COMPSCI 589
Lecture 6: Neural Networks and Deep Learning

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

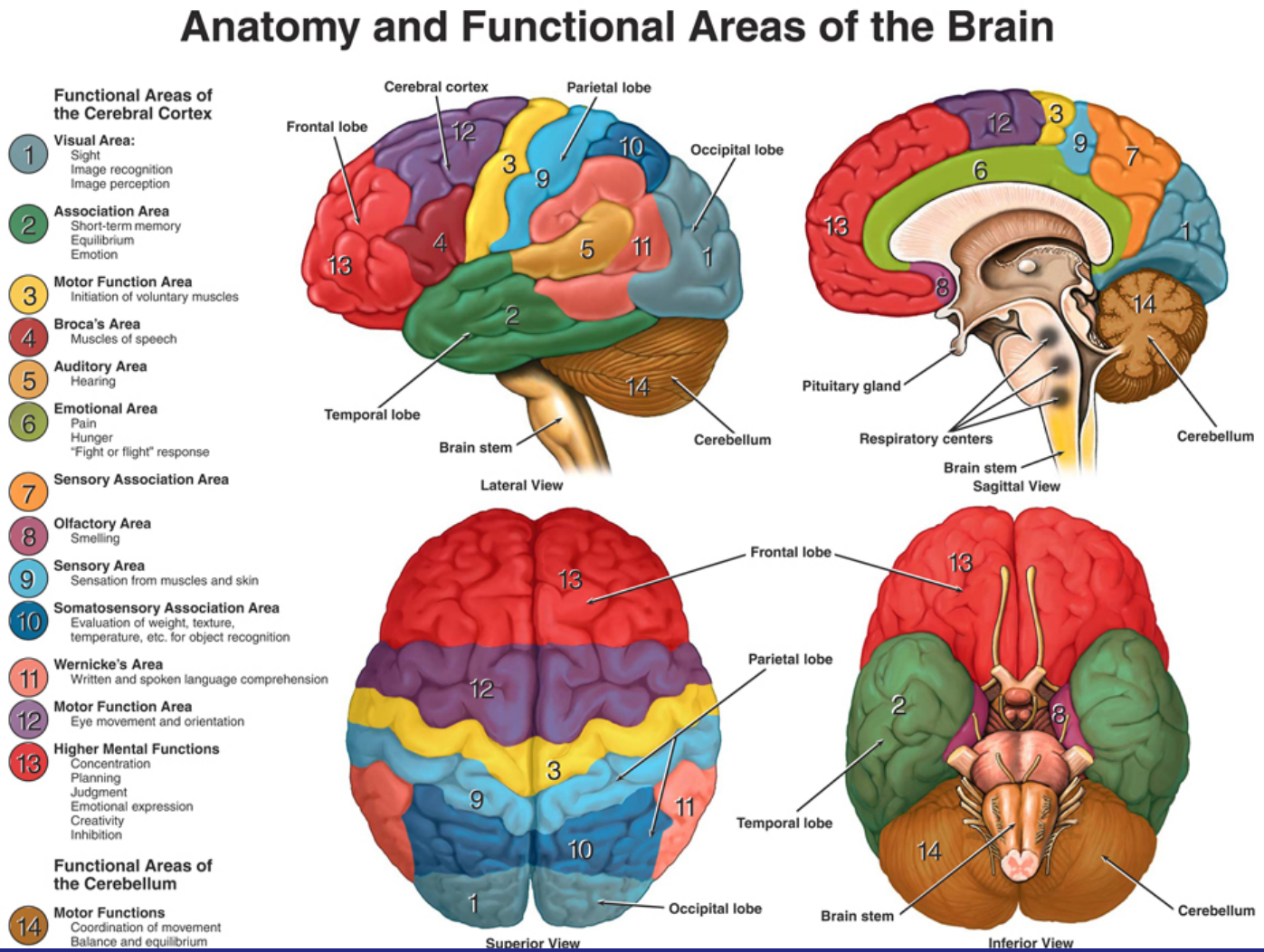
Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

Overview

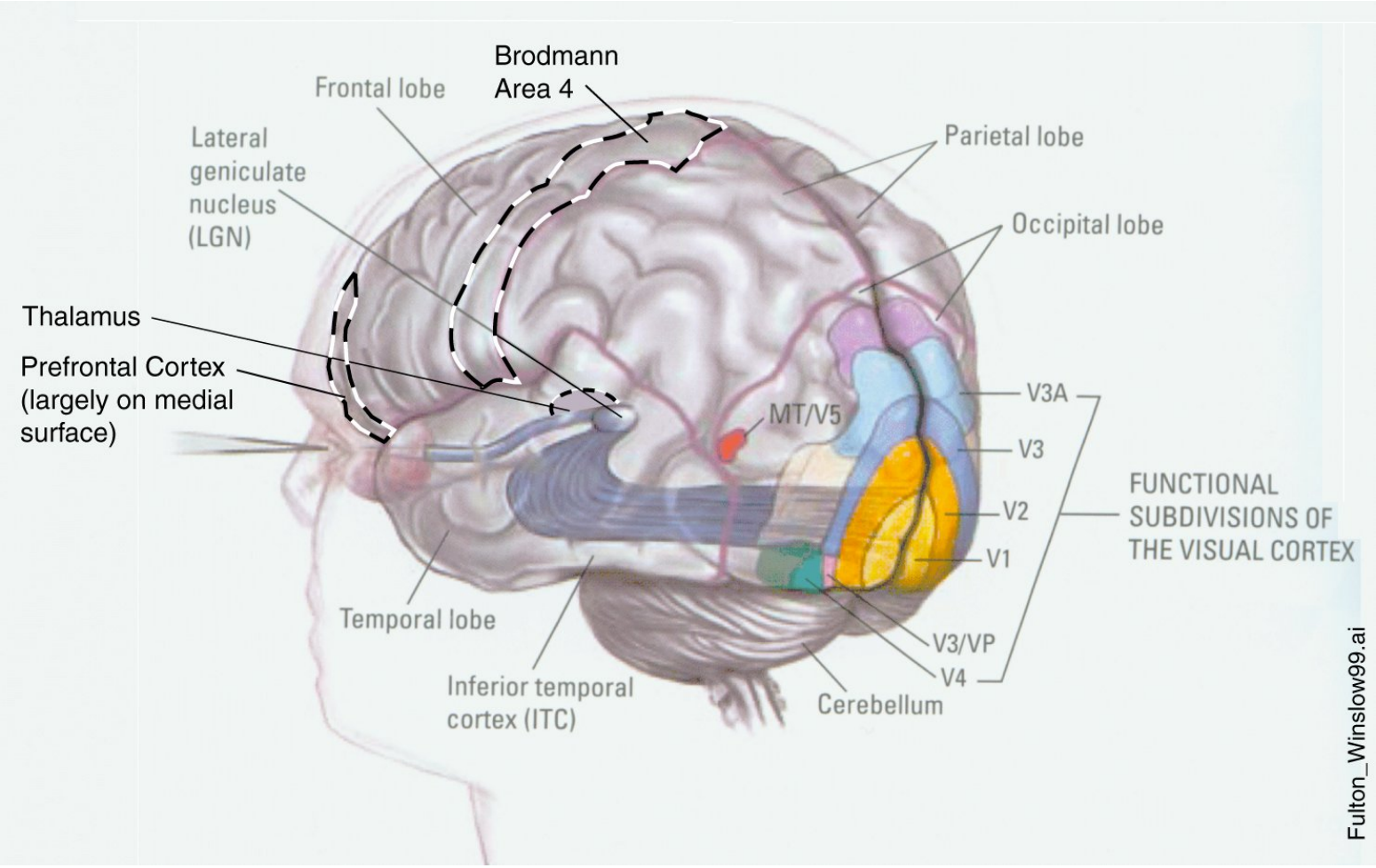
- Last class we saw how basis expansion and kernels could be used to increase the capacity of linear models in a controlled way.
- **Question:** What is the primary weakness of this approach?
- In this lecture, we'll see how artificial neural networks can learn appropriate feature representations along with decision boundaries to maximize classification accuracy.
- We'll start with the inspiration for artificial neural networks: your brain.

Your Brain

Brain Regions and Functions

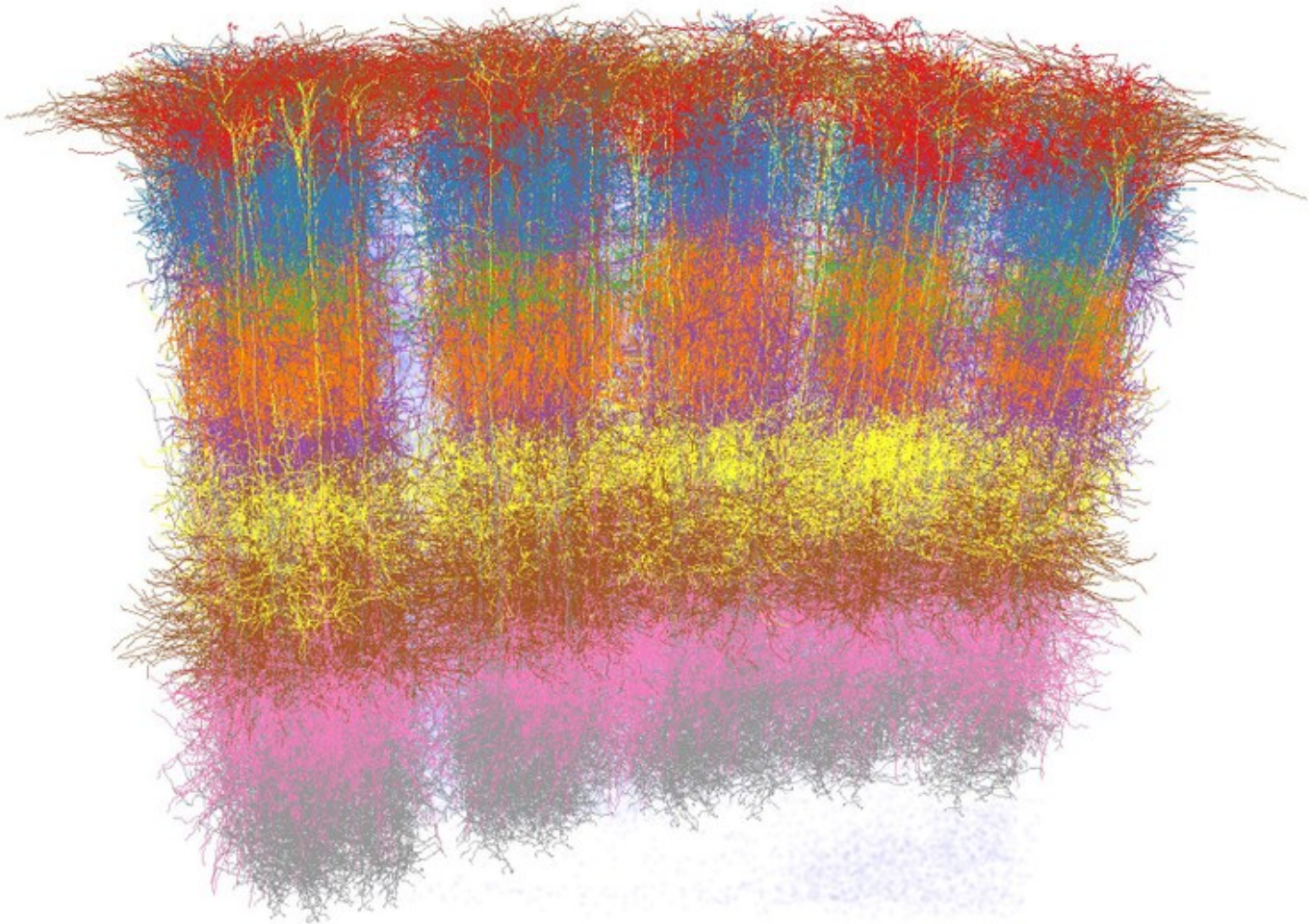


Visual Areas

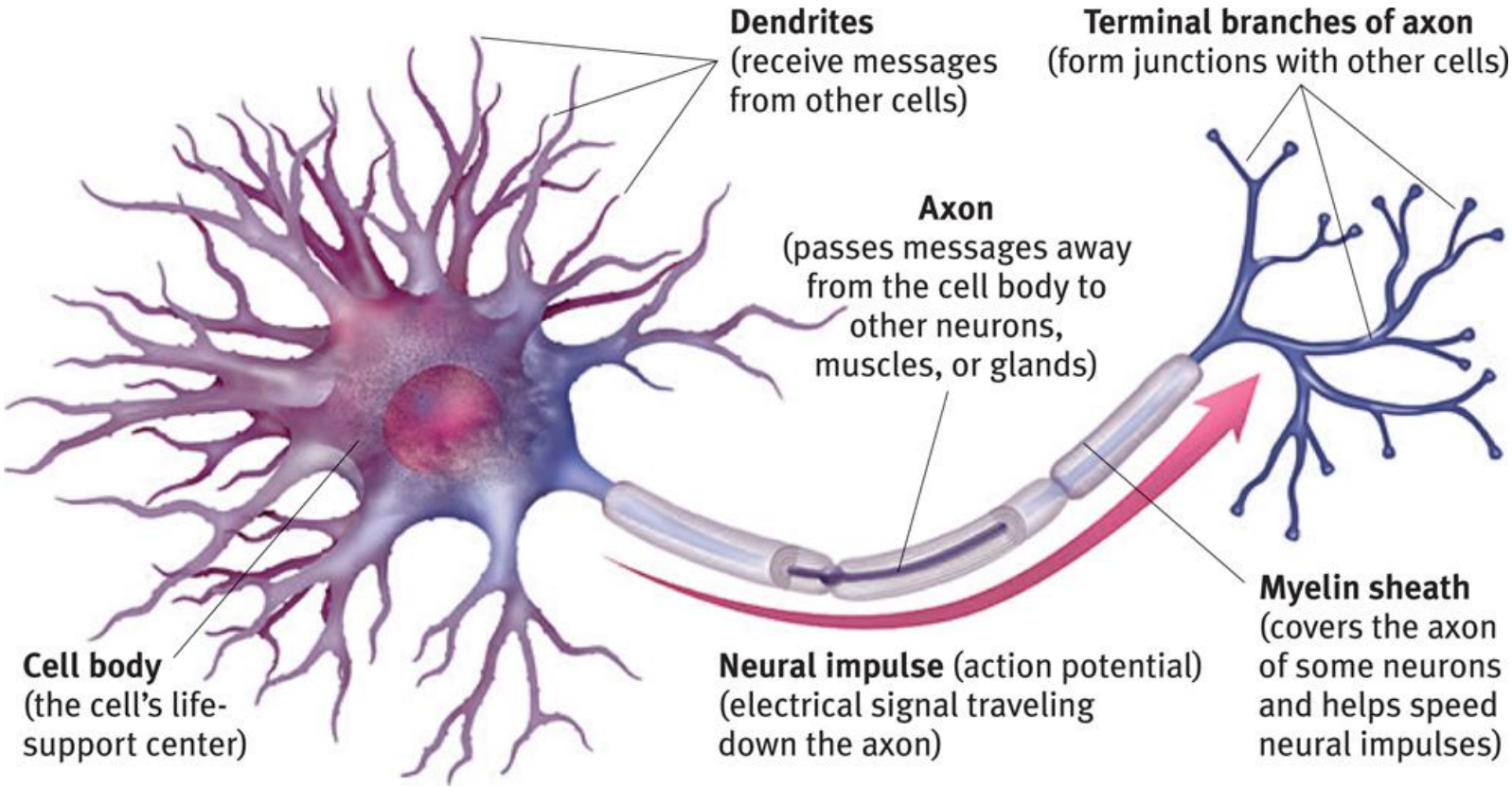


Optic nerve contains about 1 Million nerve fibers.

Cortical Columns

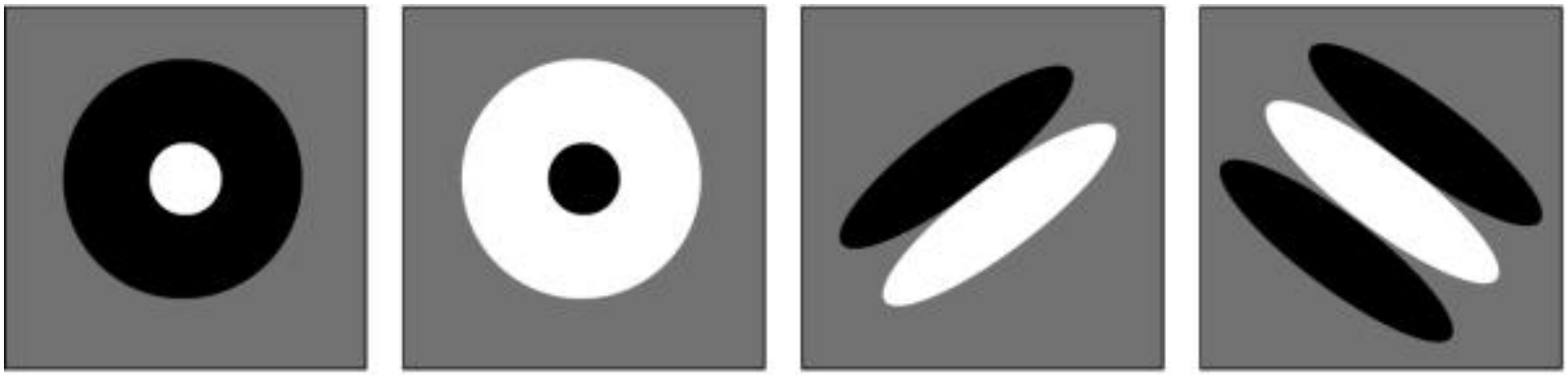


Neurons



Your brain contains about 100 Billions neurons.

Visual Receptive Fields

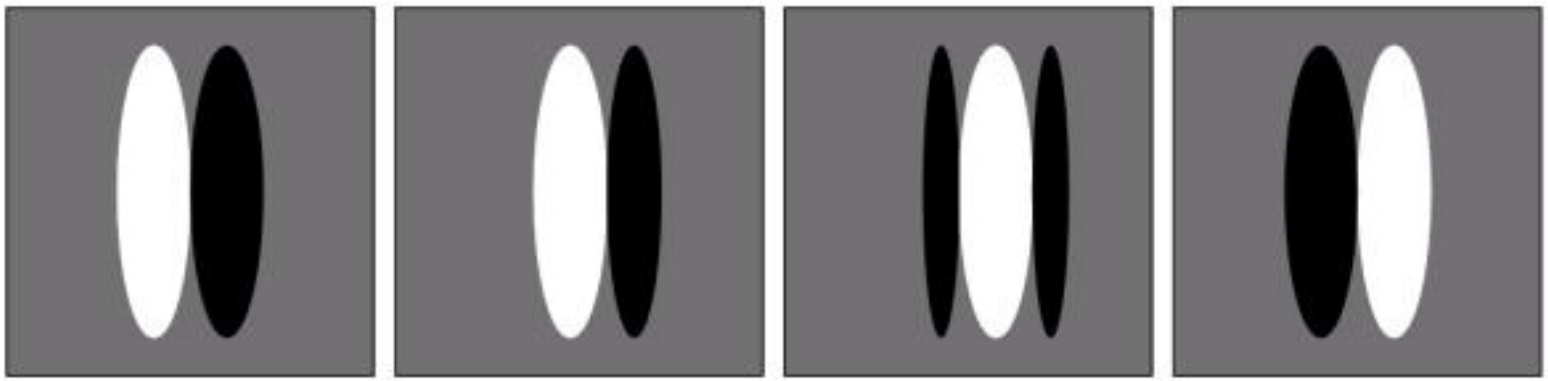


(*a*) ON cell in retina or LGN

(*b*) OFF cell in retina or LGN

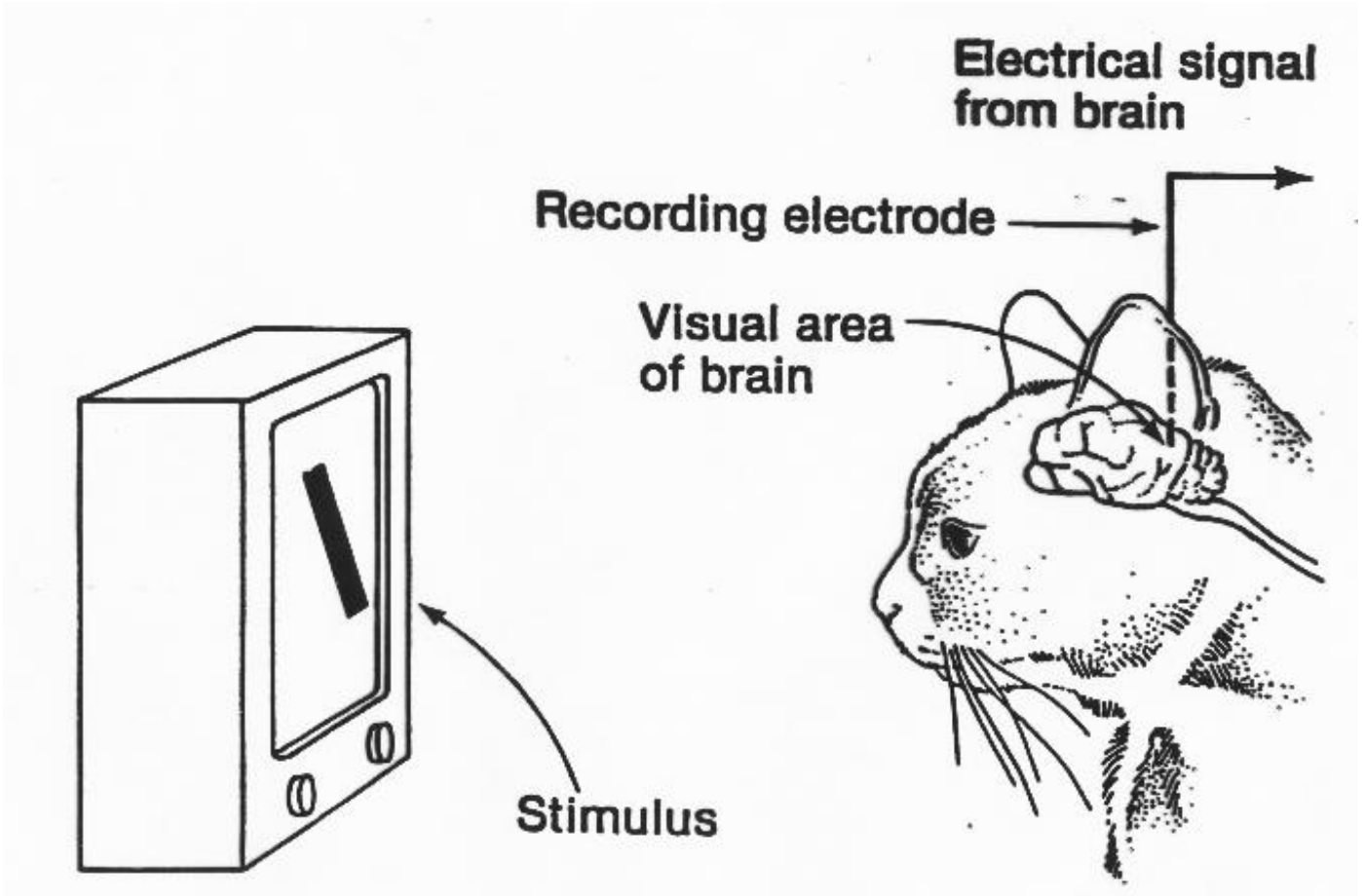
(*c*) 2-lobe V1 simple cell

(*d*) 3-lobe V1 simple cell



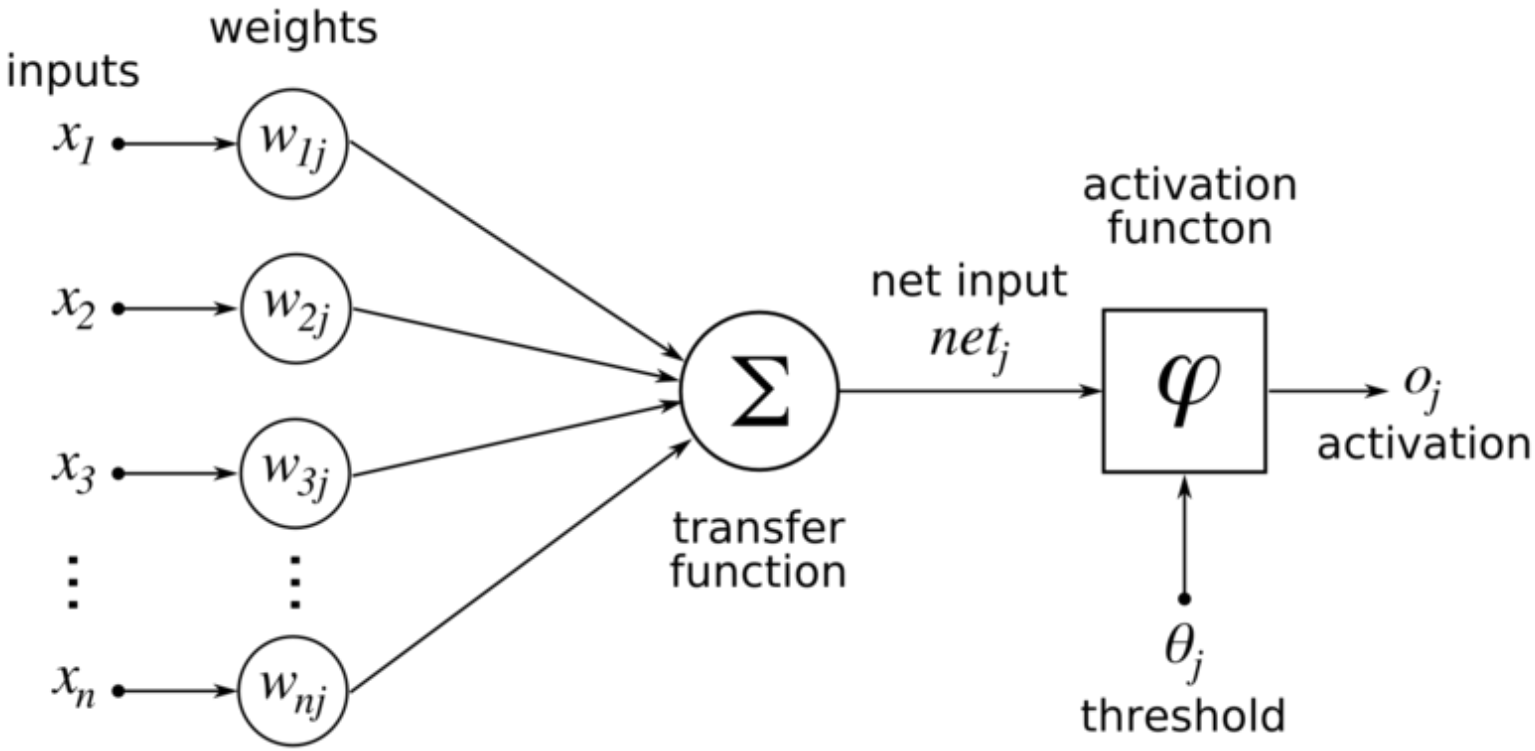
(*e*) Spatiotemporal RF of a V1 cell

Mapping Receptive Fields in Live Subjects



Hubel and Wiesel earned the Nobel Prize in Physiology or Medicine in 1981 for this work.

McCulloch and Pitts Neuron (1943)



Assuming $\varphi() = \text{sign}()$, what model is this?

The Perceptron (1950)

The Perceptron is a simple online algorithm for adapting the weights in a McCulloch/Pitts neuron. It was developed in the 1950s by Rosenblatt at Cornell.

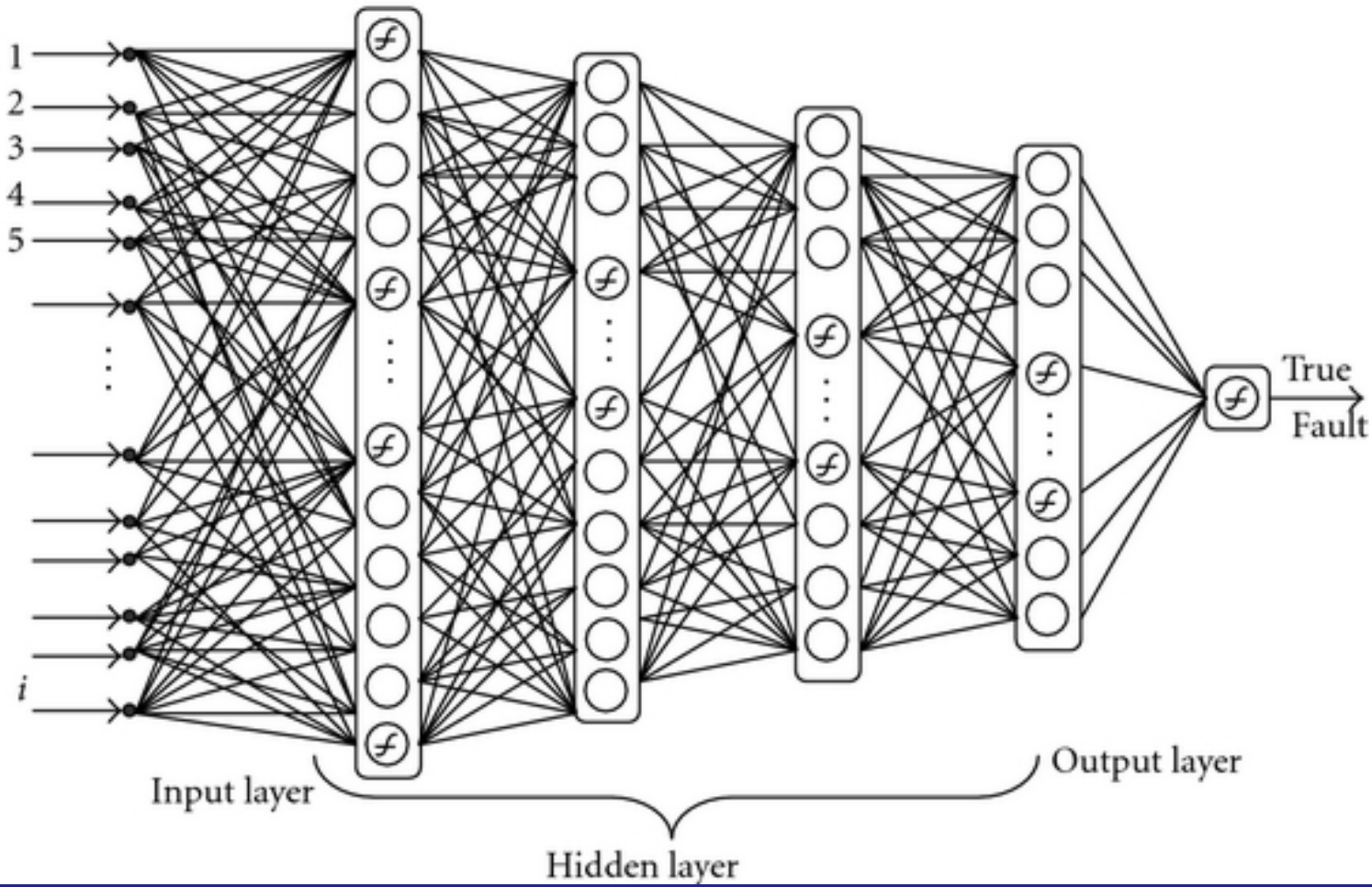
Algorithm PERCEPTRONTRAIN(**D**, *MaxIter*)

$w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \ldots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \ldots, w_D, b$

Limitations of Single Layer Perceptrons

- The representational limitations of the single-layer perceptron were not well understood at first in the AI community.
- In 1969, Minsky and Papert at MIT popularized a set of arguments showing that the single-layer perceptron could not learn certain classes of functions (including XOR).
- They also showed that more complex functions could be represented using a *multi-layer perceptron* or MLP, but no one knew how to learn them from examples.
- This led to a shift away from mathematical/statistical models in AI toward logical/symbolic models.

Multi-Layer Perceptron



Sigmoid Neural Networks

The solution to MLP learning turned out to be:

1. Make the hidden layer non-linearities smooth (sigmoid/logistic) functions:

$$h_k^{(1)} = \frac{1}{1 + \exp(-(\sum_d w_{dk} x_d + b_{dk}))} \quad (1)$$

$$h_k^{(i)} = \frac{1}{1 + \exp(-(\sum_l w_{lk} h_l^{(i-1)} + b_{lk}))} \quad (2)$$

2. Make the output layer non-linearity a smooth (sigmoid/logistic/softmax) function.

3. Use standard numerical optimization methods (gradient descent) to learn the parameters. The algorithm is known as Backpropagation and was popularized by Rumelhart, Hinton and

Sigmoid Neural Network Properties

- It can be shown that a sigmoid network with one hidden layer (of unbounded size) is a universal function approximator.
- In terms of classification, this means neural networks with one hidden layer (of unbounded size) can represent any decision boundary and thus have infinite capacity.
- It was also shown that deep networks can be exponentially more efficient at representing certain types of functions than shallow networks.
- Demo!

More Failures

- Through the 1990s is became clear that while these models could represent arbitrarily complex functions and that deep networks should work better than shallow networks, no one could effectively train deep networks.
- This led to a shift away from neural networks towards probabilistic/statistical models and SVMs through the late 1990s and 2000s.
- Only a few groups continued to work on neural network models during this time period (Hinton, LeCun, Bengio, Ng).

Deep Learning

The solution to the deep learning problem (as of right now) appears to be:

1. Have access to lots of labeled data (ie: millions of examples).
2. Make the non-linearities non-smooth again (rectified linear units, ReLU):

$$h_k^{(1)} = \max(0, \sum_d w_{dk} x_d + b_{dk})) \tag{3}$$

$$h_k^{(i)} = \max(0, \sum_l w_{lk} h_l^{(i-1)} + b_{lk})) \tag{4}$$

Deep Learning

The solution to the deep learning problem (as of right now) appears to be:

1. Have access to lots of labeled data (ie: millions of examples).
2. Make the non-linearities non-smooth again (rectified linear units, ReLU):

$$h_k^{(1)} = \max(0, \sum_d w_{dk} x_d + b_{dk})) \tag{5}$$

$$h_k^{(i)} = \max(0, \sum_l w_{lk} h_l^{(i-1)} + b_{lk})) \tag{6}$$

Deep Learning

The solution to the deep learning problem (as of right now) appears to be:

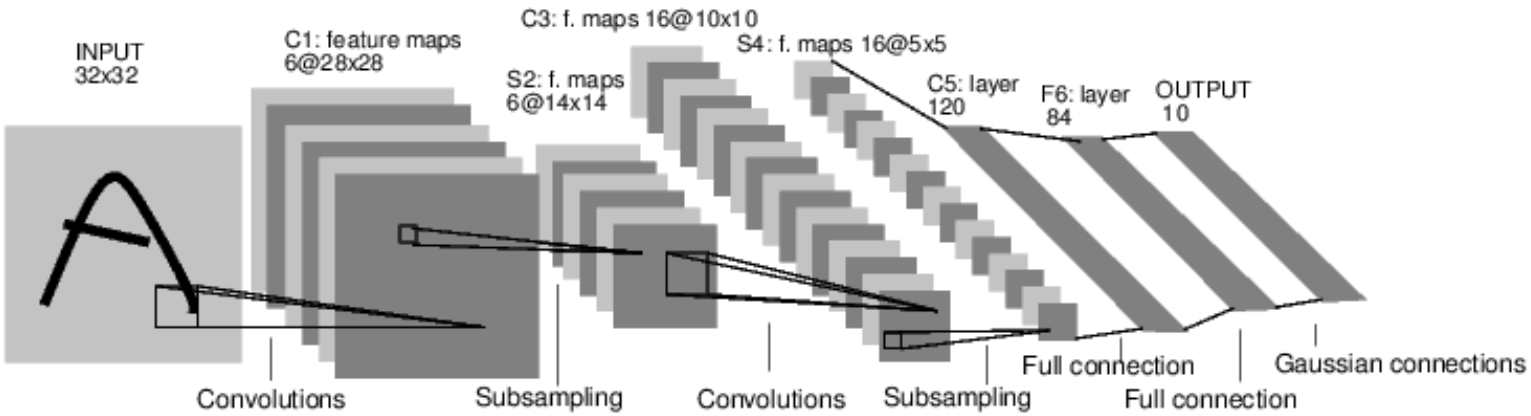
3. Use somewhat improved optimization methods for non-smooth functions (accelerated stochastic sub-gradient descent) to learn the parameters.
4. Use new regularization schemes based on randomly zeroing-out nodes in the network.
5. Do the computing on a GPU with 1000s of cores and 10s of GB of RAM for a massive 20-30X speedup (go SIMD!). Model training takes 10 days for large vision problems instead of 1 year.

Deep Learning Applications

- Deep learning models using these basic ingredients have become dominant in AI over the last few years starting with computer vision and moving on to other areas including speech recognition.
- Many companies use deep learning-based vision systems to analyze photos and images.
- Google and others switched to deep-learning based speech recognition systems after it was shown to lead to substantial reductions in the word error rate.

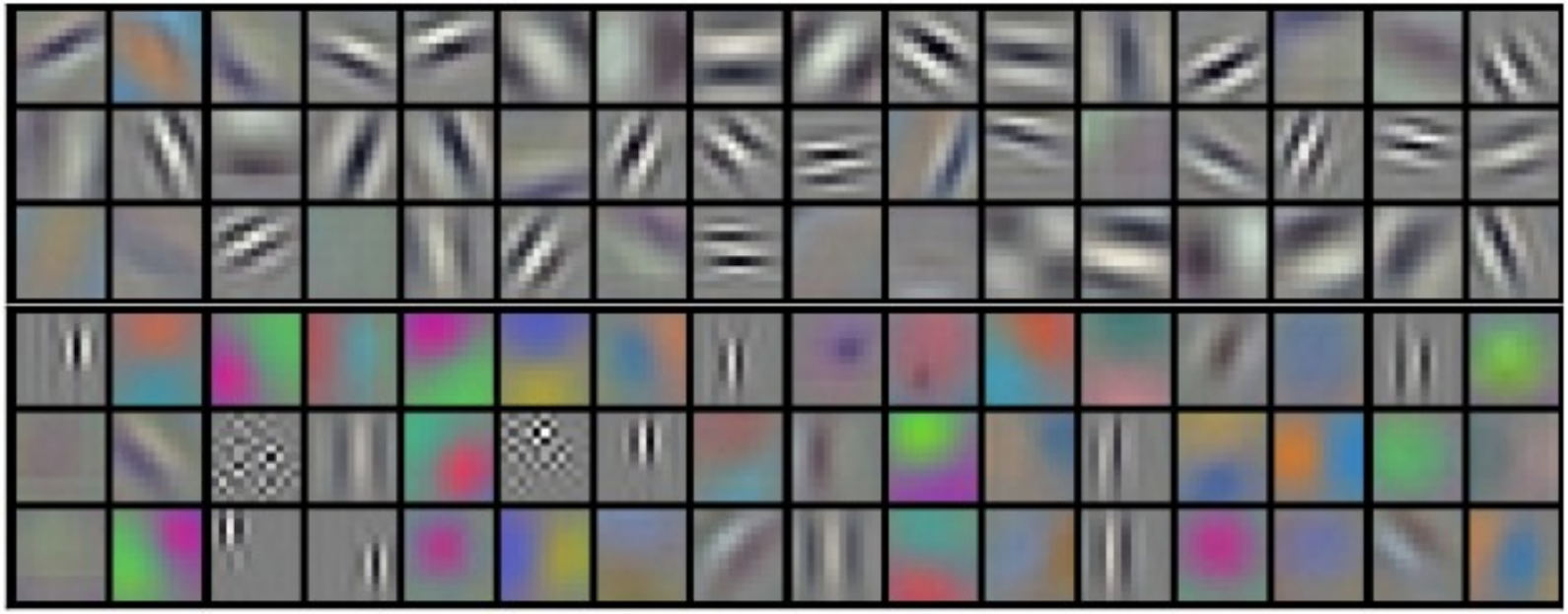
Deep Learning For Vision

Deep learning for vision uses a modified deep architecture called a *convolutional network* or convnet that learns small patches of weights (or filters) and replicates (convolves) them over an image.

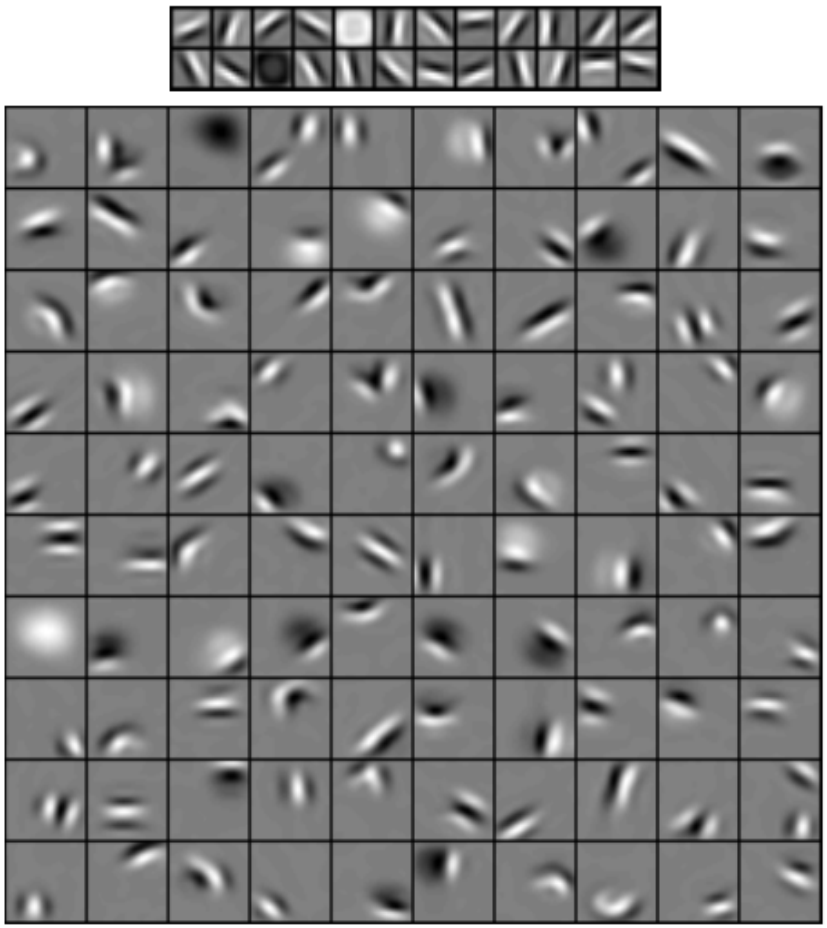


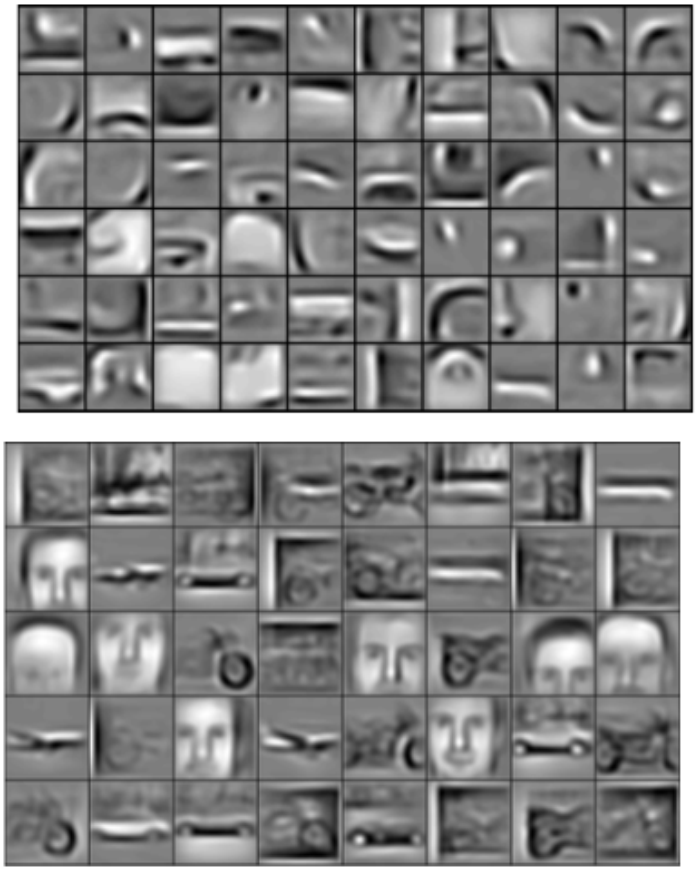
This architecture is also from the 1980s and was directly inspired by the structure of the visual areas of the brain.

Learned Receptive Fields



Learned Receptive Fields





Deep Learning Demos

- Digit Recognition: `http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html`
- Image Classification: `http://www.clarifai.com/#demo`
- Image Classification: `http://deeplearning.cs.toronto.edu/`
- Image Description: `http://deeplearning.cs.toronto.edu/i2t`
- Handwriting Generation: `http://www.cs.toronto.edu/~graves/handwriting.cgi`