COSC 325: Introduction to Machine Learning

Dr. Hector Santos-Villalobos

THE UNIVERSITY OF TENNESSEE
KNOXVILLE

Lecture 21: Artificial Neural Networks and Deep Learning

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Class Announcements

Homework
Done with all the homework!!!!!!!

Course Project:

- Amy Huang’s tip:
  - Hodges Library Studio, $3-$6, 2 BD
  - Ucopy, $15, 2 BD
- Course Project Presentation Poster Logistics
  - Please arrive early!

Quizzes:
No quiz this week.

Exams:
Exam #2 this **Thursday**, 11/21—online format.

Lectures:

- **Panel on Ethical AI 11/26**. You will get attendance points by posting a question in the Discord ***#panel-on-ethical-ai*** channel (https://discord.com/channels/1263144544082596050/1306342338926346260)

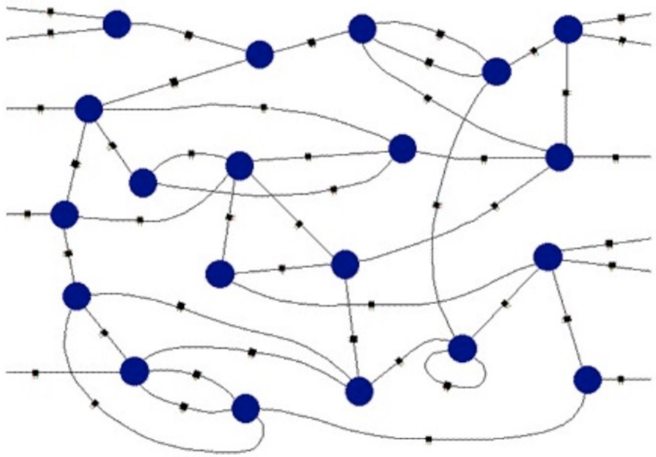
TN Voice Open!

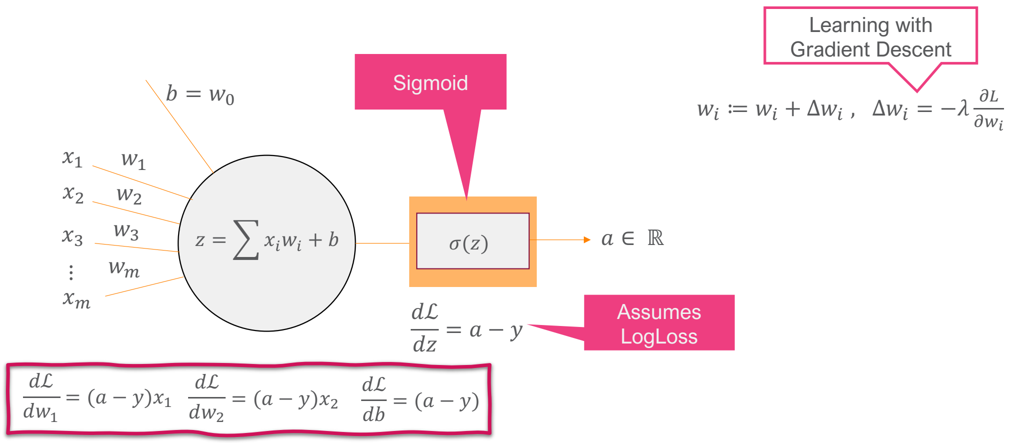
Course Project Feedback

- There is mention of issues but no mention of the mechanisms to address issues
  - E.g., missing values, outliers, etc.
- Data preprocessing steps missing
- Report depends on Jupyter notebook.
  - Report needs to be self-contained.
- EDA that provides insights about your problem and solution
  - E.g., data shape and normalization technique
- Backup claims with actual numbers or visualizations
- No clear definition of what the model should do
  - E.g., Stocks, down/upward trend prediction vs stock price regressor
- No distribution of work.
- No mention of ML technique in intro.
- Plots without legend or axis titles
- Multiple ML techniques without proper comparison. (k-fold, CIs)

Review

- ANNs
  - Hebb's Law: "Neurons that fire together wire together."
    - Connectionist Machines
  - Differentiable networks
    - We can update parameters with Gradient Descent
  - Layer weight matrix dimensions $(m^{[l-1]}, m^{[l]})$
  - Number of parameters in a layer is the number of weights and biases or $m^{[l-1]} \times m^{[l]} + m^{[l]}$

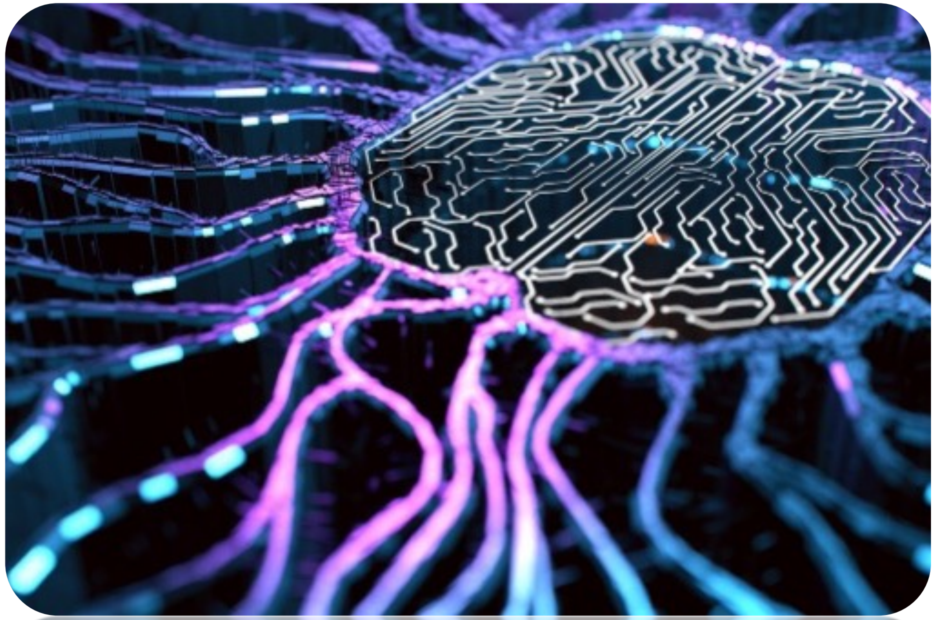




Today’s Topics

Artificial Neural Networks

*Deep Learning**



Pop Quiz

Go to Discord **panel-on-ethical-ai** channel and enter a question about AI.
Examples:

- Career in AI/ML
- Ethics in AI
- Concerns about Artificial General Intelligence
- Curiosity about a particular application

https://discord.com/channels/1263144544082596050/1306342338926346260

Backpropagation Algorithm

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Backpropagation Algorithm

1960s

Introduced in the1960s

1970's

Modern technique invented in the 1970's by Seppo Linnainmaa*

1989

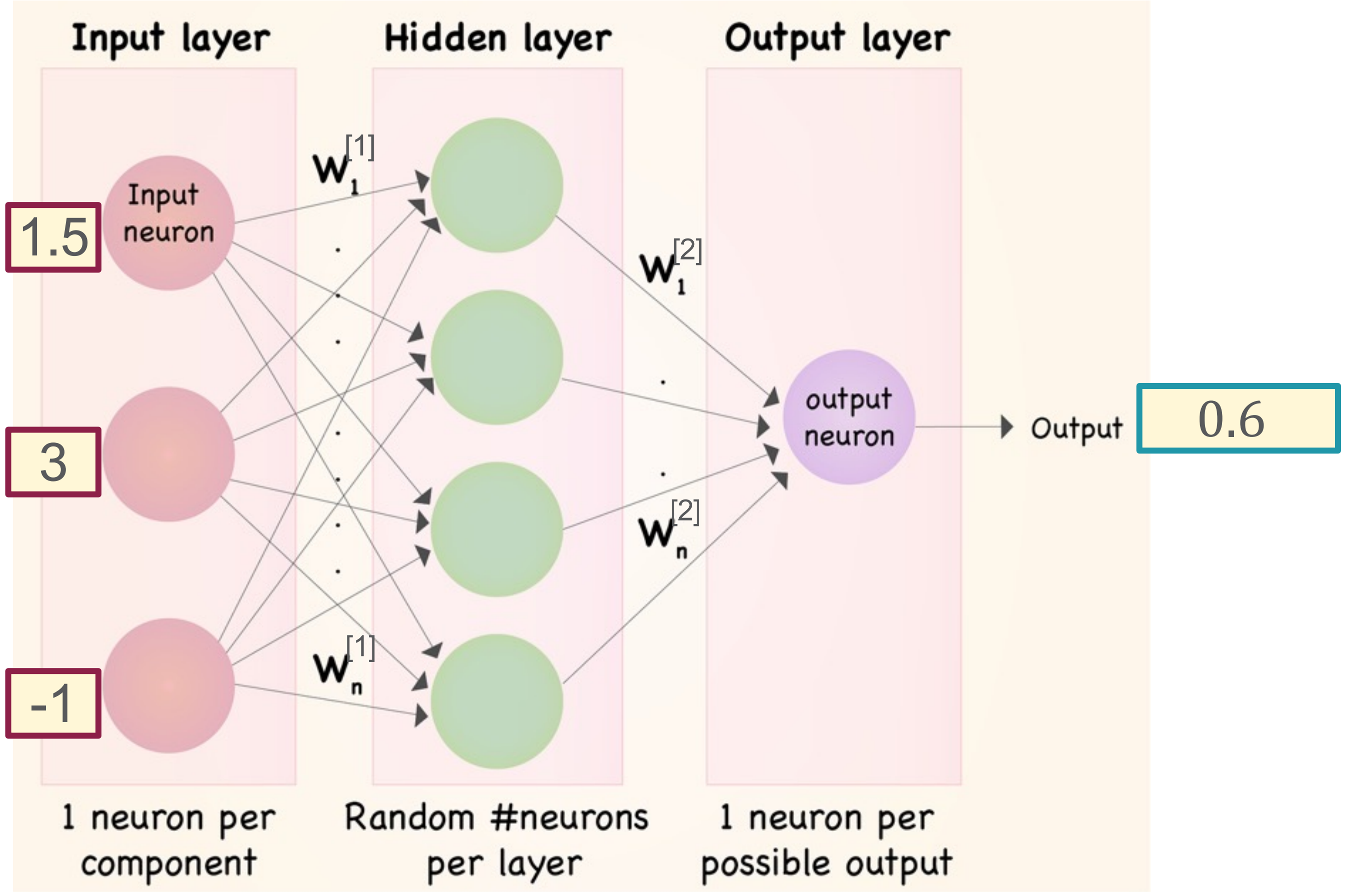
Popularized in 1989 by Rumelhart, Hinton, and Williams in the paper "Learning representations by back-propagating errors".

Jürgen Schmidhuber, "Who Invented Backpropagation?" (2014)

Feed-Forward Network

Data Samples

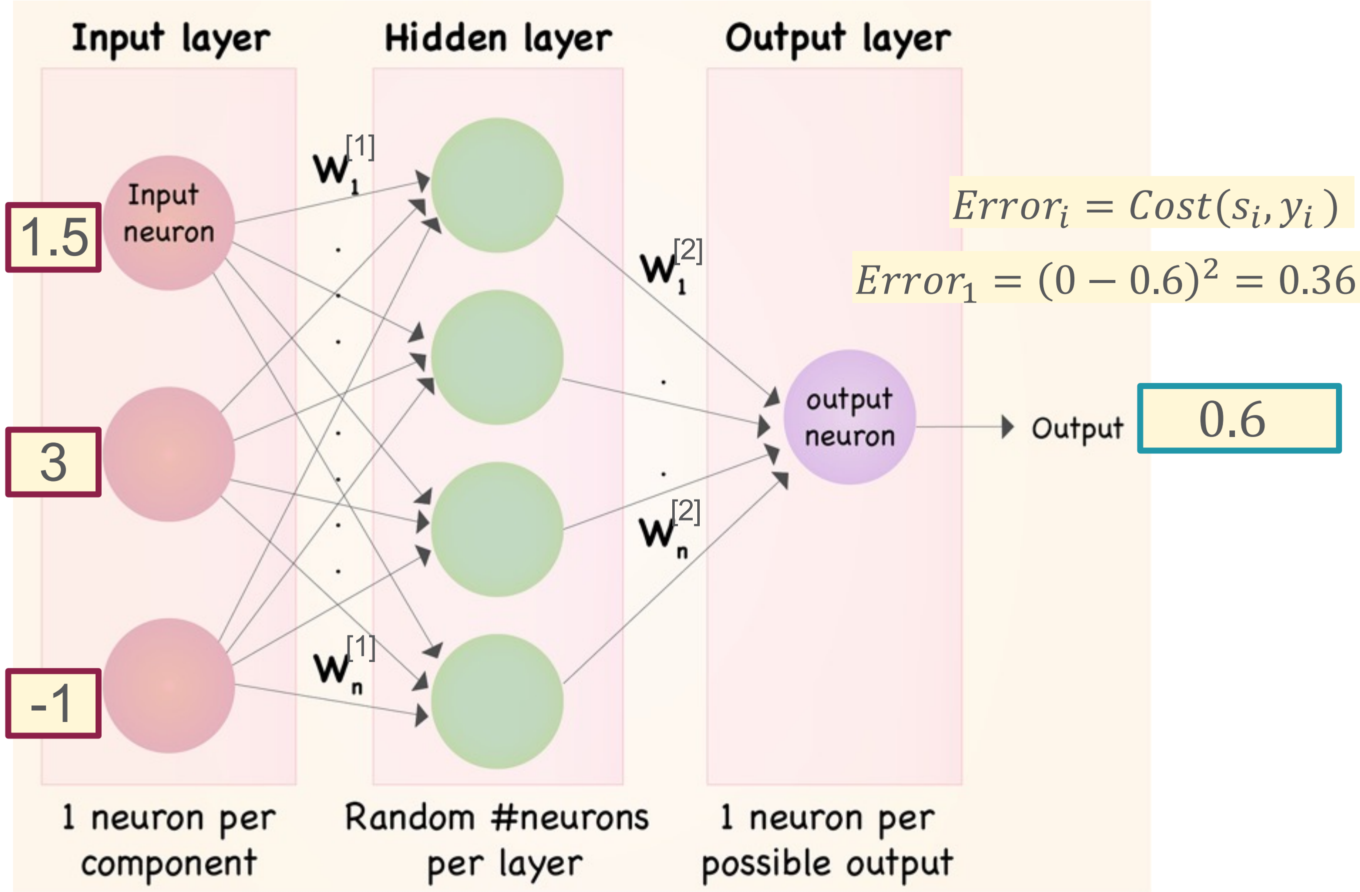
Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



Backpropagation Algorithm (High Level)

Data Samples

Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



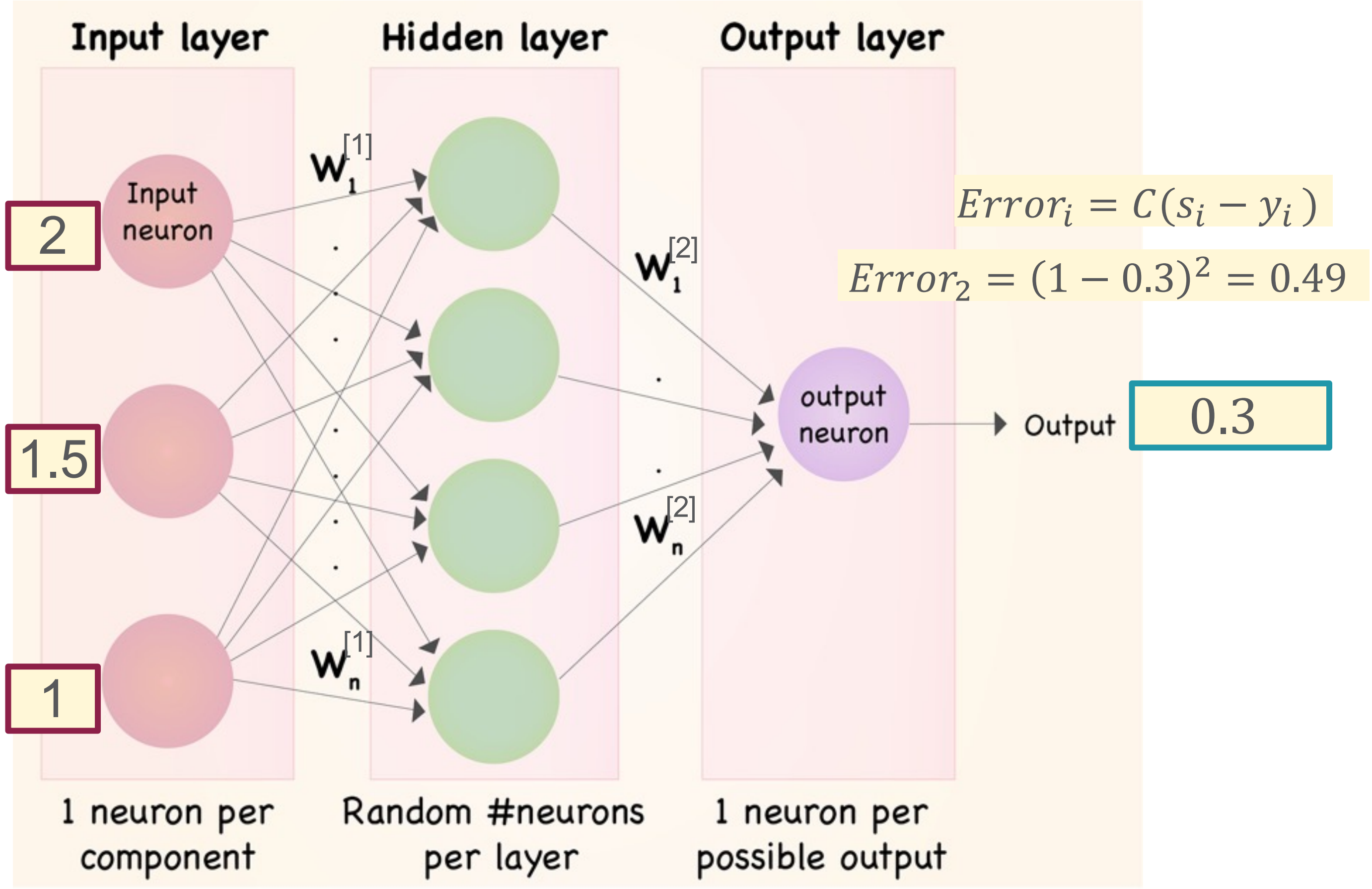
$$Error_i = Cost(s_i, y_i)$$

$$Error_1 = (0 - 0.6)^2 = 0.36$$

Backpropagation Algorithm (High Level)

Data Samples

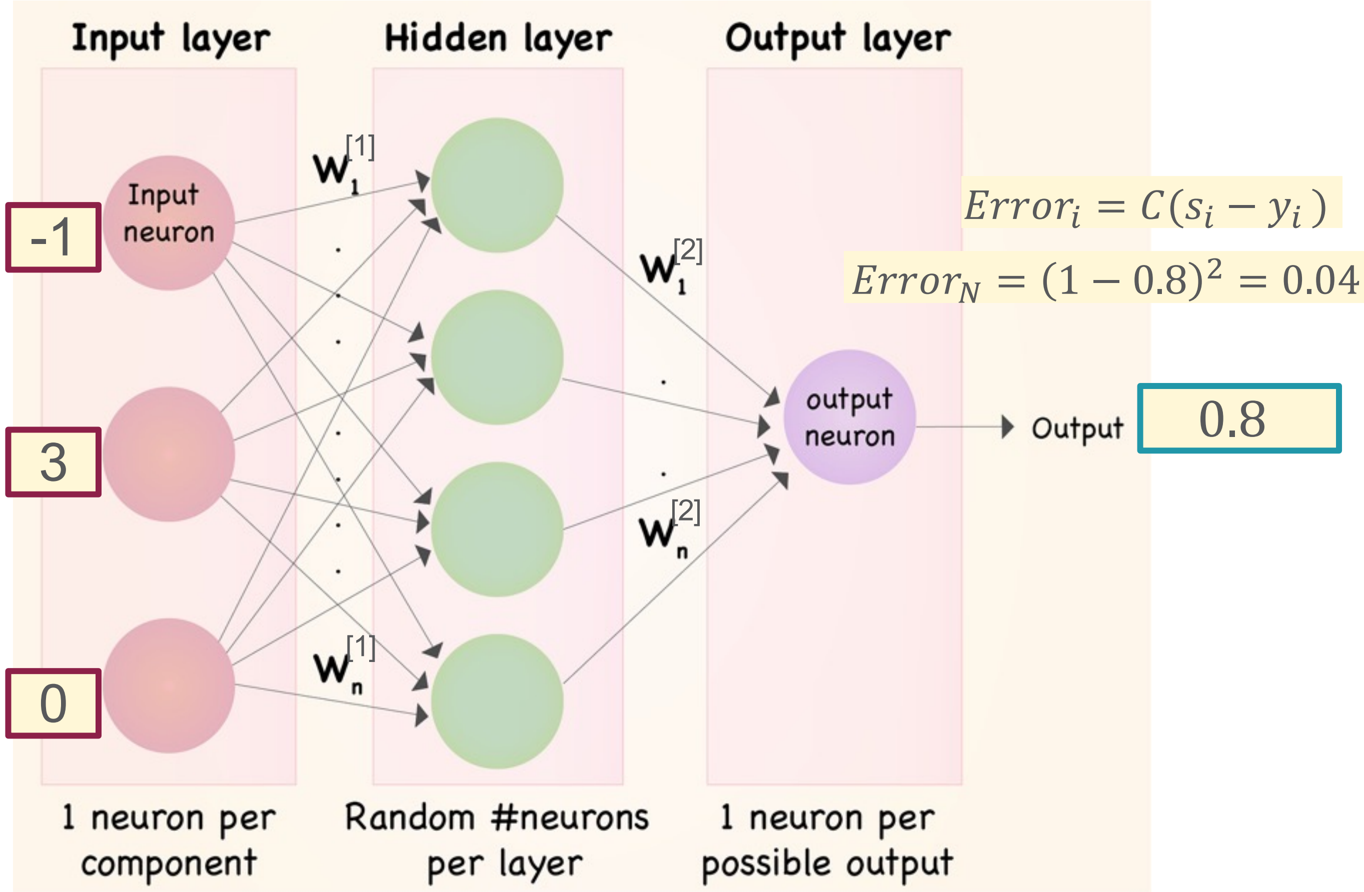
Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



Backpropagation Algorithm (High Level)

Data Samples

Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



$Error_i = C(s_i - y_i)$

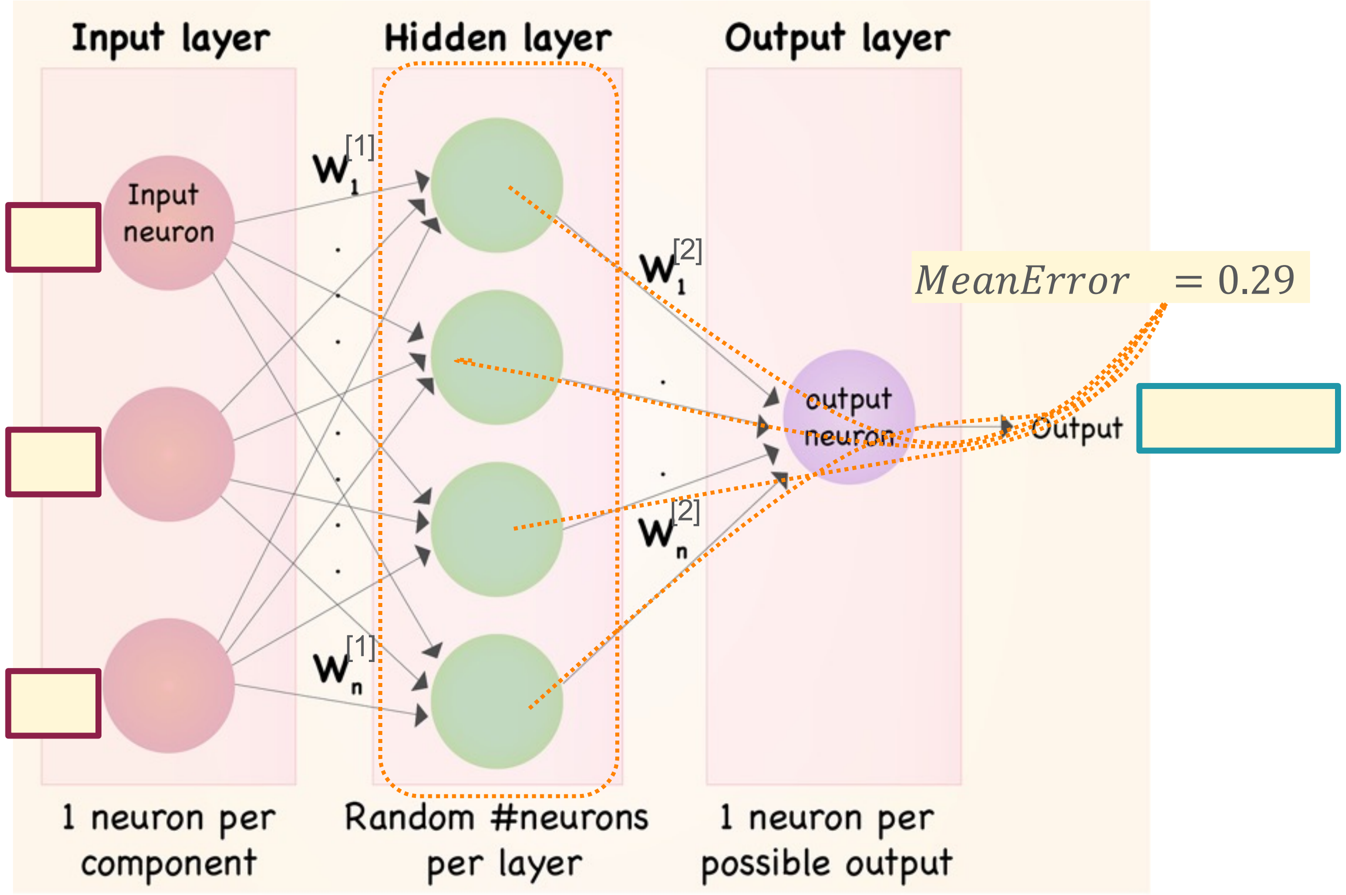
$Error_N = (1 - 0.8)^2 = 0.04$

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Backpropagation Algorithm

Data Samples

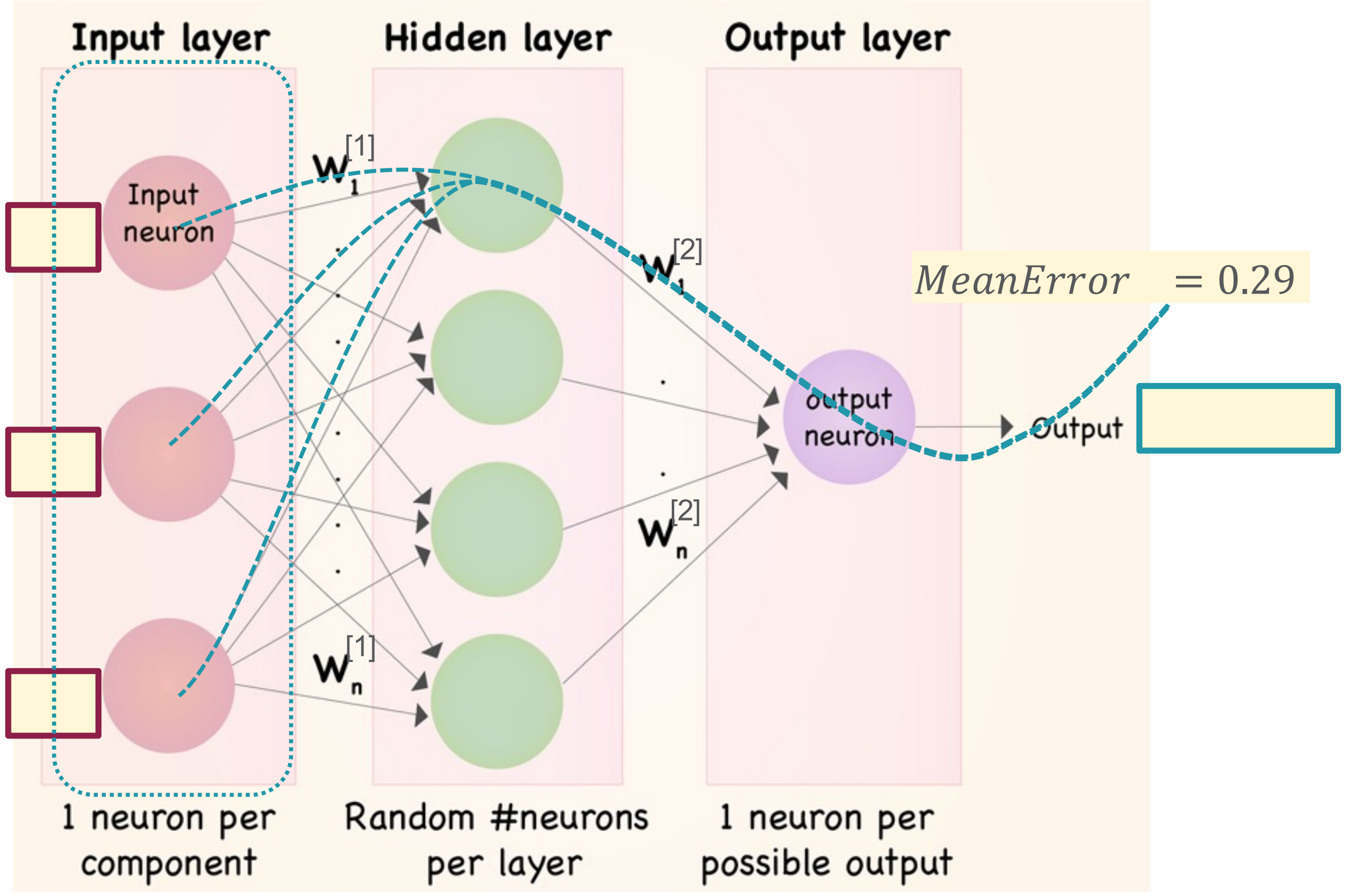
Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



Backpropagation Algorithm

Data Samples

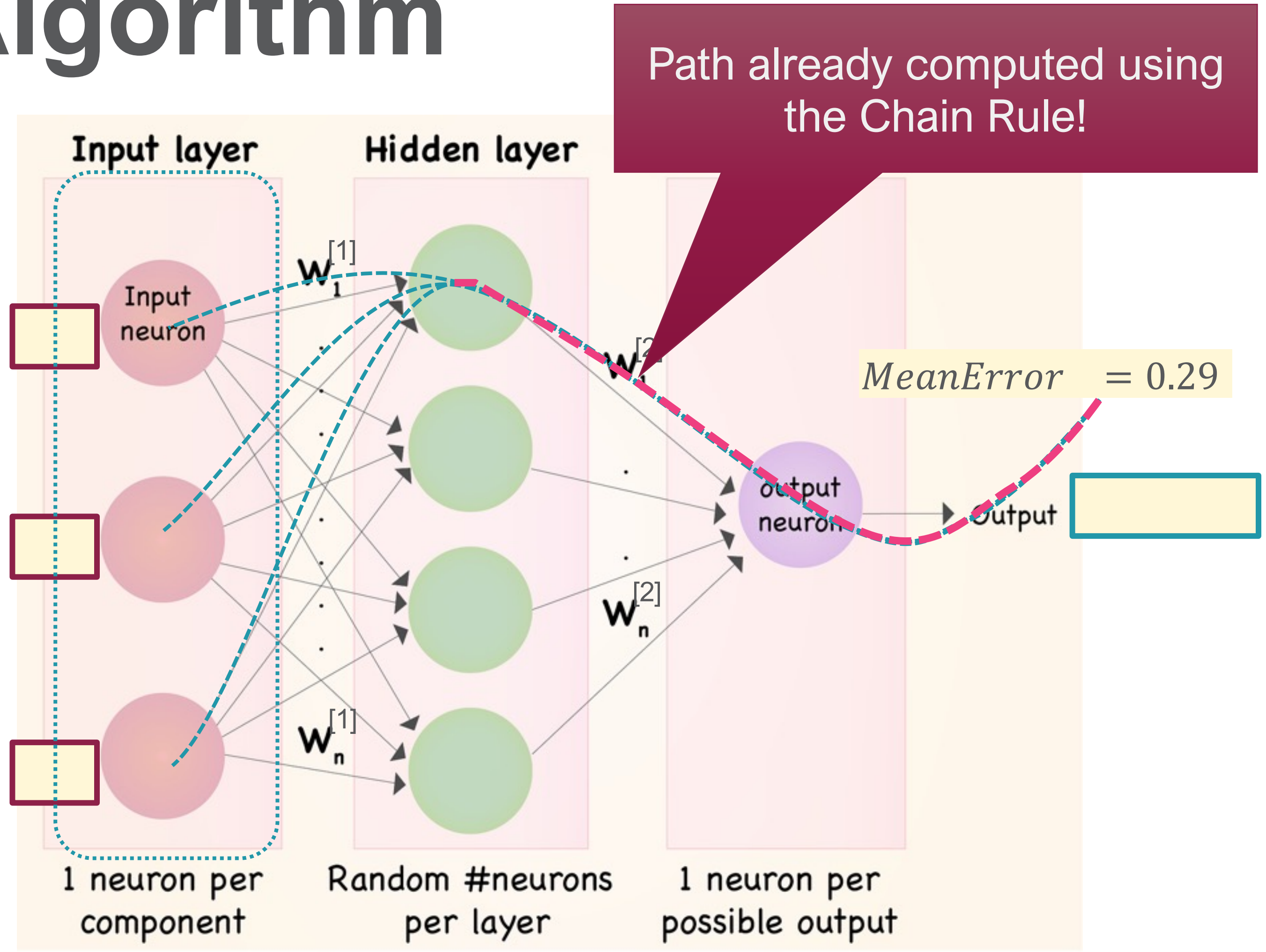
Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



Backpropagation Algorithm

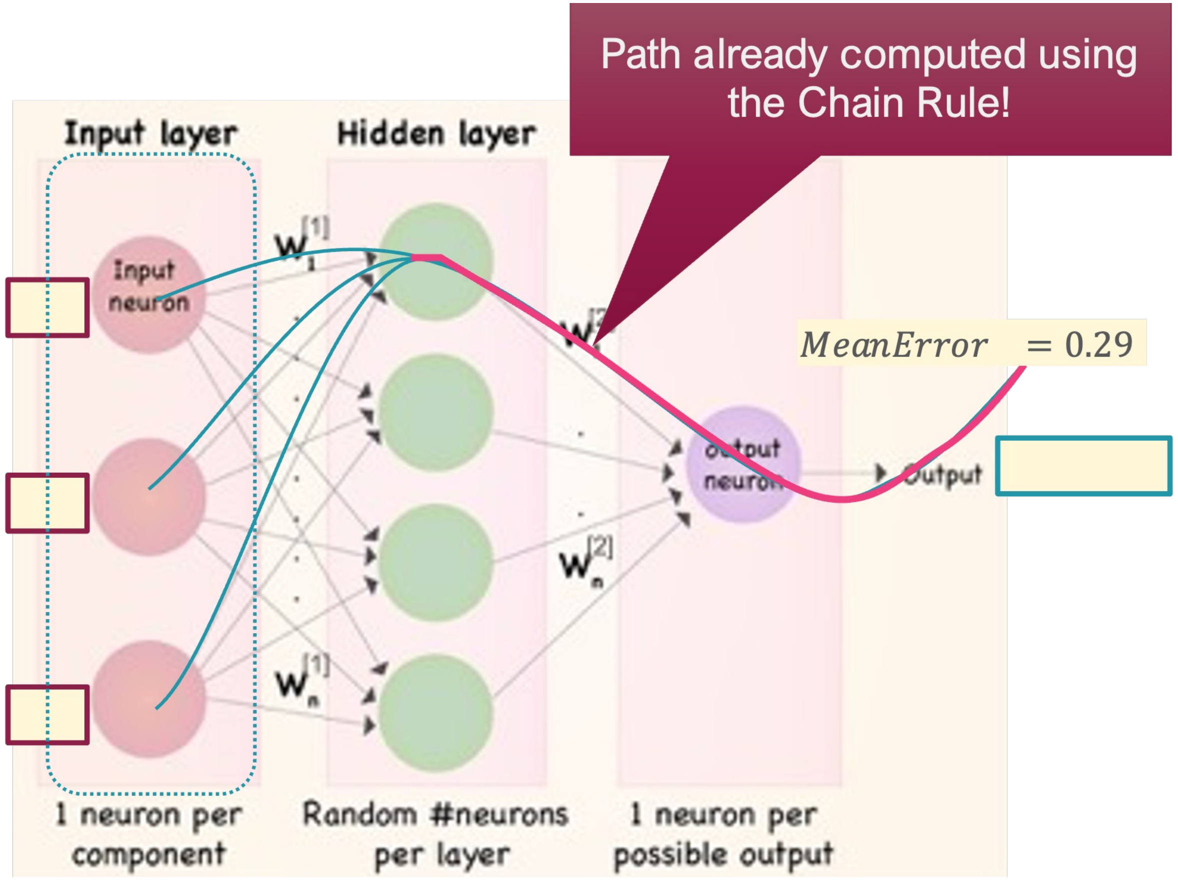
Data Samples

Sample #	x1	x2	x3	y	s
1	1.5	3	-1	0.6	0
2	2	1.5	1	0.3	1
⋮	⋮	⋮	⋮	⋮	⋮
N	-1	3	0	0.8	1



Backpropagation Algorithm

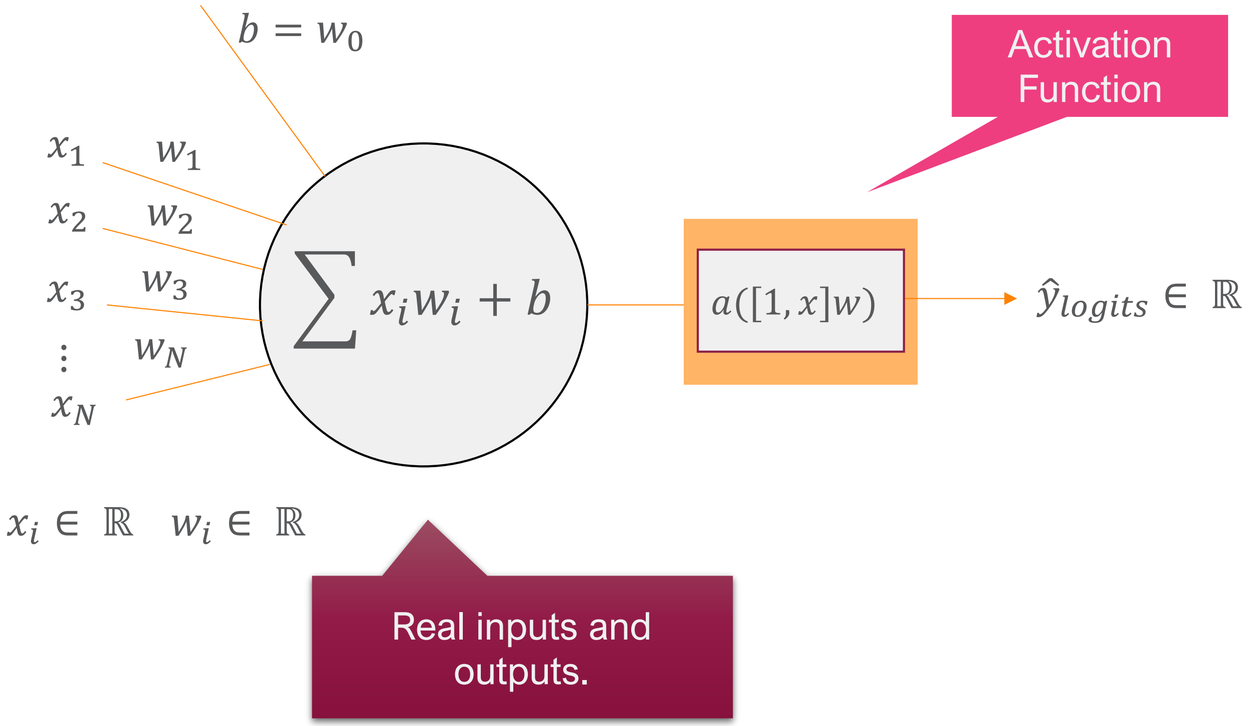
- Layer-wise computation and modularity
  - Layer-size-dependent memory
  - Parallelizability by efficient GPU-based asynchronous matrix multiplication
  - Memory scales linearly with the size of the network
- Mini-batch processing
- Simplicity of Gradient Computation
  - Straightforward
  - Iterative weight updates
  - Allows for techniques to mitigate vanishing and exploding gradients



Activation Functions

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Activation Functions



Desirable properties

- Add non-linearity to the network
- Low computational cost
- Differentiable
  - Otherwise, gradient descent (backpropagation) will not work

Linear Activation Functions

- For a linear activation function:
  - $g^{[l]}(Z^{[l]}) = Z^{[l]}$
  - Also known as an identity activation function
  - Independent of the depth of the network, the model can be collapsed into a single-layer model.
  - It still has its uses…
    - E.g., output layer for linear regression

$$a^{[1]} = z^{[1]} = W^{[1]}X + b^{[1]}$$

$$a^{[2]} = z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = z^{[2]} = W^{[2]}W^{[1]}X + b^{[1]} + b^{[2]}$$

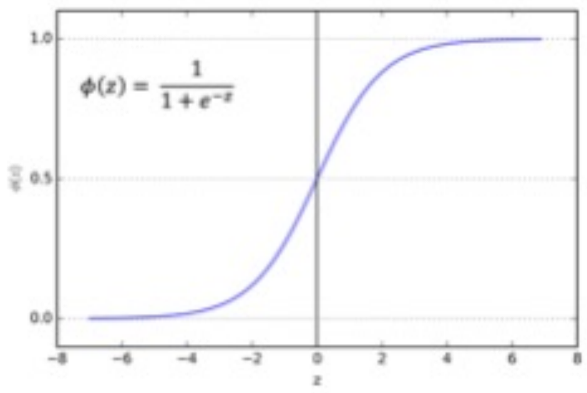
$$a^{[2]} = z^{[2]} = W'X + b'$$

$$\vdots$$

$$a^{[L-1]} = z^{[L-1]} = W'X + b'$$

Traditional Activation Functions

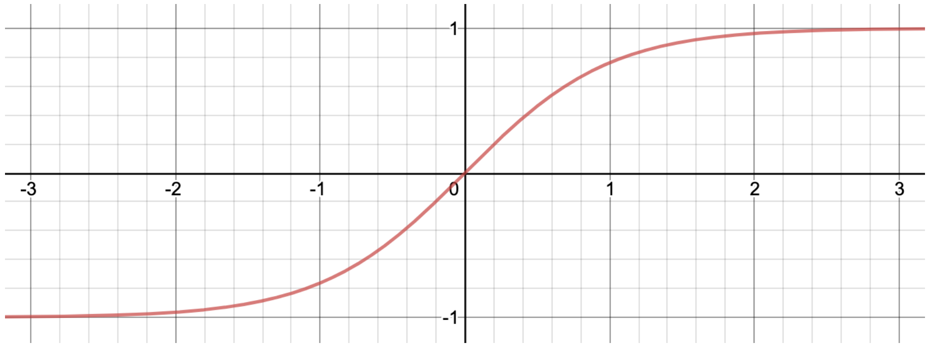
- Sigmoid function
  - Mostly used for binary output layer
  - Small derivatives for large and small z



$$g(z) = \frac{1}{1 + e^{-z}}$$

- Hyperbolic tangent function (tanh)
  - In general, works better than sigmoid
  - Good for hidden units
  - Small derivatives for large and small z

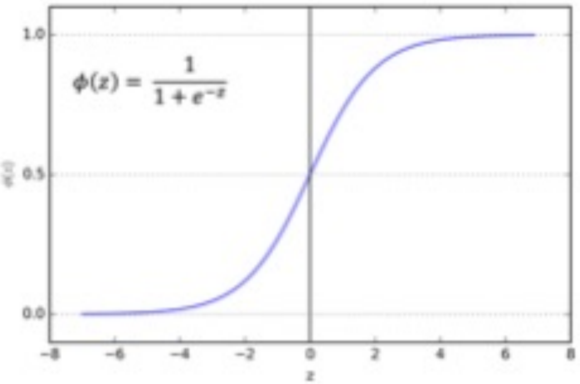
$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



Derivative Sigmoid

$$g(z) = \frac{1}{1+e^{-z}}$$

- Recall: $g'(z) = \sigma'(z) = \sigma(z)(1-\sigma(z))$



$g(3) \approx 1 \rightarrow \quad g'(10) = 1(1-1) = 0$

$g(-3) \approx 0 \rightarrow \quad g'(-10) = 0(1-0) = 0$

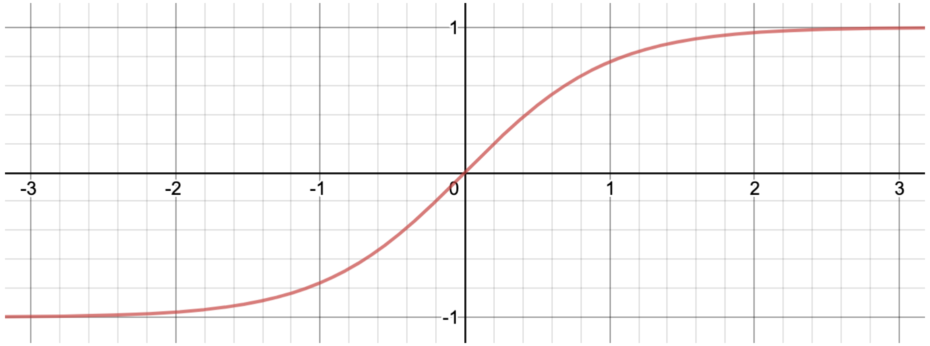
$g(2) \approx 0.88 \rightarrow \quad g'(2) = 0.88(1-0.88) = 0.1$

$g(0.5) \approx 0.62 \rightarrow g'(2) = 0.62(1-0.62) = 0.23$

Max speed ¼ @ z=0

Derivative of Tanh

$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



- $If\ g(z) = \tanh(z)\,, then$
- $g'(z) = 1 - \tanh(z)^2$
- $a = g(z), then\ g'(z) = 1 - a^2$

$$g(3) \approx 1 \rightarrow \quad g'(10) = (1 - 1^2) = 0$$

$$g(-3) \approx -1 \rightarrow \quad g'(-10) = (1 - (-1)^2) = 0$$

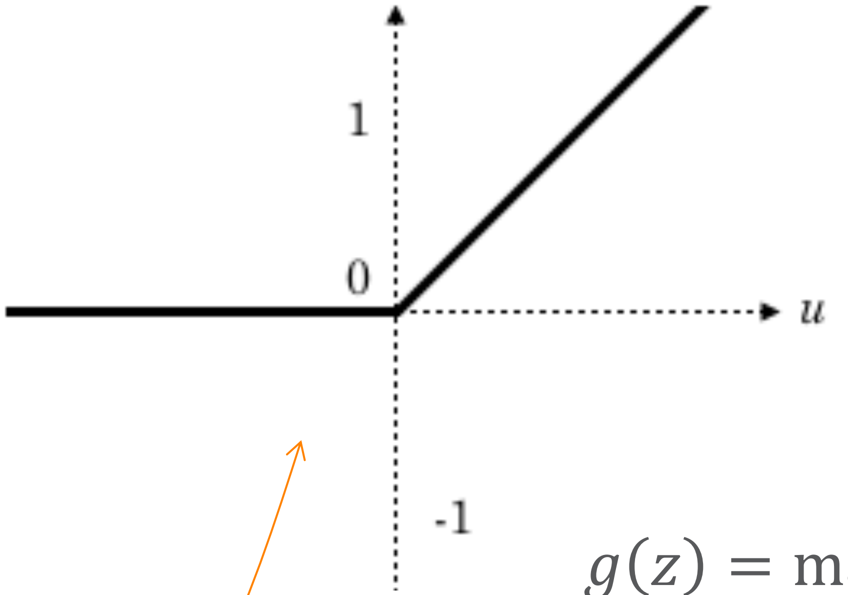
$$g(2) \approx 0.96 \rightarrow \quad g'(2) = (1 - 0.96^2) = 0.07$$

$$g(0.5) \approx 0.46 \rightarrow \quad g'(2) = (1 - 0.46^2) = 0.78$$

Max speed 1 @ z=0

Rectified Linear Unit (ReLU) Function

- The “go-to” activation function
- Derivative is very different from zero
- Derivative at zero is not defined
  - You can set $g'(0) = 0 \; or \; 1$
  - It has zero impact on performance
  - Likelihood of hitting $z = 0$ is unlikely.
- Mitigates vanishing gradients
  - Still can cause exploding gradients
- Dying ReLU problem

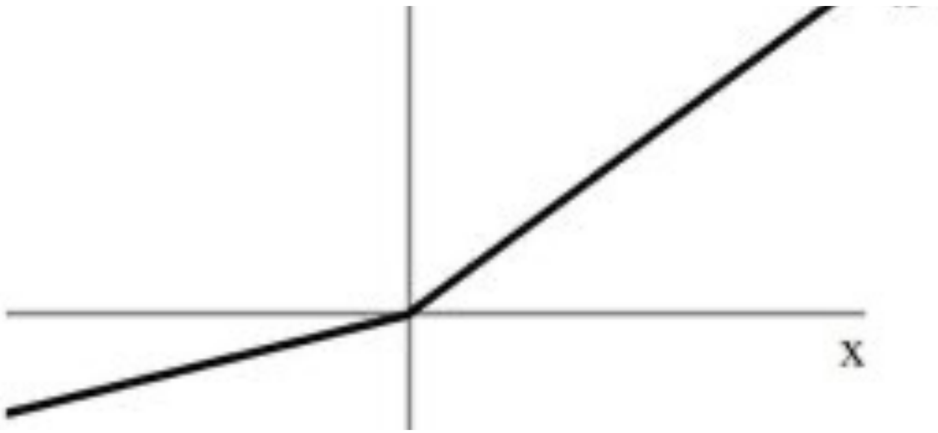


$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

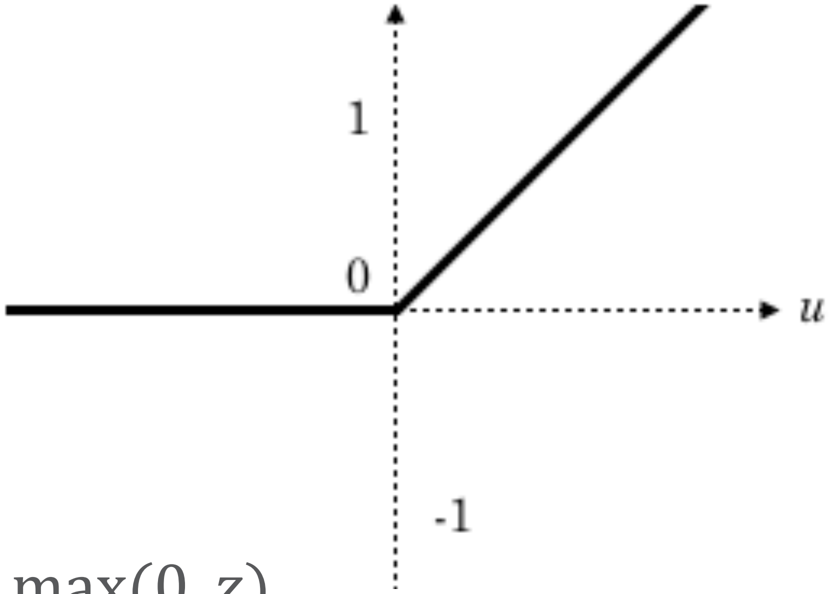
Leaky ReLU

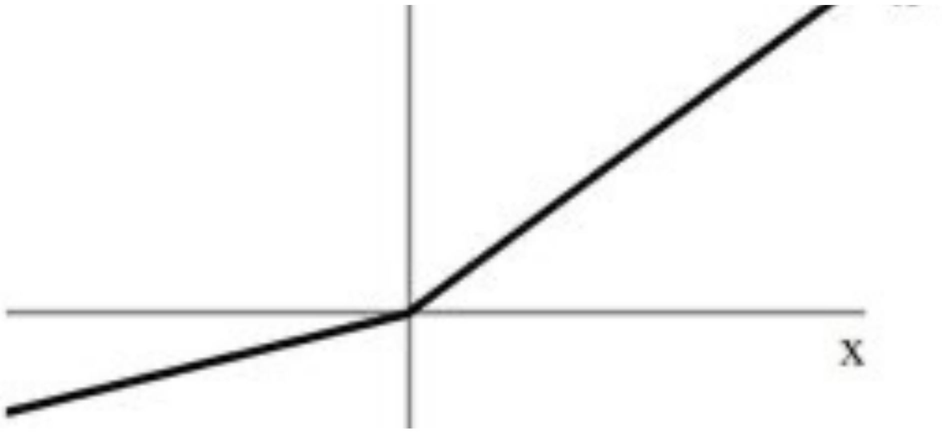
- Works better than standard ReLU
- Fixes Dying ReLU problem
- Derivative is very different from zero
- Alpha usually 0.001. It can also be hyperparameter



$$g(z) = \begin{cases} z \\ \beta z \; for \; z < 0 \end{cases} \qquad g'(z) = \begin{cases} 1 \\ \beta \; for \; z < 0 \end{cases}$$

ReLU and Leaky ReLU Derivatives



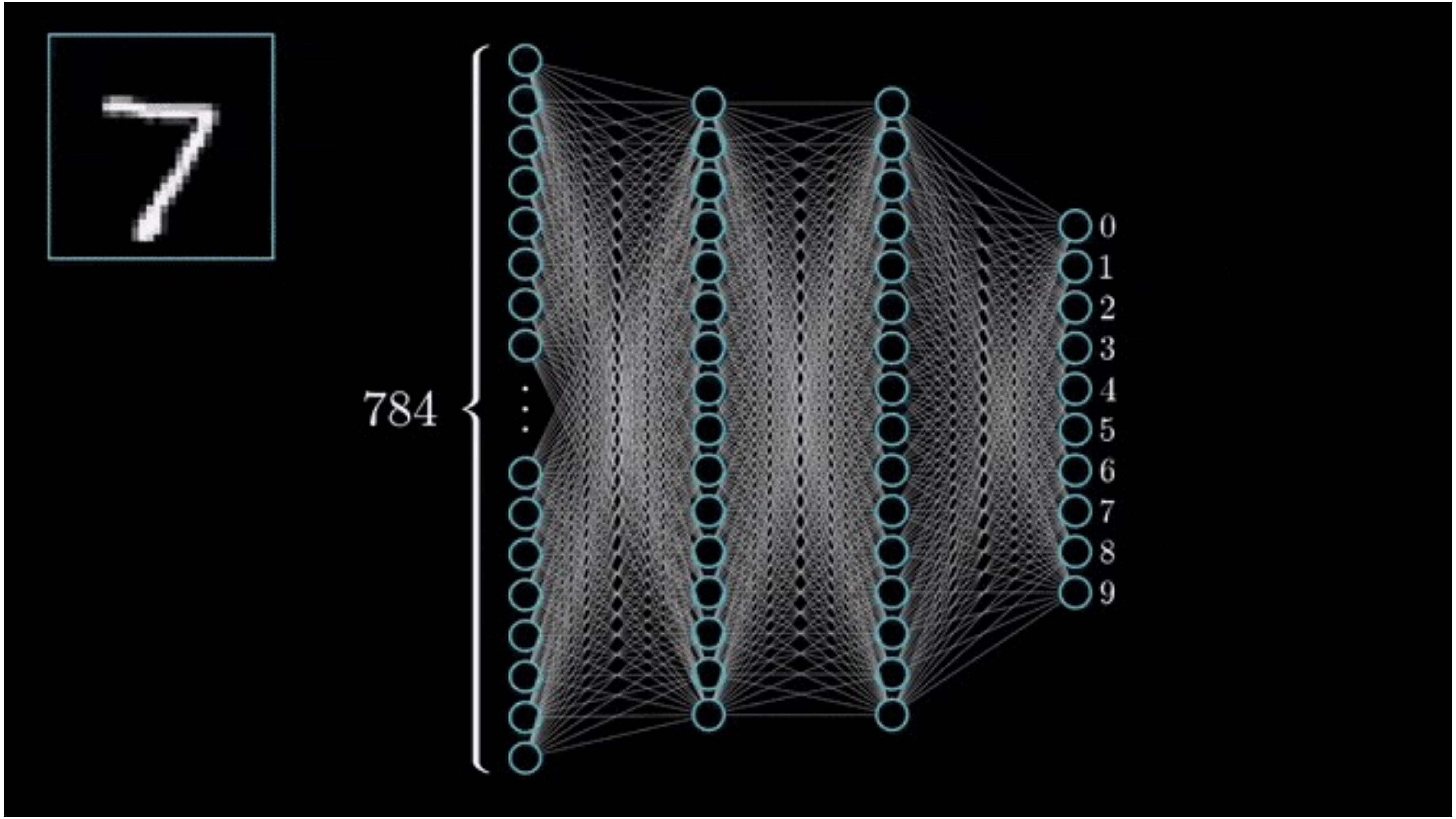


$g(z) = \max(0, z)$

$$g'(z) = \begin{cases} 1 \\ 0 \; for \; z < 0 \end{cases}$$

$$g(z) = \begin{cases} z \\ \beta z \; for \; z < 0 \end{cases}$$

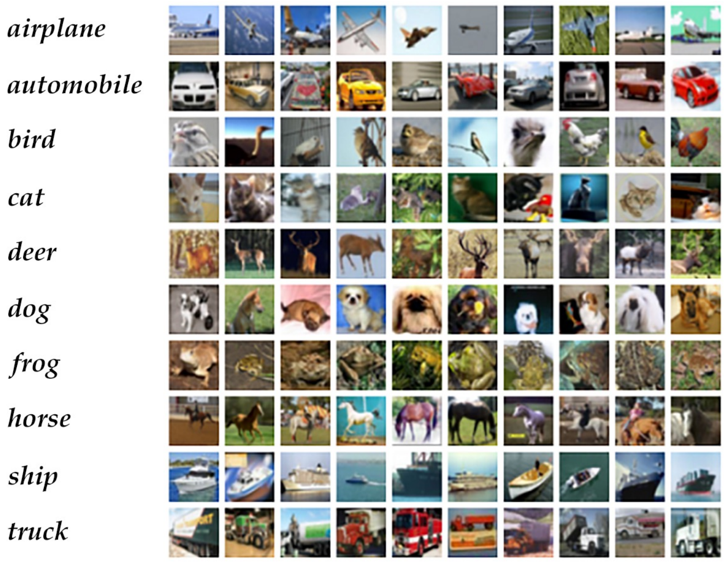
$$g'(z) = \begin{cases} 1 \\ \beta \; for \; z < 0 \end{cases}$$



Credit: Ashis Kumer Biswas, UC Denver

Limitation of Fully Connected NNs

CIFAR 10



32

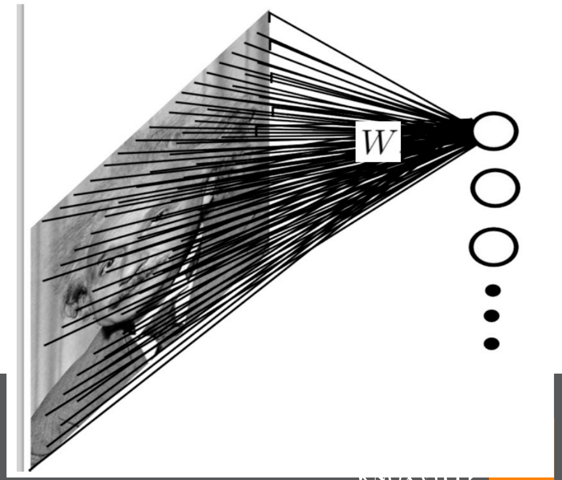
32

x3 → 3,072 $*n^{[1]}$ connections

720

480

x3 → (1M+) $*n^{[1]}$ connections



We want to work with higher-resolution images.

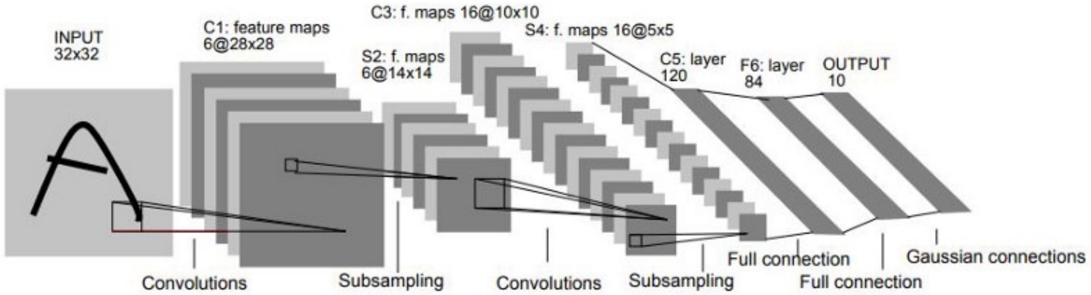
Convolutional Neural Networks

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

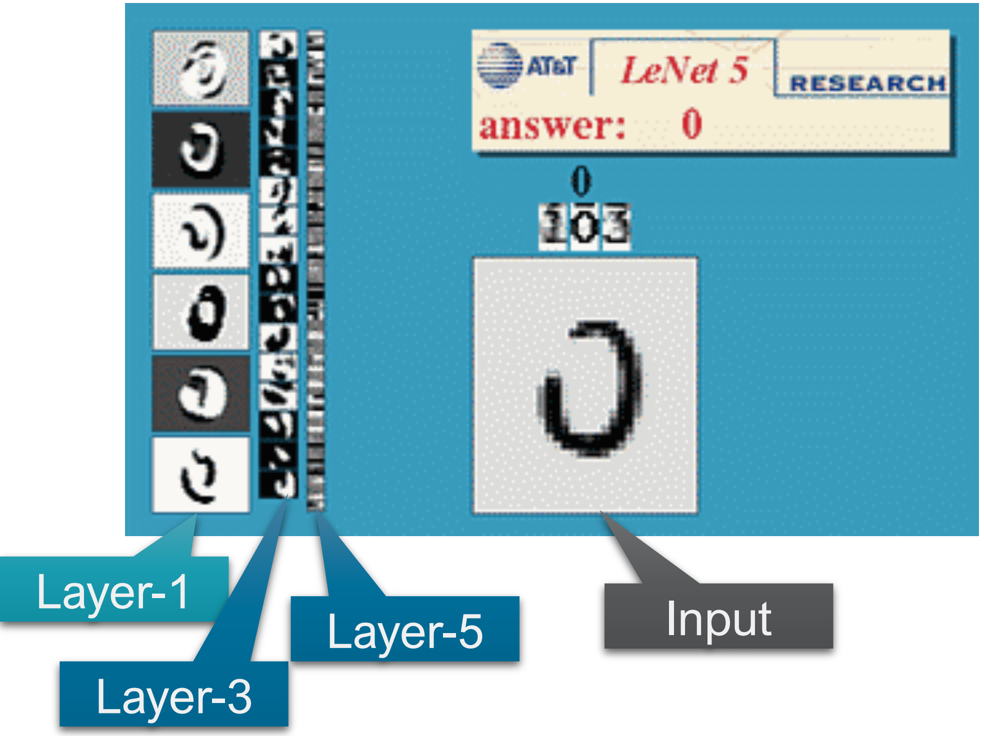
Convolutional Neural Networks

- Automated feature extraction
- Hierarchical feature learning
- Reduction of parameters needed when compared to Fully Connected (FC) networks
- Transfer learning
- Robustness to image variations

LeNet Net 1998, 60k Parameters

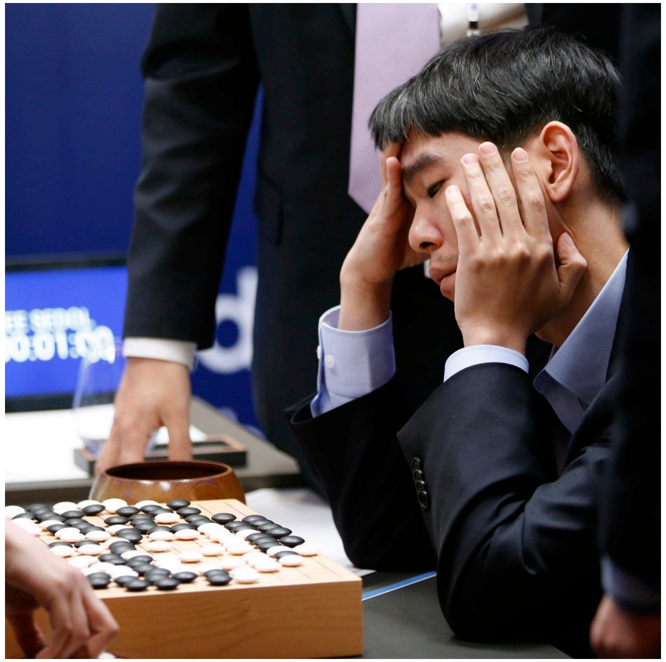


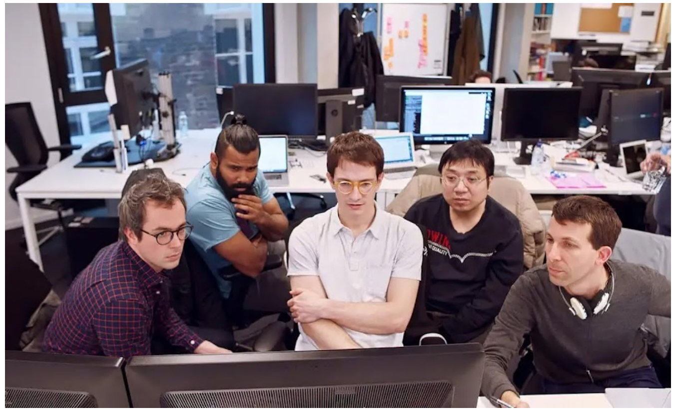
Solution to MINST Dataset and Alpha Go

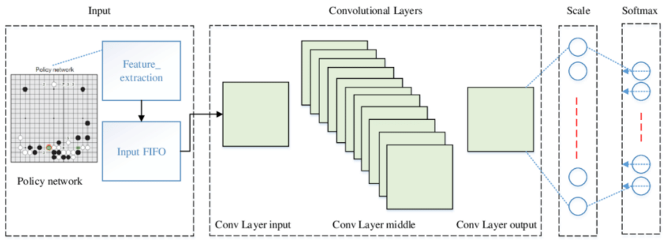


http://yann.lecun.com/exdb/lenet/index.html

"In October 2015, AlphaGo played its first game against the reigning three-time European Champion, Fan Hui. AlphaGo won the first ever match between an AI system and Go professional, scoring 5-0."

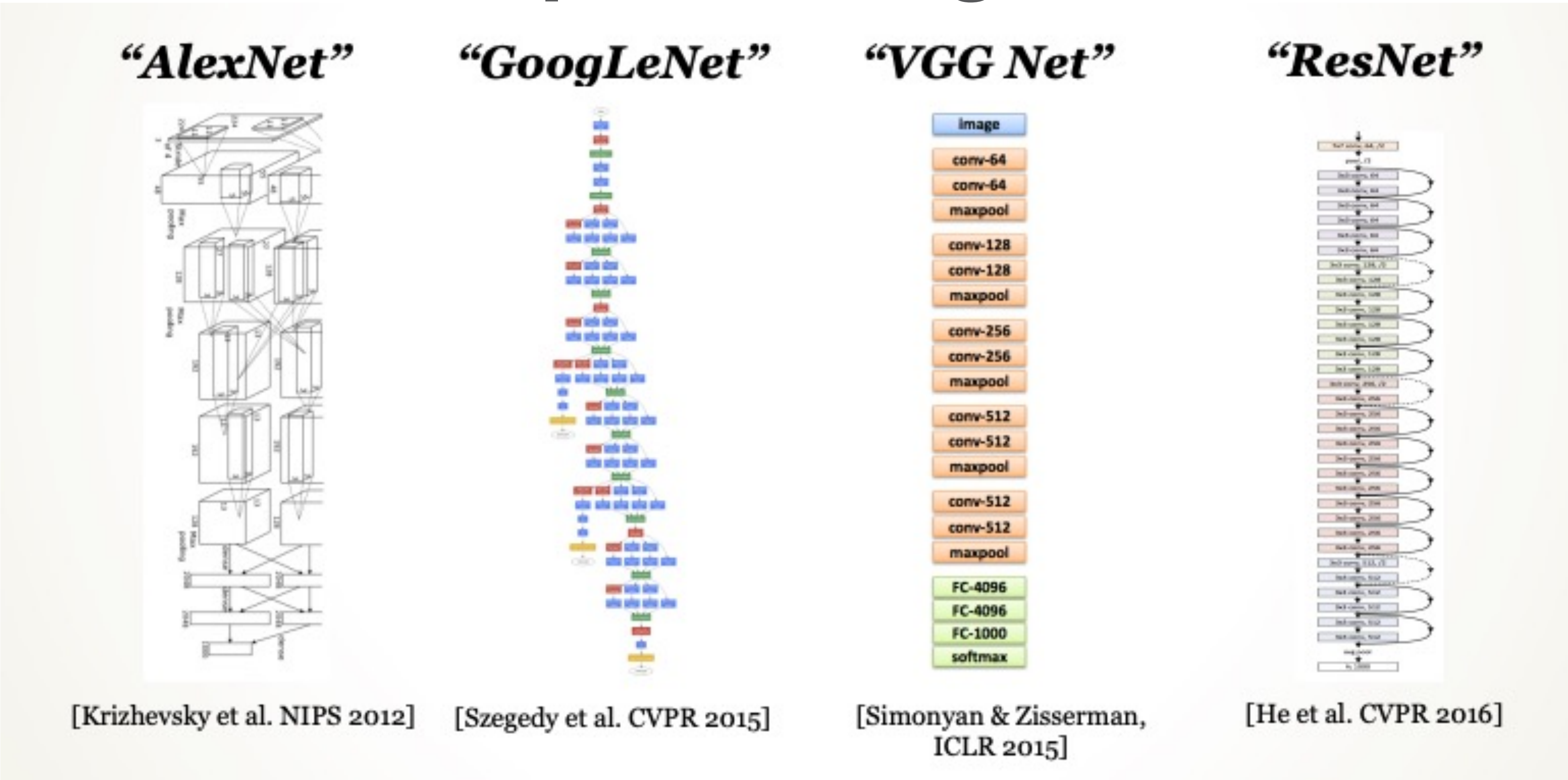






https://deepmind.google/technologies/alphago/

Evolution of Deep Learning Networks



Source: *imagenet_ilsvrc2017_v1.0.pdf*

Types of Convolutional Layers

- Convolutions (CONV)
- Fully connected (FC)
  - Typical neural connections
- Flattening
- Pooling (POOL)

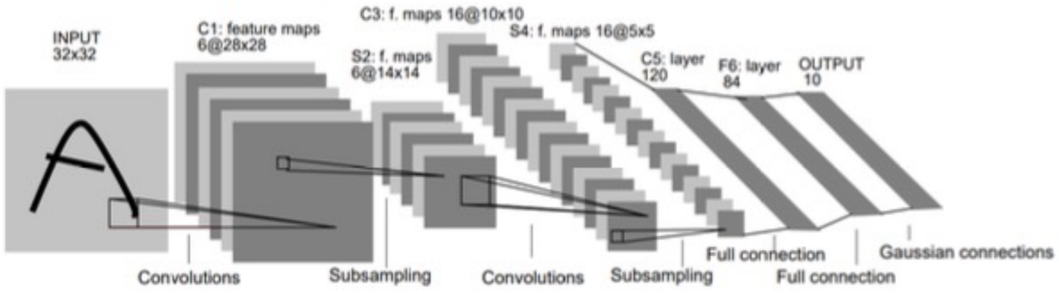
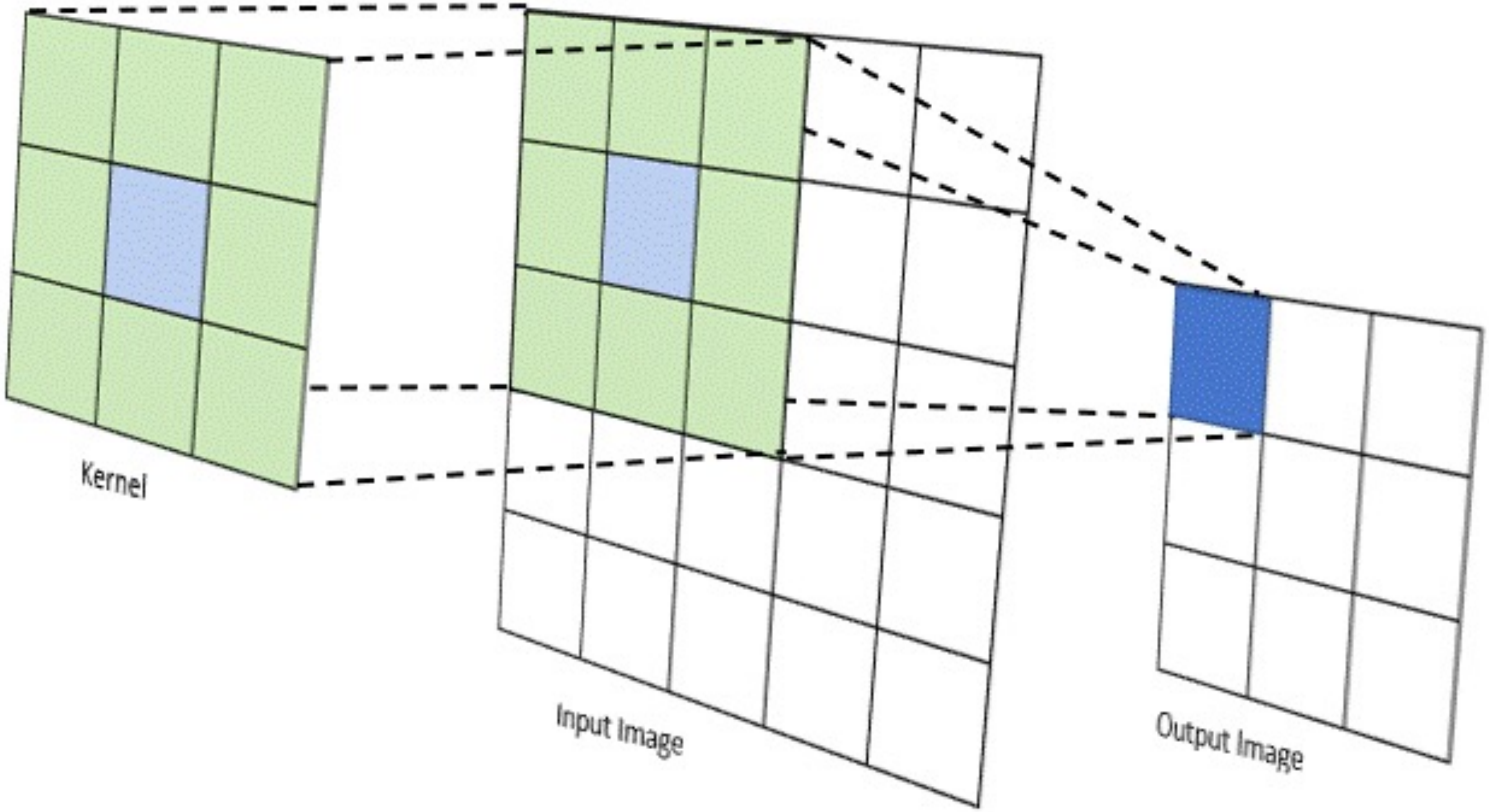


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Convolution Operation



3×3 5×5 $\to$ 3×3

In formal math, this is known as cross-correlation.

- Convolution requires a left-right and up-down flip of the filter before multiplication.

Original Filter

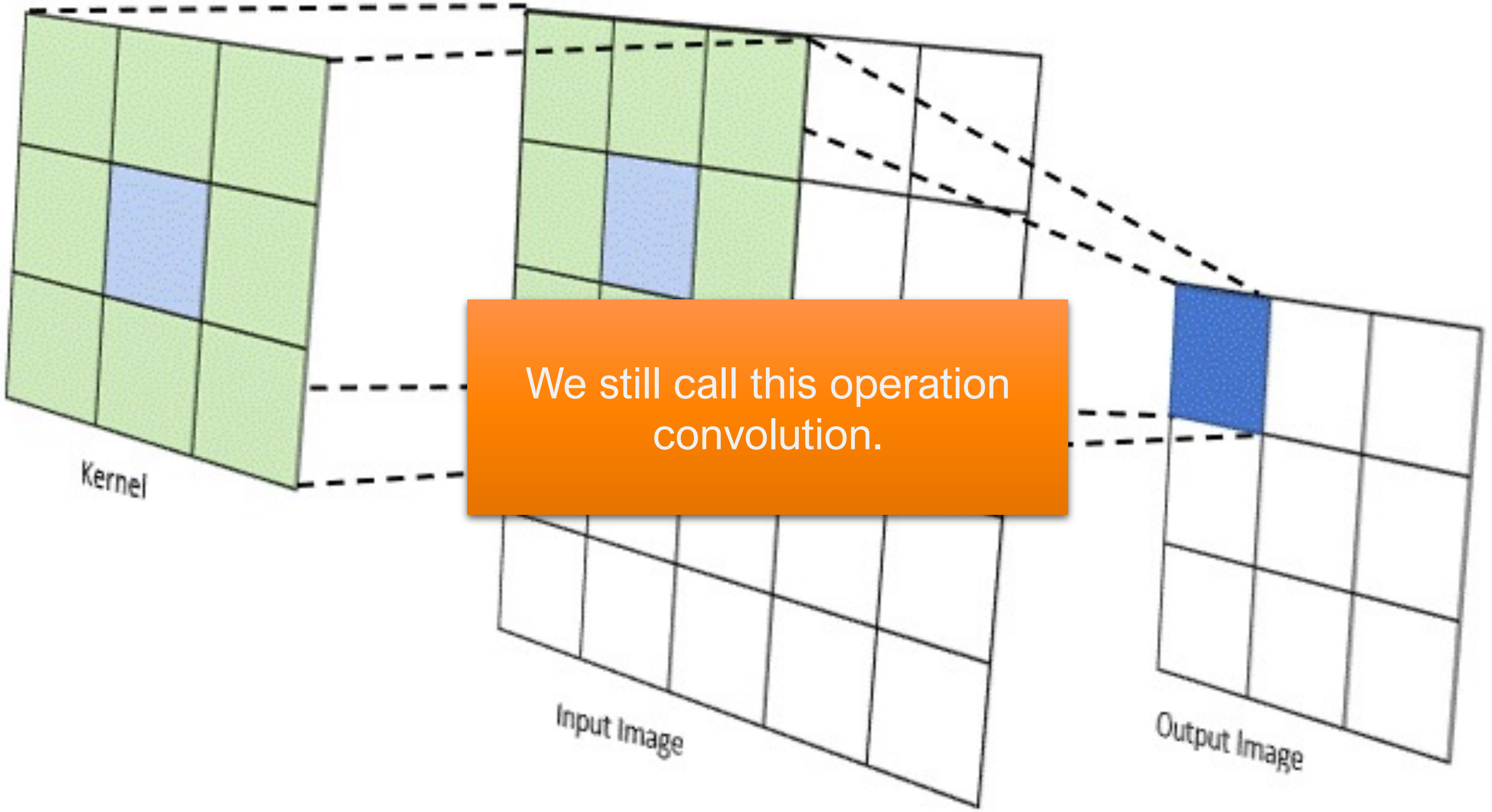
1	2	3
4	5	6
7	8	9

Before Matrix Mult.

9	8	7
6	5	4
3	2	1

Animation by Tim Terati: https://medium.com/@timothy_terati/image-convolution-filtering-a54dce7c786b

Convolution Operation



We still call this operation convolution.

3×3 $5\times5 \rightarrow$ 3×3

In formal math, this is known as

- flip
of the filter before
multiplication.

This operation has zero impact on DL model design or performance.

Original Filter

1	2	3

So, we can save computing on these steps.

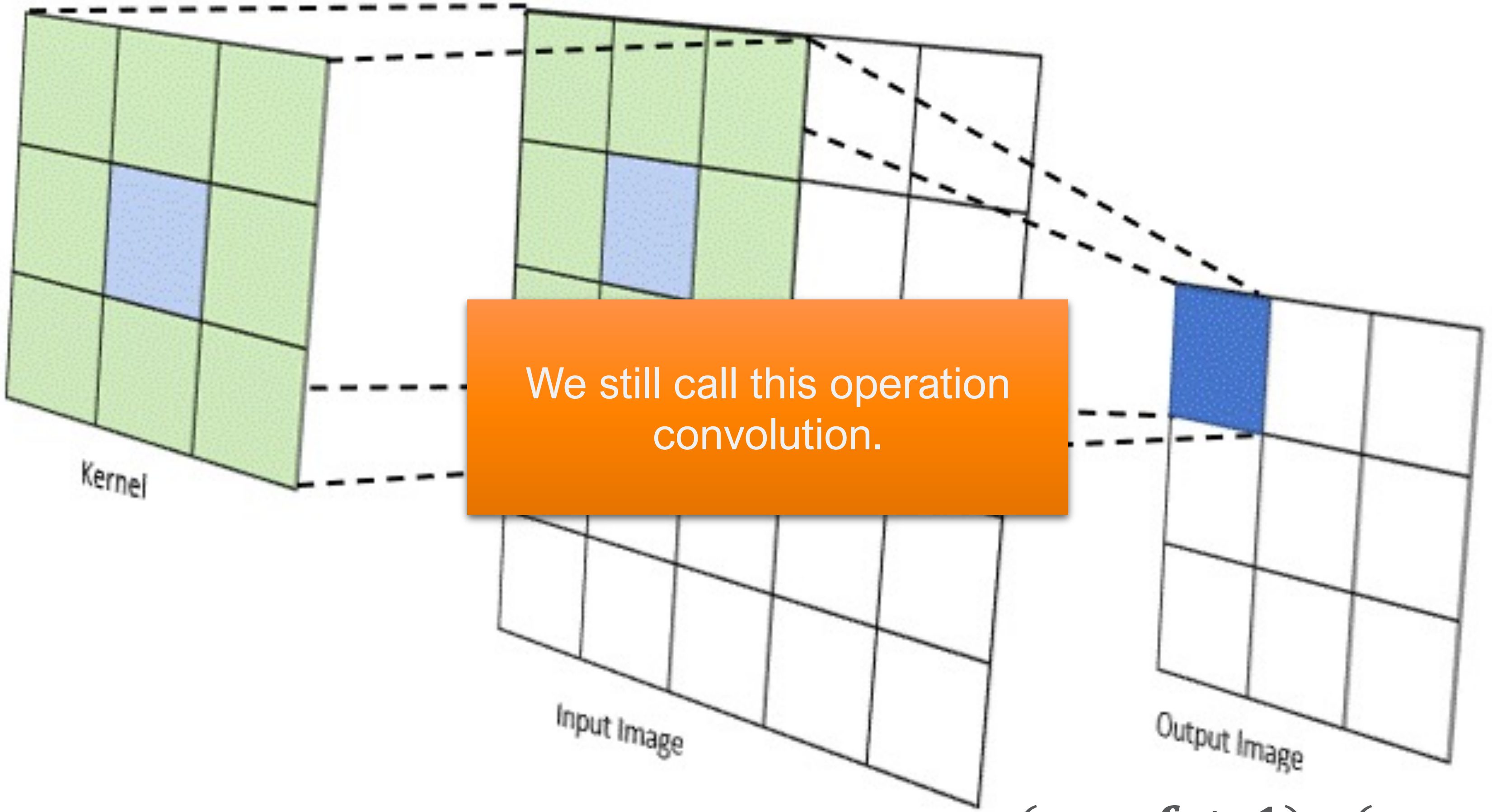
Before Matrix Mult.

9	8	7
6	5	4
3	2	1

Animation by Tim Terati: https://medium.com/@timothy_terati/image-convolution-filtering-a54dce7c786b

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Convolution Operation



$f \times f$ $\quad n_h \times n_w \rightarrow (n_h - f + 1) \times (n_w - f + 1)$

We still call this operation convolution.

In formal math, this is known as

- flip

multiplication.

This operation has zero impact on DL model design or performance.

Original Filter

1	2	3

So, we can save computing on these steps.

Before Matrix Mult.

9	8	7
6	5	4
3	2	1

Animation by Tim Terati: https://medium.com/@timothy_terati/image-convolution-filtering-a54dce7c786b

Vertical Edge Detection

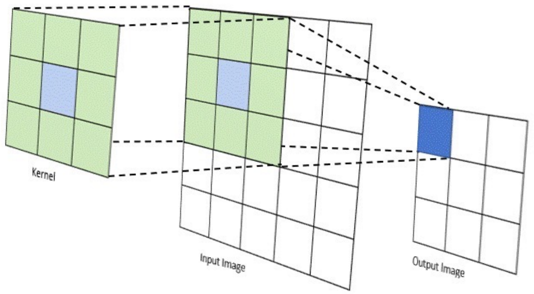
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

$*$

1	0	-1
1	0	-1
1	0	-1

$=$

Vertical Edge Detection



$^{1}5$	$^{0}5$	$^{-1}5$	0	0	0
$^{1}5$	$^{0}5$	$^{-1}5$	0	0	0
$^{1}5$	$^{0}5$	$^{-1}5$	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

$*$

1	0	-1
1	0	-1
1	0	-1

$=$

0			

$$1 * 5 + 0 * 5 + (-1) * 5 + 1 * 5 + 0 * 5 + (-1) * 5 + 1 * 5 + 0 * 5 + (-1) * 5$$

Vertical Edge Detection

5	$^{1}5$	$^{0}5$	$^{-1}0$	0	0
5	$^{1}5$	$^{0}5$	$^{-1}0$	0	0
5	$^{1}5$	$^{0}5$	$^{-1}0$	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	15		

Vertical Edge Detection

$$\begin{bmatrix} 5 & 5 & {}^{1}5 & {}^{0}0 & {}^{-1}0 & 0 \\ 5 & 5 & {}^{1}5 & {}^{0}0 & {}^{-1}0 & 0 \\ 5 & 5 & {}^{1}5 & {}^{0}0 & {}^{-1}0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 15 & 15 & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Vertical Edge Detection

5	5	5	^{1}0	^{0}0	$^{-1}$0
5	5	5	^{1}0	^{0}0	$^{-1}$0
5	5	5	^{1}0	^{0}0	$^{-1}$0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

$*$

1	0	-1
1	0	-1
1	0	-1

$=$

0	15	15	0

Vertical Edge Detection

5	5	5	0	0	0
$^{1}5$	$^{0}5$	$^{-1}5$	0	0	0
$^{1}5$	$^{0}5$	$^{-1}5$	0	0	0
$^{1}5$	$^{0}5$	$^{-1}5$	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

$*$

1	0	-1
1	0	-1
1	0	-1

$=$

0	15	15	0
0			

Vertical Edge Detection

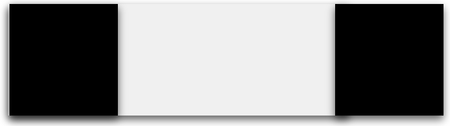
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	15	15	0
0	15	15	0
0	15	15	0
0	15	15	0



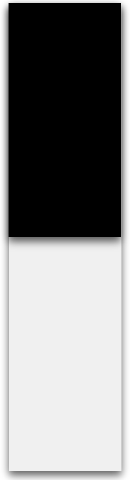
Vertical Edge Detection

It also tells us the direction of the transition.

$$\begin{bmatrix} 0 & 0 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 5 & 5 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -15 & -15 & 0 \\ 0 & -15 & -15 & 0 \\ 0 & -15 & -15 & 0 \\ 0 & -15 & -15 & 0 \end{bmatrix}$$

Horizontal Edge Det

If we use the same filter



5	5	5	5	5	5
5	5	5	5	5	5
5	5	5	5	5	5
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

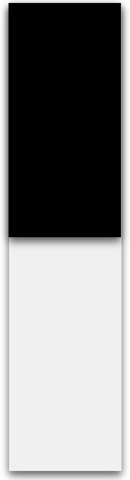
*

1	0	-1
1	0	-1
1	0	-1

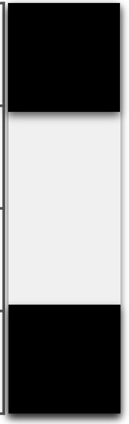
=

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

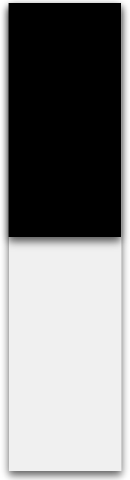
Horizontal Edge Detection



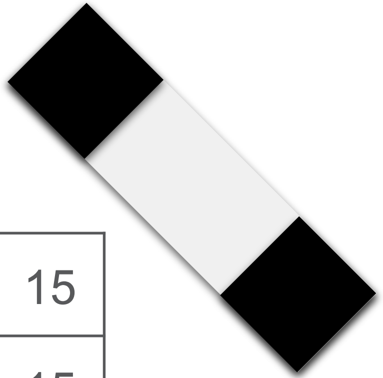
$$
\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 15 & 15 & 15 & 15 \\ 15 & 15 & 15 & 15 \\ 0 & 0 & 0 & 0 \end{bmatrix}
$$



Diagonal Edge Detection



$$\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 5 & 15 \\ 0 & 5 & 15 & 15 \\ 5 & 15 & 15 & 5 \\ 15 & 15 & 5 & 0 \end{bmatrix}$$



How does convolution help us?

1	0	-1
1	0	-1
1	0	-1

Filter

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Gaussian filter

Image

5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0
5	5	5	0	0	0

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

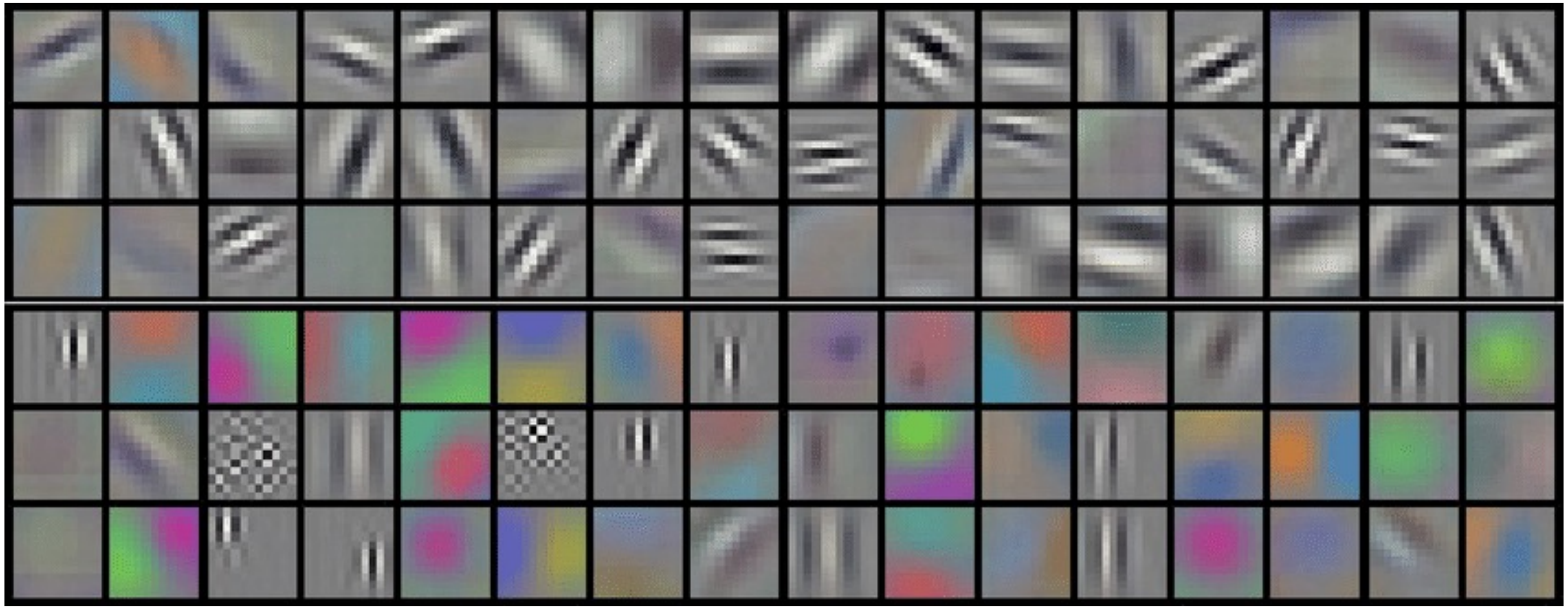
$=$

Output

How can we learn these weights?

Convolutional Neural Networks

Learn multiple filters

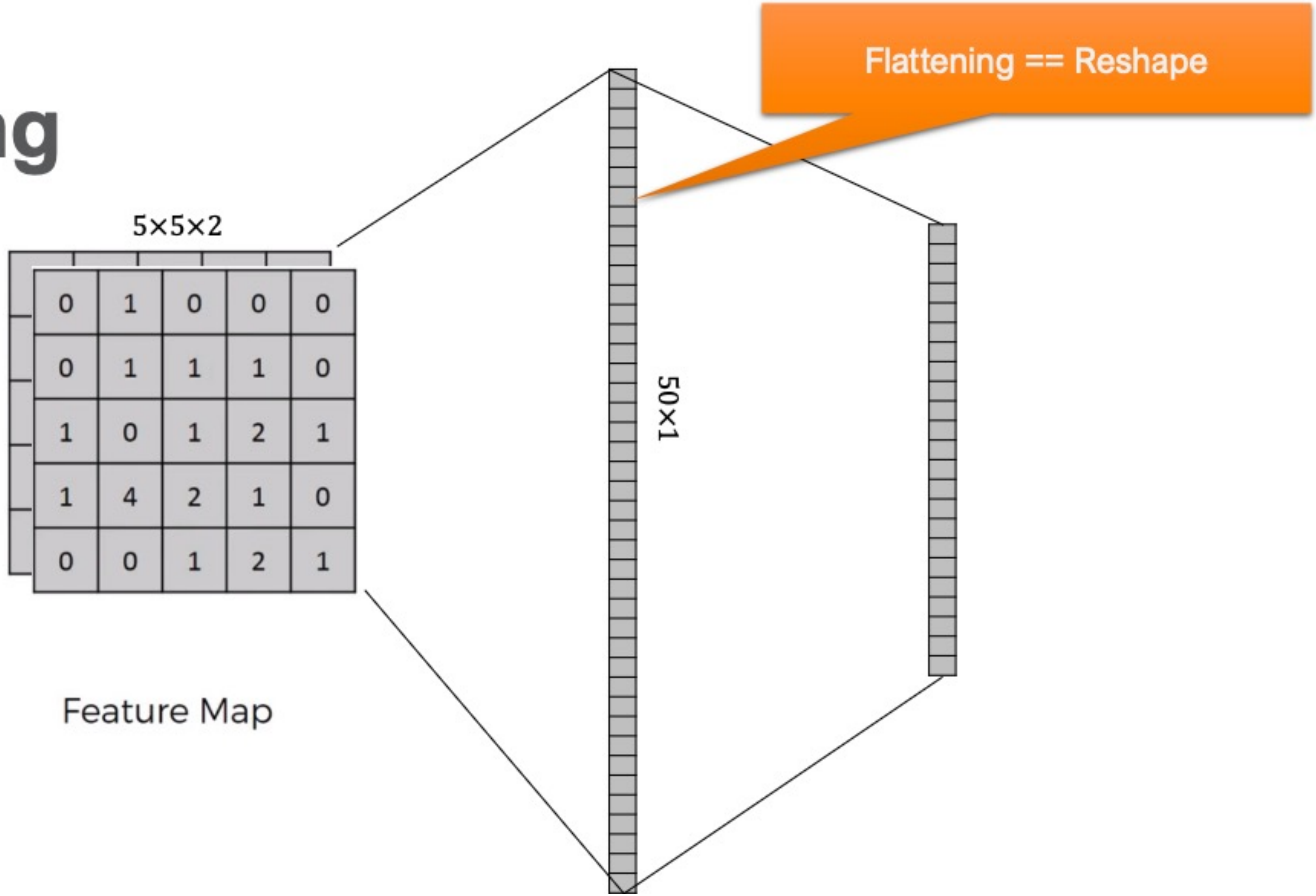


Slide: M. Ranzato

Feature Map

Image: Rob Fergus

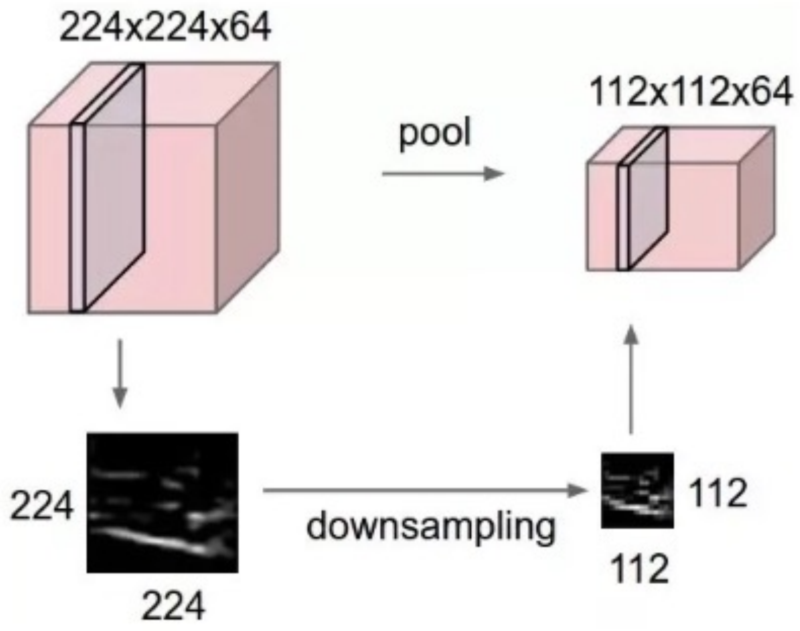
Flattening



Flattening == Reshape
5×5×2
0 1 0 0 0
0 1 1 1 0
1 0 1 2 1
1 4 2 1 0
0 0 1 2 1
50×1
Feature Map

Pooling

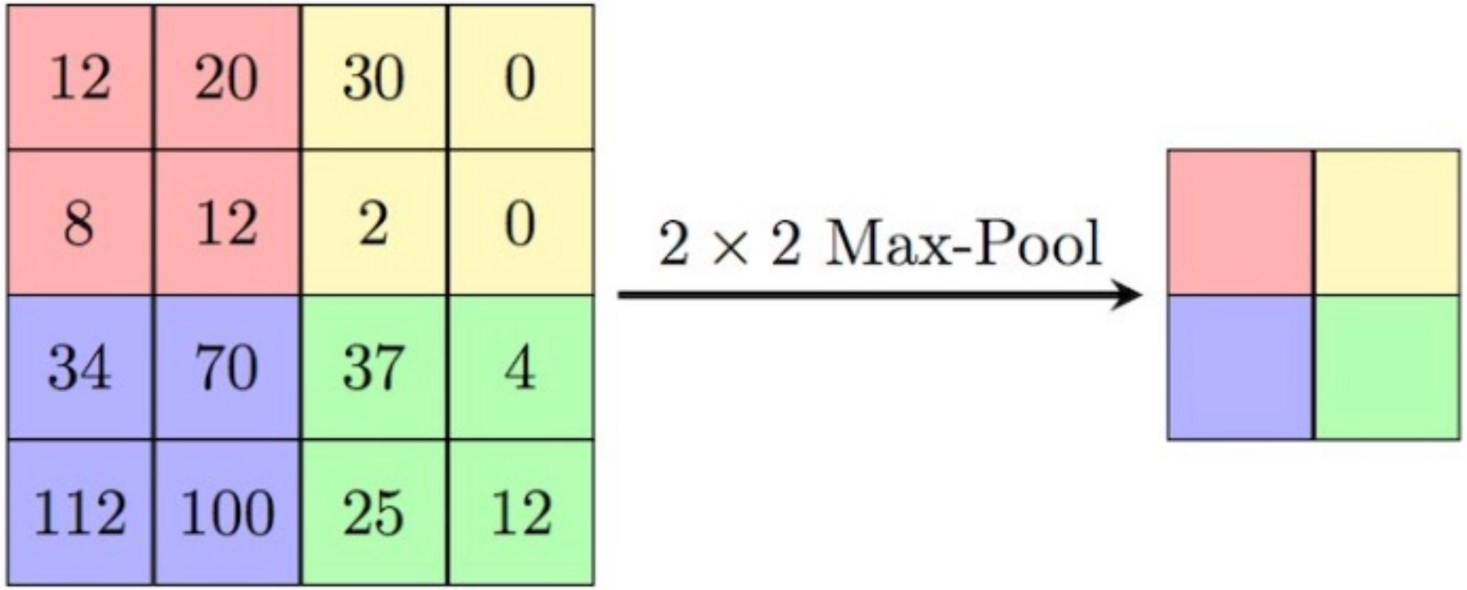
- Rarely padding is greater than zero
  - Downsampling
- No parameters, just hyperparameters
- Max Pooling
  - Outputs the largest number under the filter
  - It is a very effective and popular filter
  - Usually, the stride == kernel size ($s == f$)



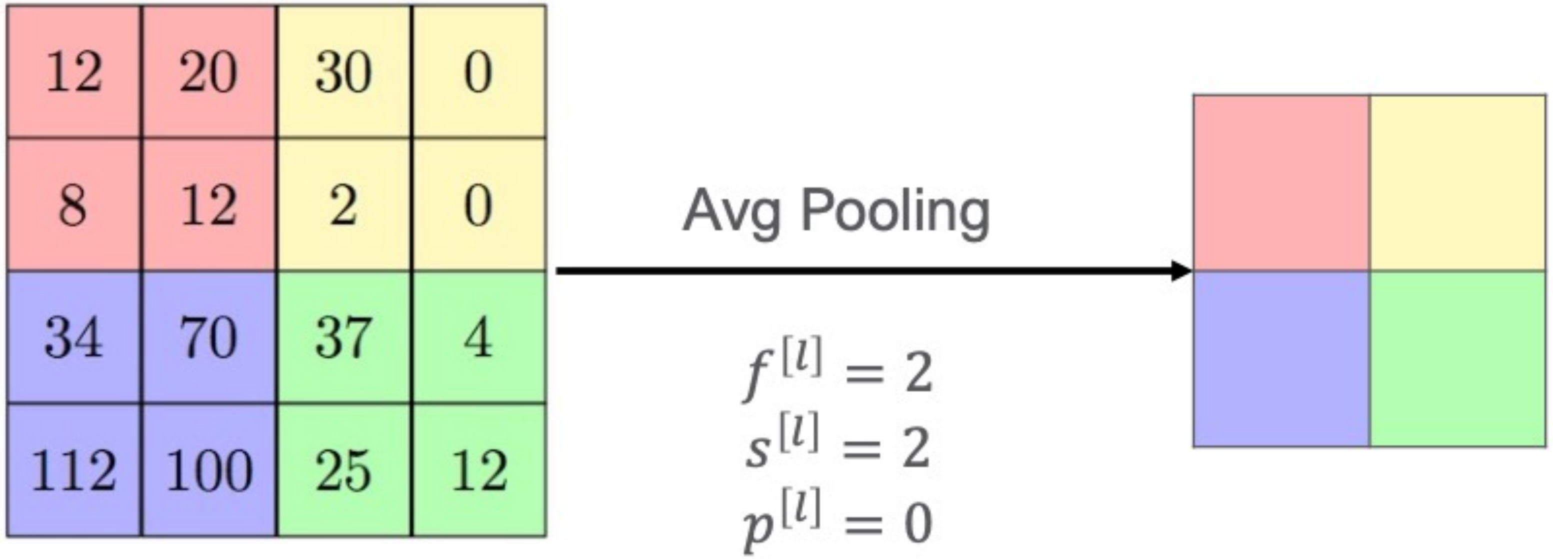
- Average Pooling
  - Outputs the average value of under the filter
  - Used on deeper layers to reduce the elements of the previous layer

Max Pooling

Typically, non-overlapping operations ($s = f$).



Average Pooling



THE UNIVERSITY OF TENNESSEE KNOXVILLE

Intuition

*

=

0	0	0	0	0	0	0
0	0					

Intuition

*

=

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	4	9	4	1	0
0	1	1	1	1	1	0

Intuition

*

=

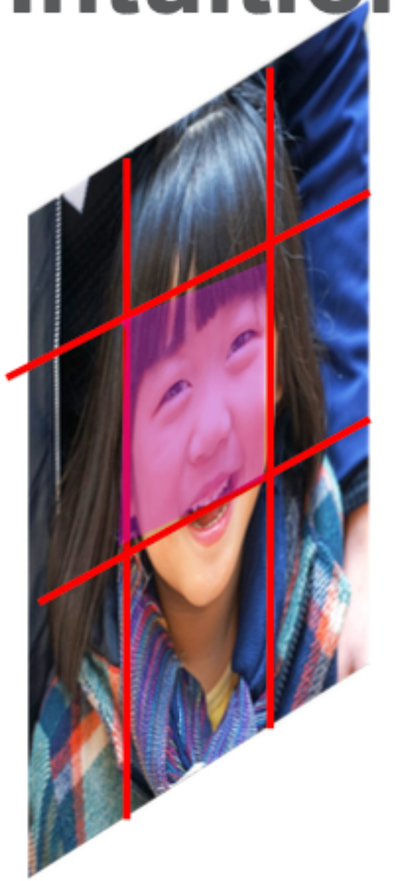
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	4	9	4	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Max-Pool

$$f^{[l]} = 2$$
$$s^{[l]} = 2$$
$$p^{[l]} = 0$$

0	1	1
1	9	4
0	0	0

Intuition



*

=

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	4	9	4	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Max-Pool

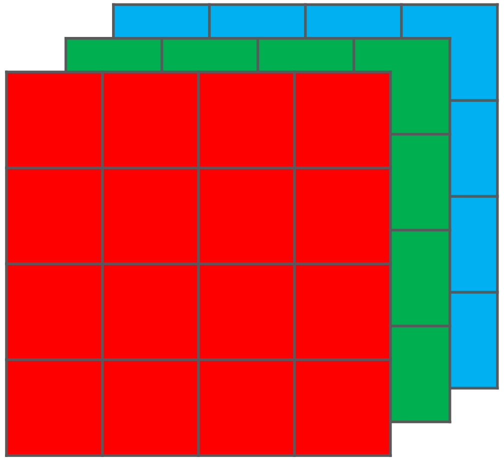
$f^{[l]} = 2$
$s^{[l]} = 2$
$p^{[l]} = 0$

0	1	1
1	9	4
0	0	0

Why CNNs?

THE UNIVERSITY OF TENNESSEE KNOXVILLE

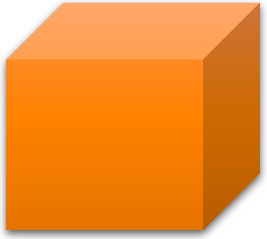
Number of parameters



$30 \times 30 \times 3$

$f.shape = (5,5,10)$

$a^{[1]}$



$26 \times 26 \times 10$ → 6,760 Responses

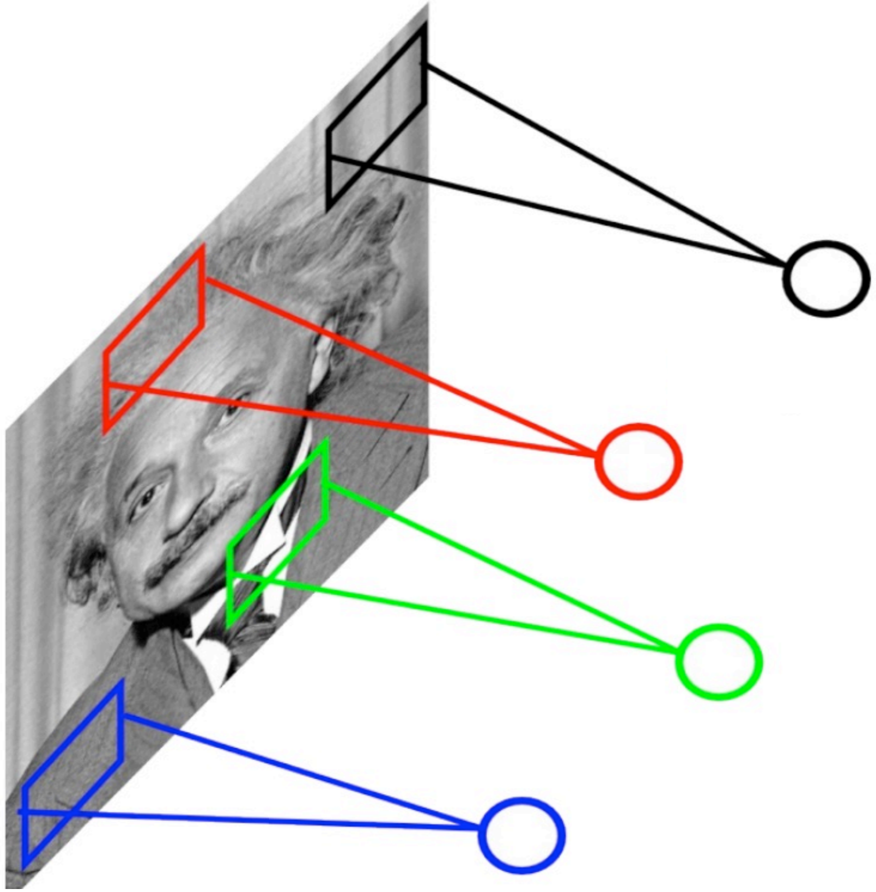
260 Parameters

$W^{[1]}.shape = (6760, 2700)$ → ~18M Parameters

Fully Connected Approach

$x_1, \ldots, x_{2700}$ → $a_1, \ldots, a_{6760}$

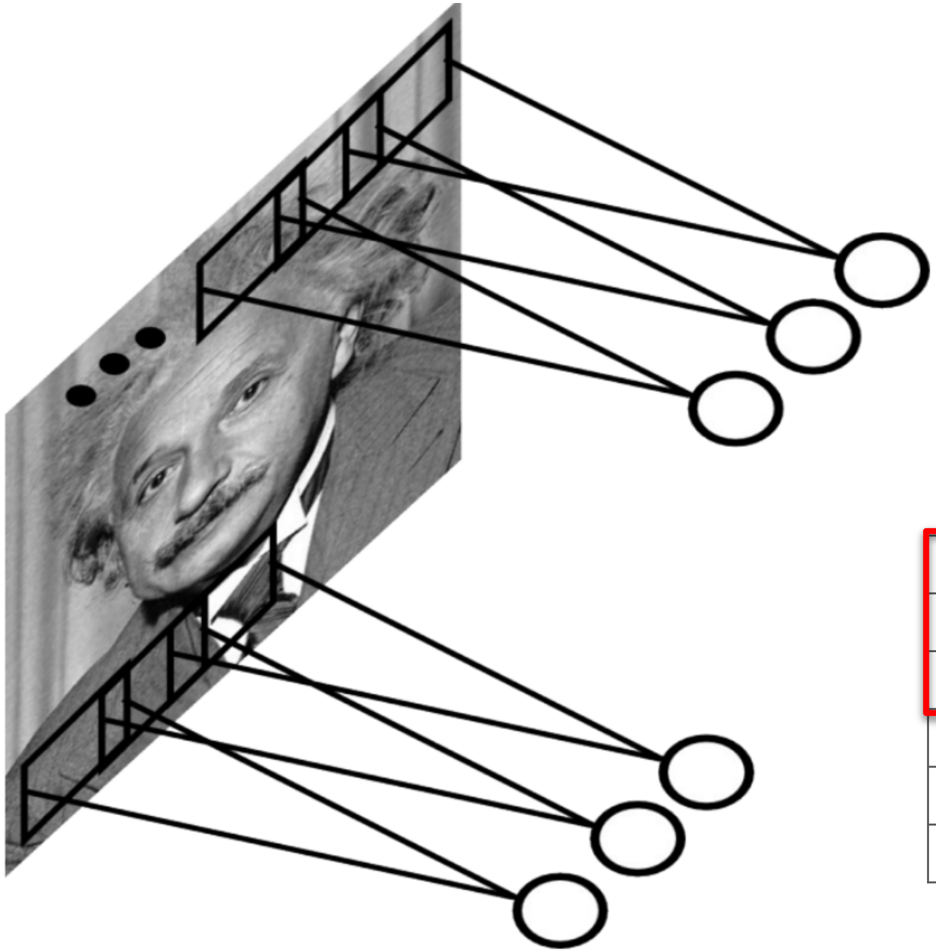
Use Local Regions (Sparsity of Connections)



The output for this window is only influenced by the pixels overlapping with the filter.

$$\begin{bmatrix} 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 15 & 15 & 0 \\ 0 & 15 & 15 & 0 \\ 0 & 15 & 15 & 0 \\ 0 & 15 & 15 & 0 \end{bmatrix}$$

Reuse The Same Kernel Everywhere



- Interesting features can happen anywhere in the image
- Share the same parameters across different locations
- Convolutions with learned kernels (i.e., filters)

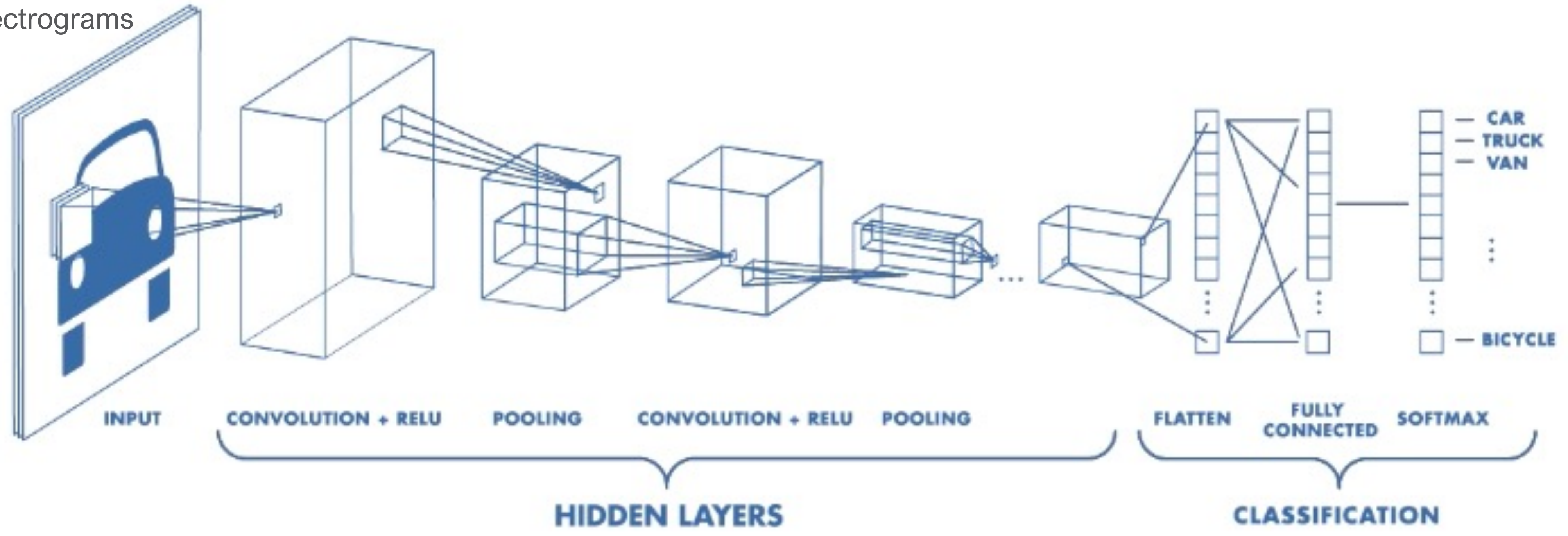
$$
\begin{bmatrix} 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 15 & 15 & 0 \\ 0 & 15 & 15 & 0 \\ 0 & 15 & 15 & 0 \\ 0 & 15 & 15 & 0 \end{bmatrix}
$$

Benefits of Sparsity and Reuse

- Uses less memory
- Needs less data
- Less prone to overfitting
- Built-in translation invariance

Convolutional Neural Networks

- Images
- 3D Objects
- DNA
- Tabular data
- Spectrograms
- Etc.



filter == kernel == neuron weights

Recap

- Locally connected (sparsity):
  - Each neuron is only connected to a few neurons in the previous layer. These are usually neurons that we expect will exhibit certain features
    - E.g., a neighborhood of pixels around a pixel in an image
- Shared weights:
  - Since we expect similar features to be present anywhere in the input, we would like these features to be detected everywhere. Therefore, we use neurons (i.e., filters) with shared weights.
- The neurons act as a feature detector:
  - Searching for certain local patterns across the input