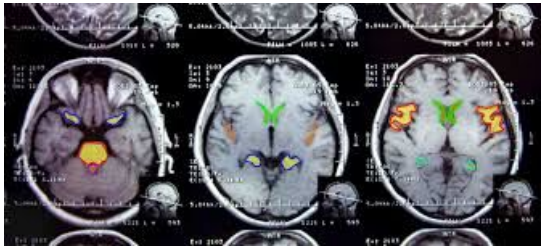
2019 Summer School on Human Language Technology

-- Computer Vision --

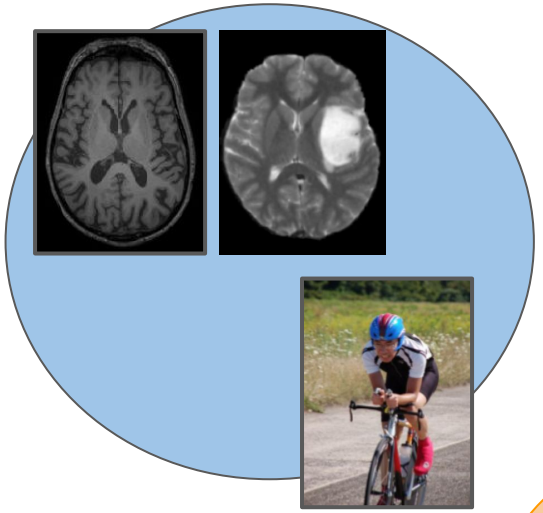


Jose Dolz
ETS Montreal

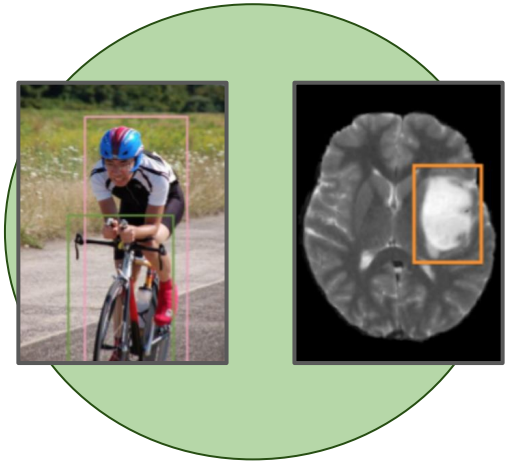
June, 12th, 2019

Introduction

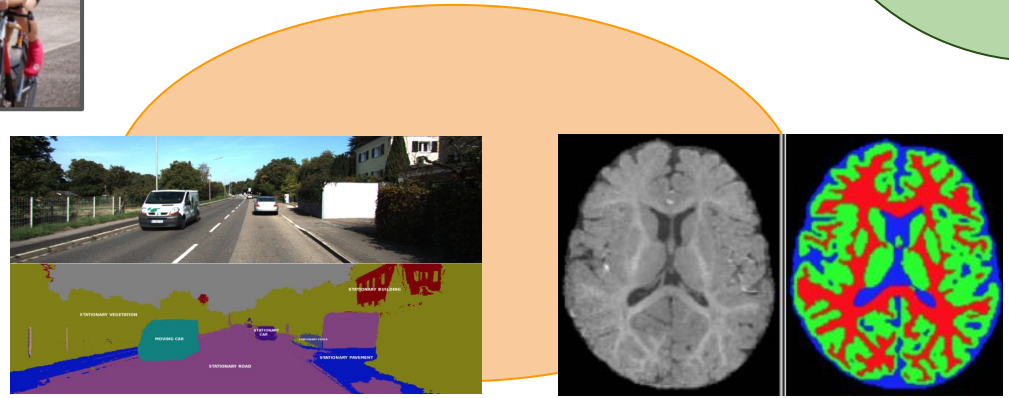
Classification



Detection



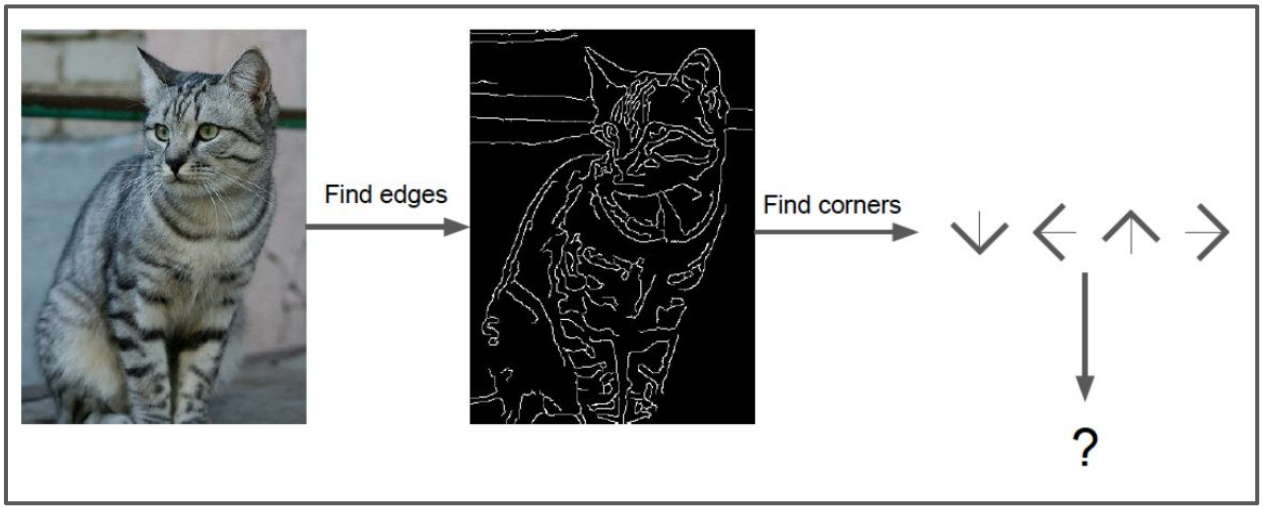
Segmentation



Introduction

Visual recognition until 2012

Hand-crafting features



Images from: Fei-Fei Li, Andrej Karpathy and Justin Johnson 2016, cs231n.

Biological inspiration

CNN architecture

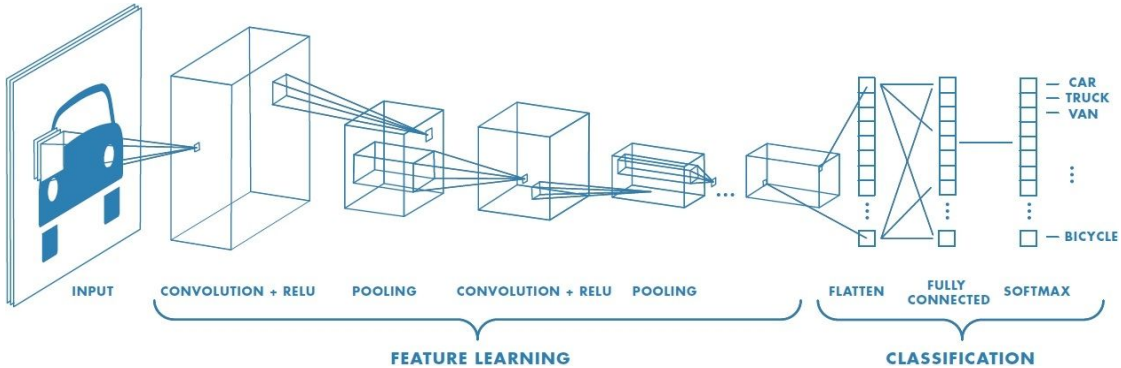


Image: https://www.mathworks.com

Human visual cortex

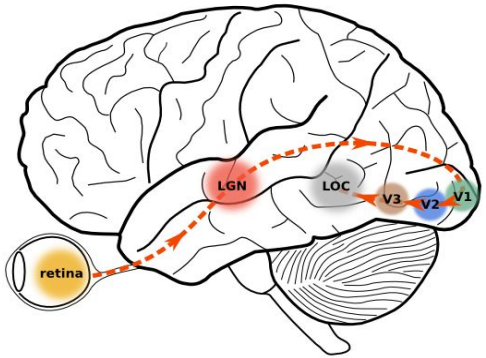


Image: https://neuwritesd.files.wordpress.com/2015/10/visual_stream_small.png

Biological inspiration

CNN architecture

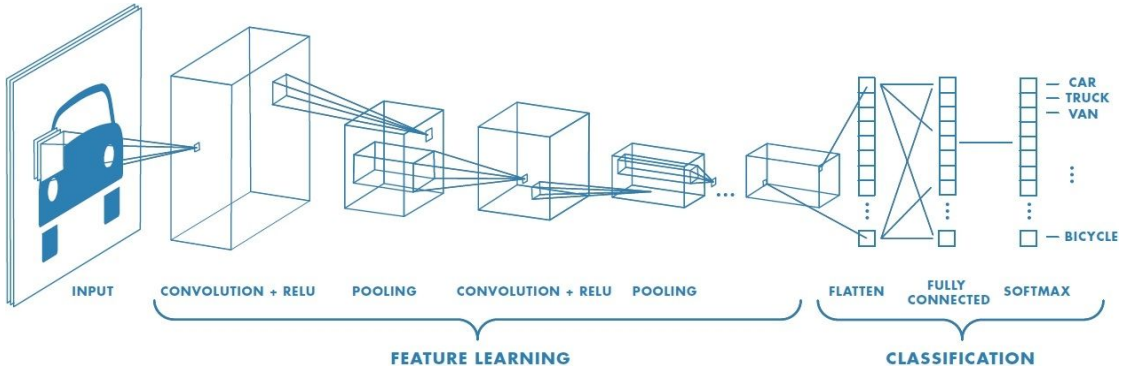
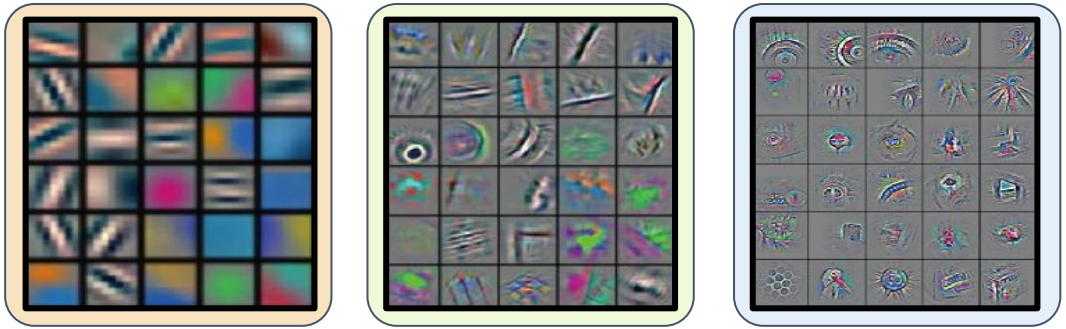


Image: https://www.mathworks.com

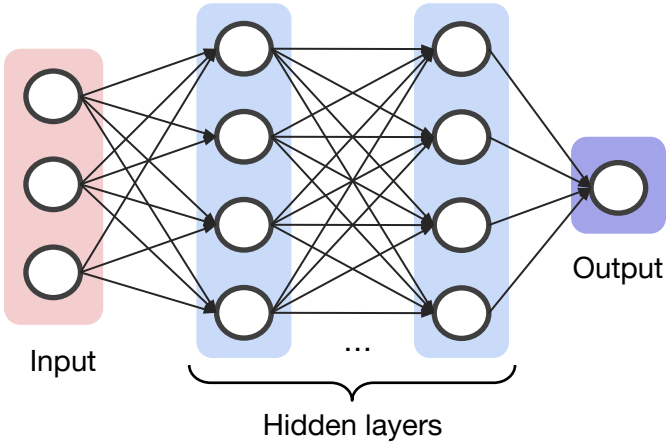


Low-level features | Mid-level features | High-level features

[Zeiles & Fergus al., ECCV 2014]

What's new?

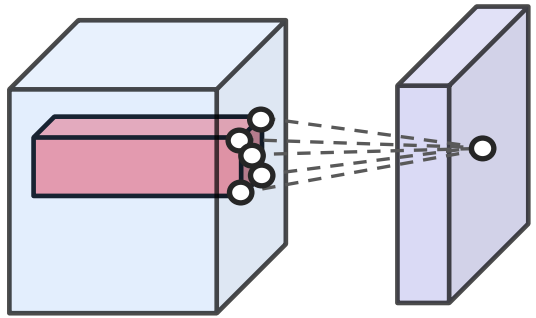
Multi-layer perceptron (MLP)



Many parameters

Limited depth

CNN



- Shared parameters
- Local connectivity

Less parameters

Deep architectures possible

Introduction

Fully connected layer

Image: 28x28 pixels

784 elements

W_k

10 x 784 values

10 classes

Introduction

Fully connected layer

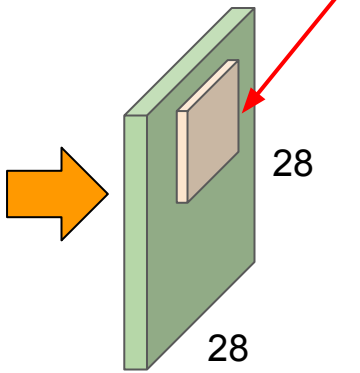
Image: 28x28 pixels

1 value: Scalar product between the input (vectorized) and a row of the matrix W

Introduction

Convolutional layer

Image: 28x28 pixels



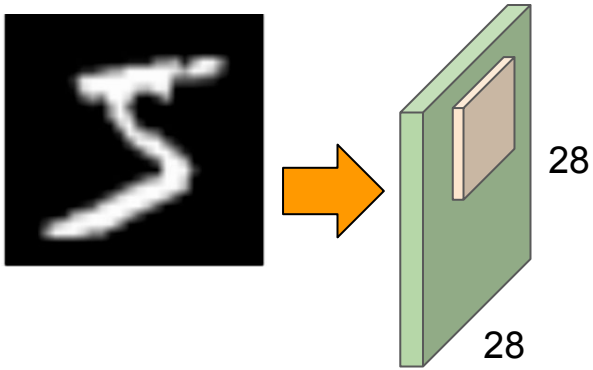
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

Introduction

Convolutional layer

Image: 28x28 pixels

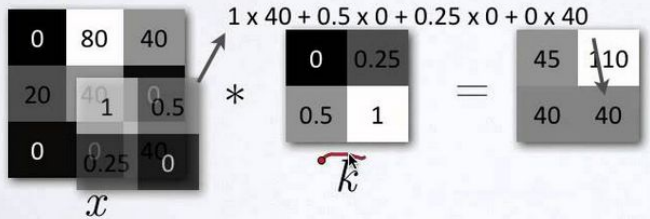


It can be formulated as:

- The convolution of an image x with a kernel k is computed as follows:

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q}\, k_{r-p,r-q}$$

- Example:



Introduction

Convolutional layer

Image: 28x28 pixels

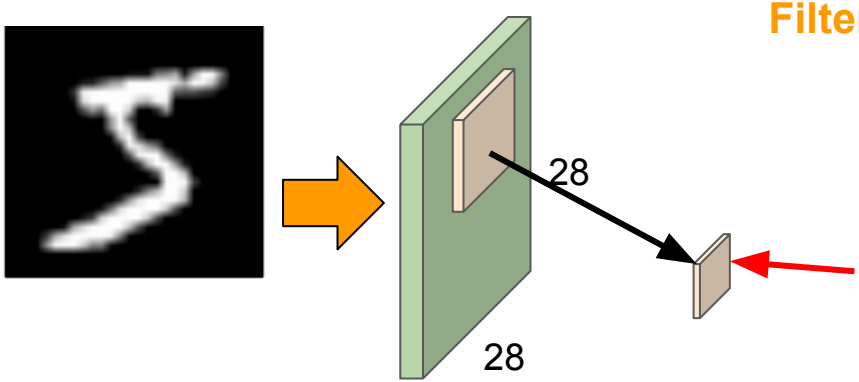


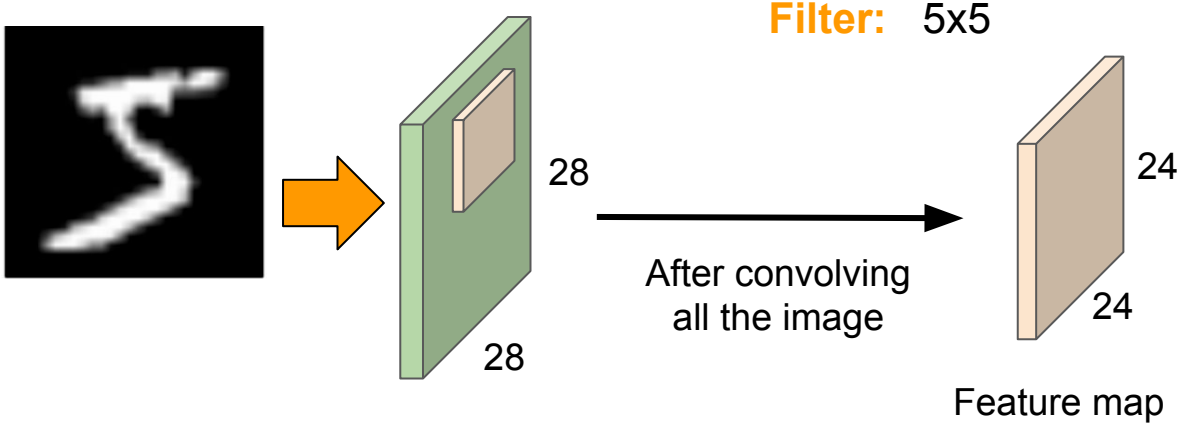
Image: 28x28x1
Filter: 5x5

Result: Scalar product between the filter and the image region.

Introduction

Convolutional layer

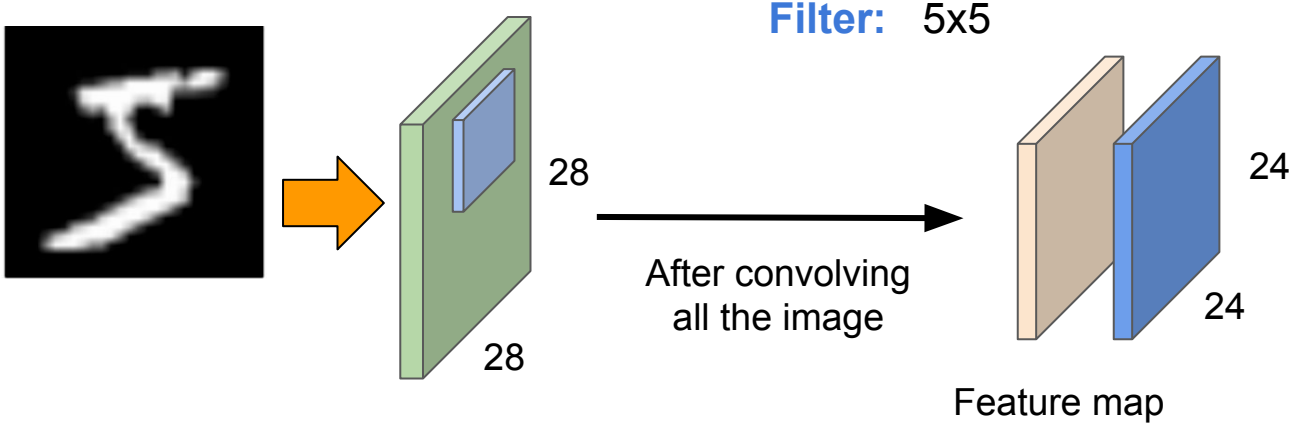
Image: 28x28 pixels



Introduction

Convolutional layer (We have several filters!!!)

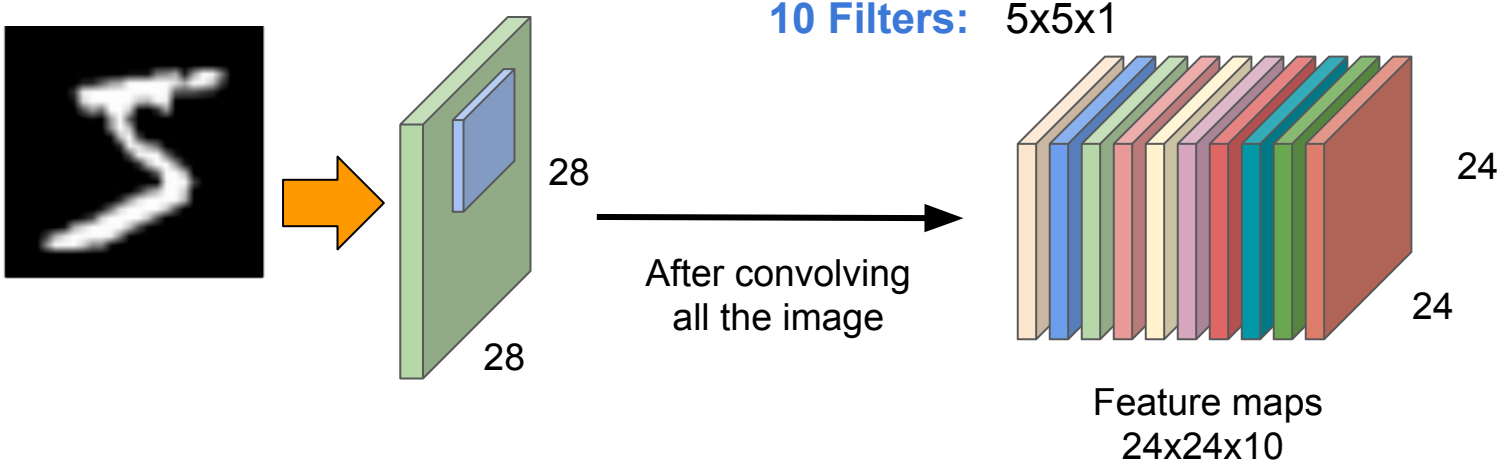
Image: 28x28 pixels



Introduction

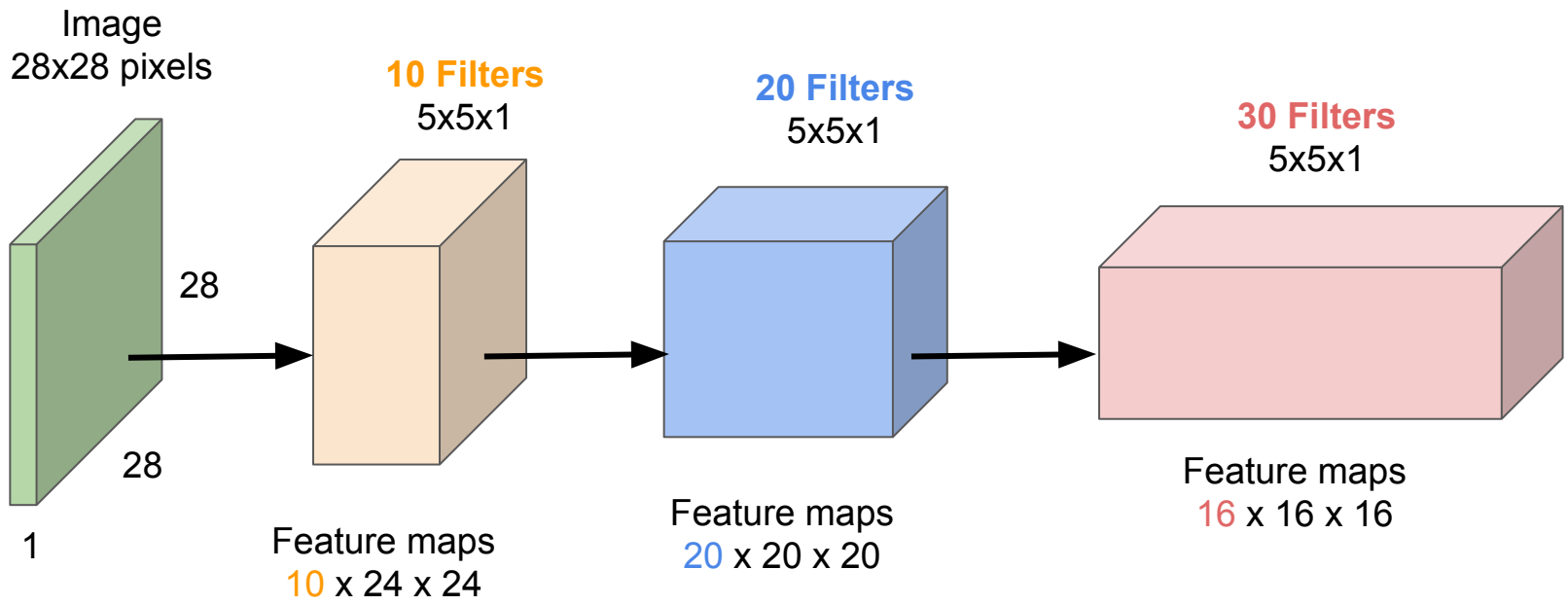
Convolutional layer (We have several filters!!!)

Image: 28x28 pixels



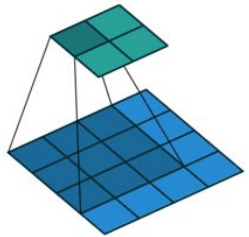
Introduction

Convolutional neural network (simplified)



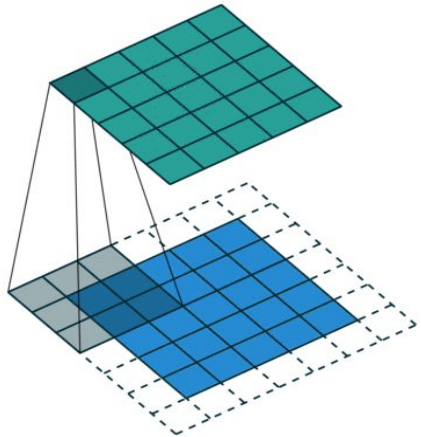
Introduction

Basics: Padding



No padding
(unit stride)

Reduced dimension
along border

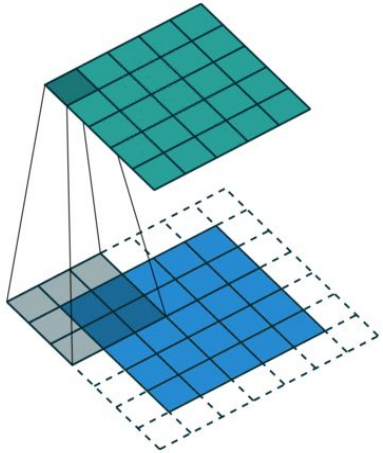


Padding
(unit stride)

Preserved spatial
dimension

Introduction

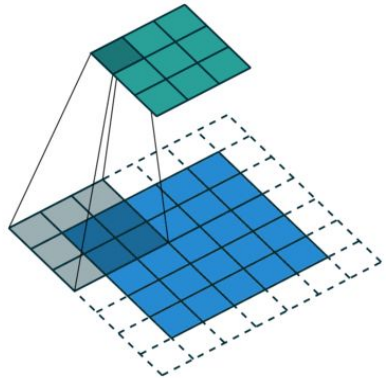
Basics: Stride



Unit stride
(padding)

Larger spatial dimension
(more computations/memory)

Overlapping receptive fields



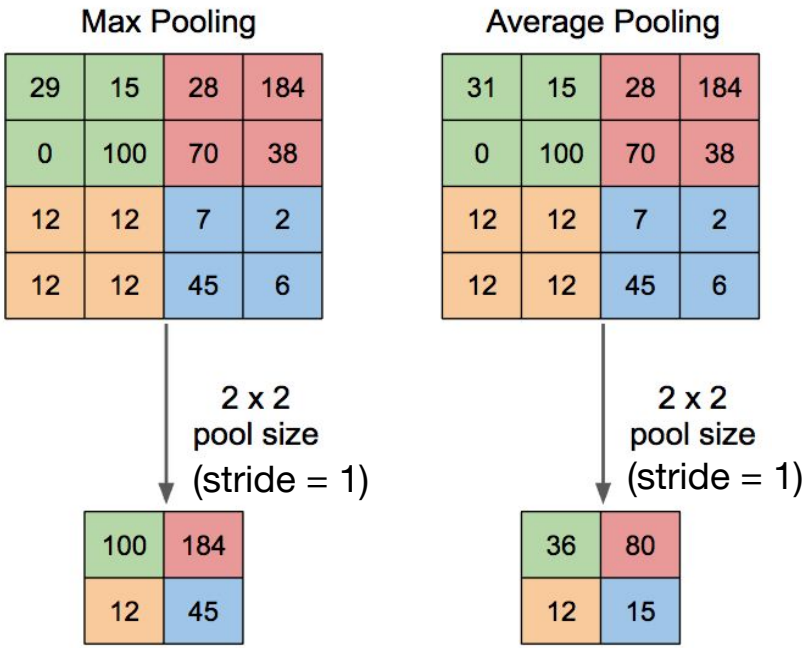
Stride = 2
(padding)

Reduced spatial dimension
(less computations/memory)

Reduced overlapping

Introduction

Basics: Pooling



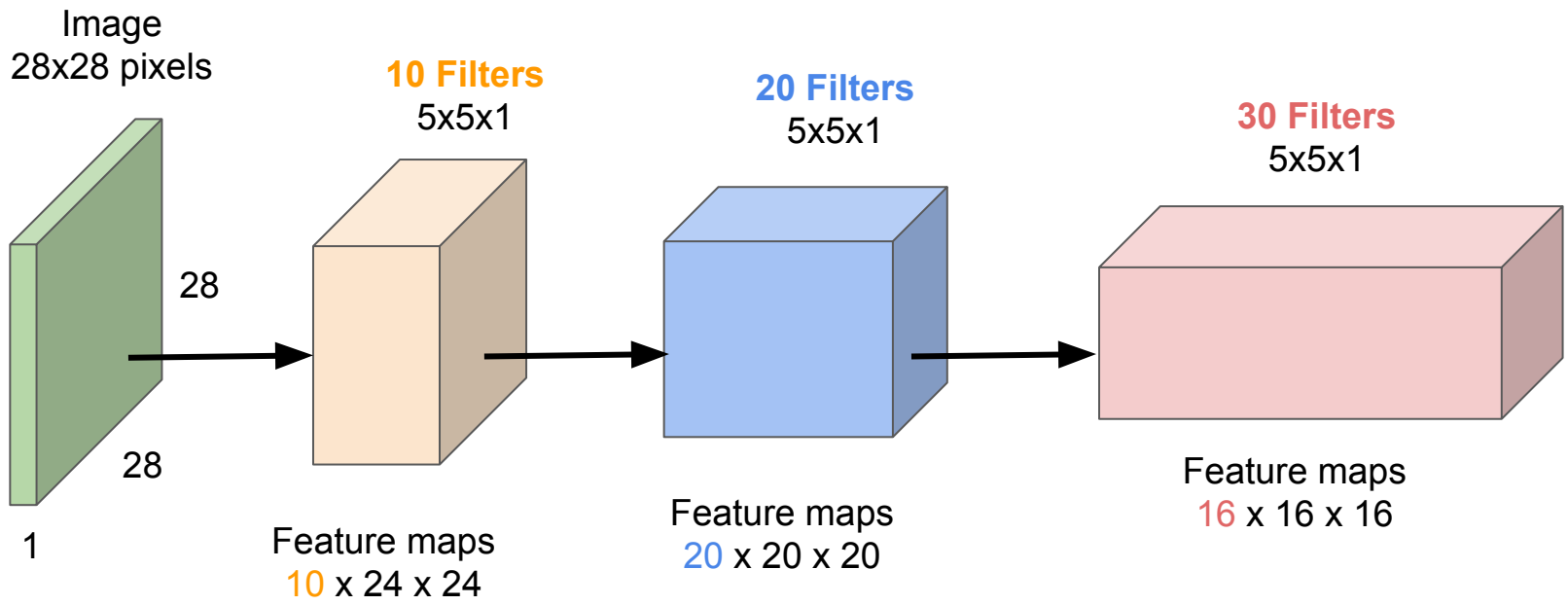
Goals

- Reduce the spatial resolution of feature maps
- Lower memory and computation requirements
- Provide partial invariance to position, scale and rotation

Pooling is typically done separately for each feature map

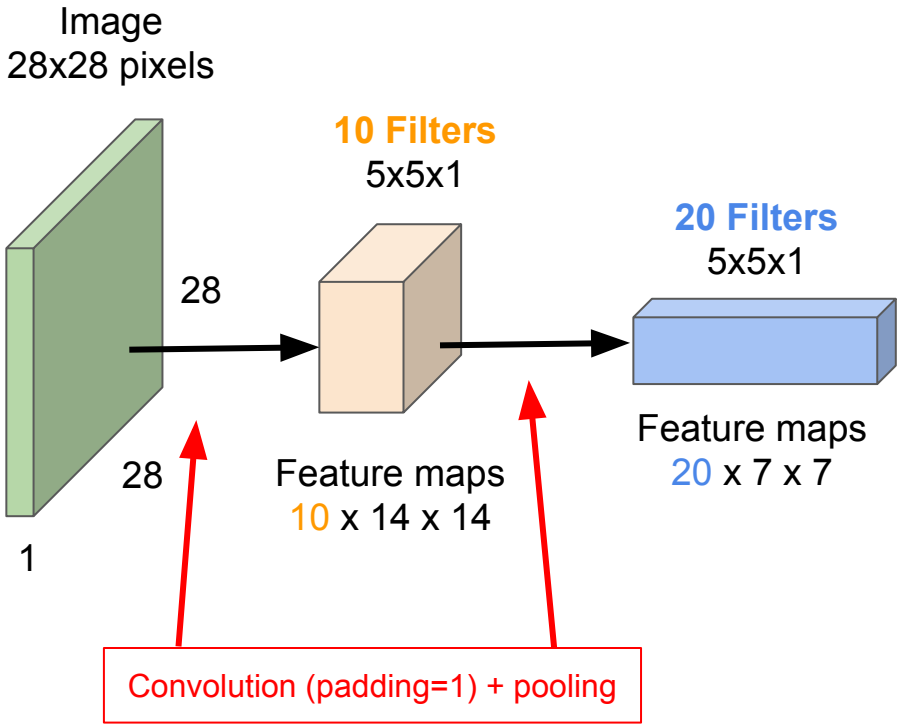
Introduction

Convolutional neural network (simplified)



Introduction

Convolutional neural network (simplified)

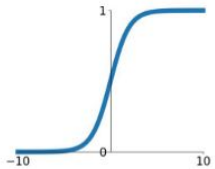


Introduction

Basics: Activation functions

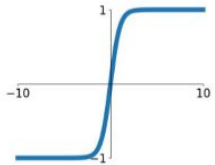
Sigmoid

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



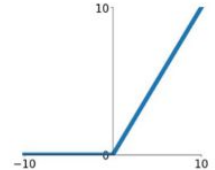
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$

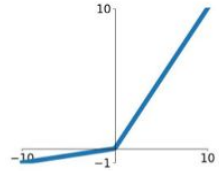


Reduces the vanishing gradient problem

Ensures that neurons are always activated

Leaky ReLU

$$\max(0.1x, x)$$

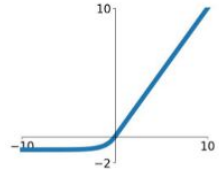


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

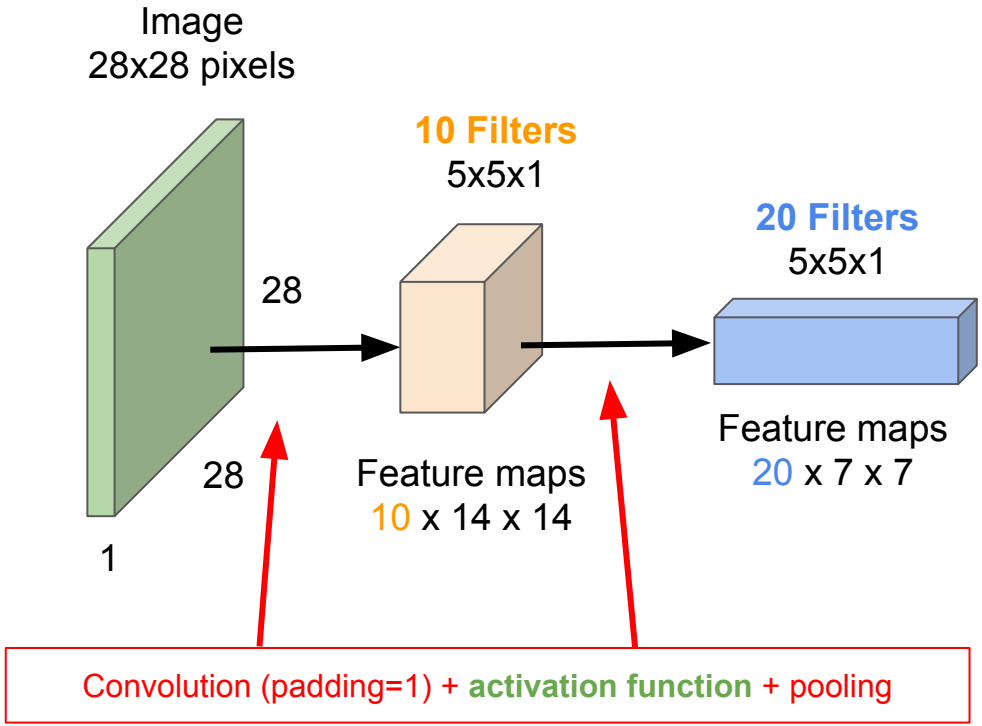
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



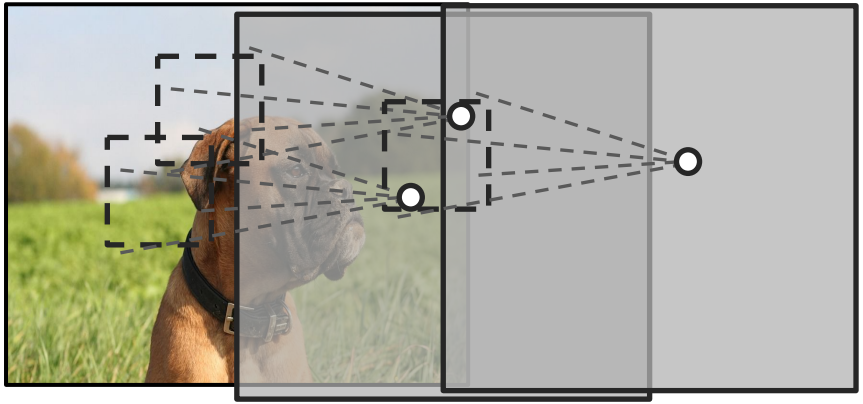
Introduction

Convolutional neural network (simplified)



Introduction

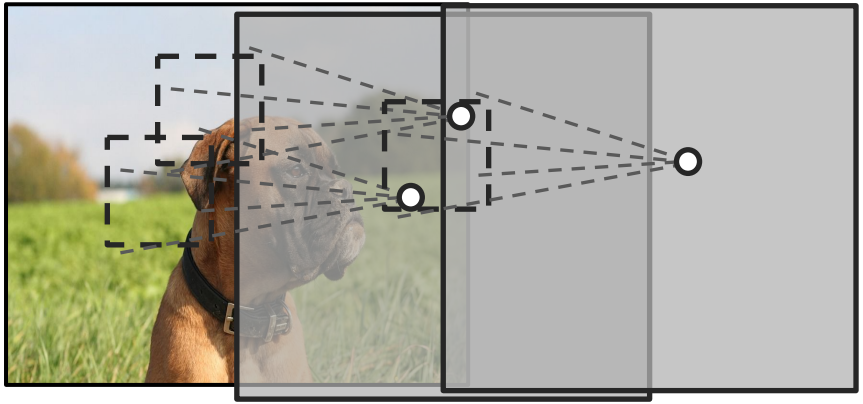
Basics: Receptive field



- Region in the input space which can the neuron's output
- A large receptive is necessary to capture spatial context
- The receptive field of neurons increases when going deeper in the network.

Introduction

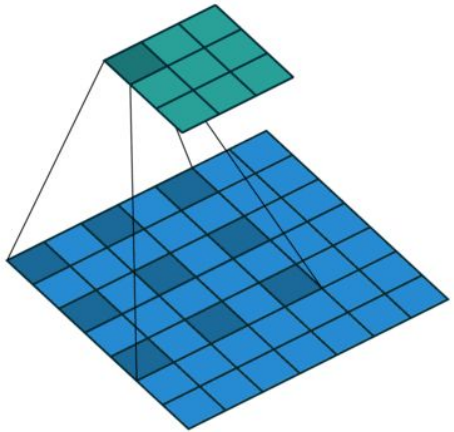
Basics: Receptive field



Question: How to get a large receptive field without having too many network parameters ?

Introduction

Basics: Dilated convolutions (à trous)

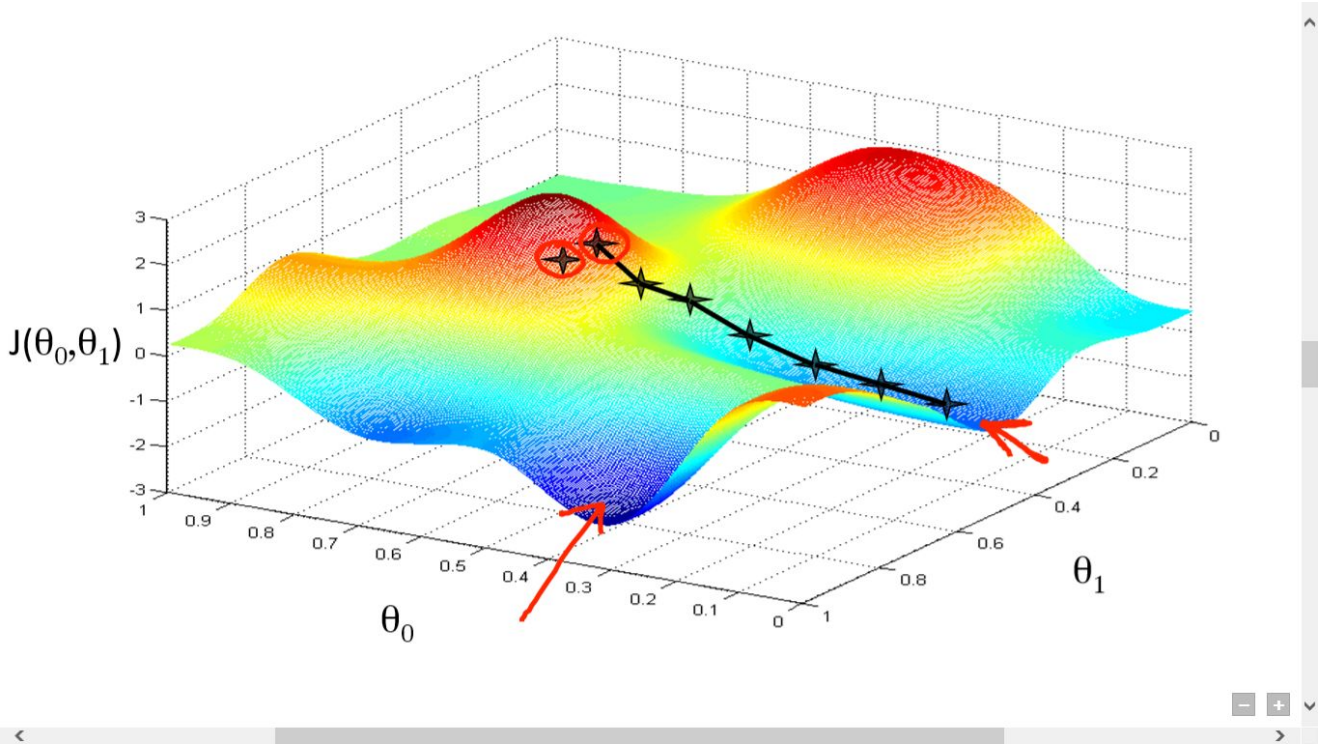


- Inflates the kernel by inserting spaces between the kernel elements
- Controlled by dilation parameter *d* (*d* = 1 gives a regular convolution)

More in optimization

CNN Training

'Simply' an optimization problem



CNN Training

Gradient descent

Optimization

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

θ ➡ Parameters

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```

- It is very slow
- Intractable for data that do not fit in memory

CNN Training

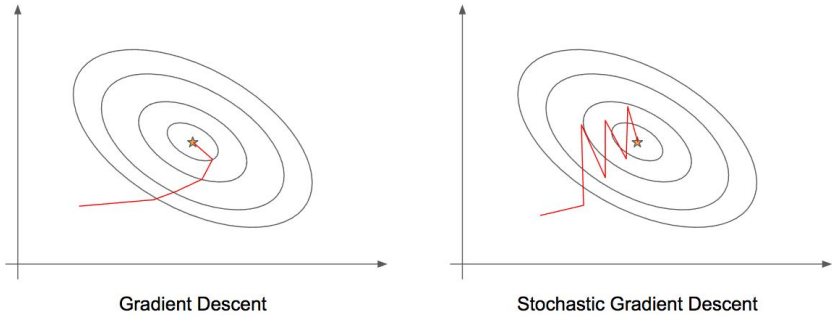
Stochastic Gradient descent (SGD)

Optimization

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(n)}; y^{(n)})$$

θ ➡ Parameters

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data, batch_size=50):
    params_grad = evaluate_gradient(loss_function, batch, params)
    params = params - learning_rate * params_grad
```



CNN Training

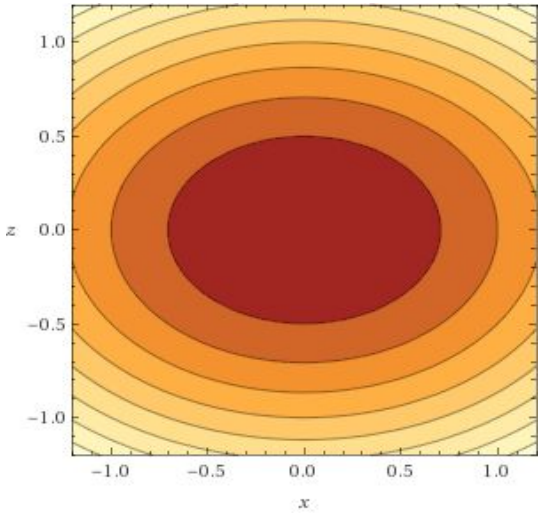
Stochastic Gradient descent (SGD)

Optimization

Problems

θ ➡ Parameters

'Condition number'



$$f(x,z) = x^2 + az^2$$

$$H_{xx} = \frac{\partial^2 f}{\partial x^2} = 2 \qquad H_{xz} = \frac{\partial^2 f}{\partial x \partial z} = 0$$

$$H_{zx} = \frac{\partial^2 f}{\partial z \partial x} = 0 \qquad H_{zz} = \frac{\partial^2 f}{\partial z^2} = 2a$$

$$\frac{H_{zz}}{H_{xx}} = a$$

CNN Training

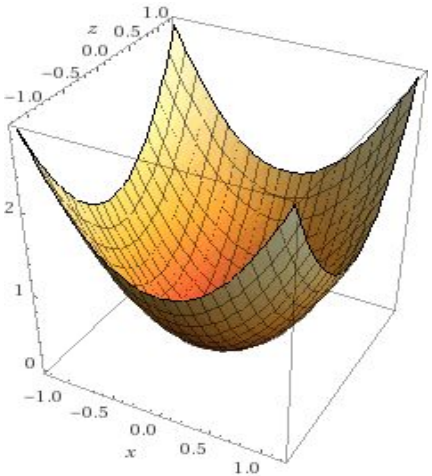
Stochastic Gradient descent (SGD)

Optimization

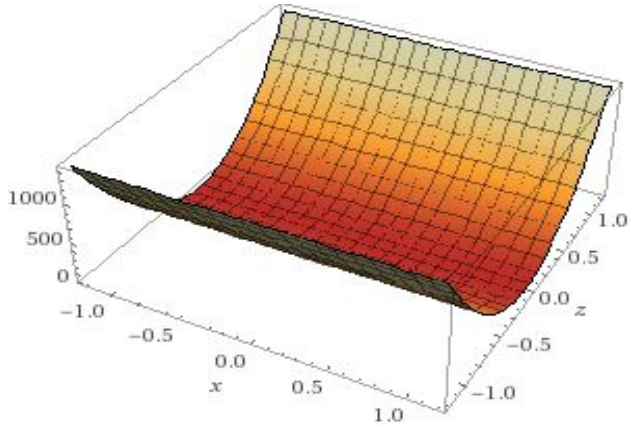
Problems

θ ➡ Parameters

'Condition number'



a = 1



a = 1000

CNN Training

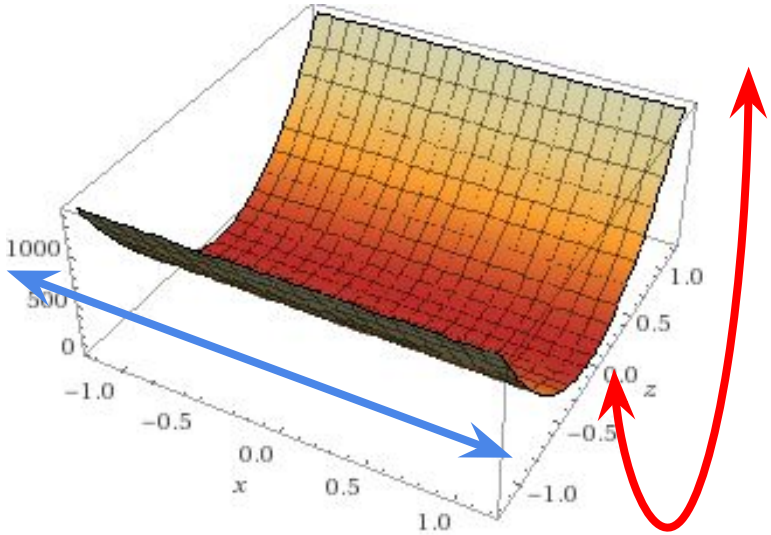
Stochastic Gradient descent (SGD)

Optimization

Problems

θ ➡ Parameters

'Condition number'



Gradient not very sensitive along this direction

Gradient very sensitive along this direction

a = 1000

CNN Training

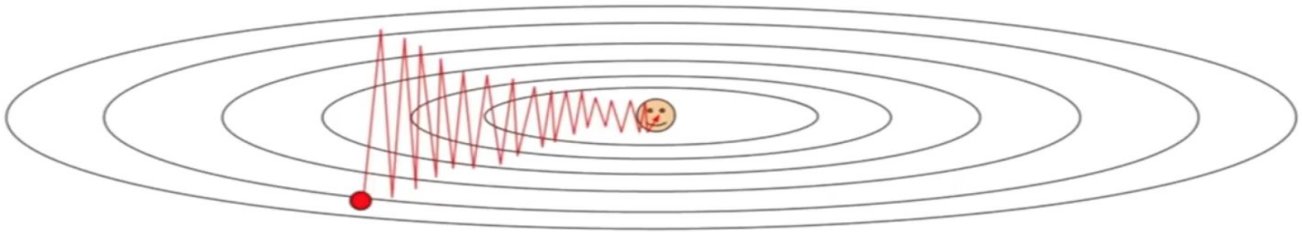
Stochastic Gradient descent (SGD)

Optimization

Problems

θ ➡ Parameters

'Condition number'



Evolution towards the minimum becomes slow

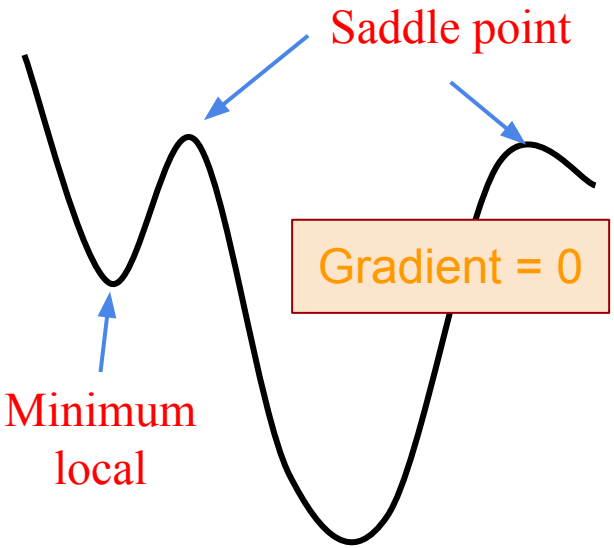
CNN Training

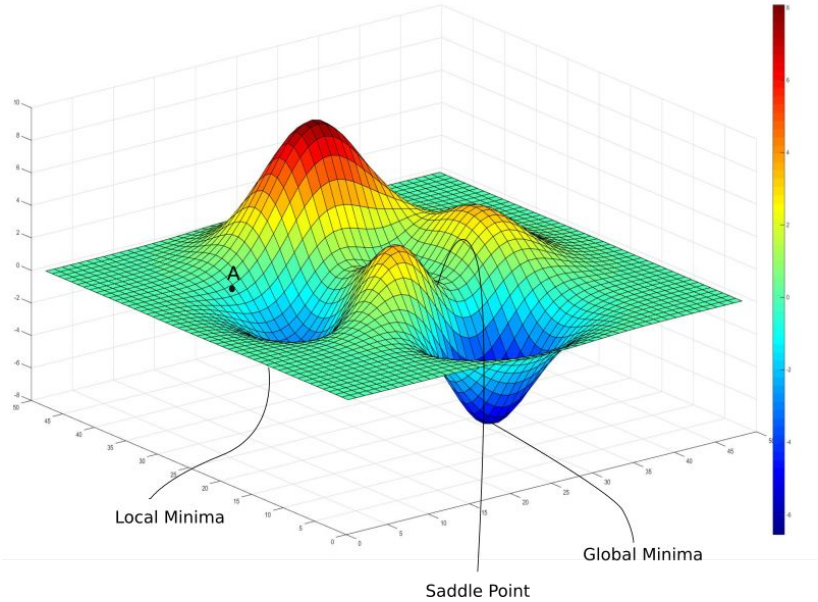
Stochastic Gradient descent (SGD)

Optimization

Problems

θ ➡ Parameters





CNN Training

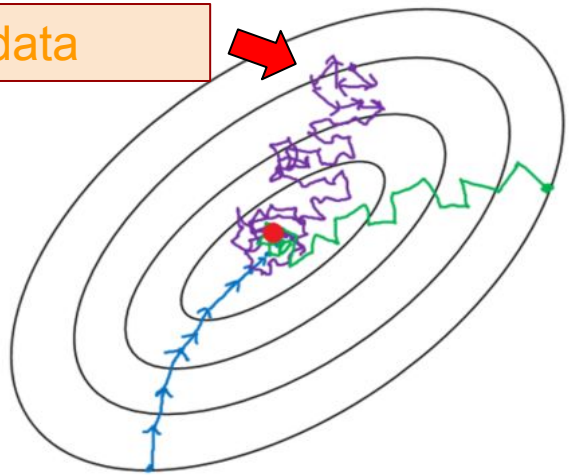
Stochastic Gradient descent (SGD)

Optimization

Problems

θ ➡ Parameters

Noisy data



Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent

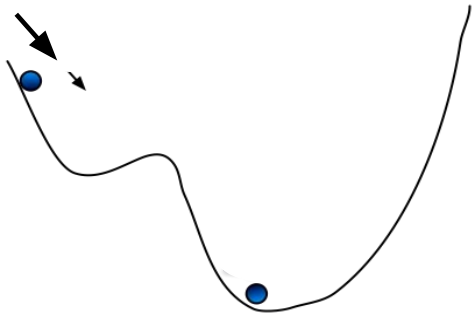
CNN Training

SGD + Momentum

Optimization

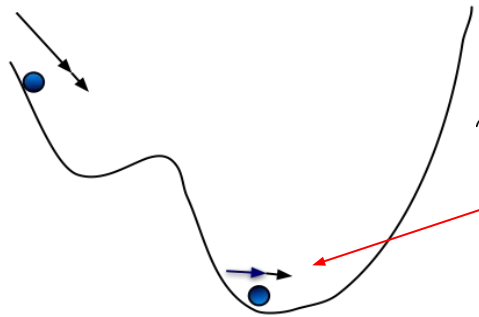
Standard gradient

$$\theta^{t+1} = \theta^t - \eta \cdot \nabla J(\theta^t)$$



With momentum

$$\theta^{t+1} = \theta^t - \eta v^{t+1}$$



Benefits:

- Accelerates learning when gradient direction is stable
- Reduces oscillations during training
- Faster convergence, possibly better solution

CNN Training

SGD + Momentum

Optimization

Standard gradient

```
for i in range (nb_epochs):
    dx = compute_grad(x)
    x = x - learning_rate * dx
```

With momentum

```
vel = 0
for i in range (nb_epochs):
    dx = compute_grad(x)
    vel = vel * rho + dx
    x = x - learning_rate * vel
```

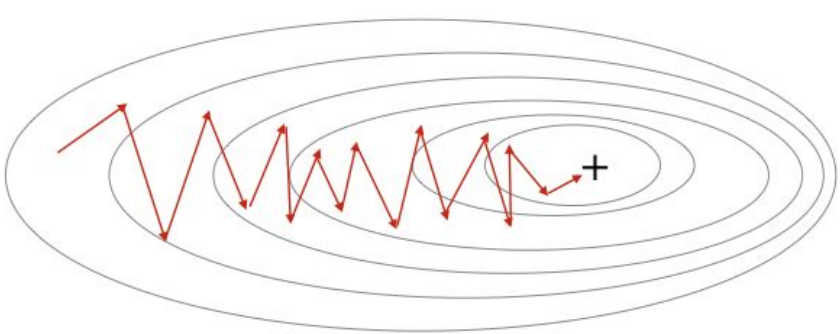
CNN Training

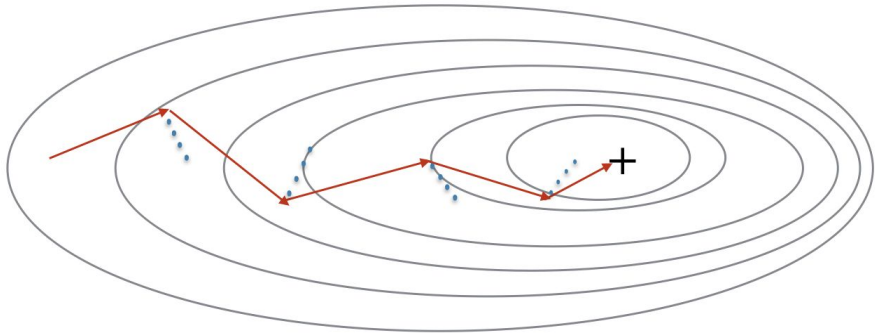
SGD + Momentum

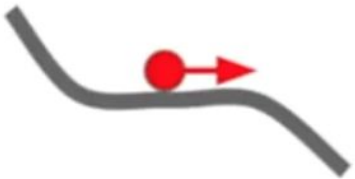
Optimization

Standard gradient

With momentum



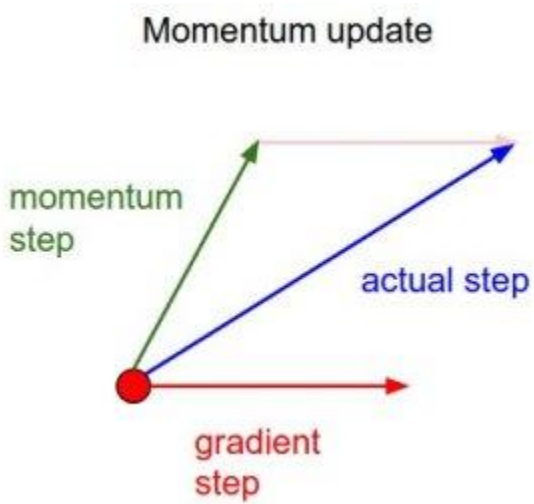


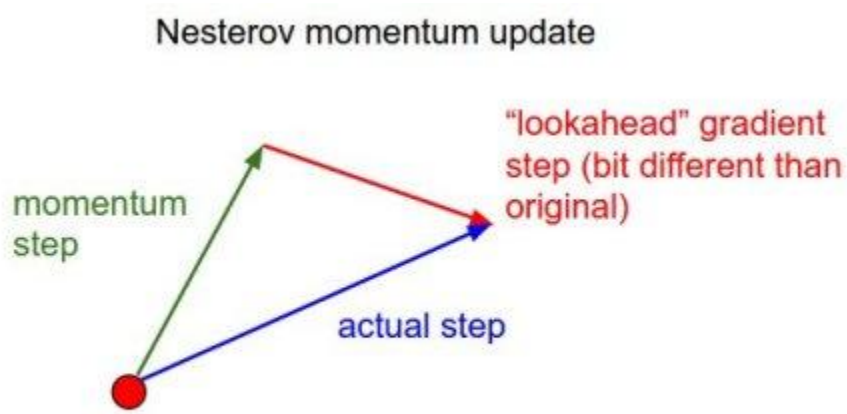


CNN Training

Nesterov Momentum

Optimization





$$v^{t+1} = \rho v^t - \eta \nabla J(\theta^t + \rho v^t)$$
$$\theta^{t+1} = \theta^t - \eta v^{t+1}$$

Source: http://cs231n.github.io/assets/nn3/nesterov.jpeg

CNN Training

Adaptative moment estimation (Adam)

Optimization

$$m_t = \gamma_1 m_{t-1} + (1-\gamma_1)\frac{\partial \mathcal{L}}{\partial w}$$

Decaying average of gradient first moment (mean)

$$v_t = \gamma_2 v_{t-1} + (1-\gamma_2)\frac{\partial \mathcal{L}^2}{\partial w}$$

Decaying average of gradient second moment (~variance)

$$\hat{v}_t = \frac{v_t}{1-\gamma_2^t} \qquad \hat{m}_t = \frac{m_t}{1-\gamma_1^t}$$

Correct values to avoid bias to zero in first steps

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{v^t + \epsilon}} m^t$$

CNN Training

Adaptative moment estimation (Adam)

Optimization

```
eps = 1e-7
first_moment = 0
second_moment = 0
for i in range (nb_epochs):
    dx = compute_grad(x)
    first_moment = beta1 * first_moment + (1 - beta1) * dx
    second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
    x = x - learning_rate * first_moment / (np.sqrt(second_moment) + eps)
```

$$m_t = \gamma_1 m_{t-1} + (1-\gamma_1)\frac{\partial \mathcal{L}}{\partial w}$$

First moment

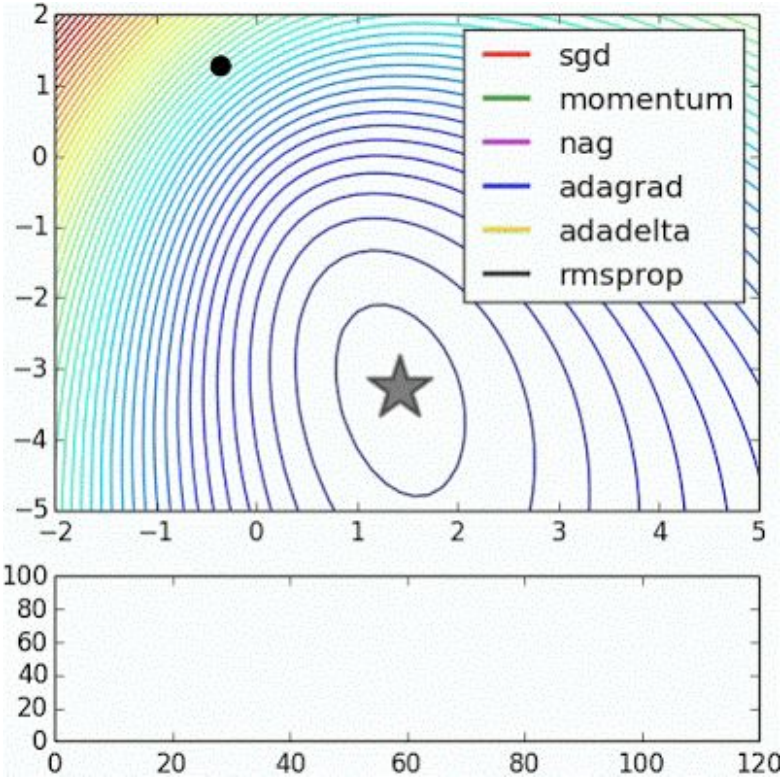
Second moment

$$v_t = \gamma_2 v_{t-1} + (1-\gamma_2)\frac{\partial \mathcal{L}^2}{\partial w}$$

CNN Training

Comparison

Optimization



CNN Training

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

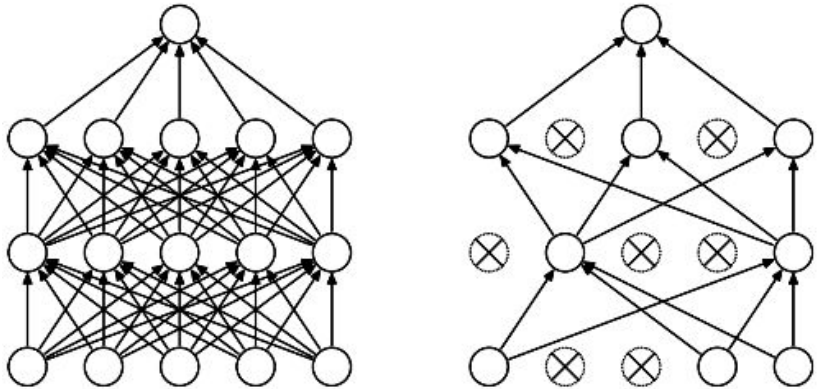
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Key ideas:

- Normalize the output of a layer using batch mean and standard deviation (mean of 0, standard deviation of 1)
- Optimal scale and shift is learned by the network (parameters gamma and beta)
- Stabilizes the learning process, faster convergence

CNN Training

Dropout



Standard network

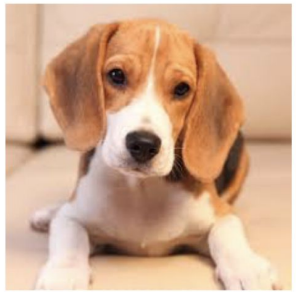
With dropout

Key ideas:

- Regularization technique to avoid overfitting
- Ignore some neurons at each iteration (forward and backward passes)
- Dropped neurons chosen randomly with probability ***p***
- Avoid co-adaptation of neurons: each neuron should work independently of others

How prediction is done?

Example: classification

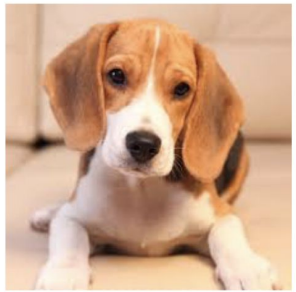


CNN output (*Logits*)

Ciel	2.45
Chien	**4.29**
Velo	-1.24

How prediction is done?

Example: classification



CNN output (*Logits*) → *Softmax* → $P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$

		exp	normalize	
Ciel	2.45	11.58	0.14	
Chien	**4.29**	**72.96**	**0.86**	SoftMax
Velo	-1.24	0.29	0.00	

How the objective is evaluated?

Example: classification

Cost function: cross-entropy  Widely used

$$KL(Q\|P) = \sum_t Q(t|\mathbf{x}) \log \frac{Q(t|\mathbf{x})}{P(t|\mathbf{x})} = -\sum_{k=1}^{K} I(k, t_{class}) \log P(t = k|\mathbf{x})$$

Kullback-Leibler Divergence

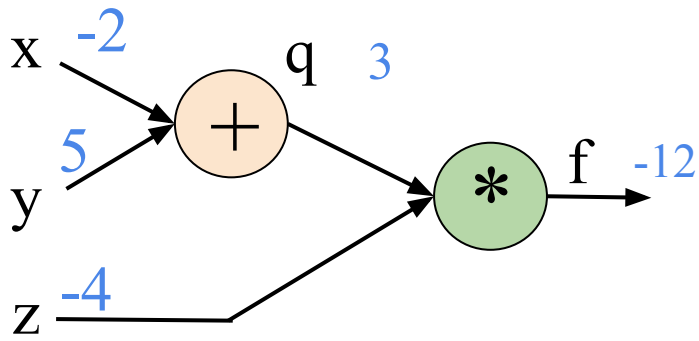
$$P(t = k|\mathbf{x}) = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4

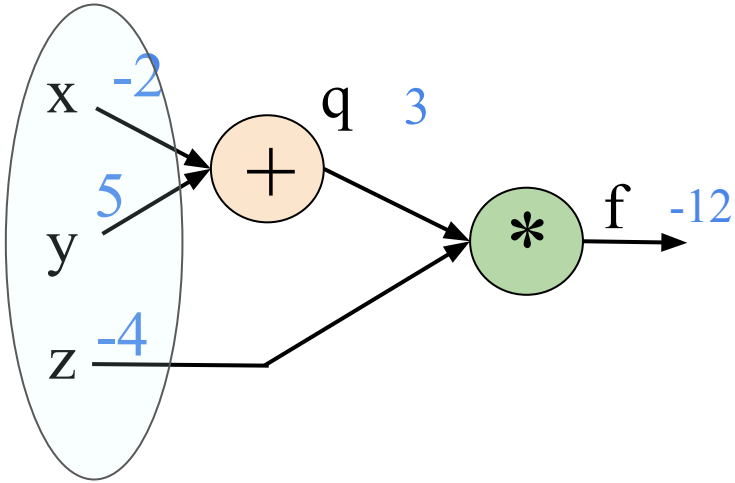


Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4



We want to know

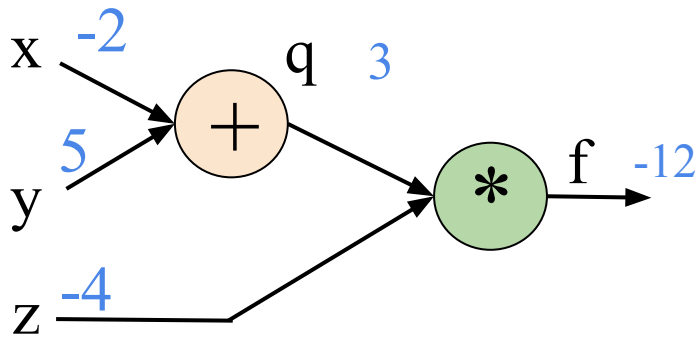
$$\frac{\partial f}{\partial x}; \frac{\partial f}{\partial y}; \frac{\partial f}{\partial z}$$

Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4



We want to know

$$\frac{\partial f}{\partial x}; \frac{\partial f}{\partial y}; \frac{\partial f}{\partial z}$$

Gradients

$$q = x + y \quad \Rightarrow \quad \frac{\partial q}{\partial x} = 1; \frac{\partial q}{\partial y} = 1$$

q respect to x and y

$$f = qz \quad \Rightarrow \quad \frac{\partial f}{\partial q} = z; \frac{\partial f}{\partial z} = q$$

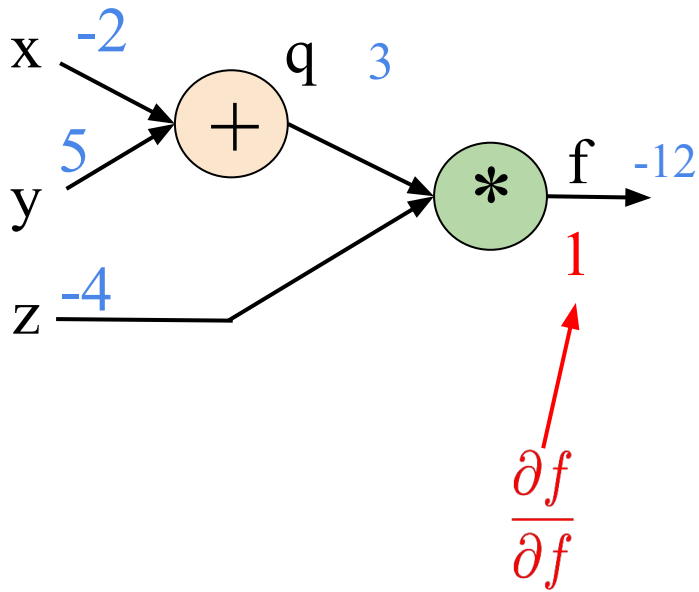
f respect to q and z

Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4



$$q = x + y \quad \Rightarrow \quad \frac{\partial q}{\partial x} = 1; \frac{\partial q}{\partial y} = 1$$

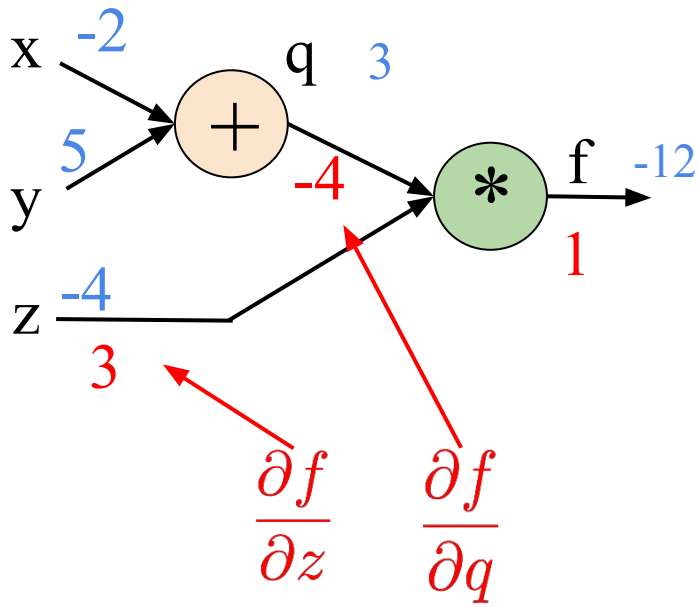
$$f = qz \quad \Rightarrow \quad \frac{\partial f}{\partial q} = z; \frac{\partial f}{\partial z} = q$$

Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4



$$q = x + y \quad \Rightarrow \quad \frac{\partial q}{\partial x} = 1; \frac{\partial q}{\partial y} = 1$$

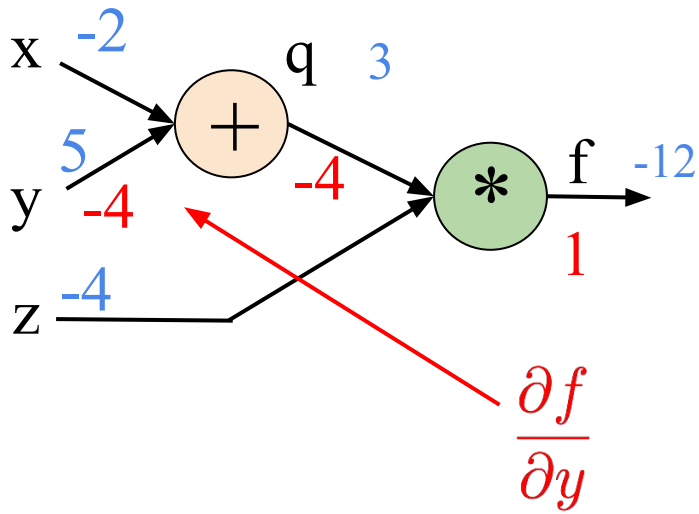
$$f = qz \quad \Rightarrow \quad \frac{\partial f}{\partial q} = z; \frac{\partial f}{\partial z} = q$$

Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4



$$q = x + y \quad \Rightarrow \quad \frac{\partial q}{\partial x} = 1; \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \Rightarrow \quad \frac{\partial f}{\partial q} = z; \frac{\partial f}{\partial z} = q$$

Chain rule

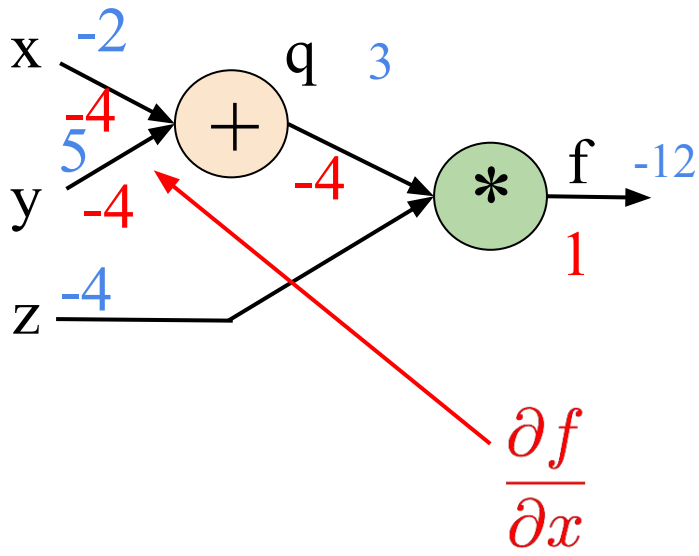
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

Back-propagating the error

The actual core of learning

$$f(x, y, z) = (x + y)z$$

x=-2 y=5 z=-4



$$q = x + y \quad \Rightarrow \quad \frac{\partial q}{\partial x} = 1; \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \Rightarrow \quad \frac{\partial f}{\partial q} = z; \frac{\partial f}{\partial z} = q$$

Chain rule

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

COMPUTER VISION TASKS

- Object classification
- Semantic segmentation
- Object detection

OBJECT CLASSIFICATION

Image classification

Dog: 85%
Cat: 10%
Horse: 5%
Chair: 0%

Goal: predict a single label (or a probability distribution over labels) for a given image.

Image classification

Challenges

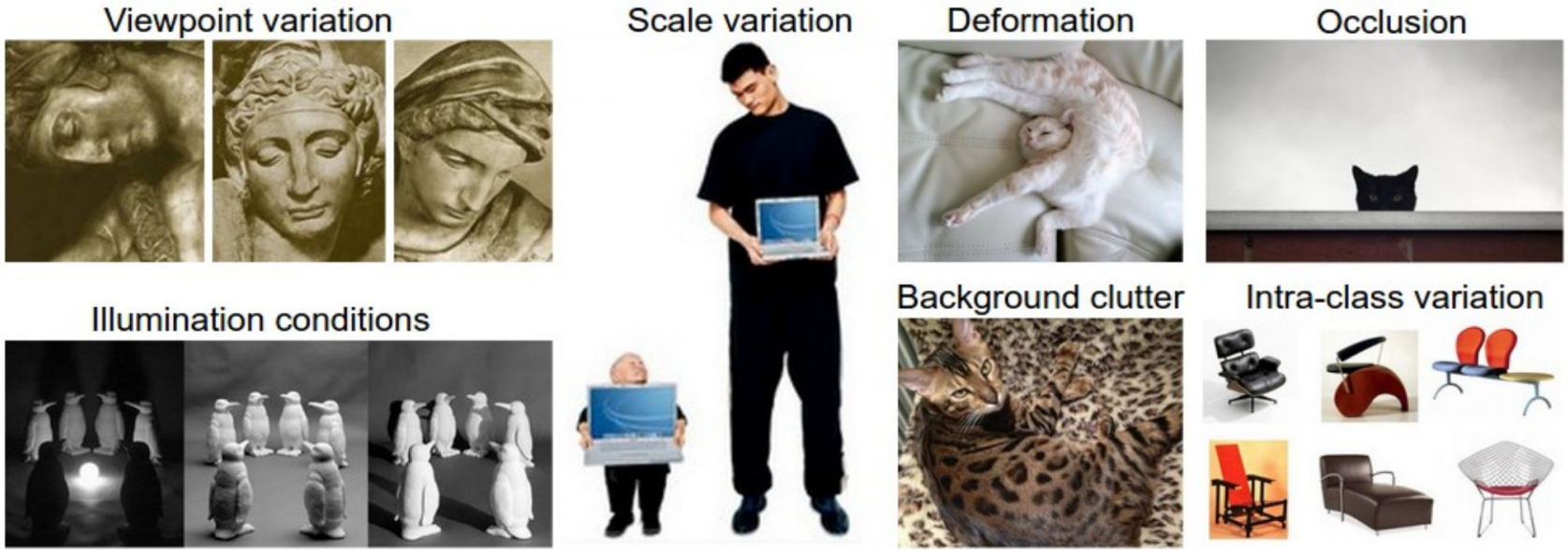
What humans see

37	49	43	43	63	45	51	56	65	59
47	64	68	37	48	56	37	47	61	47
56	67	64	39	80	66	31	48	49	33
38	49	32	75	48	49	71	35	47	27
61	62	33	64	60	49	35	40	70	49
52	32	31	56	34	32	34	27	43	36
34	77	26	36	46	27	62	76	70	65
69	36	49	37	34	41	75	61	69	46
31	40	62	30	67	43	54	77	72	72
42	69	65	76	73	61	64	34	53	66
39	52	55	64	45	78	34	76	60	57
55	68	31	56	63	77	78	69	78	35
31	47	60	43	42	51	55	57	50	78
48	41	32	35	55	39	63	50	29	40
31	28	33	60	71	68	28	40	73	76
32	63	31	80	58	67	70	67	60	38
69	49	33	35	44	66	67	38	45	46
42	50	35	40	42	66	32	29	80	30
59	59	36	47	50	31	54	68	38	61
79	29	43	30	49	63	43	62	61	35

What the computer sees

Image classification

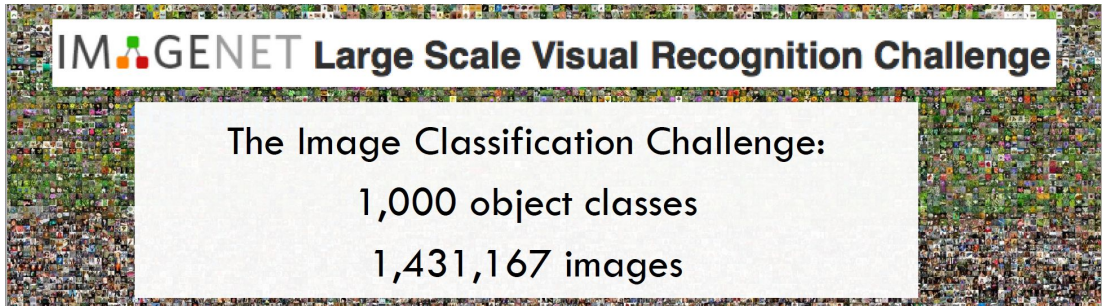
Other challenges



Easy for humans (and lesser animals). Hard for computers?

Image: http://cs231n.github.io

Image classification



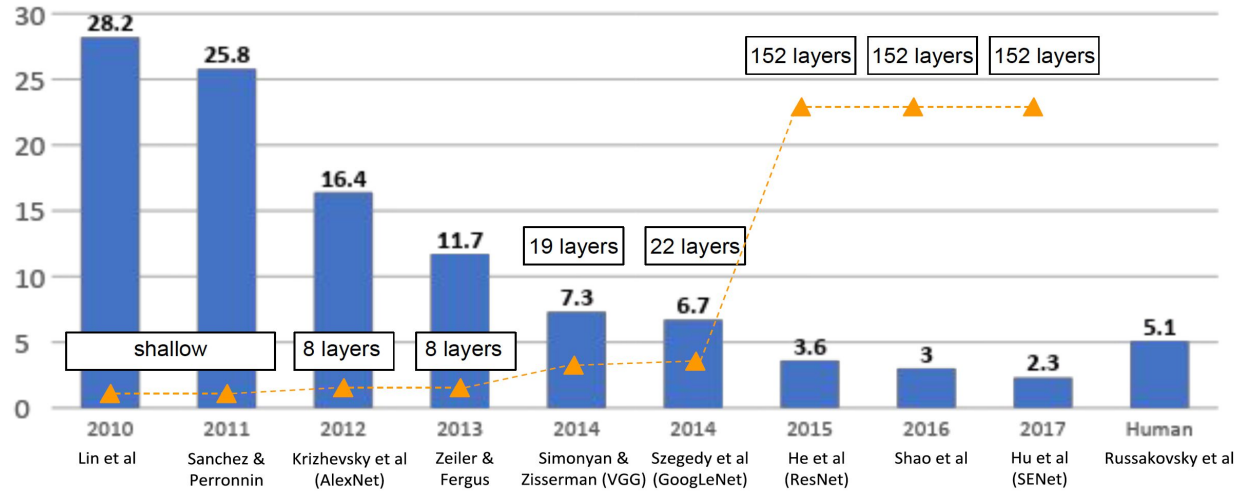


Image: http://cs231n.stanford.edu/slides/2018

Image classification

AlexNet [Krizhevsky et al, 2012]

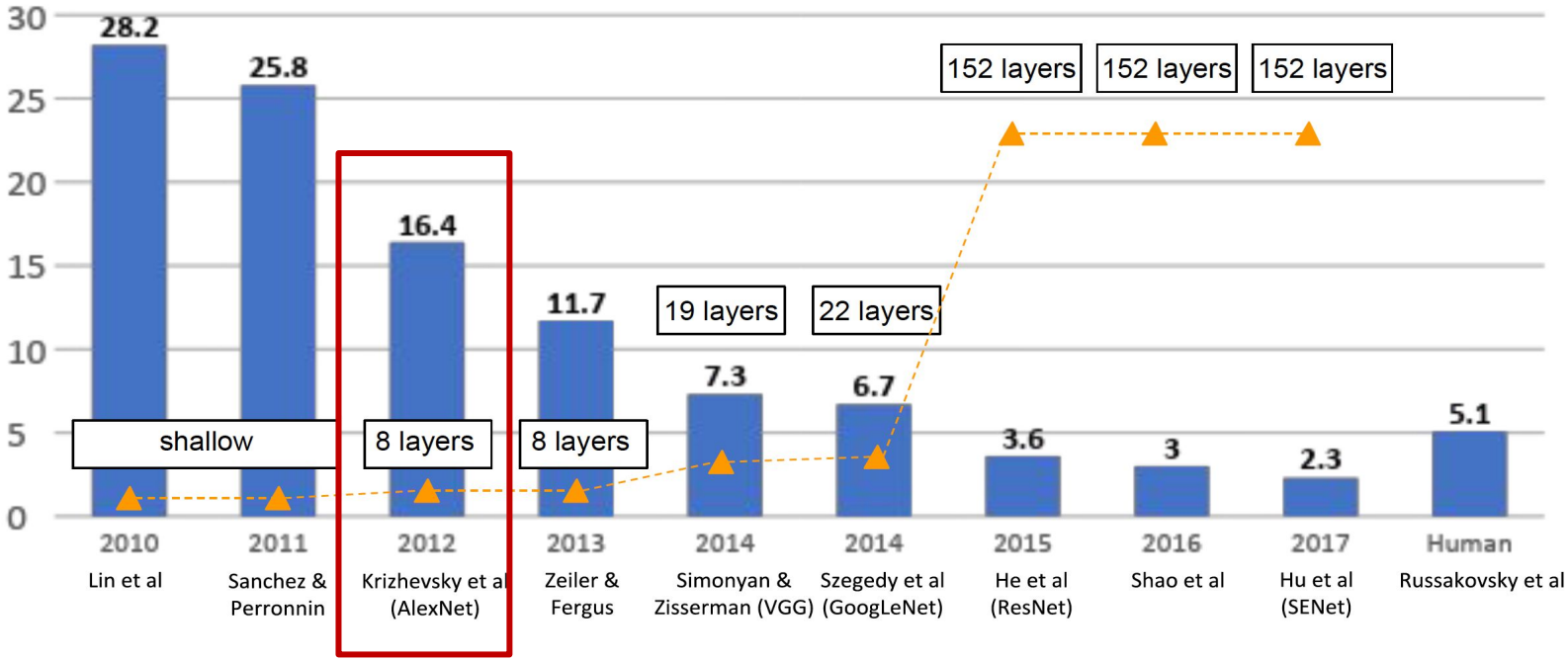
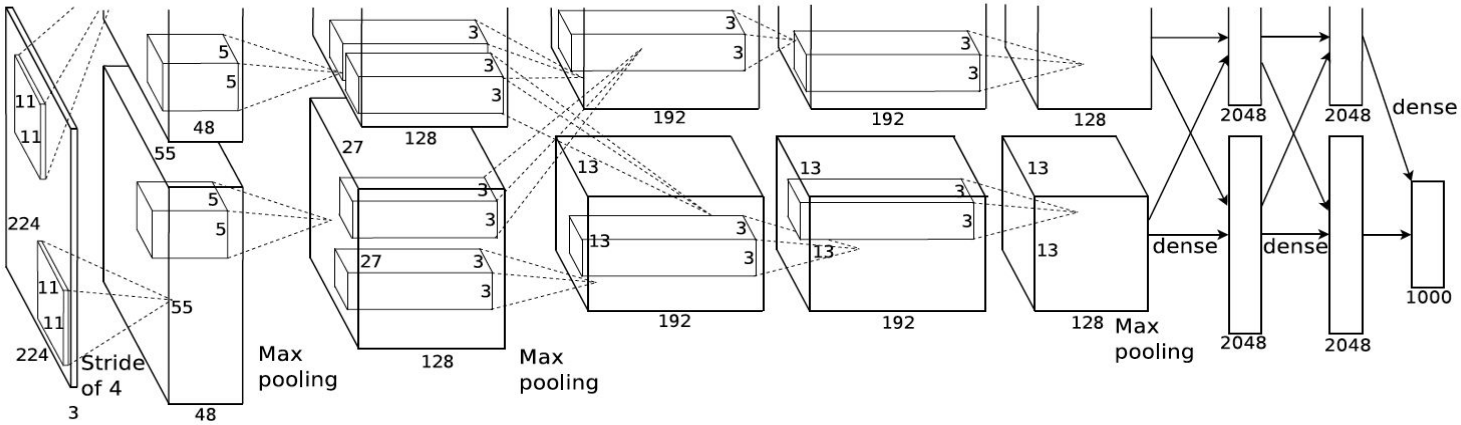


Image: http://cs231n.stanford.edu/slides/2018

Image classification

AlexNet [Krizhevsky et al, 2012]



- First model to perform well on the challenging ImageNet dataset.
- Combined techniques used in today's architectures, like ReLU, data augmentation and dropout
- Largely responsible for the rise of deep learning in computer vision

Image: Krizhevsky et al. "Imagenet classification with deep convolutional neural networks," NIPS 2012.

Image classification

ZF Net [Zeiler and Fergus, 2014]

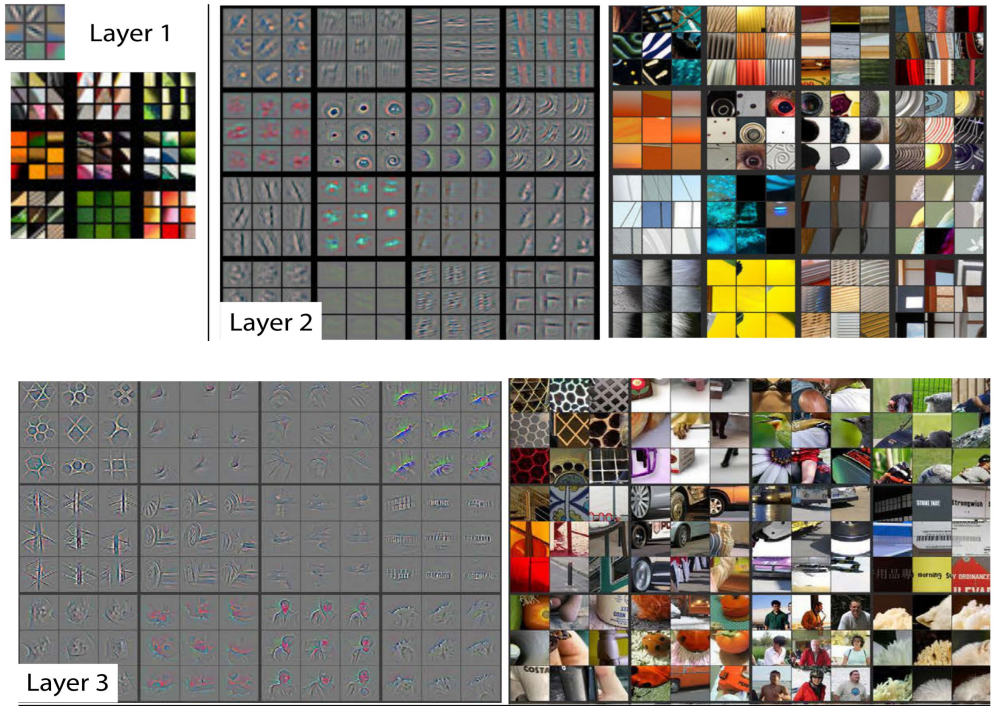


Image: Zeiler and Fergus. "Visualizing and understanding convolutional networks." ECCV, 2014.

Image classification

VGG [Simonyan and Zisserman, 2014]

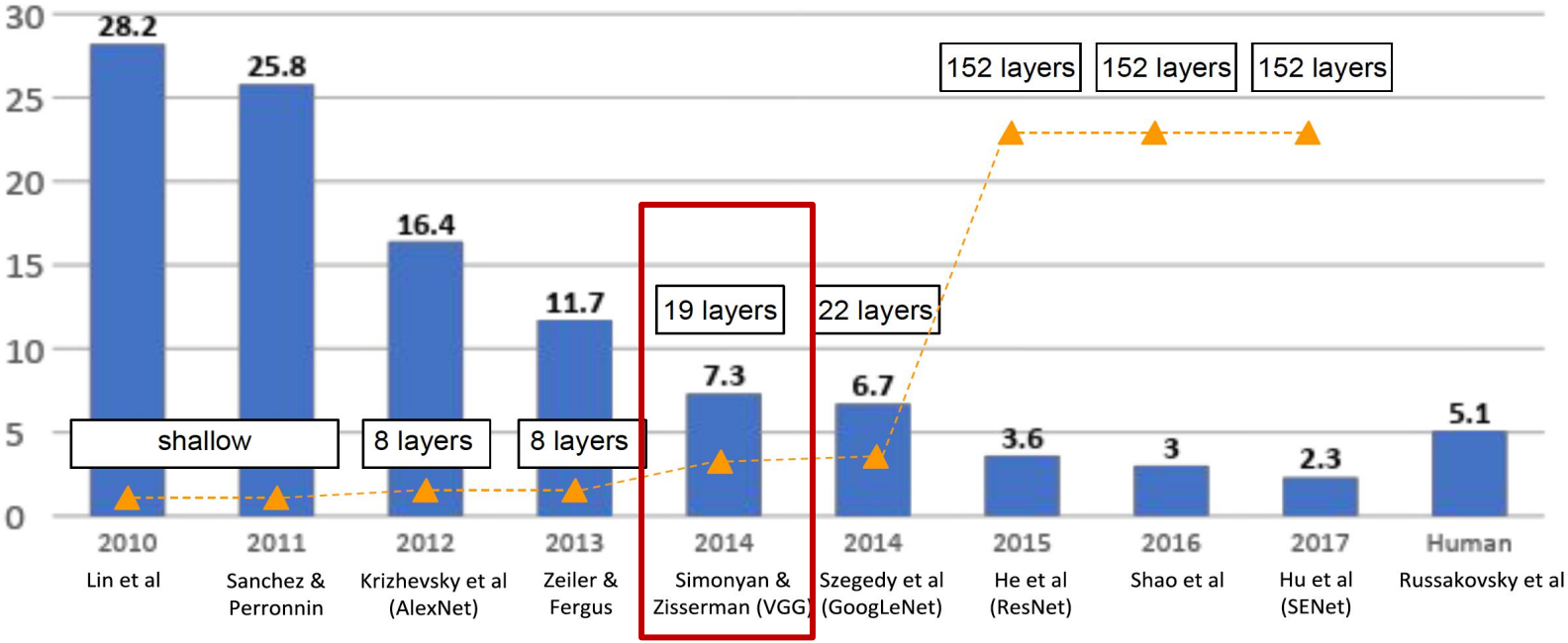


Image: http://cs231n.stanford.edu/slides/2018

Image classification

VGG [Simonyan and Zisserman, 2014]

- **Simpler structure**: only 3x3 convolutions, ReLU and 2x2 max pooling
- **Deeper network**: 16 and 19 layers (compared to 8 for AlexNet)
- **Key idea**: cascading two 3x3 convolutions gives the same receptive field as a 5x5 convolution, with much less parameters

Image: http://cs231n.stanford.edu/slides/2018

Image classification

GoogLeNet (inception v1) [Szegedy et al, 2014]

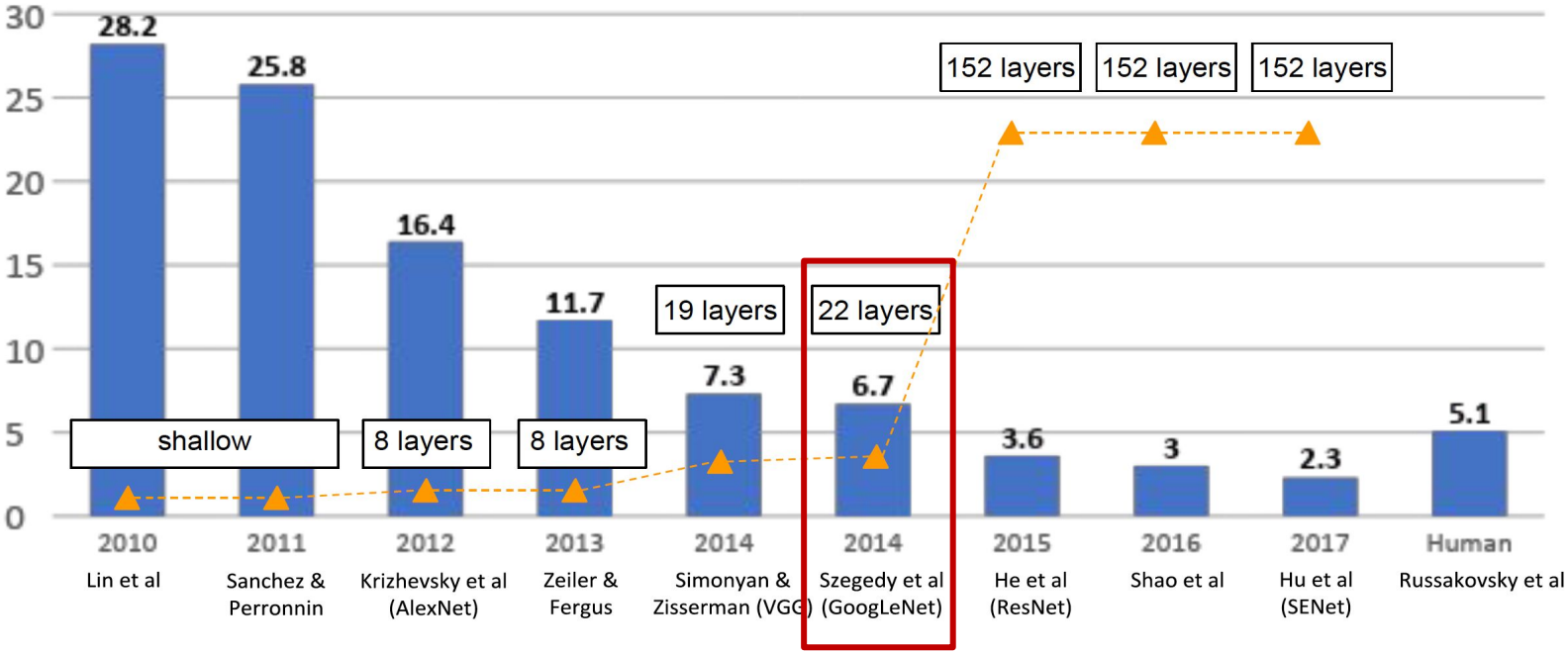
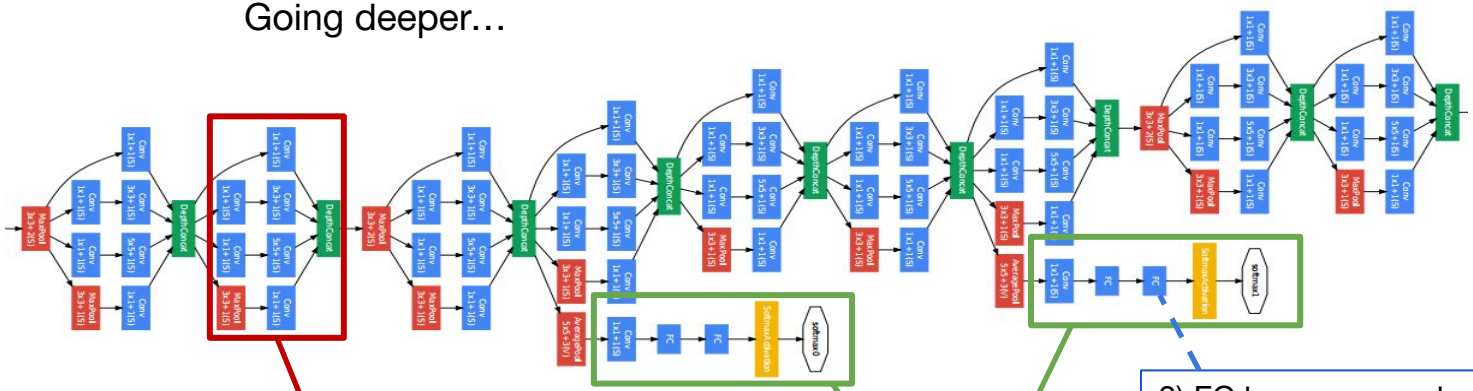


Image: http://cs231n.stanford.edu/slides/2018

Image classification

GoogLeNet (inception v1) [Szegedy et al, 2014]

Going deeper…



1) Repeating blocks called *Inception module*

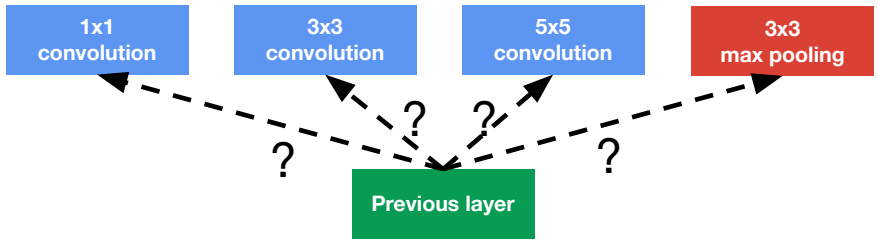
2) Intermediate classification losses to inject gradient in middle layers

3) FC layers are replaced by average pooling (fewer parameters)

Image classification

GoogLeNet (inception v1) [Szegedy et al, 2014]

Let's take a closer look at Inception modules…



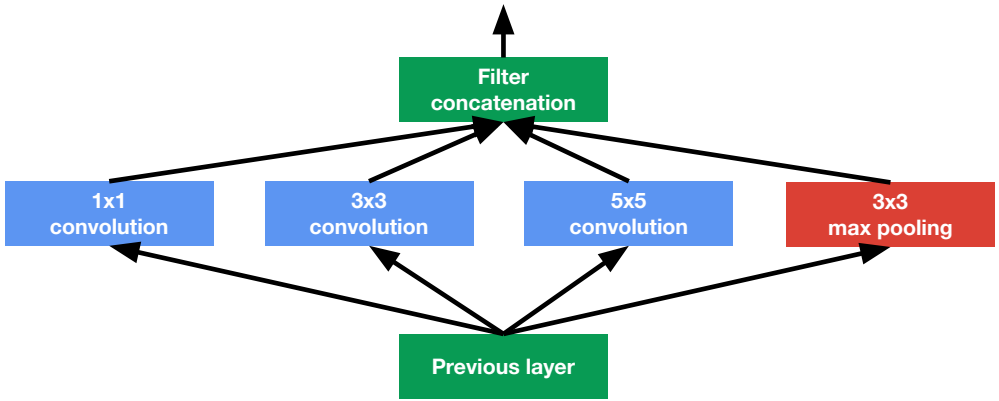
Choice for each layer:

- Convolution or pooling ?
- If convolution, what kernel size ?

Image classification

GoogLeNet (inception v1) [Szegedy et al, 2014]

Let's take a closer look at Inception modules…



Key idea:

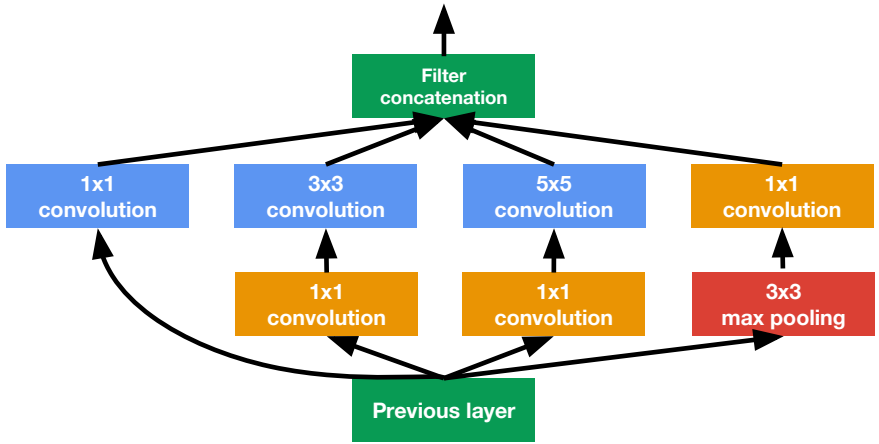
- Compute all in parallel
- Concatenate results
- Let the learning decide

Problem: this gives too many outputs and parameters

Image classification

GoogLeNet (inception v1) [Szegedy et al, 2014]

Let's take a closer look at Inception modules…

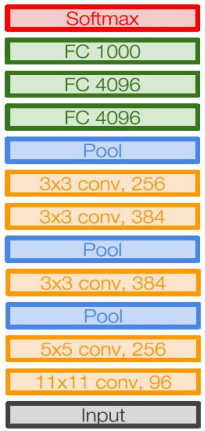


Solution: Reduce dimensionality using *bottleneck layers* composed of 1x1 convolutions

Image classification

Efficiency

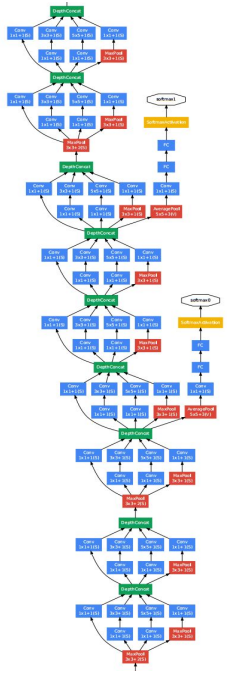
GoogLeNet (inception v1) [Szegedy et al, 2014]



AlexNet

8 layers

~62M parameters



GoogLeNet

22 layers

~5M parameters

GoogLeNet has 12x fewer parameters than AlexNet !

Image classification

ResNet [He et al, 2016]

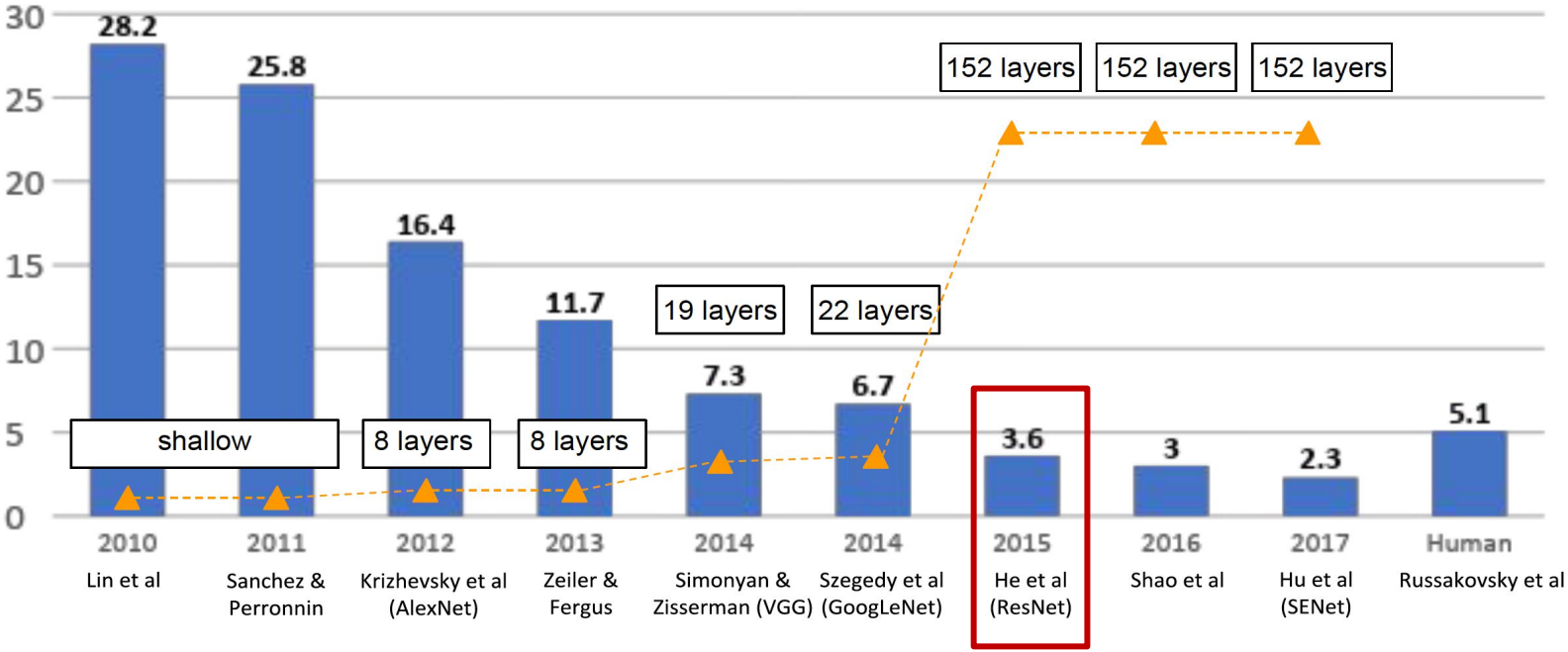
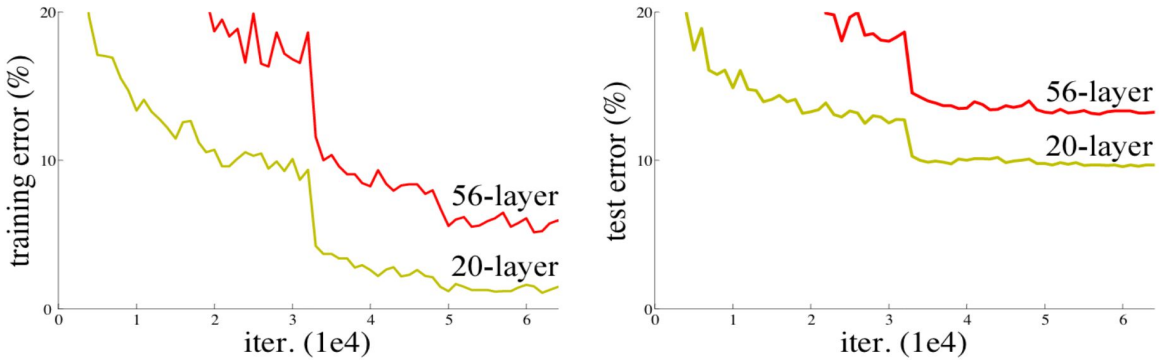


Image: http://cs231n.stanford.edu/slides/2018

Image classification

ResNet [He et al, 2016]

What happens if we make a standard CNN deeper ?



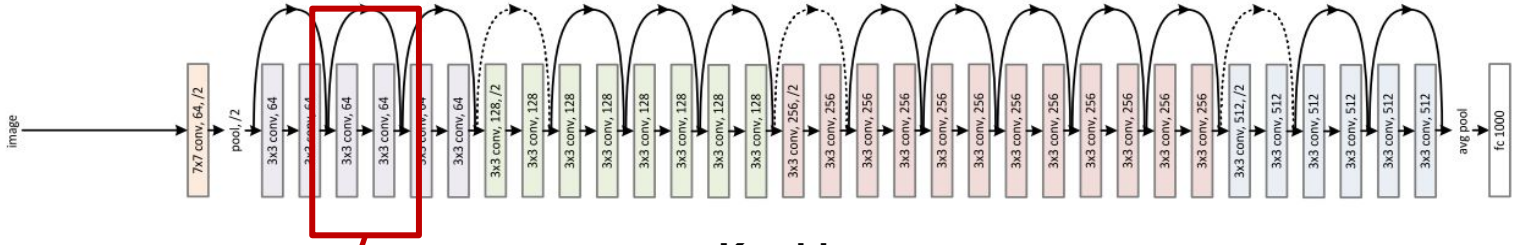
- Both training and test errors are larger
- This is not overfitting, this is an optimization problem (e.g., vanishing gradient)

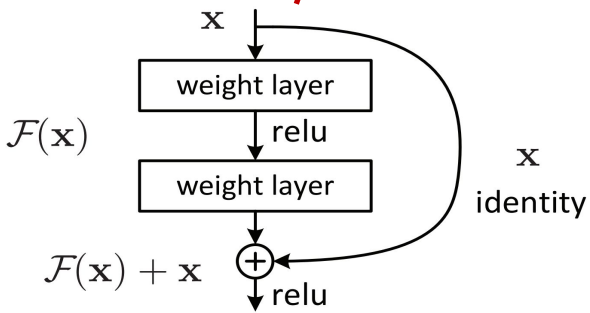
Image: He et al. "Deep residual learning for image recognition." CVPR, 2016.

Image classification

ResNet [He et al, 2016]

34-layer ResNet:





Key idea:

- Instead of computing the transformation, compute the residual required to have the transformation

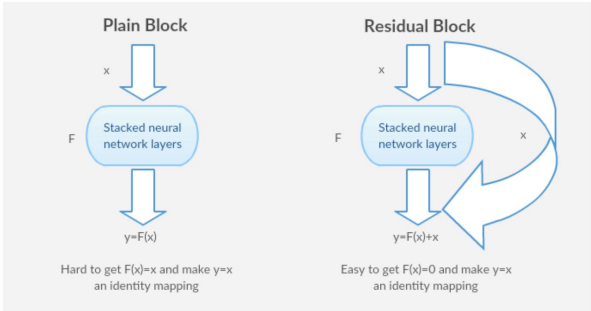
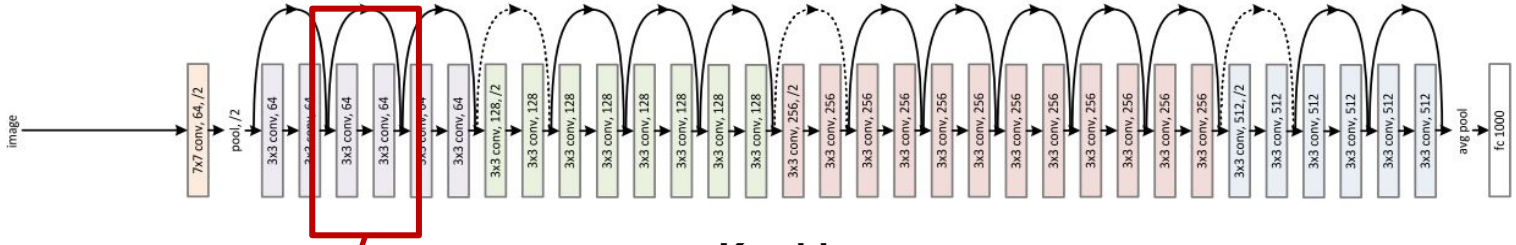


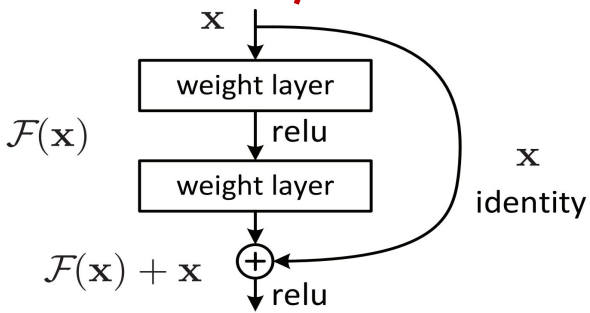
Image: He et al. "Deep residual learning for image recognition." CVPR, 2016.

Image classification

ResNet [He et al, 2016]

34-layer ResNet:





Key idea:

- Instead of computing the transformation, compute the residual required to have the transformation

Advantages:

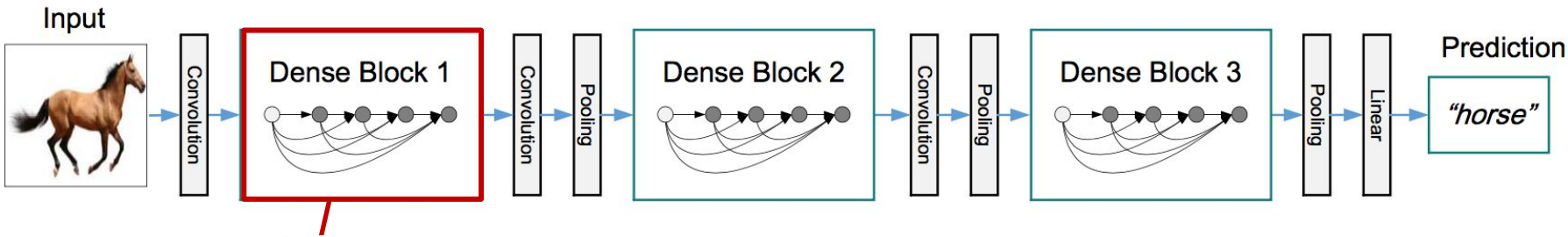
- The residual requires less information to model, so *possibly* easier to learn
- Residual connections help gradient flow during back-propagation
- Enables very deep networks (over 100 layers)

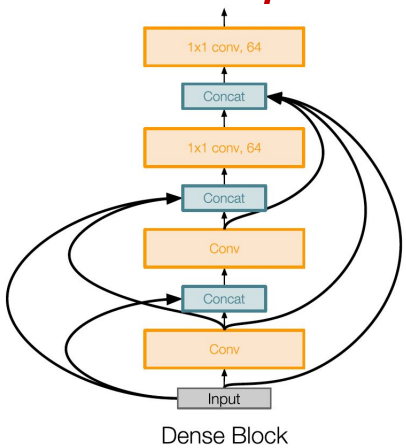
Image: He et al. "Deep residual learning for image recognition." CVPR, 2016.

Image classification

DenseNet [Huang et al, 2017]

Densely connected CNNs (DenseNet)





Dense Block

Key idea of dense blocks:

- Features computed in a layer are concatenated to the input of all subsequent layers in the block

Advantages:

- Efficient reuse of multiscale features
- Gradient can flow directly to each layer during back-propagation

Image: He et al. "Deep residual learning for image recognition." CVPR, 2016.

SEMANTIC SEGMENTATION

Semantic segmentation

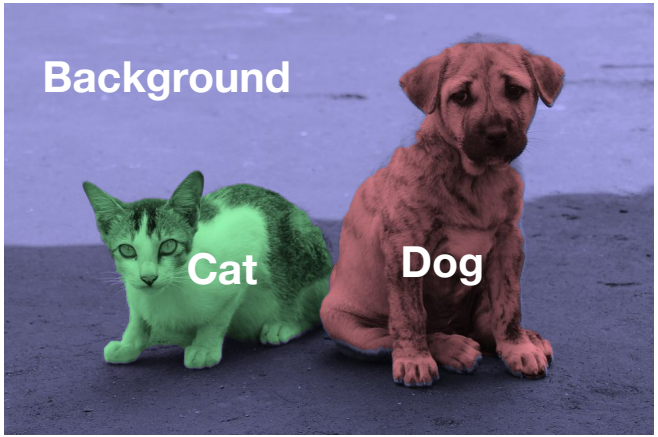


Image **Segmentation**

Goal: Assign the correct class label to each pixel of a given image

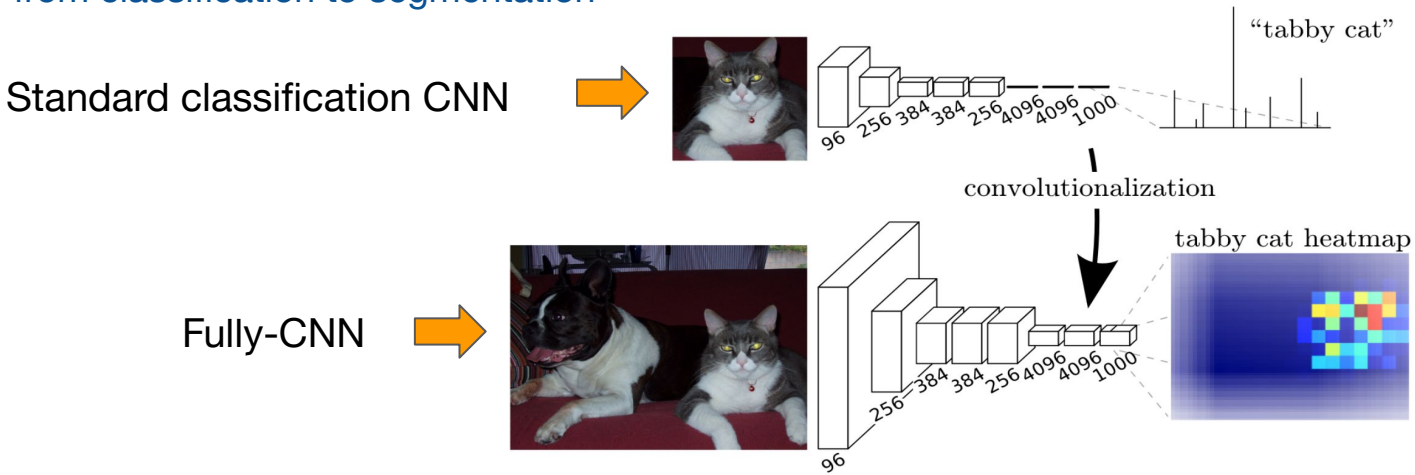
Can be seen as a dense and structured classification problem

Predictions are not independent

Possibly millions of pixels at the same time

Semantic segmentation

Fully-CNN : from classification to segmentation



Key ideas:

- Replace FC layers by 1x1 convolutions
- Treat network as a non-linear filter that can be applied to any image size

Problem: Produces very coarse segmentation maps

Solution: Add upsampling operations at the end of the network

Image: Long et al. "Fully convolutional networks for semantic segmentation." ICCV, 2015.

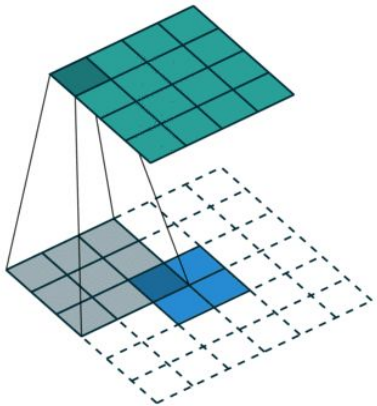
Semantic segmentation

Upsampling

Convolution as a matrix operation:

$$\begin{pmatrix} f_1 & f_2 & f_3 \end{pmatrix} * \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{pmatrix} = \begin{pmatrix} f_1 & f_2 & f_3 & 0 & 0 \\ 0 & f_1 & f_2 & f_3 & 0 \\ 0 & 0 & f_1 & f_2 & f_3 \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{pmatrix}$$

Transposed convolution:

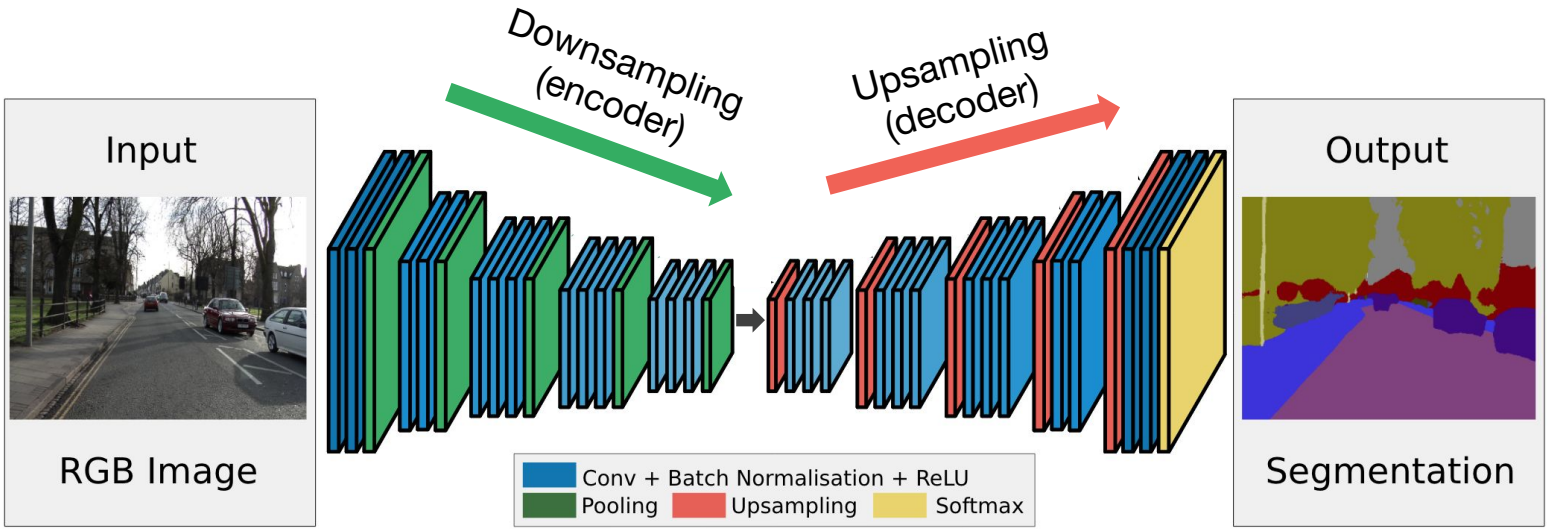


$$\begin{pmatrix} f_1 & f_2 & f_3 \end{pmatrix} *^T \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} f_1 & 0 & 0 \\ f_2 & f_1 & 0 \\ f_3 & f_2 & f_1 \\ 0 & f_3 & f_2 \\ 0 & 0 & f_3 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}$$

Image: http://deeplearning.net/software/theano/tutorial

Semantic segmentation

Encoder-decoder architecture



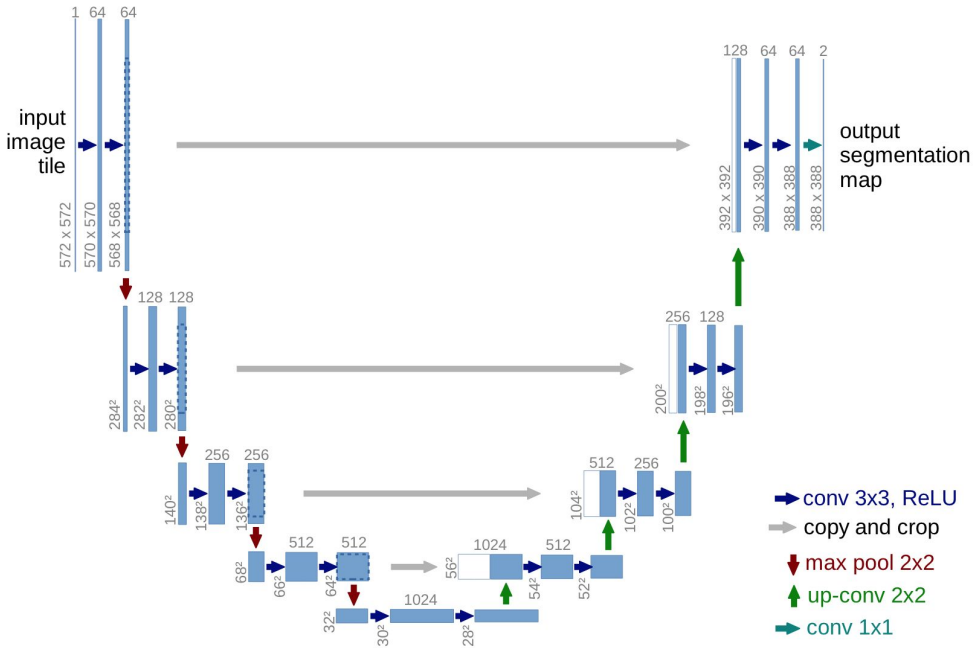
Problem: Spatial resolution is lost while downsampling

Solution: Add skip connections to copy high-resolution feature maps of the encoder to the decoder

Image adapted from:
Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." PAMI, 2017.

Semantic segmentation

UNet [Ronneberger et al, 2015]

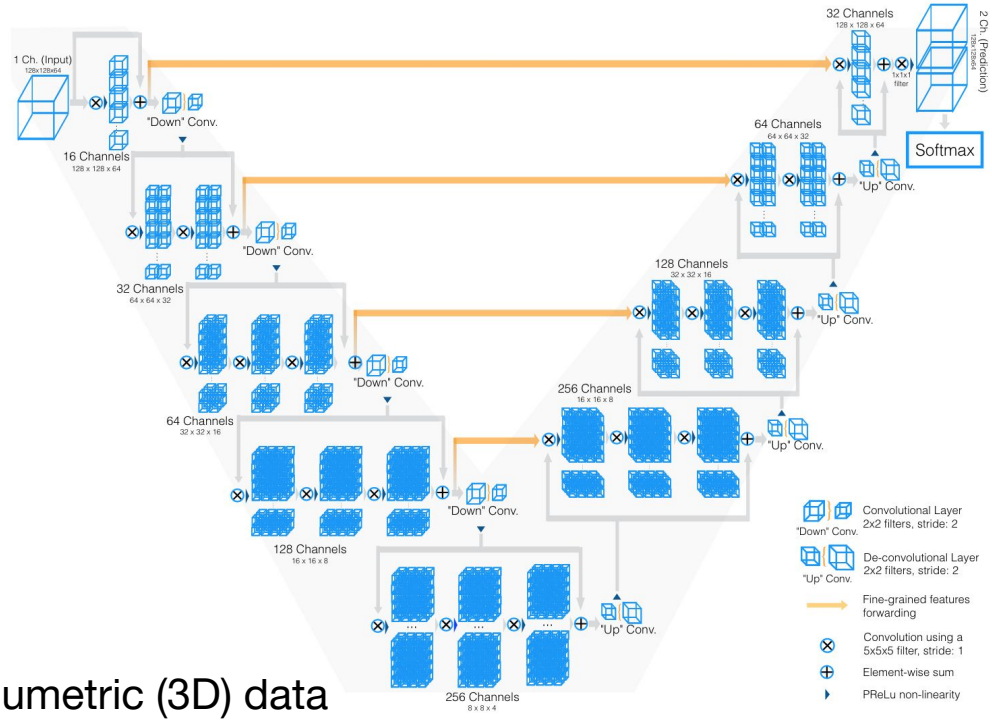


One of the most popular networks for medical image segmentation

Image: Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." MICCAI, 2015.

Semantic segmentation

VNet [Milletari et al, 2016]



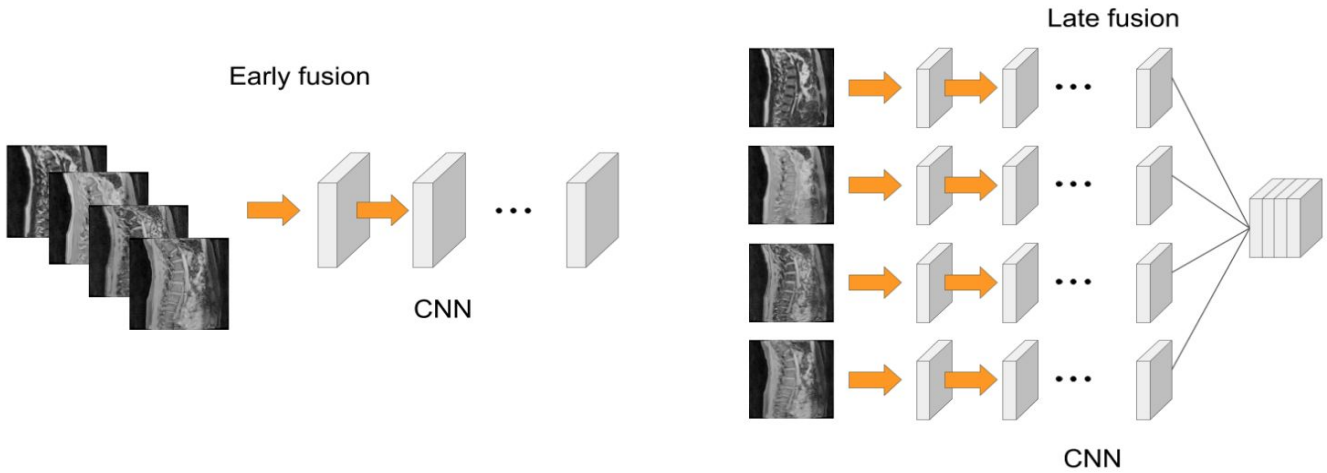
Extension of U-Net to volumetric (3D) data

Image: Milletari et al. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." International Conference on 3D Vision (3DV), 2016.

Semantic segmentation

Multi-modal segmentation

[Dolz et al, 2018]



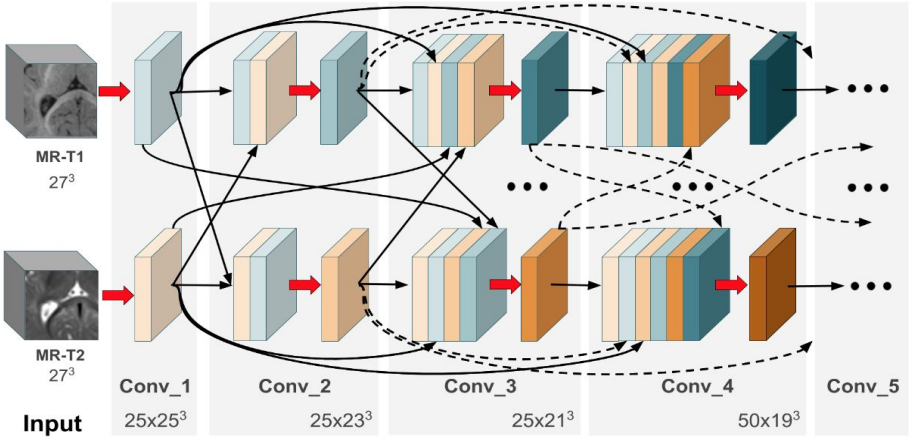
Problem: Ignores low or high level dependencies between information in different image modalities

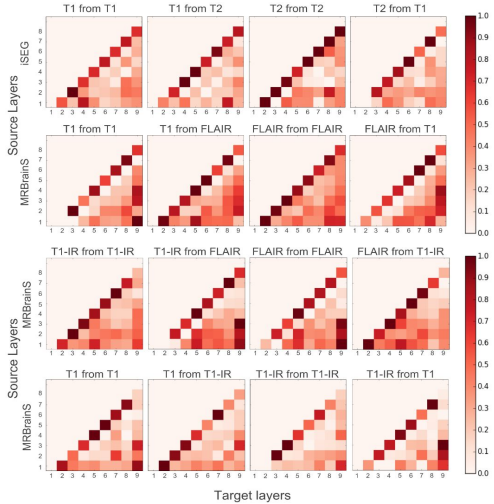
Image: Dolz et al. "IVD-Net: Intervertebral disc localization and segmentation in MRI with a multi-modal UNet." MICCAI Workshops, 2018.

Semantic segmentation

Multi-modal segmentation

HyperDense-Net [Dolz et al, 2018]





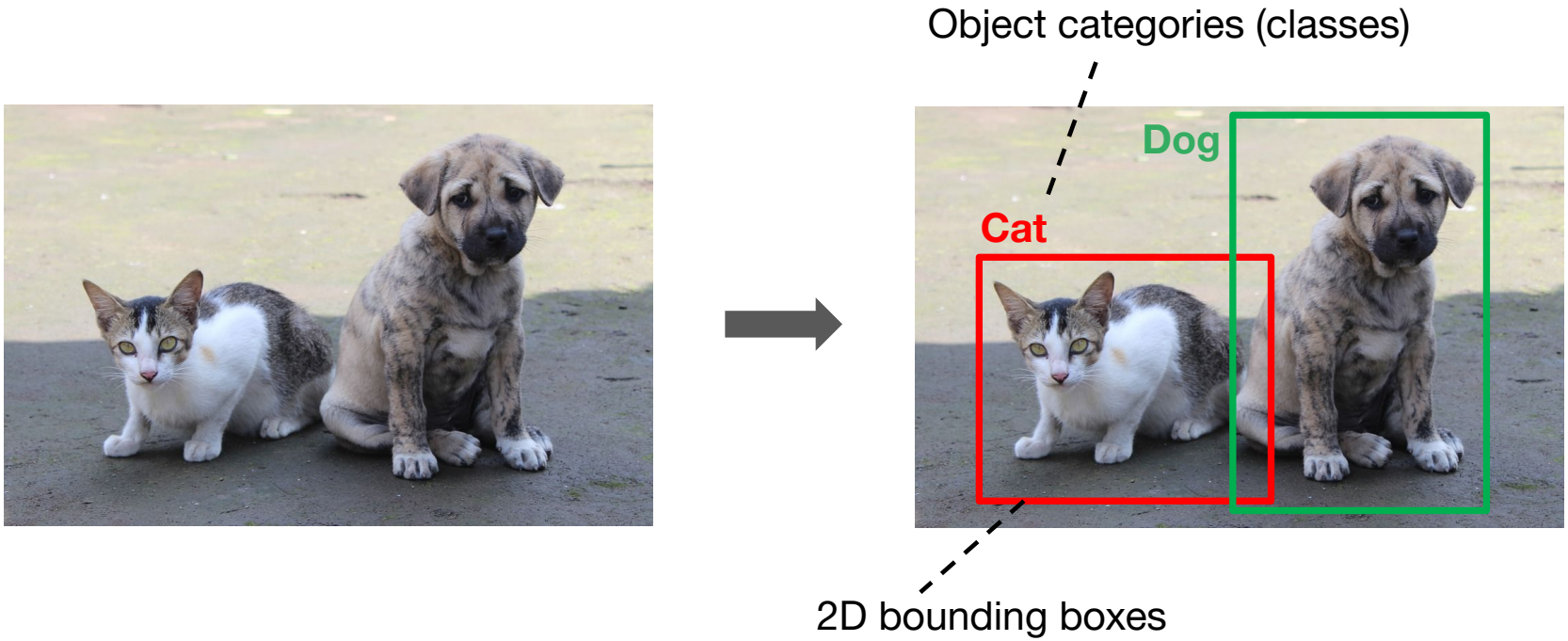
Key ideas:

- Dense connections across layers of different modalities
- Let training decide how to best combine information across modalities (see image on the right)

Image: Dolz et al. "HyperDense-Net: A hyper-densely connected CNN for multi-modal image segmentation." TMI, 2018.

OBJECT DETECTION

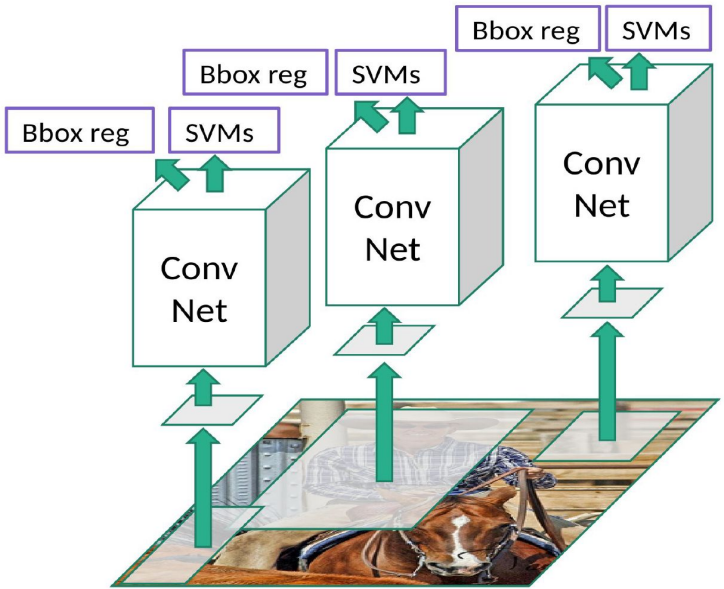
Object detection



Goal: Find objects (or people, animals) in a given image, and their location.

Object detection

R-CNN [Girshick et al, 2014]



5. Non-maxima suppression
(Removes redundant overlapping regions)

4. Classification & localization

3. Compute CNN features (AlexNet)

2. Warp proposals to fixed size

1. Extract region proposals (~2K)

- Uses the Selective Search method
- Finds regions likely to contain objects

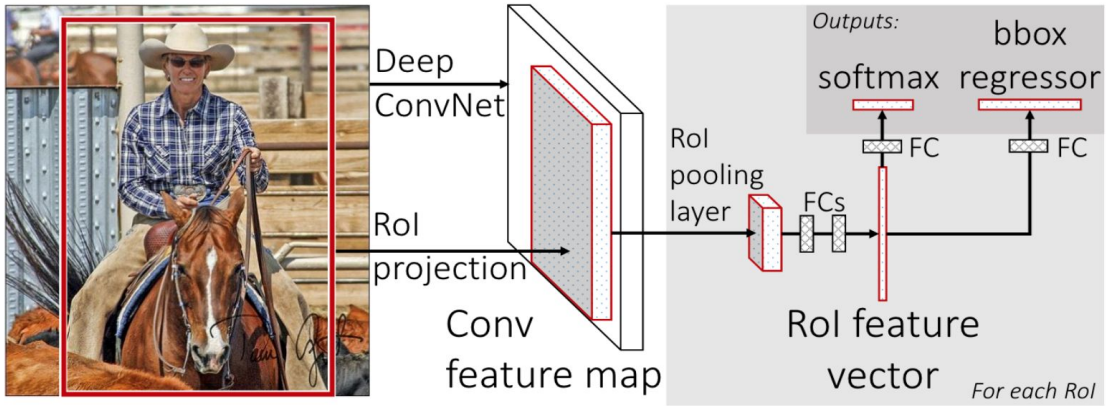
Problems:

- Multiple training stages: CNN, SVM, bounding box regressors
- Very slow (~53s per image)

Image: http://cs231n.stanford.edu/slides/2018

Object detection

Fast R-CNN [Girshick et al, 2015]



Key idea:

- Feed the input image to the CNN (instead of region proposals)
- Enables sharing CNN computations between different proposals

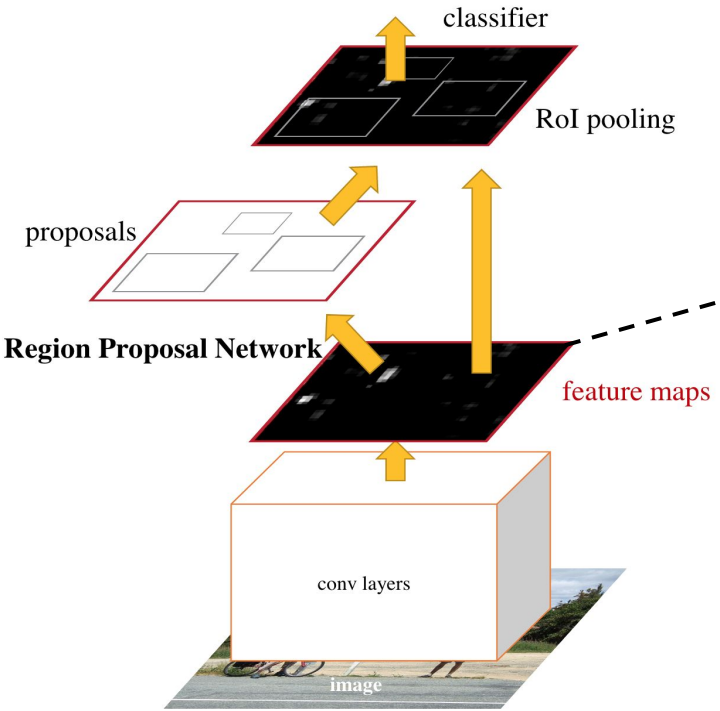
Problem: Still requires extracting proposals in a separate step

How to avoid this ?

Image: Girshick, Ross. "Fast R-CNN." ICCV. 2015.

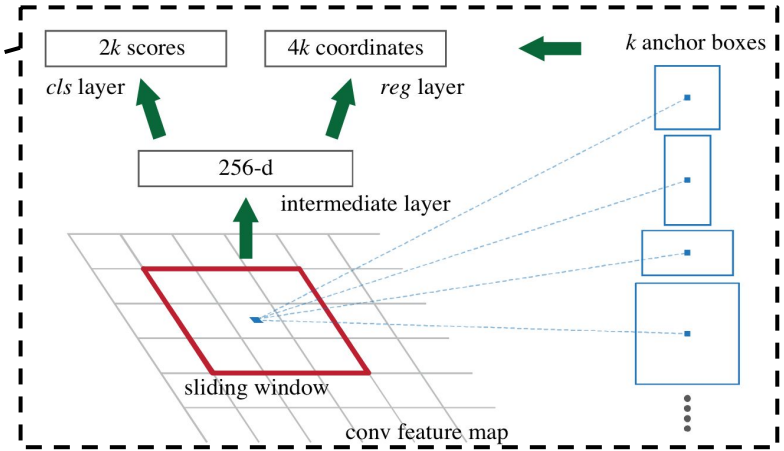
Object detection

Faster R-CNN [Ren et al, 2015]



Key ideas:

- Do the region proposal within the network
- Use anchor boxes with different scales and aspect ratios to generate RoI proposals
- Multitask loss for box regression and objectness



Images: Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.