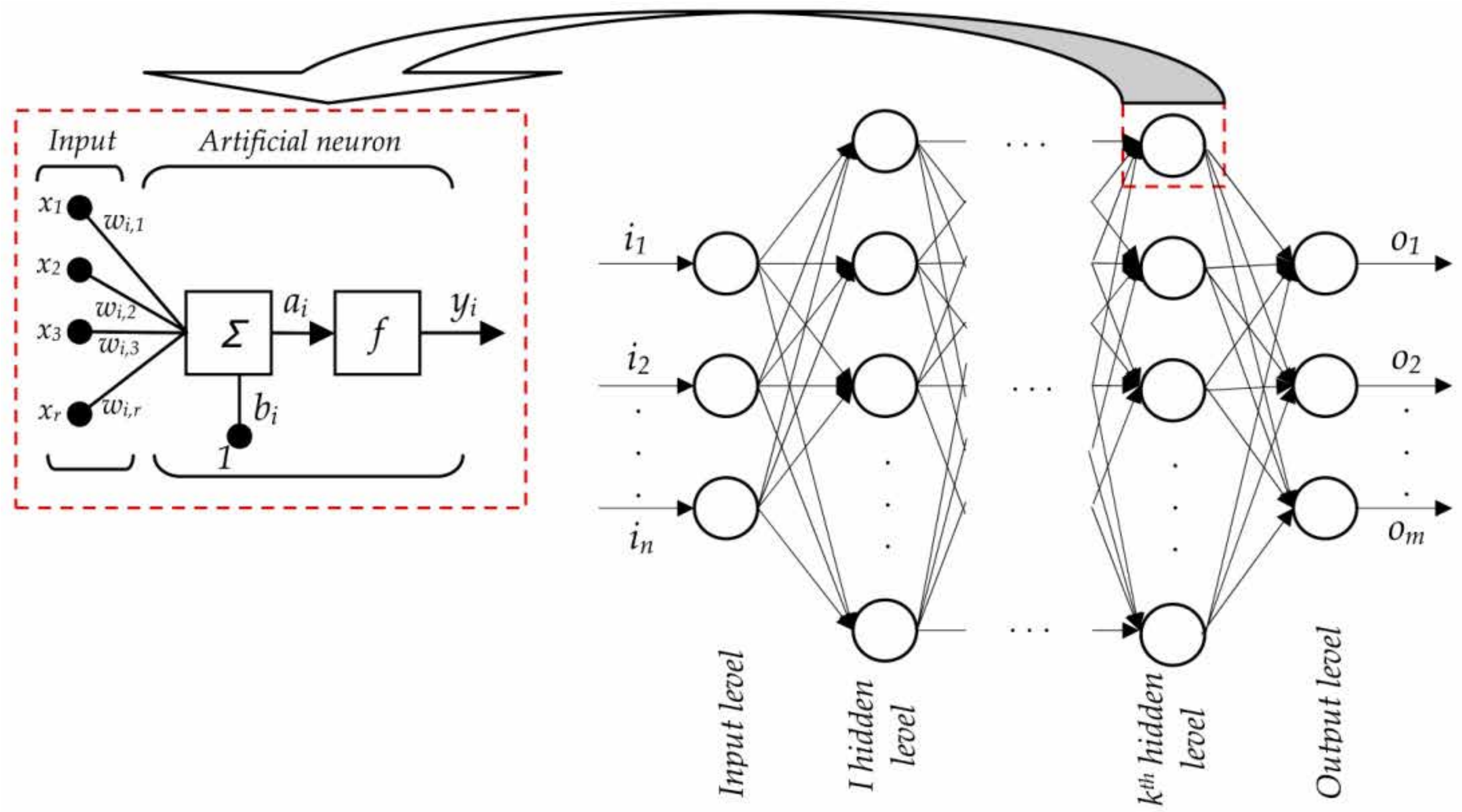
CS 466/566
Introduction to Deep Learning

Lecture 4

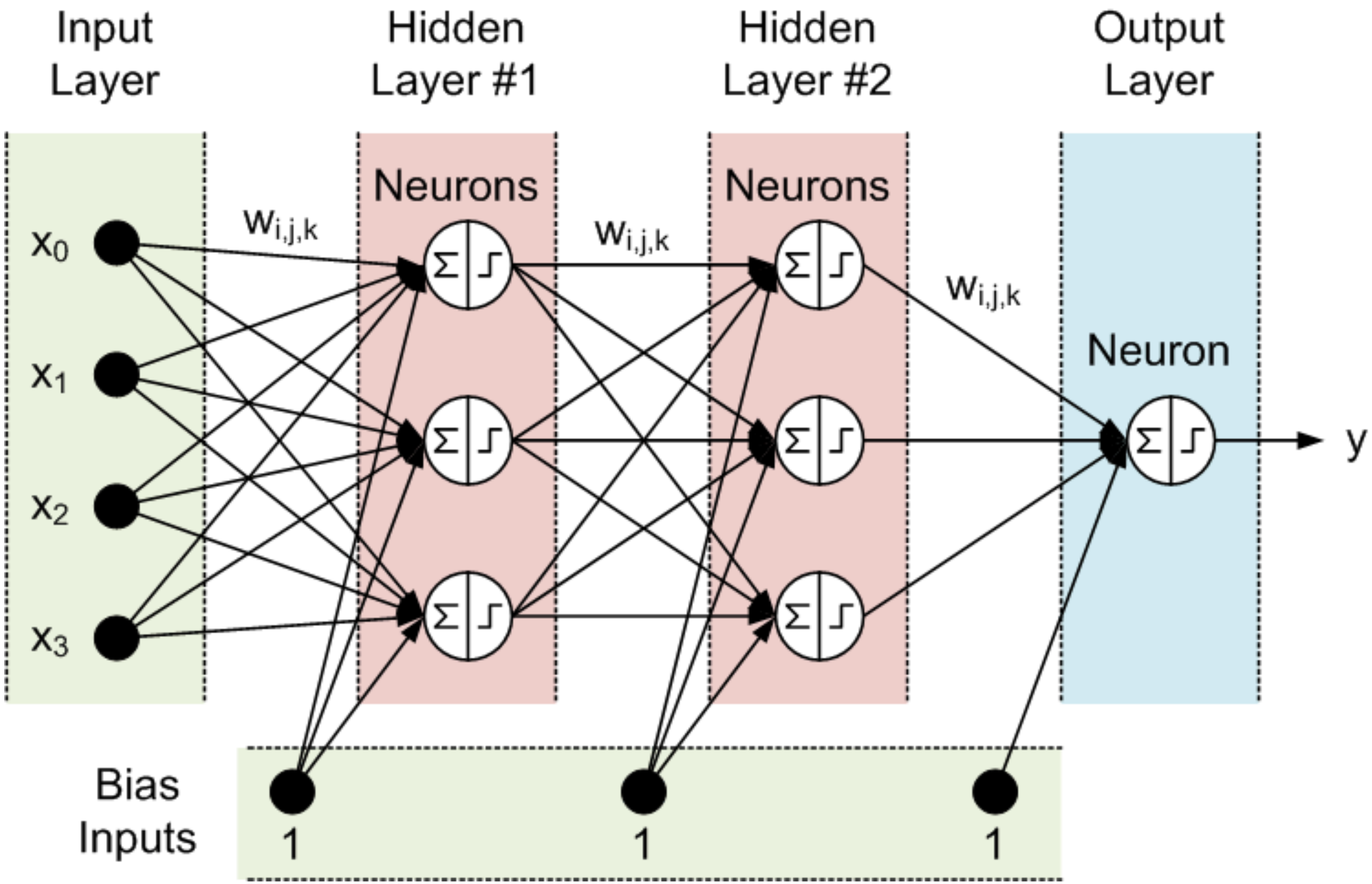
Introduction to Deep Neural Networks - Part 2

Compiled from various resources

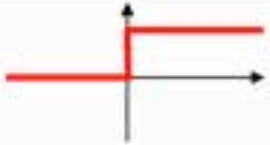
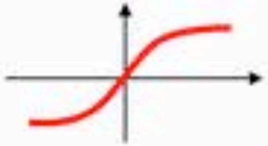
Recall #1: Neurons are **sometimes** line models



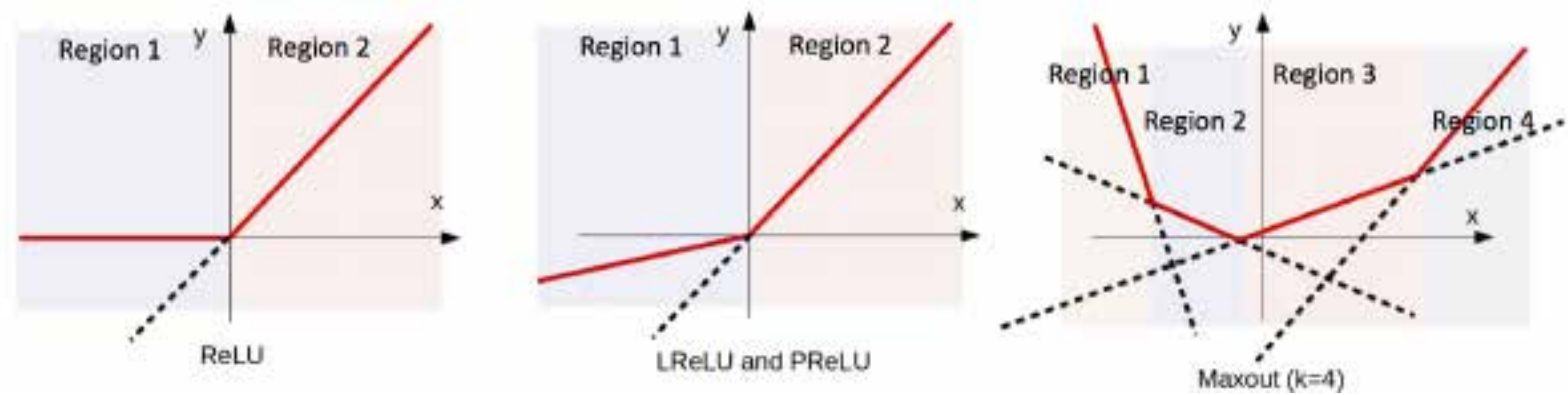
Recall #2: Bias of the line can be represented as a weight of a fake input that is always 1.



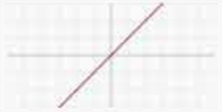
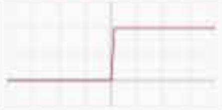
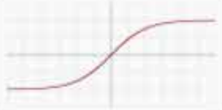
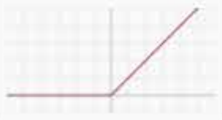
Racall #3: Activation Functions: some older ones

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Racall #3: Activation Functions: some newer ones

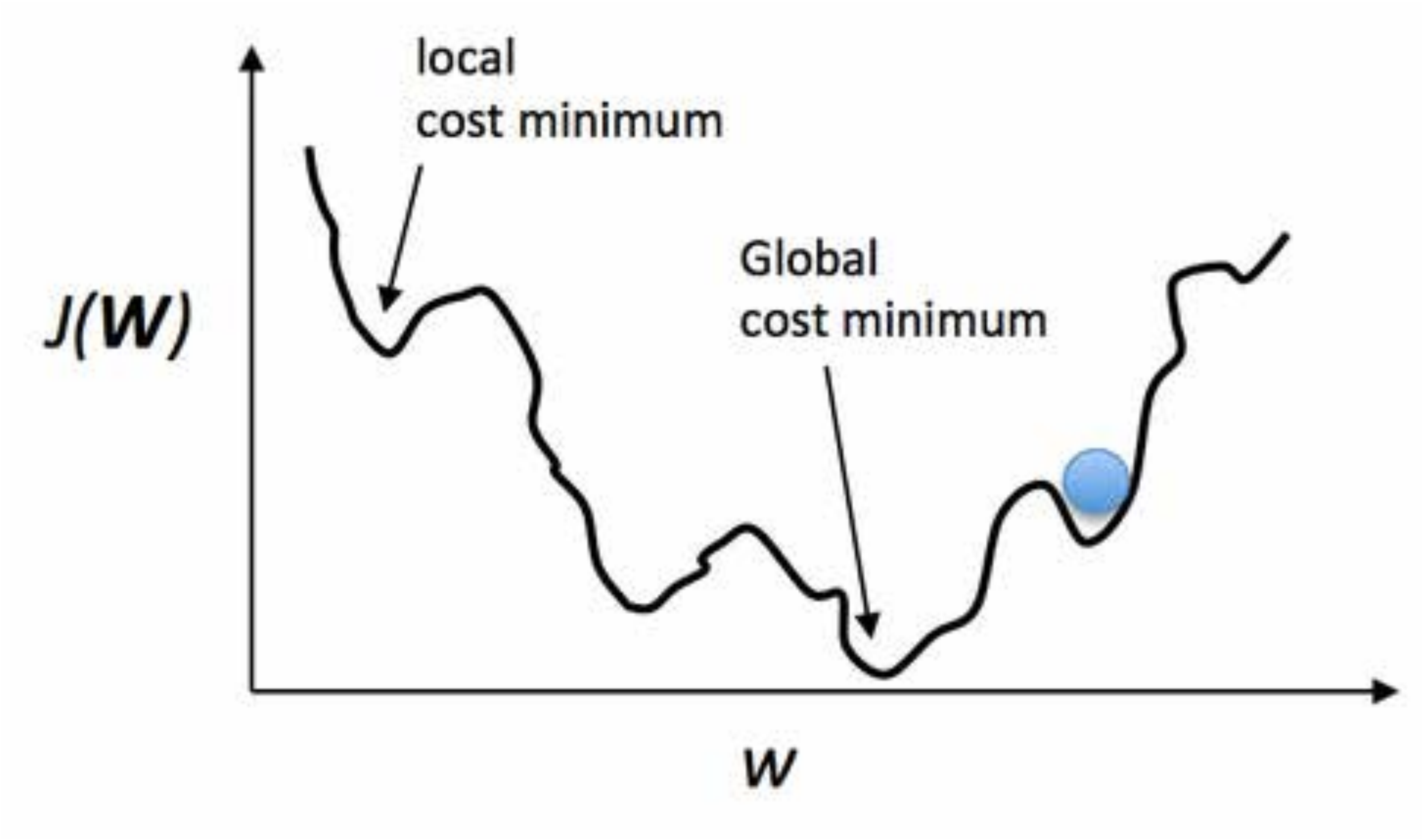


Racall #3: Activation functions

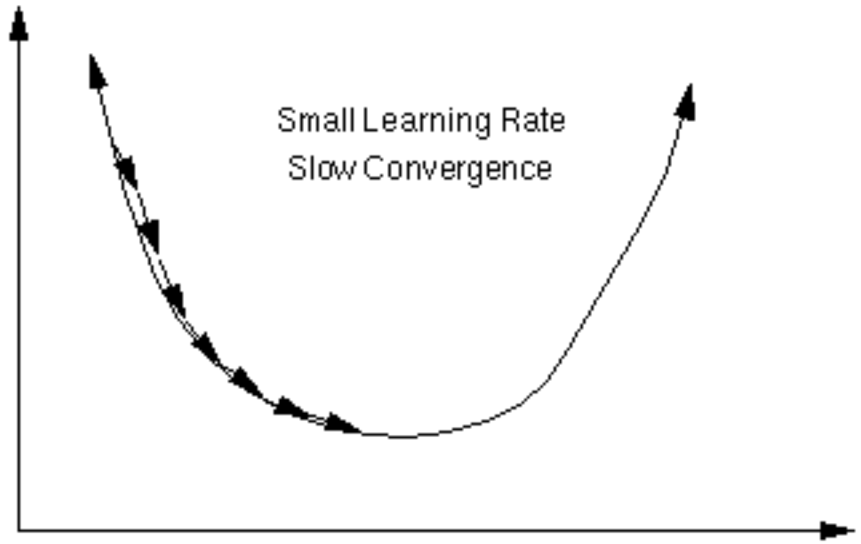
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

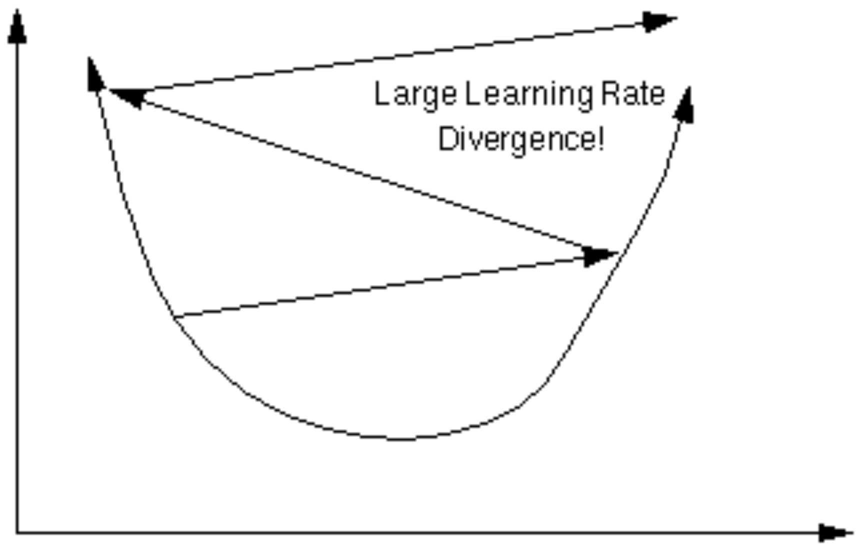
- Activation functions are the things that give non-linear separation power to Neural Networks.
- Do not confuse activations of hidden layer neurons and logistic regression.
- For a binomial (two class) classification problem, we may choose sigmoid function both for hidden layer activation and output layer at the same time.
- This doesn't mean that they are put there for the same purpose.

Recall #4: Global vs Local Cost Optimum

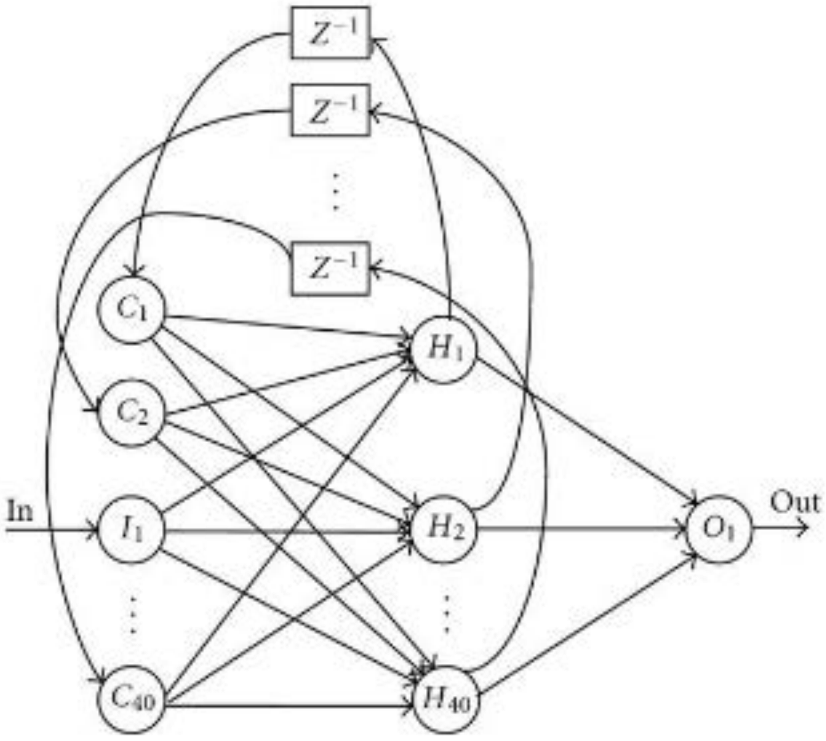


Recall #5: Learning Rate

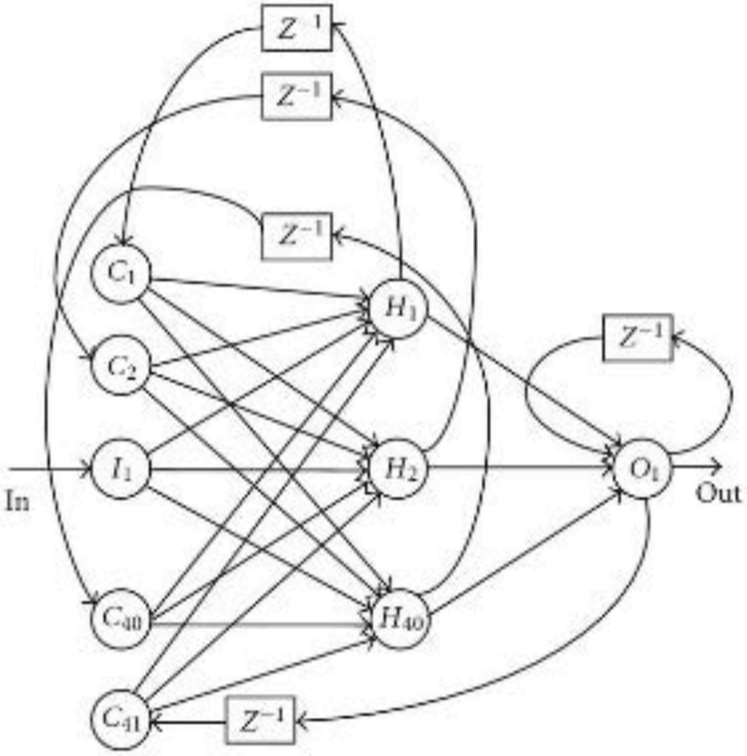




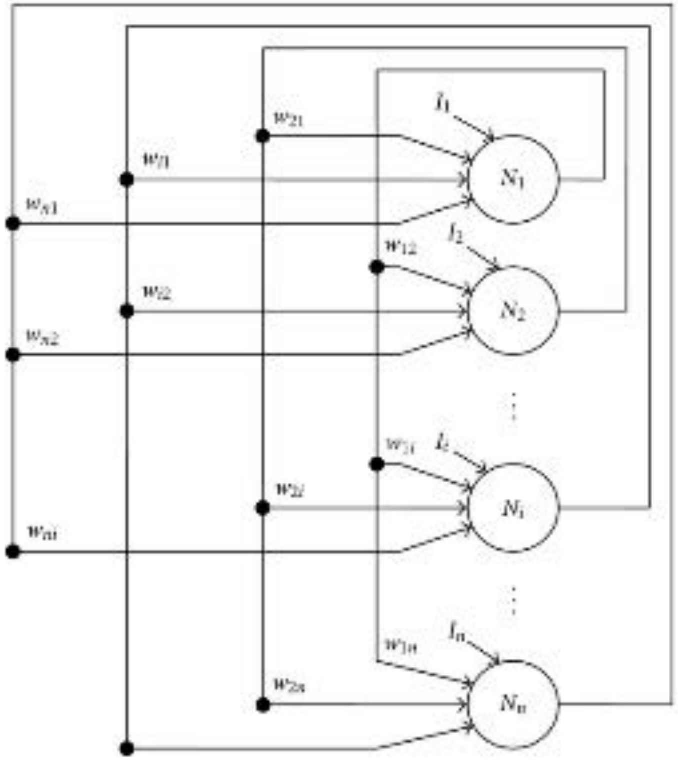
Recall #6: NNs have Different Architectures



Elman Neural Network

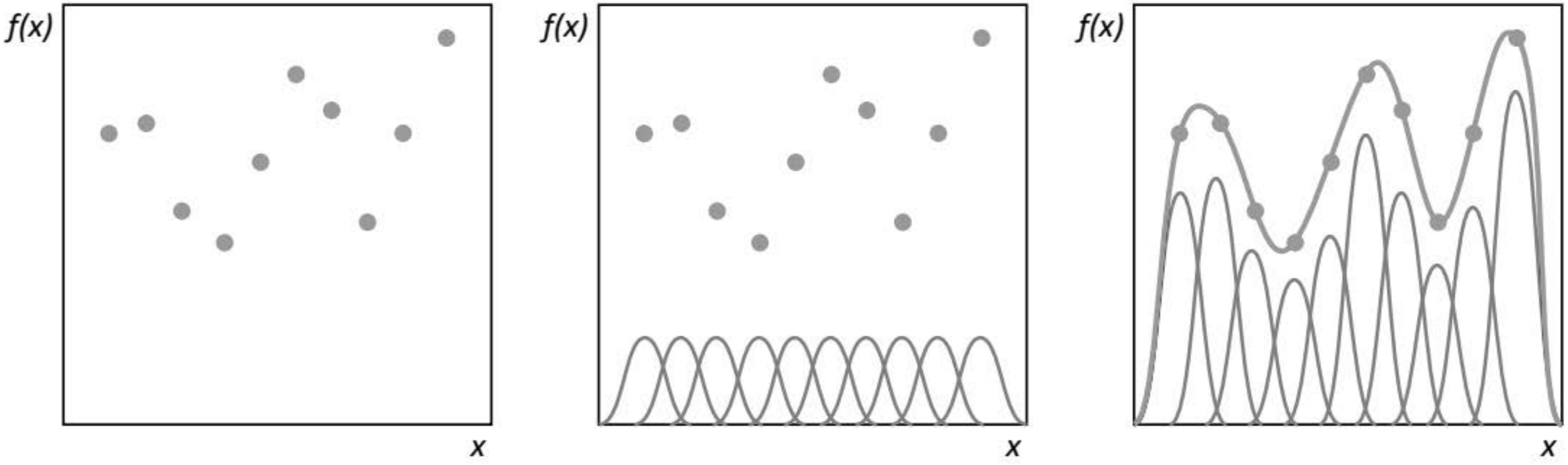


Jordan Neural Network



Hopfield Neural Network

For instance: Radial Basis Networks (Neuron activations are Radial Basis Functions)



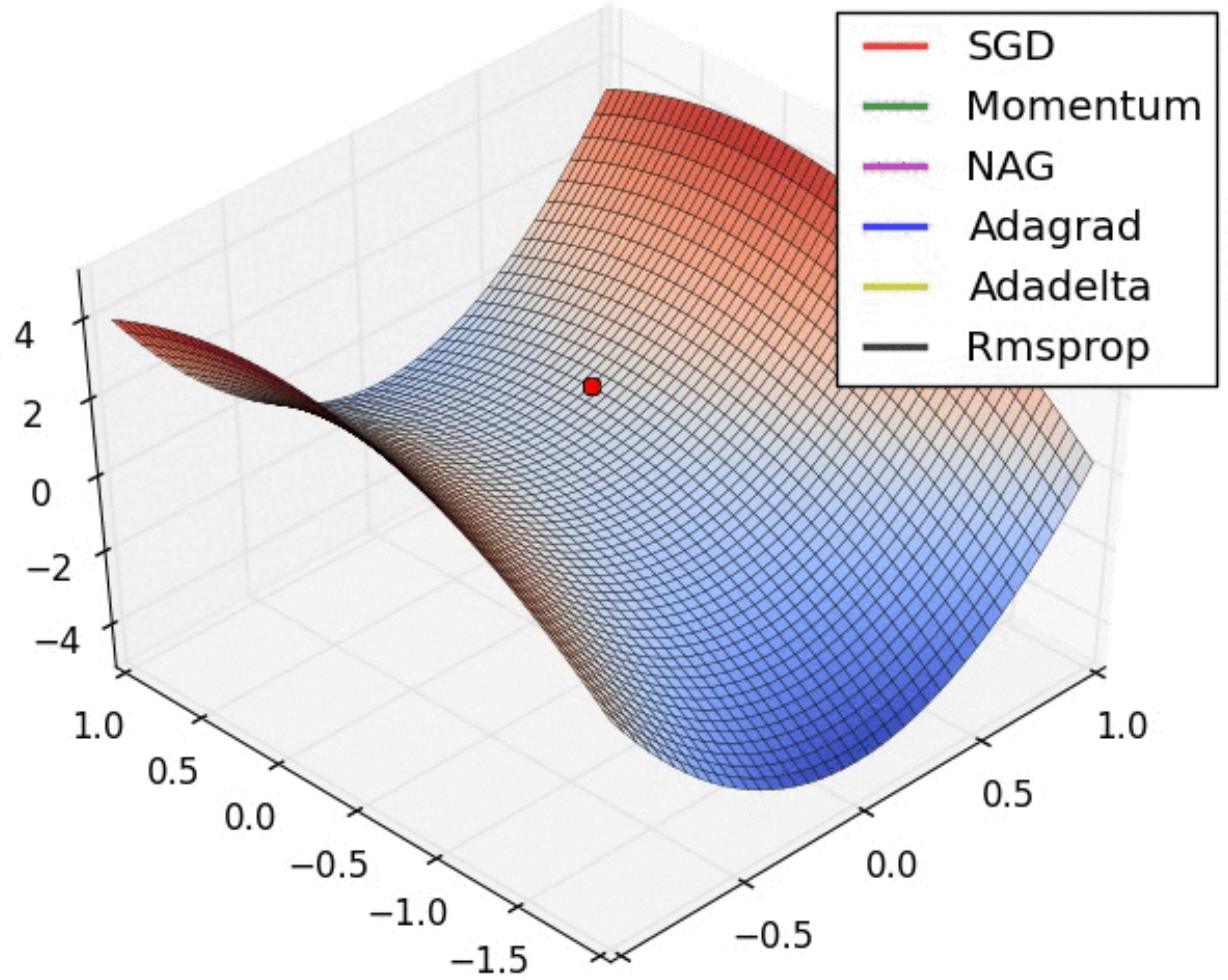
Remember: Many training algorithms exist

- Backpropagation consists of two steps:
  - The feedforward pass - the training data set is passed through the network and the output from the neural network is recorded and the error of the network is calculated
  - Backward propagation - the error signal is passed back through the network and the weights of the neural network are optimized using gradient descent.

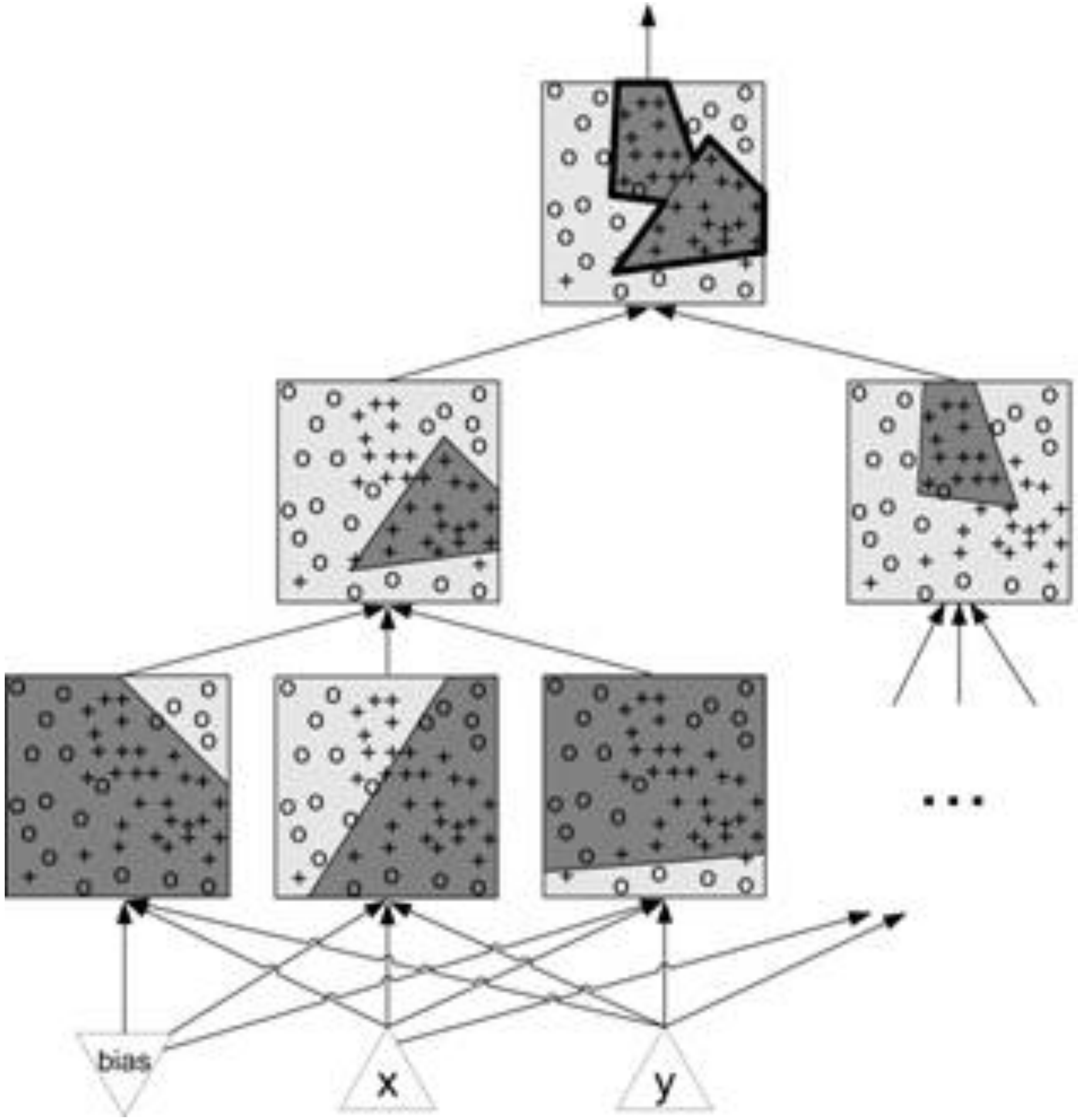
Many training algorithms exist

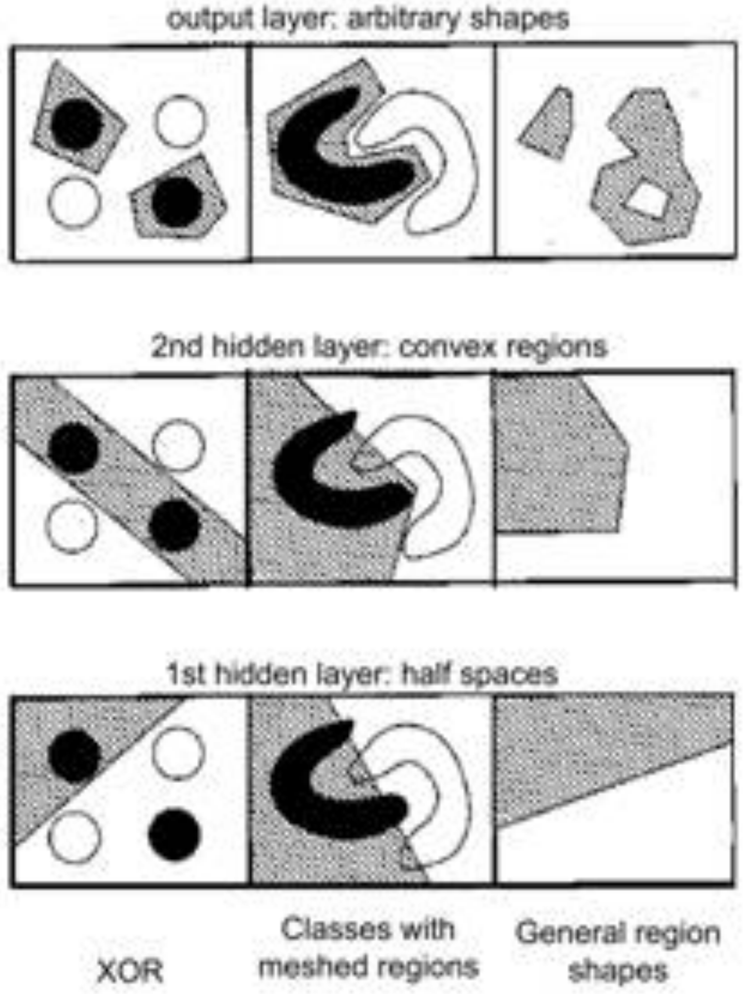
- Gradient Descent algorithm is quite slow, and is susceptible to local minima.
- Stochastic Gradient Descent (SGD)
- momentum gradient descent (QuickProp),
- Nesterov's Accelerated Momentum (NAG) gradient descent,
- the Adaptive Gradient Algorithm (AdaGrad),
- Resilient Propagation (RProp),
- Root Mean Squared Propagation (RMSProp),
- …

Many training algorithms exist

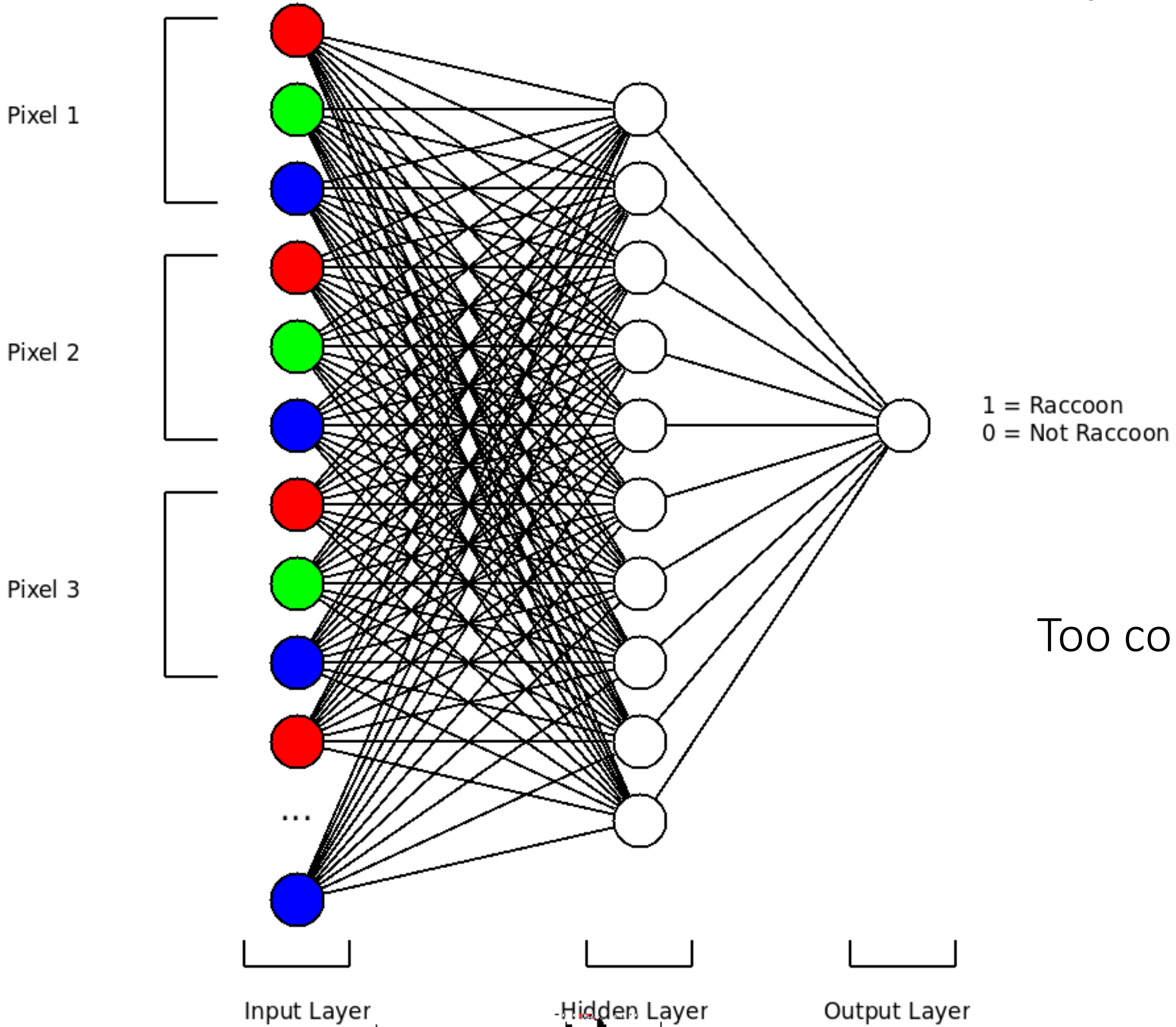


Visualization of Neurons in General



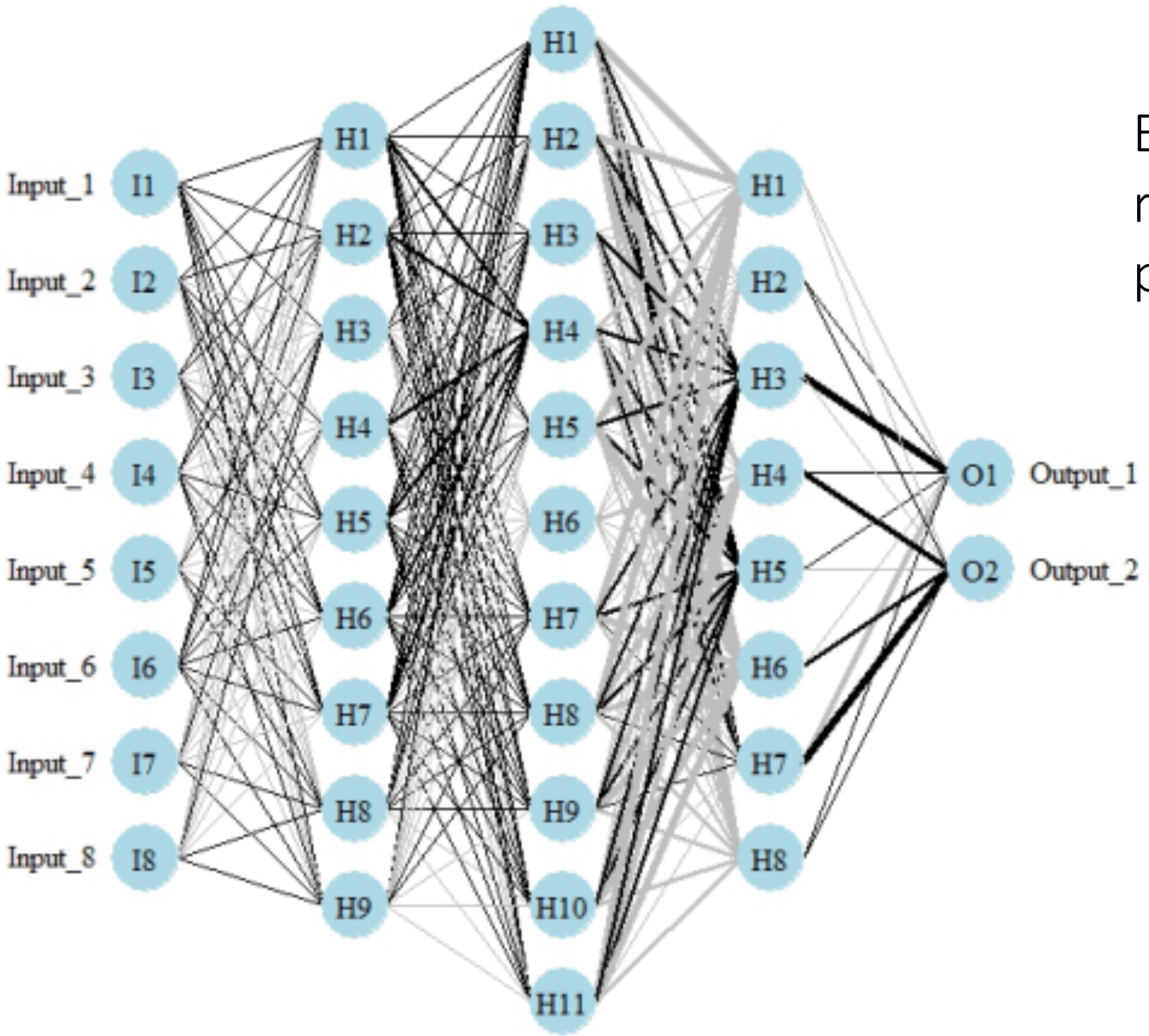


Neural Networks and Model Complexity



Too complex for a raccoon!

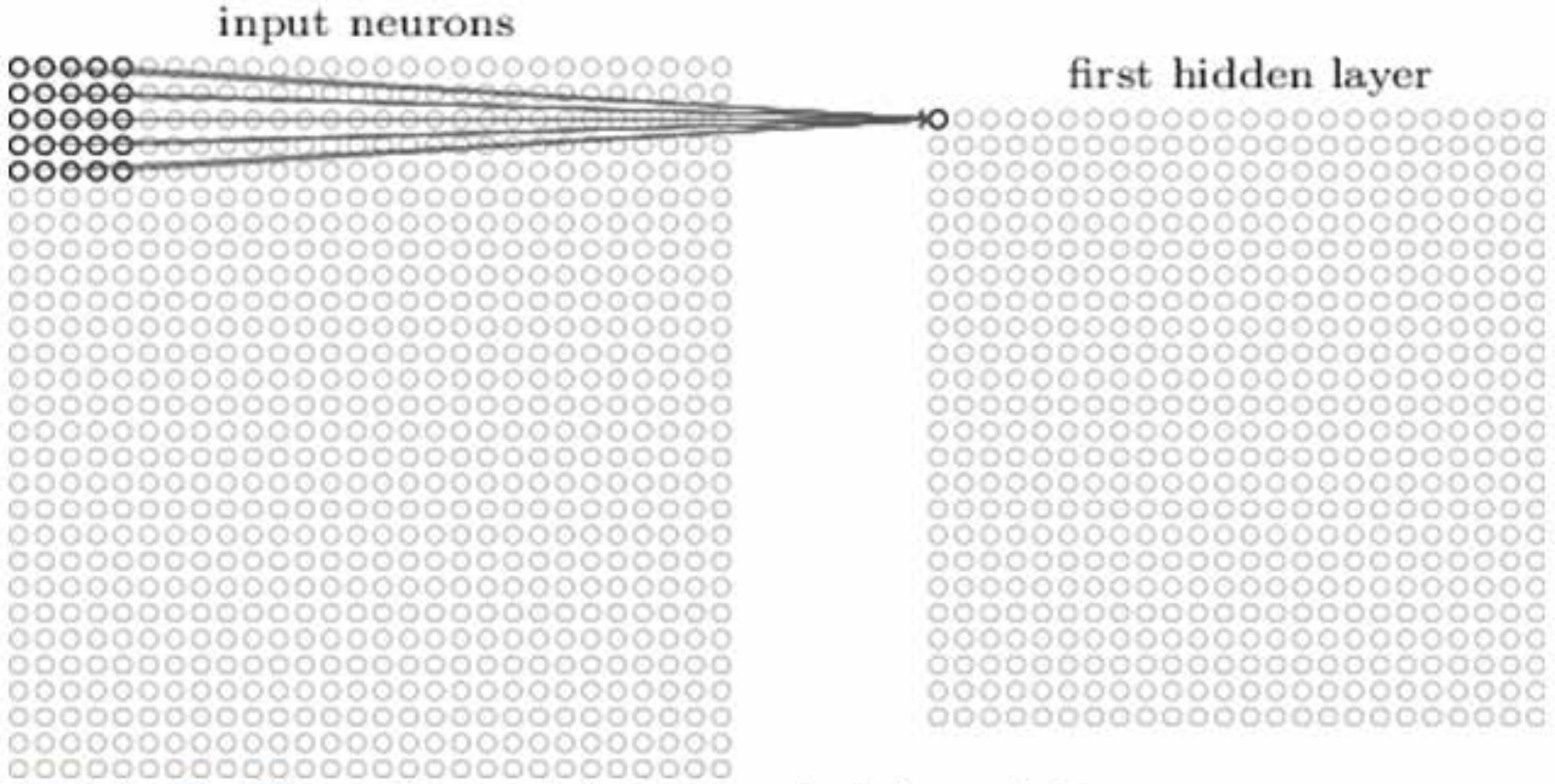
Deep and Multinomial Logistic Regression



Even more complex for a raccoon and a, let's say, panda!

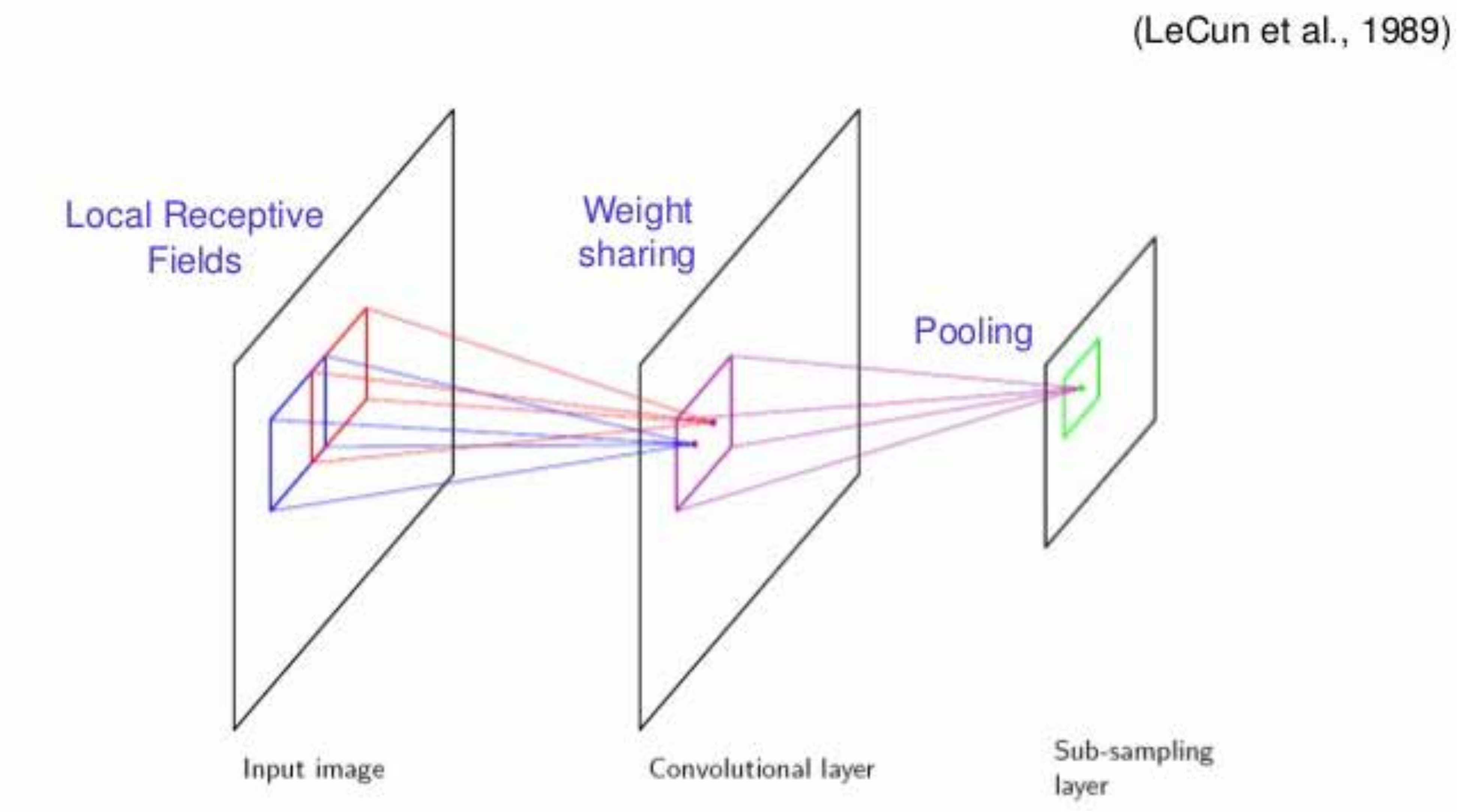
First simplification: Local Receptive Fields

- Consider a 2D representation of input neurons (e.g. M-NIST data)
- Consider a specific neuron in the hidden layer.
- Now, it will be connected to a local neighborhood only
  - Instead of connected it to each and every input

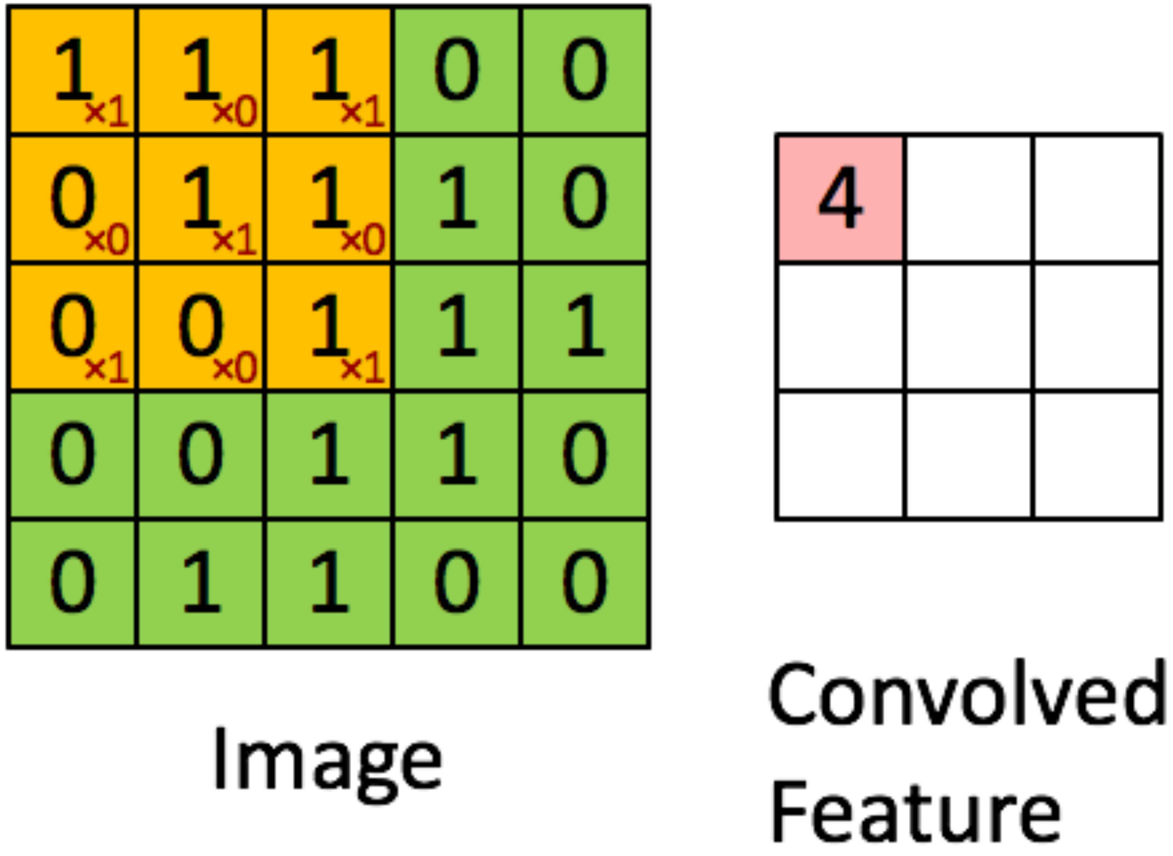


Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Second Simplification: Shared Weights

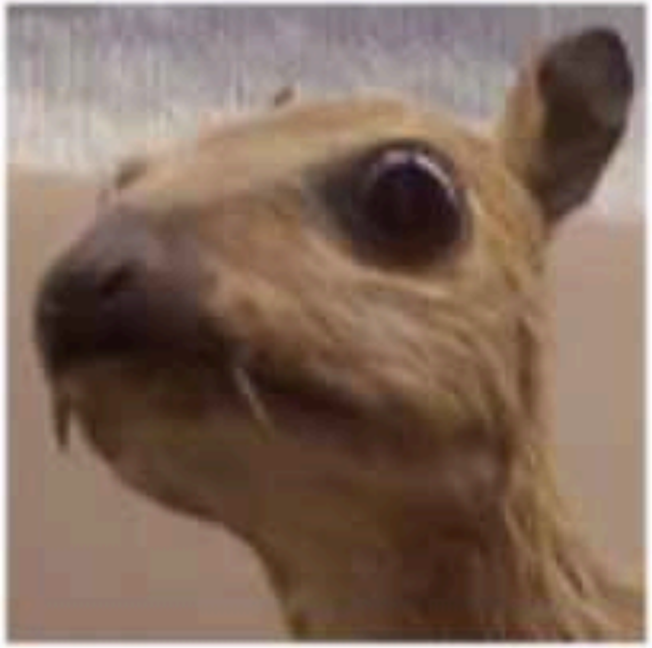


Recall: Convolution



Recall: Convolution

Input image

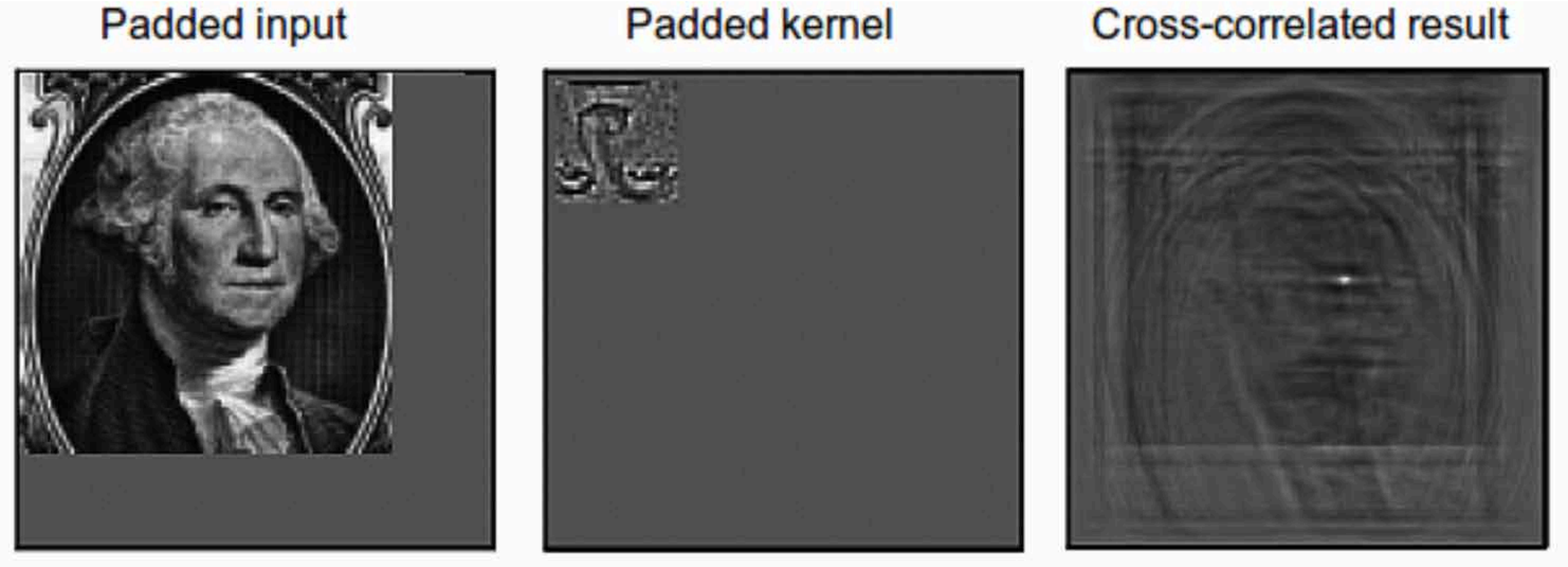


Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

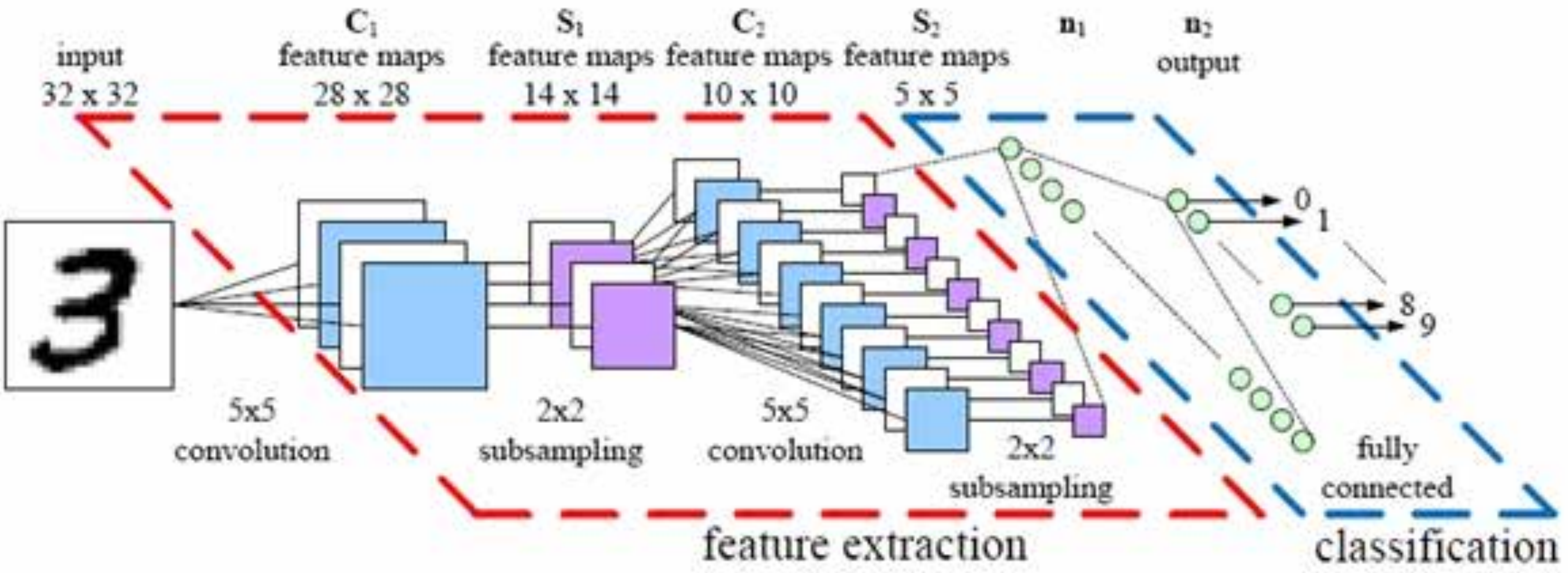
Feature map

Convolutional Filters

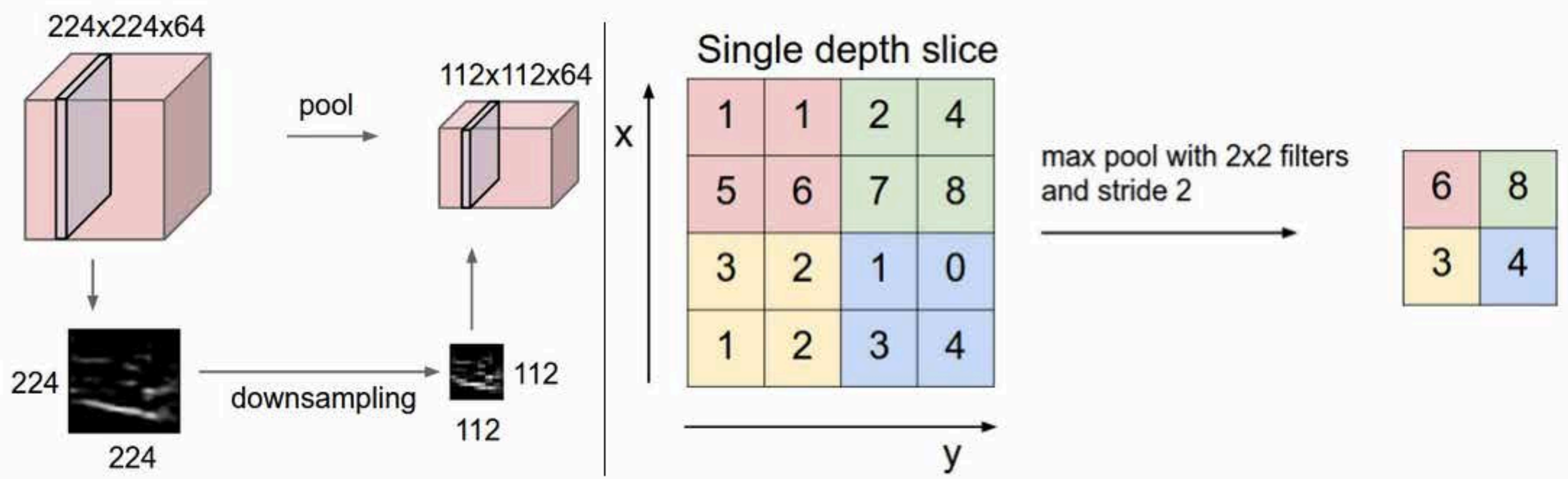


Convolutional filters can be interpreted as feature detectors, that is, the input (feature map) is filtered for a certain feature (the kernel) and the output is large if the feature is detected in the image.

CNN pipeline

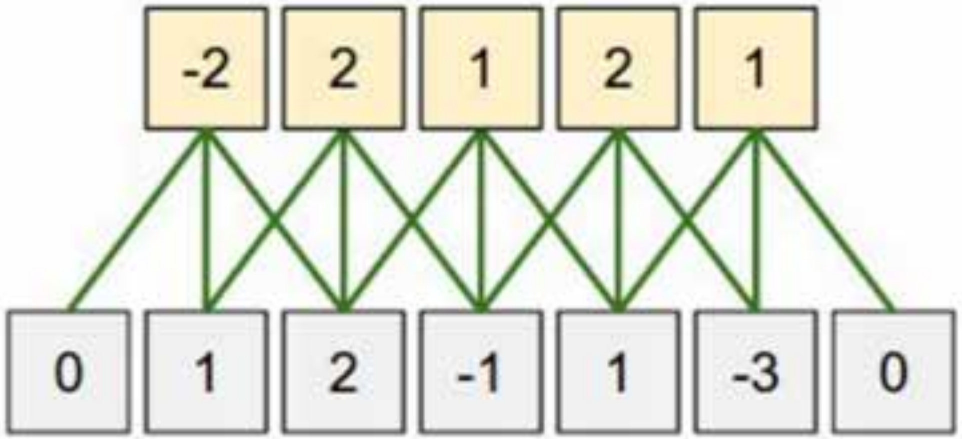


For translation invariance: Max-pooling

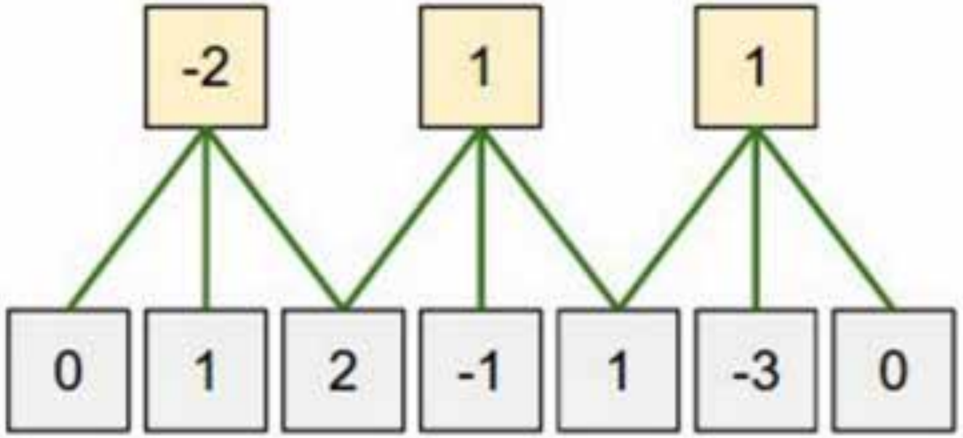


Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Stride Size

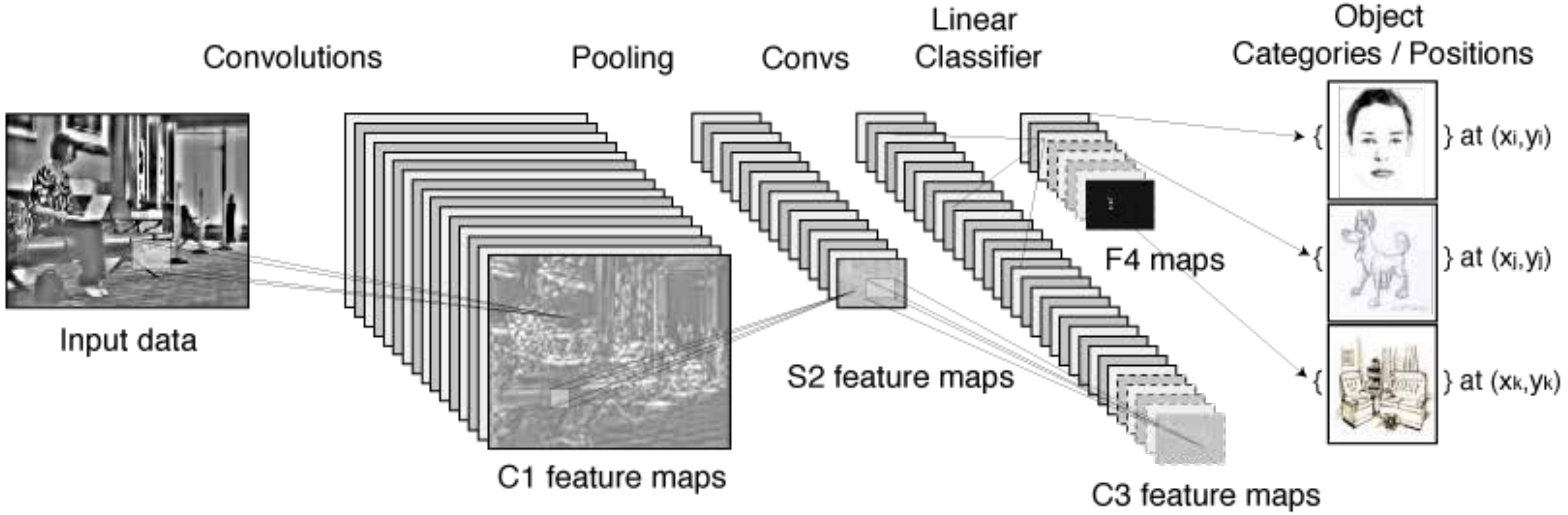


Stride = 1

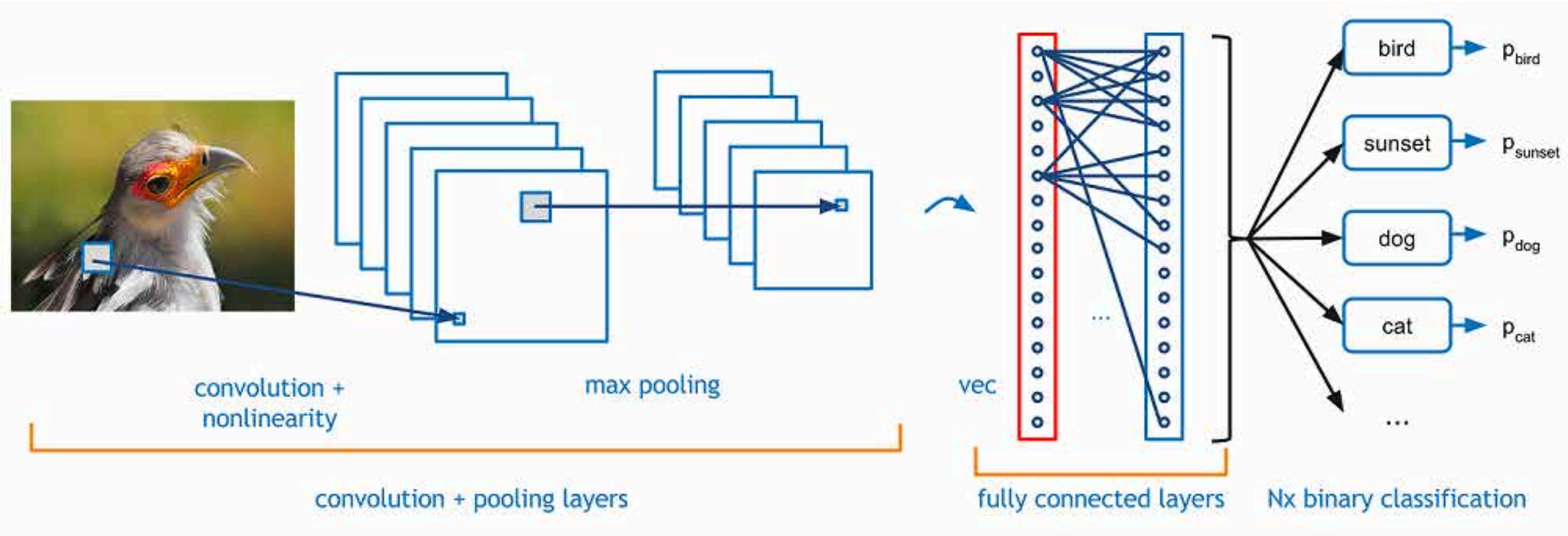


Stride = 2

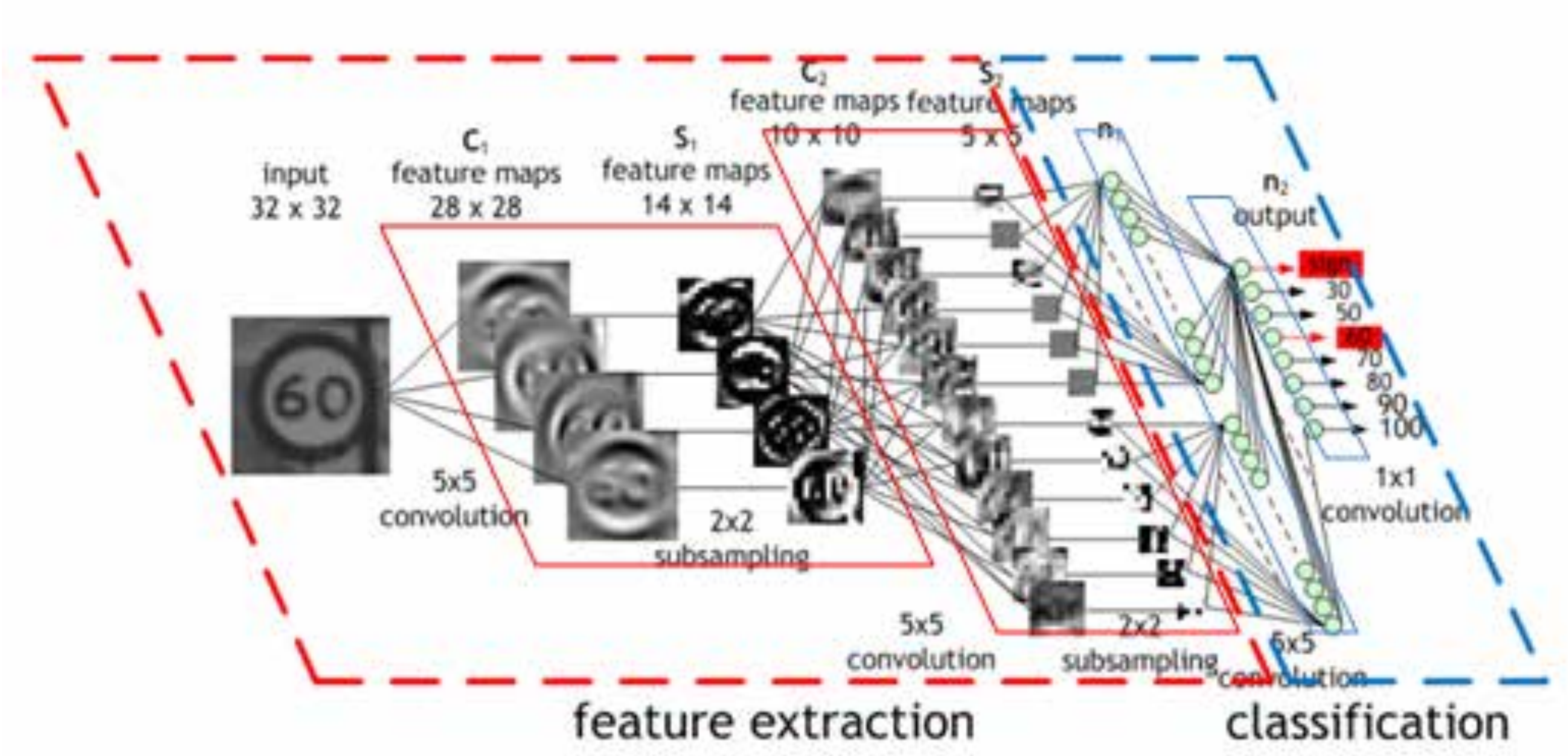
Example CNN pipeline for Object Detection



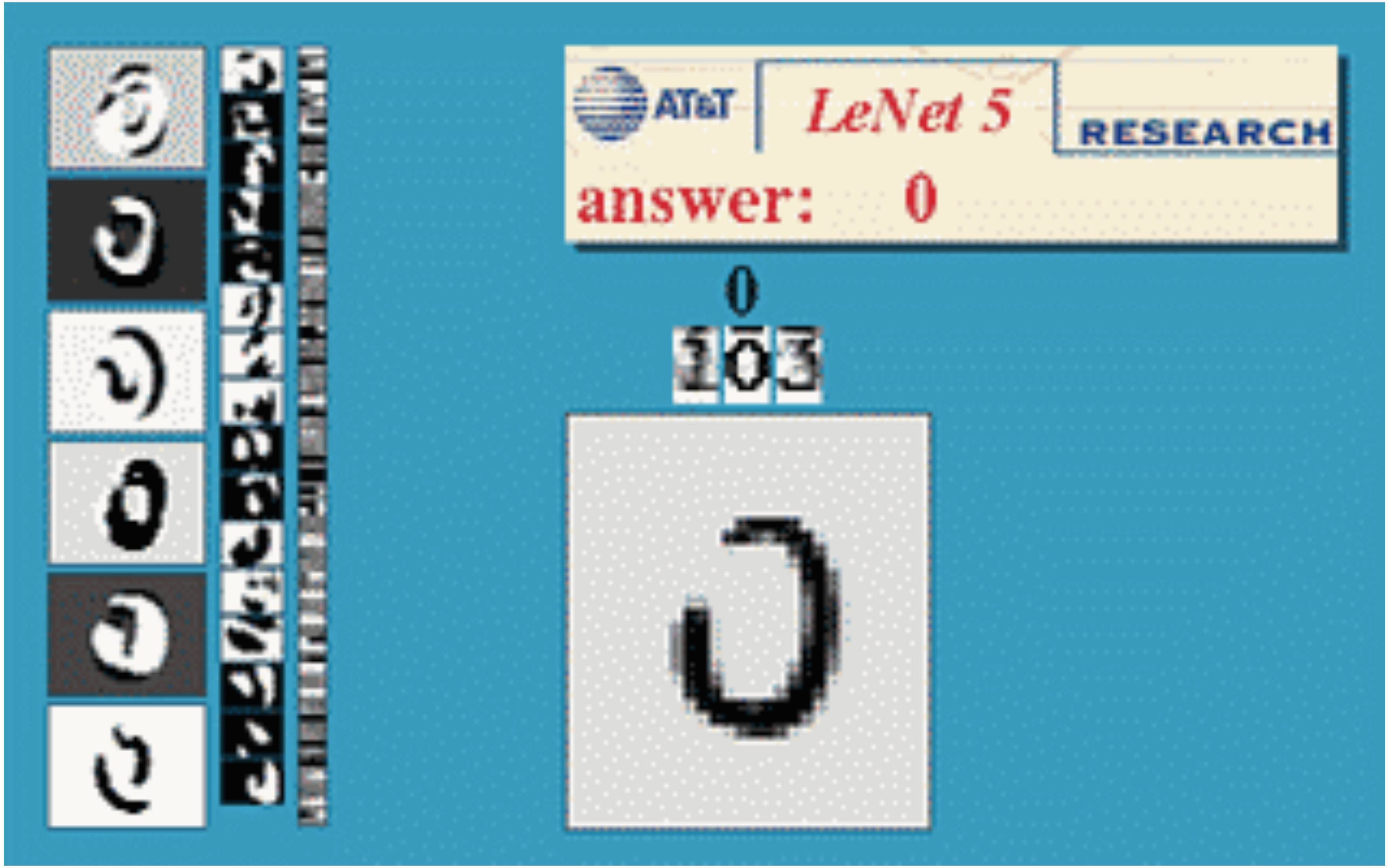
Example CNN pipeline for multiclass classification



Feature Learning/Extraction, they say.

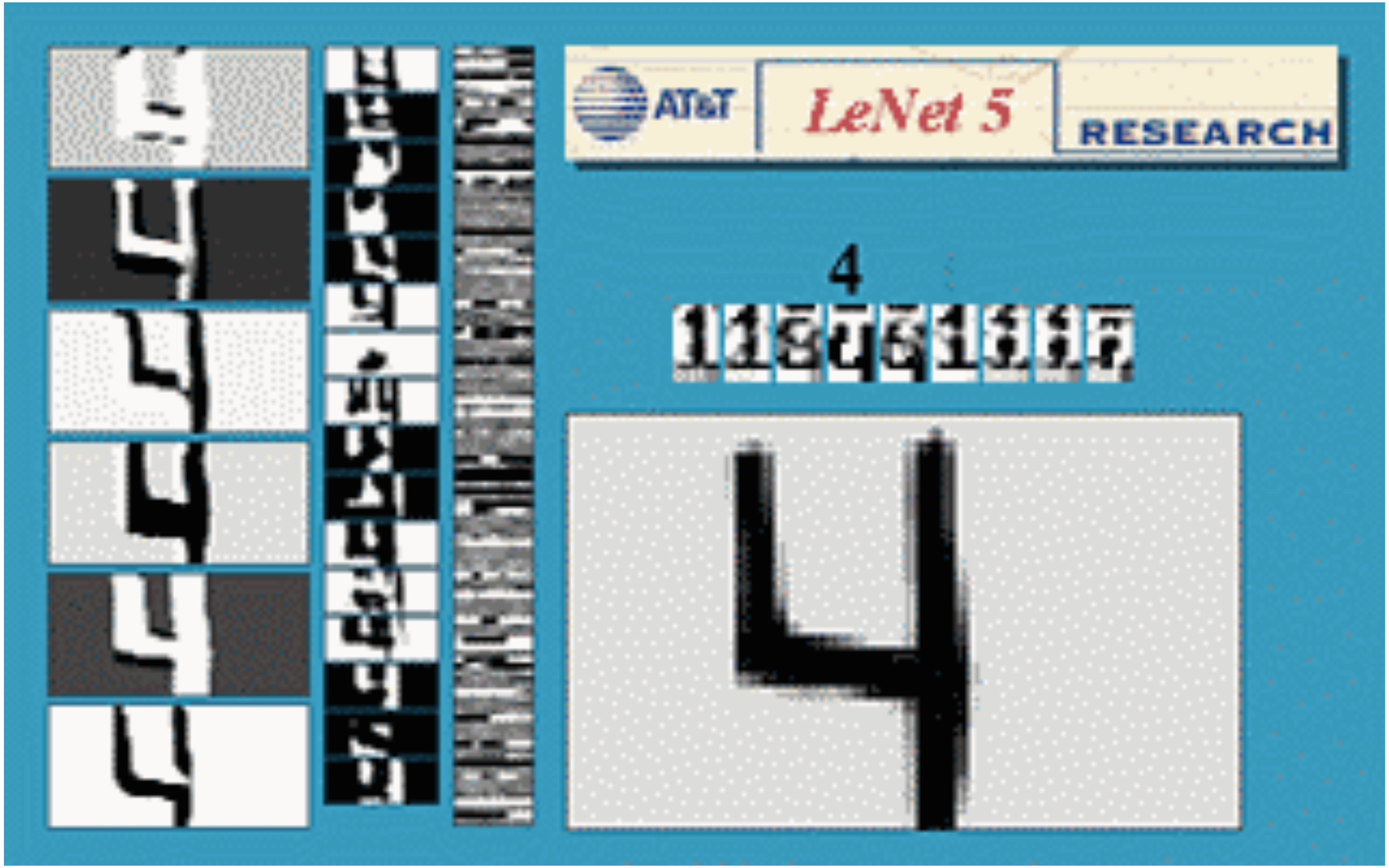


LeNet-5

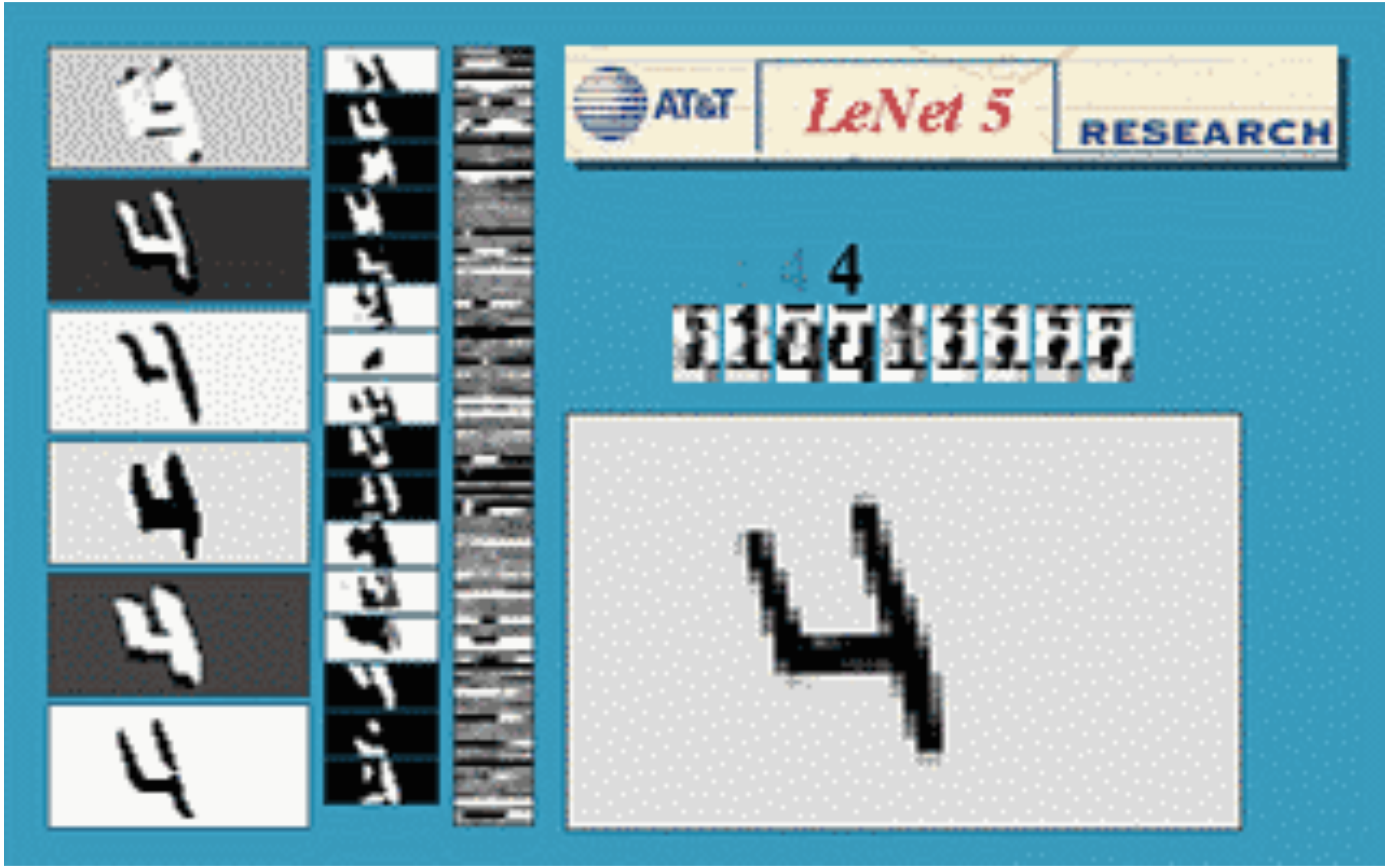


LeNet-5: Translation

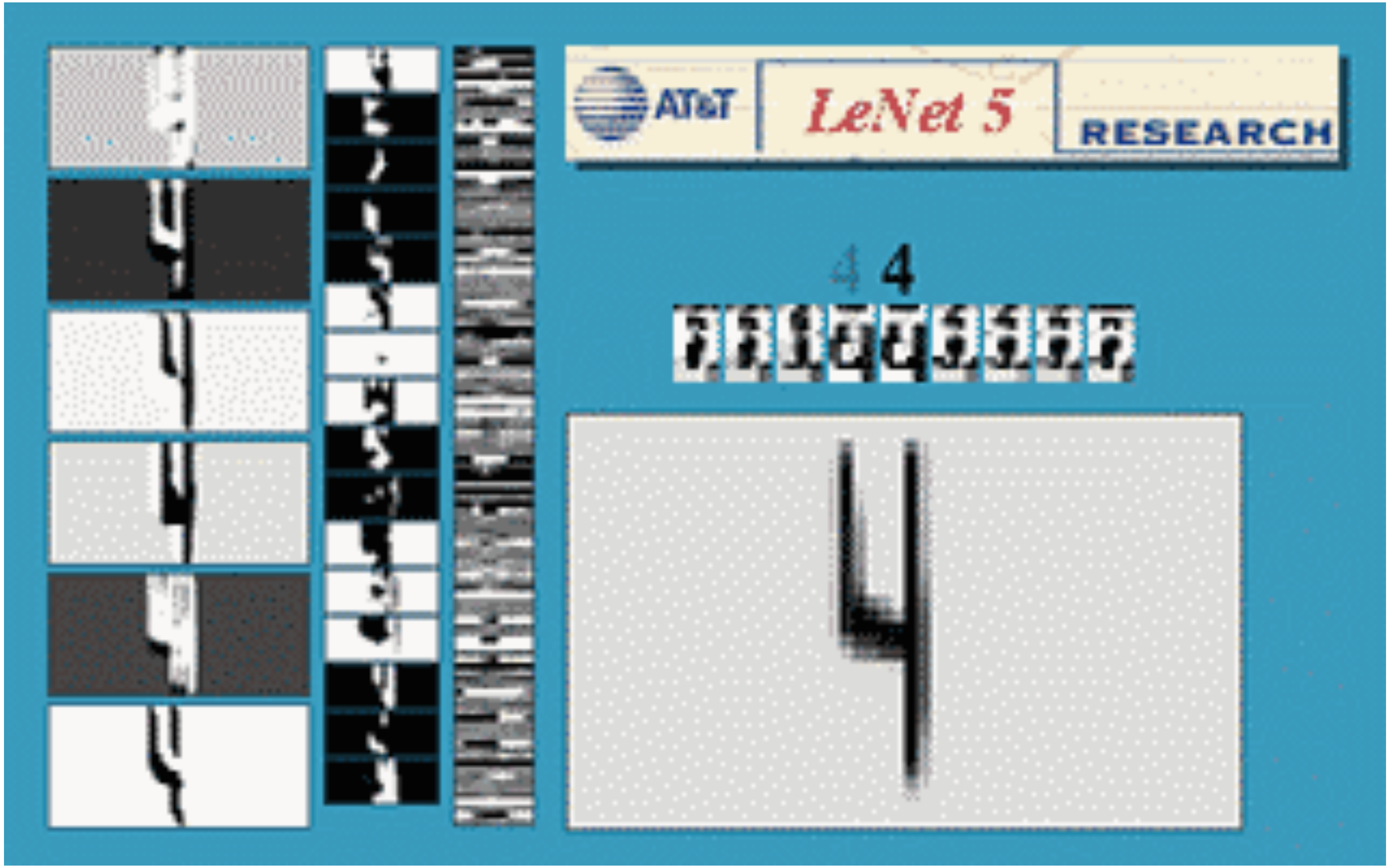
LeNet-5: Scale



LeNet-5: Rotation



LeNet-5: Squeezing



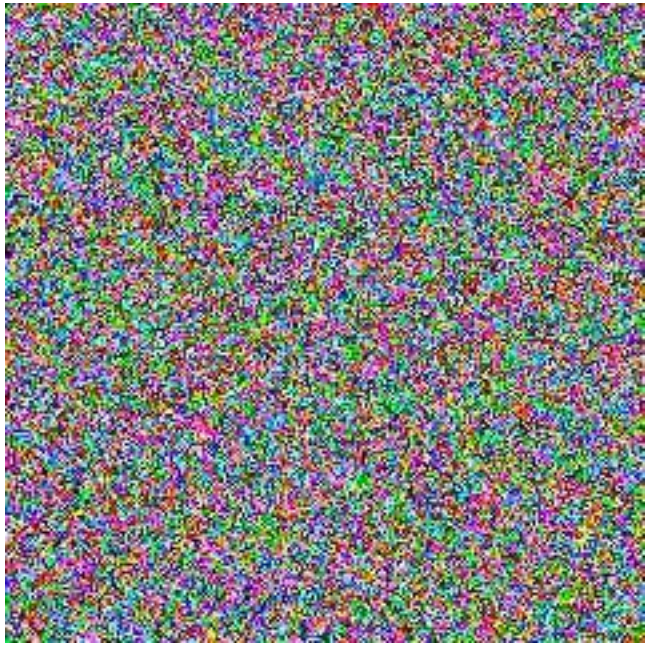
LeNet-5: Stroke Size

Visualizing Neurons

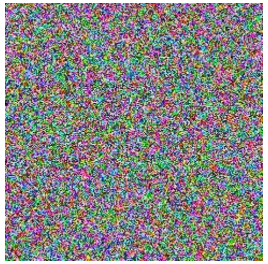
- Deep neural networks have recently been producing amazing results!
- But how do they do what they do?
- Historically, they have been thought of as “black boxes”, meaning that their inner workings were mysterious and inscrutable.
- We need to better understand exactly what each neuron has learned and thus what computation it is performing

Visualizing Neurons

- To visualize the function of a specific neuron, we synthesize inputs that cause that unit to have high activation.
- The resulting synthetic image shows what the neuron "wants to see" or "what it is looking for".
- We start by a random image (at the right):

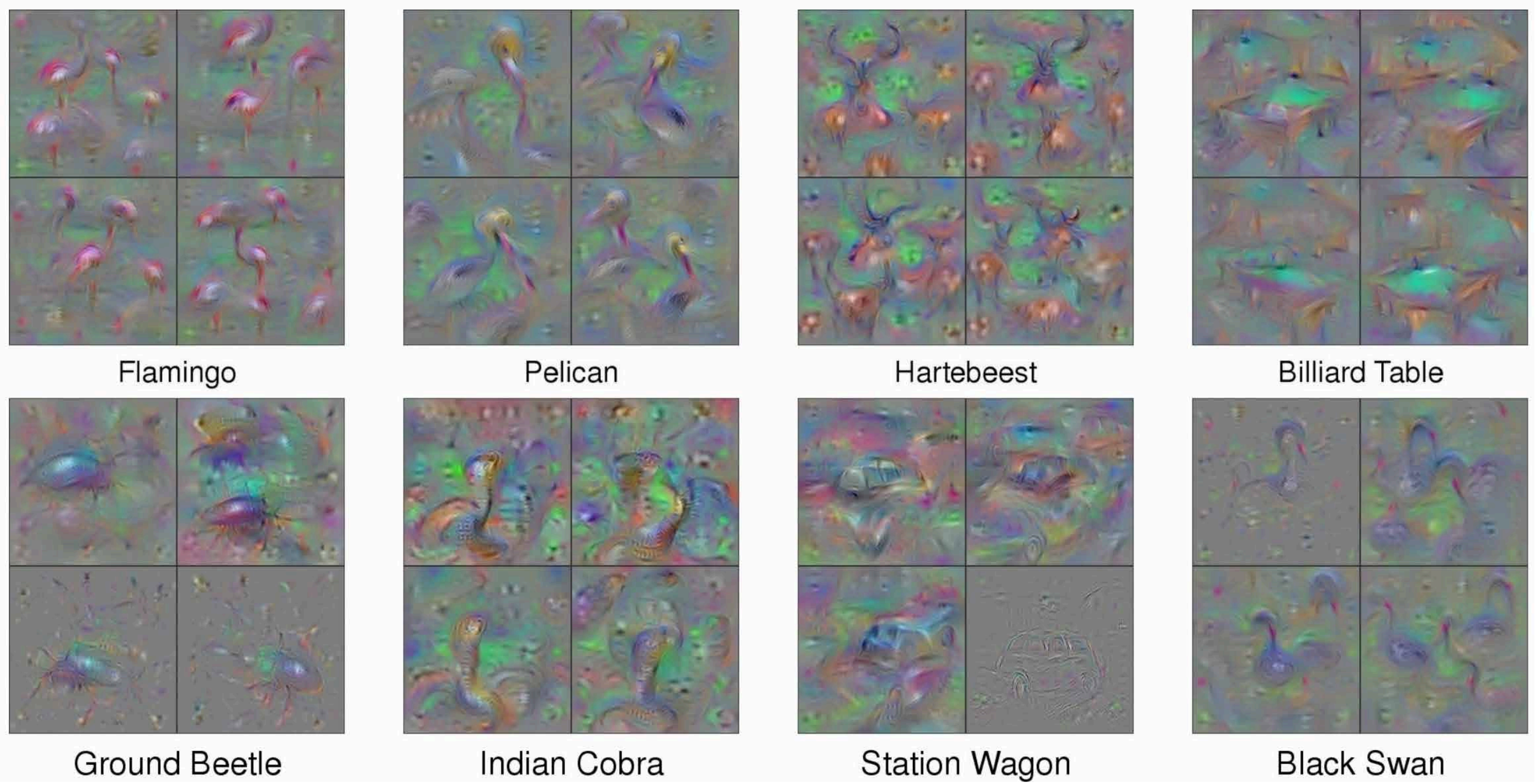


Visualizing Neurons

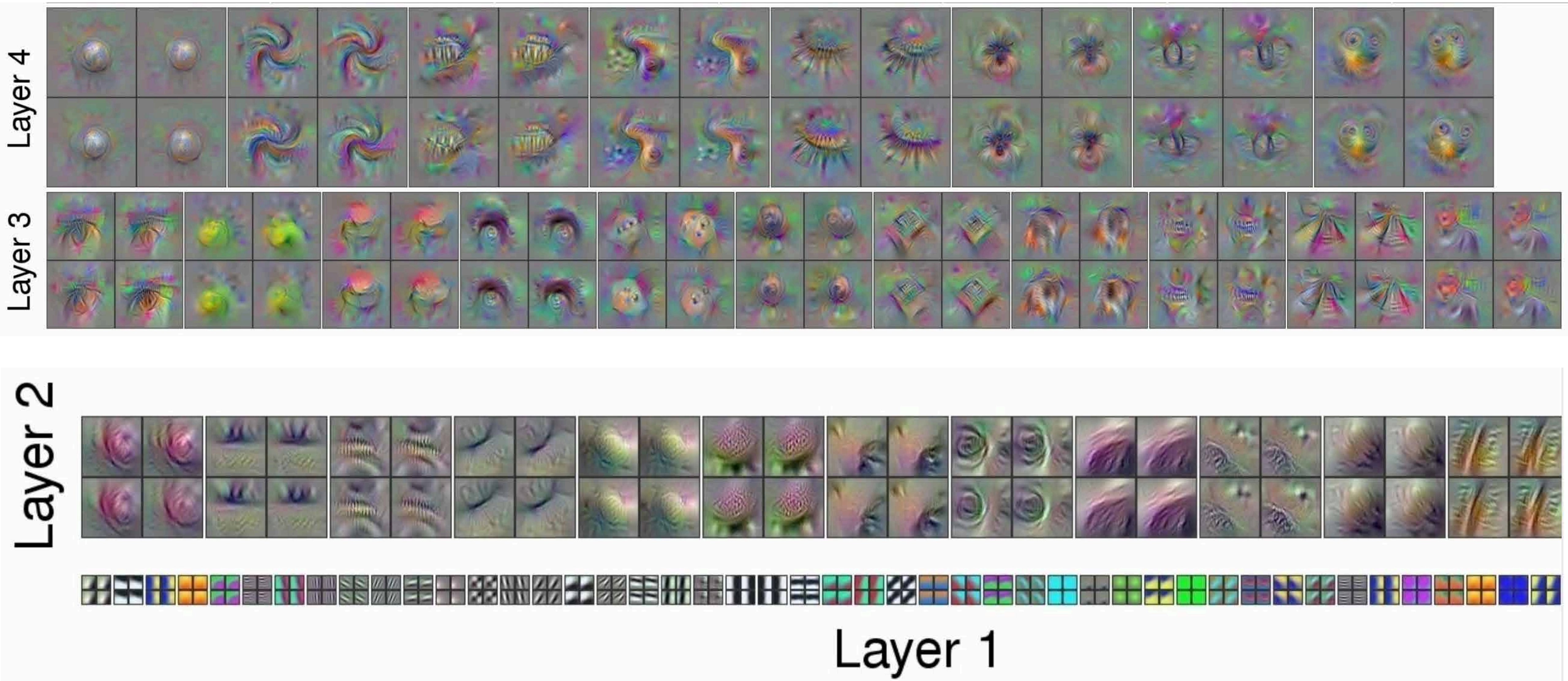


- We do a forward pass using ***this image as input*** to the network to compute the activation caused by it at some ***neuron i*** somewhere in the middle of the network.
- Then we do a backward pass (performing backprop) to compute the gradient of ***neuron i*** with respect to earlier activations in the network.
- At the end of the backward pass we are left with the gradient, or how to change the color of each pixel to increase the activation of ***neuron i***.
- We keep doing that repeatedly until we have an image ***x**** that causes high activation of the neuron in question.

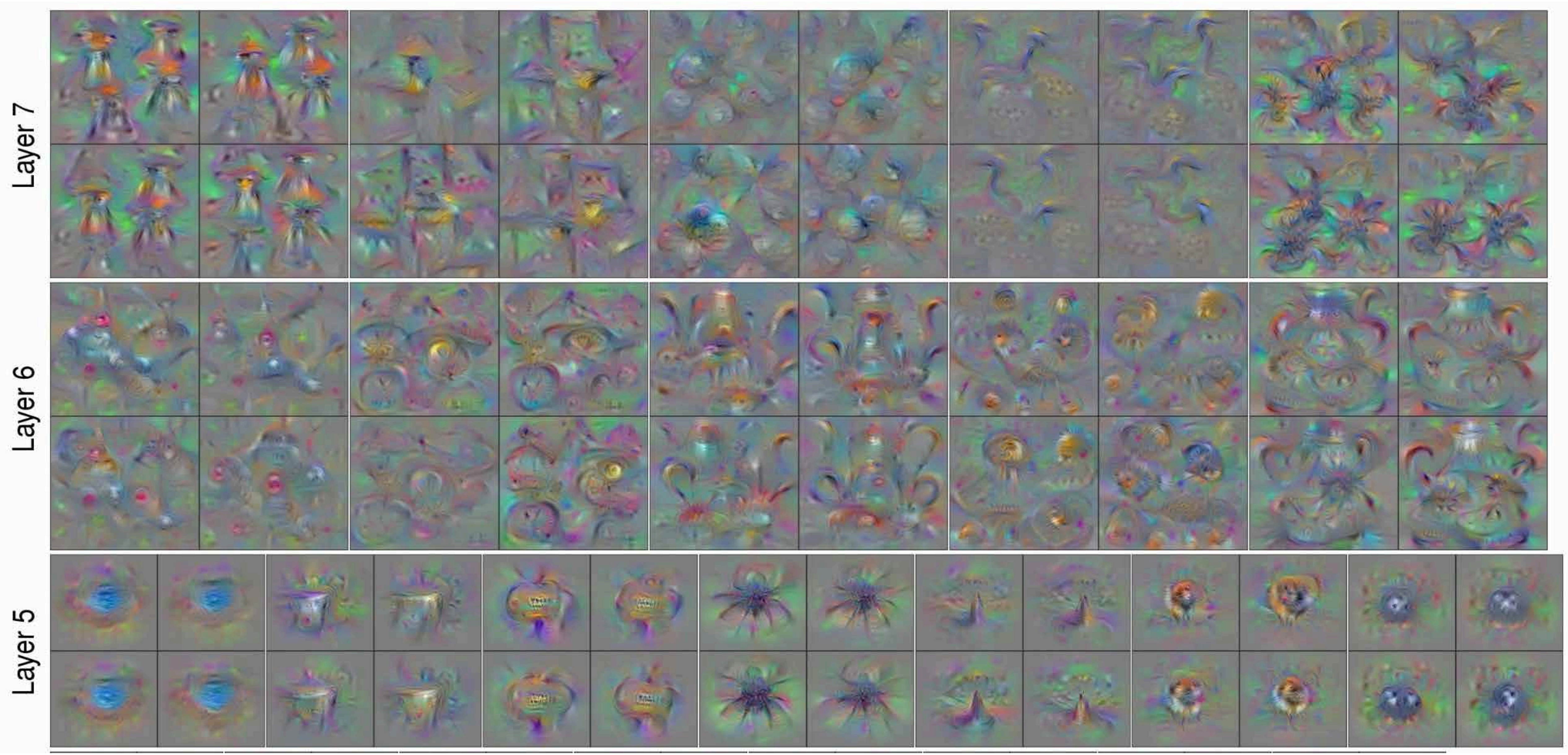
Visualizing Neurons



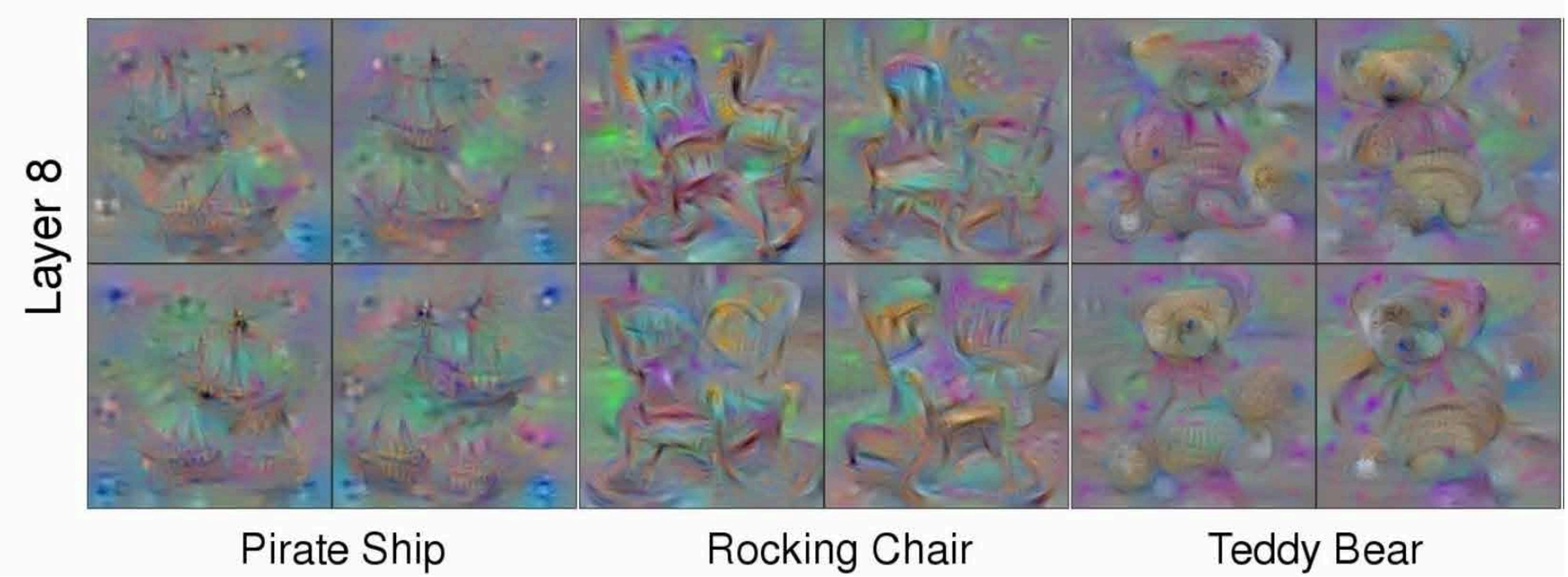
Visualizing Neurons



Visualizing Neurons



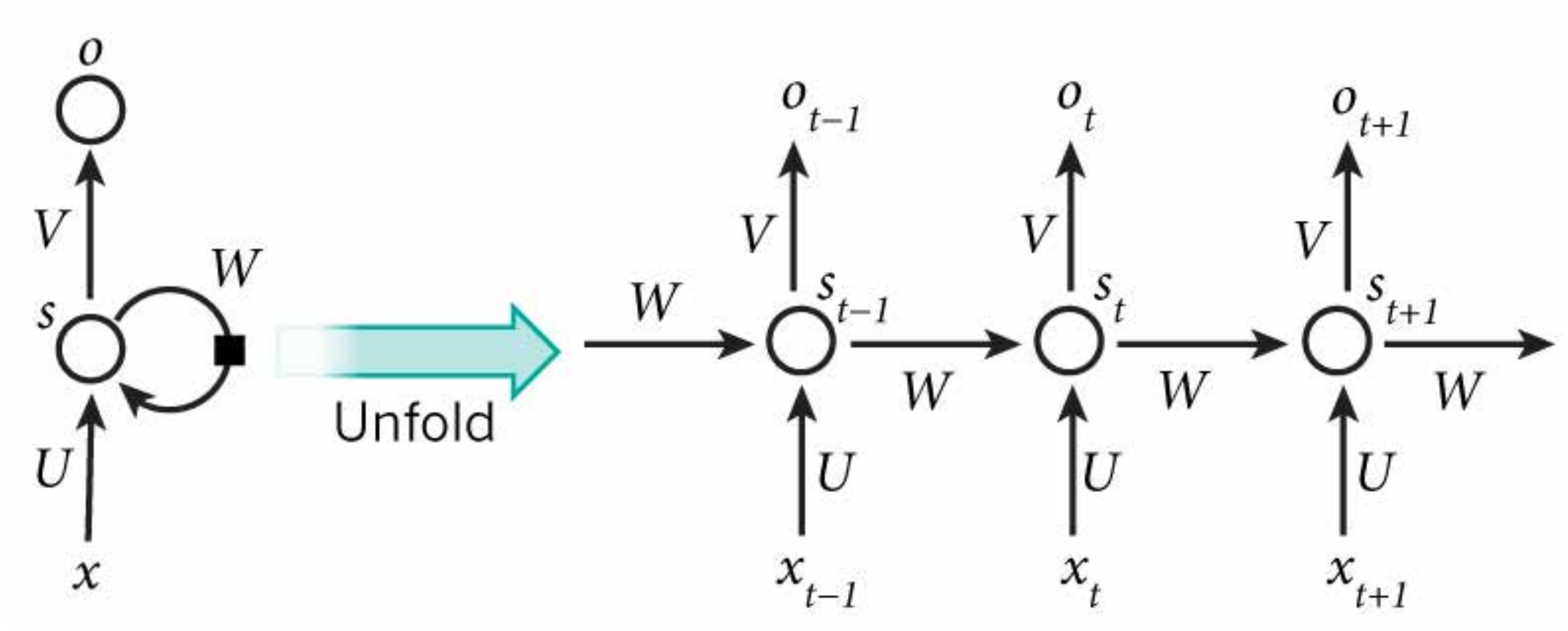
Visualizing Neurons



Recurrent Neural Networks

- The idea behind RNNs is to make use of sequential information.
- In a traditional neural network we assume that all inputs (and outputs) are independent of each other.
    - But for many tasks that’s a very bad idea.
    - If you want to predict the next word in a sentence you better know which words came before.
- RNNs are called *recurrent* because they perform the same task for every element of a sequence, with the output being depended on the previous computations.
- Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far.
- In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later).

Recurrent NN and Unfolding



By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word.

Word Embeddings: How to represent words?

1 of k encoding:

- transforms categorical features to a format that works better with classification and regression algorithms.

Let's try to encode some words:

- What's the problem in this encoding?
- We can't say that the category of "Penguin" is greater or smaller than "Human". Then they would be ordinal values, not nominal.

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

Nominal variables are used to "name," or label a series of values. **Ordinal** scales provide good information about the order of choices.

Word Embeddings: How to represent words?

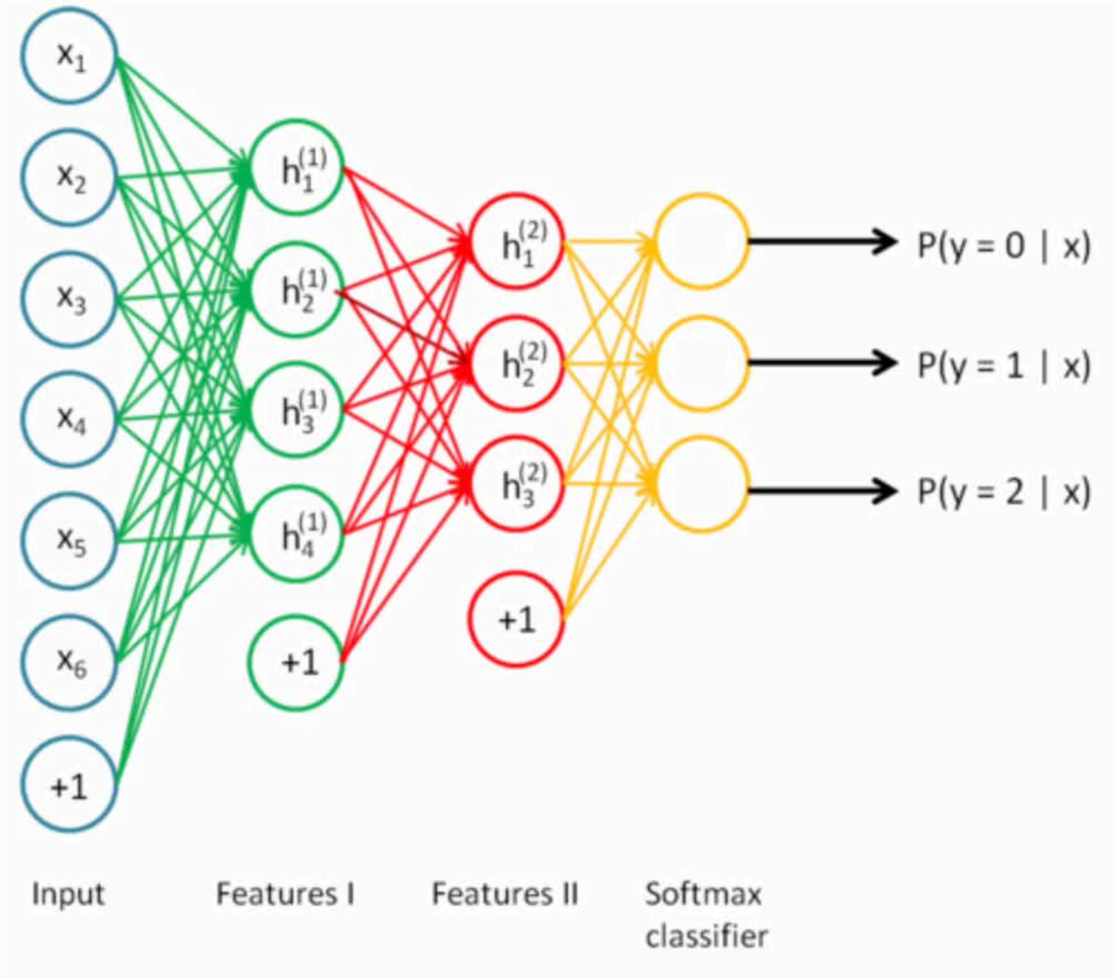
1 of k encoding:

- What we do instead is generate one boolean column for each category. Only one of these columns could take on the value 1 for each sample.

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

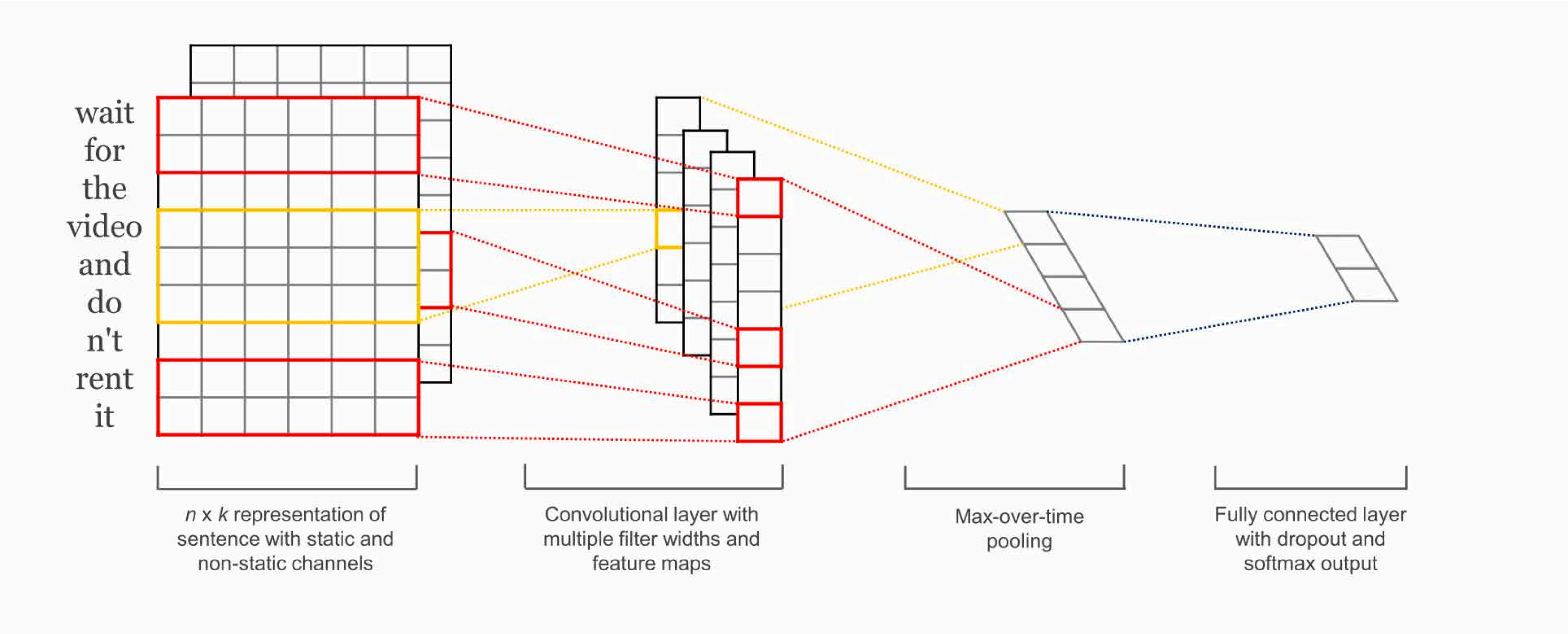
Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

Recall: Multinomial Logistic (Softmax) Regression



- Output layer is a nominal layer here.
- We use a **nominal** representation of our classes because we want to assign same amount of importance to our class probabilities.
- Using a single output and partitioning it for different classes would be using an **ordinal** representation, meaning that classes would be given importance to their order in the representation.
- There is an exception case:
  - Logistic Regression

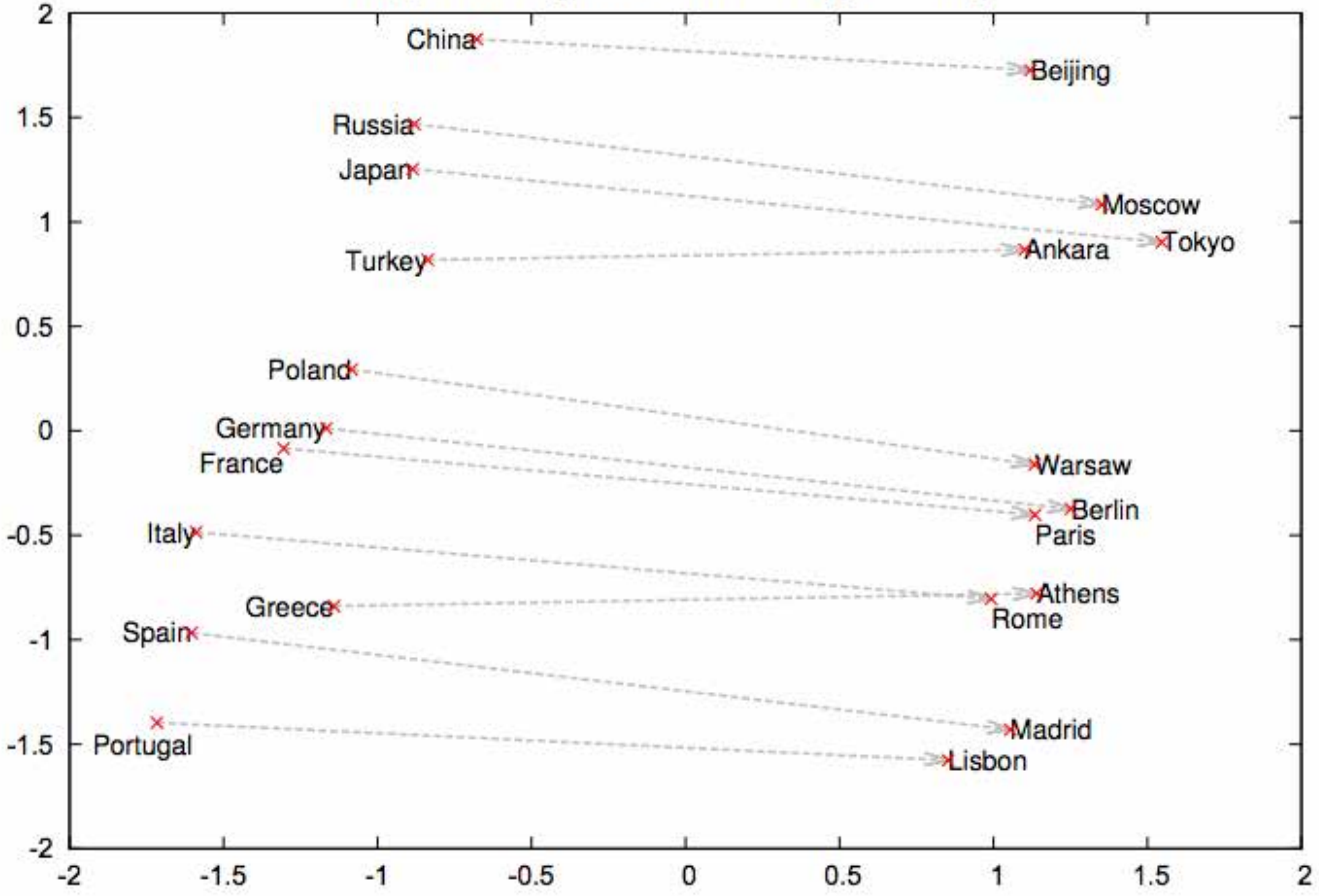
CNN training for sentence classification



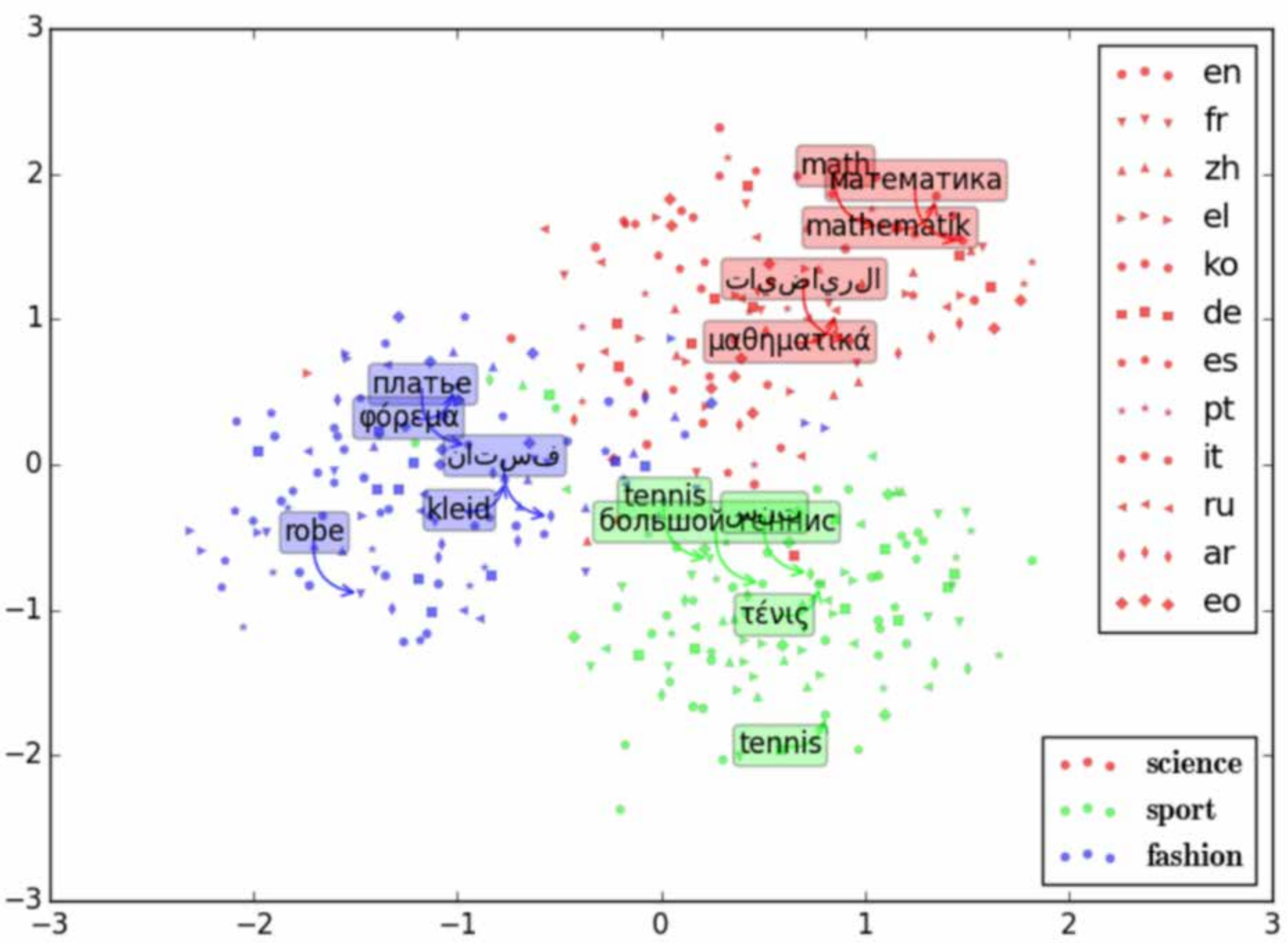
Capturing word relations

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Country and Capital Vectors Projected by PCA



Multilingual Word Embeddings



Many frameworks exist

- TENSORFLOW
- CAFFE
- TORCH
- MICROSOFT DISTRIBUTED MACHINE LEARNING TOOKIT
- MICROSOFT AZURE MACHINE LEARNING
- MXNET
- ENCOG
- H2O
- NEON
- THEANO
- SCIKIT LEARN