Deep Neural Networks:

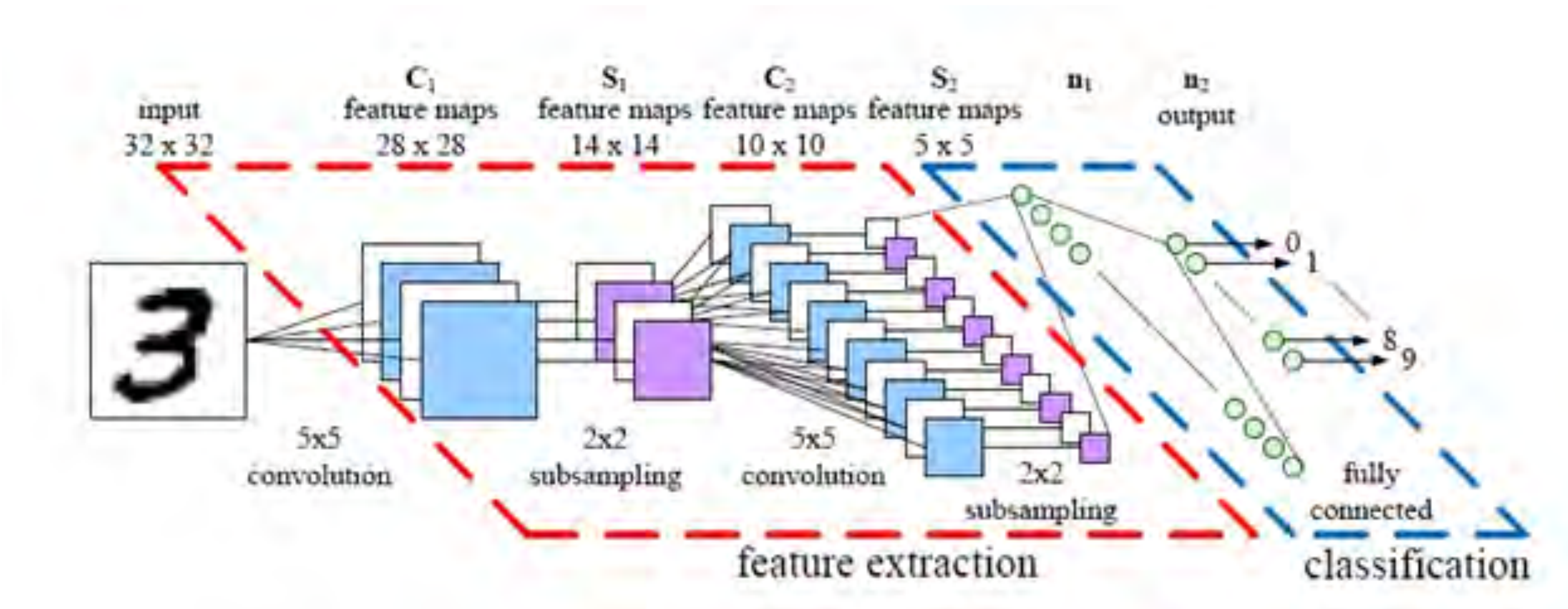
Part II Convolutional Neural Network (CNN)

Yuan-Kai Wang, 2016

Web site of this course: http://pattern-recognition.weebly.com

source: CNN for Image Classification, by S. Lazebnik, 2014.

A Convolutional Neural Network(CNN) for Image Classification



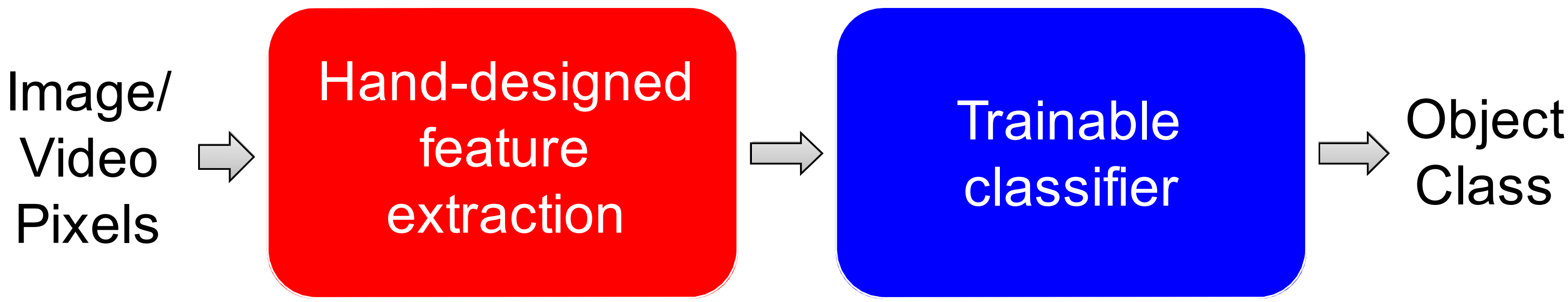
It is deep: 6 hidden layers, each with a special purpose

Deep Neural Network

Overview

- Shallow vs. deep architectures
- Background
- Stages of CNN architecture
- Visualizing CNNs
- State-of-the-art results
- Packages

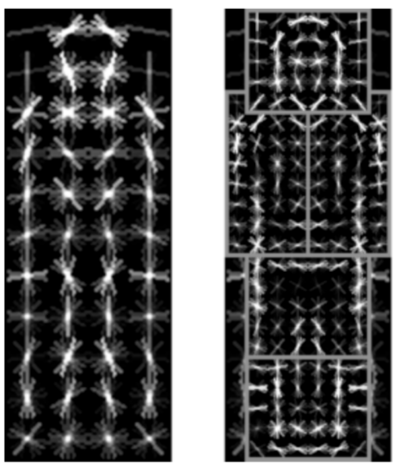
Traditional Recognition Approach



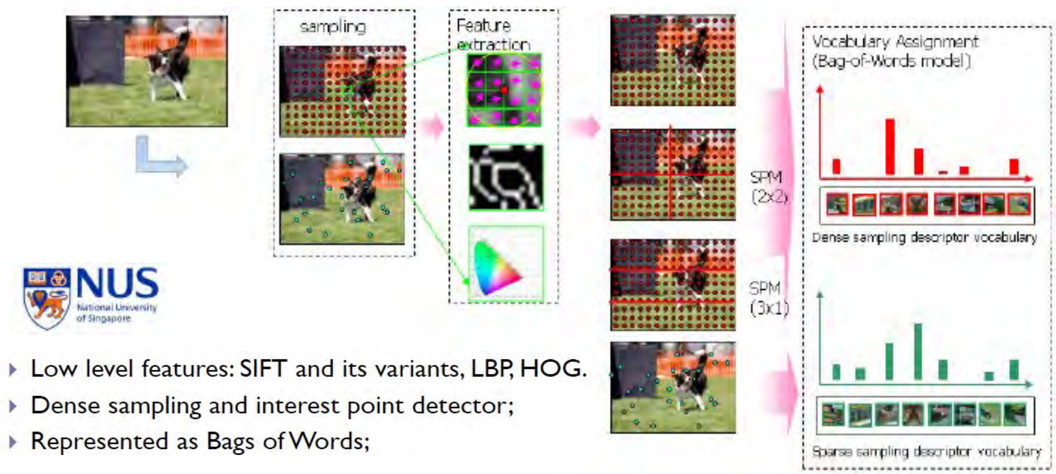
- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

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



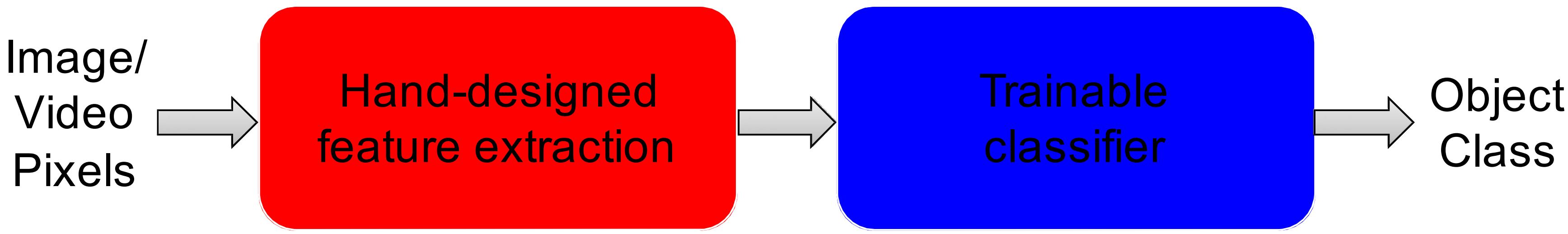
Yan & Huang
(Winner of PASCAL 2010 classification competition)

What about learning the features?

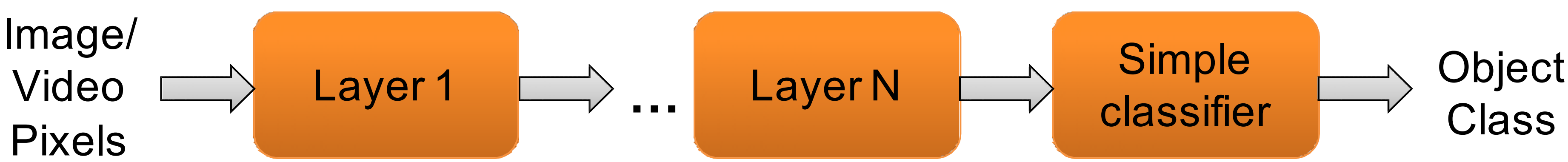
- Learn a *feature hierarchy* all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

“Shallow” vs. “deep” architectures

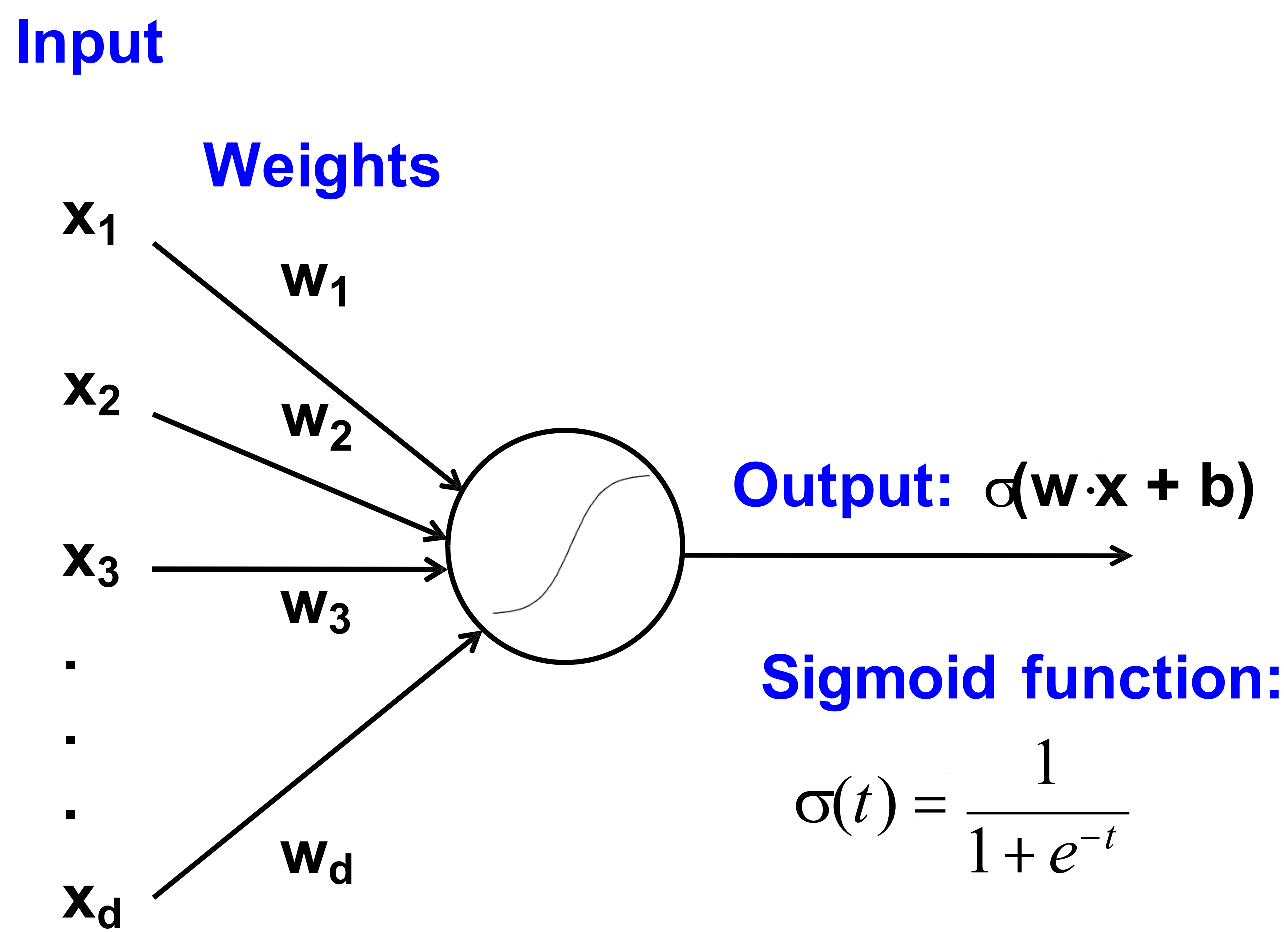
Traditional recognition: “Shallow” architecture



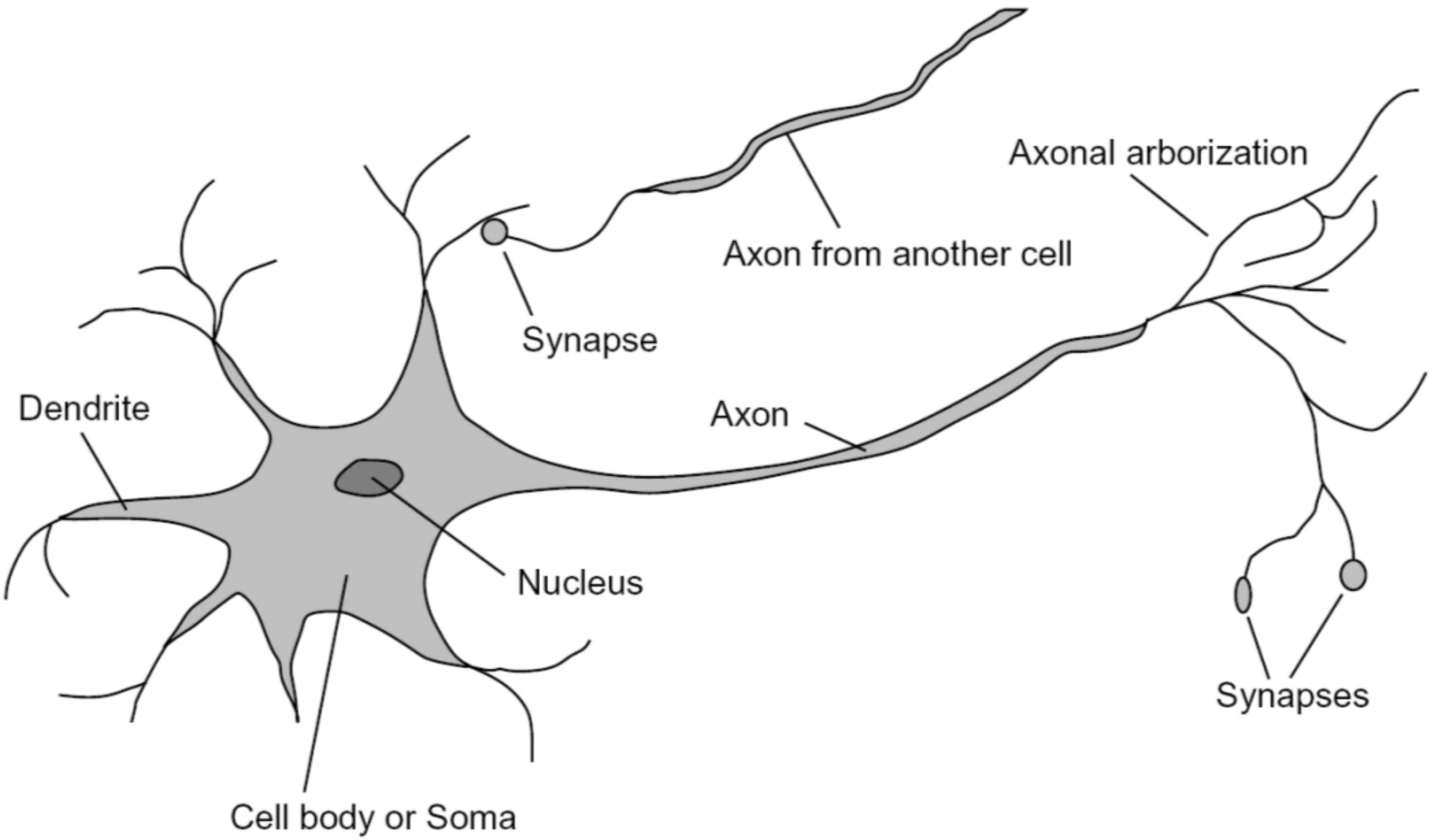
Deep learning: “Deep” architecture



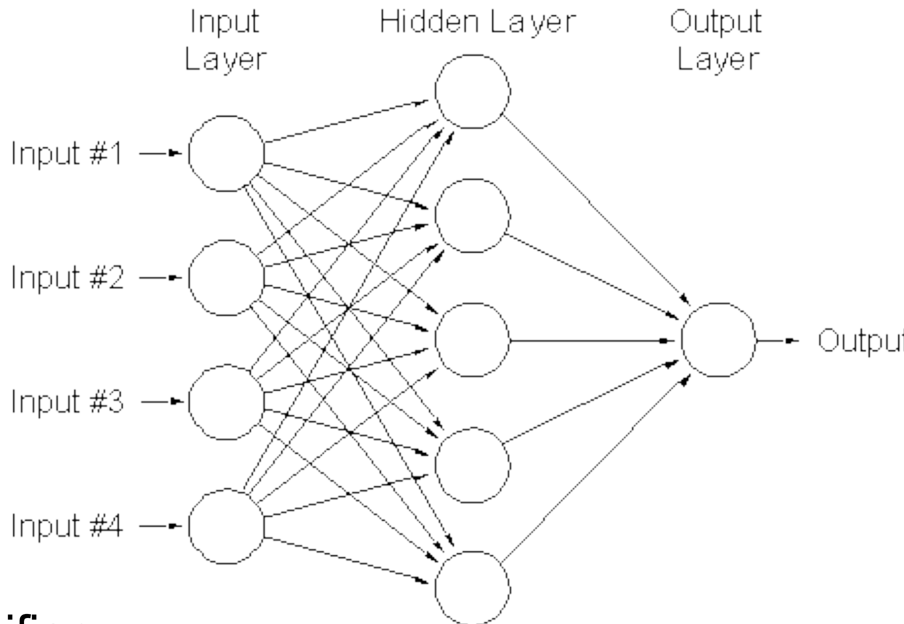
Background: Perceptrons



Inspiration: Neuron cells



Background: Multi-Layer Neural Networks



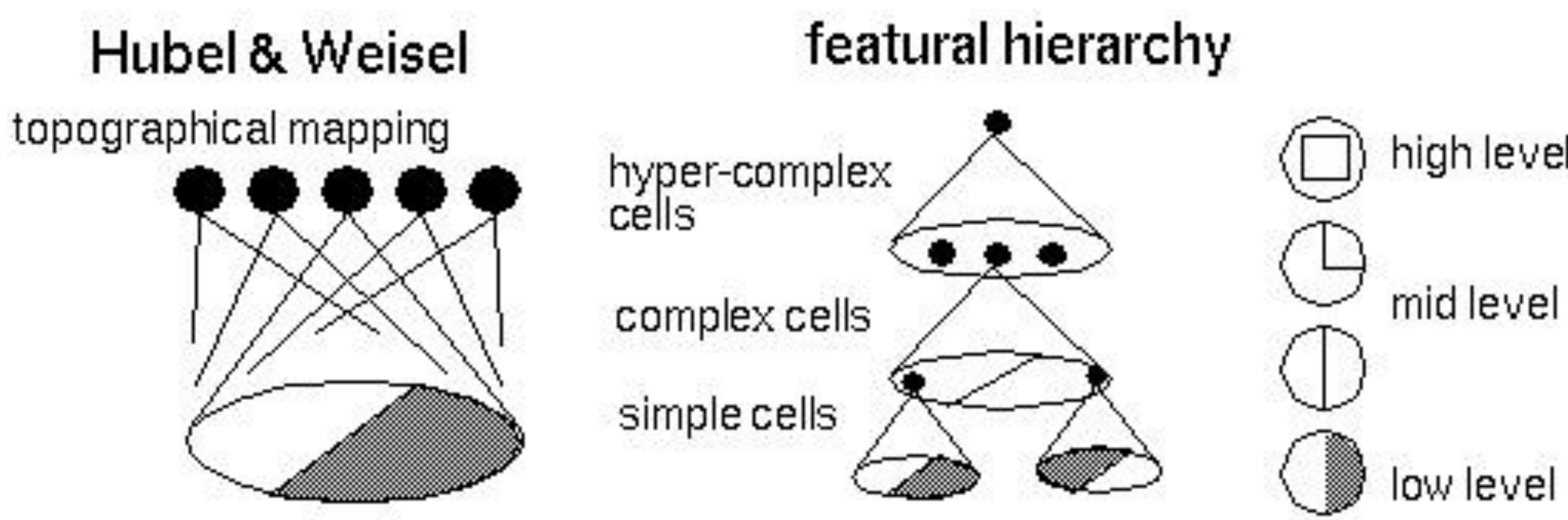
- Nonlinear classifier
- **Training:** find network weights **w** to minimize the error between true training labels y_i and estimated labels $f_{\mathbf{w}}(\mathbf{x}_i)$:

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

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

Hubel/Wiesel Architecture

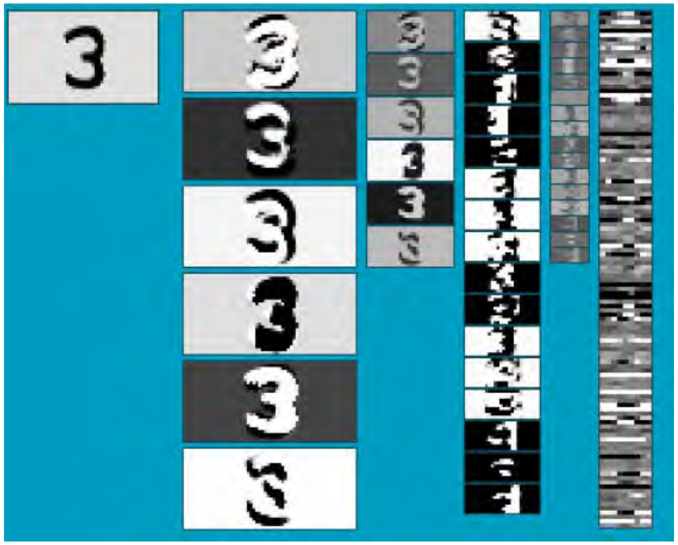
- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells

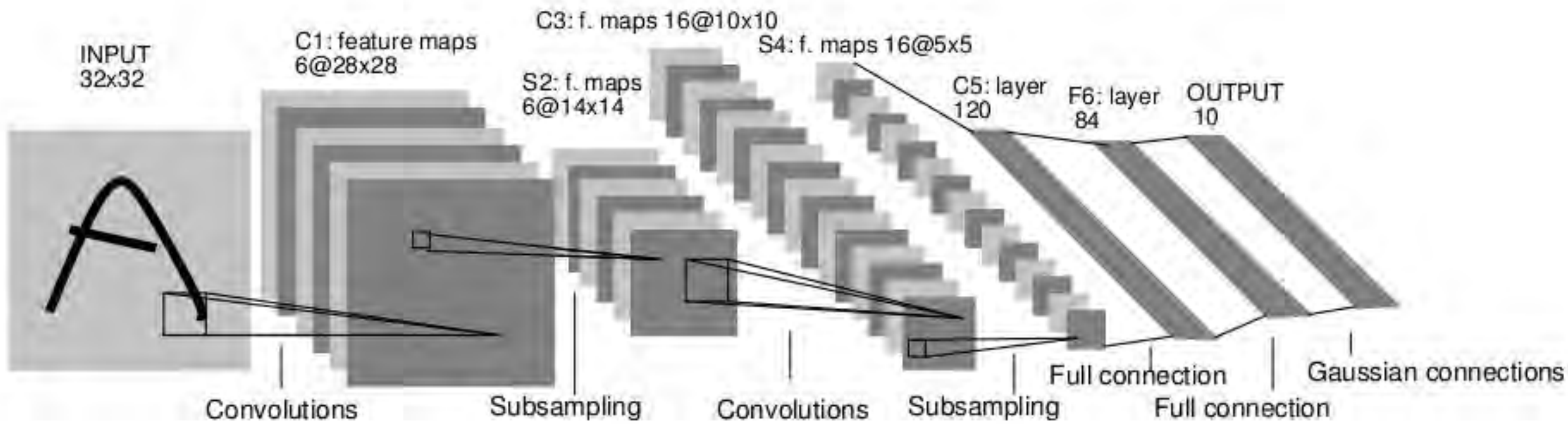


[Source]

Convolutional Neural Networks (CNN, Convnet)

- 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

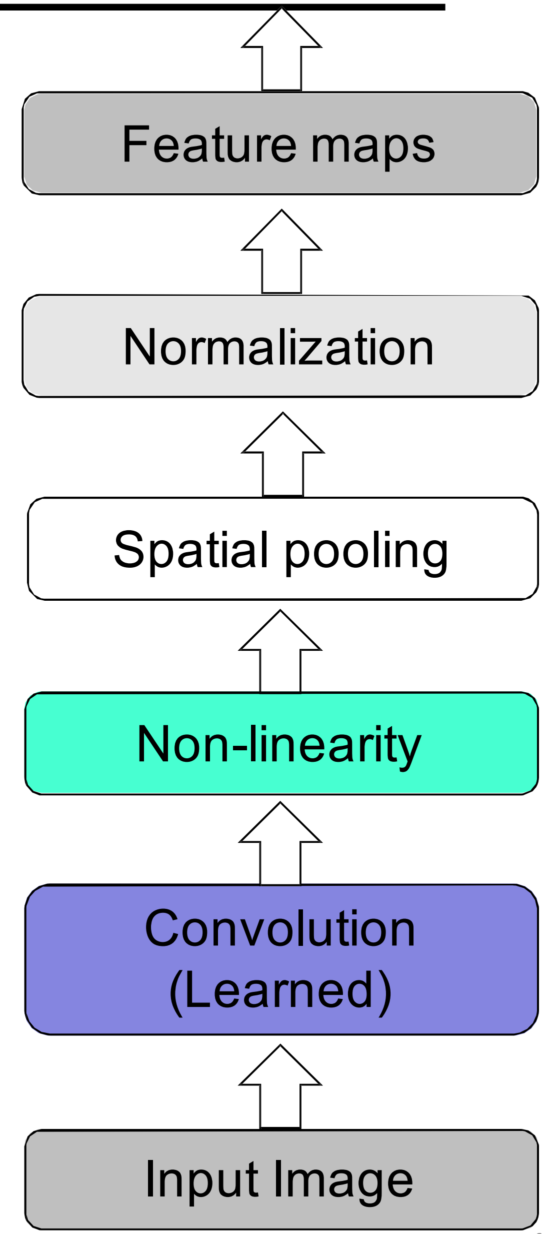




Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](), Proceedings of the IEEE 86(11): 2278–2324, 1998.

Convolutional Neural Networks (CNN, Convnet)

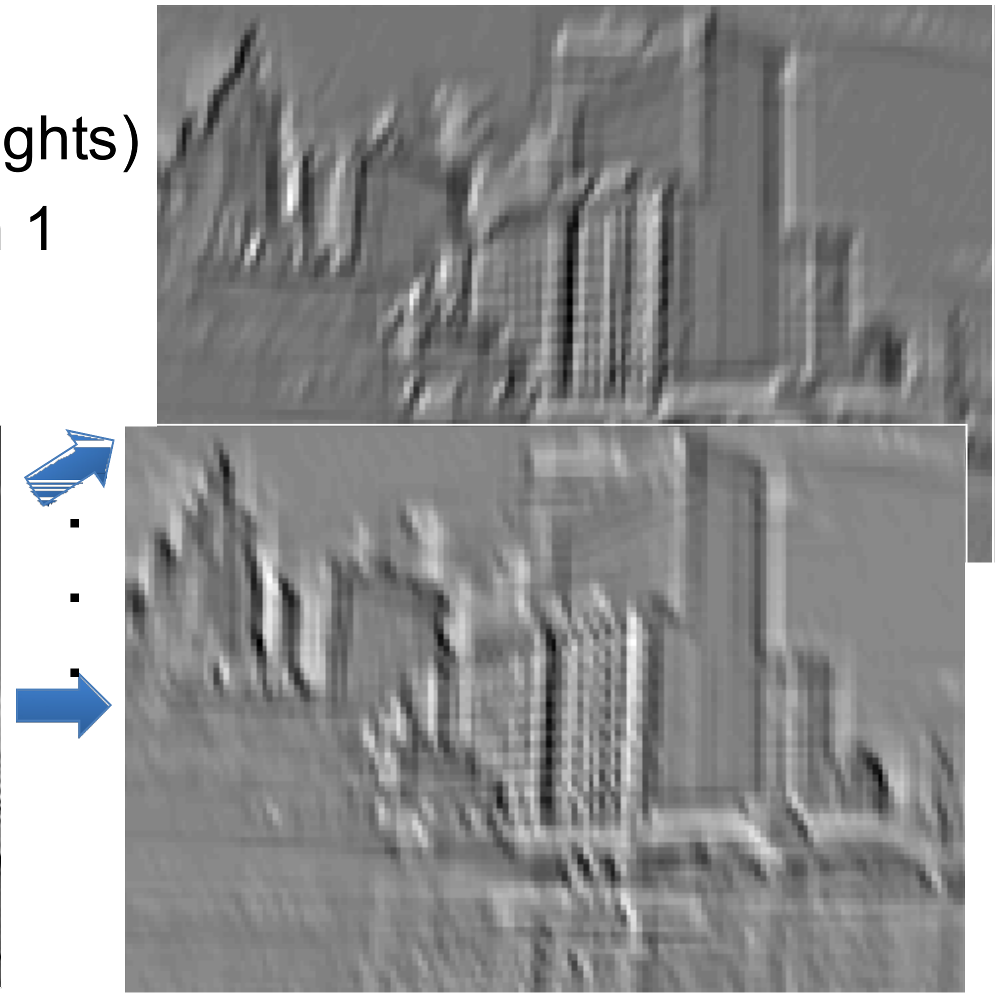
- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. Normalization
- Supervised training of convolutional filters by back-propagating classification error



1. Convolution

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

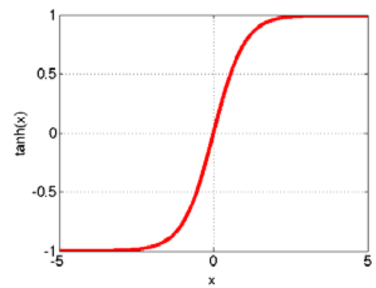
Input

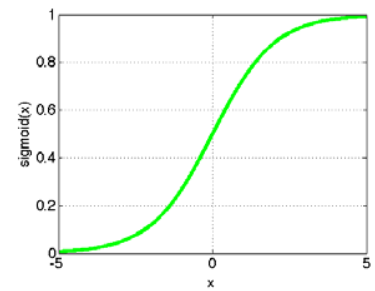


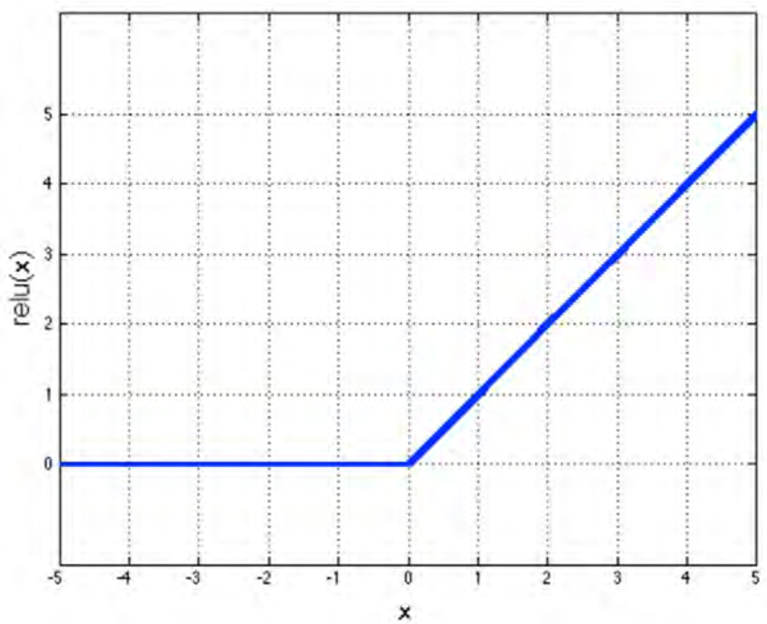
Feature Map

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

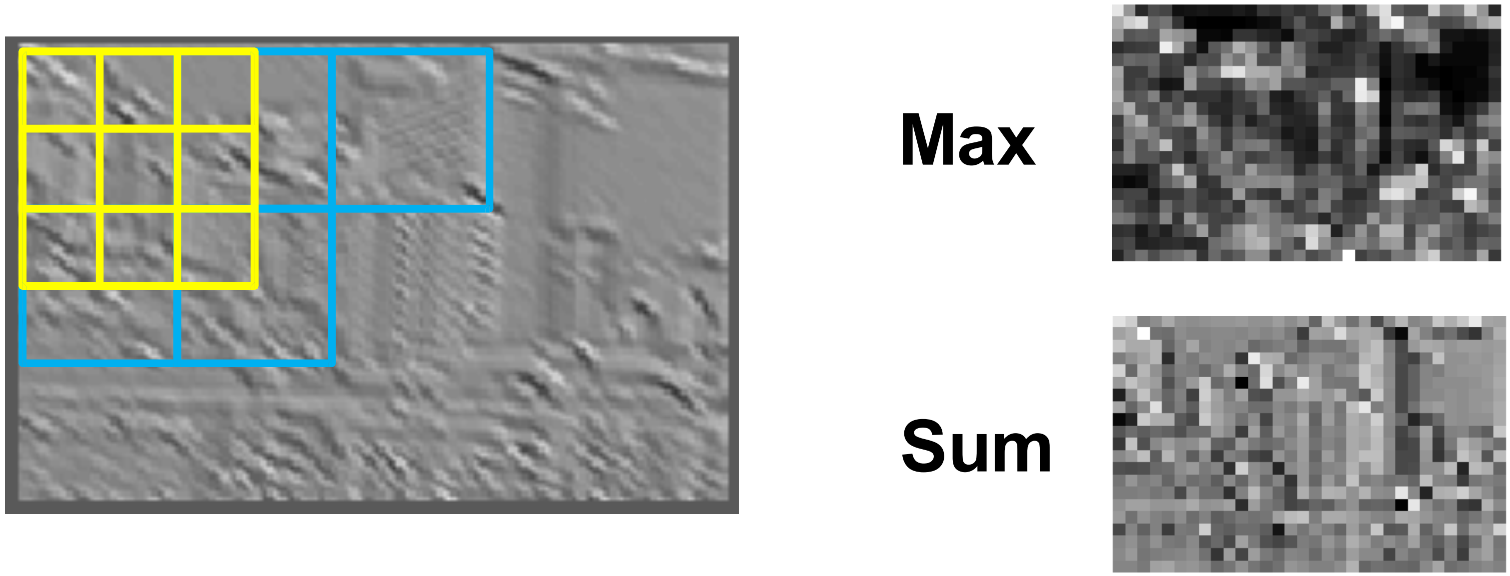






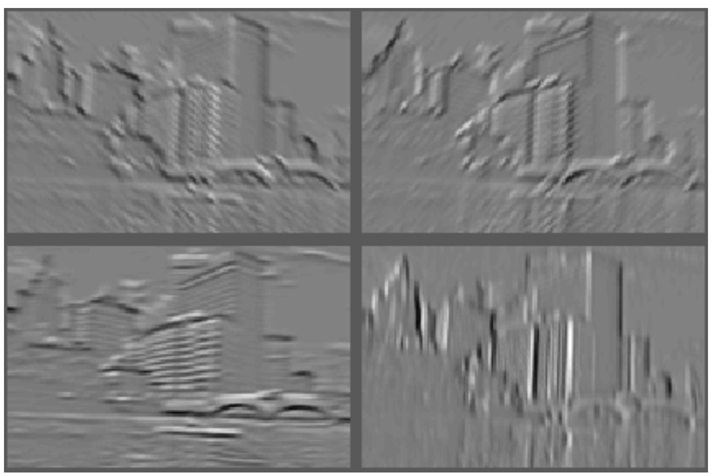
3. Spatial Pooling

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

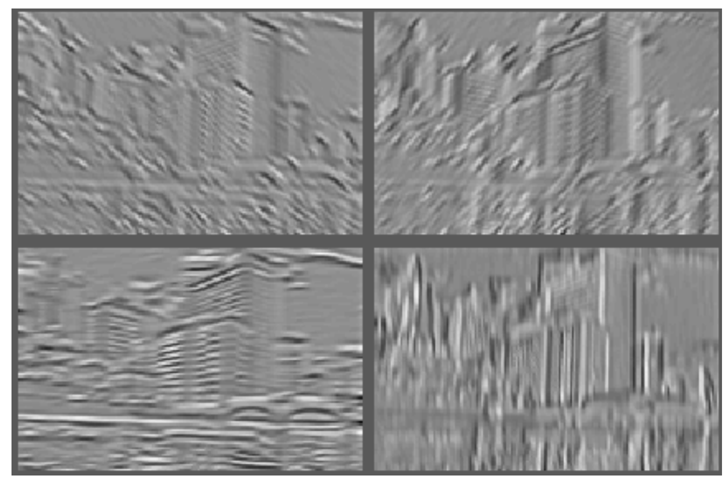


4. Normalization

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



Feature Maps



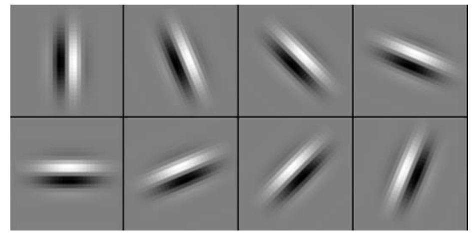
Feature Maps
After Contrast Normalization

Compare: SIFT Descriptor

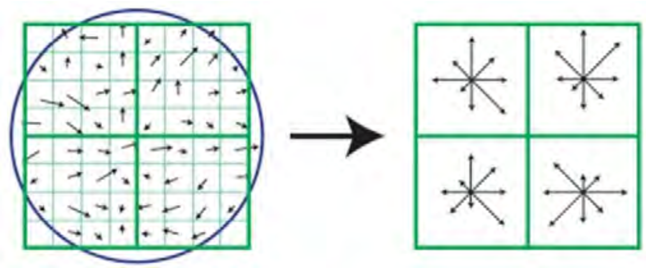
Lowe [IJCV 2004]

Image Pixels

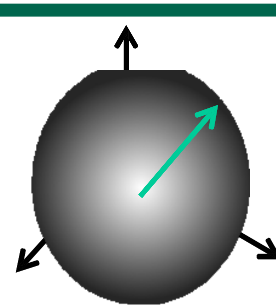
Apply oriented filters



Spatial pool (Sum)



Normalize to unit length



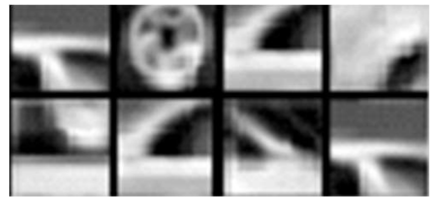
Feature Vector

Compare: Spatial Pyramid Matching

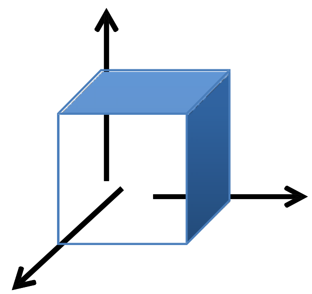
Lazebnik, Schmid, Ponce [CVPR 2006]

SIFT features

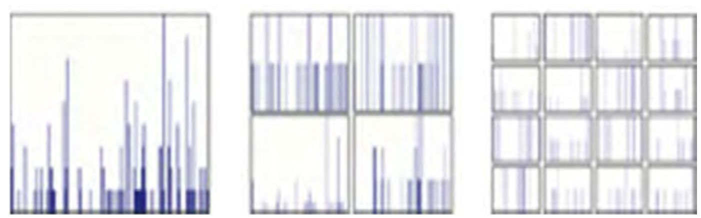
Filter with Visual Words



Take max VW response (L-inf normalization)



Multi-scale spatial pool (Sum)



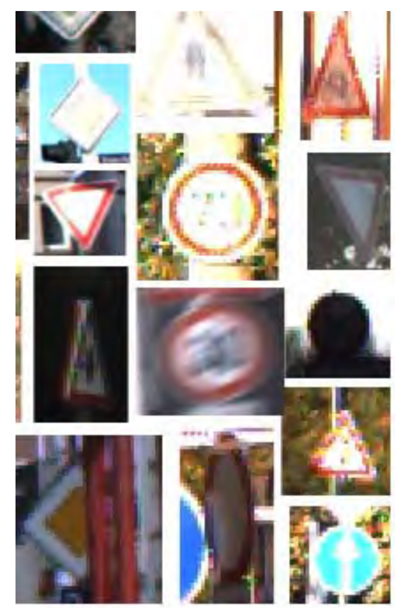
Global image descriptor

Convnet Successes

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

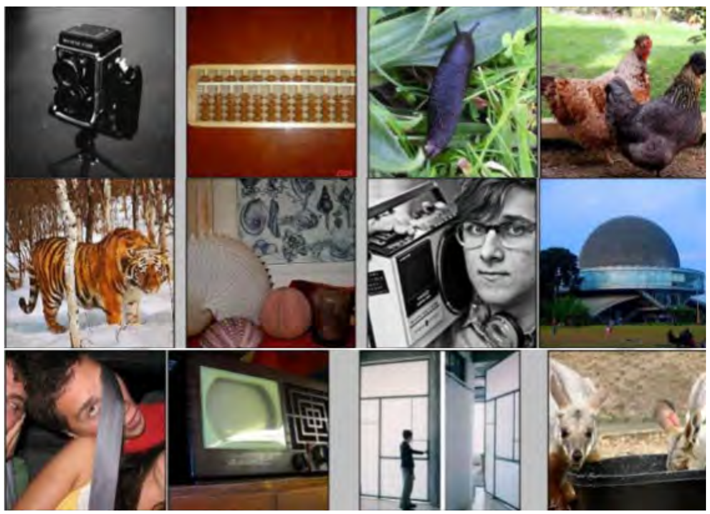
- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]

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



ImageNet Challenge 2012

IMAGENET



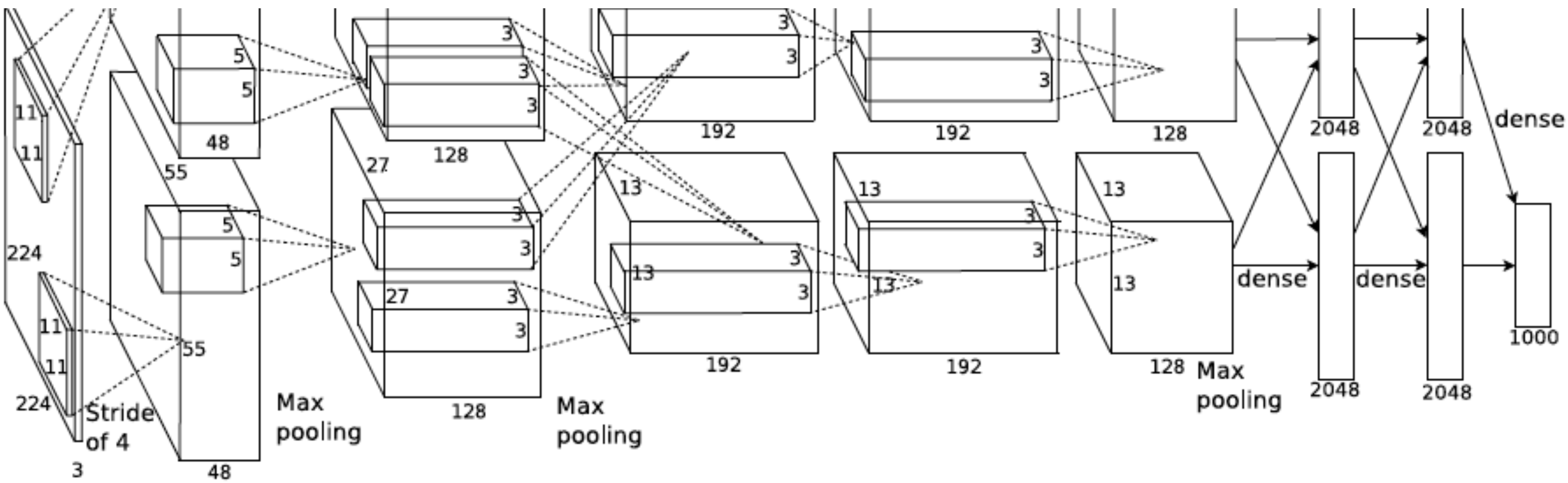
[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- 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)

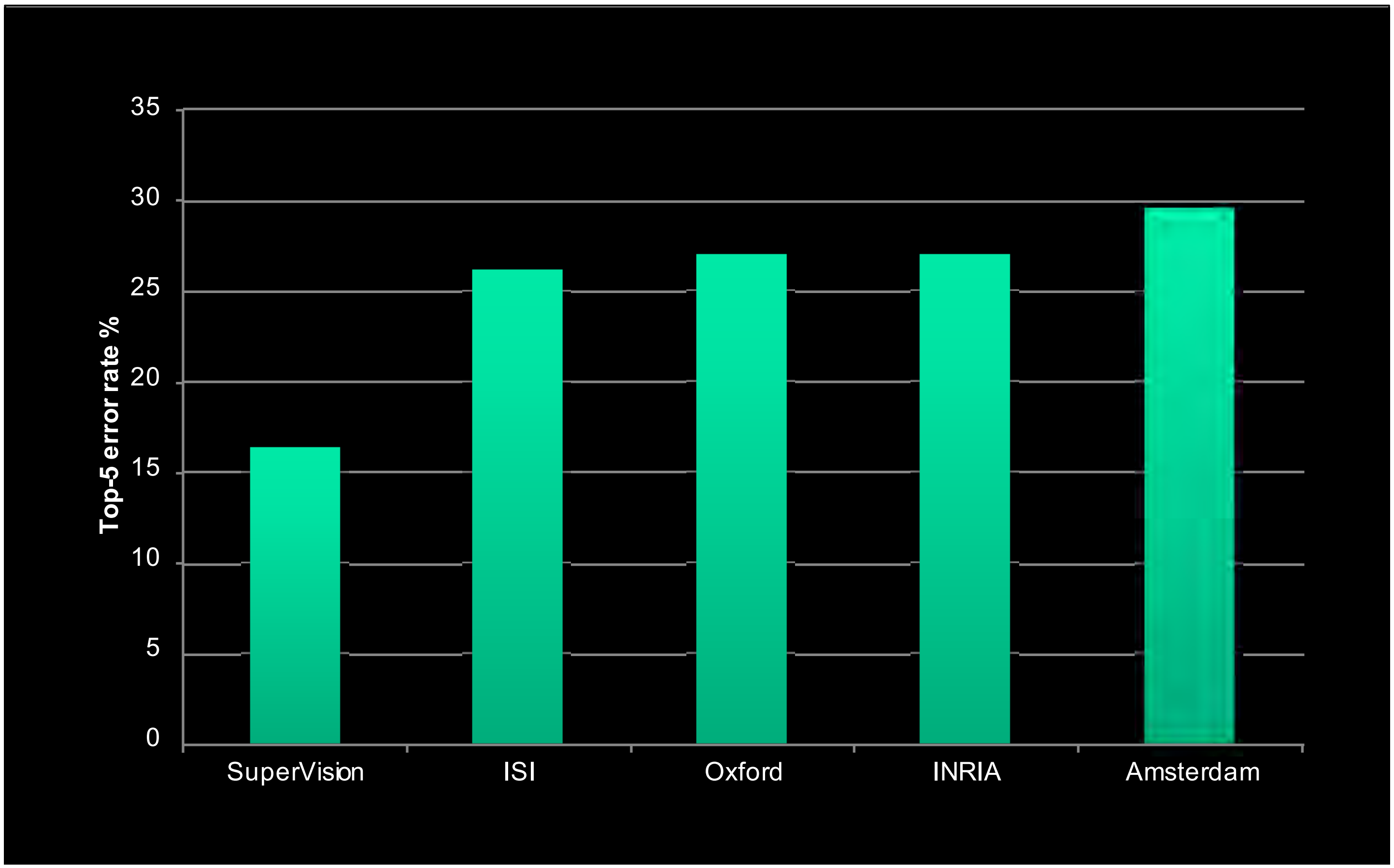


A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks], NIPS 2012

ImageNet Challenge 2012

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

Next best (non-convnet) – **26.2% error**



Visualizing Convnets

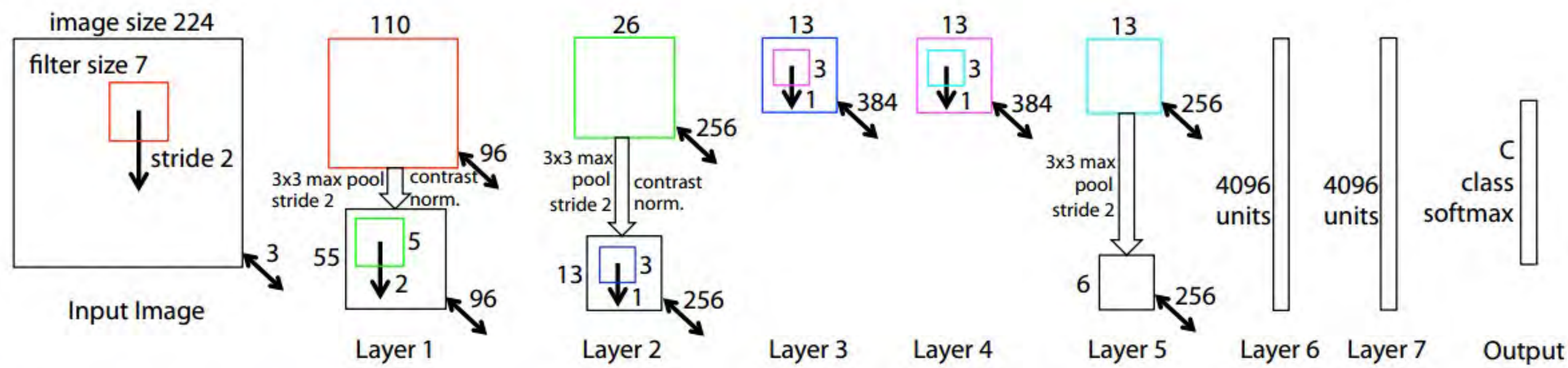
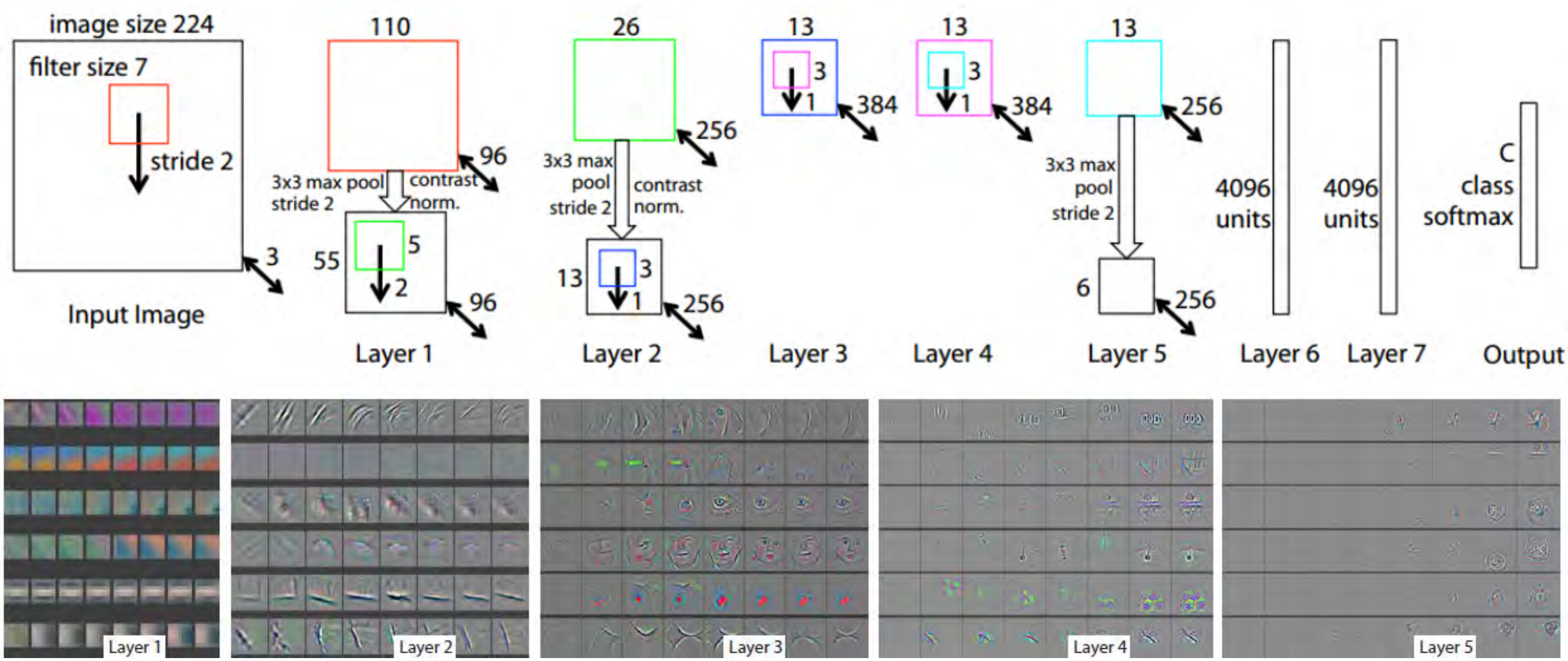


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through 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

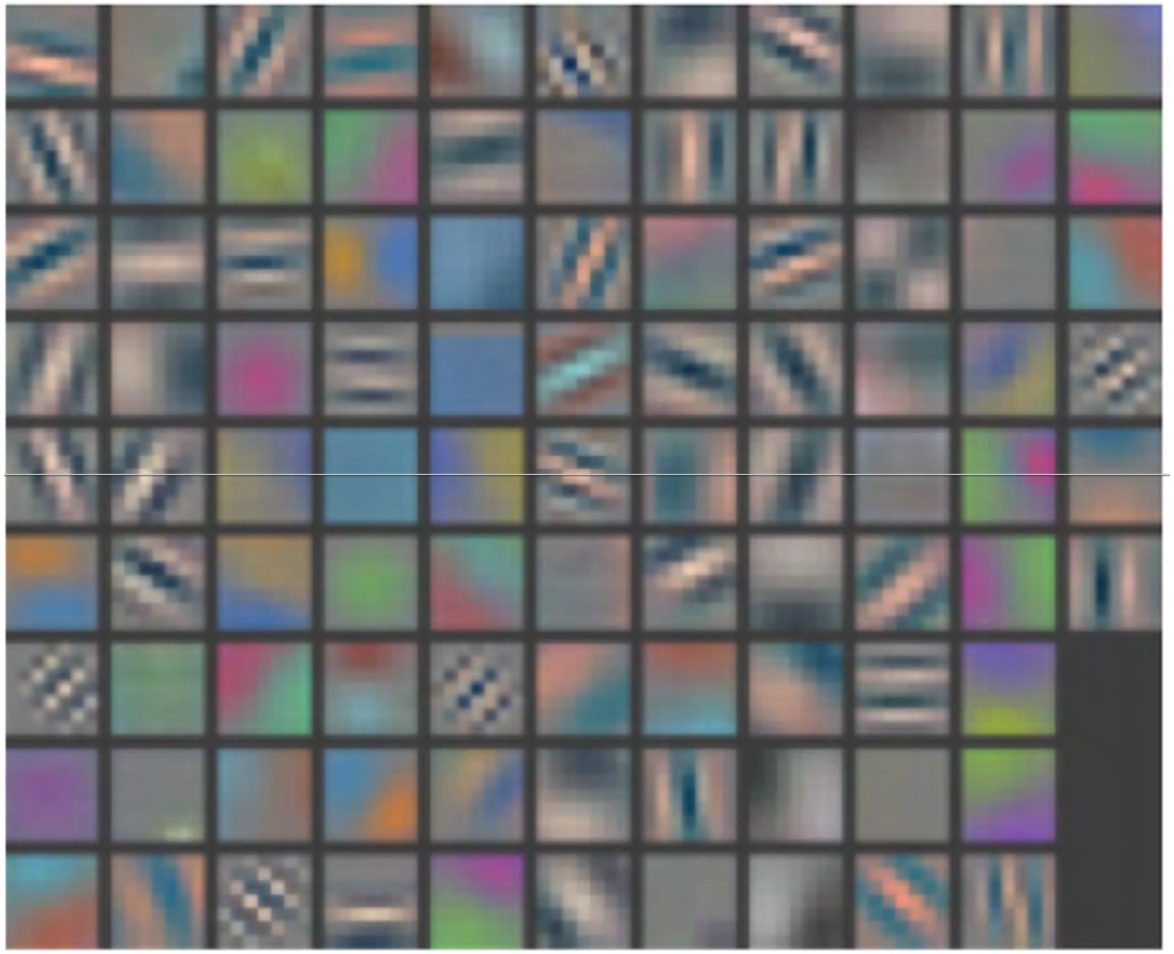
Meaning of Each Layer in Convnets



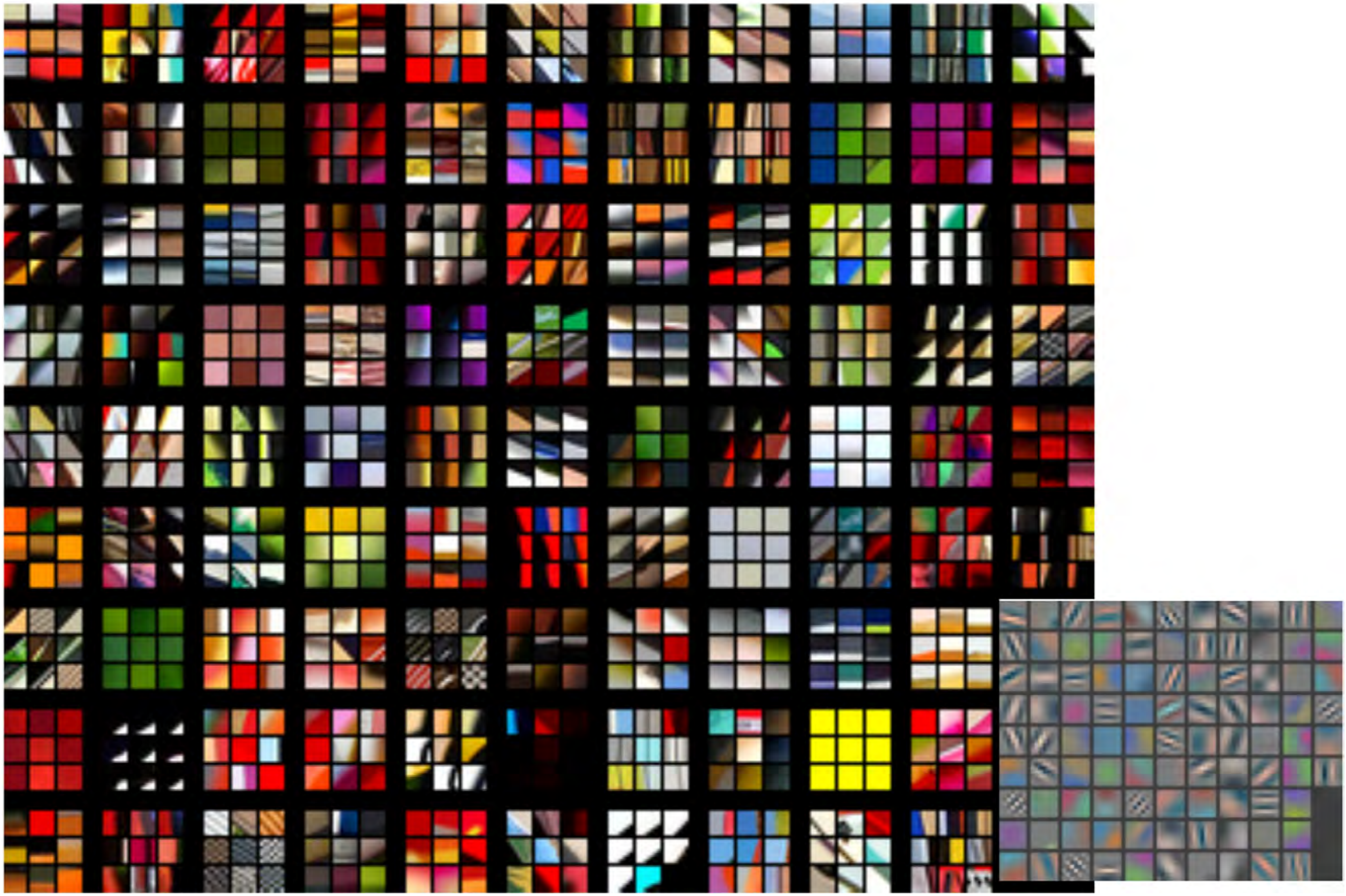
It is deep: 7 hidden layers

Deep Neural Network

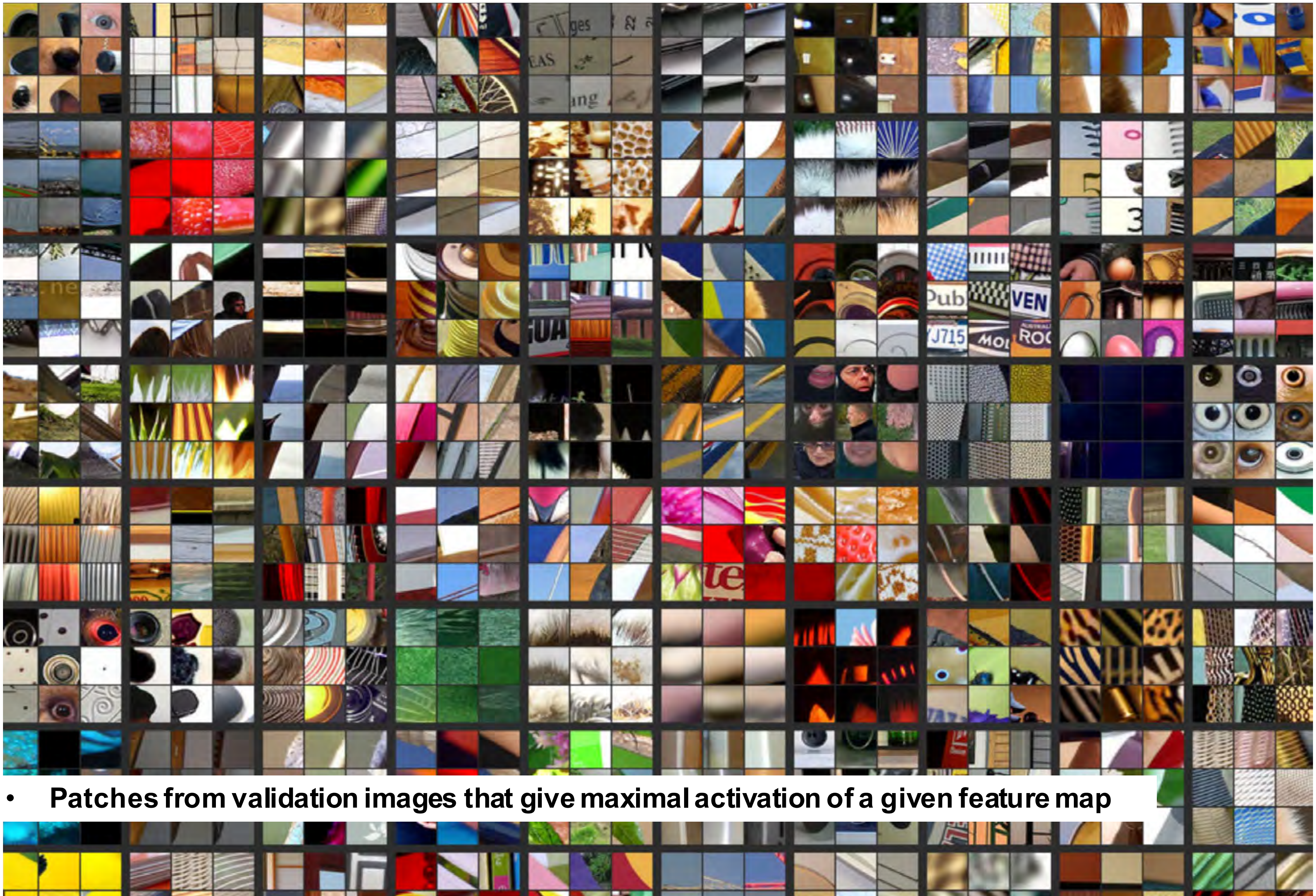
Layer 1 Filters



Layer 1: Top-9 Patches

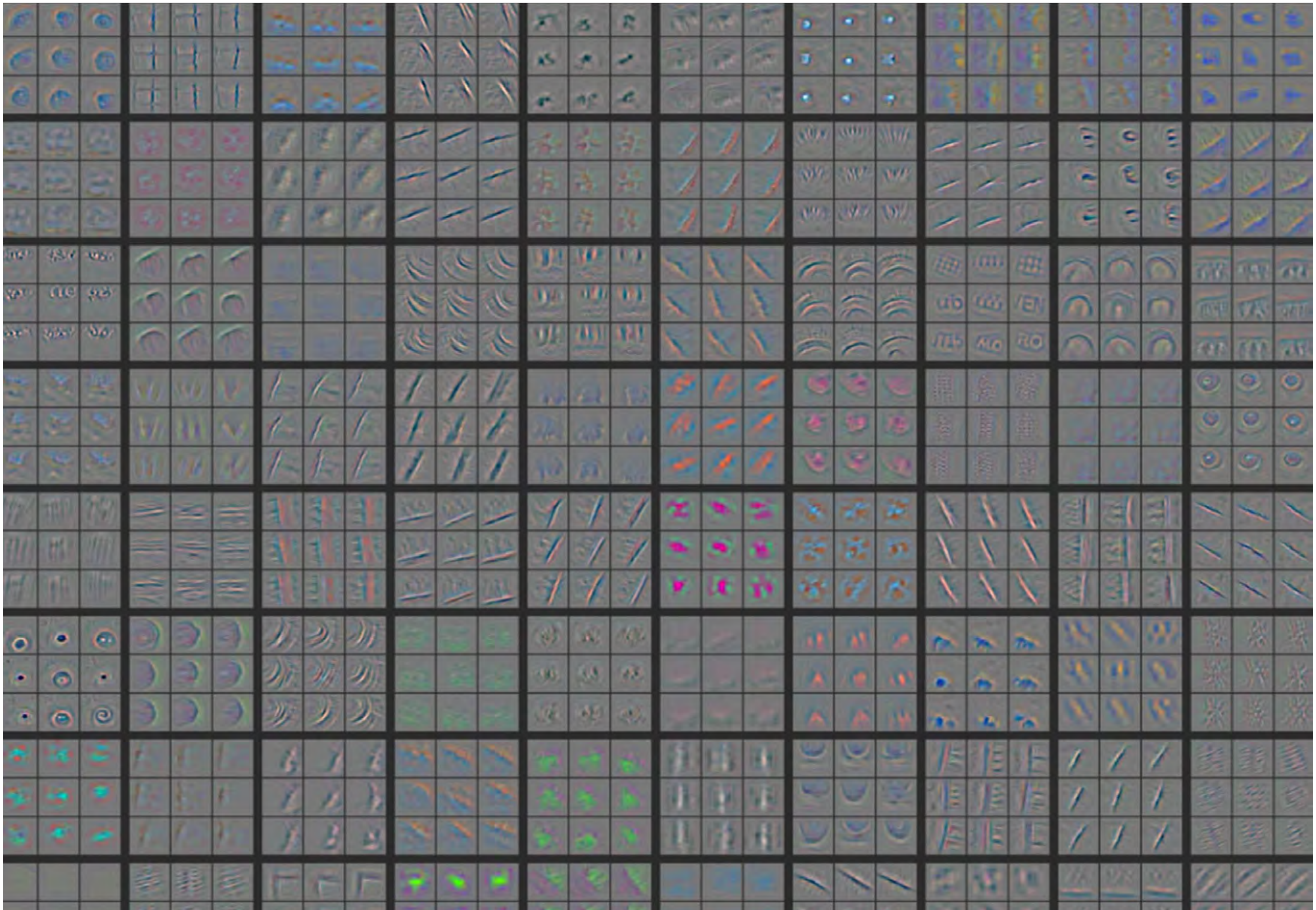


Layer 2: Top-9 Patches



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

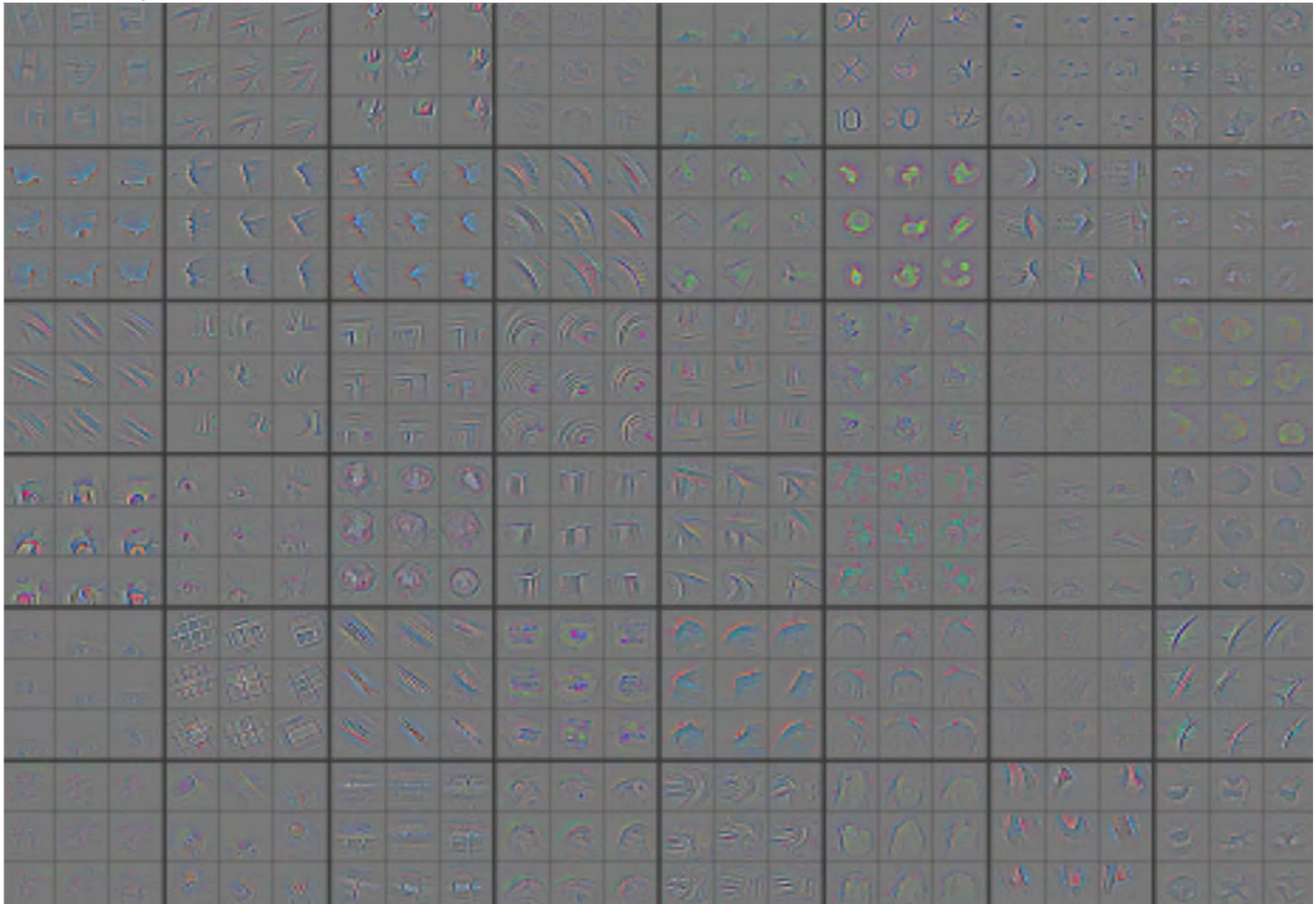
Layer 2: Top-9 Patches



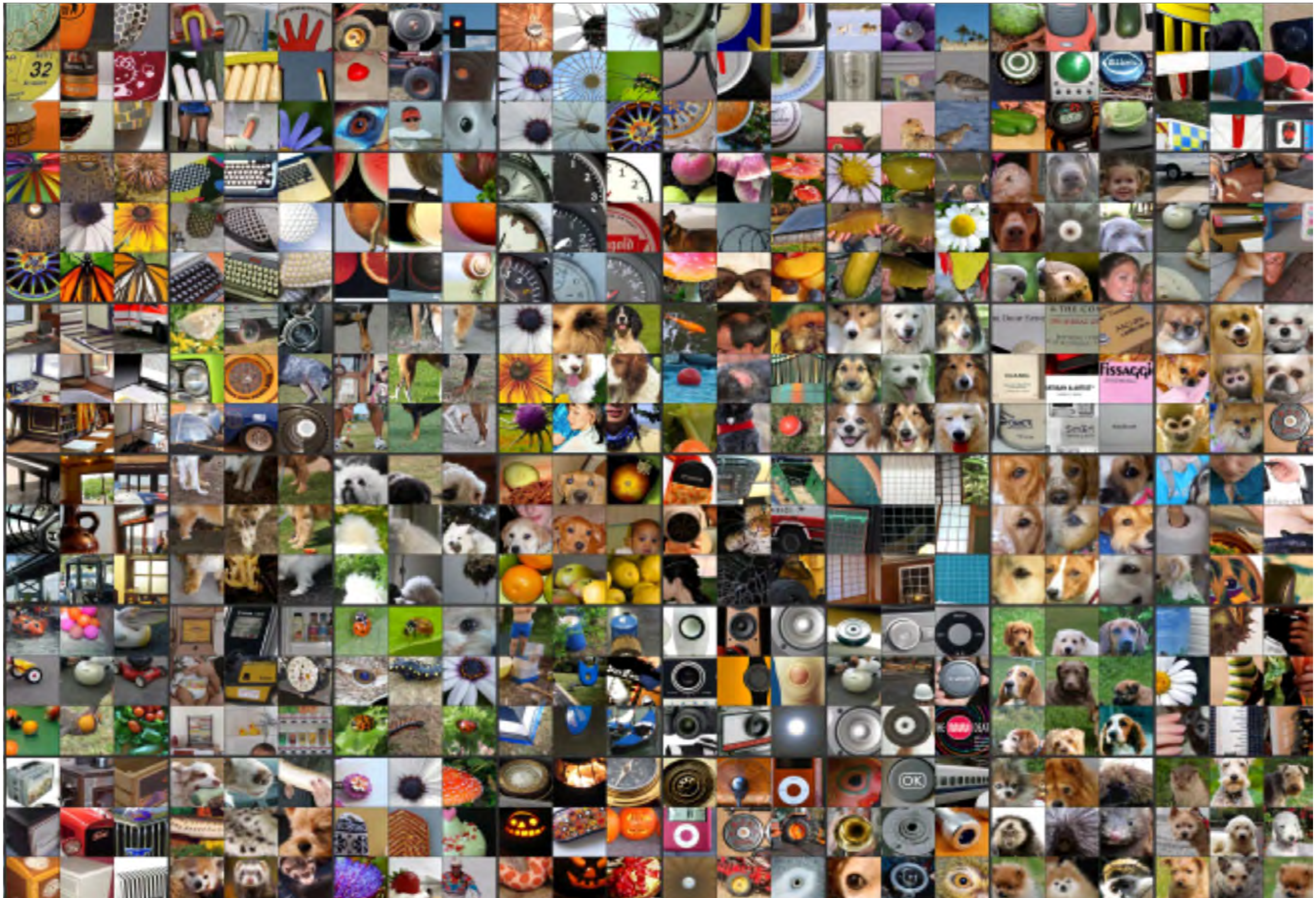
Layer 3: Top-9 Patches

Layer 3: Top-9 Patches

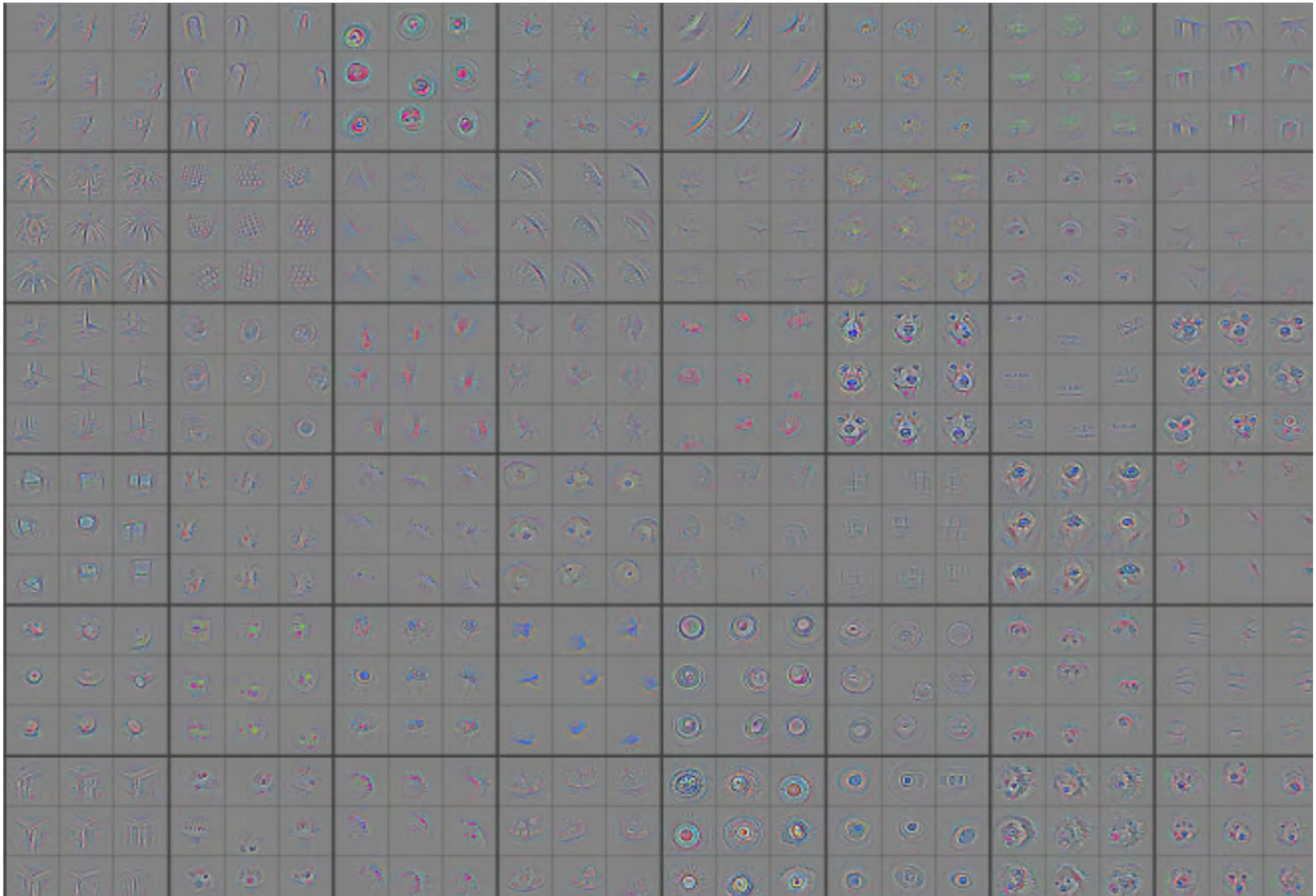
Layer 3: Top-9 Patches



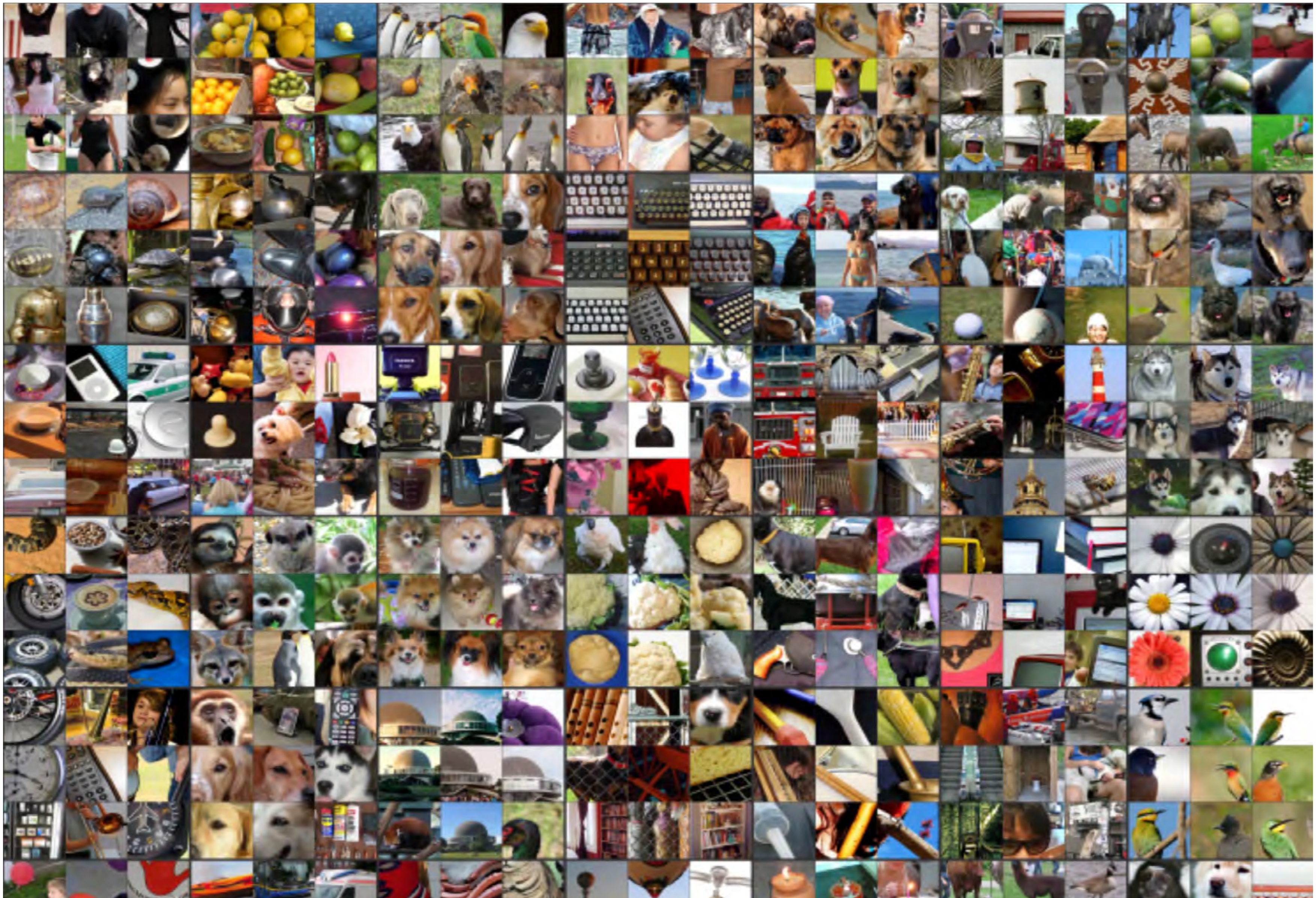
Layer 4: Top-9 Patches



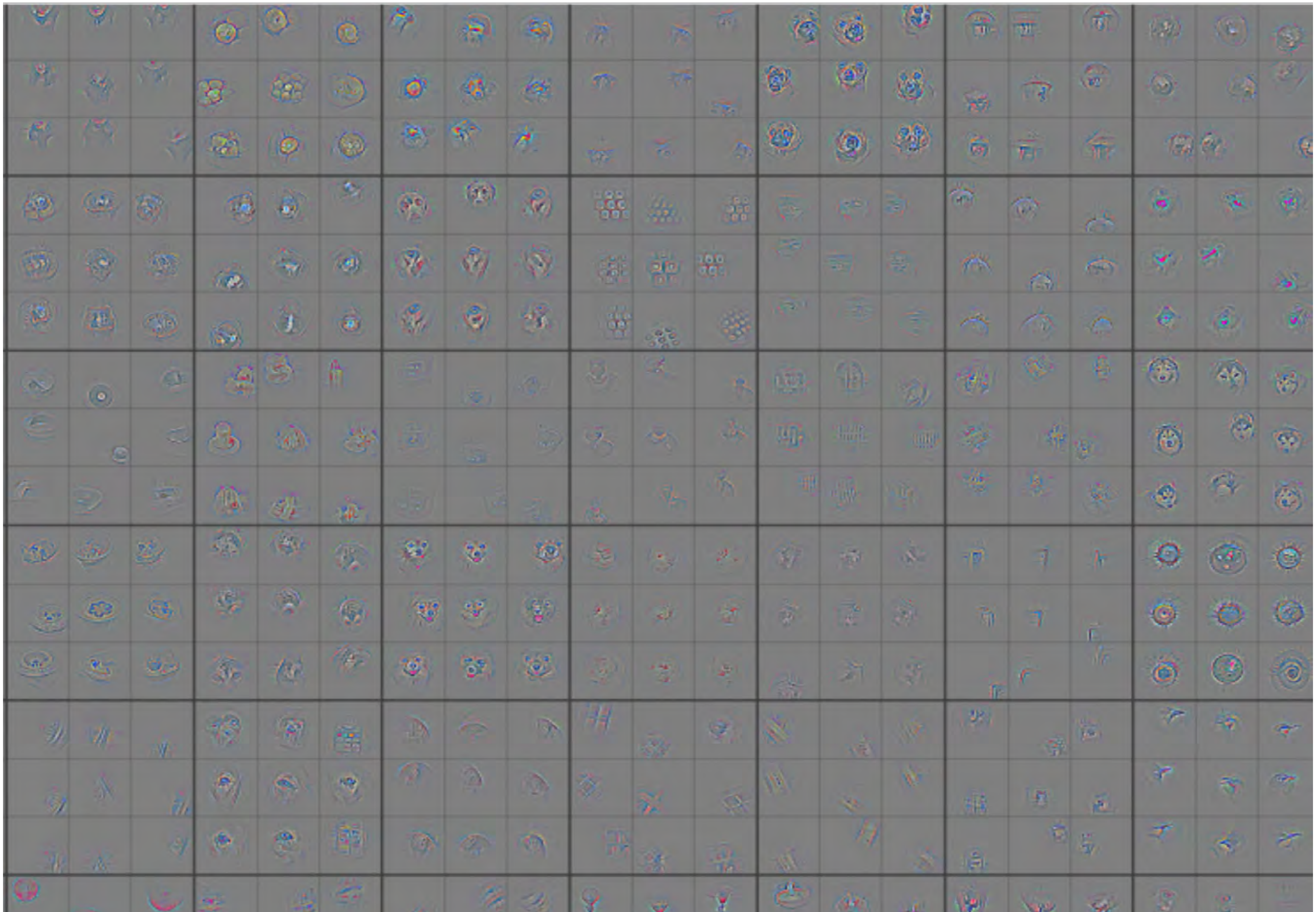
Layer 4: Top-9 Patches



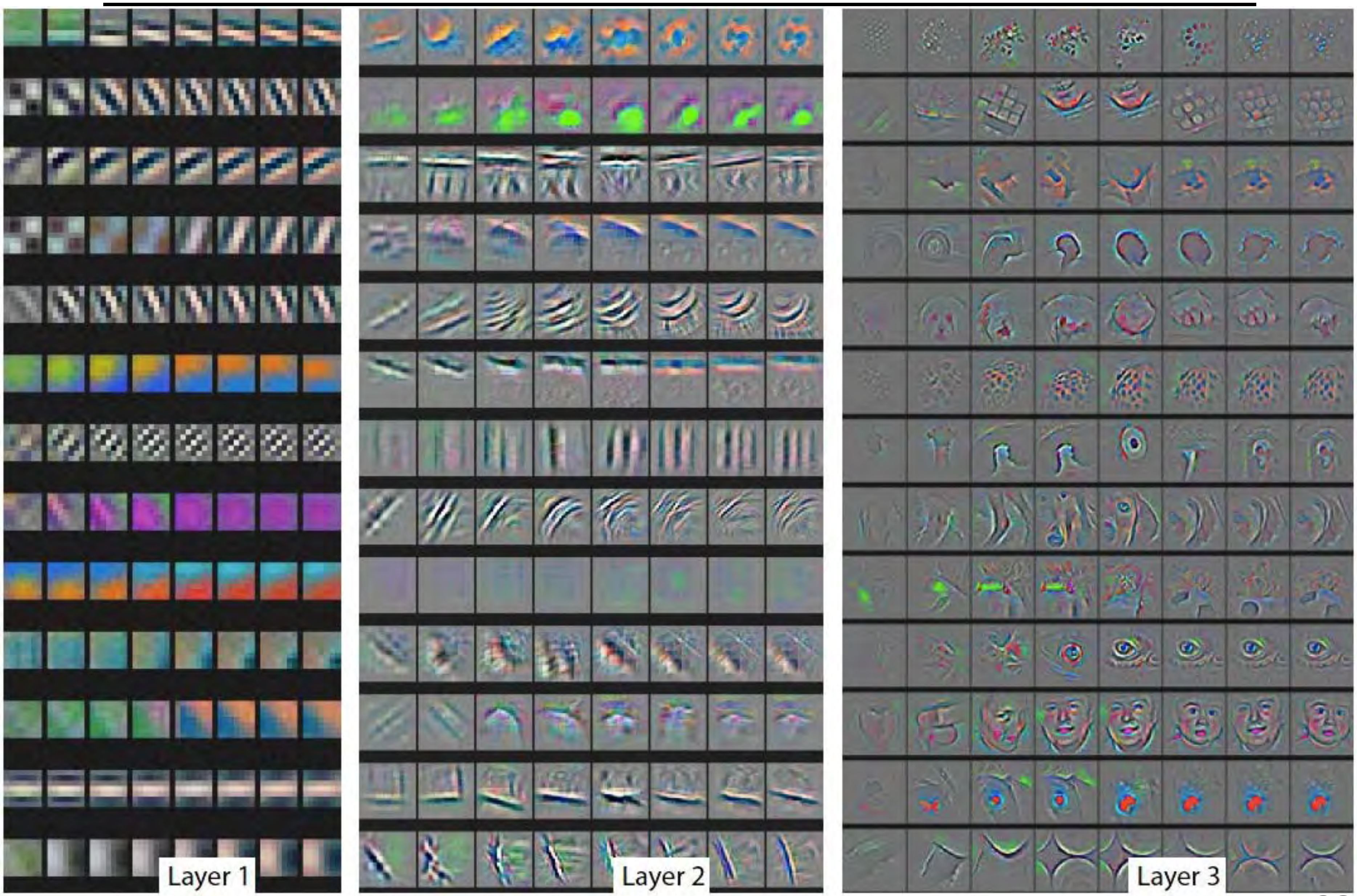
Layer 5: Top-9 Patches



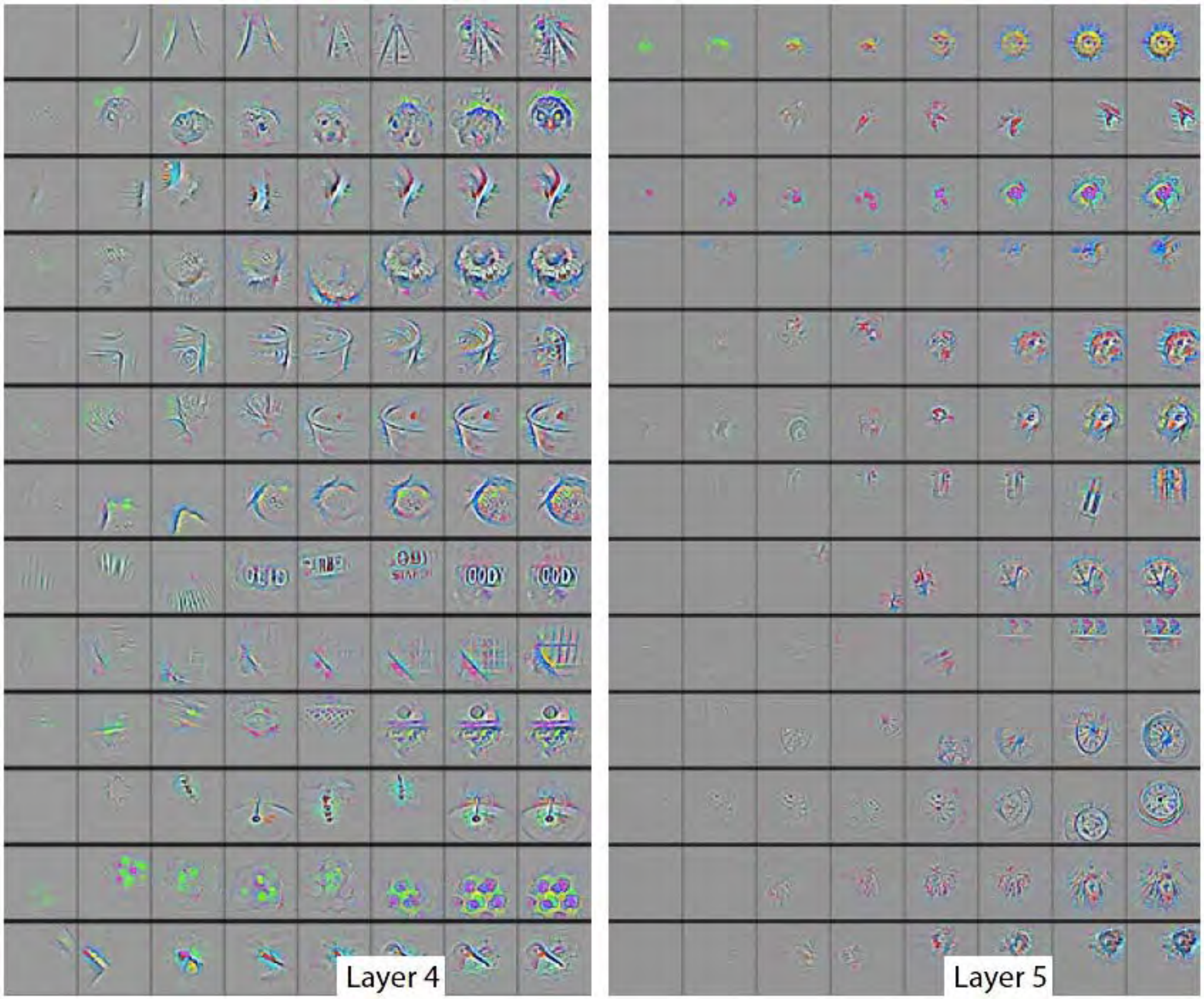
Layer 5: Top-9 Patches



Evolution of Features During Training

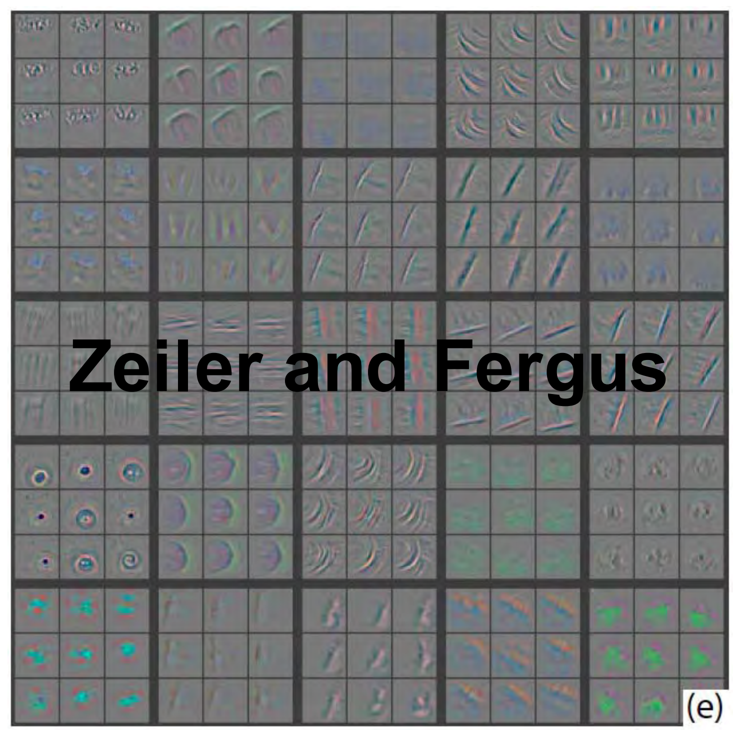


Evolution of Features During Training



Diagnosing Problems

- Visualization of Krizhevsky et al.'s architecture showed some problems with layers 1 and 2
  - Large stride of 4 used
- Alter architecture: smaller stride & filter size
  - Visualizations look better
  - Performance improves



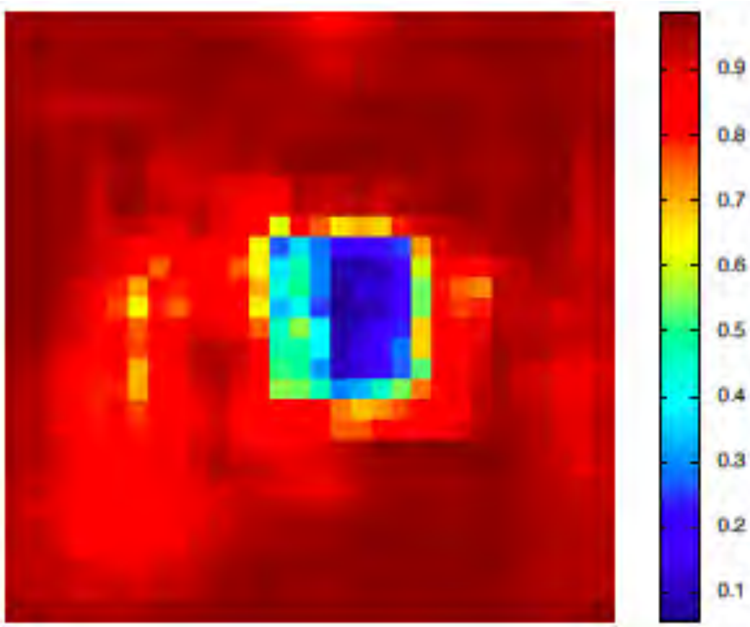
Occlusion Experiment

- Mask parts of input with occluding square

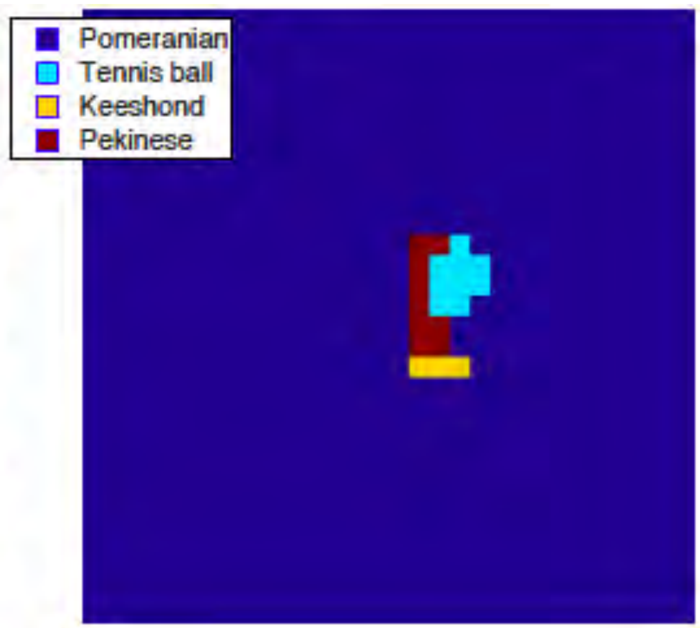
- Monitor output

Input image

p(True class)

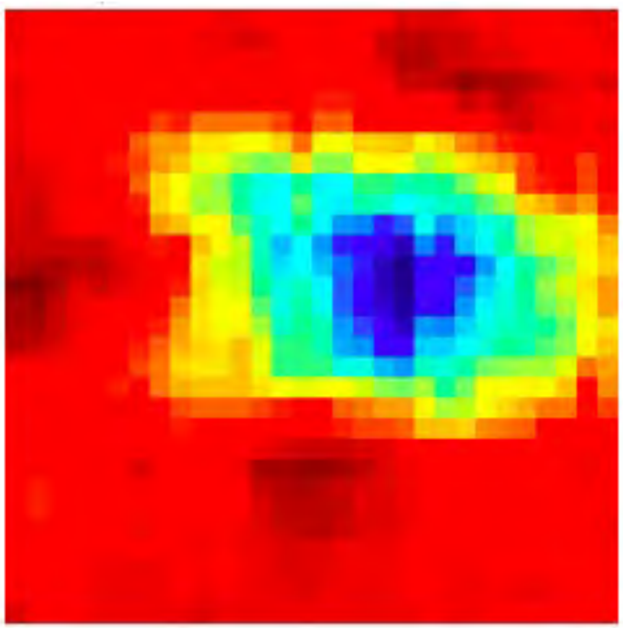


Most probable class

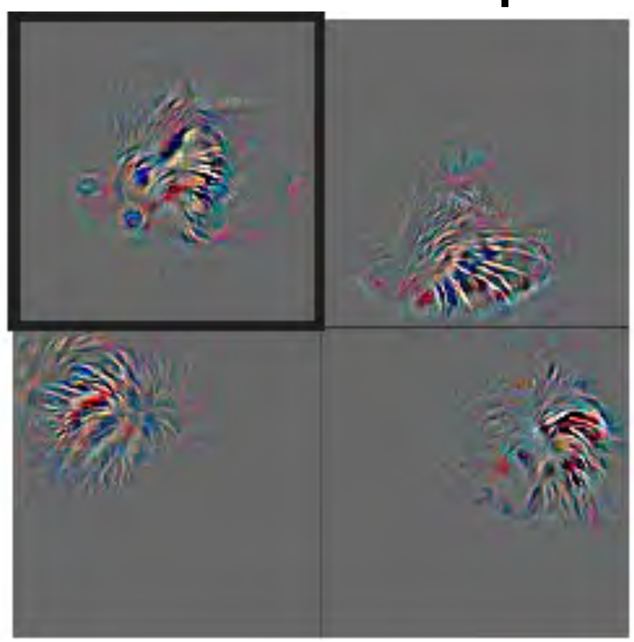


Input image

Total activation in most active 5th layer feature map

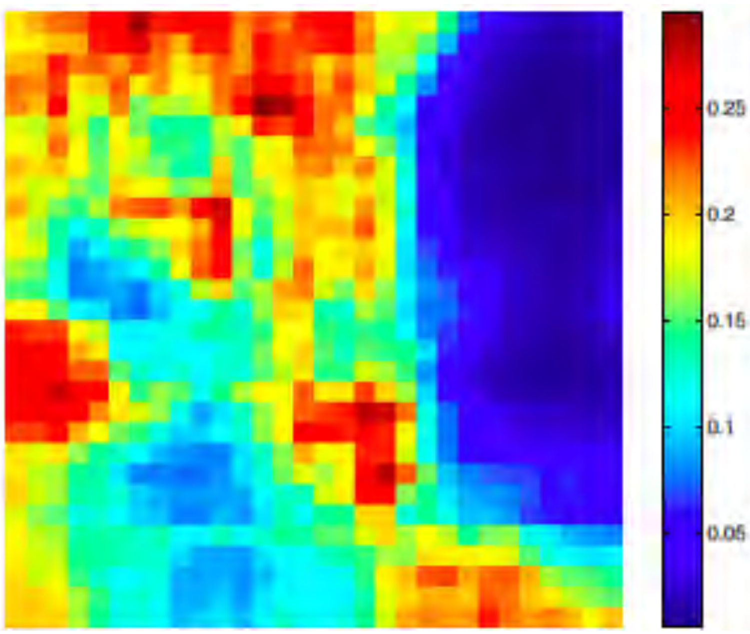


Other activations from same feature map

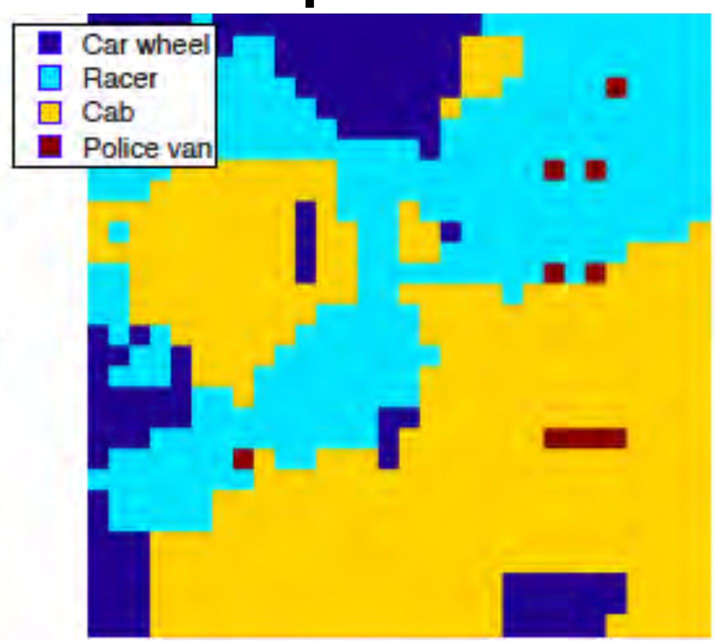


Input image

p(True class)

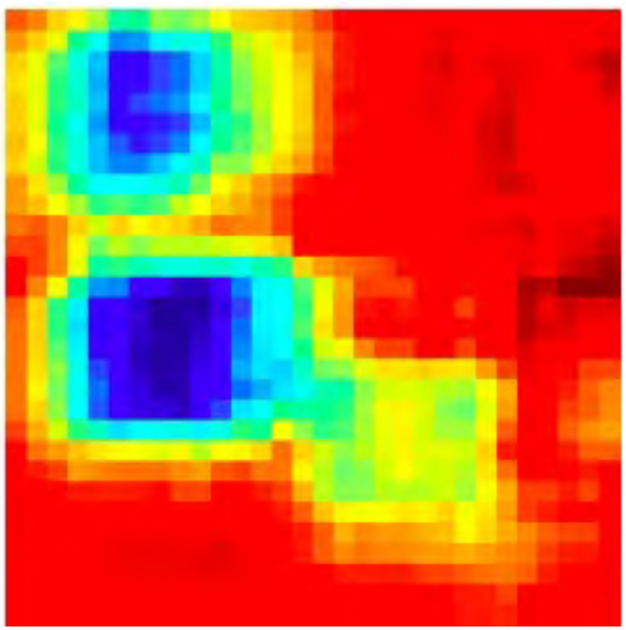


Most probable class



Input image

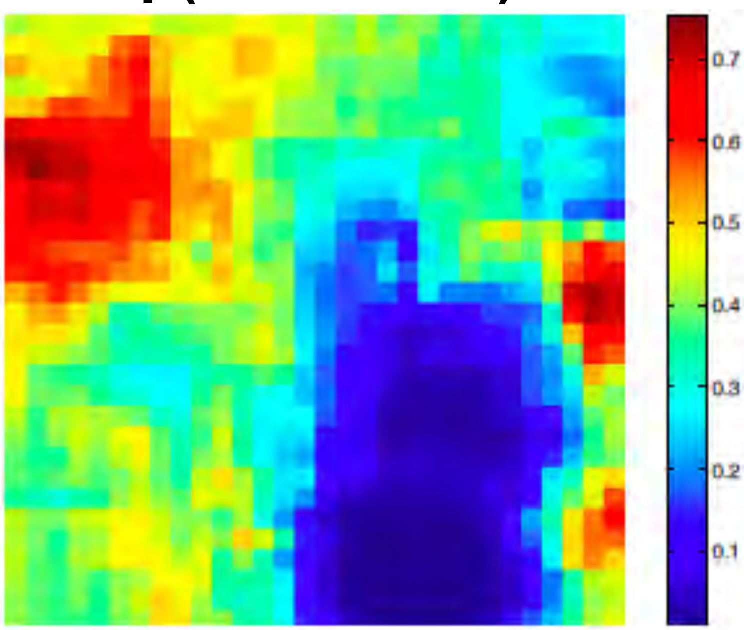
Total activation in most active 5th layer feature map



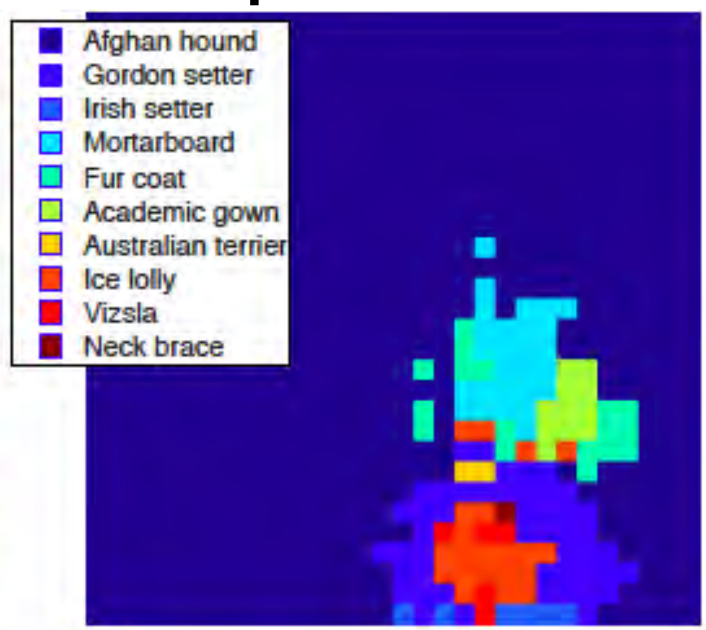
Other activations from same feature map

Input image

p(True class)

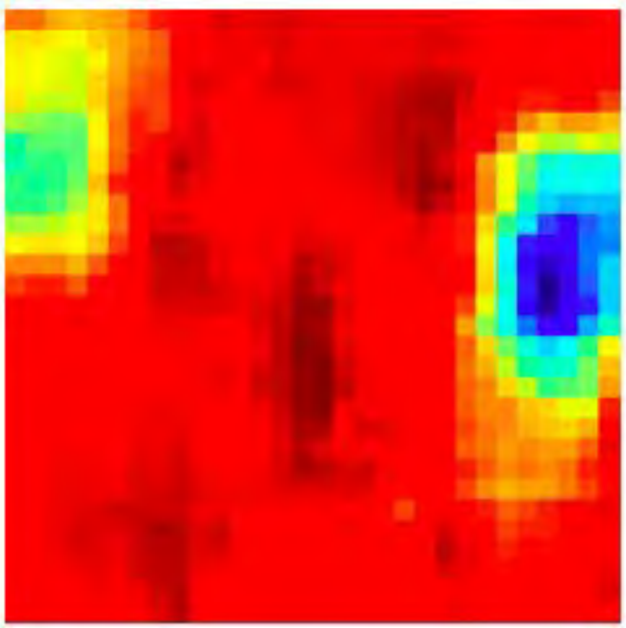


Most probable class



Input image

Total activation in most active 5th layer feature map

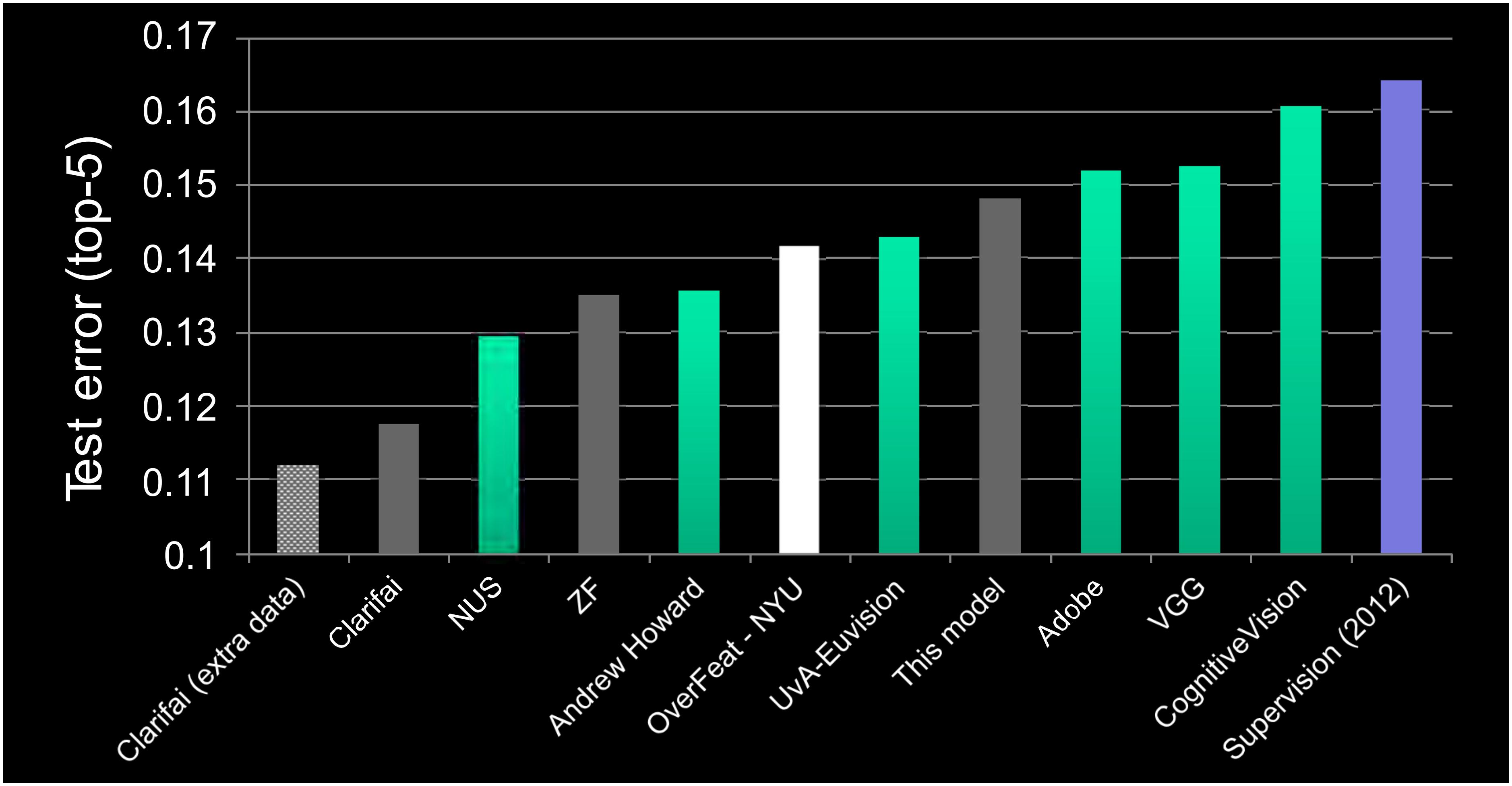


Other activations from same feature map

ImageNet Classification 2013 Results

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

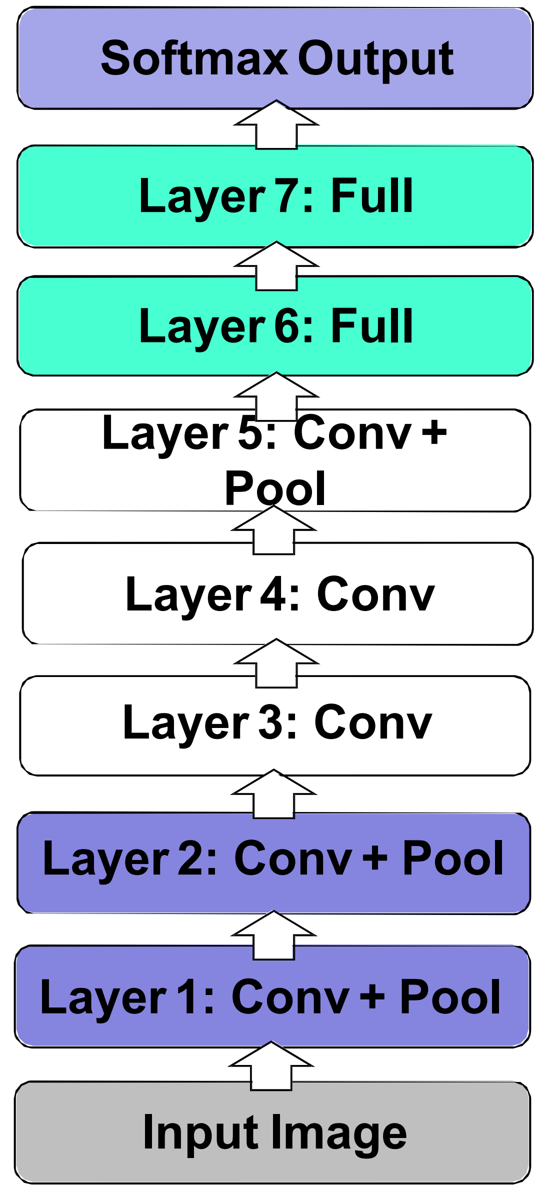
Demo: http://www.clarifai.com/



How important is depth?

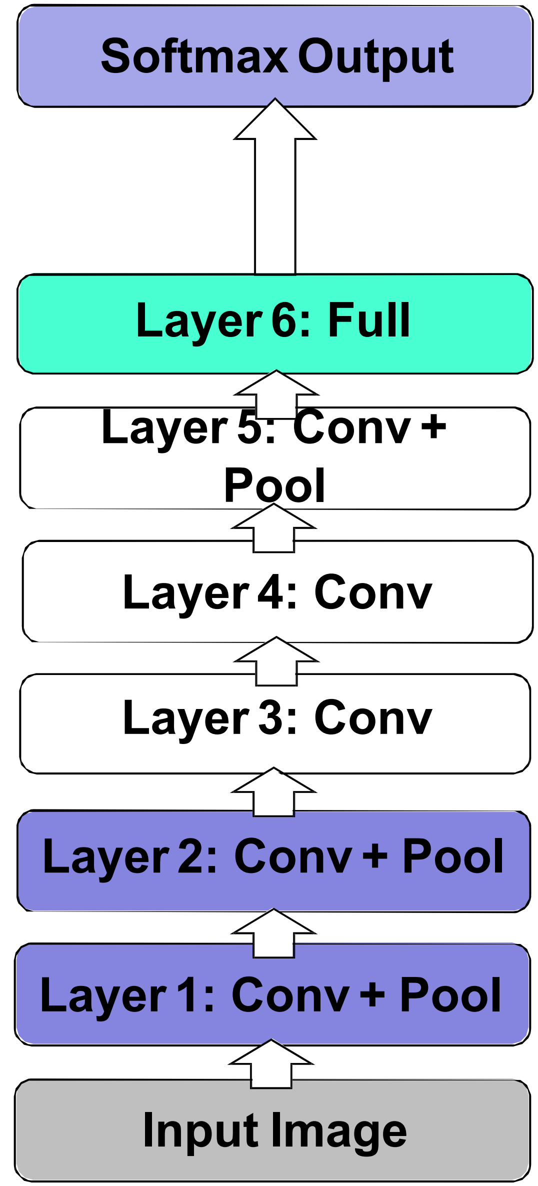
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on ImageNet
- 18.1% top-5 error



How important is depth?

- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!



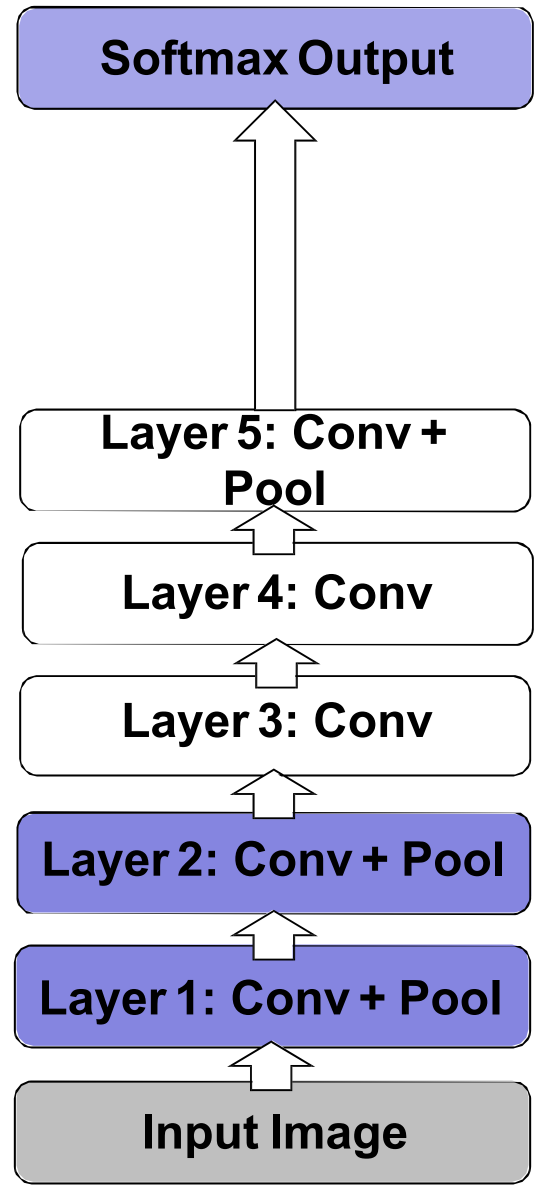
How important is depth?

Remove both fully connected layers

- Layer 6 & 7

Drop ~50 million parameters

5.7% drop in performance



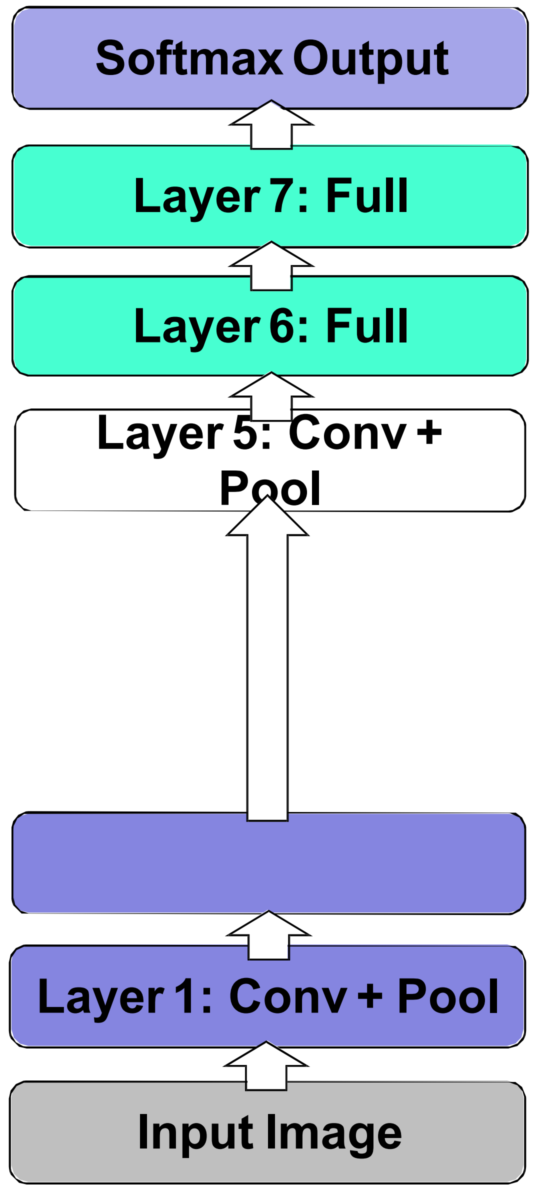
How important is depth?

Now try removing upper feature extractor layers:

- Layers 3 & 4

Drop ~1 million parameters

3.0% drop in performance



How important is depth?

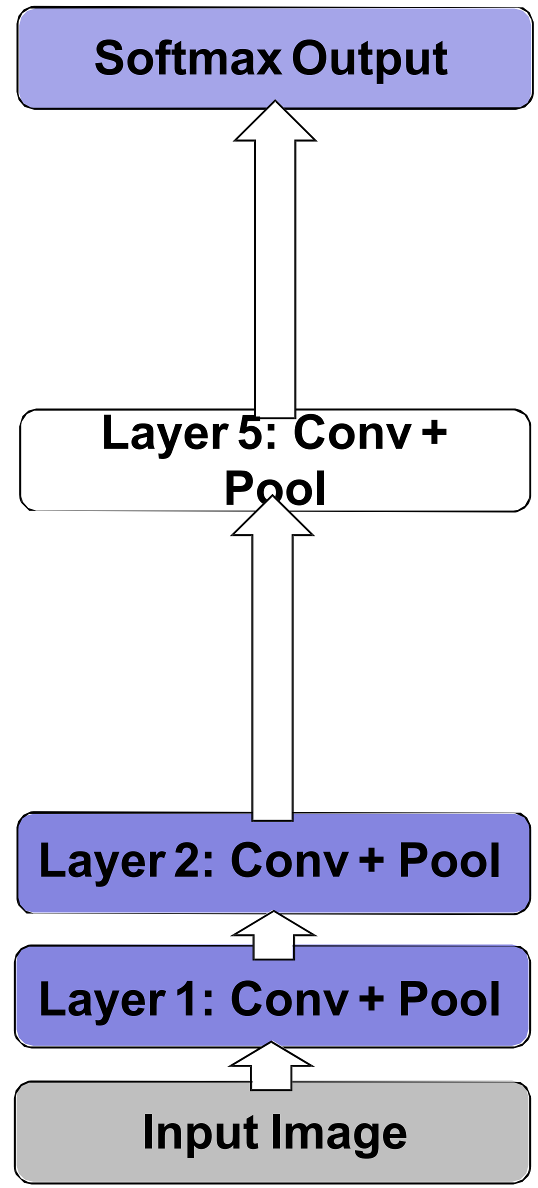
Now try removing upper feature extractor layers & fully connected:

- Layers 3, 4, 6 ,7

Now only 4 layers

33.5% drop in performance

✳Depth of network is key



Tapping off Features at each Layer

Plug features from each layer into linear SVM or soft-max

	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)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

CNN Libraries (open source)

- Cuda-convnet (Google): C/C++, Python
- Caffe (Berkeley): C/C++, Matlab, Python
- Overfeat (NYU): C/C++
- TensorFlow (Google): C++, Python
- Torch: Python
- ConvNetJS: Java script
- MatConvNet (VLFeat): Matlab
- DeepLearn Toolbox: Matlab

Using CNN Features on Other Datasets

- Take model trained on, e.g., ImageNet 2012 training set
- Take outputs of 6^{th} or 7^{th} layer before or after nonlinearity
- Classify test set of new dataset
- Optional: fine-tune features and/or classifier on new dataset

Results on misc. 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

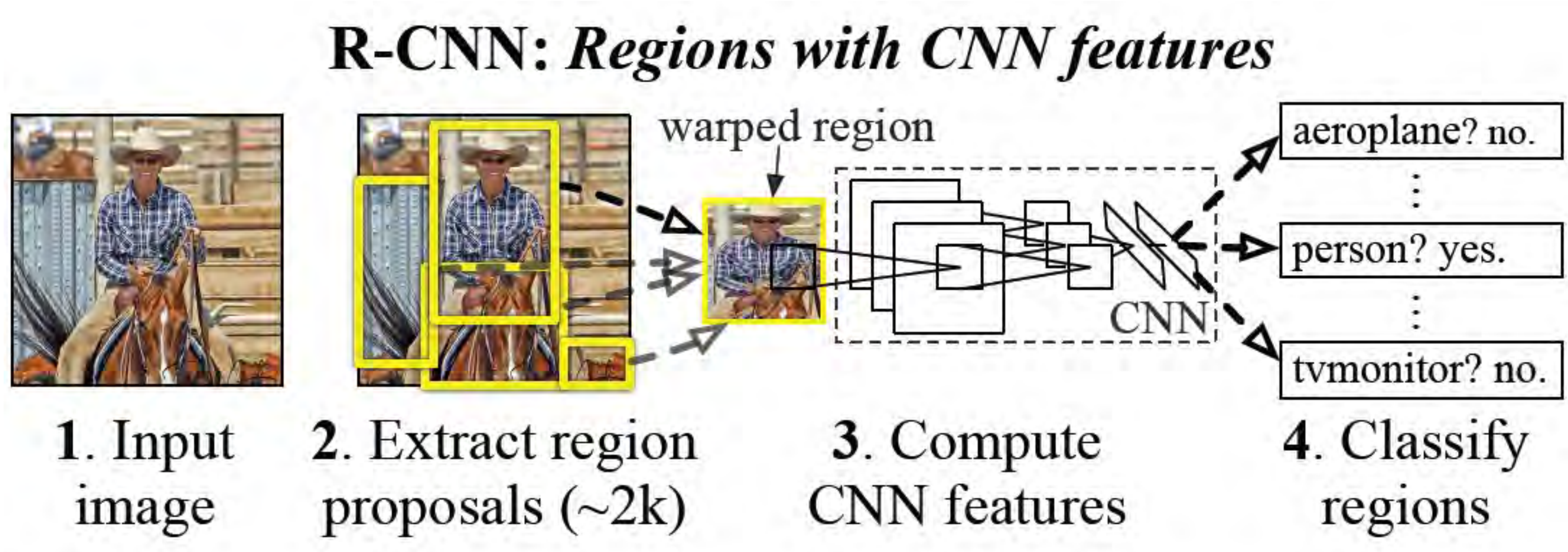
[2] MIT-67 Indoor Scenes dataset (OverFeat)

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

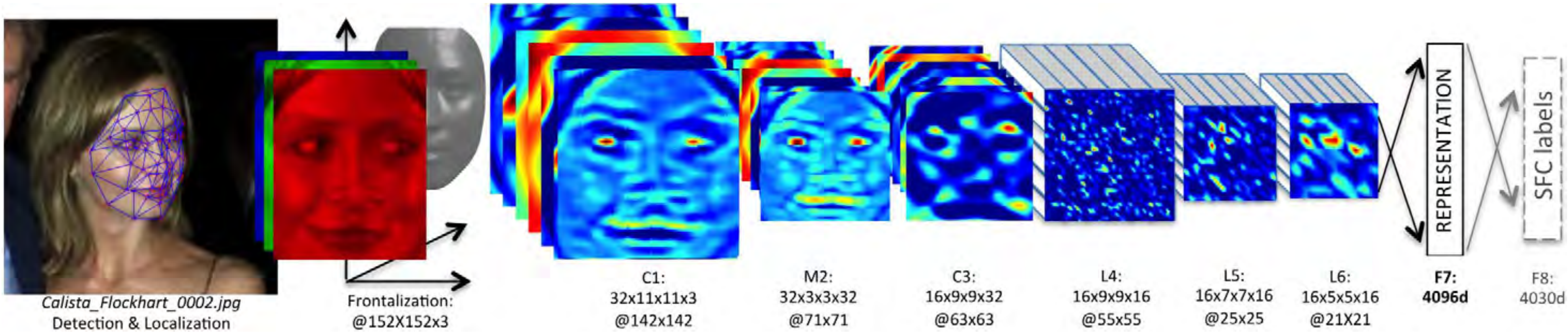
CNN features for detection

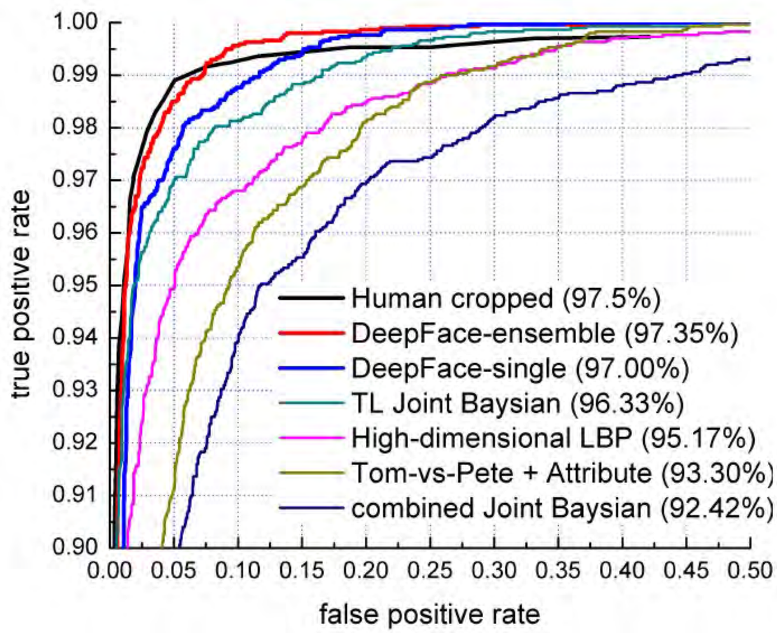


Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. **R-CNN achieves a mean average precision (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. **The popular deformable part models perform at 33.4%.**

R. Girshick, J. Donahue, T. Darrell, and J. Malik, [Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation](), CVPR 2014, to appear.

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.

References : CNNs

- Classical paper of CNN (LeNet)
  - Y. LeCun, L. Bottou, Y. Bengio, P. Huffier, "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11), 2278-2324. 1998.
- Stochastic Gradient Descent
  - G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," Neural Computation, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
- Dropout
  - N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
- Transfer learning and Pretrain
  - Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," JMLR W&CP: Proc. Unsupervised and Transfer Learning, 2012.

References : R-CNN

- R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 1, pp. 142–158, 2016.
- S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Advances in Neural Information Processing Systems, 2015, pp. 91–99. 2015.