Introduction to Neural Networks and Convolution Networks (CNNs)

Paul Rodriguez, PhD
SDSC

Outline

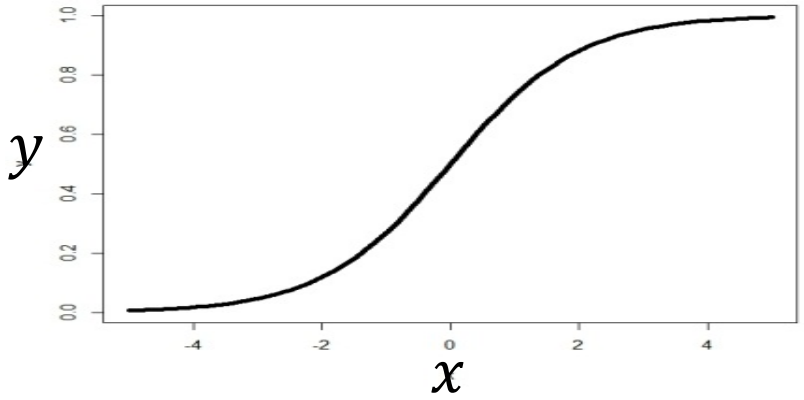
- Overview (review) of Neural Networks (aka Multilayer Perceptron)
- What is Deep Learning?
- Introduction to convolution and feature discovery
- Convolution Neural Networks

to get neural network:

Consider the Logistic Function

(aka sigmoid)

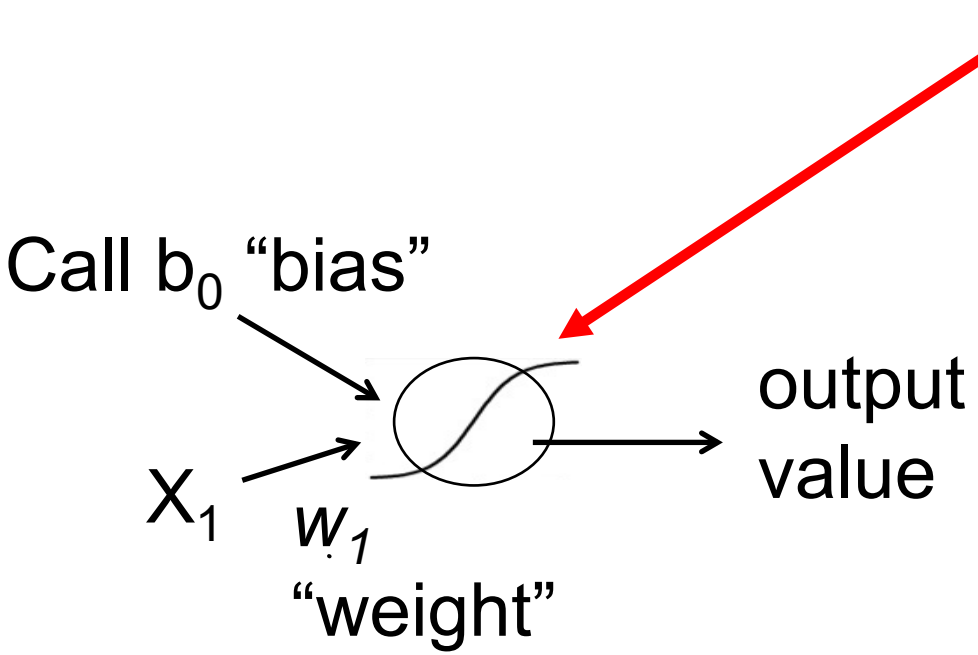
$$f(x_i) = \frac{1}{1 + exp^{(-(b+wx))}}$$



$$b = 0 \quad , \; w_1 = 1$$

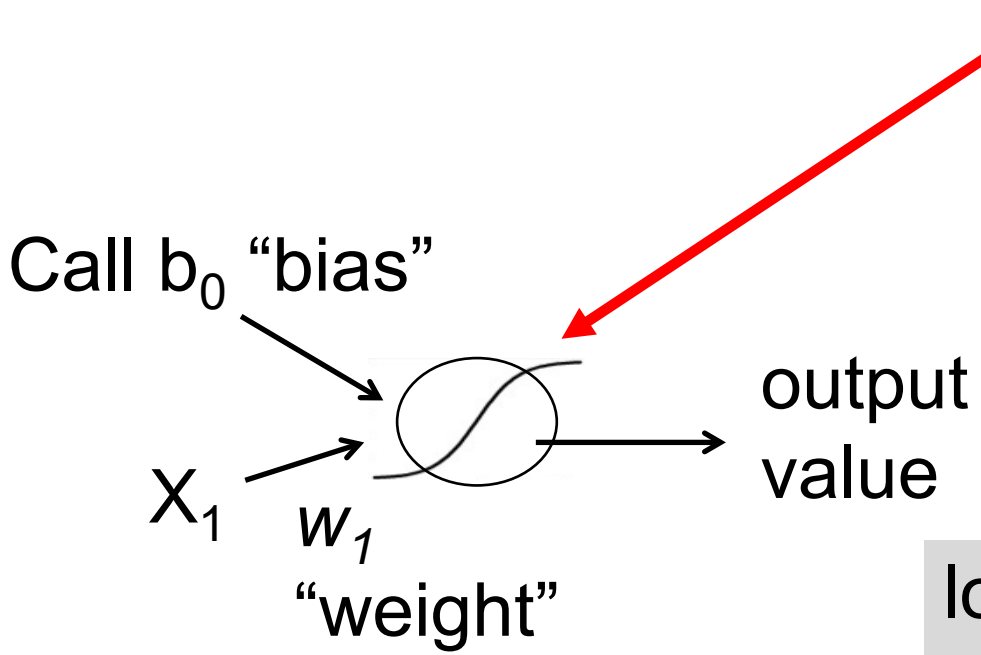
Make a graphical description of Logistic Function

$$f(x_i) = \frac{1}{1 + exp^{(-(b+wx))}}$$



Make a graphical description of Logistic Function

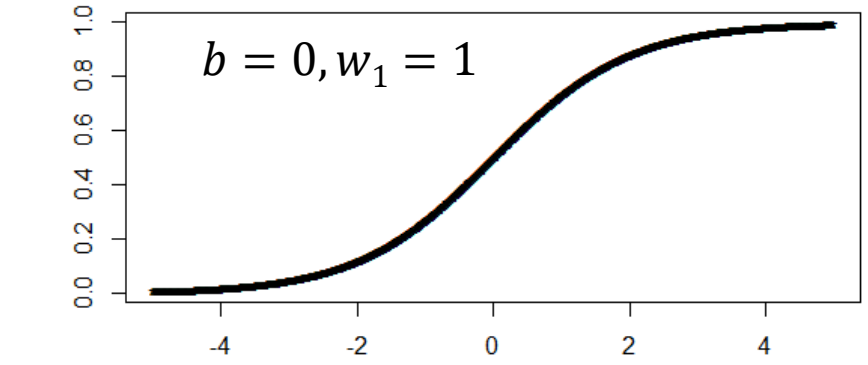
$$f(x_i) = \frac{1}{1 + exp^{(-(b+wx))}}$$

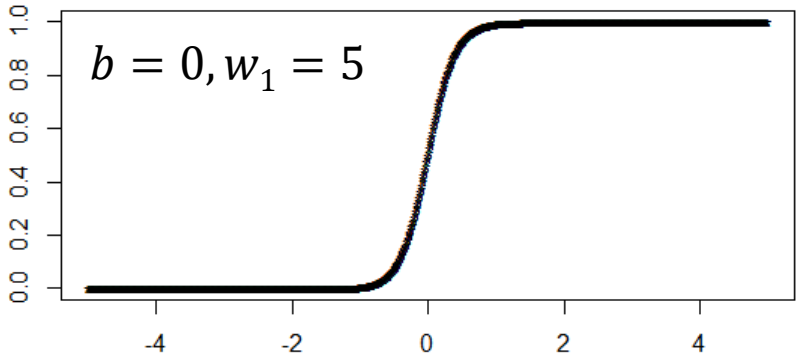


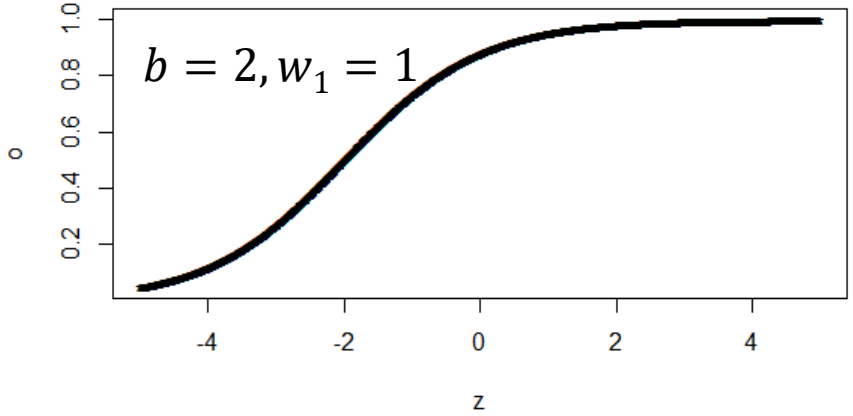
logistic function will transform input to output – call it the 'activation' function

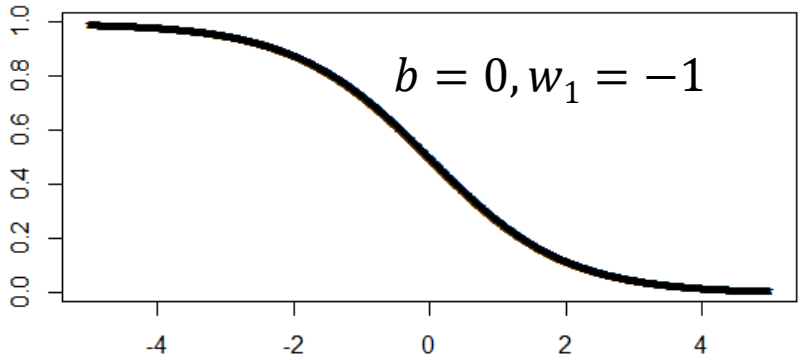
Logistic function w/various weights

$for\ y = 1/(1+exp(-(b+w_1 * x))$

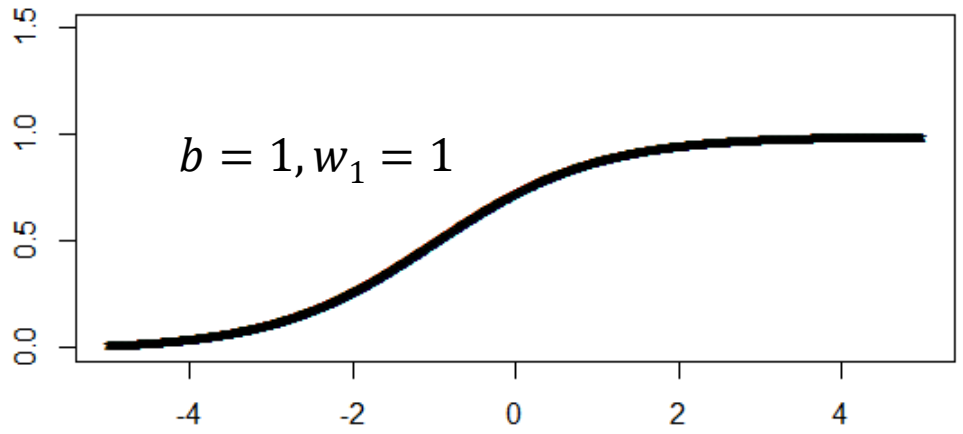


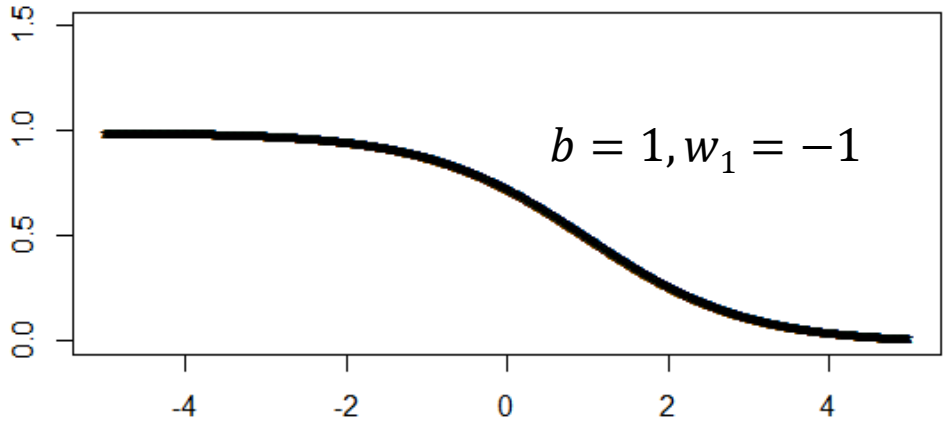




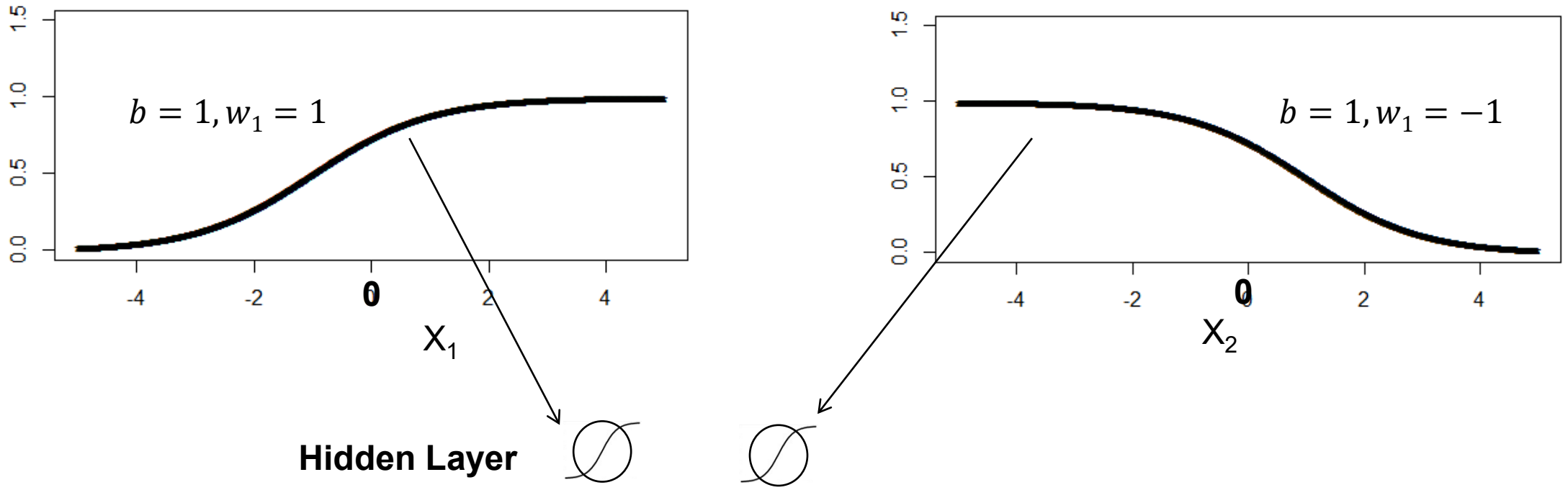


So combinations are highly flexible and nonlinear

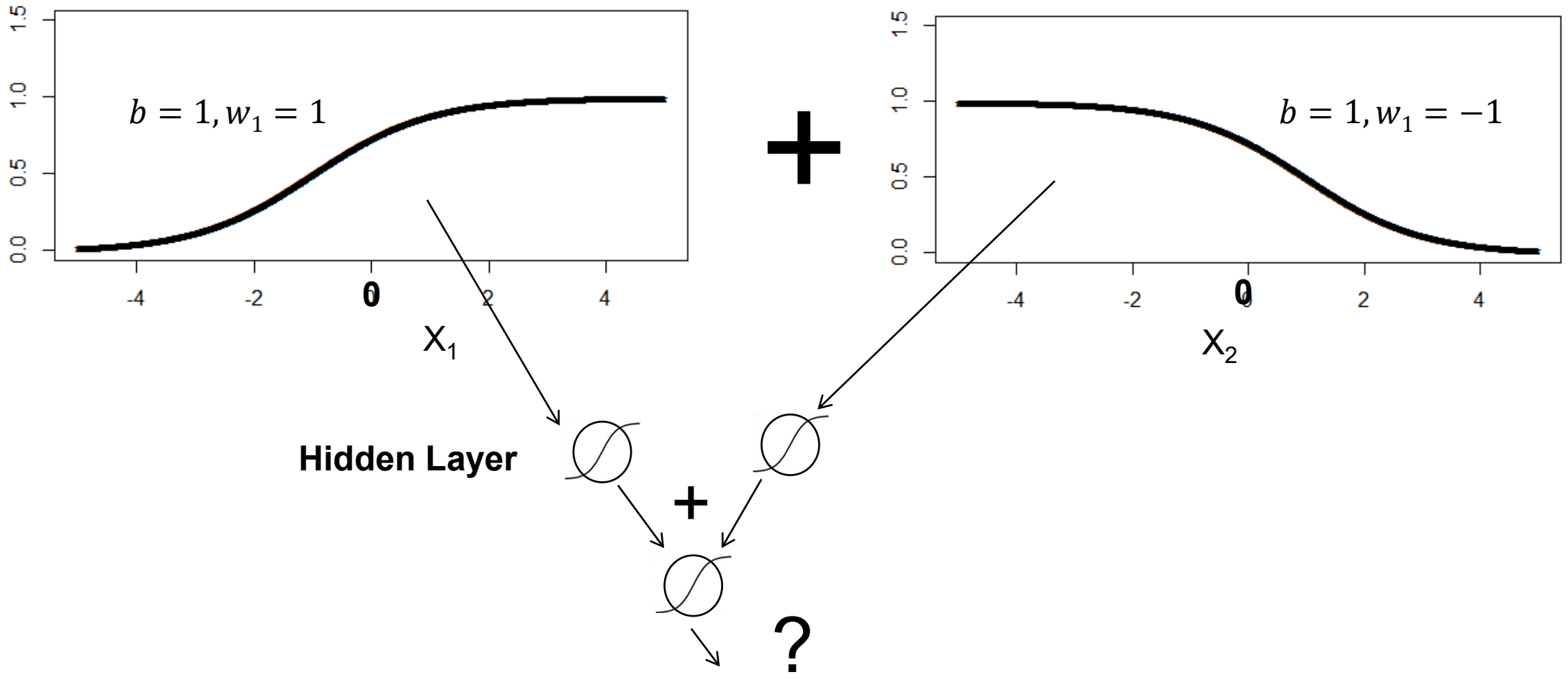




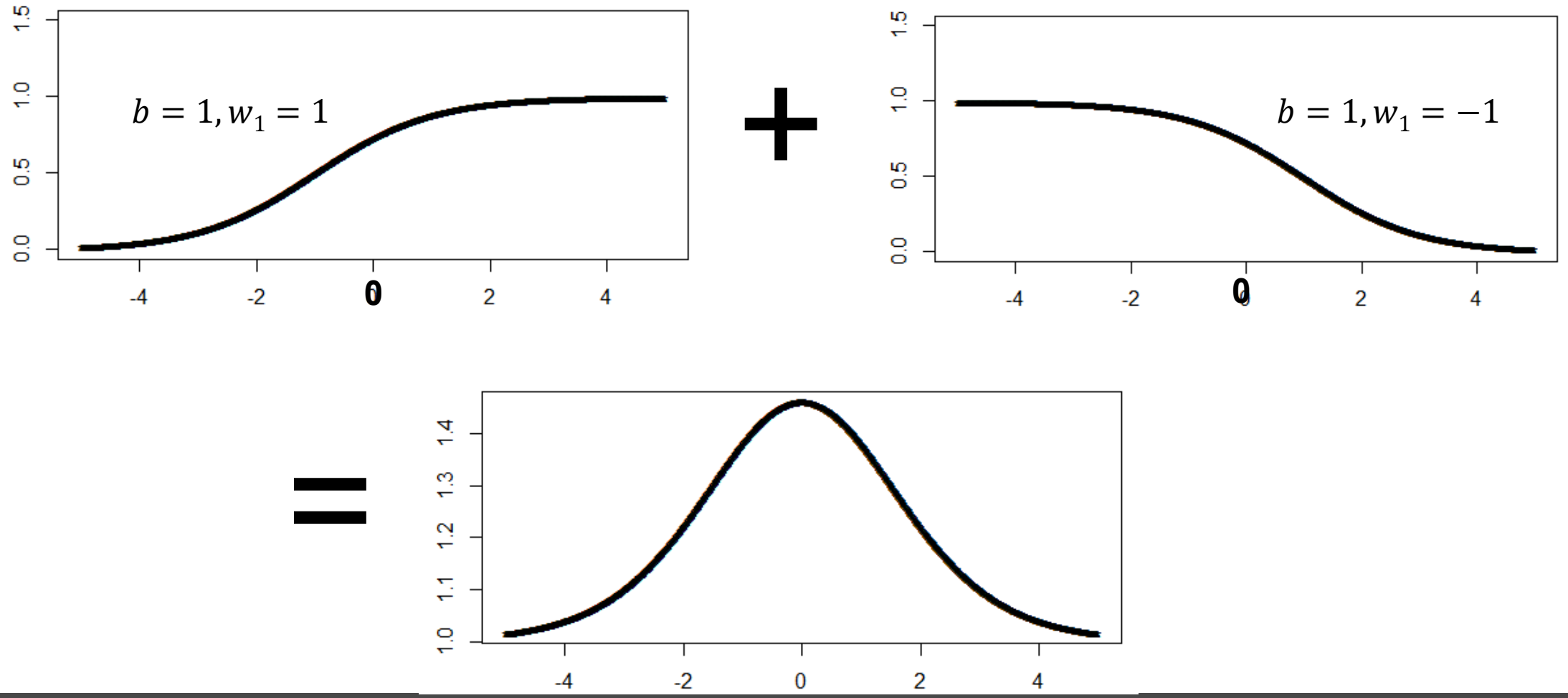
So combinations are highly flexible and nonlinear



So combinations are highly flexible and nonlinear

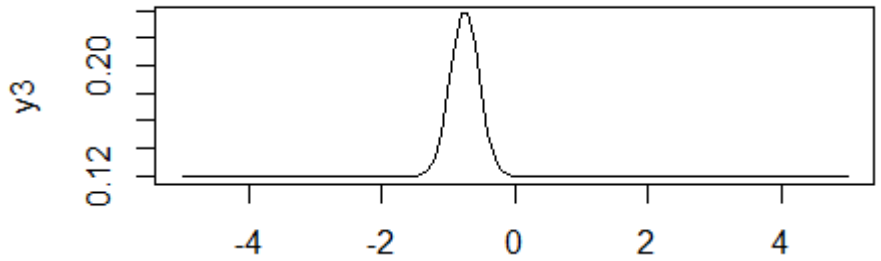


So combinations are highly flexible and nonlinear

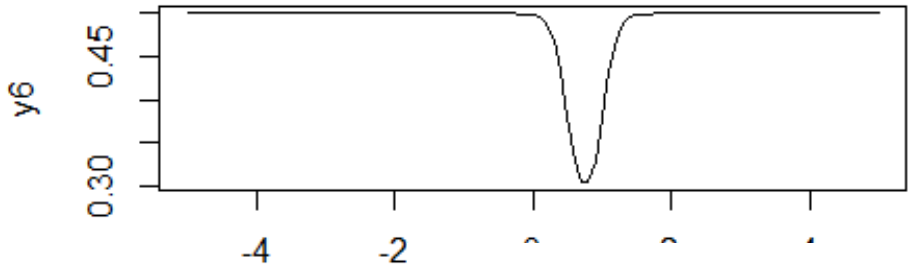


Higher level function combinations

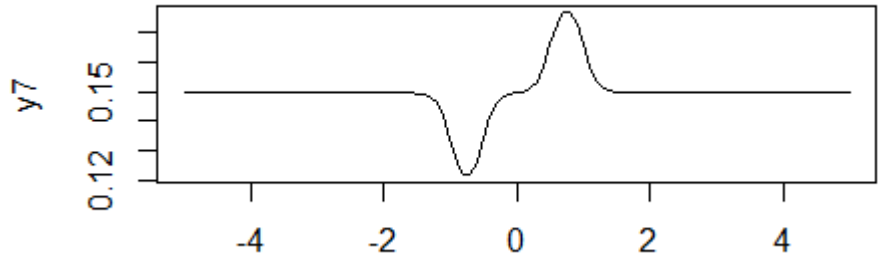
```
x=seq(-5,5,.1)
y1=1/(1+exp(10+   10*x))
y2=1/(1+exp(-5+(-10)*x))
y3=1/(1+exp(1+1*y1+1*y2))
plot(x,y3,type="l")
```




```
y4=1/(1+exp(10+ (-10)*x))
y5=1/(1+exp(-5+(10)*x))
y6=1/(1+exp(1-1*y4-1*y5))
plot(x,y6,type="l")
```

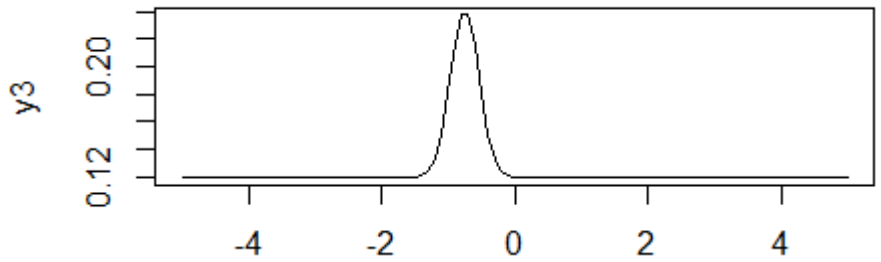



```
y7=1/(1+exp(1+2*y3+1*y6))
plot(x,y7,type="l")
```

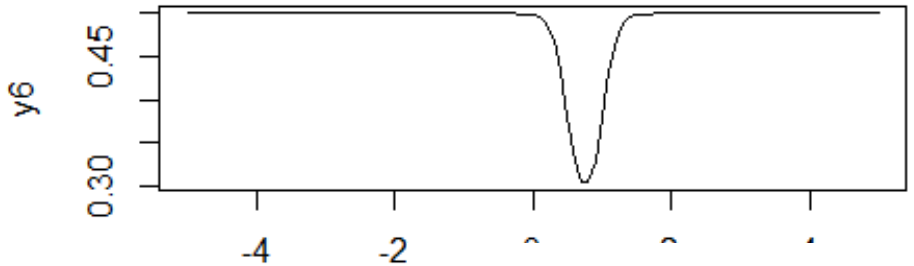


Higher level function combinations

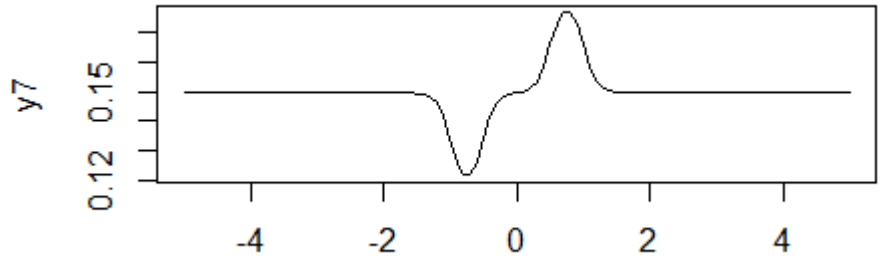
```
x=seq(-5,5,.1)
y1=1/(1+exp(10+   10*x))
y2=1/(1+exp(-5+(-10)*x))
y3=1/(1+exp(1+1*y1+1*y2))
plot(x,y3,type="l")
```




```
y4=1/(1+exp(10+ (-10)*x))
y5=1/(1+exp(-5+(10)*x))
y6=1/(1+exp(1-1*y4-1*y5))
plot(x,y6,type="l")
```

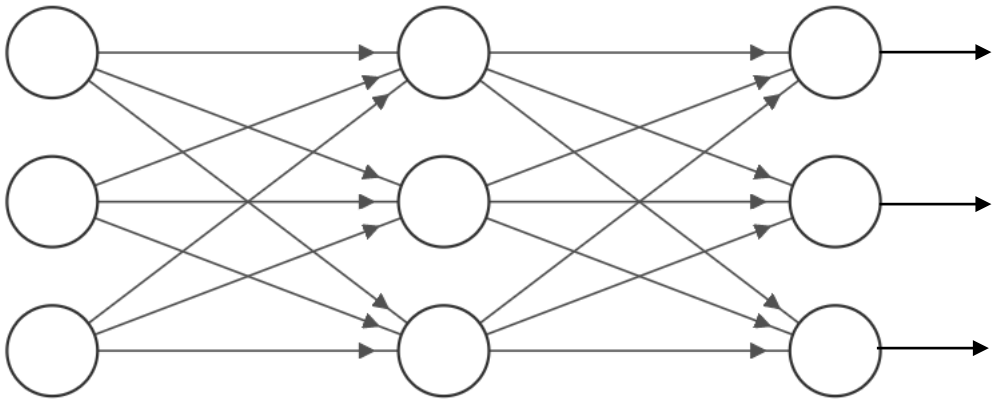



```
y7=1/(1+exp(1+2*y3+1*y6))
plot(x,y7,type="l")
```



Multiple layer networks can represent any logical or real-valued functions (unbiased, but potential to overfit)

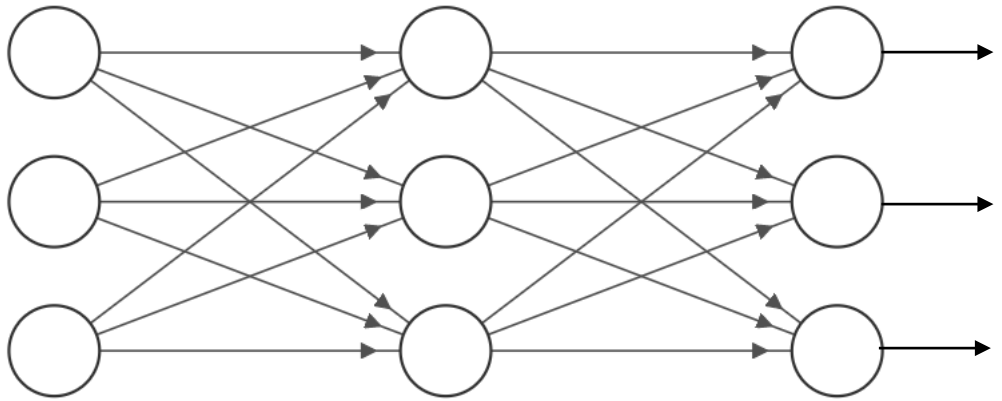
we can add layers and nodes



we can add layers and nodes

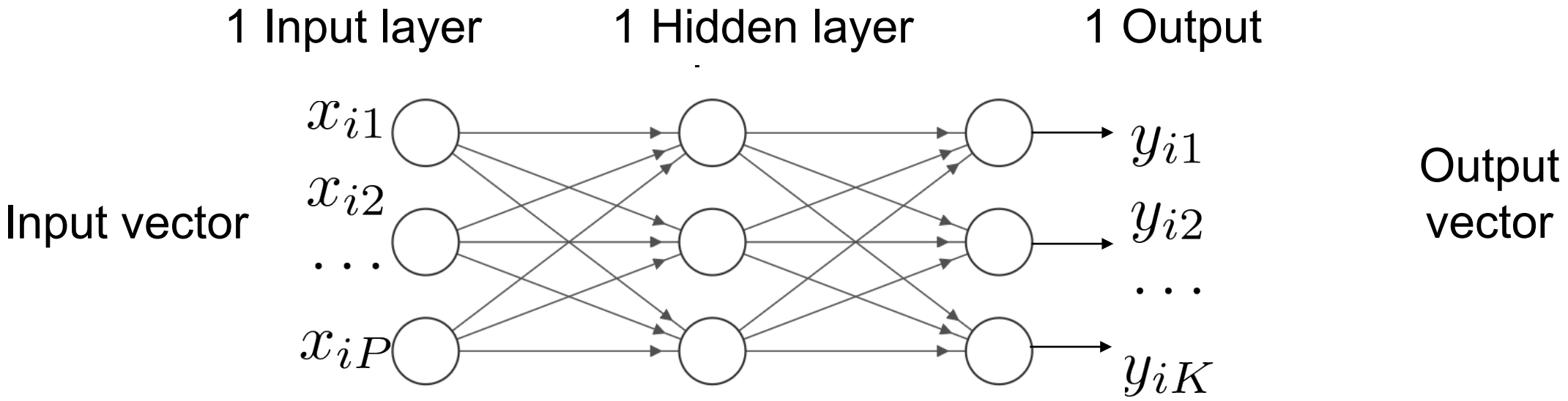
Multilayer Perceptron

1 Input layer 1 Hidden layer 1 Output



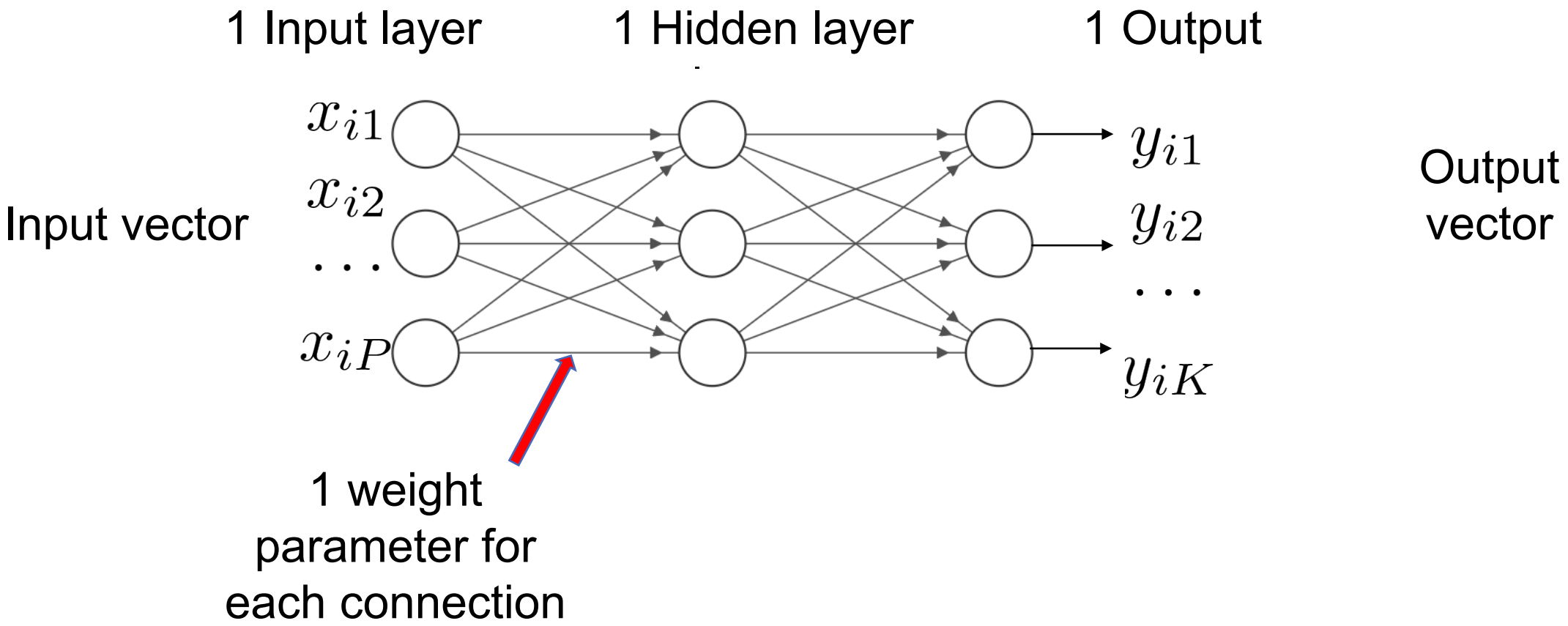
we can add layers and nodes

Multilayer Perceptron

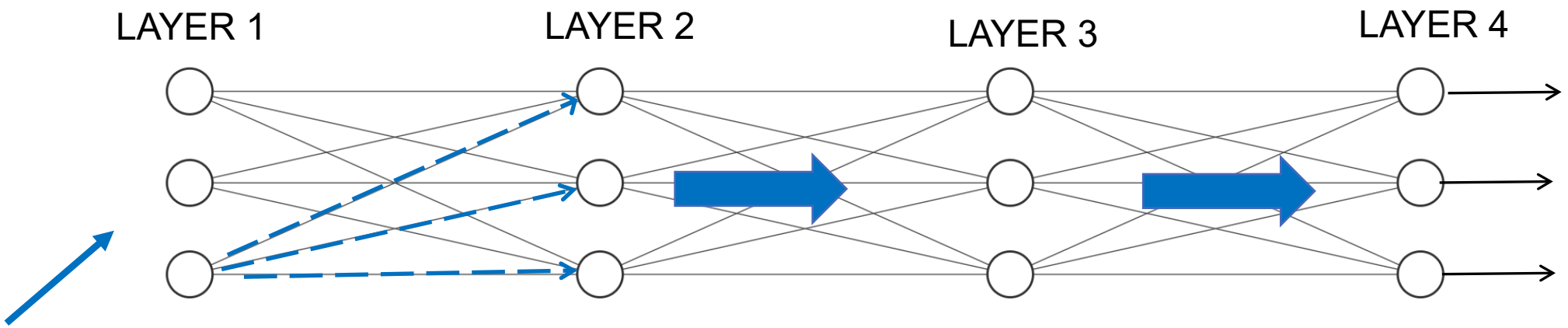


we can add layers and nodes

Multilayer Perceptron



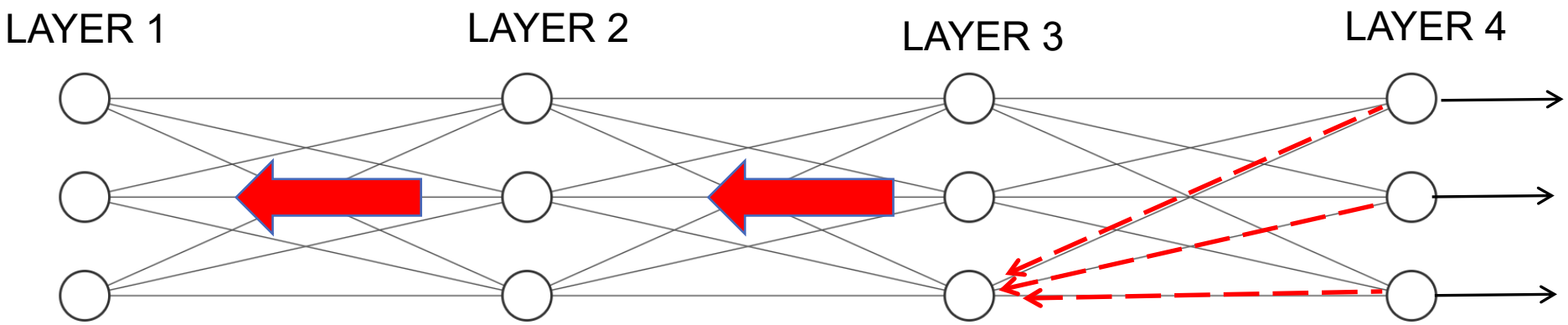
Algorithm steps:



1. **FORWARD PROPAGATE ACTIVATION**:

apply input data x_i ,
calculate all node activations

algorithm steps:



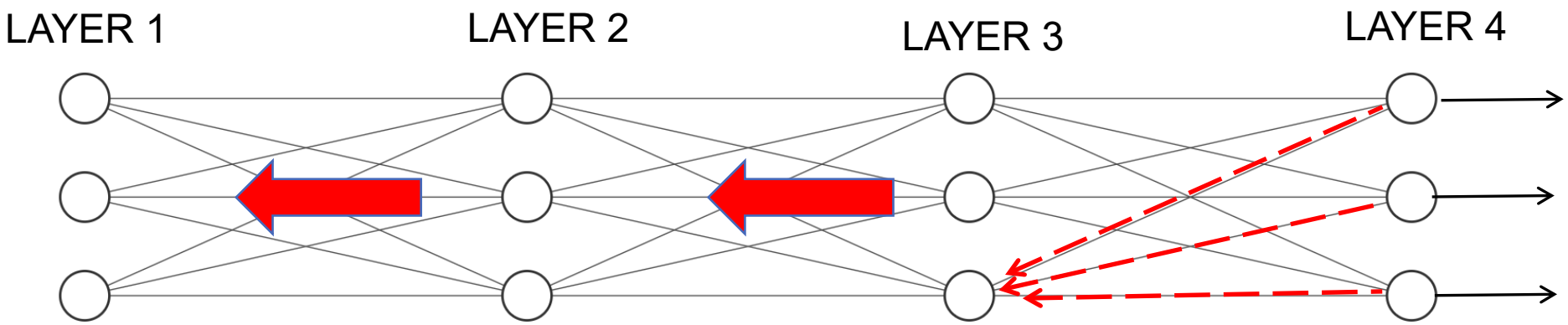
1. **FORWARD PROPAGATE ACTIVATION**:

apply input data x_i ,
calculate all node activations

2. BACKWARD PROPAGATE ERROR:

calculate error derivatives
pass it back to lower layer

algorithm steps:



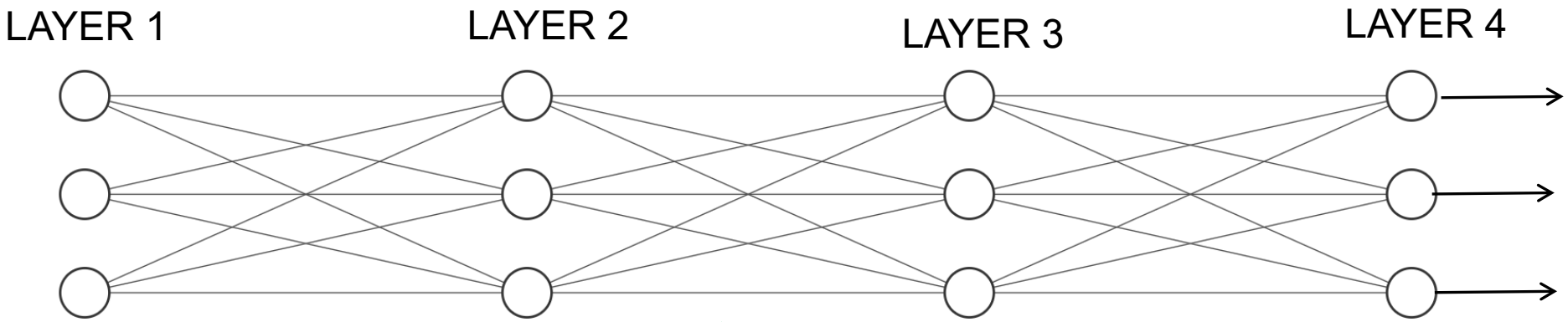
1. **FORWARD PROPAGATE ACTIVATION**:

apply input data x_i ,
calculate all node activations

2. BACKWARD PROPAGATE ERROR:
calculate error derivatives
pass it back to lower layer

Beware: error signals get diluted as you go backward - the 'vanishing gradient' problem

algorithm steps:



1. **FORWARD PROPAGATE ACTIVATION**:

apply input data x_i ,
calculate all node activations

2. BACKWARD PROPAGATE ERROR:
calculate Error (or Loss) derivatives
pass it back to lower layer

3. Update weights and bias terms

$$w_{ji} = w_{ji} - \eta \frac{dE}{dw_{ji}}$$

NN Algorithm

INITIALIZE WEIGHTS

LOOP until stopping criterion:

FORWARD PROPAGATION: calculate all node activations

BACKWARD PROPAGATION: calculate all derivatives to ***minimize Loss (dL)***

UPDATE WEIGHTS: $w \leftarrow w - learning\_rate * \frac{dL}{dw}$

STOP: when validation loss reaches minimum or converges

NN Algorithm
[heuristics, options to learn faster and/or better]

INITIALIZE WEIGHTS **[use truncated distributions]**

LOOP until stopping criterion: **[work in batches of input]**

FORWARD PROPAGATION: calculate all node activations

BACKWARD PROPAGATION: calculate all derivatives to ***minimize Loss (dL)***

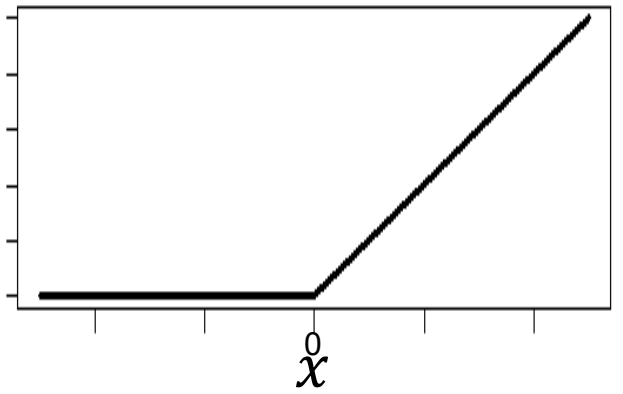
UPDATE WEIGHTS: $w \leftarrow w - learning\_rate * \frac{dL}{dw}$ **[adapt learning rate, use momentum]**

STOP: when validation loss reaches minimum or converges **[several metrics of loss are possible]**

A heuristic for deep networks

RELU activation function

RELU (rectified linear



It is unscaled (bad!)

But *df/da* is constant (good!)

$$f(a) = \begin{cases} a & a > 0 \\ 0 & a <= 0 \end{cases}$$

where $a = XW$

RELU helps mitigates vanishing gradients

terminology and cheat sheet on output activations:

Type of Problem	Y outputs	Output Activation Function	Output **PREDICTION** (what you decide to predict)	Output Loss Function	Evaluative Measure
Regression: map into to real valued prediction	if $Y \in (-\infty, +\infty)^K$	(linear) $\hat{Y} = XW$	$\hat{Y}$:	Sum Squared Error (SSE)	Root Mean Squared Error (RMSE)
Multivariate output of 0's and 1's (mulit-binary)	if $Y \in [0,1]^K$	(sigmoid) $\hat{Y} = \frac{1}{1+exp^{-(XW)}}$	1 or 0	SSE	RMSE
Binary Classification	if $Y \in \{0,1\}$	$\hat{Y} = \frac{1}{1+exp^{-(XW)}}$	A probability given by $\hat{Y}$: $P(y=1\|x)$	Cross Entropy $L = -ylog(\hat{y}) - (1-y)(log(\hat{y}))$	Accuracy, ROC
Multiclassification	if $Y \in \{0,1\}^K$	$\hat{Y_k} = \frac{exp^{-(XW_k)}}{\sum_k exp^{-(XW_k)}}$	(softmax) Max class	Cross Entropy $L = -\sum_k y_k log(\hat{y_k})$	Accuracy

Summary:

Pro:

- Neural Networks in general, are flexible, powerful learners
- Hidden layers learn a nonlinear transformation of input
- Many heuristics about what works

Summary:

Pro:

- Neural Networks in general, are flexible, powerful learners
- Hidden layers learn a nonlinear transformation of input
- Many heuristics about what works

Con:

- Hard to interpret
- Needs more data
- Lots of parameters

What is deep learning?

What is deep learning?

Deep learning refers to learning complex and varied transformations of the input

What is deep learning?

Deep learning refers to learning complex and varied transformations of the input

Deep learning refers to **discovering** useful features of the input

What is deep learning?

Deep learning refers to learning complex and varied transformations of the input

Deep learning refers to **discovering** useful features of the input

Deep learning is a neural network with many layers

pause

onto Convolution Networks

Image features

- **MNIST - A database of handwritten printed digits**

  (National Inst. of Standards and Technology)

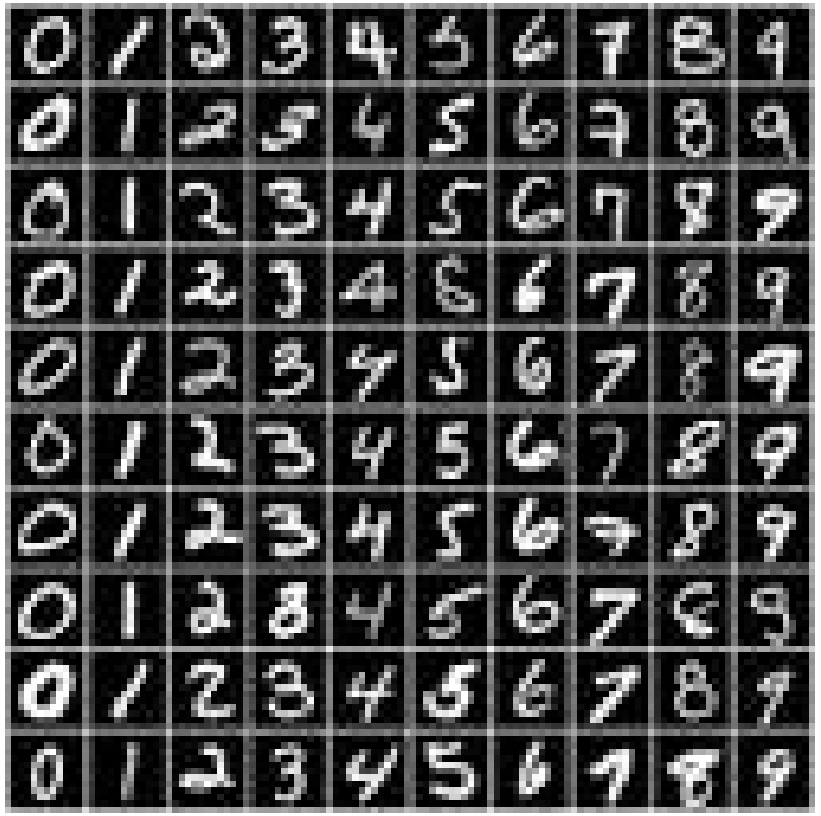


Image features

- **MNIST - A database of handwritten printed digits**

  (National Inst. of Standards and Technology)

How to classify digits?

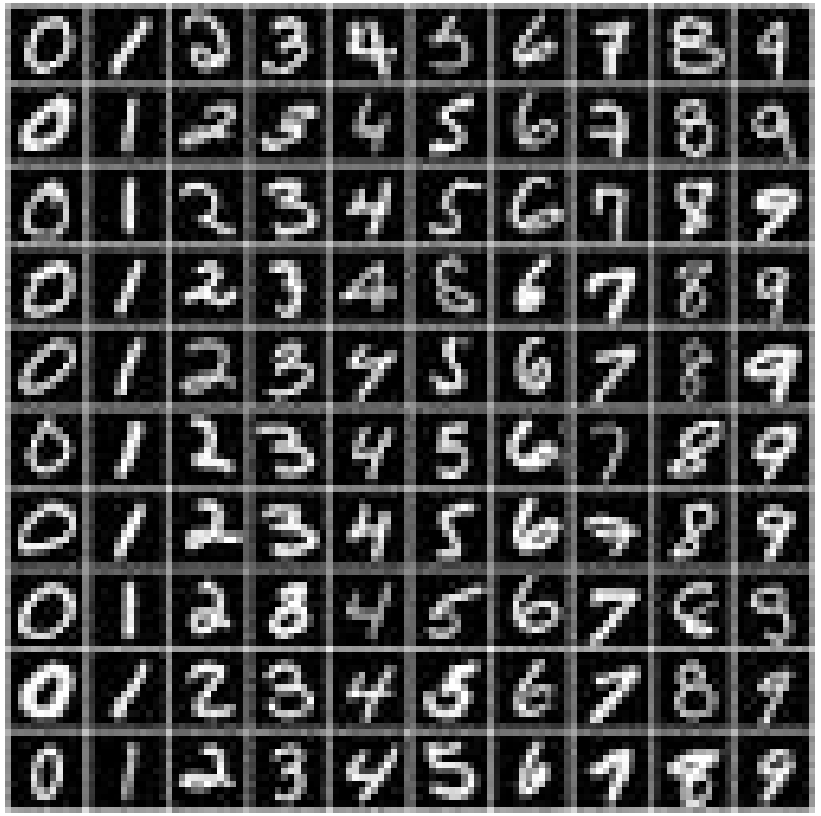
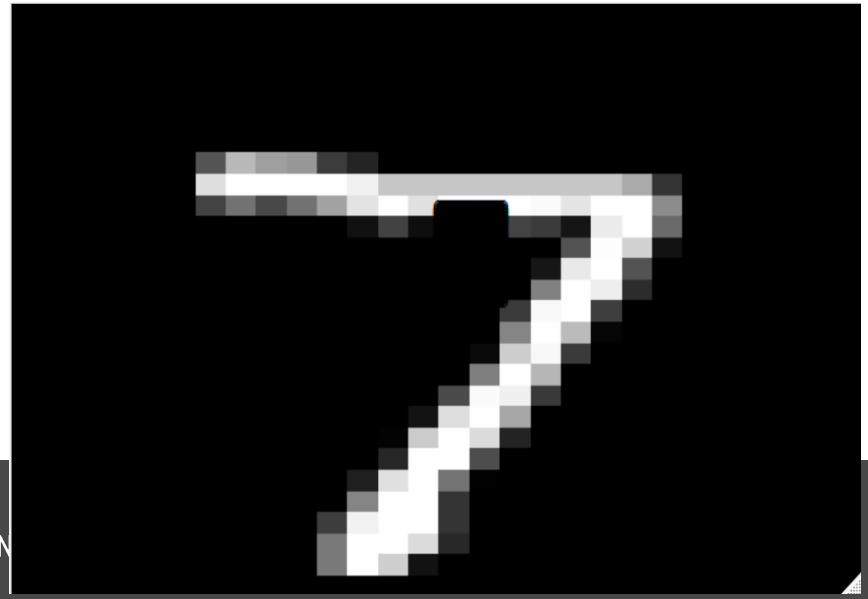


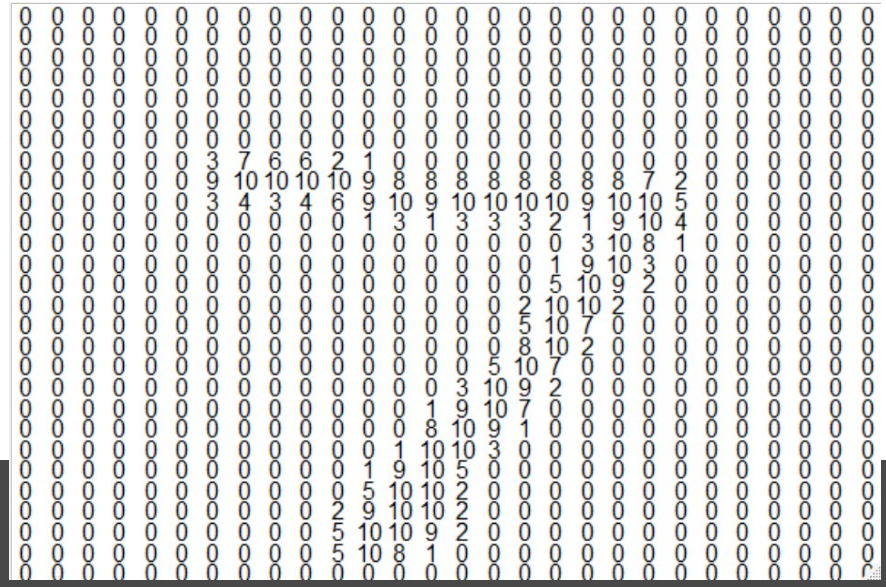
Image features

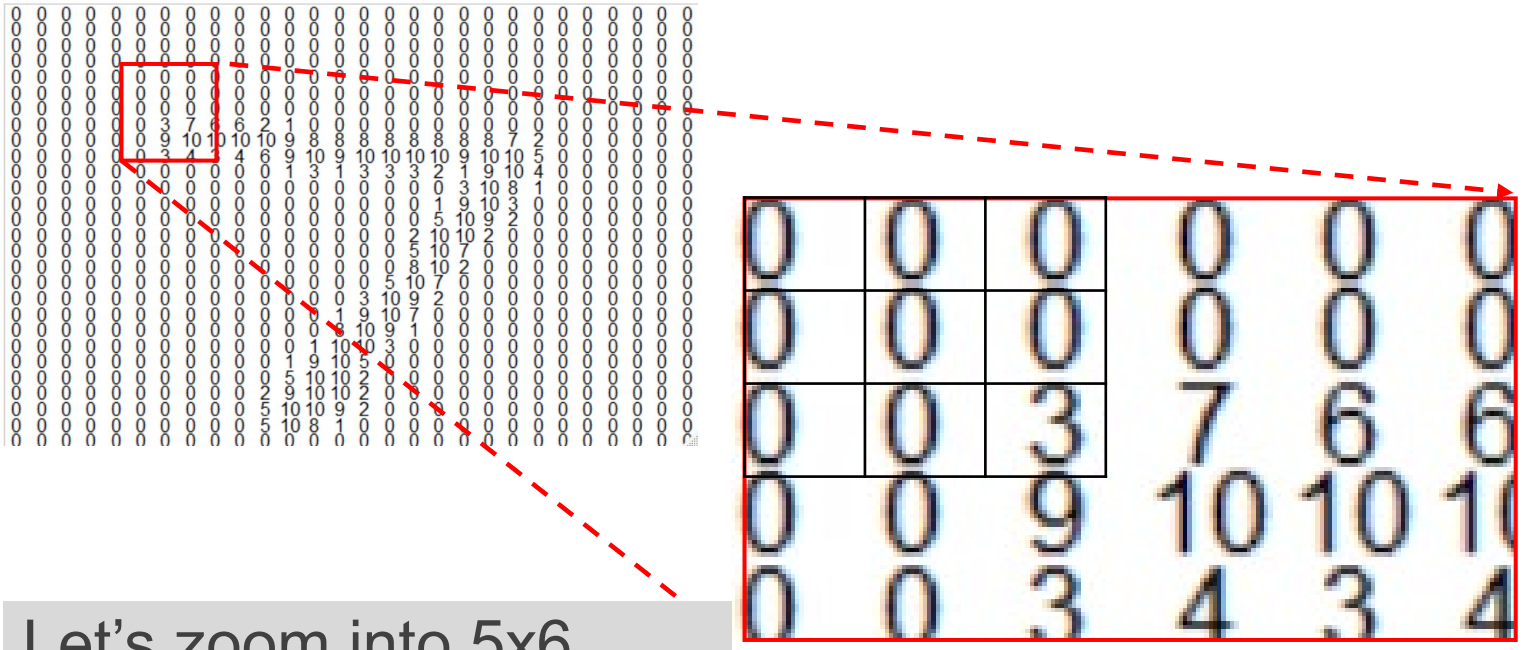
- **MNIST - A database of handwritten printed digits**

  (National Inst. of Standards and Technology)

How to classify digits?

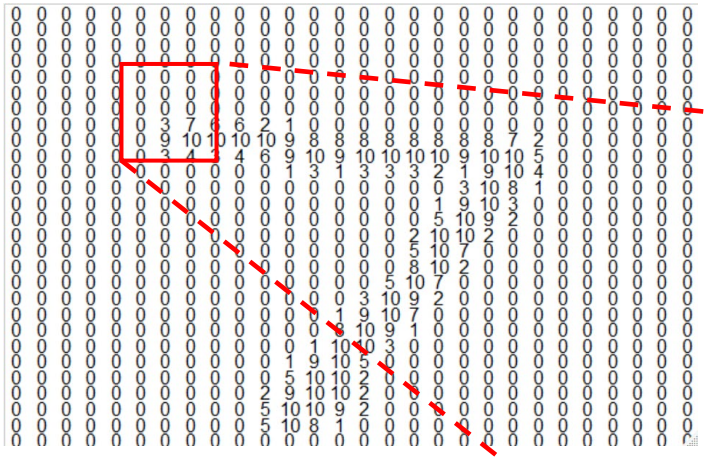


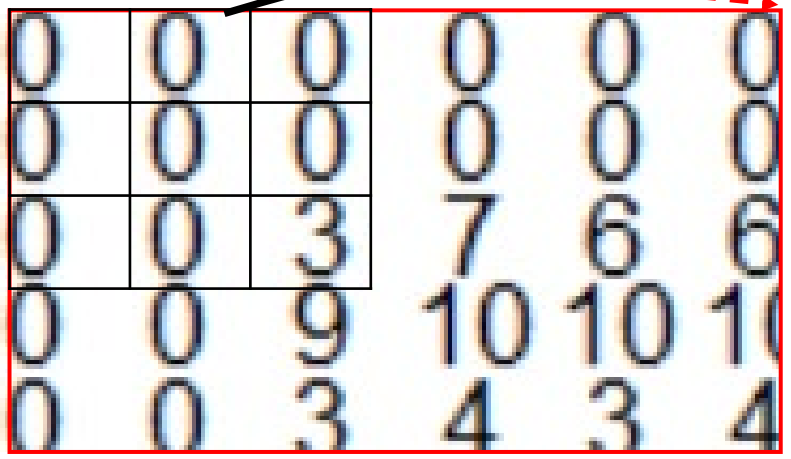




Let’s zoom into 5x6 window of pixels near the tip of ‘7’

Take a 3x3 patch of pixels and apply a ‘filter’ template – designed to find an edge





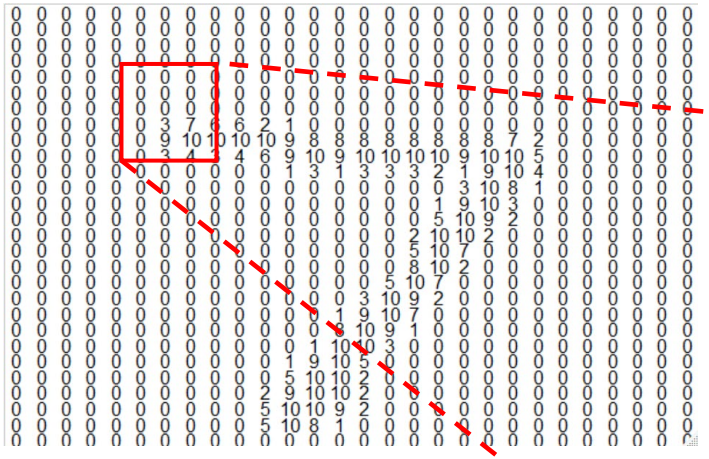
-1	0	+1
-1	0	+1
-1	0	+1

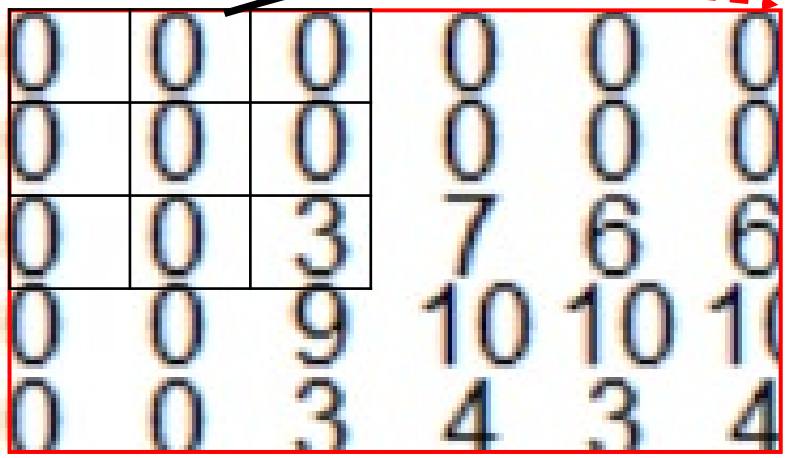
X

1. Multiply 3x3 patch of pixels with 3x3 filter

Let's zoom into 5x6 window of pixels near the tip of '7'

Take a 3x3 patch of pixels and apply a 'filter' template – designed to find an edge





(our weight parameters)

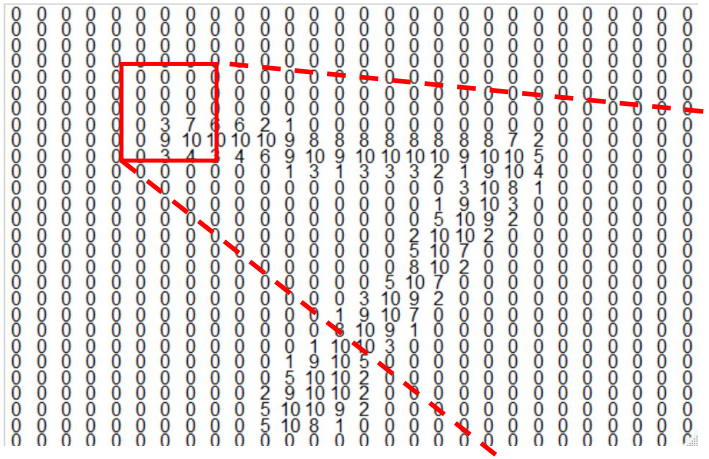
-1	0	+1
-1	0	+1
-1	0	+1

X

1. Multiply 3x3 patch of pixels with 3x3 filter “**W**”

Let’s zoom into 5x6 window of pixels near the tip of ‘7’

Take a 3x3 patch of pixels and apply a ‘filter’ template – designed to find an edge



0	0	0
0	0	0
0	0	3

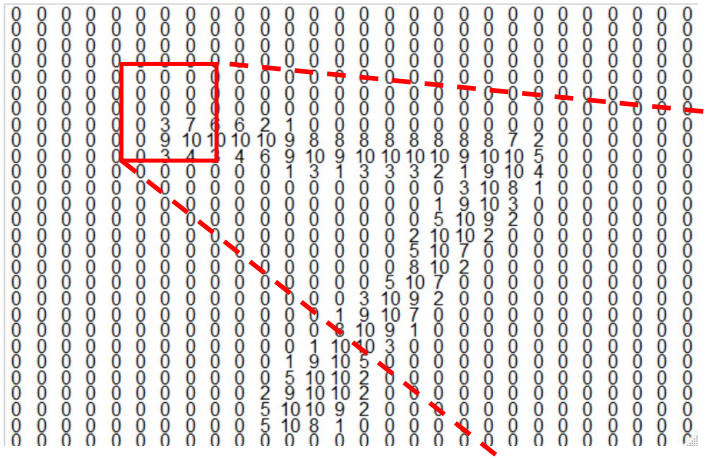
X

-1	0	+1
-1	0	+1
-1	0	+1

1. Multiply 3x3 patch of pixels with 3x3 filter "**W**"

2. Put answer in new cell of output map

3			



0	0	0
0	0	0
0	0	3

0	0	0
0	0	0
7	6	6

0	0	0
0	0	0
9	10	10

0	0	0
0	0	0
3	4	3

X

-1	0	+1
-1	0	+1
-1	0	+1

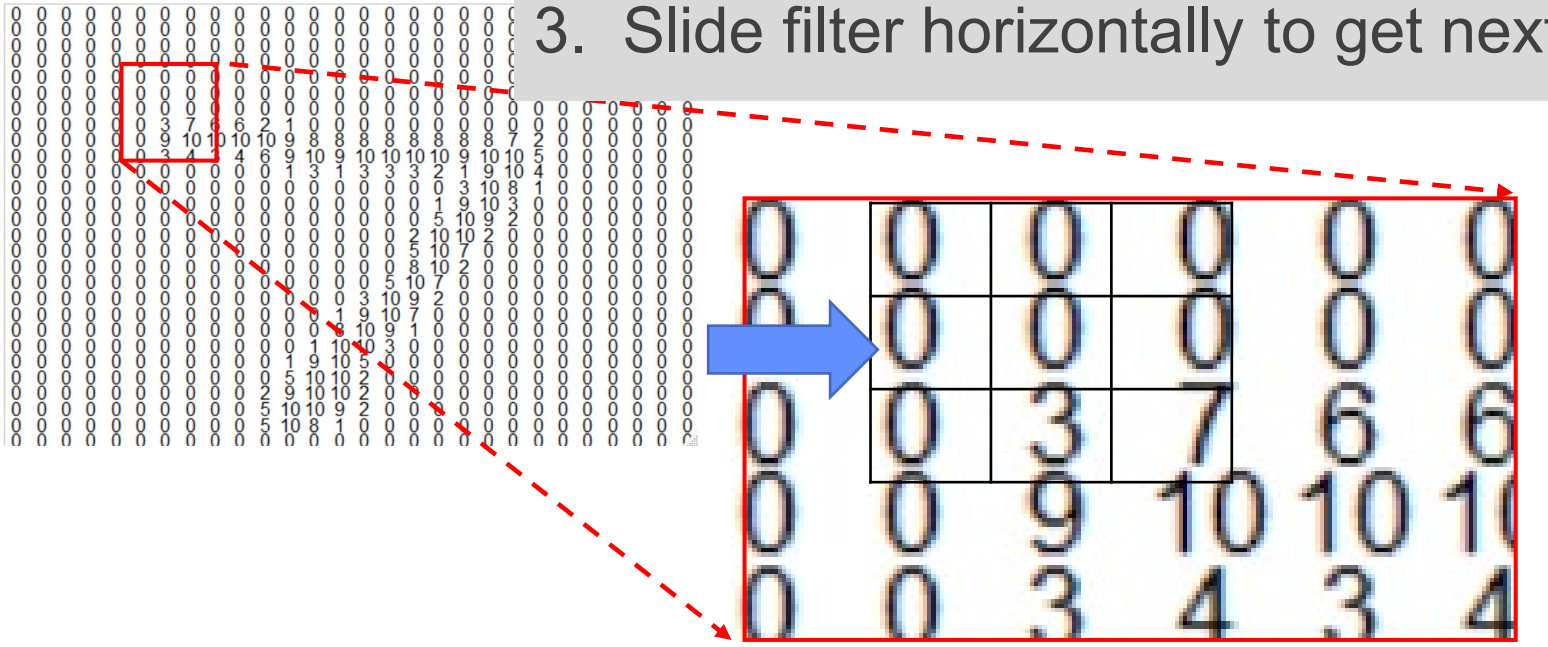
1. Multiply 3x3 patch of pixels with 3x3 filter "**W**"

This is "image window"∗W

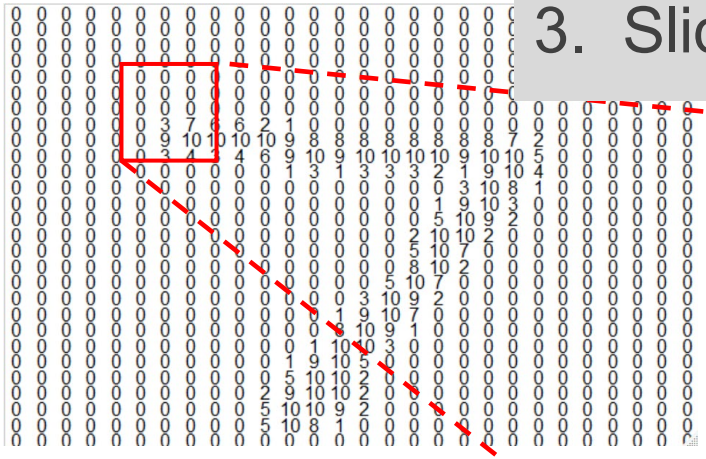
3			

2. Put answer in new cell of output map

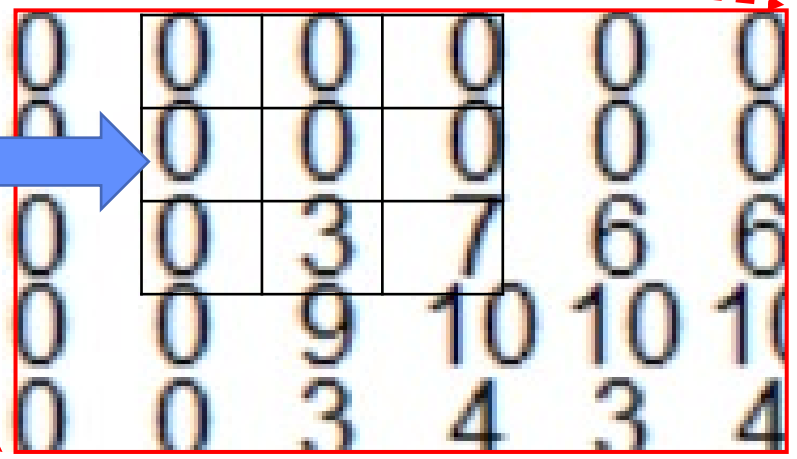
3. Slide filter horizontally to get next output value



3	7		



3. Slide filter horizontally to get next output value



-1	0	+1
-1	0	+1
-1	0	+1

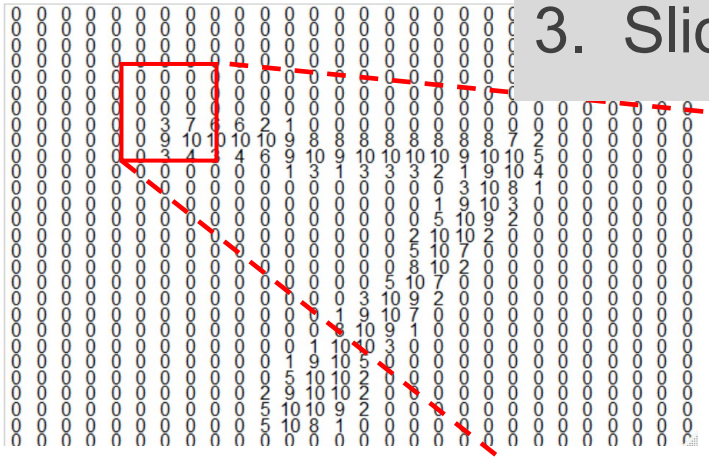
X

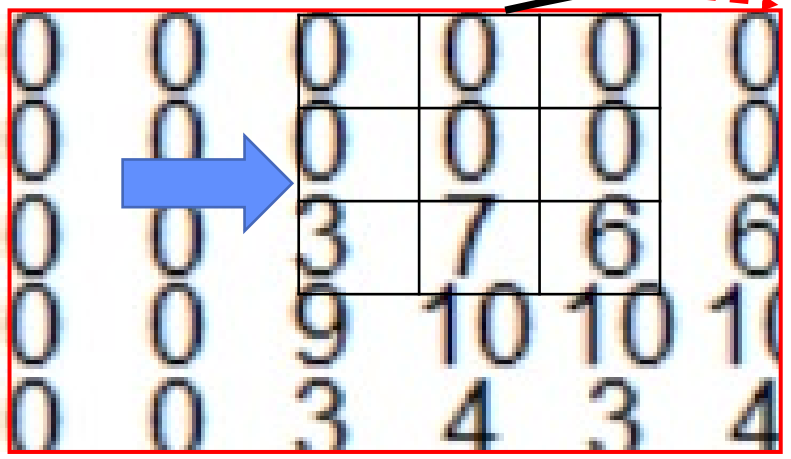
1. Multiply 3x3 patch of pixels with 3x3 filter "**W**"

2. Put answer in new cell of output map

3	7		

3. Slide filter horizontally to get next output value

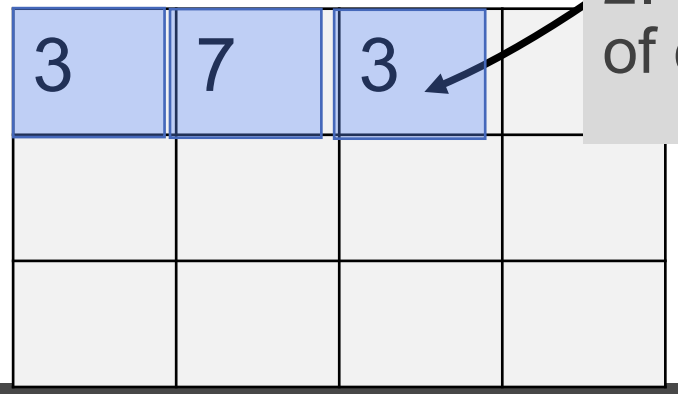




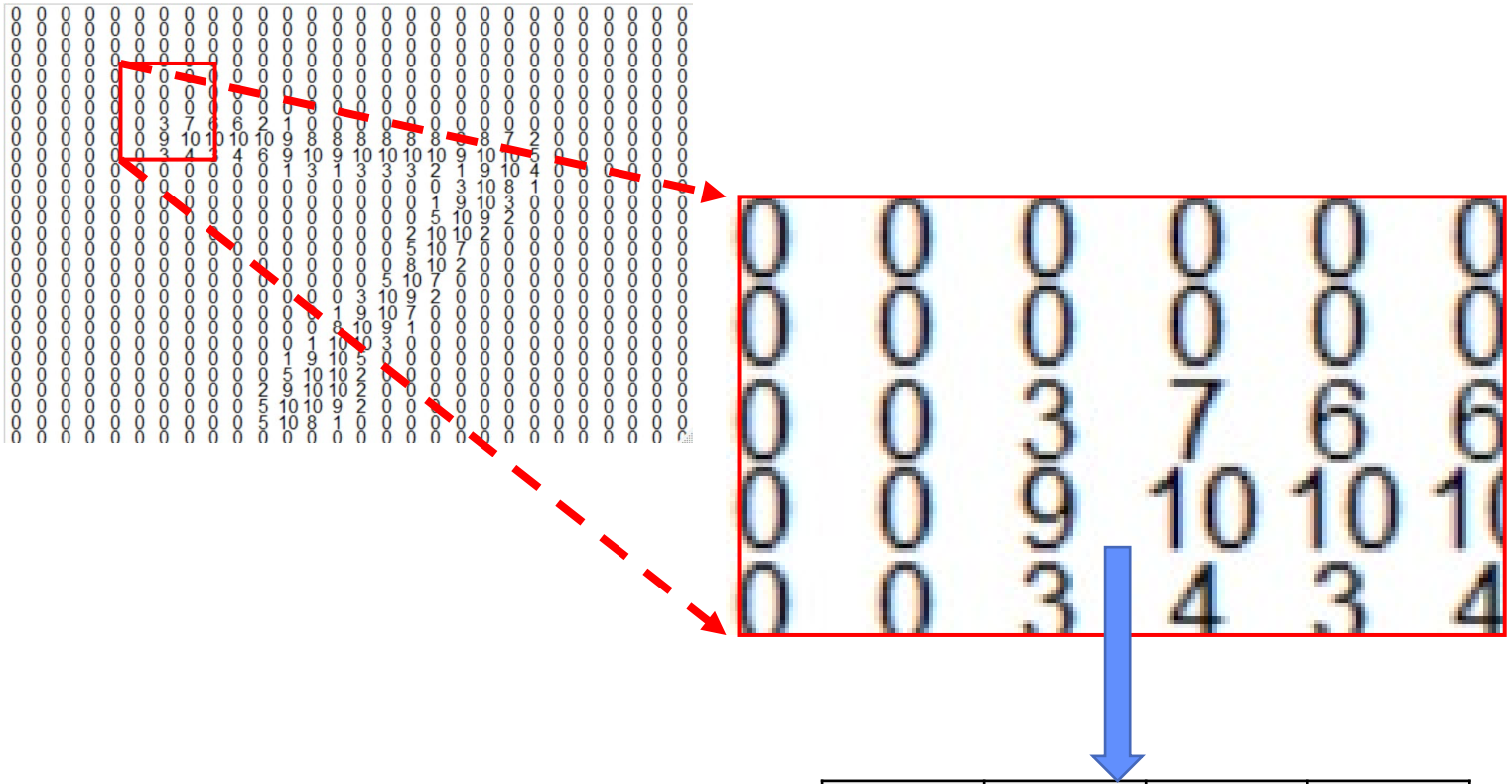
-1	0	+1
-1	0	+1
-1	0	+1

1. Multiply 3x3 patch of pixels with 3x3 filter "**W**"

2. Put answer in new cell of output map

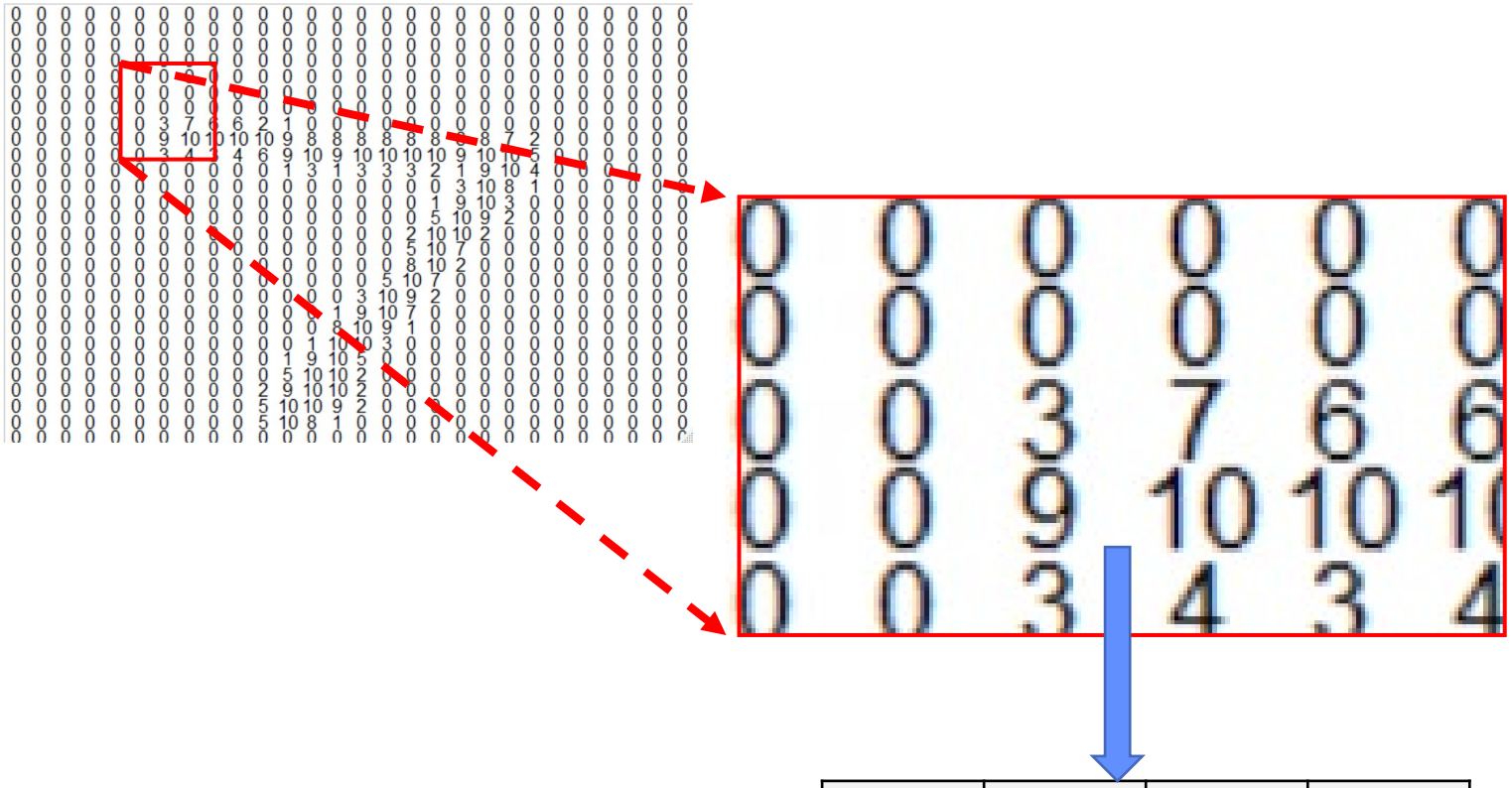


NOTE: sliding a filter is known as a "convolution" operation



After vertical and horizontal sliding the 5x6 patch is now a 3x5 ***feature map.***

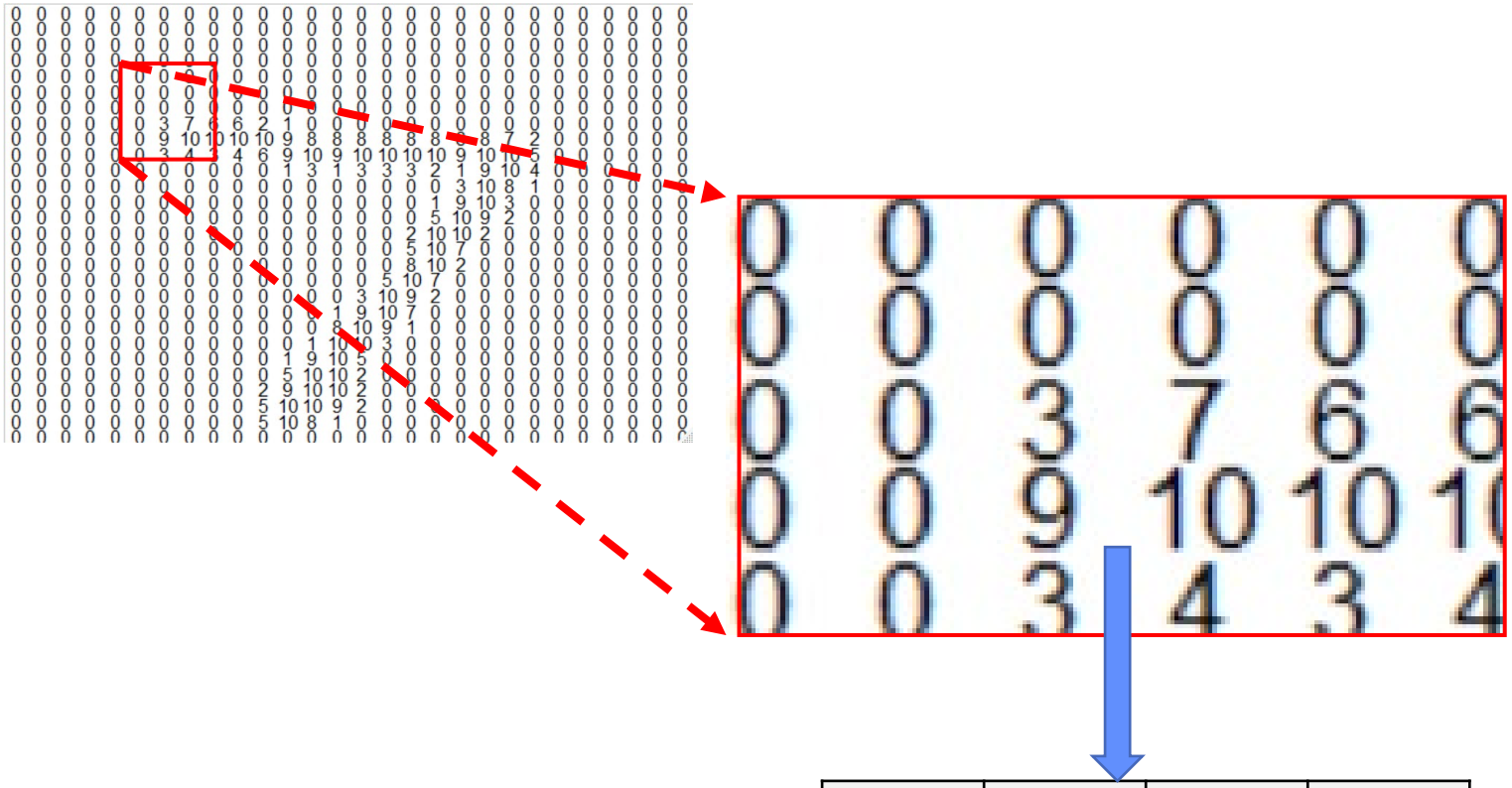
3	7	3	-1
12	17	4	-1
15	21	4	-1



3	7	3	-1
12	17	4	-1
15	21	4	-1

After vertical and horizontal sliding the 5x6 patch is now a 3x5 ***feature map.***

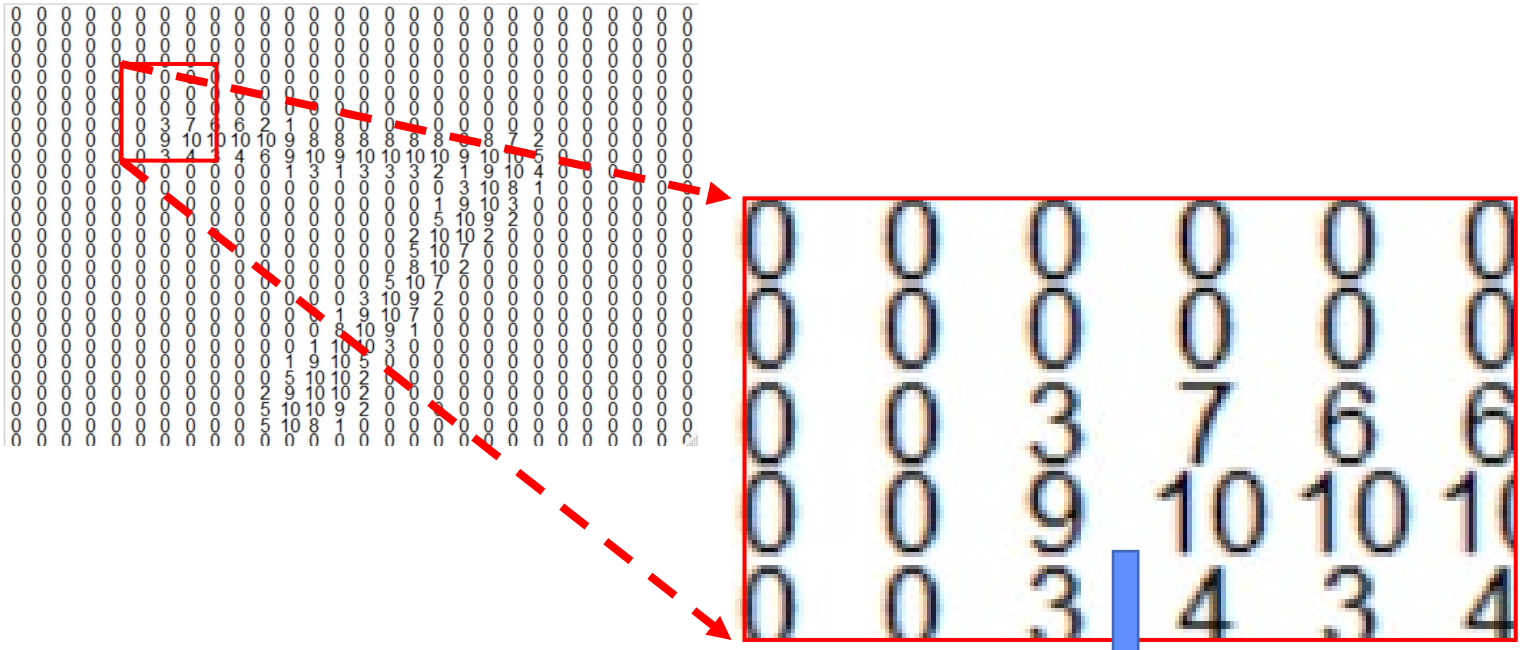
What do the highest values in the feature map represent?



3	7	3	-1
12	17	4	-1
15	21	4	-1

Optional next step:

Use another filter, and take maximum over elements - “max pooling”



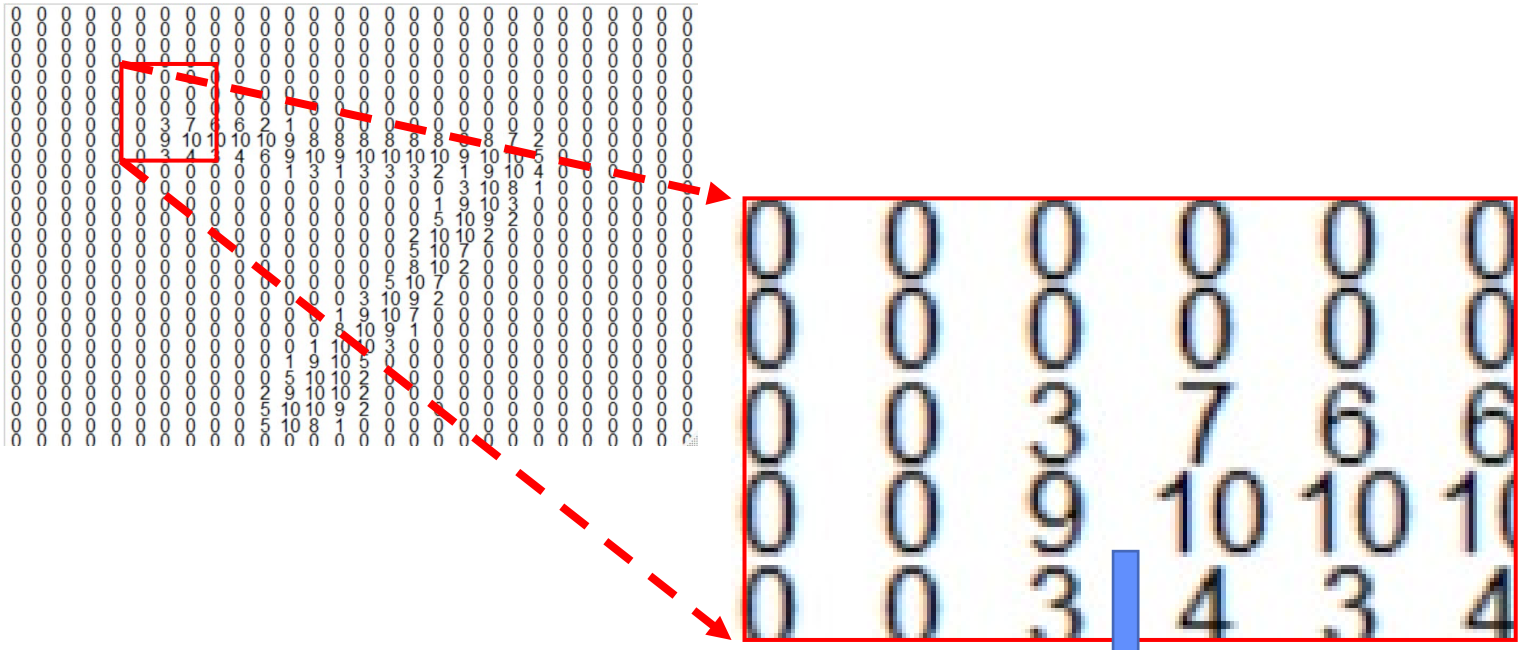
3	7	3	-1
12	17	4	1
15	21	4	-1

Optional next step:

Use another filter, and take maximum over elements - “max pooling”

2x2 filter has max=17

17		



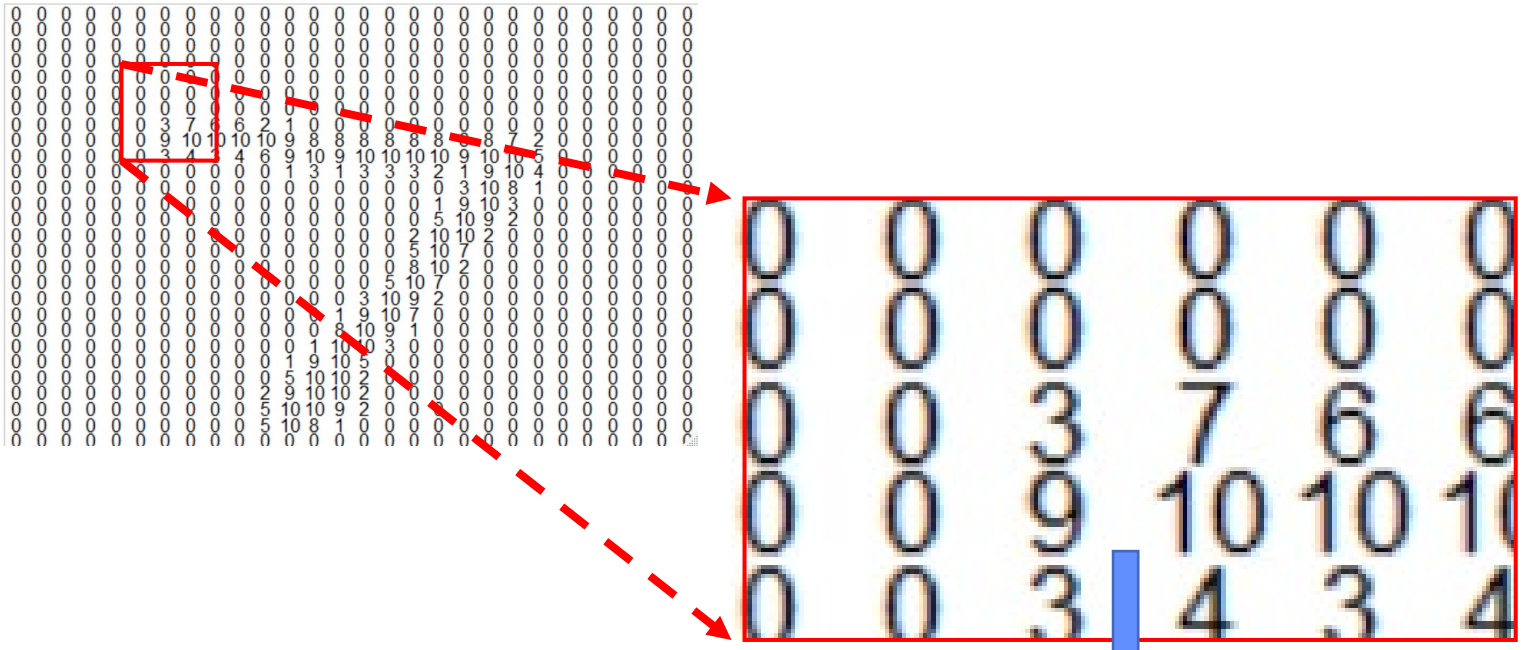
3	7	3	-1
12	17	4	-1
15	21	4	-1

Optional next step:

Use another filter, and take maximum over elements - “max pooling”

Slide filter …

17	17	4
21	21	4



*After convolution and pooling 5x6 patch is **transformed** into a 2x3 feature map of 'edge gradients'*

3	7	3	-1
12	17	4	-1
15	21	4	-1

Slide filter …

17	17	4
21	21	4

Feature engineering

In Computer Vision there are many kinds of edge detectors and many ways to scale them

-1	0	+1
-1	0	+1
-1	0	+1

But building features is hard, so if you have enough data …

Convolution Neural Network (CNN)

In CNNs the filter values are weight parameters that are learned (**feature discovery**)

W_{11}	W_{12}	W_{13}
W_{21}	W_{22}	W_{23}
W_{31}	W_{32}	W_{33}

Convolution Neural Network (CNN)

In CNNs the filter values are weight parameters that are learned (**feature discovery**)

W_{11}	W_{12}	W_{13}
W_{21}	W_{22}	W_{23}
W_{31}	W_{32}	W_{33}

A convolution layer is a set of feature maps, where each map is derived from convolution of 1 filter with input

Convolution Neural Network (CNN)

More hyperparameters:

Size of filter (smaller is more general)

Convolution Neural Network (CNN)

More hyperparameters:

Size of filter (smaller is more general)

Number of pixels to slide over (1 or 2 is usually fine)

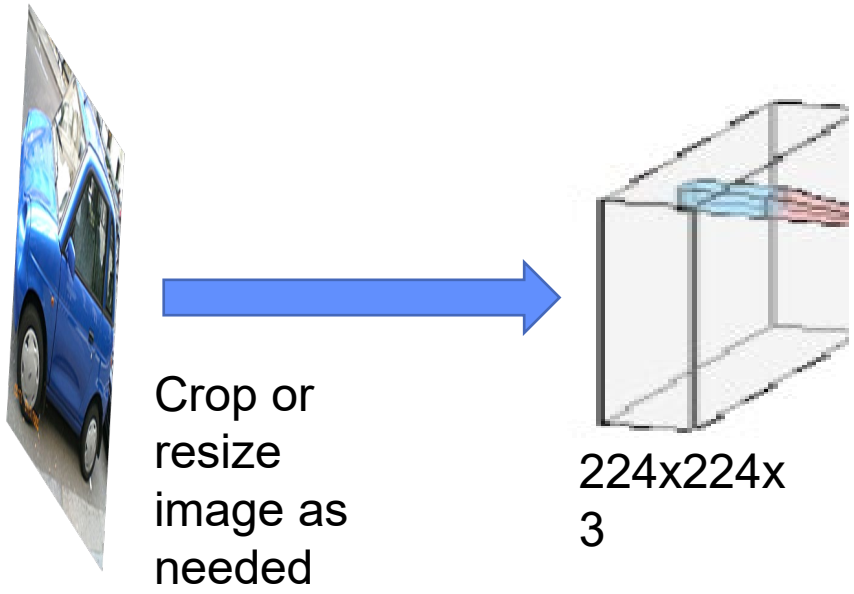
Convolution Neural Network (CNN)

More hyperparameters:

- Size of filter (smaller is more general)
- Number of pixels to slide over (1 or 2 is usually fine)
- Number of filters (depends on the problem!)
- Max pooling or not (usually some pooling layers)

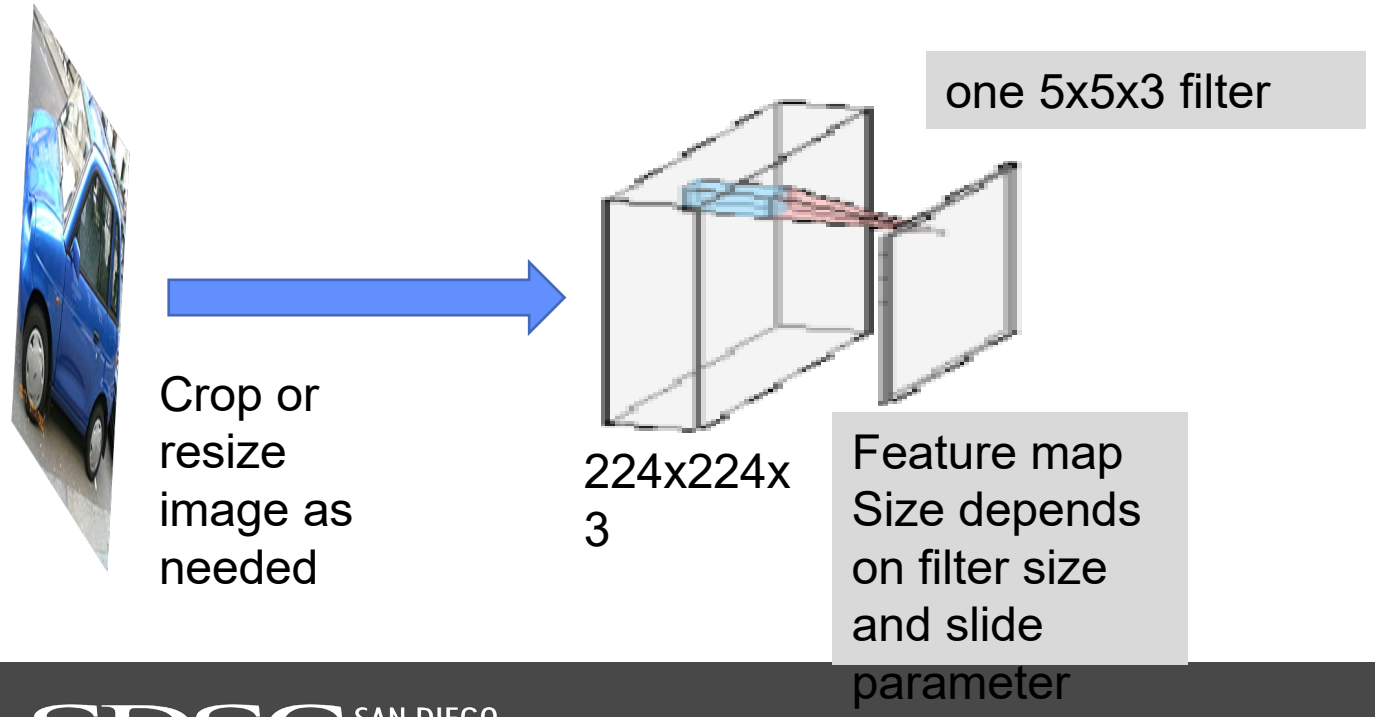
Convolution with image

- **Make 1 layer, using HxWx3 image (3 for Red,Green,Blue channels)**



Convolution with image

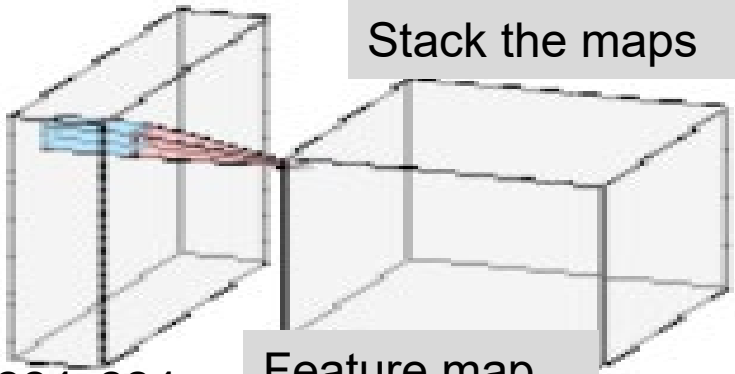
- **Make 1 layer, using HxWx3 image (3 for RGB channels)**



Convolution with image

- **Make 1 layer, using HxWx3 image (3 for RGB channels)**

Crop or resize image as needed



224x224x 3

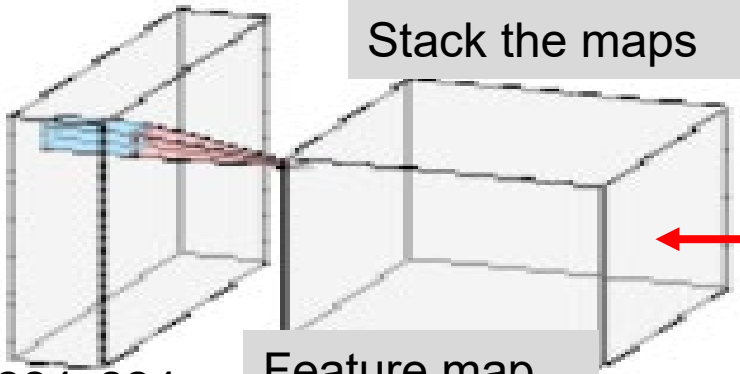
Repeat for many filters, all 5x5x3.
Each filter leads to 1 feature map
Stack the maps

Feature map Size depends on filter size and slide parameter

Convolution with image

- **Make 1 layer, using HxWx3 image (3 for RGB channels)**

Crop or resize image as needed



224x224x 3

Repeat for many filters, all 5x5x3.
Each filter leads to 1 feature map
Stack the maps

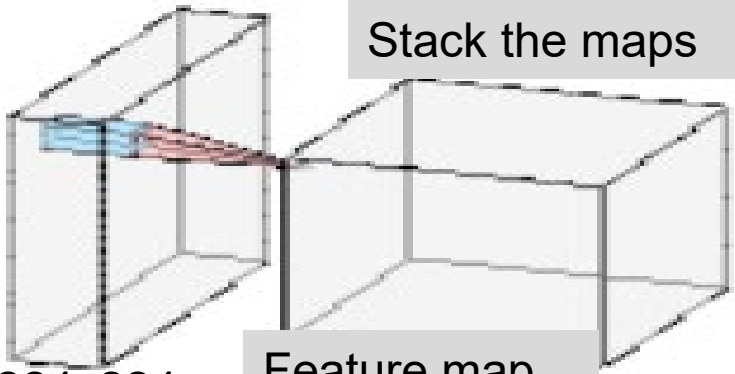
Feature map Size depends on filter size and slide parameter

This is a "convolution layer"
The thickness is the number of different feature maps, sometimes called 'channels' or 'number of filters

Convolution with image

- **Make 1 layer, using HxWx3 image (3 for RGB channels)**

Crop or resize image as needed



224x224x 3

Repeat for many filters, all 5x5x3.
Each filter leads to 1 feature map
Stack the maps

Feature map Size depends on filter size and slide parameter

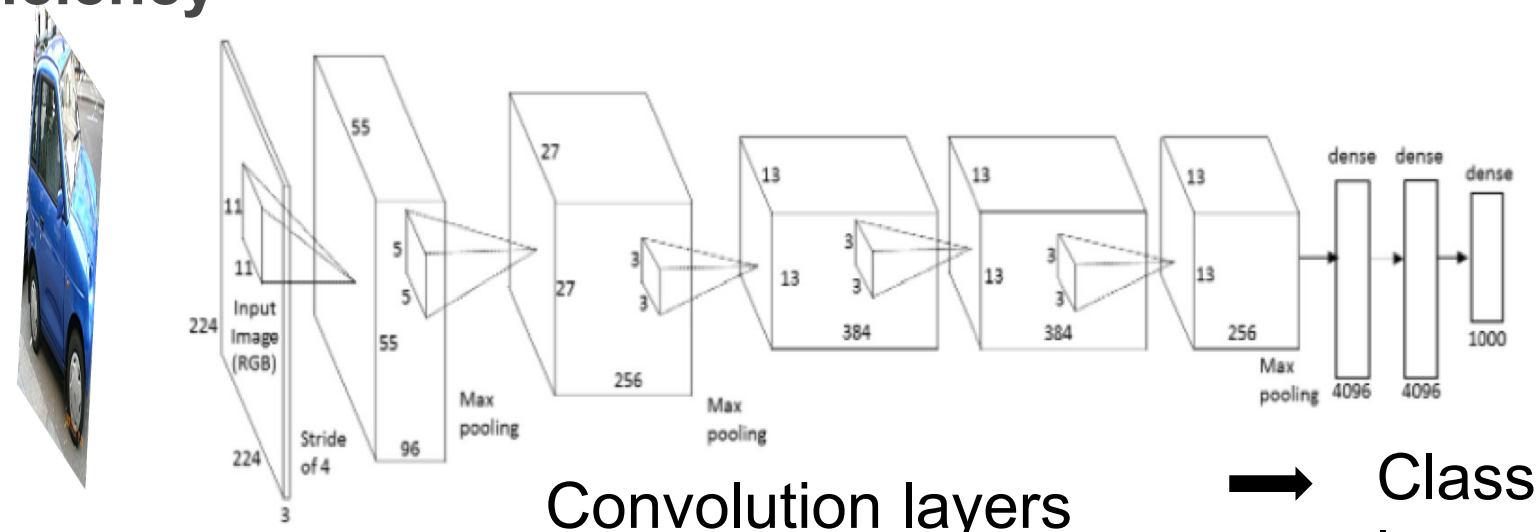
Add a layer of max-pool to down sample each feature map

OR

Add a next convolution layer

Large Scale Versions

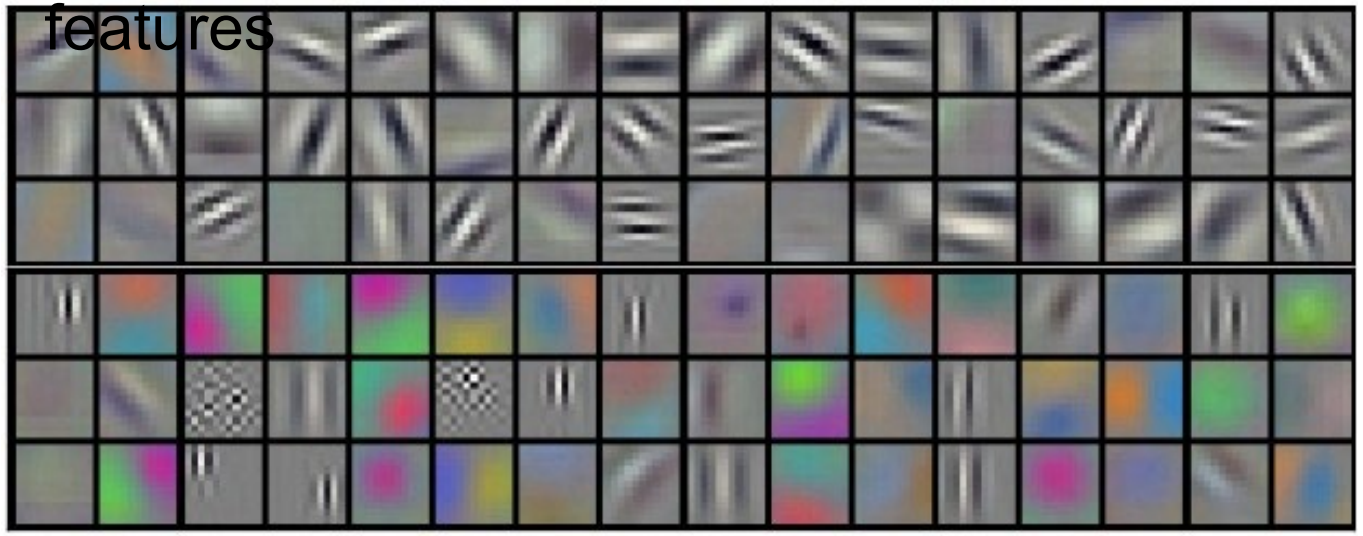
- Large (deep) Convolution Networks are turning out to be feasible with GPUs (some are 100+ layers)
- Need large amounts of data and many heuristics to avoid overfitting and increase efficiency



Convolution layers → Classification layers and output

What Learned Convolutions Look Like

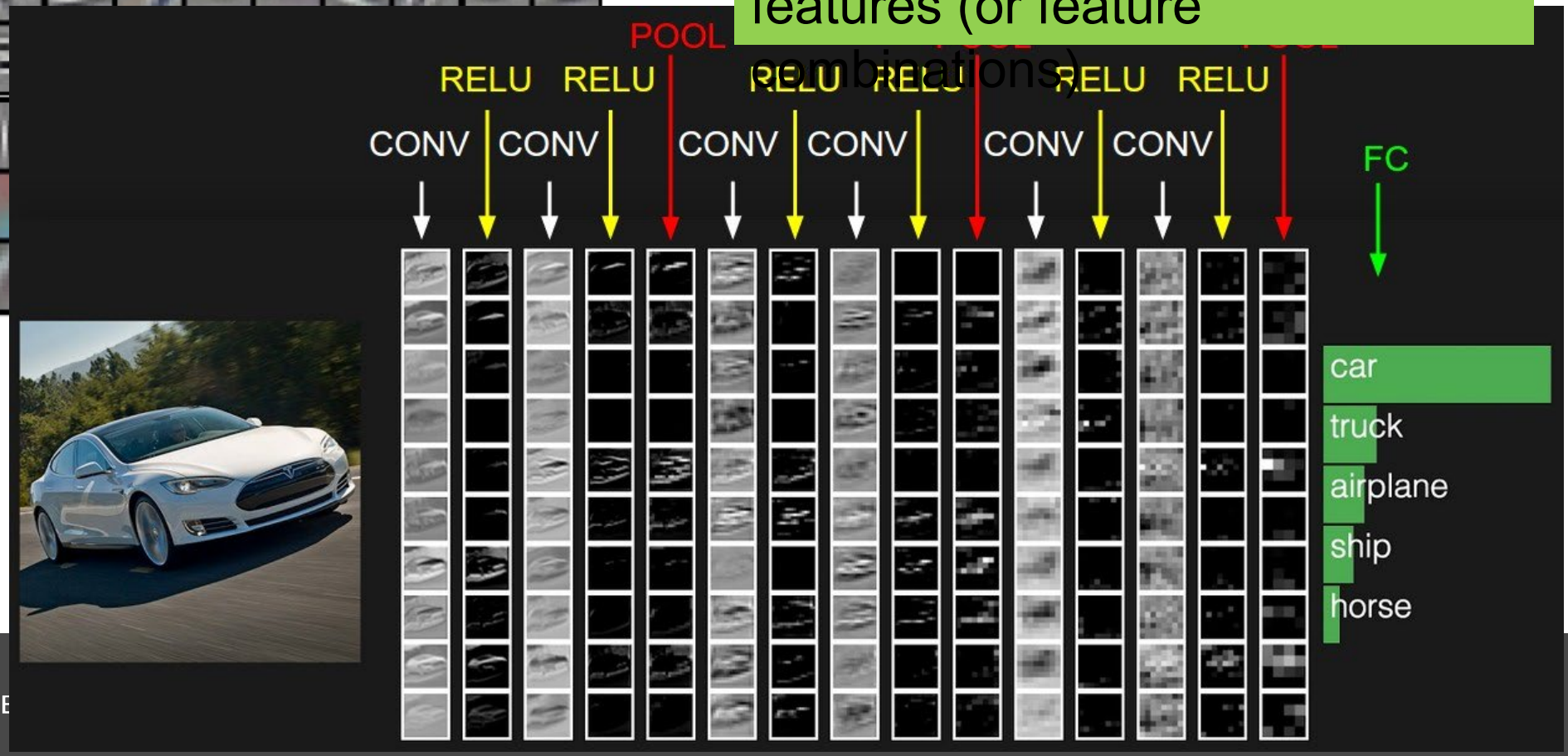
First convolution layer filters are simple features



What Learned Convolutions

First convolution layer filters are simple features

Higher layers are more abstract features (or feature combinations)



Convolution Neural Network Summary

CNNs works because convolution layers have a special architecture and function – it is biased to do certain kind of transformations

Low layers have less filters that represent simple local features for all classes

Higher layers have more filters that cover large regions that represent object class features