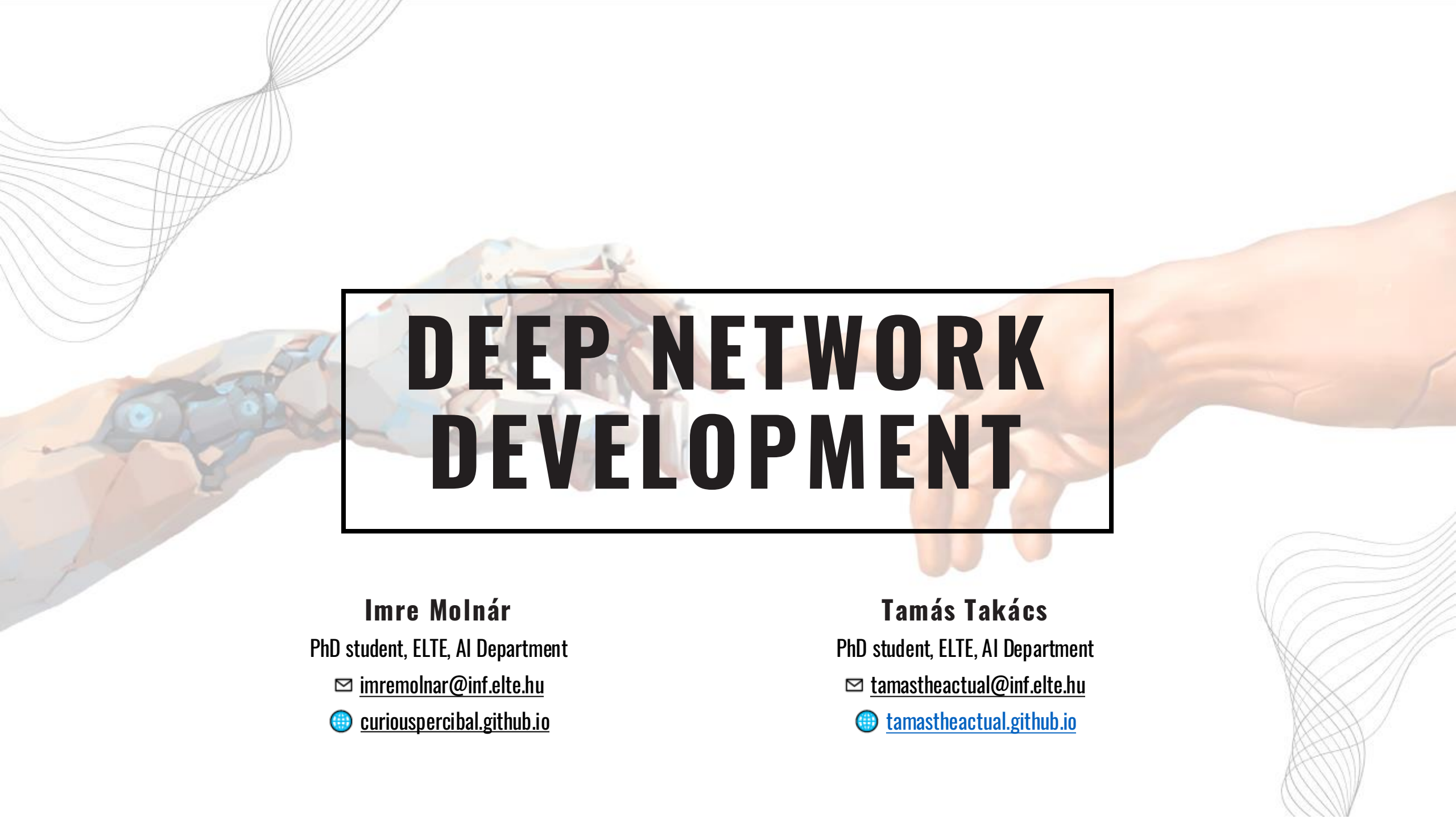


DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

imremolnar@inf.elte.hu

curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

tamastheactual@inf.elte.hu

tamastheactual.github.io

Deep Network Development

Lecture 3.

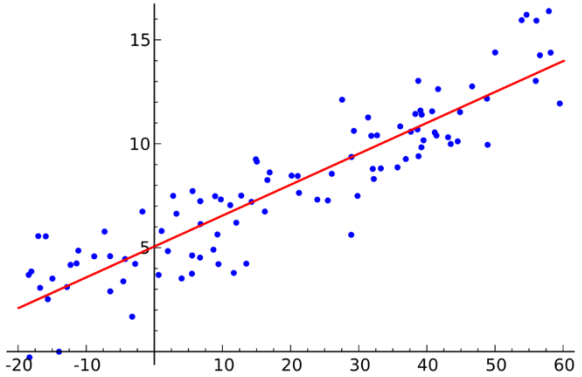
Image Classification & Convolutional Neural Networks

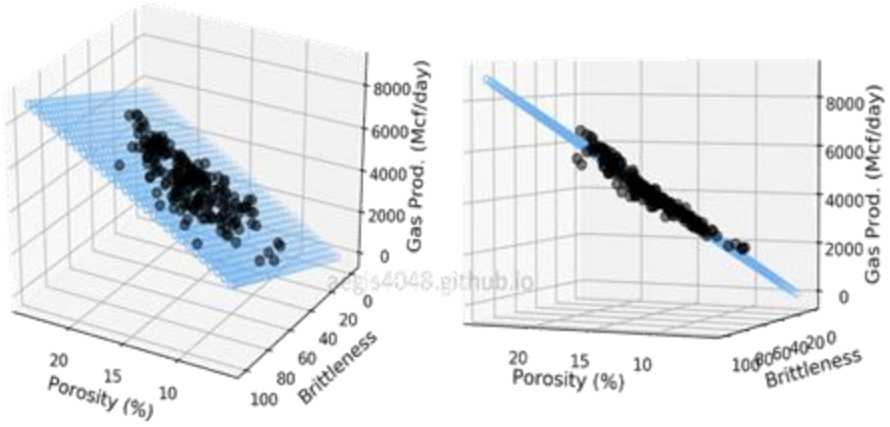
Budapest, 28th February 2025

1 Image Classification

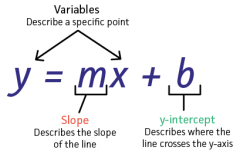
2 Convolutional Neural Networks

Previously on Lecture 2

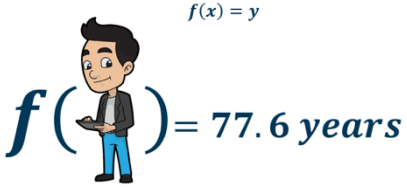


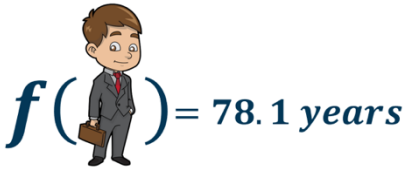


Use Linear Regression to predict Life Expectancy



Life Expectancy function
$f(x) = y$





$\hat{y} = \theta_1 x + \theta_0$ $\quad \hat{\mathbf{Y}} = X\theta$

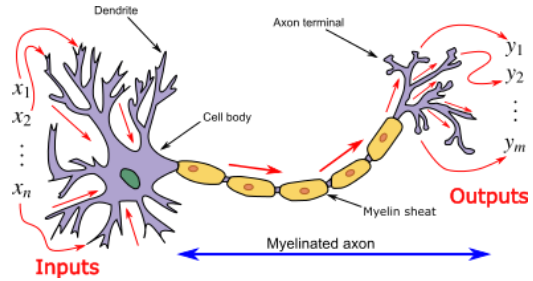
$\theta_1, \theta_0 = ?$

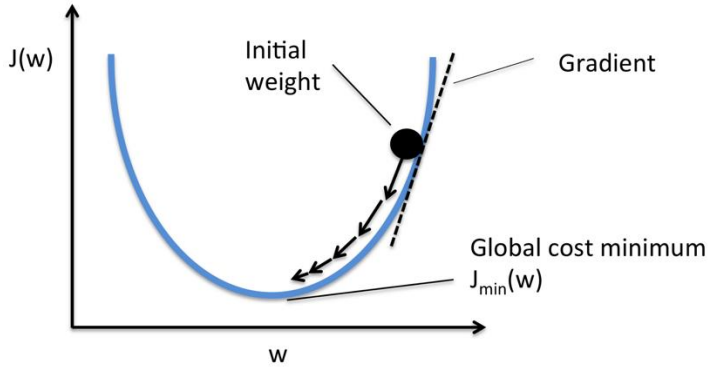
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_1 x^{(i)} + \theta_0 - y^{(i)}\right)^2$$

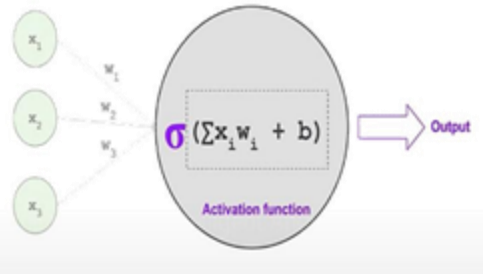
$$\boldsymbol{J(\theta) = (X\theta - Y)^T(X\theta - Y)}$$

Normal Equation:

$$\boldsymbol{\theta^* = (X^T X)^{-1}(X^T Y)}$$

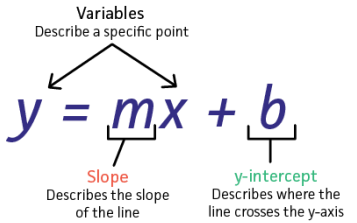


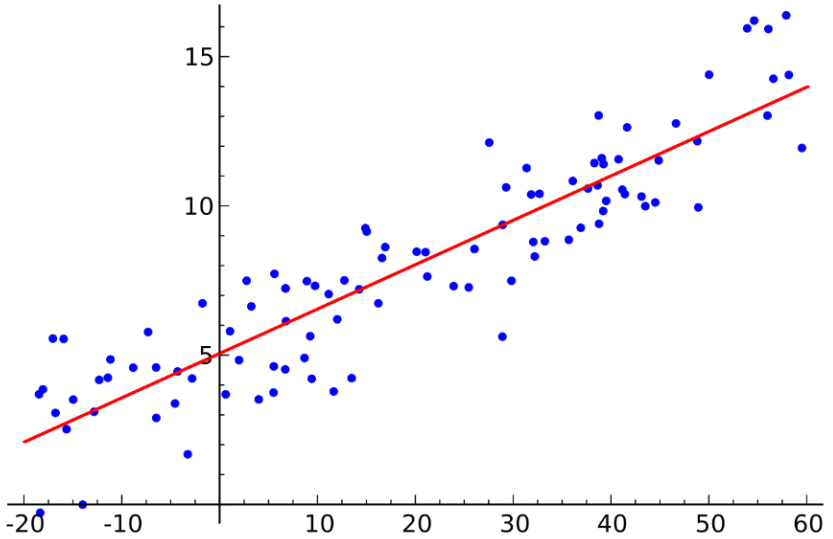




Linear Regression

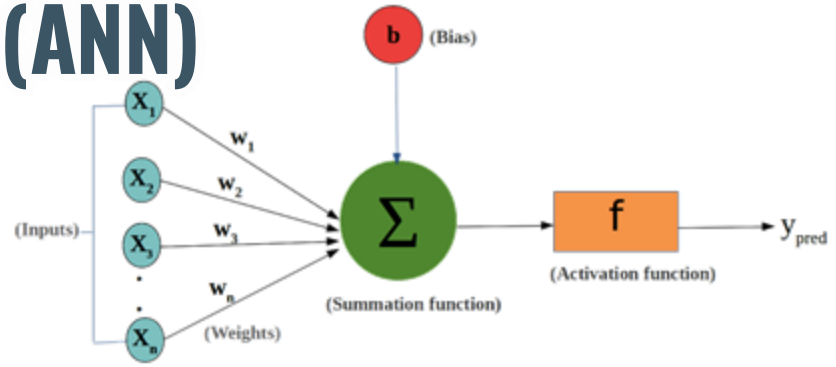
- Supervised learning
- Have: (x, y)
  - x – input
  - y – target
- *Goal: Learn a function to map* $\boldsymbol{x \rightarrow y}$.
  - $\boldsymbol{h(x)} = \hat{y}$
  - $\hat{y} = \boldsymbol{\theta_1 x + \theta_0}$
- Regression – Predict real-valued / continuous output:
  - $\hat{y} \in \mathbb{R}$

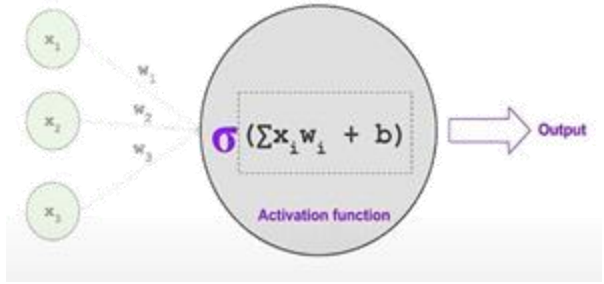




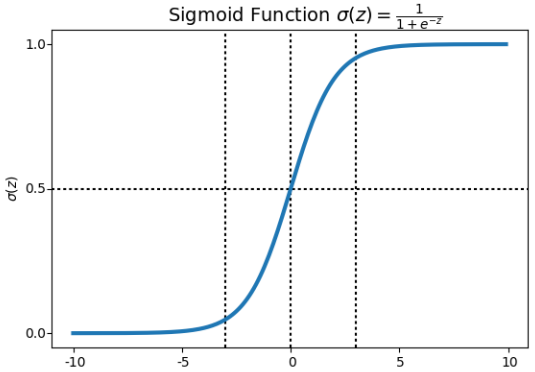
- Given the height and weight of a person, predict the age of the person.
- Age can be: 10,25,33,71,...

Artificial Neural Network (ANN)





- Using ANN for linear regression
- x_j – the inputs
- w_j – parameters we will train
- b – bias parameter
- **g – nonlinear activation function**
- o – the output
- One neuron with m inputs does the following:
  - $o = g\left(\sum_{j=0}^{m} w_j x_j + b\right)$



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

Activation function

- Adds non-linearity
- Output limited to a range (i.e. 0-1)
- Given the height and weight information, predict if it is a human or an animal.
- 0: human; 1: animal

Deep Network Development

Lecture 3.

Image Classification

Budapest, 28th February 2025

1 Image Classification

2 Convolutional Neural Networks

Linear Regression vs Classification

Linear Regression

- Supervised learning
- Have: (x, y)
  - x: input
  - y: target $\in \mathbb{R}$
- *Goal: Learn a function to map $x \rightarrow y$.*
  - $h(x) = \hat{y}$
  - $\hat{y} = \sum_{j=0}^{m} w_j x_j + b$
- Regression – Predict real-valued / continuous output:
  - $\hat{y} \in \mathbb{R}$

Binary Classification

- Supervised learning
- Have: (x, y)
  - x: input
  - y: target $\in \{0, 1\}$
- *Goal: Learn a function to map $x \rightarrow y$.*
  - $h(x) = \hat{y}$
  - $\hat{y} = g\left(\sum_{j=0}^{m} w_j x_j + b\right)$ Add non-linear activation function
- Classification – Predict discrete set of values:
  - $\hat{y} \in \{0, 1\}$

Binary Classification

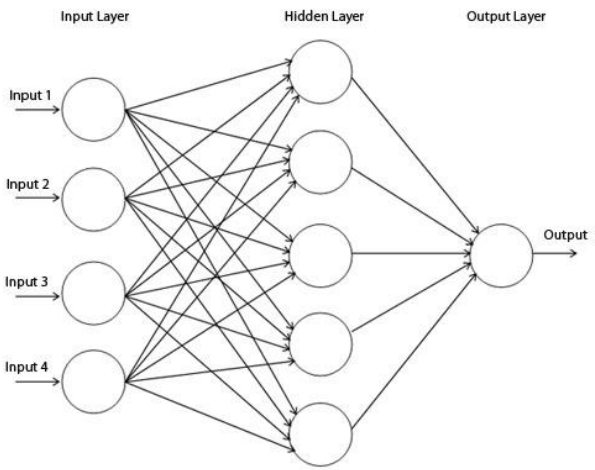
Binary classification – which of the two classes does the input belong to?

- Cat vs Dog
- Dog vs Mop

Binary Classification

Binary Classification

- Binary classification – Two classes
- Is x in the target class C?
- Output $P(x \in C)$, the probability of x is in C
- The network will have 1 output
- h(x) must be between 0 and 1

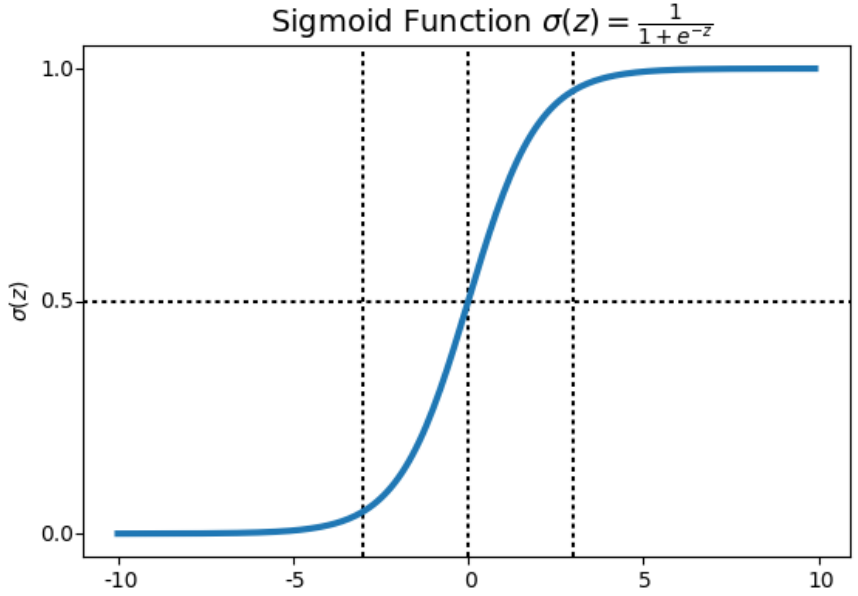


Dog vs Mop

Binary Classification

- h(x) must be between 0 and 1
- Use the sigmoid activation on last layer

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



- Linear regression becomes Logistic Regression (Binary Classification), if we add a Sigmoid activation function to the output
- Linear Regression: $\boldsymbol{h}(\boldsymbol{x}) = Wx + b$ (continuous output)
- Binary Classification: $\boldsymbol{h}(\boldsymbol{x}) = g(Wx + b)$(discrete output between 0 and 1)

Linear Regression

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

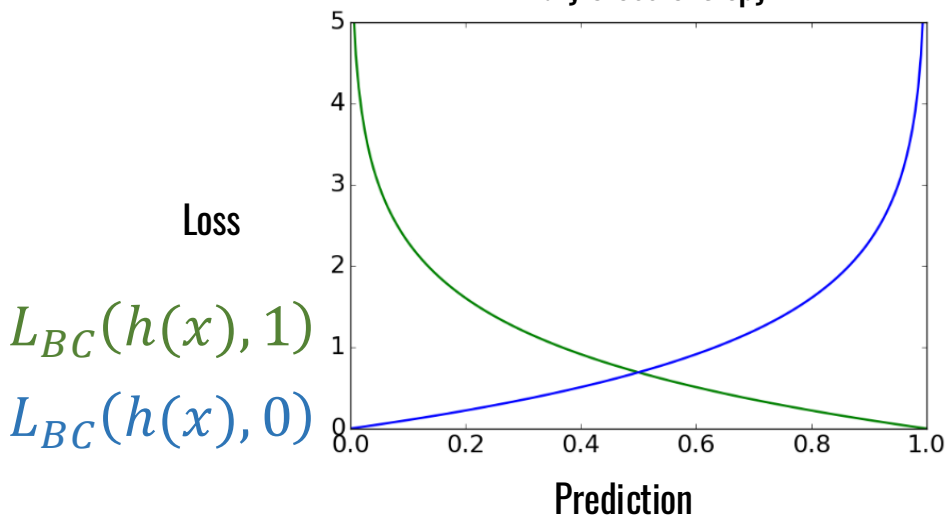
Binary Classification Loss

- Binary cross-entropy loss

$$L_{BC}(h(x), y) = -y \log h(x) - (1 - y) \log(1 - h(x))$$

- Assumptions:
  y is 0 or 1
  h(x) = ŷ is between 0 and 1

Binary cross-entropy

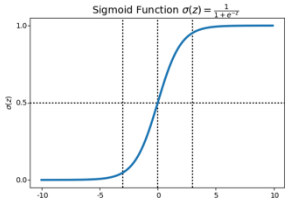


$$L_{BC}(h(x), y) = -y \log h(x) - (1 - y) \log(1 - h(x))$$
$$L_{BC}(1,1) = -1 \log 1 - (1 - 1) \log(1 - 1)$$
$$L_{BC}(1,1) = 0 - (0) \log(0)$$
$$L_{BC}(1,1) = 0$$

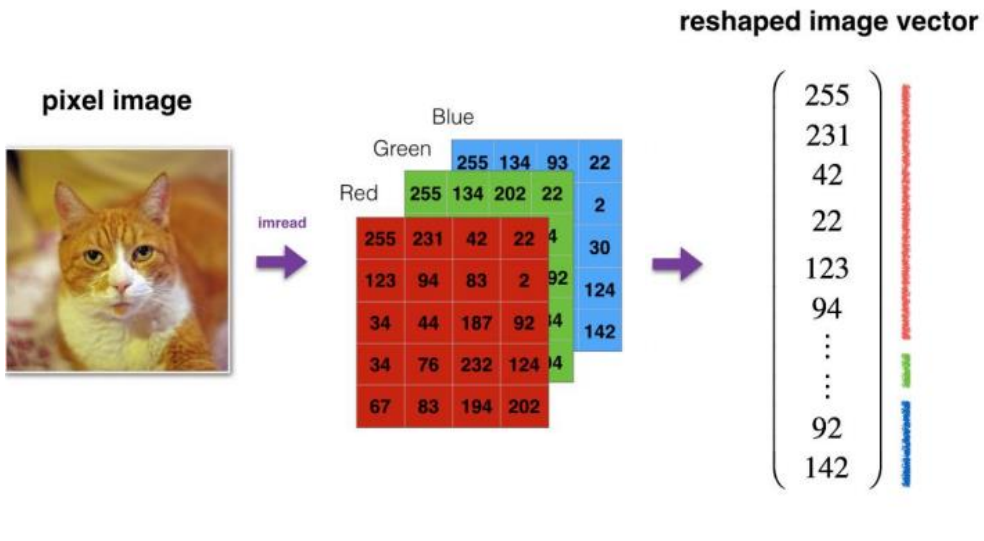
log(0) is undefined -> clipping

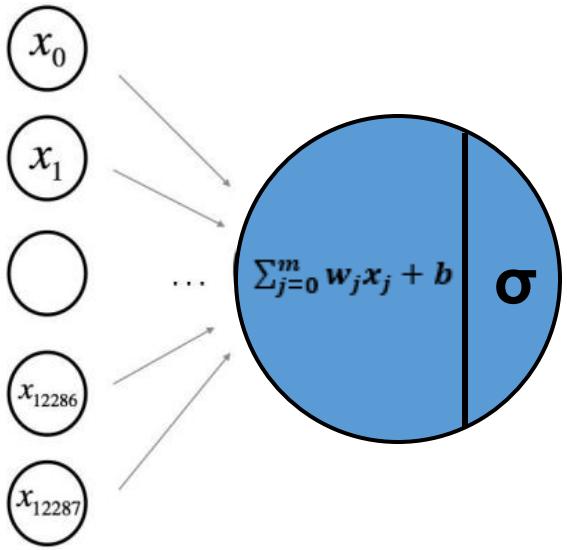
Binary Classification example

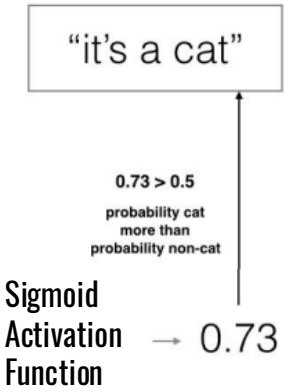
Classes C = {0: Dog, 1: Cat}



INPUT **NEURAL NETWORK** **OUTPUT** **TARGET**







Y = 1

LOSS

$$L_{BC}(h(x), y) = -y \log h(x) - (1 - y) \log(1 - h(x))$$
$$L_{BC}(0.73, 1) = -1 \log 0.73 - (1 - 1) \log(1 - 0.73)$$
$$L_{BC}(0.73, 1) = -1 * -0.13667714 - (0) \log(0.23)$$
$$L_{BC}(0.73, 1) = 0.13667714$$

$$\boldsymbol{L_{BC}(0.73, 1) = 0.14}$$

More neurons and categories

Next

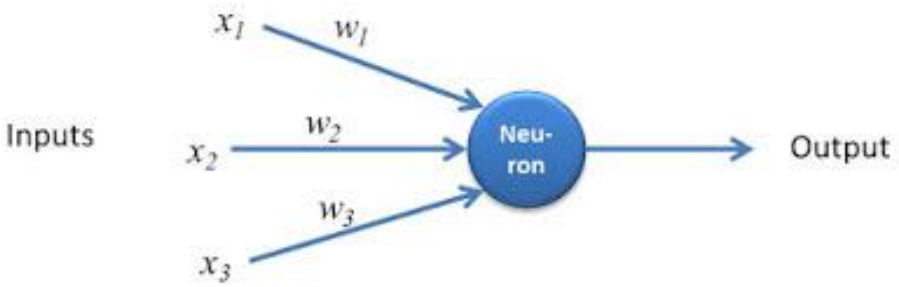
- Single artificial neuron
- Binary Classification (0 or 1)

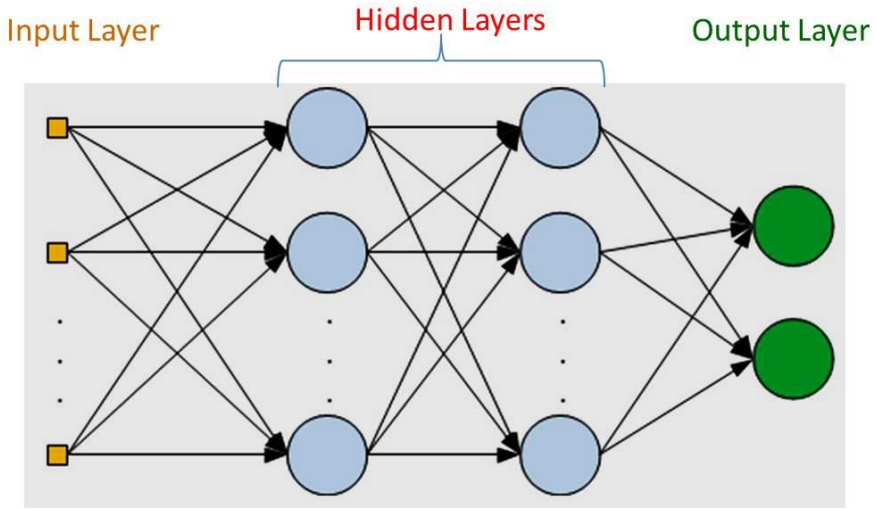
→

- Deep Neural Network
- Multi-Class Classification (0,1,2,3…)

Deep Neural Networks

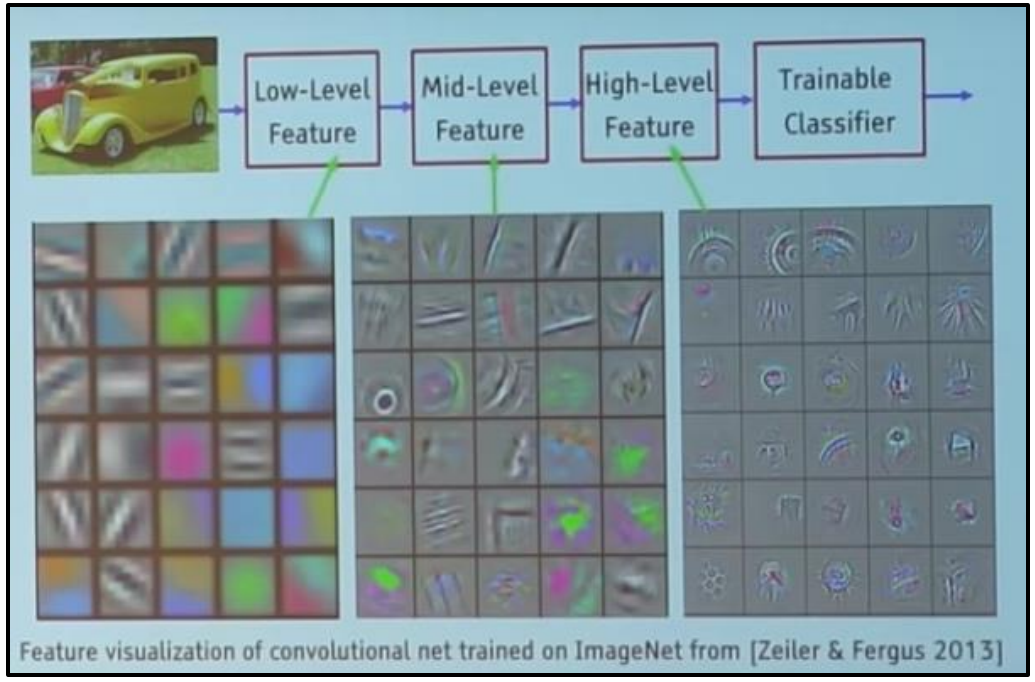
- Neural networks are built up from neurons, that have inputs and outputs
- Neurons are organised into layers
- Layers refine the output of the previous layers

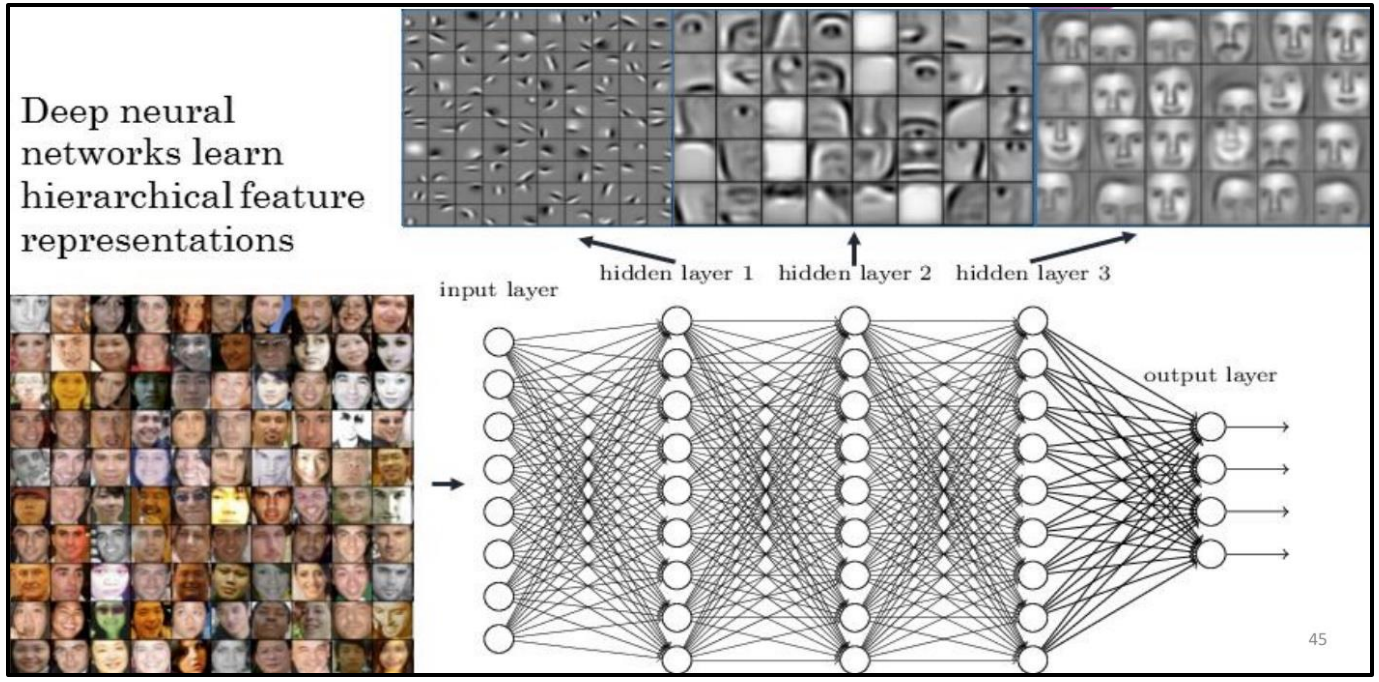




Deep Neural Networks

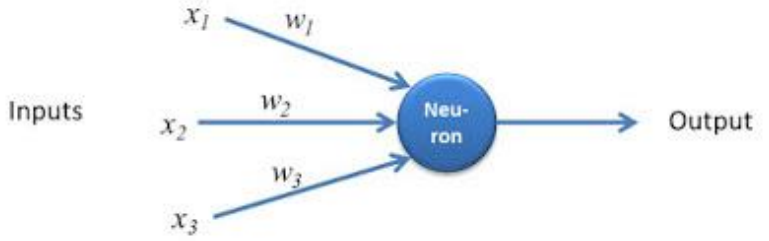
- Each layer refines the previous layer
- Visualization demo: https://adamharley.com/nn_vis/



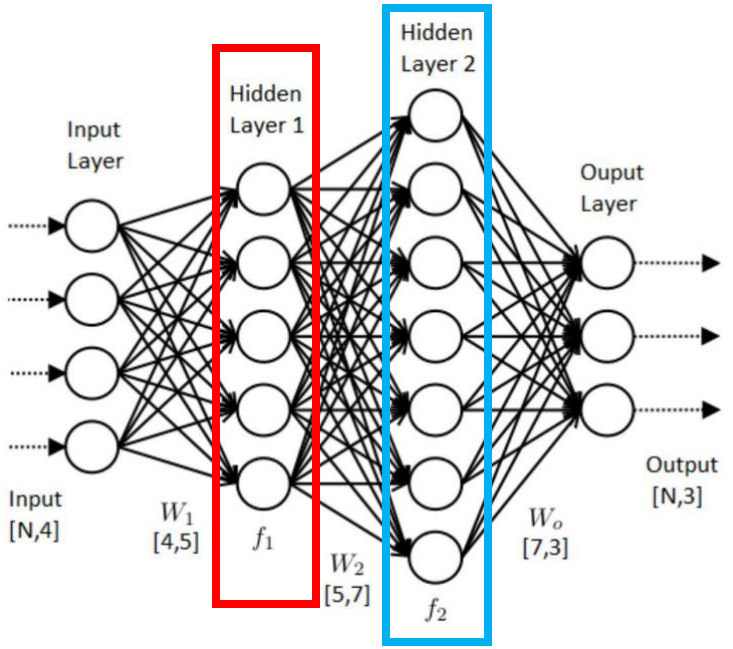


Deep Neural Networks

- Artificial Neural Networks ⟺ Feed Forward Neural Networks ⟺ Fully Connected Networks



- A neural network is built up from neurons
- Neurons are organised into layers
- Layers refine the output of the previous layers
  - $\boldsymbol{h_1(x)} = g\left(W^{(1)}x + b^{(1)}\right)$
  - $h_2(x) = g\left(W^{(2)}\boldsymbol{h_1} + b^{(2)}\right)$
  - $h_2(x) = g\left(W^{(2)}\boldsymbol{g}\left(\boldsymbol{W^{(1)}x} + \boldsymbol{b^{(1)}}\right) + b^{(2)}\right)$



Deep Neural Networks [1, 2]

- How to find *W* and *b*?
- Given $h = g\left(W^{(2)}\boldsymbol{g}\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + b^{(2)}\right)$

- Select a loss function L that measures how good *h* is:
  - the smaller L(h(x),y) , the better *h* is

- Update $\boldsymbol{W}^{(1)}$:
  $$\boldsymbol{W_t}^{(1)} = \boldsymbol{W_{t\_1}}^{(1)} - \alpha\nabla J(\boldsymbol{W}^{(1)})$$
- Update $\boldsymbol{W}^{(2)}$:
  $$\boldsymbol{W_t}^{(2)} = \boldsymbol{W_{t\_1}}^{(2)} - \alpha\nabla J(\boldsymbol{W}^{(2)})$$
- Update $\boldsymbol{b}^{(1)}$:
  $$\boldsymbol{b_t}^{(1)} = \boldsymbol{b_{t\_1}}^{(1)} - \alpha\nabla J(\boldsymbol{b}^{(1)})$$
- Update $\boldsymbol{b}^{(2)}$:
  $$\boldsymbol{b_t}^{(2)} = \boldsymbol{b_{t\_1}}^{(2)} - \alpha\nabla J(\boldsymbol{b}^{(2)})$$

Terminology
- L(h(x),y) – loss or error function for a single data point of the dataset
- C(θ), J(W) – cost or objective function for the entire dataset (usually average)
- The update rule can also be in the form:

$$w_i \leftarrow w_i + \Delta w_i$$

$$b \leftarrow b + \Delta b$$

$$\Delta w_i = -\alpha\frac{\partial L}{\partial w_i}$$

$$\Delta b = -\alpha\frac{\partial L}{\partial b}$$

[1] Backpropagation derivation: https://www.cs.put.poznan.pl/pliskowski/pub/teaching/eio/lab1/eio-supplementary.pdf
[2] Another backpropagation derivation (be aware of notation differences): https://www.cs.swarthmore.edu/~meeden/cs81/s10/BackPropDeriv.pdf

Deep Neural Networks

1. Select W_0 randomly
2. Calculate $h(X)$ for all training examples
3. Calculate the cost:

$$J(W) = \frac{1}{N}\sum_{i=1}^{N} L(h(x_i), y_i)$$

4. Update W:

$$W_t = W_{(t-1)} - \alpha \nabla J(W)$$

5. Repeat steps 2-4 until convergence

Deep Neural Networks

1. Select W_0 randomly
2. Calculate $h(X)$ for all training examples
3. Calculate the cost function:

$$J(W) = \frac{1}{N}\sum_{i=1}^{N} L(h(x_i), y_i)$$

4. Update W:

$$W_t = W_{(t-1)} - \alpha \nabla J(W)$$

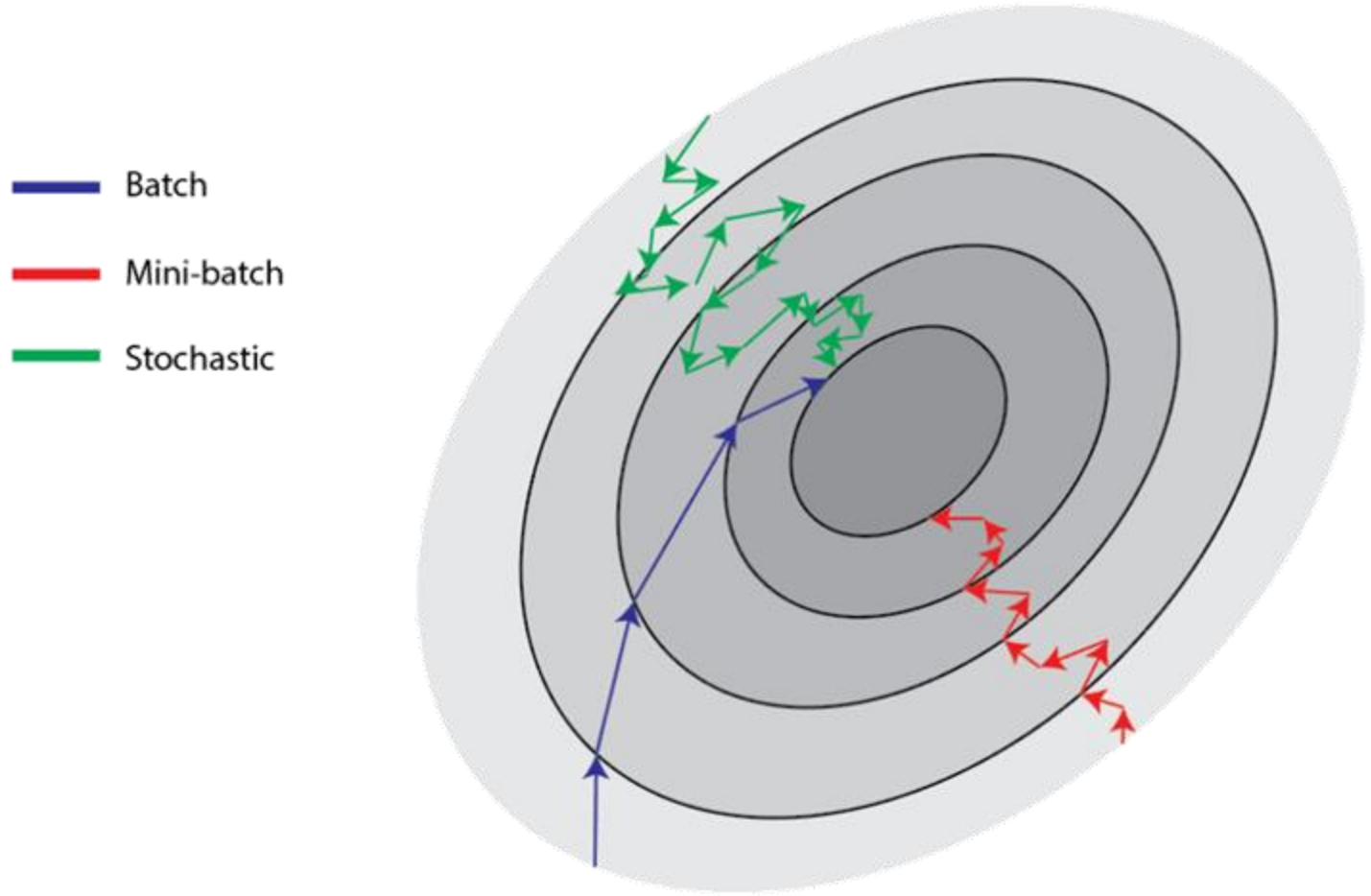
5. Repeat steps 2-4 until convergence

Batch size – the number of training samples used to optimize the model's parameters

Types (with Gradient Descent)

- **Batch Gradient Descent**
  - Batch size = Size of dataset
- **Mini-Batch Gradient Descent**
  - Batch size = a subset of the dataset examples; typically, 2^n , e.g., 8,16,32,128,...
  - In each iteration, a mini-batch is randomly sampled from the dataset
- **Stochastic Gradient Descent**
  - Batch size = 1
  - In each iteration, a single example is randomly sampled from the dataset

Training Deep Neural Networks



Hyperparameter vs Parameter

Hyperparameters

To create and train Neural Networks we must make certain decisions (we choose the values):

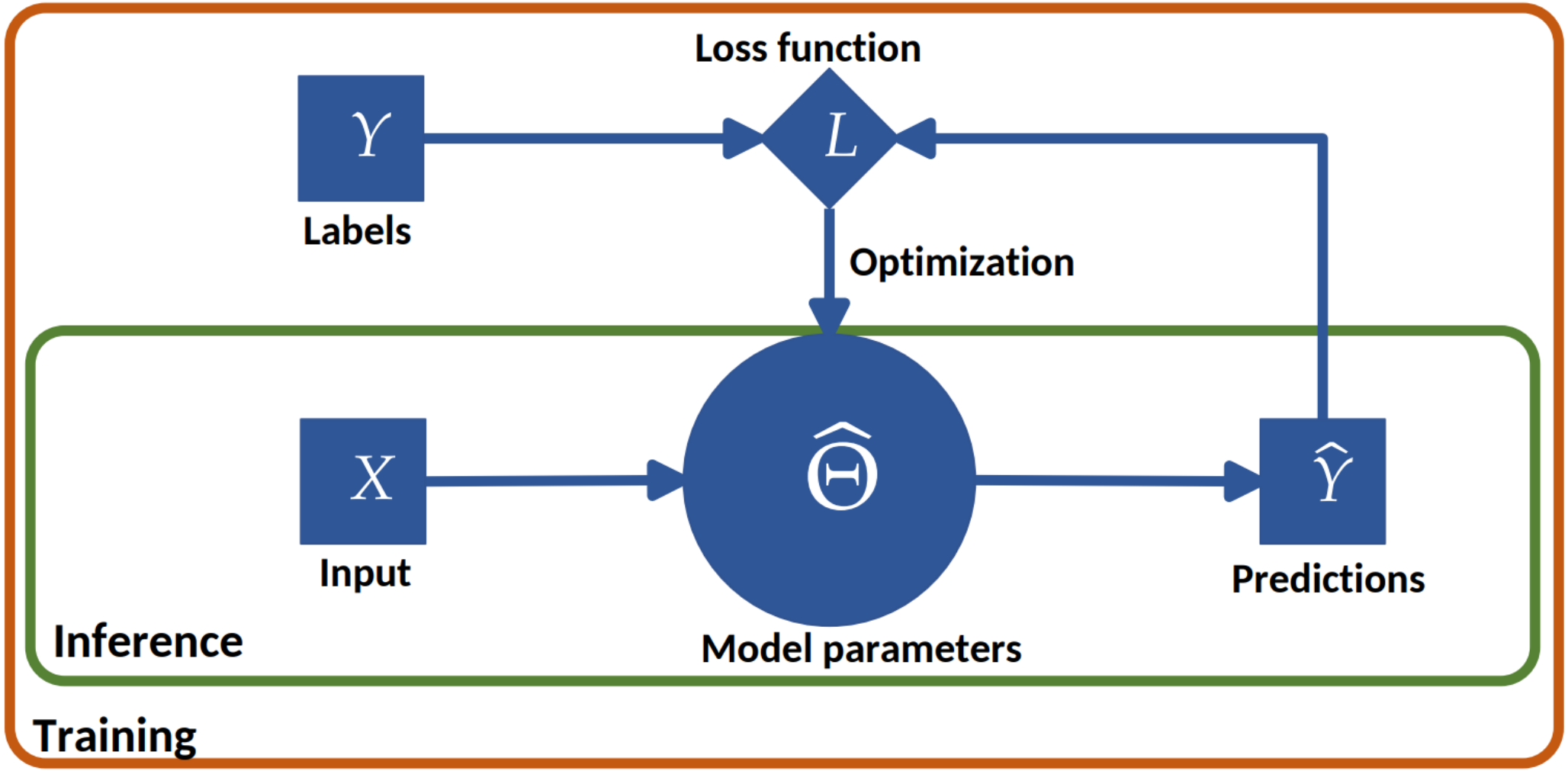
- Number of layers; Number of neurons in each layer
- Activation functions
- Learning rate
- Batch size
- Many more (optimization related)

Parameters

What the Neural Network learns. The weights and biases of the network are optimized with an algorithm (Gradient Descent, etc.). We do not choose the values for them.[3]

3. We can choose the initial values (weight initialization), it can be initialized with zeros, ones, random values, etc. However, the network learns the optimal values through training.

Deep Neural Networks



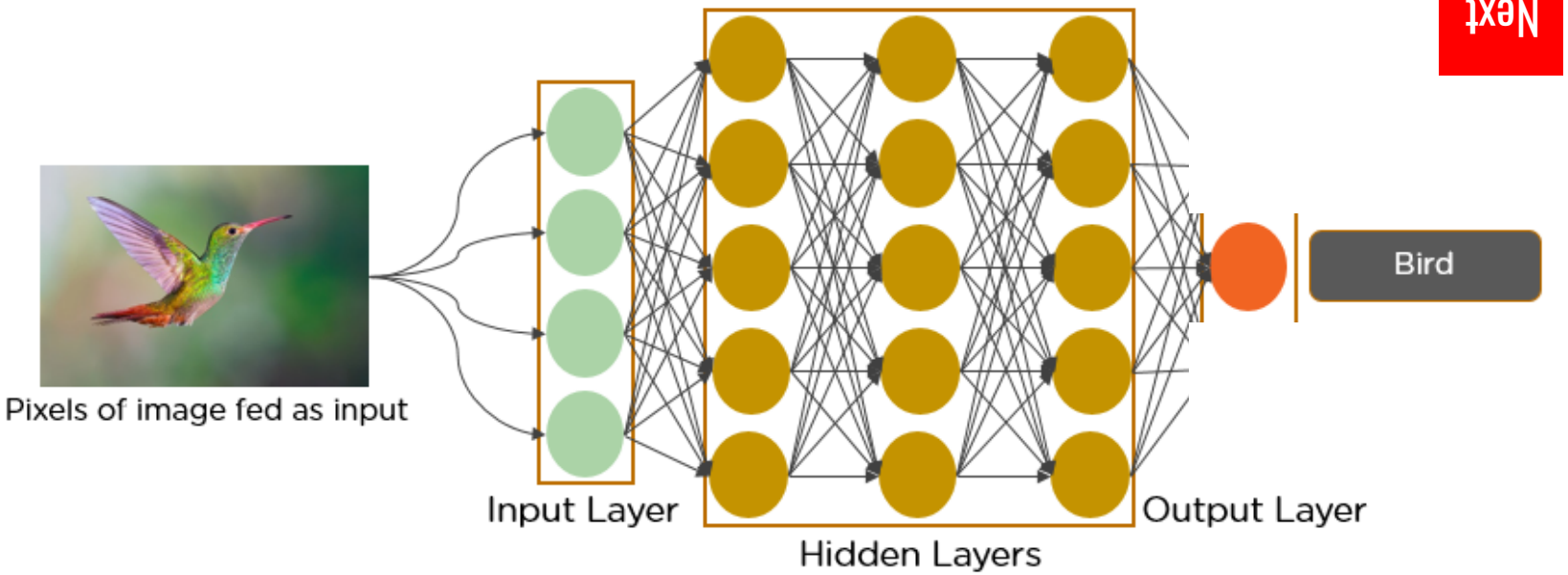
More neurons and categories

- Single artificial neuron
- Binary Classification (0 or 1)

→

- Deep Neural Network
- Multi-Class Classification (0,1,2,3…)

Next

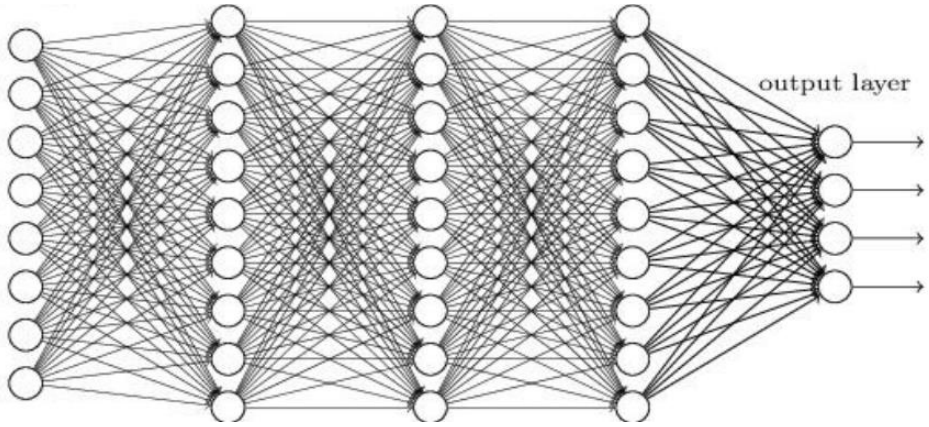


Multi-Class Classification

- With *K* classes
- The network will have K outputs
- Softmax activation for squashing outputs between 0 and 1

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

This is basically a probability distribution on the classes.
They sum up to 1.

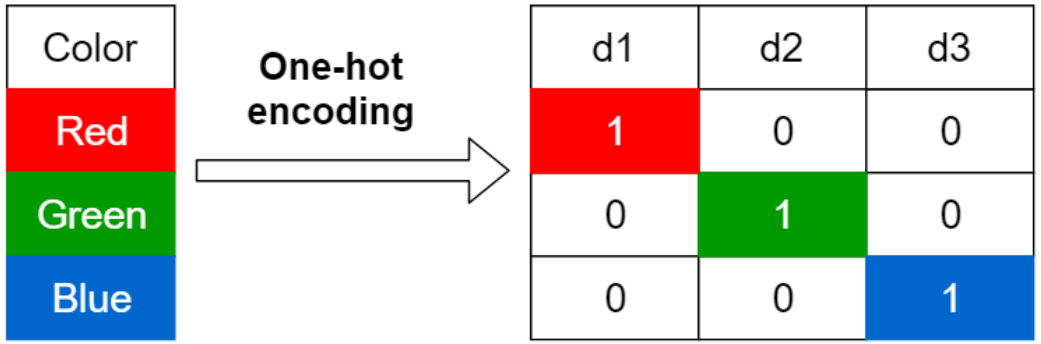


Multi-Class Classification

Output of the neural network is a K long vector

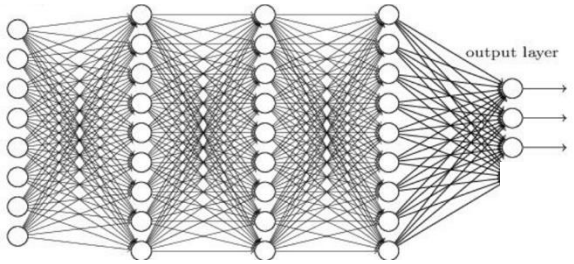
How should we encode the ground-truth values?
One-hot encoding (K=3):
(cat) 1 -> [1, 0, 0]
(dog) 2 -> [0, 1, 0]
(horse) 3 -> [0, 0, 1,]

Just as h(x): values between 0 and 1, sum up to 1

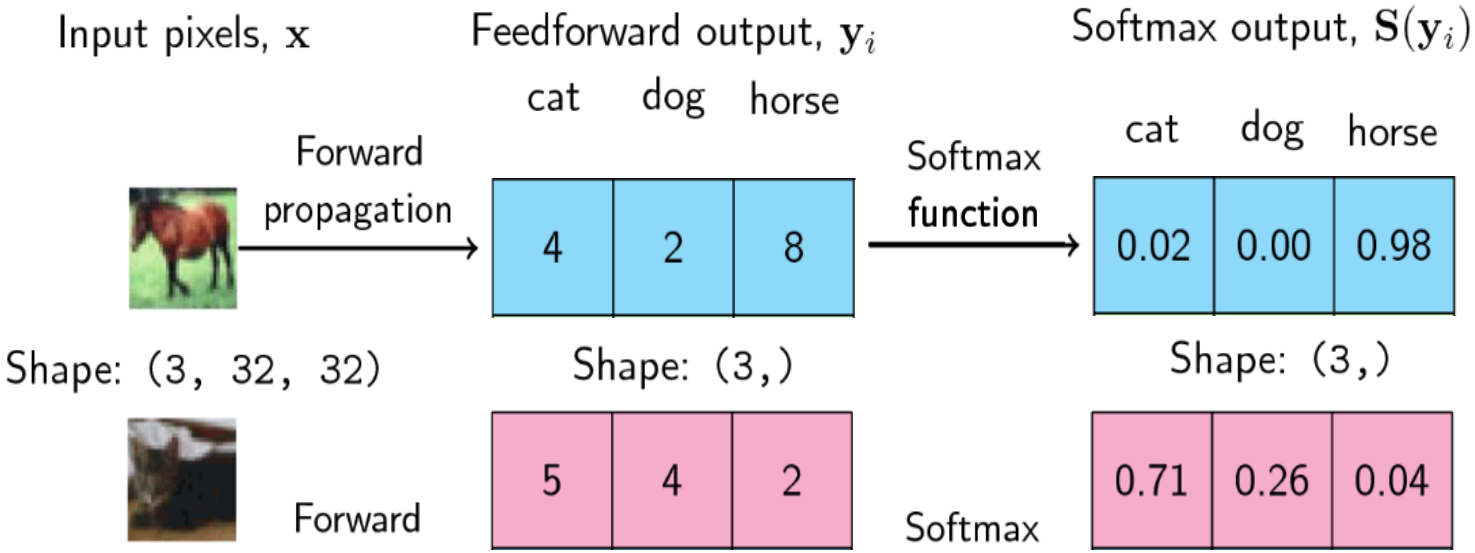


Multi-Class Classification

- K = 3
- Softmax activation $\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$
- Probability distribution on the classes. They sum up to 1.



(Cat, Dog, Horse)



Multi-Class Classification Loss

With *K* classes
Categorical Cross-entropy loss

$$L(h(x), y) = -\sum_{i=1}^{K} y_i \log h_i(x)$$

Assumptions:

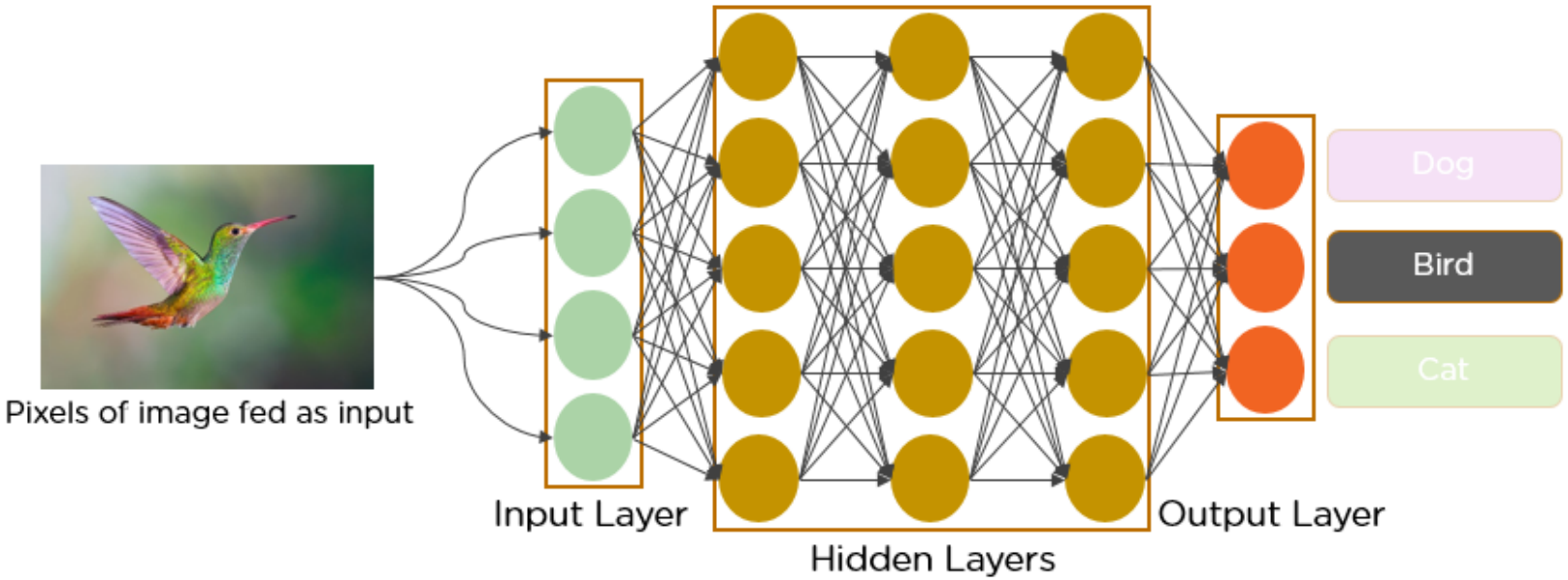
- *y* is a vector of *K elements with values* 0 or 1, exactly one element is 1
- $h_i(x)$ is a vector of K elements between 0 and 1, the sum of all elements is equal to 1

More neurons and categories

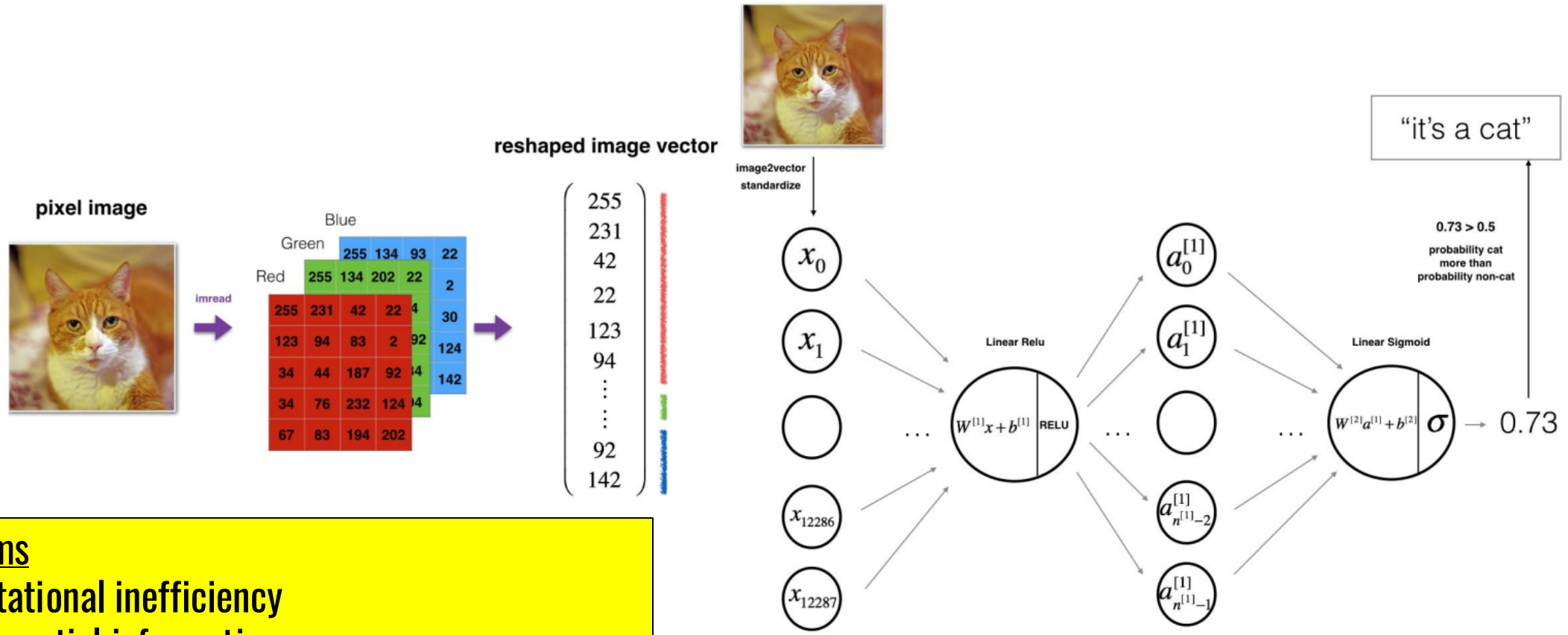
- Single artificial neuron
- Binary Classification (0 or 1)

→

- Deep Neural Network
- Multi-Class Classification (0,1,2,3...)



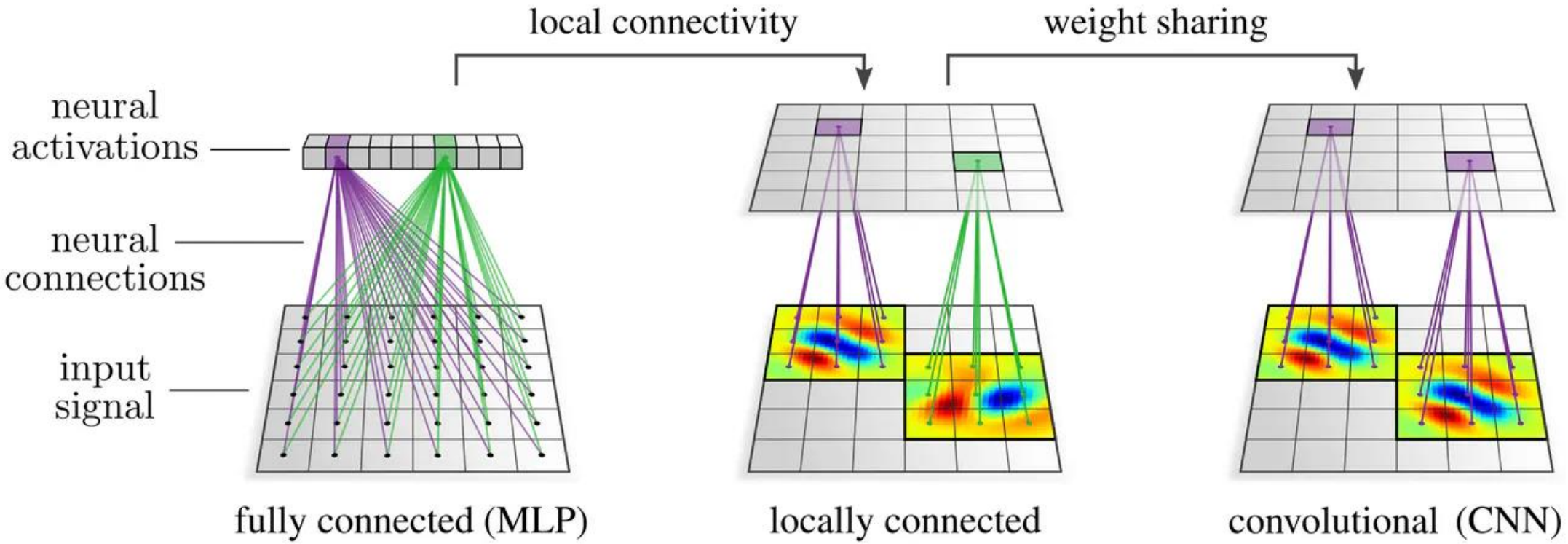
Disadvantages of Feed Forward Neural Networks



Problems

Computational inefficiency
Loss of spatial information
Independent weights/features

Disadvantages of Feed Forward Neural Networks (Intro to CNNs)



Deep Network Development

Lecture 3.

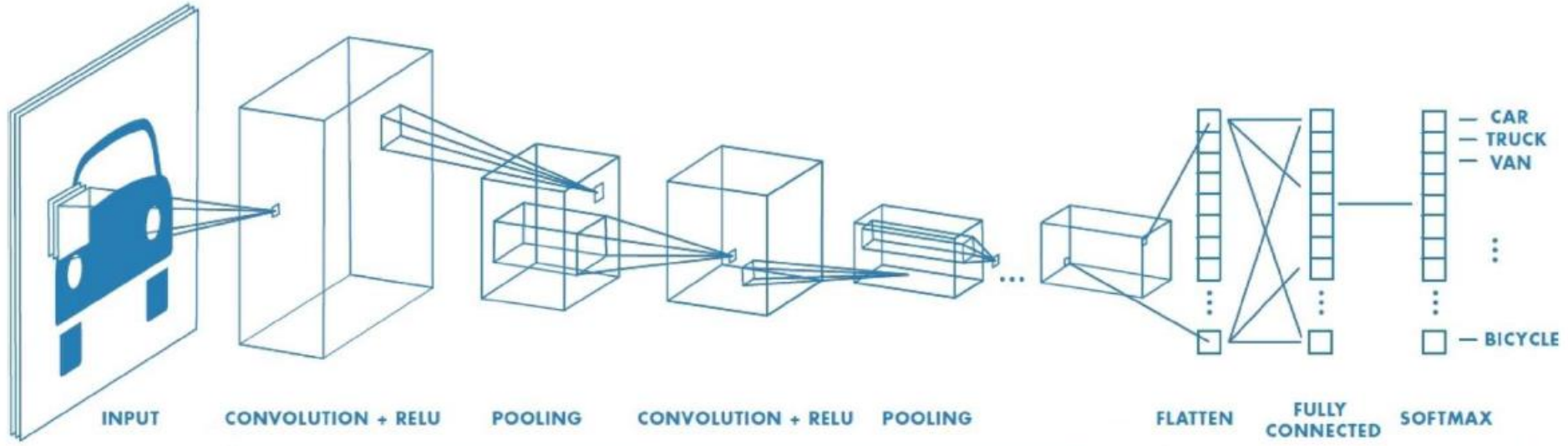
Convolutional Neural Networks

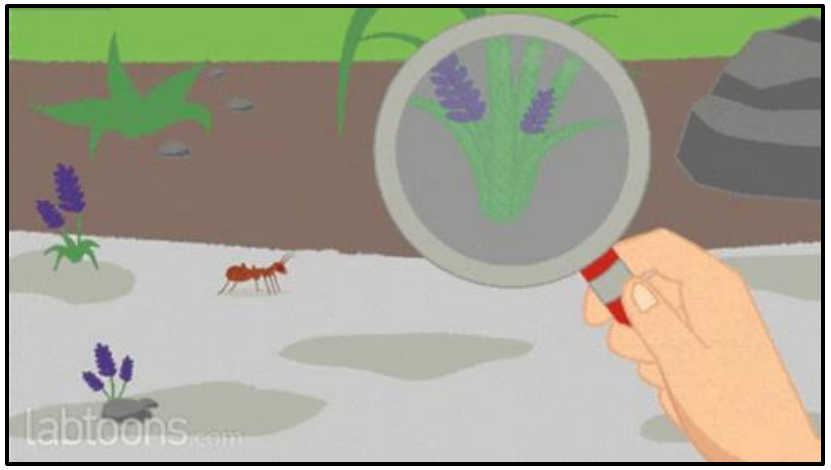
Budapest, 24th September 2024

1 Image Classification

2 Convolutional Neural Networks

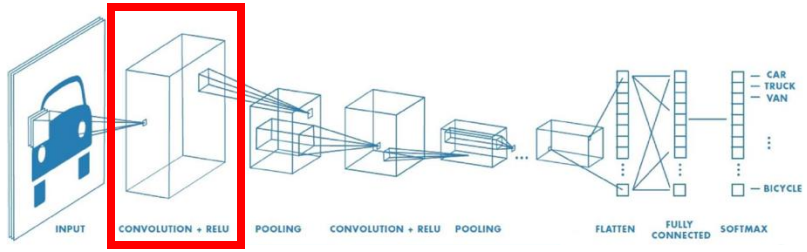
How do Convolutional Neural Networks (CNNs) work?





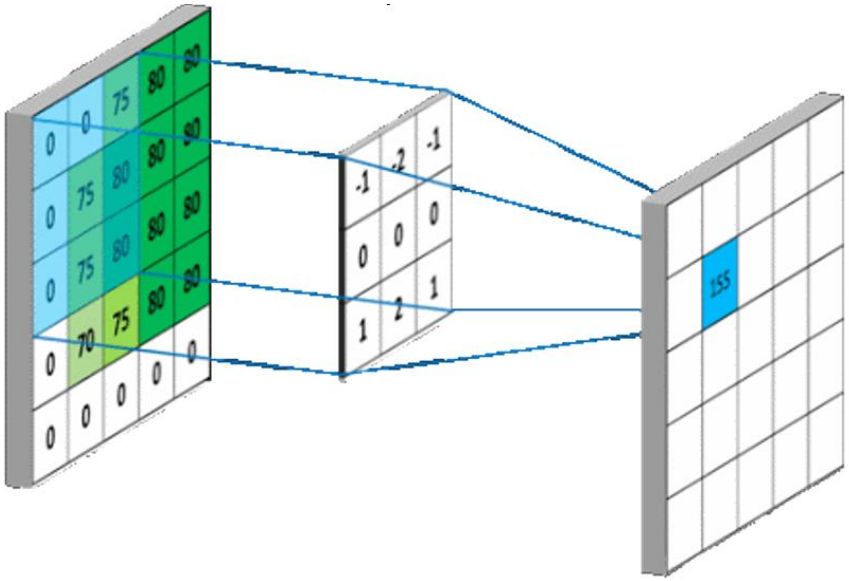
CNNs extract relevant features from an image.
Like a magnifying glass “enhancing” details of the image.

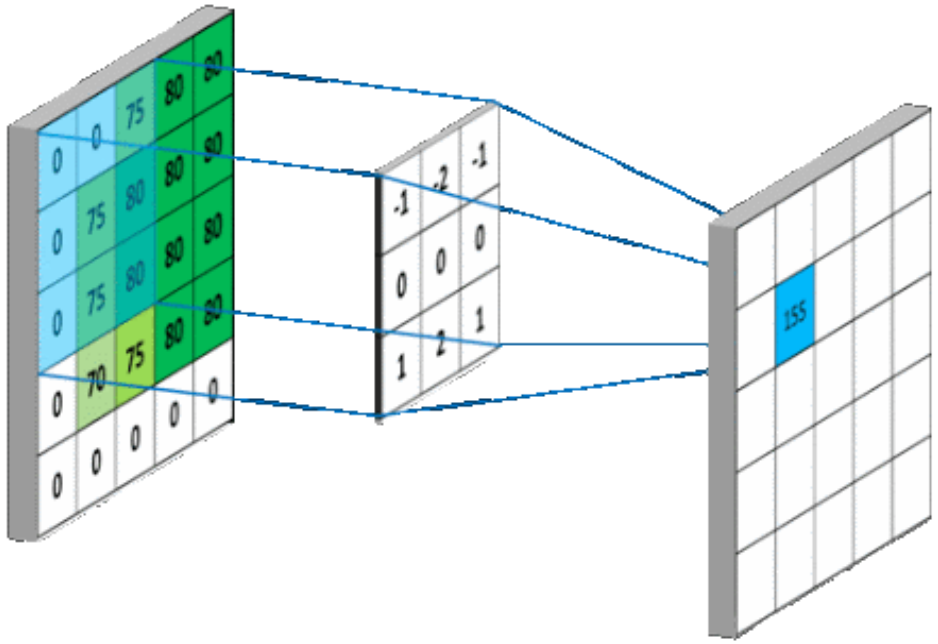
Convolutional layer



Convolution
Input: 5x5
Filter: 3x3
Output: 3x3

Filter / Kernel size = f x f





Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3
- Output = ?

10	2	4	1	1
0	4	0	2	2
1	8	16	0	0
8	1	1	0	0
0	0	2	6	1

*

2	2	2
1	0	1
2	2	2

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10 x 2	2 x 2	4 x 2	1	1
0 x 1	4 x 0	0 x 1	2	2
1 x 2	8 x 2	16 x 2	0	0
8	1	1	0	0
0	0	2	6	1

$$10 \times 2 + 2 \times 2 + 4 \times 2 + 0 \times 1 + 4 \times 0 + 0 \times 1 + 1 \times 2 + 8 \times 2 + 16 \times 2 = 82$$

- Output = ?

82		

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10	2 x 2	4 x 2	1 x 2	1
0	4 x 1	0 x 0	2 x 1	2
1	8 x 2	16 x 2	0 x 2	0
8	1	1	0	0
0	0	2	6	1

$$2 \times 2 + 4 \times 2 + 1 \times 2 + 4 \times 1 + 0 \times 0 + 2 \times 1 + 8 \times 2 + 16 \times 2 + 0 \times 2 = 68$$

- Output = ?

82	68	

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10	2	4 x 2	1 x 2	1 x 2
0	4	0 x 1	2 x 0	2 x 1
1	8	16 x 2	0 x 2	0 x 2
8	1	1	0	0
0	0	2	6	1

$4 \times 2 + 1 \times 2 + 1 \times 2 +$
$0 \times 1 + 2 \times 0 + 2 \times 1 +$
$16 \times 2 + 0 \times 2 + 0 \times 2 = 46$

- Output = ?

82	68	46

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10	2	4	1	1
0 x 2	4 x 2	0 x 2	2	2
1 x 1	8 x 0	16 x 1	0	0
8 x 2	1 x 2	1 x 2	0	0
0	0	2	6	1

$0 \times 2 + 4 \times 2 + 0 \times 2 +$
$1 \times 1 + 8 \times 0 + 16 \times 1 +$
$8 \times 2 + 1 \times 2 + 1 \times 2 = 46$

- Output = ?

82	68	46
45		

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

- Output = 3 x 3

10	2	4	1	1
0	4	0	2	2
1	8	16	0	0
8	1	1	0	0
0	0	2	6	1

*

2	2	2
1	0	1
2	2	2

=

82	68	46
45	24	26
63	65	51

Convolutional layer

Original

Sharpen

Edge Detect

"Strong" Edge Detect

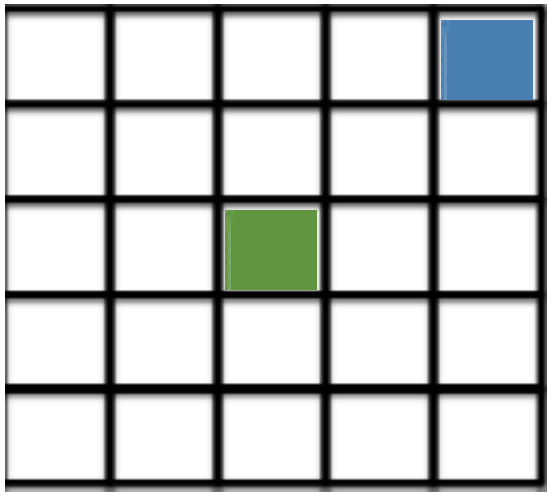
w1	w2	w3
w4	w5	w6
w7	w8	w9

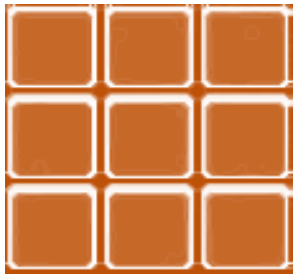
Learnable filter[1]

1. The filter weights are like the weights in Feed Forward Networks (FFN). Similar to FFN, a bias term is added after applying the filter weights.

Convolutional layer - Padding

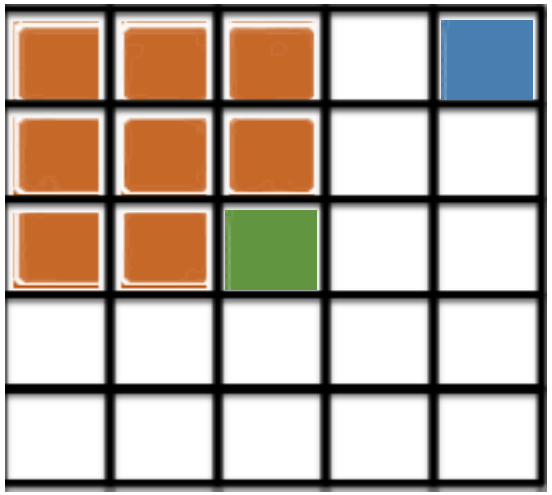
- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 0**
- **Number of times green pixel is in the filter’s receptive field: 0**





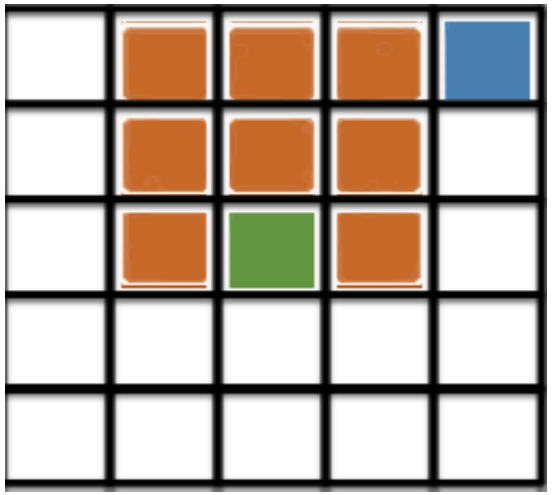
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 0**
- **Number of times green pixel is in the filter’s receptive field: 1**



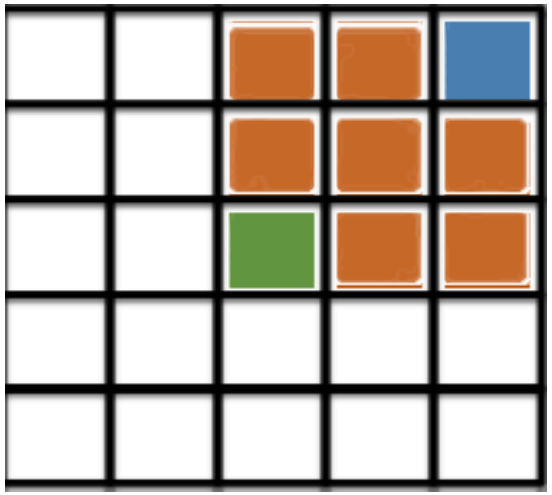
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 0**
- **Number of times green pixel is in the filter’s receptive field: 2**



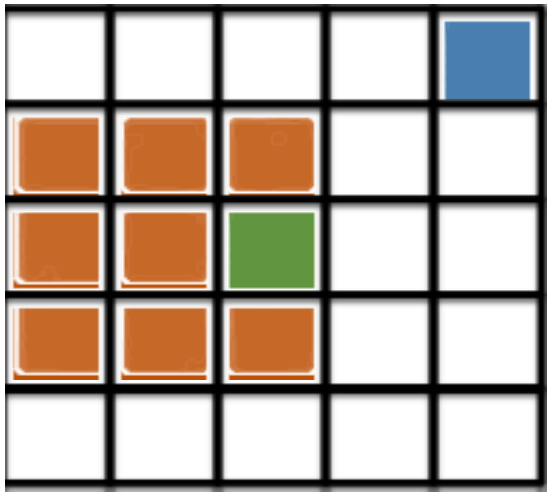
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 3**



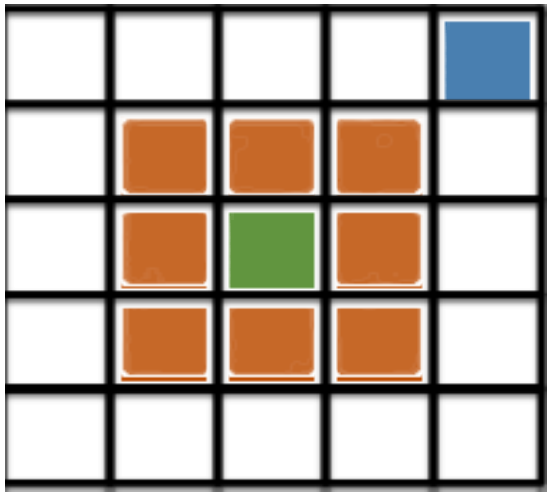
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 4**



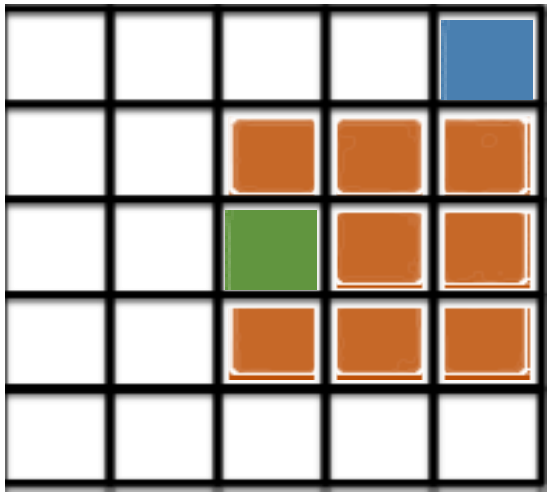
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 5**



Convolutional layer - Padding

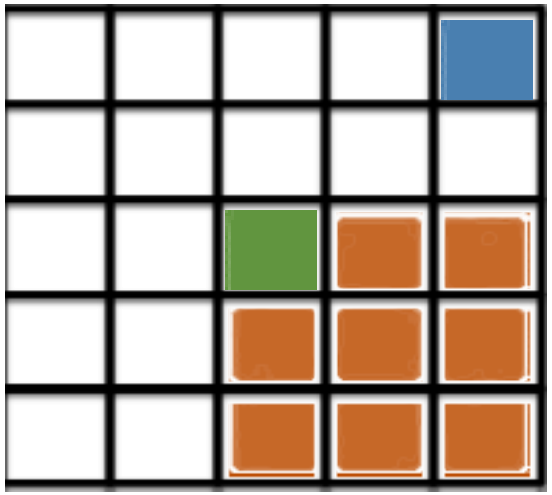
- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 6**



Convolutional layer - Padding

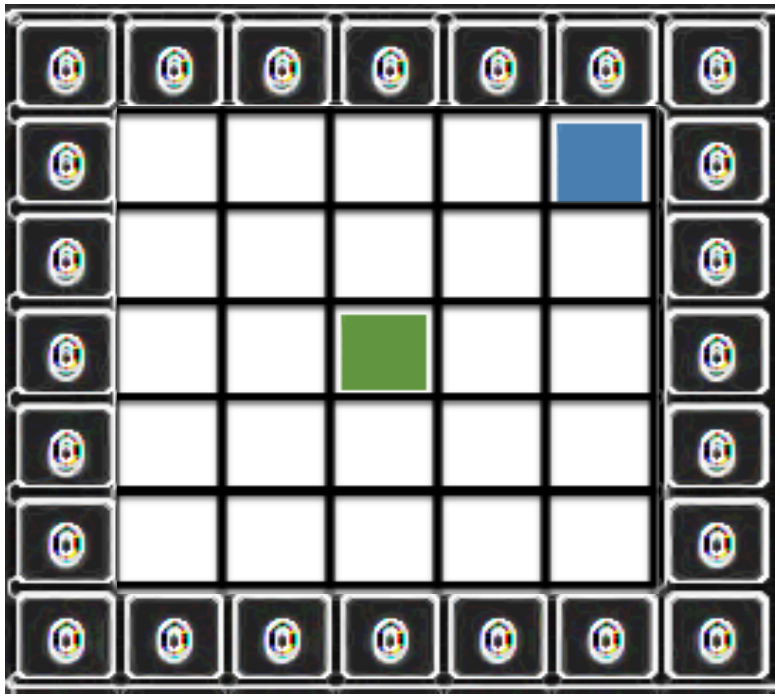
- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 9**

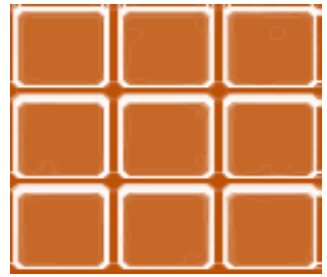
… Skip to the end



Convolutional layer - Padding

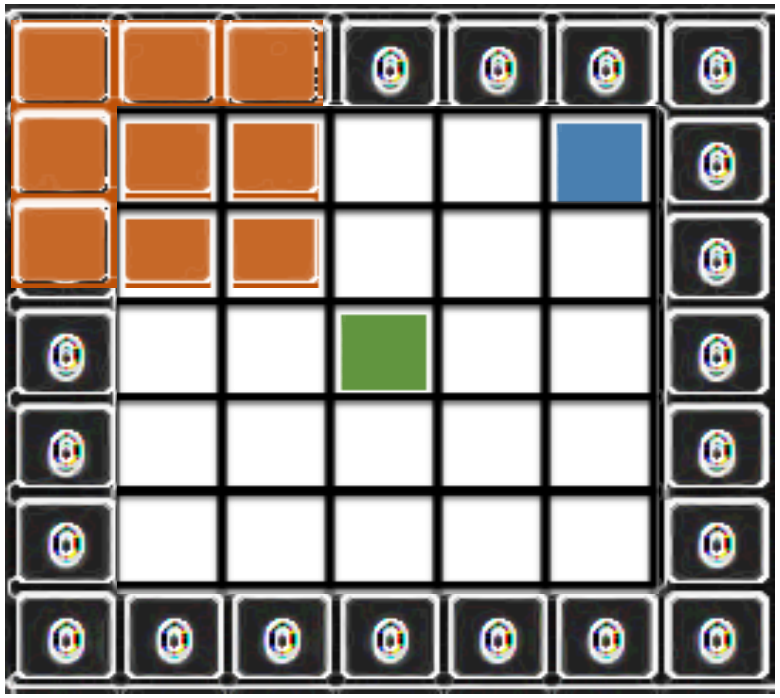
- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 0
- Number of times green pixel is in the filter's receptive field: 0





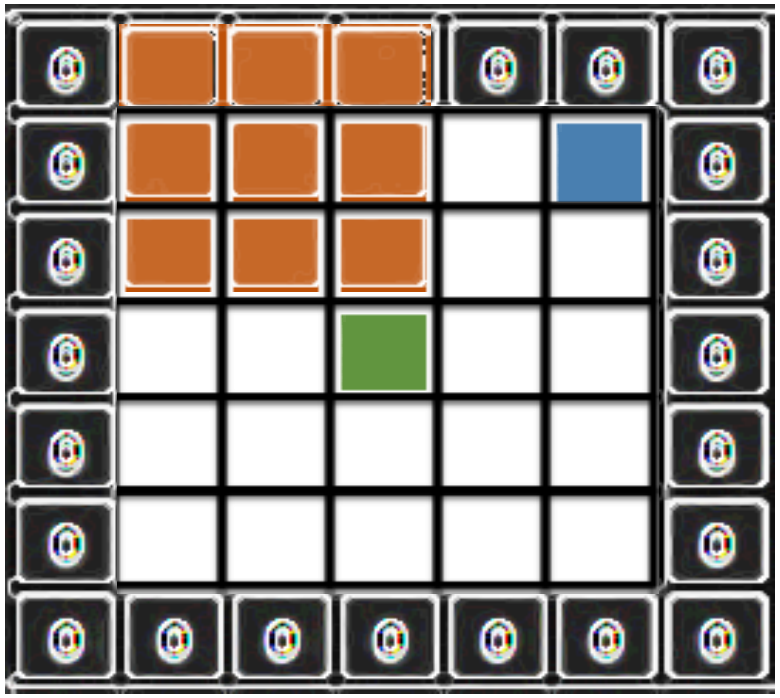
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 0
- Number of times green pixel is in the filter's receptive field: 0



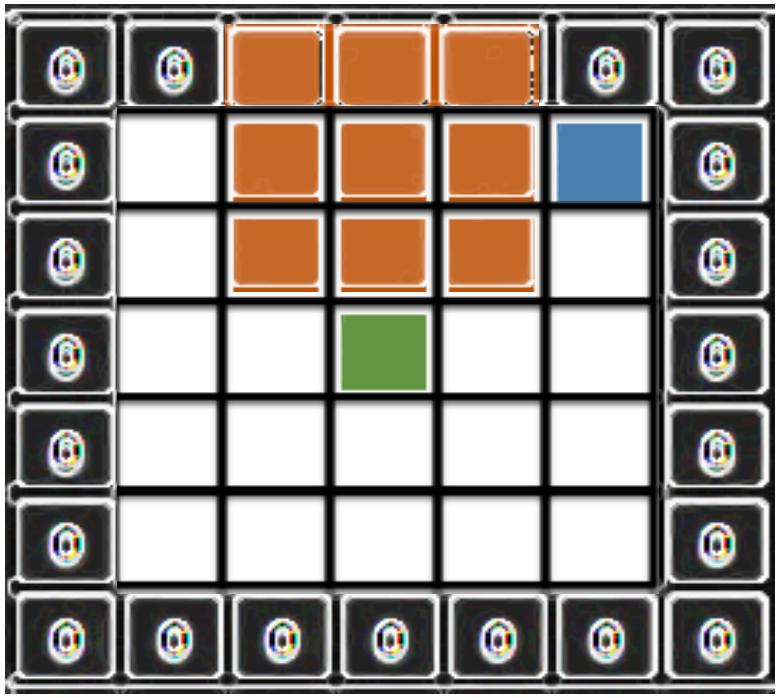
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 0
- Number of times green pixel is in the filter's receptive field: 0



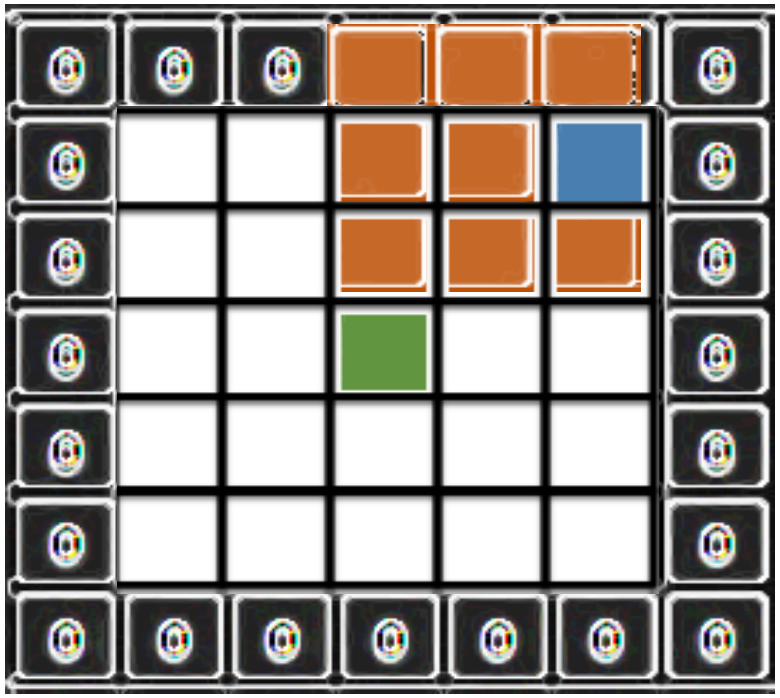
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 0
- Number of times green pixel is in the filter's receptive field: 0



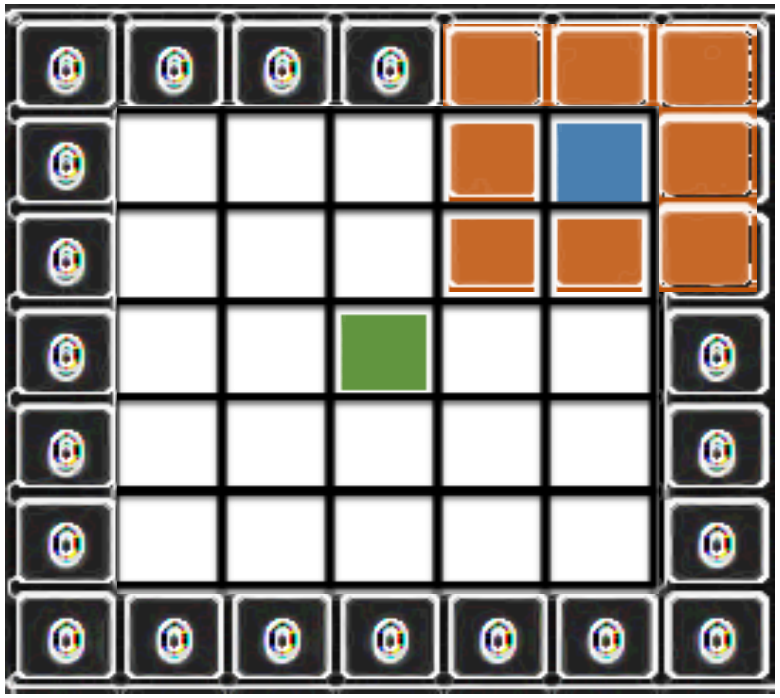
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 1
- Number of times green pixel is in the filter's receptive field: 0



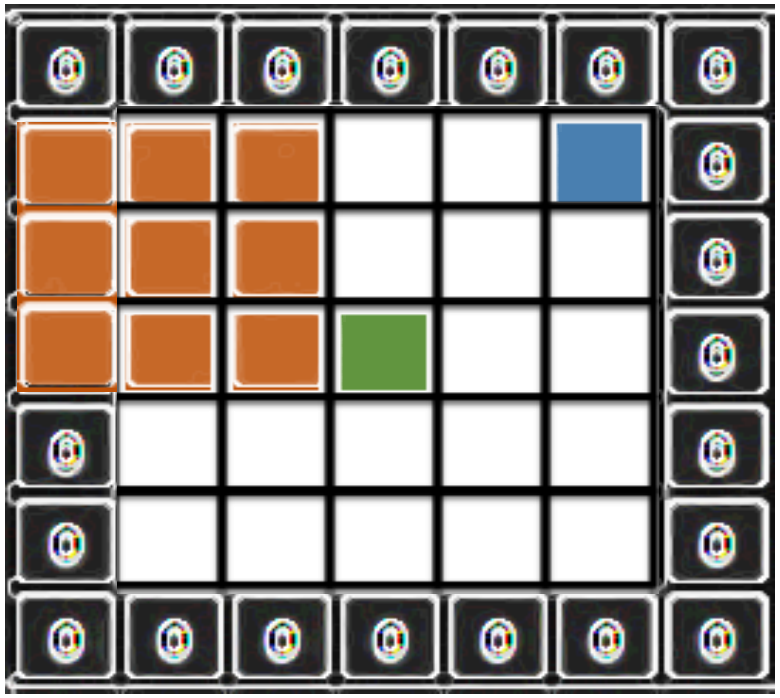
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 2
- Number of times green pixel is in the filter's receptive field: 0



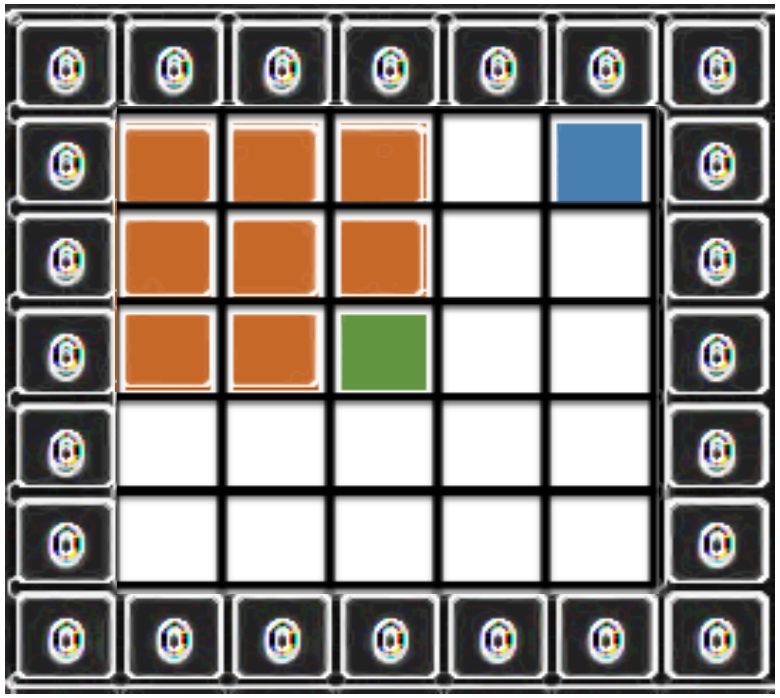
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 2
- Number of times green pixel is in the filter's receptive field: 0



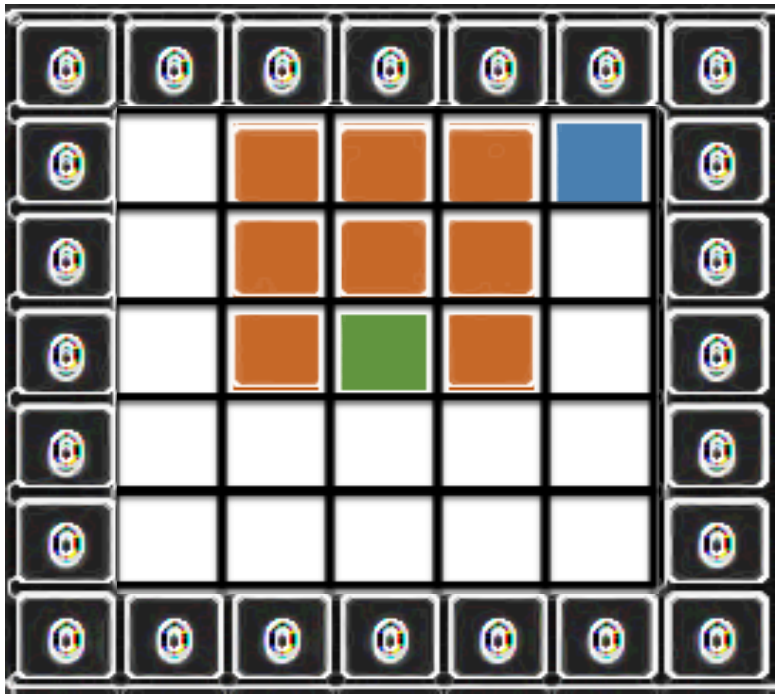
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 2
- Number of times green pixel is in the filter's receptive field: 1



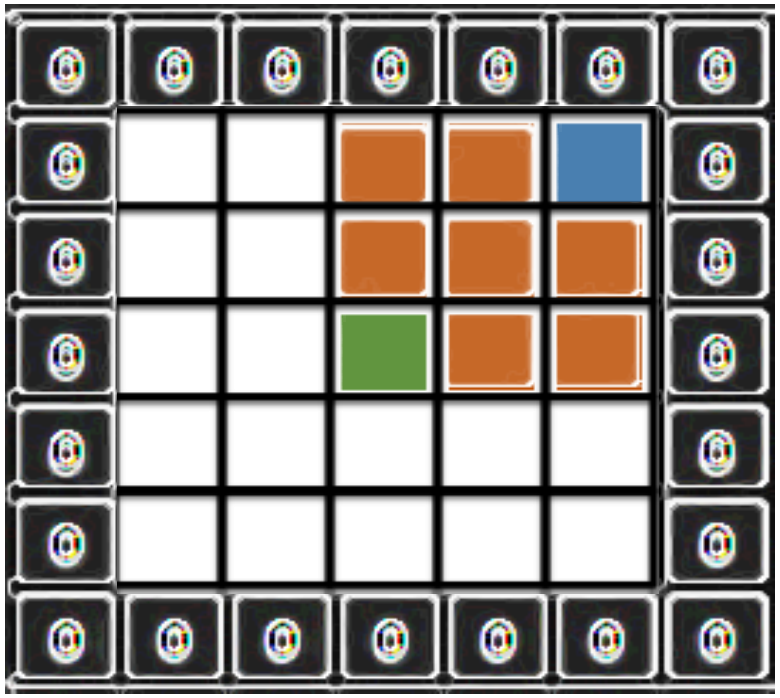
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 2
- Number of times green pixel is in the filter's receptive field: 2



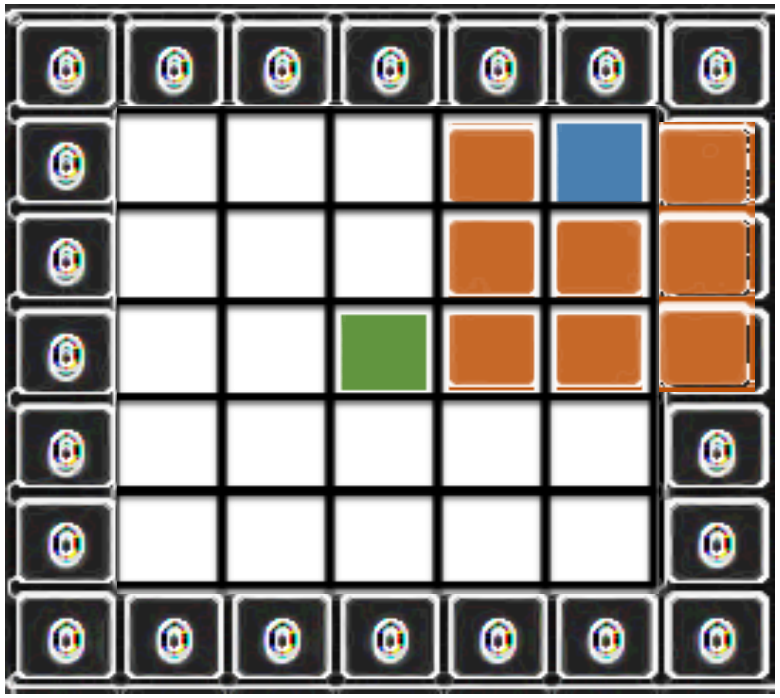
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 3
- Number of times green pixel is in the filter's receptive field: 3



Convolutional layer - Padding

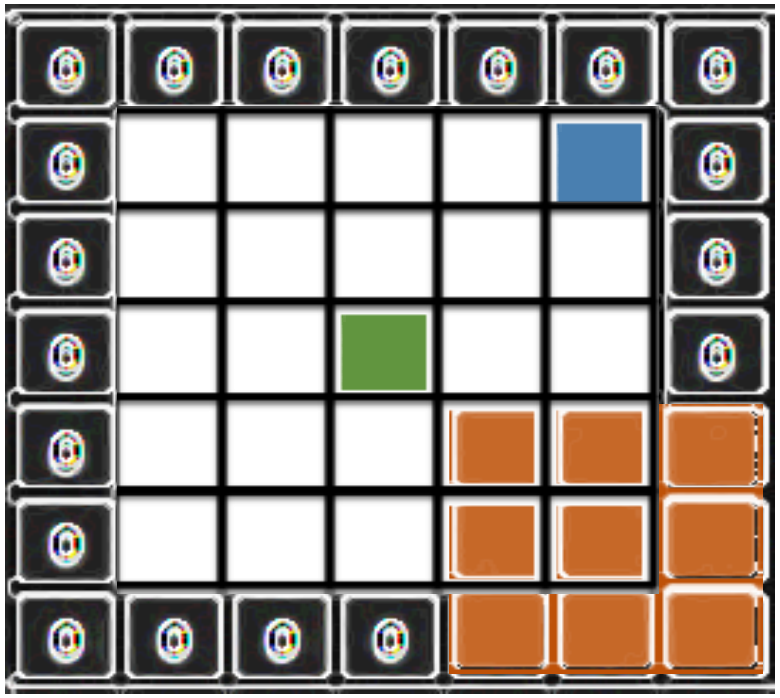
- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 4
- Number of times green pixel is in the filter's receptive field: 3



Convolutional layer - Padding

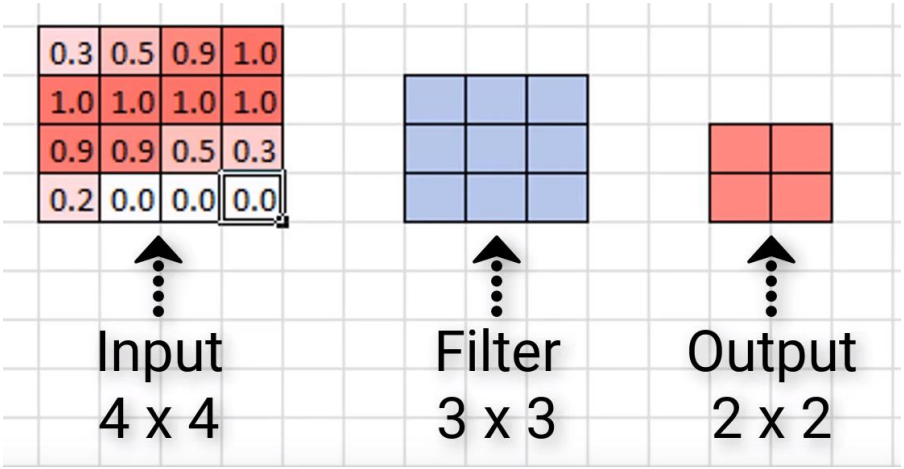
- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 4
- Number of times green pixel is in the filter's receptive field: 9

... Skip to the end

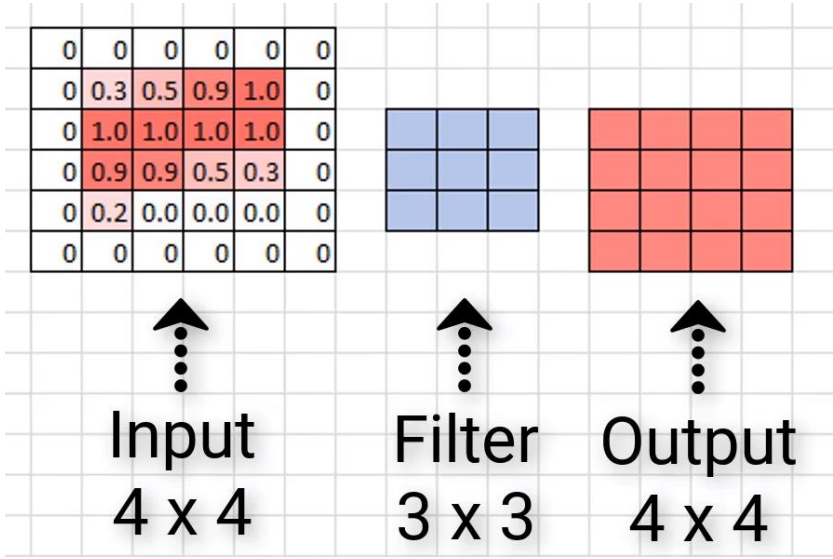


Convolutional layer - Padding

Valid Padding (No Padding)
p = 0

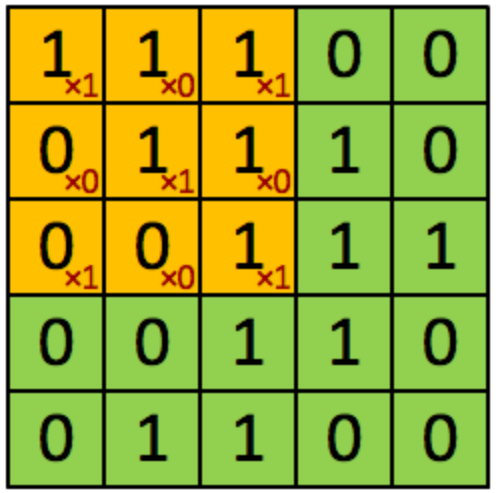


Same Padding
p ≠ 0

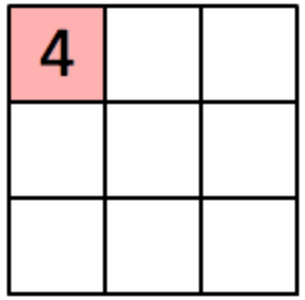


Convolutional layer - Stride

Stride
s = 1

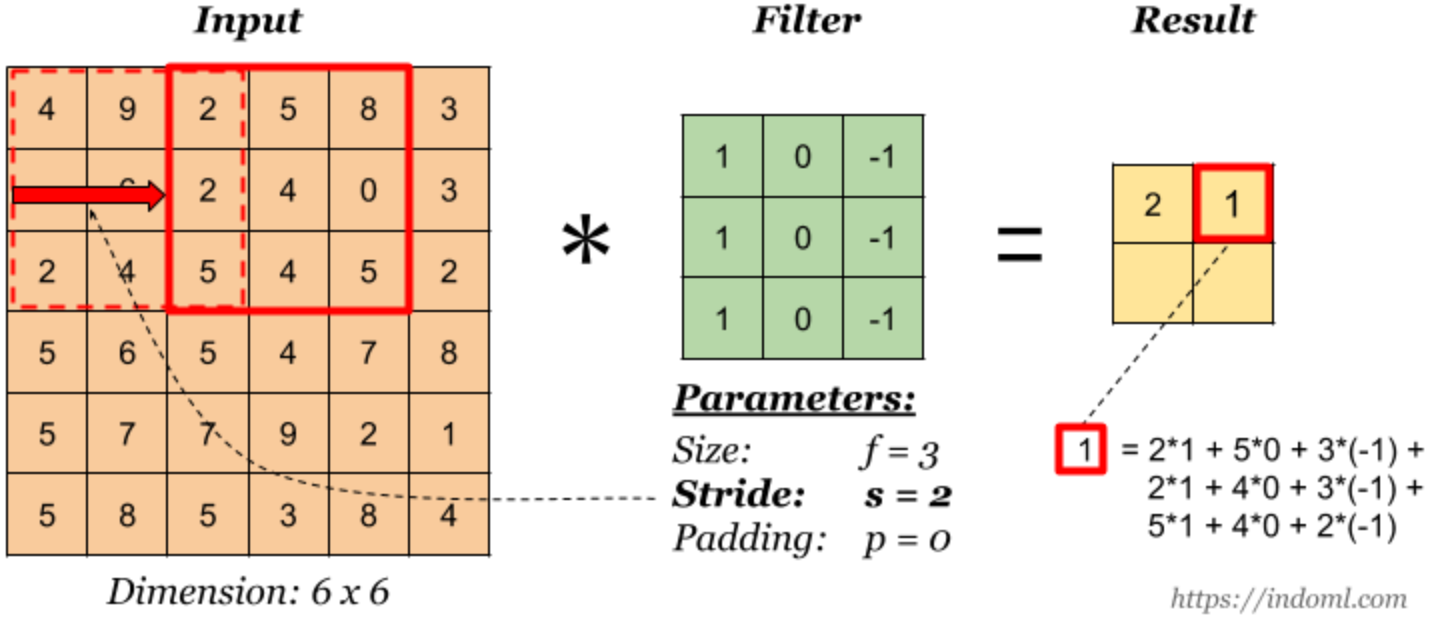


Image

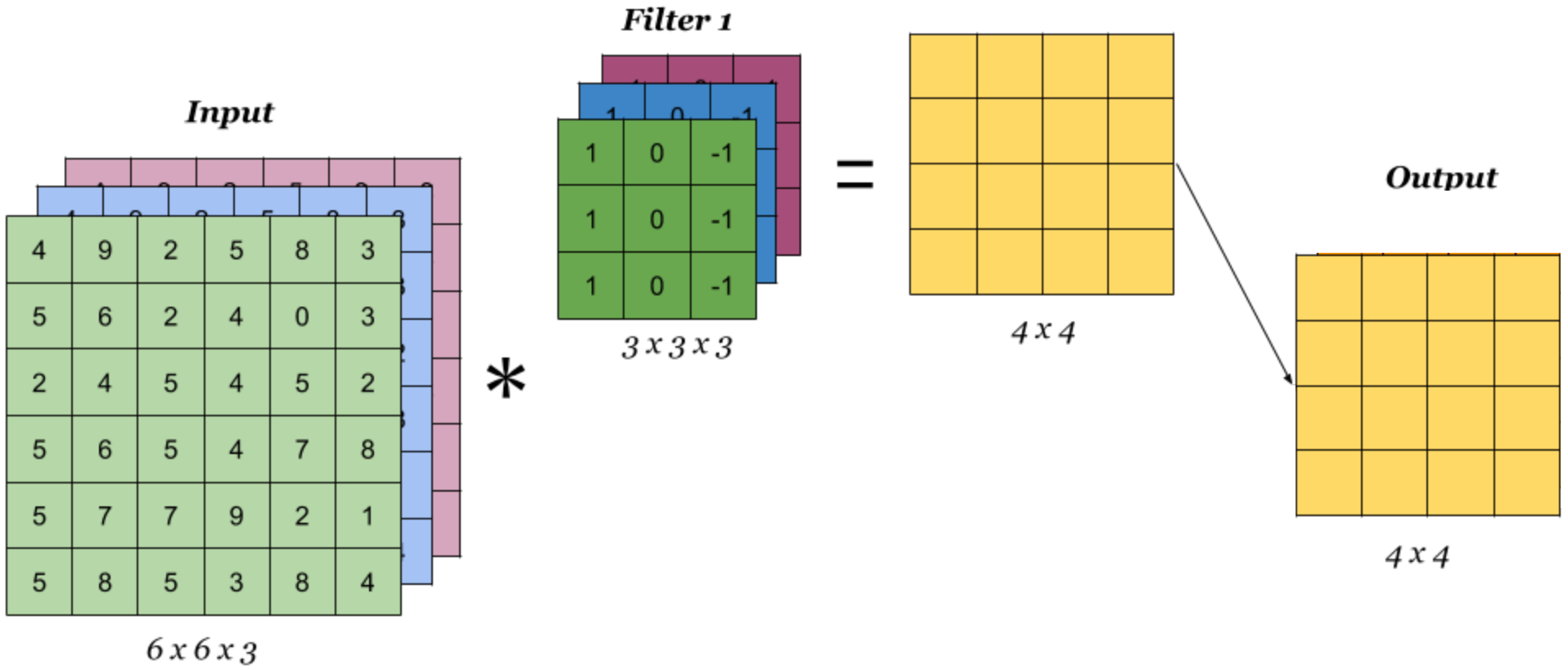


Convolved Feature

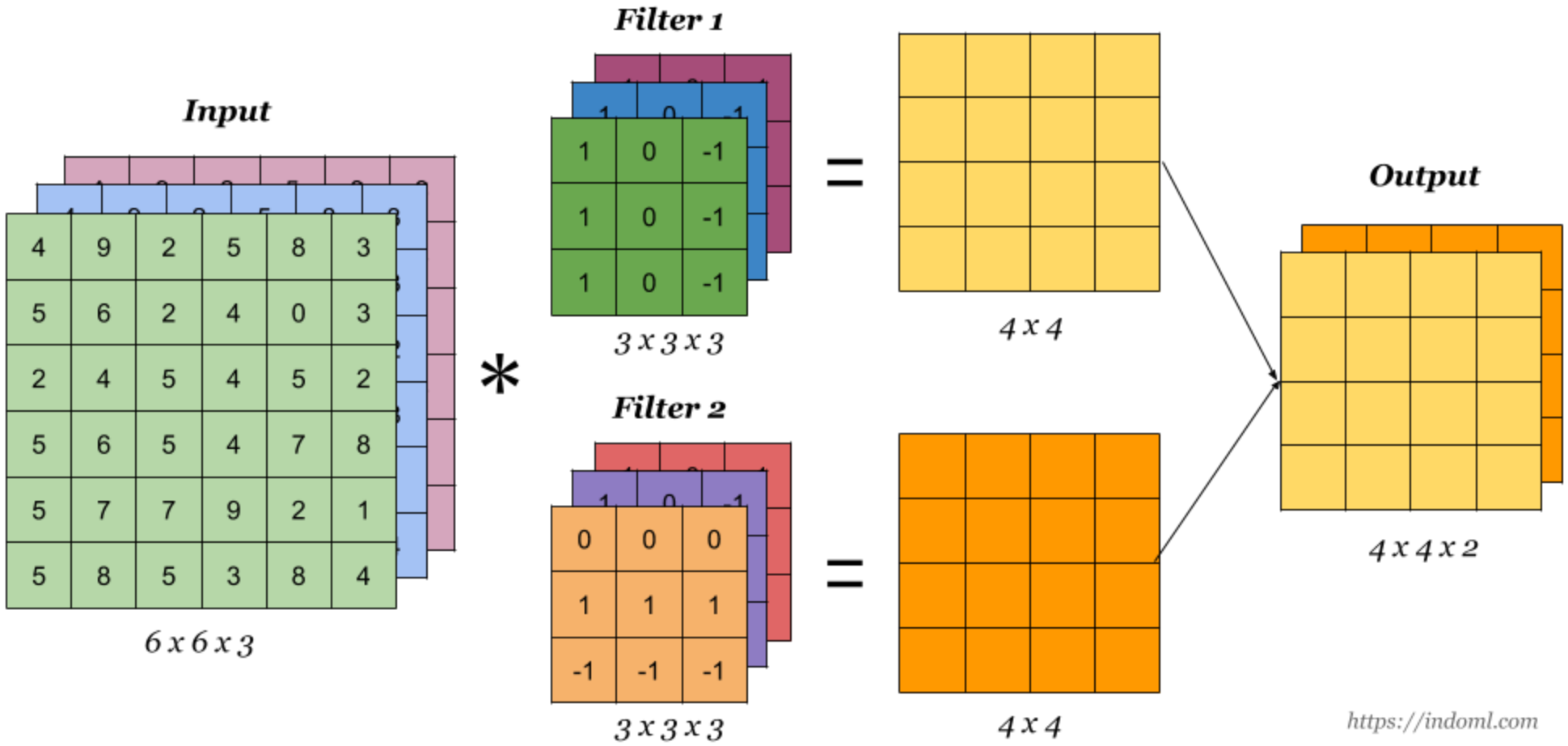
Stride
s = 2



Convolutional layer – Multiple filters



Convolutional layer – Multiple filters



Convolutional Layer

Summary of convolutions

n × m **image** f × f **filter** c **channels**

padding p **stride** s

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{m + 2p - f}{s} + 1 \right\rfloor \times \mathbf{C}$$

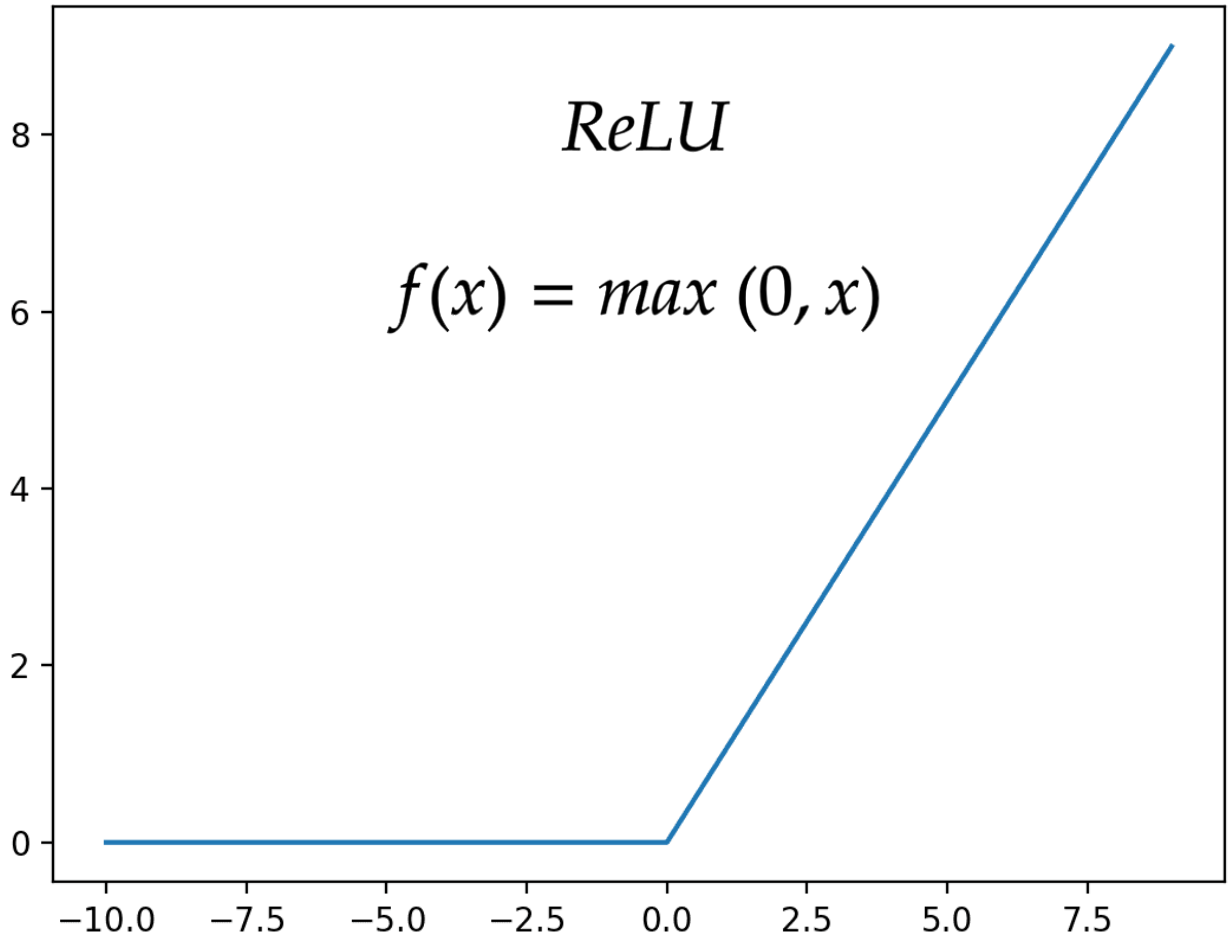
Example:

Input Image = 224x224x3

Filter = 3x3x3 (x32)
n = 224; p = 0; s = 1; f = 3; c_o = 32

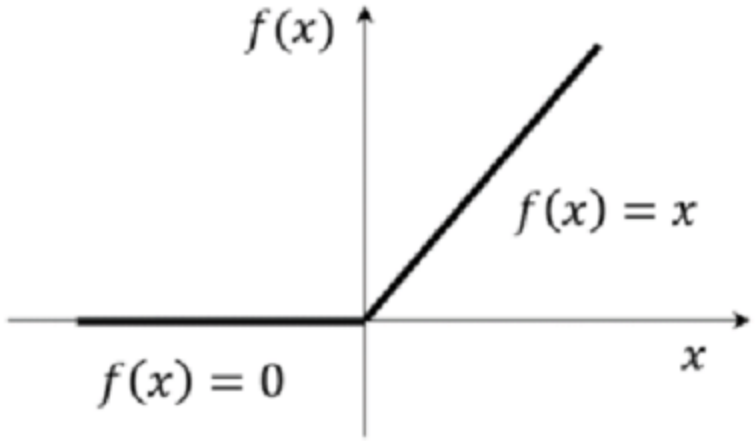
Output = 222x222x32

ReLU activation function



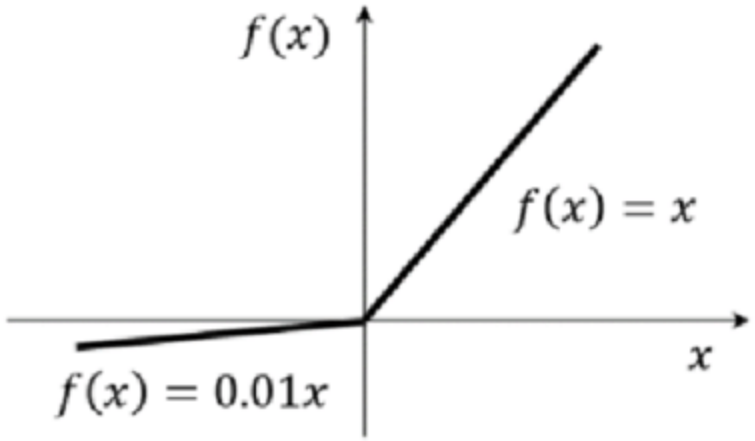
LeakyReLU activation function

$$ReLU = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$



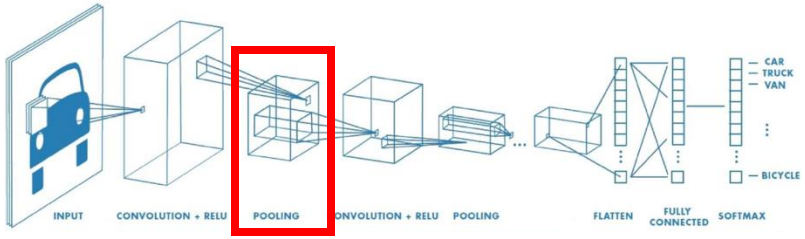
ReLU activation function

$$LealyReLU = \begin{cases} \gamma x, & x \leq 0 \\ x, & x > 0 \end{cases}$$



LeakyReLU activation function

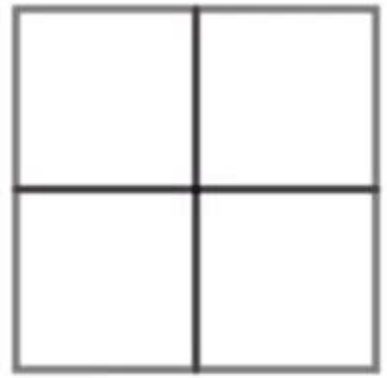
Pooling layer – Max Pooling



Max Pooling
Filter: 2x2
Stride: 2

4	6	1	1
1	3	1	3
4	0	0	8
8	5	4	0

Input (4x4)

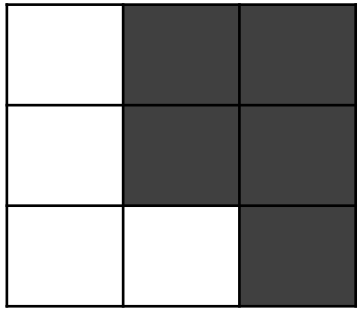


Output (2x2)

Convolutional layer – Convolution properties

- **Translation Equivariant**

$f(t(x)) = t(f(x))$,
where **t** is a transformation



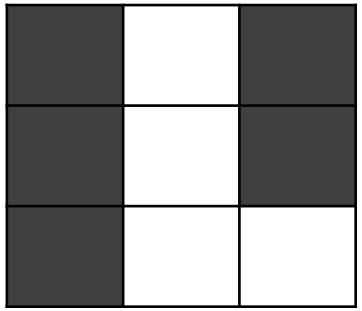
Input 3x3		
20	0	0
20	0	0
20	20	0

*

Filter 2x2	
2	0
0	2

=

Output 2x2	
40	0
80	0



0	20	0
0	20	0
0	20	20

*

2	0
0	2

=

40	40
40	80

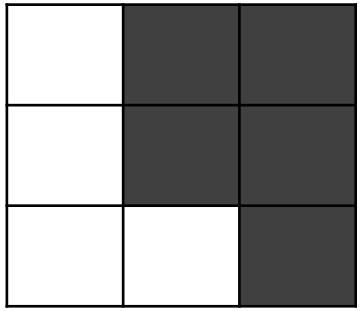
Both the **input** and the **output** shifted to the **right**!

1. Translation equivariance means that the model responds predictably to translations, while translation invariance means that the model produces the same output regardless of translations

Pooling layer – Max Pooling properties

- **Translation Invariant**

$f\big(t(x)\big) = f(x)$,
where **t** is a transformation



Input 3x3		
20	0	0
20	0	0
20	20	0

*

Filter 2x2	
2	0
0	2

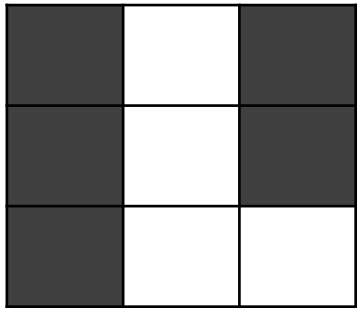
=

Output 2x2	
40	0
80	0

Pooling 2x2 →

80

Change in the **input** results in the **same output**



0	20	0
0	20	0
0	20	20

*

2	0
0	2

=

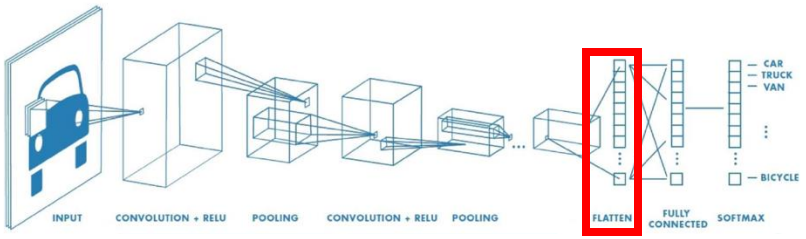
40	40
40	80

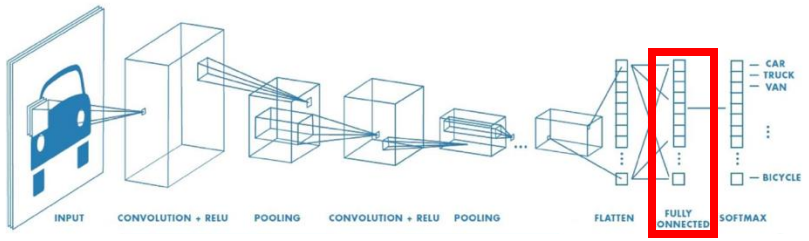
Pooling 2x2 →

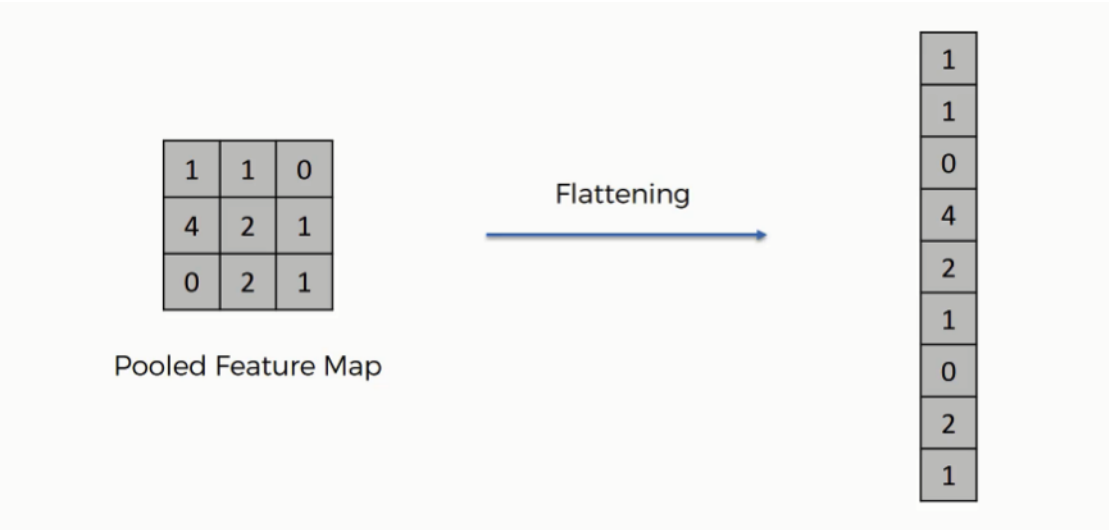
80

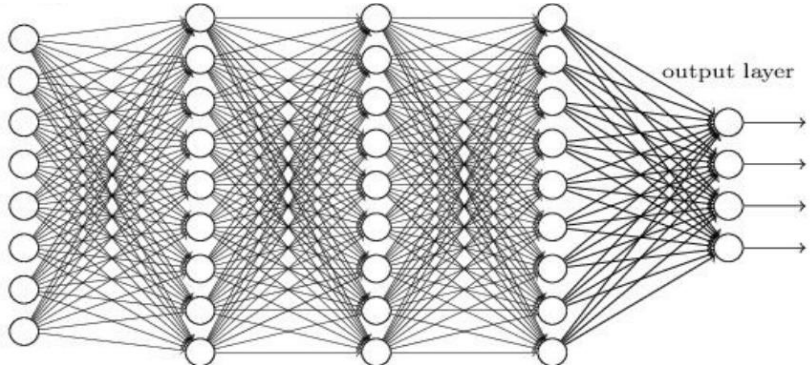
1. Translation equivariance means that the model responds predictably to translations, while translation invariance means that the model produces the same output regardless of translations

Flatten and Fully Connected layer





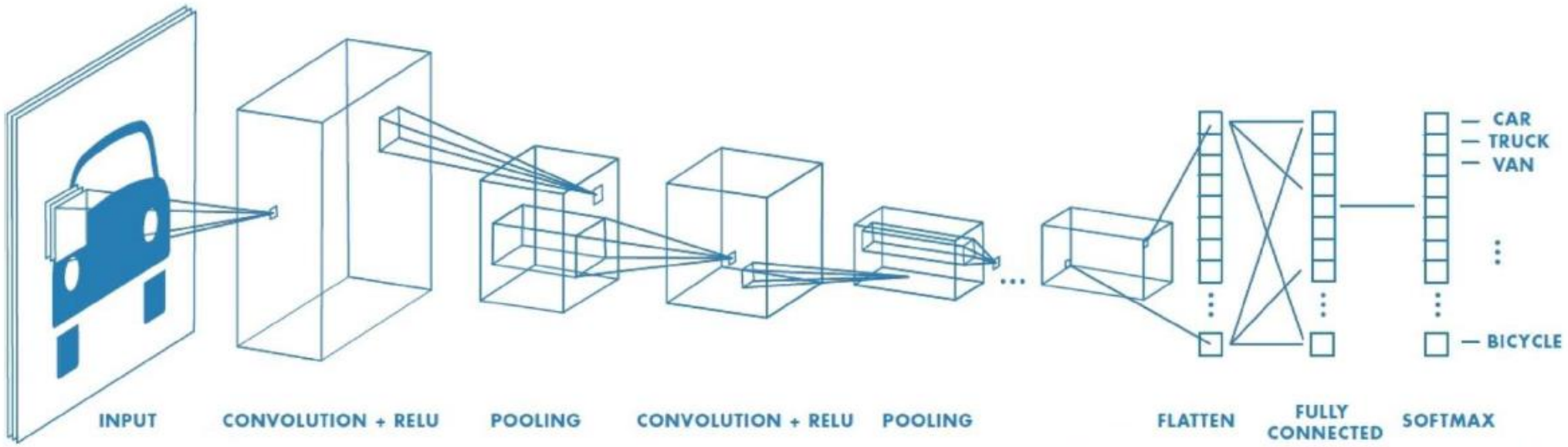




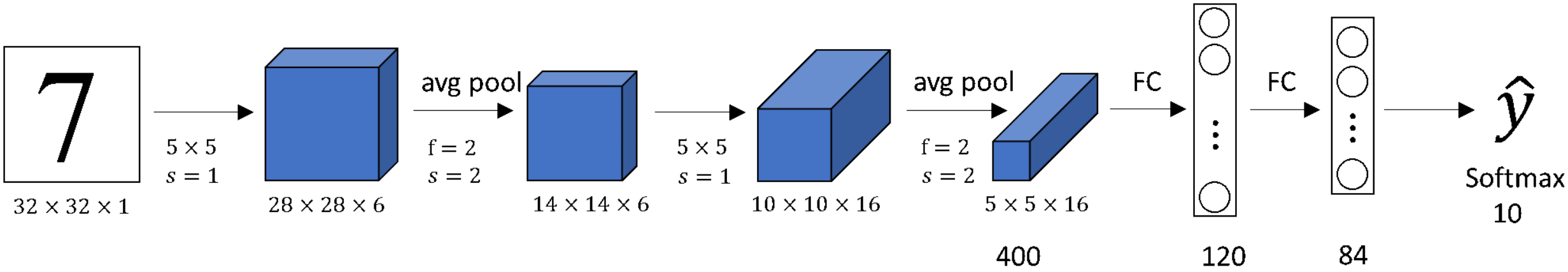
Convolutional Neural Network

Visualization:

- https://poloclub.github.io/cnn-explainer/
- https://distill.pub/2017/feature-visualization/

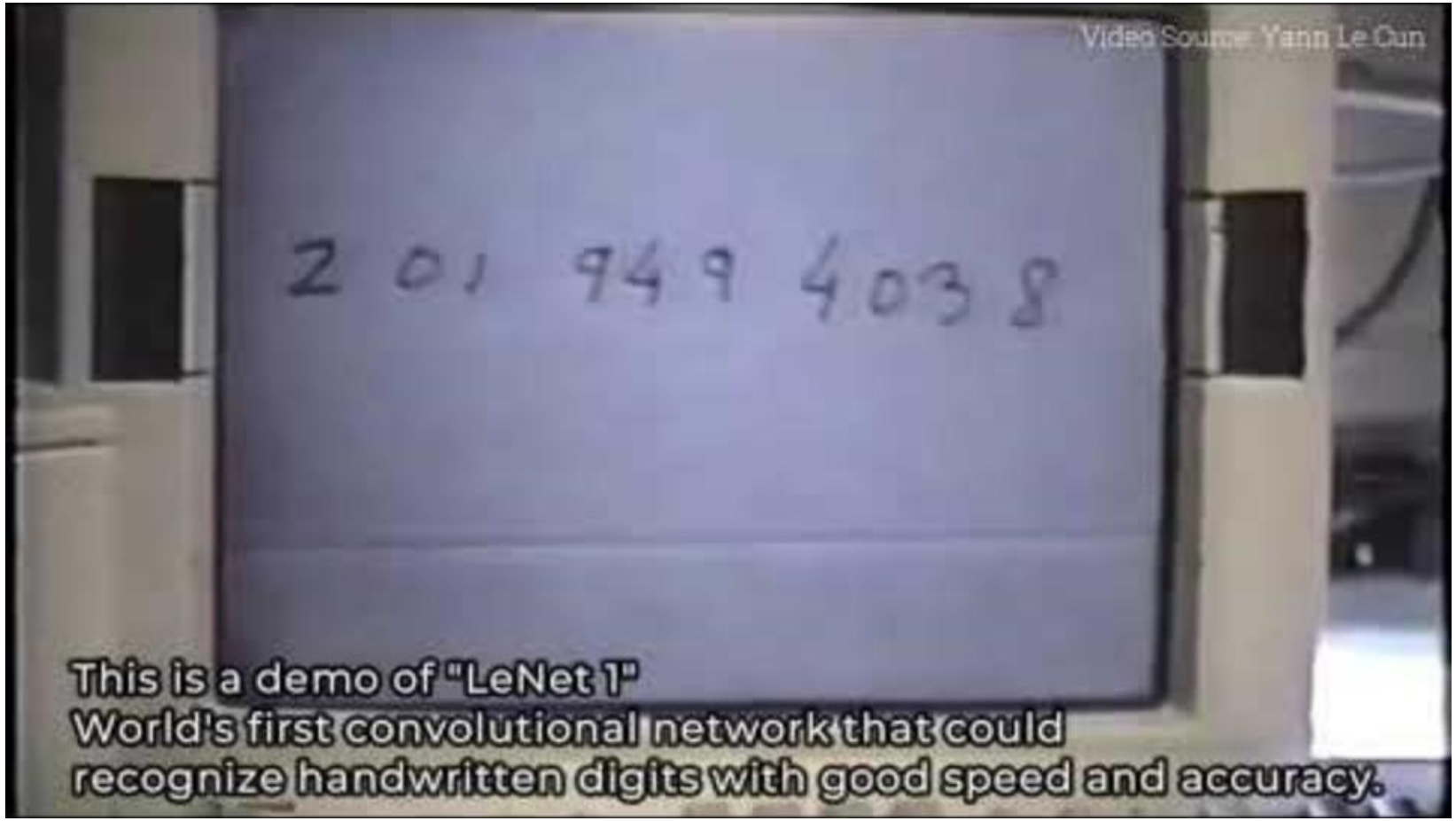


“LeNet 5” Y. Lecun et al. (1998) [3]



[3] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

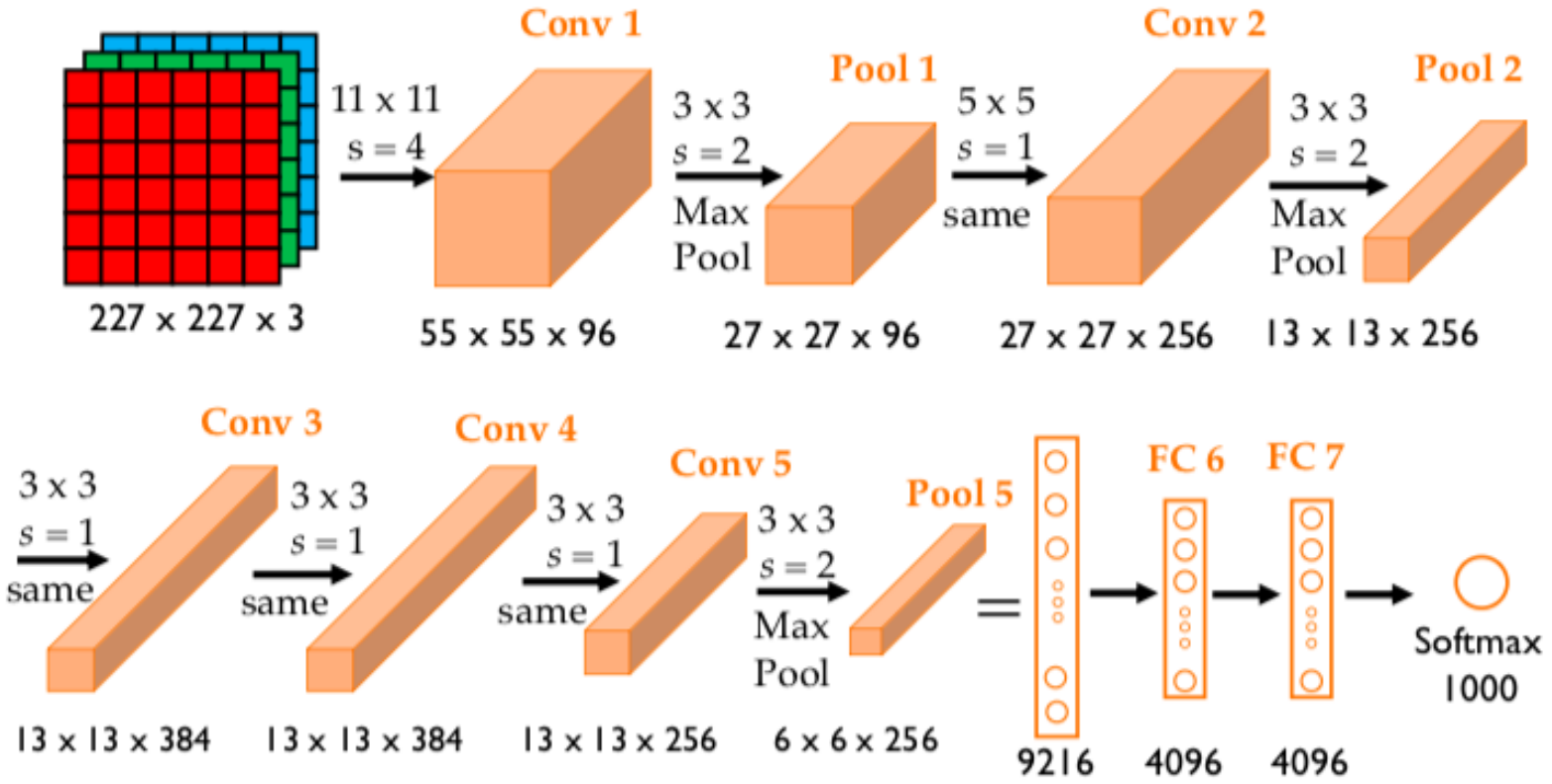
“LeNet 5” Y. Lecun et al. (1998) [3]



[3] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

“AlexNet” A Krizhevskyet al. (2012) [4]

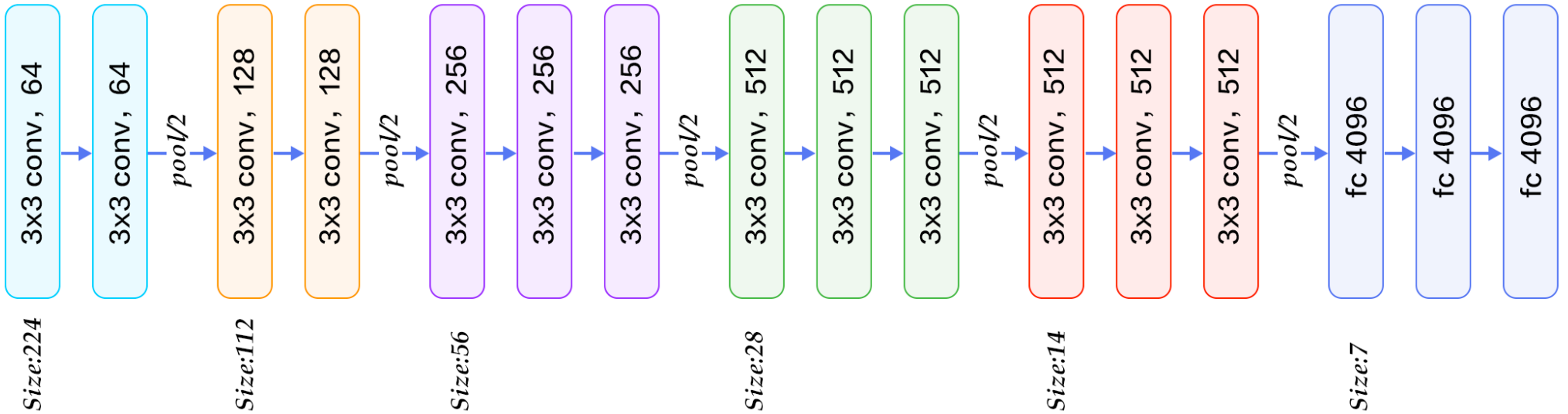
- Trained on the ImageNet dataset
- Contains more than 1M images
- 1000 classes



[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems (Vol. 25). Curran Associates, Inc. https://doi.org/10.1145/3065386

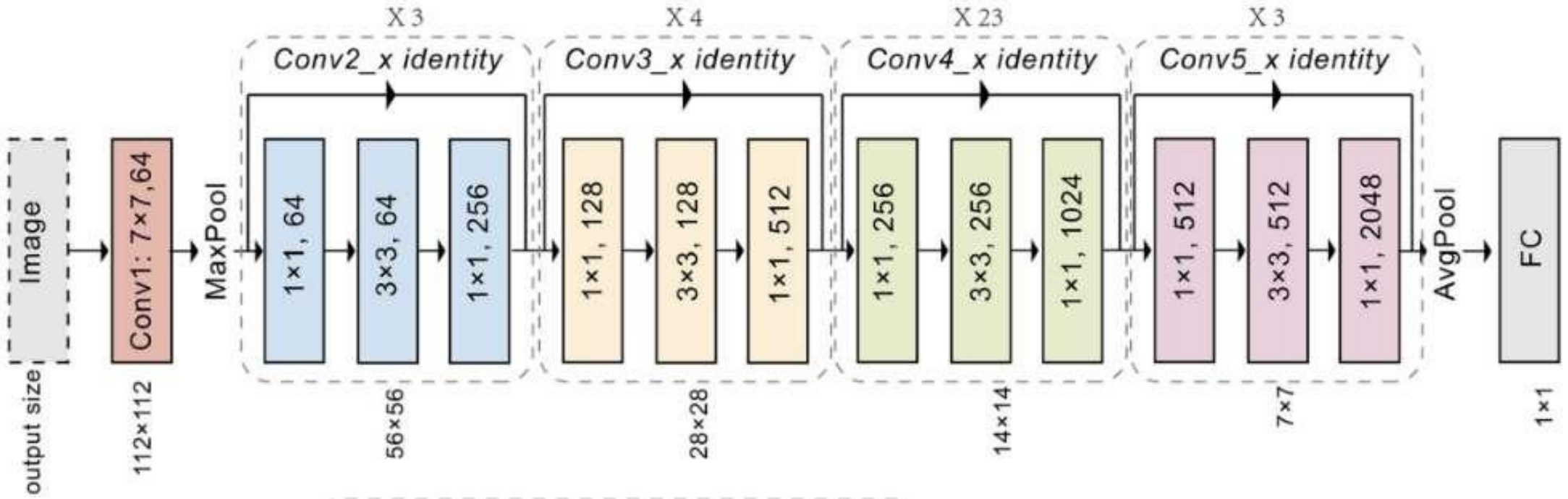
“VGGNet” Simonyan and Zisserman (2014) [5]

- The number of filters doubles in each block



[5] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://arxiv.org/abs/1409.1556

“ResNet” Kaiming He et al. (2015) [6]



[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Summary

- **Fully Connected Networks** (Feed Forward Networks) calculate a weighted sum of the inputs
- We can add non-linear activation functions like **Sigmoid** to transform linear regression to logistic regression (classification)

- **Classification** is a supervised learning task
  - Binary: the target is 0 or 1
  - Multiclass: the target is one element out of a discrete set of elements (usually use one-hot encoding)

- **Convolutional Neural Networks** (CNNs) are efficient algorithms for images
  - Weight sharing
  - Capture spatial patterns

- By changing the CNN configurations, we create **different architectures**

Links

Backpropagation related

- https://medium.com/binaryandmore/beginners-guide-to-deriving-and-implementing-backpropagation-e3c1a5a1e536
- (matrix form): https://sudeepraja.github.io/Neural/

Resources

Books:

- Courville, Goodfellow, Bengio: Deep Learning
  Freely available: https://www.deeplearningbook.org/
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J.: Dive into Deep Learning
  Freely available: https://d2l.ai/

Courses:

- Deep Learning specialization by Andrew NG
- https://www.coursera.org/specializations/deep-learning

That’s all for today!