An Introduction to Deep Learning

(part 3)

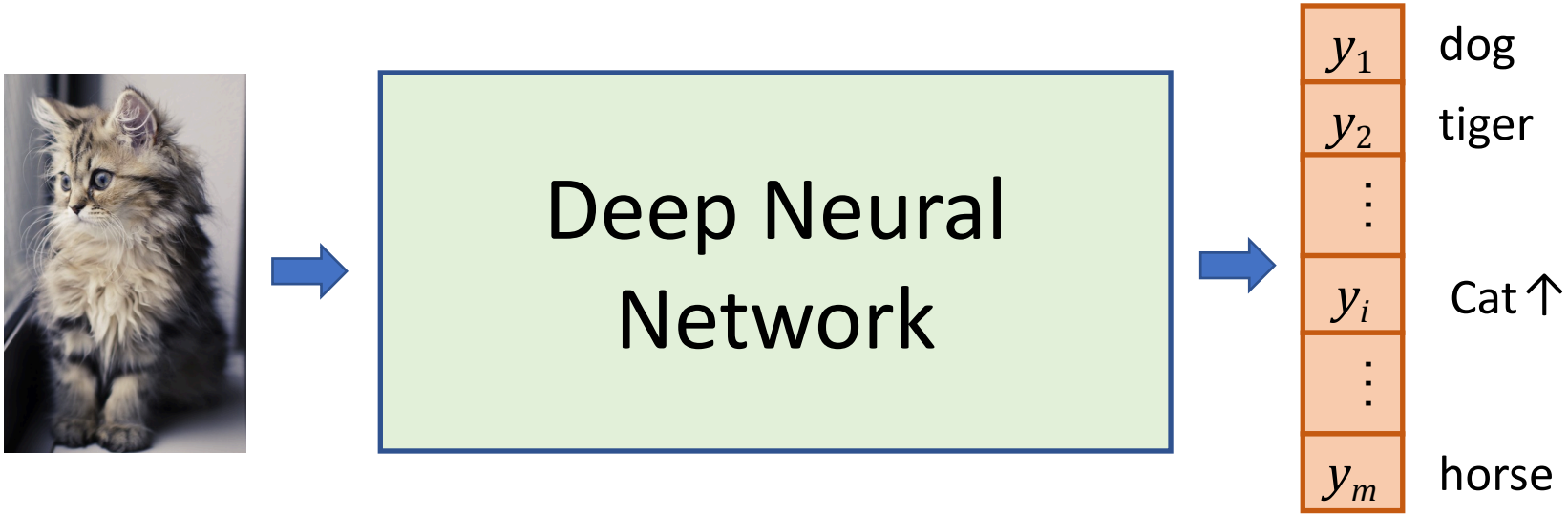
Farnoosh Heidarivincheh

EMAT31530 - February 2021

Convolutional Neural Networks

So far, we have discussed

- Neural networks with Fully-Connected layers
- The optimization process / How the network is trained



Outline

- Convolutional Neural Networks
  - Different Layers
  - Popular Architectures
- Other Popular Neural Networks
  - Recurrent Neural Networks
  - Long Short-Term Memory Networks
  - Auto-Encoders
  - Generative Adversarial Networks

Convolutional Neural Networks

Convolutional Neural Networks

Image classification with **CNN**

- No need to pre-process images
- CNNs outperform traditional approaches

Convolutional Neural Network (CNN)

y_1 dog

y_2 tiger

$\vdots$

y_i Cat ↑

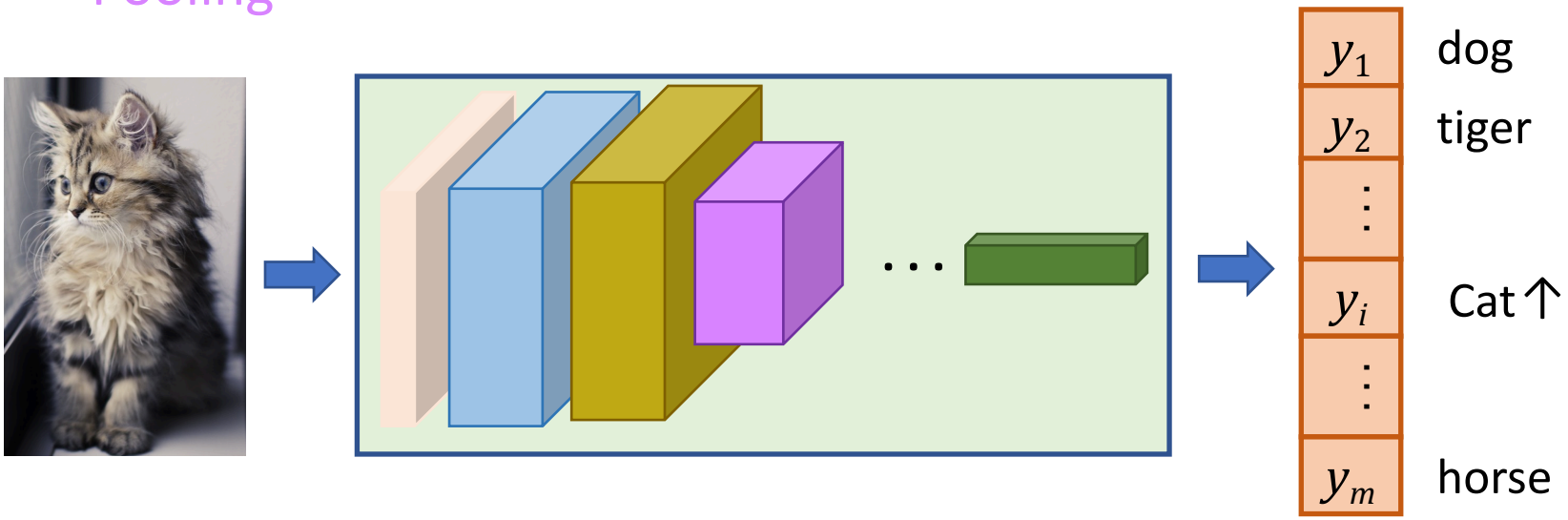
$\vdots$

y_m horse

Convolutional Neural Networks

Image classification with **CNN** - Layers in a CNN:

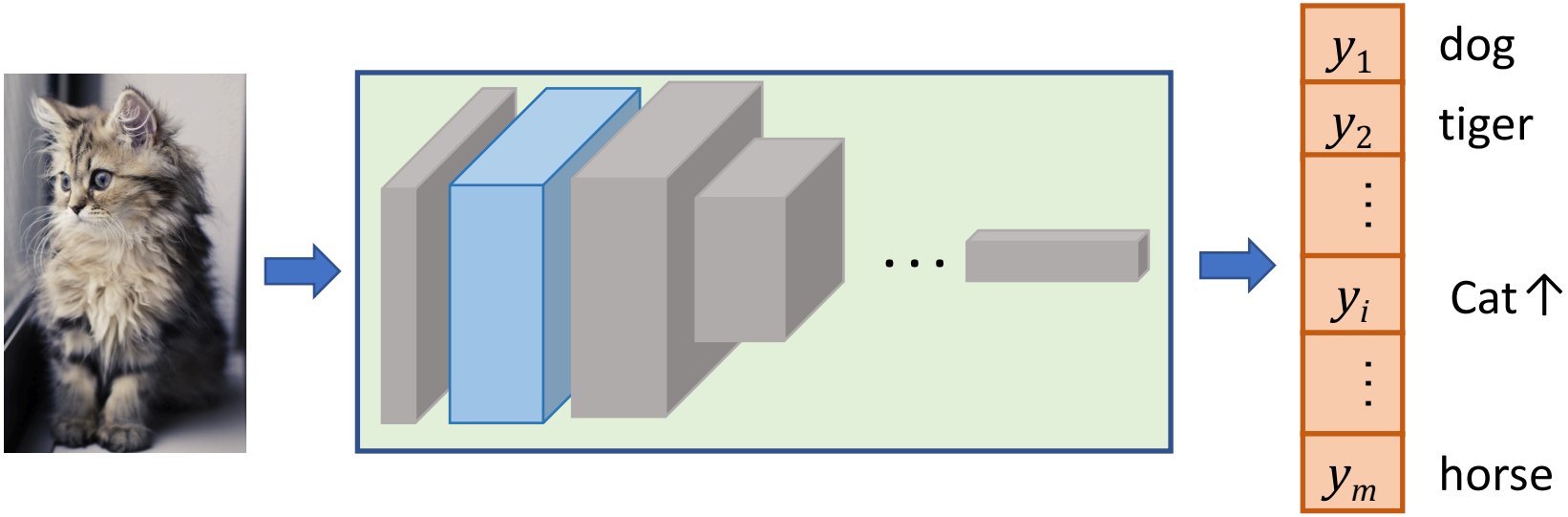
- Input
- Convolutional
- Activation (ReLU, …)
- Pooling
- Fully-Connected (FC)
- Drop-out
- Batch Normalisation (BN), ...



Convolutional Neural Networks

Convolutional layer

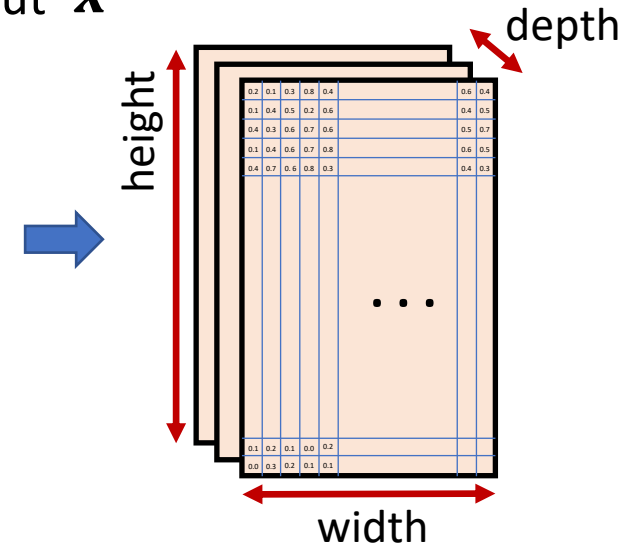
- It is the main building block of a CNN
- its neurons can have more than one dimension
- It captures the correlation among neighboring data points



Convolutional Neural Networks

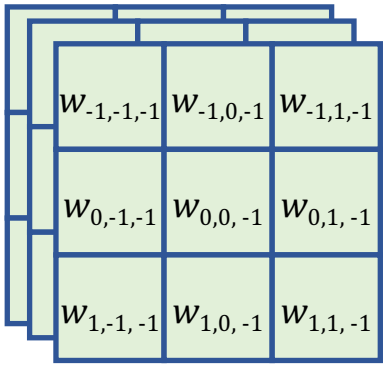
Convolutional layer

RGB image input $\boldsymbol{x}$



Convolution kernel

$\boldsymbol{w}$

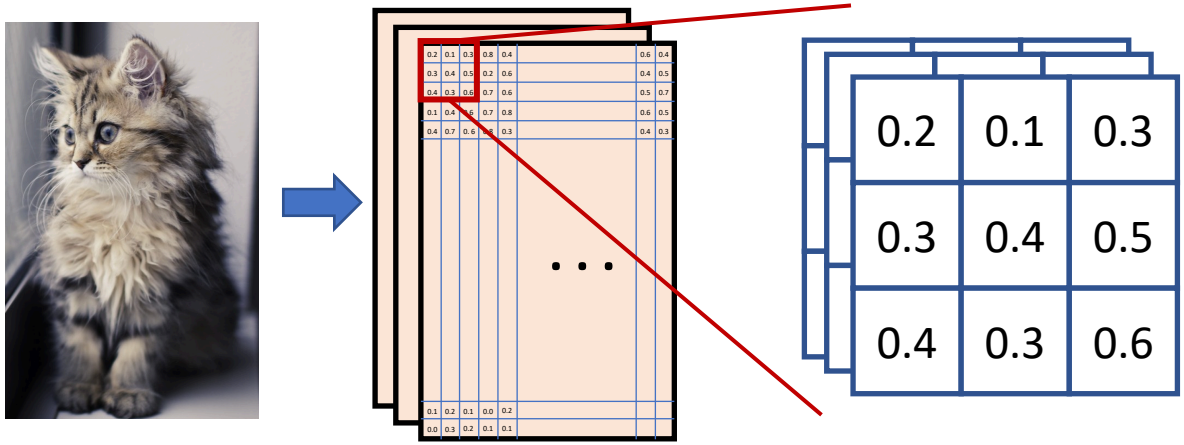


Kernel size: 3×3×3

Convolutional Neural Networks

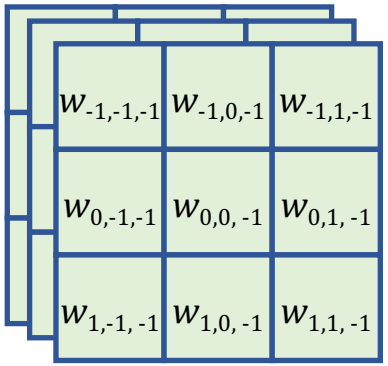
Convolutional layer

RGB image input $\boldsymbol{x}$



Convolution kernel

$\boldsymbol{w}$



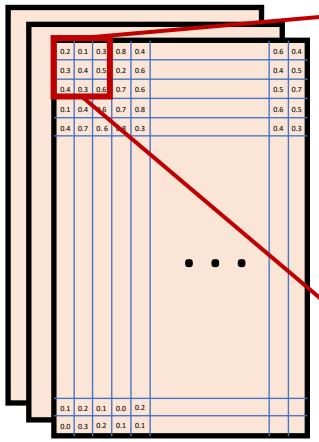
Kernel size: 3×3×3

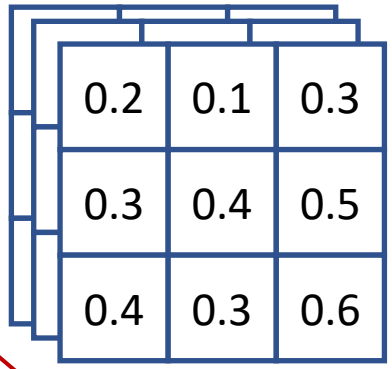
Convolutional Neural Networks

Convolutional layer

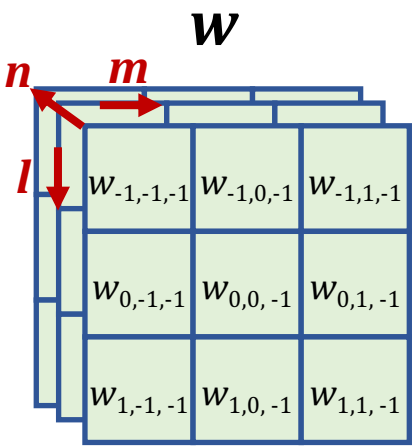
$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$





$\boldsymbol{w}$

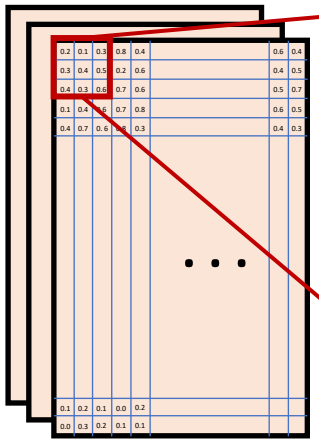


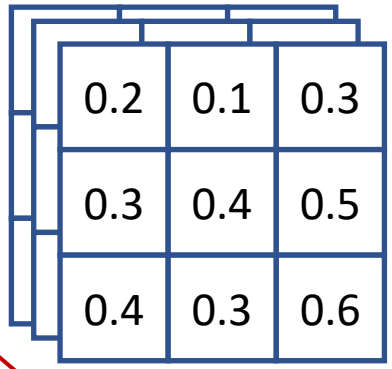
Convolutional Neural Networks

Convolutional layer

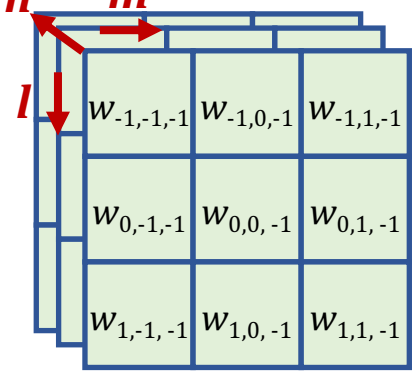
$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$





$\boldsymbol{w}$

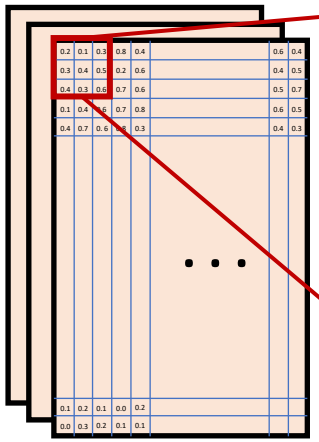


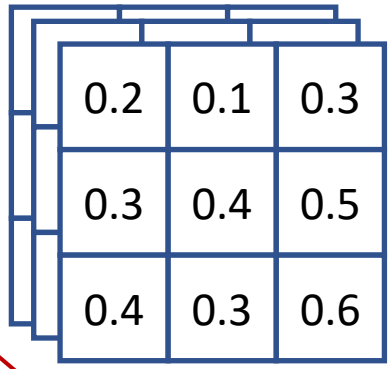
Convolutional Neural Networks

Convolutional layer

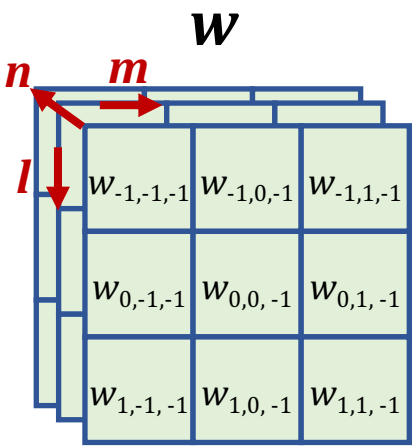
$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$





$\boldsymbol{w}$

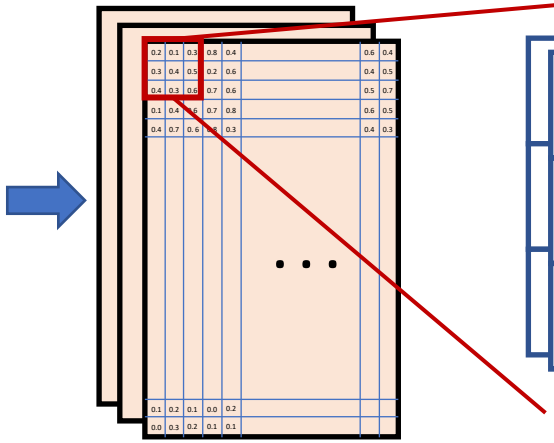


Convolutional Neural Networks

Convolutional layer

$$\text{bias } w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$



0.2	0.1	0.3
0.3	0.4	0.5
0.4	0.3	0.6

$\boldsymbol{w}$

$\boldsymbol{n}$ $\boldsymbol{m}$ $\boldsymbol{l}$

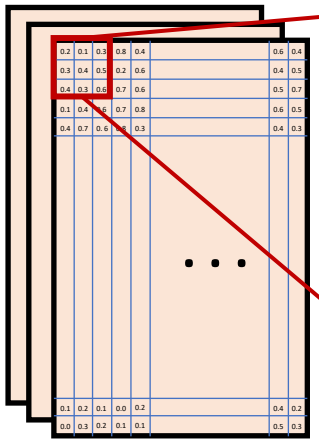
$w_{-1,-1,-1}$	$w_{-1,0,-1}$	$w_{-1,1,-1}$
$w_{0,-1,-1}$	$w_{0,0,-1}$	$w_{0,1,-1}$
$w_{1,-1,-1}$	$w_{1,0,-1}$	$w_{1,1,-1}$

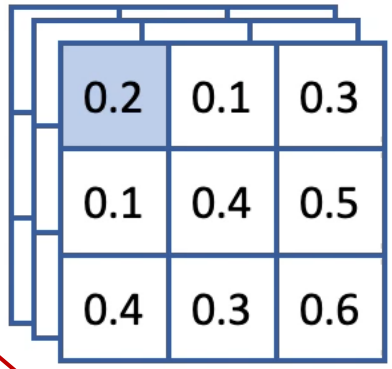
Convolutional Neural Networks

Convolutional layer

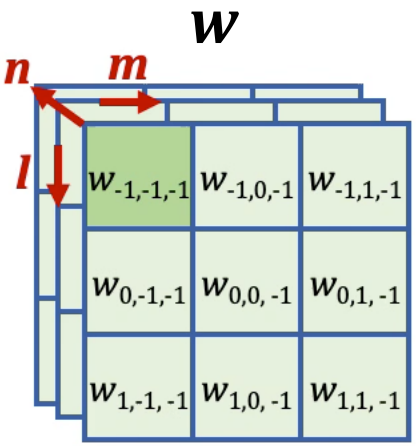
$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$





$\boldsymbol{w}$



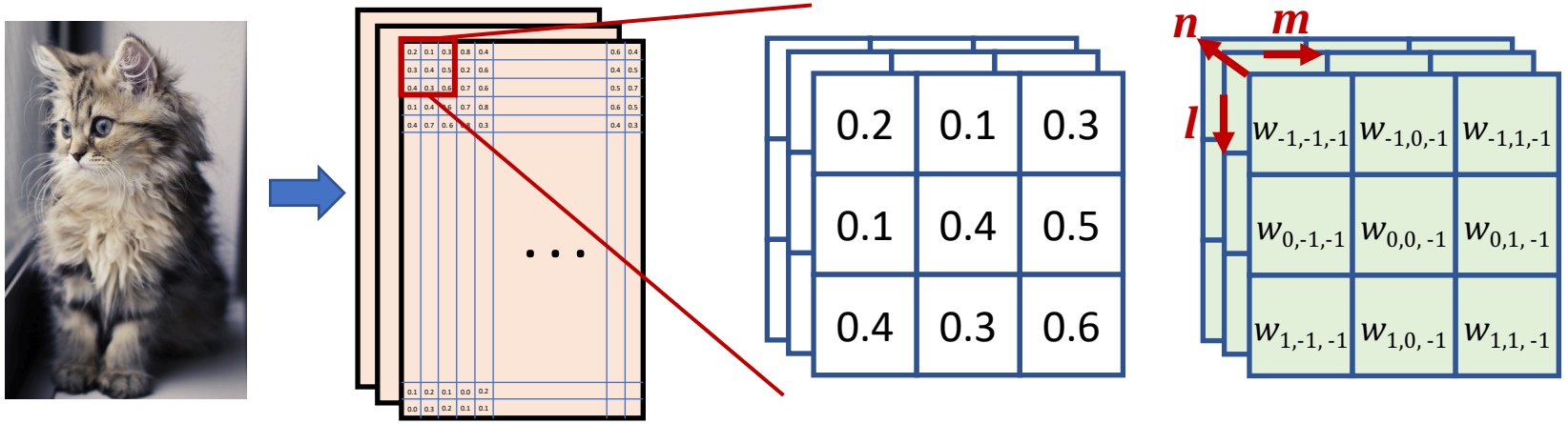
$w_b + 0.2 \times w_{-1,-1,-1}$

Convolutional Neural Networks

Convolutional layer

$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$



This is repeated for every pixel of the image (if *stride*=1)

Image is padded for the bordering pixels

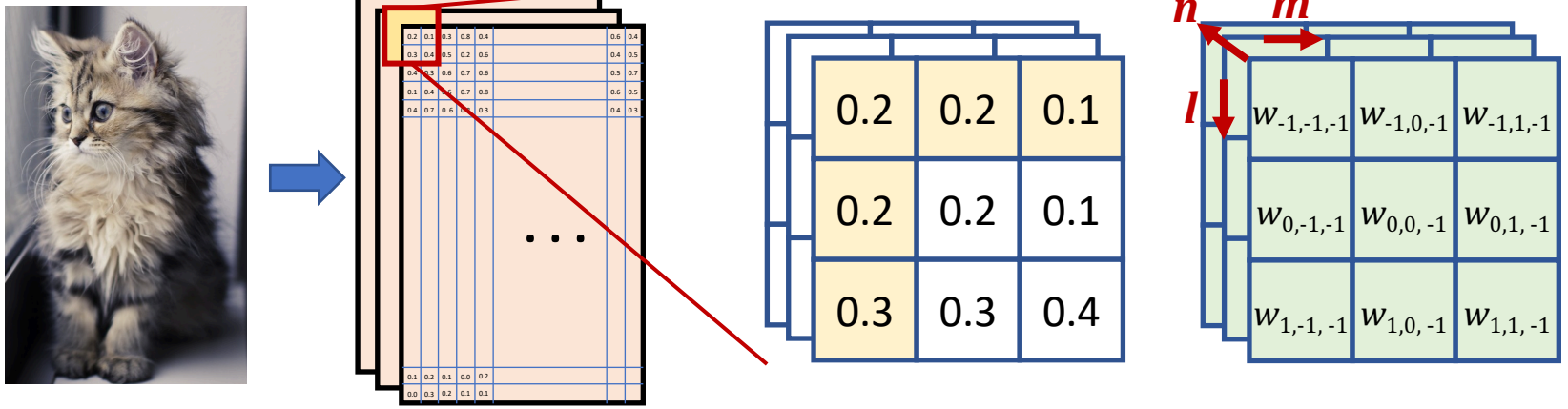
So that the output is the same size as its input

Convolutional Neural Networks

Convolutional layer

$$w_b + \sum_{l,m,n=-1}^{1} x_{i+l,j+m,k+n} w_{l,m,n}$$

RGB image input $\boldsymbol{x}$



This is repeated for every pixel of the image (if *stride*=1)

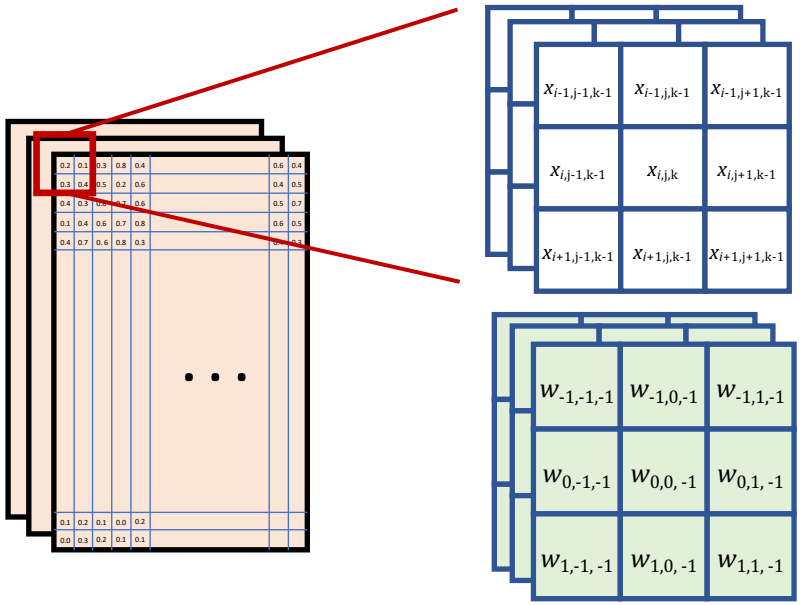
Image is padded for the bordering pixels

So that the output is the same size as its input

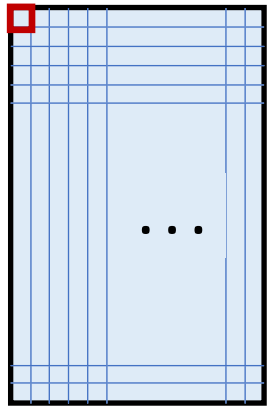
Convolutional Neural Networks

Convolutional layer

RGB image input



Conv layer output



This is repeated for every pixel of the image (if *stride*=1)

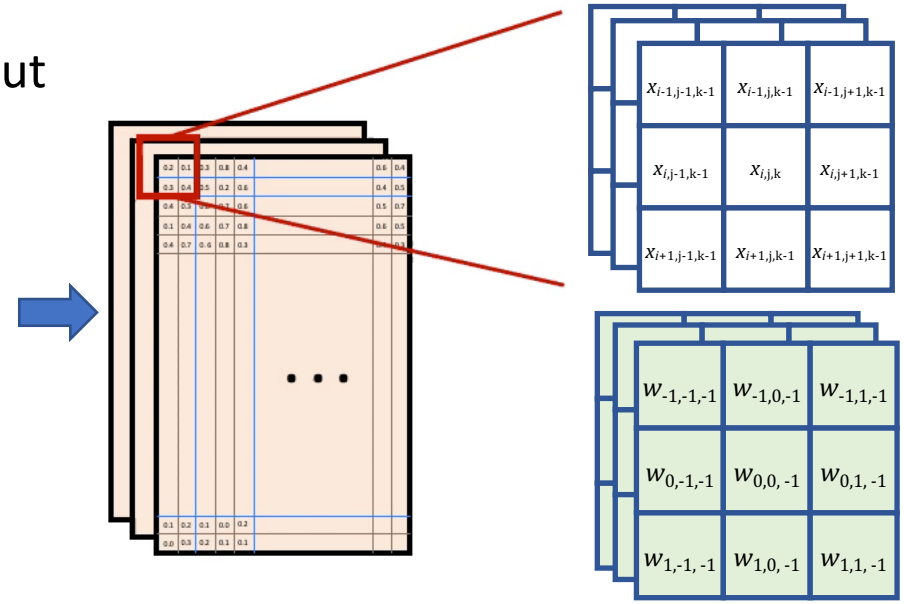
Image is padded for the bordering pixels

So that the output is the same size as its input

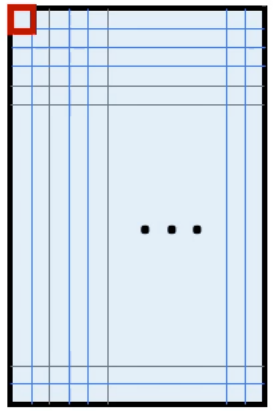
Convolutional Neural Networks

Convolutional layer

RGB image input



Conv layer output



This is repeated for every pixel of the image (if *stride*=1)

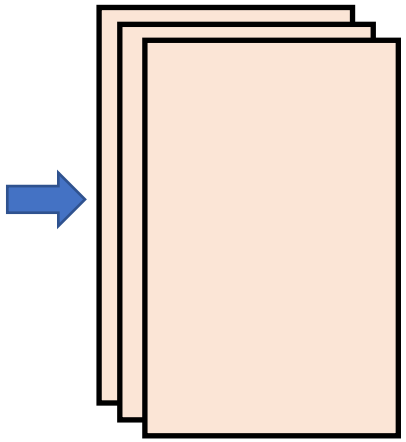
Image is padded for the bordering pixels

So that the output is the same size as its input

Convolutional Neural Networks

Convolutional layer

RGB image input x



$w^{(1)}$

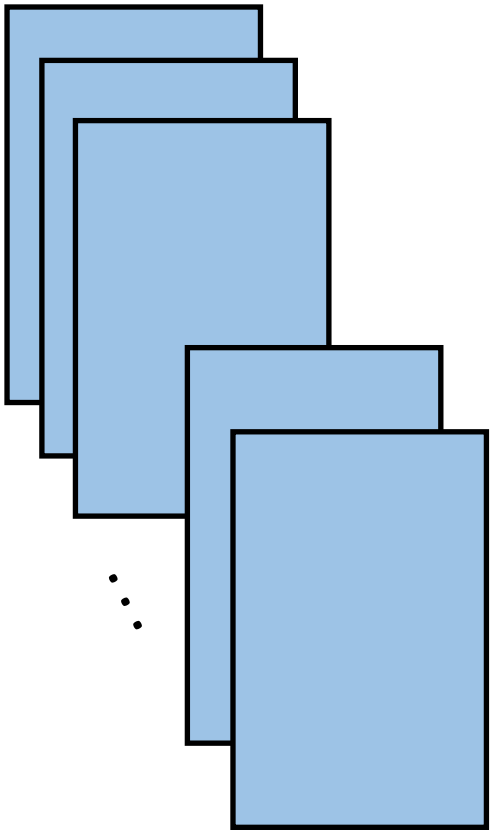
$w^{(2)}$

$w^{(3)}$

$w^{(c-1)}$

$w^{(c)}$

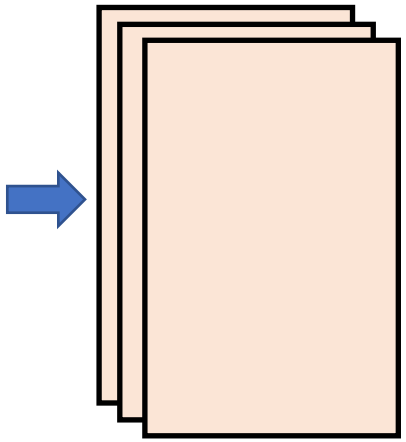
Conv layer output

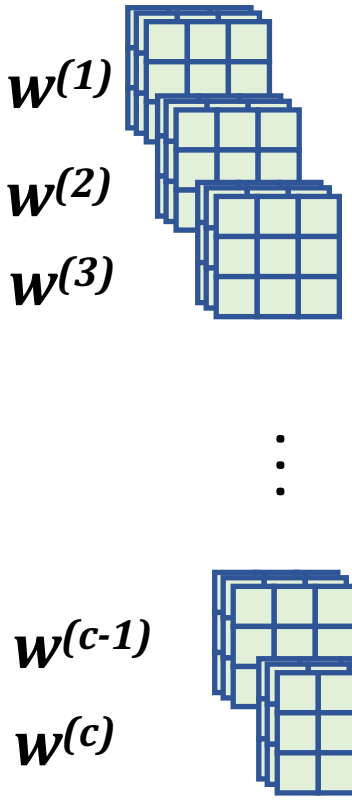


Convolutional Neural Networks

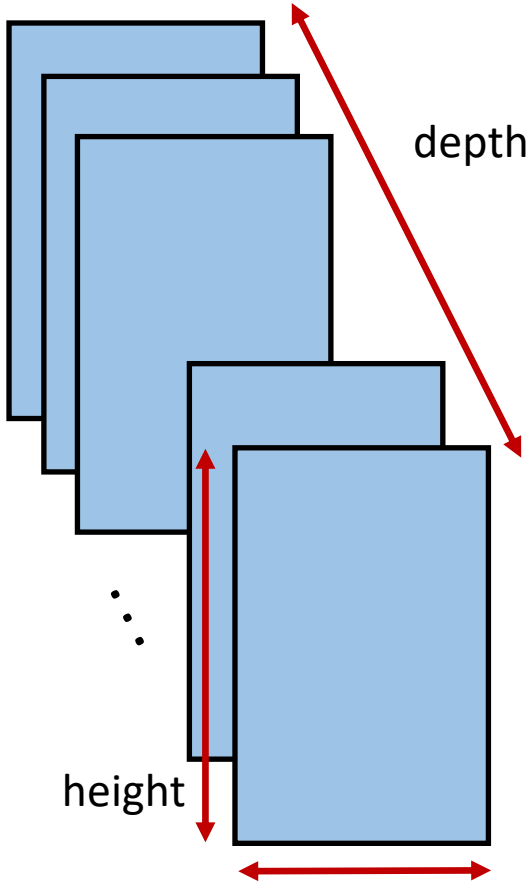
Convolutional layer

RGB image input x





Conv layer output

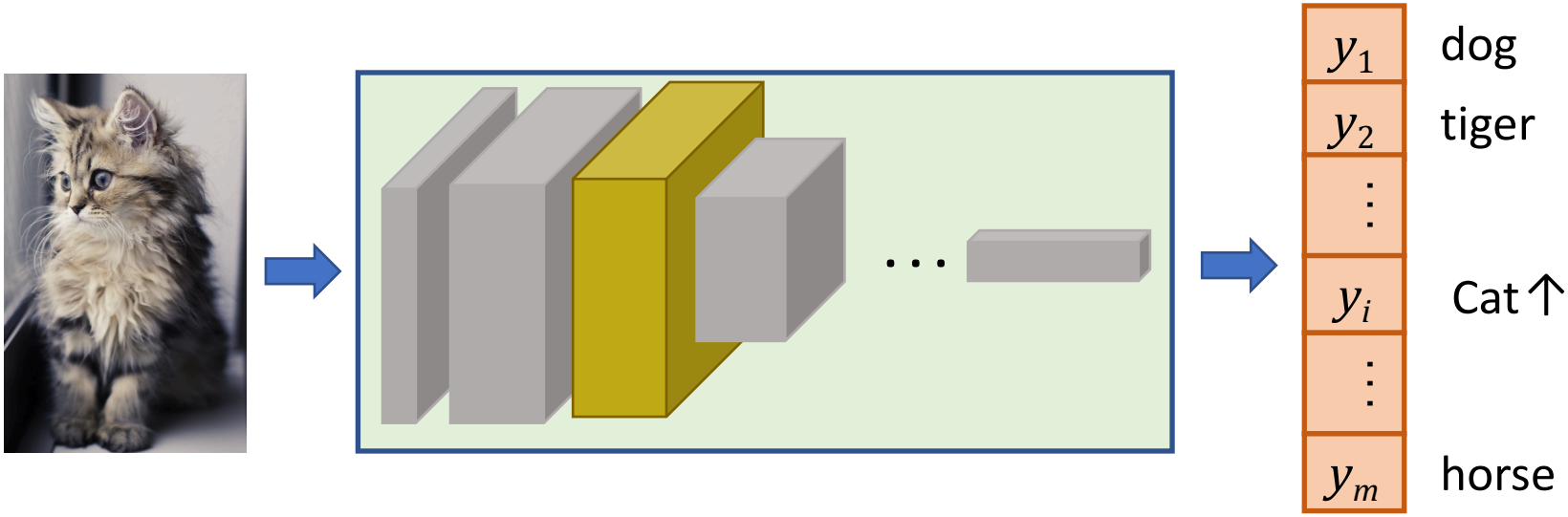


This can be used as the input to another layer

Convolutional Neural Networks

Activation layer

- It introduces non-linearity to the network
- Non-linearity helps the network learn complex patterns
- The most common activations are Sigmoid, ReLU and Tanh

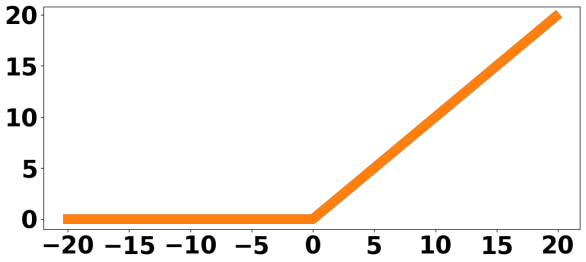


Convolutional Neural Networks

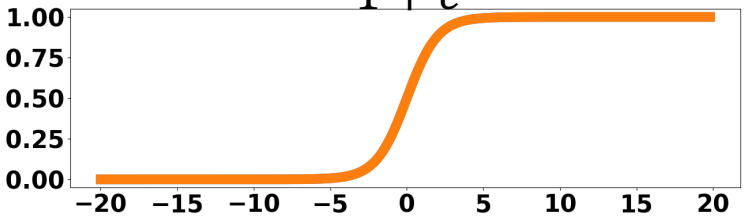
Activation layer

$$ReLU(x) = \max(0, x)$$

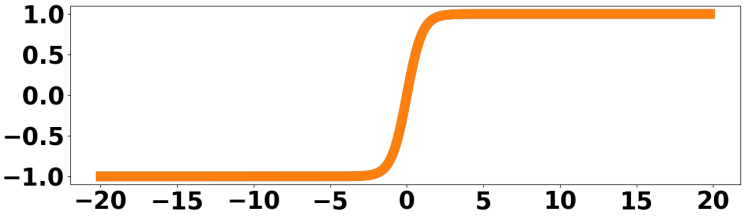
(*Rectified Linear Unit*)

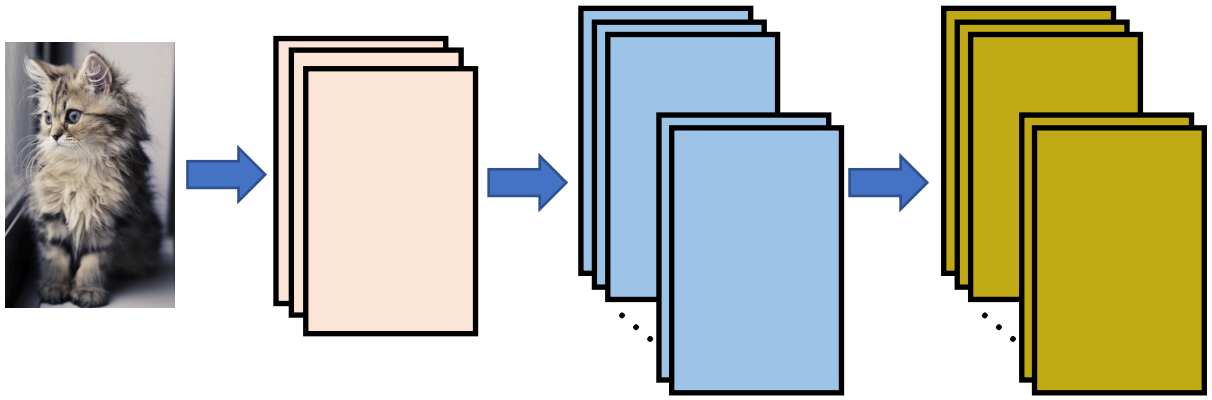


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



$$tanh(x)$$

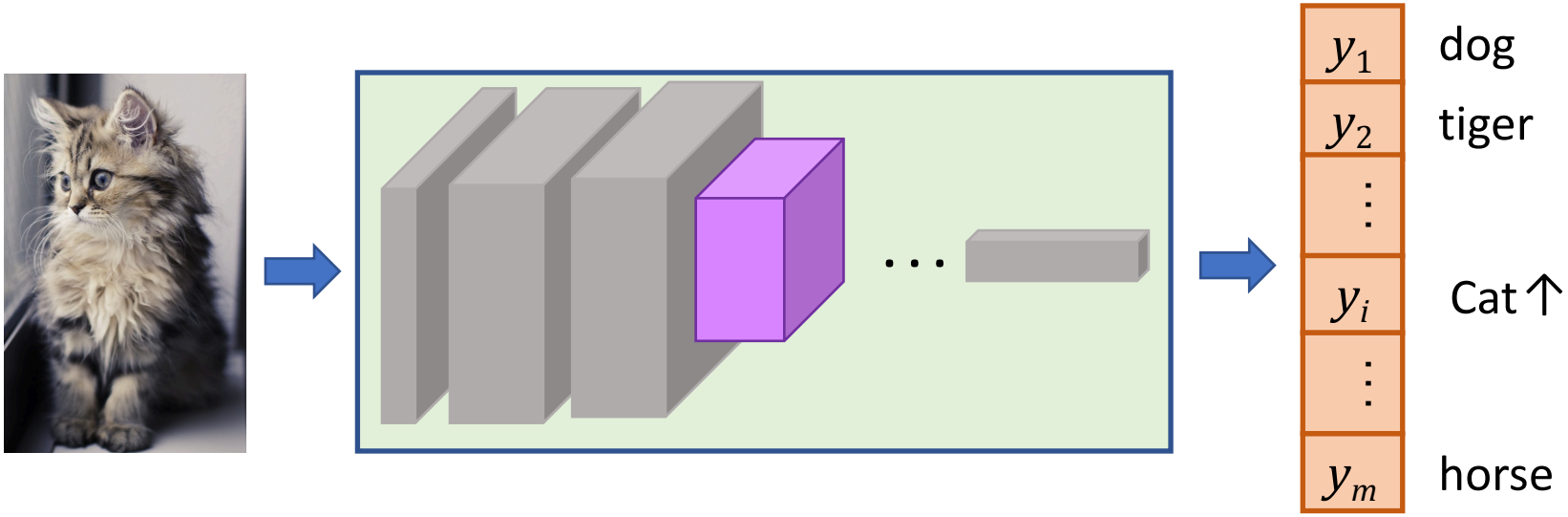




Convolutional Neural Networks

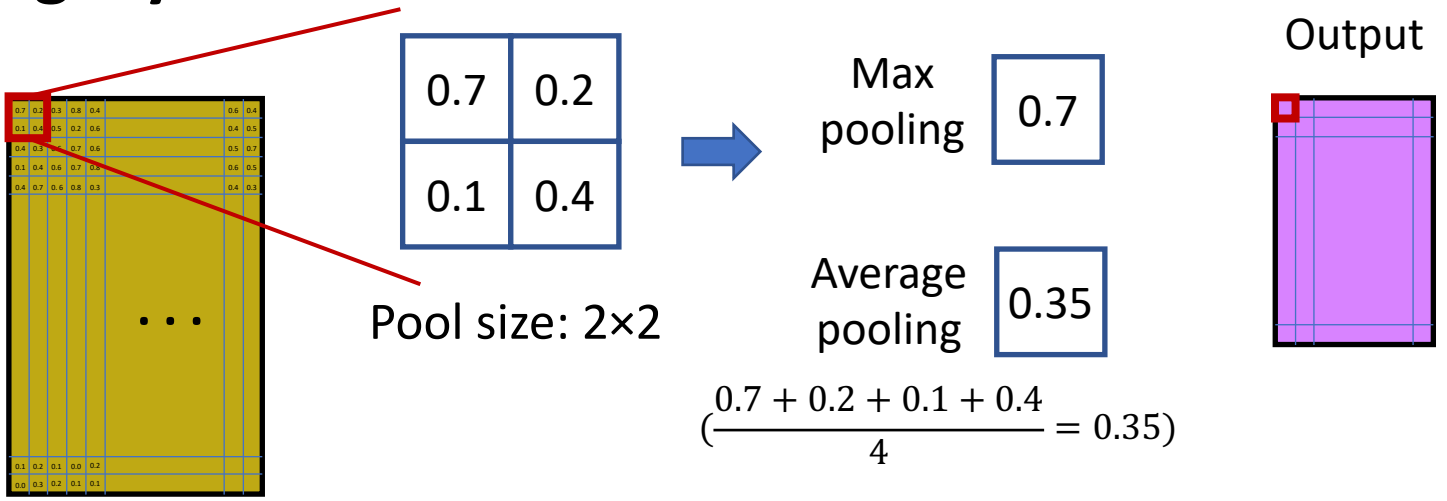
Pooling layer

- It down-samples its input
- It reduces the network sensitivity to the features' location
- Two common pooling layers are average and max pooling



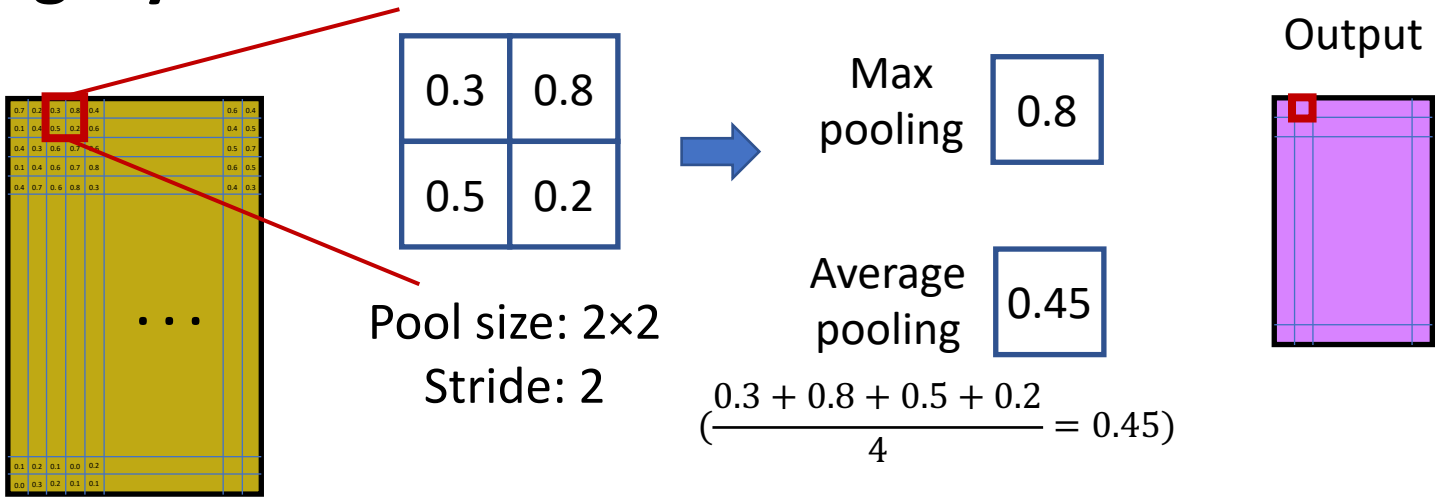
Convolutional Neural Networks

Pooling layer



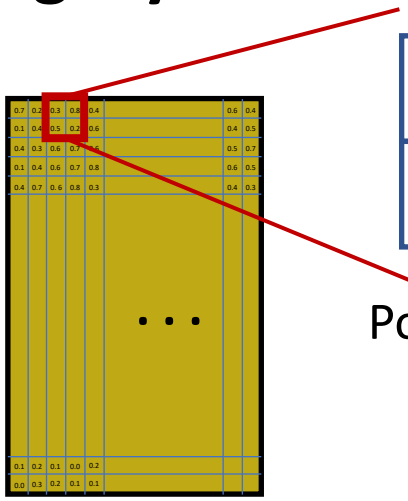
Convolutional Neural Networks

Pooling layer



Convolutional Neural Networks

Pooling layer



0.3	0.8
0.5	0.2

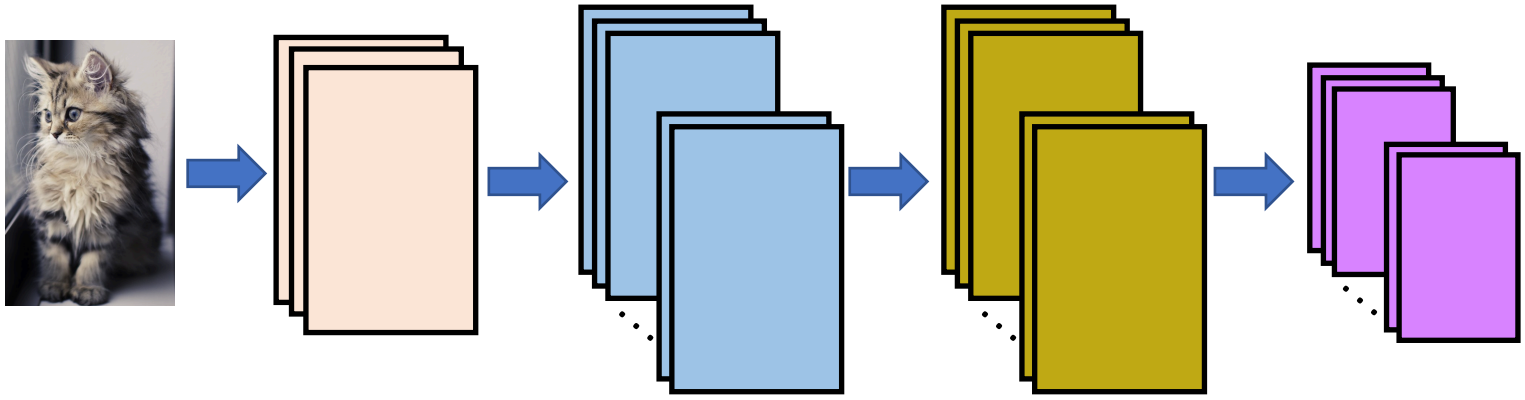
Pool size: 2×2
Stride: 2

Max pooling → 0.8

Average pooling → 0.45

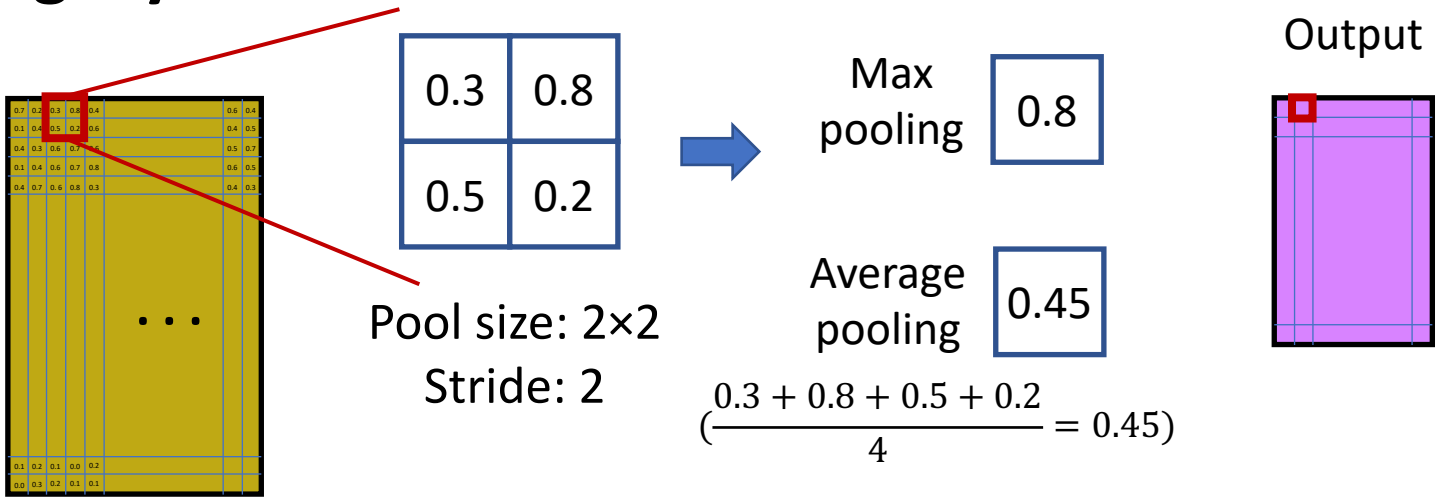
$$\left(\frac{0.3 + 0.8 + 0.5 + 0.2}{4} = 0.45\right)$$

Output



Convolutional Neural Networks

Pooling layer

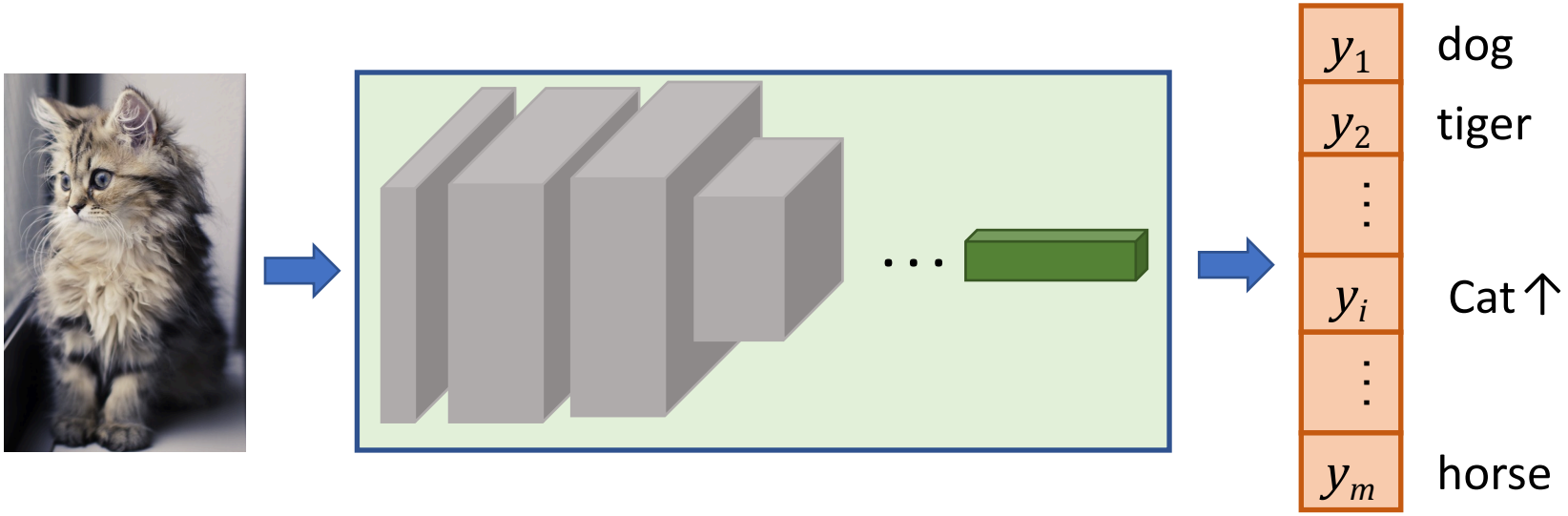


Pooling layer is not learnable

Convolutional Neural Networks

Fully-Connected layer

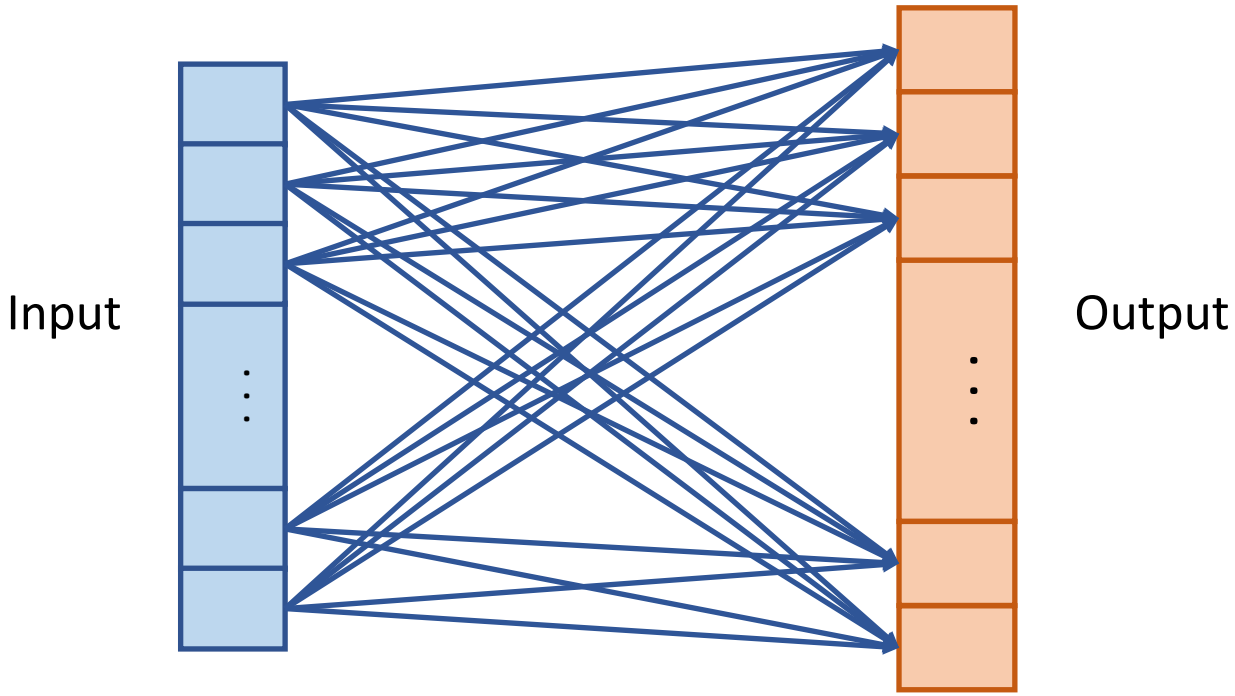
- It connects all its inputs to all the outputs
- It is usually used as the last layers of a network (before classification)



Convolutional Neural Networks

Fully-Connected layer

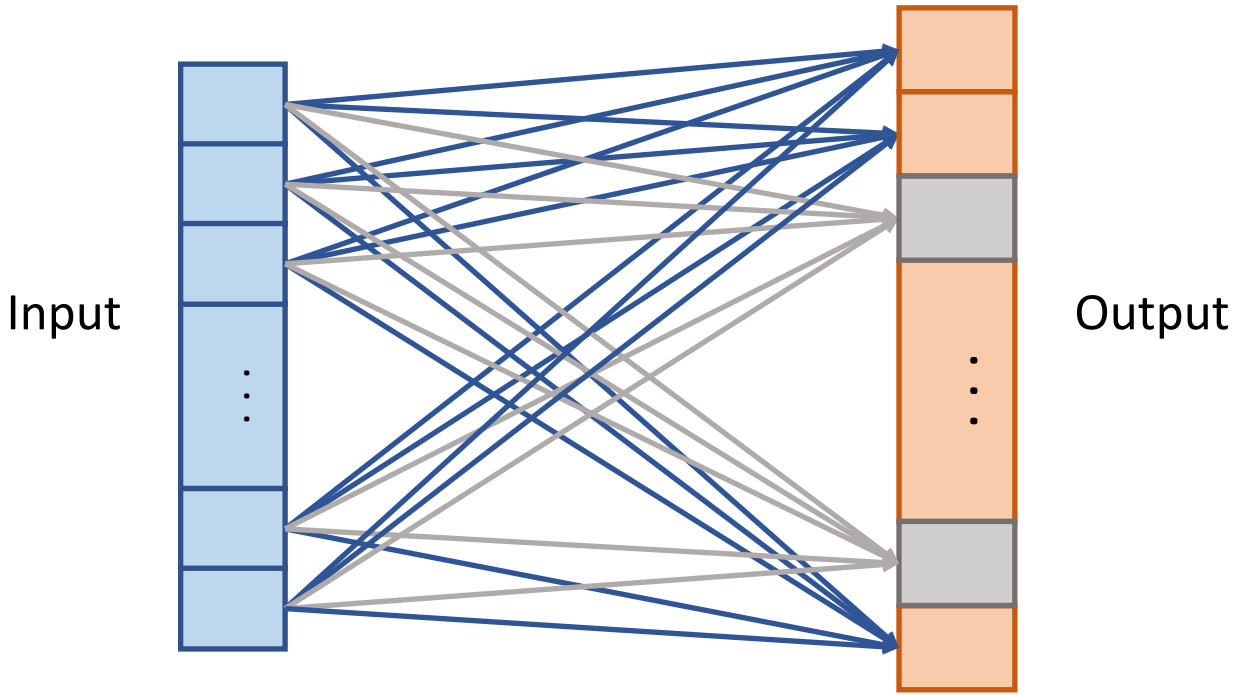
- It connects all its inputs to all the outputs
- It is usually used as the last layers of a network (before classification)



Convolutional Neural Networks

Drop-out layer

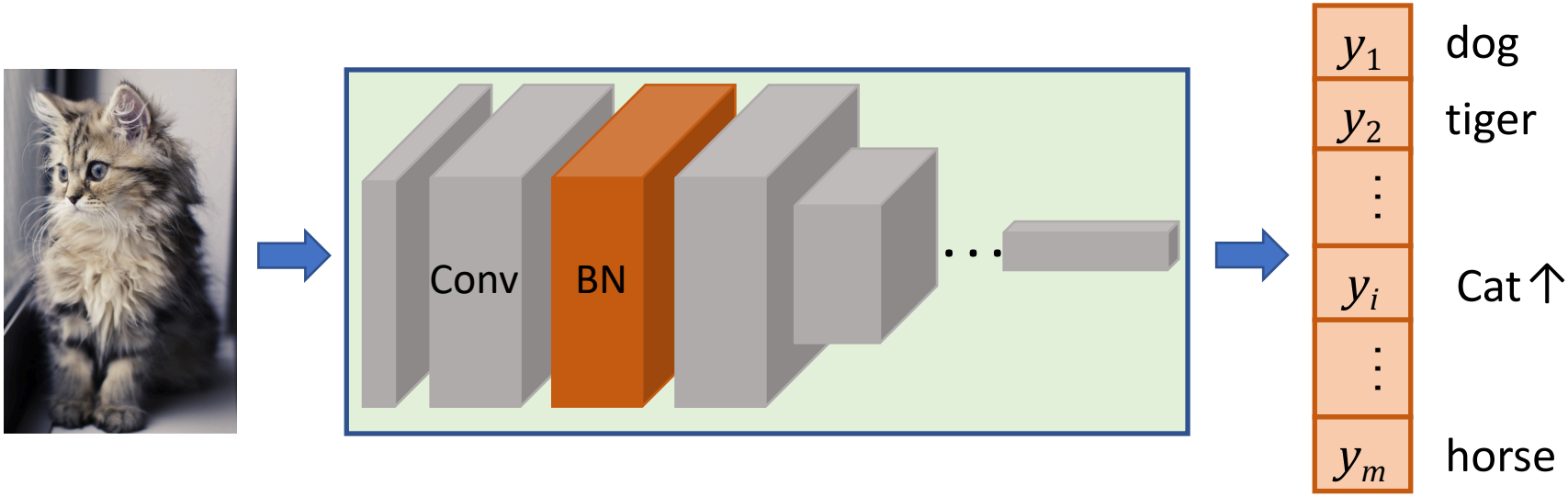
- It randomly drops out (sets to 0) neurons during training
- This prevents over-fitting (regularisation)



Convolutional Neural Networks

Batch-Normalisation layer

- It normalises its input across mini-batches
- It stabilises the network against the changes in the distribution of its input
- It helps the network generalise better to test data (regularisation)



Convolutional Neural Networks

Batch-Normalisation layer

It subtracts the mini-batch mean from its input
and divides it by the mini-batch standard deviation

$$\hat{x} = \frac{\boldsymbol{x} - m}{s}$$

This is then shifted and scaled by two learnable parameters

$$x^{BN} = \hat{x}.\gamma + \beta$$

The BN output has a distribution with
mean β and standard deviation γ

Popular
CNN Architectures

Popular CNN Architectures

CNN – VGG16

- It has 138 million parameters
- Its output layer has 1000 neurons

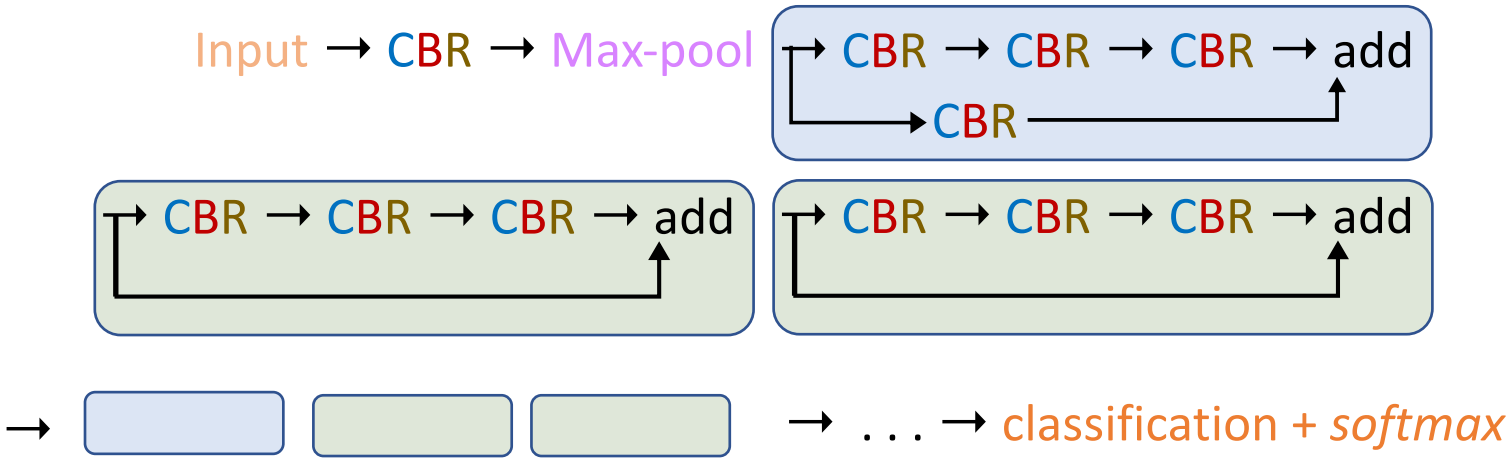
Input → Conv → ReLU → Conv → ReLU → Max-pool
→ Conv → ReLU → Conv → ReLU → Max-pool
→ Conv → ReLU → Conv → ReLU → Conv → ReLU → Max-pool
→ Conv → ReLU → Conv → ReLU → Conv → ReLU → Max-pool
→ Conv → ReLU → Conv → ReLU → Conv → ReLU → Max-pool
→ FC → ReLU → FC → ReLU → classification + *softmax*

Simonyan and Zisserman, Very Deep Convolutional Networks for Large Scale Image Recognition, 2014

Popular CNN Architectures

CNN – ResNet

- It uses residual (shortcut) connections
- It has increased the number of layers (152)
- It has 26 million parameters (CBR: Conv → BN → ReLU)



He et al, Deep Residual Learning for Image Recognition, 2016

Popular CNN Architectures

CNN for image classification

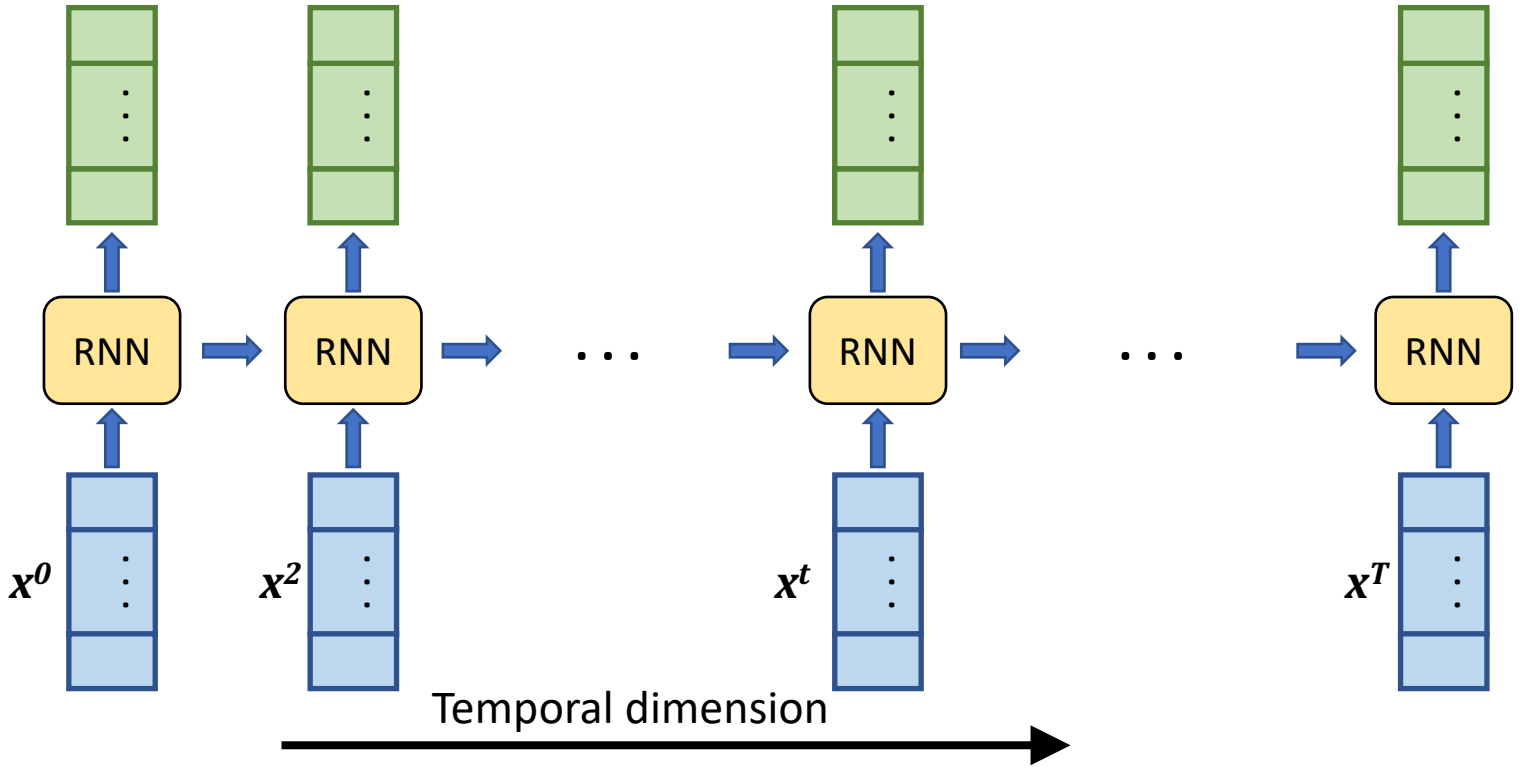
- AlexNet
- Inception-v1/ v2/ v4
- Inception-ResNets
- Xception
- DenseNet
- ResNeXt

Other Popular Neural Networks

Other Popular Neural Networks

Recurrent Neural Networks (RNN)

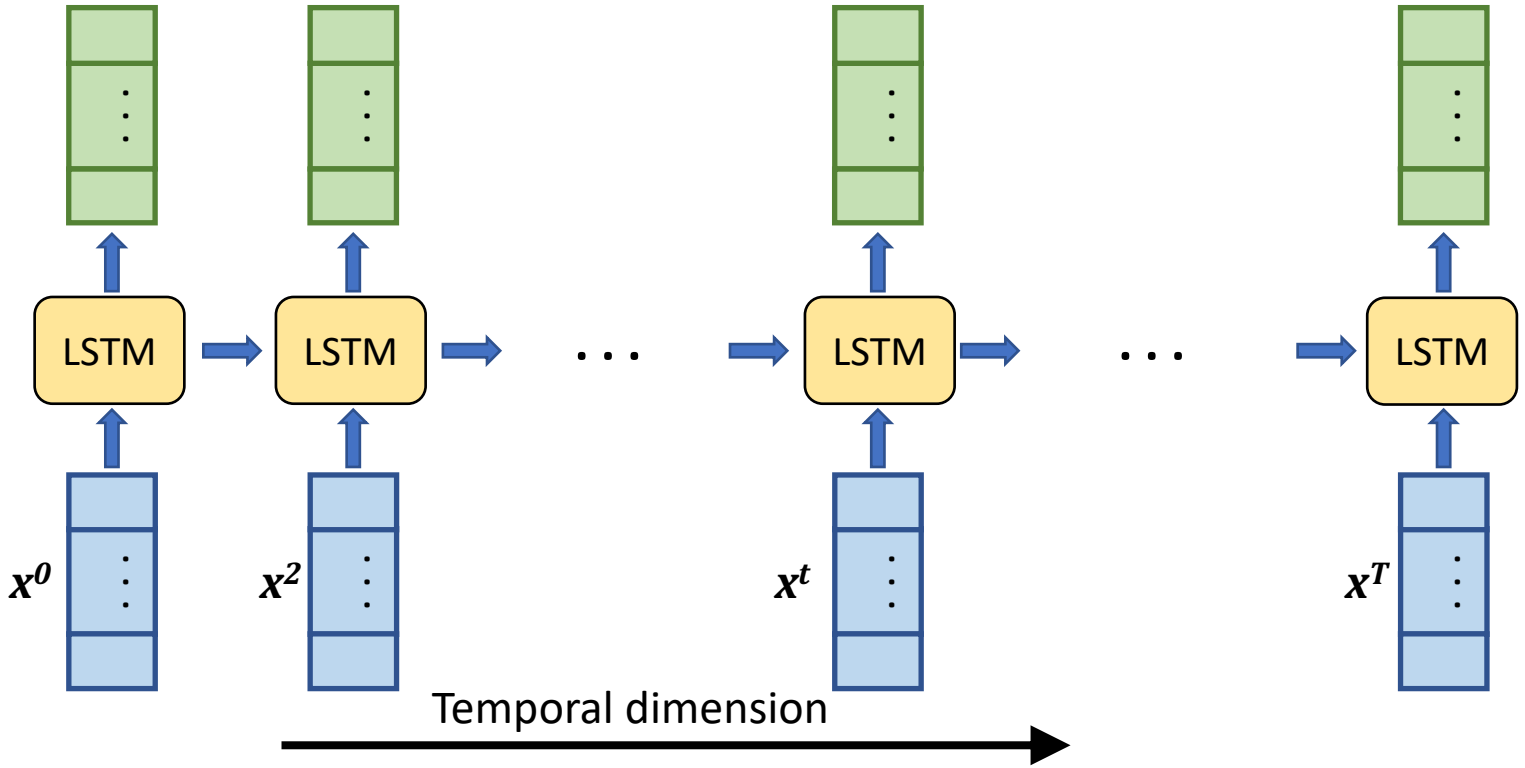
- They process sequences of data
- They back-propagate loss through time



Other Popular Neural Networks

Long Short-Term Memory Networks (LSTM)

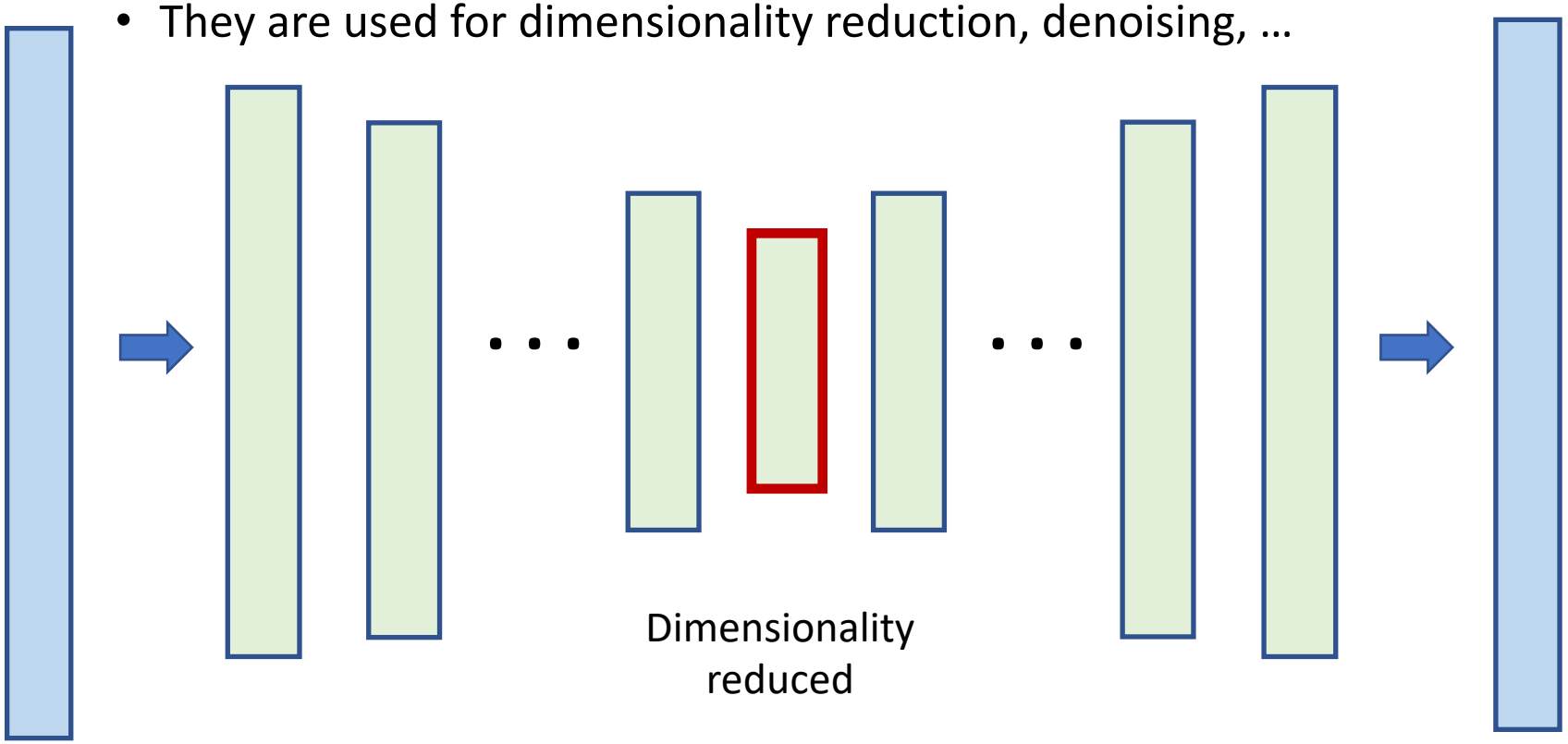
- They are a special type of Recurrent Networks
- They can store information in their memory for long durations



Other Popular Neural Networks

Auto-Encoders

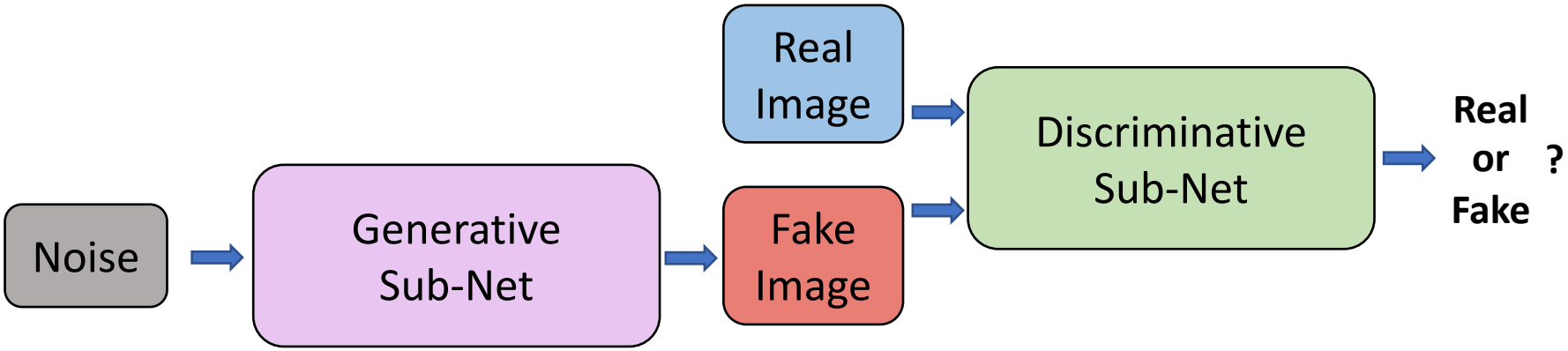
- They reconstruct their input
- They are used for dimensionality reduction, denoising, …



Other Popular Neural Networks

Generative Adversarial Networks (GAN)

- They are generative models – e.g. when trained they can generate fake images out of noise
- They have two sub-network (discriminative and generative)
- The two sub-networks compete against each other (adversarial)



Summary and Conclusion

- We saw different types of layers in a CNN

  Convolutional, Activation, Pooling

  Fully-Connected, Drop-out, Batch Normalisation, ...

- We saw some popular neural networks
  - CNN (VGG and ResNet)
  - RNN and LSTM
  - Auto-Encoder
  - GAN

Thanks for your attention