5th layer feature map

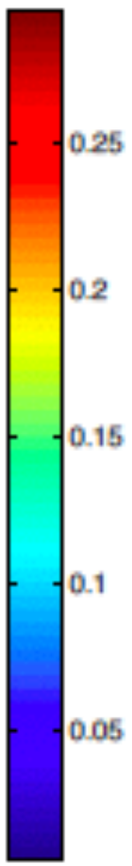


Other activations from same feature map

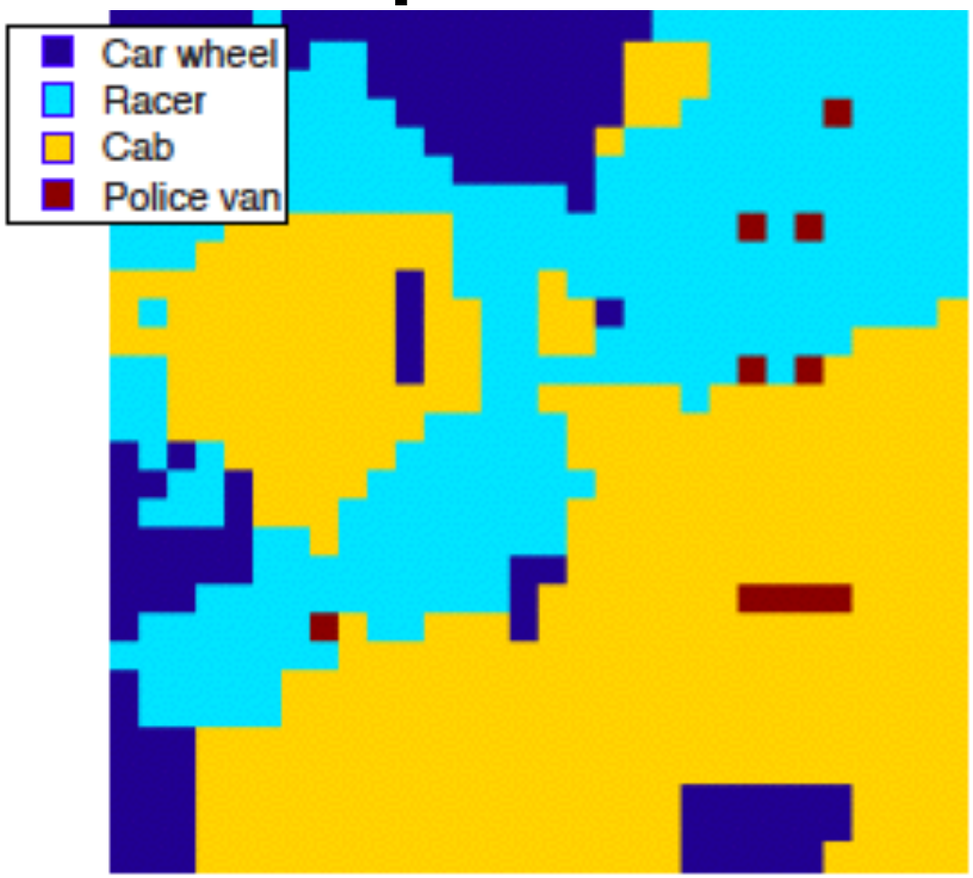


p(True class)

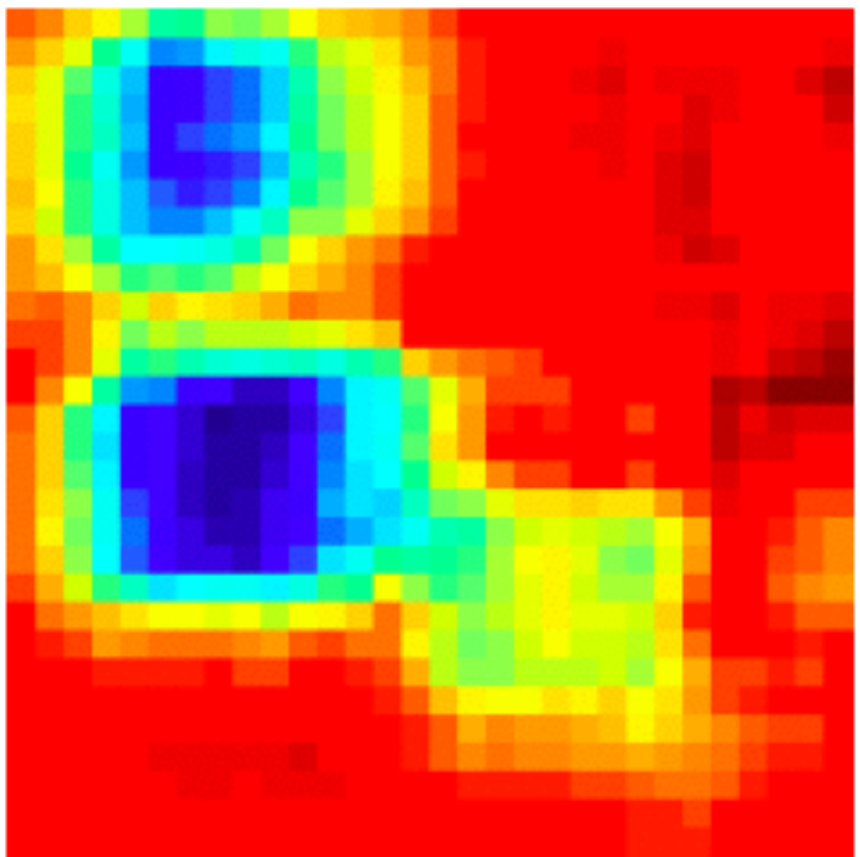




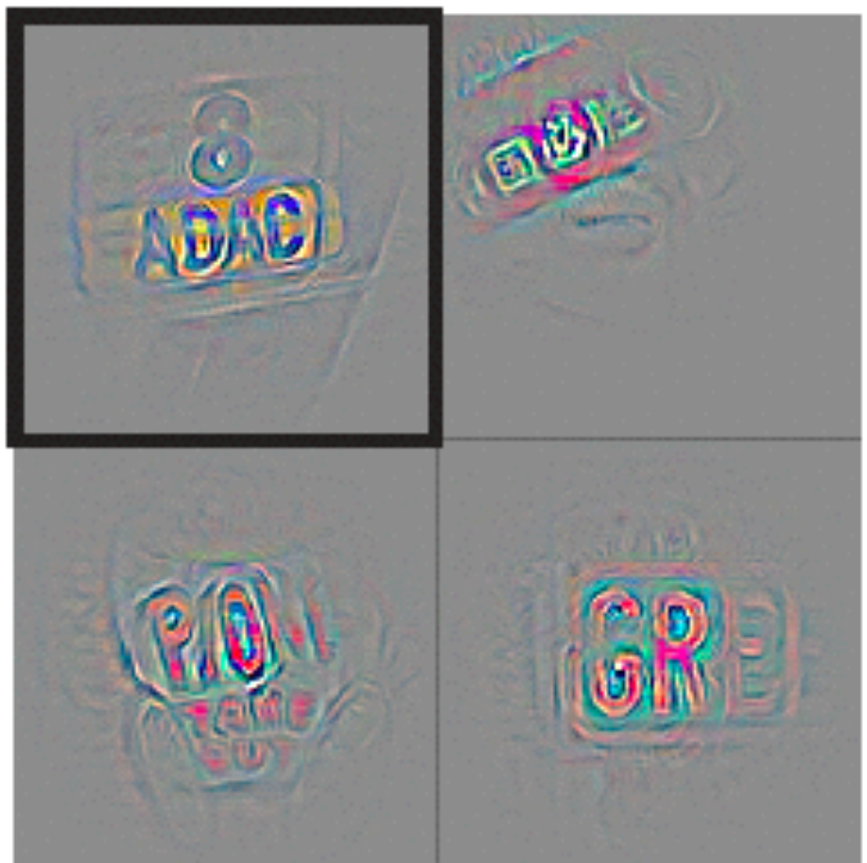
Most probable class

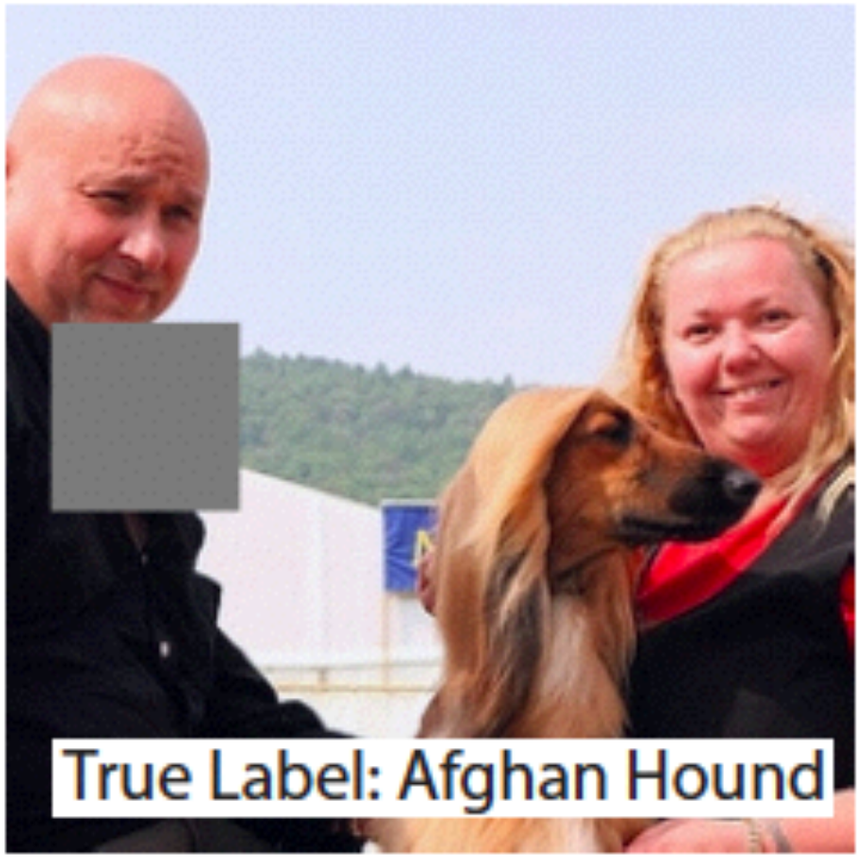


Total activation in most active 5th layer feature map

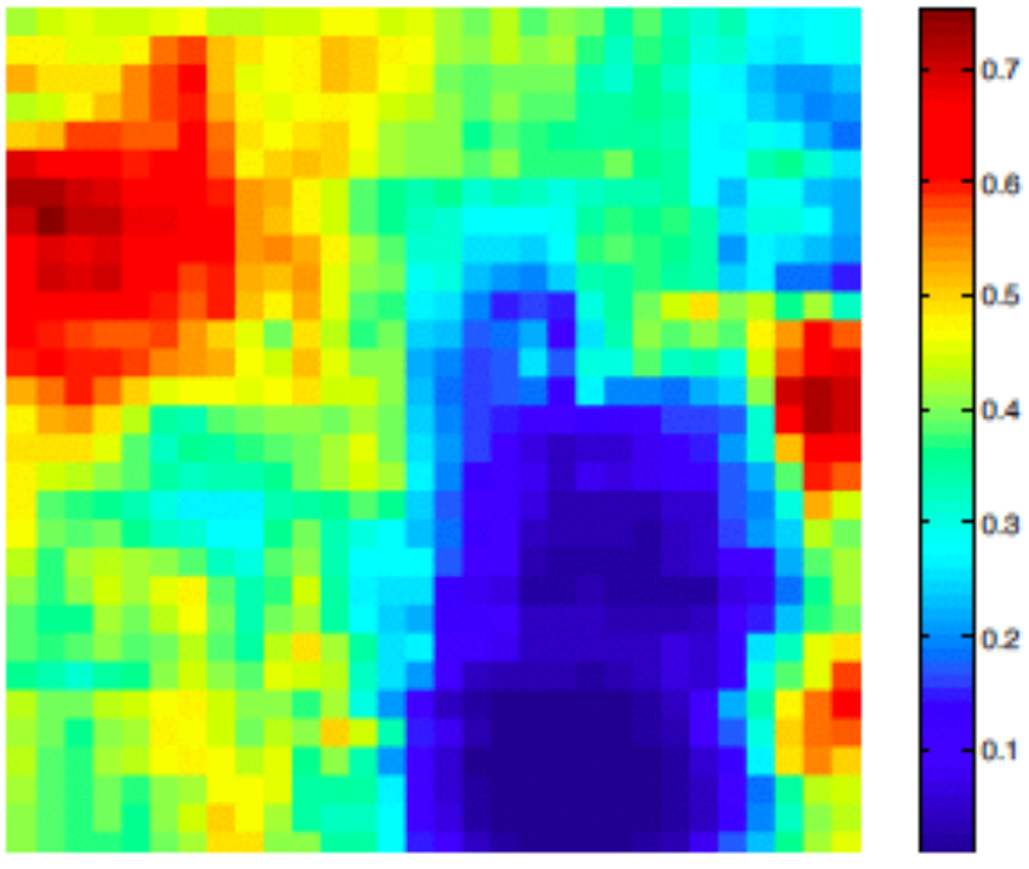


Other activations from same feature map

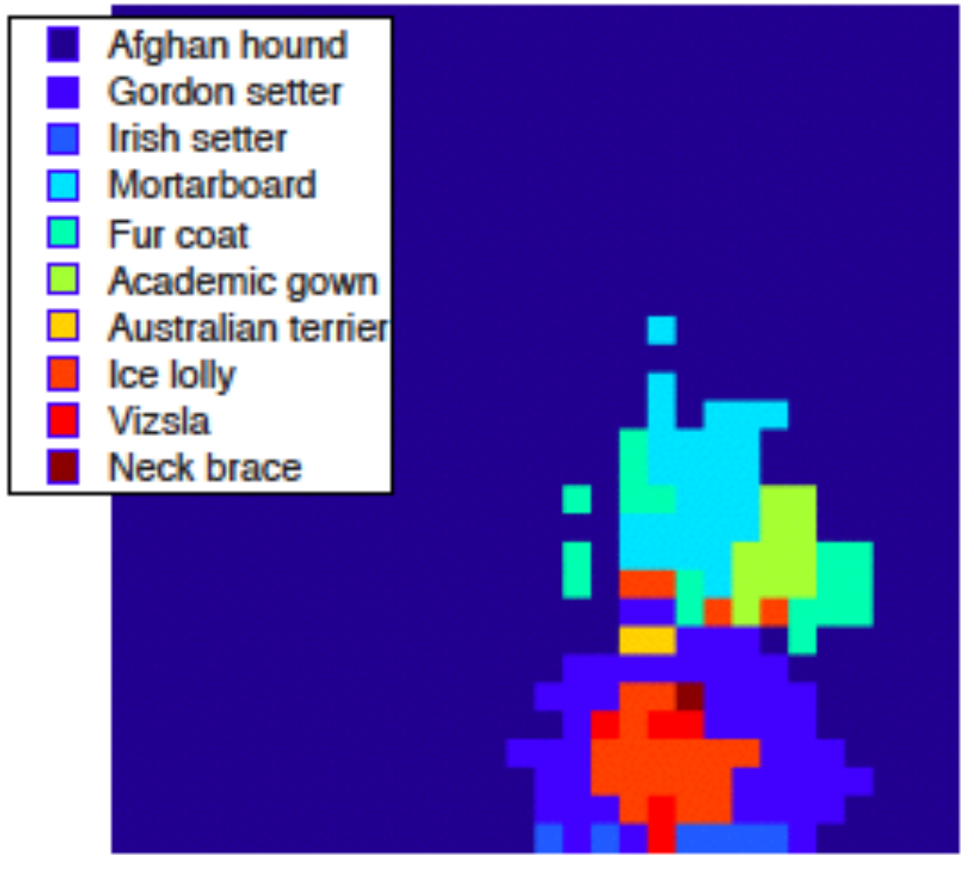




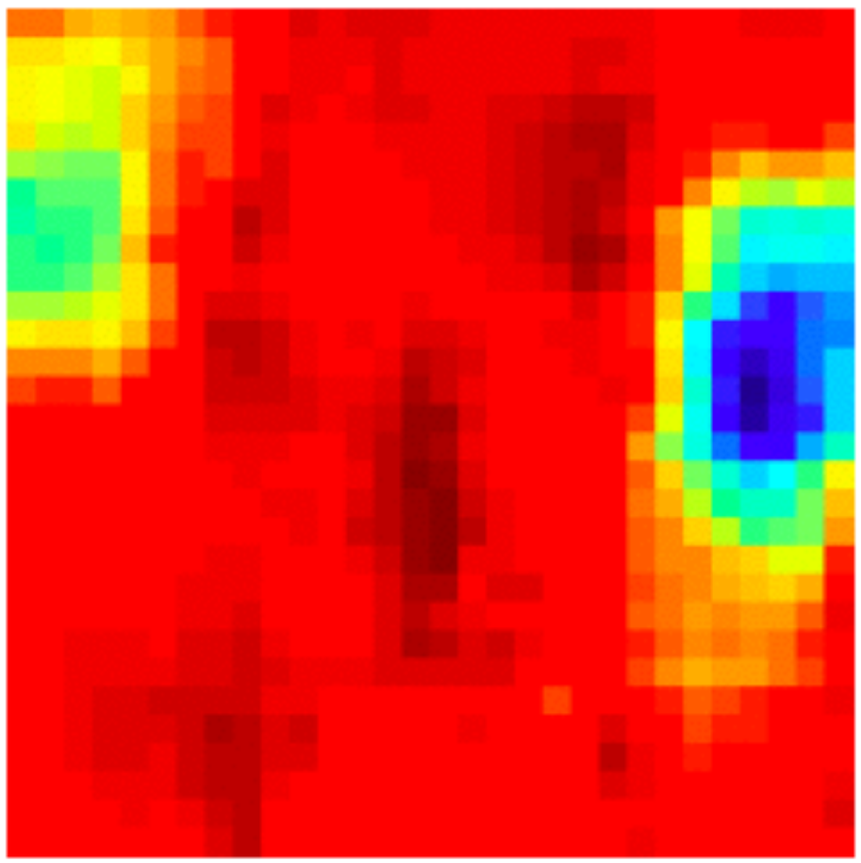
p(True class)



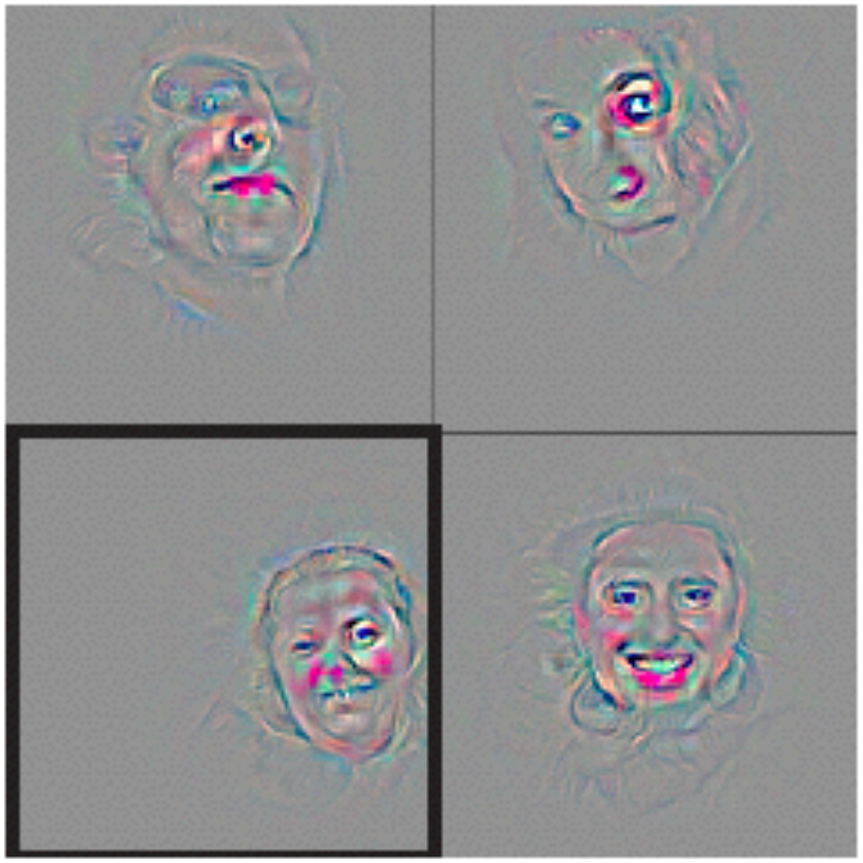
Most probable class



Total activation in most active 5th layer feature map

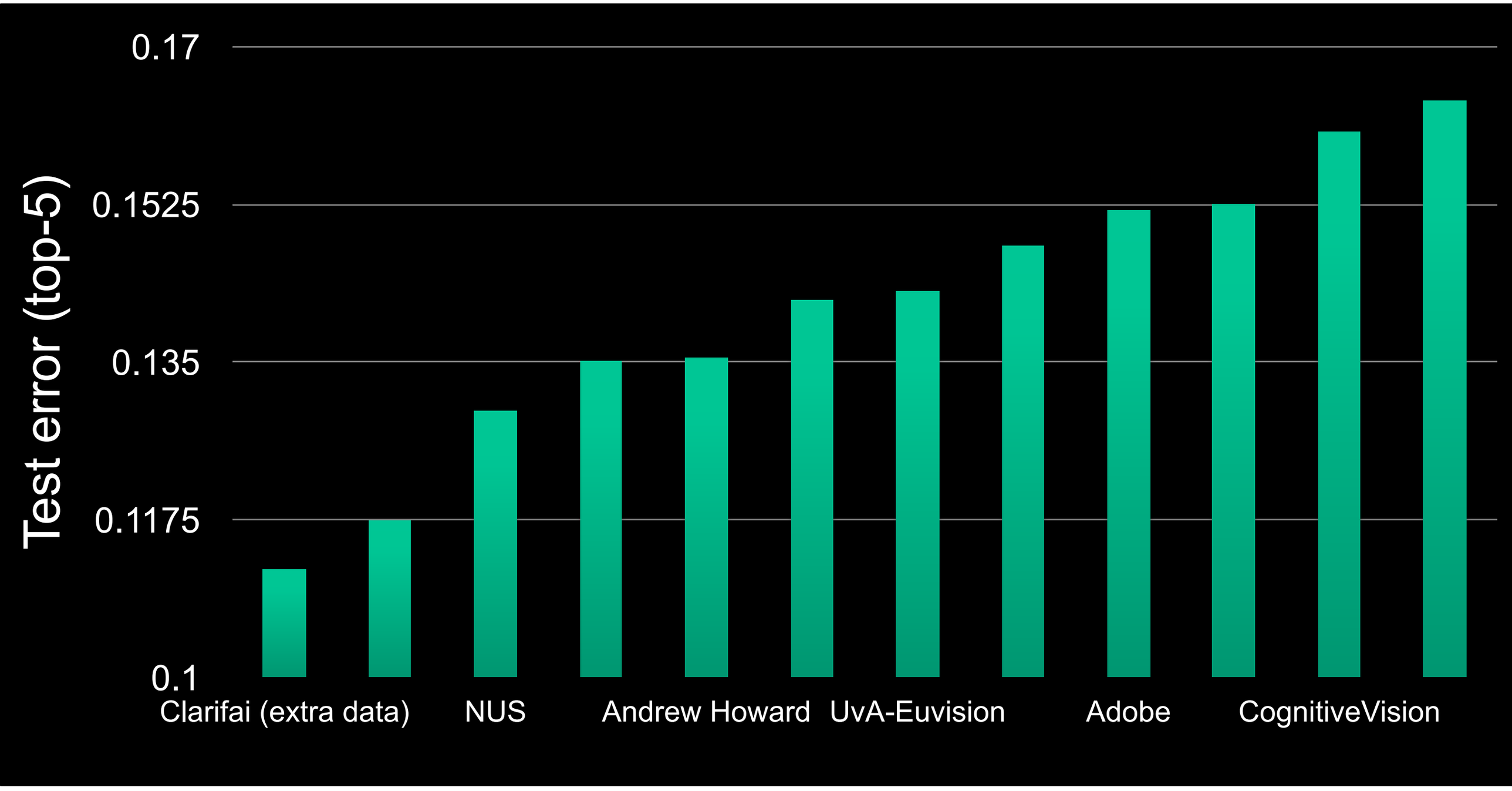


Other activations from same feature map



ImageNet Classification 2013 Results

http://www.image-net.org/challenges/LSVRC/2013/results.php



ImageNet 2014 - Test error at 0.07 (Google & Oxford groups)

http://image-net.org/challenges/LSVRC/2014/results

CNNs for small datasets

- Take model trained on ImageNet
- Take outputs of 6^{th} or 7^{th} layer before or after nonlinearity as features
- Train linear SVMs on these features (like retraining the last layer of the network)
- Optionally back-propagate: fine-tune features and/or classifier on new dataset

Tapping off features at each Layer

Plug features from each layer into linear SVM

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$

Higher layers are better

Results on benchmarks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

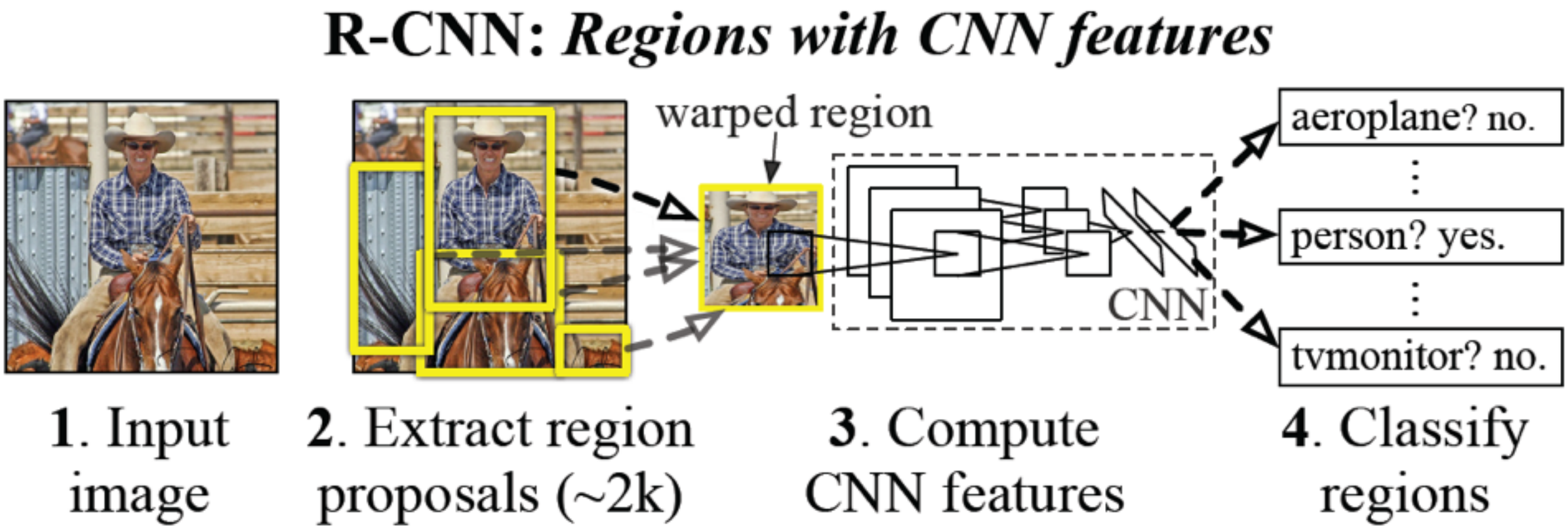
[2] MIT-67 Indoor Scenes dataset (OverFeat)

Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, **DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition**, arXiv preprint, 2014

[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, **CNN Features off-the-shelf: an Astounding Baseline for Recognition**, arXiv preprint, 2014

CNN features for detection



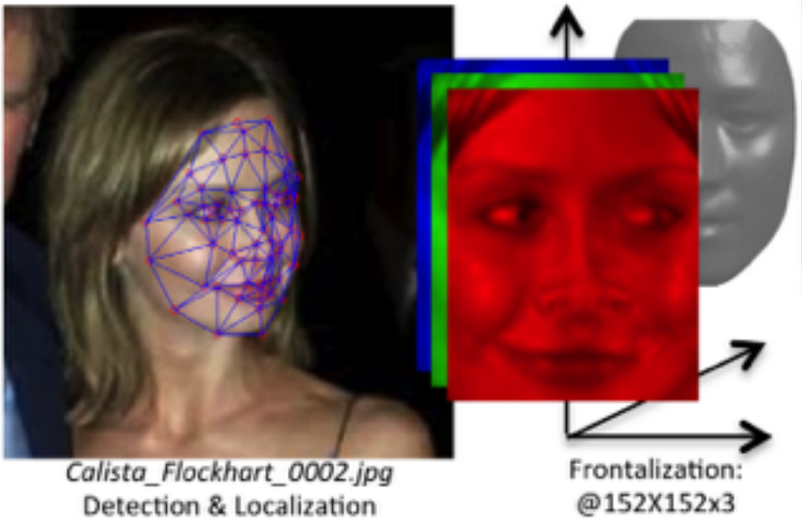
R-CNN achieves mAP of **53.7%** on PASCAL VOC 2010

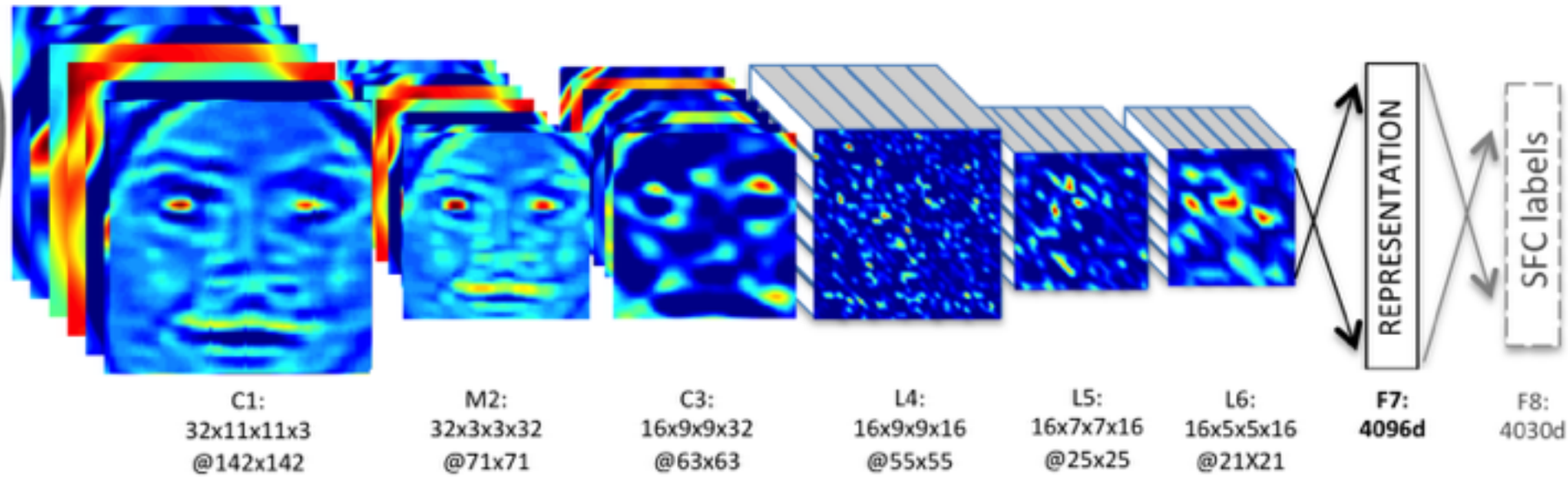
For comparison, **Uijlings et al. (2013)** report **35.1%** mAP using the *same region proposals*, but with a spatial pyramid and bag-of-visual-words approach.

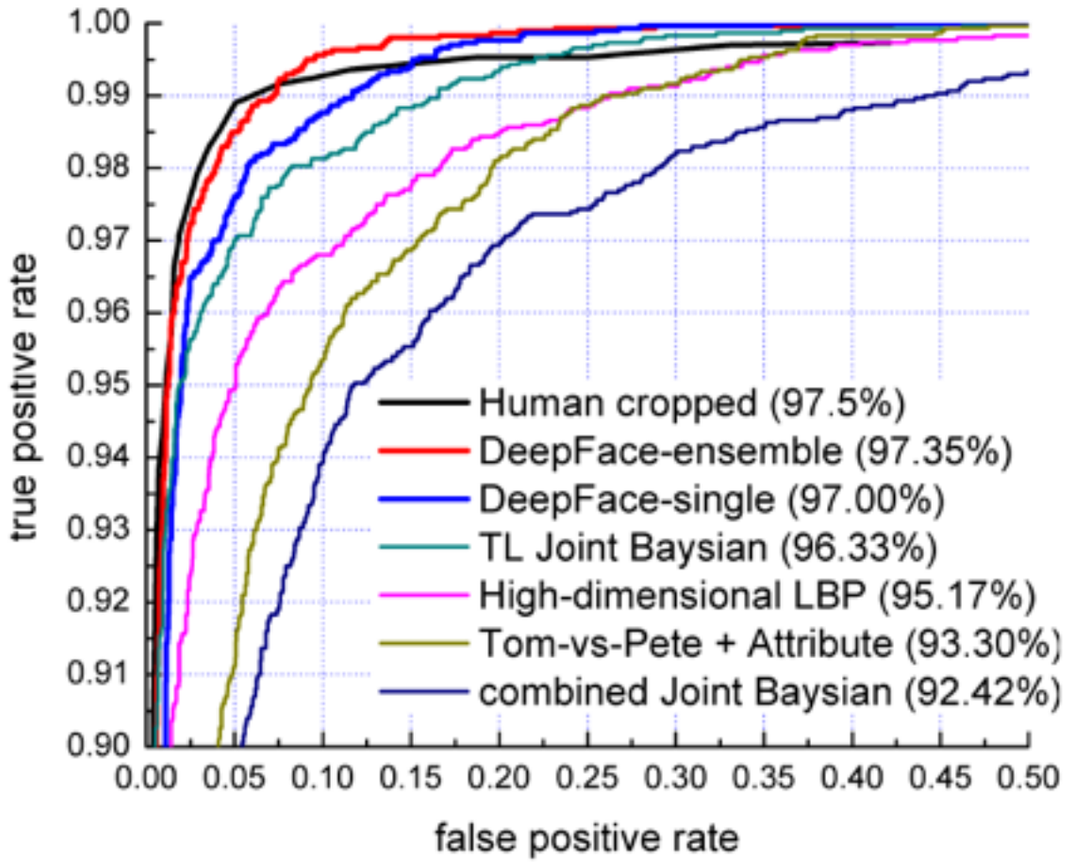
Part-based model with HOG (DPM, Poselets) ~ **33.5%**

R. Girshick, J. Donahue, T. Darrell, and J. Malik, **Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation**, CVPR 2014

CNN features for face verification







Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR 2014, to appear.

Open-source CNN software

- **Cuda-convnet** (Alex Krizhevsky, Google)
  - High speed convolutions on the GPU
- **Caffe** (Y. Jia, Berkeley)
  - Replacement of deprecated **Decaf**
  - High performance CNNs
  - Flexible CPU/GPU computations
- **Overfeat** (NYU)
- **MatConvNet** (Andrea Vedaldi, Oxford)
  - An easy to use toolbox for CNNs from MATLAB
  - Comparable performance/features with Caffe