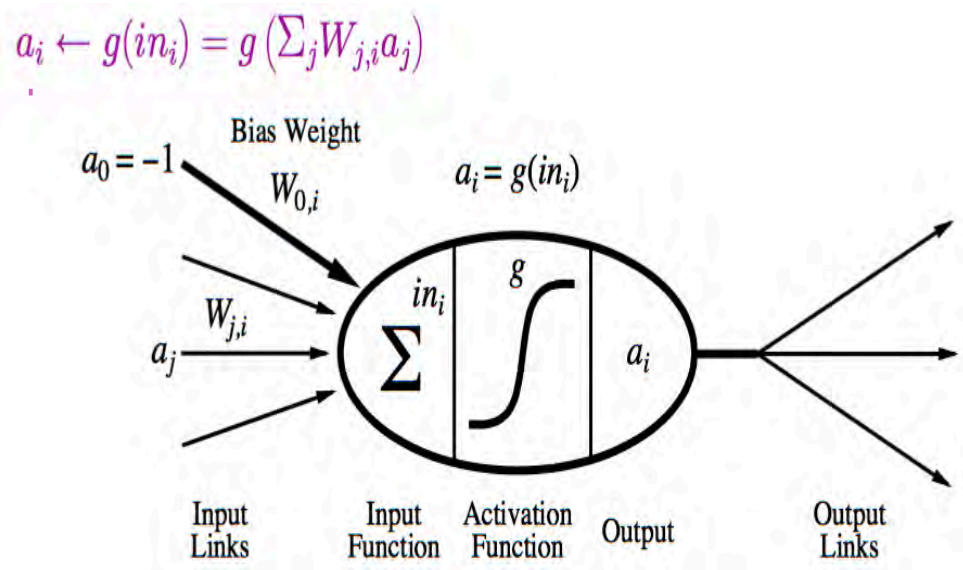
Introduction to Convolutional Neural Networks

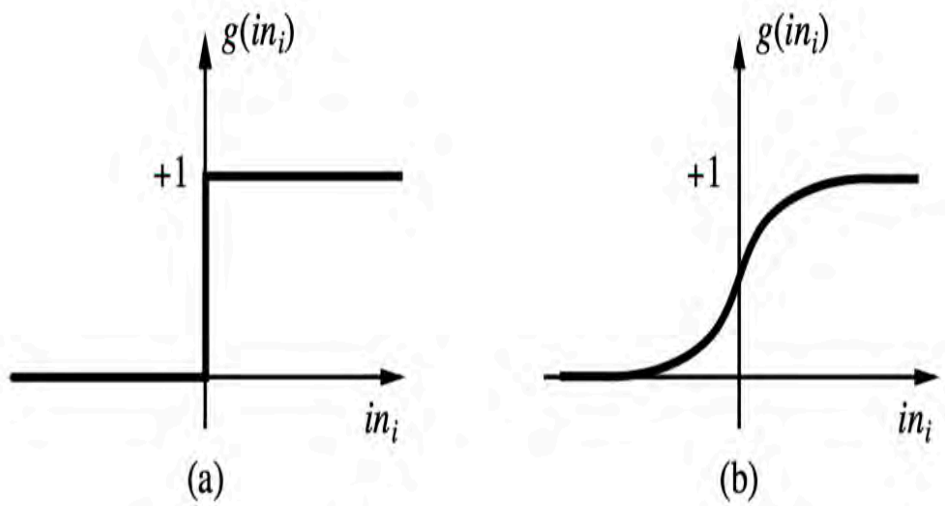
CHEM 430

What is a Neuron ?

- A neuron is a computational unit in a neural network that exchanges messages with other neurons



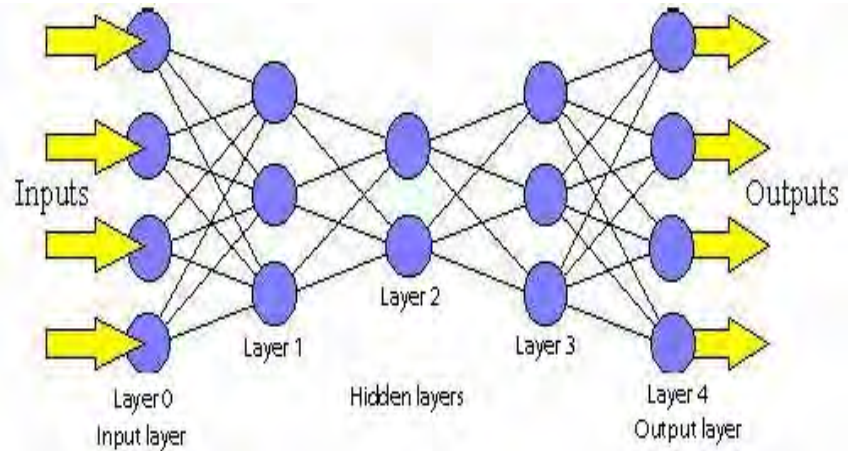
Possible activation functons:



Step function/ threshold function

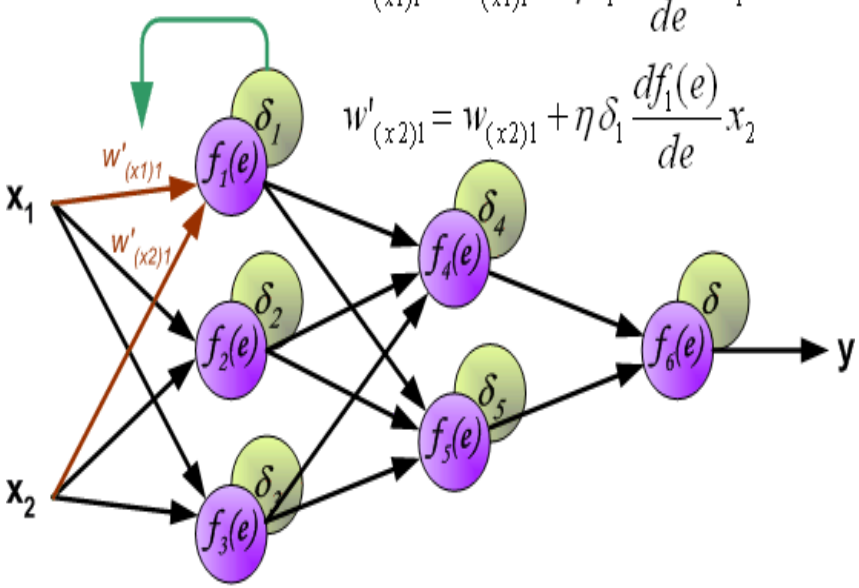
Sigmoid function (a.k.a, logistic function)

Feed Forward & Backpropagation



$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



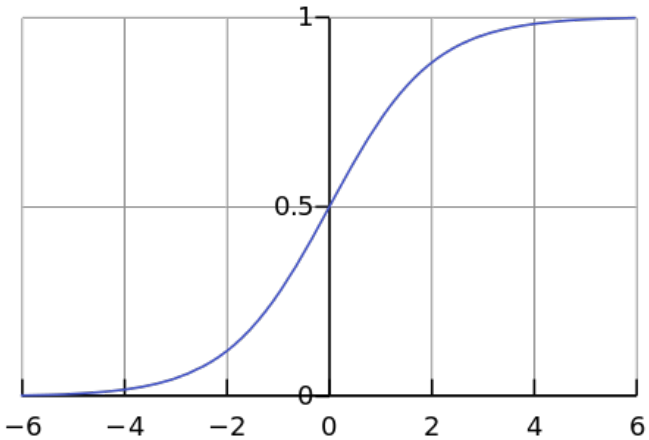
Feed forward algorithm:

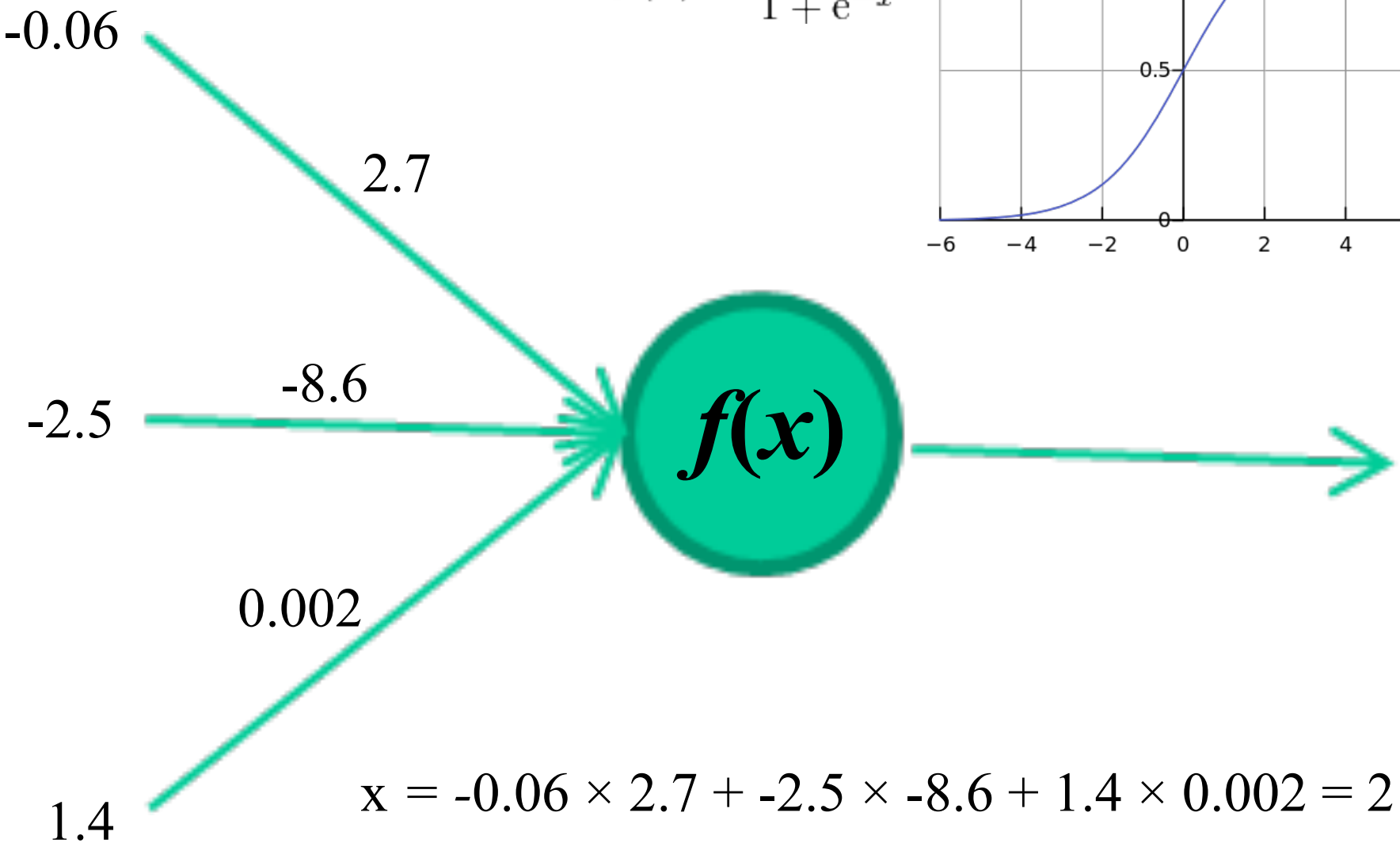
- Activate the neurons from the left to the right.

Backpropagation:

- Randomly initialize the parameters
- Calculate total error at the right, $f6\ (e)$
- Then calculate contributions to error, δn, at each step going backwards.

$$f(x) = \frac{1}{1 + e^{-x}}$$





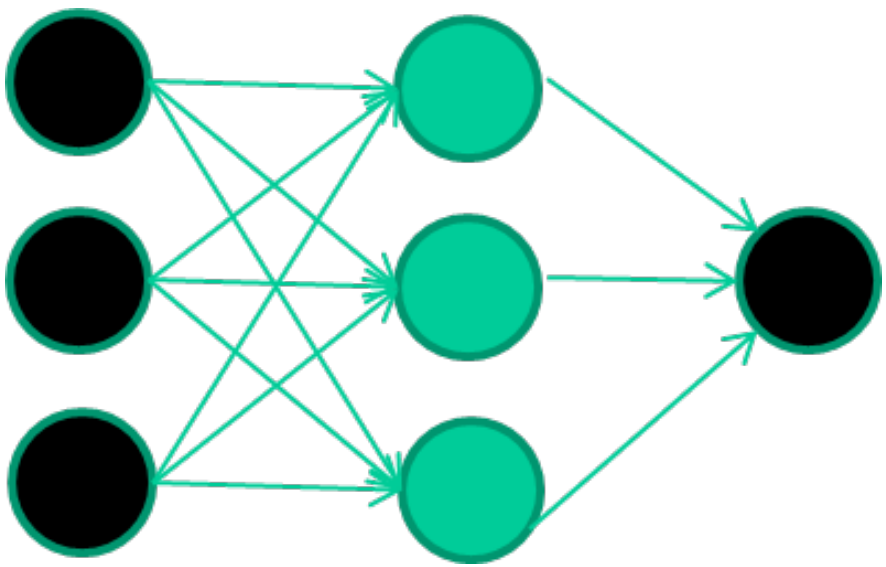
$$x = -0.06 \times 2.7 + -2.5 \times -8.6 + 1.4 \times 0.002 = 21.34$$

Training data

Initialise with random weights

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

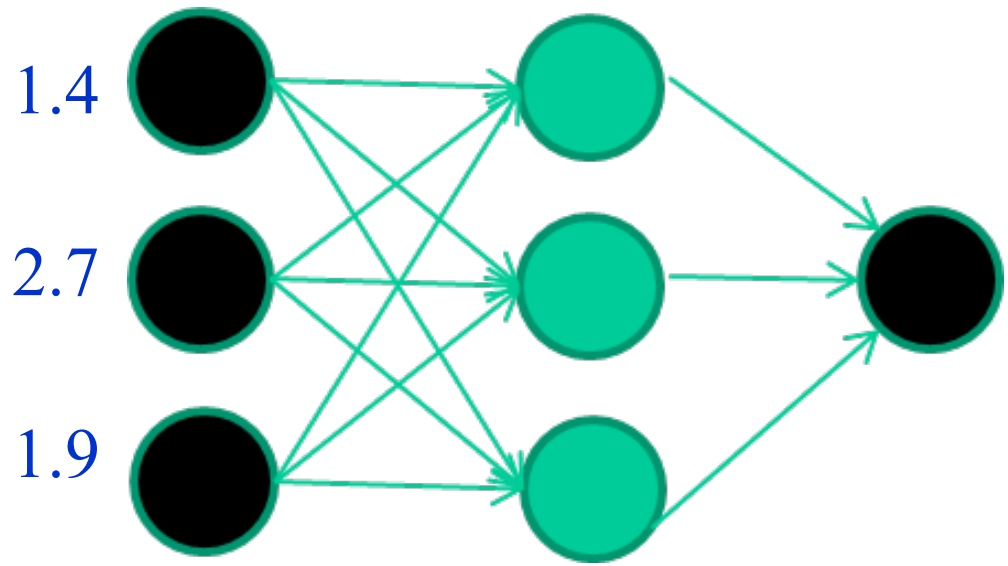
etc …



Training data

Present a training pattern

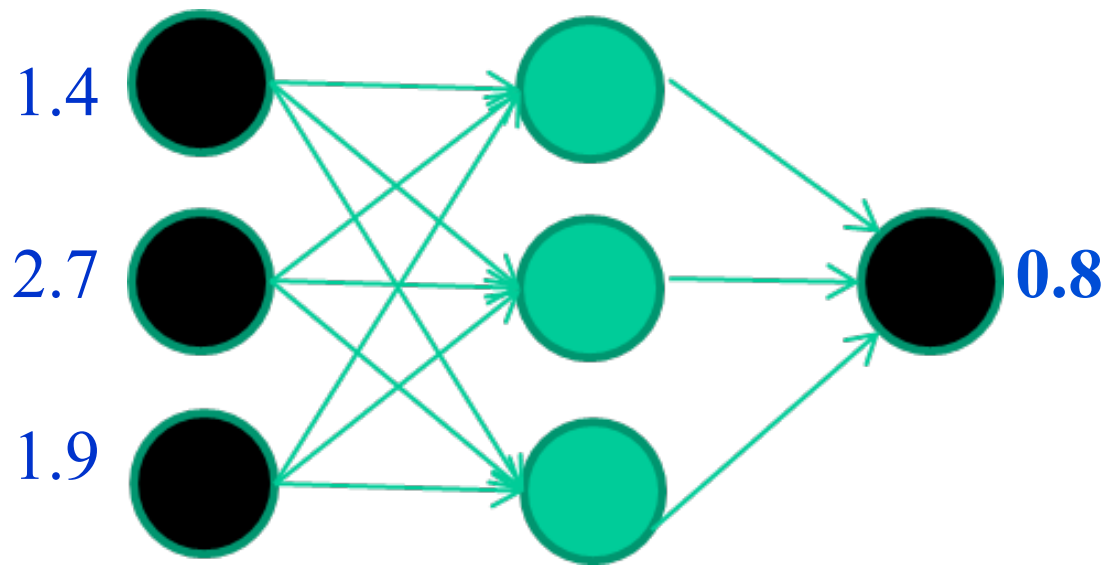
Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

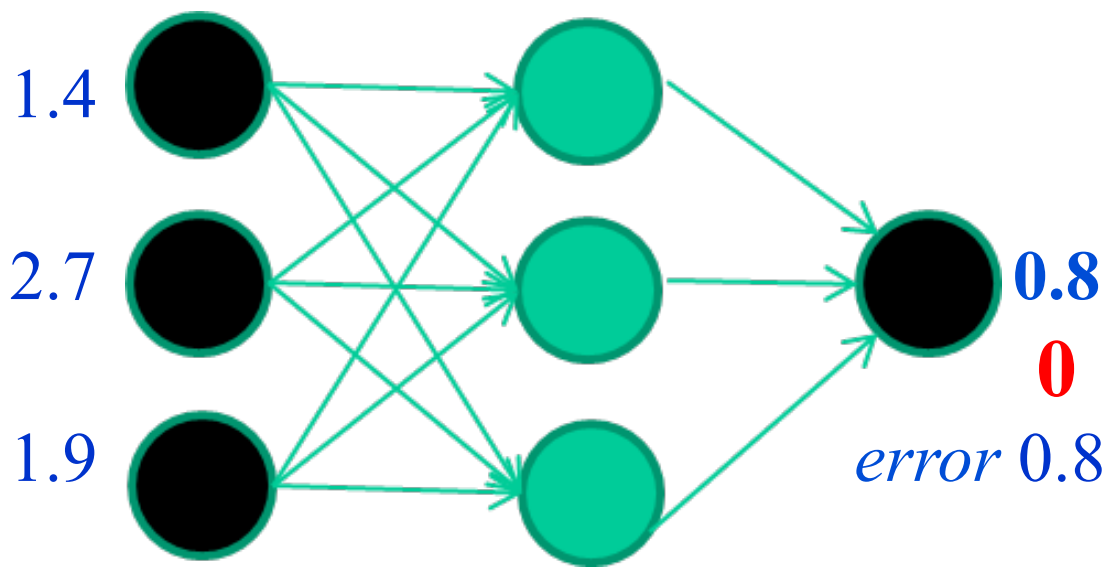
Feed it through to get output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

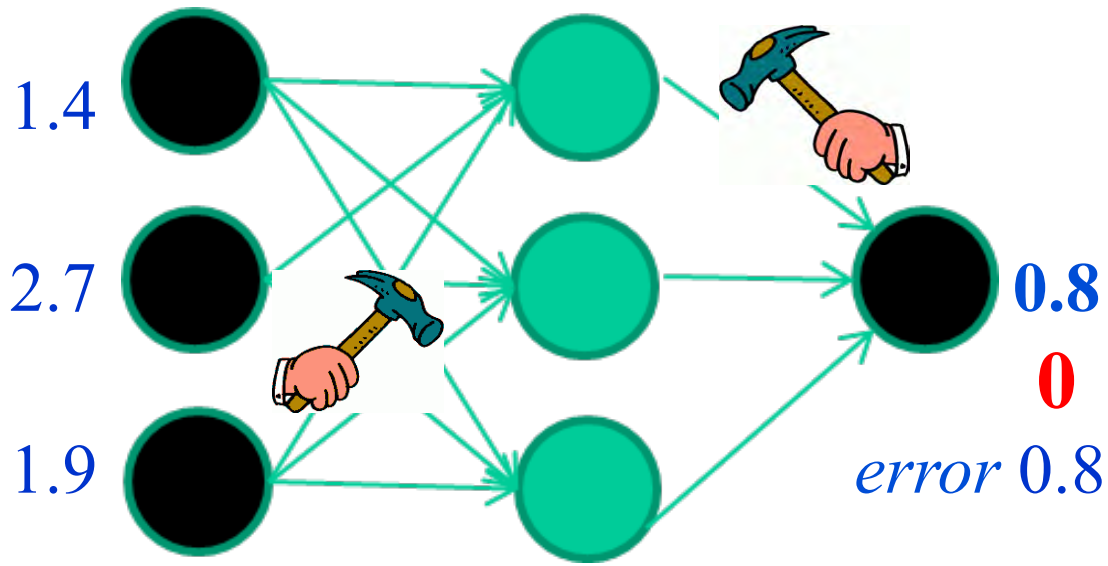
Compare with target output



Training data

Adjust weights based on error

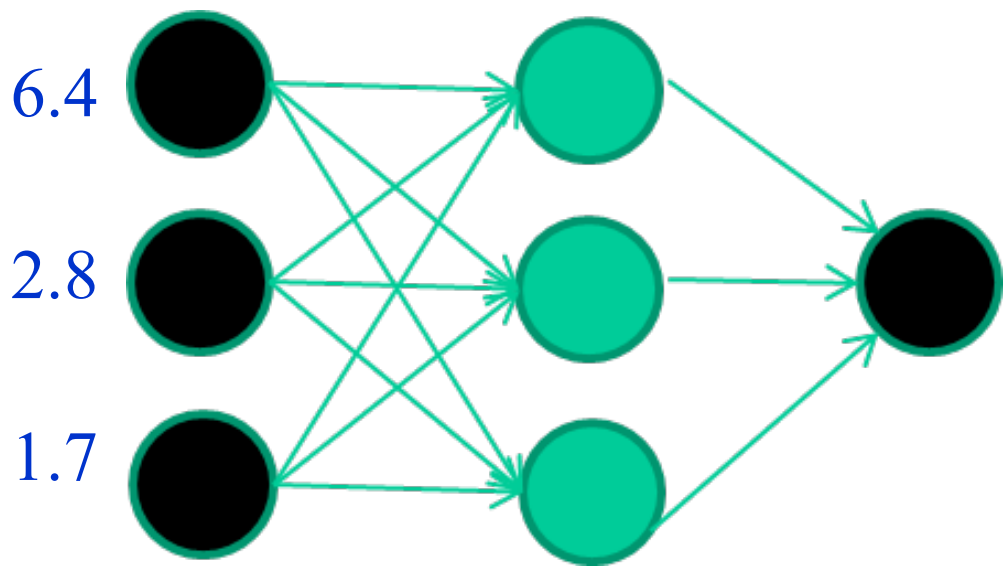
Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

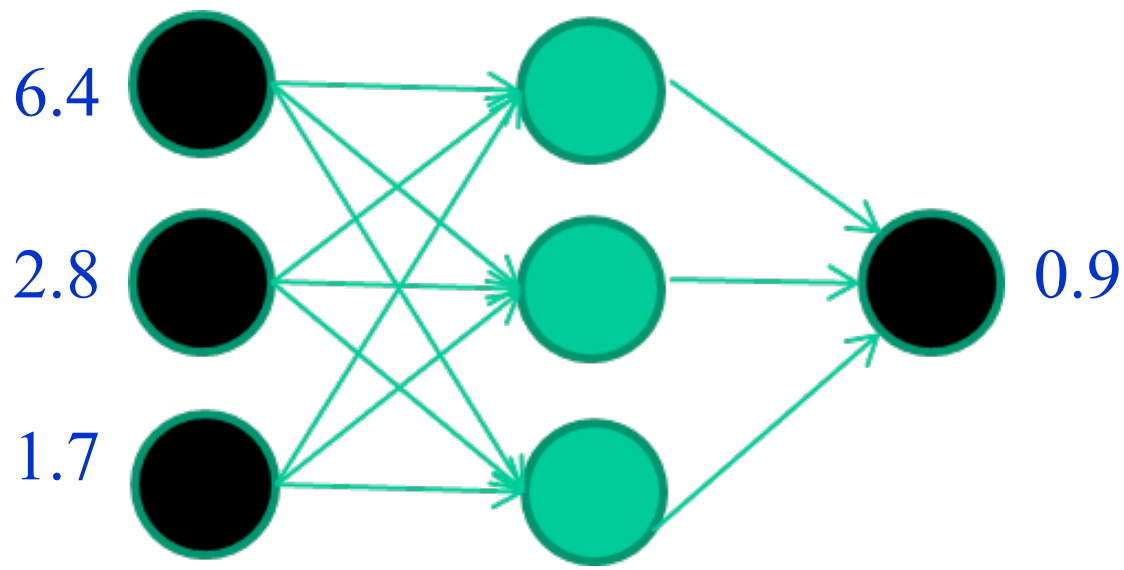
Present a training pattern



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

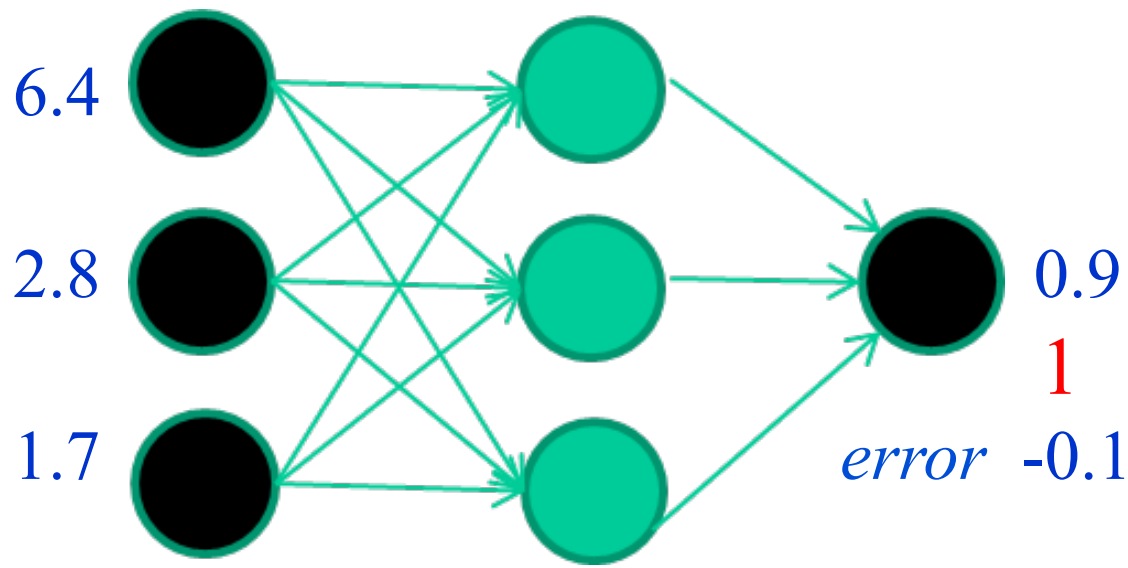
Feed it through to get output



Training data

Compare with target output

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

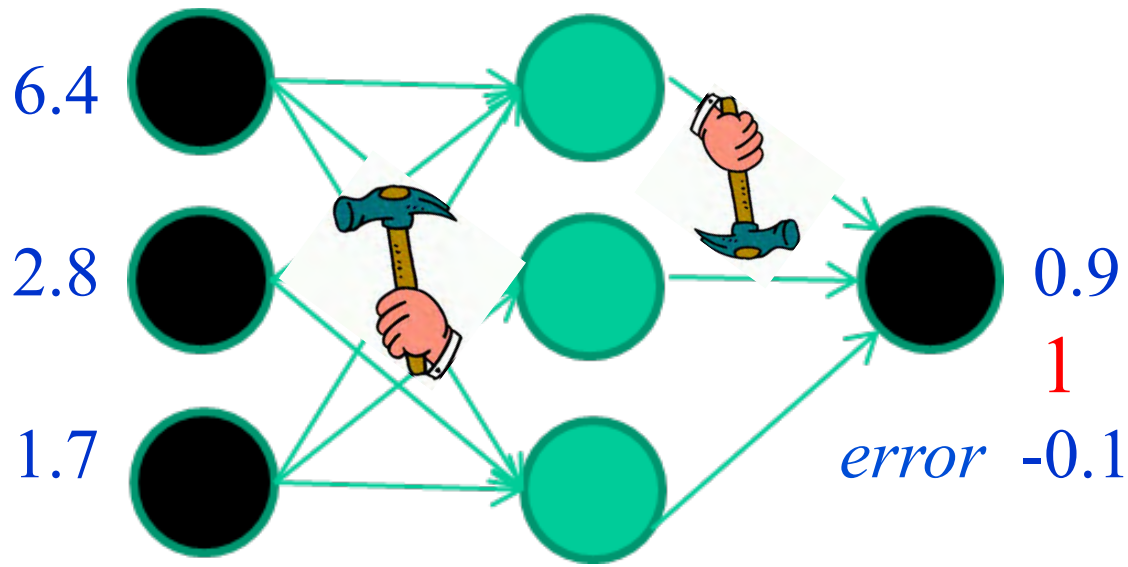


Training data

Adjust weights based on error

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

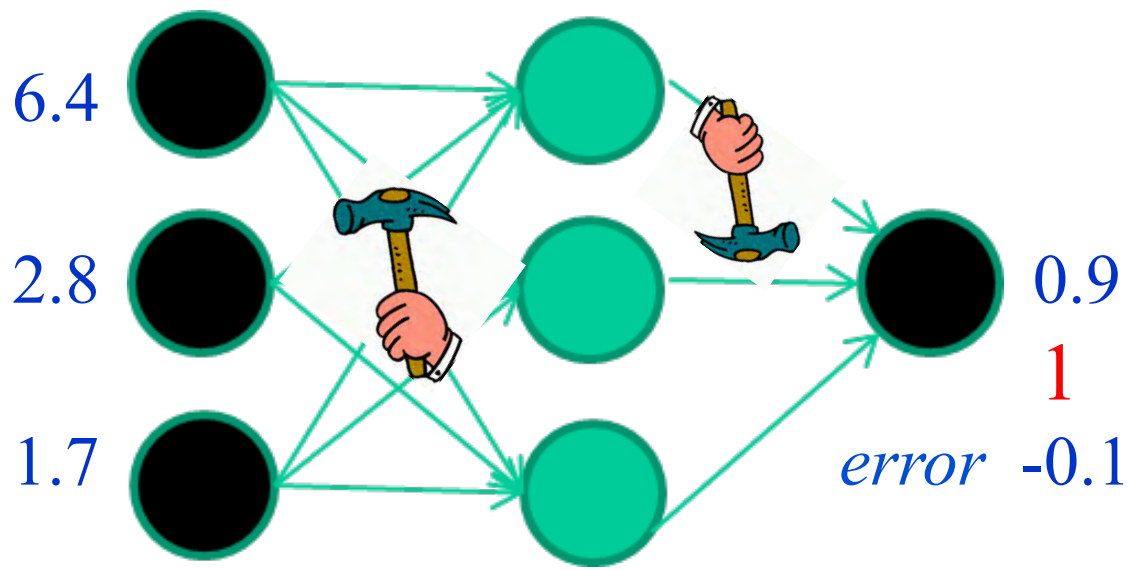
etc …



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

And so on ….

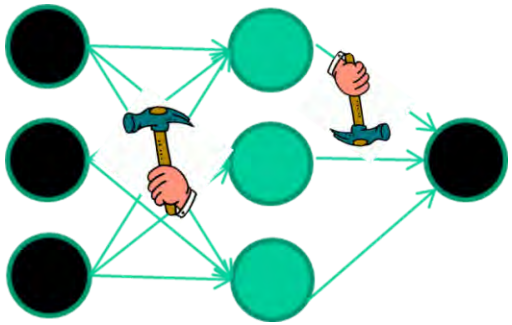


Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

Algorithms for weight adjustment are designed to make changes that will reduce the error

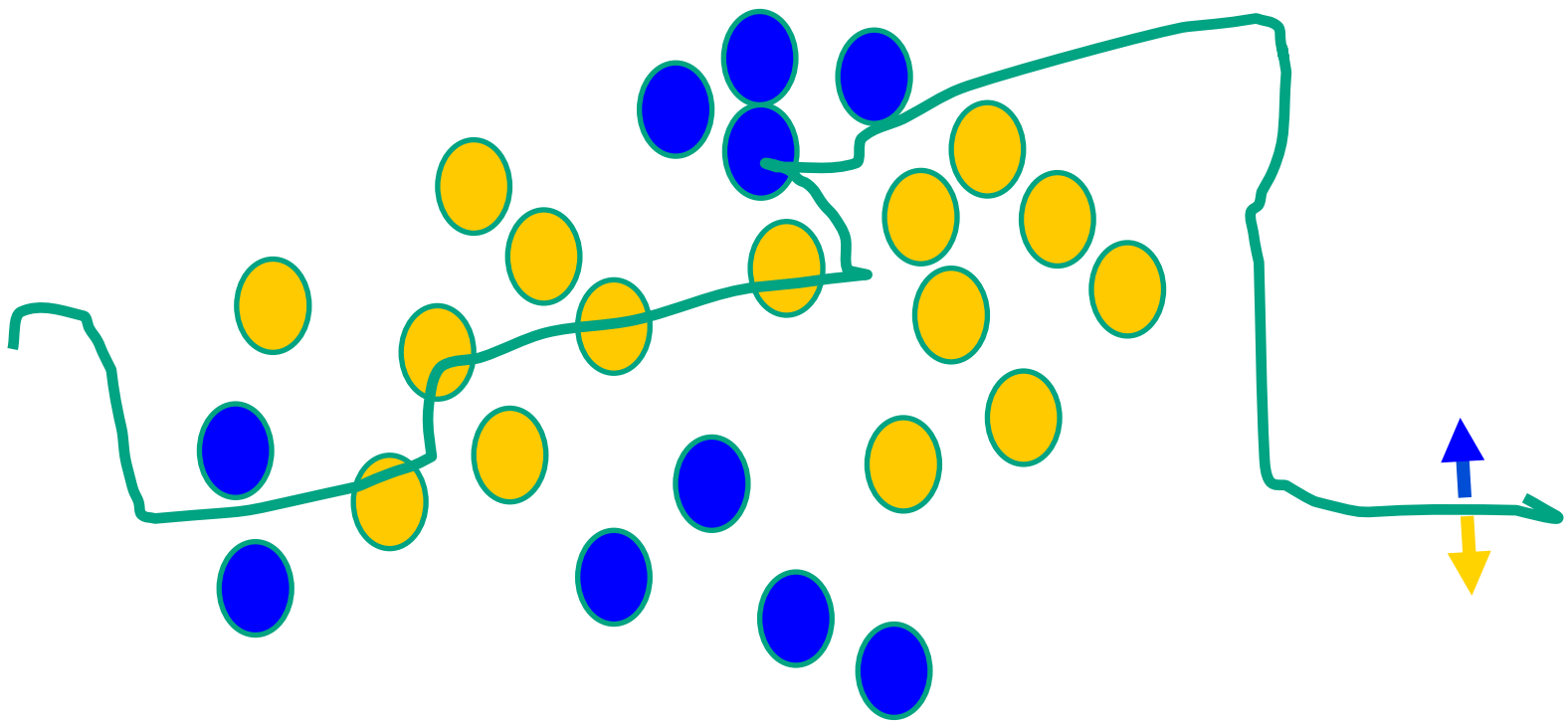
The Main Points to Remember

- weight-learning algorithms for NNs are “simple”
- they work by making thousands and thousands of tiny adjustments, each making the network do better at the most recent pattern, but perhaps a little worse on many others
- but, by “luck”, eventually this tends to be good enough to learn effective classifiers for many real applications



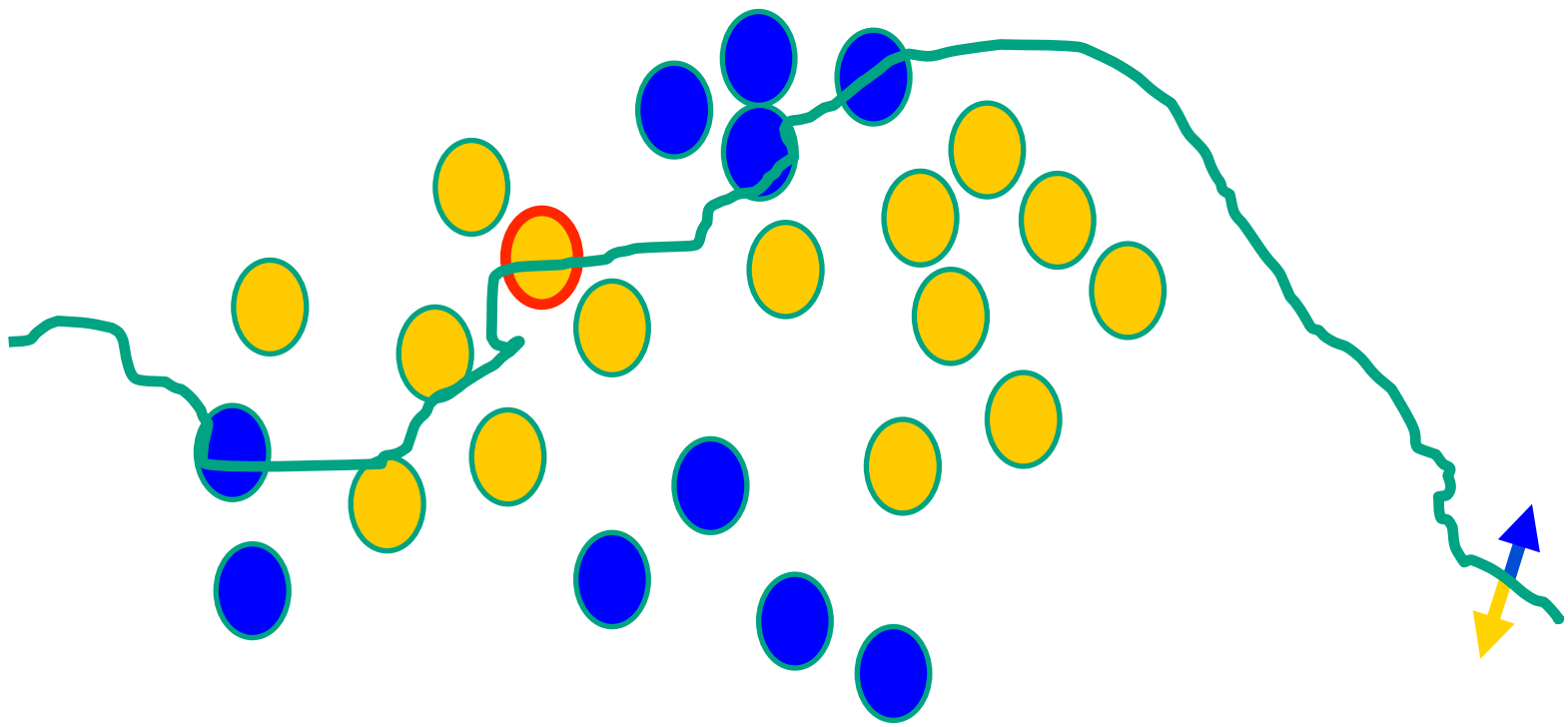
The Decision Boundary Perspective

Initial random weights



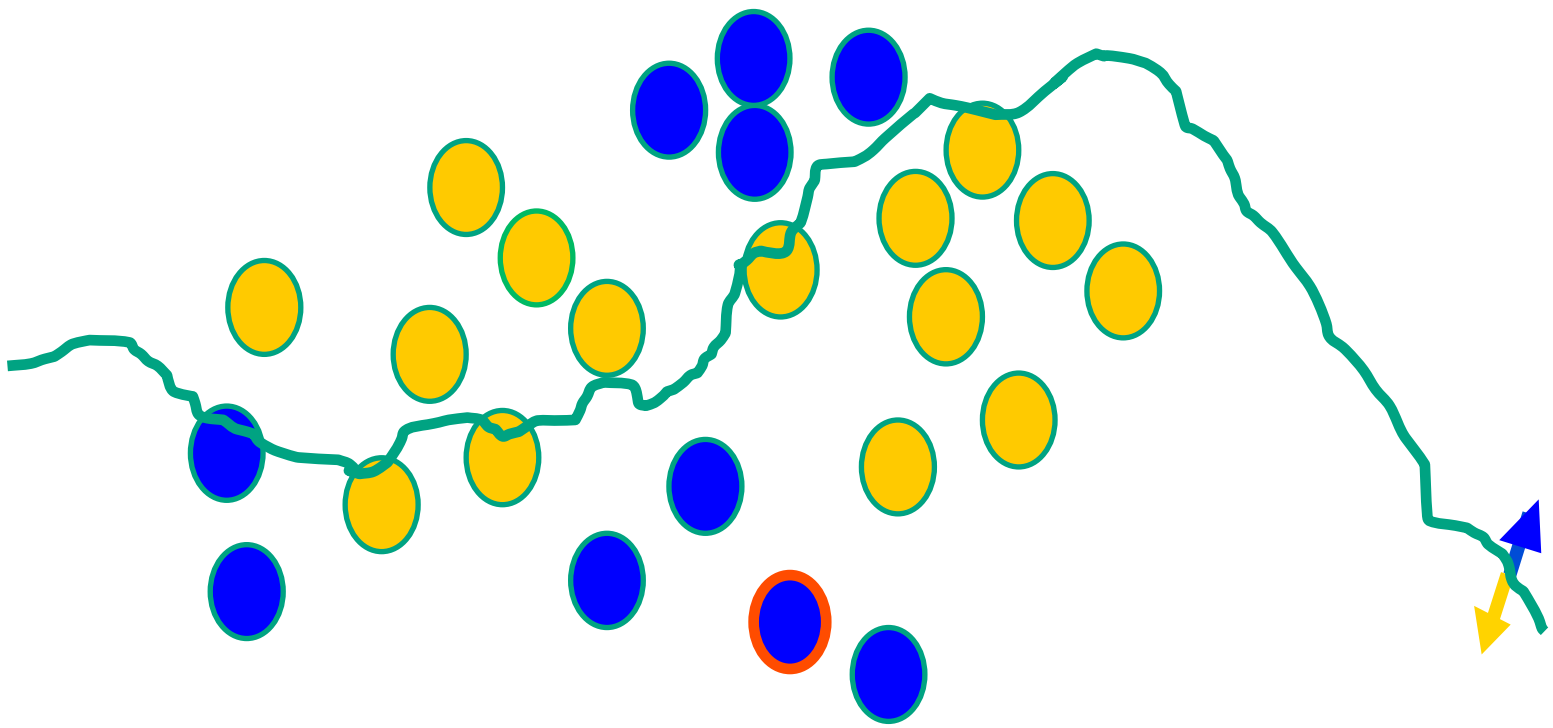
The Decision Boundary Perspective

Present a training instance / adjust the weights



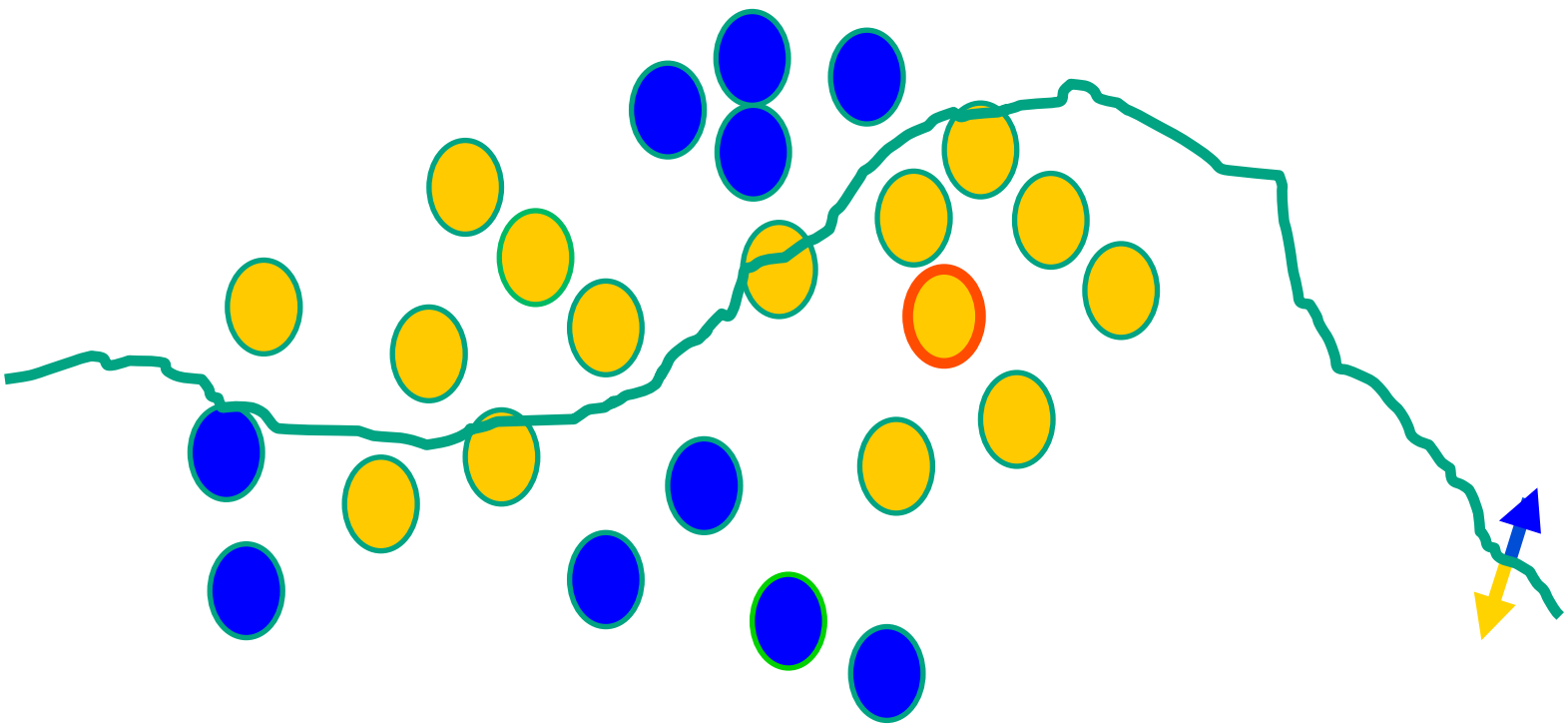
The Decision Boundary Perspective

Present a training instance / adjust the weights



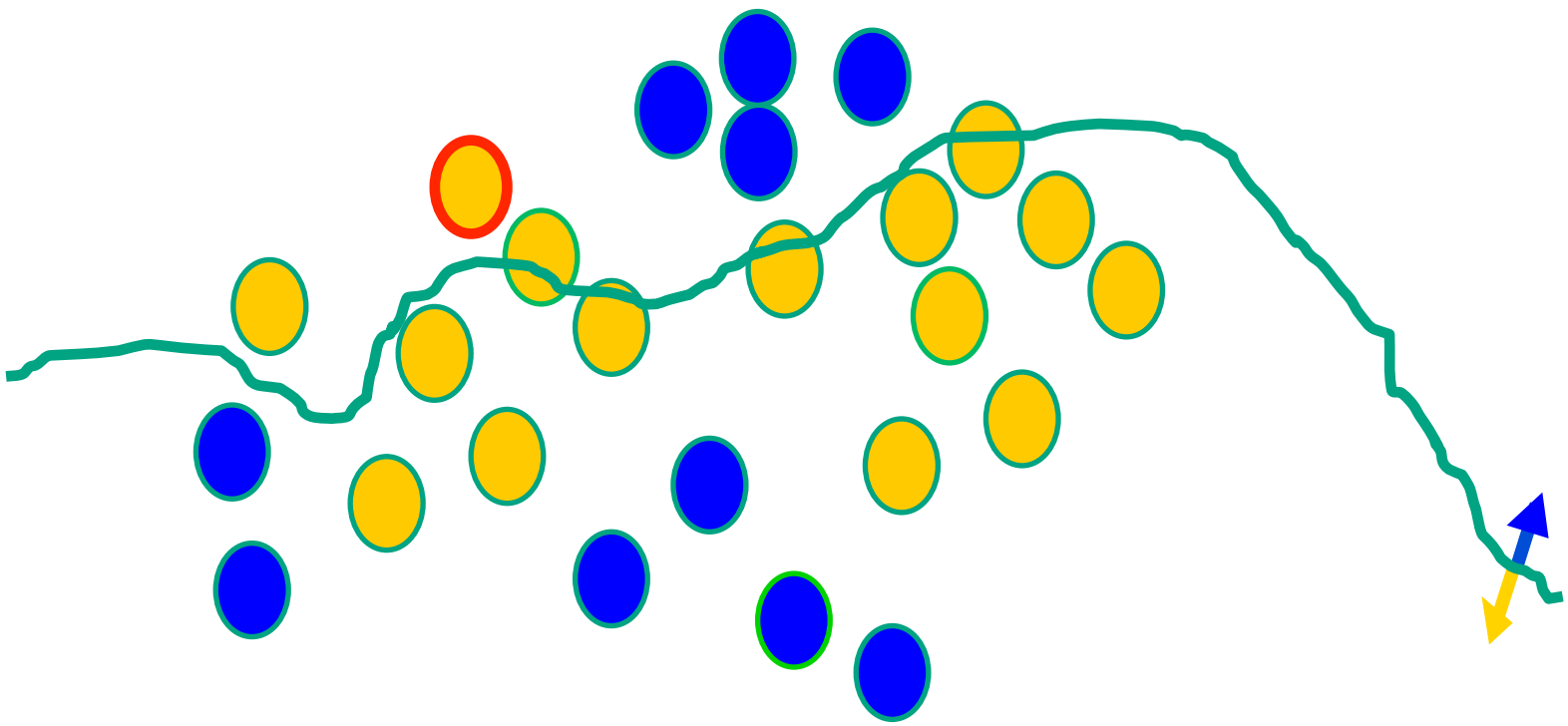
The Decision Boundary Perspective

Present a training instance / adjust the weights



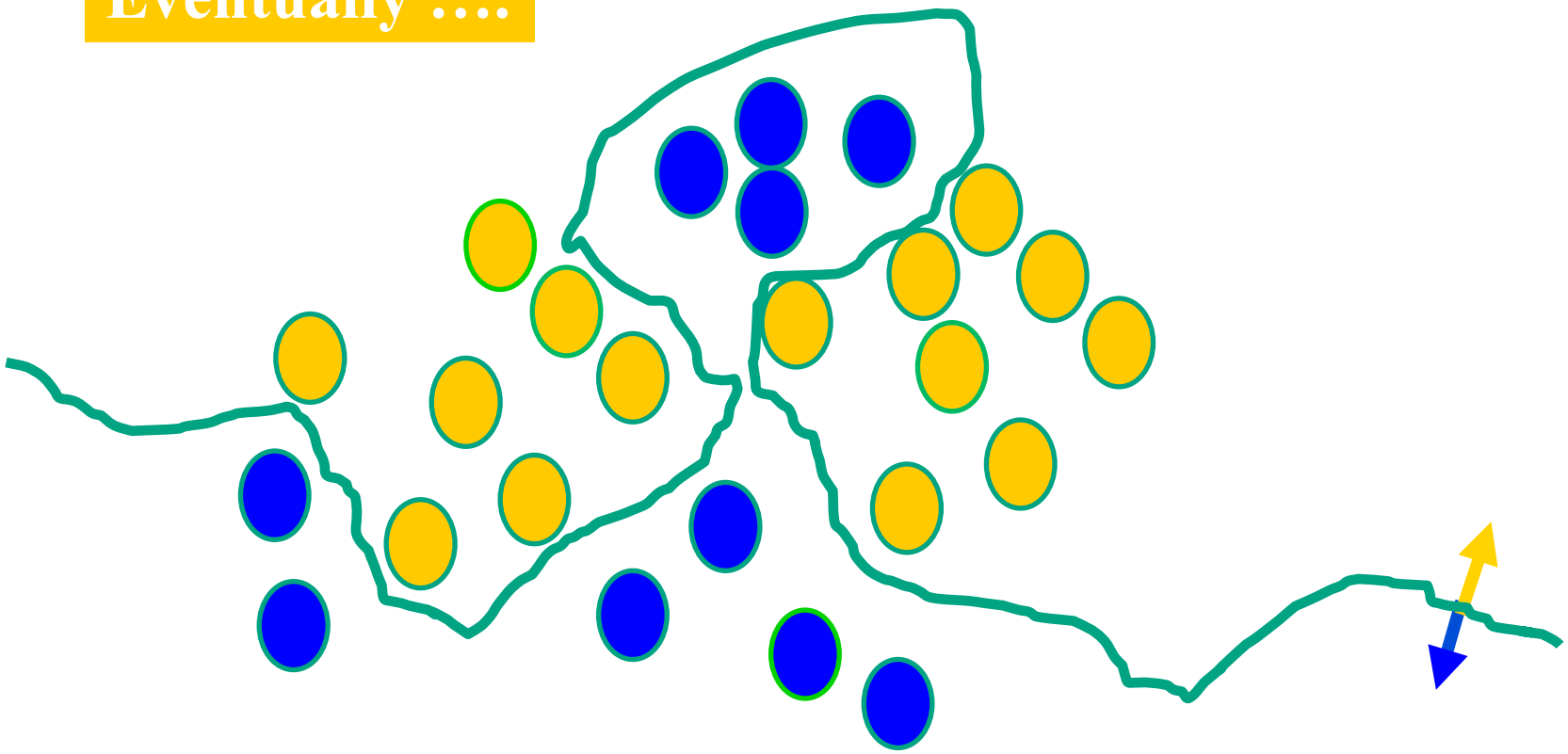
The Decision Boundary Perspective

Present a training instance / adjust the weights



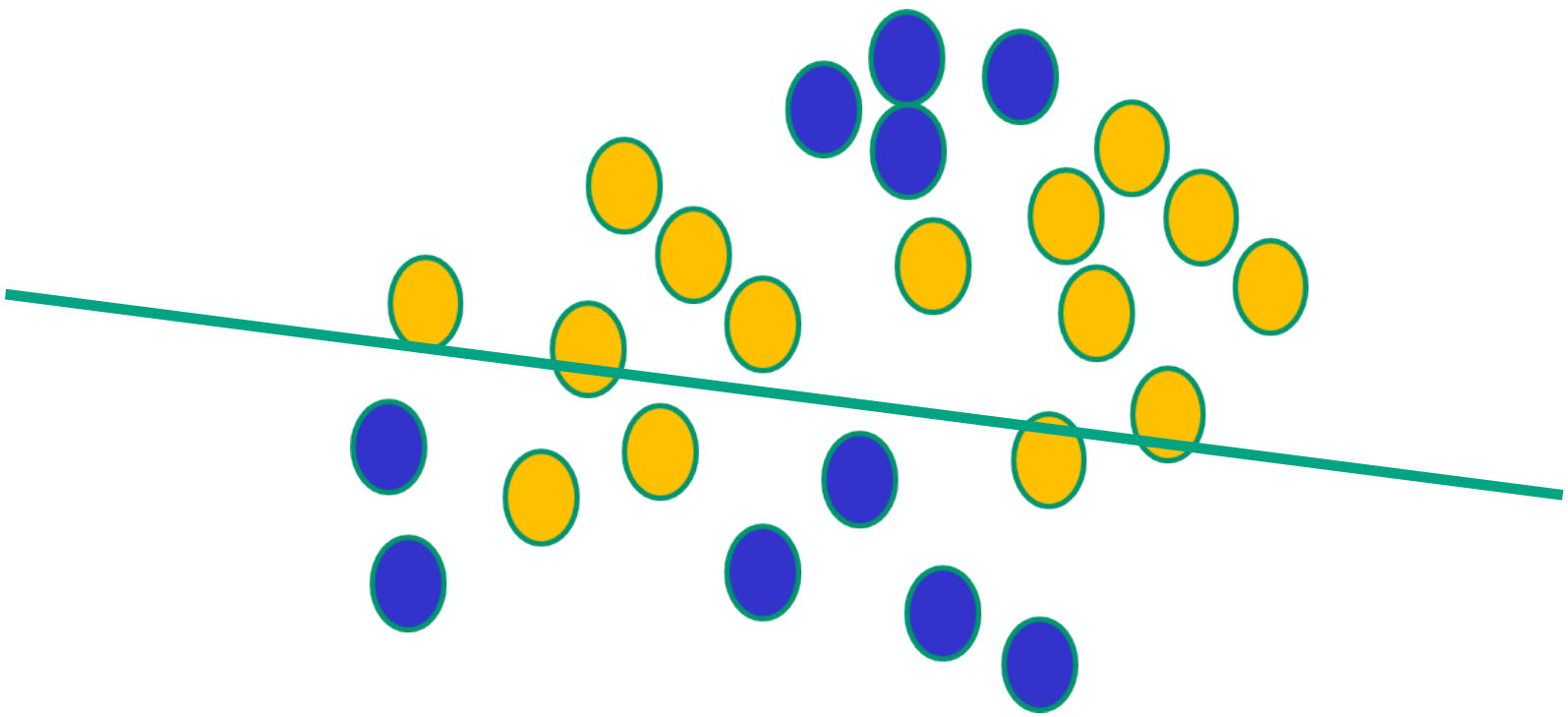
The Decision Boundary Perspective

Eventually ….

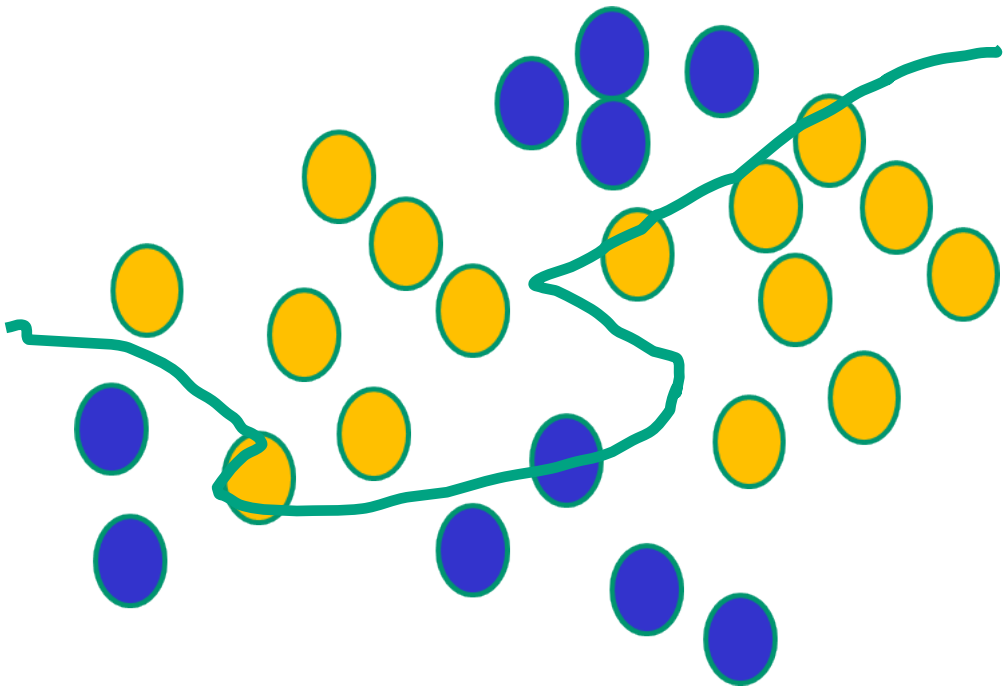


Linear Activation Funtions

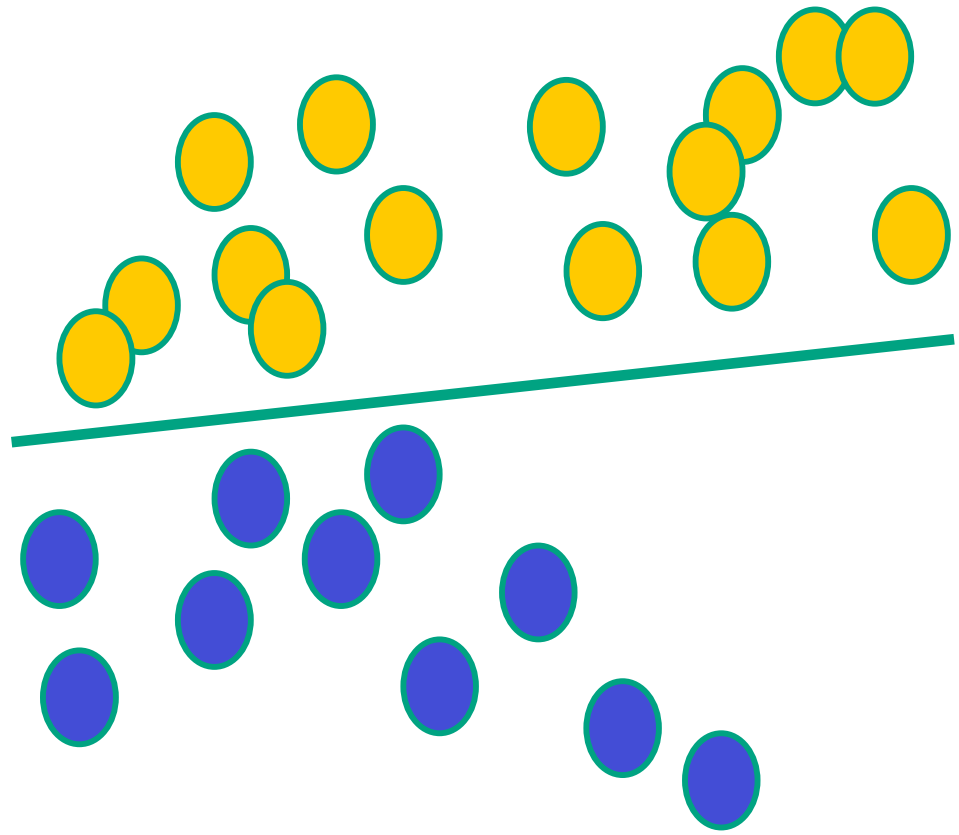
If $f(x)$ is linear, the NN can ***only*** draw straight decision boundaries (even if there are many layers of units)



NNs use nonlinear $f(x)$ so they can draw complex boundaries, but keep the data unchanged



SVMs only draw straight lines, but they transform the data first in a way that makes that OK



Limitations of Neural Networks

Random initialization + **densely connected networks** lead to:

- High cost
  - Each neuron in the neural network can be considered as a logistic regression.
  - Training the entire neural network is to train all the interconnected logistic regressions.
- Difficult to train as the number of hidden layers increases
  - Recall that logistic regression is trained by gradient descent.
  - In backpropagation, gradient is progressively getting more dilute. That is, below top layers, the correction signal δn is minimal.
- Stuck in local optima
  - The objective function of the neural network is usually not convex.
  - The random initialization does not guarantee starting from the proximity of global optima.
- Solution
  - Deep Learning/Learning multiple levels of representation

What exactly is Deep Learning ?

Why is it generally better than other methods on image, speech and certain other types of data?

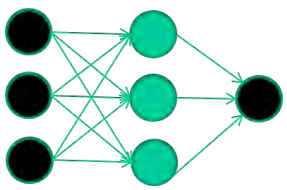
The short answer:

"Deep Learning" means using a **neural network** with **several layers of nodes** between input and output

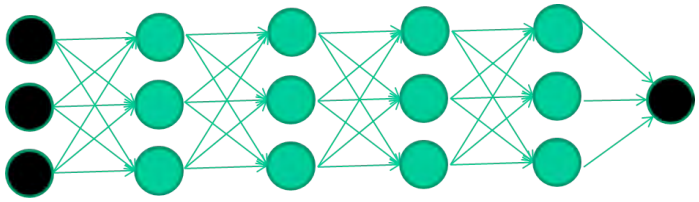
The series of layers between input & output do **feature identification and processing in a series of stages,** just as our brains seem to.

Multi-layer neural networks have been around for about 25 years... So, what's actually new?

We have always had good algorithms for learning the weights in networks with 1 hidden layer

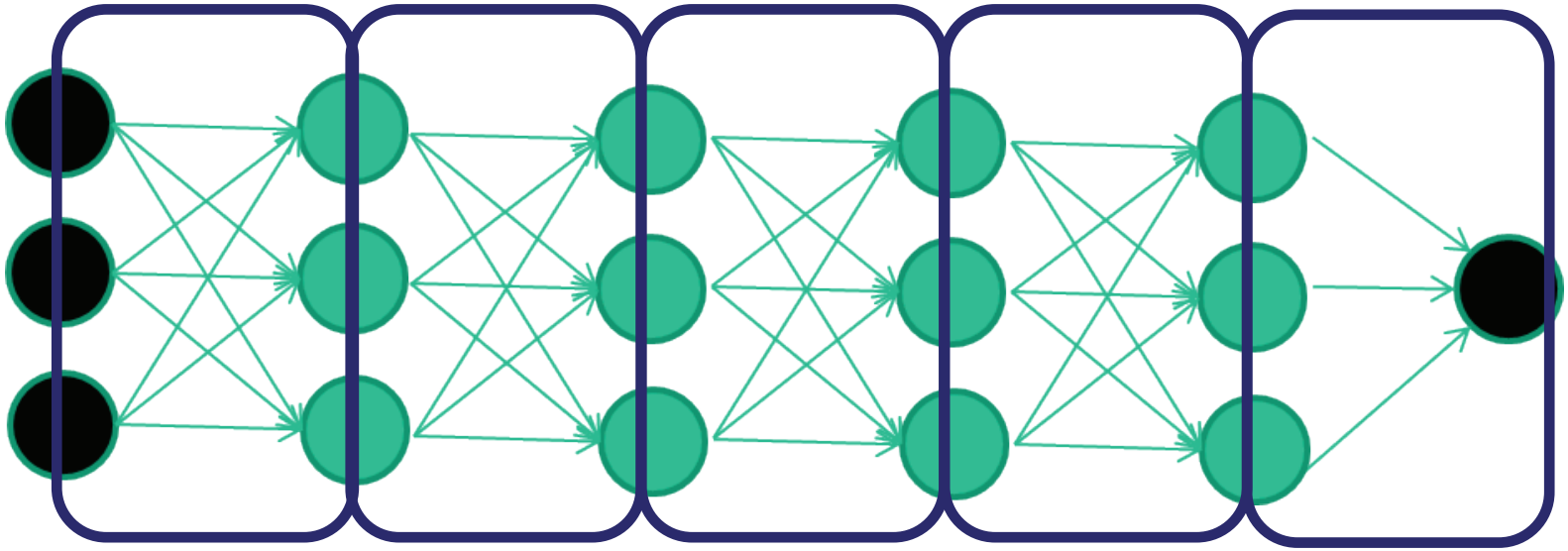


But these algorithms are not good at learning the weights for networks with more hidden layers



What's new is: **<u>algorithms for training many-layer networks</u>**

How to Train a Multi-Layer Network



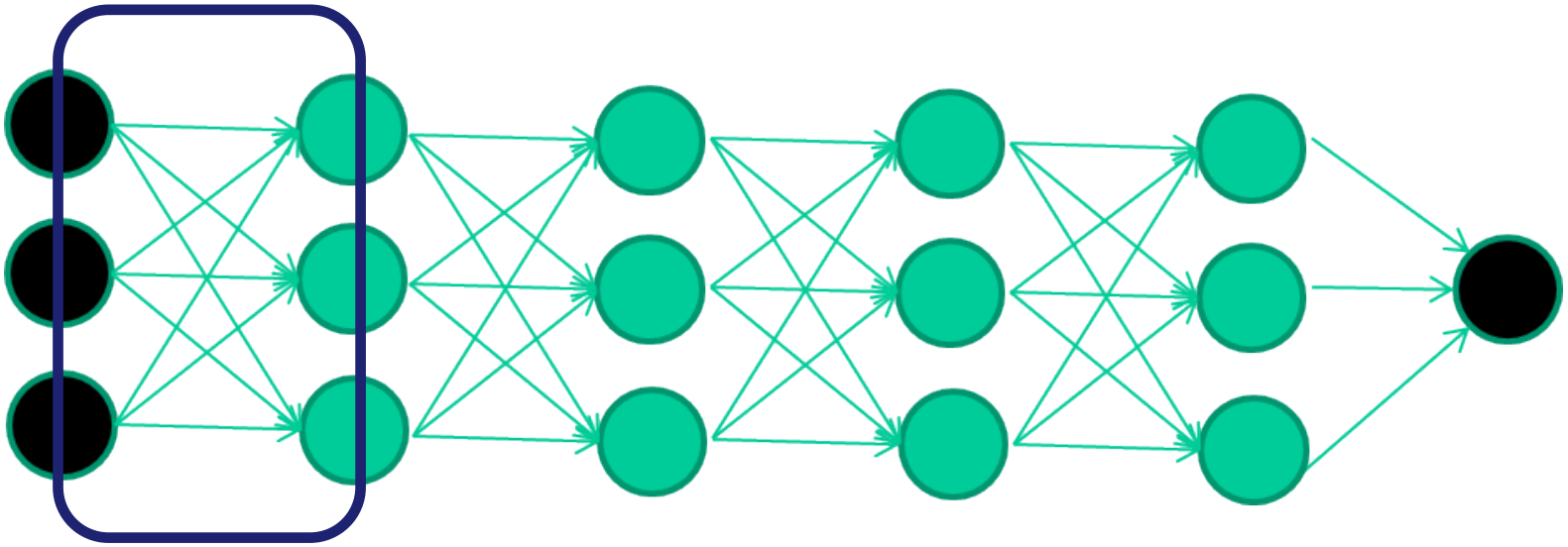
Train **this** layer first

then **this** layer

then **this** layer

then **this** layer

finally **this** layer



EACH of the (non-output) layers is trained to be an ***auto-encoder.***

Basically, it is forced to learn good features that describe what comes from the previous layer.

Networks for Deep Learning

- ***Deep Belief Networks*** and ***Autoencoders*** employs layer-wise unsupervised learning to initialize each layer and capture multiple levels of representation simultaneously.
  - Hinton, G. E, Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527-1554.
  - Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks, Advances in Neural Information Processing Systems 19
- ***Convolutional Neural Network*** organizes neurons based on animal's visual cortex system, which allows for learning patterns at both local level and global level.
  - Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

Yann LeCun

(60, born in Paris,
now lives in NYC)

LeNet image recognition

inventor of backpropagation methods for training, and of convolutional neural nets

current director of Artificial Intellegence at Facebook

Deep Belief Networks

- A *deep belief network* (DBN) is a probabilistic, generative model made up of multiple layers of hidden units.
  - A composition of simple learning modules that make up each layer
- A DBN can be used to generatively pre-train a DNN by using the learned DBN weights as the initial DNN weights.
  - Back-propagation or other discriminative algorithms can then be applied for fine-tuning of these weights.
- Advantages:
  - Particularly helpful when limited training data are available
  - These pre-trained weights are closer to the optimal weights than are randomly chosen initial weights.

Convolutional Neural Networks

- Convolutional Neural Networks are inspired by mammalian visual cortex.
  - The visual cortex contains a complex arrangement of cells, which are sensitive to small sub-regions of the visual field, called a receptive field. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images.
  - Two basic cell types:
    - Simple cells respond maximally to specific edge-like patterns within their receptive field.
    - Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.

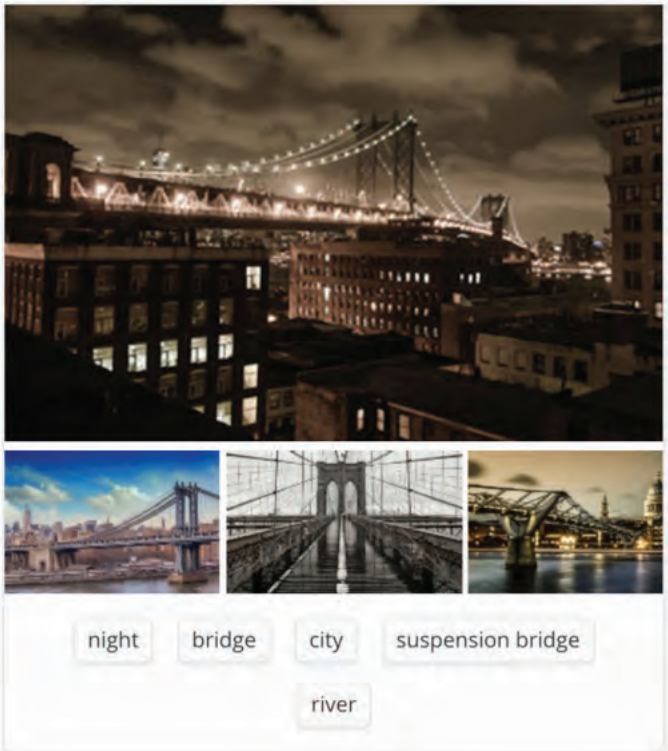
"Godfathers of AI" Awarded 2018 Turing Prize

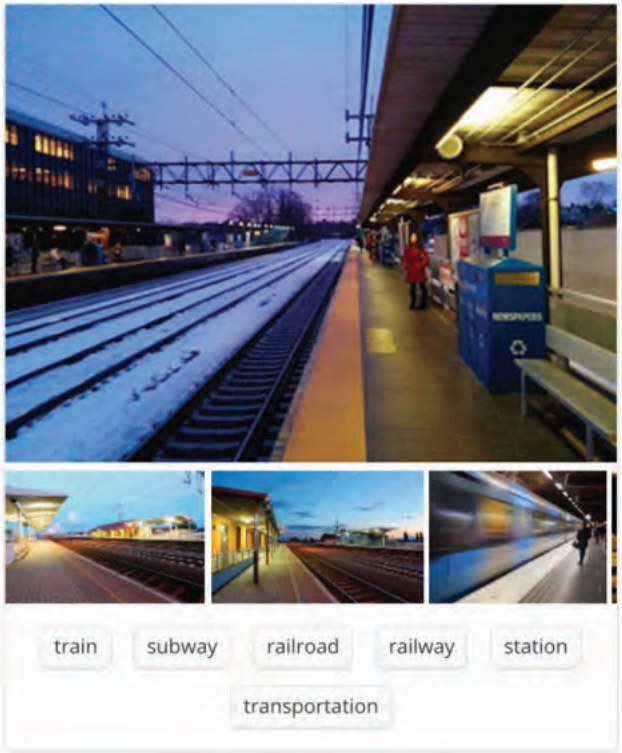
Yann LeCun
(Facebook & New York Univ.)

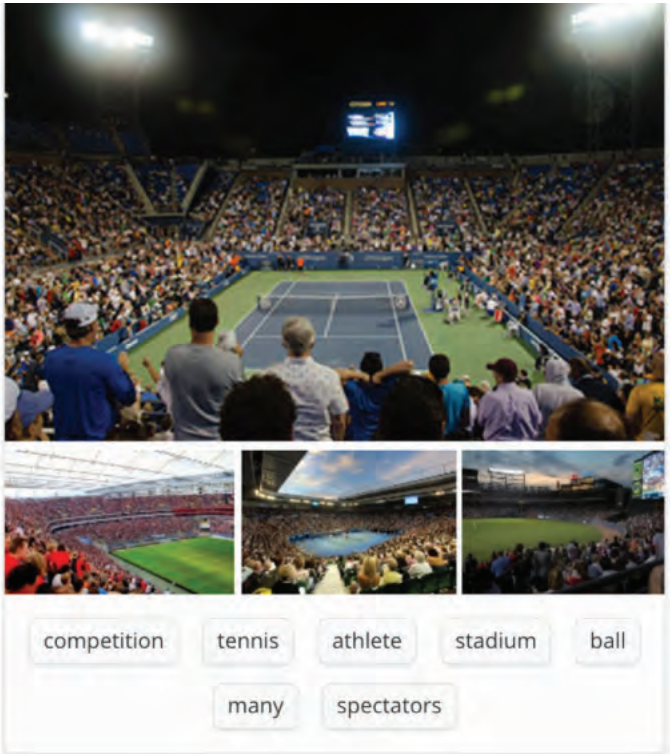
Geoff Hinton
(Google & Univ. Toronto)

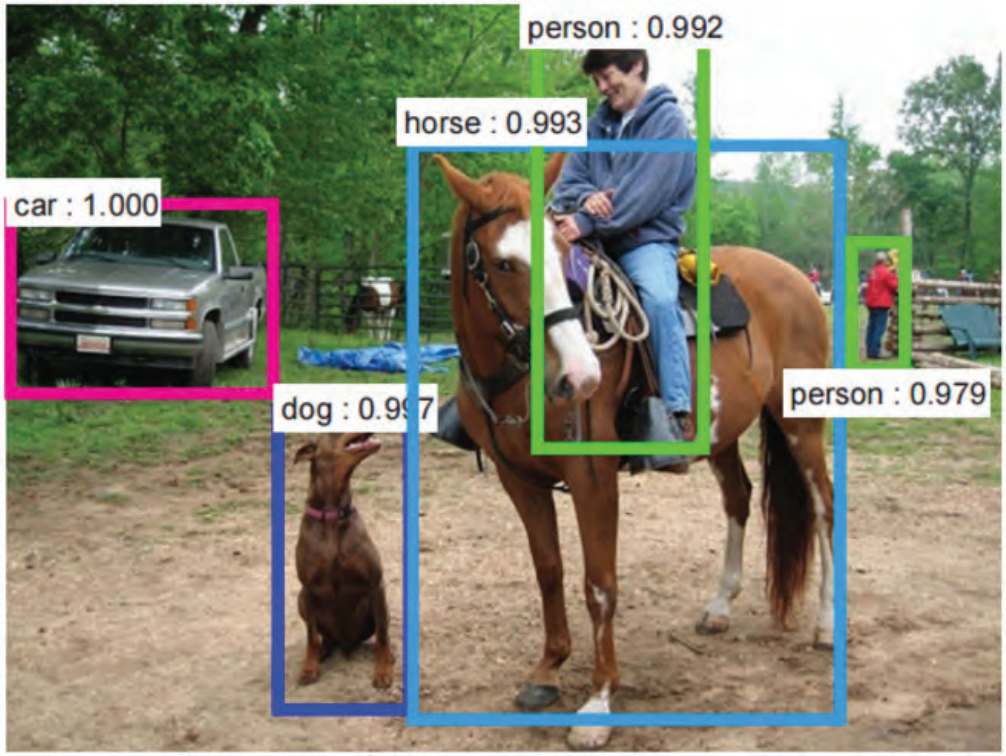
Yoshua Bengio
(Element AI & Univ. Montreal)

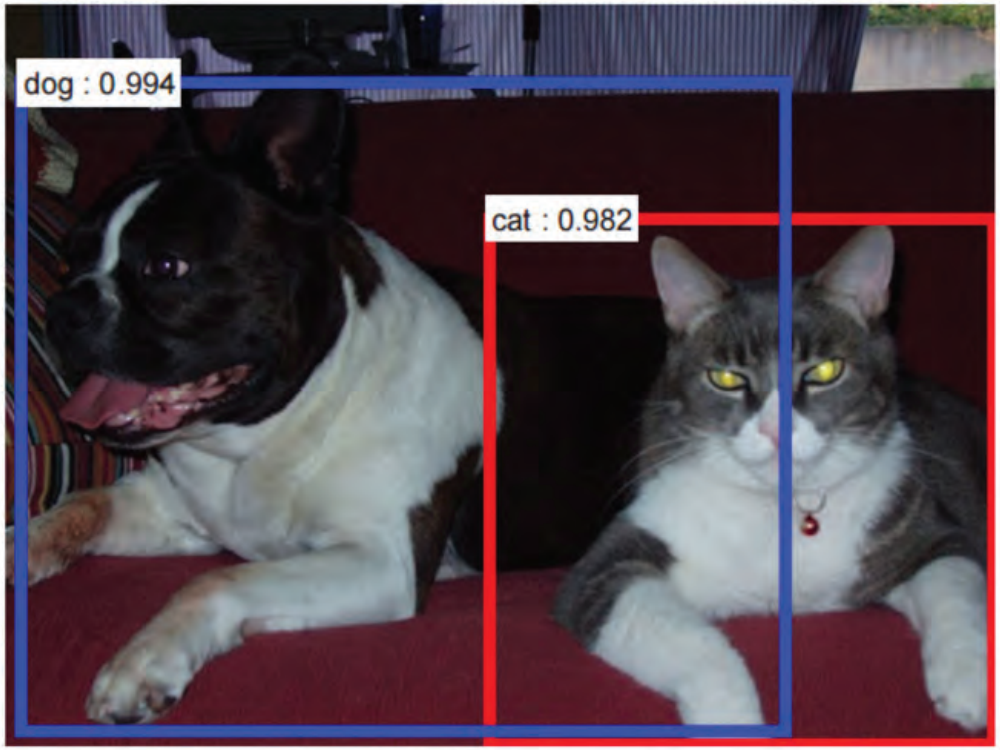
Convolutional Neural Network for Image Classification

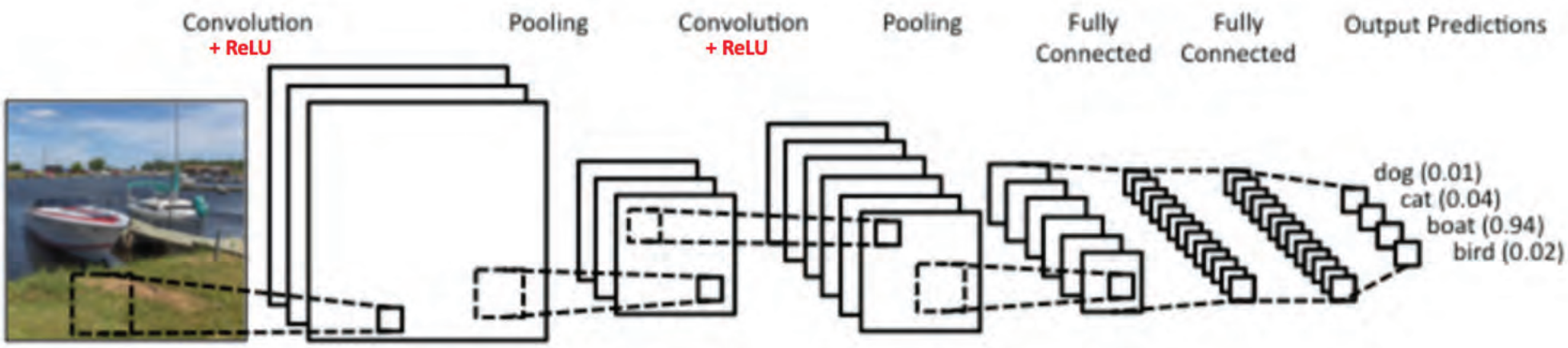




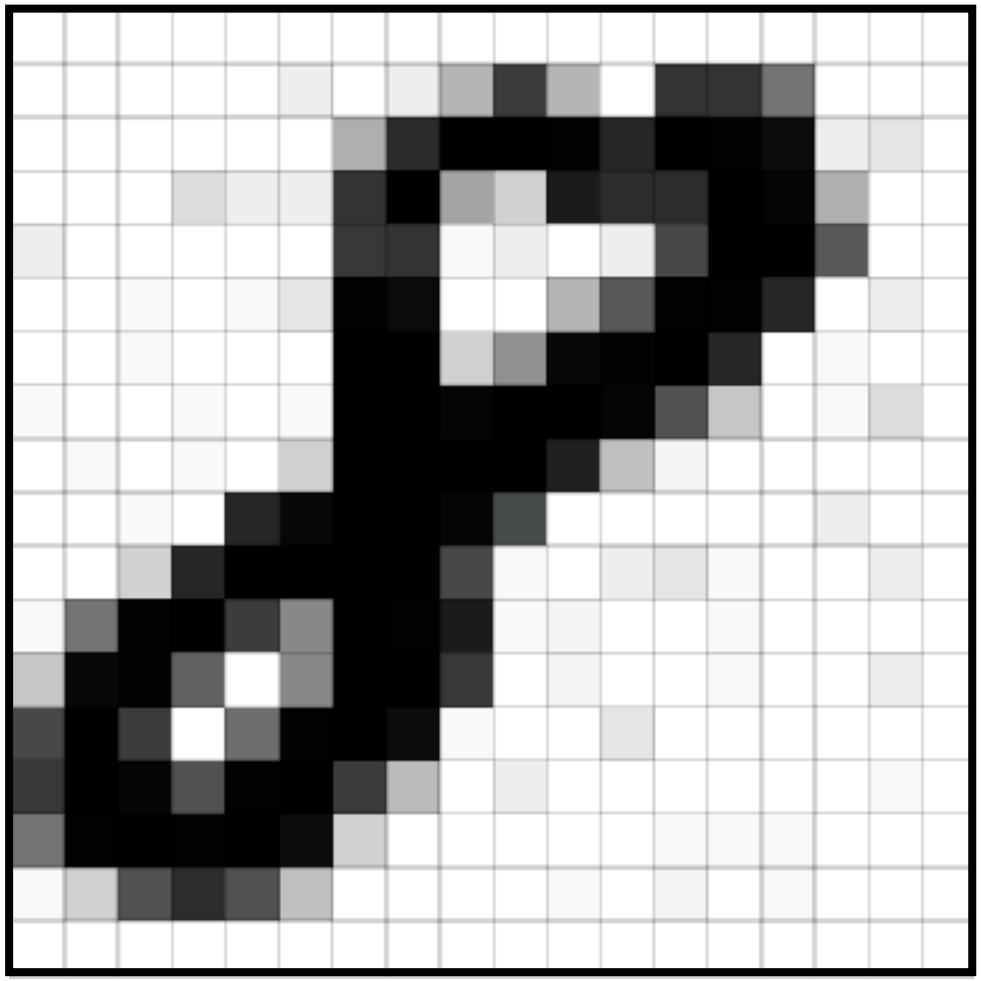



person : 0.992
horse : 0.993
car : 1.000
dog : 0.997
person : 0.979


dog : 0.994
cat : 0.982


Convolution
+ ReLU
Pooling
Convolution
+ ReLU
Pooling
Fully
Connected
Fully
Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

Representation of an Image as Pixels



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

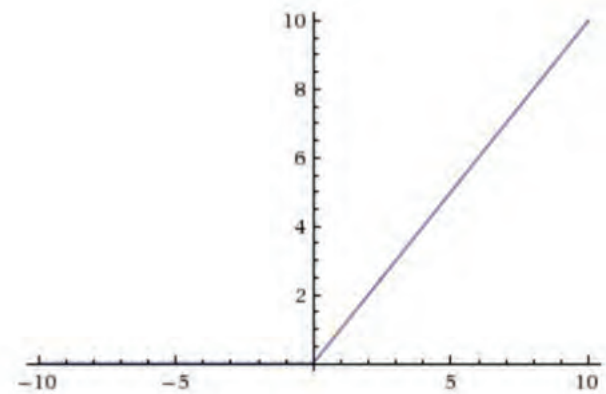
1	0	1
0	1	0
1	0	1

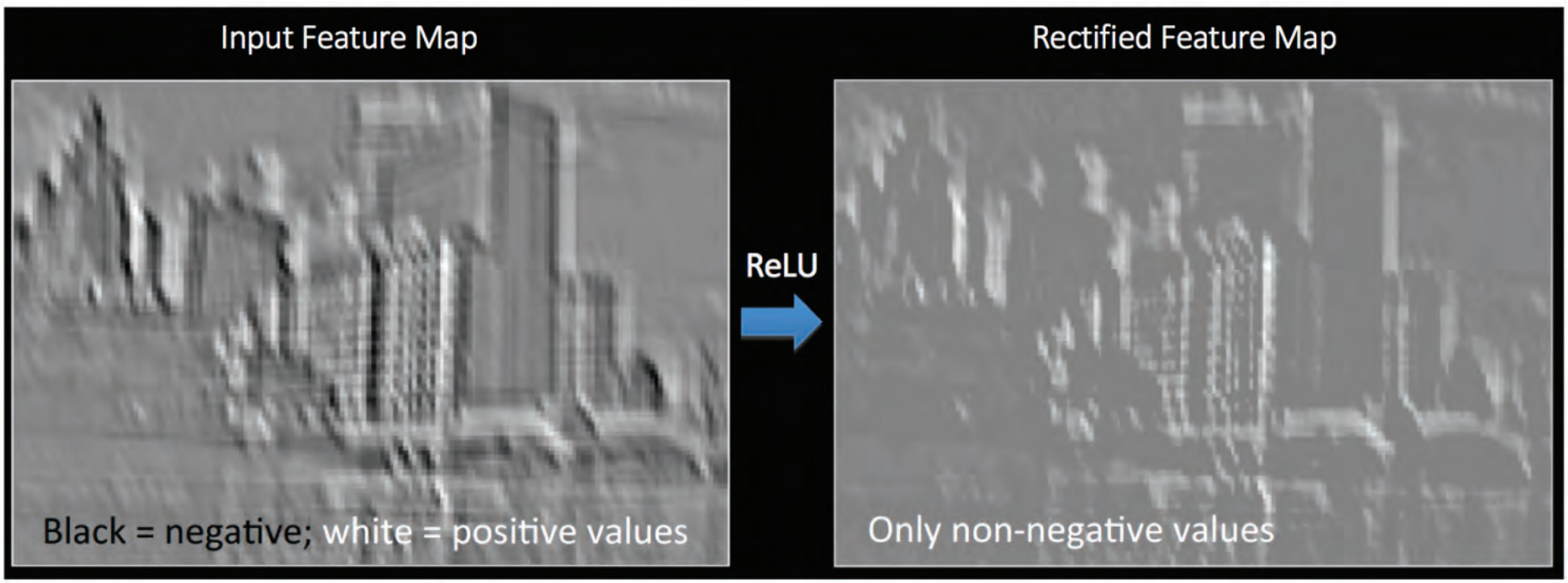
Filter

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

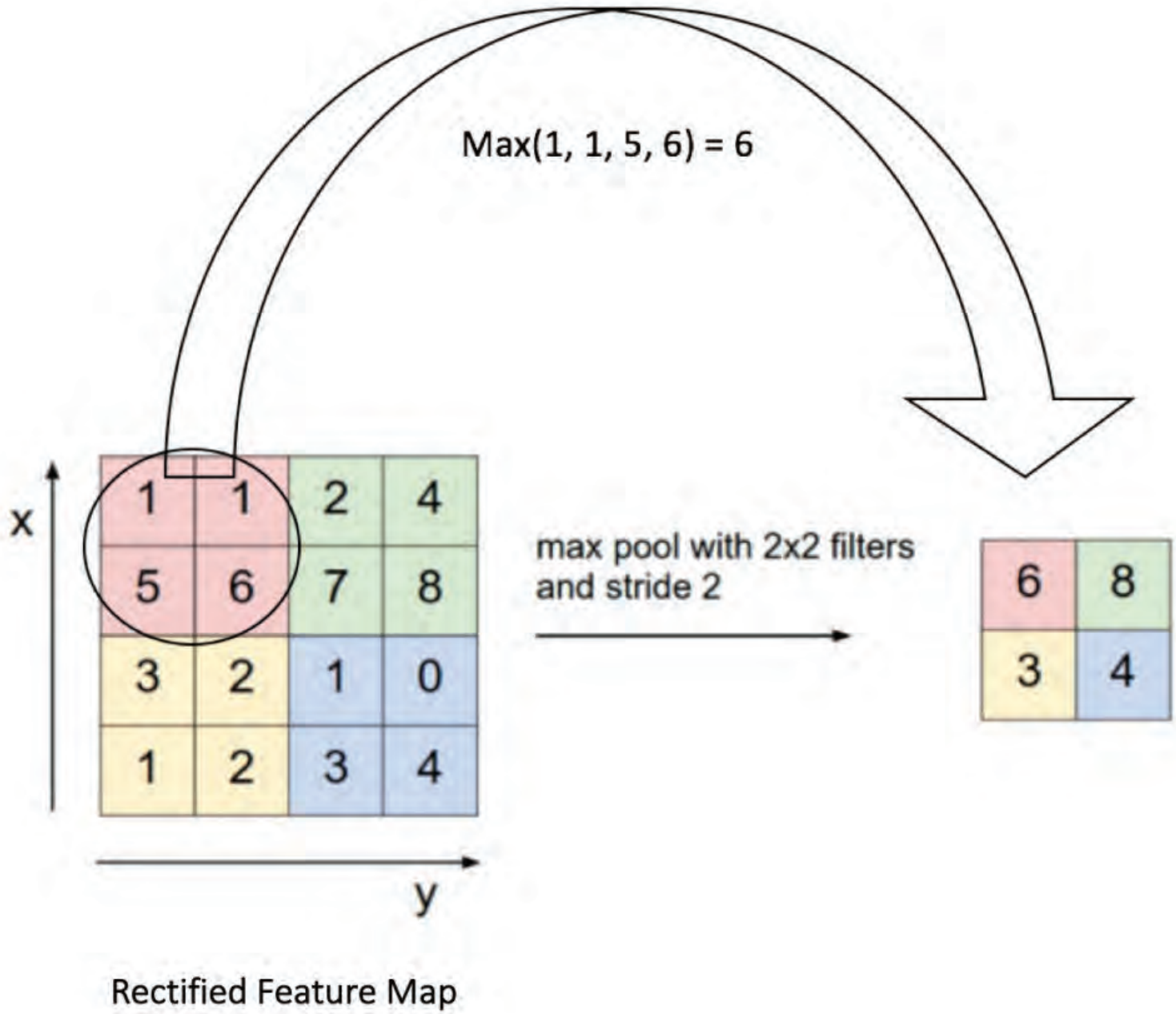
The ReLU (Rectified Linear Unit) Operation

Output = Max(zero, Input)

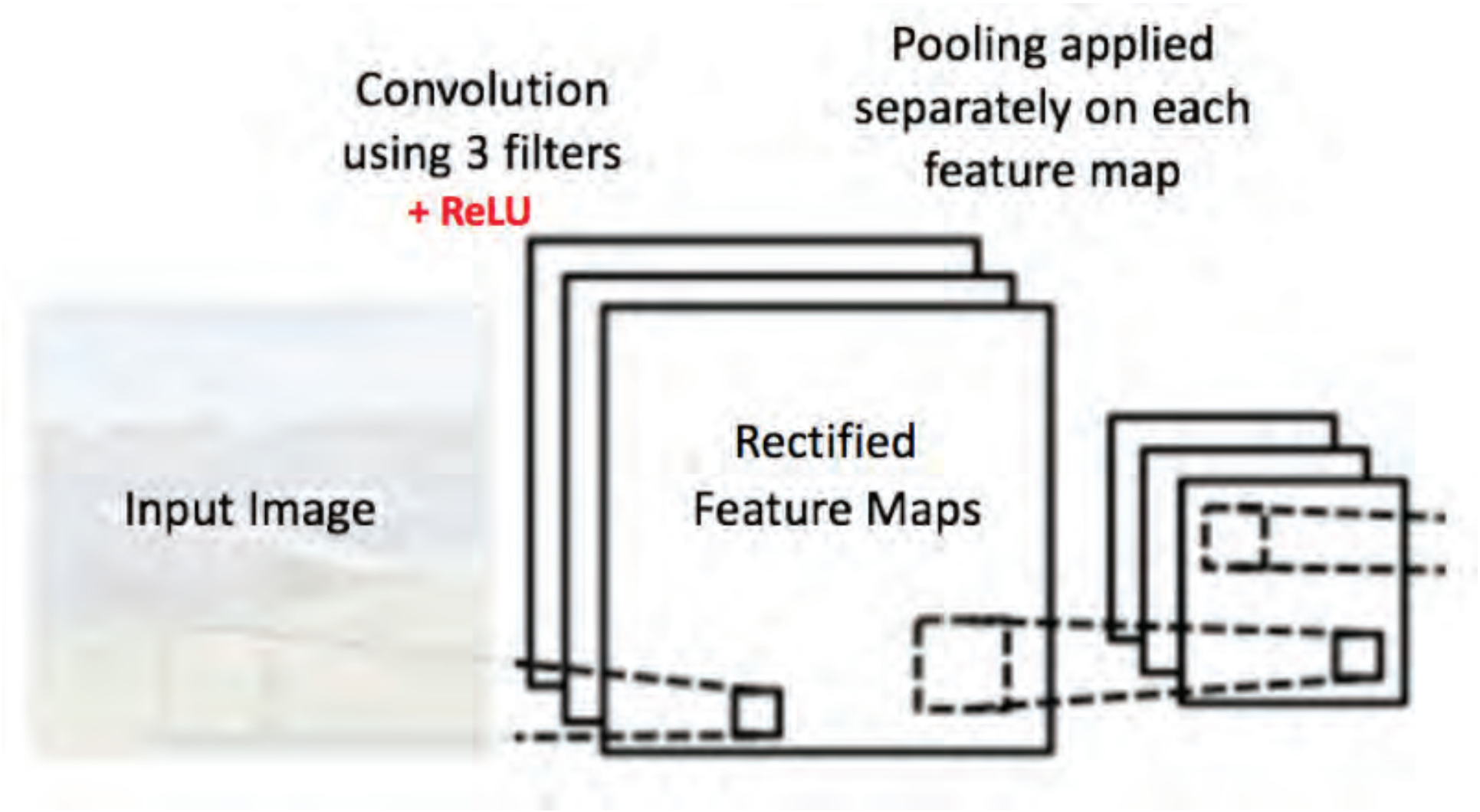




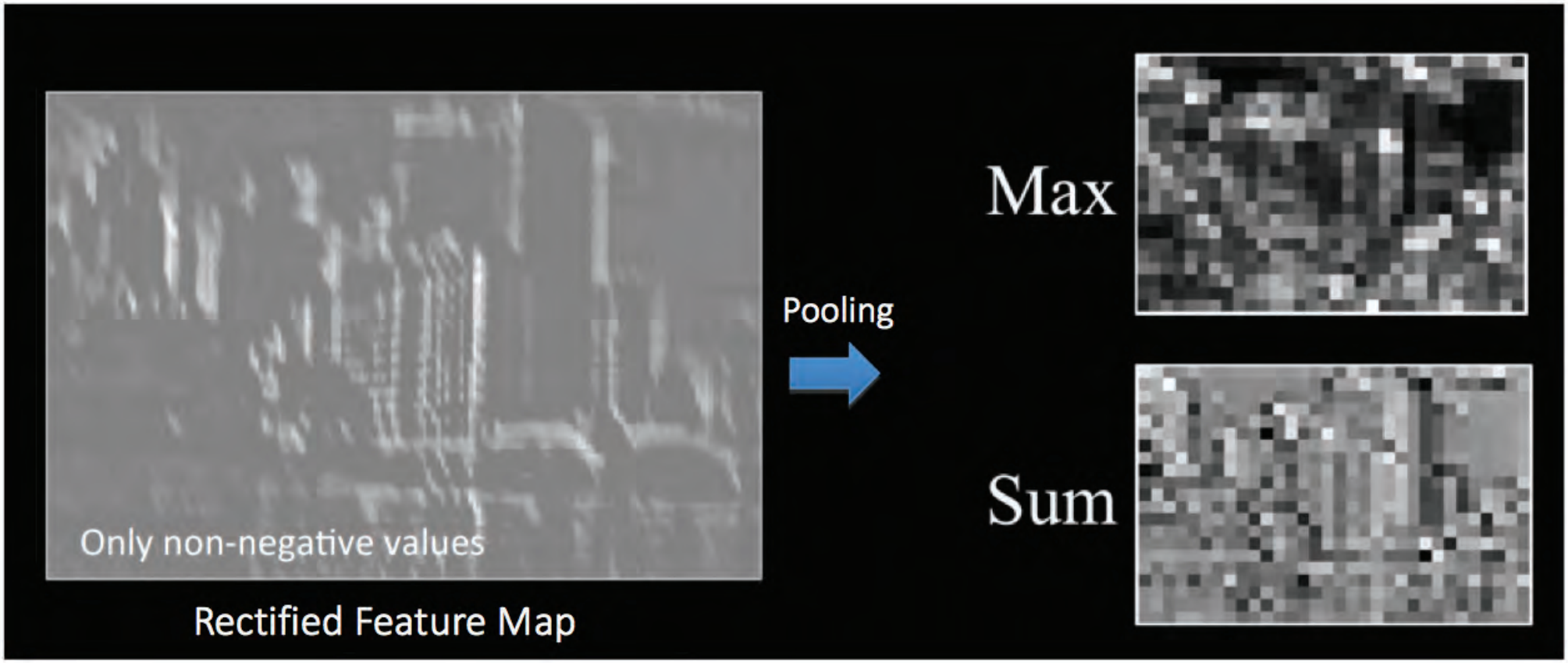
The Max Pooling Operation



Pooling Applied to Rectified Feature Maps



Pooling Applied to Rectified Feature Maps



Training of a Convolutional Neural Net

Step 1: Initialize all filters and parameters / weights with random values.

Step 2: The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.

Lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]

Since weights are randomly assigned for the first training example, output probabilities are also random.

Step 3: Calculate the total error at the output layer (summation over all 4 classes)
Total Error = $\sum \frac{1}{2}$ (target probability – output probability) ^{2}

Step 4: Use Backpropagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.

The weights are adjusted in proportion to their contribution to the total error.

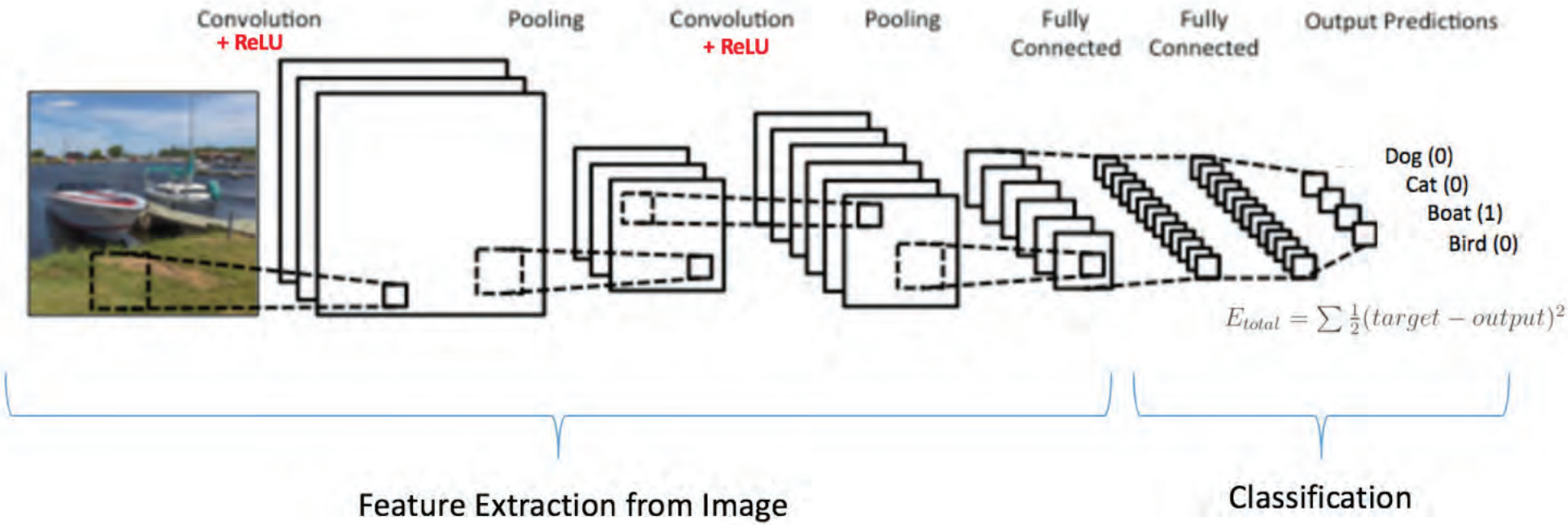
When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].

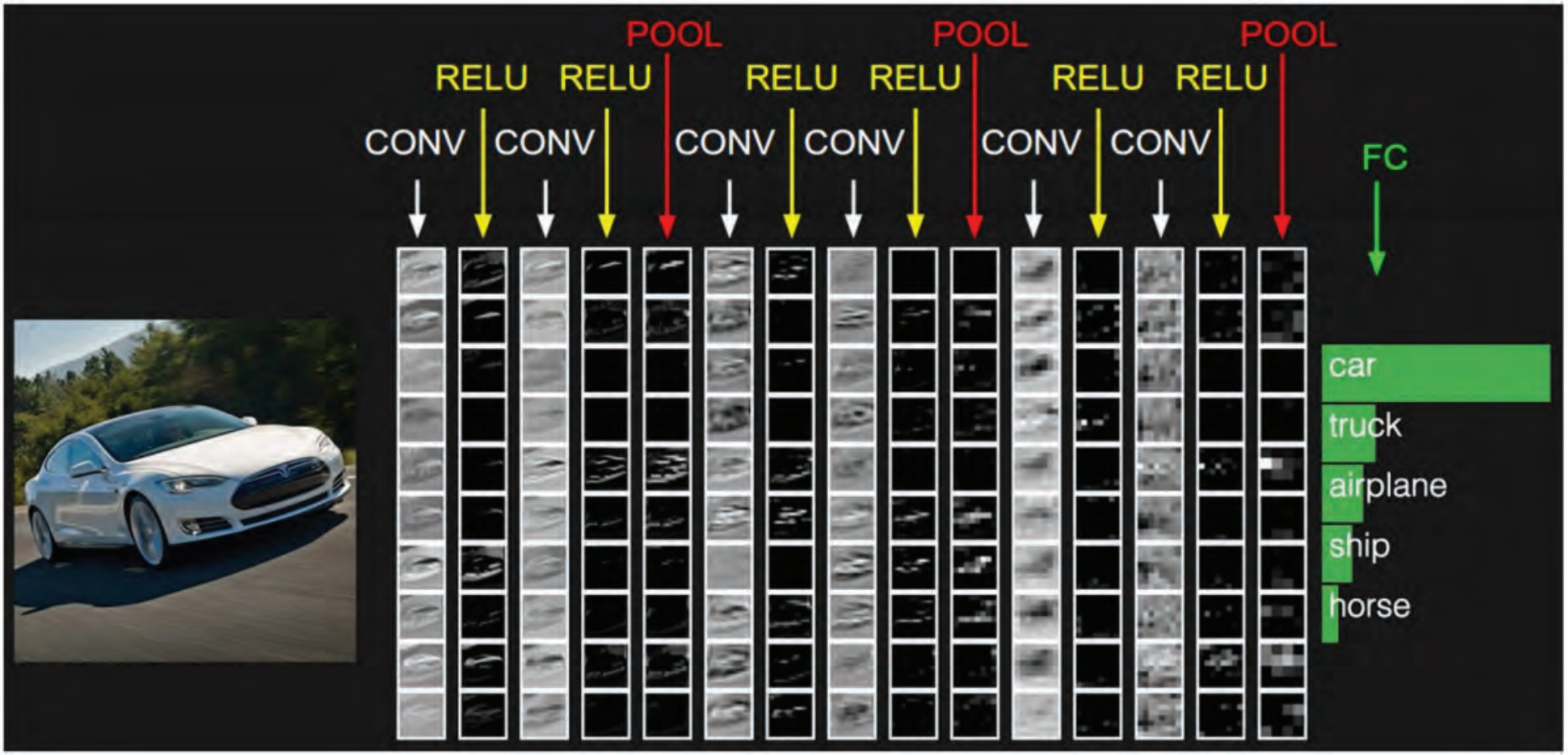
This means that the network has learnt to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.

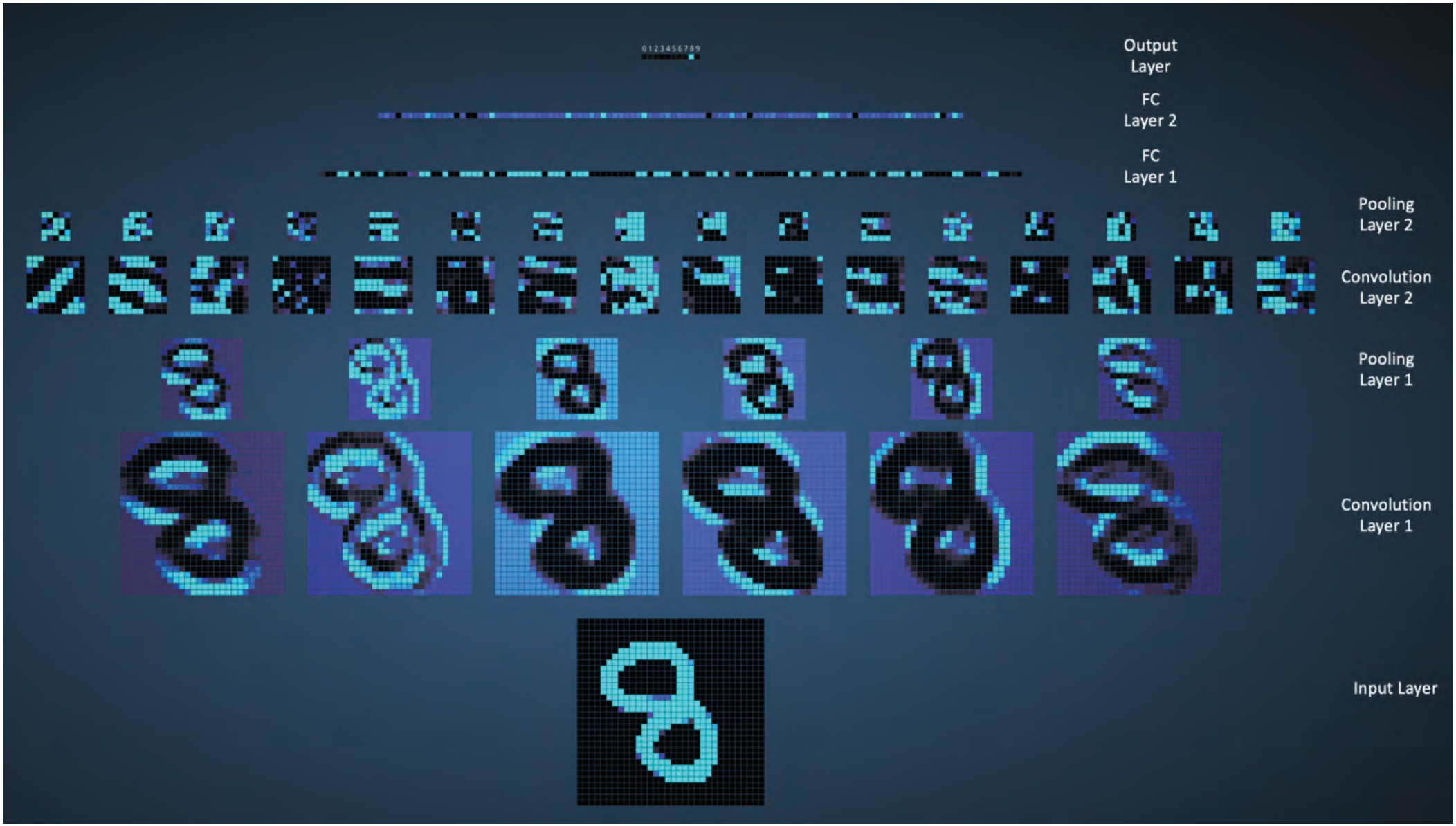
Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

Step 5: Repeat steps 2-4 with all images in the training set.

Convolutional Neural Nets: Putting It All Together




POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse


0123456789
Output Layer
FC Layer 2
FC Layer 1
Pooling Layer 2
Convolution Layer 2
Pooling Layer 1
Convolution Layer 1
Input Layer

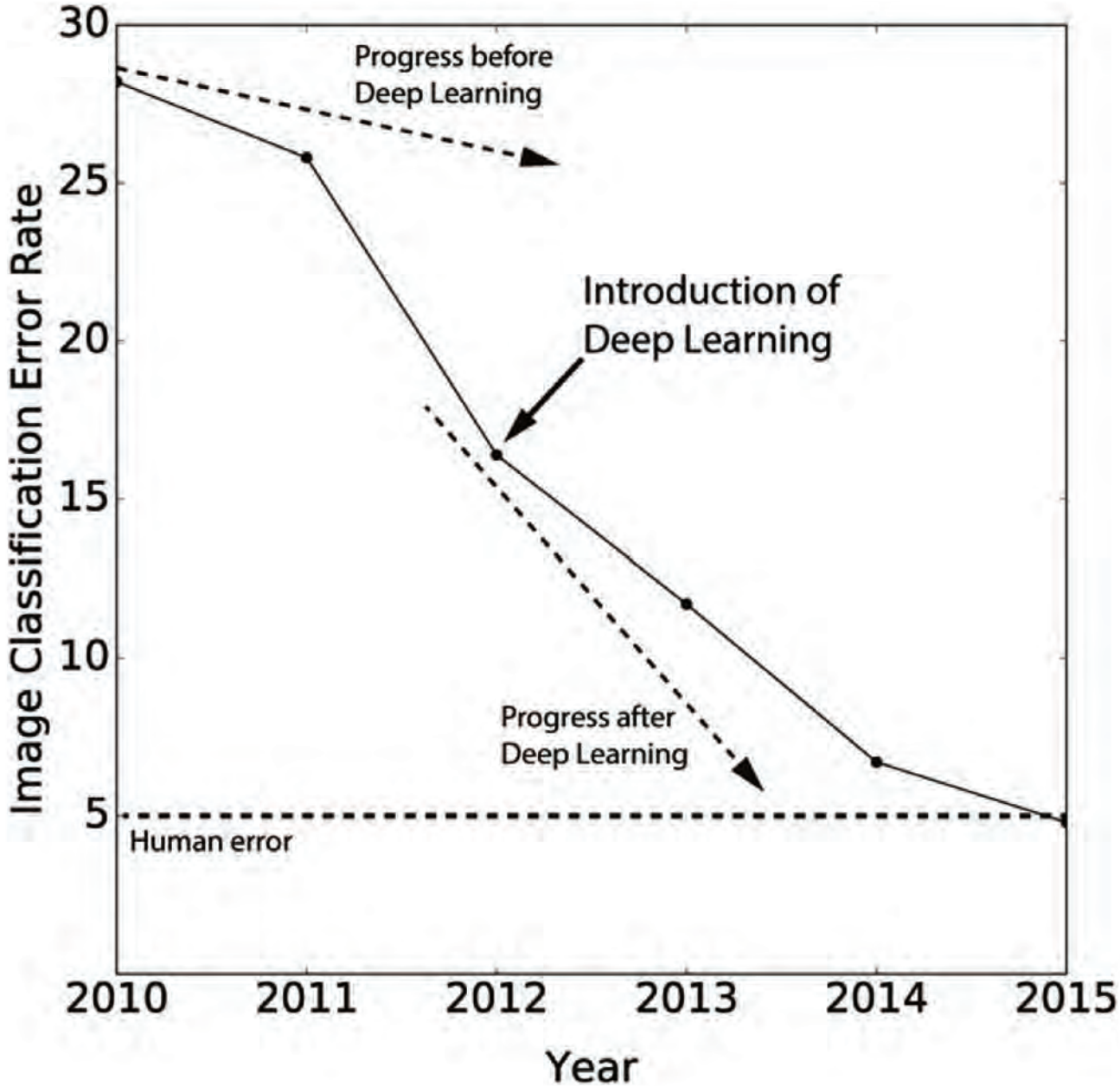


Figure 4. Historical error rate of the best performing image classification algorithms in the annual ImageNet competition.[37] Established models of computer vision stagnated at 25 30%. The introduction of deep learning in 2012 led to a significant improvement to ∼15%, and human level accu racy (∼5%) for image classification was achieved by 2015.

TensorFlow is an open source library for machine learning tasks developed by Google and first released in November 2015.

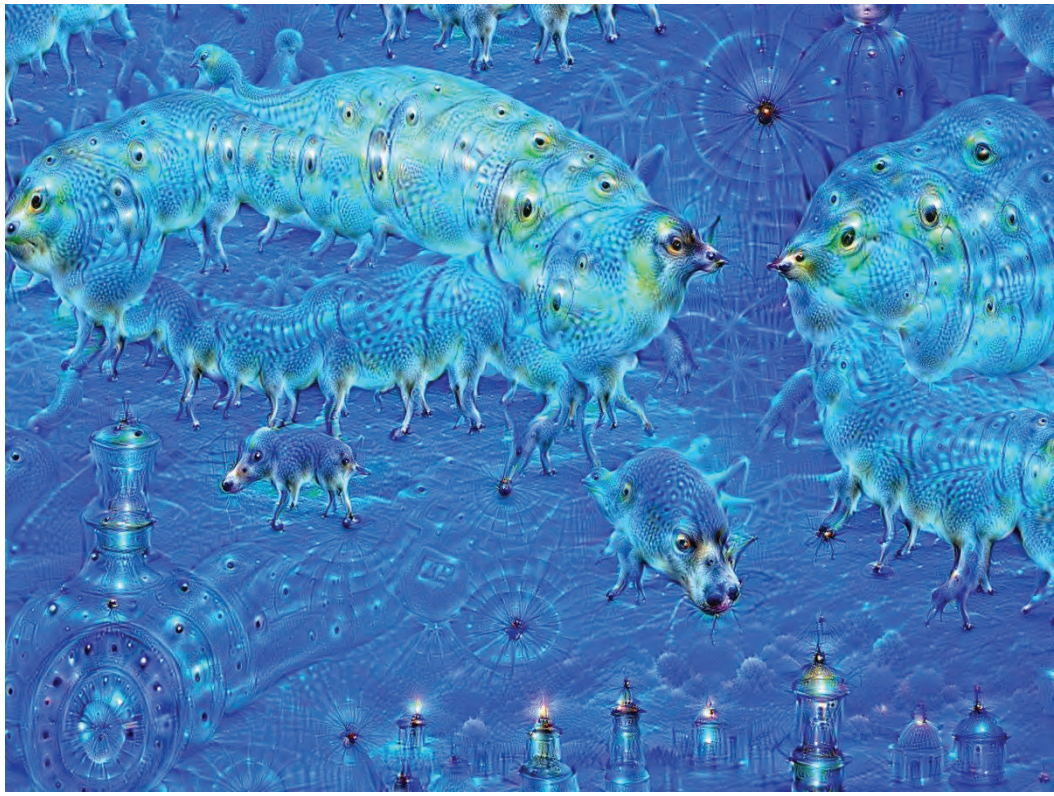
It is a “second generation” system for machine learning, based on deep learning neural networks.

RackBrain now handles a large number of Google searches, and is powered by TensorFlow.

TensorFlow calculations are generally expressed as stateful dataflow graphs. Nodes in the graph represent mathematical operations, while edges are multidimensional arrays ("tensors") communicated between them. The name, TensorFlow, derives from the operations which neural networks perform on the arrays themselves.

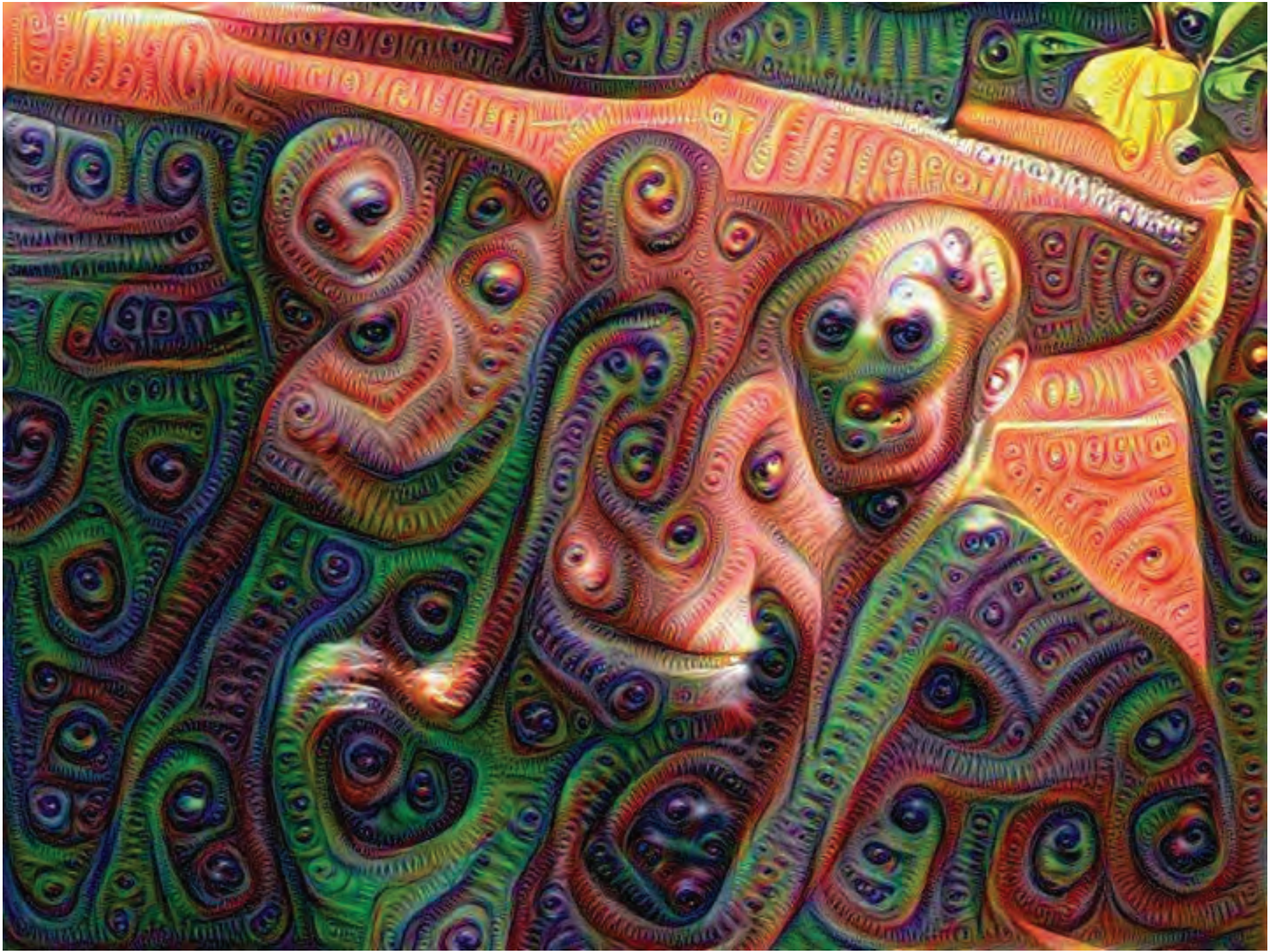
DeepDream - Convolutional Neural Network

Original Image



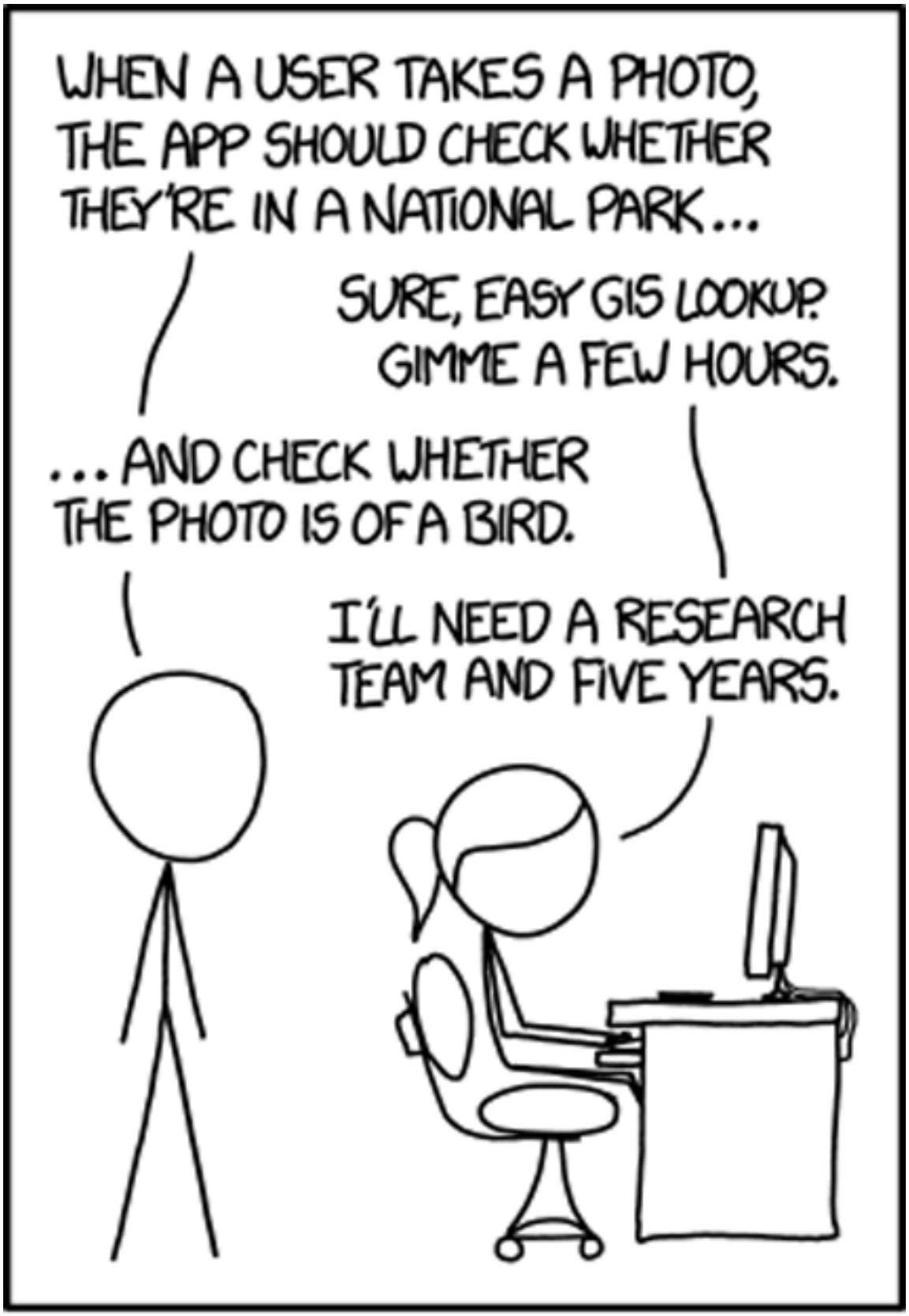
After 10 Iterations of DeepDream

Three Men in a Pool (DeepDream)



In *Nature*, 27 January 2016

- "DeepMind's program AlphaGo beat Fan Hui, the European Go champion, five times out of five in tournament conditions..."
- "AlphaGo was not preprogrammed to play Go: rather, it learned using a general-purpose algorithm that allowed it to interpret the game's patterns."
- "...AlphaGo program applied **deep learning** in neural networks (convolutional NN) — brain-inspired programs in which connections between layers of simulated neurons are strengthened through examples and experience."



IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.