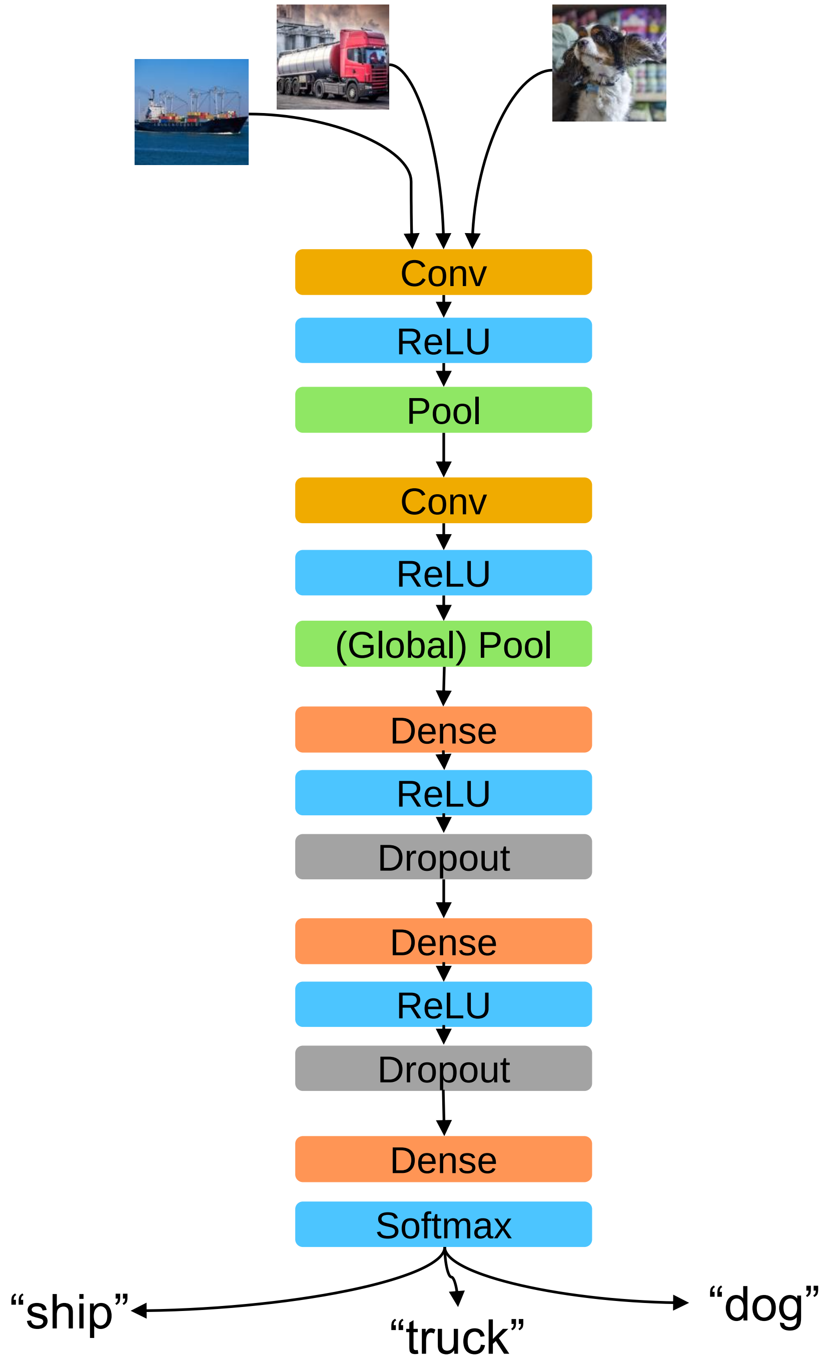
Week 4: Convolutional Networks

Unit 1: Introduction to CNNs

Introduction to CNNs
Overview

Content:
- Biological inspiration
- Challenges for computer vision
- The need for spatial invariance
- A naïve approach to neural networks for computer vision
- Convolutional neural networks

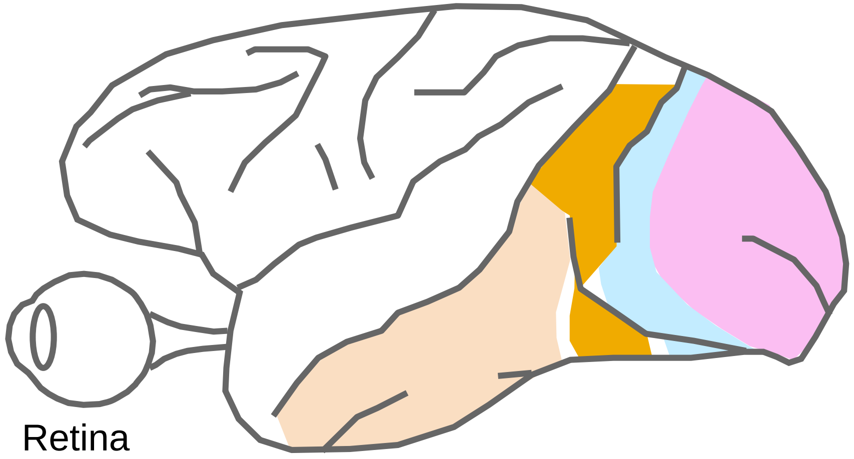


Introduction to CNNs

Biological inspiration

Example: The primate visual cortex

- Signals arriving from the retina are processed hierarchically by subsequent brain areas
- This is reminiscent of processing by subsequent layers of a deep artificial neural network



David Daniel Cox, Thomas Dean, *Neural Networks and Neuroscience-Inspired Computer Vision*, Current Biology, Volume 24, Issue 18, 2014, Pages R921-R929

Introduction to CNNs

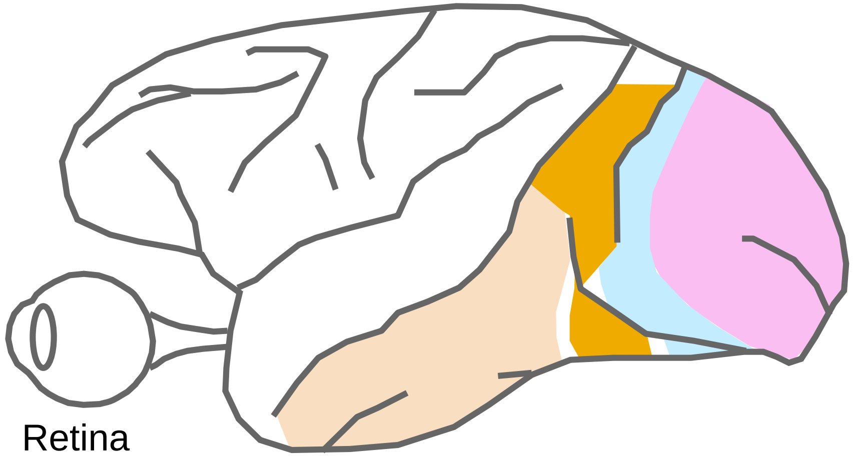
Biological inspiration

Example: The primate visual cortex

- Signals arriving from the retina are processed hierarchically by subsequent brain areas
- This is reminiscent of processing by subsequent layers of a deep artificial neural network

Of course, the correspondence is not perfect:

- In the primate visual cortex, there are many forward and backward connections



David Daniel Cox, Thomas Dean, *Neural Networks and Neuroscience-Inspired Computer Vision*, Current Biology, Volume 24, Issue 18, 2014, Pages R921-R929

Introduction to CNNs
Biological inspiration

Hubel and Wiesel (1950s and ’60s) studied feline visual cortex

Two types of cells identified:

- *Simple* cells fire in response to stimuli of a particular shape and orientation

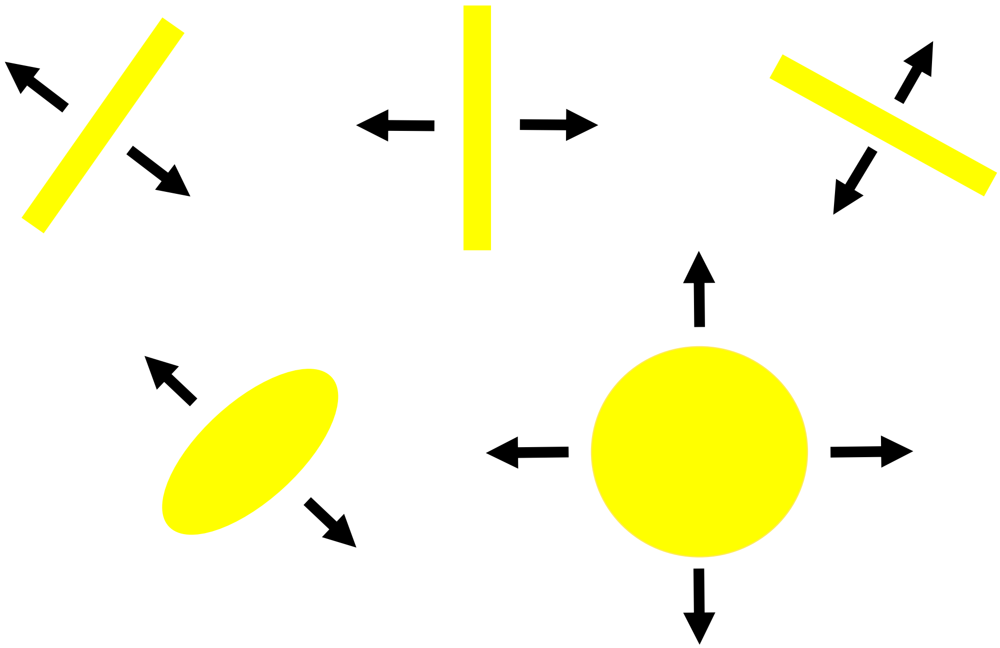
Hubel, David H., and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." The Journal of physiology 160.1 (1962): 106

Introduction to CNNs
Biological inspiration

Hubel and Wiesel (1950s and ’60s) studied feline visual cortex

Two types of cells identified:

- *Simple* cells fire in response to stimuli of a particular shape and orientation
- *Complex* cells additionally fire only when the stimulus moves in a particular direction

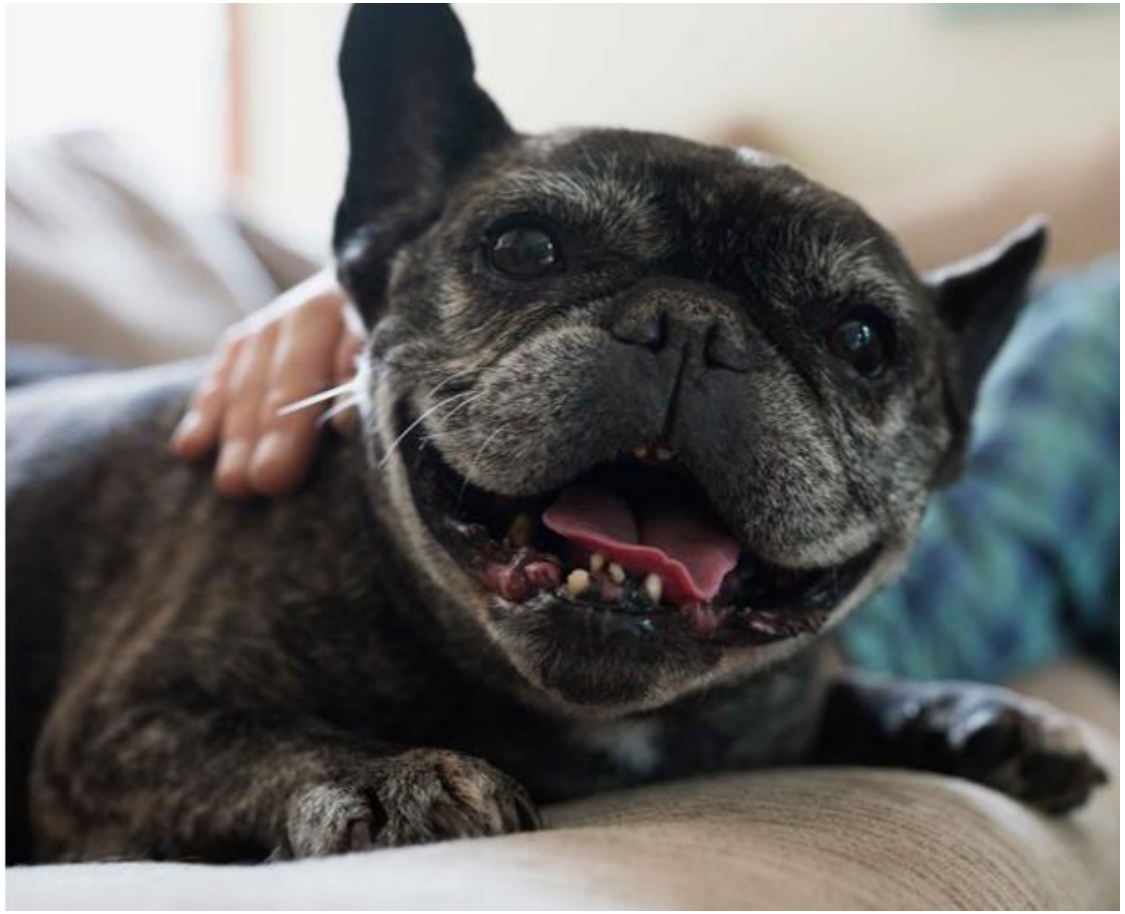


Hubel, David H., and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." The Journal of physiology 160.1 (1962): 106

Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole



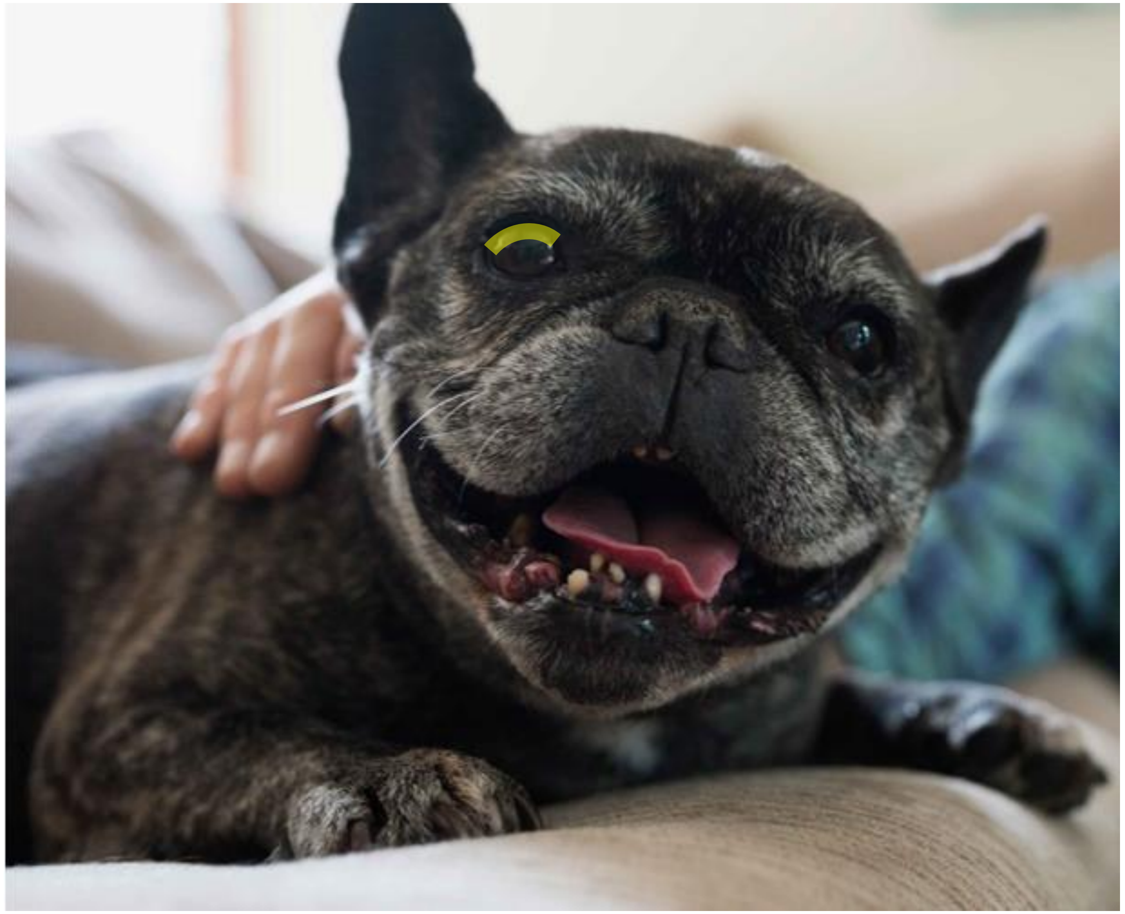
Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye



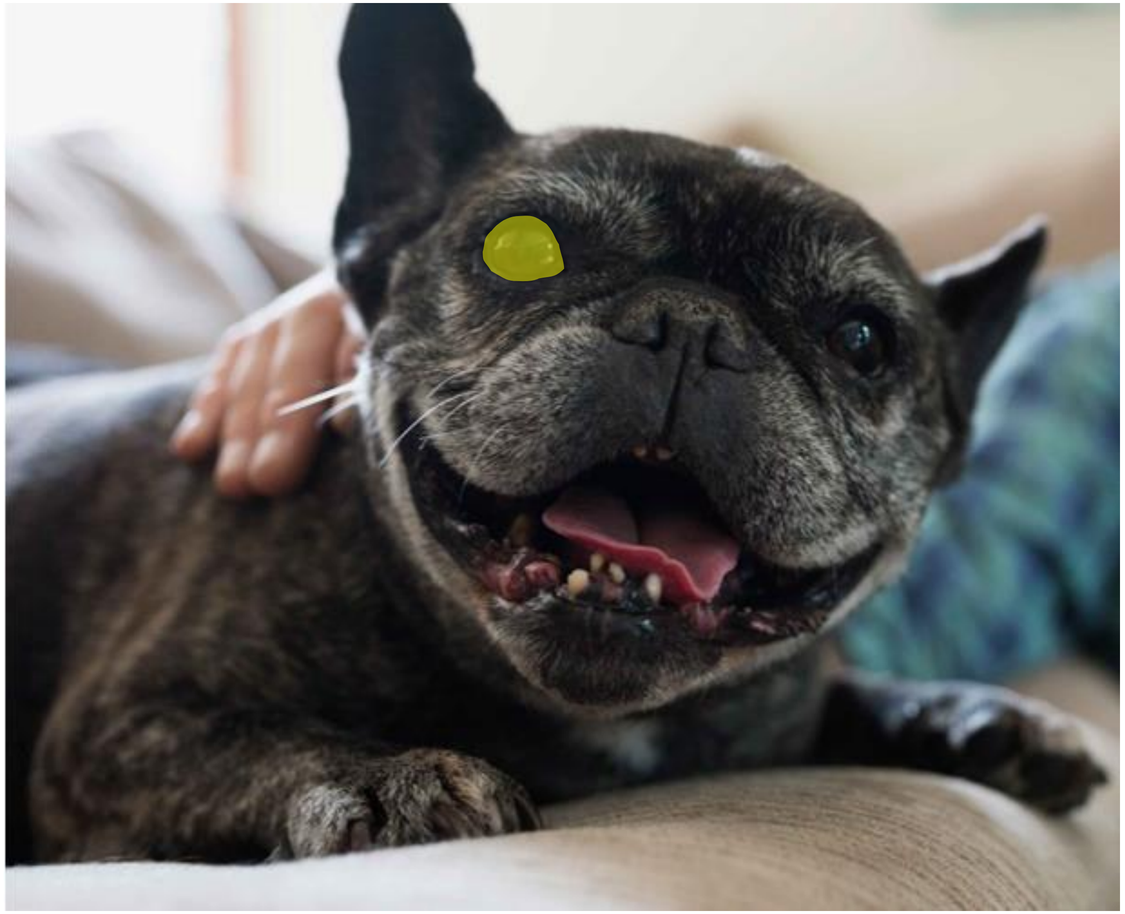
Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye
- Combine those component parts into an abstract representation of an eye



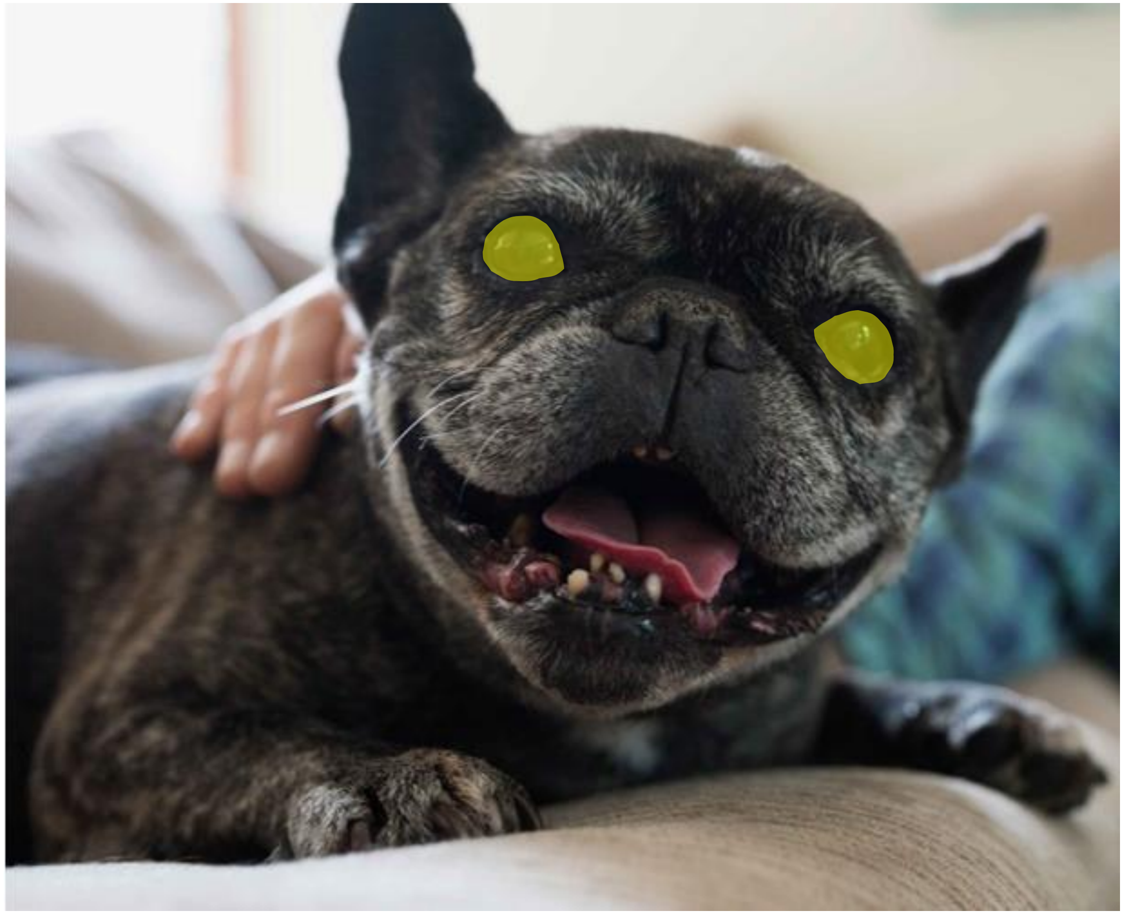
Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye
- Combine those component parts into an abstract representation of an eye
- Identify other body parts of the dog, like a second eye



Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye
- Combine those component parts into an abstract representation of an eye
- Identify other body parts of the dog, like a second eye, ears

Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye
- Combine those component parts into an abstract representation of an eye
- Identify other body parts of the dog, like a second eye, ears, snout

Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye
- Combine those component parts into an abstract representation of an eye
- Identify other body parts of the dog, like a second eye, ears, snout, legs, etc.

Introduction to CNNs

Challenges for computer vision – From images to abstract representations

Object recognition involves combining many irregular features into a whole

For example, to recognize a dog, an animal's brain must:

- Identify edges of shapes, like the arcs forming the outline of an eye
- Combine those component parts into an abstract representation of an eye
- Identify other body parts of the dog, like a second eye, ears, snout, legs, etc.
- Combine these parts into an abstract representation of a dog

Introduction to CNNs

Challenges for computer vision – Data dimensionality

Image data is very high-dimensional

- A 512×512 RGB image has $512 \cdot 512 \cdot 3 = 786{,}432$ features

Introduction to CNNs

Challenges for computer vision – Invariance

Does this image contain a dog?

Introduction to CNNs

Challenges for computer vision – Invariance

Does this image contain a dog?

Many invariances must be learned by a “naïve” machine learning model, e.g.:

- the position of the dog in the image (spatial invariance)

Introduction to CNNs

Challenges for computer vision – Invariance

Does this image contain a dog?

Many invariances must be learned by a “naïve” machine learning model, e.g.:

- the position of the dog in the image (spatial invariance)
- the size of the dog in the image (scale invariance)

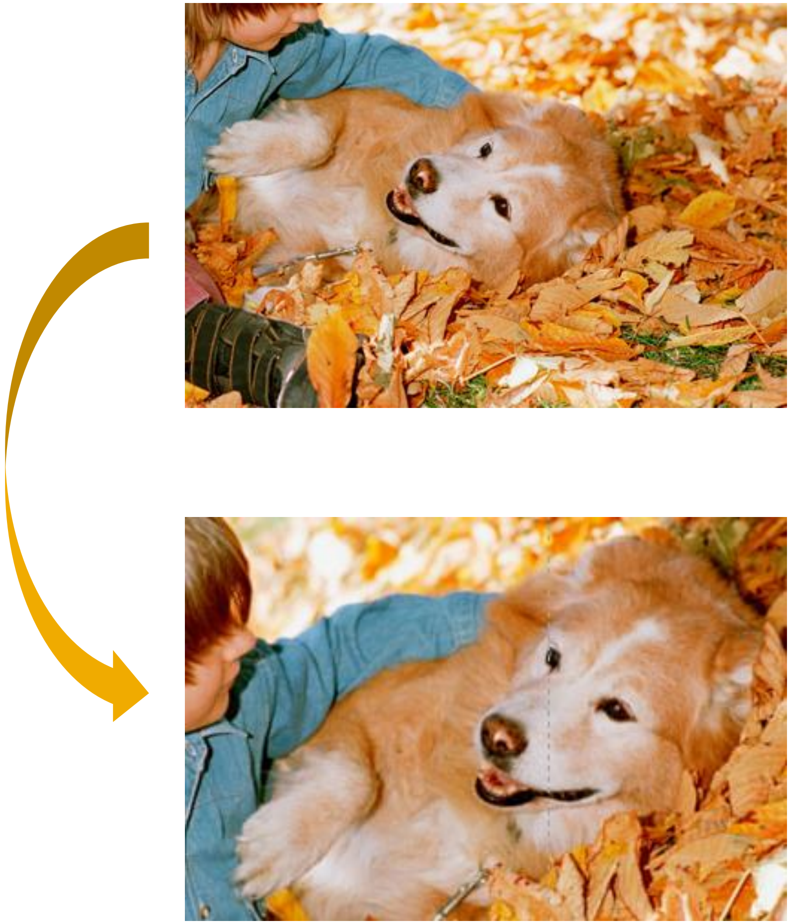
Introduction to CNNs

Challenges for computer vision – Invariance

Does this image contain a dog?

Many invariances must be learned by a “naïve” machine learning model, e.g.:

- the position of the dog in the image (spatial invariance)
- the size of the dog in the image (scale invariance)
- the angle at which the image was taken (2D rotational invariance)



Introduction to CNNs

Challenges for computer vision – Invariance

Does this image contain a dog?

Many invariances must be learned by a “naïve” machine learning model, e.g.:

- the position of the dog in the image (spatial invariance)
- the size of the dog in the image (scale invariance)
- the angle at which the image was taken (2D rotational invariance)
- the orientation of the dog and the parts of its body in 3D space

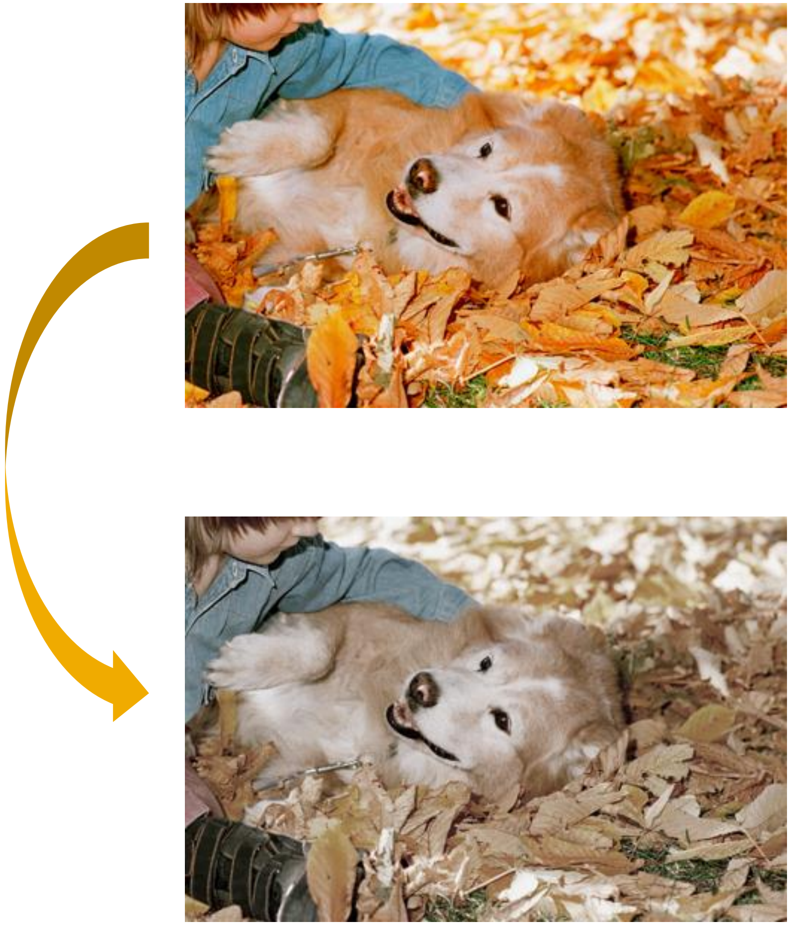
Introduction to CNNs

Challenges for computer vision – Invariance

Does this image contain a dog?

Many invariances must be learned by a “naïve” machine learning model, e.g.:

- the position of the dog in the image (spatial invariance)
- the size of the dog in the image (scale invariance)
- the angle at which the image was taken (2D rotational invariance)
- the orientation of the dog and the parts of its body in 3D space
- and more…



Introduction to CNNs

The need for spatial invariance

Does this image contain a dog?

Let’s focus on spatial invariance.

- Where the dog appears is irrelevant
- The same is true for the component questions:
  – *Is there an edge in some orientation?*
  – *Is this an eye?*
  – *Does it have legs?*

Introduction to CNNs

The need for spatial invariance

Does this image contain a dog?

Let's focus on spatial invariance.

- Where the dog appears is irrelevant
- The same is true for the component questions:
  - *Is there an edge in some orientation?*
  - *Is this an eye?*
  - *Does it have legs?*
- *However, the spatial relationships between the components of the image remain important*

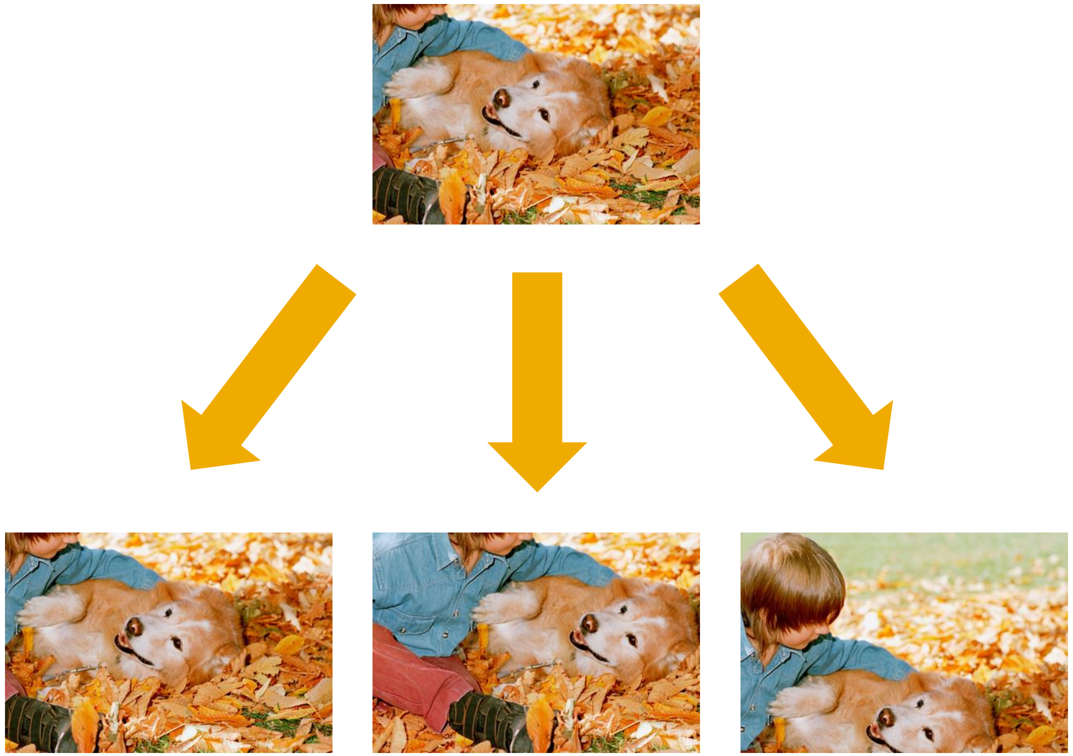
dog

not a dog

Introduction to CNNs

A naïve approach to neural networks for computer vision

A first idea for dealing with spatial invariance:
Augment the data by generating many translations of the images in the training set, then apply a feed-forward neural network.



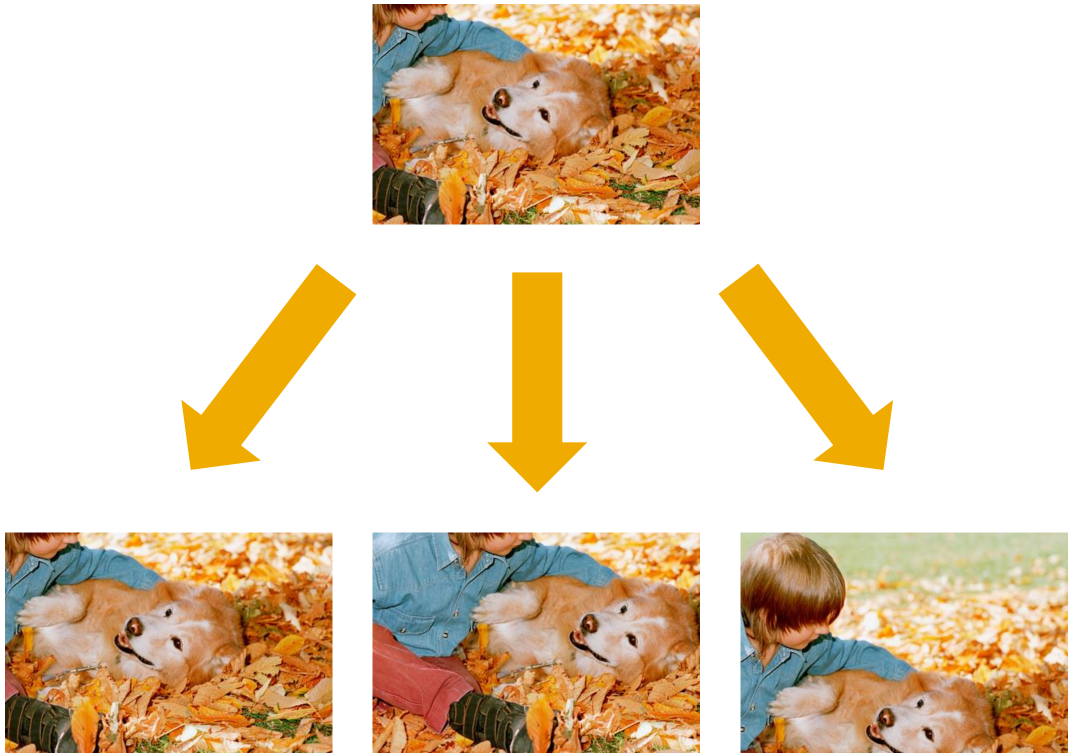
Introduction to CNNs

A naïve approach to neural networks for computer vision

A first idea for dealing with spatial invariance: Augment the data by generating many translations of the images in the training set, then apply a feed-forward neural network.

Problems:

- Lengthens training time by massively increasing size of data set
- The feed-forward network requires many parameters (for a 512×512 RGB image, $786{,}432$ times the size of the first hidden layer in the input layer alone)



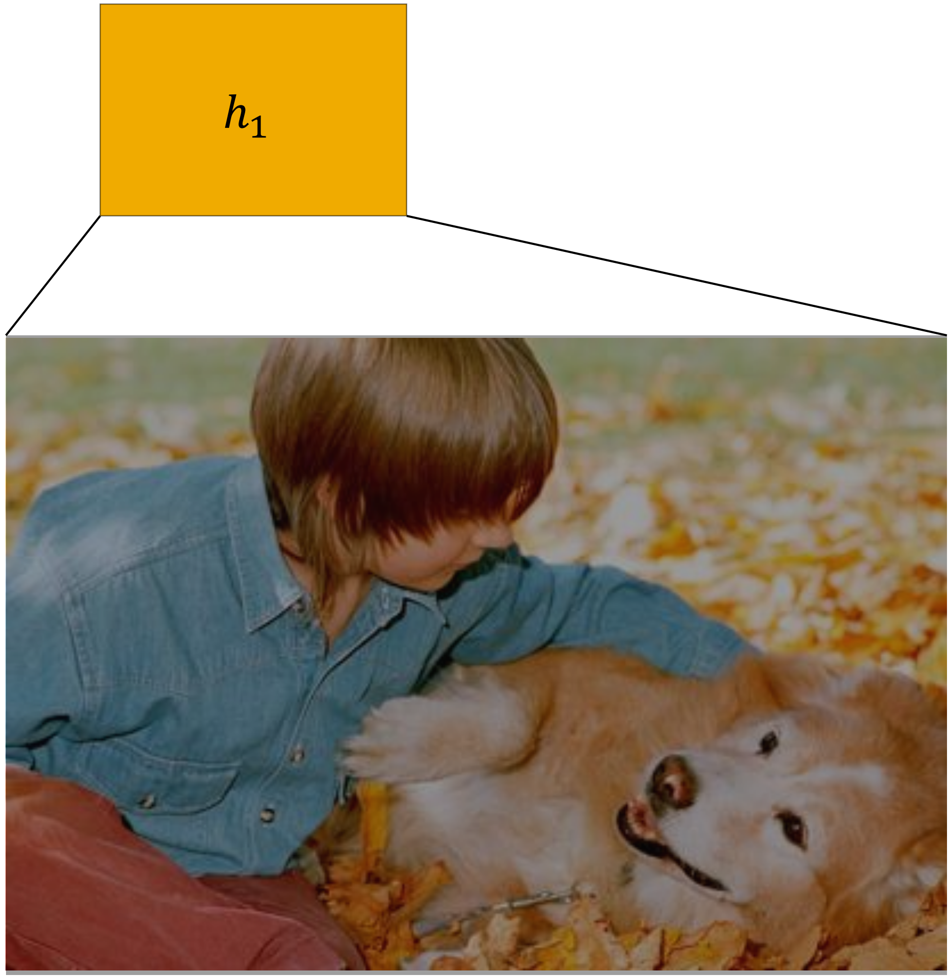
Introduction to CNNs

A naïve approach to neural networks for computer vision

A feed-forward neural network connects each pixel to each hidden node

A lot of parameters:

$$image\ width\ \times\ image\ height\ \times\ size\ of\ hidden\ layer$$



Introduction to CNNs

Convolutional neural networks

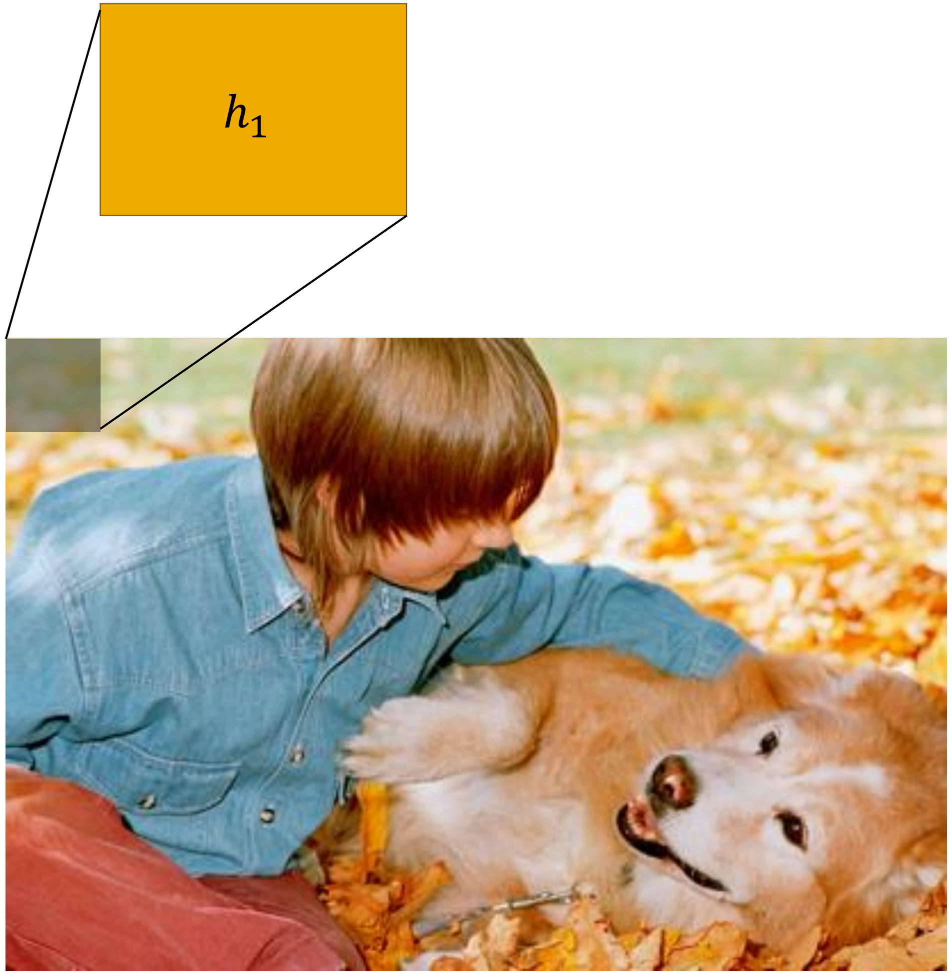
CNNs apply *the same transformation*
(feature maps) to each patch of the image

Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

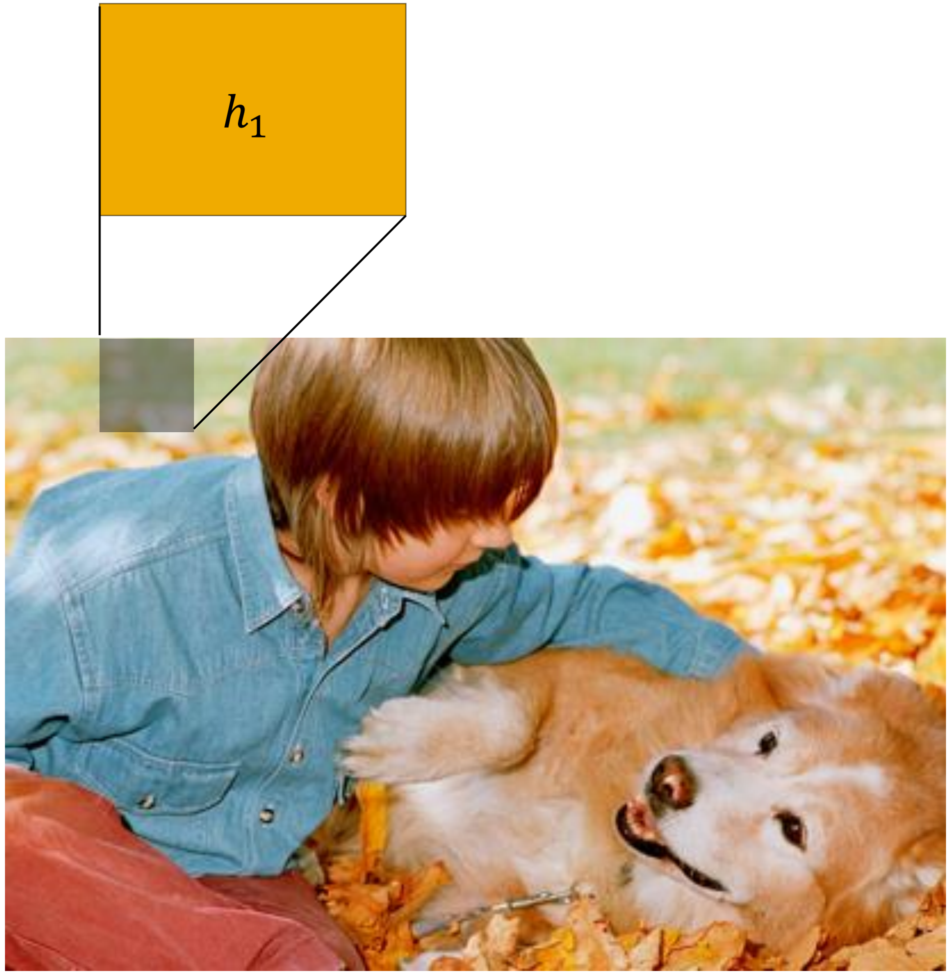
Each small patch is connected to the first hidden layer of the network



Introduction to CNNs
Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network

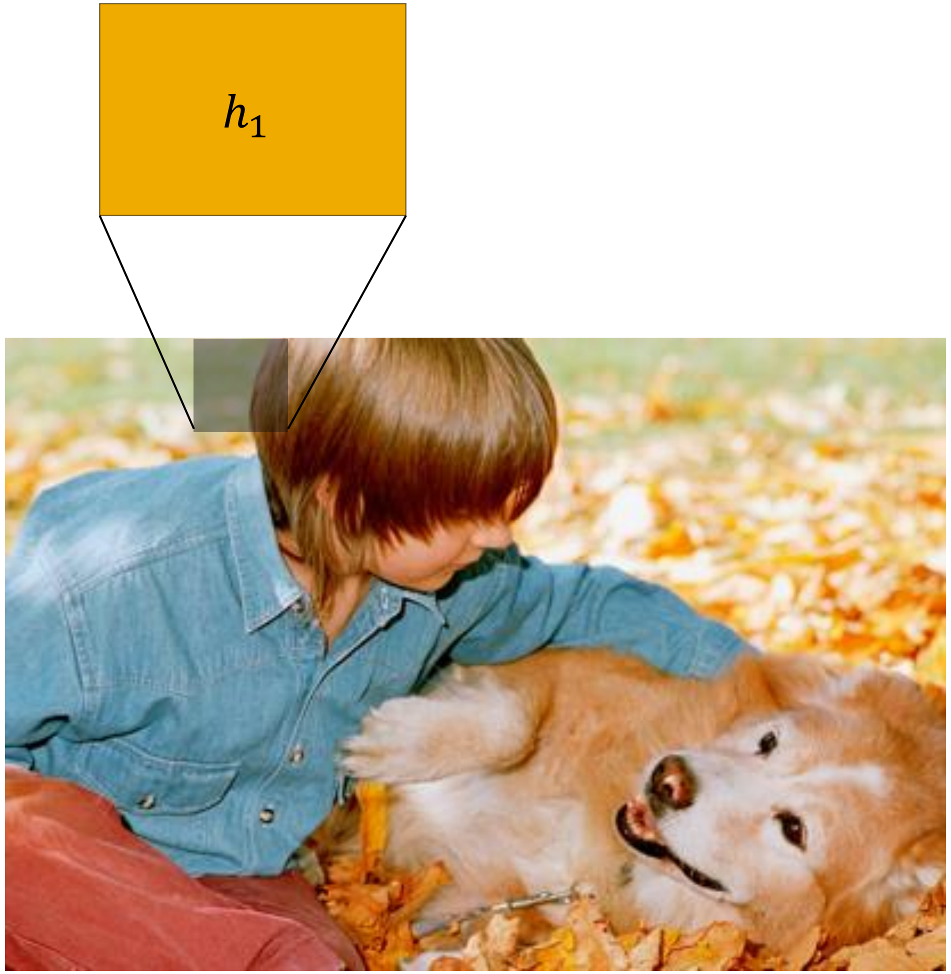


Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network

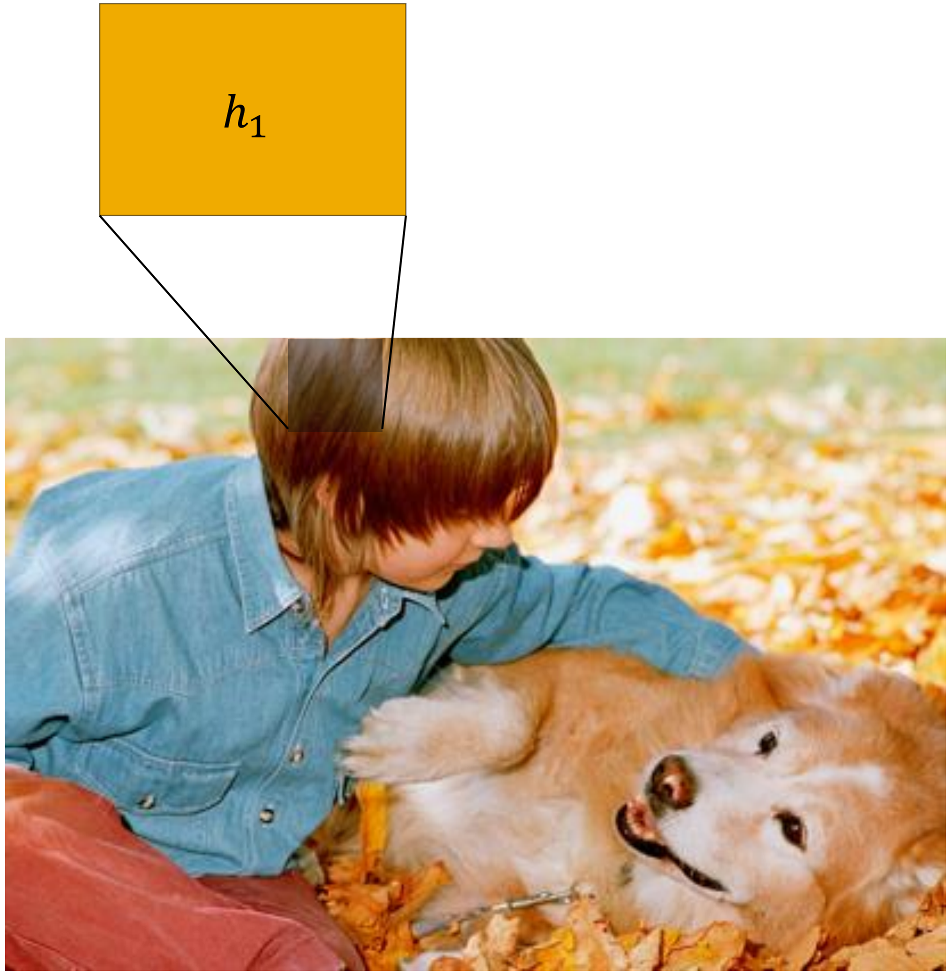


Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network

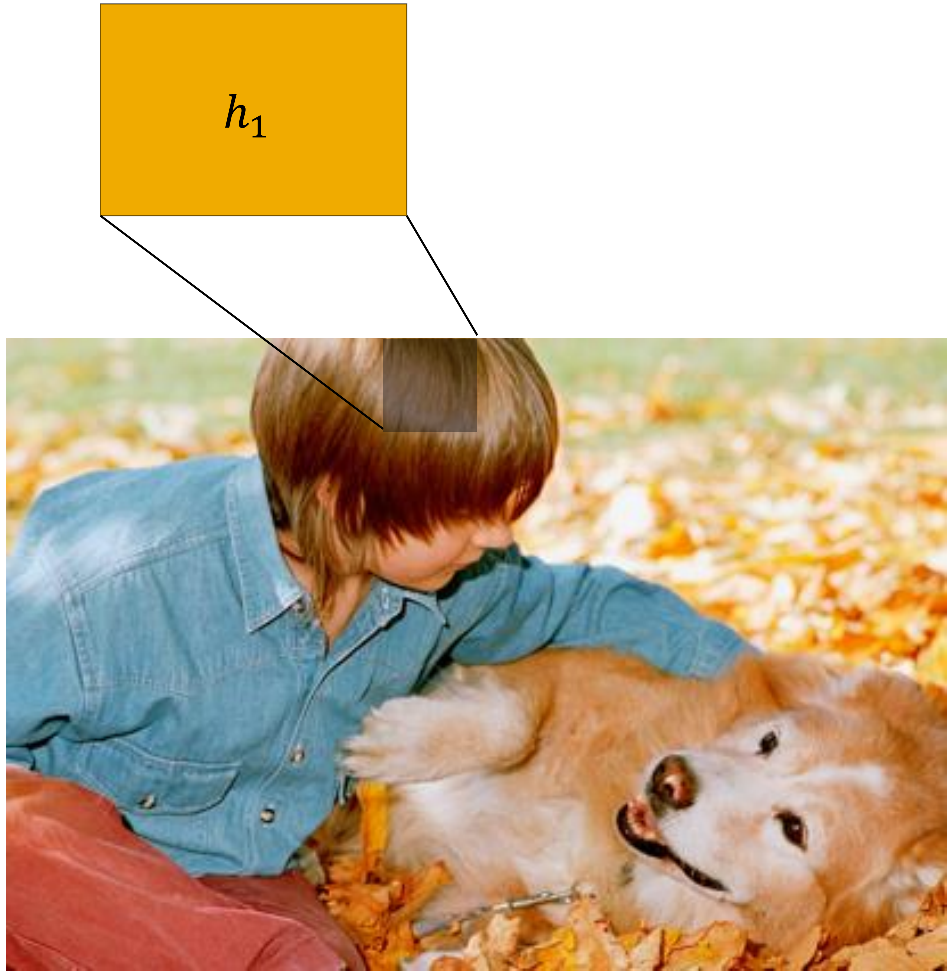


Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network

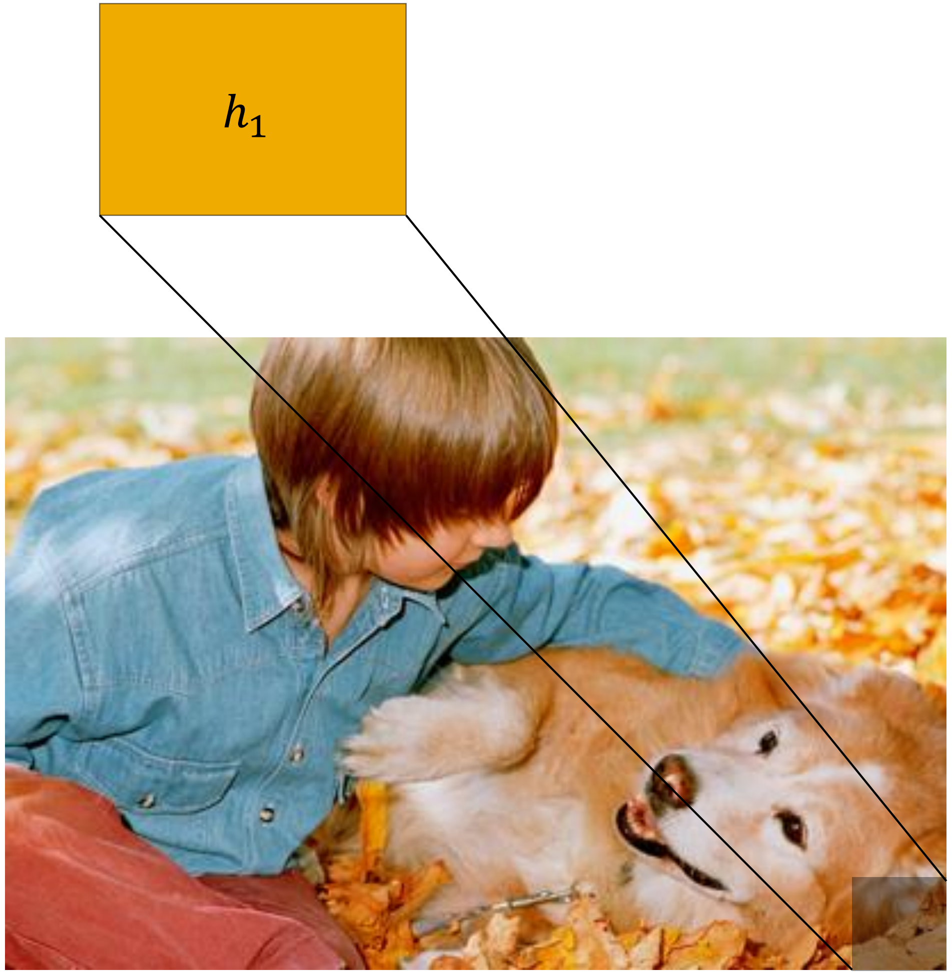


Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network



Introduction to CNNs

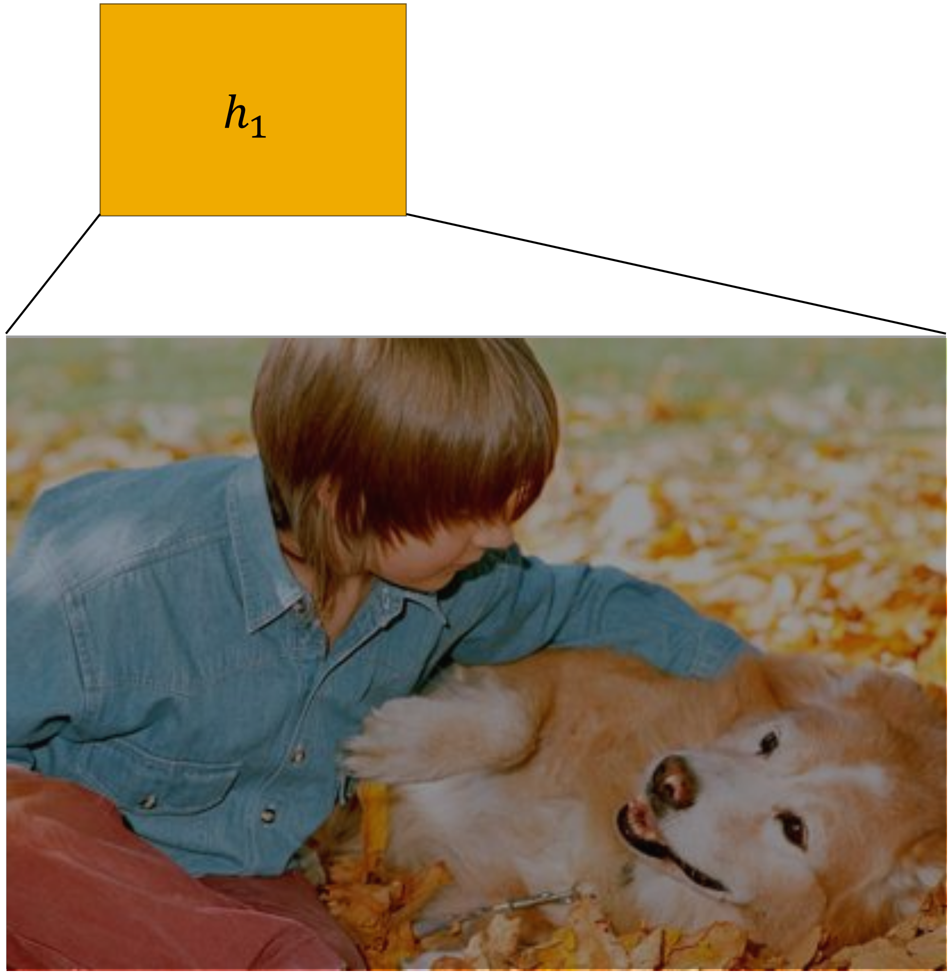
Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network

Contrast with a feed-forward neural network, which connects each pixel to each hidden node

Significantly more parameters:
$image\ width \times image\ heigh \times size\ of\ hidden\ layer$



Introduction to CNNs
Convolutional neural networks

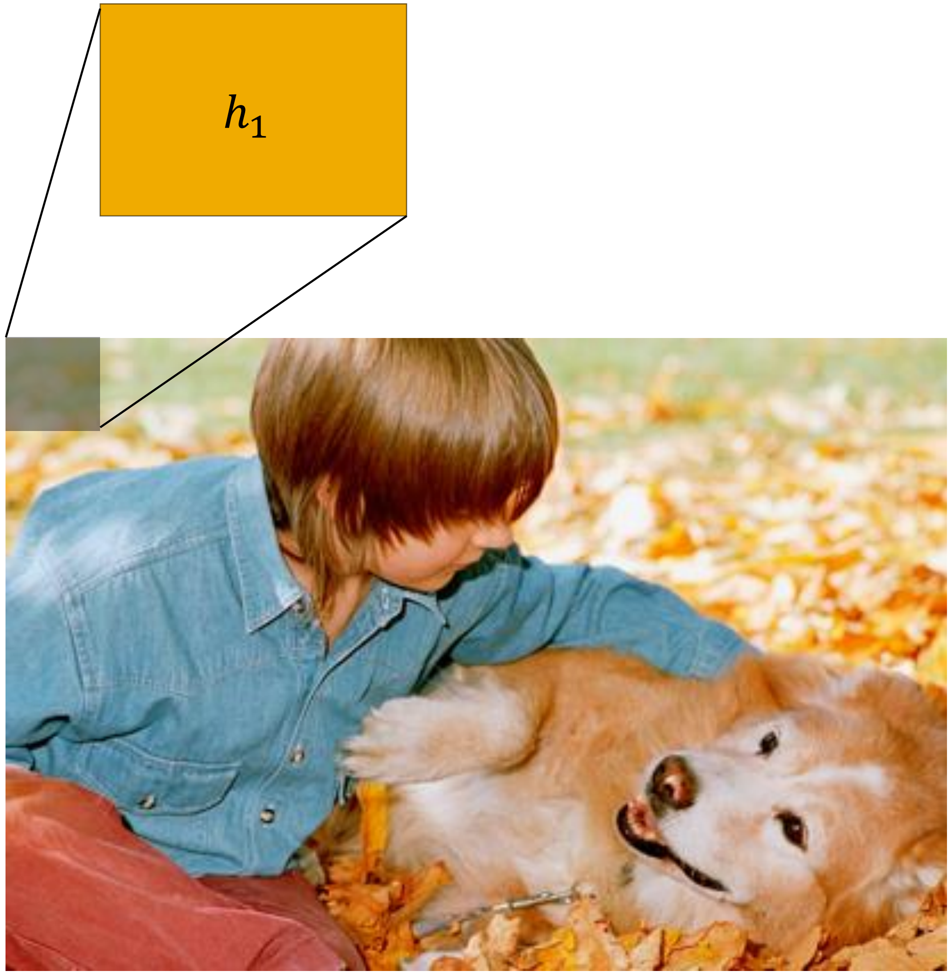
CNNs apply *the same transformation* (feature maps) to each patch of the image

Each small patch is connected to the first hidden layer of the network

Contrast with a feed-forward neural network, which connects each pixel to each hidden node

Significantly more parameters:
$image\ width\ \times\ image\ heigh\ \times\ size\ of\ hidden\ layer$

instead of:
$patch\ width\ \times\ patch\ height\ \times\ size\ of\ hidden\ layer$



Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

In deeper layers of the network, the features become effectively larger and more abstract

Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

In deeper layers of the network, the features become effectively larger and more abstract

Introduction to CNNs

Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

In deeper layers of the network, the features become effectively larger and more abstract

Advantages:

- Fewer parameters to tune
- Spatial invariance built in mathematically
- Retains spatial relationship between features

Introduction to CNNs
Convolutional neural networks

CNNs apply *the same transformation* (feature maps) to each patch of the image

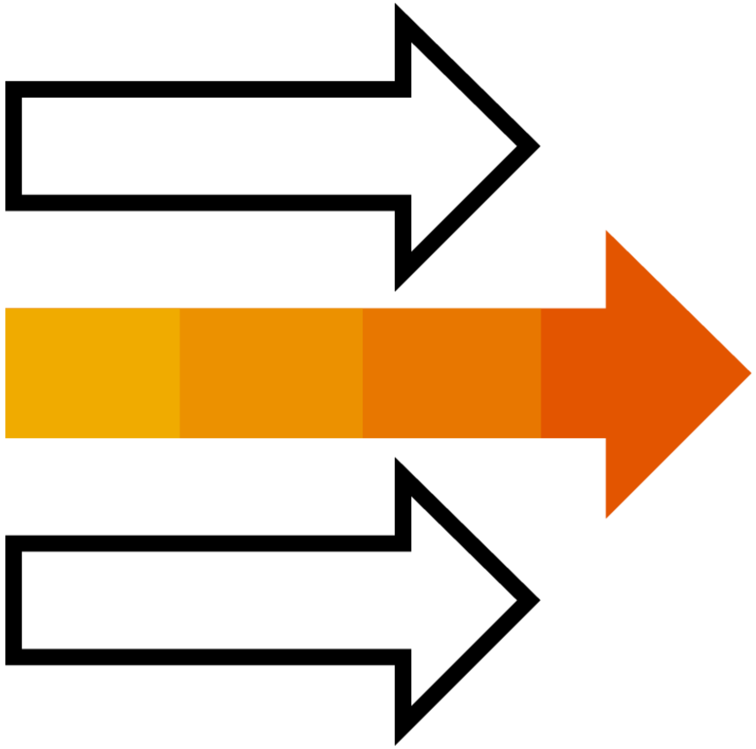
In deeper layers of the network, the features become effectively larger and more abstract

The next lecture will explain CNNs and these advantages in detail

Advantages:
- Fewer parameters to tune
- Spatial invariance built in mathematically
- Retains spatial relationship between features

Introduction to CNNs
Coming up next

CNN Architecture I
- Convolutions
- Non-linearity
- Weight initialization



Thank you.

Contact information:

open@sap.com

Week 4: Convolutional Networks

Unit 2: CNN Architecture Part I

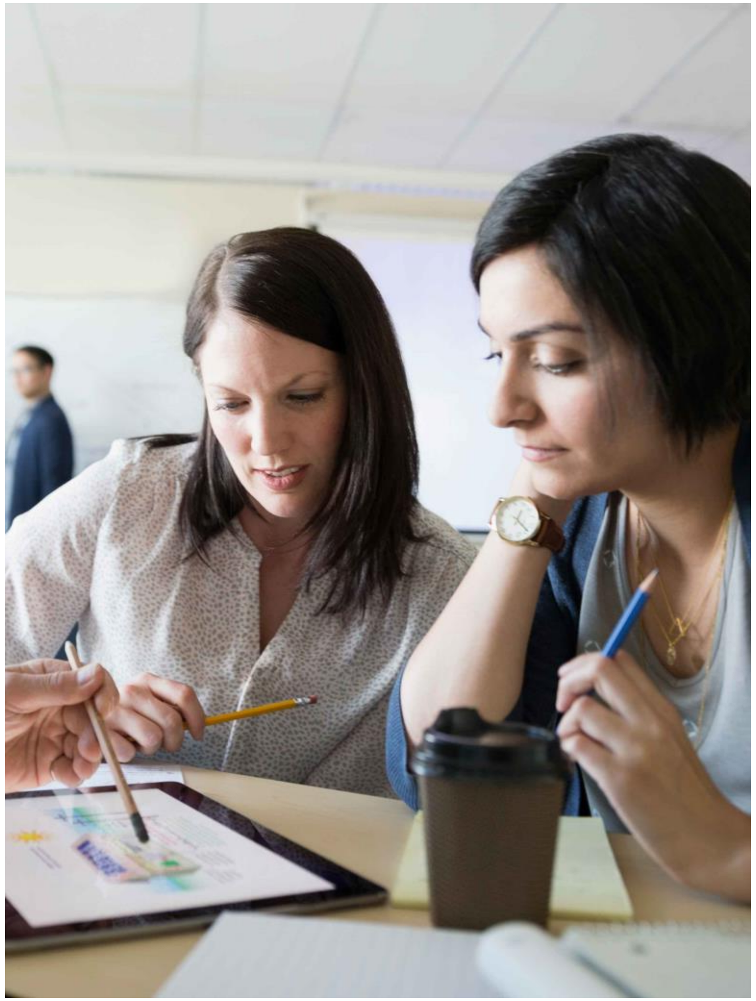
Run Simple

openSAP
open.sap.com

CNN Architecture Part I
What we covered in the last unit

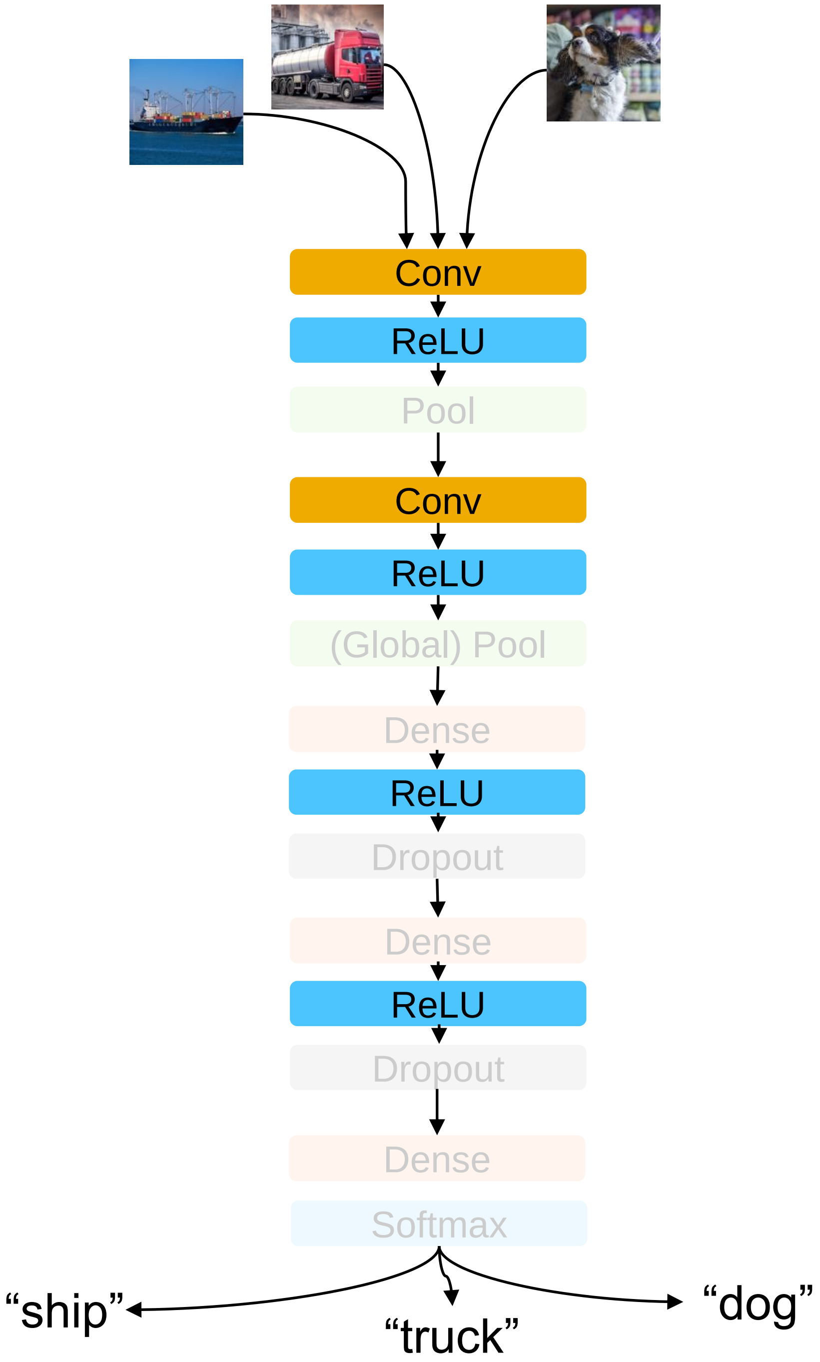
Introduction to Convolutional Networks

- Biological inspiration
- Spatial invariance
- Basic idea: convolutional network



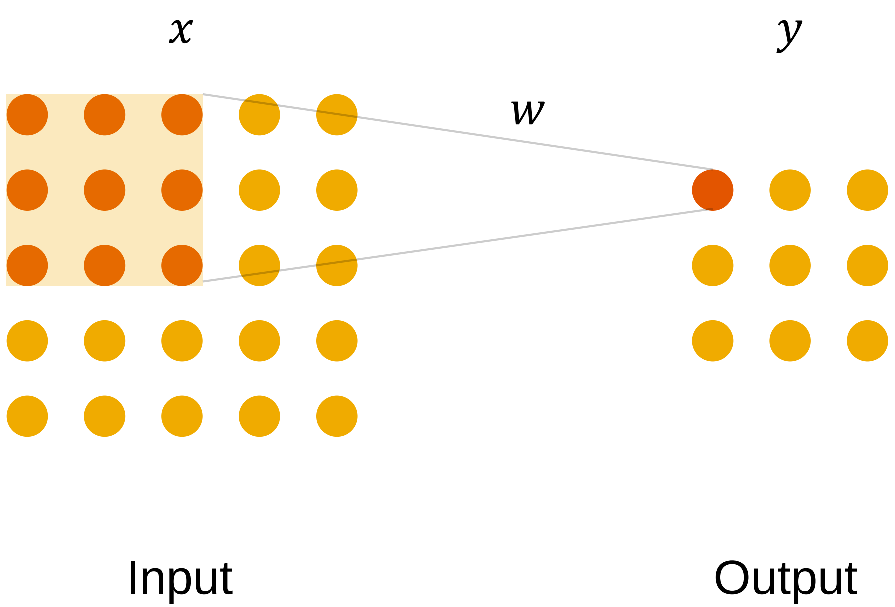
CNN Architecture Part I
Overview

Content:
- Convolutions
- ReLU non-linearity (activation function)
- Weight Initialization
- pooling / global pooling
- dense (fully connected) layers
- dropout
- softmax



CNN Architecture Part I

Convolutions



"valid" convolution

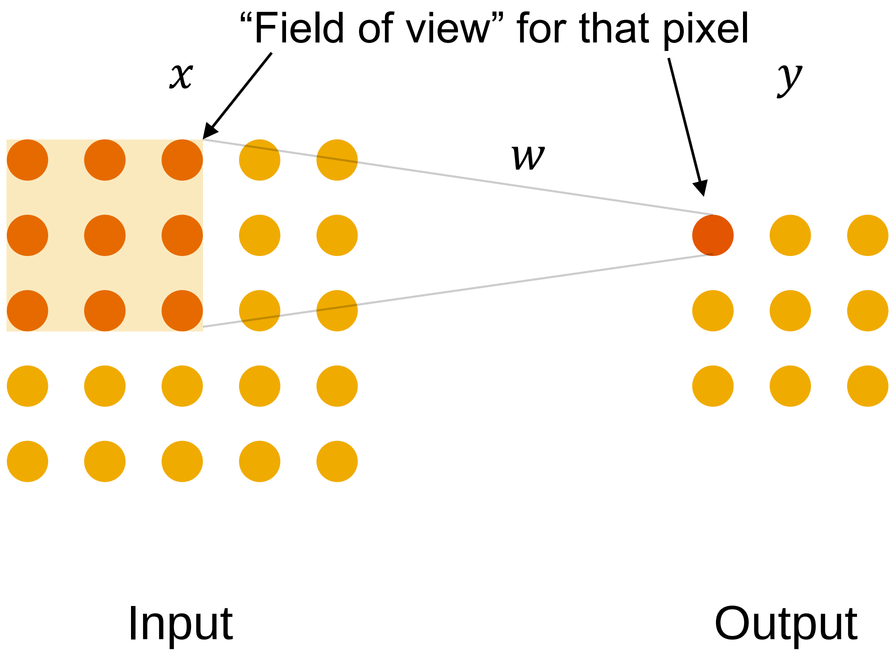
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



“valid” convolution

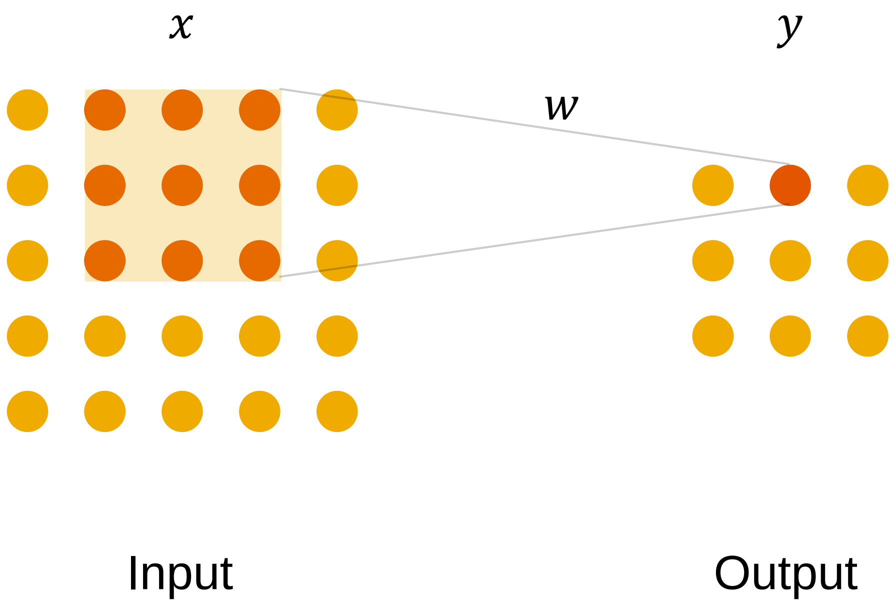
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



"valid" convolution

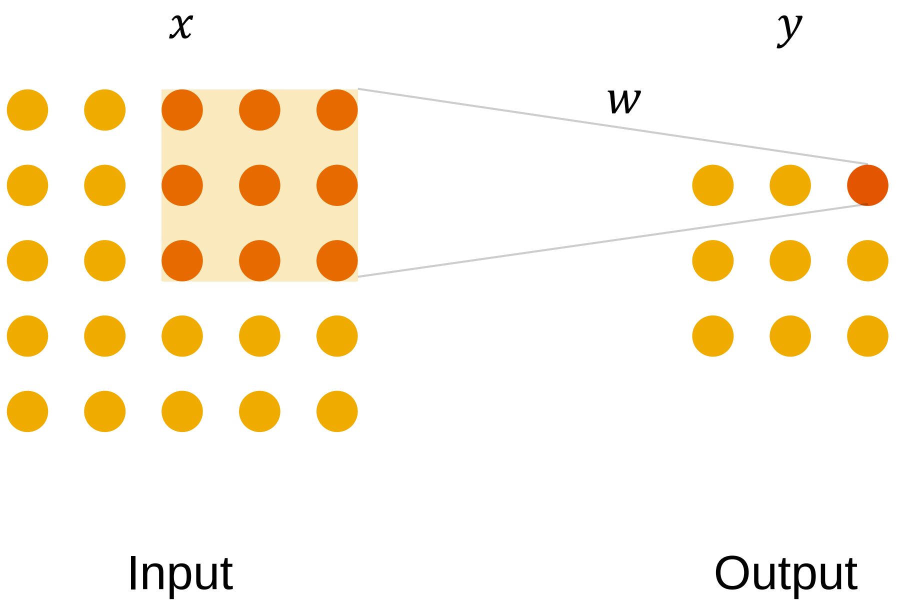
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



"valid" convolution

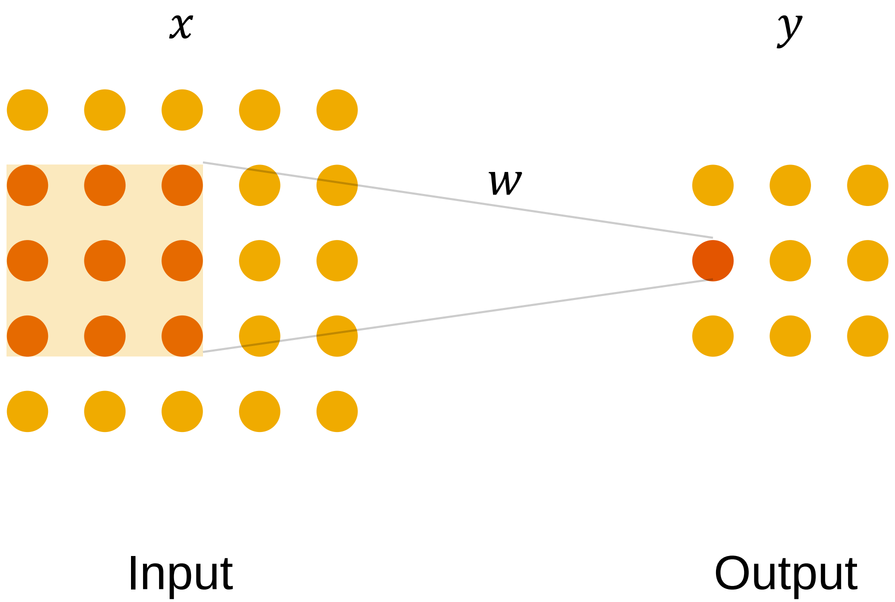
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



"valid" convolution

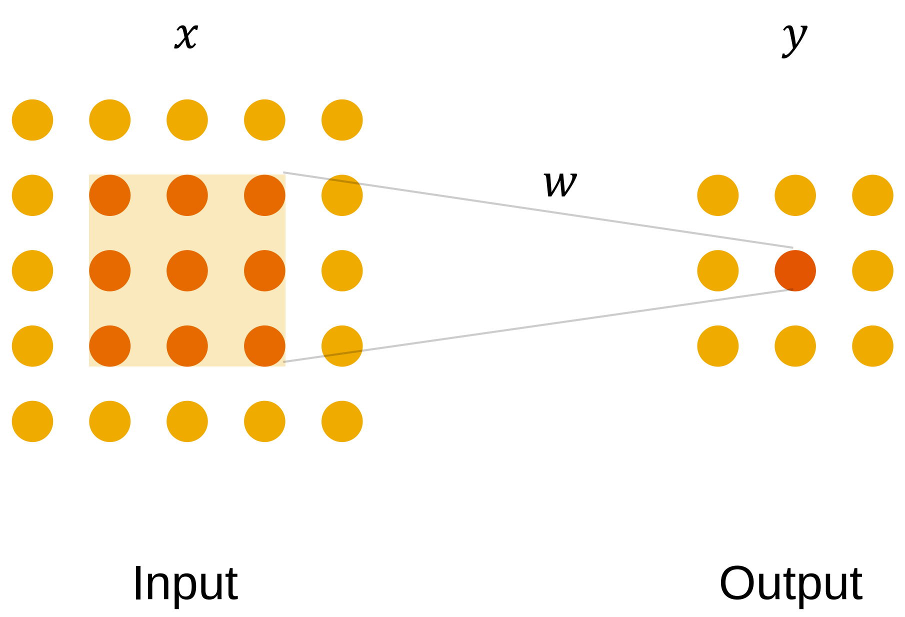
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



"valid" convolution

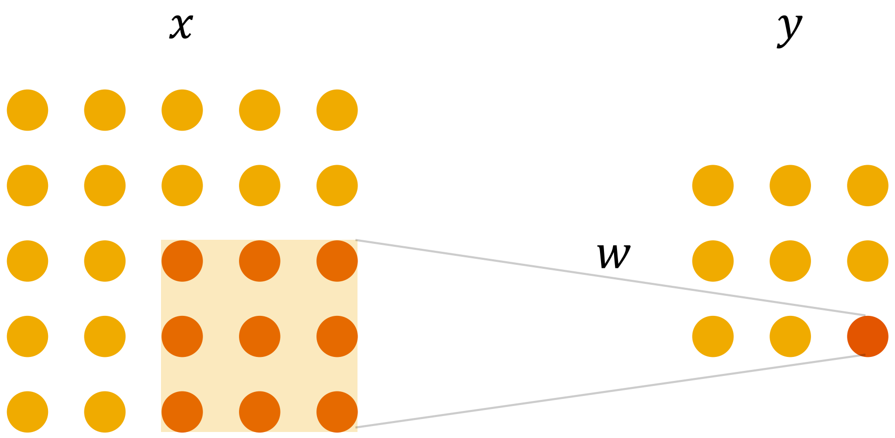
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



Input Output

"valid" convolution

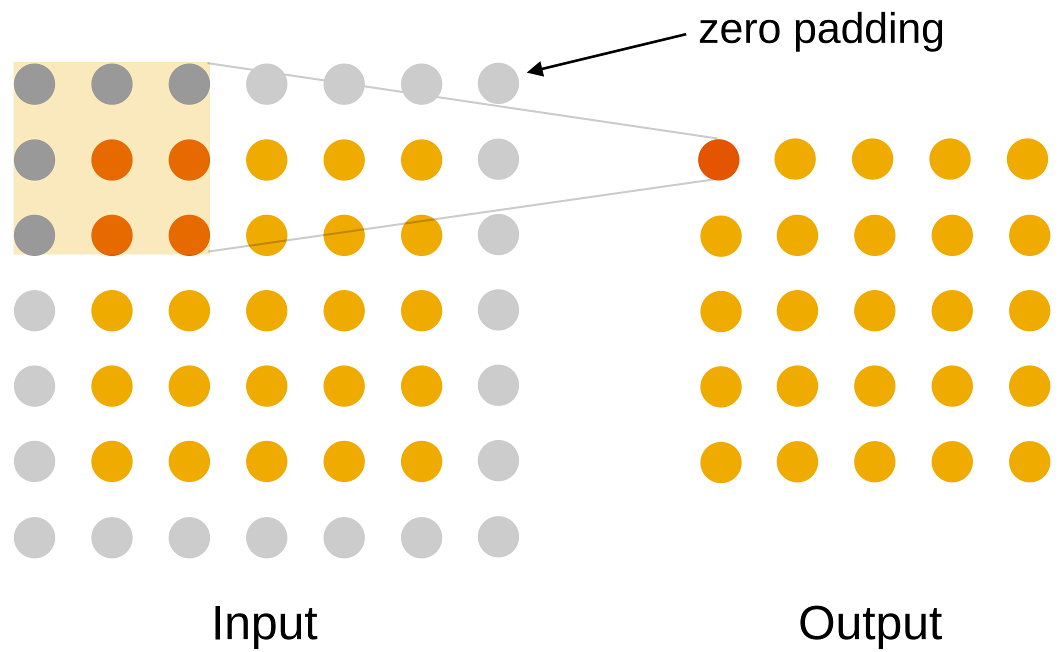
$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions



"same" convolution

$$y_{ij} = \sum_{kl} w_{kl} x_{(k+i)(l+j)}$$

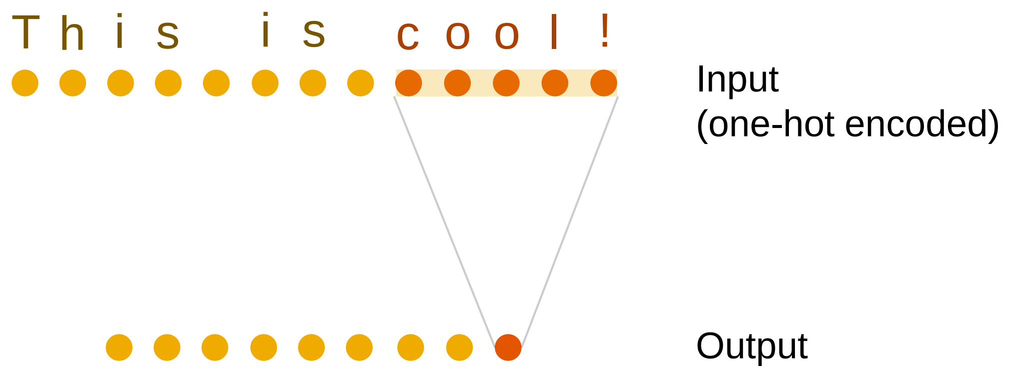
Weights are ***shared*** between pixels (neurons)!

- Imparts spatial invariance to network
- Reduces number of weights

CNN Architecture Part I

Convolutions in 1D

The same ideas generalize to 1D as well!



1D convolution:

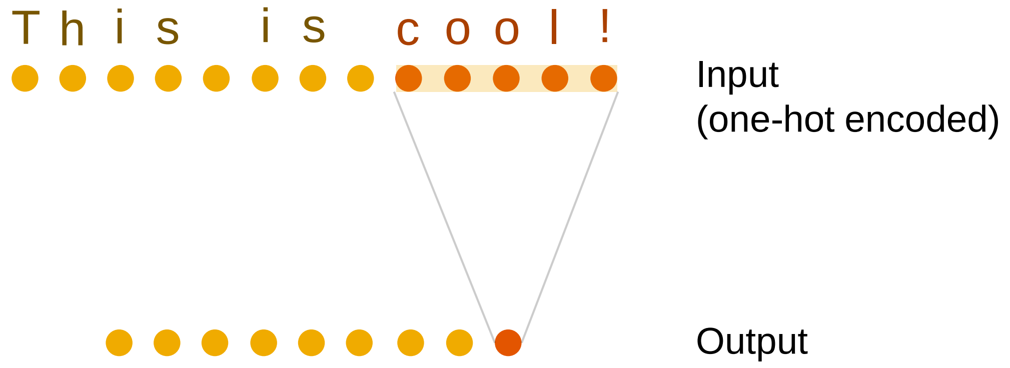
e.g. audio signal

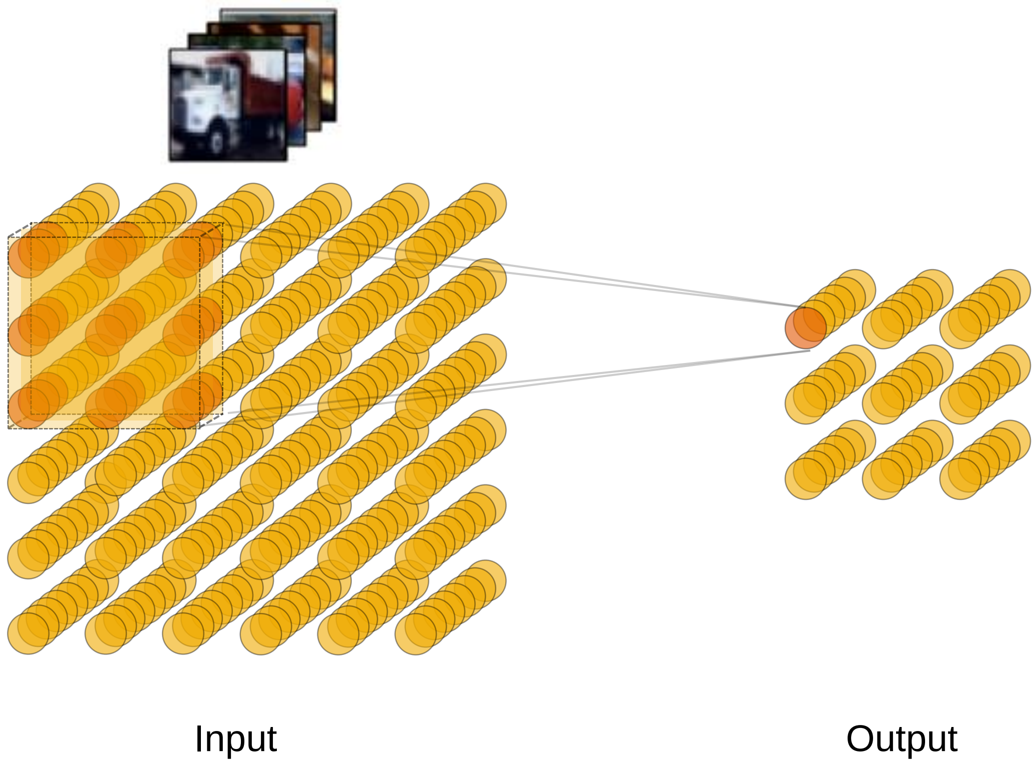
natural language processing

CNN Architecture Part I

Convolutions in 1D & 3D

The same ideas generalize to 1D & 3D as well!





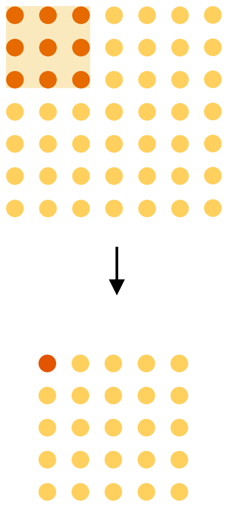
1D convolution:
e.g. audio signal
natural language processing

3D convolution:
e.g. video processing

CNN Architecture Part I

Convolutions: popular kernel shapes

3x3 kernel

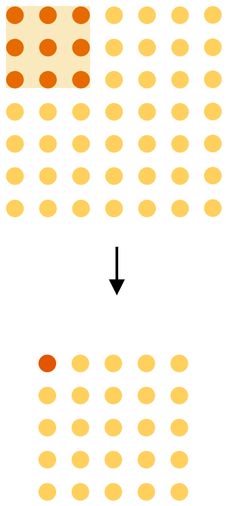


Most widely used in
deep models

CNN Architecture Part I

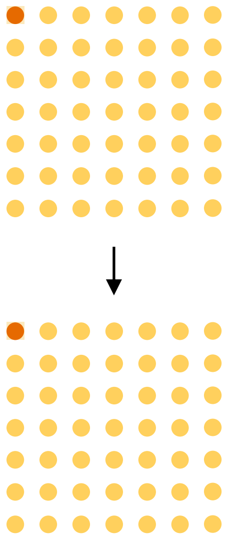
Convolutions: popular kernel shapes

3x3 kernel



Most widely used in deep models

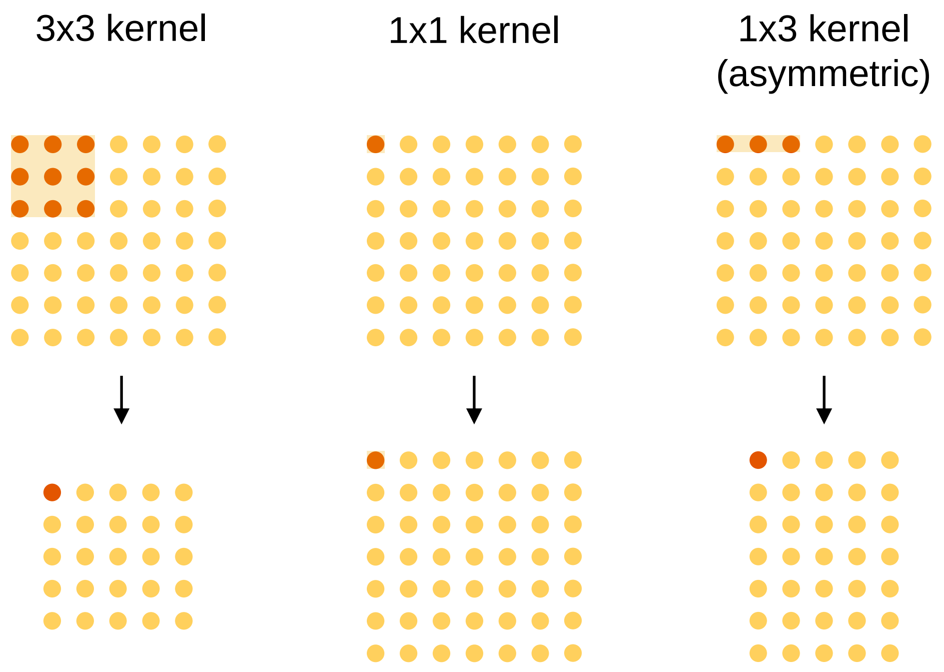
1x1 kernel



Will become clear shortly

CNN Architecture Part I

Convolutions: popular kernel shapes



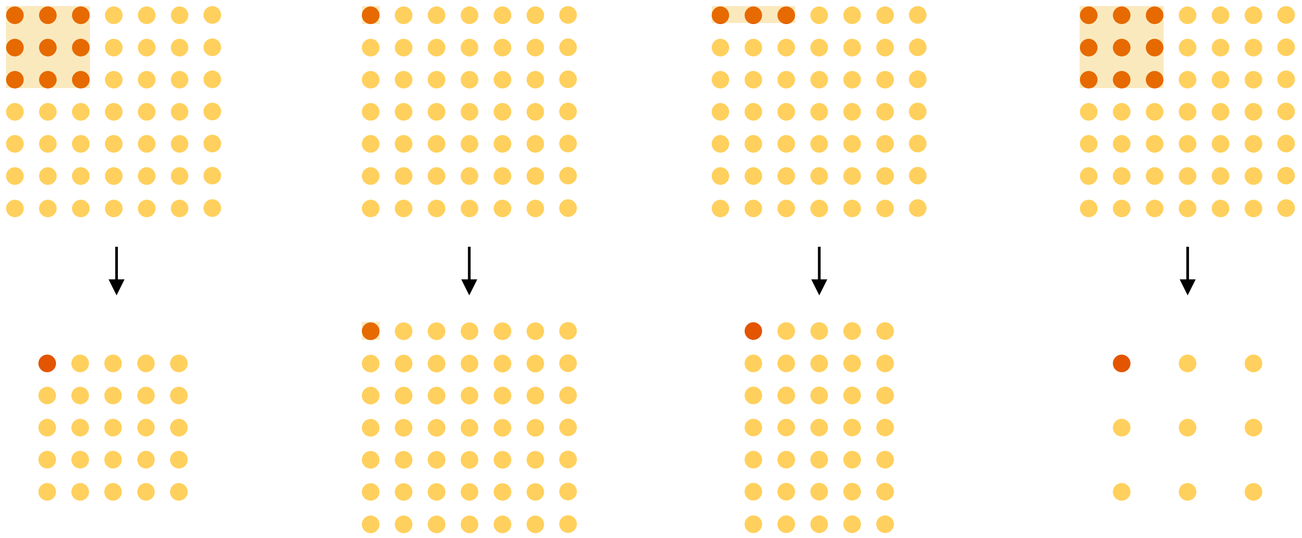
Most widely used in deep models

Will become clear shortly

CNN Architecture Part I

Convolutions: popular kernel shapes

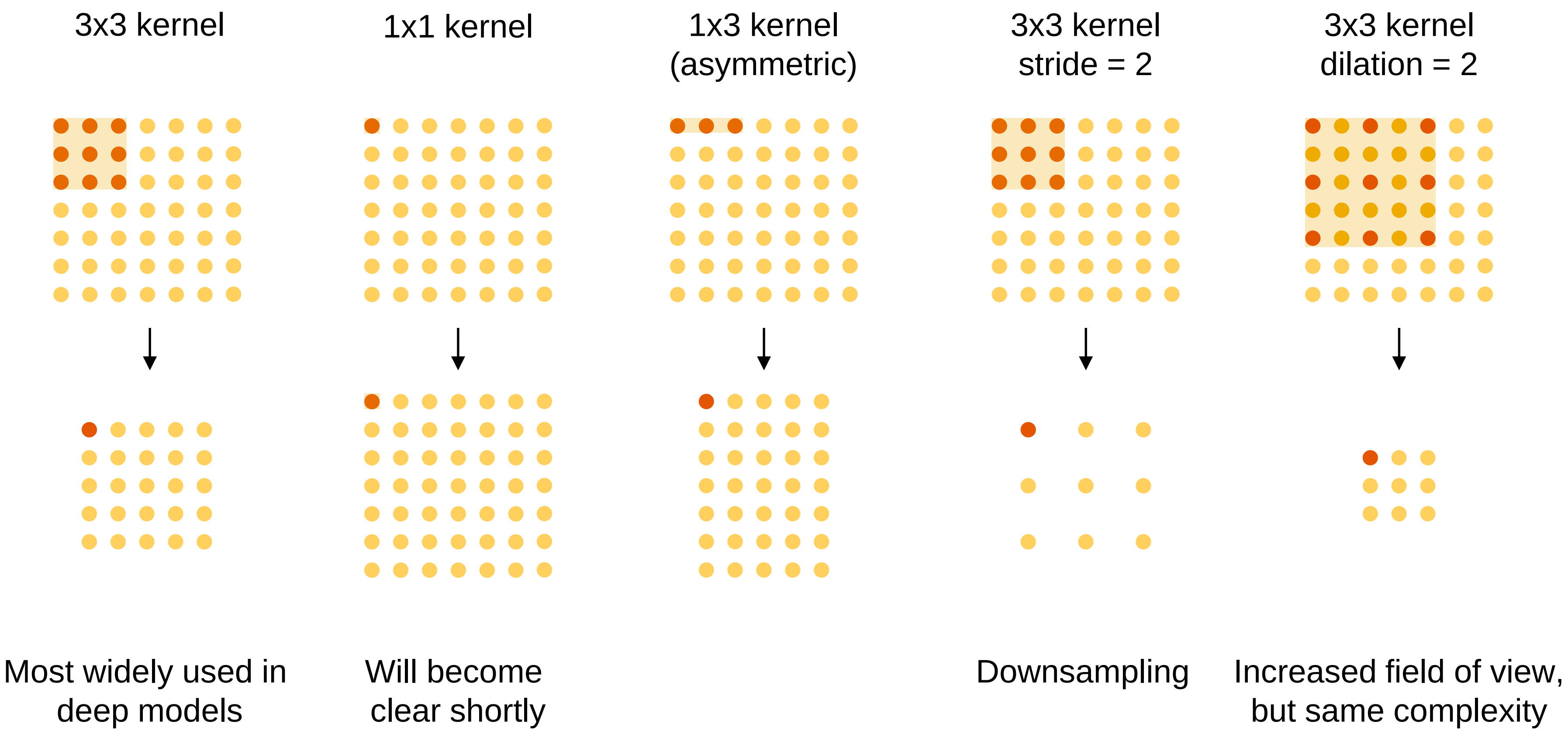
3x3 kernel | 1x1 kernel | 1x3 kernel (asymmetric) | 3x3 kernel stride = 2



Most widely used in deep models | Will become clear shortly | Downsampling

CNN Architecture Part I

Convolutions: popular kernel shapes

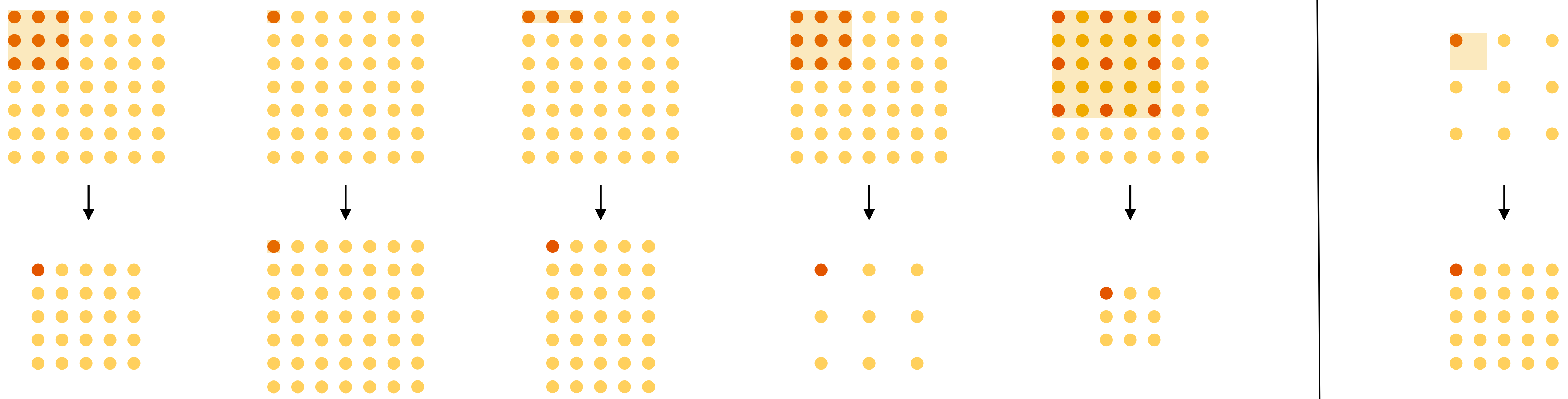


CNN Architecture Part I

Convolutions: popular kernel shapes

Transposed Convolution
aka deconvolution
aka fractionally strided convolution

3x3 kernel | 1x1 kernel | 1x3 kernel (asymmetric) | 3x3 kernel stride = 2 | 3x3 kernel dilation = 2 | 2x2 kernel stride = 2



Most widely used in deep models | Will become clear shortly | Downsampling | Increased field of view, but same complexity | Upsampling

CNN Architecture Part I

Convolutions: popular kernel shapes

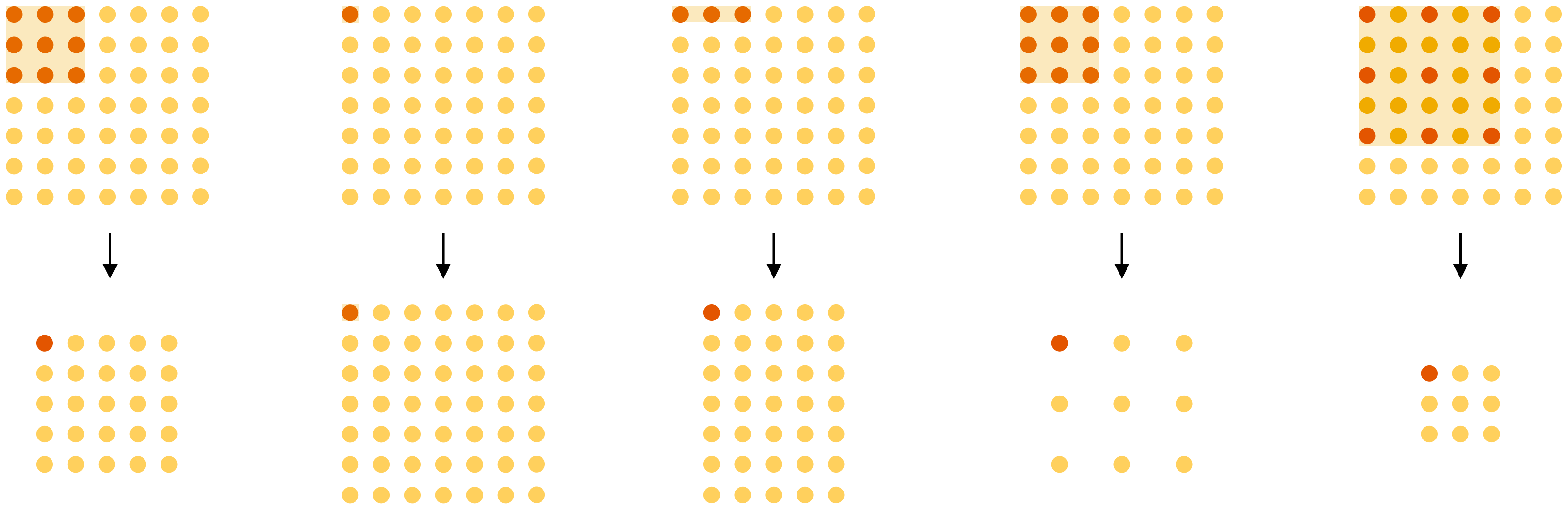
3x3 kernel

1x1 kernel

1x3 kernel (asymmetric)

3x3 kernel stride = 2

3x3 kernel dilation = 2



Most widely used in deep models

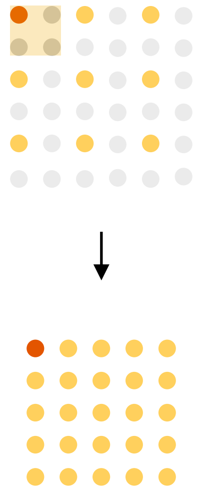
Will become clear shortly

Downsampling

Increased field of view, but same complexity

Transposed Convolution
aka deconvolution
aka fractionally strided convolution

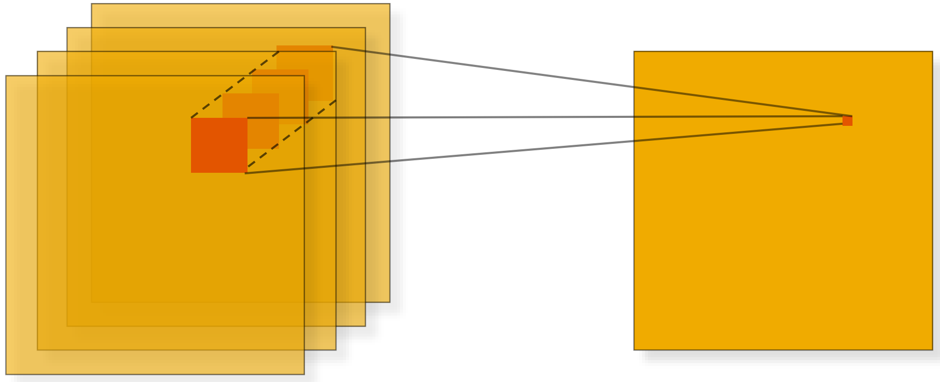
2x2 kernel stride = 2



Upsampling

CNN Architecture Part I

Convolutions



Input

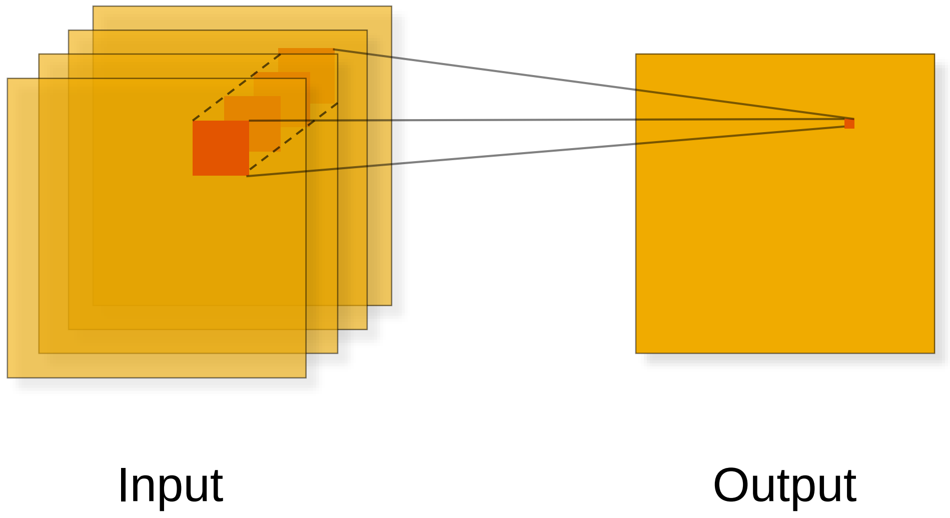
Output

$$y_{ij} = \sum_{C} \sum_{kl} w_{Ckl} x_{C(k+i)(l+j)}$$

We are not just summing over pixels, but also across input channels!

CNN Architecture Part I

Convolutions



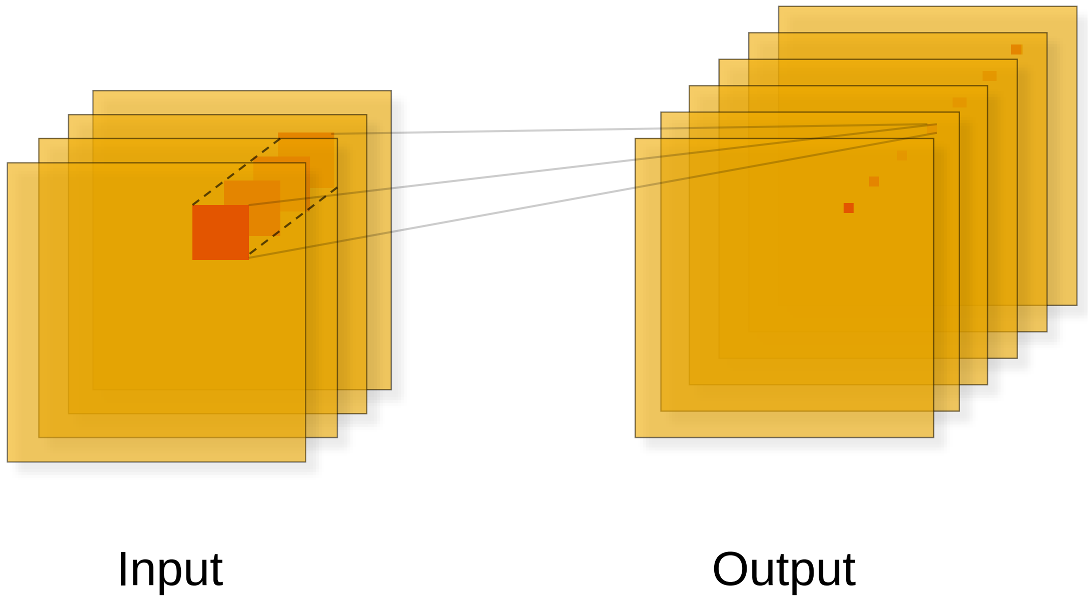
$$y_{ij} = \sum_{C} \sum_{kl} w_{Ckl} x_{C(k+i)(l+j)}$$

One “shared” set of weights per input channel

We are not just summing over pixels, but also across input channels!

CNN Architecture Part I

Convolutions



Multiple output channels

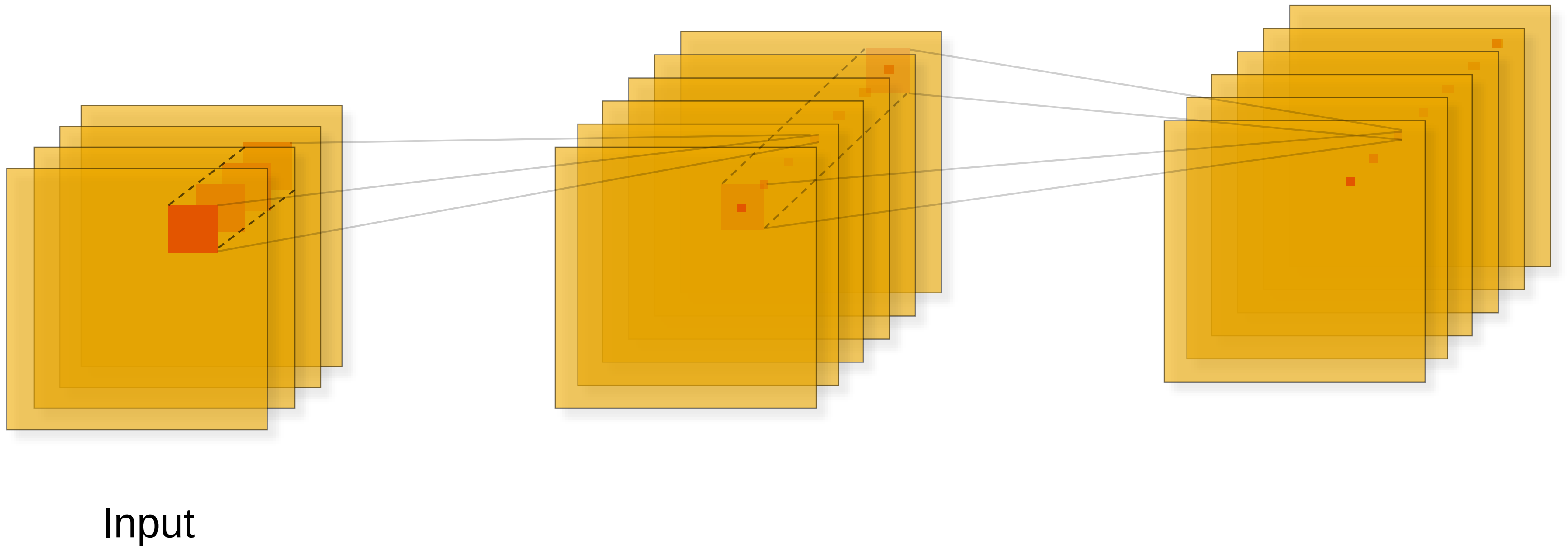
$$y^{I}_{ij} = \sum_{C} \sum_{kl} w^{I}_{Ckl} x_{C(k+i)(l+j)}$$

One ”shared” set of weights per input channel *and* per output channel

We can have multiple output channels (feature maps) as well!

CNN Architecture Part I

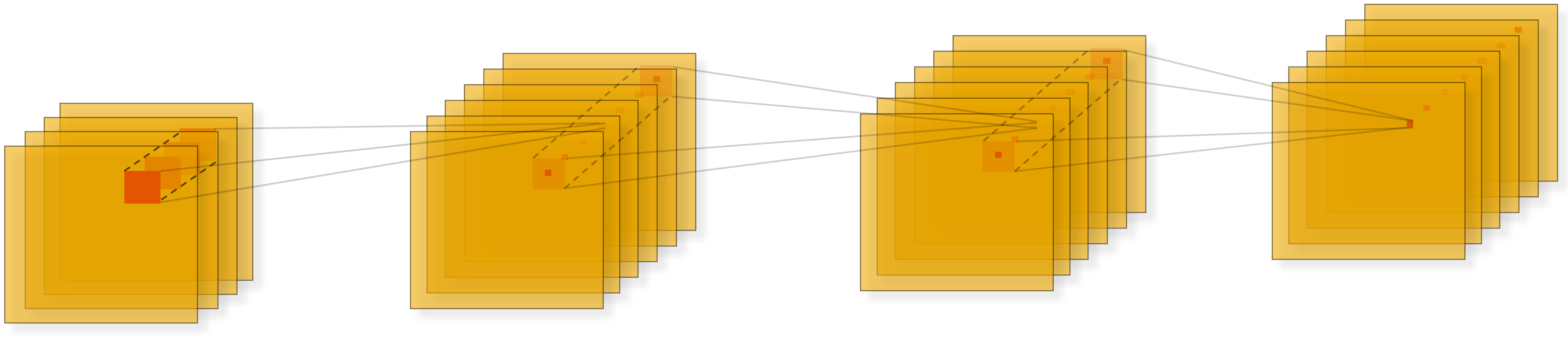
Convolutions: going deeper



We can chain multiple convolutions to build a deep convolutional neural network!

CNN Architecture Part I

Convolutions: going deeper

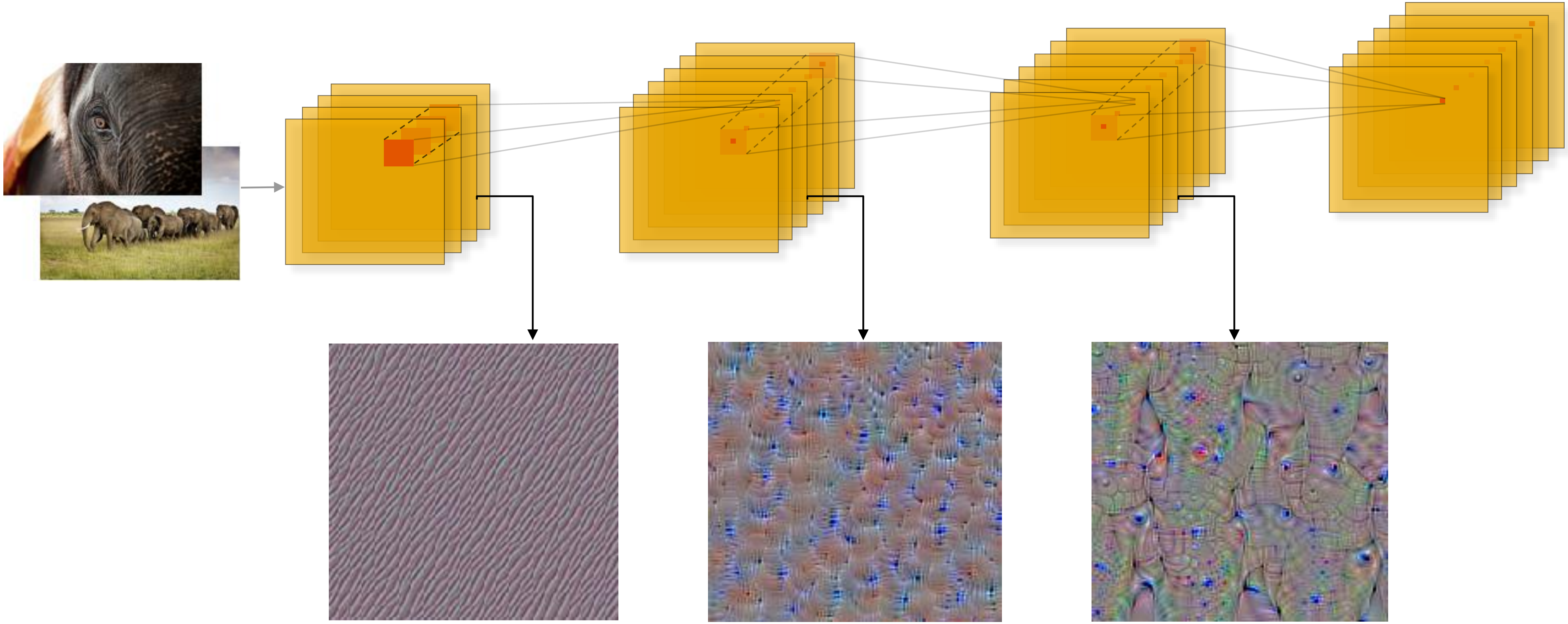


Input

We can chain multiple convolutions to build a deep convolutional neural network!

CNN Architecture Part I

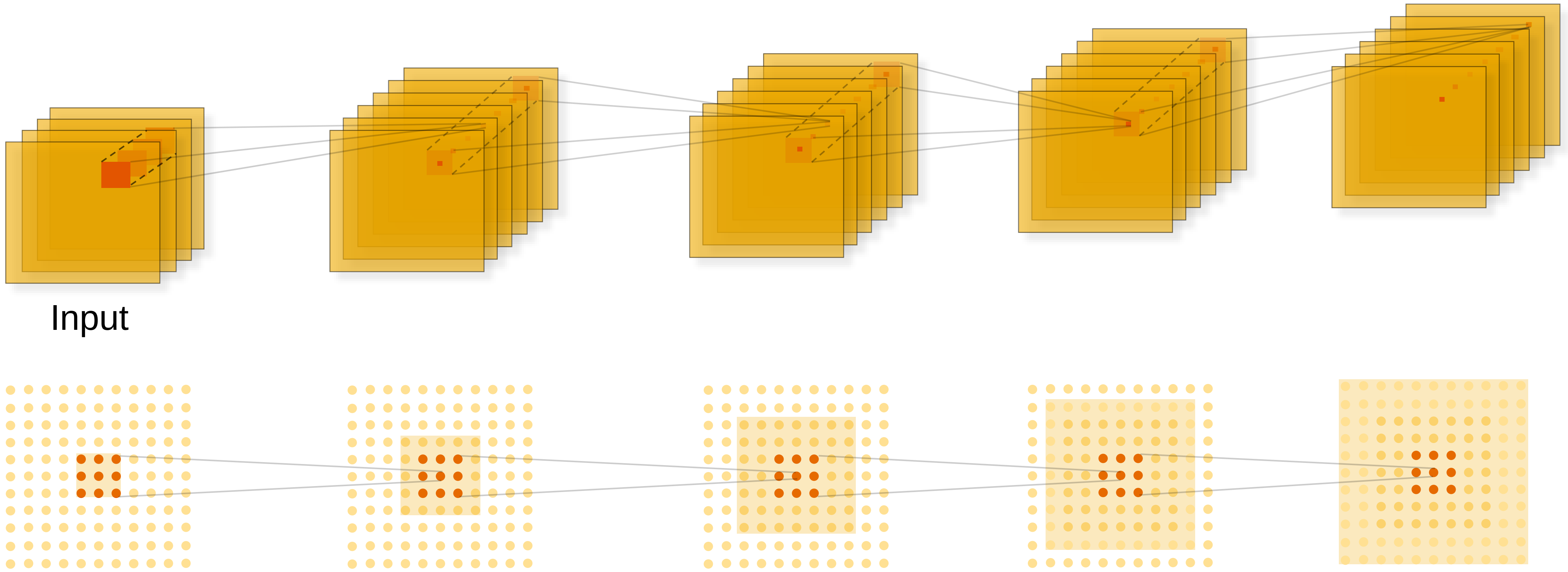
Convolutions: going deeper



Low level (local) features → **High level** (global) features

CNN Architecture Part I

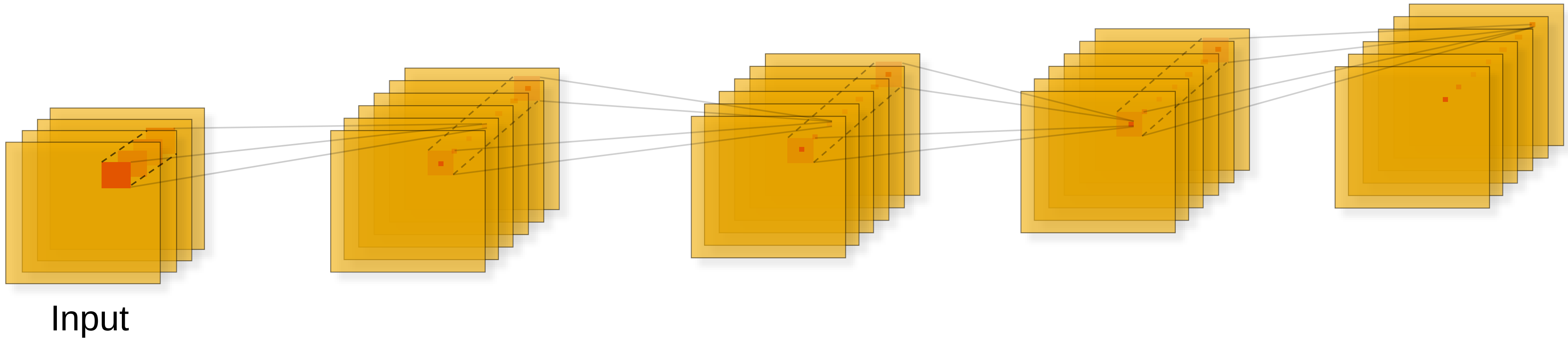
Convolutions: going deeper



Note: After each layer, the effective field of view *within the input image* is increased!

CNN Architecture Part I

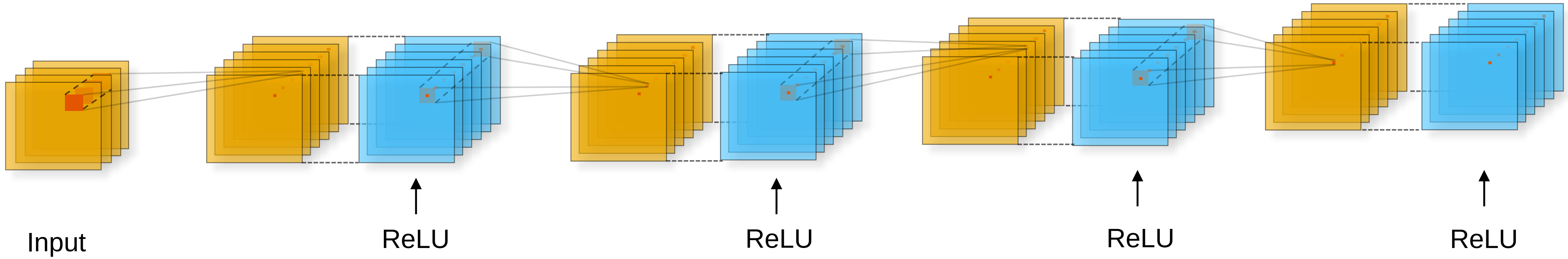
Non-linearity: ReLUs



After each convolution, we want to have a **non-linearity** to increase expressive power!

CNN Architecture Part I

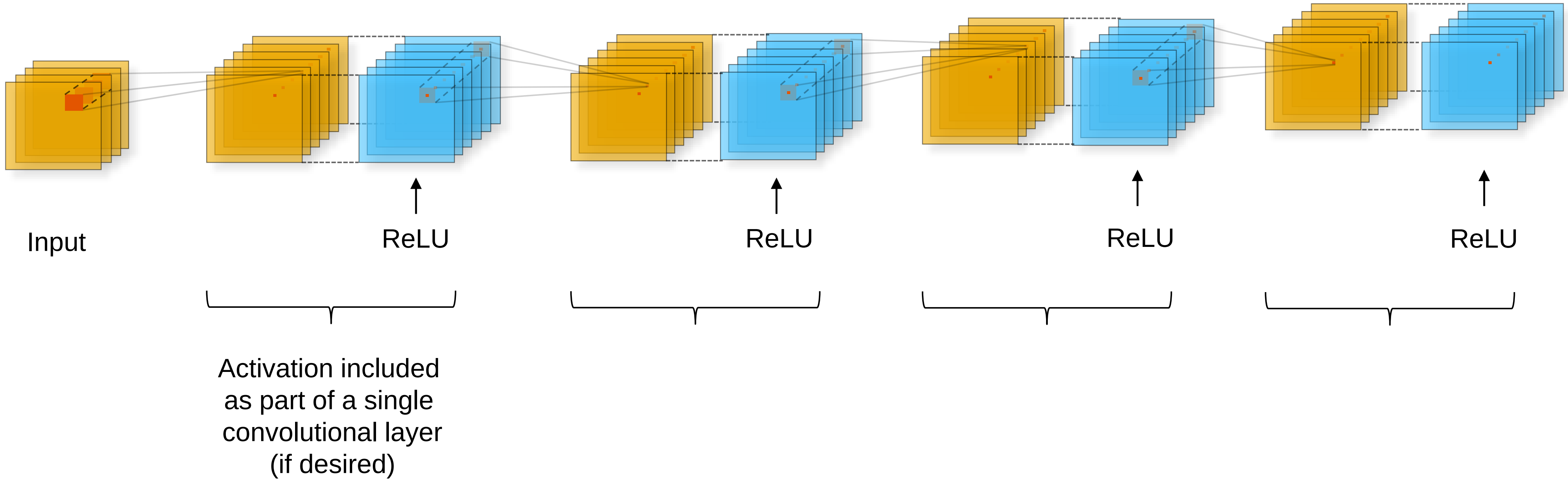
Non-linearity: ReLUs



After each convolution, we want to have a **non-linearity** to increase expressive power!

CNN Architecture Part I

Non-linearity: ReLUs



After each convolution, we want to have a **non-linearity** to increase expressive power!

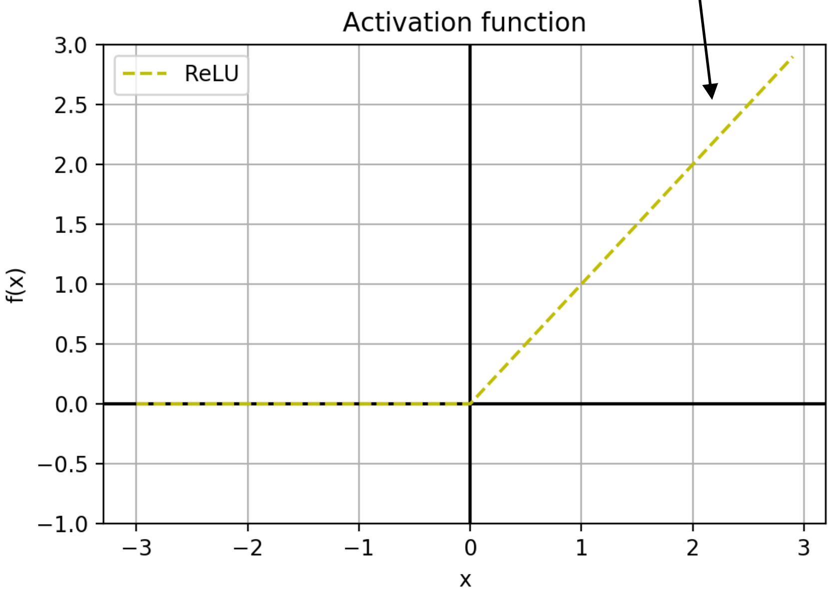
CNN Architecture Part I

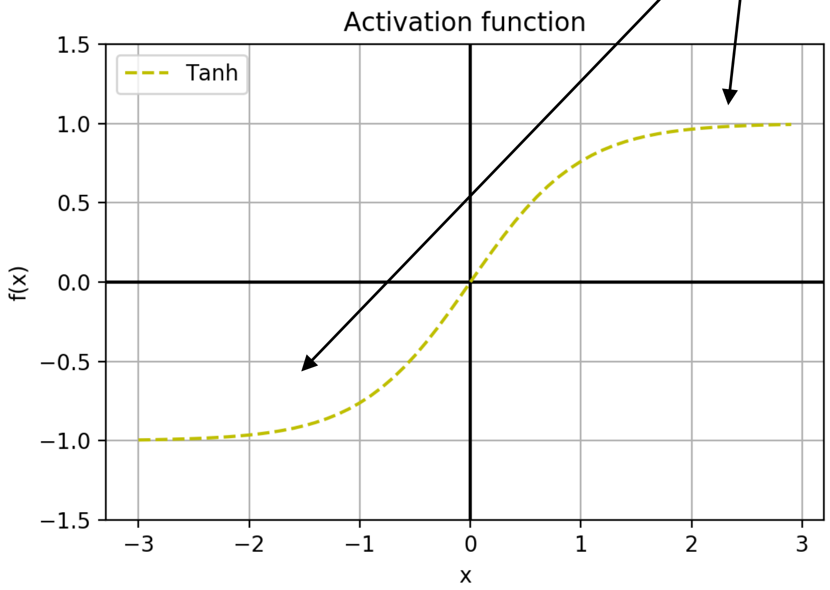
Non-linearity: rectified linear unit (ReLU)

No saturation

Saturation (derivative ~ 0)

ReLU





Classical activation functions

- ReLUs do not **saturate**
  (reduced vanishing gradient problem; see RNNs, Week 3)
- Induce sparsity

Deep Sparse Rectifier Neural Networks
Xavier Glorot, Antoine Bordes, Yoshua Bengio;

Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR 15:315-323, 2011

CNN Architecture Part I

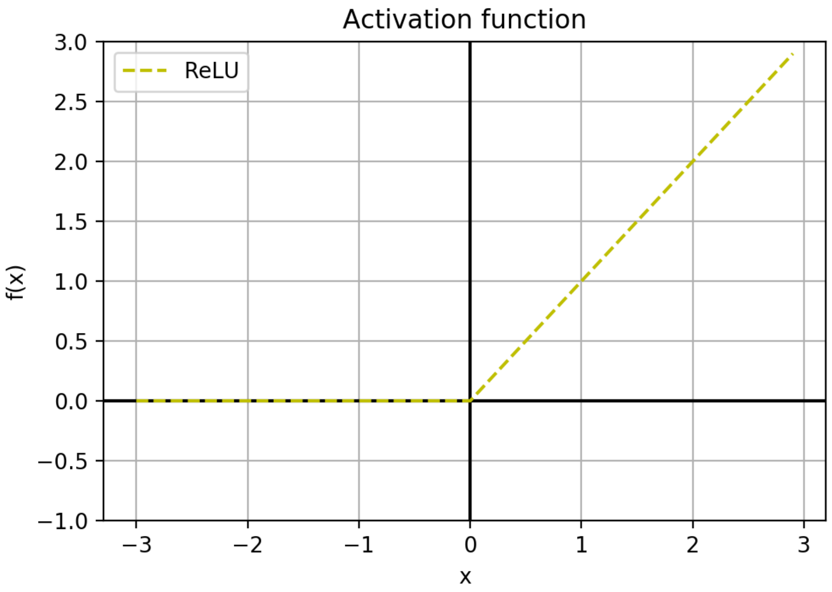
Non-linearity: parametric rectified linear unit (PReLU)

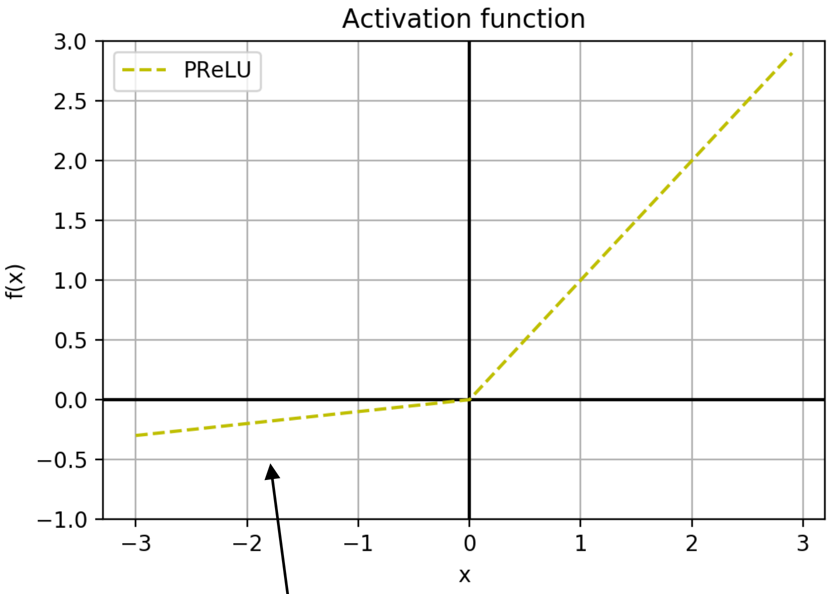
Delving Deep into Rectifiers:
Surpassing Human-Level Performance on ImageNet Classification

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

ProceedingICCV '15 Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) Pages 1026-1034

ReLU





PReLU

- ReLUs do not **<u>saturate</u>**
- Induce sparsity
- But they may “die” during training (never ever activate again)

Parametric (leaky) ReLU (PReLU) does not “die”

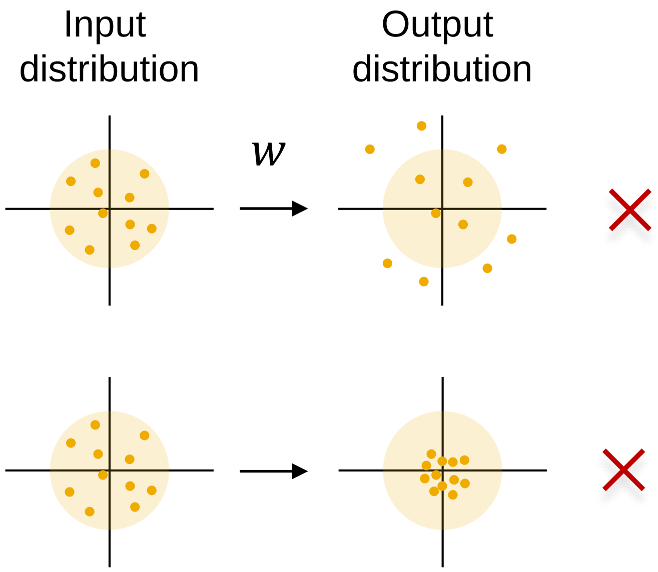
CNN Architecture Part I

Weight initialization

- Weights in each layer are **randomly initialized**

CNN Architecture Part I
Weight initialization

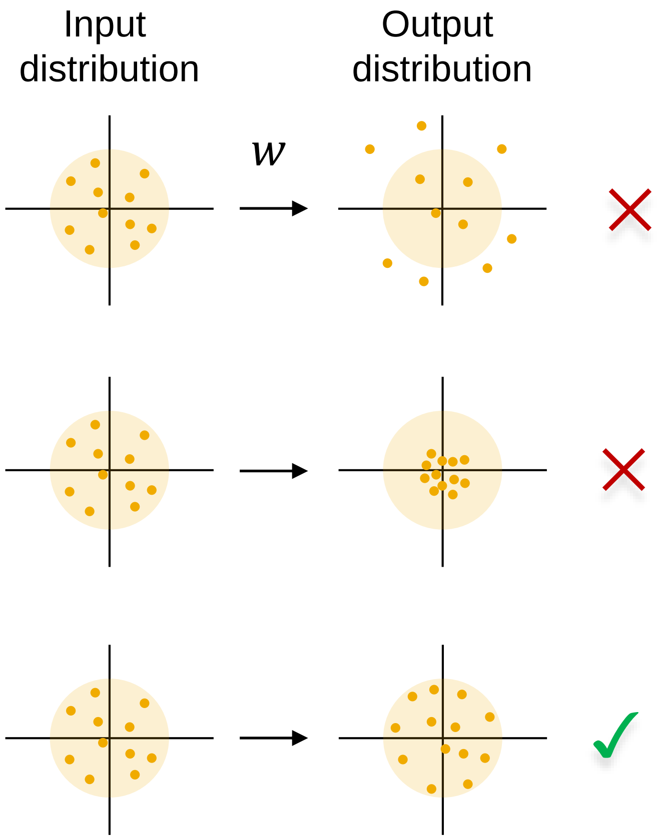
- Weights in each layer are **randomly initialized**
- But they should be initialized such that an input signal is ***neither amplified nor damped***!



CNN Architecture Part I

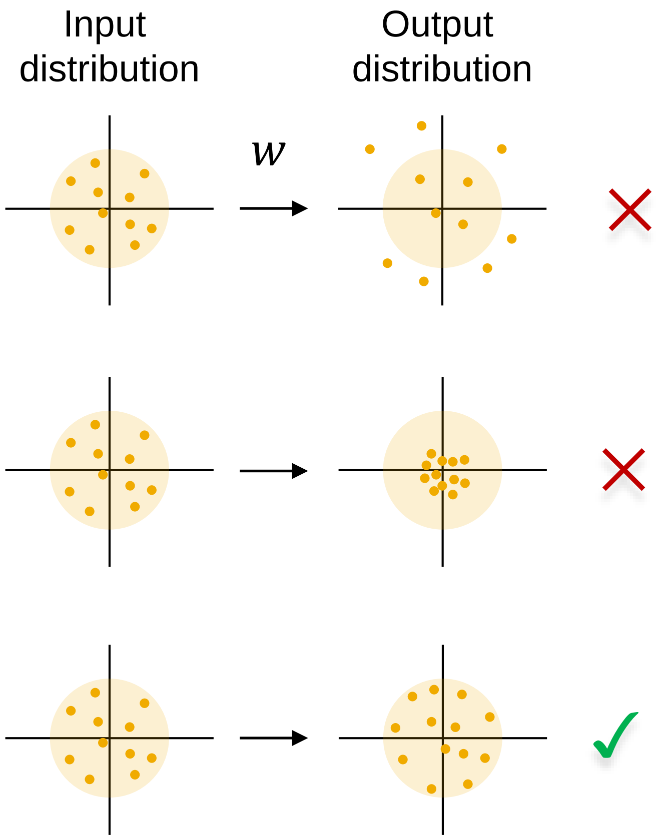
Weight initialization

- Weights in each layer are **randomly initialized**
- But they should be initialized such that an input signal is ***neither amplified nor damped***!
- This property **depends on the number of input/output connections**, and **the non-linearity**



CNN Architecture Part I

Weight initialization



- Weights in each layer are **randomly initialized**
- But they should be initialized such that an input signal is ***neither amplified nor damped***!
- This property **depends on the number of input/output connections**, and **the non-linearity**

For ReLU, use **“He’s” method** (*aka* Gaussian / variance scaling initialization)

$$w \sim P_{\text{Gaussian}}, \quad Var = \sqrt{\frac{2}{N}},$$

N: number of input or output channels, or average of the two

Delving Deep into Rectifiers:
Surpassing Human-Level Performance on ImageNet Classification

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

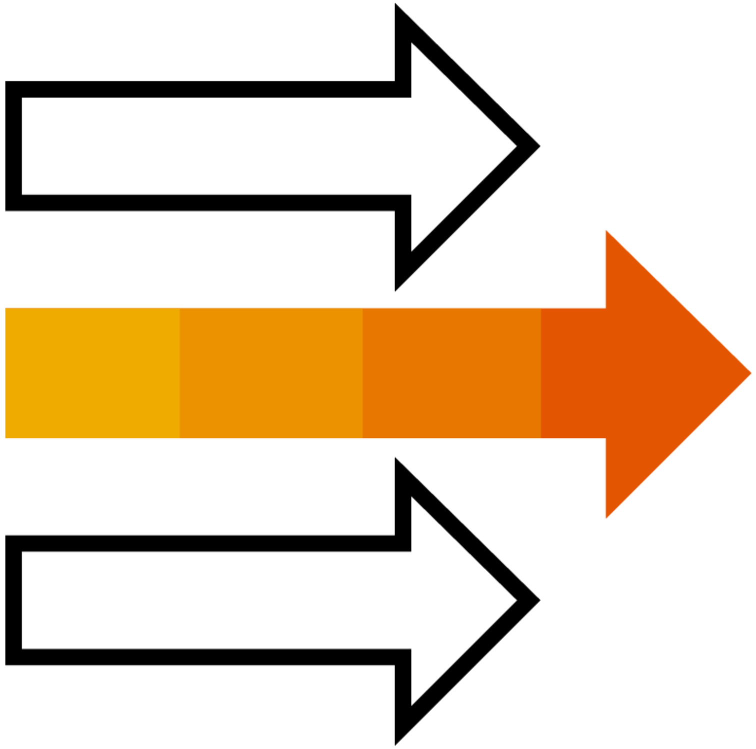
ProceedingICCV '15 Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) Pages 1026-1034

CNN Architecture Part I

Coming up next

CNN Architecture II

- Pooling
- Dense layers
- Dropout
- Softmax



Thank you.

Contact information:

open@sap.com

Week 4: Convolutional Networks

Unit 3: CNN Architecture Part II

Run Simple

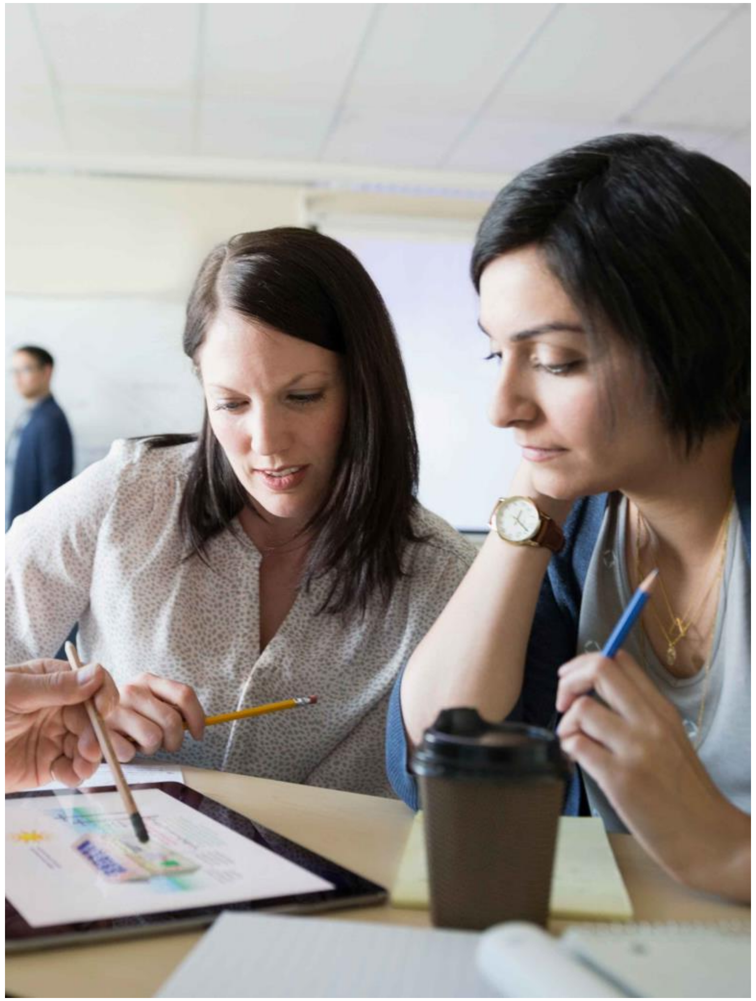
openSAP
open.sap.com

CNN Architecture Part II

What we covered in the last unit

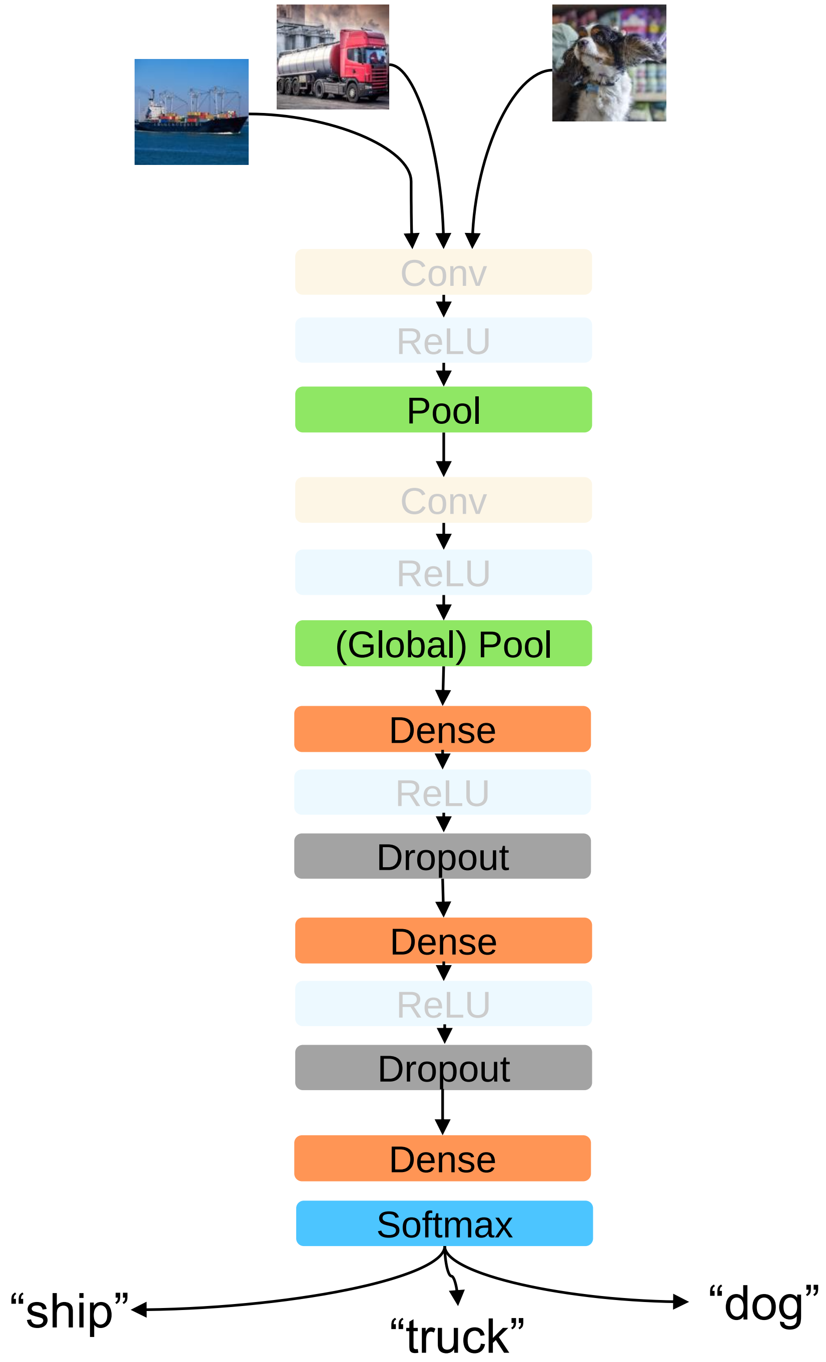
CNN Architecture Part I

- Convolutions
- Non-linearity
- Weight initialization



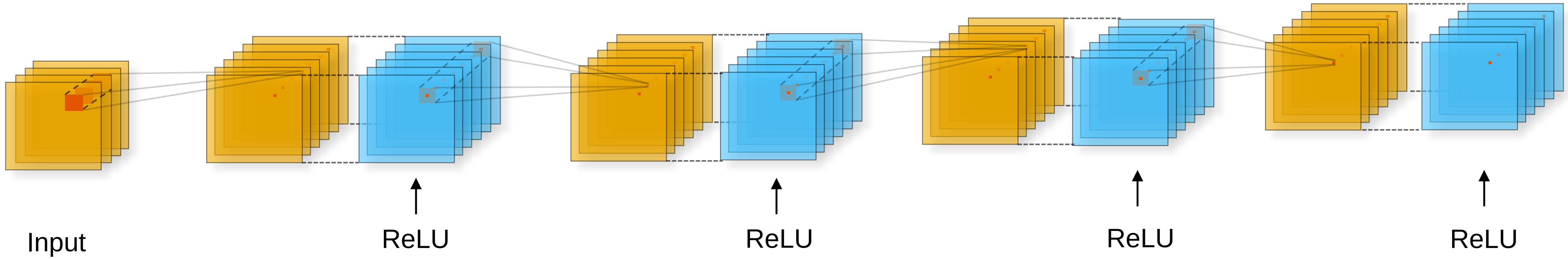
CNN Architecture Part II
Overview

Content:
- Convolutions
- ReLU non-linearity (activation function)
- Weight initialization
- Pooling / global pooling
- Dense (fully connected) layers
- Dropout
- Softmax



CNN Architecture Part II

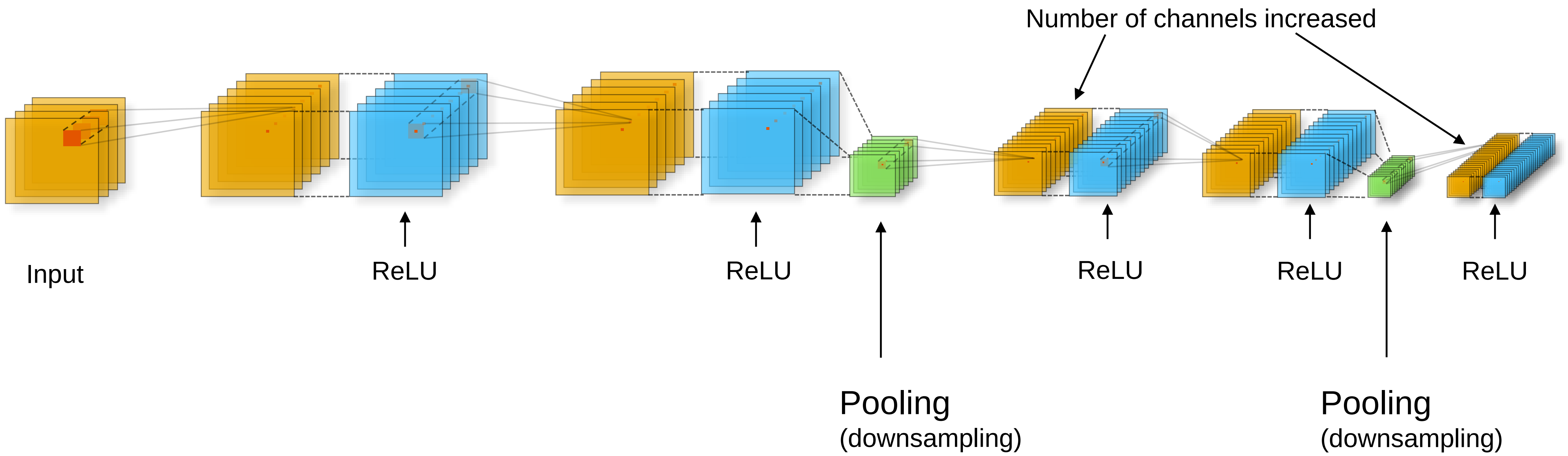
Reducing resolution: pooling



After each convolution (block), we may want to decrease resolution but increase the number of channels

CNN Architecture Part II

Reducing resolution: pooling



After each convolution (block), we may want to **decrease resolution** but ***increase* the number of channels**:

- Increase spatial invariance
- Increase level of abstraction
- Decrease computational load

CNN Architecture Part II

Reducing resolution: pooling

0	1	2	1
1	0	0	1
2	3	1	0
1	2	5	2

$\xrightarrow[\text{Pooling}]{\text{Average}}$

Example: 2x2 pooling, 2x2 stride

0.5	1
2	2

CNN Architecture Part II

Reducing resolution: pooling

Example: 2x2 pooling, 2x2 stride

0	1	2	1
1	0	0	1
2	3	1	0
1	2	5	2

Average Pooling →

0.5	1
2	2

0	1	2	1
1	0	0	1
2	3	1	0
1	2	5	2

Max Pooling →

1	2
3	5

Better at capturing edges

CNN Architecture Part II

Reducing resolution: global pooling

0	1	2	1
1	0	0	1
2	3	1	0
1	2	5	2

Global Average Pooling →

1.375

Take the average over an entire channel

CNN Architecture Part II

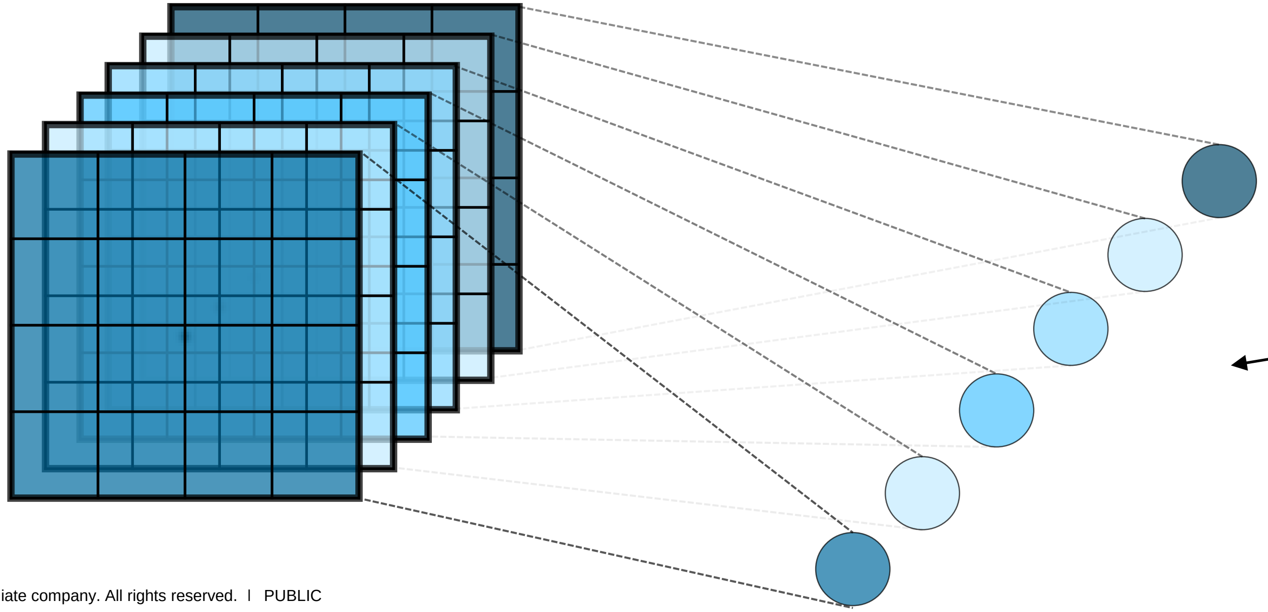
Reducing resolution: global pooling

0	1	2	1
1	0	0	1
2	3	1	0
1	2	5	2

$\xrightarrow[\text{Pooling}]{\text{Global Average}}$

1.375

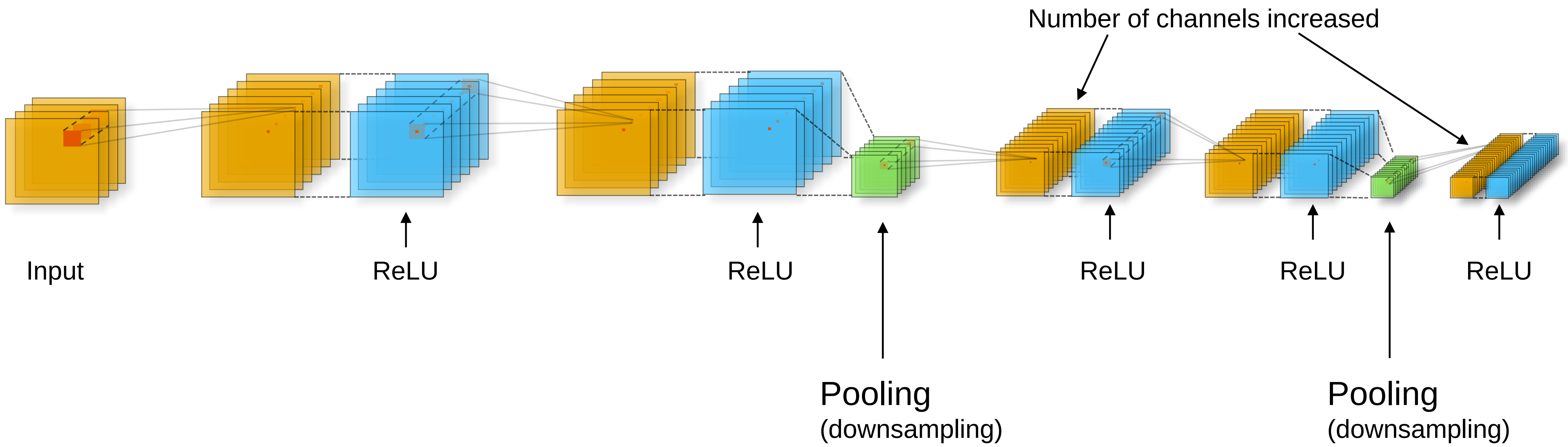
Take the average over an entire channel



Outcome of global pooling can be used as input for a dense layer to reduce complexity

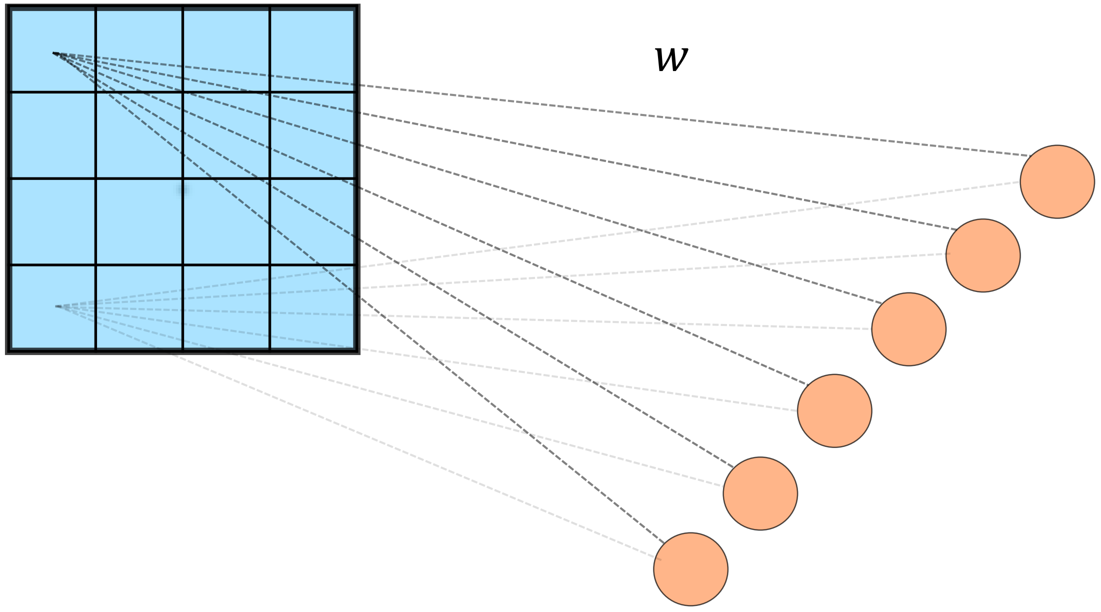
CNN Architecture Part II

From convolutions to dense layers



For classification tasks, we want to add dense layers

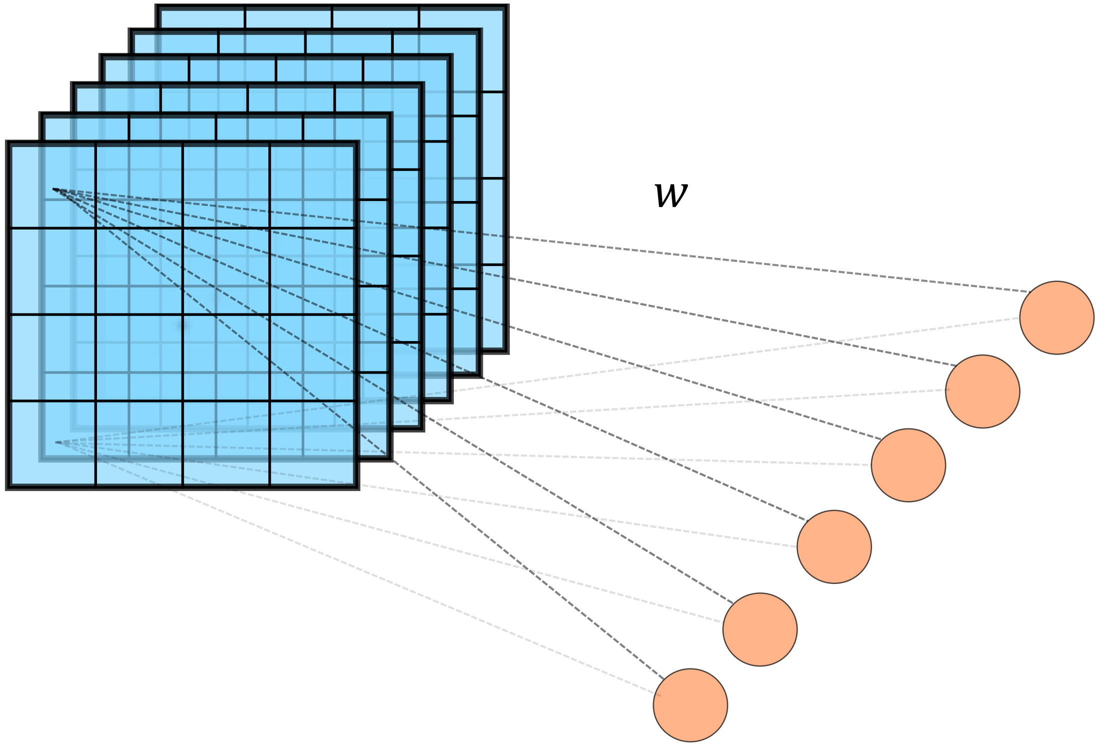
CNN Architecture Part II
From convolutions to dense layers



Every pixel in all output channels is connected to every neuron in the dense layer

CNN Architecture Part II

From convolutions to dense layers

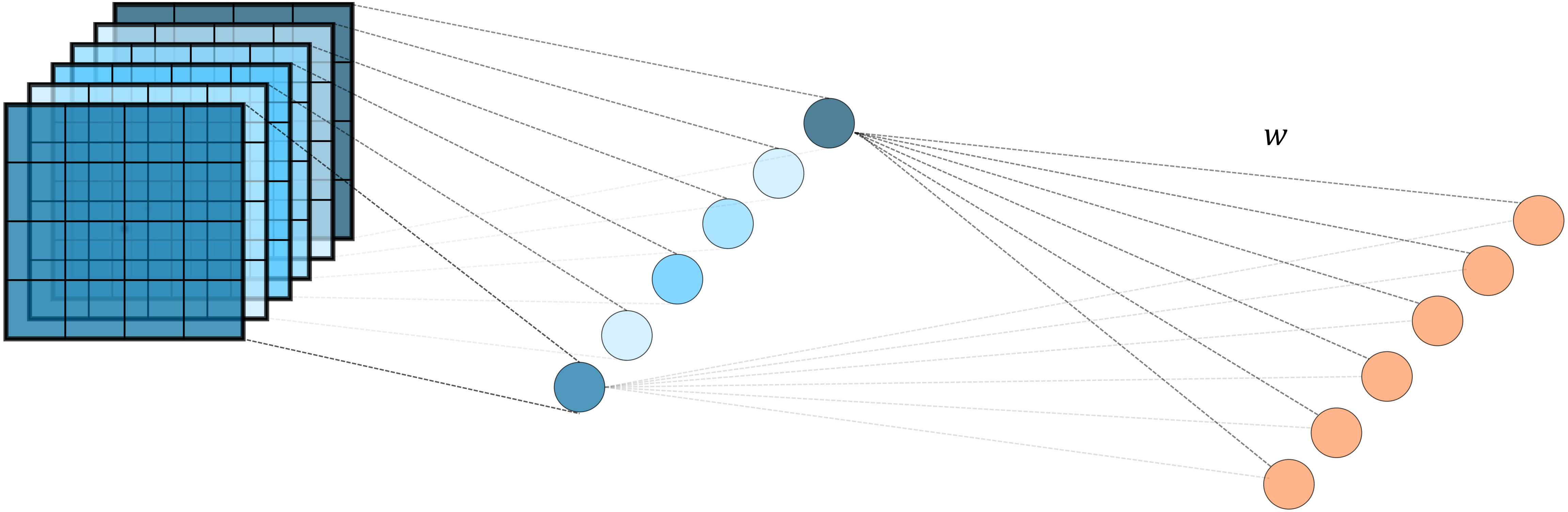


Every pixel *in all output channels* is connected to every neuron in the dense layer

→ weight matrix may become very large!

CNN Architecture Part II

From convolutions to dense layers



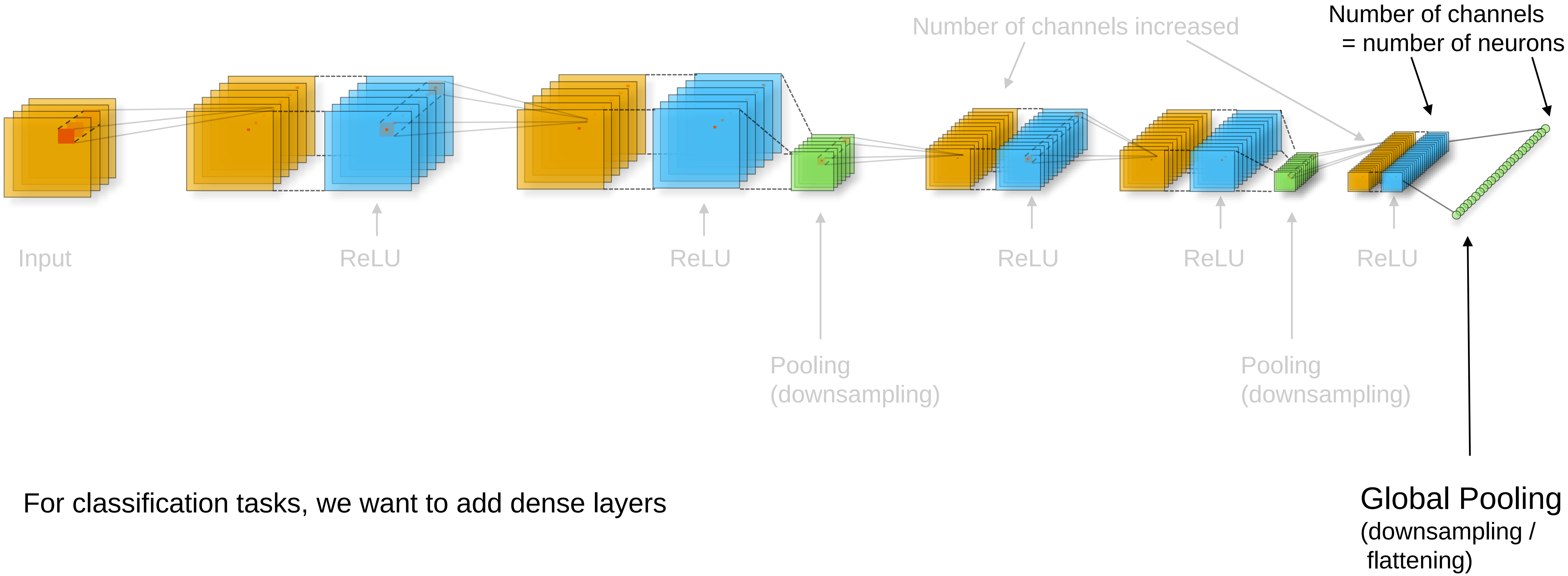
Every pixel *in all output channels* is connected to every neuron in the dense layer

→ weight matrix may become very large!

→ use ***global pooling***

CNN Architecture Part II

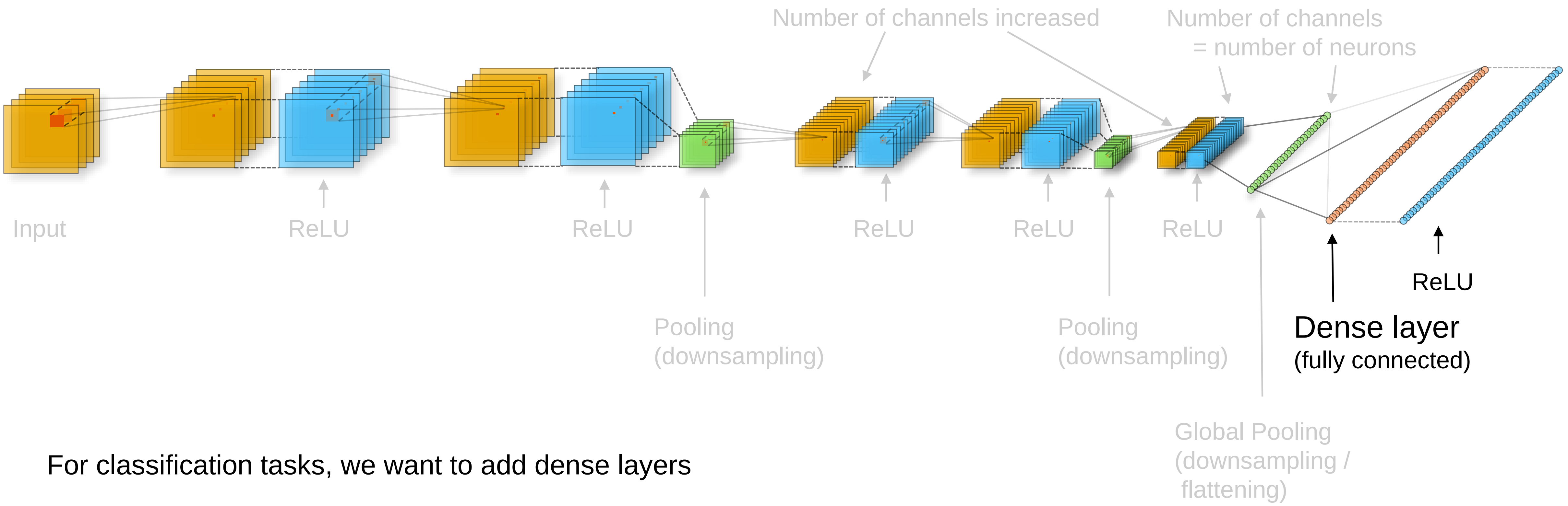
From convolutions to dense layers



For classification tasks, we want to add dense layers

CNN Architecture Part II

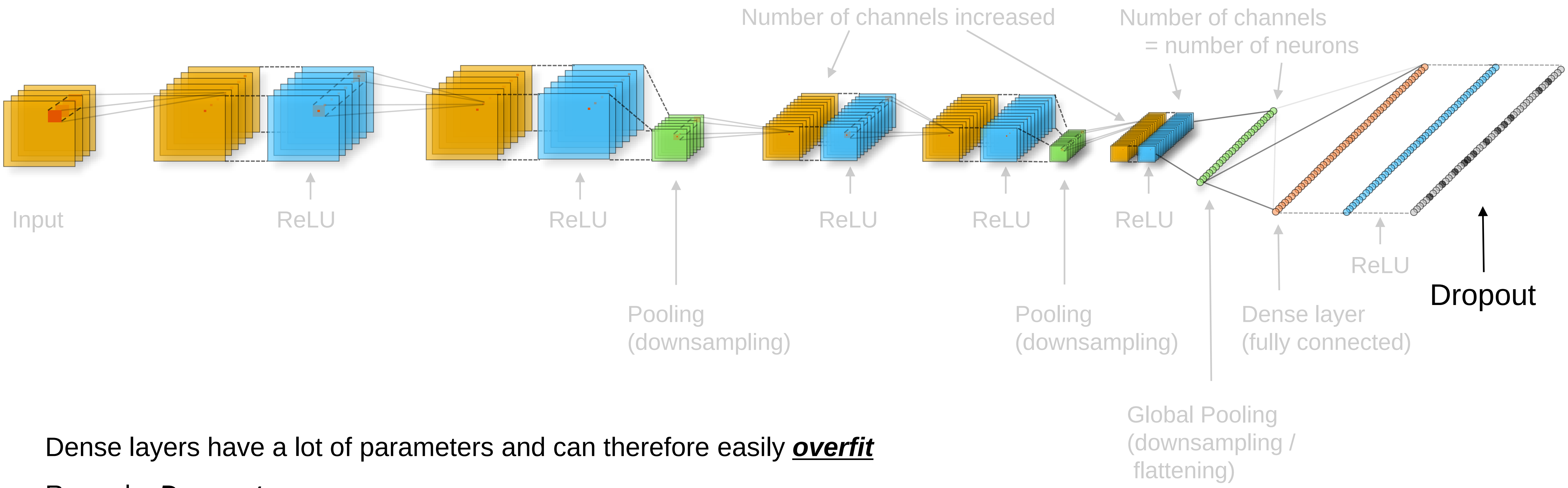
From convolutions to dense layers



For classification tasks, we want to add dense layers

CNN Architecture Part II

Risk of overfitting: dropout



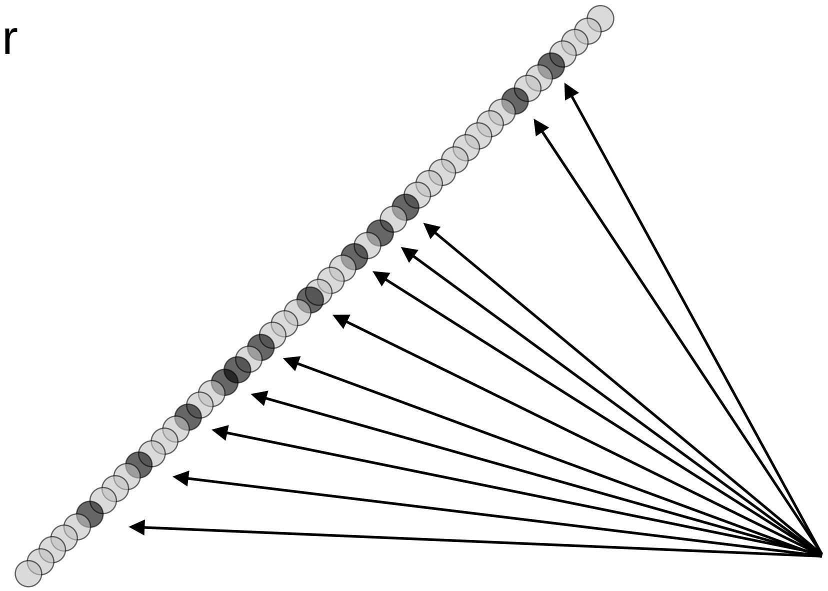
Dense layers have a lot of parameters and can therefore easily ***overfit***

Remedy: ***Dropout***

CNN Architecture Part II

Risk of overfitting: dropout

Dropout Layer



Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov
Journal of Machine Learning Research 15 (2014) 1929-1958

During training, **randomly set neurons to zero** with probability P (e.g. 50% chance)

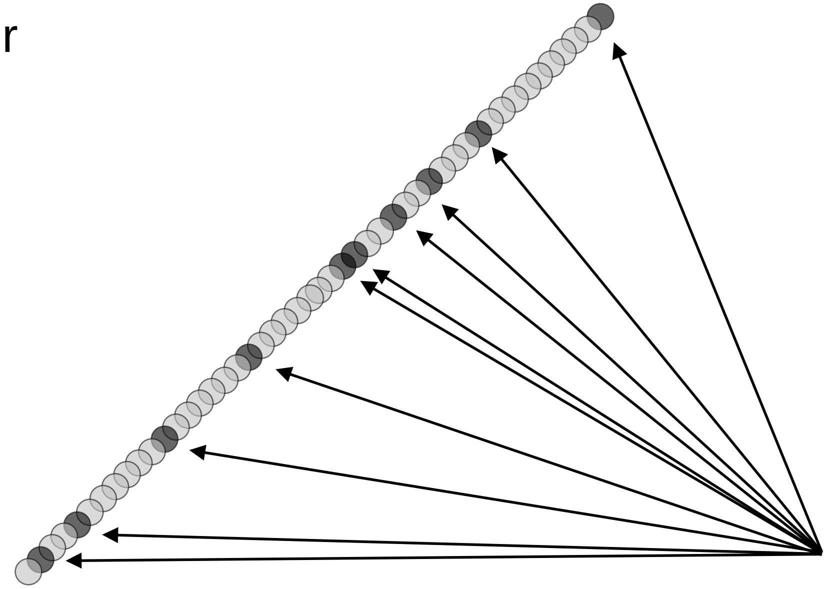
Dense layers have a lot of parameters and can therefore easily ***overfit***

Remedy: ***Dropout***

CNN Architecture Part II

Risk of overfitting: dropout

Dropout Layer



Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov
Journal of Machine Learning Research 15 (2014) 1929-1958

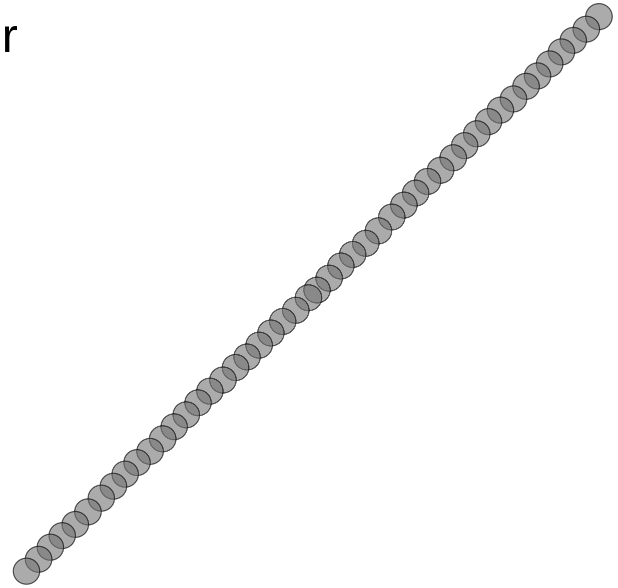
During training, **randomly set neurons to zero** with probability P (e.g. 50% chance)

Dense layers have a lot of parameters and can therefore easily ***overfit***

Remedy: ***Dropout***

CNN Architecture Part II
Risk of overfitting: dropout

Dropout Layer



Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov
Journal of Machine Learning Research 15 (2014) 1929-1958

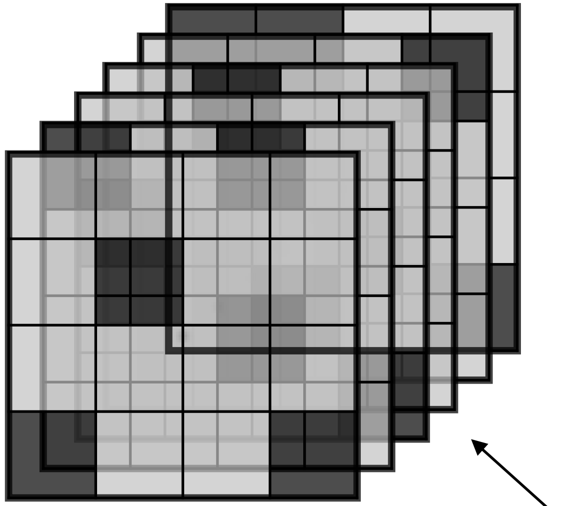
During inference, **rescale neurons** with 1 / P (e.g. factor 2)

Dense layers have a lot of parameters and can therefore easily ***overfit***

Remedy: ***Dropout***

CNN Architecture Part II

Risk of overfitting: dropout

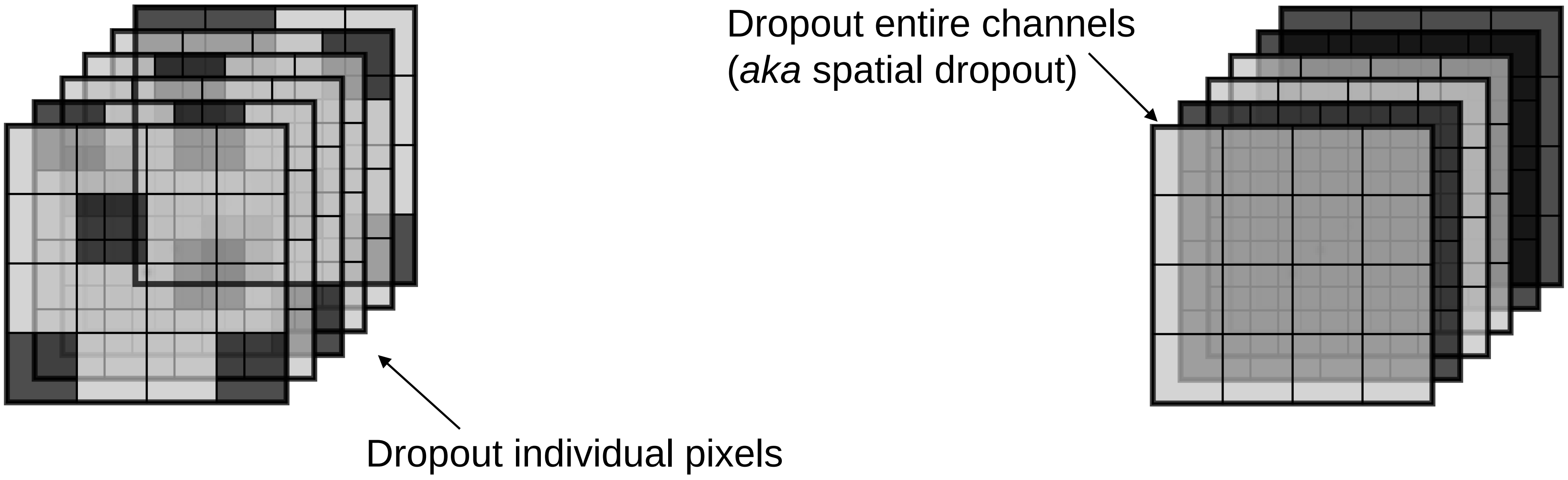


Dropout individual pixels

Dropout can also be applied to **convolutional layers**

CNN Architecture Part II

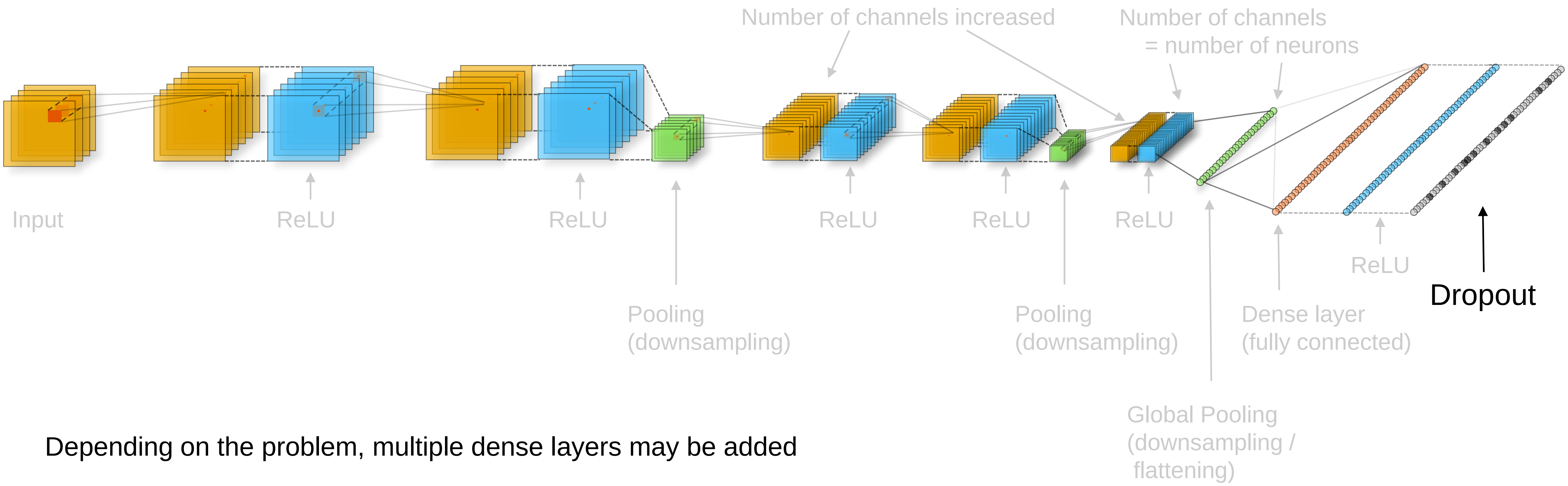
Risk of overfitting: dropout



Dropout can also be applied to **convolutional layers**

CNN Architecture Part II

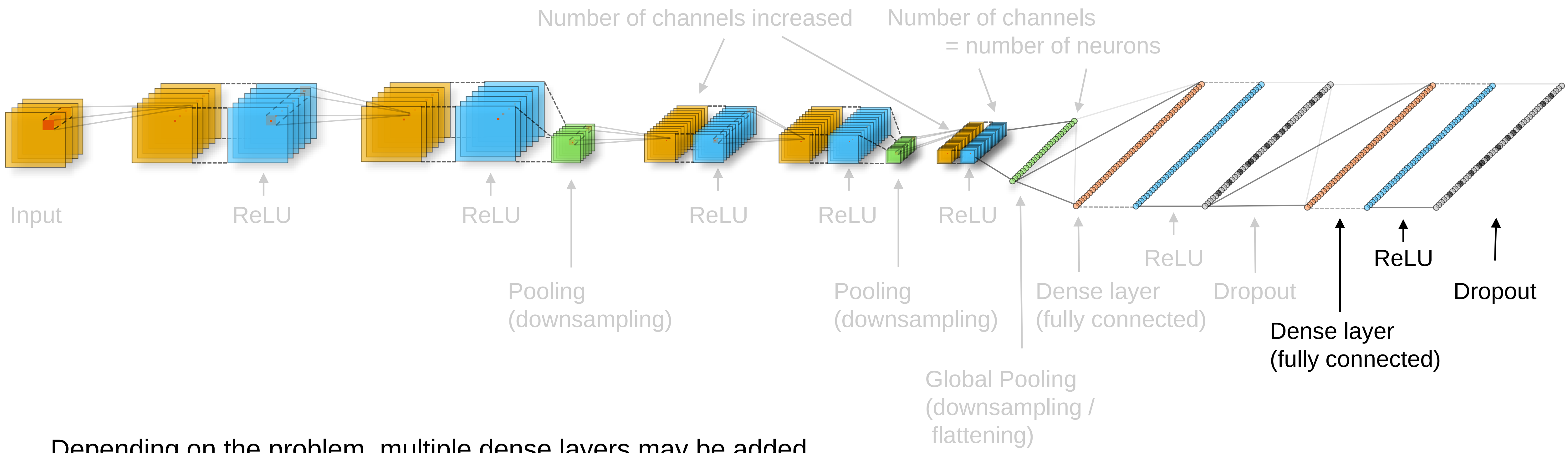
Multiple dense layers



Depending on the problem, multiple dense layers may be added

CNN Architecture Part II

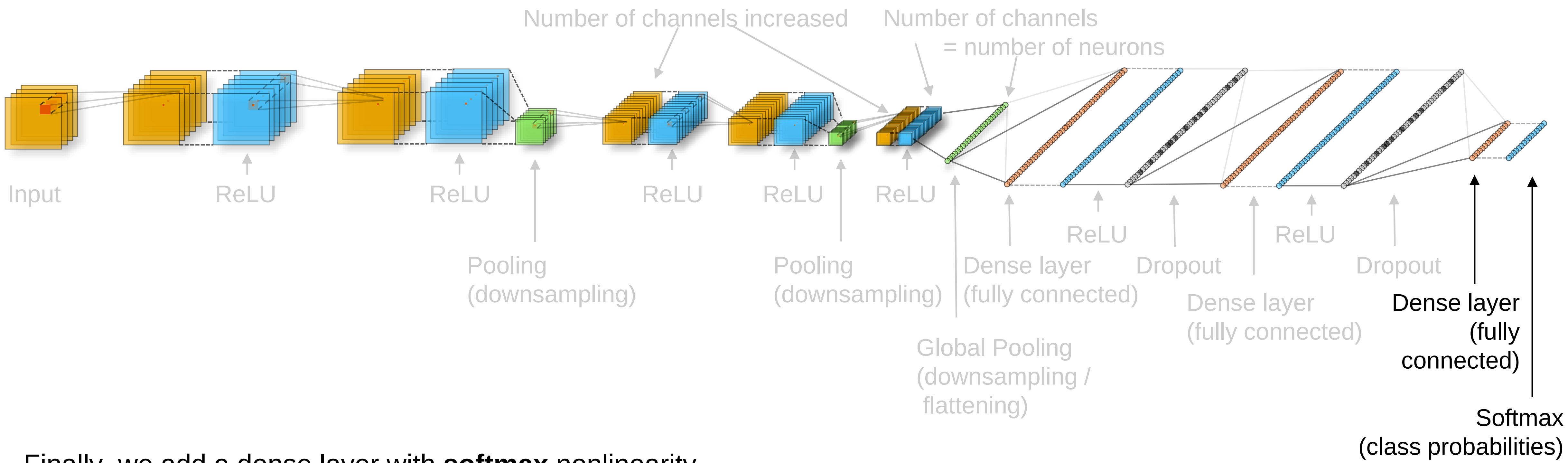
Multiple dense layers



Depending on the problem, multiple dense layers may be added

CNN Architecture Part II

Final dense & classification layer



Finally, we add a dense layer with **softmax** nonlinearity

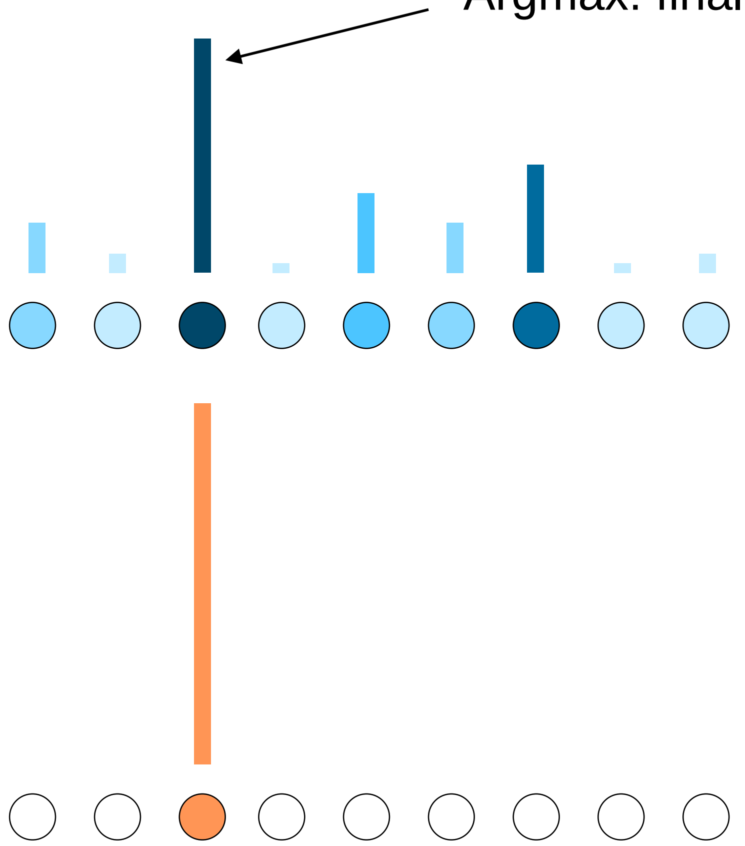
The number of neurons in the final dense layer must equal the number of classes

CNN Architecture Part II

Final dense & classification layer

Argmax: final class prediction

Softmax output:



$$\sum_i P_i = 1$$

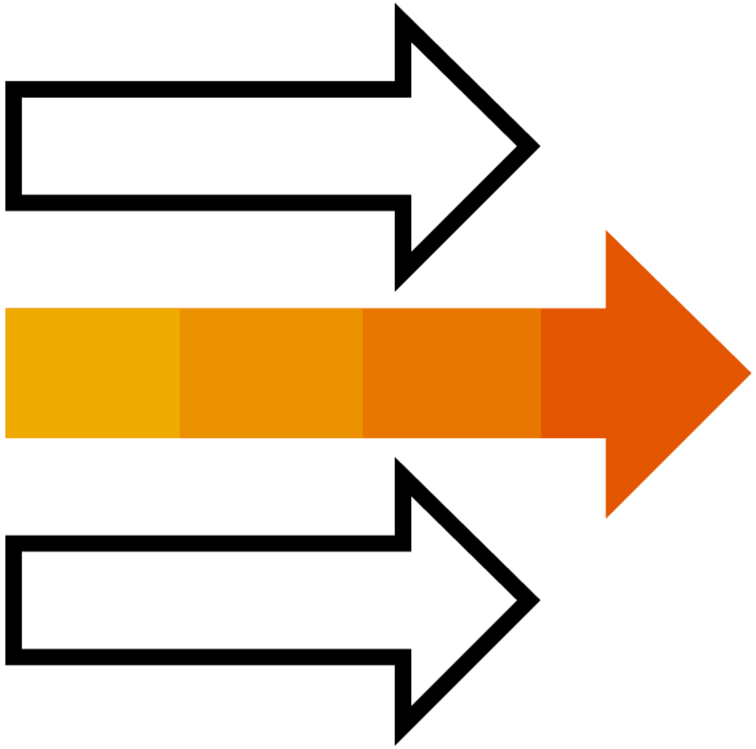
Training label:

CNN Architecture Part II

Coming up next

Accelerating Deep CNN Training

- Computational considerations
- Batch normalization
- Transfer learning
- Residual networks



Thank you.

Contact information:

open@sap.com

Week 4: Convolutional Networks

Unit 4: Accelerating Deep CNN Training

Run Simple

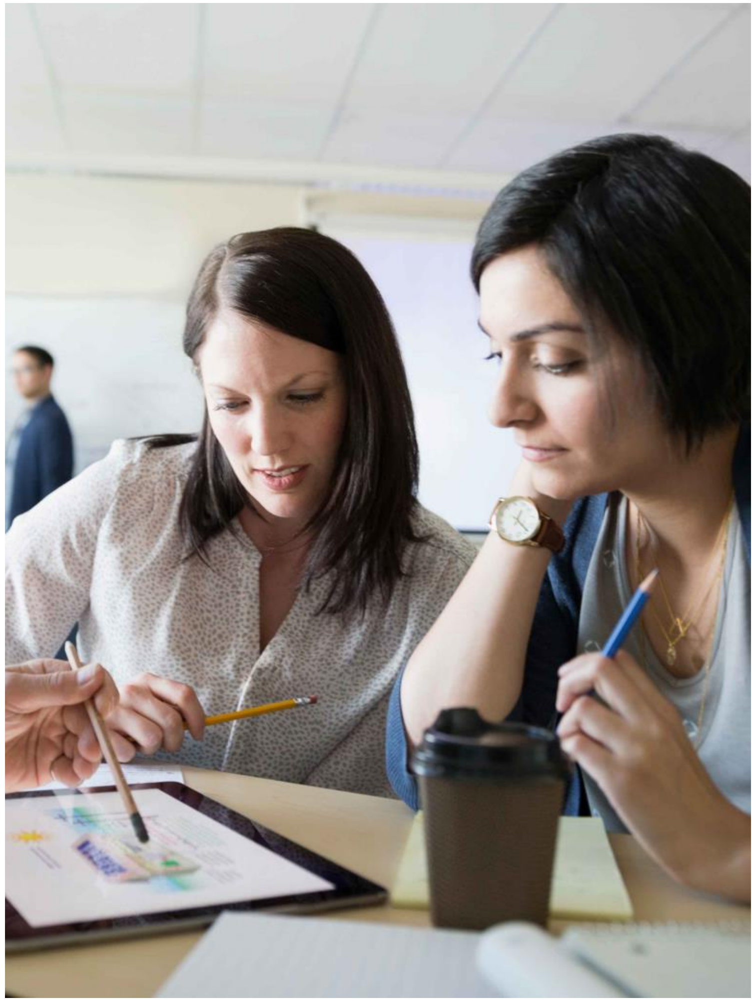
openSAP
open.sap.com

Accelerating Deep CNN Training

What we covered in the last unit

CNN Architecture Part II

- Pooling
- Dense layers
- Dropout
- Softmax

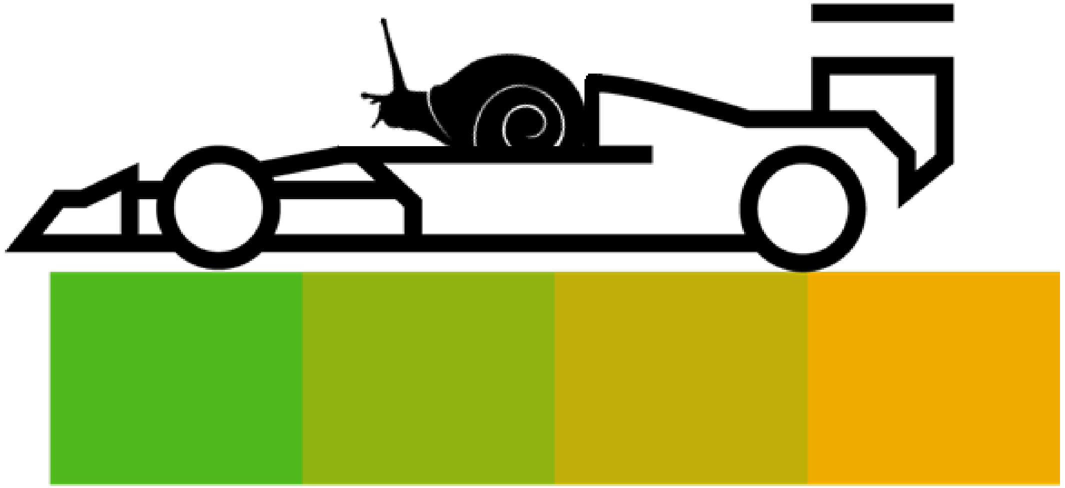


Accelerating Deep CNN Training

Overview

Content:

- Computational considerations
- Batch normalization
- Transfer learning
- Residual networks

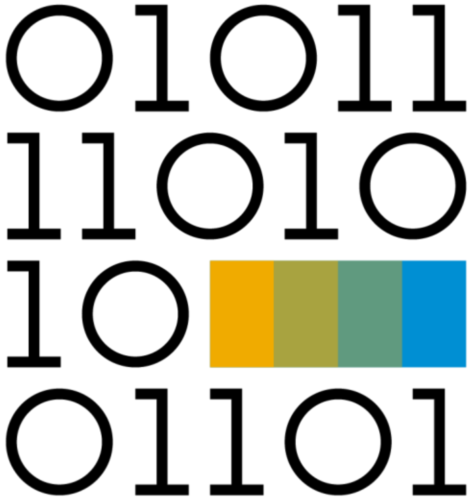


Accelerating Deep CNN Training

Overview

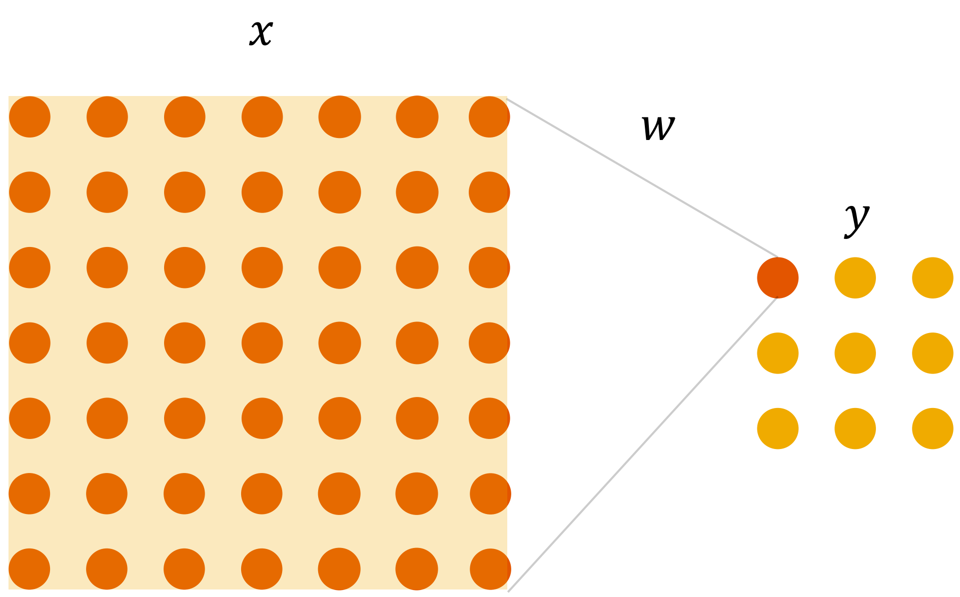
Content:

- **Computational considerations**
- Batch normalization
- Transfer learning
- Residual networks



Accelerating Deep CNN Training

Computational considerations: convolutional filter size



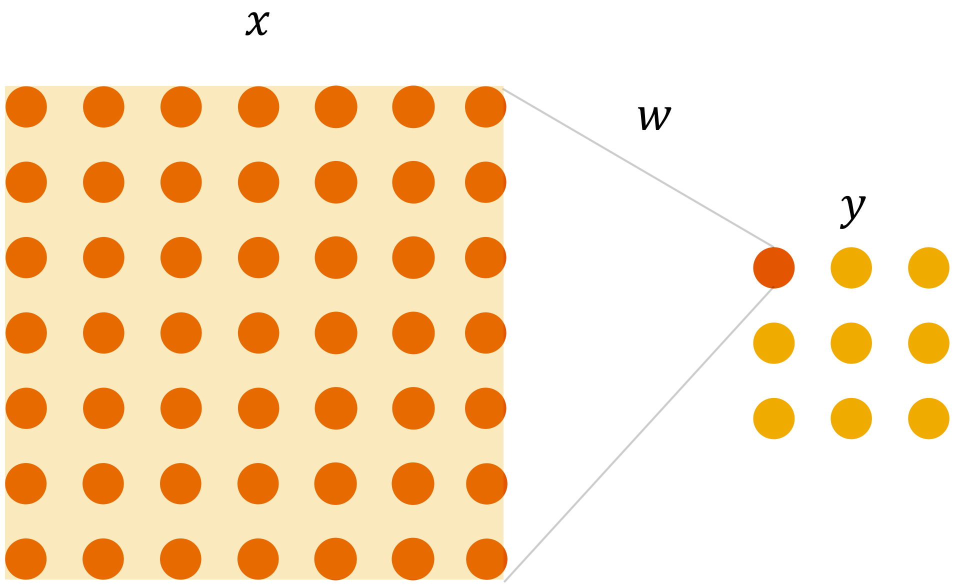
7 x 7 Convolution

100 channels on each layer

49 x 100^2 parameters to optimize

Accelerating Deep CNN Training

Computational considerations: convolutional filter size

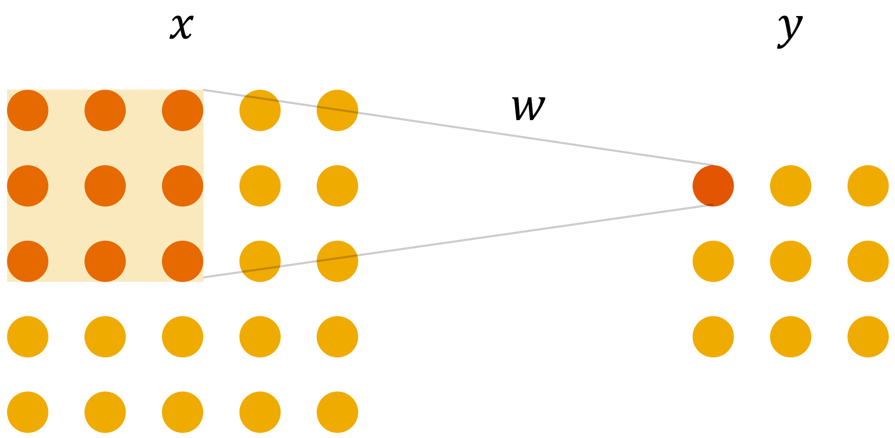


7 x 7 Convolution

100 channels on each layer

49 x 100^2 parameters to optimize

80% less →



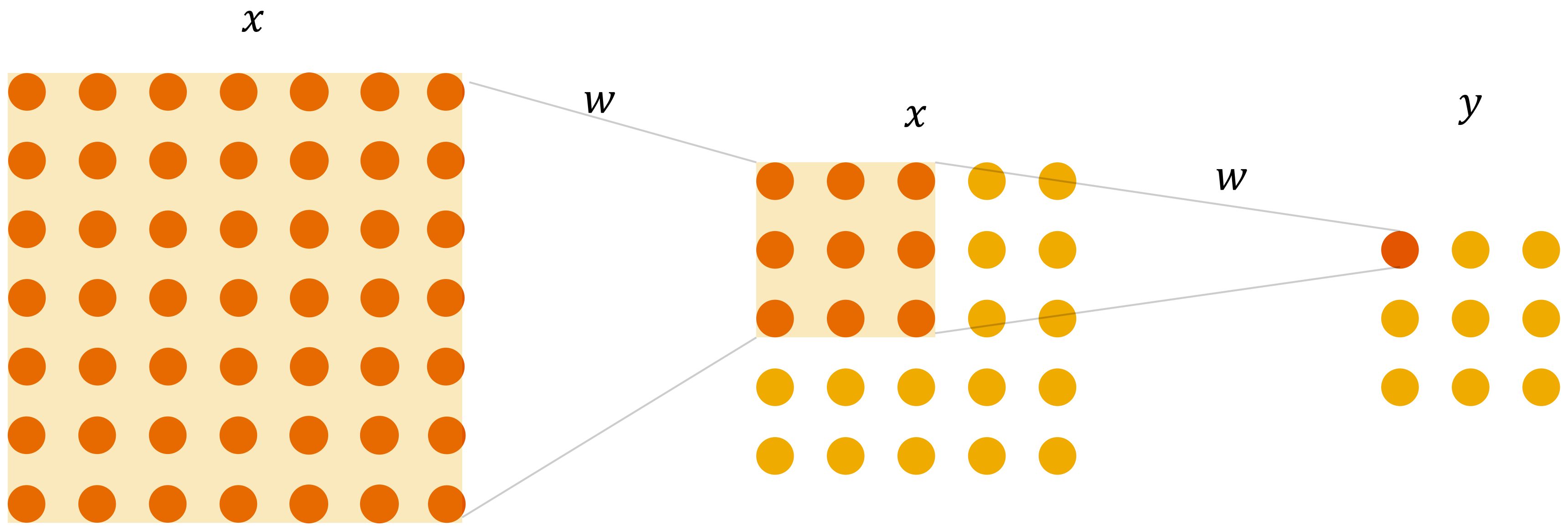
3 x 3 Convolution

100 channels on each layer

9 x 100^2 parameters to optimize

Accelerating Deep CNN Training

Computational considerations: convolutional filter size



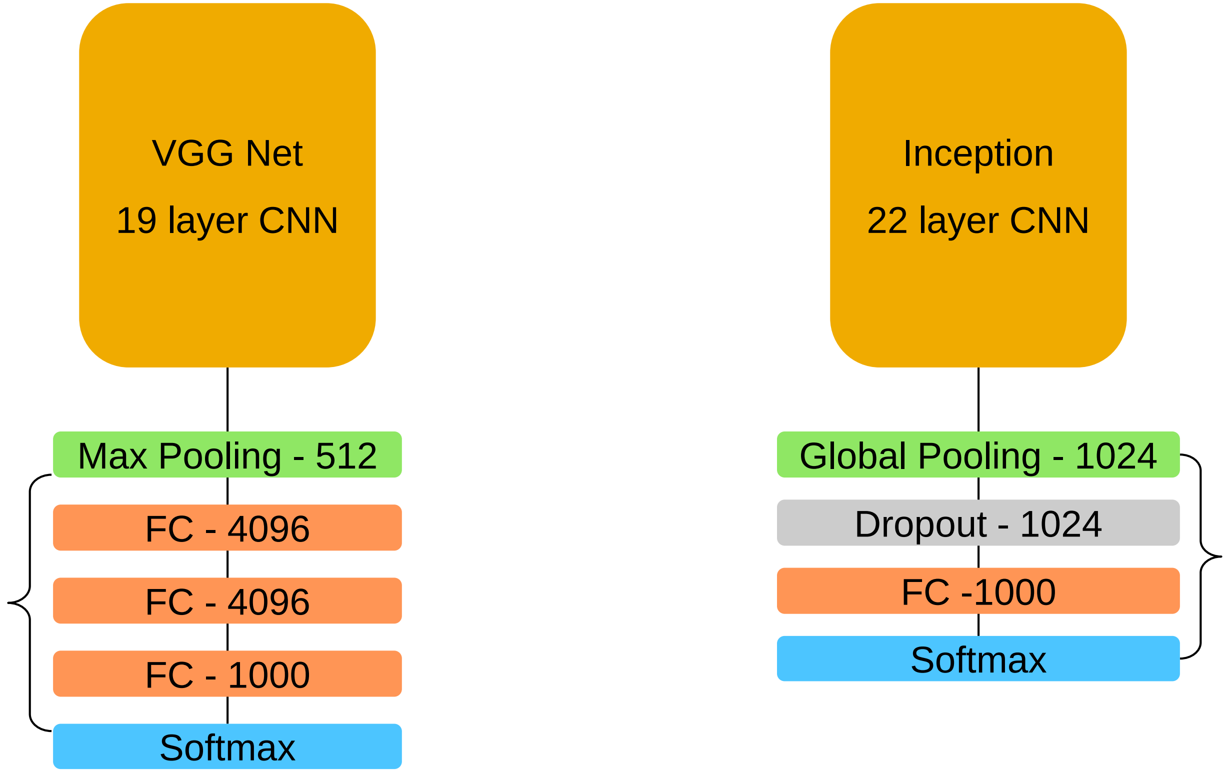
Combination of two 3x3 convolutional layers (2 strides) have the same field of view.

Still **fewer** parameters than 7 x 7, **less** overfitting, **more** nonlinearity!

VGG Net: *Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.*

Accelerating Deep CNN Training

Computational considerations: replacing fully connected (dense) layers



VGG Net:

$w_1 = 512 * 4096$

$w_2 = 4096^2$

$w_3 = 4096 * 1000$

~ 23 M parameters

Inception:

$w_1 = 0$

$w_2 = 1024 * 1000$

~ 1M parameters

VGG Net: Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

Inception: Szegedy C. et al. (2015). Going Deeper with Convolutions. CoRR. abs/1409.4842.

Accelerating Deep CNN Training

Overview

Content:

- Computational considerations
- **Batch normalization**
- Transfer learning
- Residual networks

$$\sqrt{b^2 - 4ac}$$

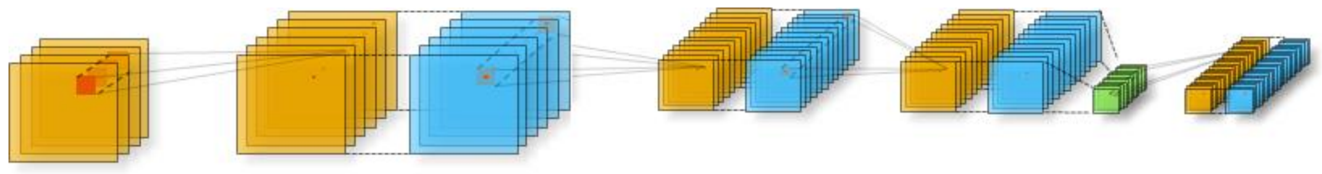
$$\int xdy$$

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

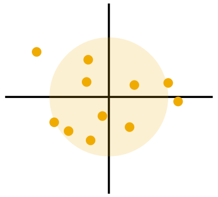
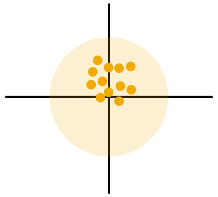
$$e^{-i\omega t}$$

Accelerating Deep CNN Training

Batch normalization: motivation

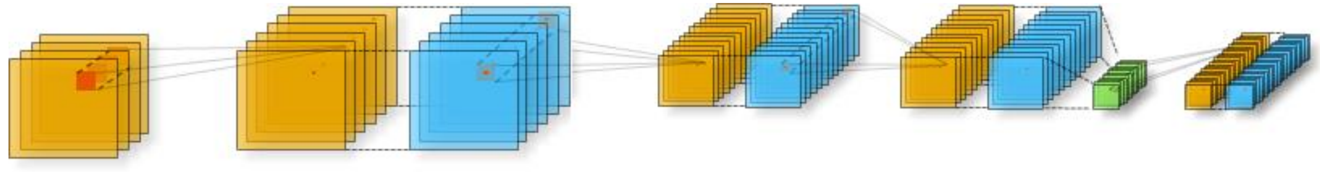


- During training, the output distribution of each layer changes due to weight updates

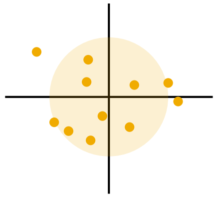
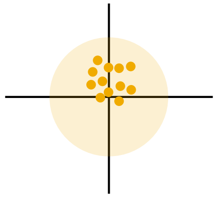
T=0: Conv 1 →  → Conv 2 →  → Conv 3 → ….

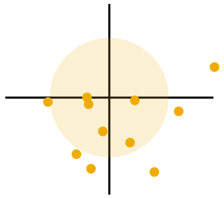
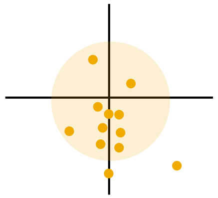
Accelerating Deep CNN Training

Batch normalization: motivation



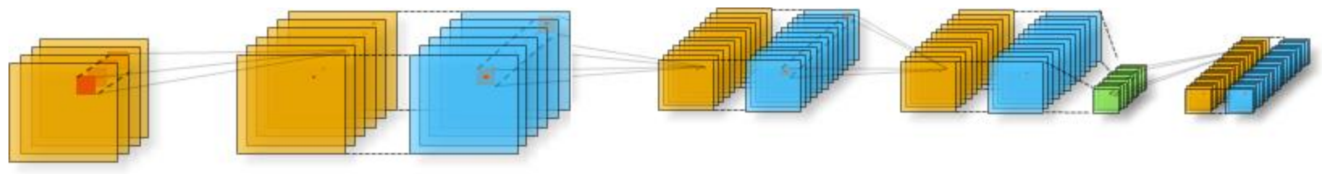
- During training, the output distribution of each layer changes due to weight updates

T=0: Conv 1 →  → Conv 2 →  → Conv 3 → ….

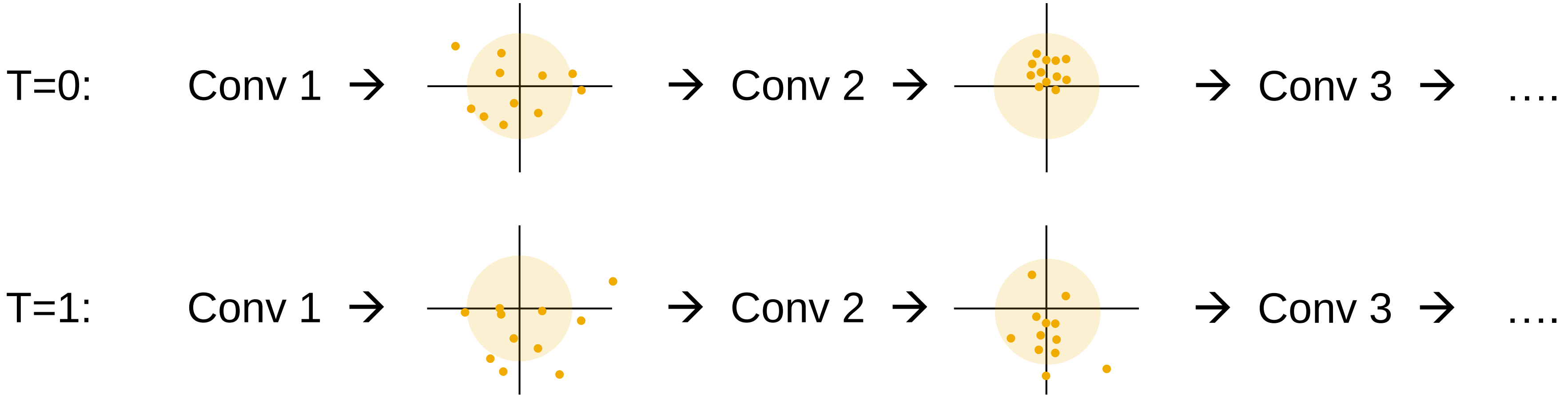
T=1: Conv 1 →  → Conv 2 →  → Conv 3 → ….

Accelerating Deep CNN Training

Batch normalization: motivation



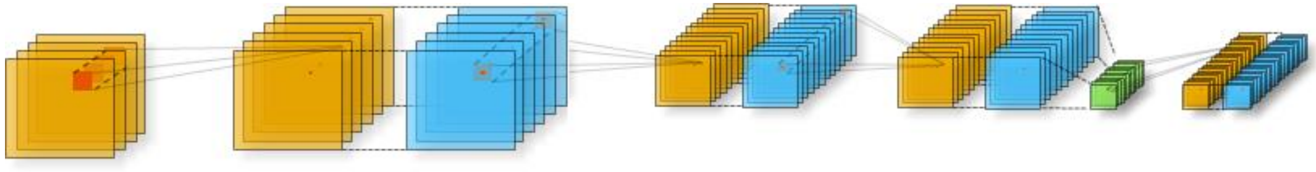
- During training, the output distribution of each layer changes due to weight updates
- Each following layer has to adapt to the new output distribution of the previous layer
- This makes training rather hard



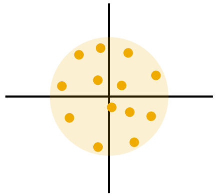
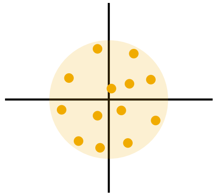
Aka “**internal covariate shift**”

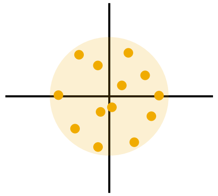
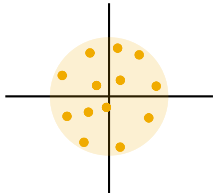
Accelerating Deep CNN Training

Batch normalization: motivation



- **Idea**: Normalize output distribution after each layer

T=0: Conv 1 →  → Conv 2 →  → Conv 3 → ….

T=1: Conv 1 →  → Conv 2 →  → Conv 3 → ….

Accelerating Deep CNN Training

Batch normalization algorithm

Input: Values of x over a mini-batch $B = \{x_{i..m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}\,(x_i)\}$

Batch Normalization: *Ioffe, S. & Szegedy C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR, abs/1502.03167.*

Accelerating Deep CNN Training

Batch normalization algorithm

Input: Values of x over a mini-batch $B = \{x_{i..m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}\,(x_i)\}$

1. **Normalize** every batch **by mean and variance** of the batch.

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\alpha_\beta^2 + \epsilon}}$$

Batch Normalization: *Ioffe, S. & Szegedy C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR, abs/1502.03167.*

Accelerating Deep CNN Training

Batch normalization algorithm

Input: Values of x over a mini-batch $B = \{x_{i..m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}\,(x_i)\}$

1. **Normalize** every batch **by mean and variance** of the batch.

$$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\alpha_\beta^2 + \epsilon}}$$

2. Introduce **two new parameters** γ, β.

Batch Normalization: *Ioffe, S. & Szegedy C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR, abs/1502.03167.*

Accelerating Deep CNN Training

Batch normalization algorithm

Input: Values of x over a mini-batch $B = \{x_{i..m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}\,(x_i)\}$

1. **Normalize** every batch **by mean and variance** of the batch.

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\alpha_\beta^2 + \epsilon}}$$

2. Introduce **two new parameters** γ, β.

3. **Scale and shift** the activations with γ and β.

$$y_i \leftarrow \gamma \widehat{x_i} + \beta \equiv BN_{\gamma,\beta}\,(x_i)$$

Batch Normalization: *Ioffe, S. & Szegedy C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR, abs/1502.03167.*

Accelerating Deep CNN Training

Batch normalization algorithm

Input: Values of x over a mini-batch $B = \{x_{i..m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}\,(x_i)\}$

1. **Normalize** every batch **by mean and variance** of the batch.

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\alpha_\beta^2 + \epsilon}}$$

2. Introduce **two new parameters** γ, β.

3. **Scale and shift** the activations with γ and β.

$$y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}\,(x_i)$$

4. **Learn** the parameters γ and β during training.

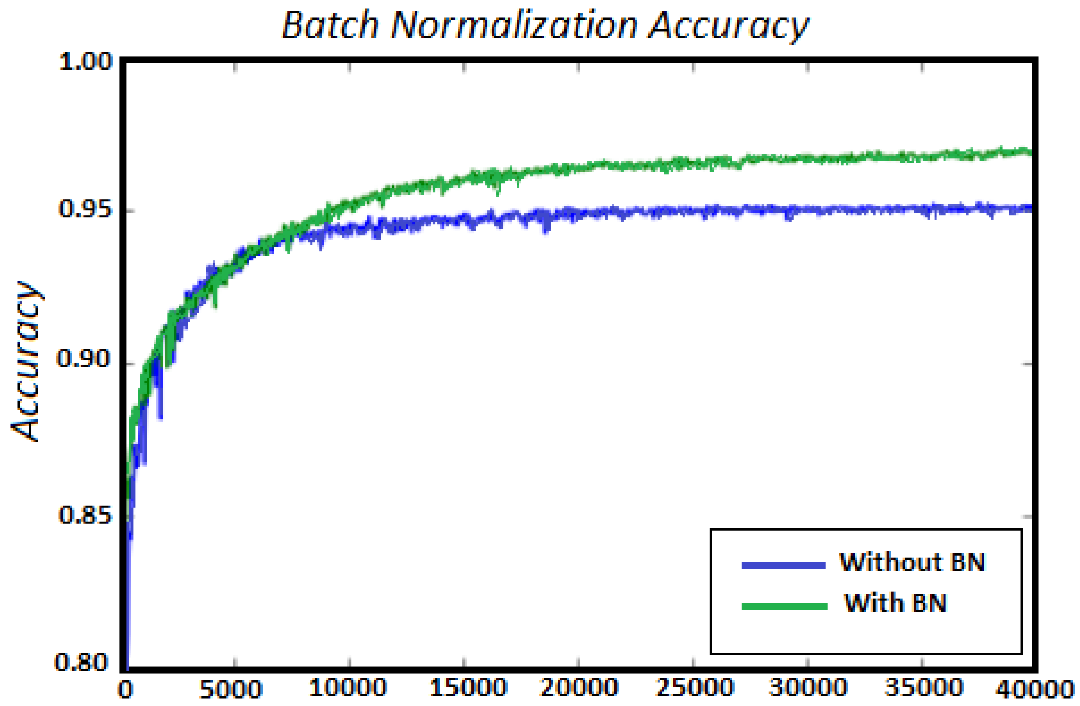
Batch Normalization: *Ioffe, S. & Szegedy C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR, abs/1502.03167.*

Accelerating Deep CNN Training

Batch normalization algorithm

Benefits:

- Higher accuracy in earlier training steps
- Reduction in overfitting
- Less need for dropout layers

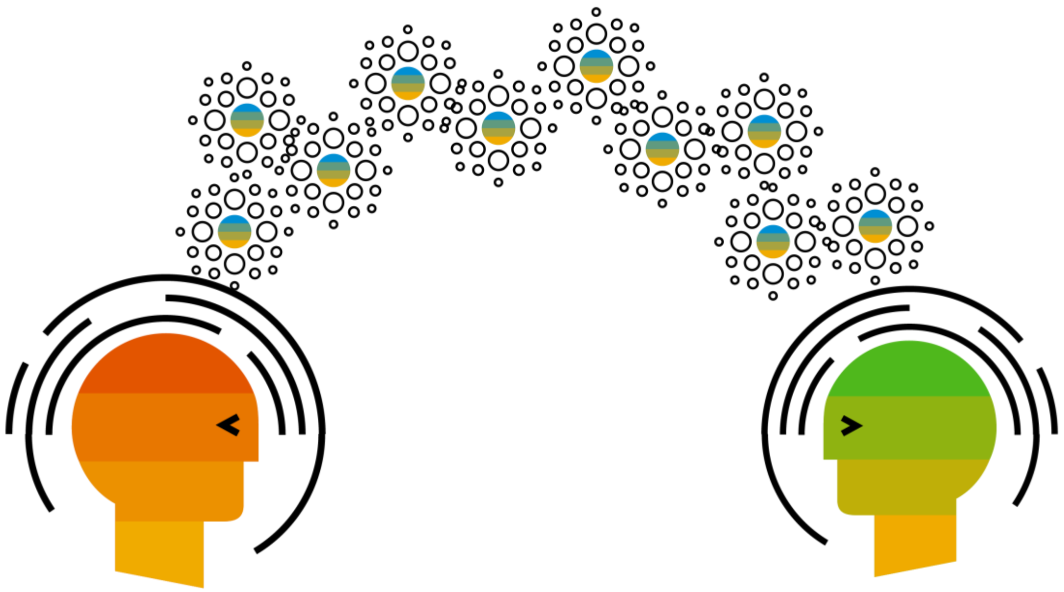


Accelerating Deep CNN Training

Overview

Content:

- Computational considerations
- Batch normalization
- **Transfer learning**
- Residual networks



Accelerating Deep CNN Training

Transfer learning: motivation

Problem: Deep neural networks require very large data sets to train, and large computational resources to build from scratch

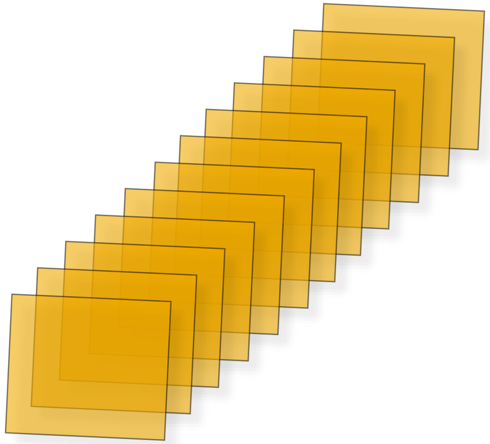
Accelerating Deep CNN Training

Transfer learning: motivation

Problem: Deep neural networks require very large data sets to train, and large computational resources to build from scratch

Famous CNN models **AlexNet, ZFNet, VGGNet, GoogleNet, Microsoft ResNet…**

- Trained on millions of images
- Powerful GPUs used for training
- Training takes days or even weeks

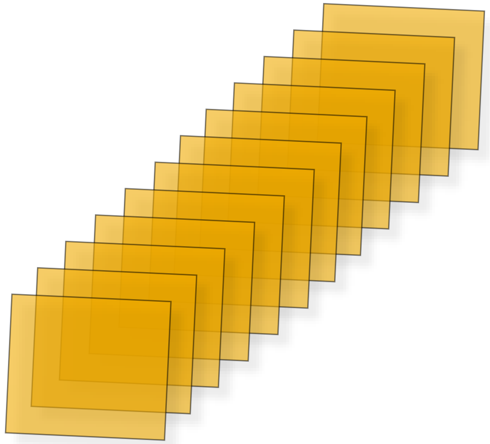


Accelerating Deep CNN Training

Transfer learning: motivation

Problem: Deep neural networks require very large data sets to train, and large computational resources to build from scratch

Famous CNN models **AlexNet, ZFNet, VGGNet, GoogleNet, Microsoft ResNet…**

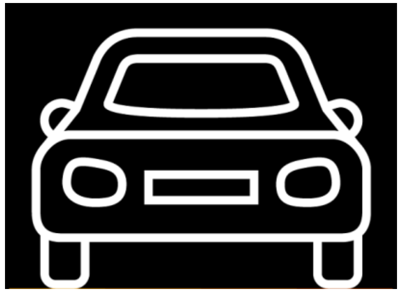


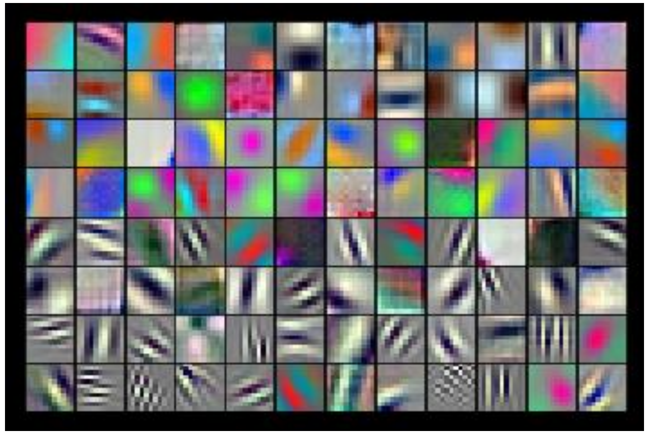
- Trained on millions of images
- Powerful GPUs used for training
- Training takes days or even weeks

Solution: Transfer learning (recycle pretrained model)

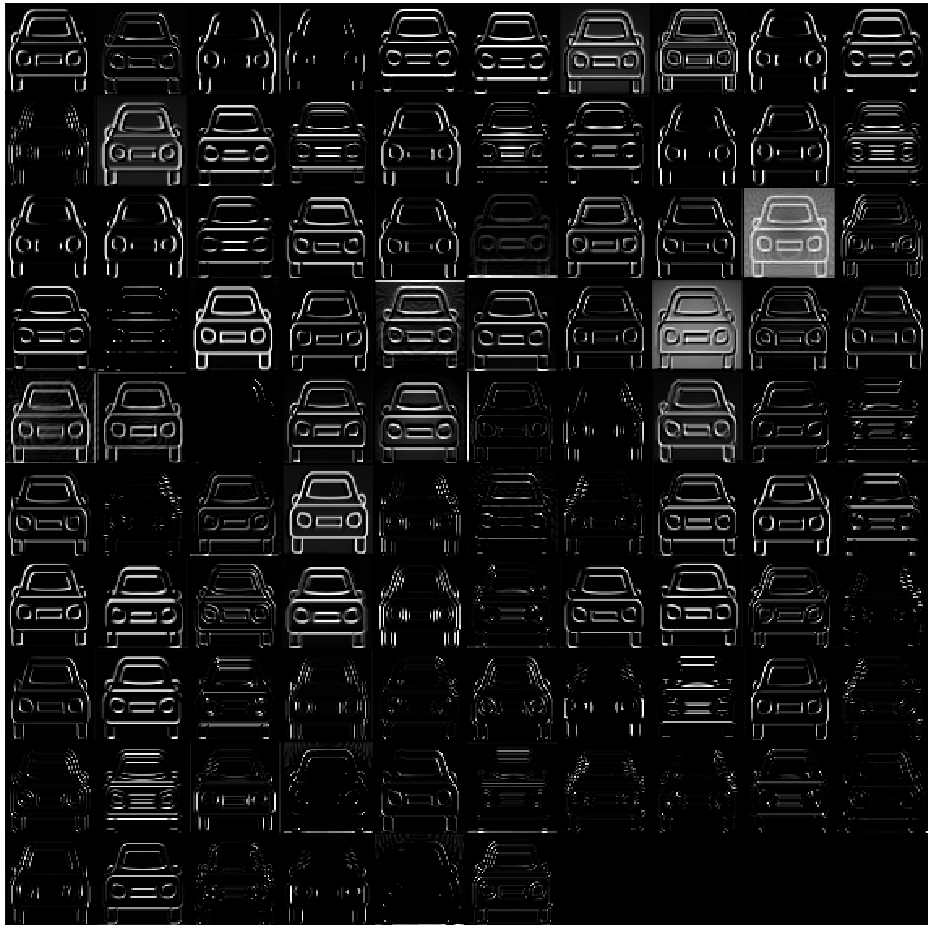
Accelerating Deep CNN Training

Transfer learning: reusing joint features





Low-level filters
conv-1 AlexNet



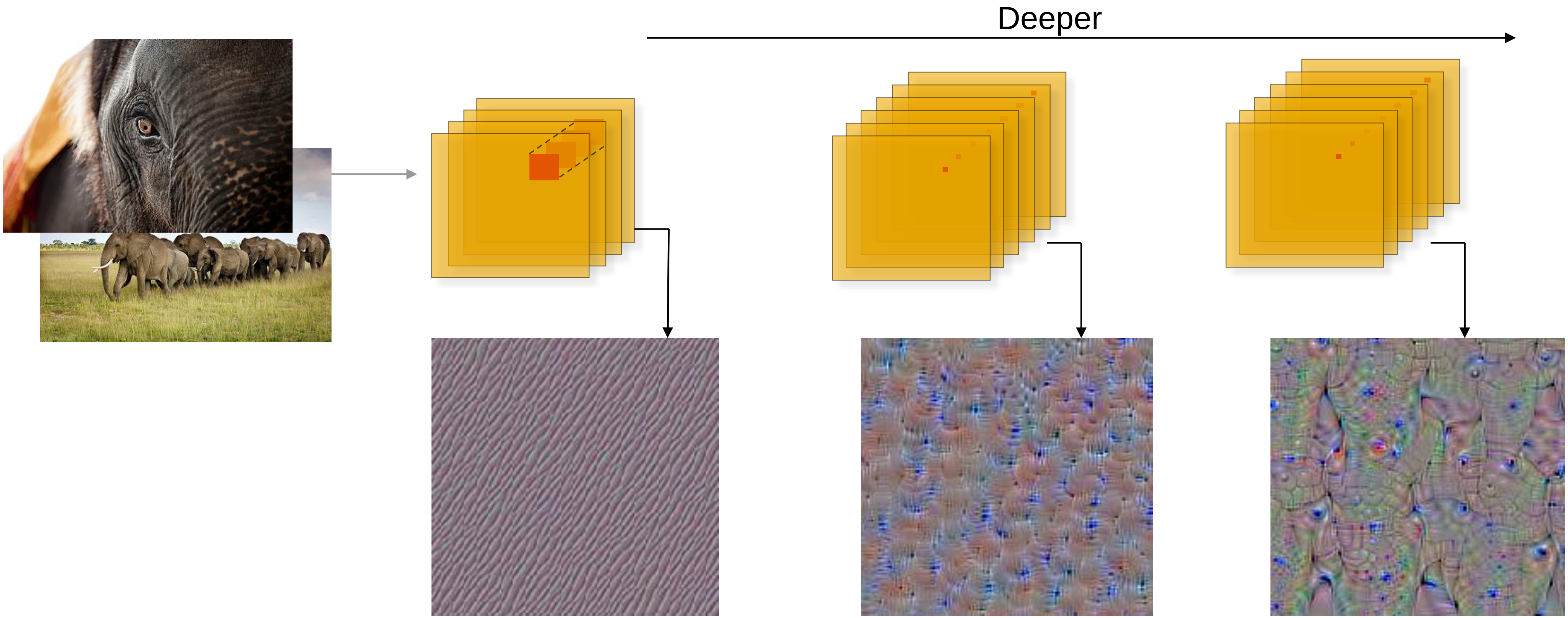
Low-level features
Feature map-1 AlexNet

DeCAF: *Donahue, J. et al. (2014).* DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *CoRR, abs/. 1310.1531*

AlexNet: *Krizhevsky, A. et al.(2012). ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM* 60, 6.

Accelerating Deep CNN Training

Transfer learning: reusing joint features



Low-level activations

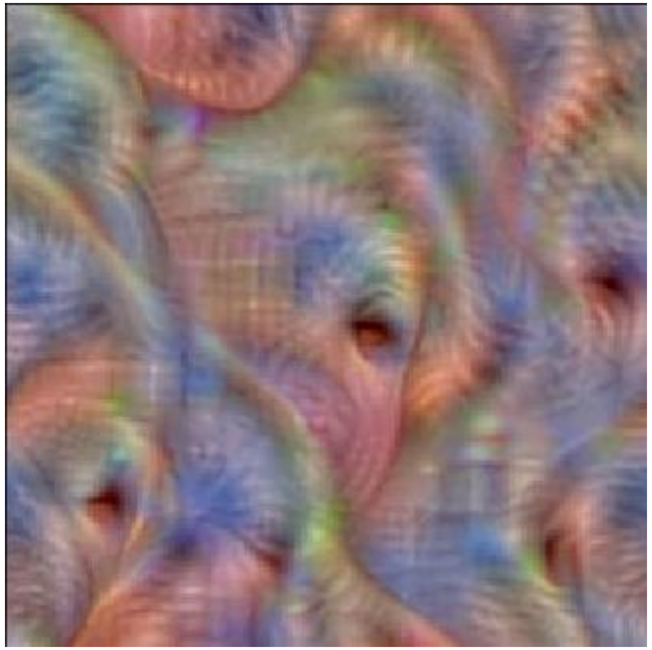
Mid-level activations

High-level activations

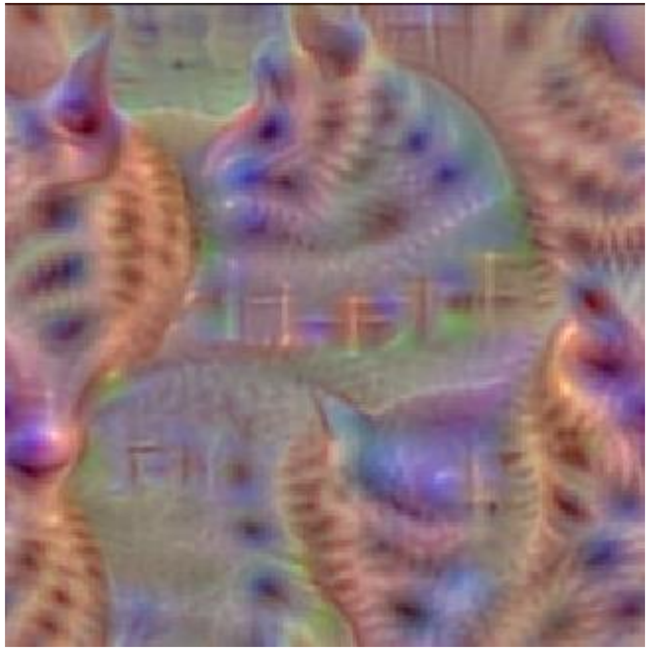
VGG Net: Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556

Accelerating Deep CNN Training

Transfer learning: last layer activations



Dog



Cat

Elephant

VGG Net: Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556

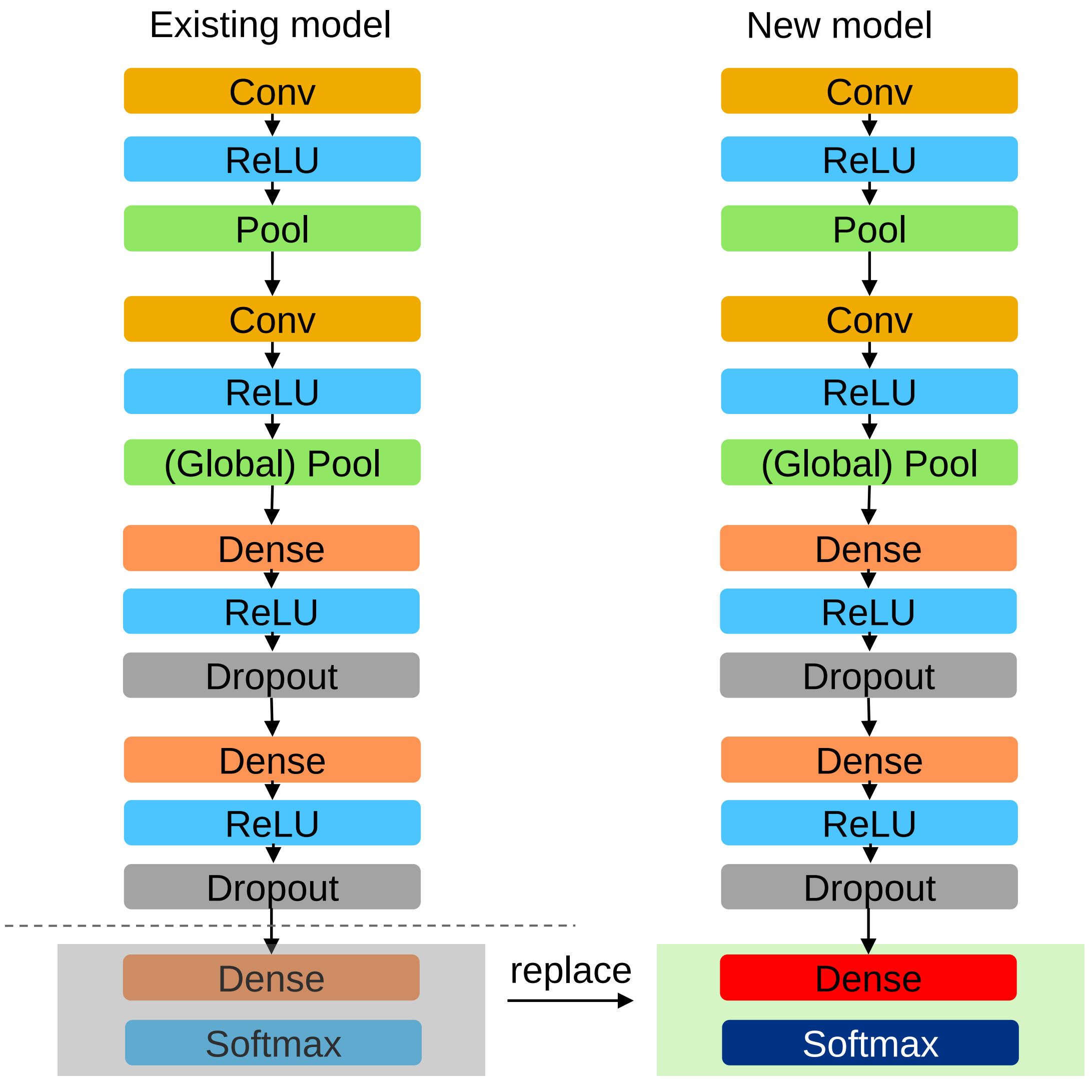
Accelerating Deep CNN Training

Transfer learning: application

Scenario 1:

Replacing last layer with new classifier

- Retrain on small data set
- Similar features



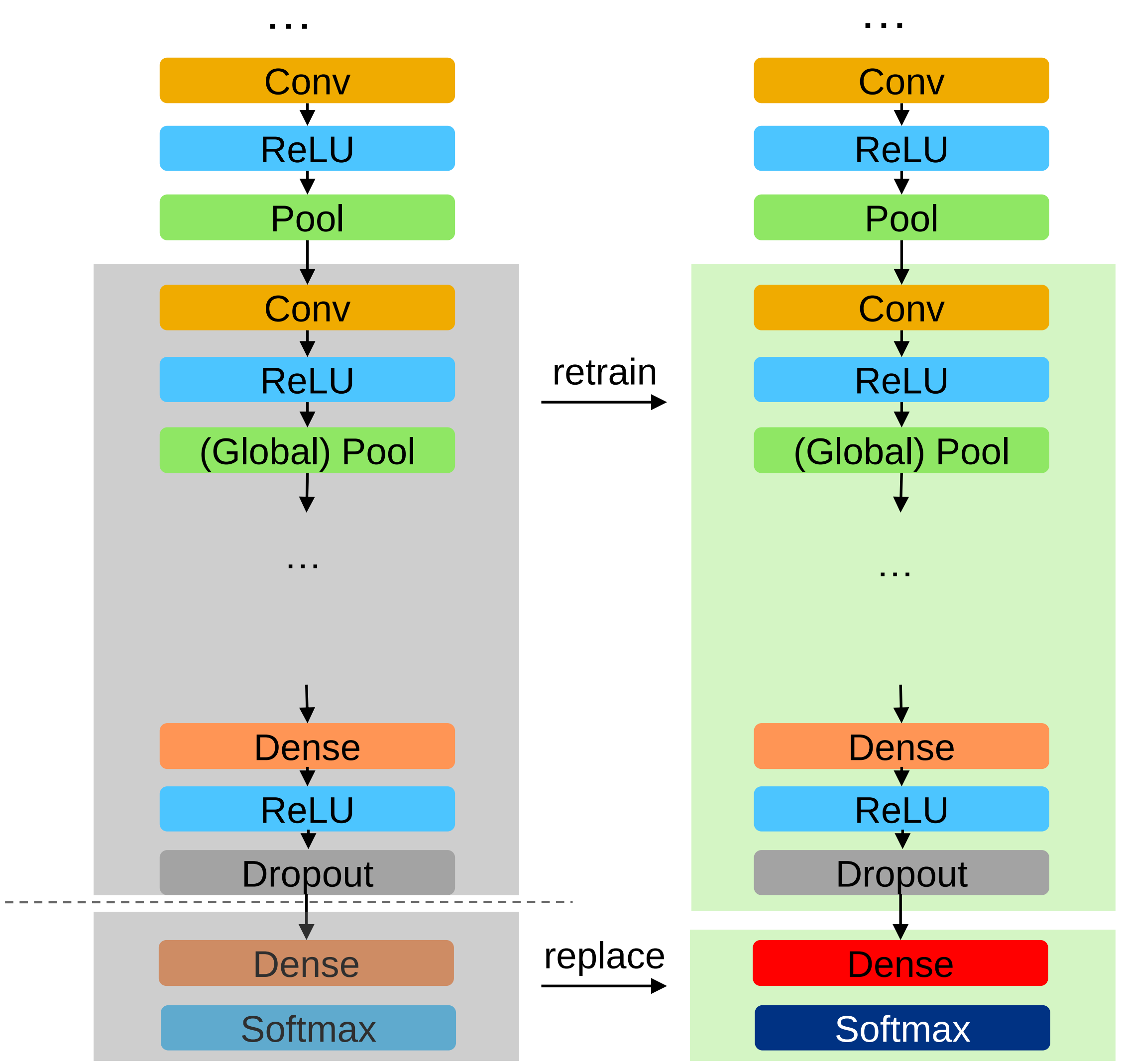
Accelerating Deep CNN Training

Transfer learning: application

Scenario 1:

Fine-tuning more layers

- Retrain on large data set
- Similar features

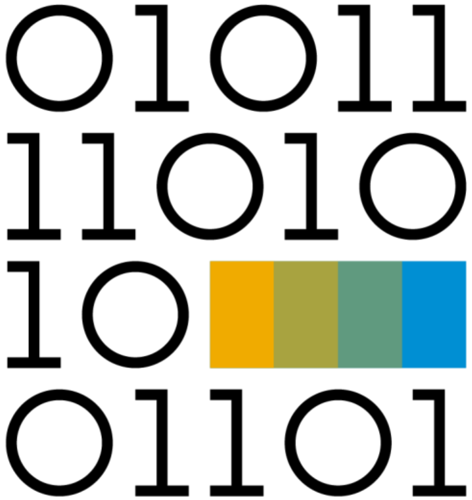


Accelerating Deep CNN Training

Overview

Content:

- Computational considerations
- Batch normalization
- Transfer learning
- **Residual networks**



Accelerating Deep CNN Training

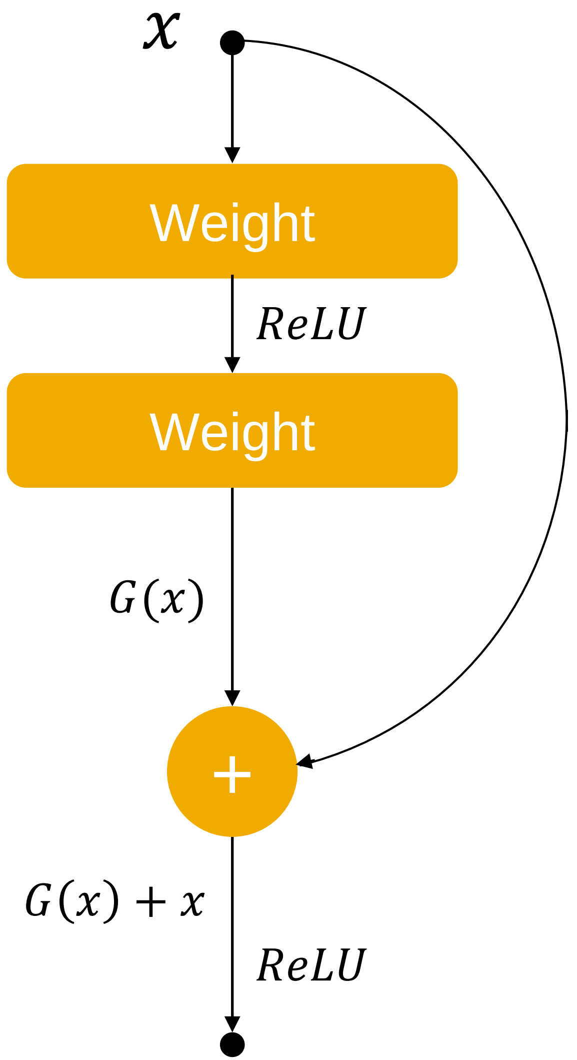
Deep residual networks: Extremely Deep Networks *[He, Zhang, Ren, Sun, 2015]*

- ***Intuition:*** Deeper networks are more expressive
- ***Problem:*** Deep networks are hard to train

Accelerating Deep CNN Training

Deep residual networks: Extremely Deep Networks *[He, Zhang, Ren, Sun, 2015]*

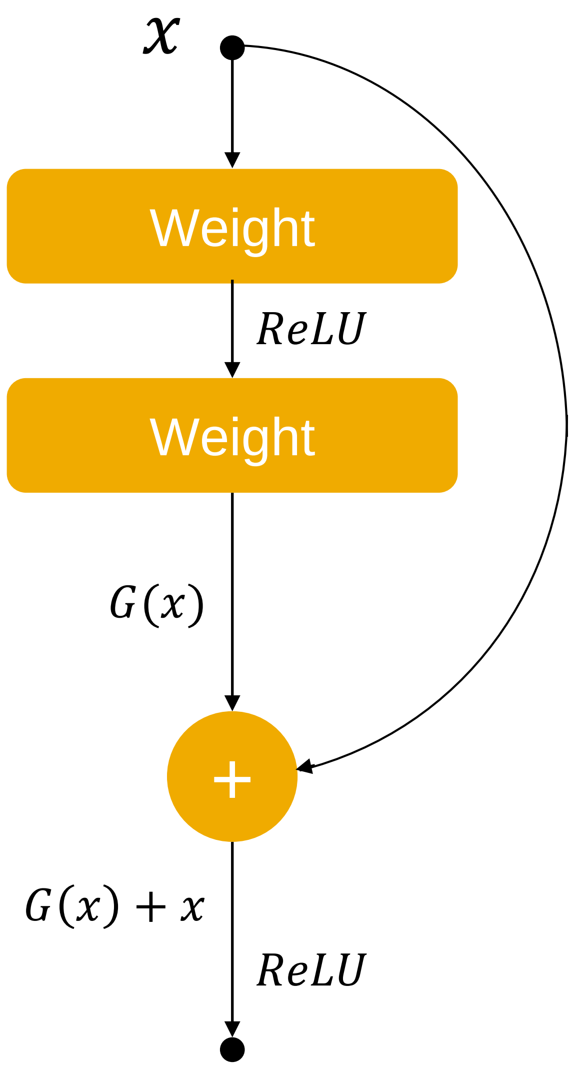
- ***Intuition:*** Deeper networks are <u>more expressive</u>
- ***Problem:*** Deep networks are <u>hard to train</u>
- ***Idea:*** The modeled function of each building block has a higher resemblance to the <u>identity function</u> (than to the zero function)

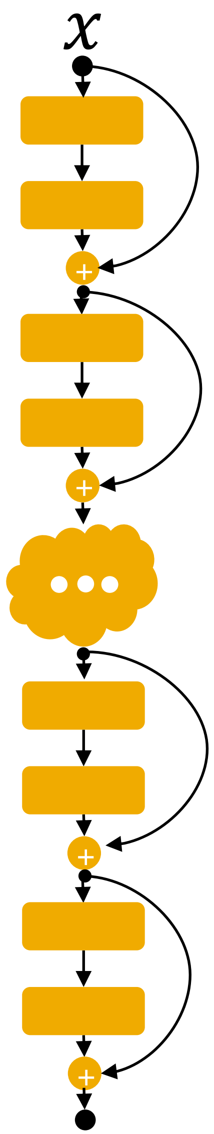


Accelerating Deep CNN Training

Deep residual networks: Extremely Deep Networks *[He, Zhang, Ren, Sun, 2015]*

- ***Intuition:*** Deeper networks are <u>more expressive</u>
- ***Problem:*** Deep networks are <u>hard to train</u>
- ***Idea:*** The modeled function of each building block has a higher resemblance to the <u>identity function</u> (than to the zero function)
- The signal can *directly skip* through multiple layers (gradient more easily flows from top to bottom layers)
- Allows very deep architectures (typically more than 100 layers)



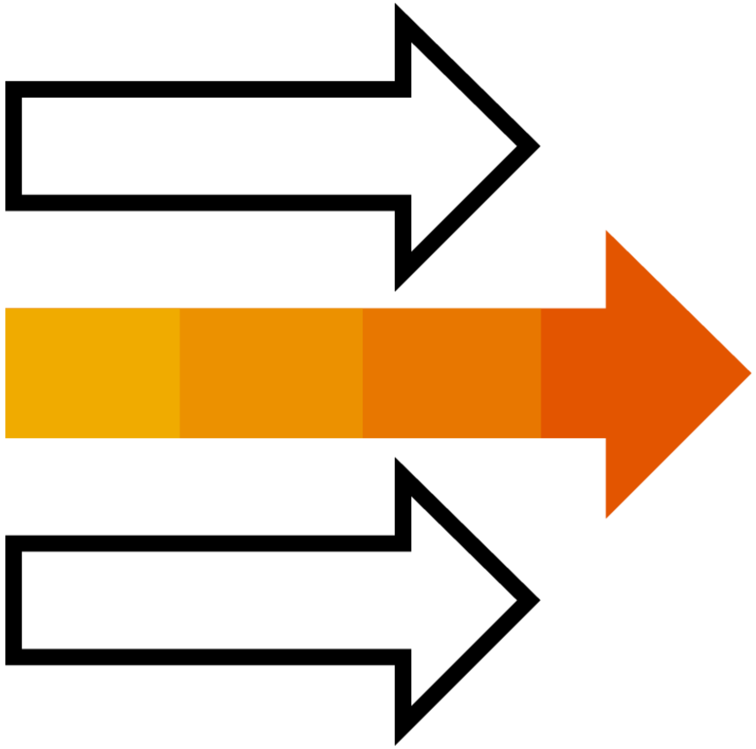


Accelerating Deep CNN Training

Coming up next

Applications of CNNs

- Object detection
- Semantic image segmentation



Thank you.

Contact information:

open@sap.com

Week 4: Convolutional Networks

Unit 5: Applications of CNNs

Run Simple

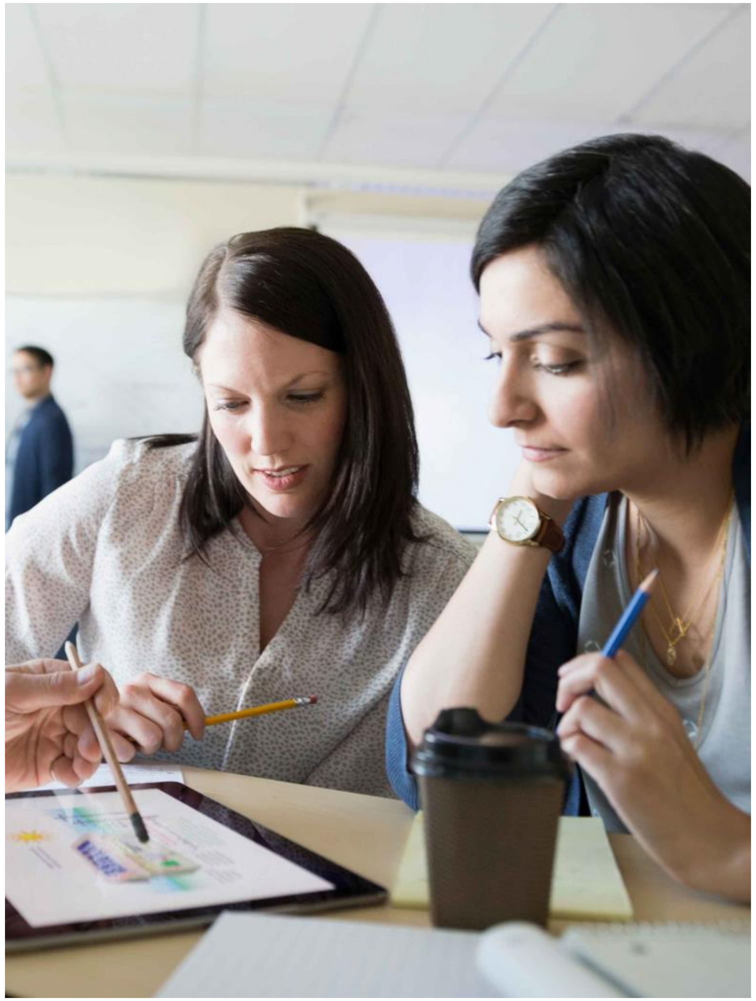
openSAP
open.sap.com

Applications of CNNs

What we covered in the last unit

Accelerating Deep CNN Training

- Computational considerations
- Batch normalization
- Transfer learning
- Residual networks

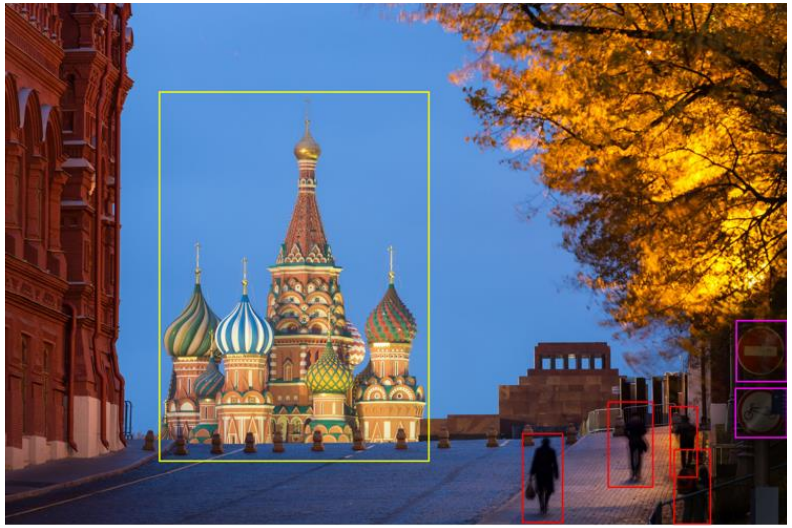


Applications of CNNs

Overview

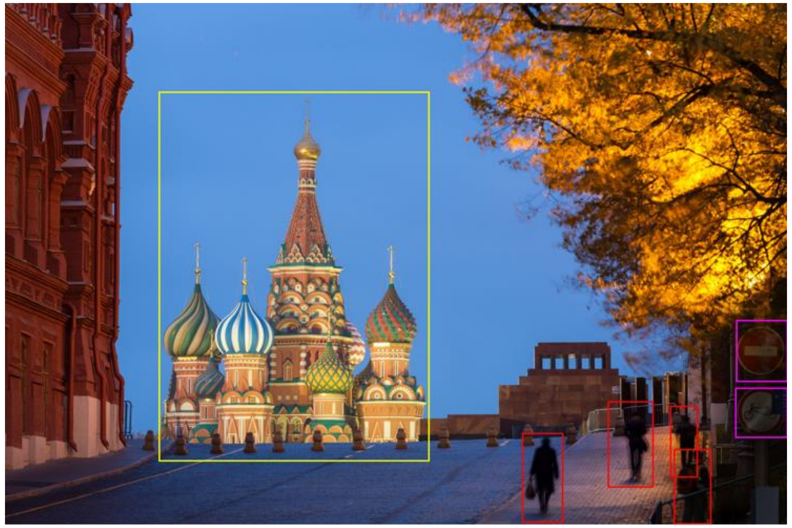
Content:

- Object detection
  - Two-stage detectors
  - One-stage detectors
- Segmentation



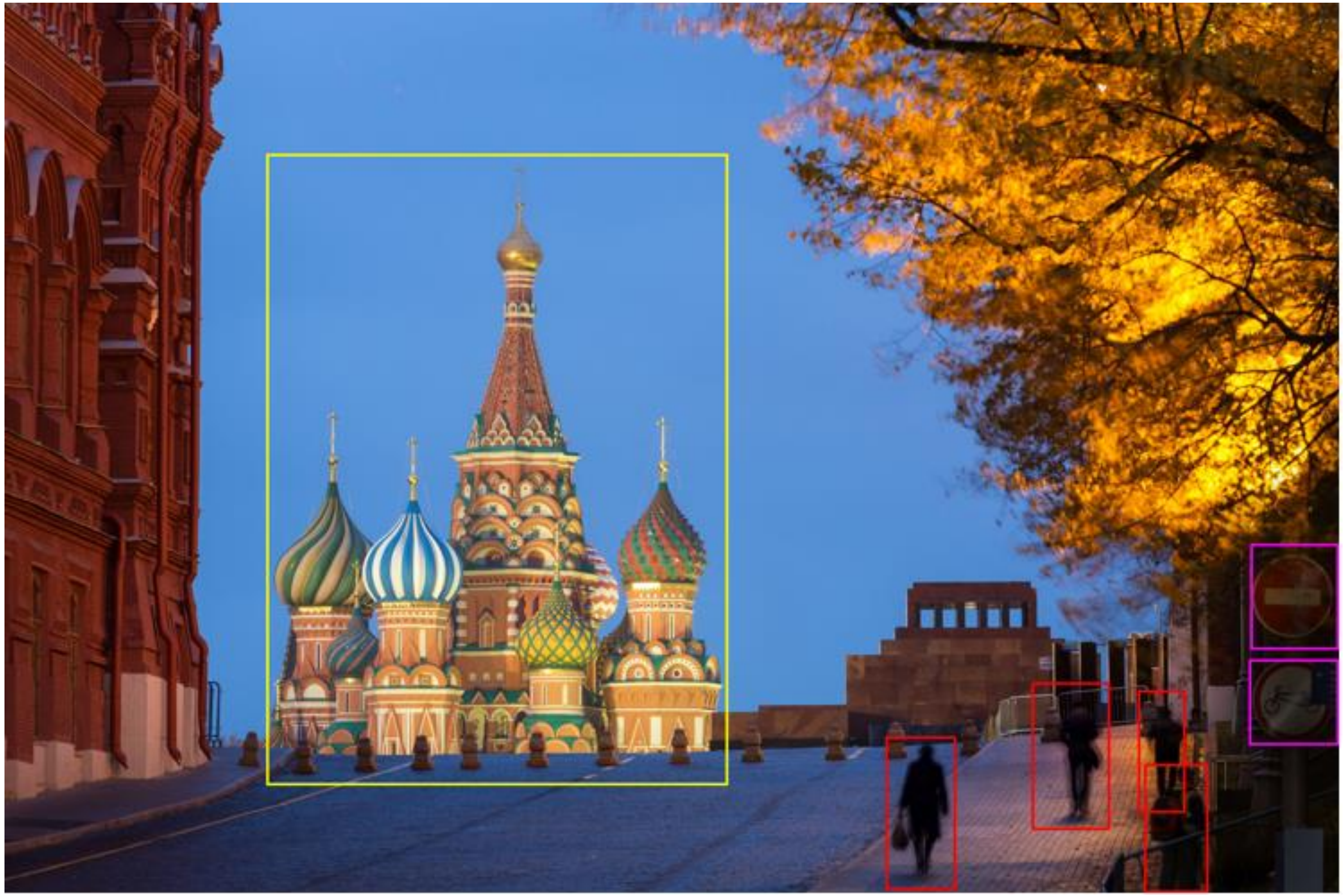
Applications of CNNs
Overview

Content:
- **Object detection**
  - **Two-stage detectors**
  - One-stage detectors
- Segmentation



Applications of CNNs

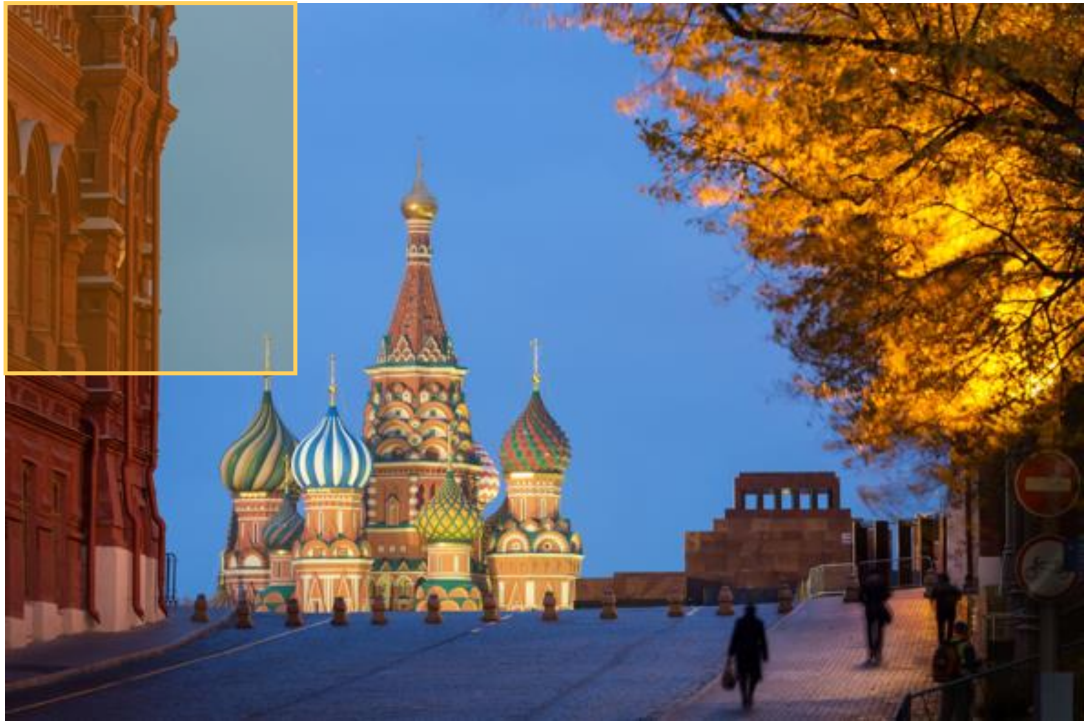
Object detection: example



Applications of CNNs

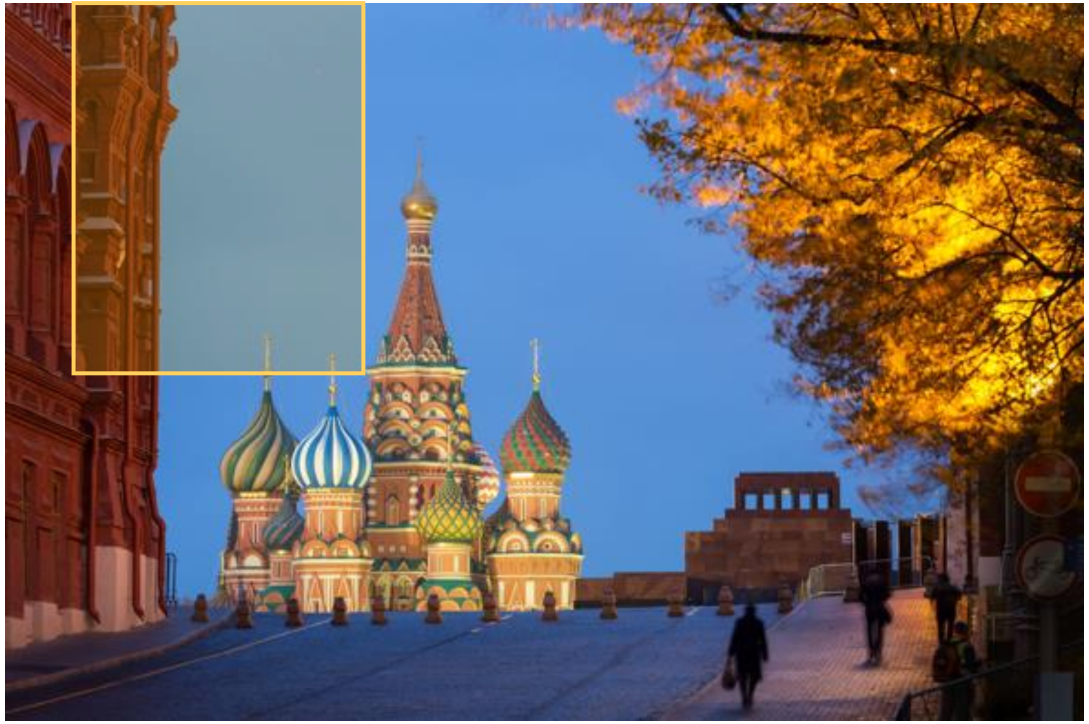
Object detection – Naïve approach

1. Specify sliding window size



Applications of CNNs
Object detection – Naïve approach

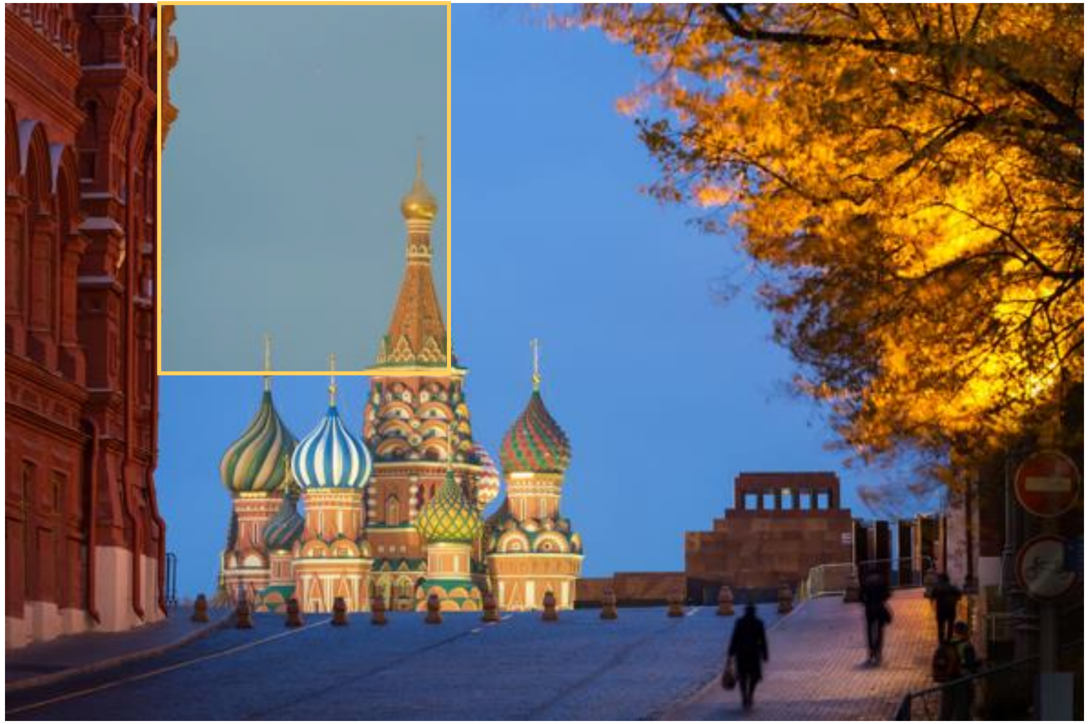
1. Specify sliding window size



Applications of CNNs

Object detection – Naïve approach

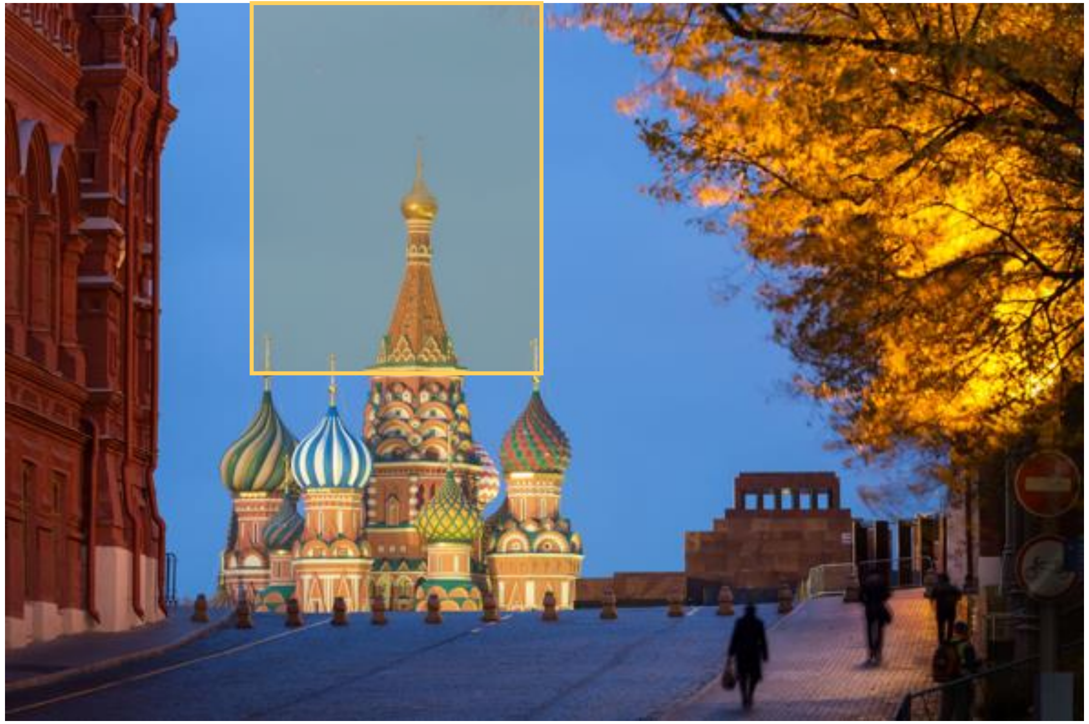
1. Specify sliding window size



Applications of CNNs

Object detection – Naïve approach

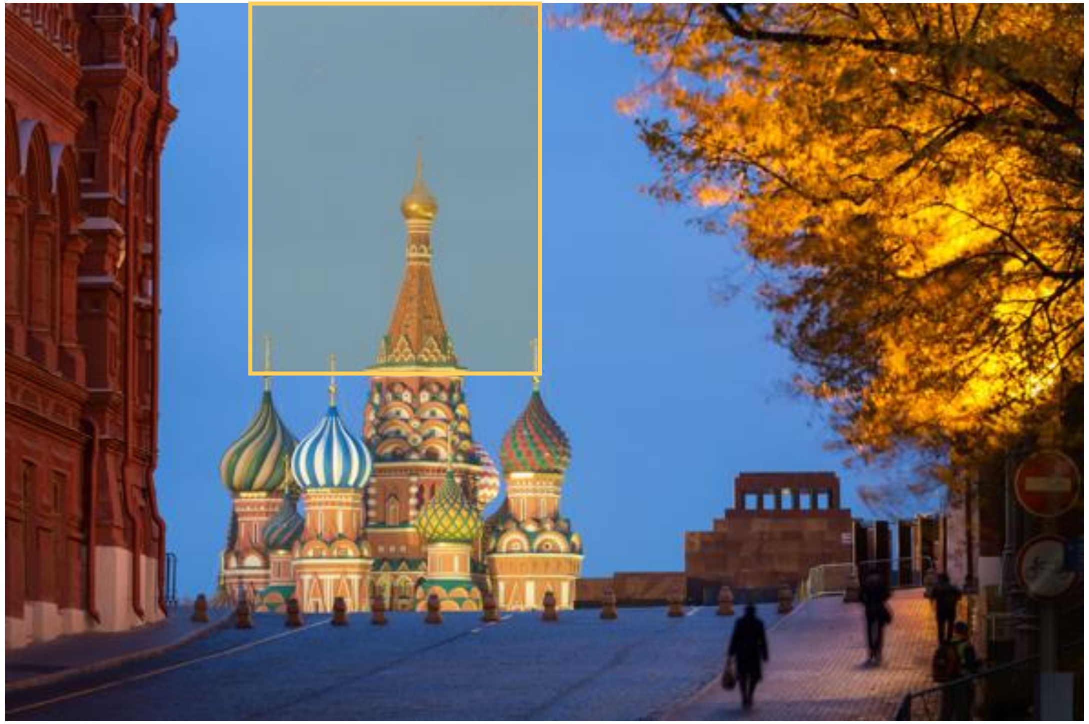
1. Specify sliding window size

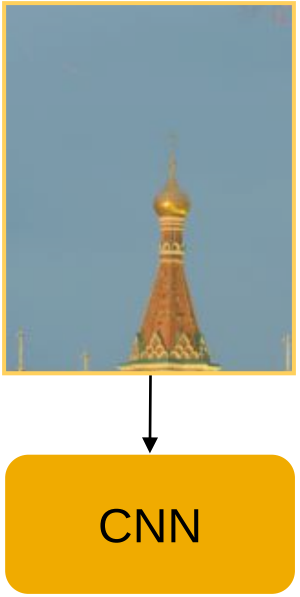


Applications of CNNs

Object detection – Naïve approach

1. Specify sliding window size
2. Run pre-trained object-classification CNN on each window (*softmax* output)

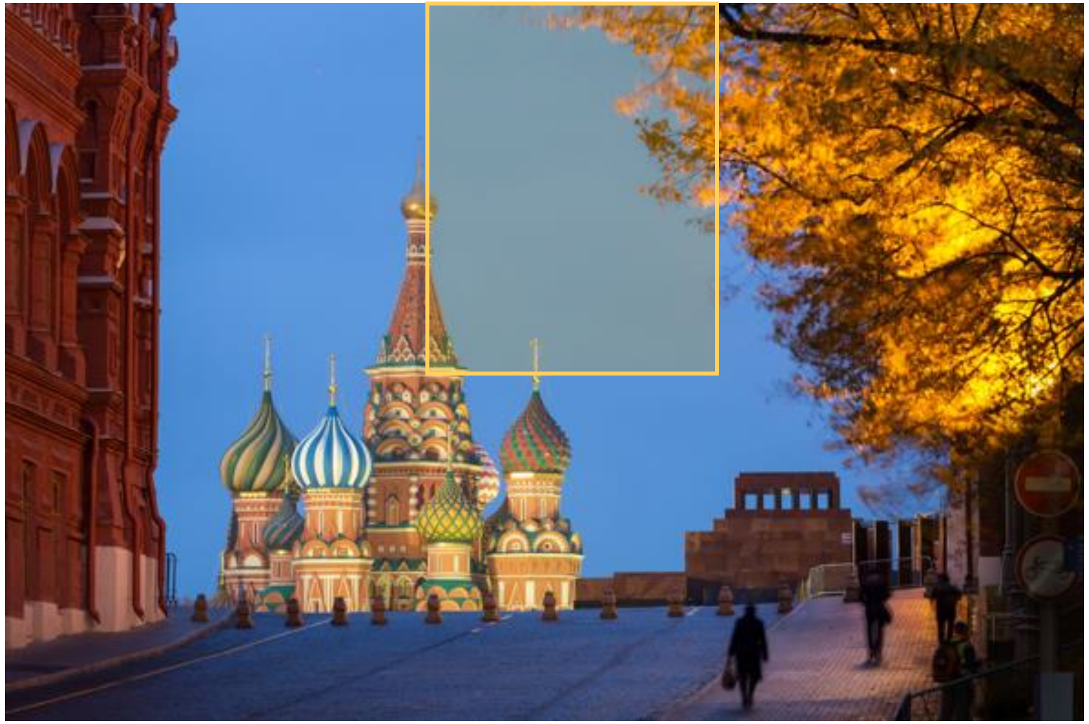


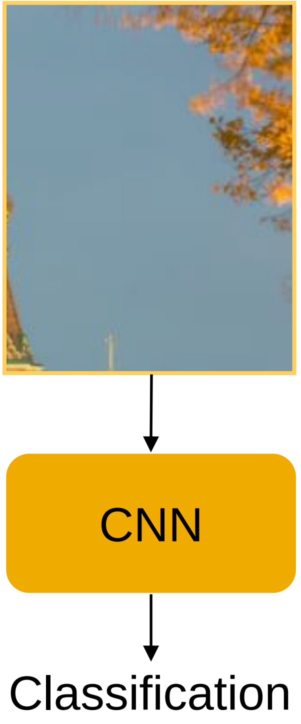


Applications of CNNs

Object detection – Naïve approach

1. Specify sliding window size
2. Run pre-trained object-classification CNN on each window (*softmax* output)
3. Associate current sliding window with object class if probability is sufficiently high (e.g. > 0.5)

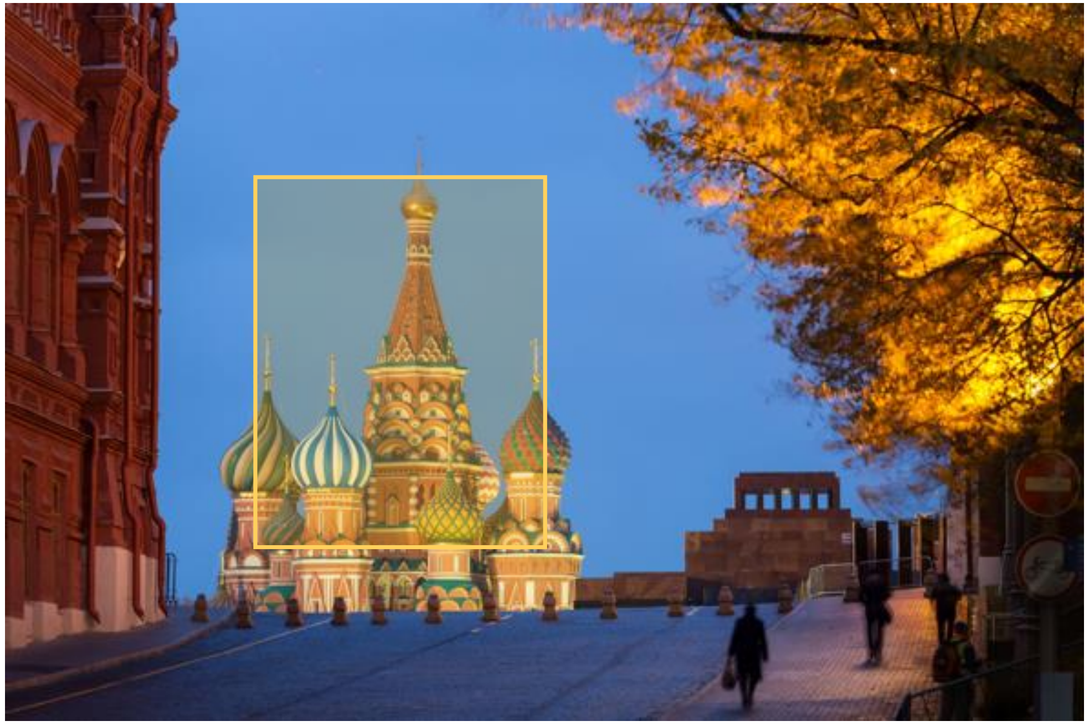




Applications of CNNs

Object detection – Naïve approach

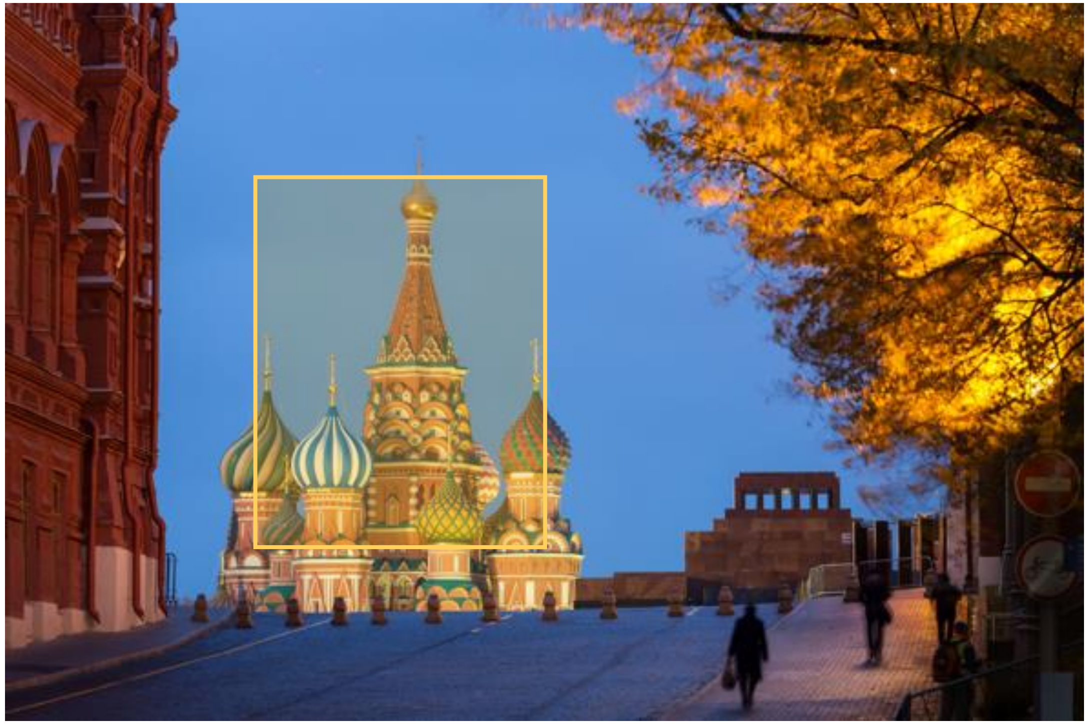
1. Specify sliding window size
2. Run pre-trained object-classification CNN on each window (*softmax* output)
3. Associate current sliding window with object class if probability is sufficiently high (e.g. > 0.5)



Applications of CNNs

Object detection – Naïve approach

1. Specify sliding window size
2. Run pre-trained object-classification CNN on each window (*softmax* output)
3. Associate current sliding window with object class if probability is sufficiently high (e.g. > 0.5)



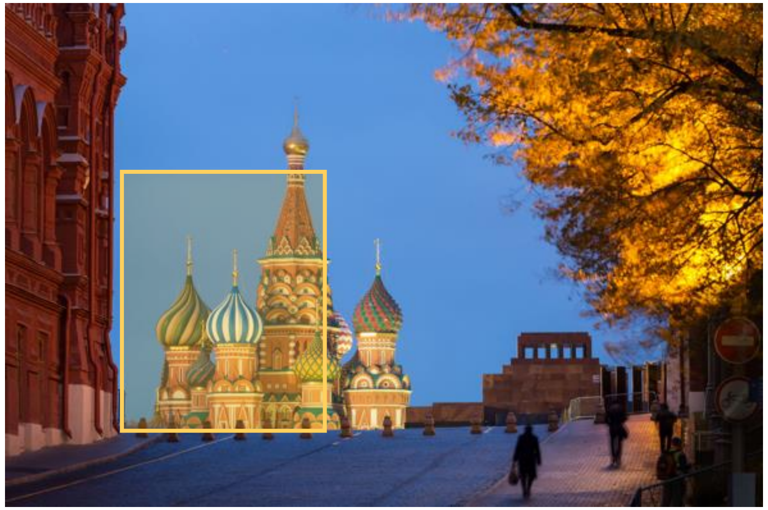
Problem:

Intractable even for small images

Applications of CNNs

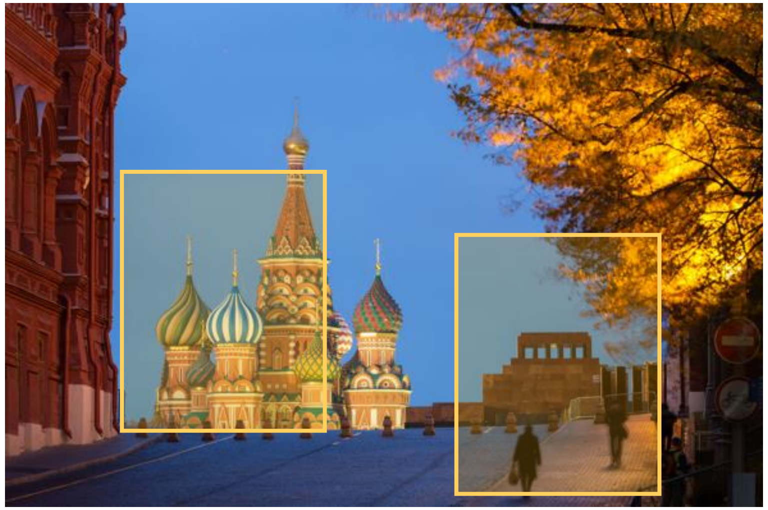
Two-stage detectors: key idea

- Use proposal algorithm (e.g. "selective search")



Applications of CNNs

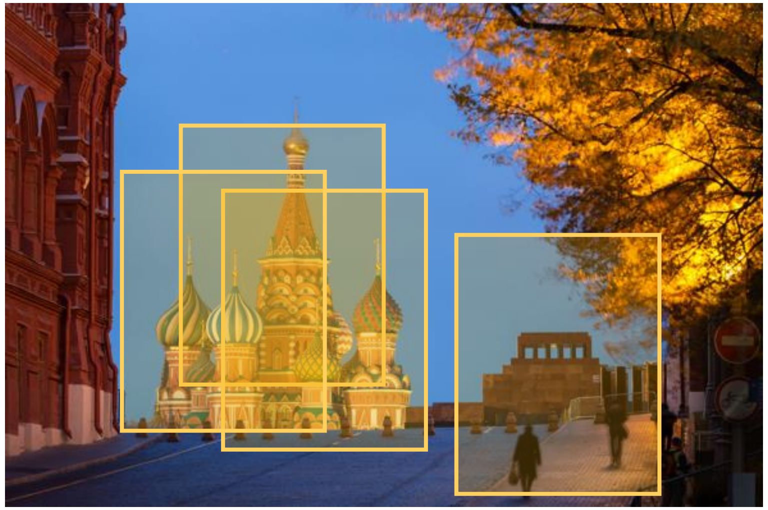
Two-stage detectors: key idea



- Use proposal algorithm (e.g. “selective search”)
- Boxes are proposed based on e.g.:
  – Texture
  – Color
  – Intensity

Applications of CNNs
Two-stage detectors: key idea

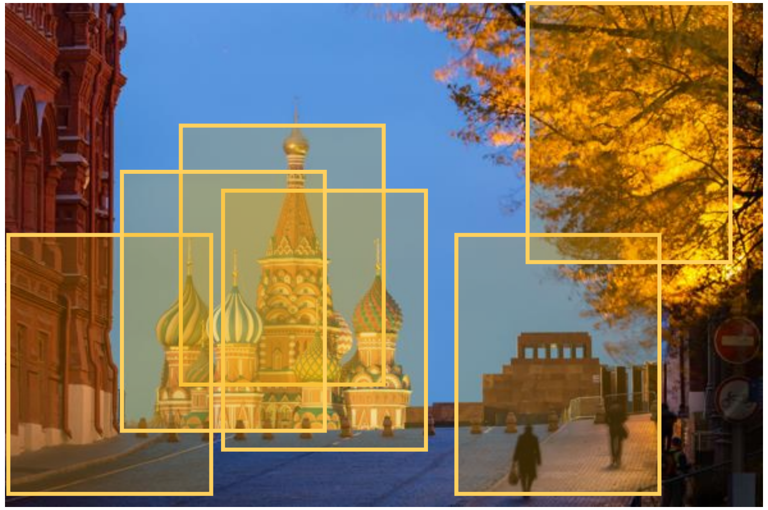
- Use proposal algorithm (e.g. “selective search”)
- Boxes are proposed based on e.g.:
  – Texture
  – Color
  – Intensity



Applications of CNNs

Two-stage detectors: key idea

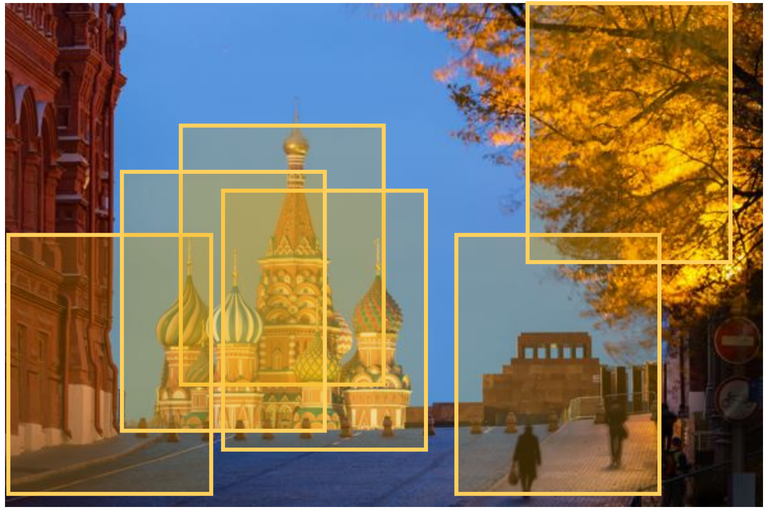
- Use proposal algorithm (e.g. “selective search”)
- Boxes are proposed based on e.g.:
  – Texture
  – Color
  – Intensity



Applications of CNNs

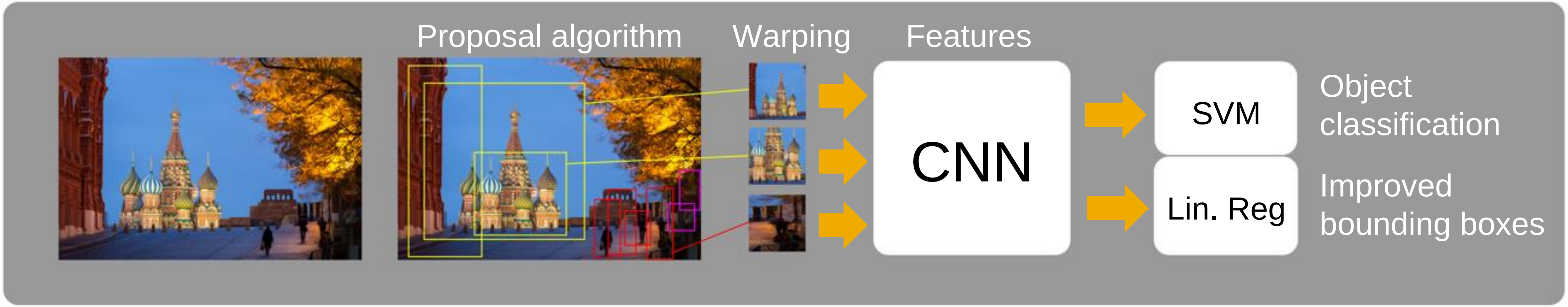
Two-stage detectors: key idea

- Use proposal algorithm (e.g. “selective search”)
- Boxes are proposed based on e.g.:
  - Texture
  - Color
  - Intensity
- Evaluate **<u>only</u>** on “proposals”



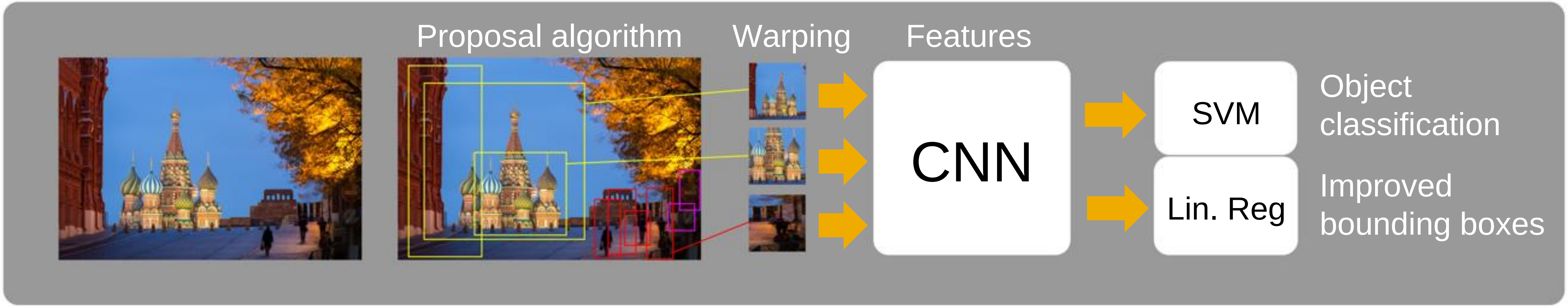
Applications of CNNs

Two-stage detectors: R-CNN [*Girshick, Donahue, Darrell, Malik, 2013]*



Applications of CNNs

Two-stage detectors: R-CNN [*Girshick, Donahue, Darrell, Malik, 2013]*

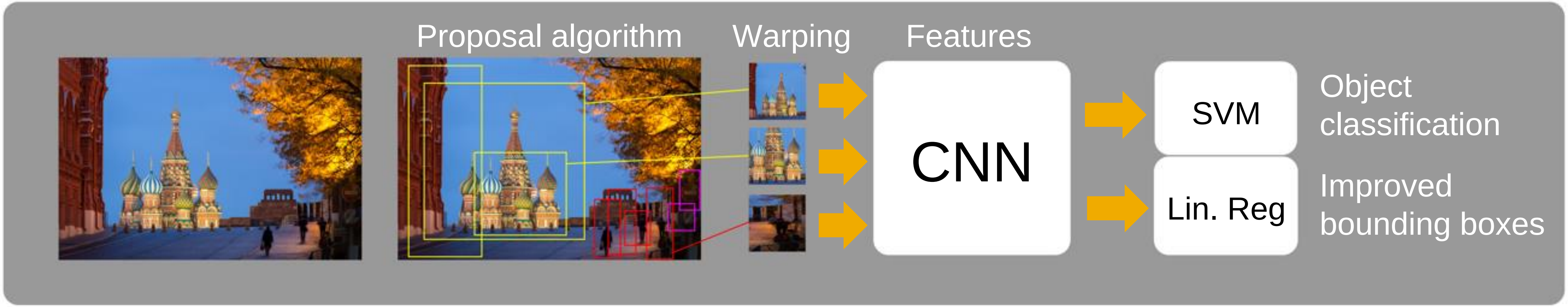


Advantages:

- Several orders of magnitude faster
- Allows improvement of bounding boxes

Applications of CNNs

Two-stage detectors: R-CNN [*Girshick, Donahue, Darrell, Malik, 2013]*



Advantages:

- Several orders of magnitude faster
- Allows improvement of bounding boxes

Disadvantages:

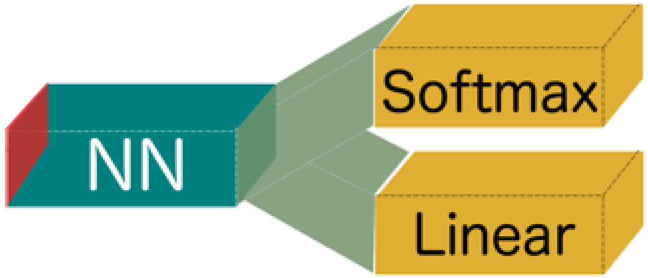
- CNN cannot adapt to changes in SVM/linear regression
  – *Implication:* Degrades power of model
- CNN often evaluates boxes that have large overlaps
  – *Implication:* Suboptimal performance

Applications of CNNs

Two-stage detectors: Fast R-CNN *[Girshick 2015]*

Key Idea:

- Softmax output replaces SVM
- Combine regression and classification into one neural net (end-to-end)



Fully connected

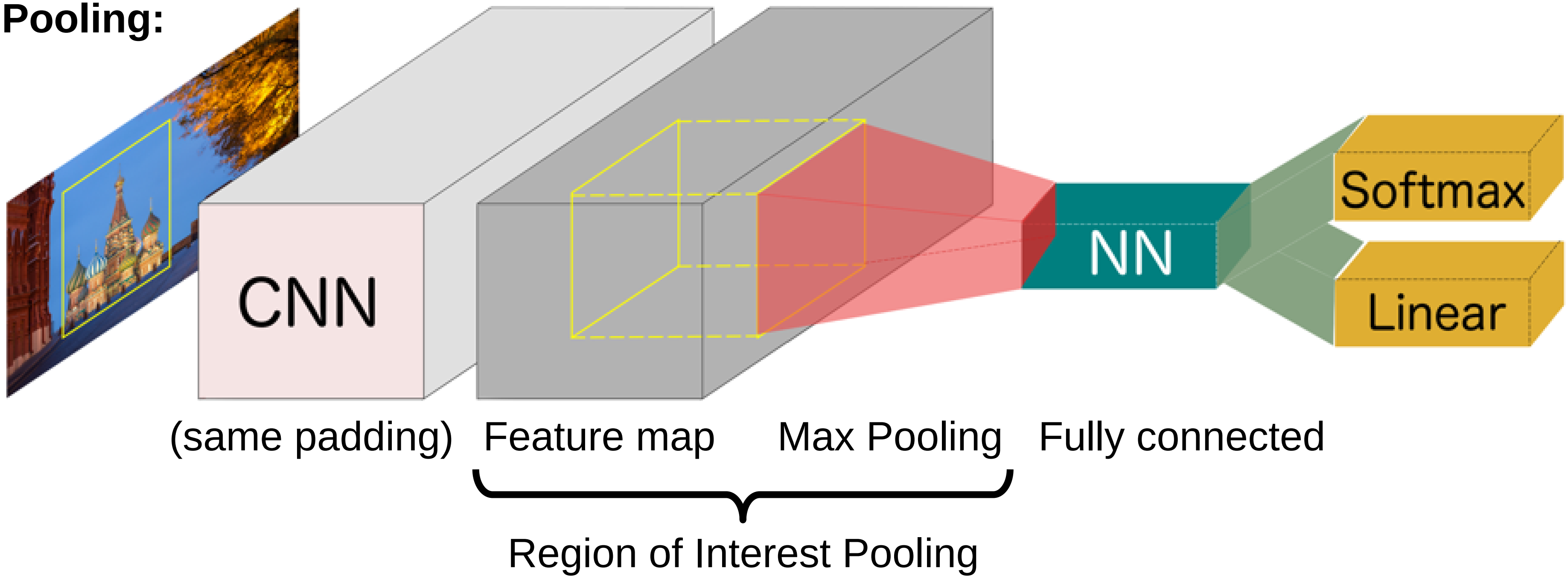
Applications of CNNs

Two-stage detectors: Fast R-CNN *[Girshick 2015]*

Key Idea:

- Softmax output replaces SVM
- Combine regression and classification into one neural net (end-to-end)
- Evaluate CNN only one time/image (share results among suggestions)

Region of Interest Pooling:



Applications of CNNs

Two-stage detectors: Faster R-CNN *[Ren, He, Girshick, Sun, 2015]*

Key Problem: Region proposal becomes bottleneck

Applications of CNNs

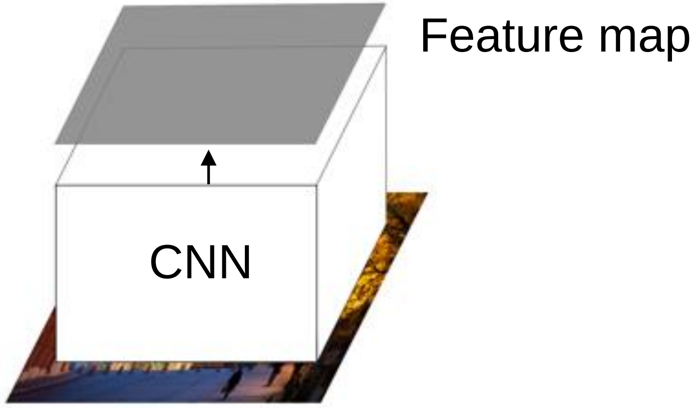
Two-stage detectors: Faster R-CNN *[Ren, He, Girshick, Sun, 2015]*

Key Problem: Region proposal becomes bottleneck
Solution: Add so-called “Region Proposal Network”

Applications of CNNs

Two-stage detectors: Faster R-CNN *[Ren, He, Girshick, Sun, 2015]*

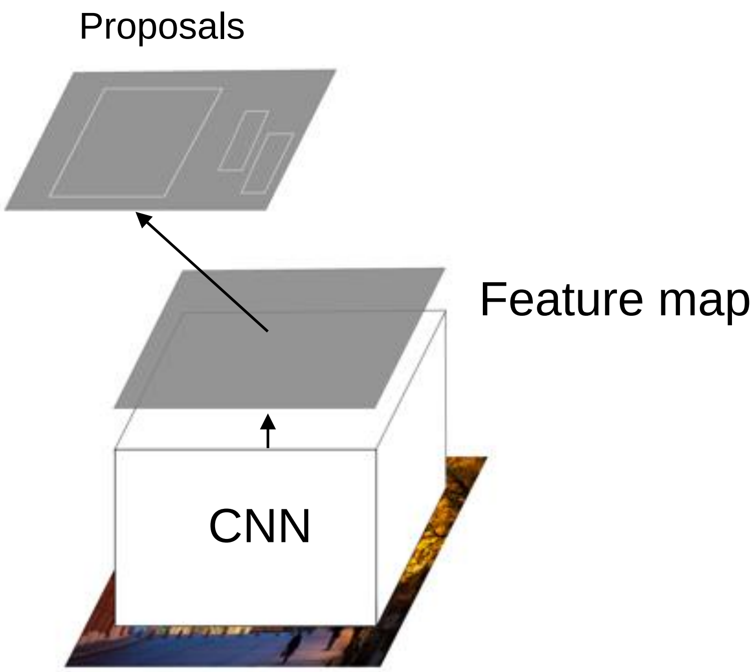
Key Problem: Region proposal becomes bottleneck
Solution: Add so-called "Region Proposal Network"



Applications of CNNs

Two-stage detectors: Faster R-CNN *[Ren, He, Girshick, Sun, 2015]*

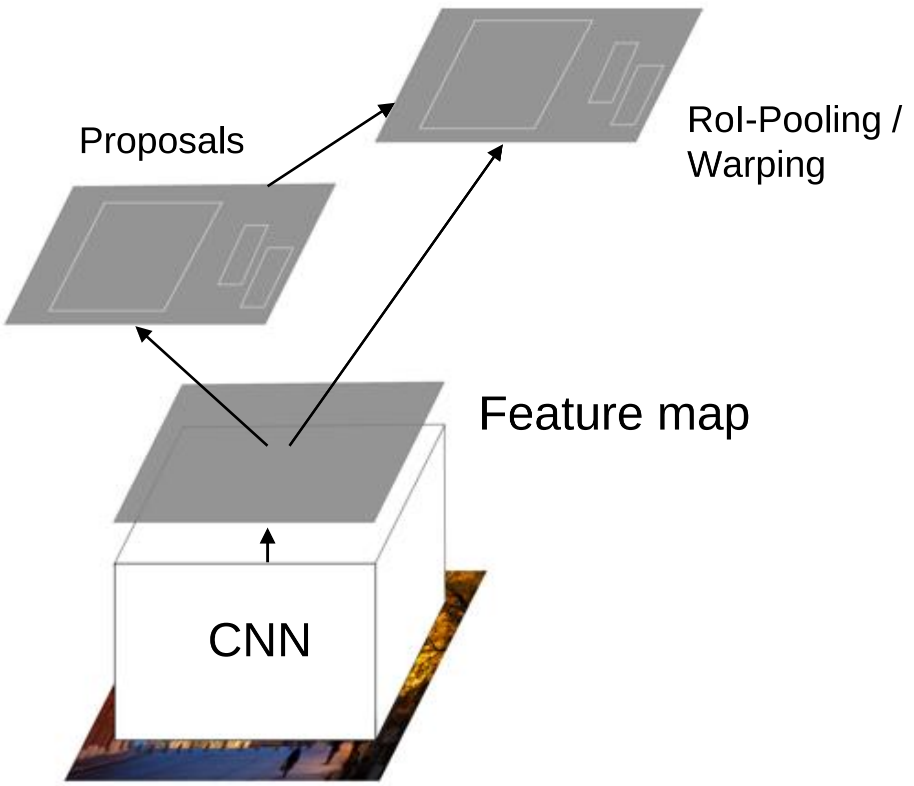
Key Problem: Region proposal becomes bottleneck
Solution: Add so-called “Region Proposal Network”



Applications of CNNs

Two-stage detectors: Faster R-CNN *[Ren, He, Girshick, Sun, 2015]*

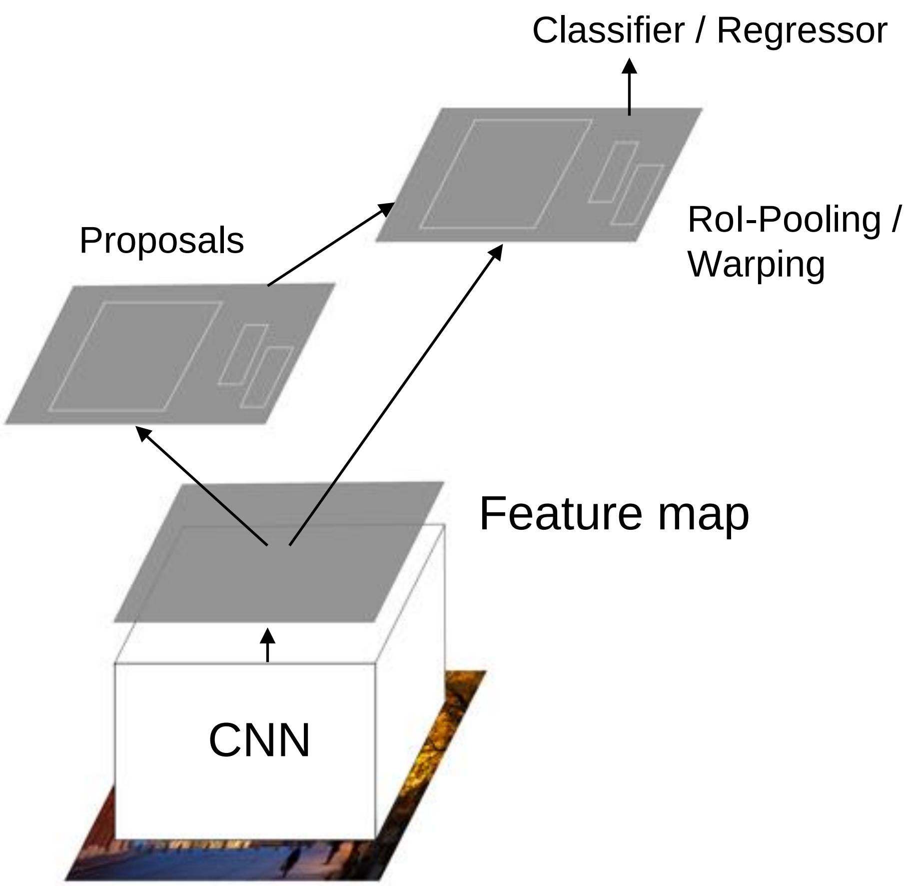
Key Problem: Region proposal becomes bottleneck
Solution: Add so-called "Region Proposal Network"



Applications of CNNs

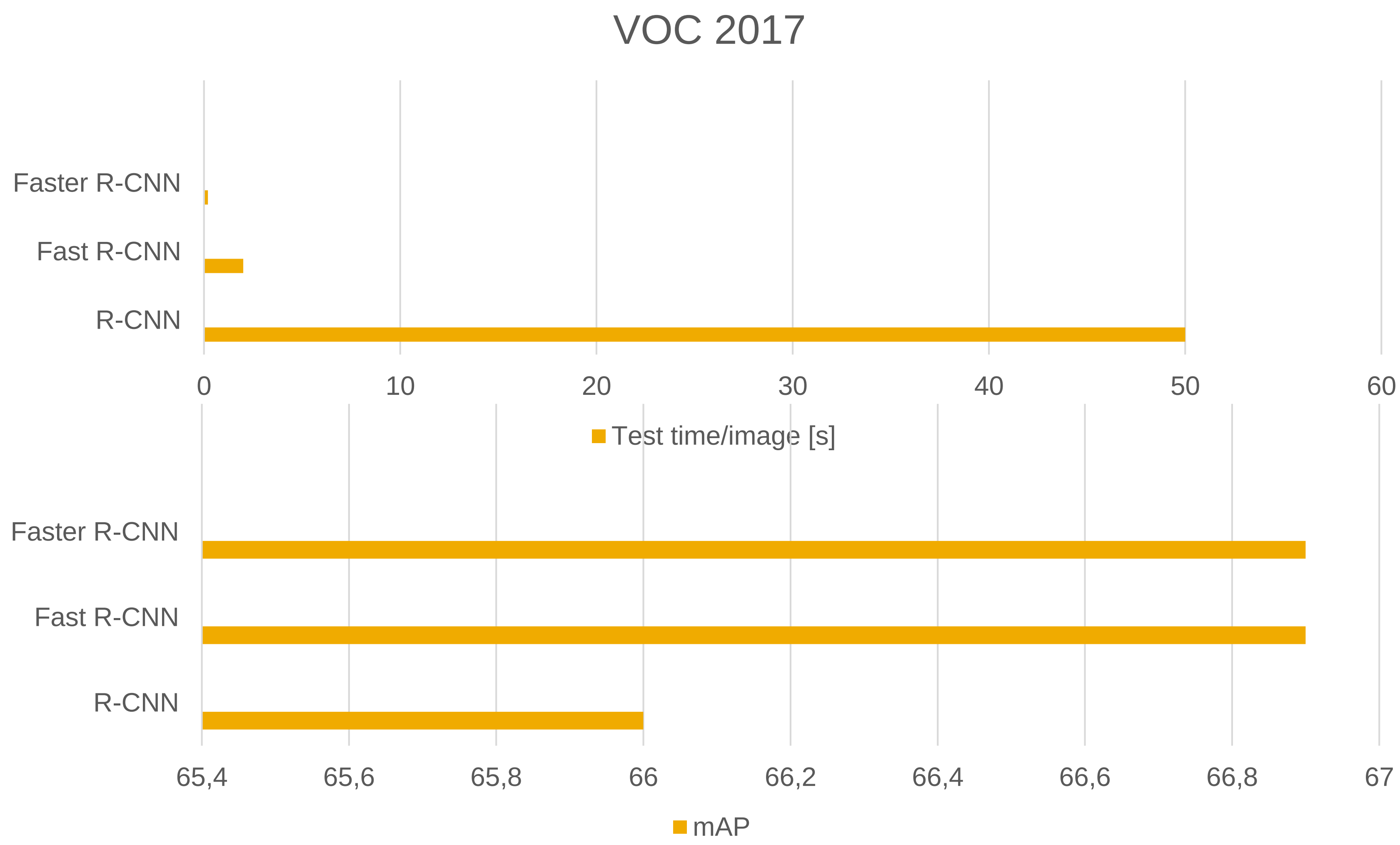
Two-stage detectors: Faster R-CNN *[Ren, He, Girshick, Sun, 2015]*

Key Problem: Region proposal becomes bottleneck
Solution: Add so-called “Region Proposal Network”



Applications of CNNs

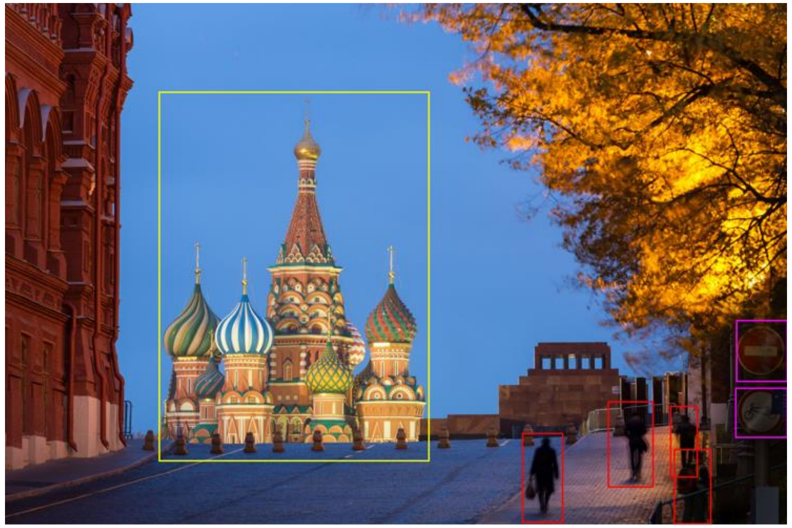
Two-stage detectors: comparison *[Li, Karpathy, Johnson]*



Applications of CNNs
Overview

Content:

- **Object detection**
  - Two-stage detectors
  - **One-stage detectors**
- Segmentation



Applications of CNNs

Object detection: one-stage detectors

How can we make this faster?

The R-CNN class detectors are accurate – but very slow!

Methodology:

- First stage: Generate region proposals
- Second stage: Classify & refine bbox proposals

Applications of CNNs

Object detection: one-stage detectors

How can we make this faster?

The R-CNN class detectors are accurate – but very slow!

Methodology:

- First stage: Generate region proposals
- Second stage: Classify & refine bbox proposals

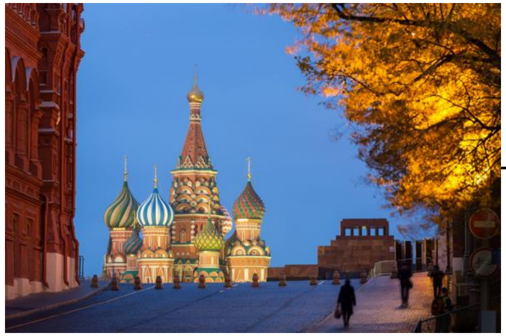
CHANGE METHODOLOGY

One-stage detectors:

- **Idea**: Take advantage of fully convolutional networks to compute bounding box regression and class probabilities in **one** pass
- **Implication**: Architecture is considerably simpler!

Applications of CNNs

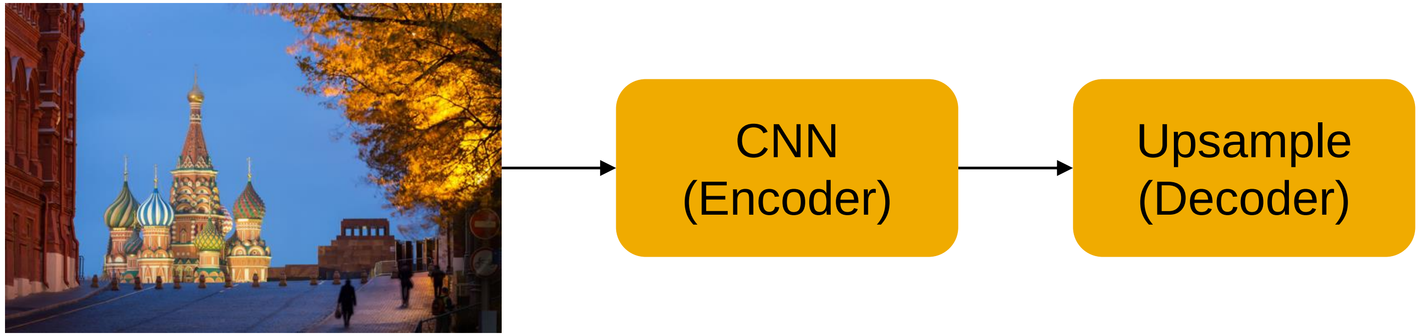
One-stage detectors e.g.: YOLO(9000) – You Only Look Once [*Redmon, Divvala, Girshick, Farhadi, 2015*]



CNN
(Encoder)

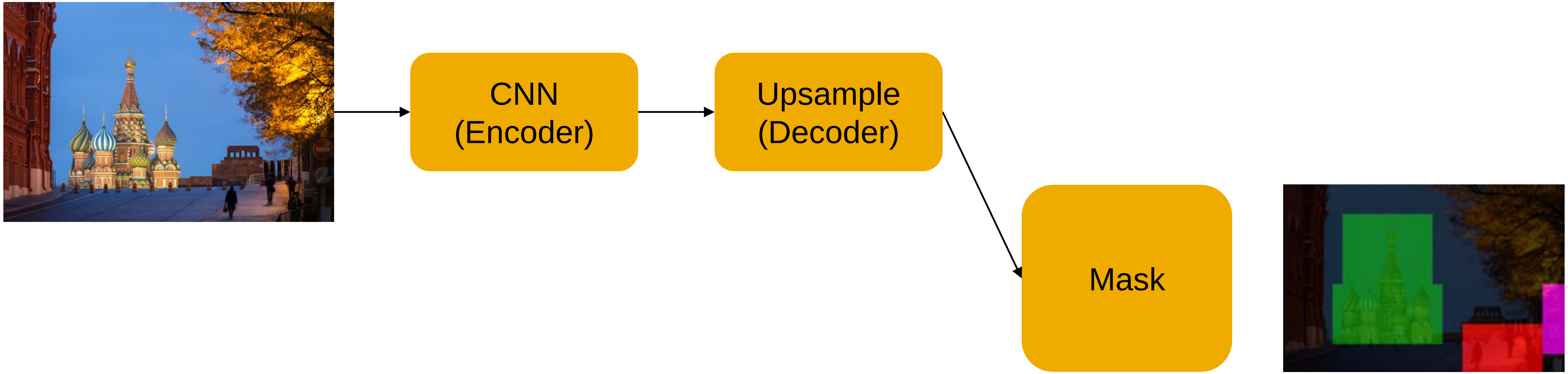
Applications of CNNs

One-stage detectors e.g.: YOLO(9000) – You Only Look Once [*Redmon, Divvala, Girshick, Farhadi, 2015*]



Applications of CNNs

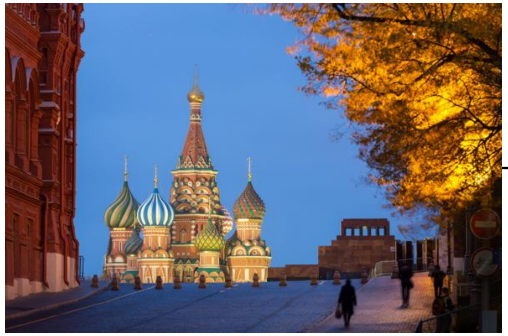
One-stage detectors e.g.: YOLO(9000) – You Only Look Once [*Redmon, Divvala, Girshick, Farhadi, 2015*]



Each pixel predicts
a class

Applications of CNNs

One-stage detectors e.g.: YOLO(9000) – You Only Look Once [*Redmon, Divvala, Girshick, Farhadi, 2015*]



CNN (Encoder) → Upsample (Decoder) → Box coordinates / Mask

Box coordinates

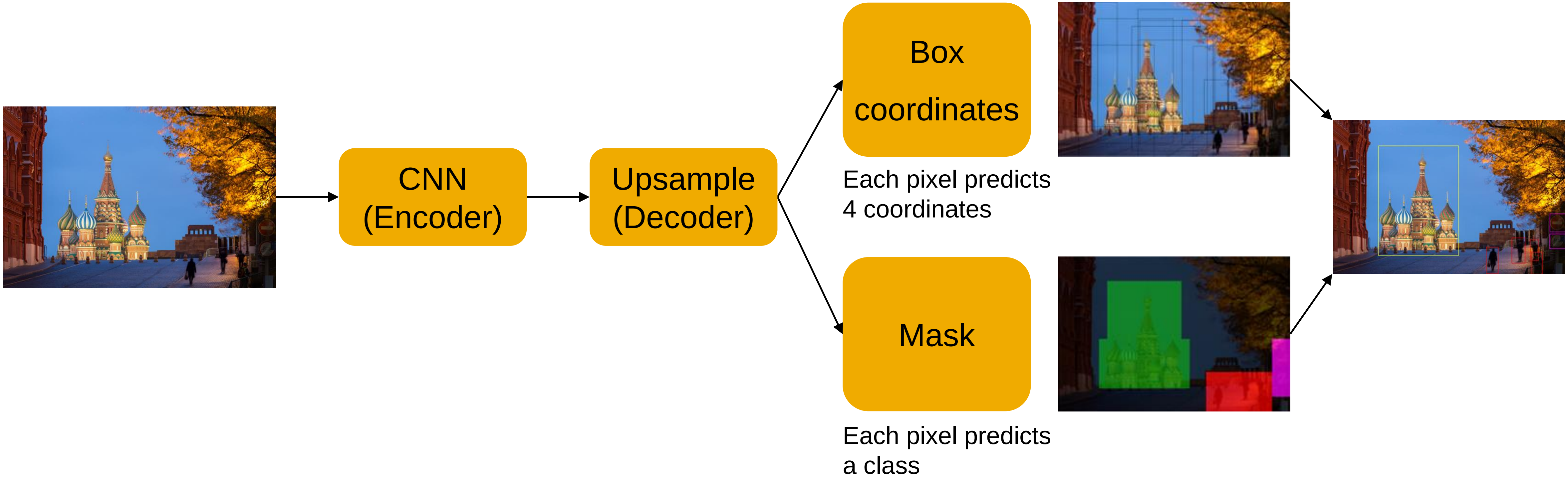
Each pixel predicts 4 coordinates

Mask

Each pixel predicts a class

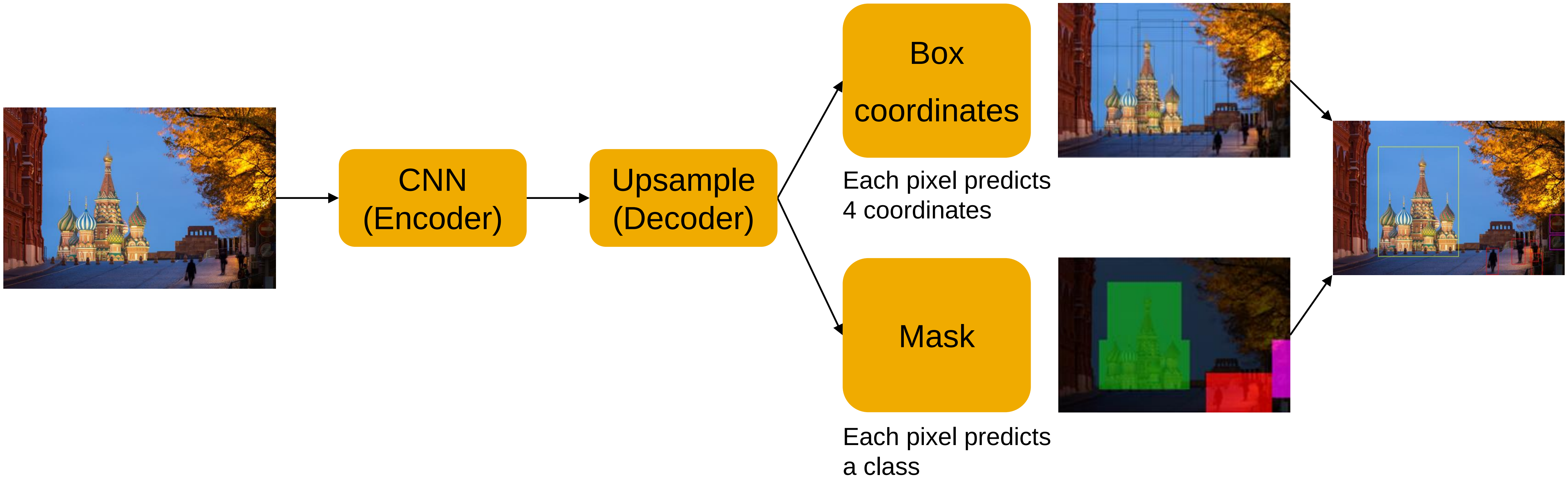
Applications of CNNs

One-stage detectors e.g.: YOLO(9000) – You Only Look Once [*Redmon, Divvala, Girshick, Farhadi, 2015*]



Applications of CNNs

One-stage detectors e.g.: YOLO(9000) – You Only Look Once [*Redmon, Divvala, Girshick, Farhadi, 2015*]



Also see: Single Shot Multibox Detector (SSD)

Applications of CNNs

One-stage detectors: comparison

Problems: One-stage detectors are less accurate than two-stage detectors:

Applications of CNNs

One-stage detectors: comparison

Problems: One-stage detectors are less accurate than two-stage detectors:

- Number of bounding box predictions is considerably larger (c.f. R-CNN’s 2k proposals vs. 100k in most one-stage detectors)

Applications of CNNs

One-stage detectors: comparison

Problems: One-stage detectors are less accurate than two-stage detectors:

- Number of bounding box predictions is considerably larger (c.f. R-CNN’s 2k proposals vs. 100k in most one-stage detectors)
- This leads to more class imbalance since most of the bounding boxes **belong to background**

Applications of CNNs

One-stage detectors: RetinaNet *[Lin, Goyal, Girshick, He, Dollar, 2017]*

Problems: One-stage detectors are less accurate than two-stage detectors:

- Number of bounding box predictions is considerably larger (c.f. R-CNN’s 2k proposals vs. 100k in most one-stage detectors)
- This leads to more class imbalance since most of the bounding boxes **belong to background**

Remedy: Introduce so-called *Focal Loss*:

Applications of CNNs

One-stage detectors: RetinaNet *[Lin, Goyal, Girshick, He, Dollar, 2017]*

Problems: One-stage detectors are less accurate than two-stage detectors:

- Number of bounding box predictions is considerably larger (c.f. R-CNN’s 2k proposals vs. 100k in most one-stage detectors)
- This leads to more class imbalance since most of the bounding boxes **belong to background**

Remedy: Introduce so-called *Focal Loss*:

- Down-weight contribution of examples that are particularly easy to learn (i.e. background patches)

Applications of CNNs

One-stage detectors: RetinaNet *[Lin, Goyal, Girshick, He, Dollar, 2017]*

Problems: One-stage detectors are less accurate than two-stage detectors:

- Number of bounding box predictions is considerably larger (c.f. R-CNN's 2k proposals vs. 100k in most one-stage detectors)
- This leads to more class imbalance since most of the bounding boxes **belong to background**

Remedy: Introduce so-called *Focal Loss*:

- Down-weight contribution of examples that are particularly easy to learn (i.e. background patches)

- $$\mathrm{L}(p, y) = \begin{cases} -\log p\,, & y = +1 \\ -\log(1-p)\,, & y = -1 \end{cases}$$

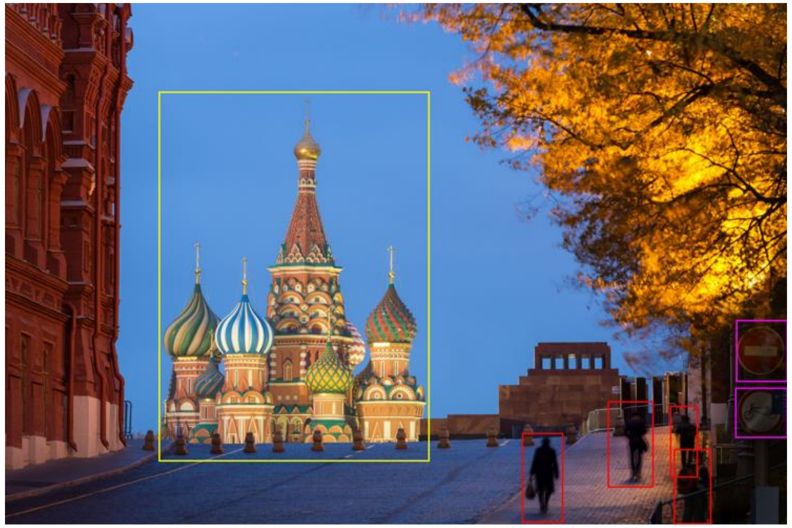
- $$\mathrm{F}(p, y) = \begin{cases} -(1-p)^{\gamma}\log p\,, & y = +1 \\ -p^{\gamma}\log(1-p)\,, & y = -1 \end{cases}$$

For more details: RetinaNet – https://arxiv.org/abs/1708.02002

Applications of CNNs
Overview

Content:

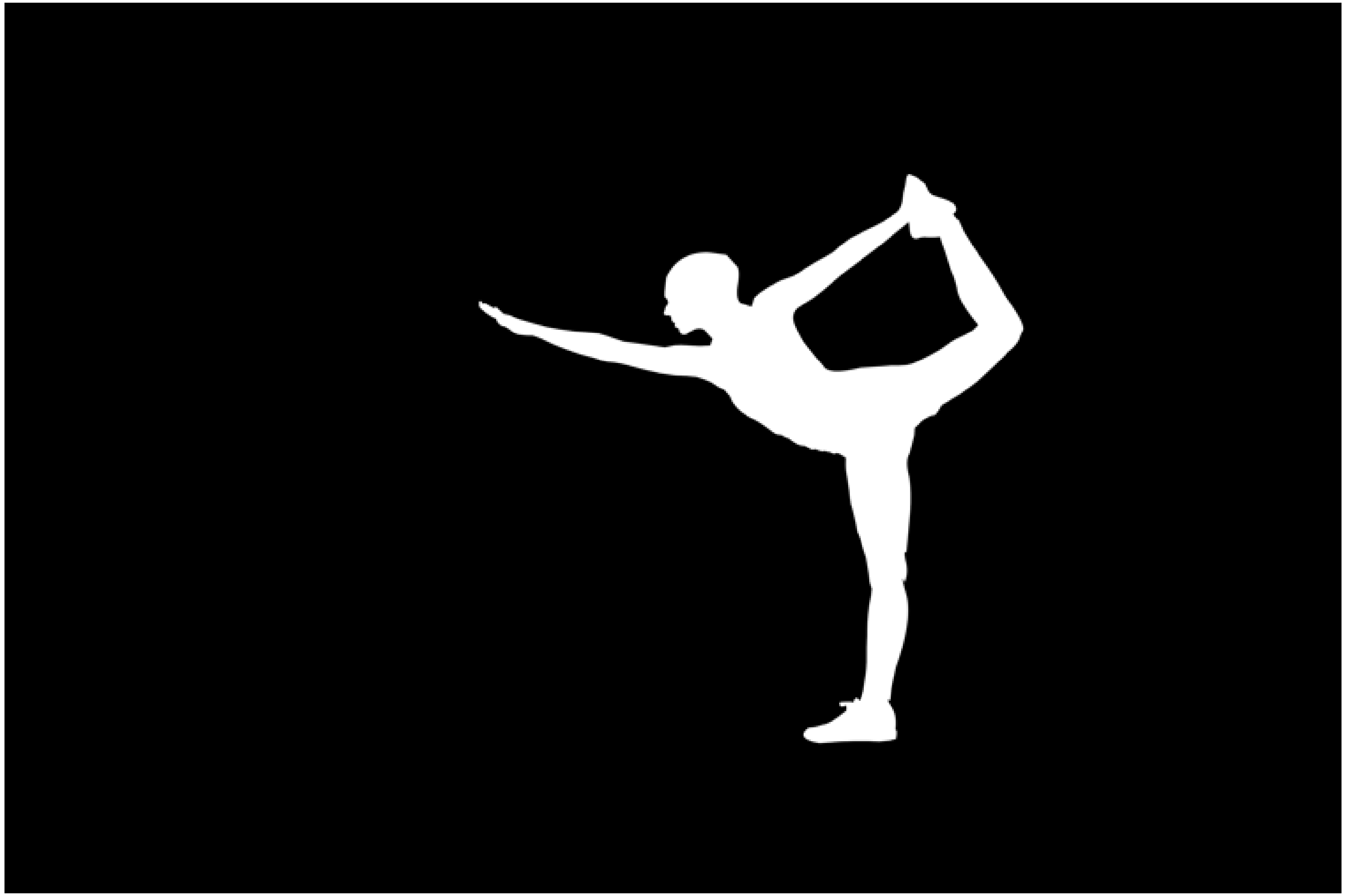
- Object detection
  - Two-stage detectors
  - One-stage detectors
- **Segmentation**



Applications of CNNs

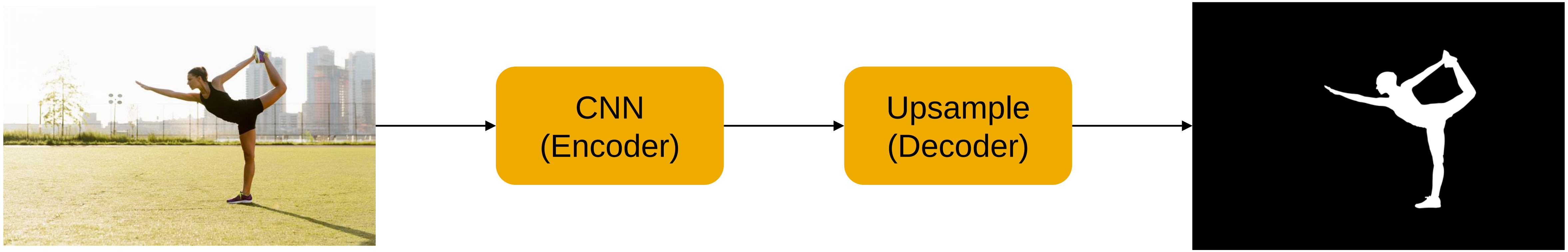
Image segmentation: example

Applications of CNNs
Image segmentation: example



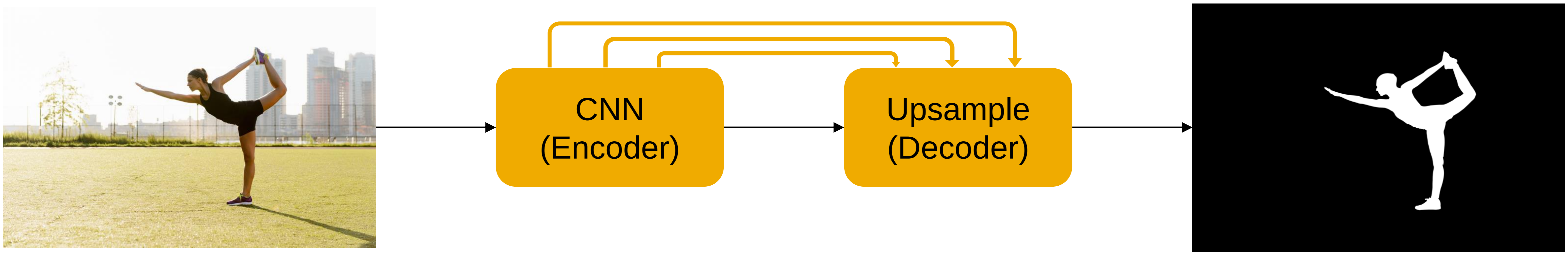
Applications of CNNs

Image segmentation: generic architecture



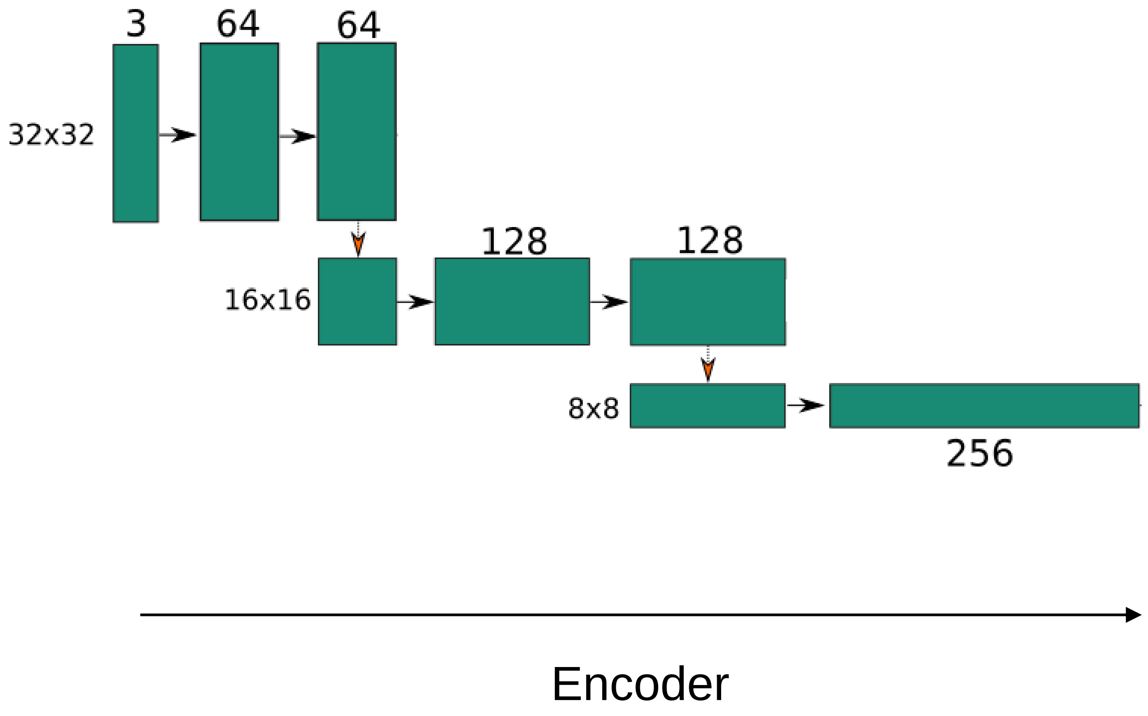
Applications of CNNs

Image segmentation: generic architecture



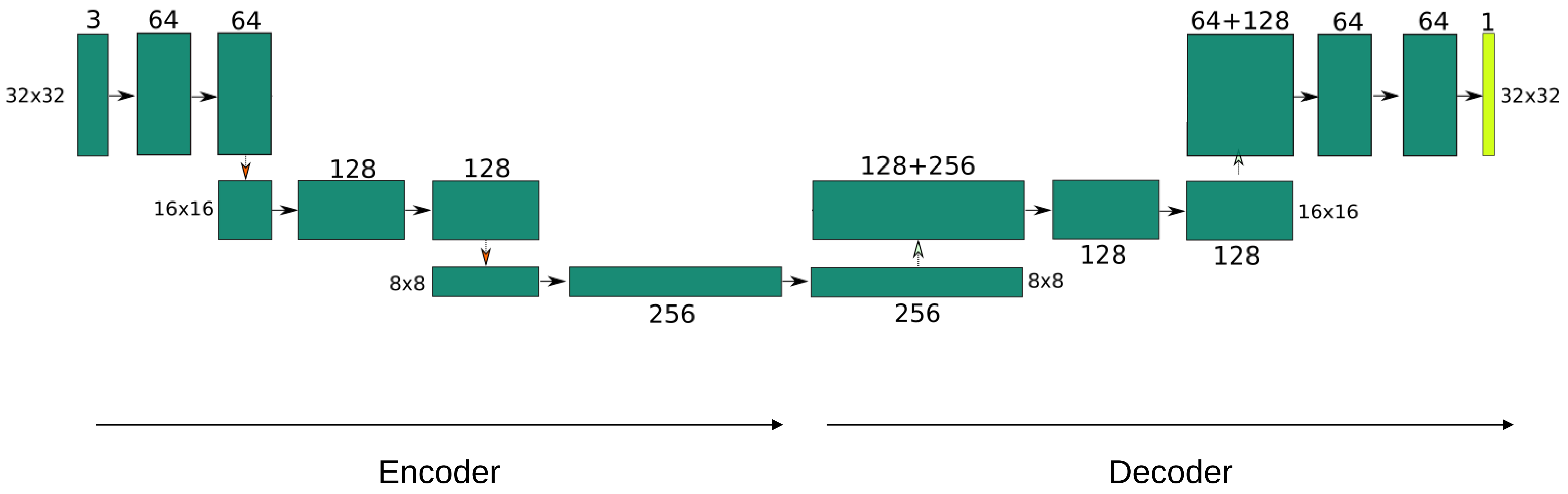
Applications of CNNs

Image segmentation e.g. U-Net architecture



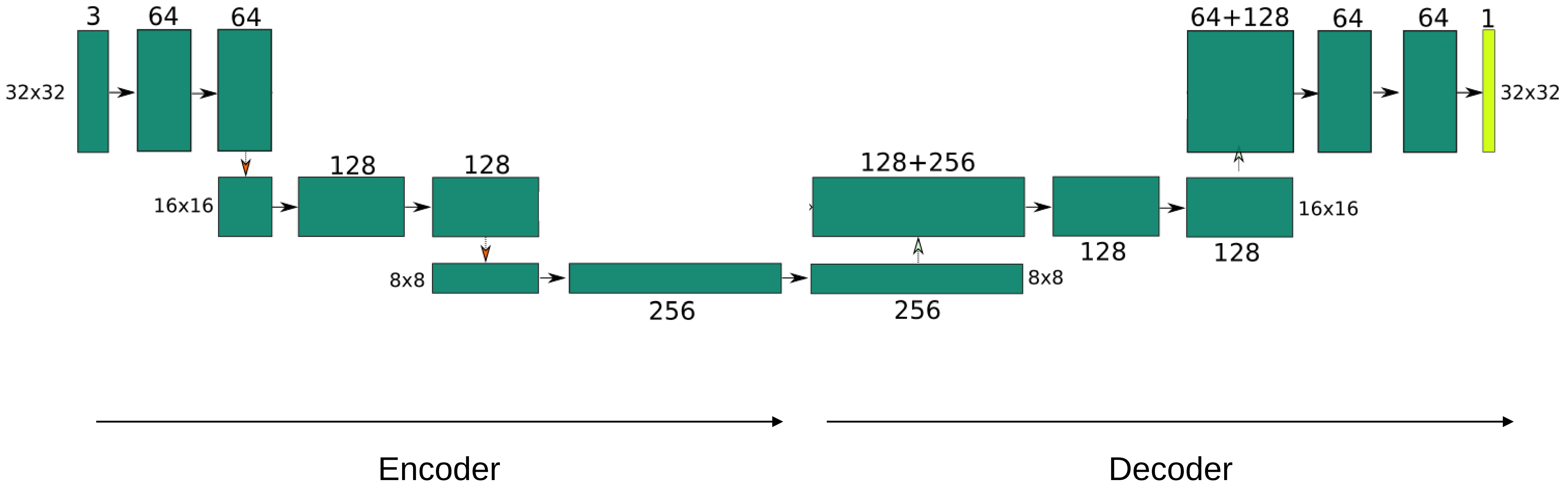
Applications of CNNs

Image segmentation e.g. U-Net architecture



Applications of CNNs

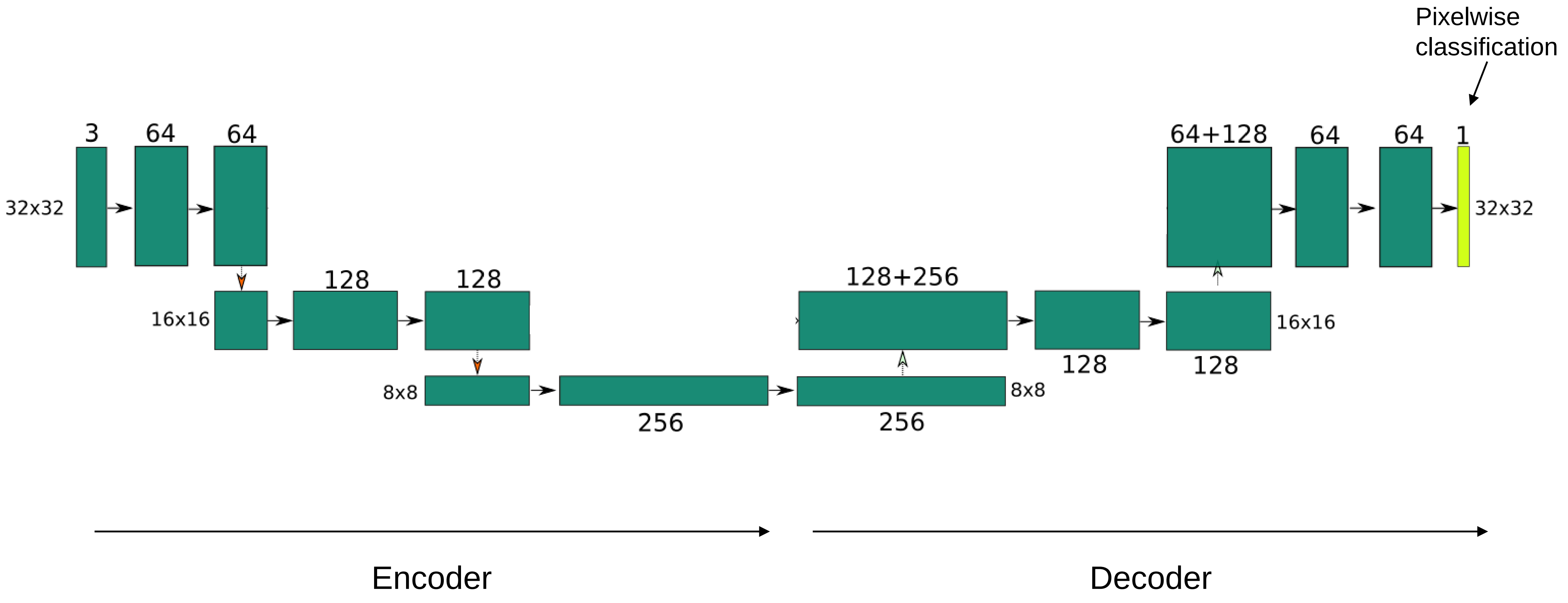
Image segmentation e.g. U-Net architecture



Upsampling: Transposed Convolutions (see Unit 3)

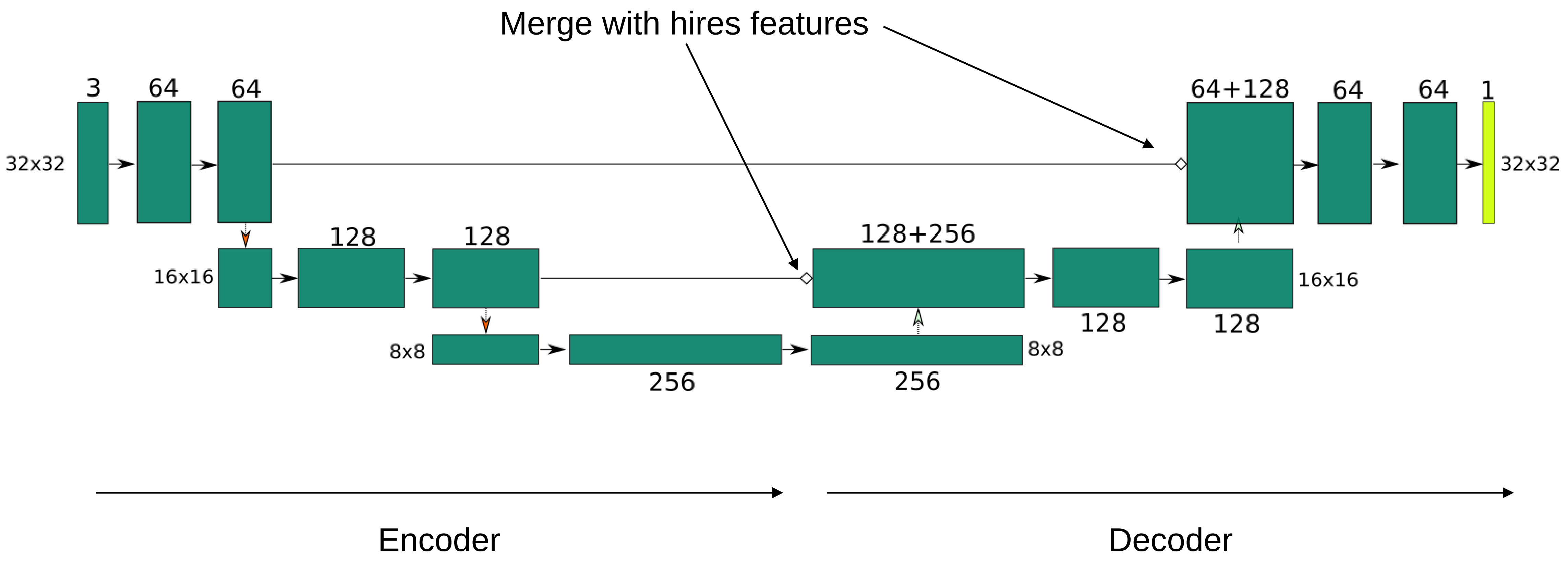
Applications of CNNs

Image segmentation e.g. U-Net architecture



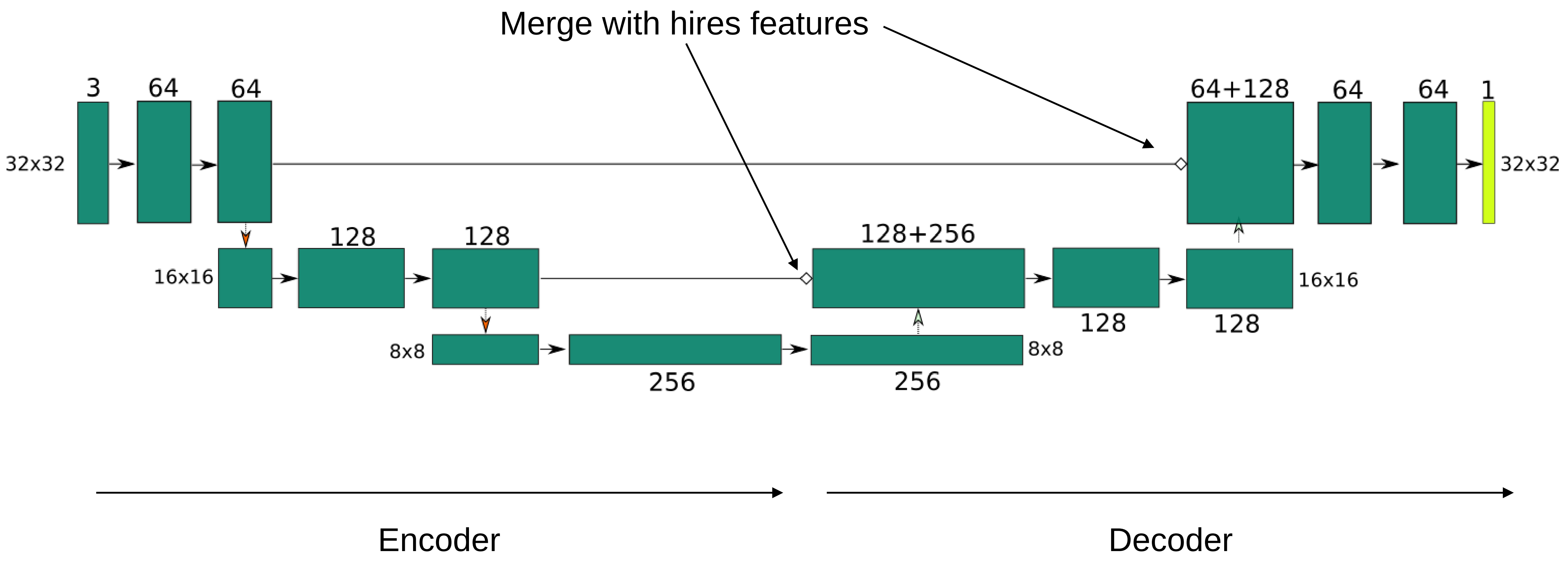
Applications of CNNs

Image segmentation e.g. U-Net architecture



Applications of CNNs

Image segmentation e.g. U-Net architecture



Also see: e.g. SegNet
e.g. Pyramid Scene Parsing Network

Applications of CNNs

Image segmentation: challenges

- **Efficiency / Accuracy**
  - Encoder downsampling (low resolution) versus dense prediction

Applications of CNNs

Image segmentation: challenges

- **Efficiency / Accuracy**
  – Encoder downsampling (low resolution) versus dense prediction
    → Make use of dilated convolutions
    → Feature fusion

Applications of CNNs

Image segmentation: challenges

- **Efficiency / Accuracy**
  - Encoder downsampling (low resolution) versus dense prediction
    → Make use of dilated convolutions
    → Feature fusion
- **Data availability**
  - Ground truth: Hand-segmented pictures (very costly to acquire!)
  - → Data augmentation is essential!

Applications of CNNs

Image segmentation: data augmentation

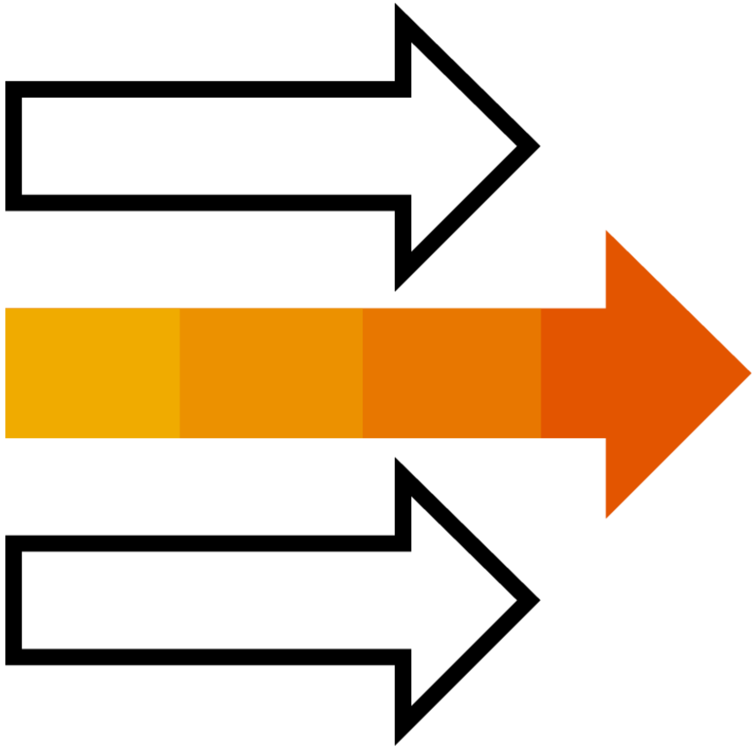
- Data augmentation is indispensable when there is little training data
- Use randomized “on-the-fly” modifications

Applications of CNNs

Coming up next

Industry Applications of Deep Learning

- Machine learning in customer service
- Machine learning in banking
- Medical image segmentation



Thank you.

Contact information:

open@sap.com