Introduction to Deep Learning

Shangsong Liang
Sun Yat-sen University

DL is providing breakthrough results in speech recognition and image classification …

From this Hinton et al 2012 paper:

http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/38131.pdf

modeling technique	#params [10^6]	WER Hub5'00-SWB	WER RT03S-FSH
GMM, 40 mix DT 309h SI	29.4	23.6	27.4
NN 1 hidden-layer×4634 units	43.6	26.0	29.4
+ 2×5 neighboring frames	45.1	22.4	25.7
DBN-DNN 7 hidden layers×2048 units	45.1	17.1	19.6
+ updated state alignment	45.1	16.4	18.6
+ sparsification	15.2 nz	16.1	18.5
GMM 72 mix DT 2000h SA	102.4	17.1	18.6

task	hours of training data	DNN-HMM	GMM-HMM with same data	GMM-HMM with more data
Switchboard (test set 1)	309	18.5	27.4	18.6 (2000 hrs)
Switchboard (test set 2)	309	16.1	23.6	17.1 (2000 hrs)
English Broadcast News	50	17.5	18.8	
Bing Voice Search (Sentence error rates)	24	30.4	36.2	
Google Voice Input	5,870	12.3		16.0 (>>5,870hrs)
Youtube	1,400	47.6	52.3	

go here: http://yann.lecun.com/exdb/mnist/

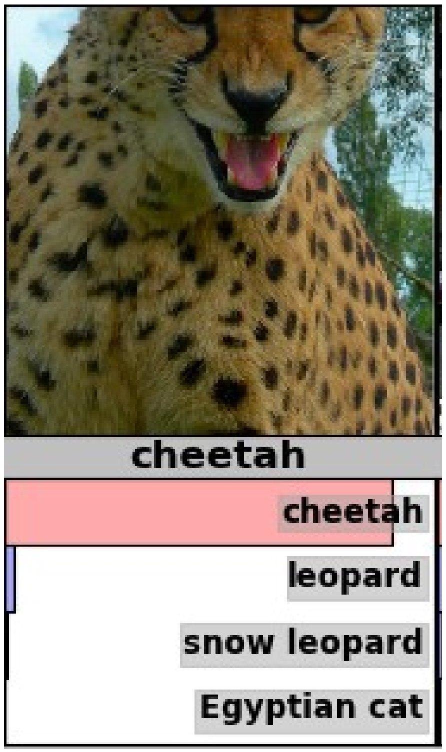
From here:
http://people.idsia.ch/~juergen/cvpr2012.pdf

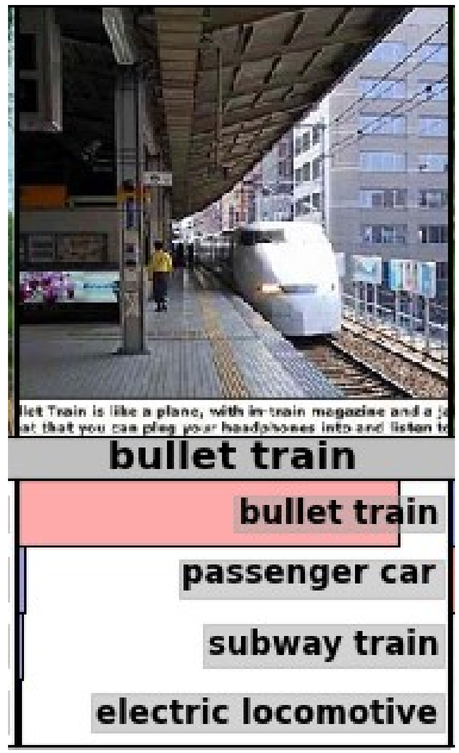
Dataset	Best result of others [%]	MCDNN [%]	Relative improv. [%]
MNIST	0.39	0.23	41
NIST SD 19	see Table 4	see Table 4	30-80
HWDB1.0 on.	7.61	5.61	26
HWDB1.0 off.	10.01	6.5	35
CIFAR10	18.50	11.21	39
traffic signs	1.69	0.54	72
NORB	5.00	2.70	46

Deep Learning is providing breakthrough results in speech recognition, image classification, etc.

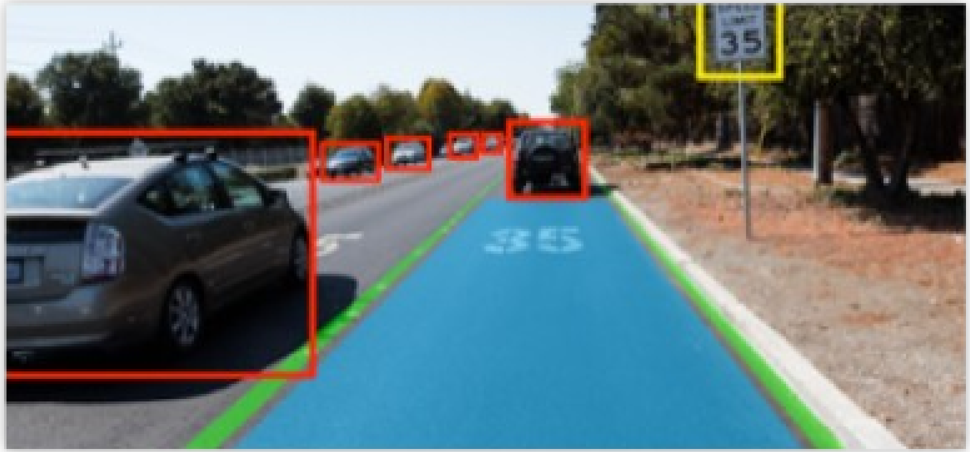
google inception network

Examples from the test set (with the network's guesses)

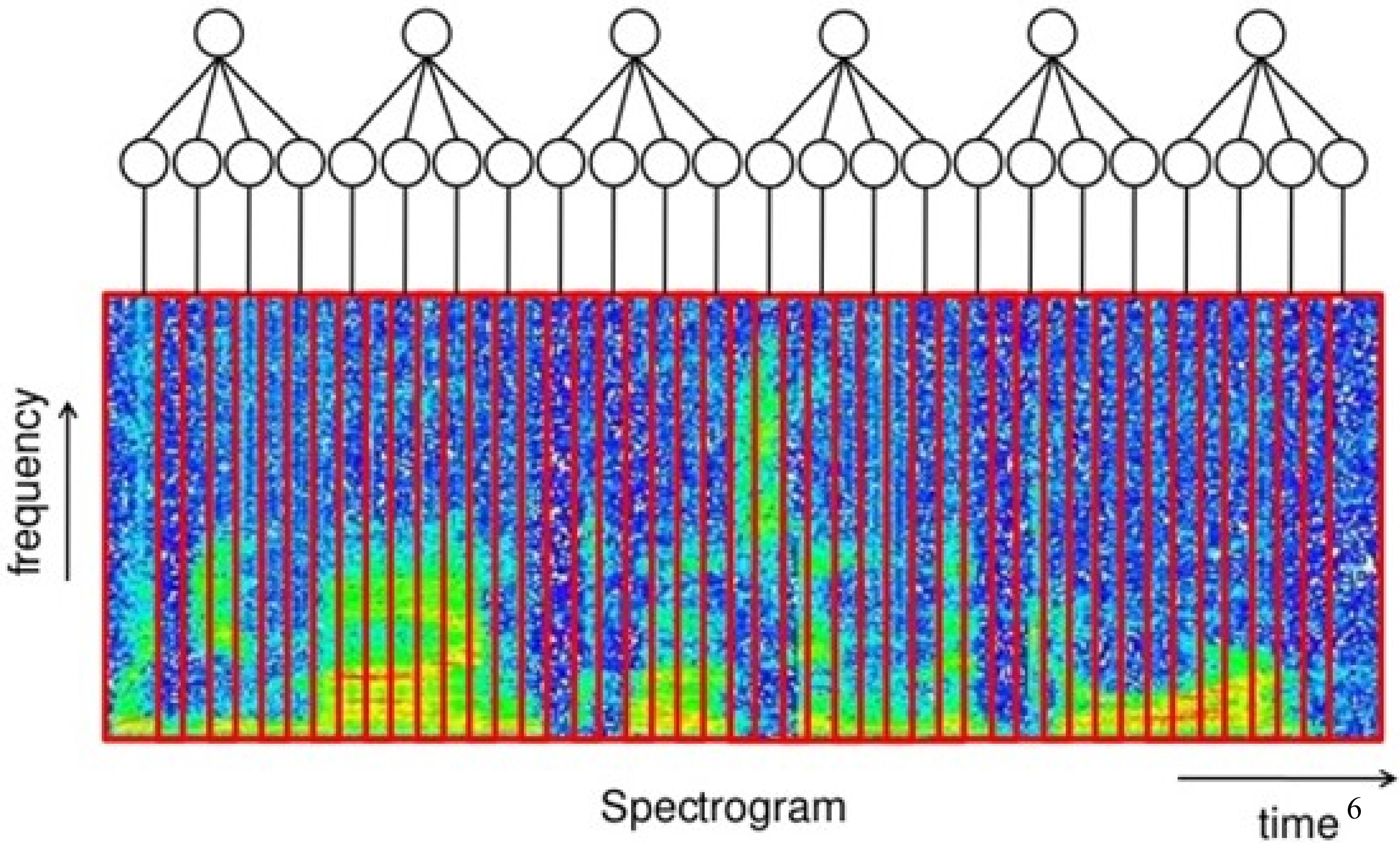




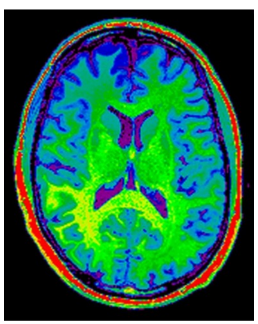
Video analyses and decision making

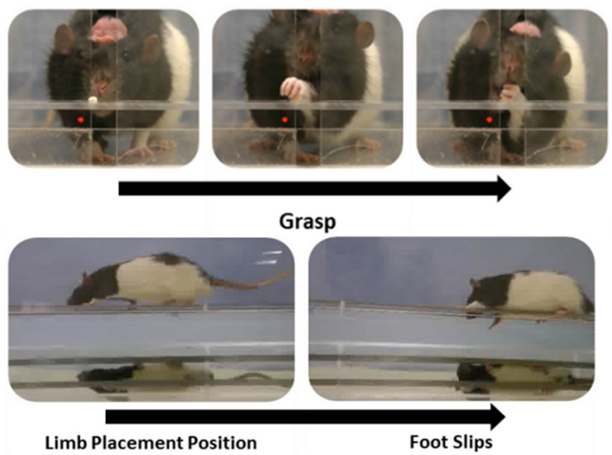


Speech recognition

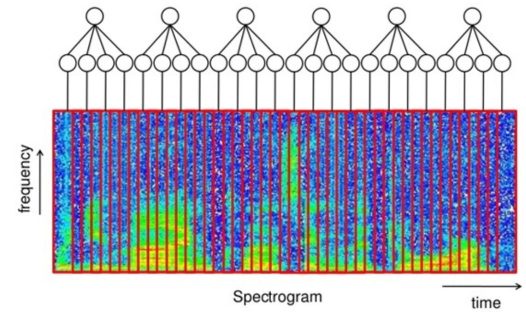


Neuroscience data is similar to other types of data



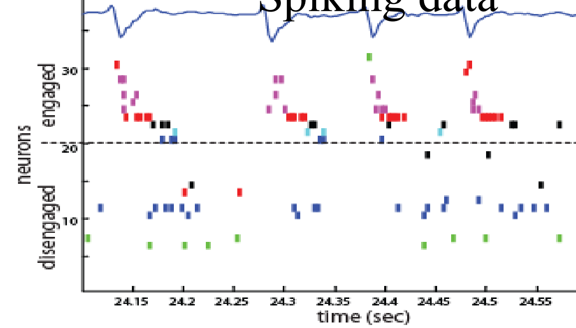


Ryait et al. & Luczak

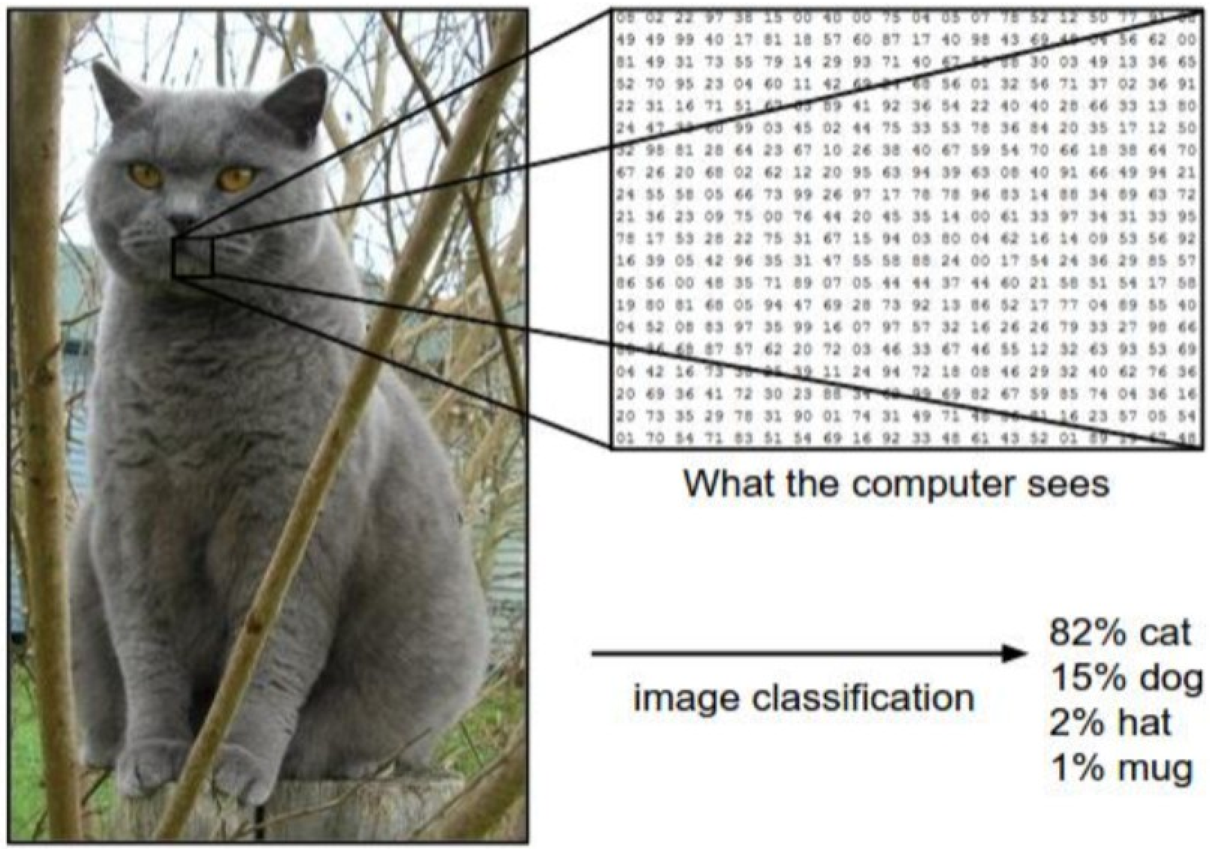


EEG / LFP

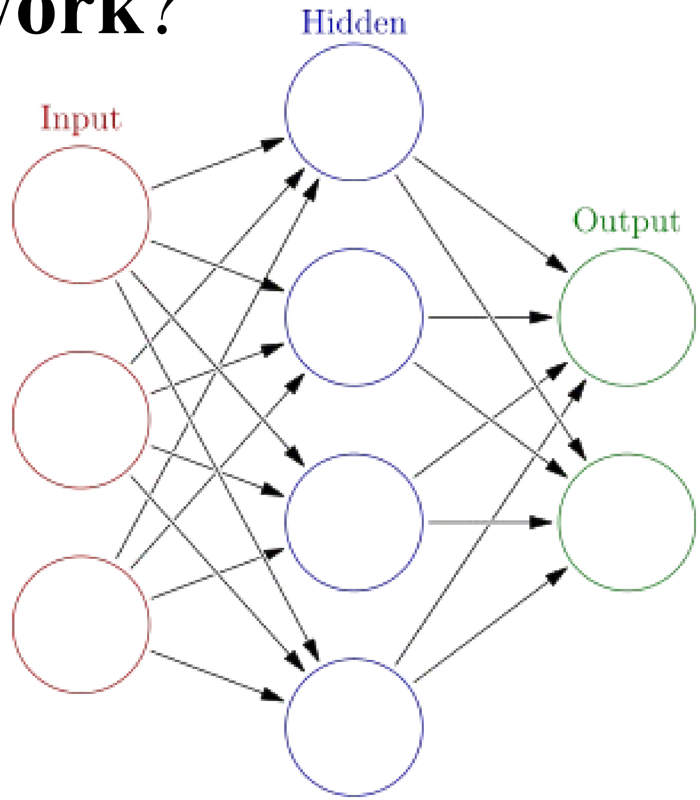
Spiking data



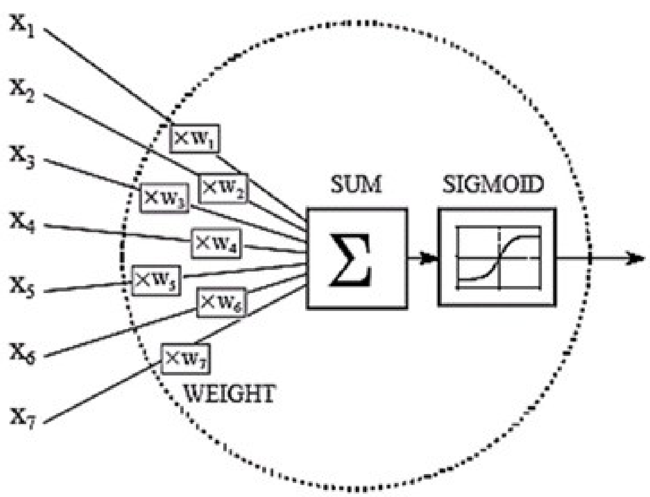
Images are Numbers

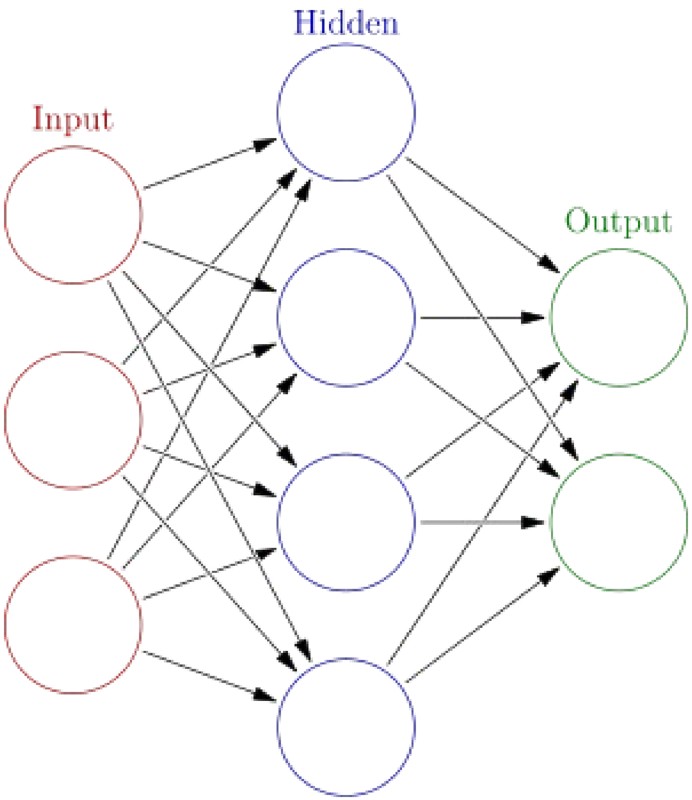


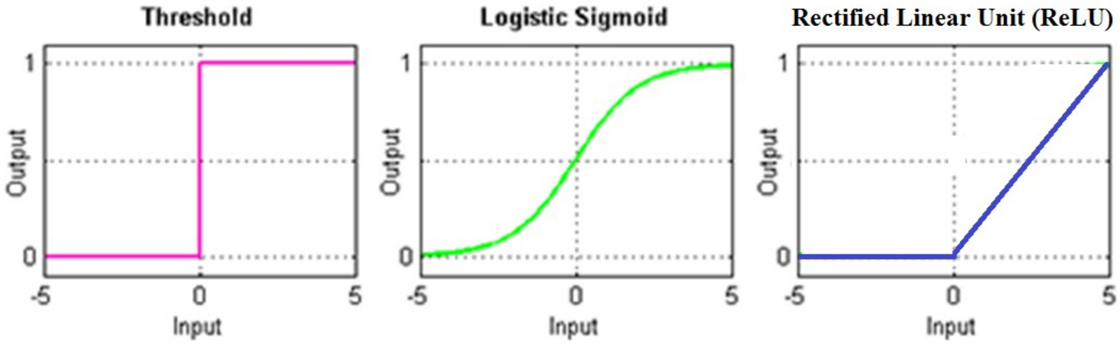
What is Artificial Neuronal Network?



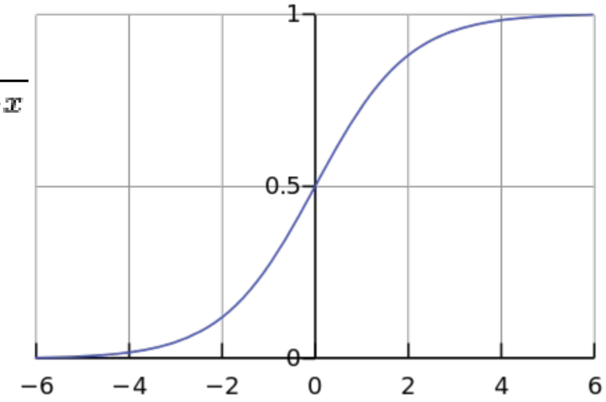
Artificial neural

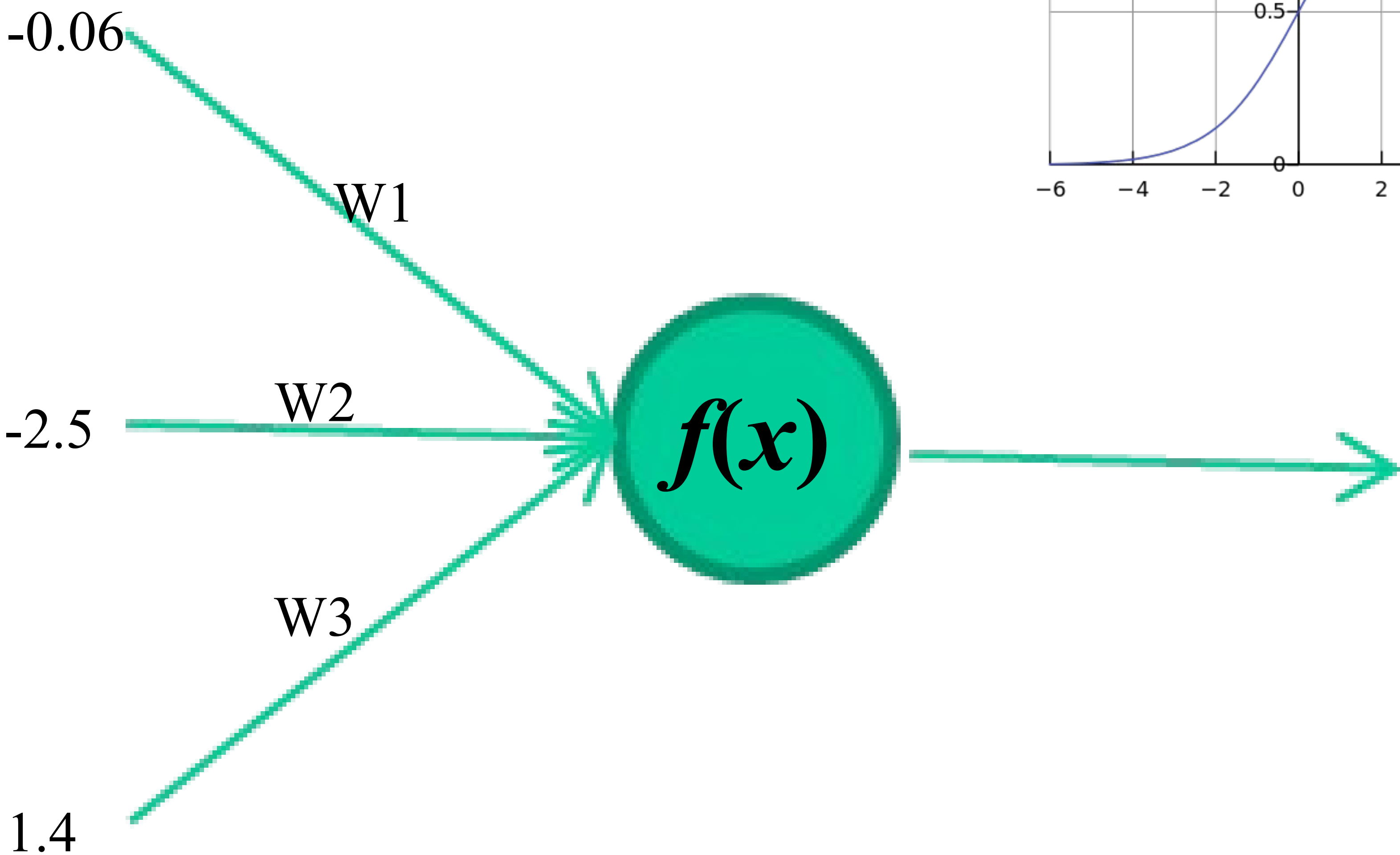




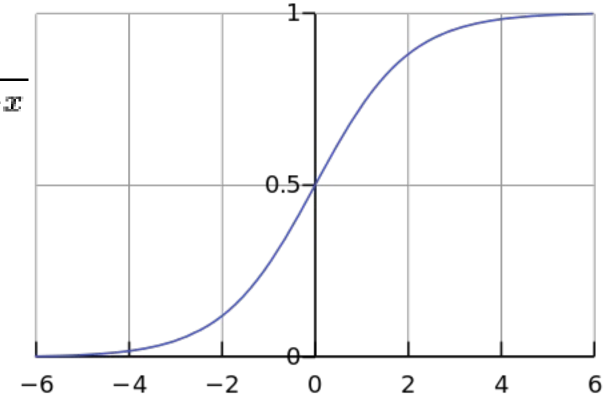


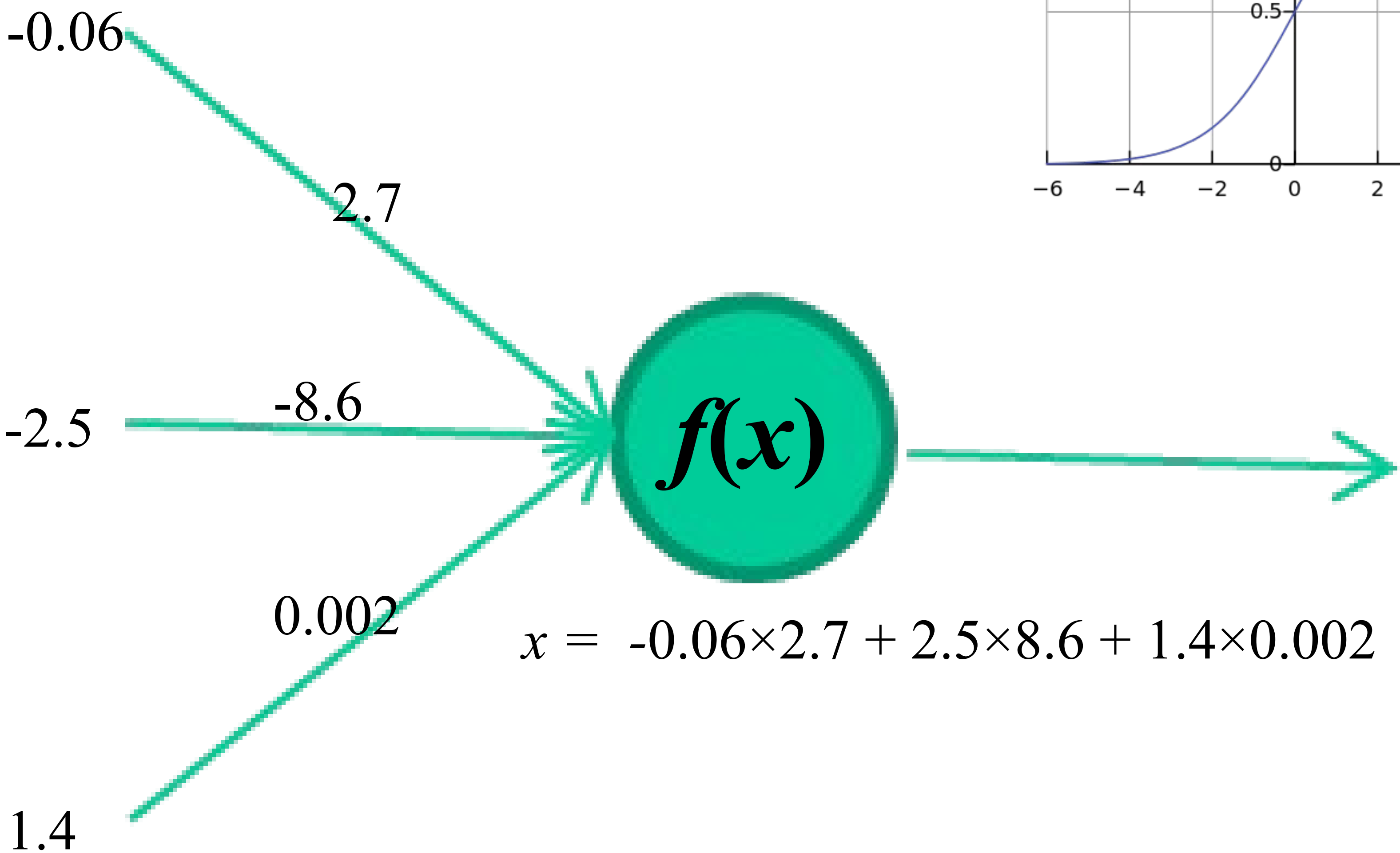
$$f(x) = \frac{1}{1+e^{-x}}$$





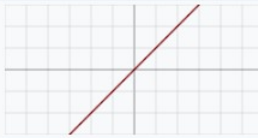
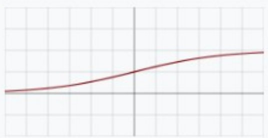
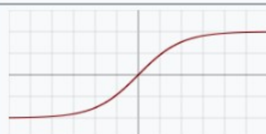
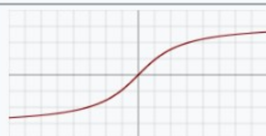
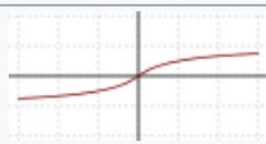
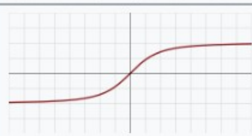
$$f(x) = \frac{1}{1+e^{-x}}$$



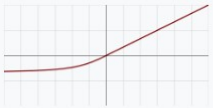
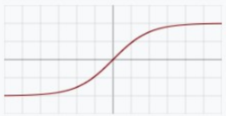
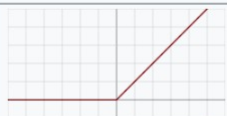
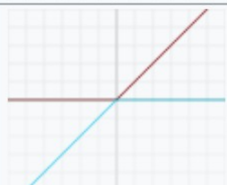
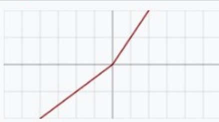
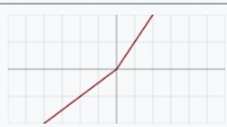
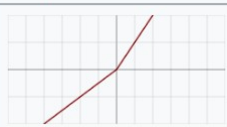


$$x = -0.06\times2.7 + 2.5\times8.6 + 1.4\times0.002 = 21.34$$

Activation function

Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ [1]	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
ArSinH		$f(x) = \sinh^{-1}(x) = \ln\left(x + \sqrt{x^2 + 1}\right)$	$f'(x) = \frac{1}{\sqrt{x^2 + 1}}$
ElliotSig[9][10][11] Softsign[12][13]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$
Inverse square root unit (ISRU)[14]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1 + \alpha x^2}}\right)^3$

Activation function

Inverse square root linear unit (ISRLU)[14]		$f(x)=\begin{cases}\frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}\left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Square Nonlinearity (SQNL)[11]		$f(x)=\begin{cases}1 & : x>2.0\\ x-\frac{x^2}{4} & : 0\leq x\leq 2.0\\ x+\frac{x^2}{4} & : -2.0\leq x<0\\ -1 & : x<-2.0\end{cases}$	$f'(x)=1\mp\frac{x}{2}$
Rectified linear unit (ReLU)[15]		$f(x)=\begin{cases}0 & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Bipolar rectified linear unit (BReLU)[16]		$f(x_i)=\begin{cases}ReLU(x_i) & \text{if } i \bmod 2=0\\ -ReLU(-x_i) & \text{if } i \bmod 2\neq 0\end{cases}$	$f'(x_i)=\begin{cases}ReLU'(x_i) & \text{if } i \bmod 2=0\\ -ReLU'(-x_i) & \text{if } i \bmod 2\neq 0\end{cases}$
Leaky rectified linear unit (Leaky ReLU)[17]		$f(x)=\begin{cases}0.01x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0.01 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Parametric rectified linear unit (PReLU)[18]		$f(\alpha,x)=\begin{cases}\alpha x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(\alpha,x)=\begin{cases}\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Randomized leaky rectified linear unit (RReLU)[19]		$f(\alpha,x)=\begin{cases}\alpha x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$ [3]	$f'(\alpha,x)=\begin{cases}\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$

So, 1. **what exactly is deep learning** ?

And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

So, 1. **what exactly is deep learning** ?

And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

The short answers

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

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

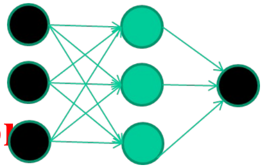
hmmm… OK, but:

3. **multilayer neural networks have been around for 25 years. What's actually new?**

hmmm… OK, but:

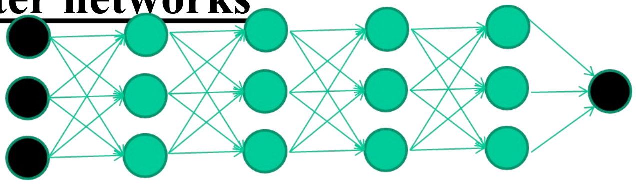
3. **multilayer neural networks have been around for 25 years. What's actually new?**

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: **algorithms for training many-later networks**



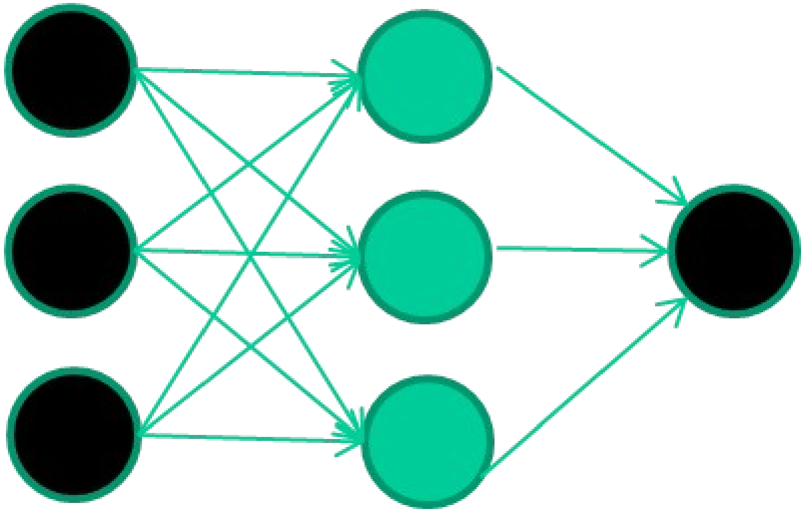
longer answers

1. reminder/quick-explanation of how neural network weights are learned;
2. the idea of **unsupervised feature learning** (why ‘intermediate features’ are important for difficult classification tasks, and how NNs seem to naturally learn them)
3. The ‘breakthrough’ – the simple trick for training Deep neural networks

A dataset

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

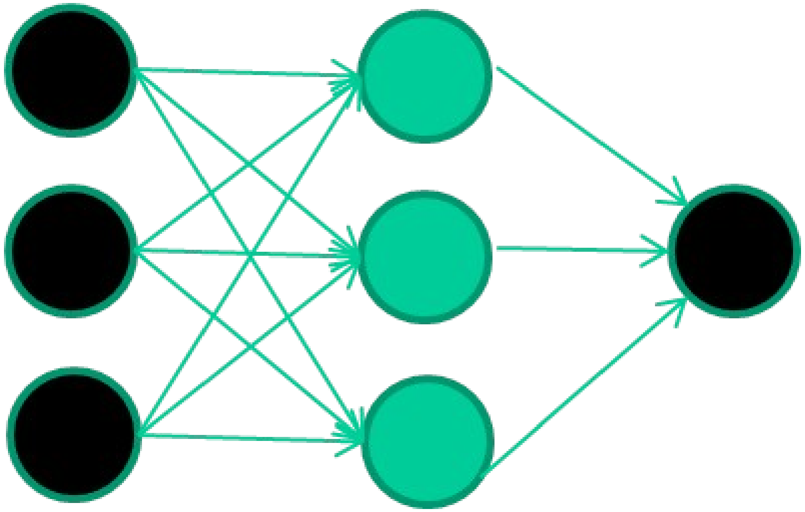
etc …



Training the neural network

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 …

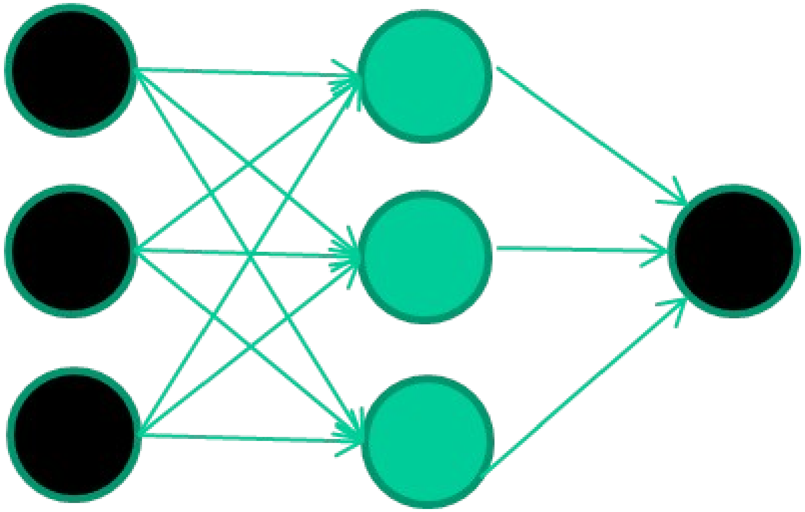


Training data

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 …

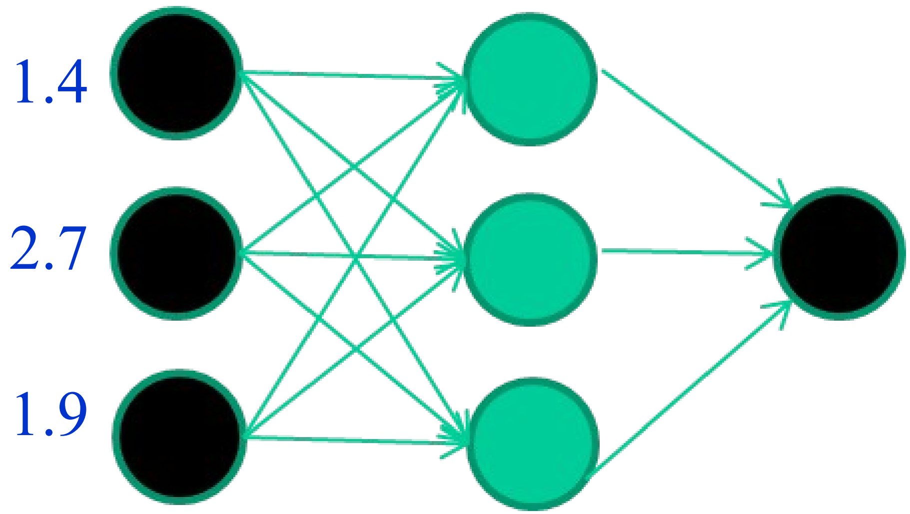
Initialise with random weights



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 …			

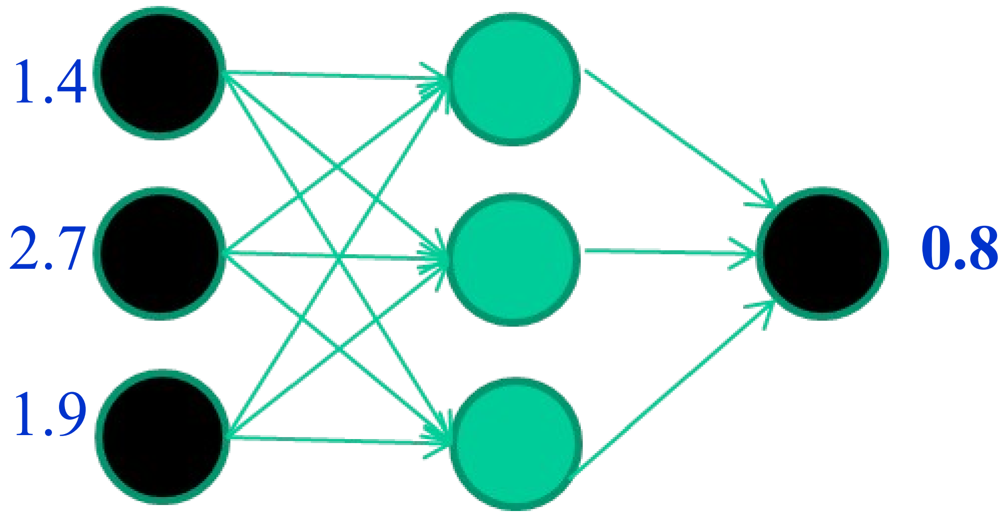
Present a training pattern



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 …			

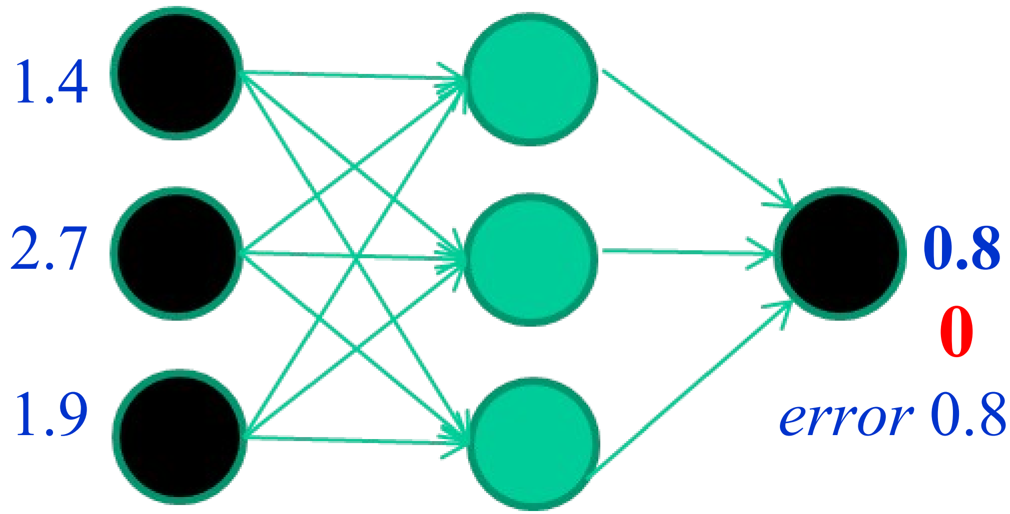
Feed it through to get output



Training data

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 …			

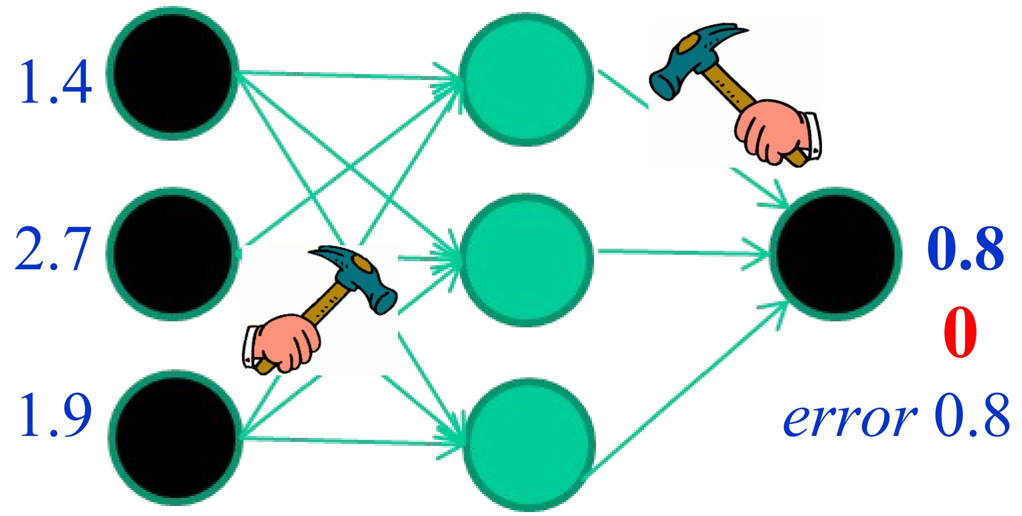
Compare with target output



Training data

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 …			

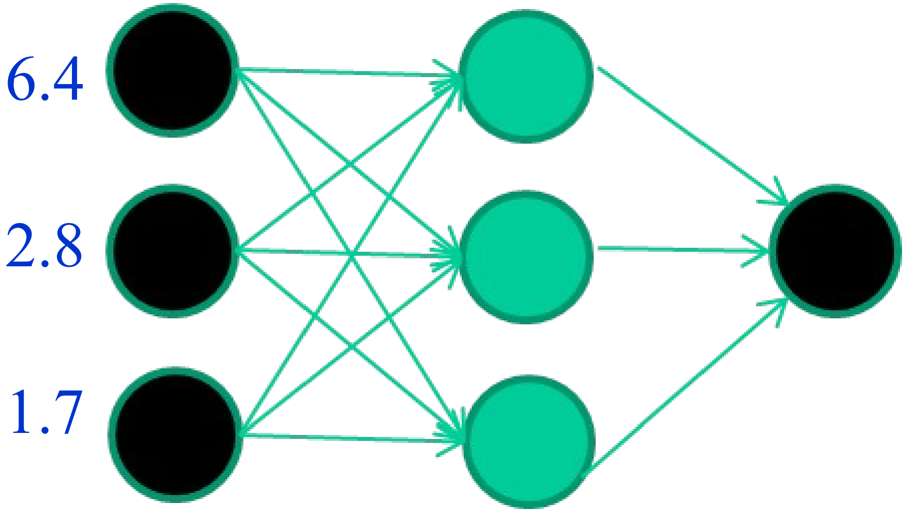
Adjust weights based on error



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 …			

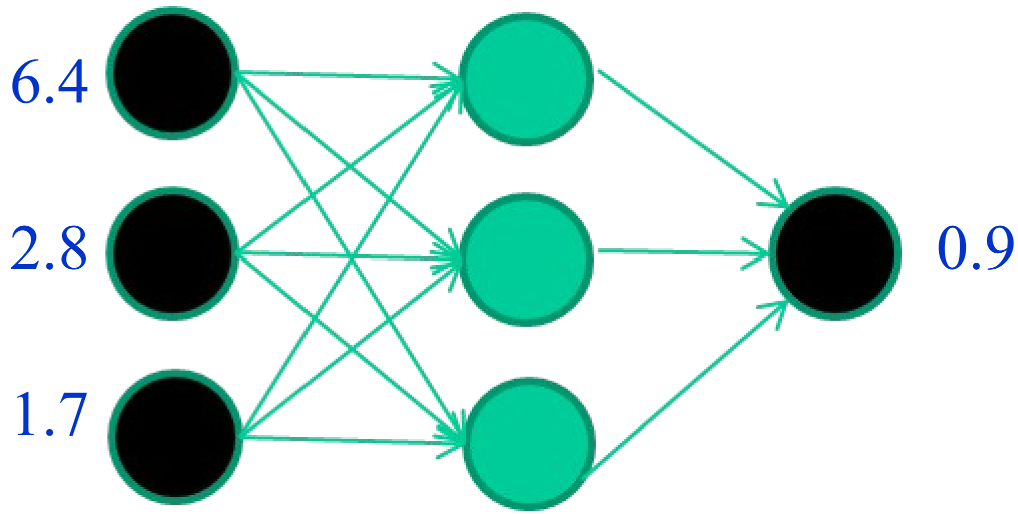
Present a training pattern



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 …			

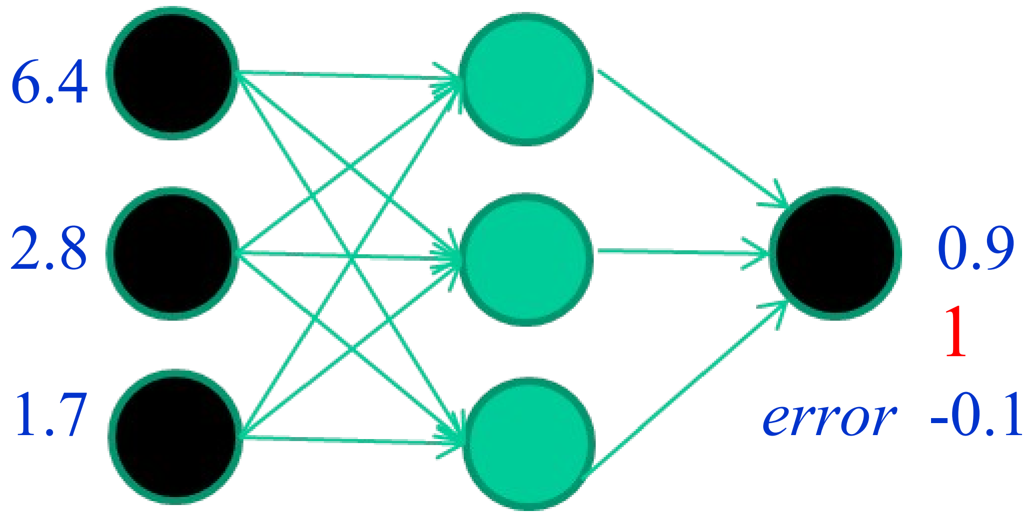
Feed it through to get output



Training data

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 …			

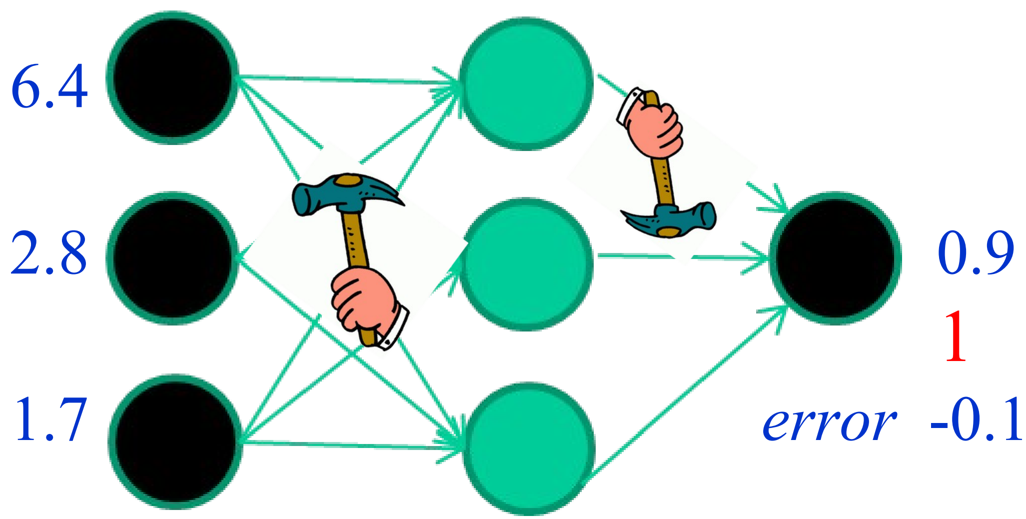
Compare with target output



Training data

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 …			

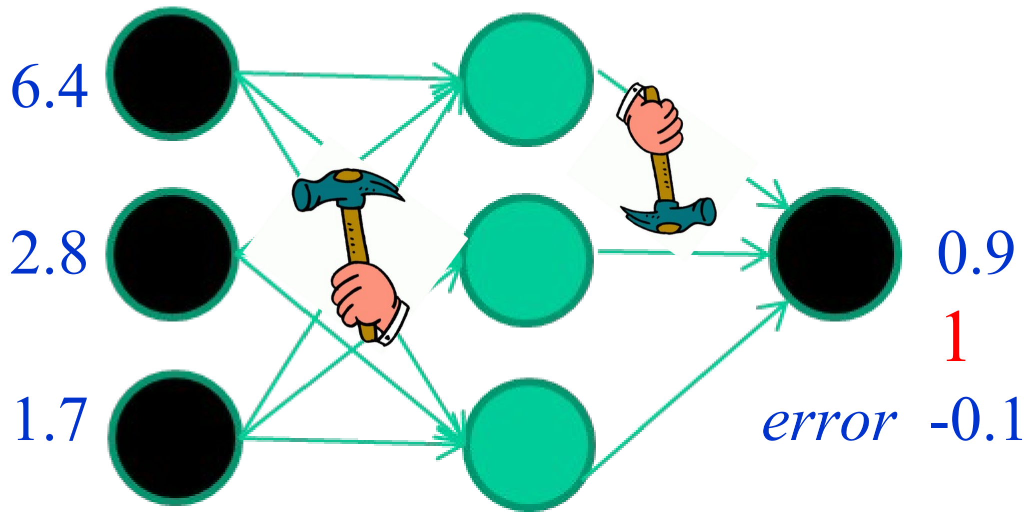
Adjust weights based on error



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 ….

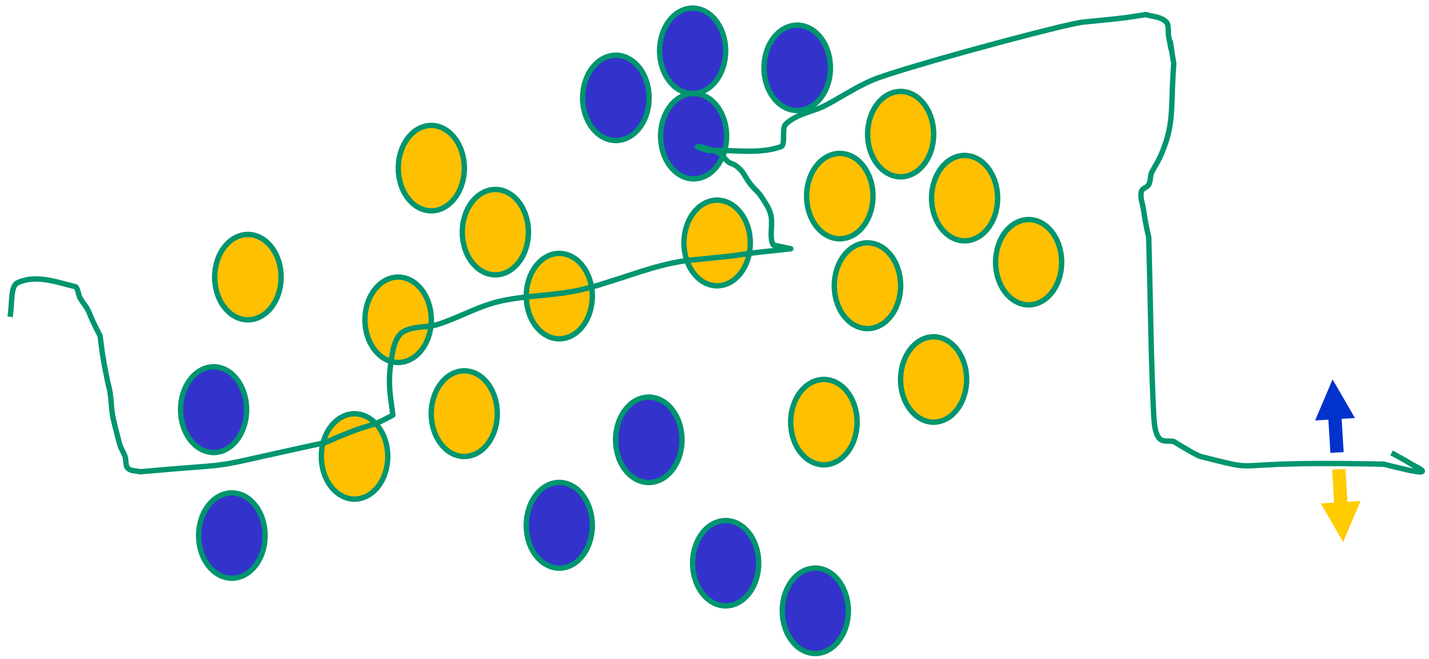


Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

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

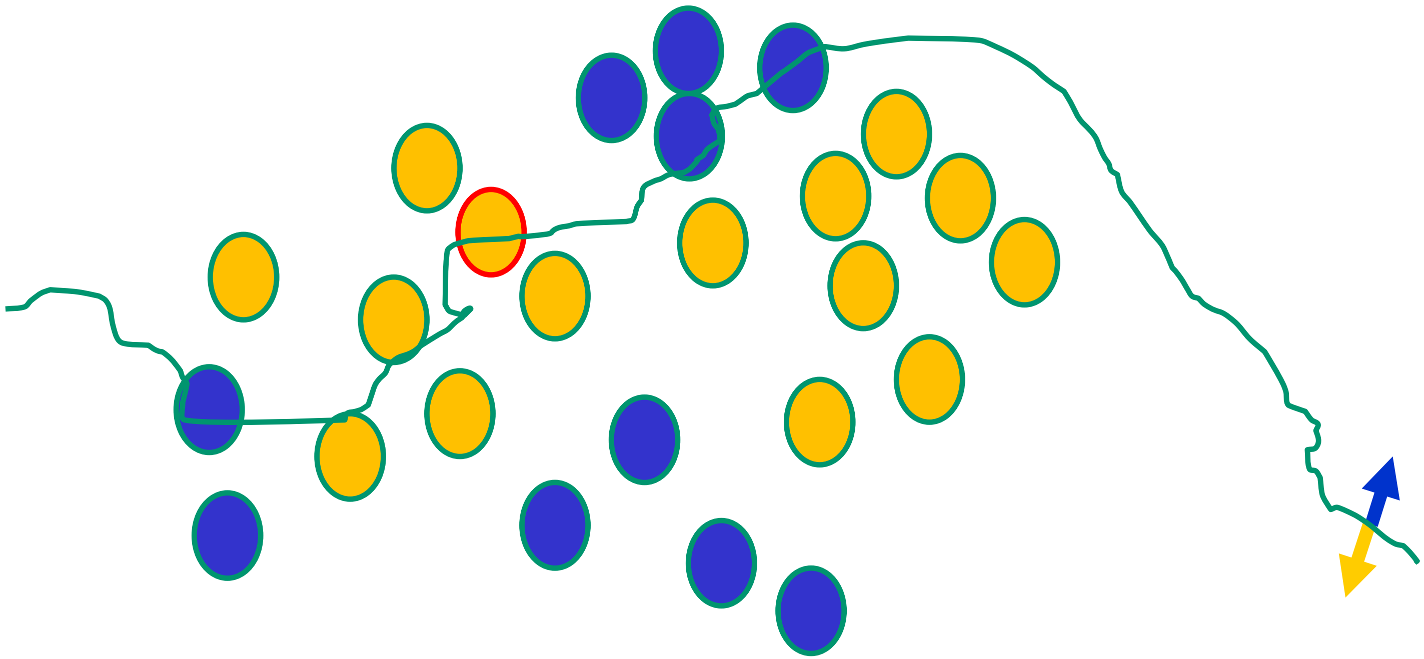
The decision boundary perspective…

Initial random weights



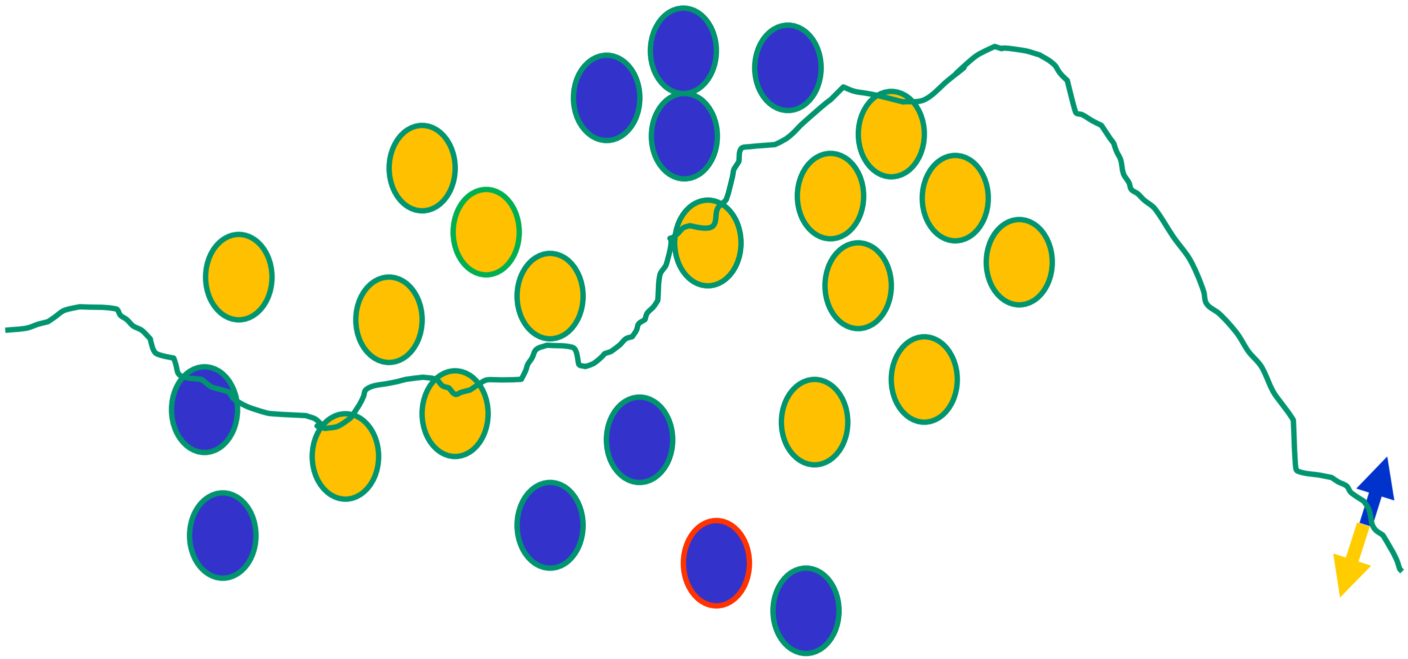
The decision boundary perspective…

Present a training instance / adjust the weights



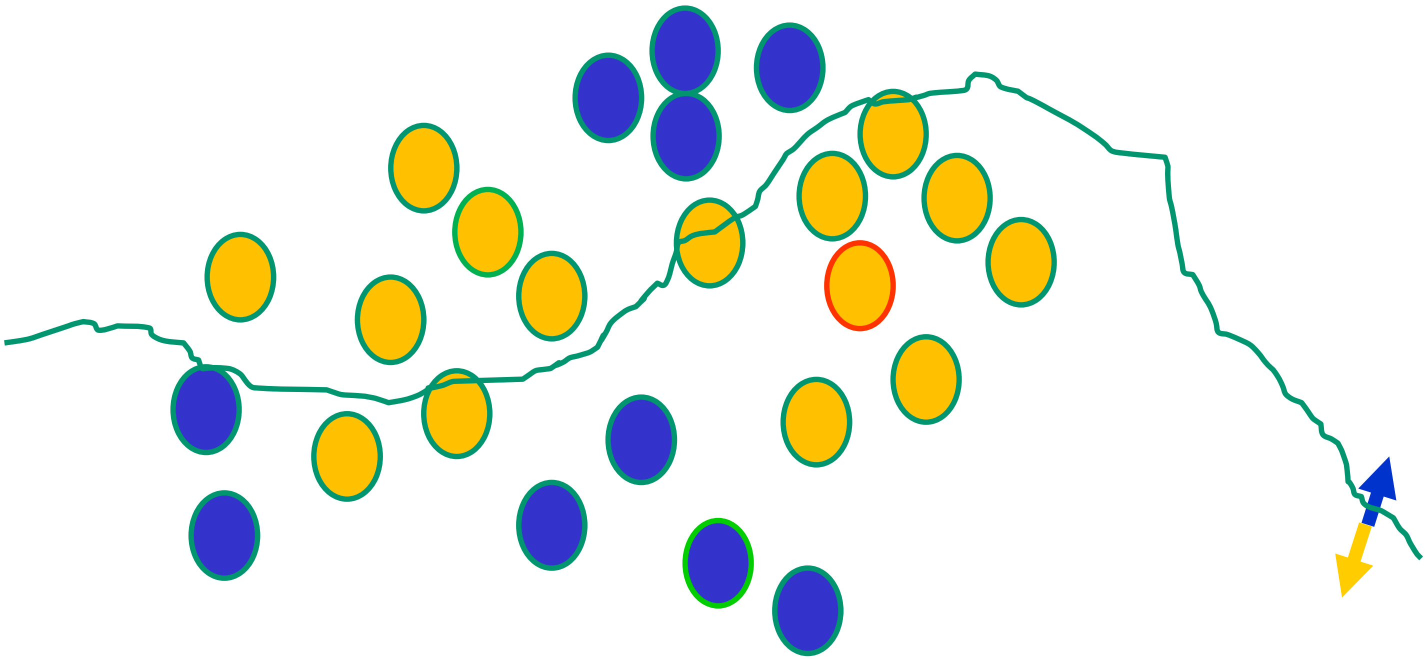
The decision boundary perspective…

Present a training instance / adjust the weights



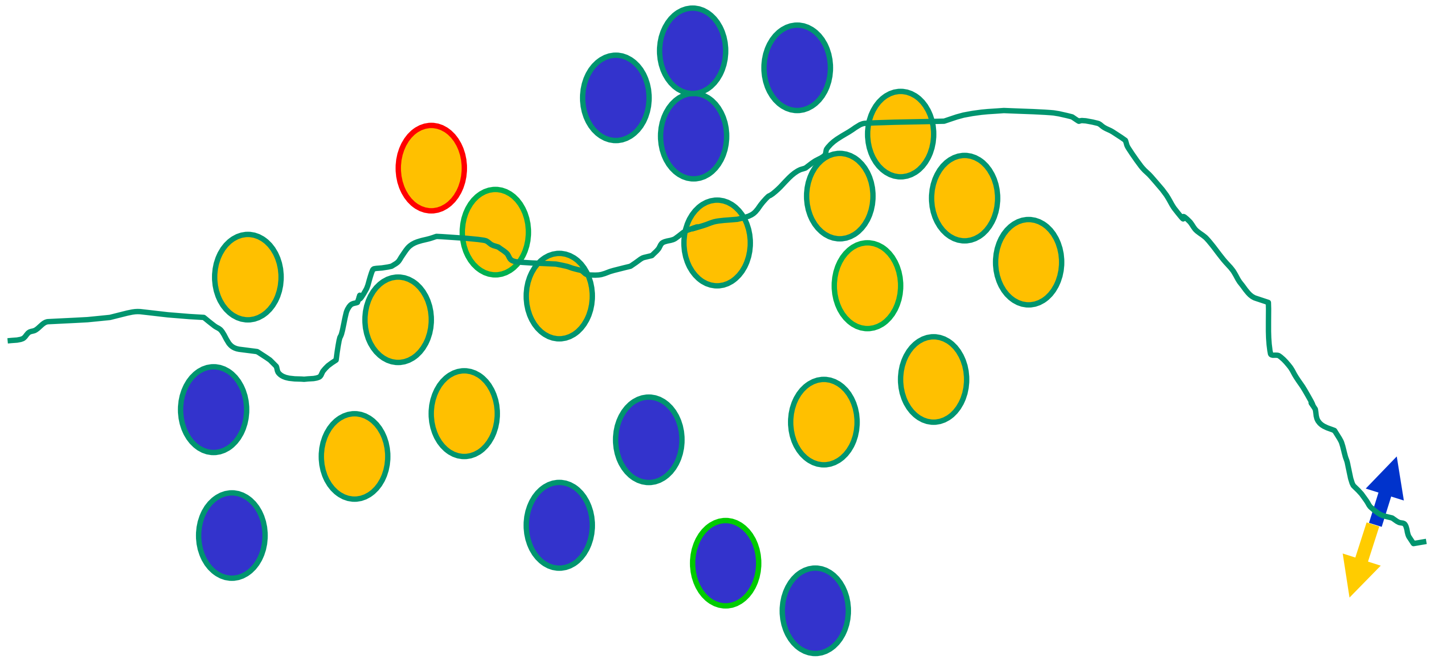
The decision boundary perspective…

Present a training instance / adjust the weights



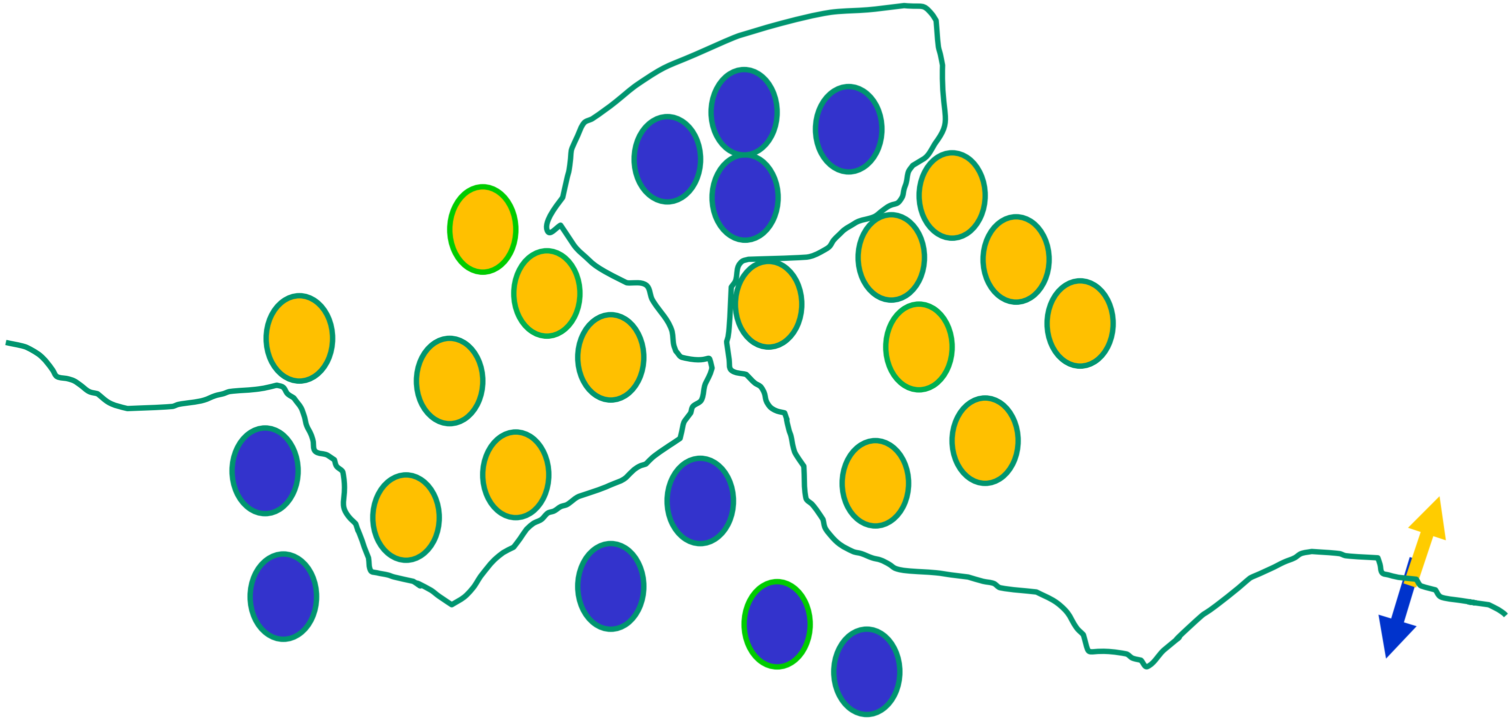
The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

Eventually ….

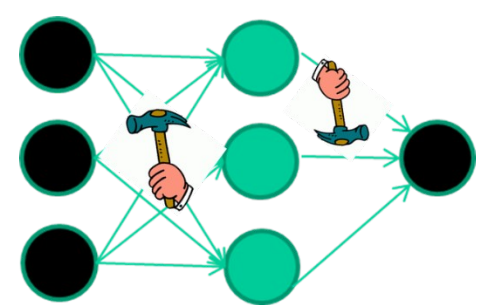


The point I am trying to make

- weight-learning algorithms for NNs are dumb

- 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 dumb luck, eventually this tends to be good enough to learn effective classifiers for many real applications



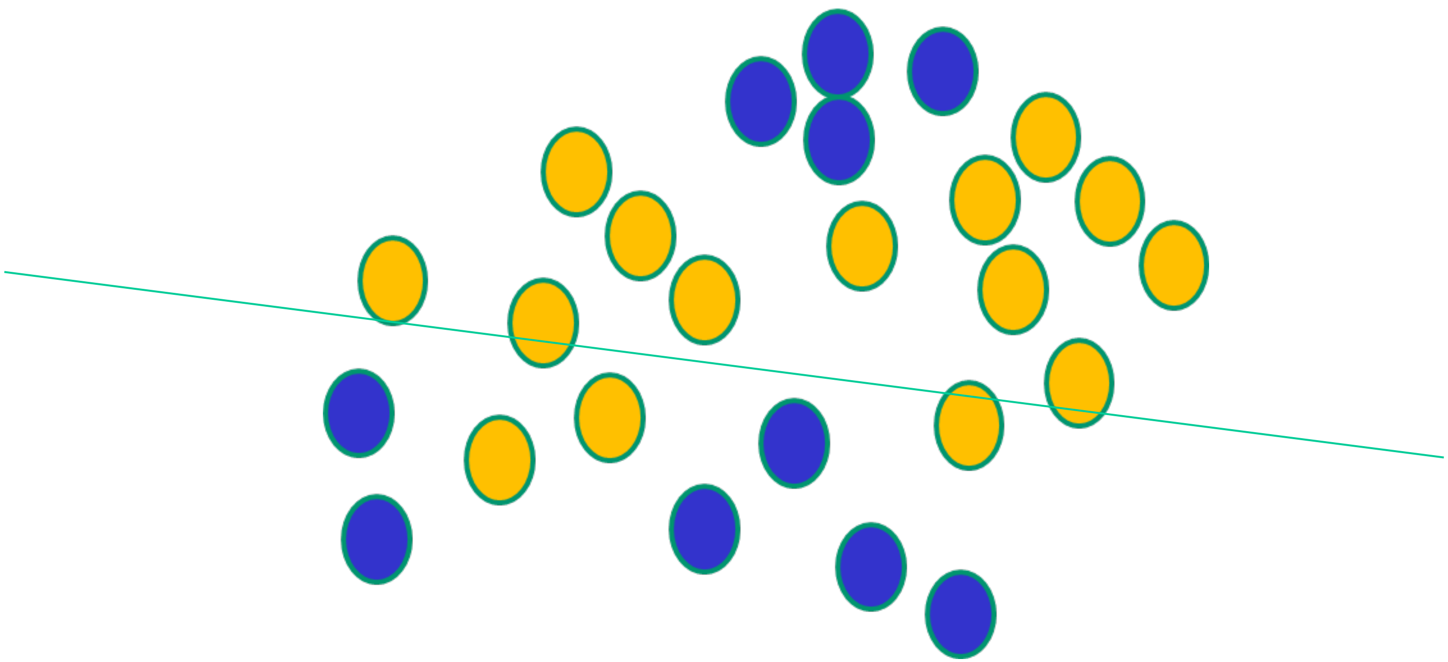
Some other points

Detail of a standard NN weight learning algorithm – **later**

If $f(x)$ is **non-linear**, a network with 1 hidden layer can, in theory, learn perfectly any classification problem. A set of weights exists that can produce the targets from the inputs. The problem is finding them.

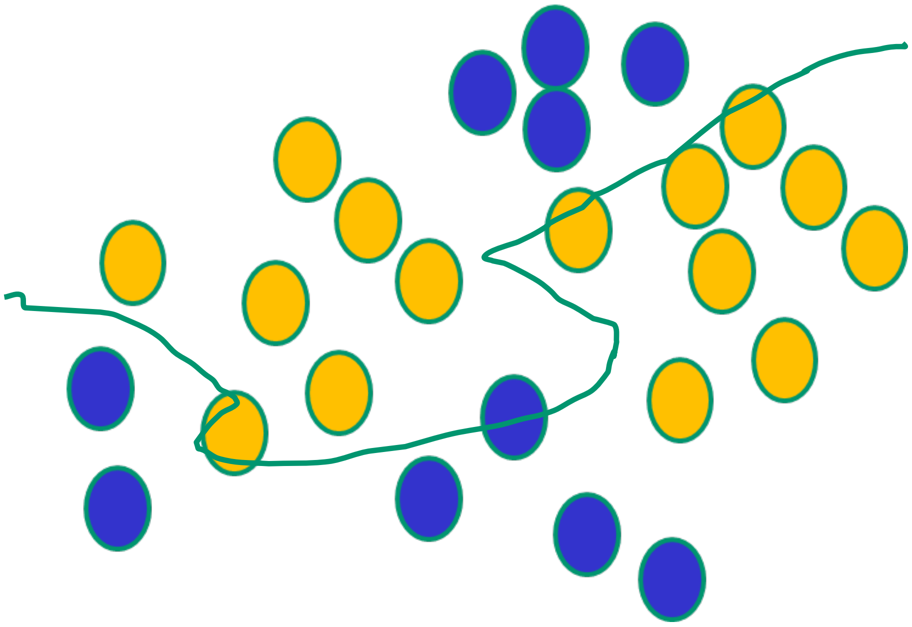
Some other ‘by the way’ points

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



Some other ‘by the way’ points

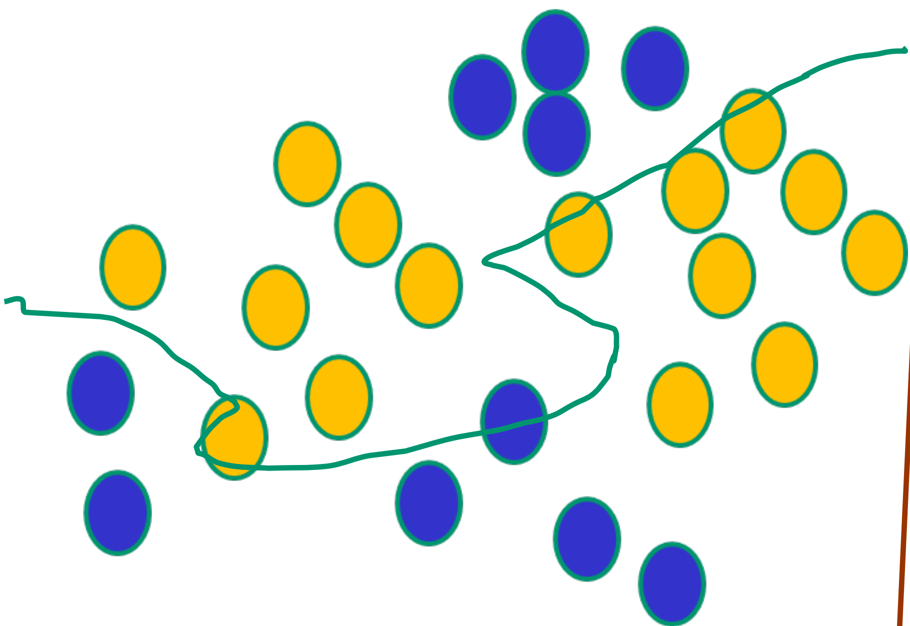
NNs use nonlinear f(x) so they can draw complex boundaries, but keep the data unchanged

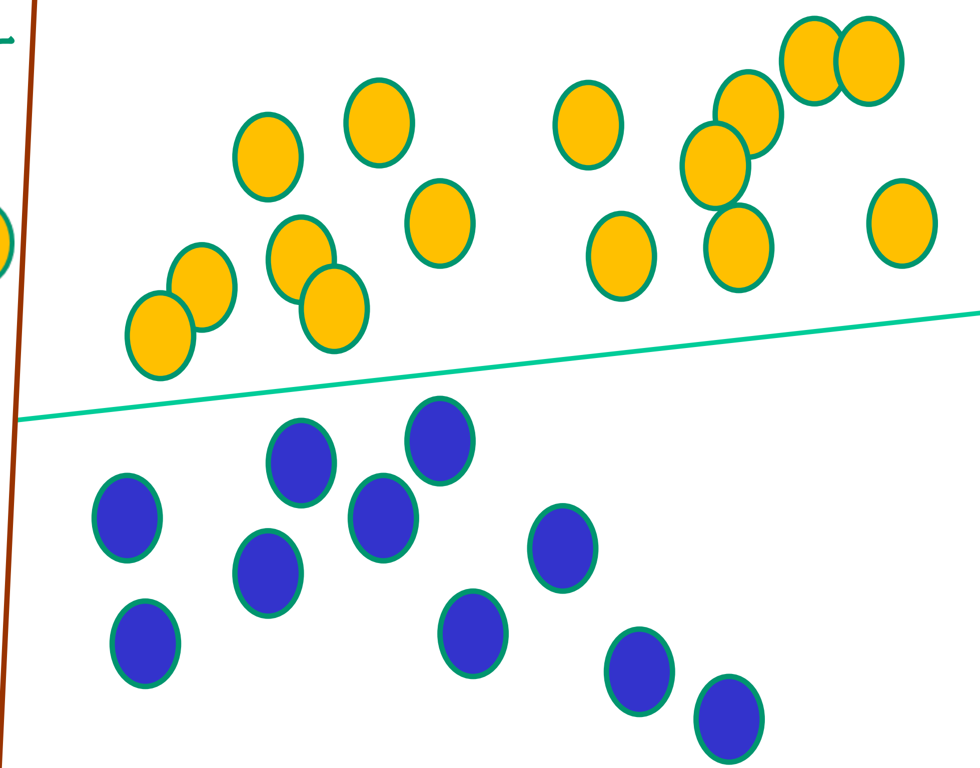


Some other 'by the way' points

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





Feature detectors

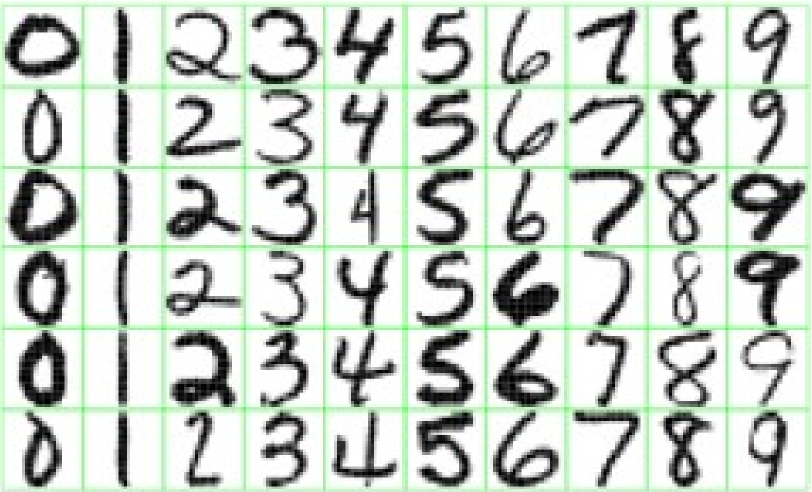
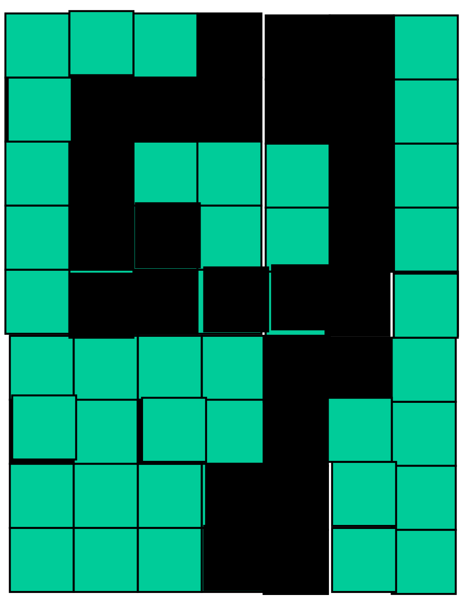
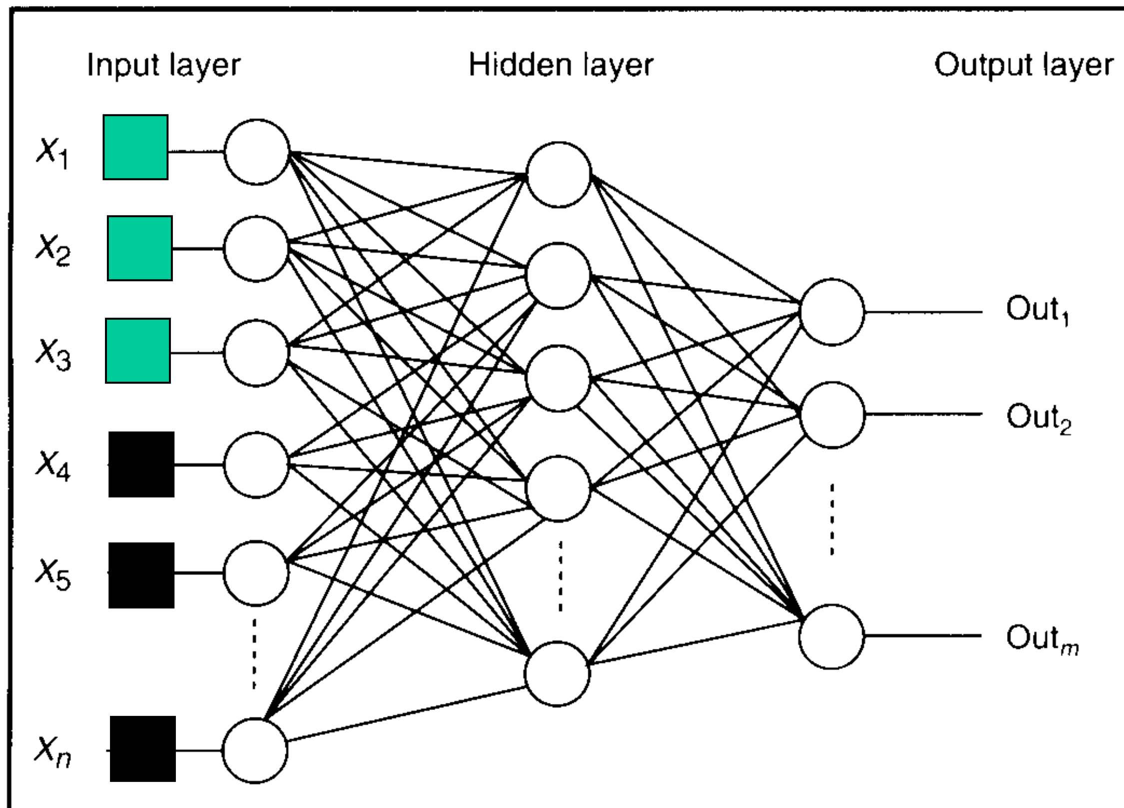
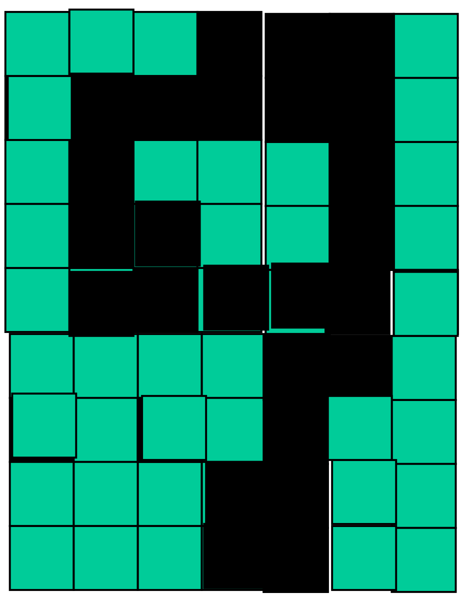


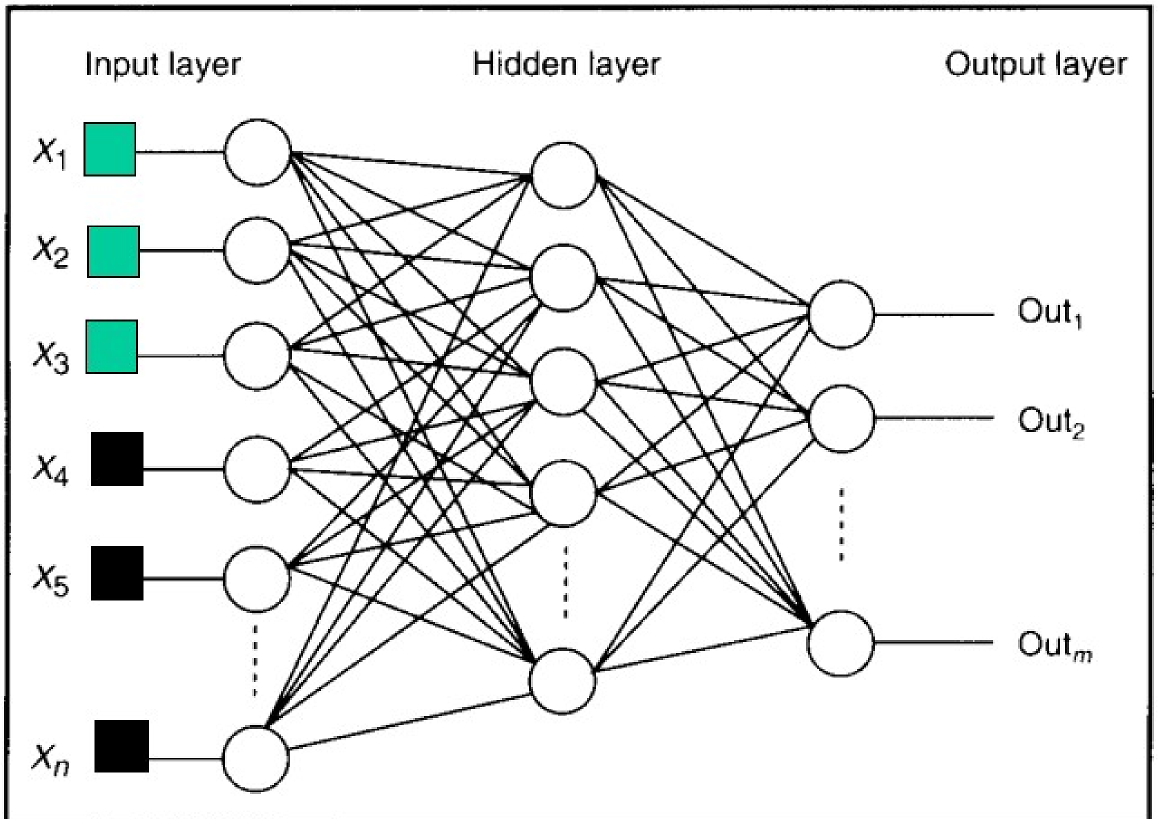
Figure 1.2: *Examples of handwritten digits postal envelopes.*



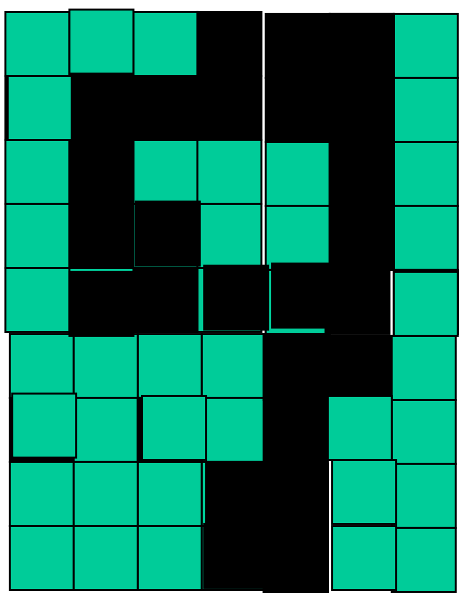


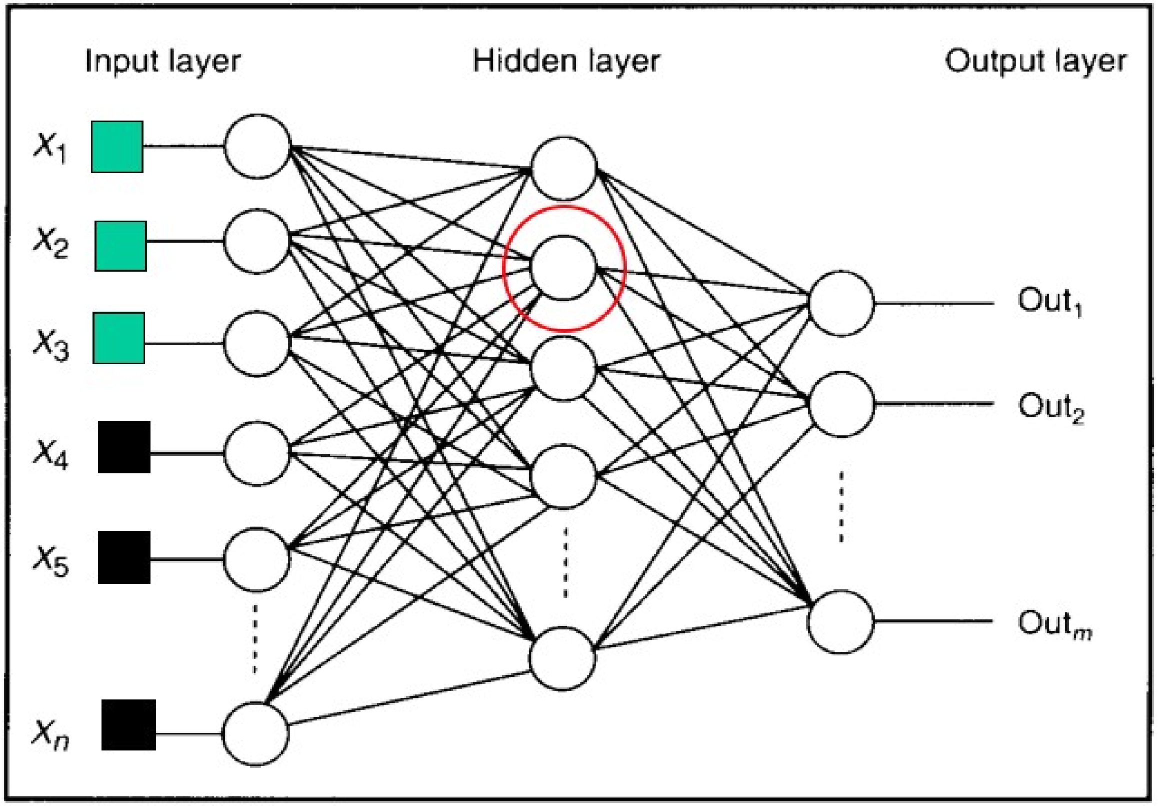
Feature detectors



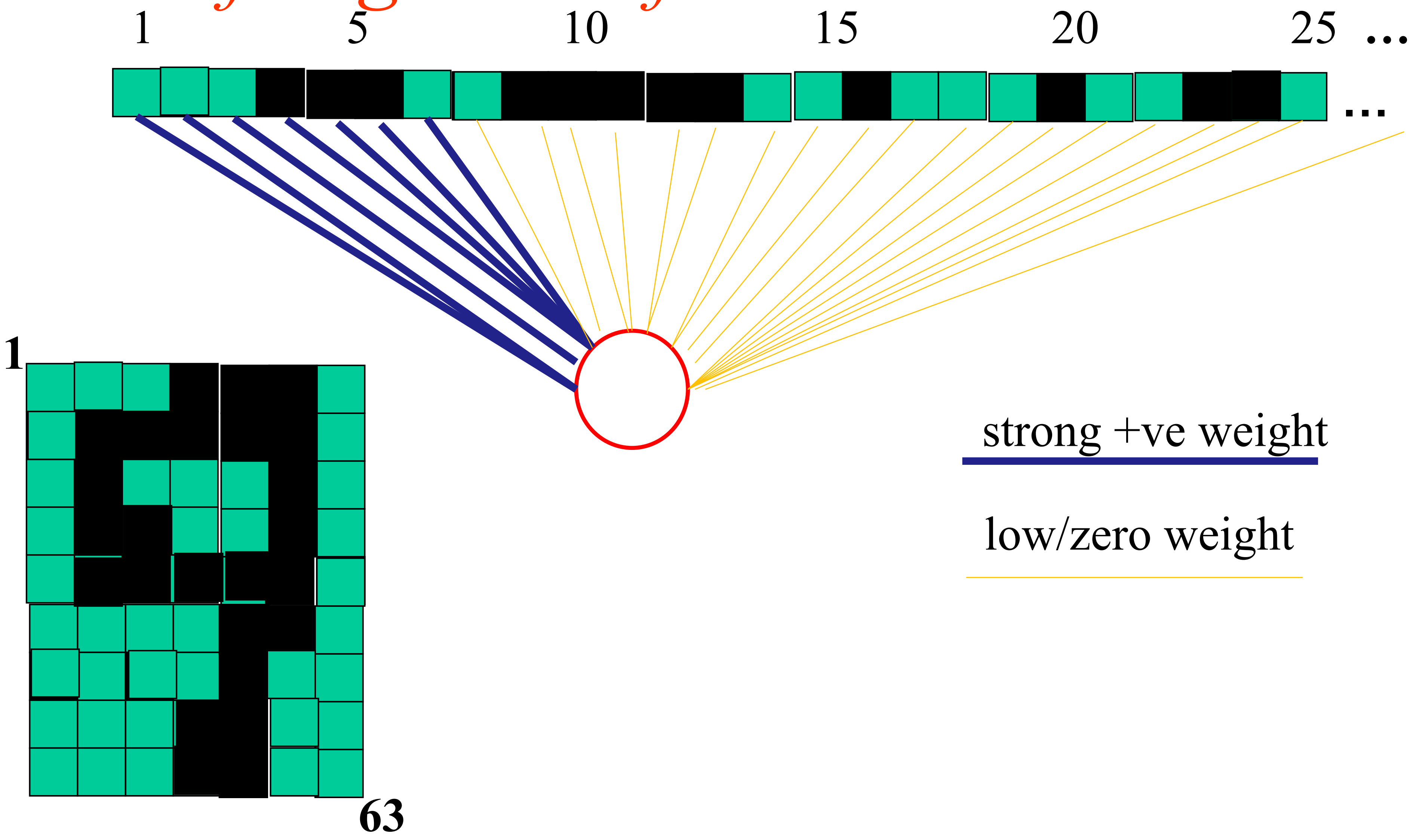


what is this unit doing?

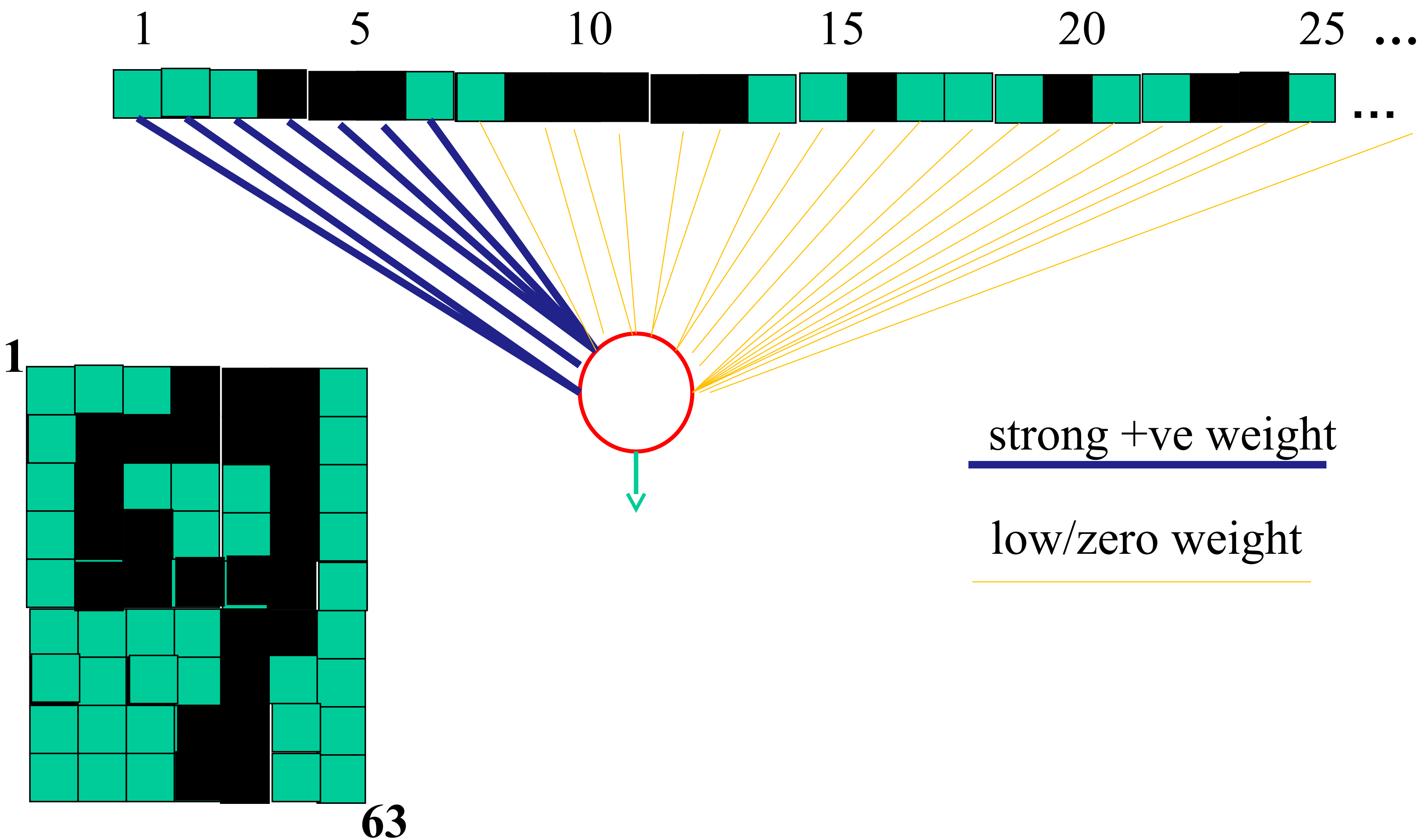




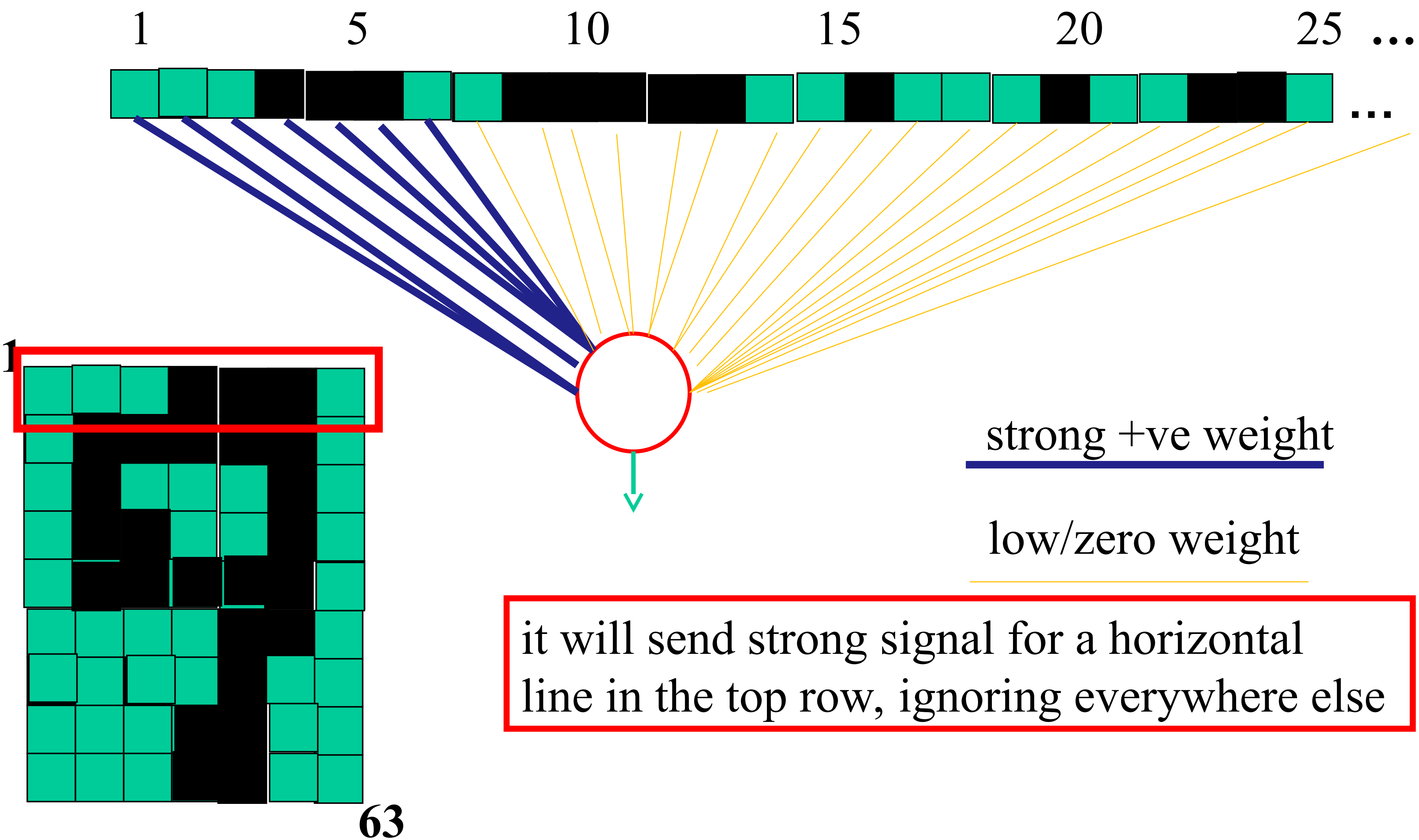
Hidden layer units become *self-organised feature detectors*



What does this unit detect?

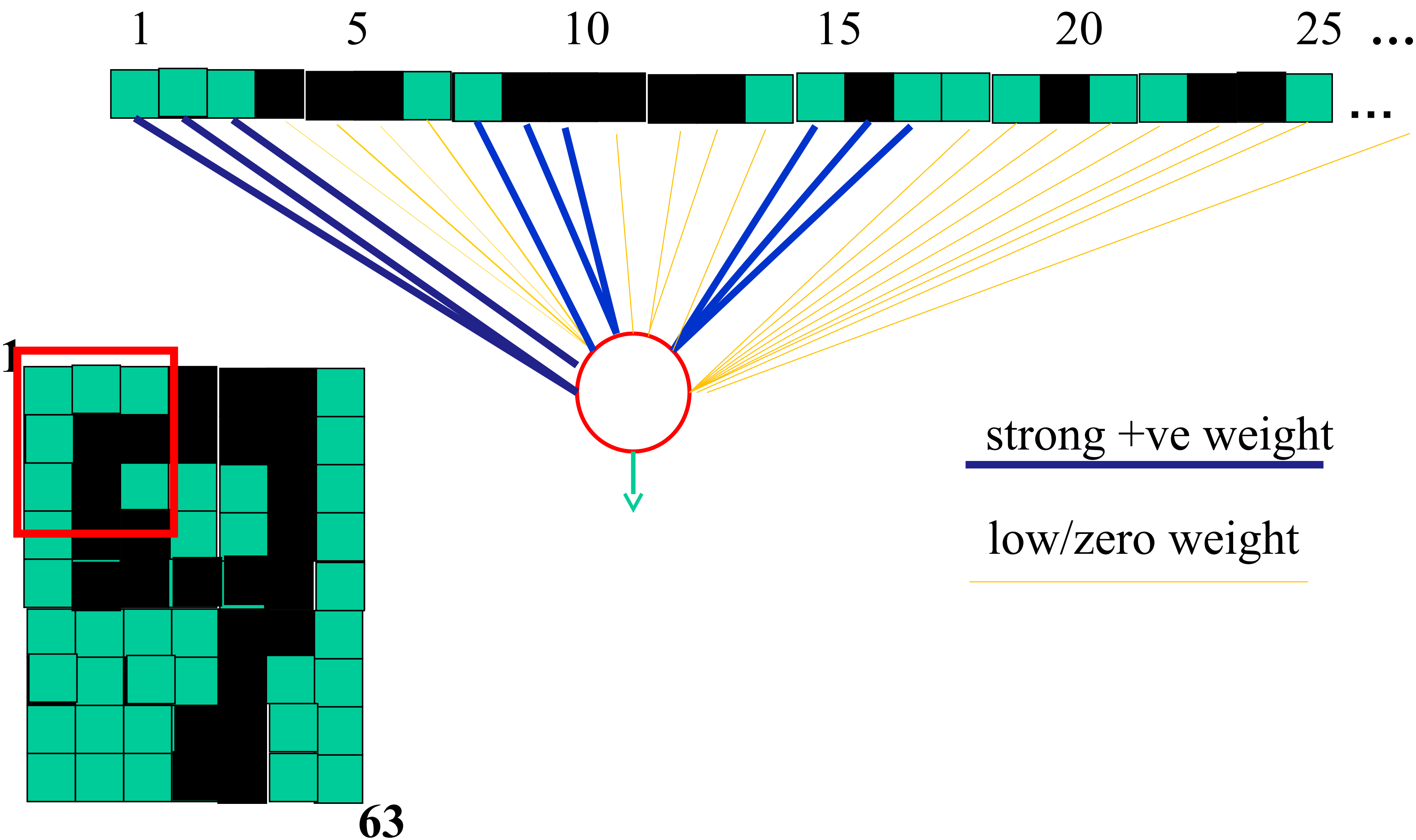


What does this unit detect?

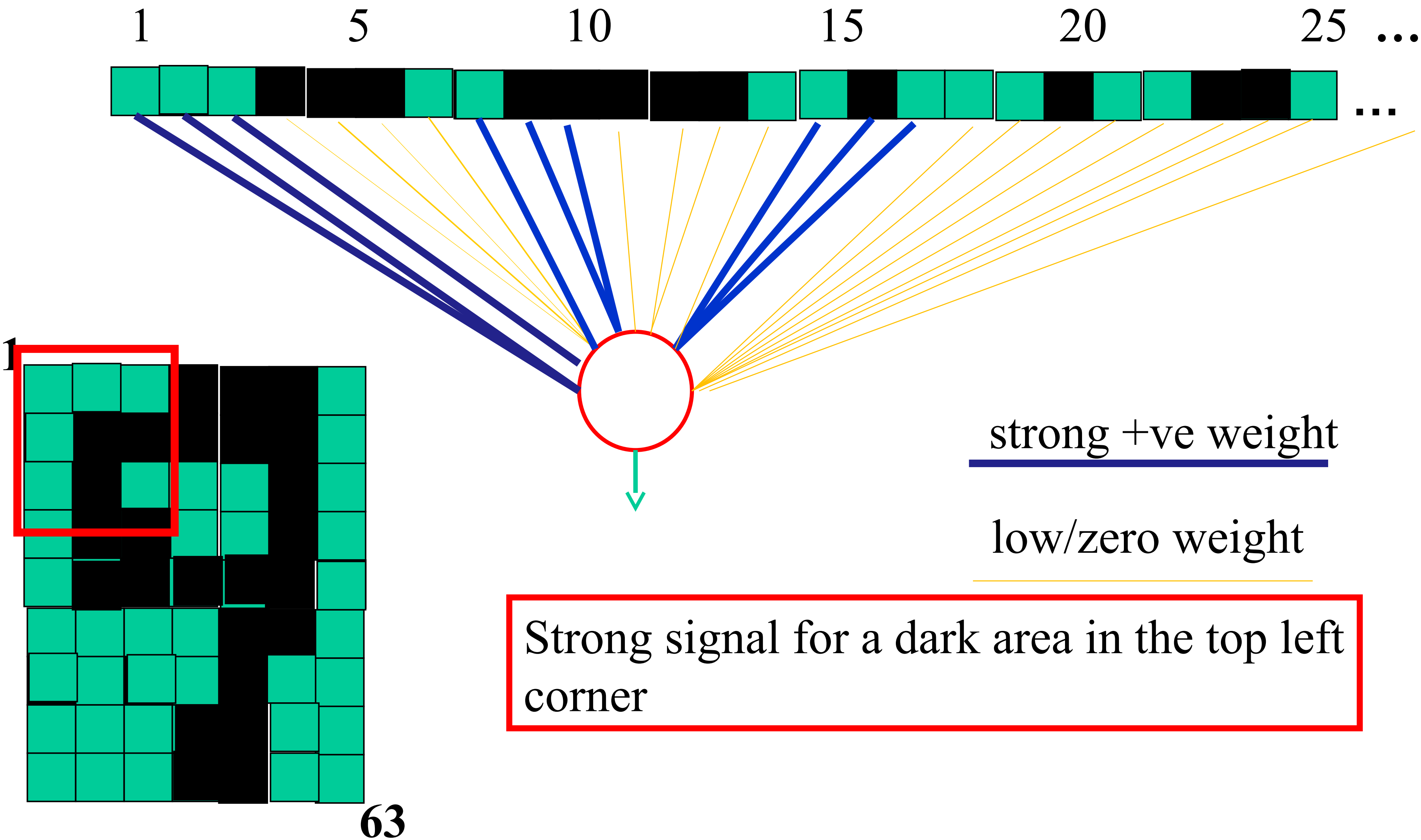


it will send strong signal for a horizontal line in the top row, ignoring everywhere else

What does this unit detect?



What does this unit detect?



Strong signal for a dark area in the top left corner

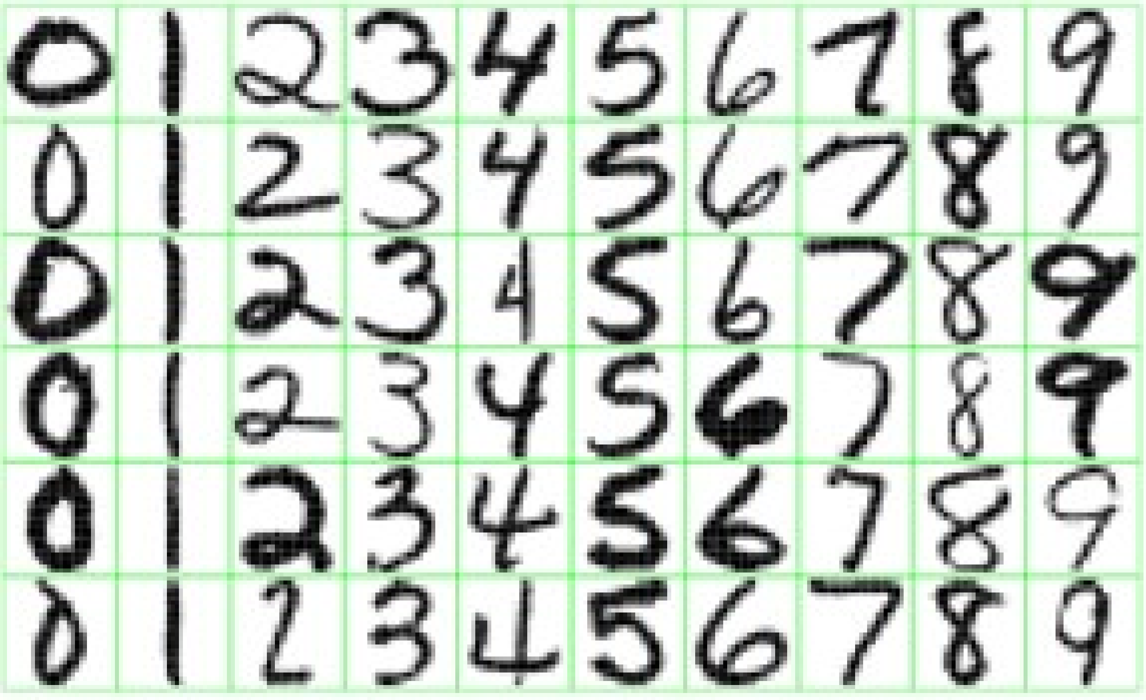
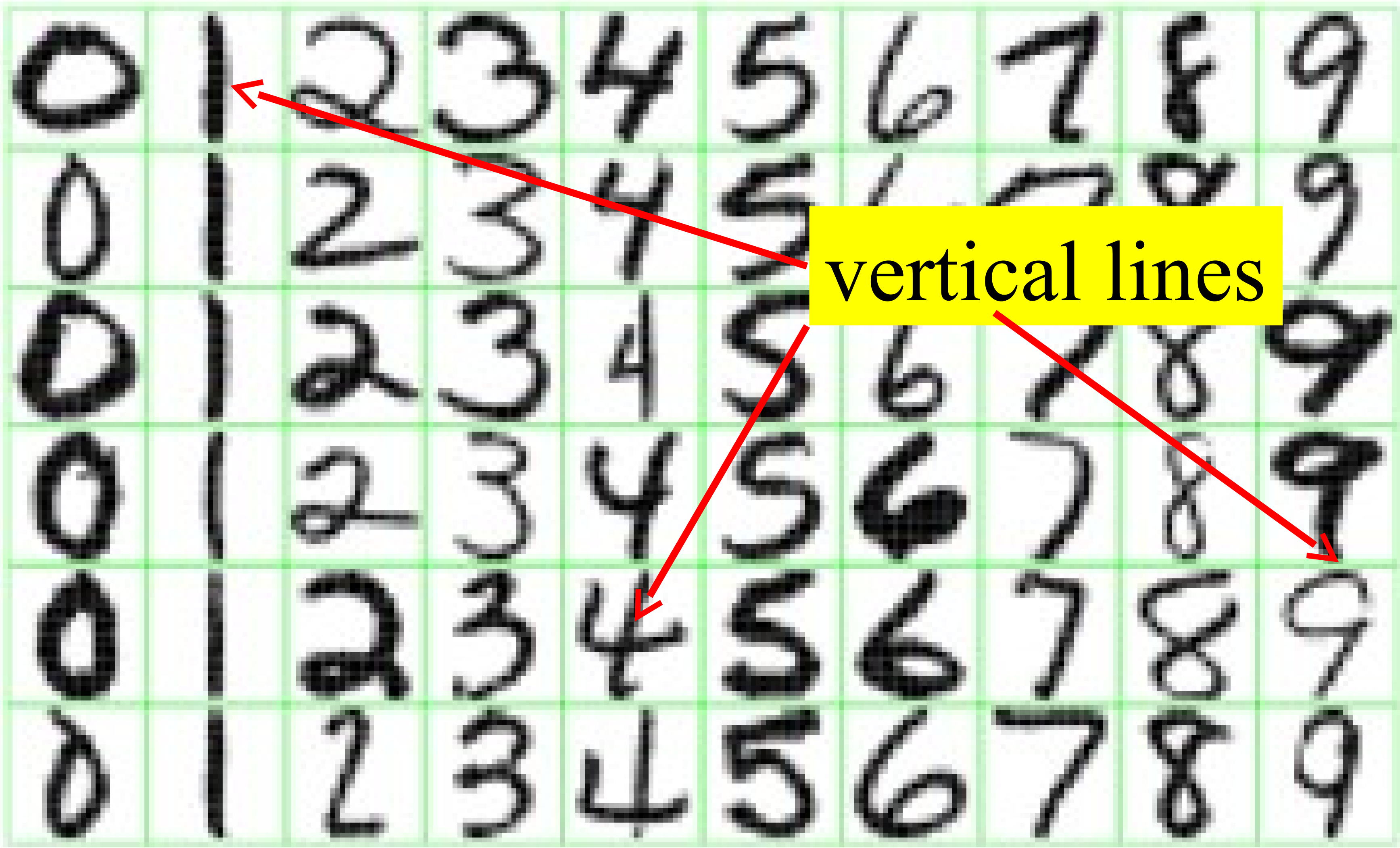


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

What features might you expect a good NN to learn, when trained with data like this?



1

Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

1

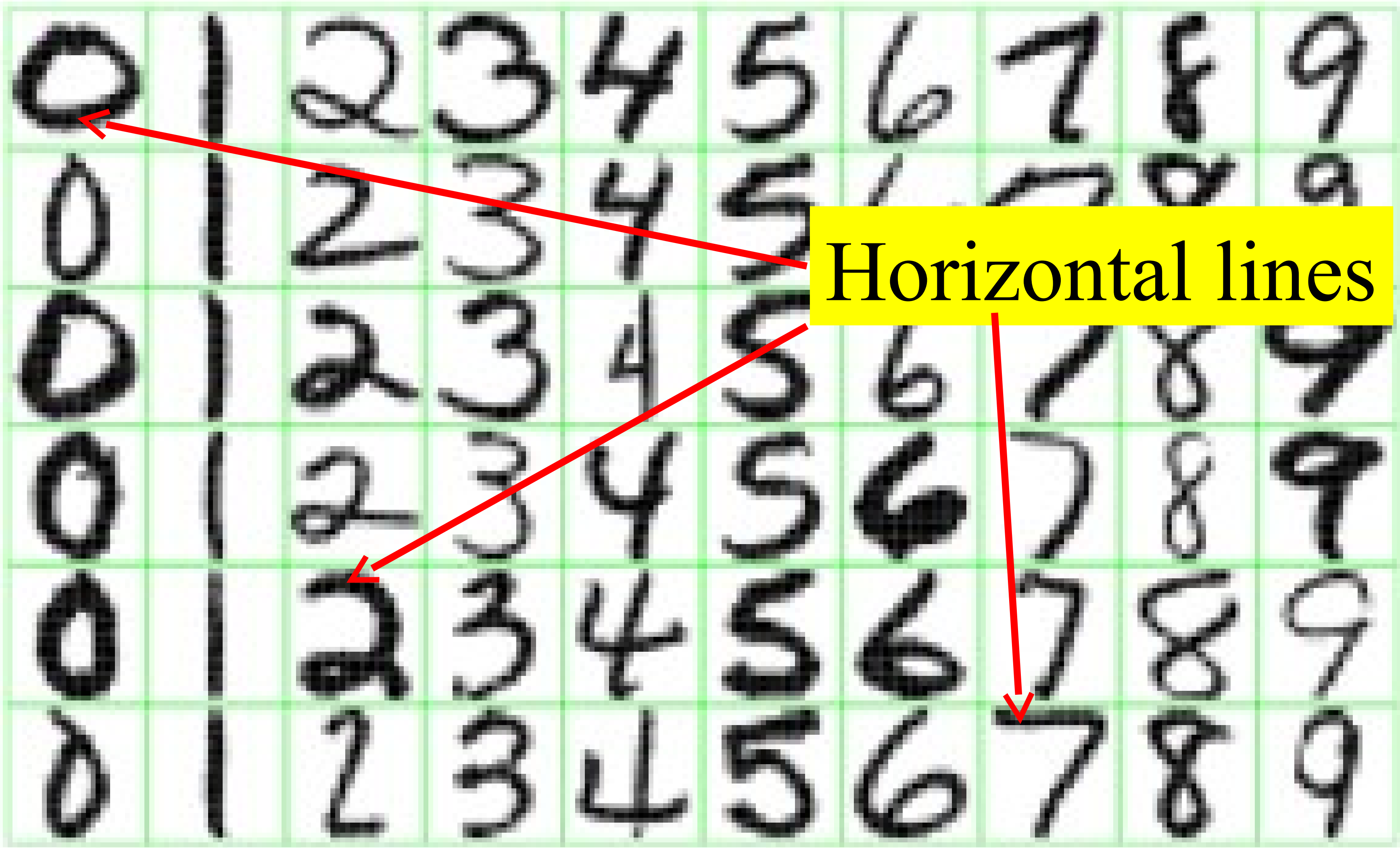


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

1

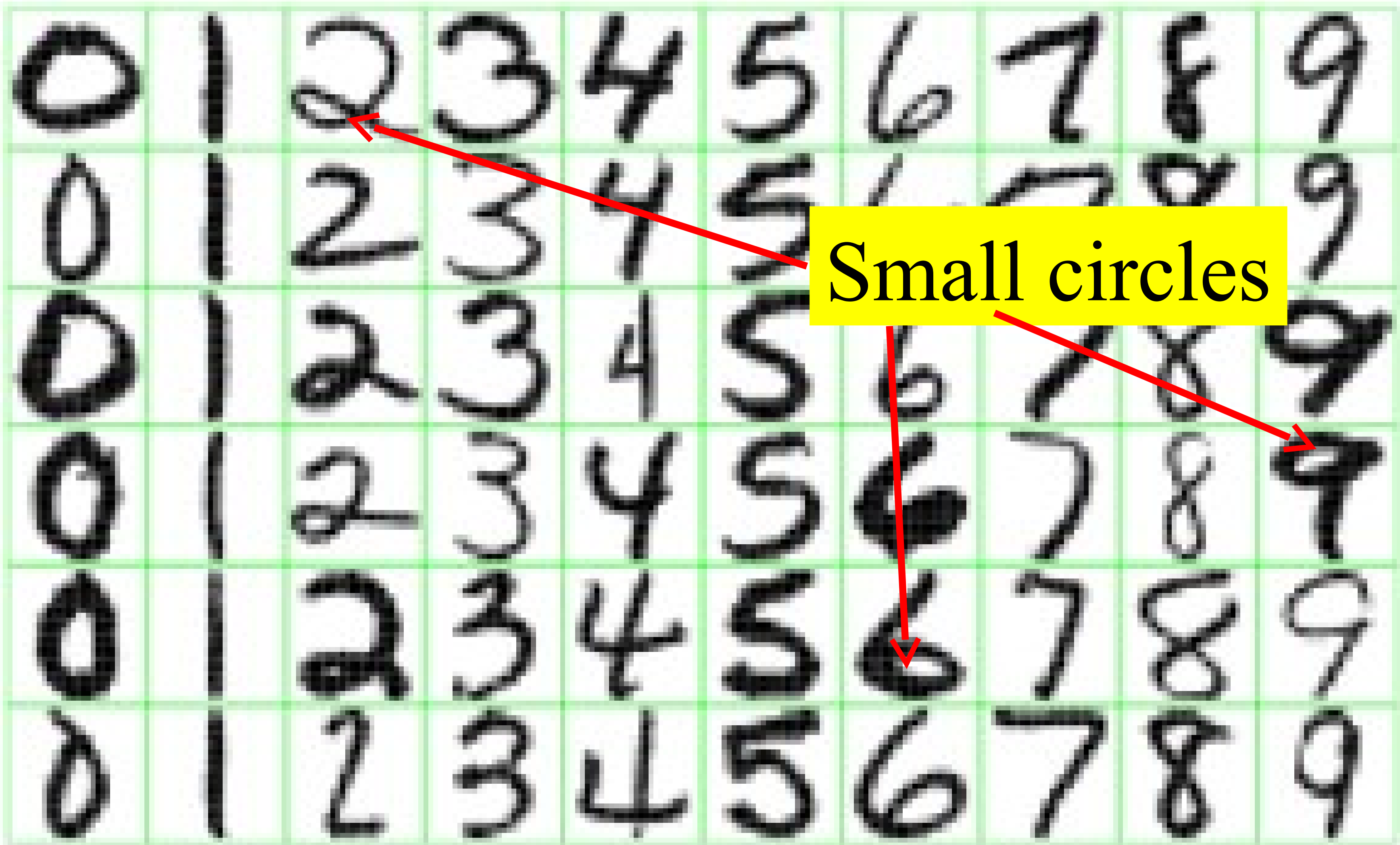
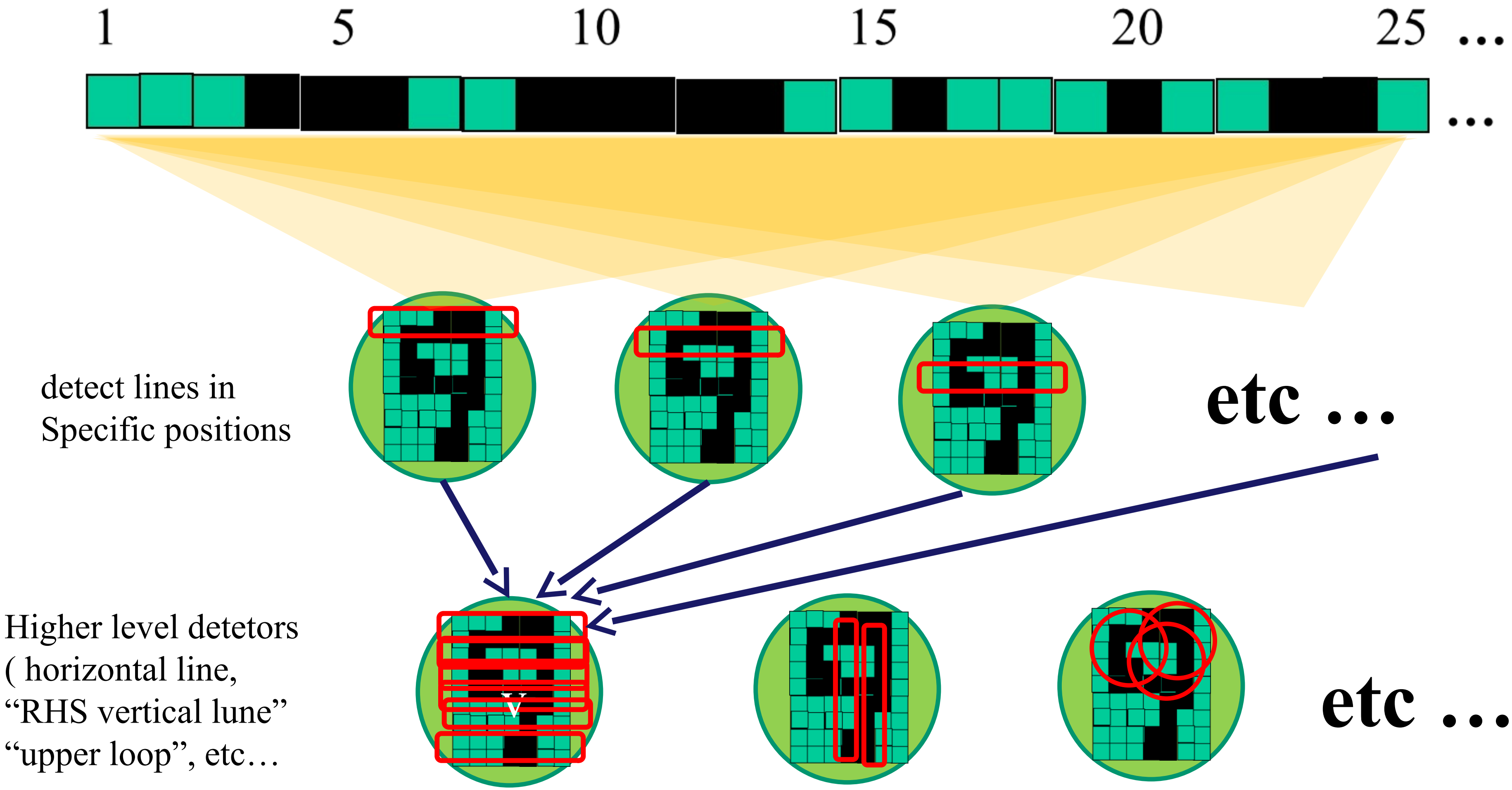


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

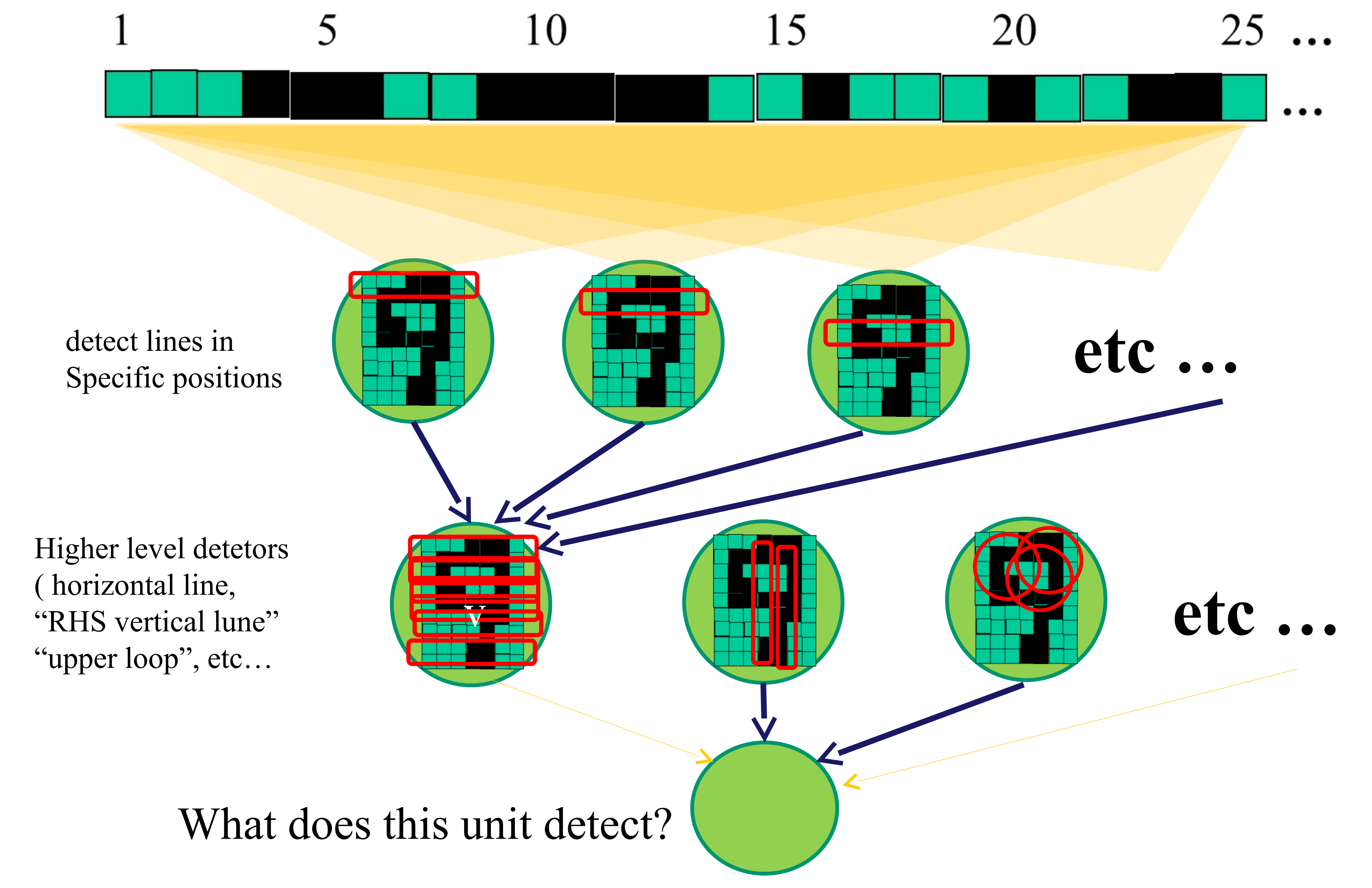
1

But what about position invariance ???
our example unit detectors were tied to
specific parts of the image

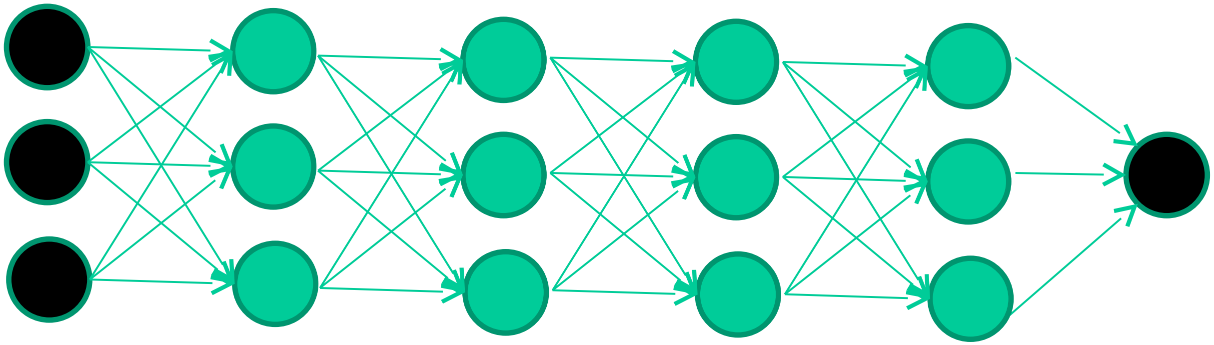
successive layers can learn higher-level features …



successive layers can learn higher-level features …

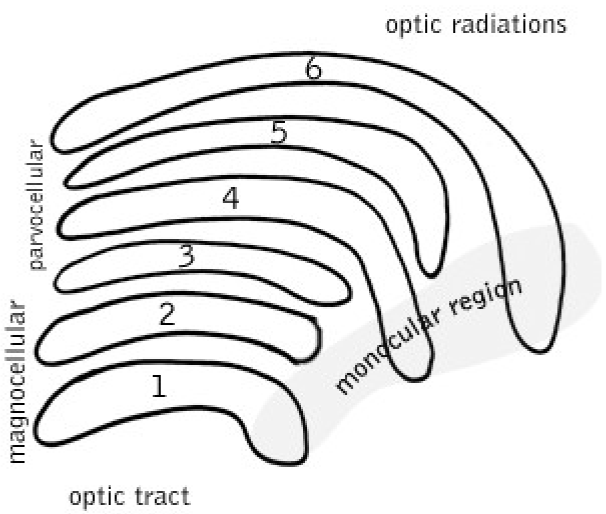


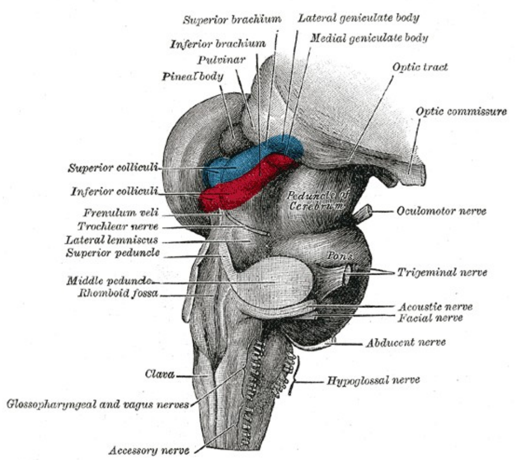
So: *multiple layers make sense*



So: *multiple layers make sense*

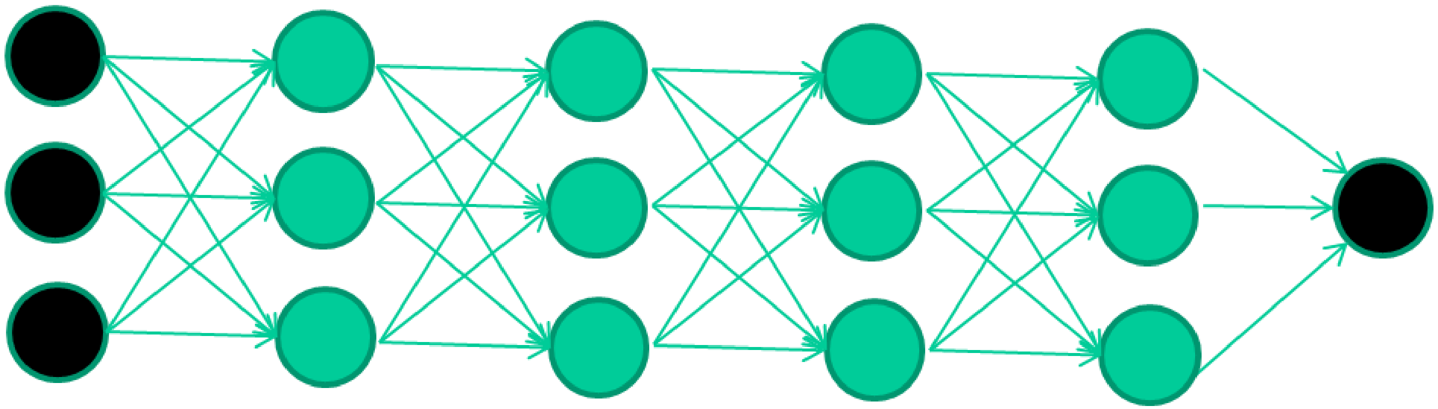
Your brain works that way



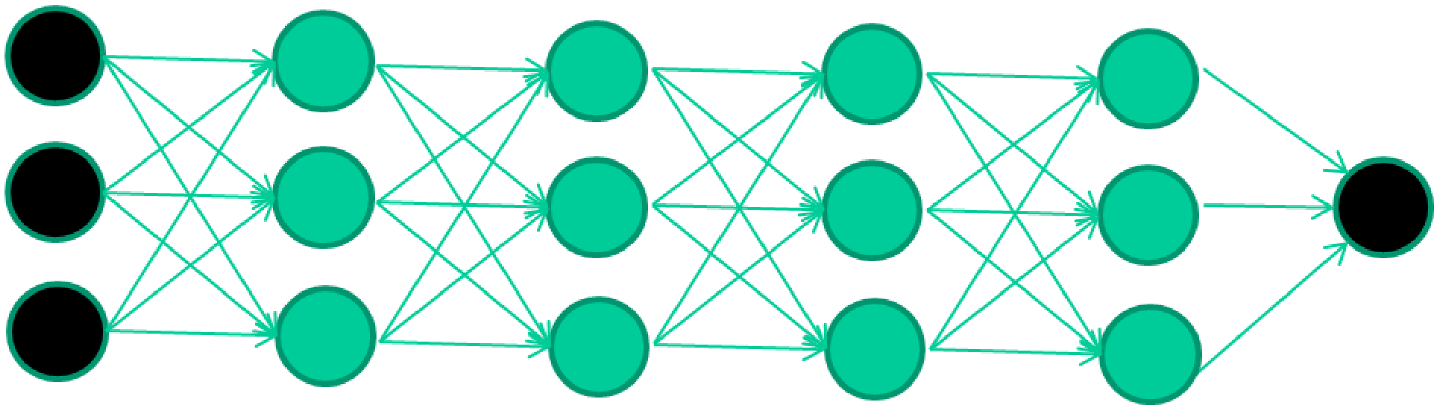


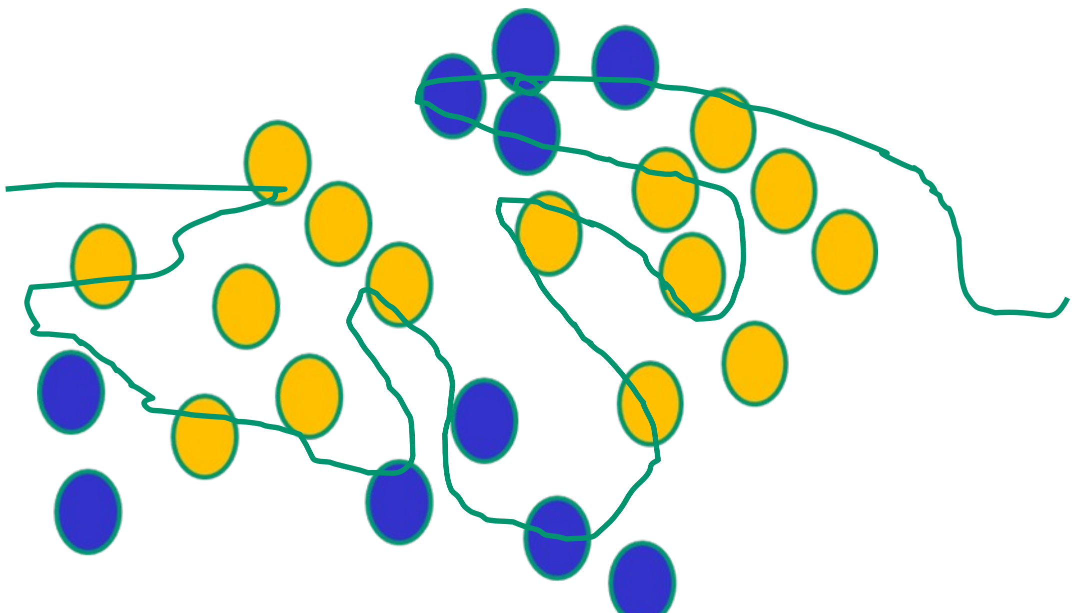
So: *multiple layers make sense*

Many-layer neural network architectures should be capable of learning the true underlying features and 'feature logic', and therefore generalise very well …



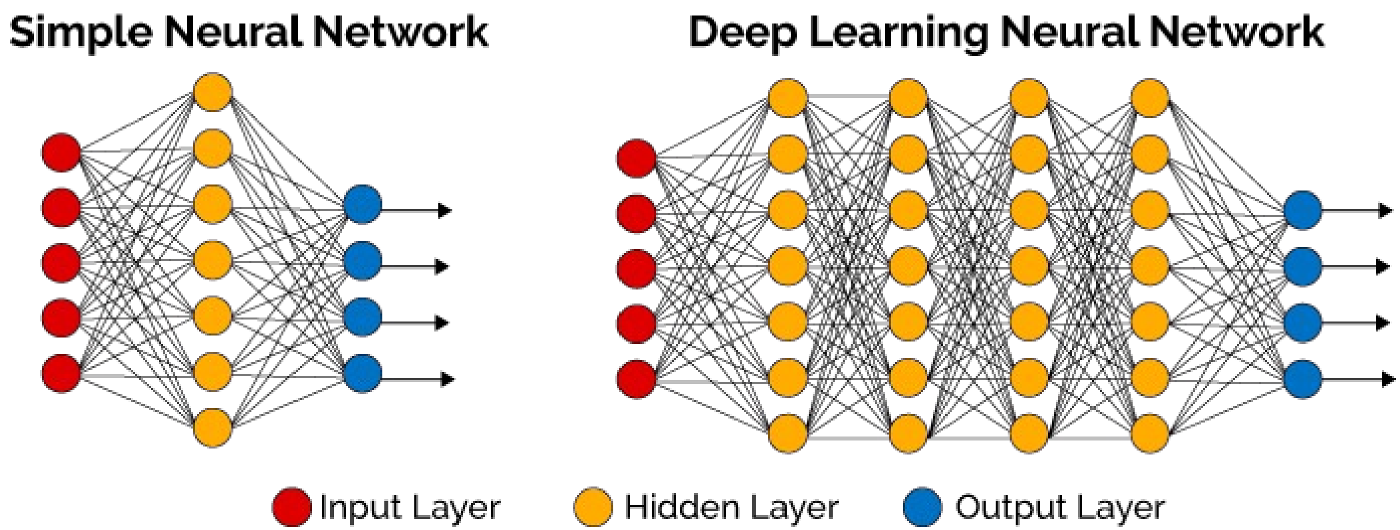
But, until very recently, our weight-learning algorithms simply did not work on multi-layer architectures



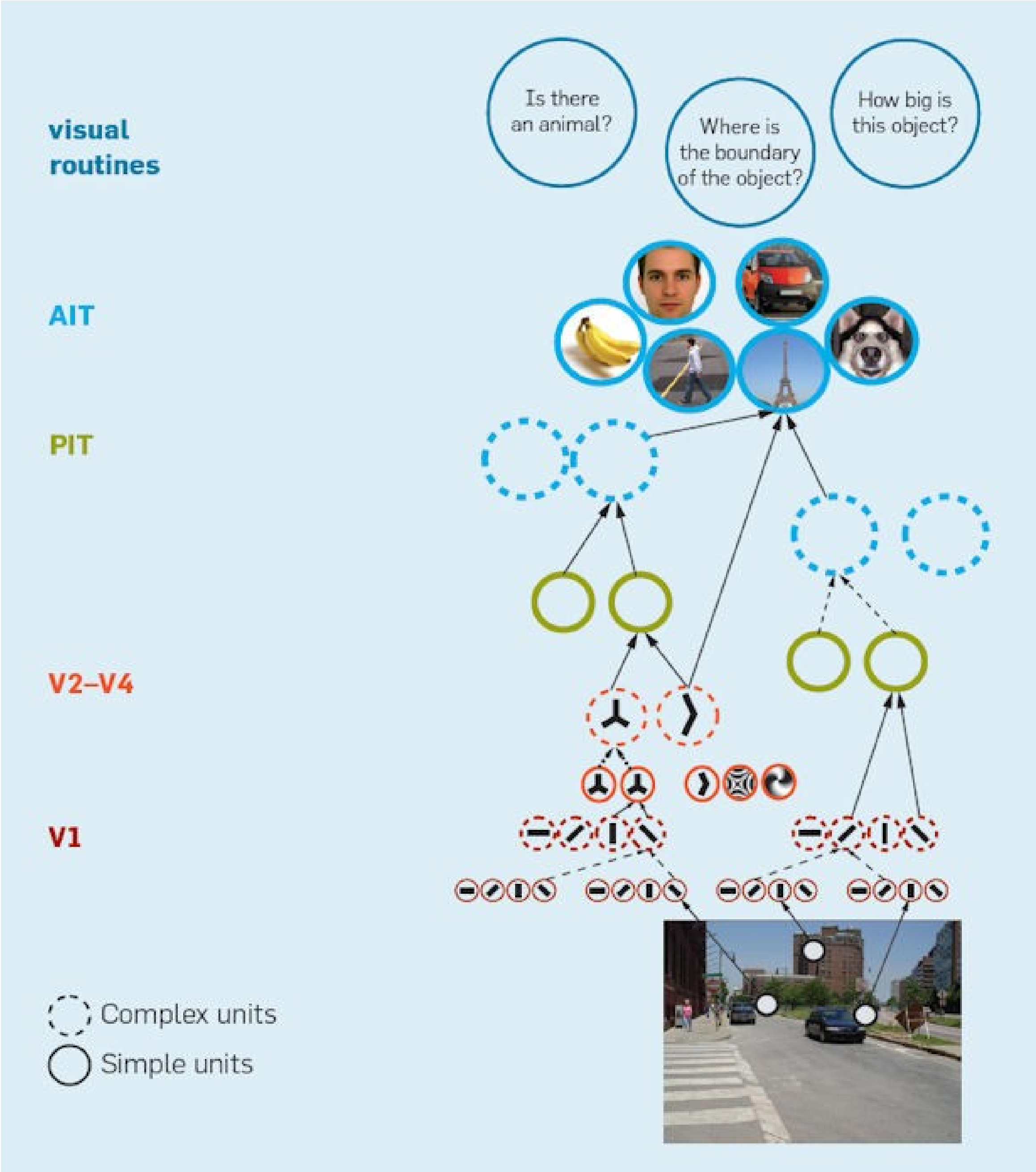


Along came deep learning …

What is deep learning ?



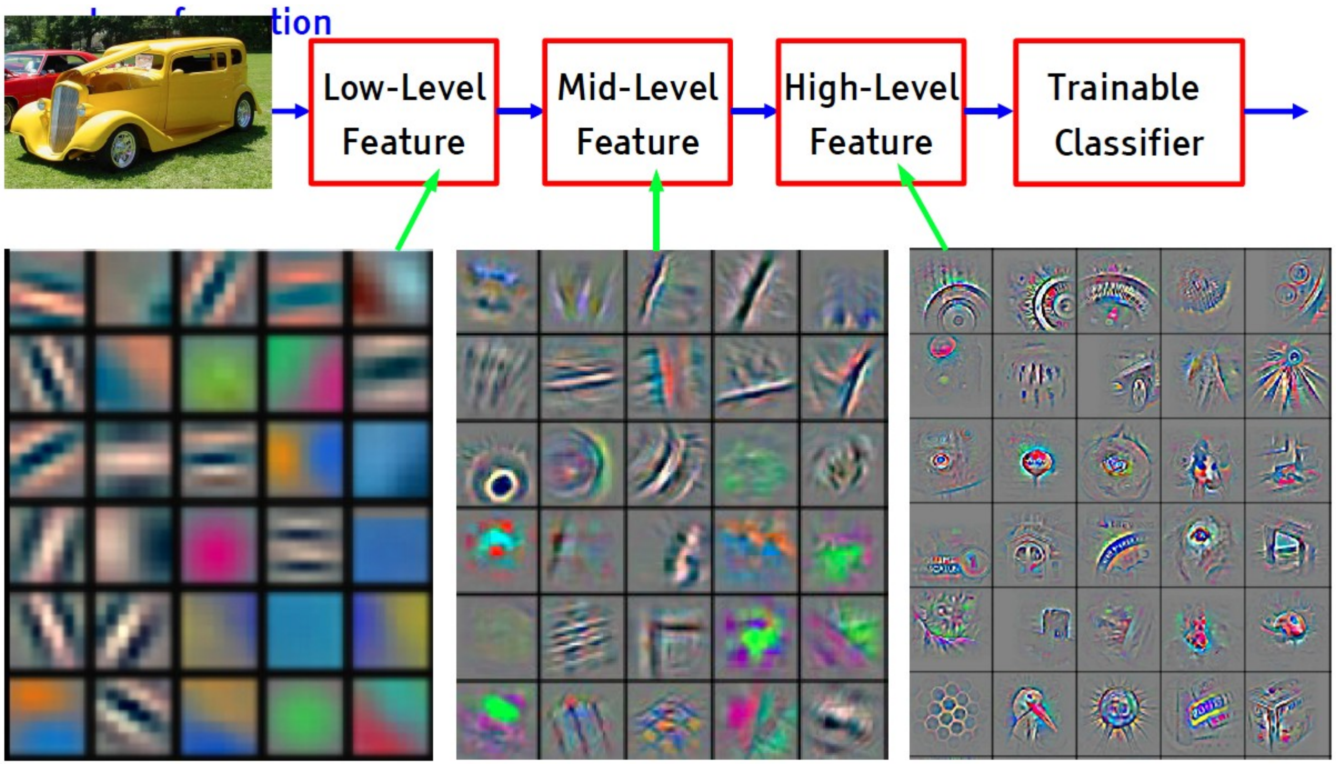
A network with 1 hidden layer can, in theory, learn perfectly any classification problem. A set of weights exists that can produce the targets from the inputs. The problem is finding them.



Hierarchical models

Riesenhuber & Poggio. Nature Neurosci 1999

Deep Learning = Learning Hierarchical Representations

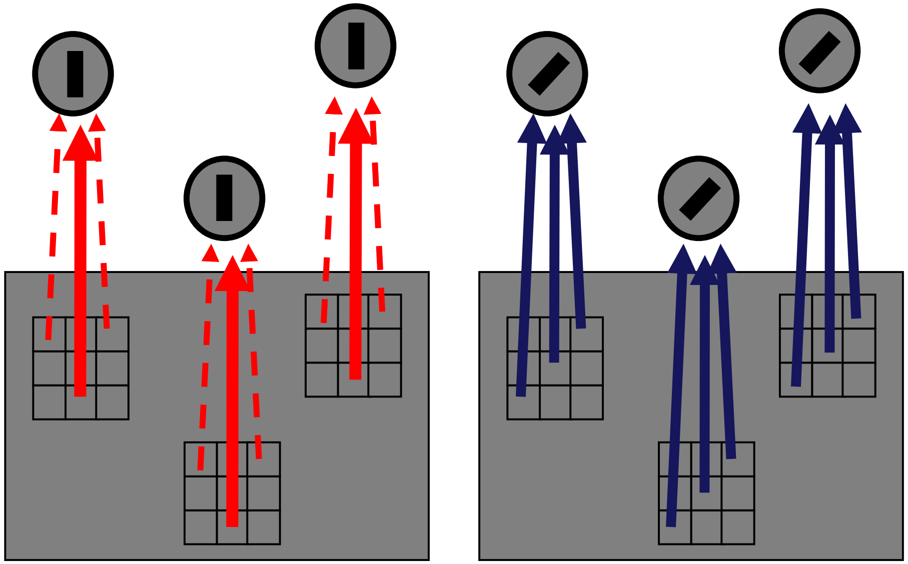


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

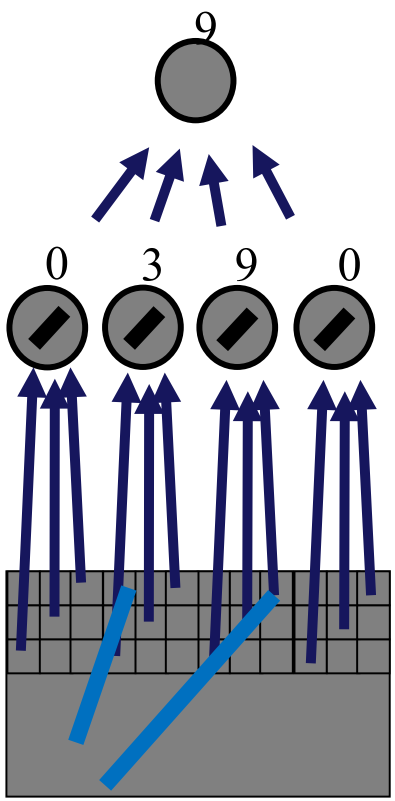
Convolutional Networks (ConvNet or CNN)

(currently the dominant approach for neural networks)

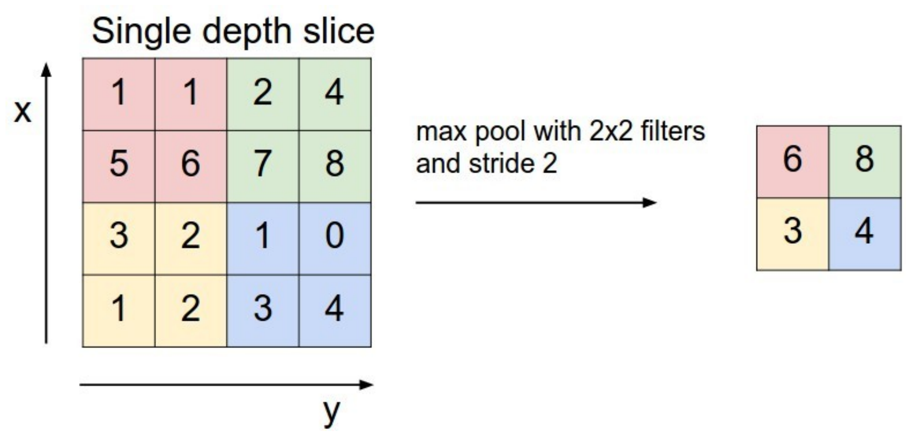
- Use many different copies of the same feature detector with different positions.
  - Replication greatly reduces the number of free parameters to be learned.
- Use several different feature types, each with its own map of replicated detectors.
  - Allows each patch of image to be represented in several ways.



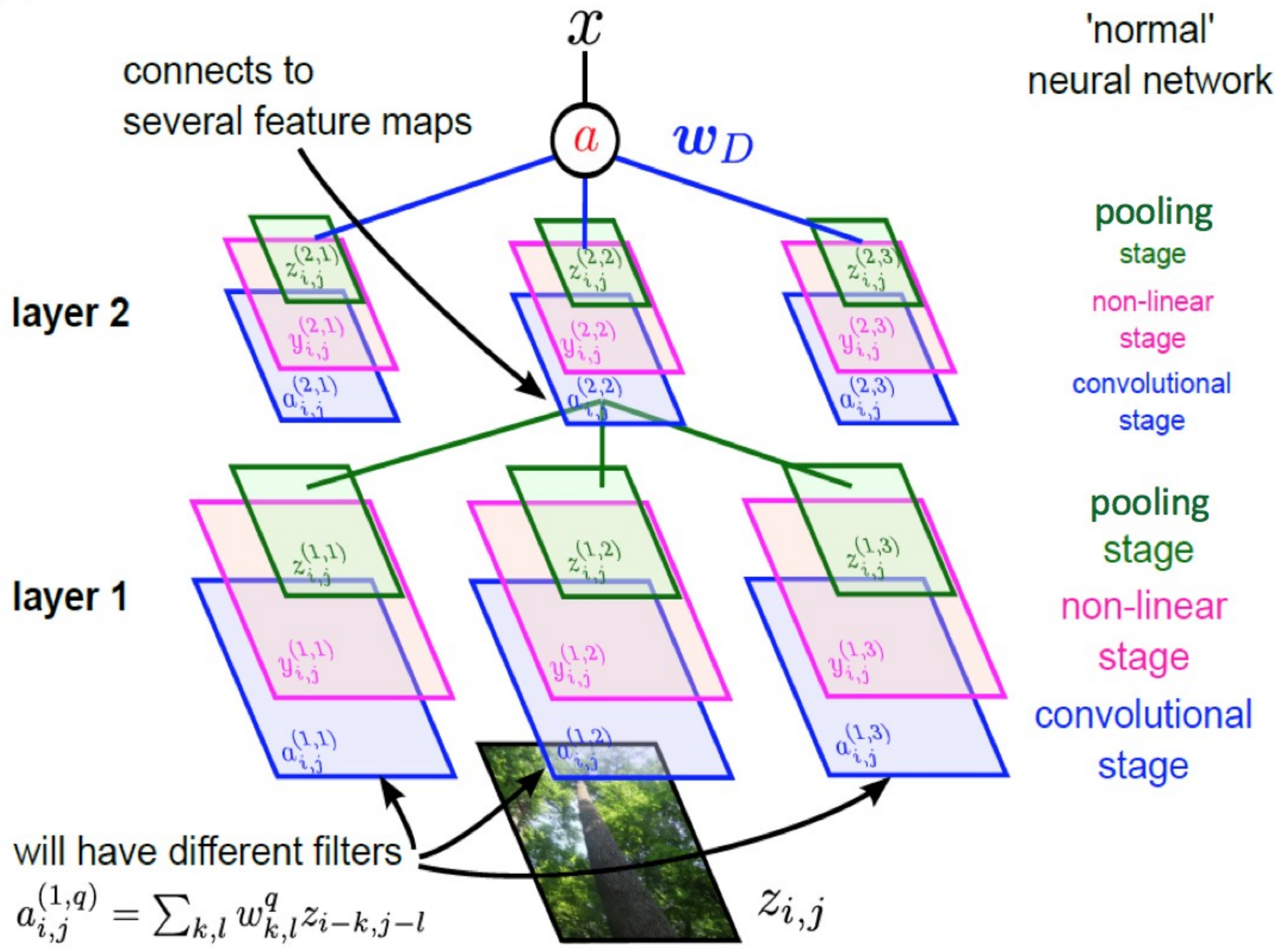
CNN Architecture: Pooling Layer



- Pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value of the features in that region.
- Intuition: to progressively **reduce the spatial size** of the representation to **reduce the amount of parameters and computation** in the network, and hence to also **control overfitting**

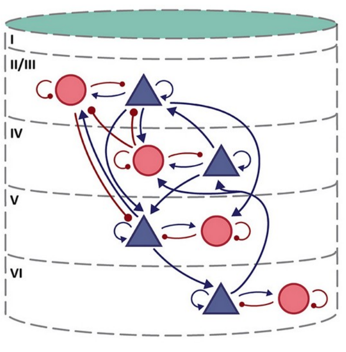


Full CNN

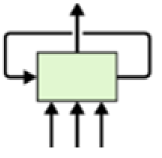


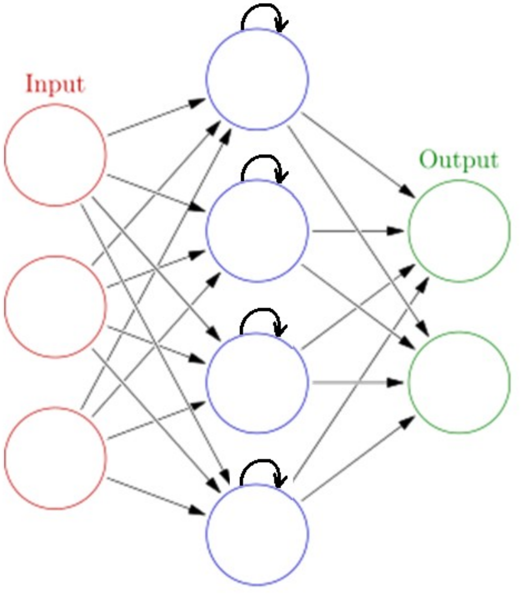
x
'normal'
neural network
connects to
several feature maps
a
$\boldsymbol{w}_D$
pooling
stage
non-linear
stage
convolutional
stage
layer 2
$z_{i,j}^{(2,1)}$
$y_{i,j}^{(2,1)}$
$a_{i,j}^{(2,1)}$
$z_{i,j}^{(2,2)}$
$y_{i,j}^{(2,2)}$
$a_{i,j}^{(2,2)}$
$z_{i,j}^{(2,3)}$
$y_{i,j}^{(2,3)}$
$a_{i,j}^{(2,3)}$
layer 1
$z_{i,j}^{(1,1)}$
$y_{i,j}^{(1,1)}$
$a_{i,j}^{(1,1)}$
$z_{i,j}^{(1,2)}$
$y_{i,j}^{(1,2)}$
$a_{i,j}^{(1,2)}$
$z_{i,j}^{(1,3)}$
$y_{i,j}^{(1,3)}$
$a_{i,j}^{(1,3)}$
pooling
stage
non-linear
stage
convolutional
stage
will have different filters
$a_{i,j}^{(1,q)} = \sum_{k,l} w_{k,l}^q z_{i-k,j-l}$
$z_{i,j}$

Recurrent Neural Networks and LSTM neurons

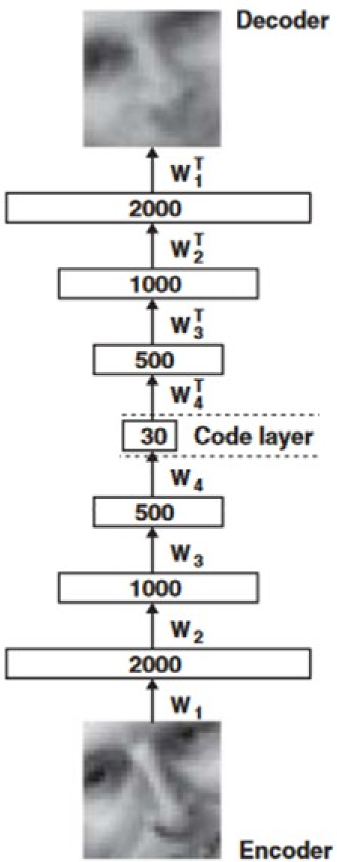


Potjans and Diesmann (2014)





Note: No top-down feedback connections from top layers



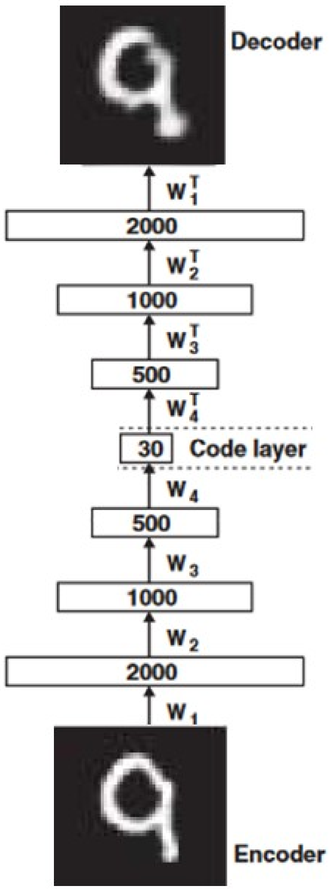
Autoencoder

- Train the neural network to reproduce its input vector as its output

- This forces it to compress as much information as possible into few numbers in the central bottleneck.

- These few (here 30) numbers are then a good way to represent data.

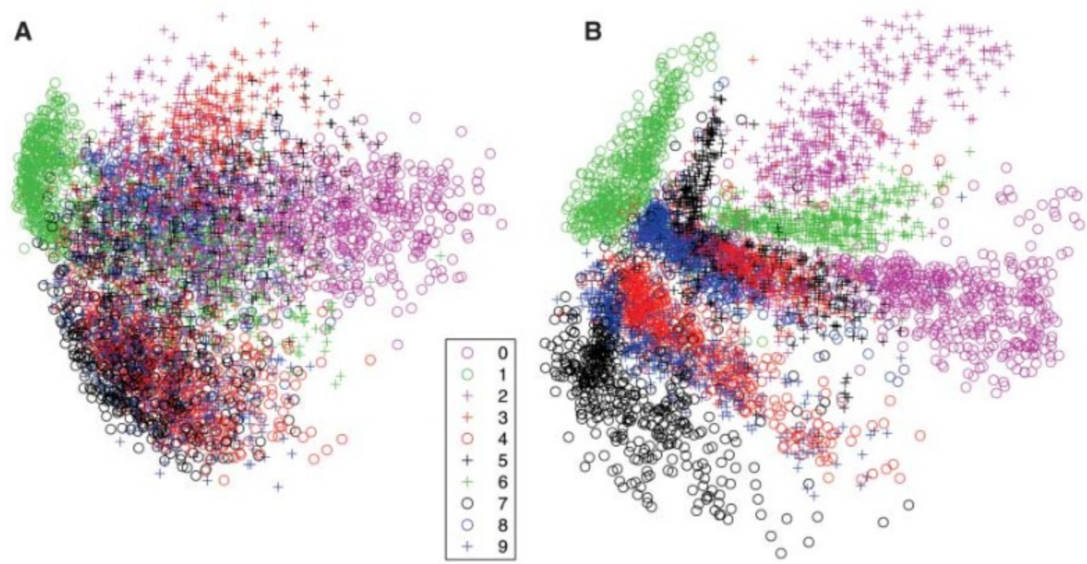
Autoencoder



Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov 28 JULY 2006 VOL 313 SCIENCE

Fig. 3. **(A)** The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. **(B)** The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



Convolutional Autoencoder

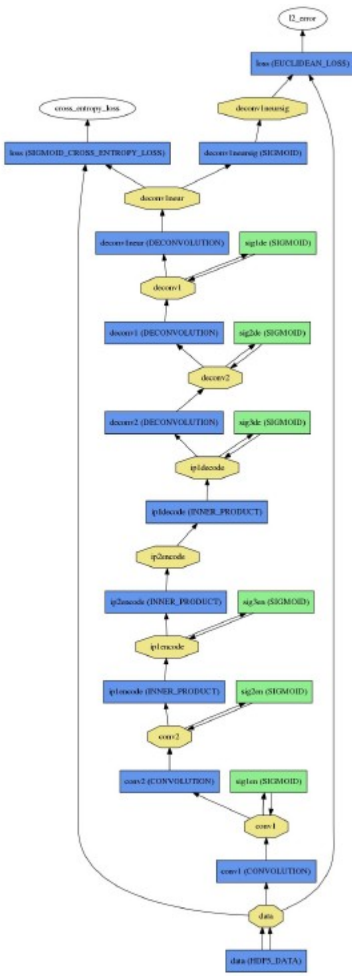


Figure 3. CAE model in Caffe

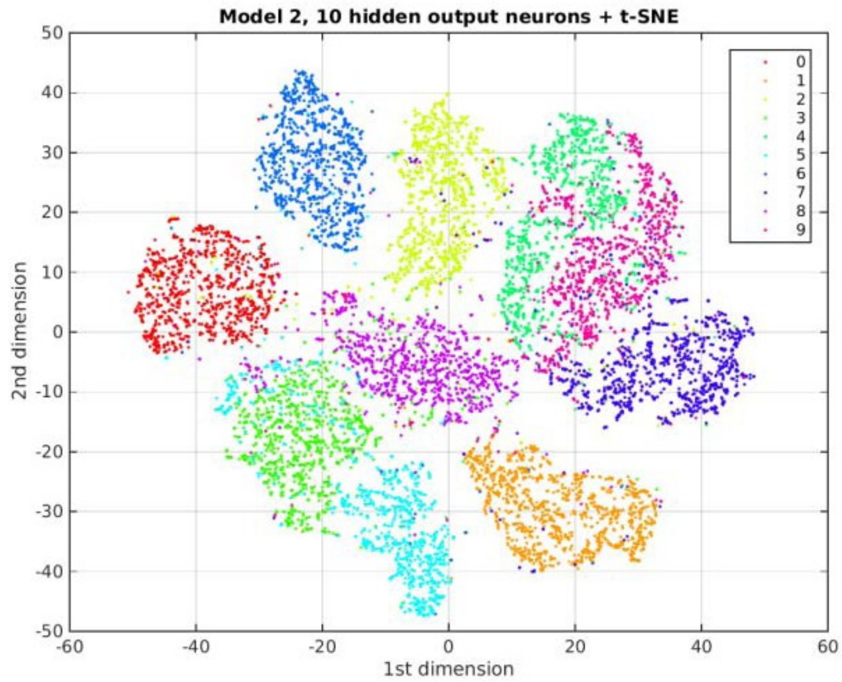
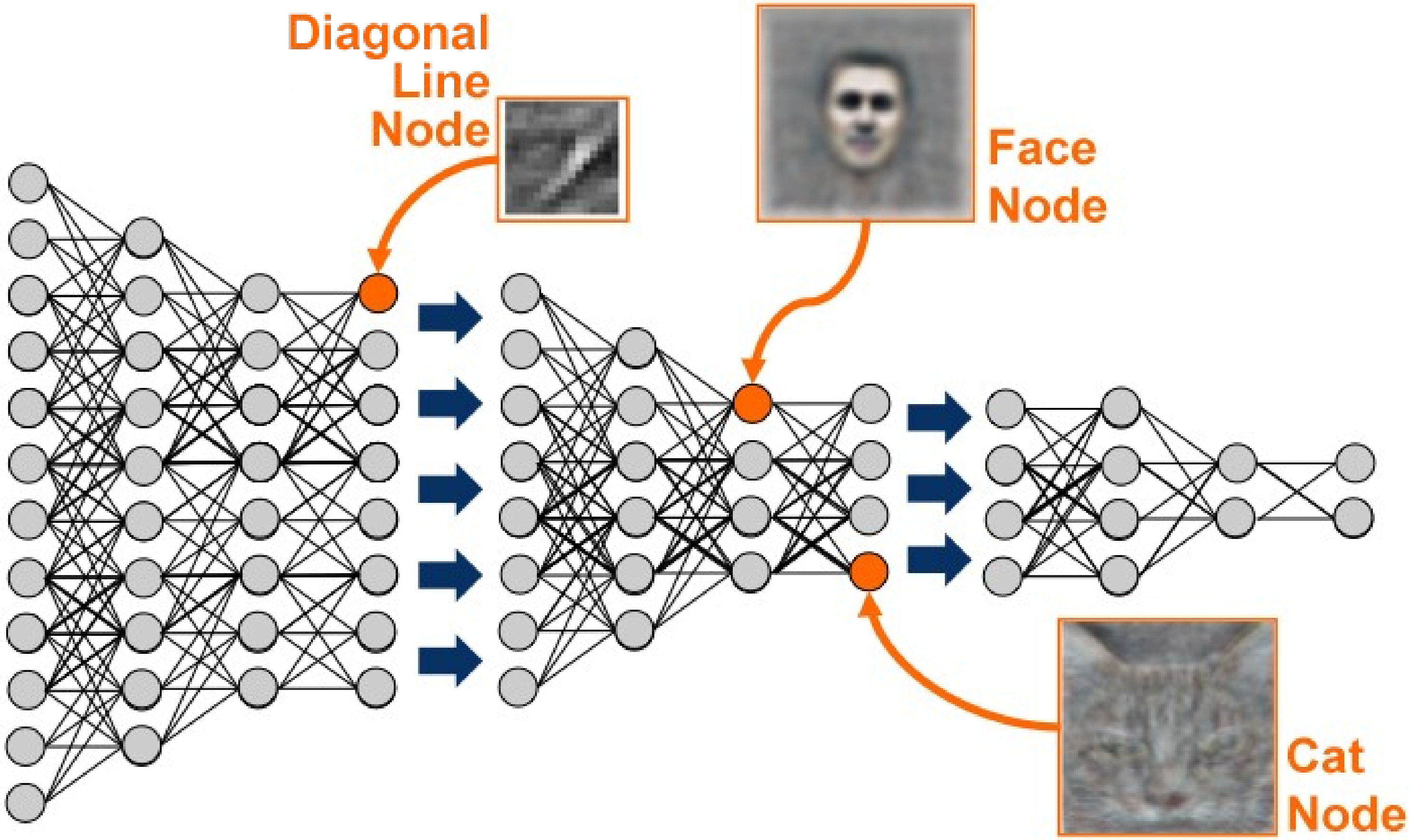


Figure 5. Visualization of MNIST test set in a 2D space by 10-dimensional CAE Model 2 + t-SNE

Turchenko & Luczak, IEEE IDAACS 2017

Deep Neuronal Networks

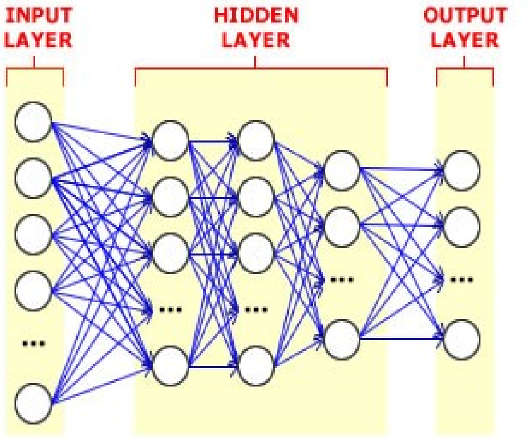


Le et al. (2013) *ICASSP, IEEE International Conference*

http://theanalyticsstore.ie/deep-learning/

And that's that

- That's the basic idea
- There are many many types of deep learning,
- different kinds of autoencoder, variations on architectures and training algorithms, etc…
- Very fast growing area …



Thanks

- Shangsong Liang
- Sun Yat-sen University
- liangshangsong@gmail.com