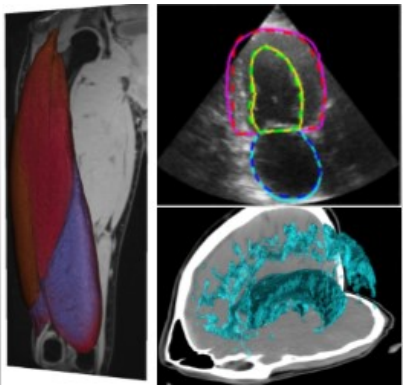
LABEX PRIMES
UNIVERSITÉ DE LYON

cnrs

80 years of building new worlds through knowledge

Deep learning for medical imaging school

April 15—19 2019, Campus de la Doua, Lyon



# Advanced concepts in deep learning

Pierre-Marc Jodoin

and

Christian Desrosiers

UNIVERSITÉ DE SHERBROOKE

ÉTS
Le génie pour l'industrie

# Outline

1. **Refresher on CNNs**
2. **More on optimization**
3. **Problems and architectures**
    i. Image classification
    ii. Semantic segmentation
    iii. Object detection
    iv. Instance segmentation
4. **Further topics**
    i. Handling spatial variance
    ii. Convolutional autoencoders
    iii. Graph convolutions
    iv. Recurrent neural networks

# CNN REFRESHER

# Convolutional neural networks

CNN architecture

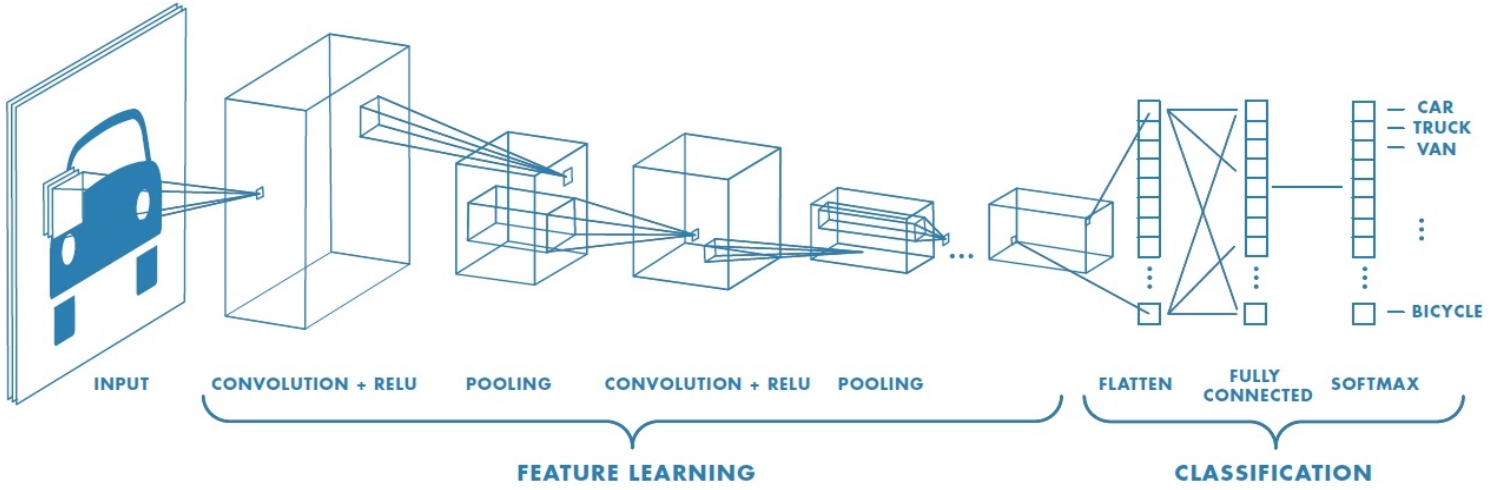



Image: https://www.mathworks.com

Human visual cortex

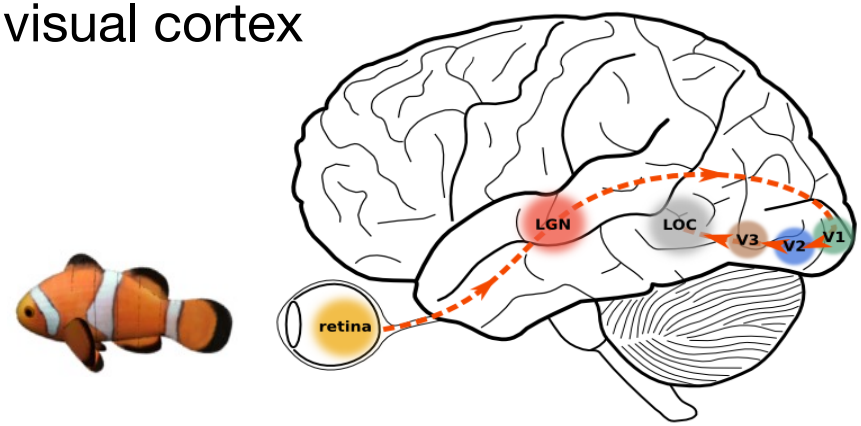
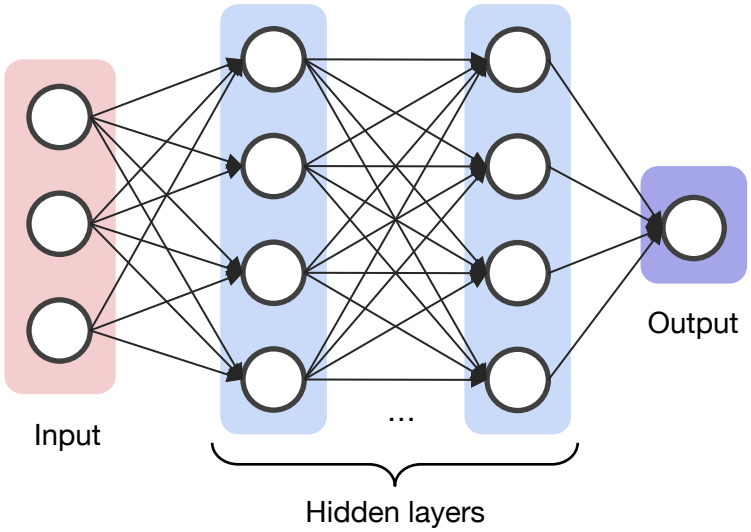



Image: https://neuwritesd.files.wordpress.com/2015/10/visual_stream_small.png

# Convolutional neural networks

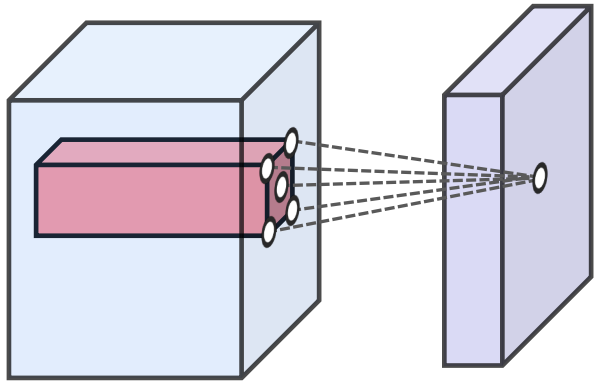
Multi-layer perception (MLP)




Many parameters

Limited depth

CNN



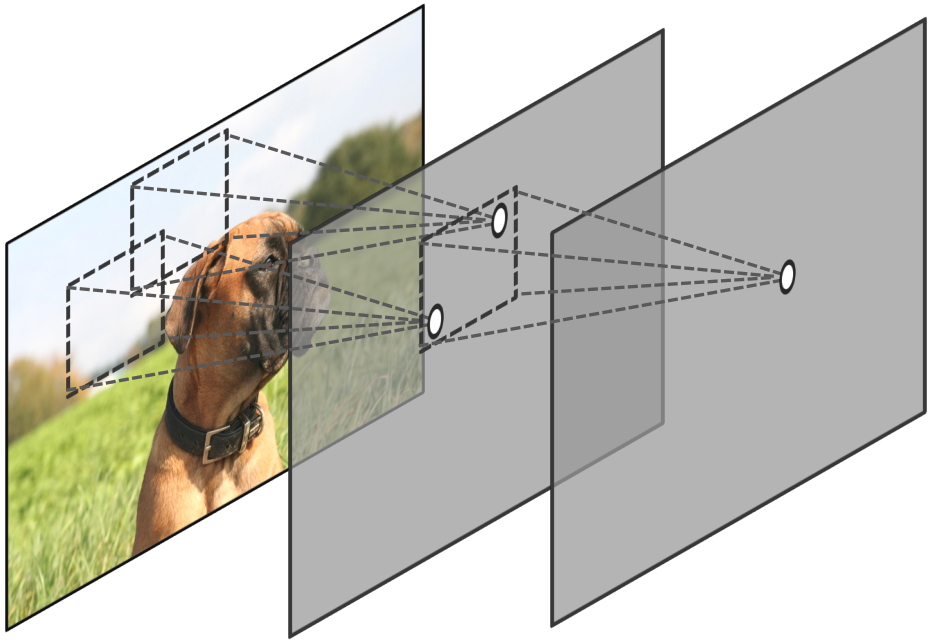
- Shared parameters
- Local connectivity

Less parameters

Deep architectures possible

# Convolutional neural networks

**Receptive field of a neuron**



Region in the input space which can affect the neuron's output

A large receptive is necessary to capture spatial context

However, it increases the number of parameters

# Convolutional neural networks

**Building blocks**

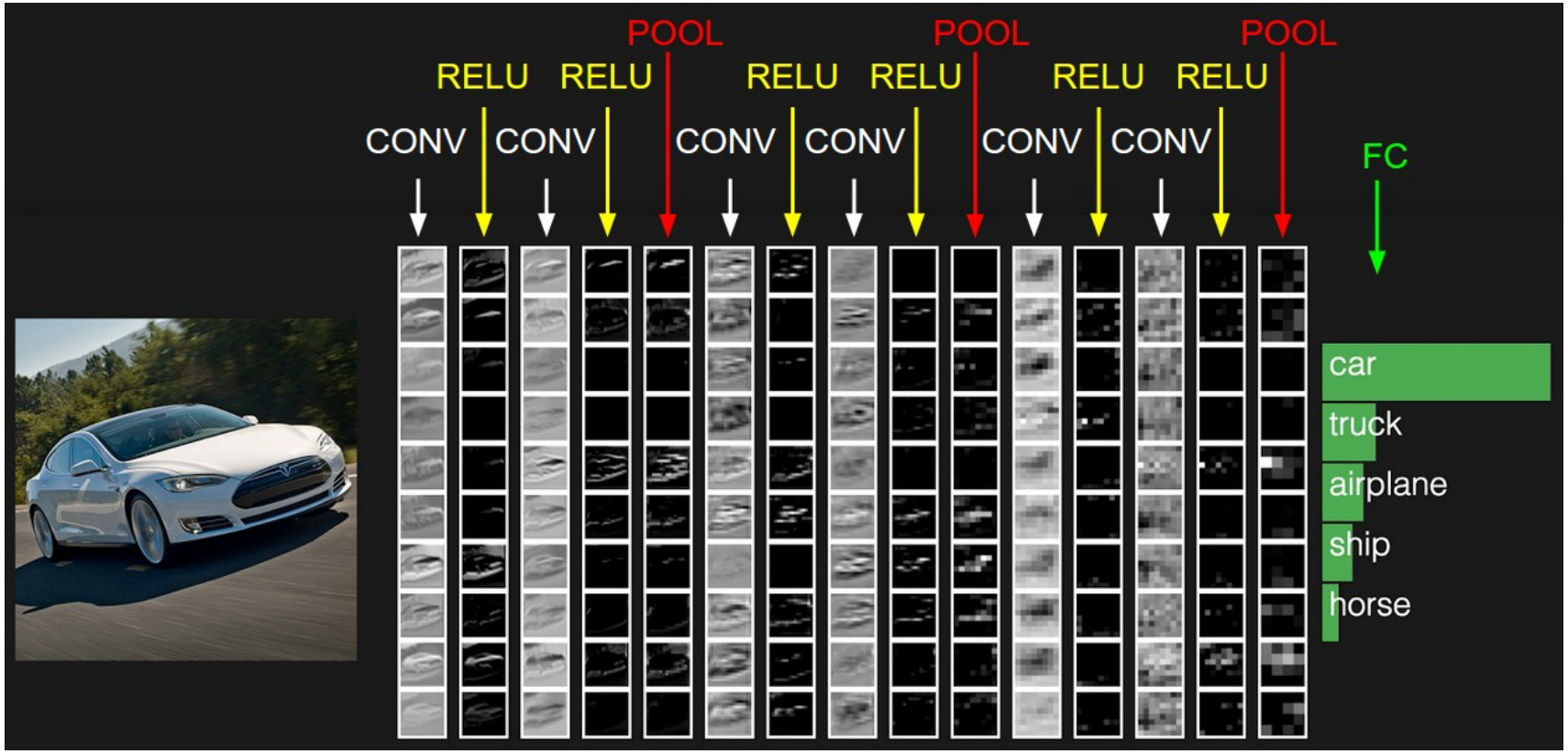
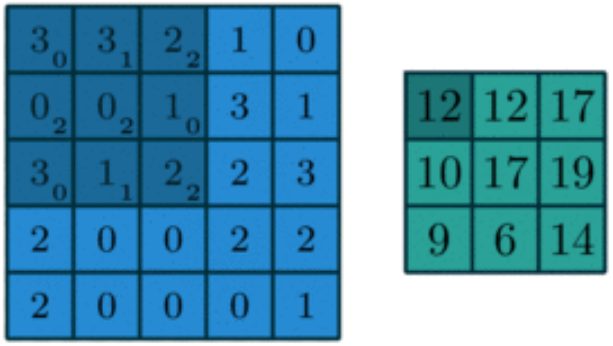



Image: http://cs231n.github.io

# Convolutional neural networks

**Convolution**




$$\begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} * \begin{pmatrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 12 & 12 & 17 \\ 10 & 17 & 19 \\ 9 & 6 & 14 \end{pmatrix}$$

Kernel (3 x 3)

Feature map

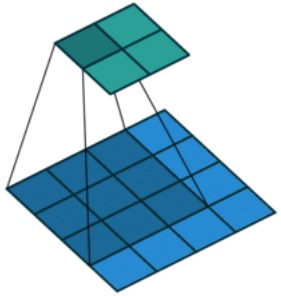
Output

General formulation (2D)

$$G = h * F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i-u, j-v]$$

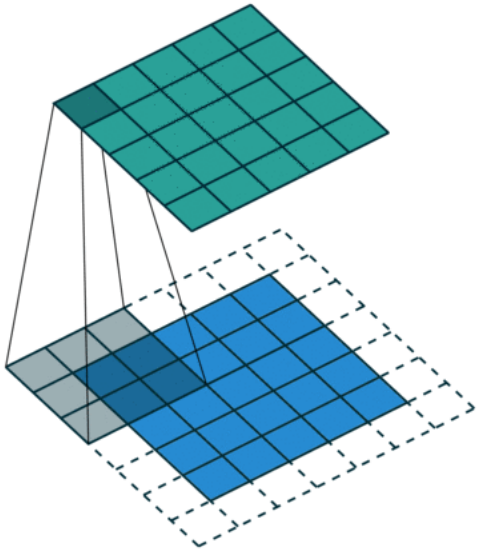
# Convolutional neural networks

**Convolution padding**



No padding
(unit stride)

Reduced dimension
along border



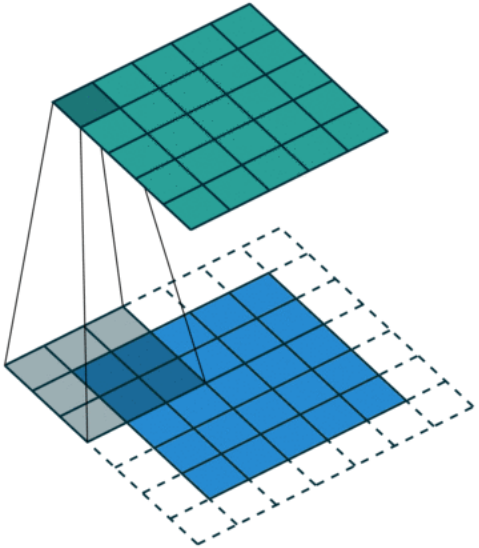
Padding
(unit stride)

Preserved spatial
dimension

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

# Convolutional neural networks

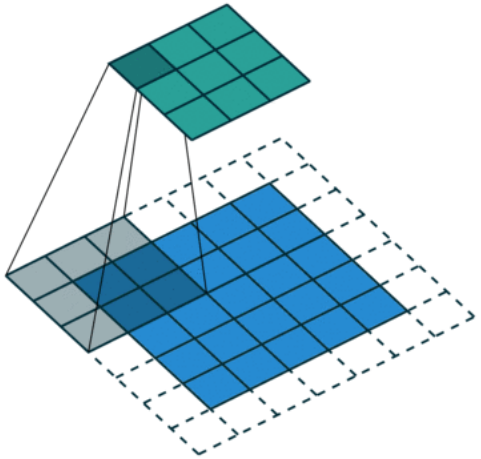
**Convolution stride**



Unit stride
(padding)

Larger spatial dimension
(more computations/memory)

Overlapping receptive fields



Stride = 2
(padding)

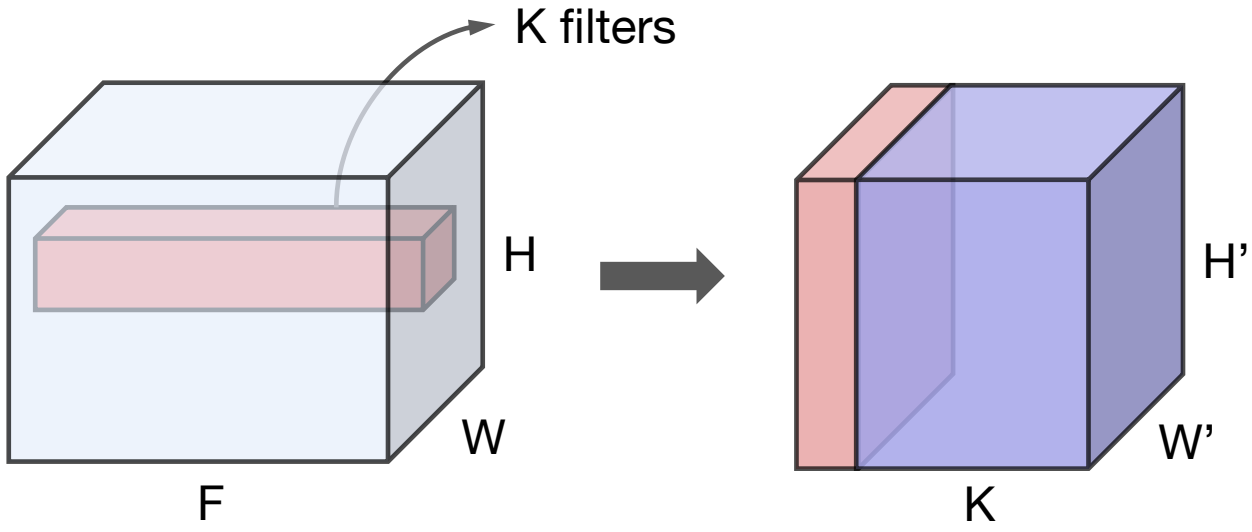
Reduced spatial dimension
(less computations/memory)

Reduced overlapping

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

# Convolutional neural networks

**Convolution dimension**



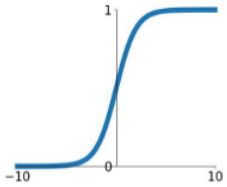

- Kernel depth is same as the number of input channels (e.g., 3 in this example)
  - Kernel of a 2D convolution is a 3D tensor
  - Kernel of a 3D convolution is a 4D tensor
- Number of output channels corresponds to the number of filters

# Convolutional neural networks

**Activation function**

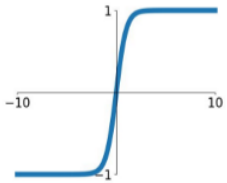
**Sigmoid**

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



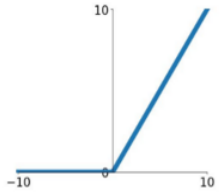

**tanh**

$$\tanh(x)$$




**ReLU**

$$\max(0, x)$$

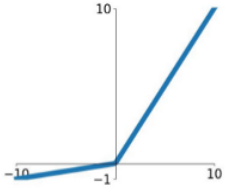



Reduces the vanishing gradient problem

Ensures that neurons are always activated

**Leaky ReLU**

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

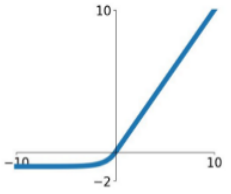



**Maxout**

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

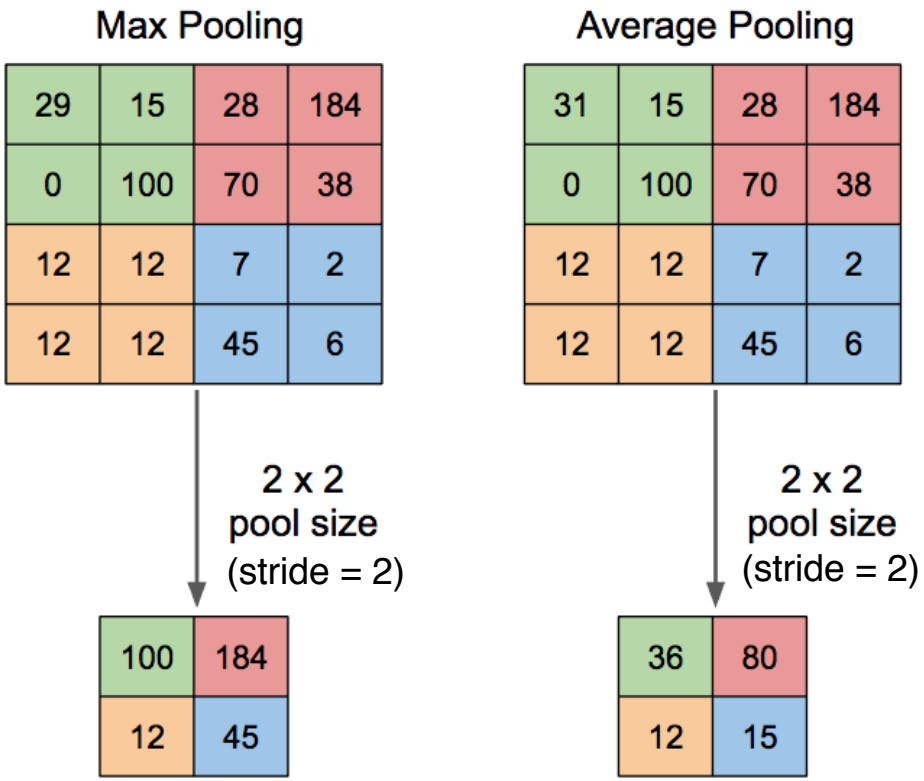
**ELU**

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



# Convolutional neural networks

## Pooling




**Goals**

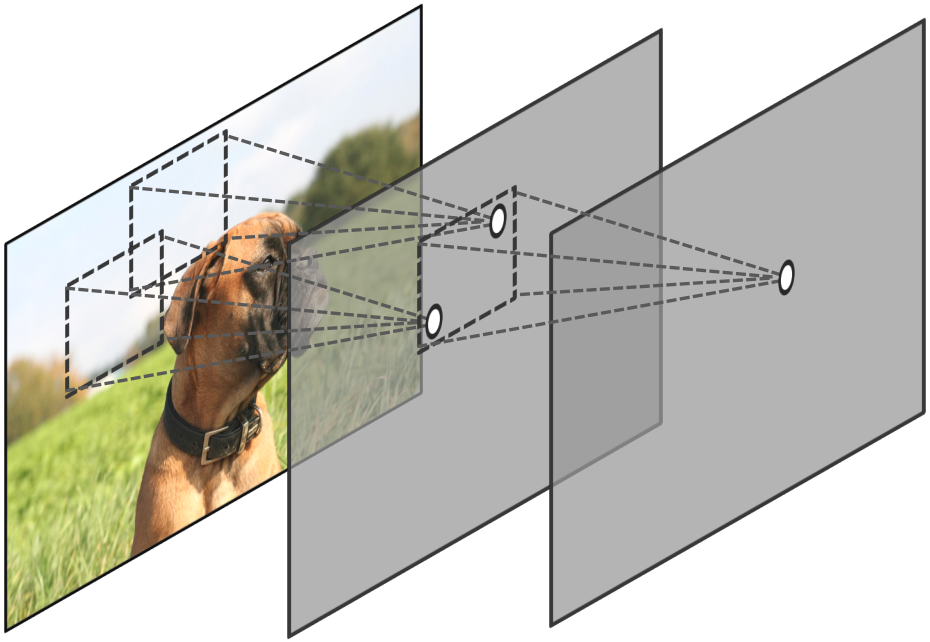
- Reduce the spatial resolution of feature maps
- Lower memory and computation requirements
- Provide partial invariance to position, scale and rotation

Pooling is typically done separately for each feature map (channel)

Images: https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/

# Convolutional neural networks

**Receptive field of a neuron (revisited)**

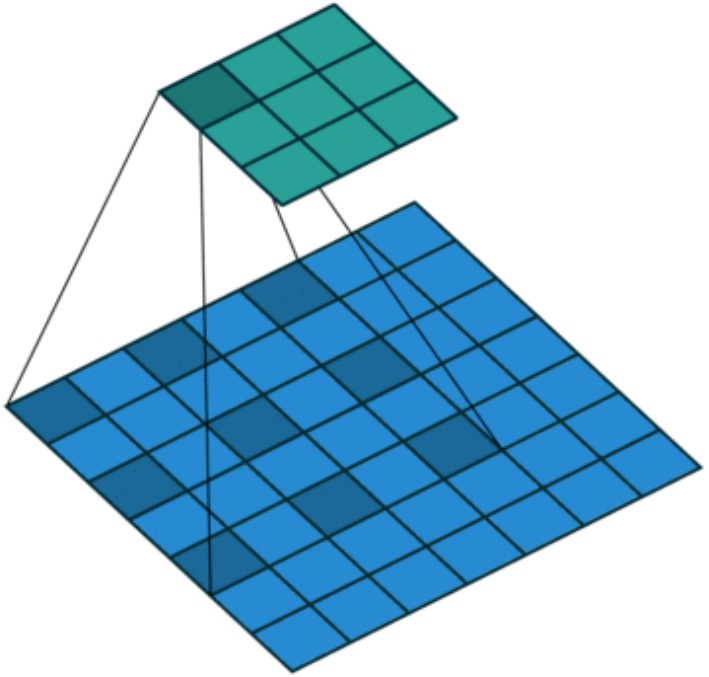


The receptive field of neurons increases when going deeper in the network.

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

# Convolutional neural networks

**Dilated convolution (*à trous*)**



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

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

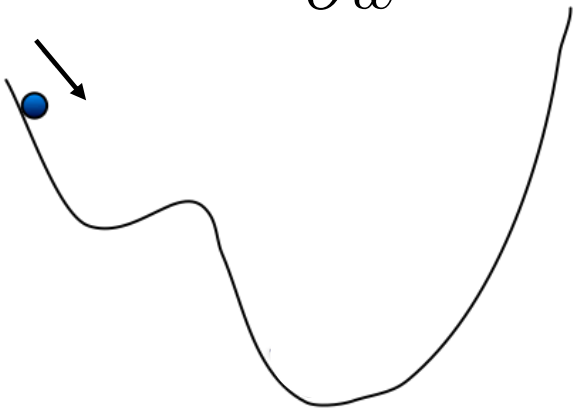
# MORE ON OPTIMIZATION

# Momentum in optimization

$$w_t = w_{t-1} - v_t$$

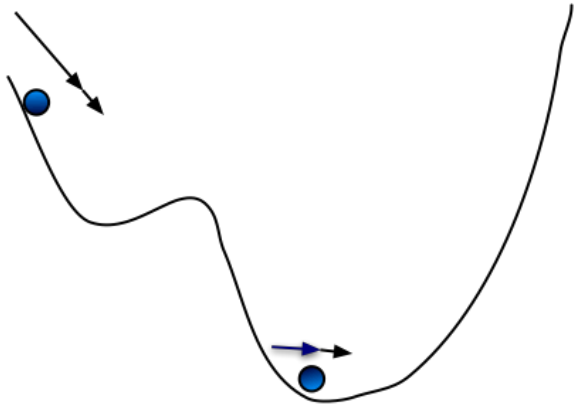
Standard gradient

$$v_t = \eta \frac{\partial \mathcal{L}}{\partial w}$$



With momentum

$$v_t = \beta v_{t-1} + \eta \frac{\partial \mathcal{L}}{\partial w}$$



**Benefits**:

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

# Momentum in optimization

**Adaptive moment estimation (Adam)**

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

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

$$\lim_{t\to\infty} \gamma_j^t = 0,\ j \in \{1,2\}$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

Reduce step size if large variance

# Momentum in optimization

**Adaptive moment estimation (Adam)**

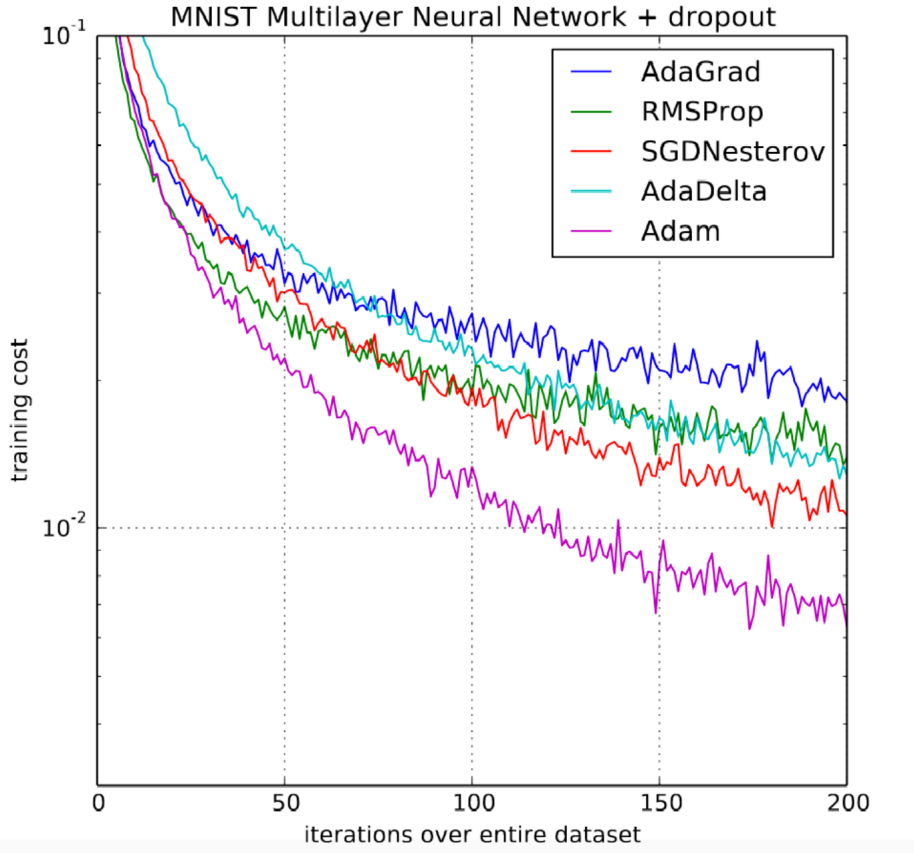



Image: Kingma et al. "Adam: A method for stochastic optimization." arXiv, 2014

# Batch normalization

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

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

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

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

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

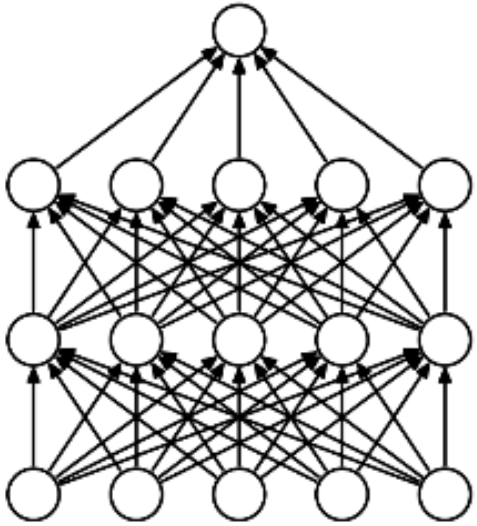
**Key ideas:**

- 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

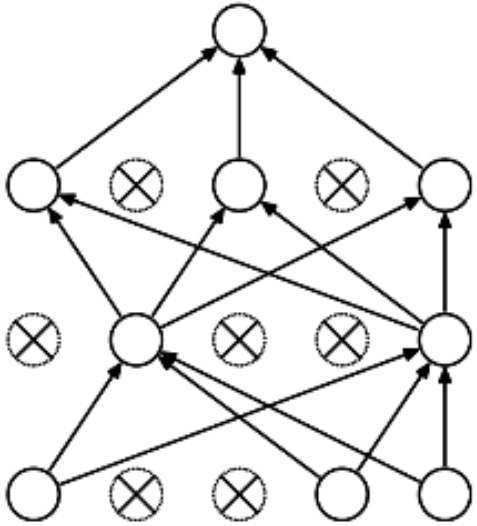


Image: Ioffe et al. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv, 2015.

# Dropout regularization



Standard network



With dropout

Tip: higher ***p*** for layers with more parameters

**Key ideas**

- During training drop neurons randomly with probability ***p***
- Avoid co-adaptation of neurons: each neuron should work independently of others
- ~ Implicit learning of network ensemble

Image: Srivastava et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

# IMAGE 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**

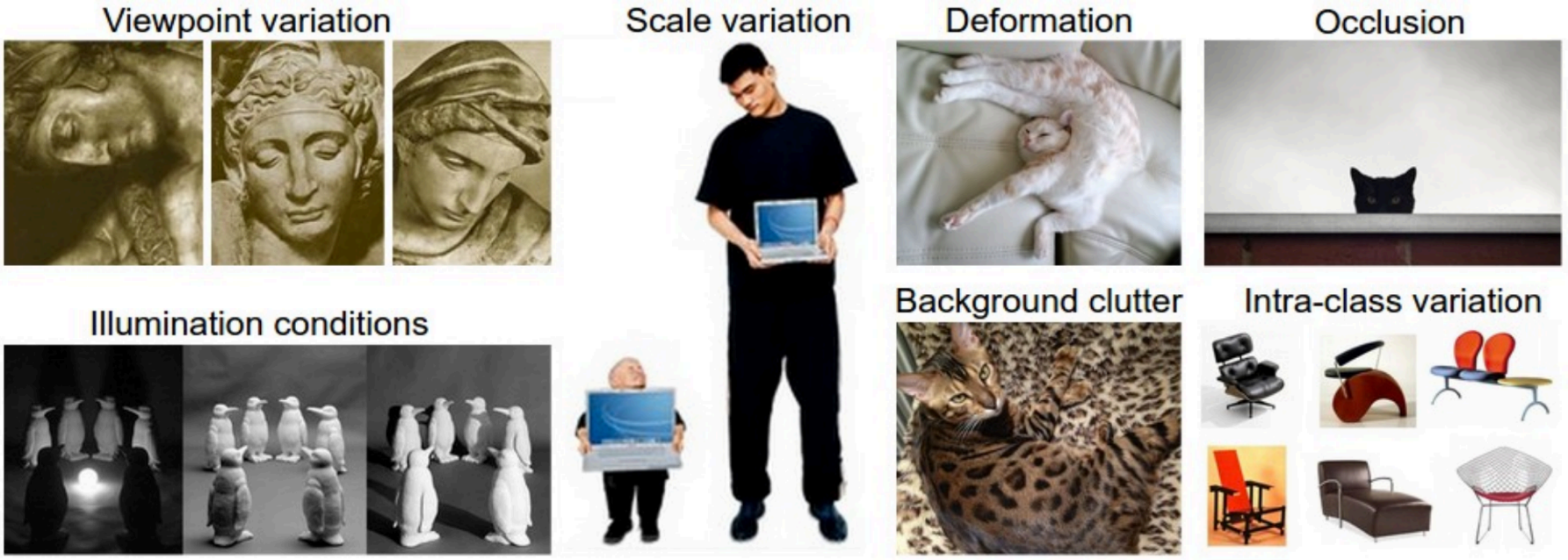
What humans see

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 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

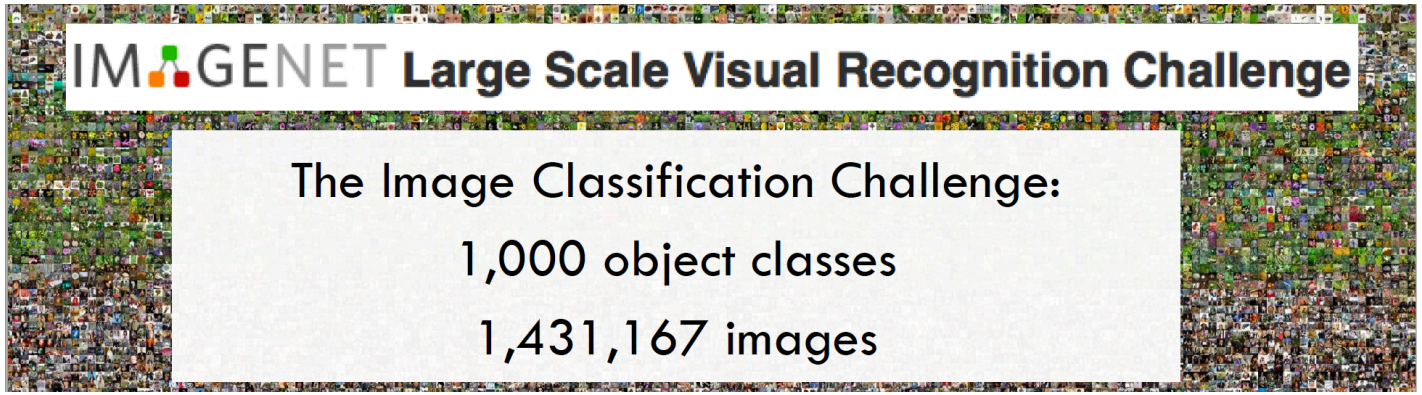
**Other challenges**




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

Image: http://cs231n.github.io

# Image classification



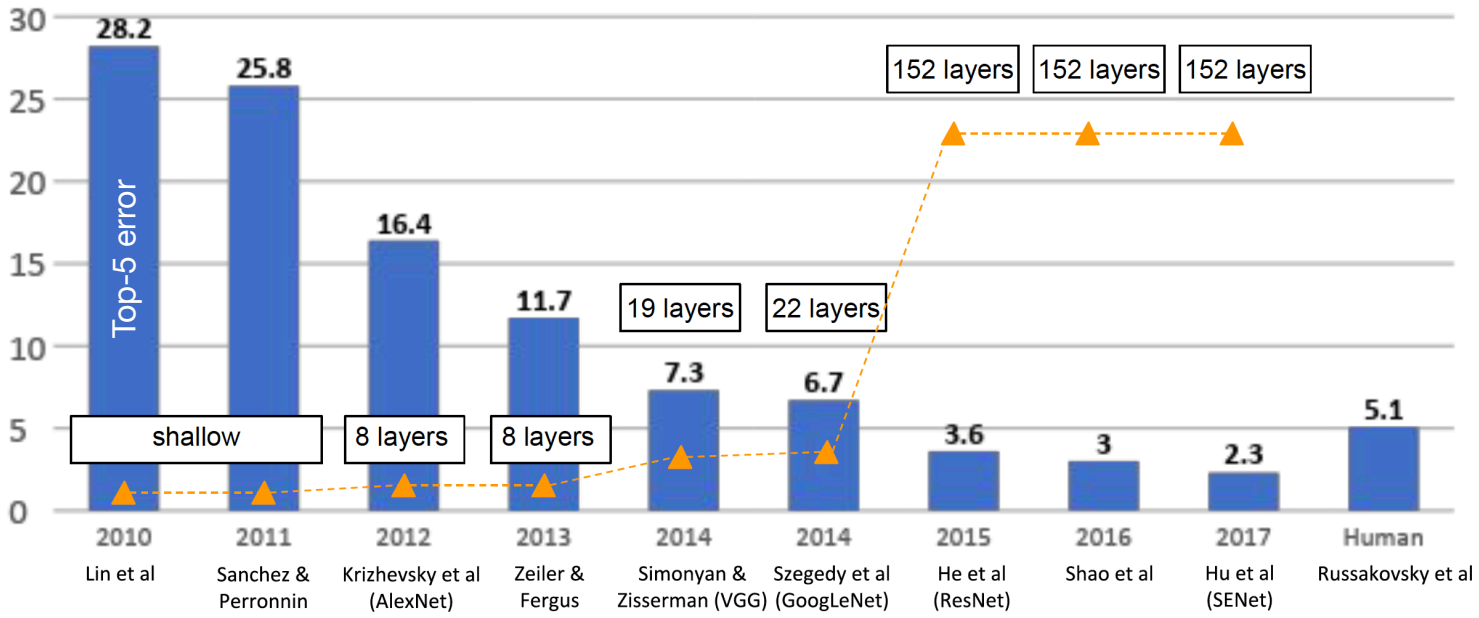




Image: http://cs231n.stanford.edu/slides/2018

# Image classification

**AlexNet** **[Krizhevsky et al, 2012]**

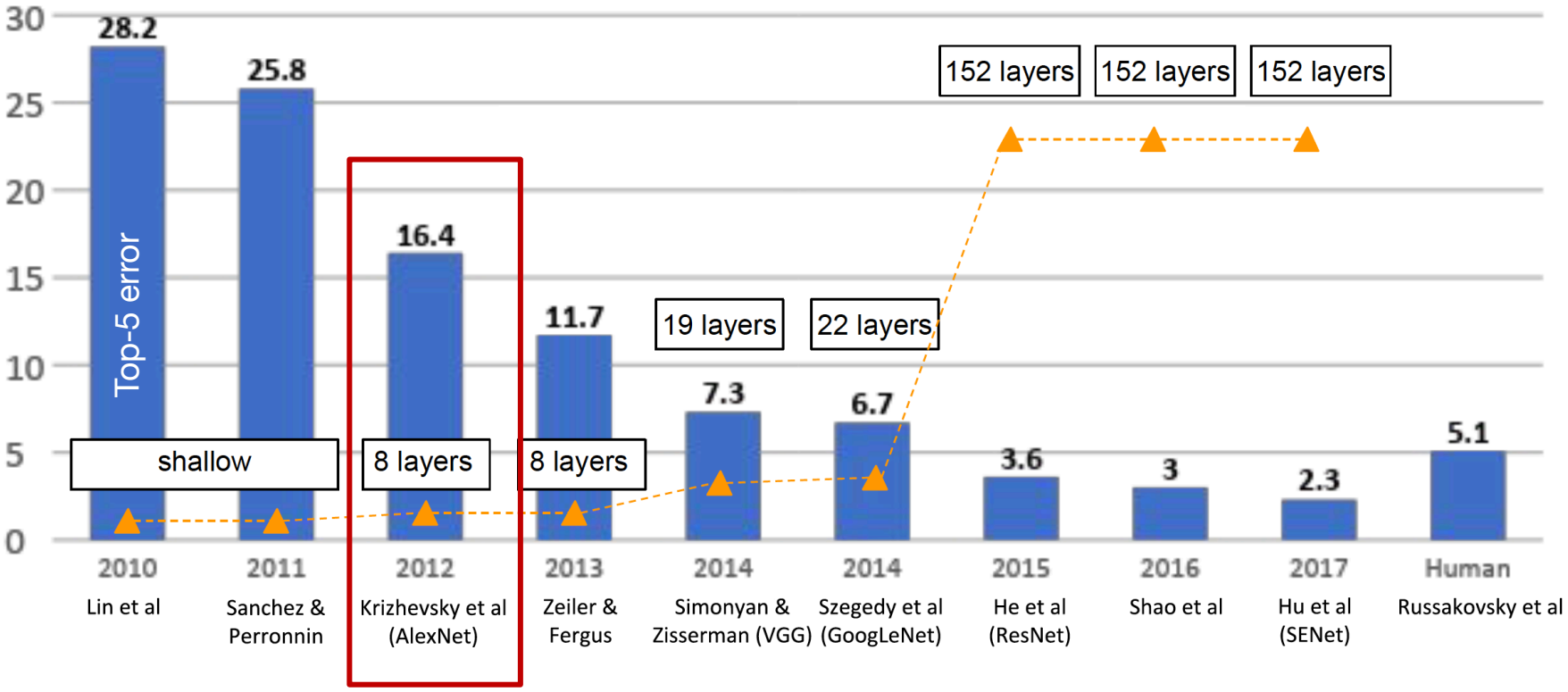
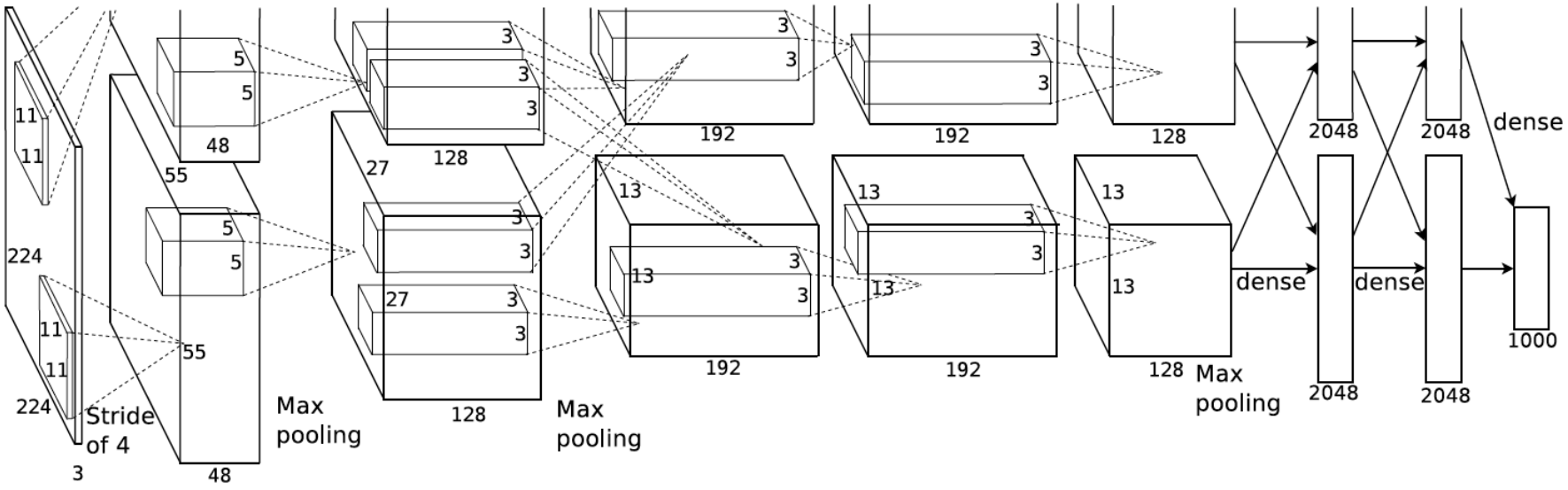



Image: http://cs231n.stanford.edu/slides/2018

# Image classification

**AlexNet** **[Krizhevsky et al, 2012]**




- First model to perform well on the ImageNet dataset (~11% lower error than runner up)
- Combined techniques used in today's architectures, like ReLU, data augmentation and dropout
- Used GPUs for training
- Largely responsible for the deep learning revolution in computer vision



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

# Image classification

**ZF Net** **[Zeiler and Fergus, 2014]**

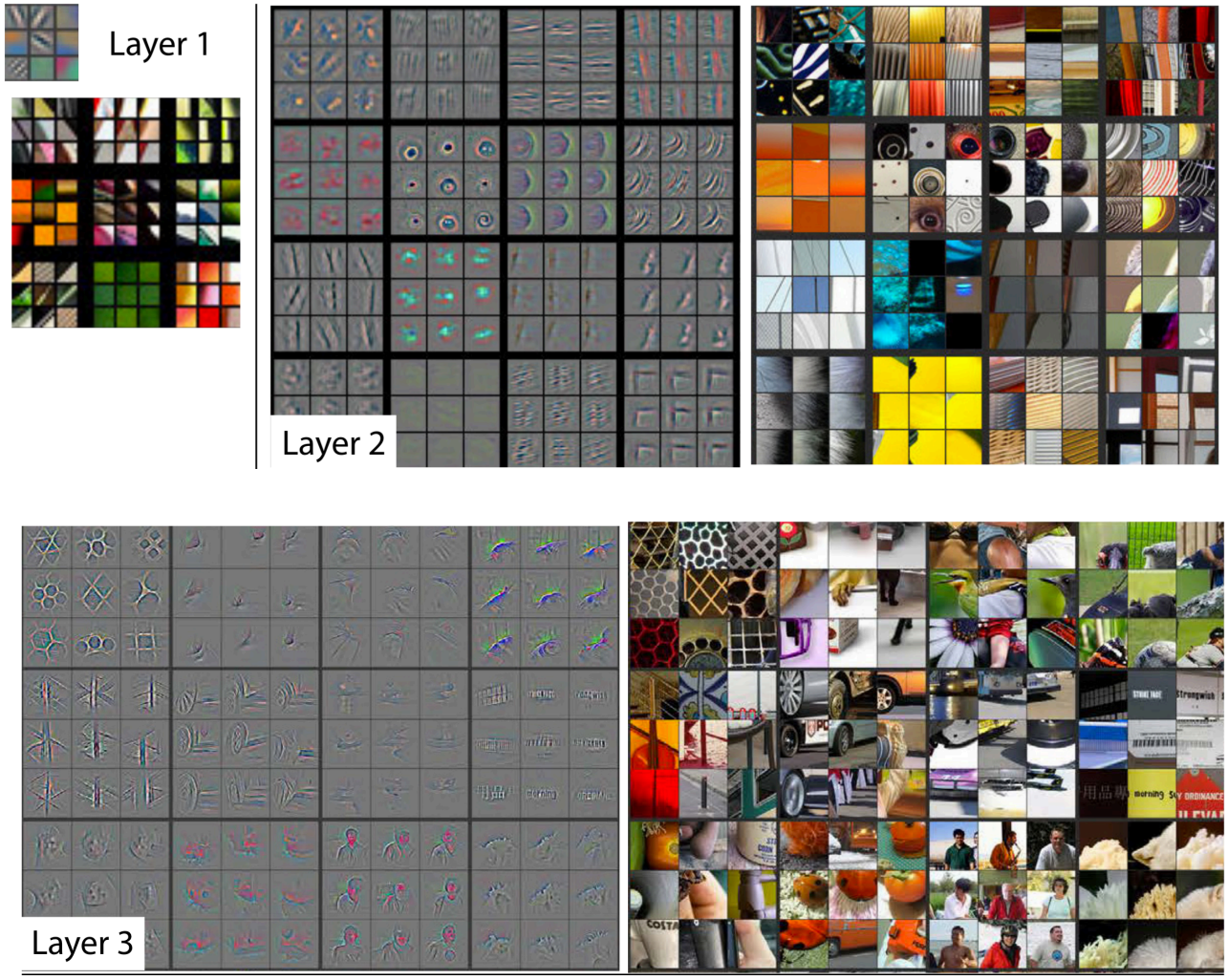



Image: Zeiler and Fergus. "Visualizing and understanding convolutional networks." ECCV, 2014.

# Image classification

**VGG [Simonyan and Zisserman, 2014]**

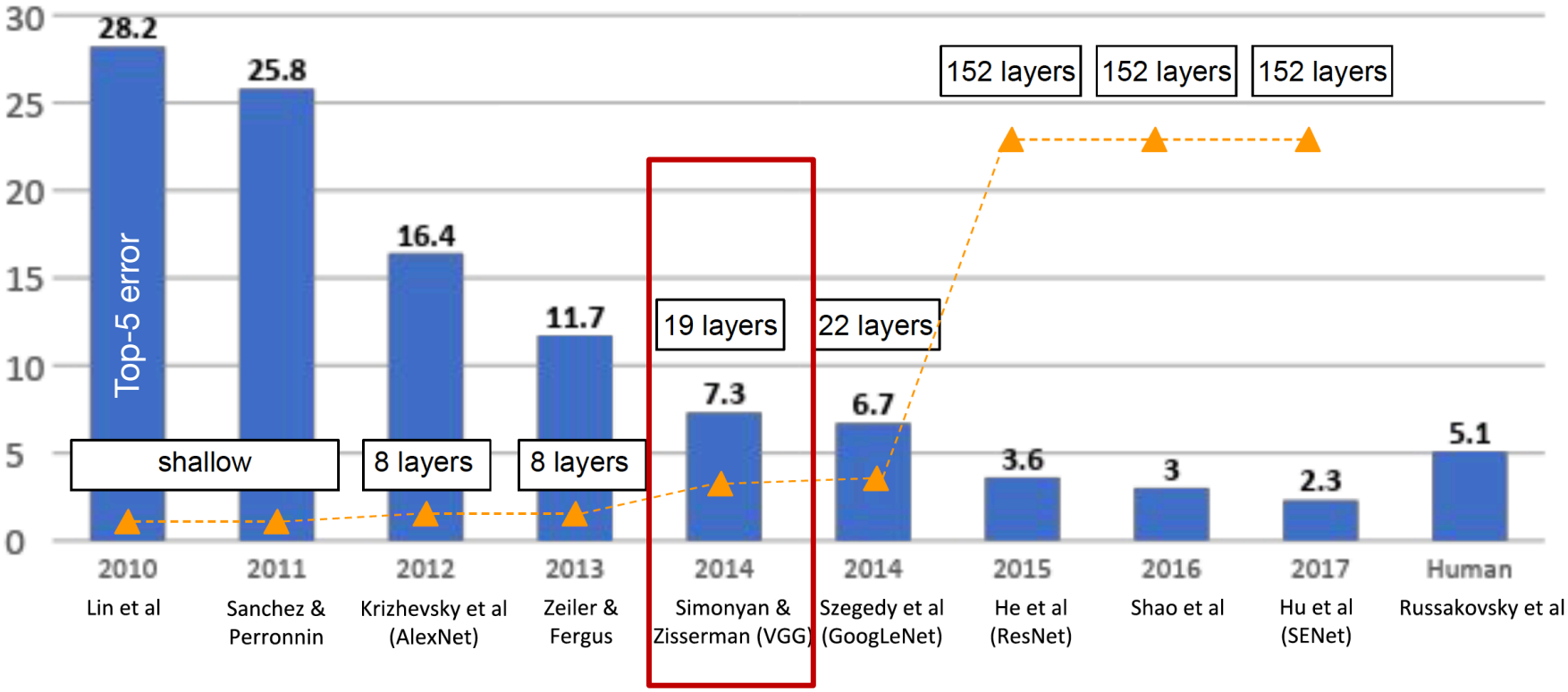



Image: http://cs231n.stanford.edu/slides/2018

# 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]**

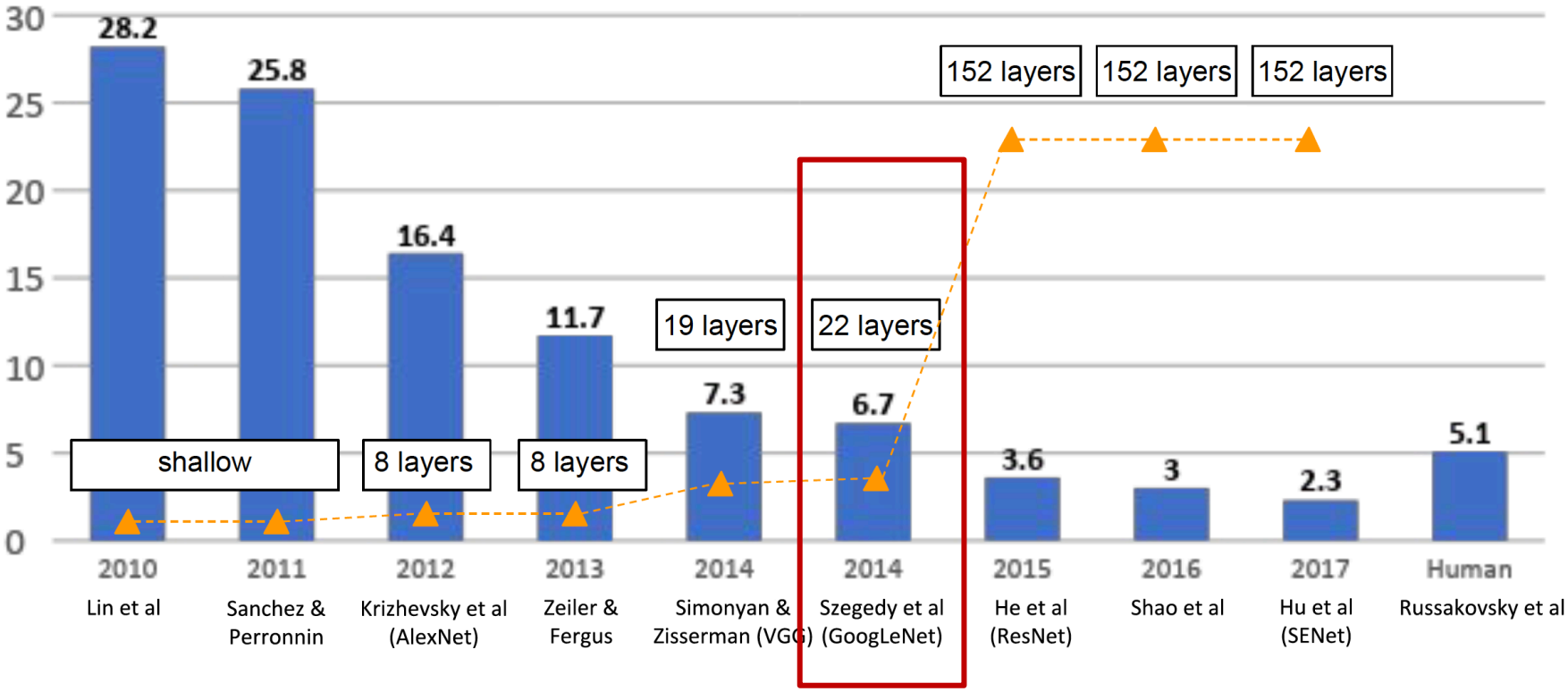
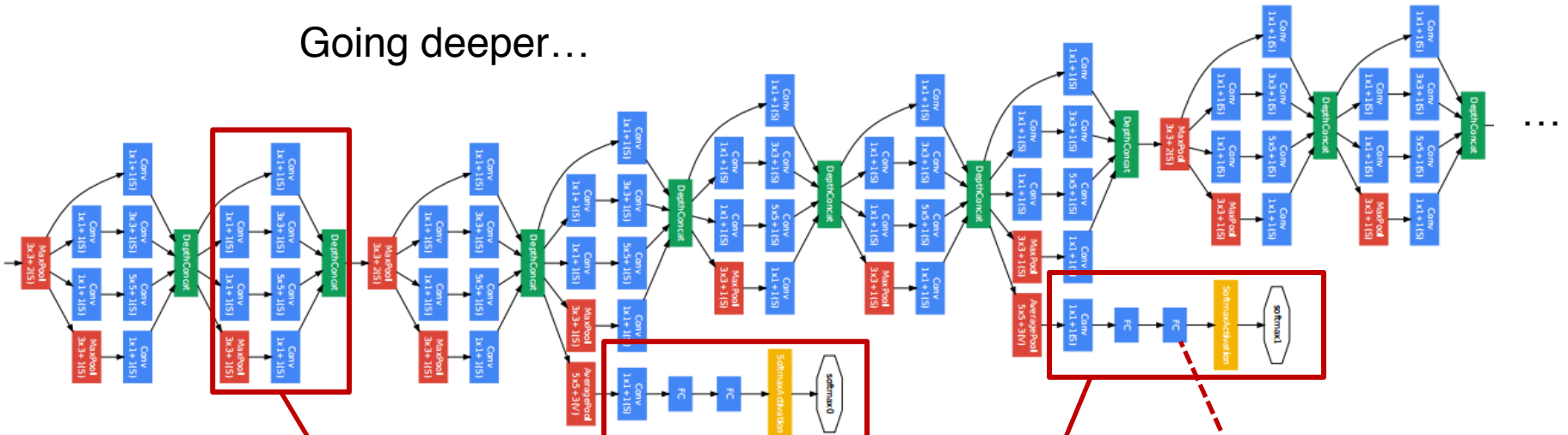



Image: http://cs231n.stanford.edu/slides/2018

# Image classification

**GoogLeNet (Inception v1)** **[Szegedy et al, 2014]**

Going deeper…



1) Repeating blocks called *Inception module*

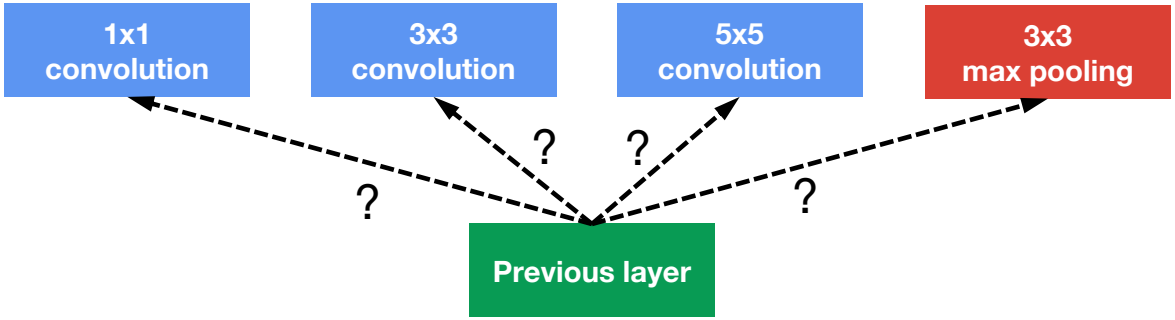
2) Intermediate classification losses to inject gradient in middle layers

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

Let's take a closer look at Inception modules…

# Image classification

**Inception module**

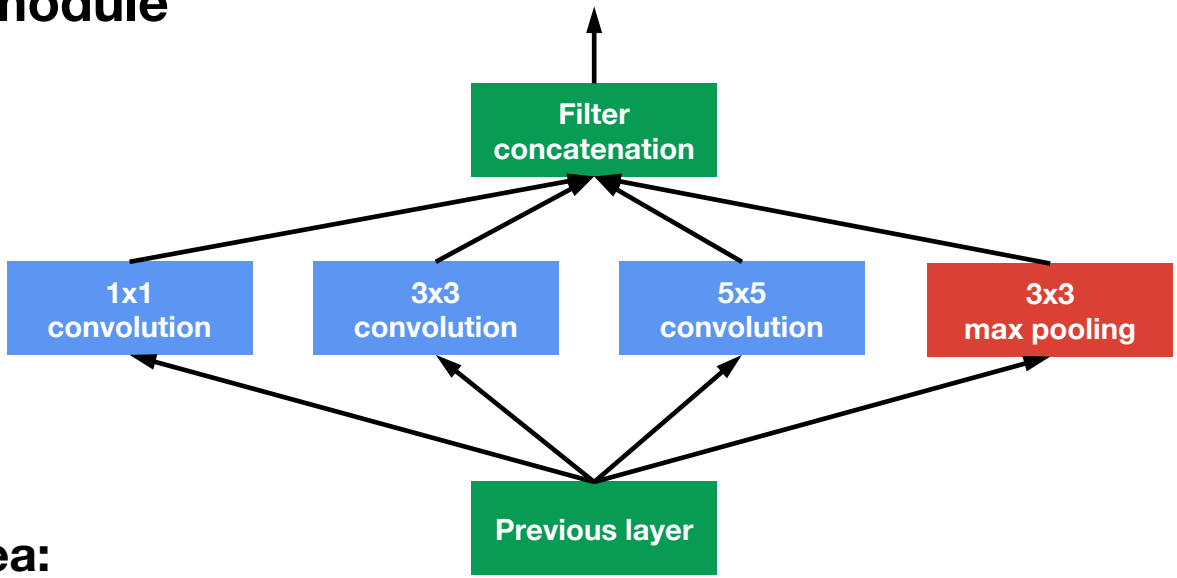



**Choice for each layer:**

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

# Image classification

**Inception module**



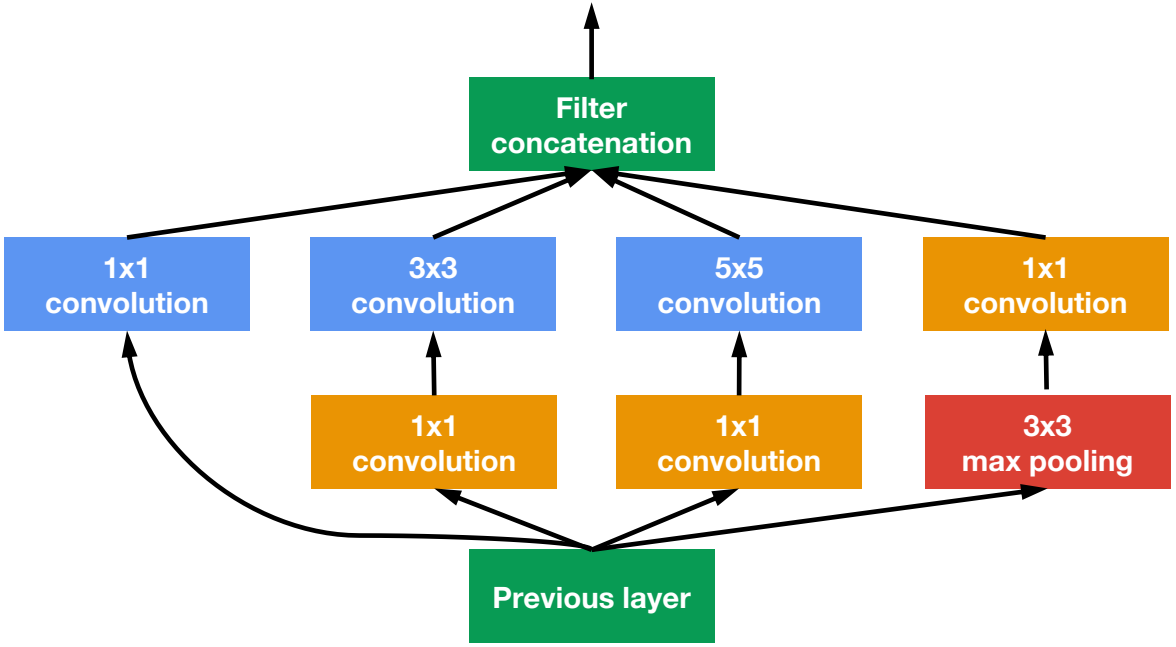

**Key idea:**

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

**Problem**: this gives too many outputs and parameters

# Image classification

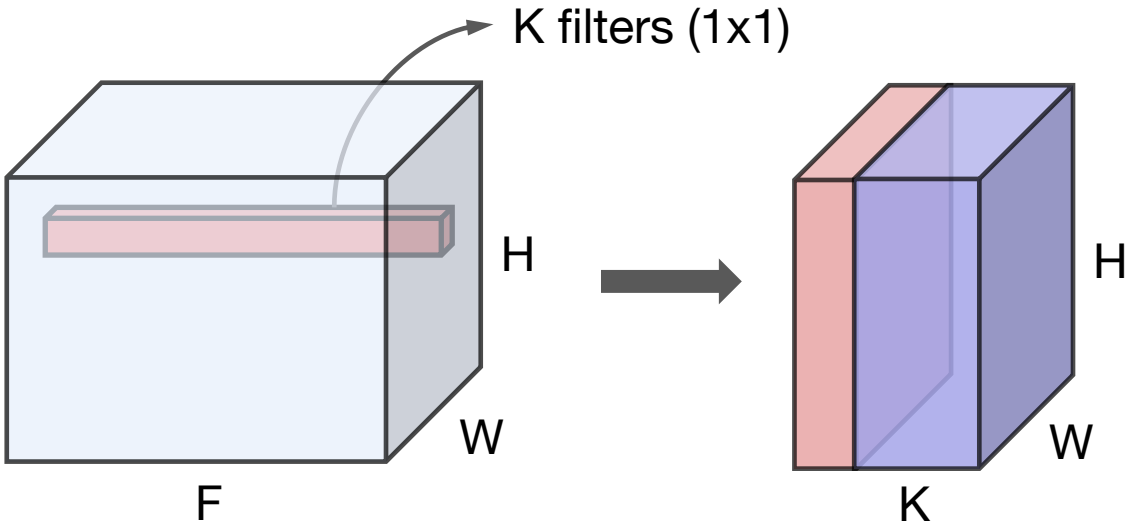
**Inception module**




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

# Image classification

**1x1 convolution**




**Intuition**: Acts like a learnable feature pooling function

To reduce size, choose K < F

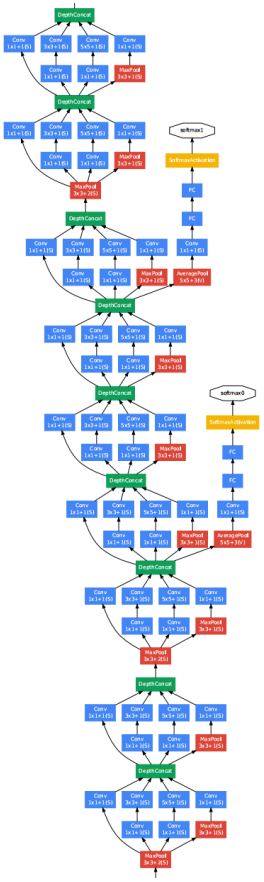
# Image classification

**Efficiency**


**AlexNet**

8 layers

~62M parameters



**GoogLeNet**

22 layers

~5M parameters

GoogLeNet has 12x fewer parameters than AlexNet !

# Image classification

**ResNet** **[He et al, 2016]**

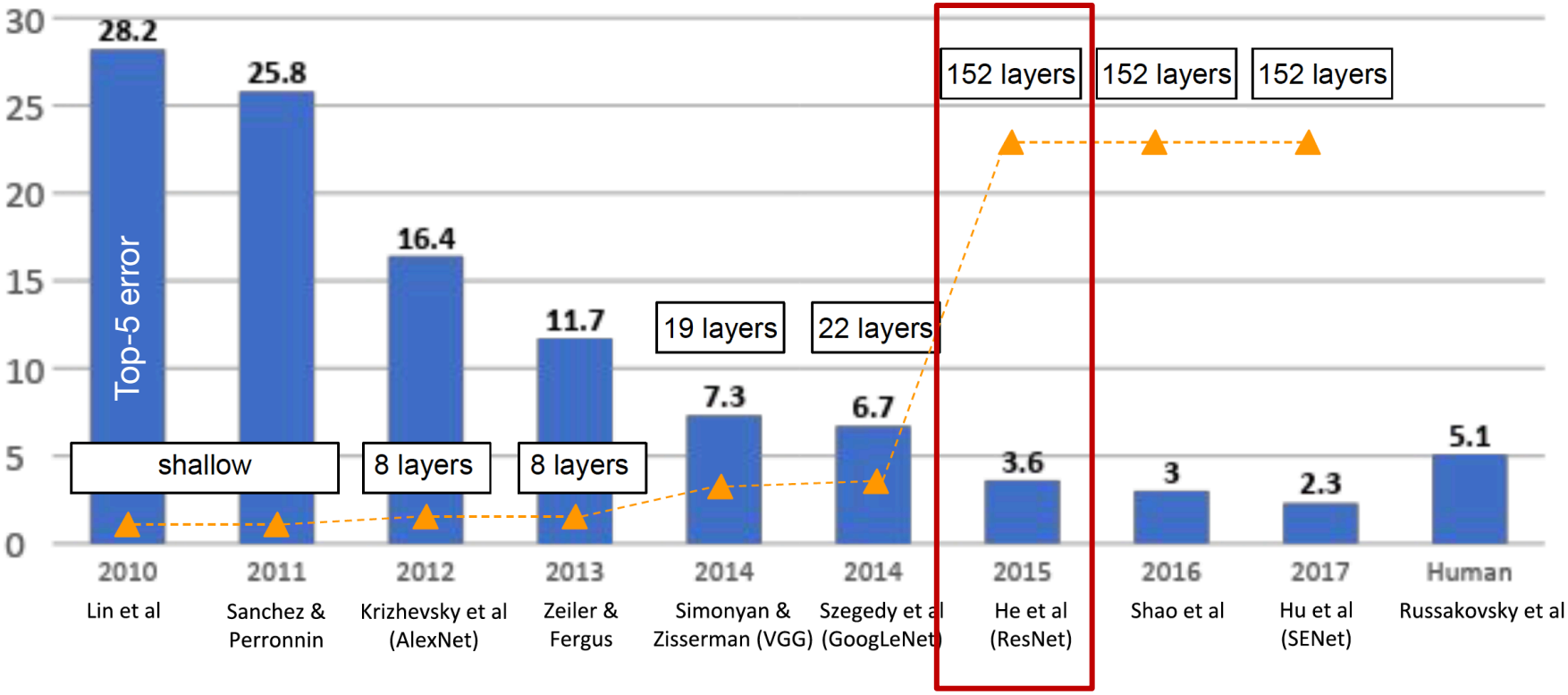
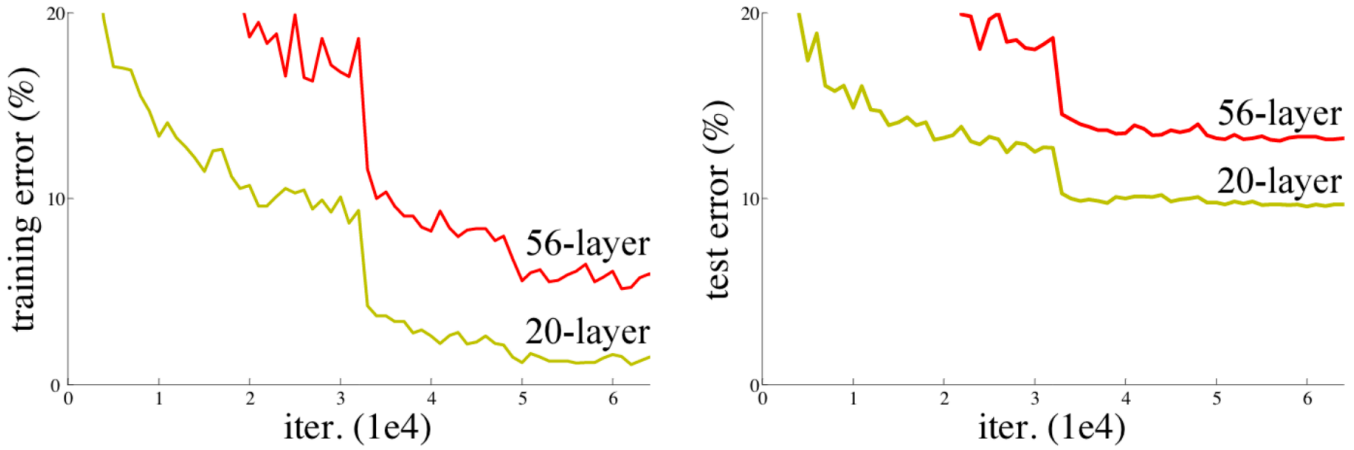



Image: http://cs231n.stanford.edu/slides/2018

# Image classification

**ResNet** **[He et al, 2016]**

What happens if we make a standard CNN deeper ?



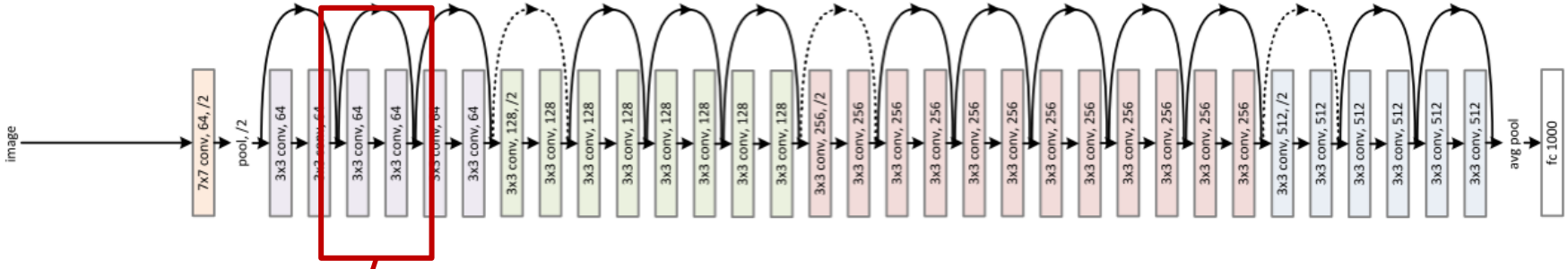

- Both training and test errors are larger
- This is not overfitting, its an optimization problem (e.g., vanishing gradient)

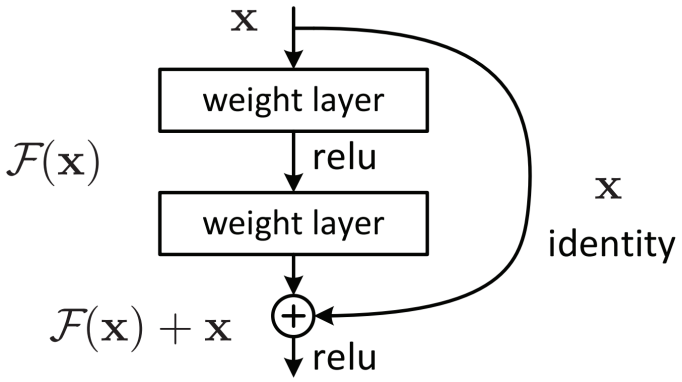
Image: He et al. "Deep residual learning for image recognition." CVPR, 2016.

# 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

**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

## Densely connected CNN (DenseNet) [Huang et al, 2016]

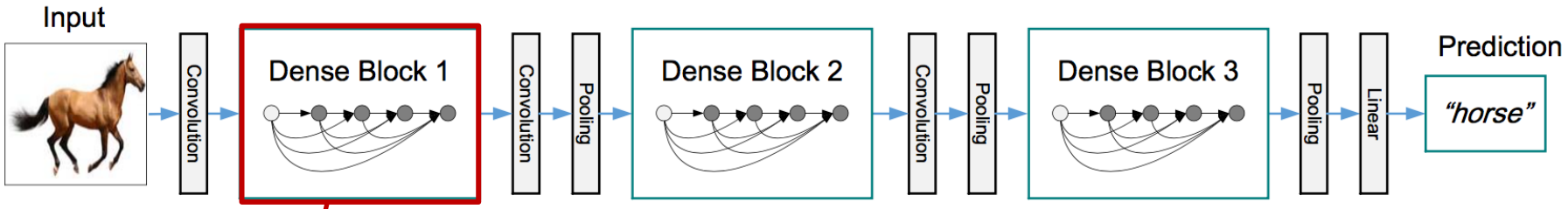
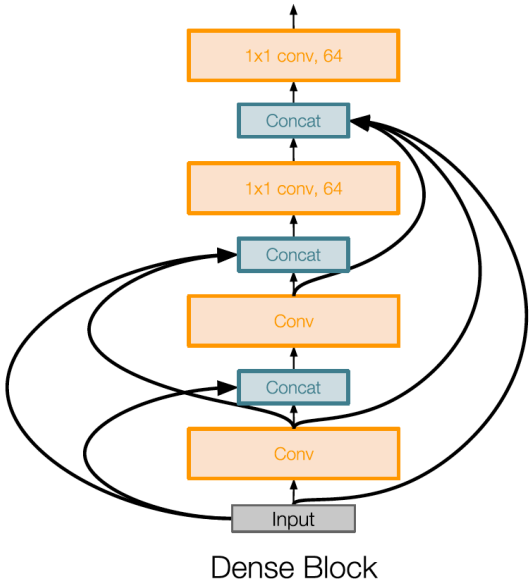



Image: Huang et al. "Densely connected convolutional networks." CVPR, 2017.




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

# Image classification

**Alzheimer's diagnosis by 3D CNN adaptation [Hosseini-Asl, 2016]**

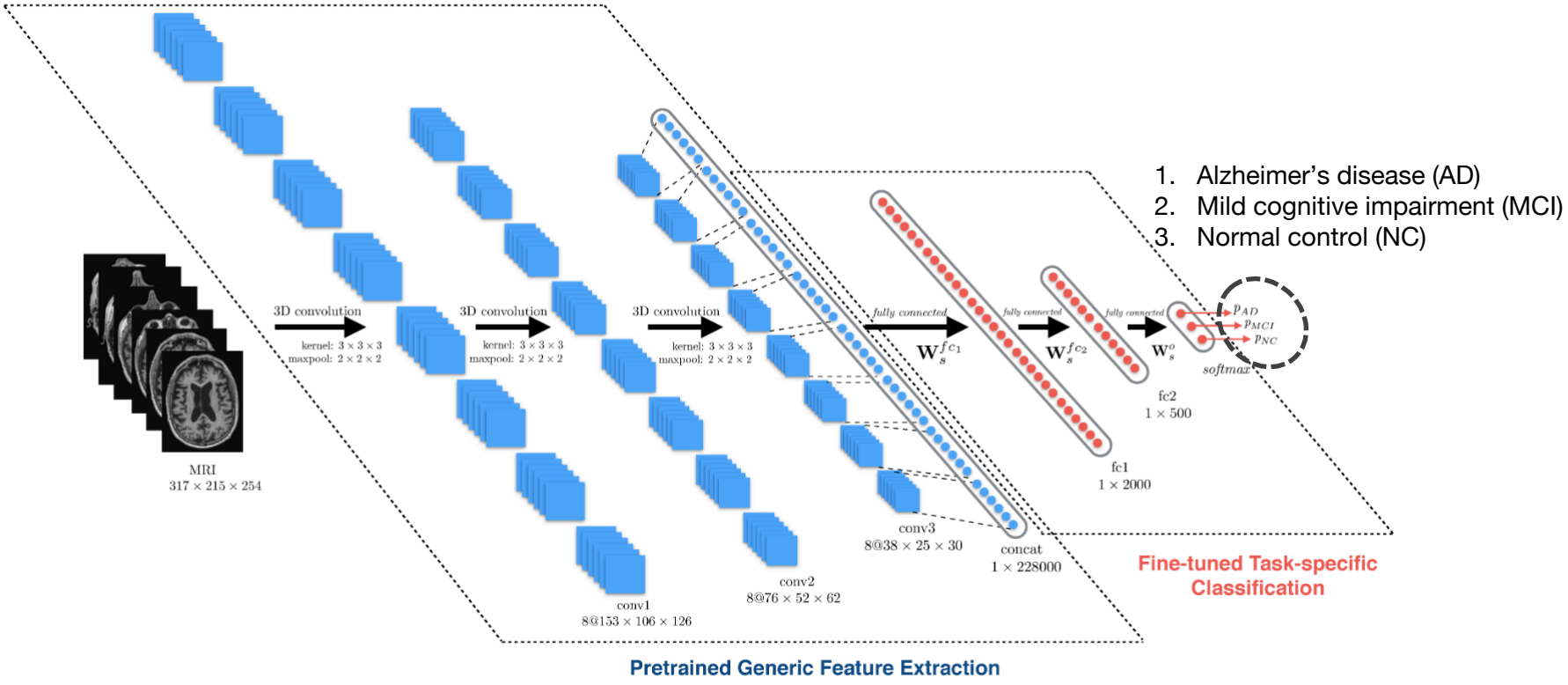



Image: Hosseini-Asl et al. "Alzheimer's disease diagnostics by adaptation of 3D convolutional network.", ICIP, 2016.

# SEMANTIC SEGMENTATION

# Semantic segmentation

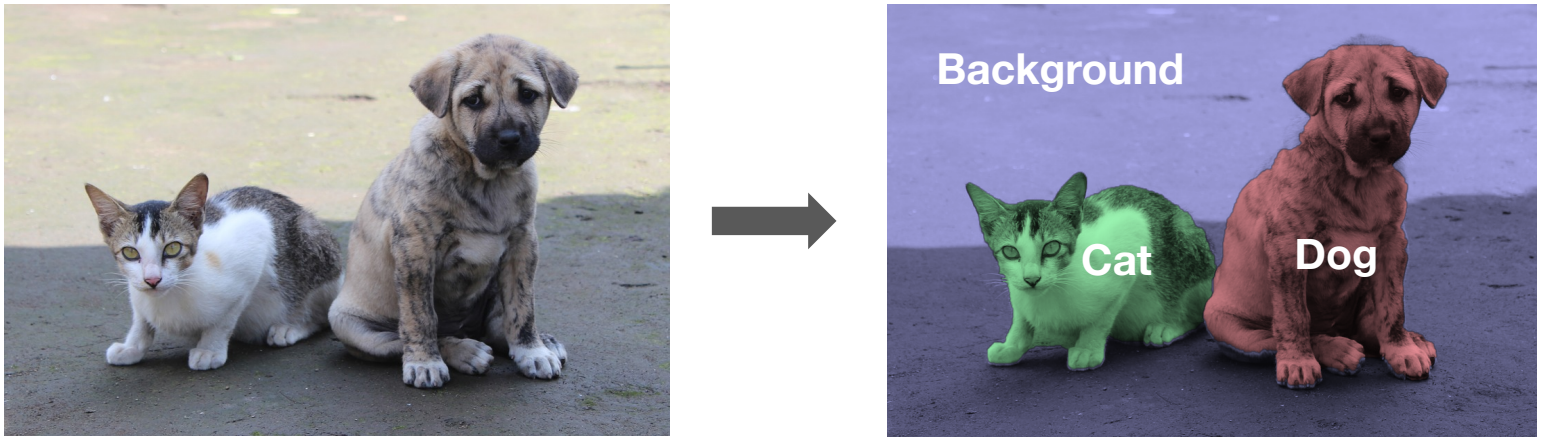



**Image** **Segmentation**

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

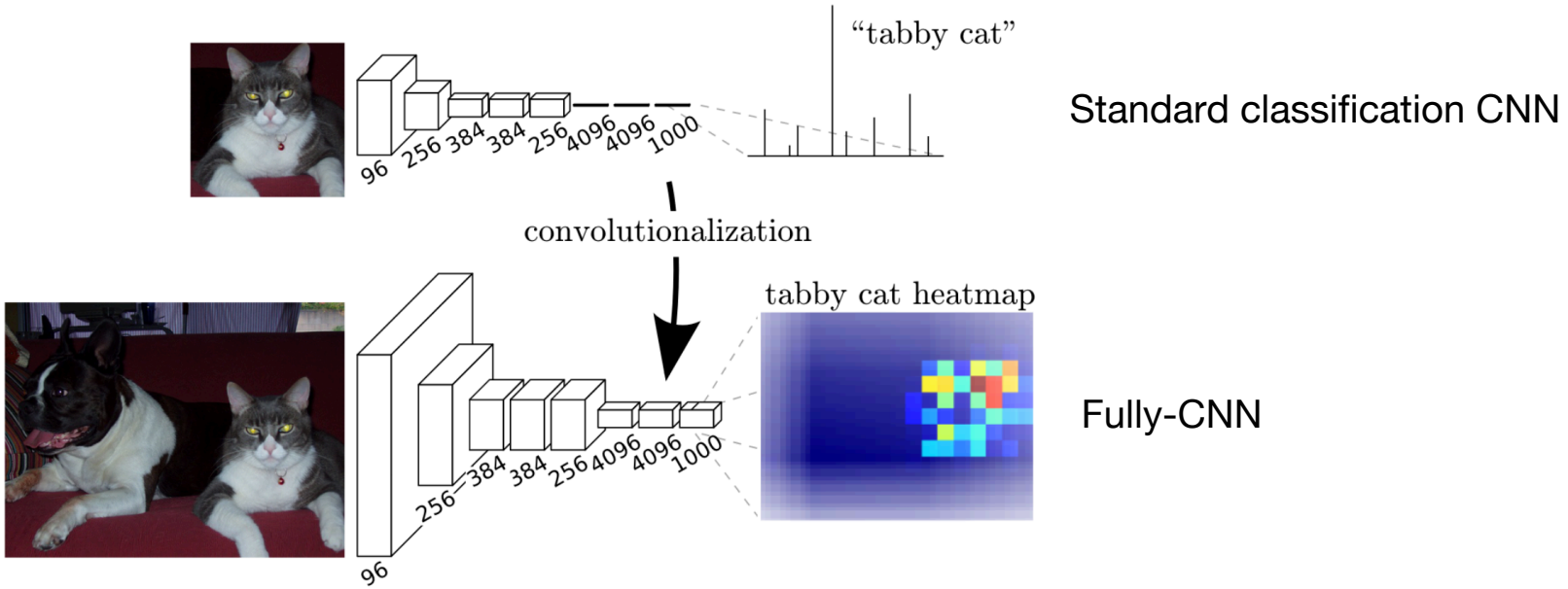
Can be seen as a dense and structured classification problem

Predictions are not independent

Possibly millions of pixels at the same time

# Semantic segmentation

**Fully-CNN : from classification to segmentation**




Standard classification CNN

Fully-CNN

**Key ideas:**

- Replace FC layers by 1x1 convolutions
- Network becomes 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



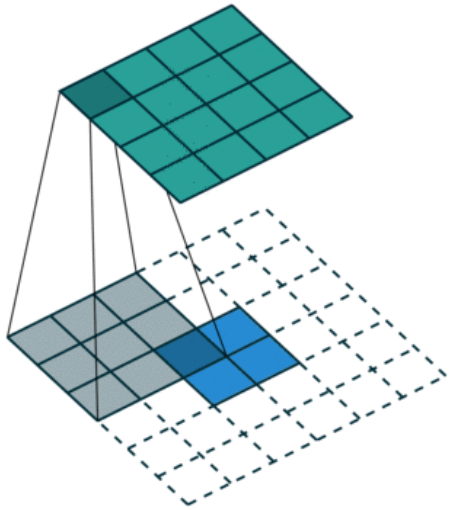
Image: Long et al. "Fully convolutional networks for semantic segmentation." ICCV, 2015.

# Semantic segmentation

**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:**

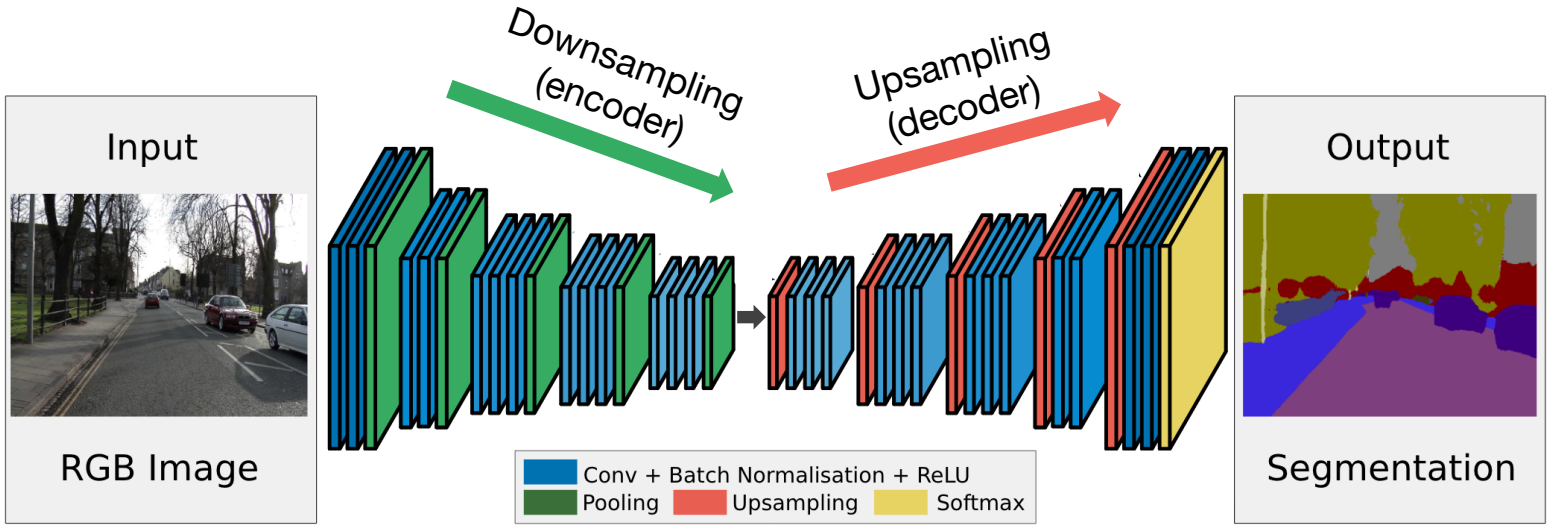


$$\begin{pmatrix} f_1 & f_2 & f_3 \end{pmatrix} *^T \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} f_1 & 0 & 0 \\ f_2 & f_1 & 0 \\ f_3 & f_2 & f_1 \\ 0 & f_3 & f_2 \\ 0 & 0 & f_3 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}$$

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

# Semantic segmentation

**Encoder-decoder architecture:**



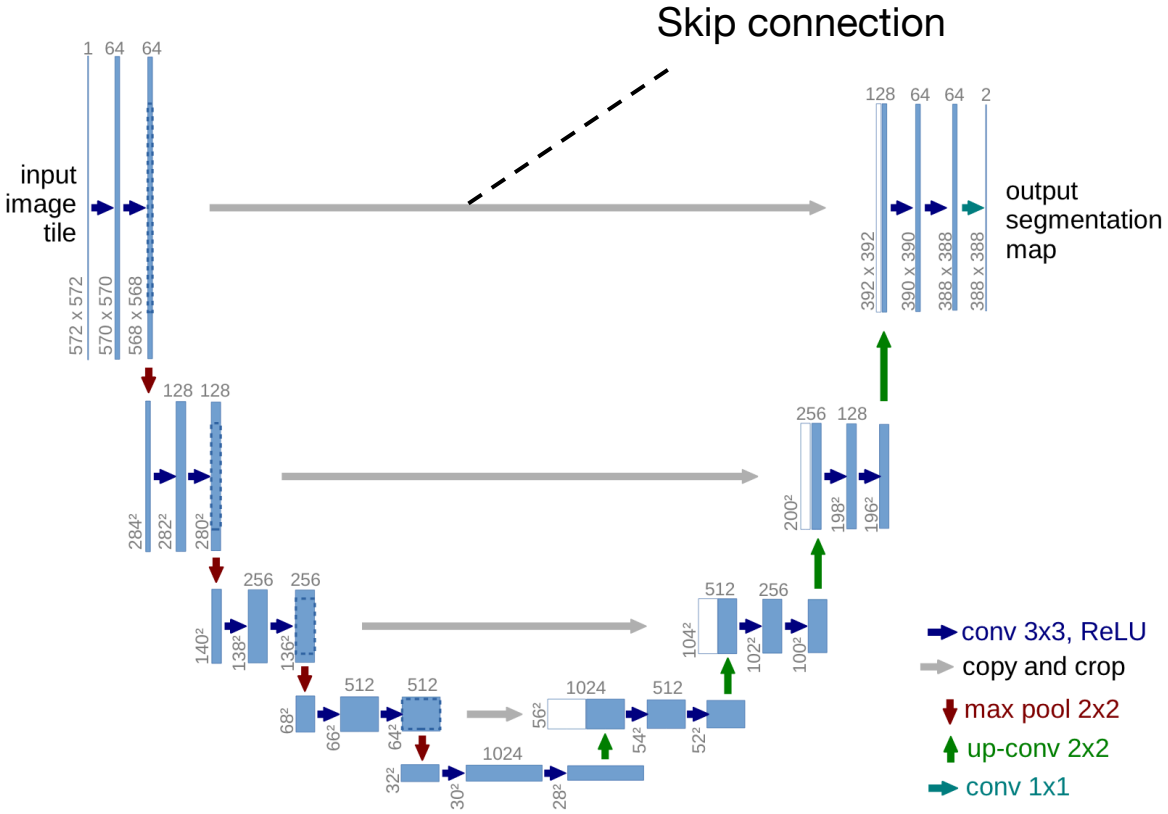

**Problem**: Spatial resolution is lost while downsampling

**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

**U-Net [Ronneberger et al., 2015]**



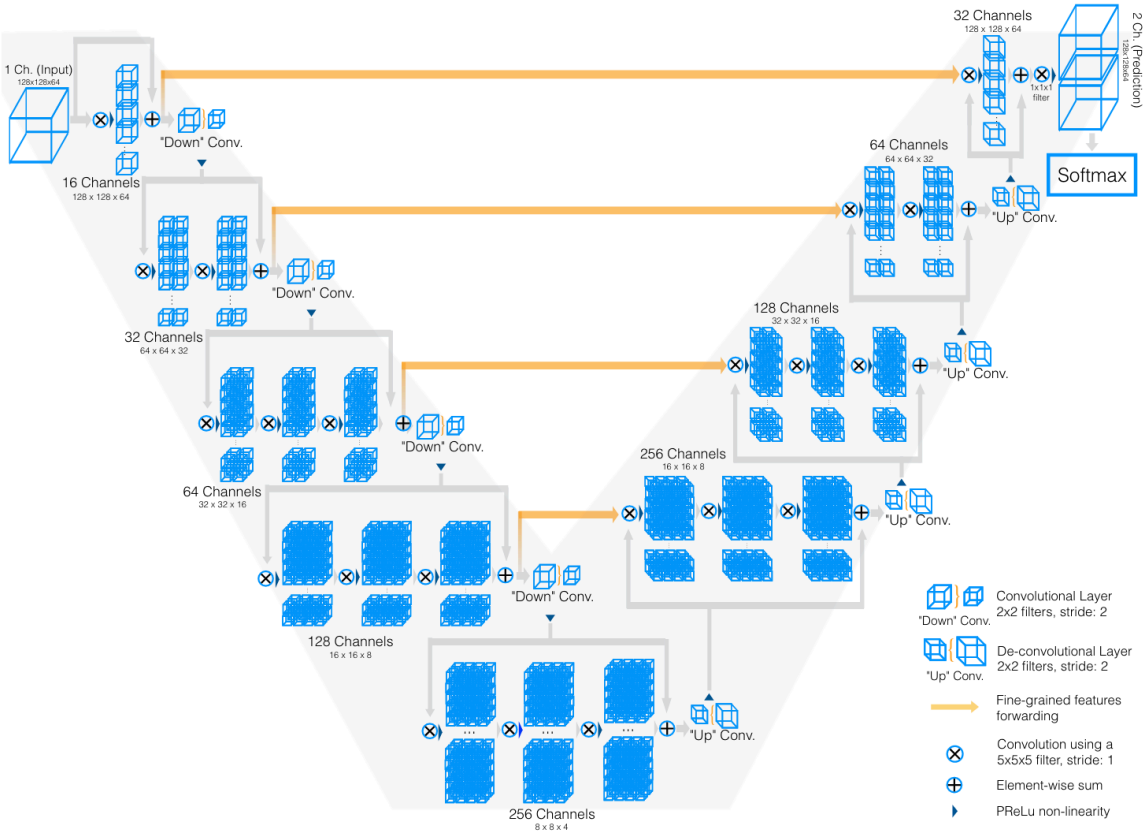

One of the most popular networks for medical image segmentation



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

# Semantic segmentation

**V-Net [Milletari et al., 2016]**

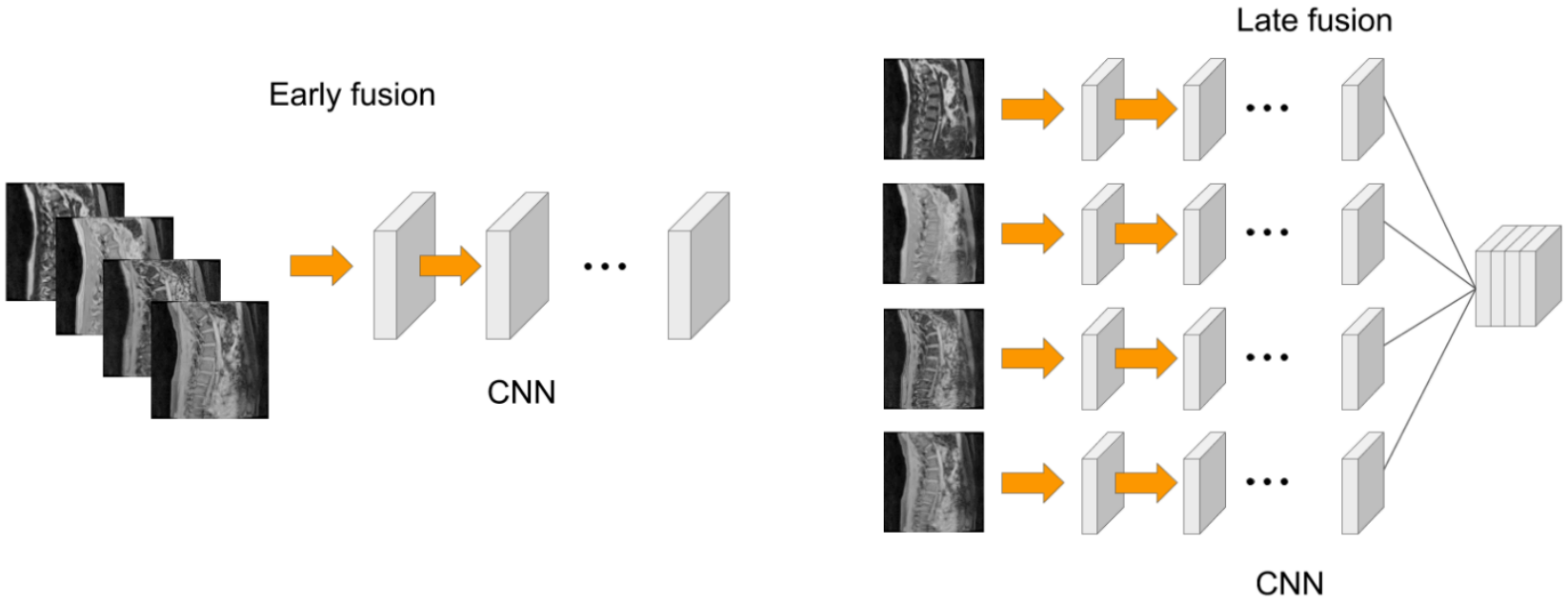



Extension of U-Net to volumetric (3D) data

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

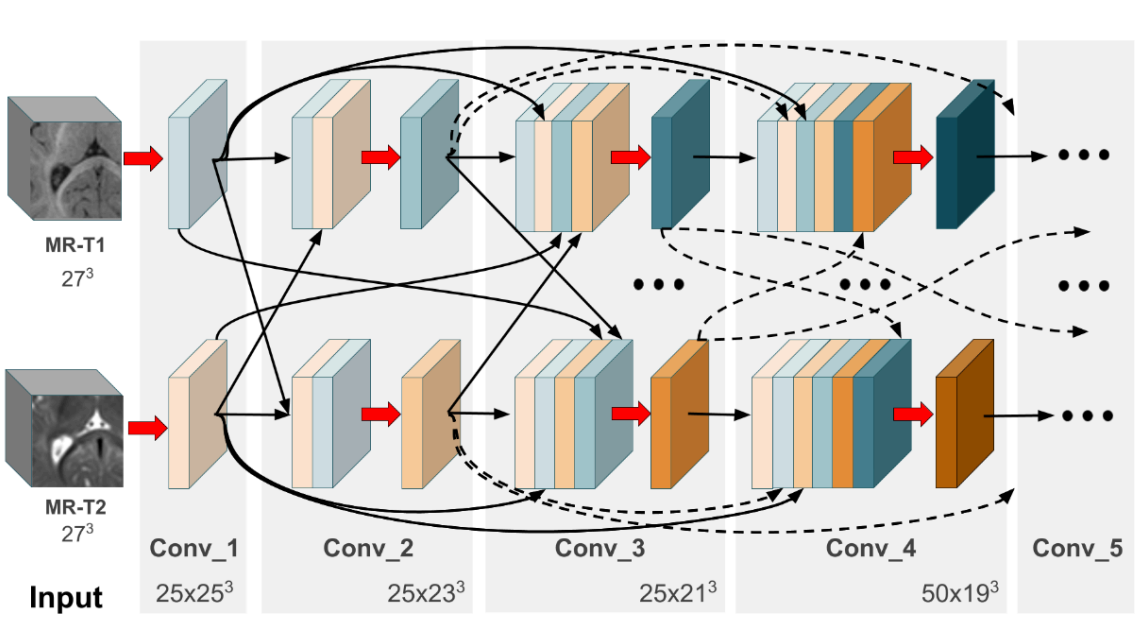


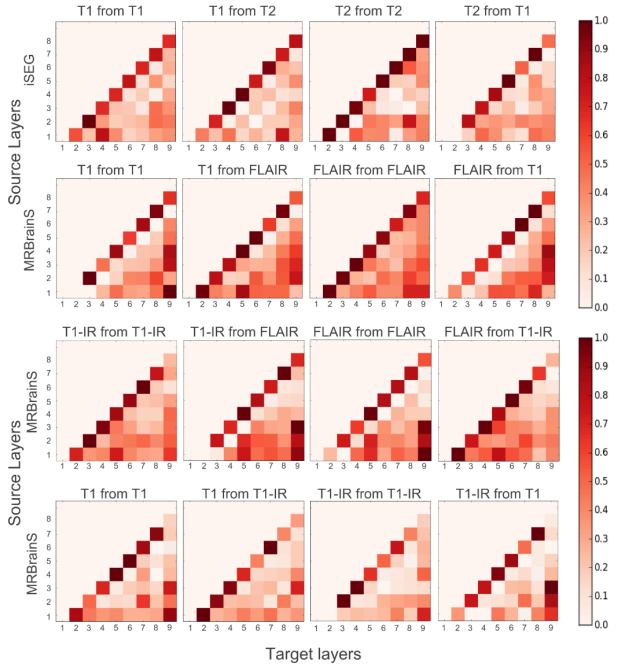

**Problem**: Ignores low or high level dependencies between information in different image modalities

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

# Semantic segmentation

## HyperDense-Net [Dolz et al., 2018]



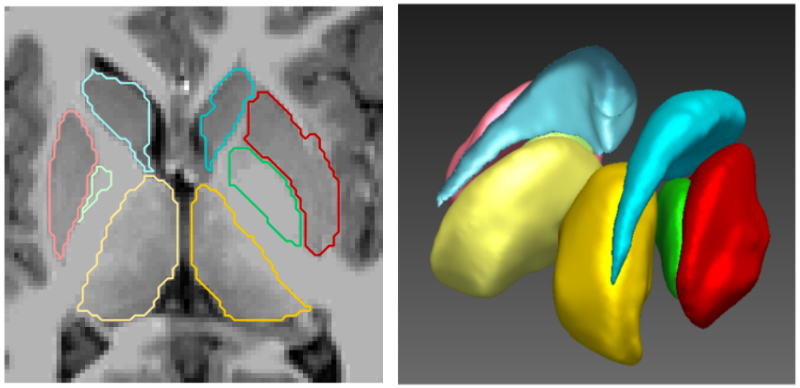

**Key ideas:**

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

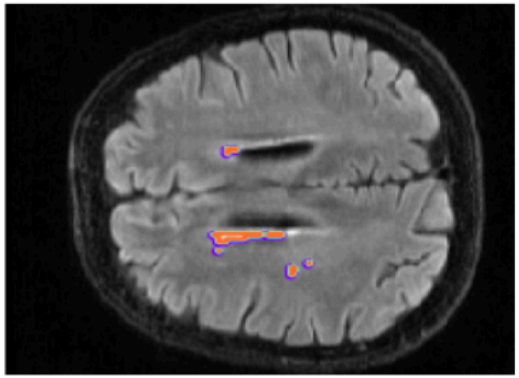
Image: Dolz et al. "HyperDense-Net: A hyper-densely connected CNN for multi-modal image segmentation." TMI, 2018.

# Semantic segmentation

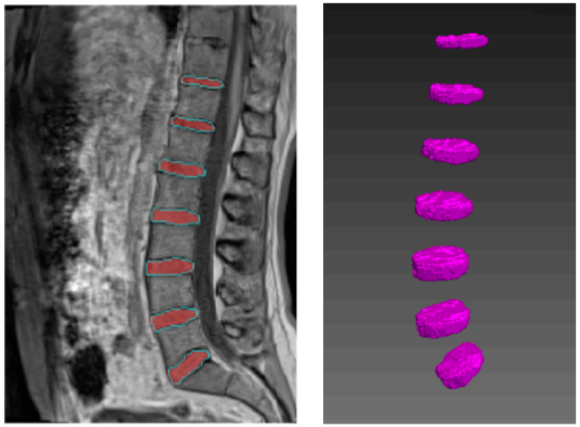
**Some medical imaging applications (all in MRI)**



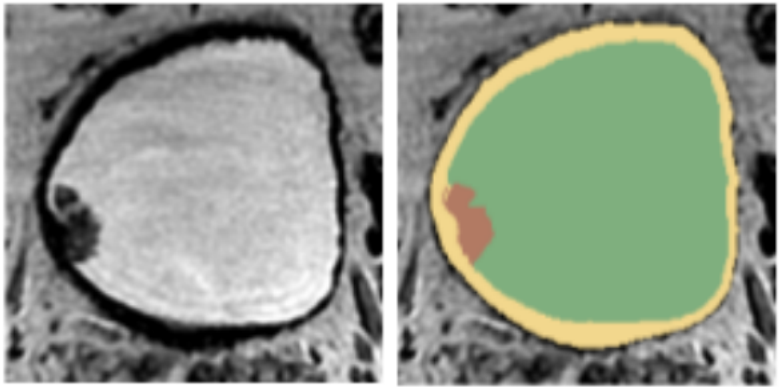
**Subcortical brain structures**
[Dolz et al, 2018]



**White matter hyperintensities**
[Dolz et al, 2019]



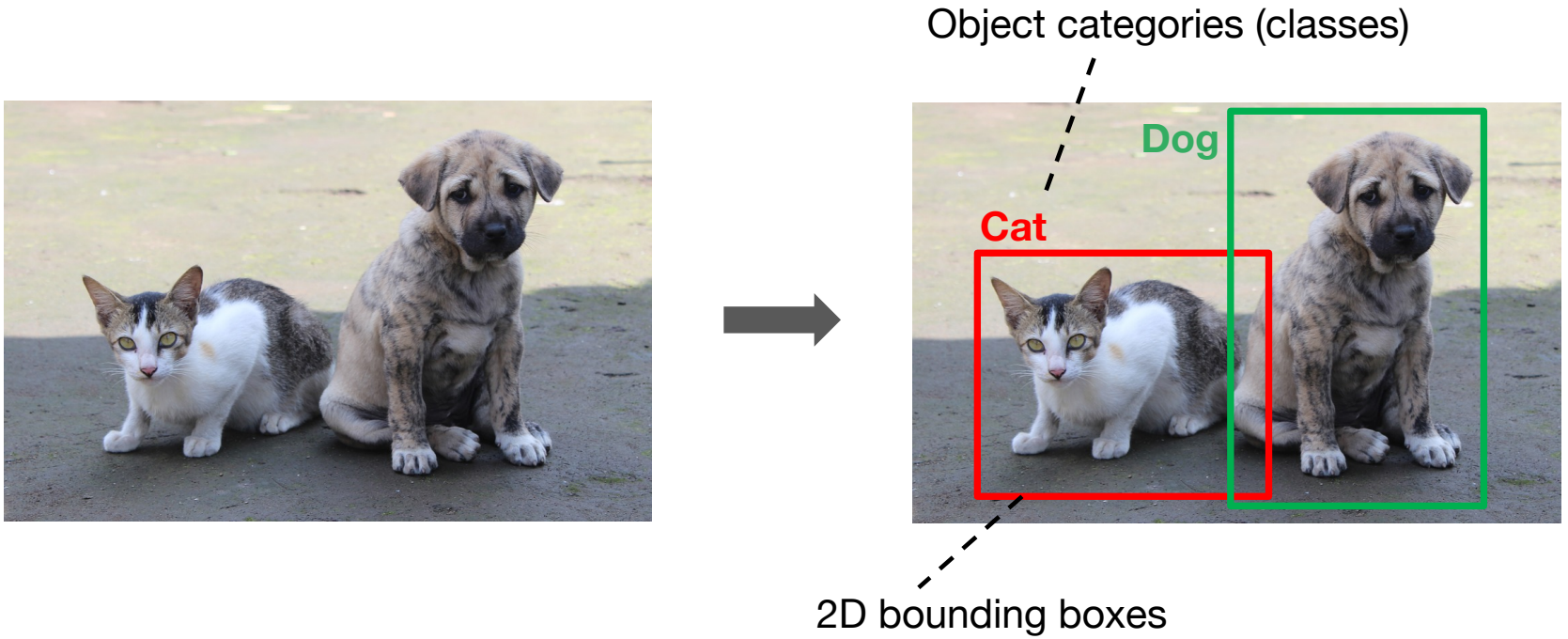
**Intervertebral disks**
[Dolz et al, 2019]



**Prostate wall and tumor**
[Dolz et al, 2018]

# OBJECT DETECTION

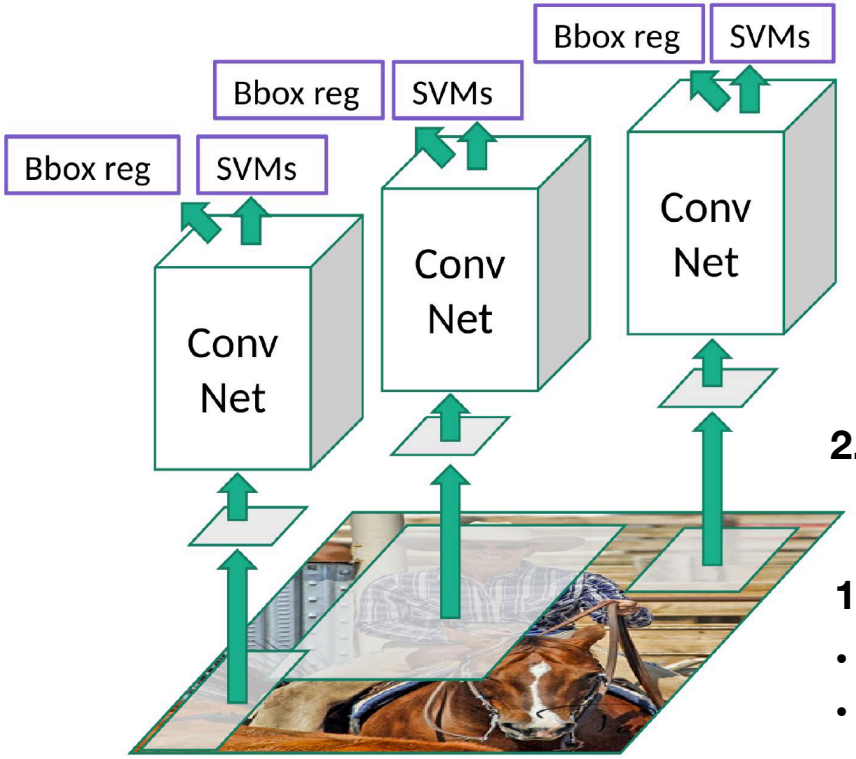
# Object detection




**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

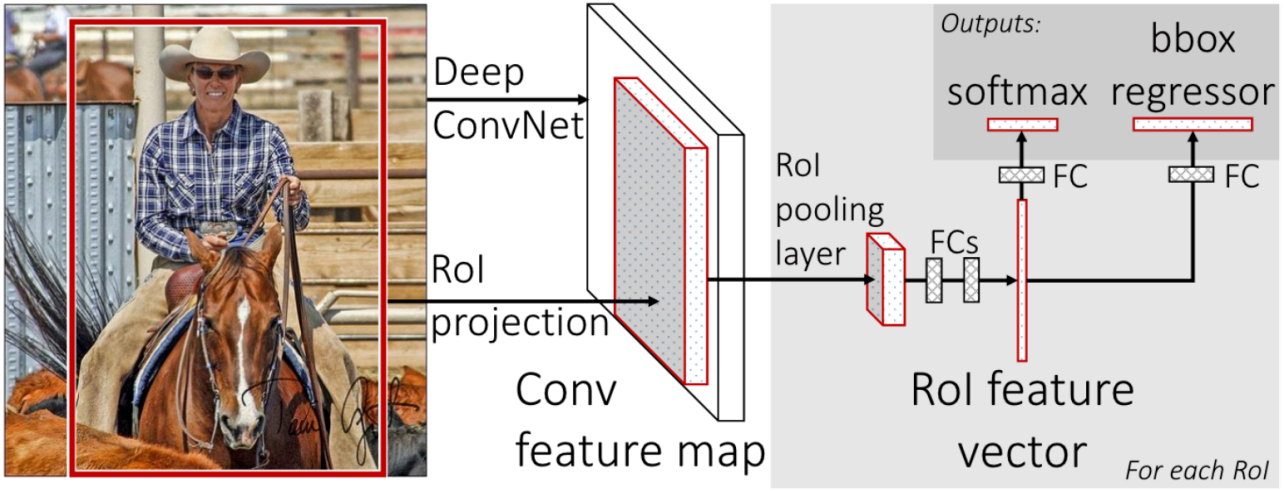
**Problems**:

- 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, 2015]**




**Key idea:**

- Swap the order of generating region proposals and running the CNN
- Enables sharing CNN computations between different proposals

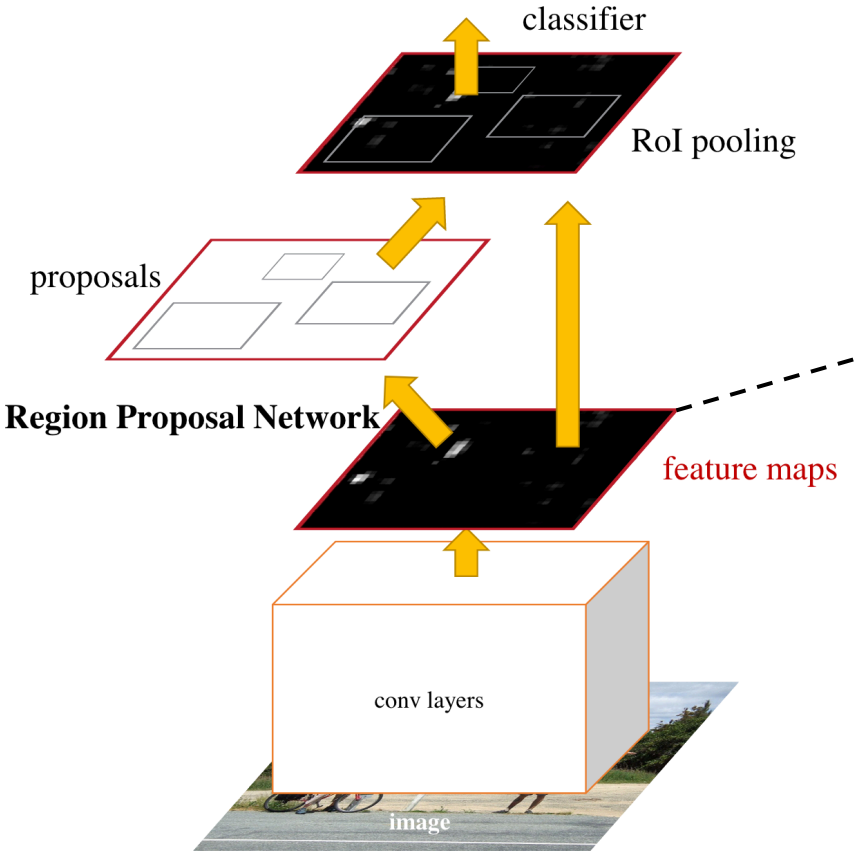
**Problem**: Still requires extracting proposals in a separate step

**How to avoid this ?**

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

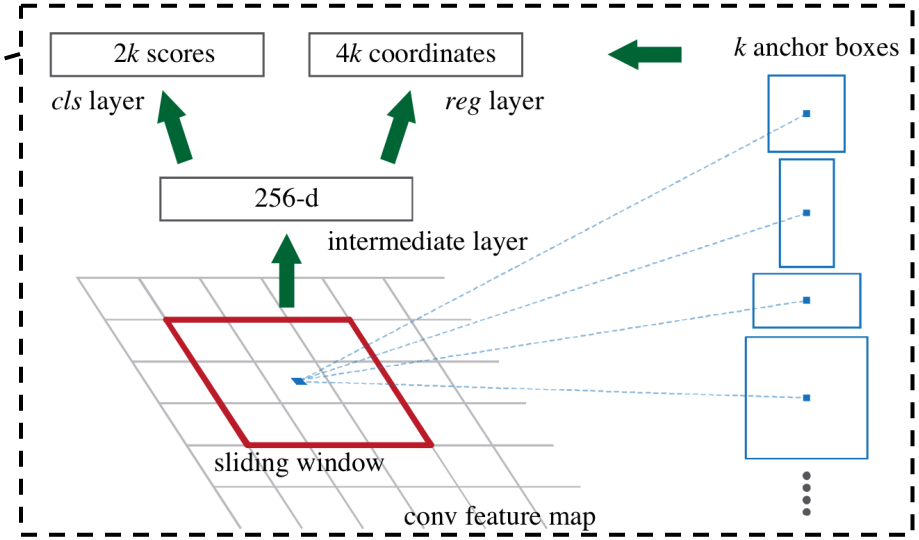
# Object detection

## Faster R-CNN [Ren et al., 2015]




**Key ideas:**

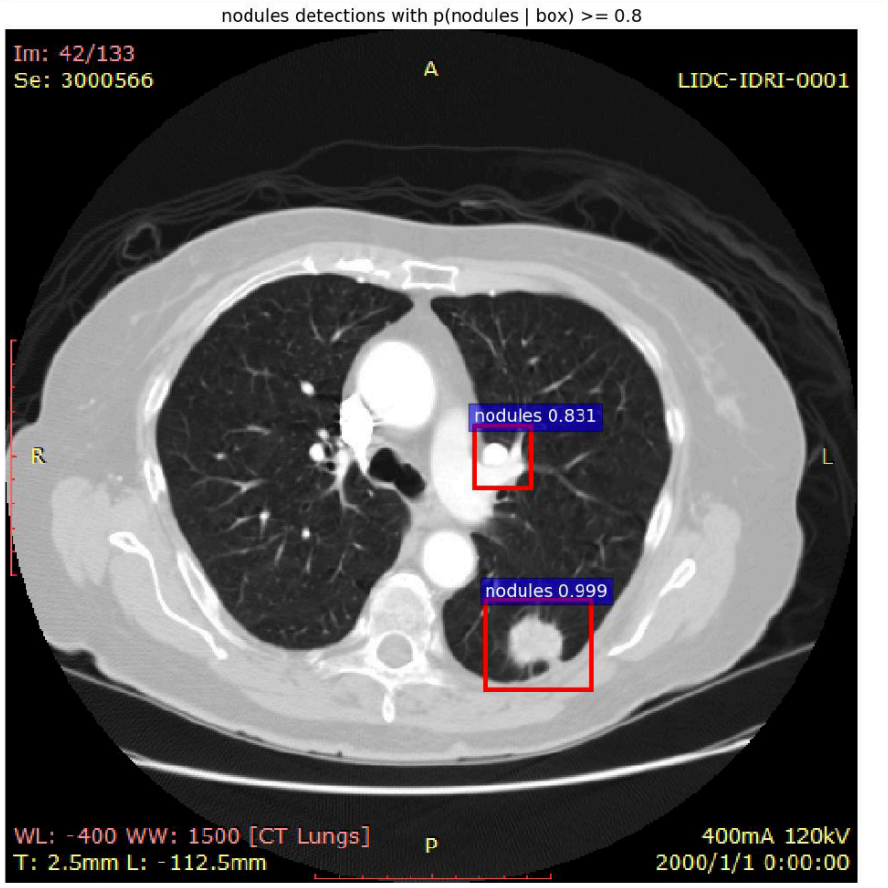
- Do the region proposal within the network
- Use anchor boxes with different scales and aspect ratios to generate RoI proposals
- Multitask loss for box regression and objectness




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

# Object detection

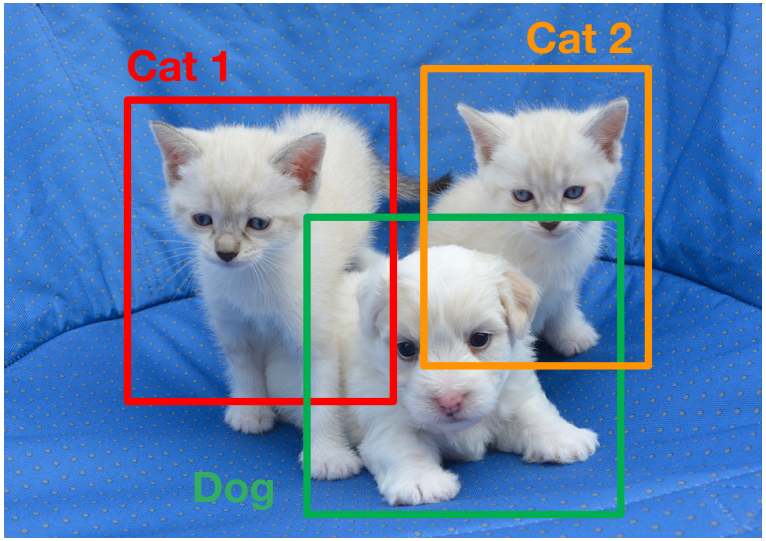
**Pulmonary nodule detection** **[Fan et al., 2018]**

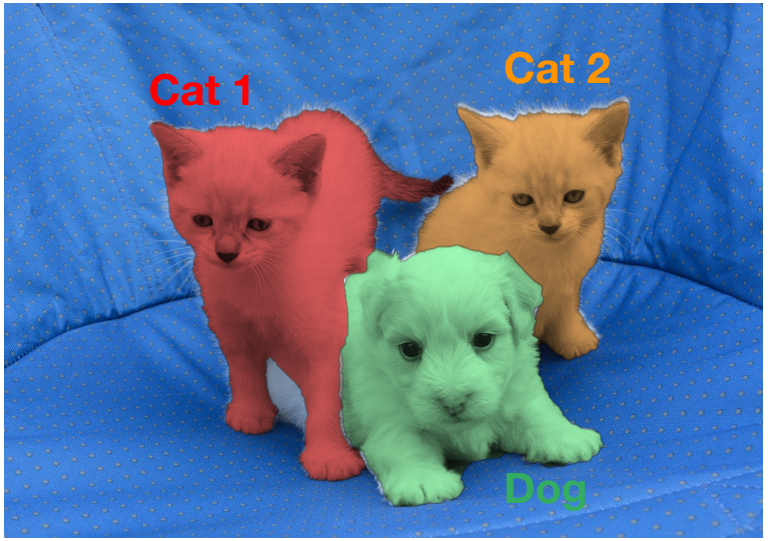



Images: Fan et al. "A modified faster R-CNN method to improve the performance of the pulmonary nodule detection." ICDIP, 2018

# INSTANCE SEGMENTATION

# Instance segmentation

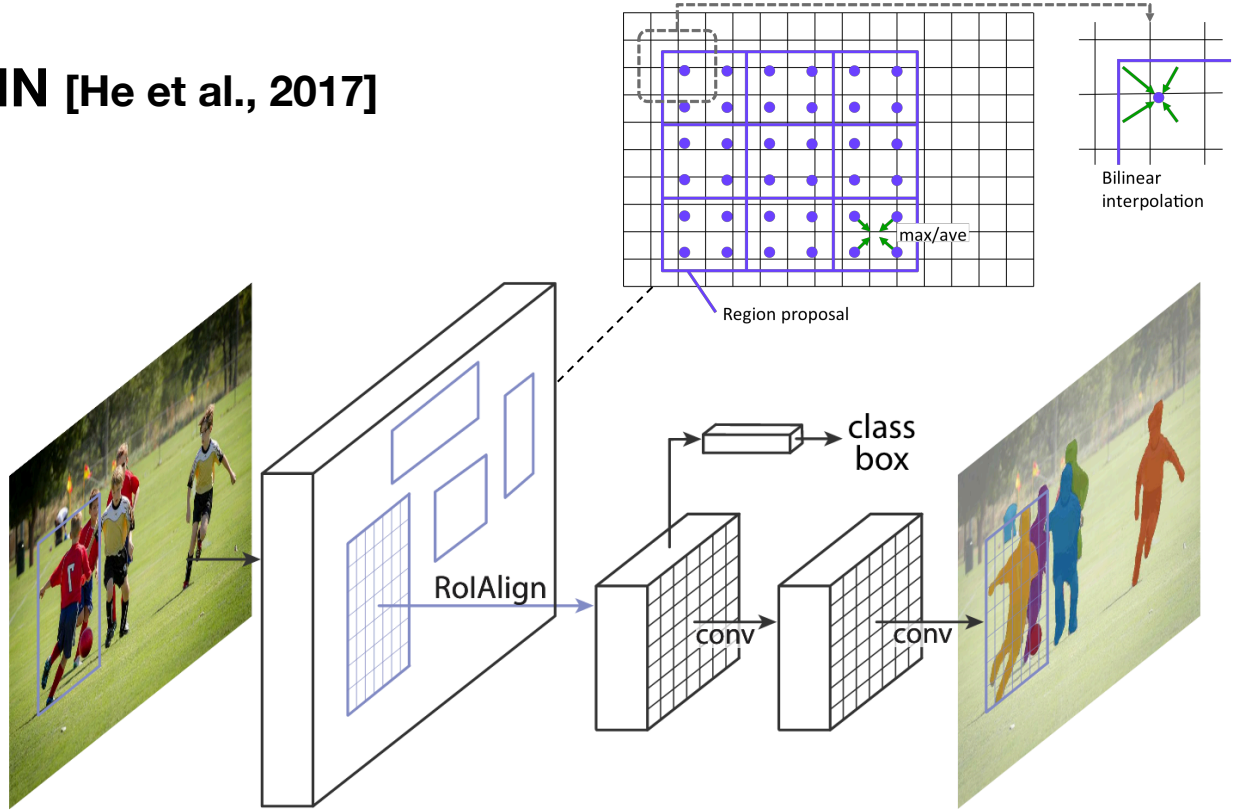



**Goal:** Detect and segment all instances of objects in an image

# Instance segmentation

**Mask R-CNN [He et al., 2017]**



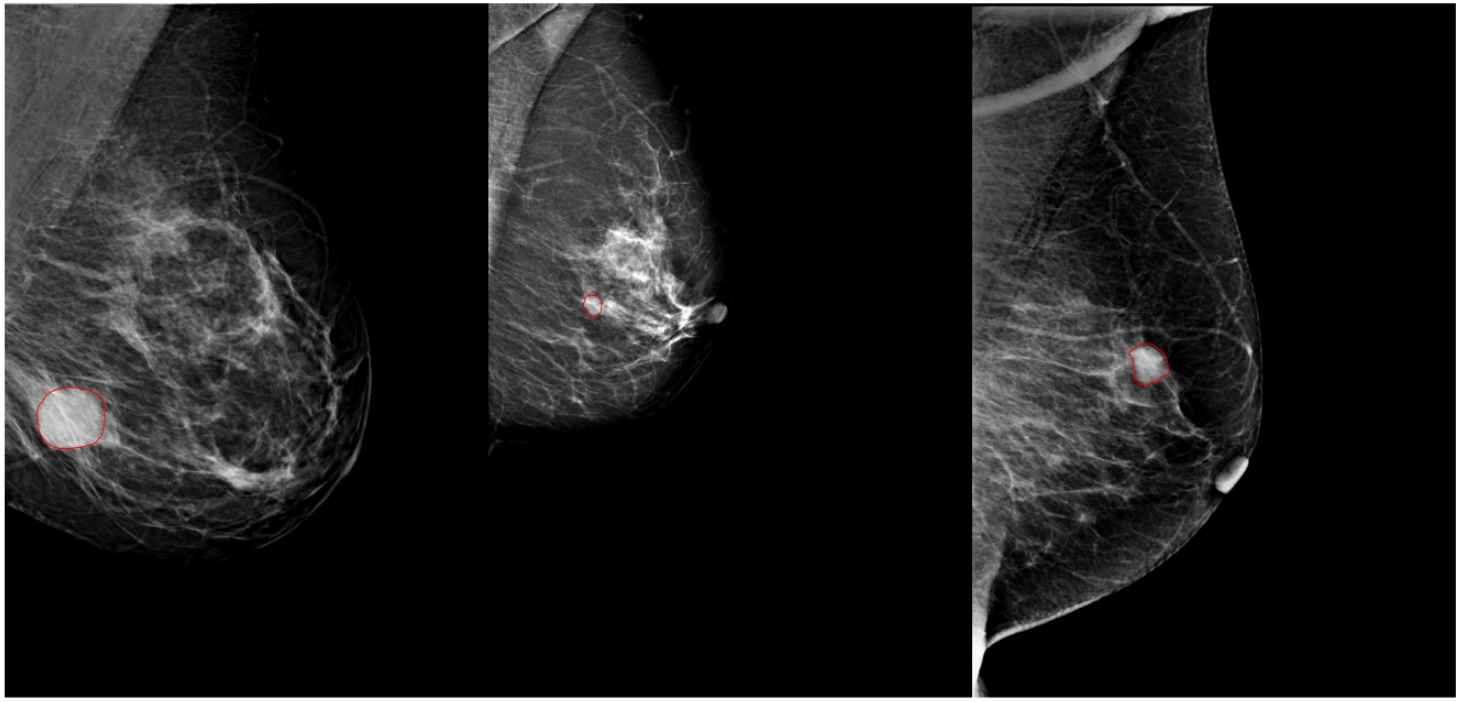

**Key ideas:**

- Same architecture as Faster R-CNN but with a third branch for segmentation
- Generates a fixed-size segmentation mask for each class in every RoI
- Use the box classification branch to select the right mask
- RoI pooling requires interpolation

Images: He et al. "Mask R-CNN." ICCV, 2017

# Instance segmentation

**Soft tissue lesion detection in mammography [Teuwen et al., 2017]**



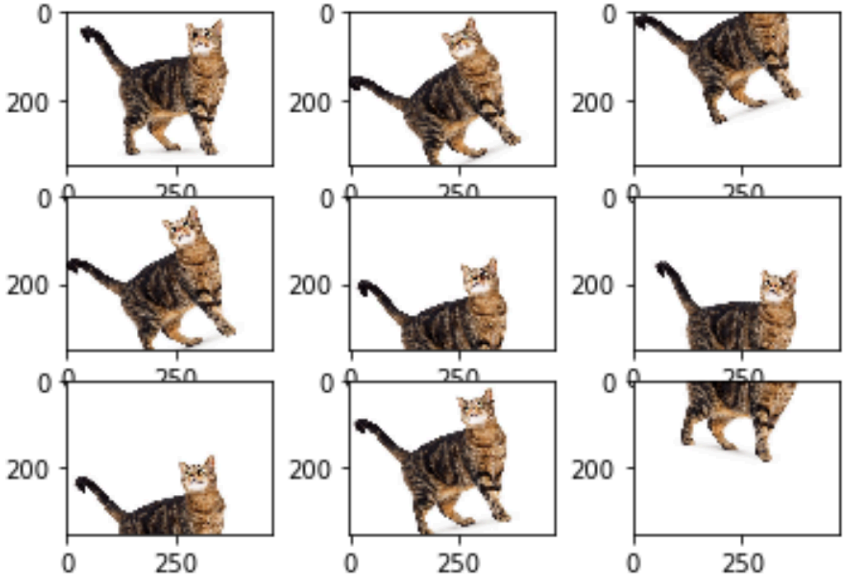


Images: Teuwen et al. "Soft tissue lesion detection in mammography using deep neural networks for object detection." MIDL, 2018.

# HANDLING SPATIAL VARIANCE

# Handling spatial invariance

**Data augmentation**




**Key idea:**

- Augment the training set by applying small transformations to the original images
- Try to make the network invariant to these transformations (better generalization)

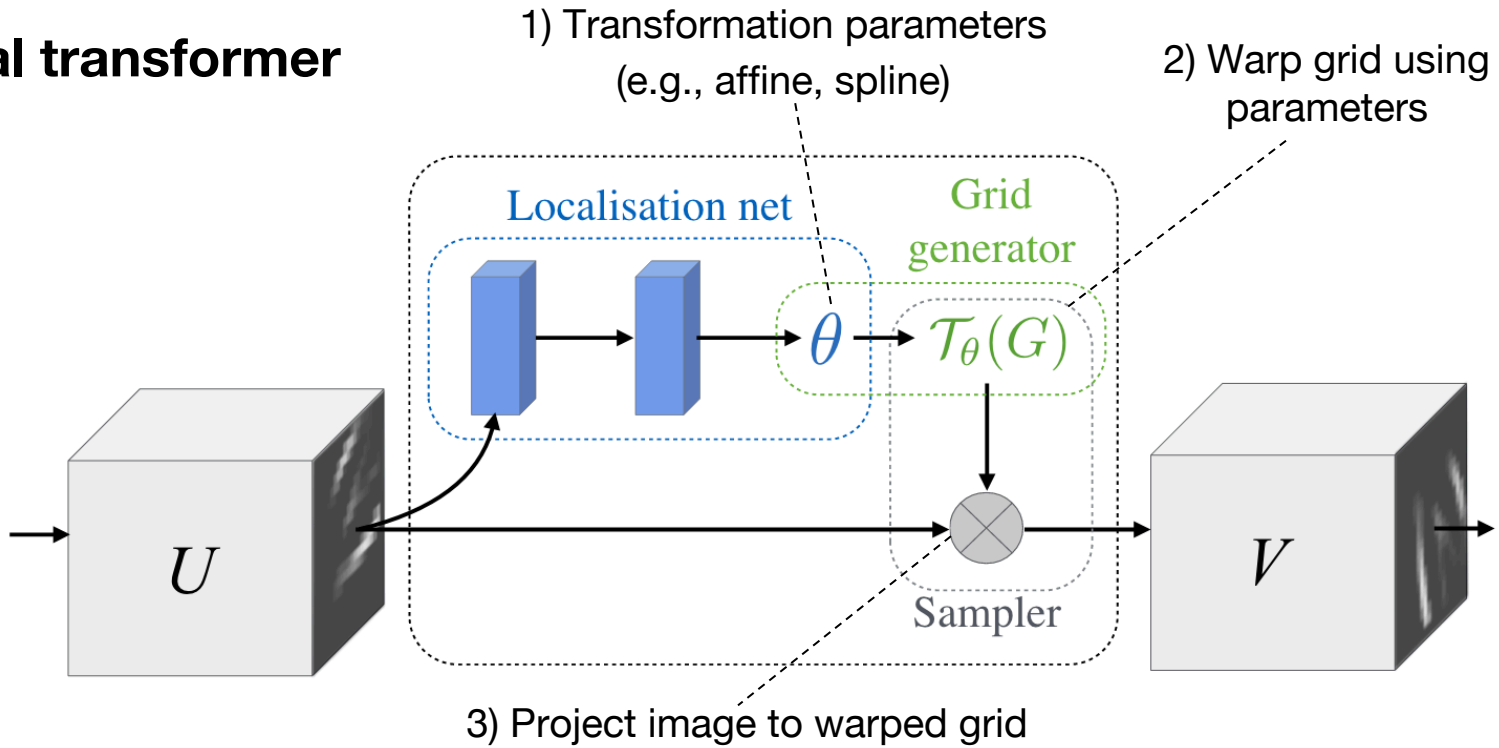
**Is there a better way of doing this ?**



Images: https://towardsdatascience.com

# Handling spatial invariance

**Spatial transformer**




**Key ideas:**

- Transforms the input image so that learning in subsequent layers is easier
- Does pose normalization (tilted or scaled objects) and enhances spatial attention
- Unlike in max-pooling, spatial invariance is learned

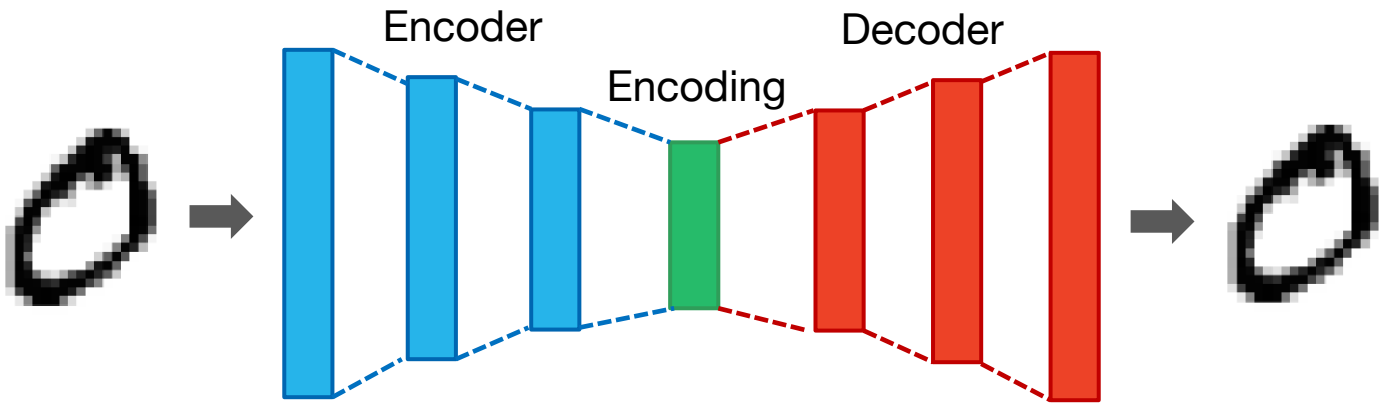


Images: Jaderberg et al. "Spatial transformer networks." NIPS, 2015.

# CONVOLUTION AUTOENCODER: GOING UNSUPSERVISED

# Convolutional autoencoder

**Standard model**



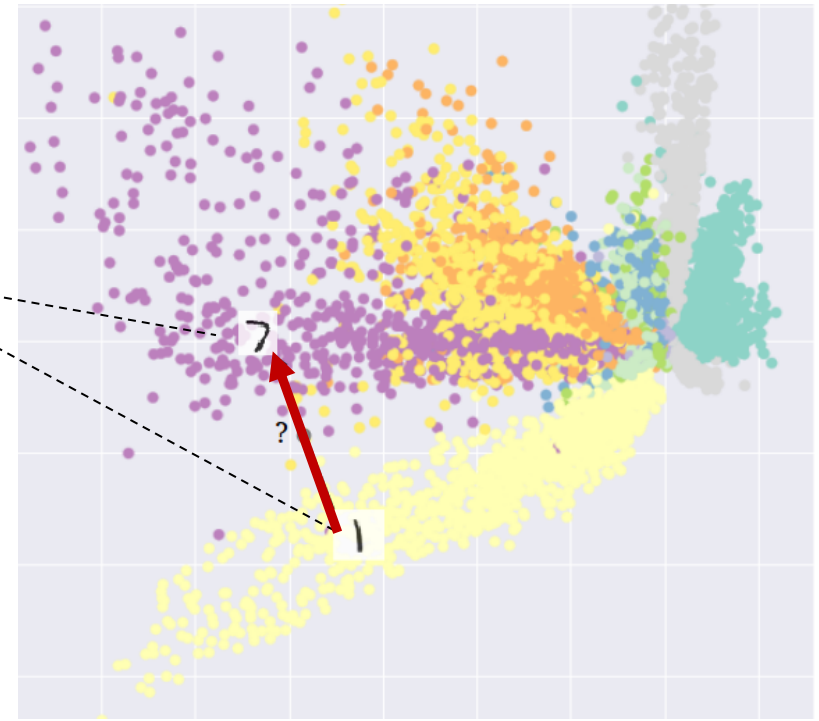

**Key idea:**

- Same encoder-decoder architecture, but the input and output images are the same
- Used for denoising images or having a low-dimensional representation of images

# Convolutional autoencoder

**The encoding space**

What would happen if we interpolated between two point the encoding space ?

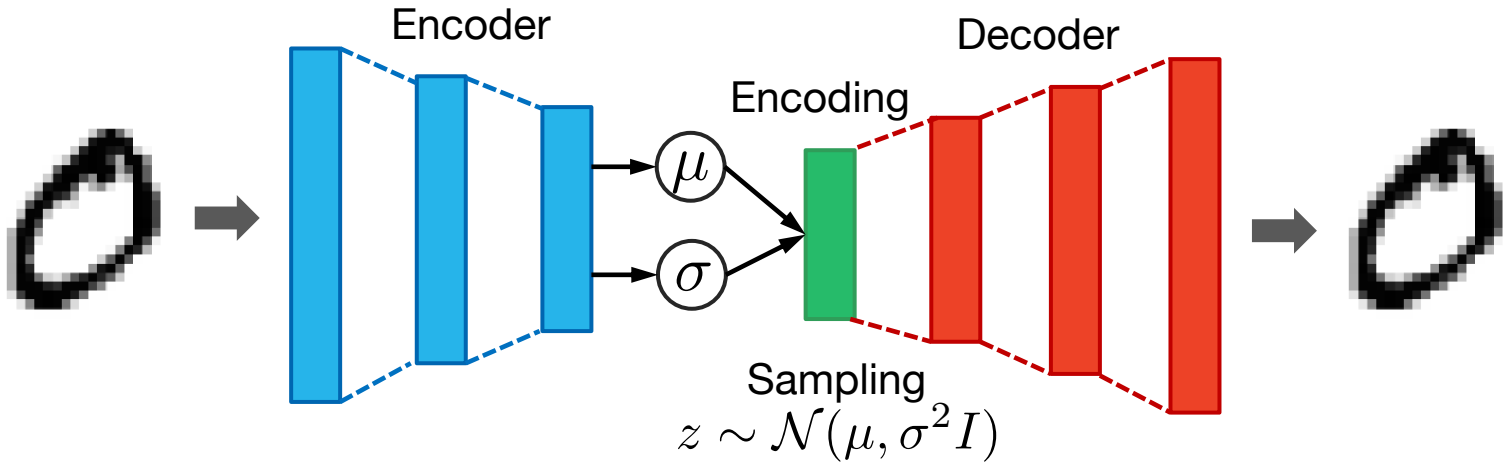



**Problem:** encoding space is not “regular”

Image adapted from: https://towardsdatascience.com

# Convolutional autoencoder

**Variational autoencoder (VAE)**

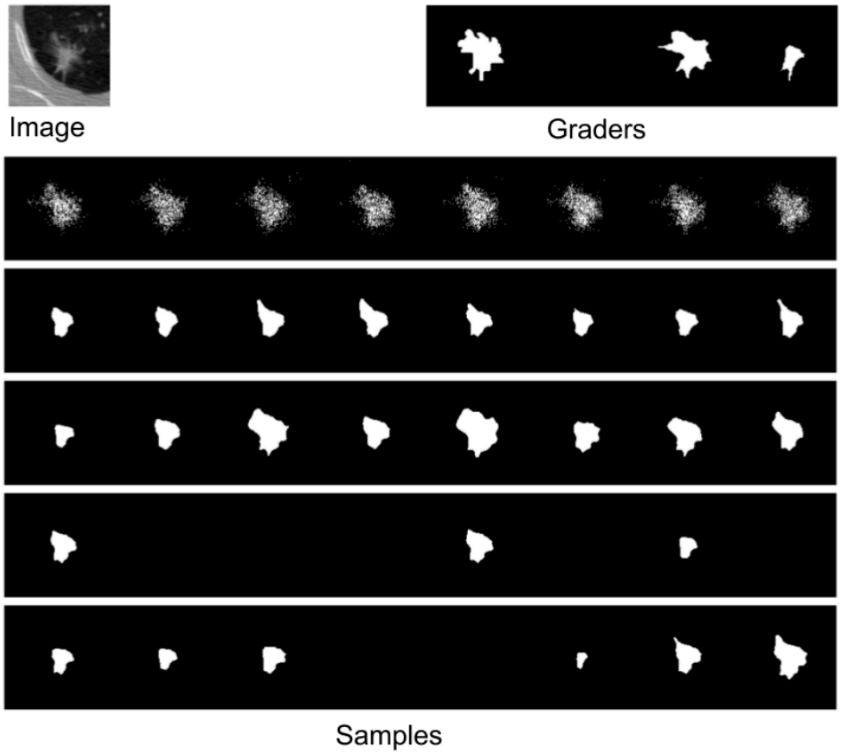



**Key idea:**

- Instead of mapping an image to its encoding, map to the parameters of a distribution (e.g., mu and sigma for Gaussian)
- Sample encodings based on this distribution, and minimize expected reconstruction error for these samples
- Make the distribution as close to a zero-mean, unit variance spherical Gaussian

# Convolutional autoencoder

**Probabilistic U-Net** **[Kohl et al., 2018]**




**Key idea:** Use VAE to generate multiple segmentation hypotheses

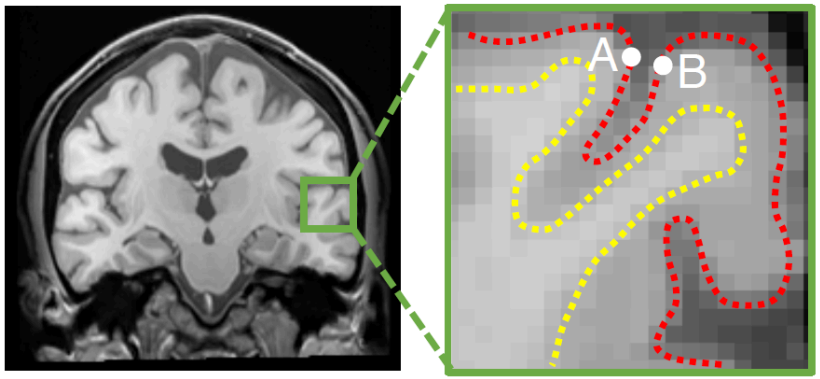


Image: Kohl et al. "A probabilistic U-Net for segmentation of ambiguous images." NIPS, 2018.

# GRAPH CONVOLUTIONS: LEARNING BEYOND IMAGES

# Graph convolutions

**Motivation: learning on surfaces (cortex)**




**Problem:** Two adjacent points in an image are not necessarily close on the surface

**Solution**: Model the surface as a mesh graph and perform learning on this graph

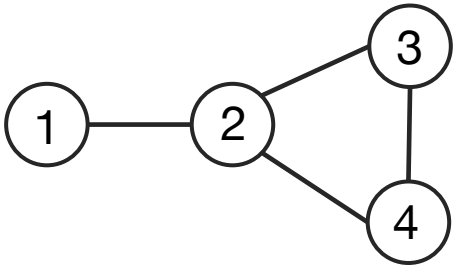


Image: Kohl et al. "A probabilistic U-Net for segmentation of ambiguous images." NIPS, 2018.

# Graph convolutions

**Graph Laplacian matrix**

$$D_{ii} = \sum_j A_{ij} \quad \text{(node degree)}$$




$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$$L_{\text{norm}} = I - D^{-1/2} A D^{-1/2}$$

**Spectral graph convolution**

Graph Fourier transform

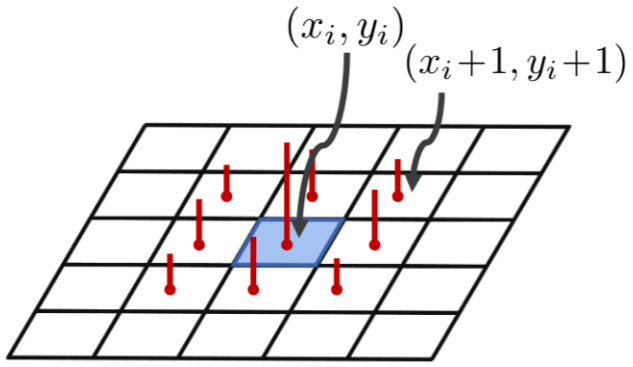
$$L_{\text{norm}} = U \Lambda U^T$$

$$x *_G g = \mathcal{F}^{-1}\big(\mathcal{F}(x) \odot \mathcal{F}(g)\big)$$

$$= U\big(U^T x \odot U^T g\big)$$

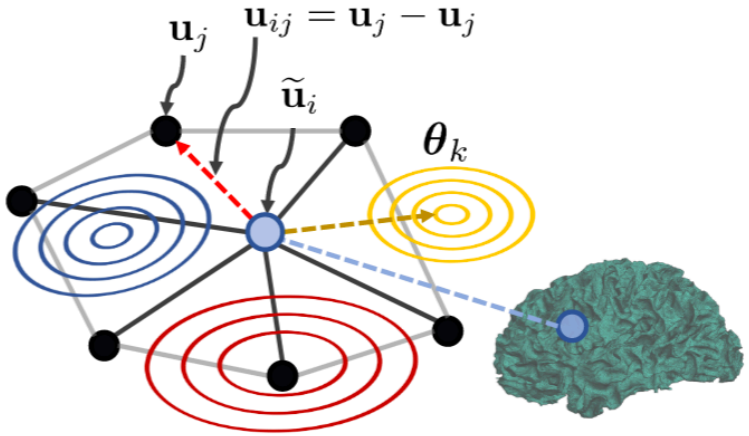
# Graph convolutions

## Convolution on a grid




$$z_{ip}^{(l)} = \sum_{q=1}^{M_l} \sum_{k=1}^{K_l} w_{pqk}^{(l)} \cdot y_{i+k,\,q}^{(l)} + b_p^{(l)}$$

## Geometric graph convolution




$$z_{ip}^{(l)} = \sum_{j \in \mathcal{N}_i} \sum_{q=1}^{M_l} \sum_{k=1}^{K_l} w_{pqk}^{(l)} \cdot y_{jq}^{(l)} \cdot \varphi_{ij}(\boldsymbol{\theta}_k^{(l)}) + b_p^{(l)}$$

$$\varphi_{ij}(\boldsymbol{\theta}_k) = \exp\left( -\tfrac{1}{2} (\mathbf{u}_{ij} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{u}_{ij} - \boldsymbol{\mu}_k) \right)$$

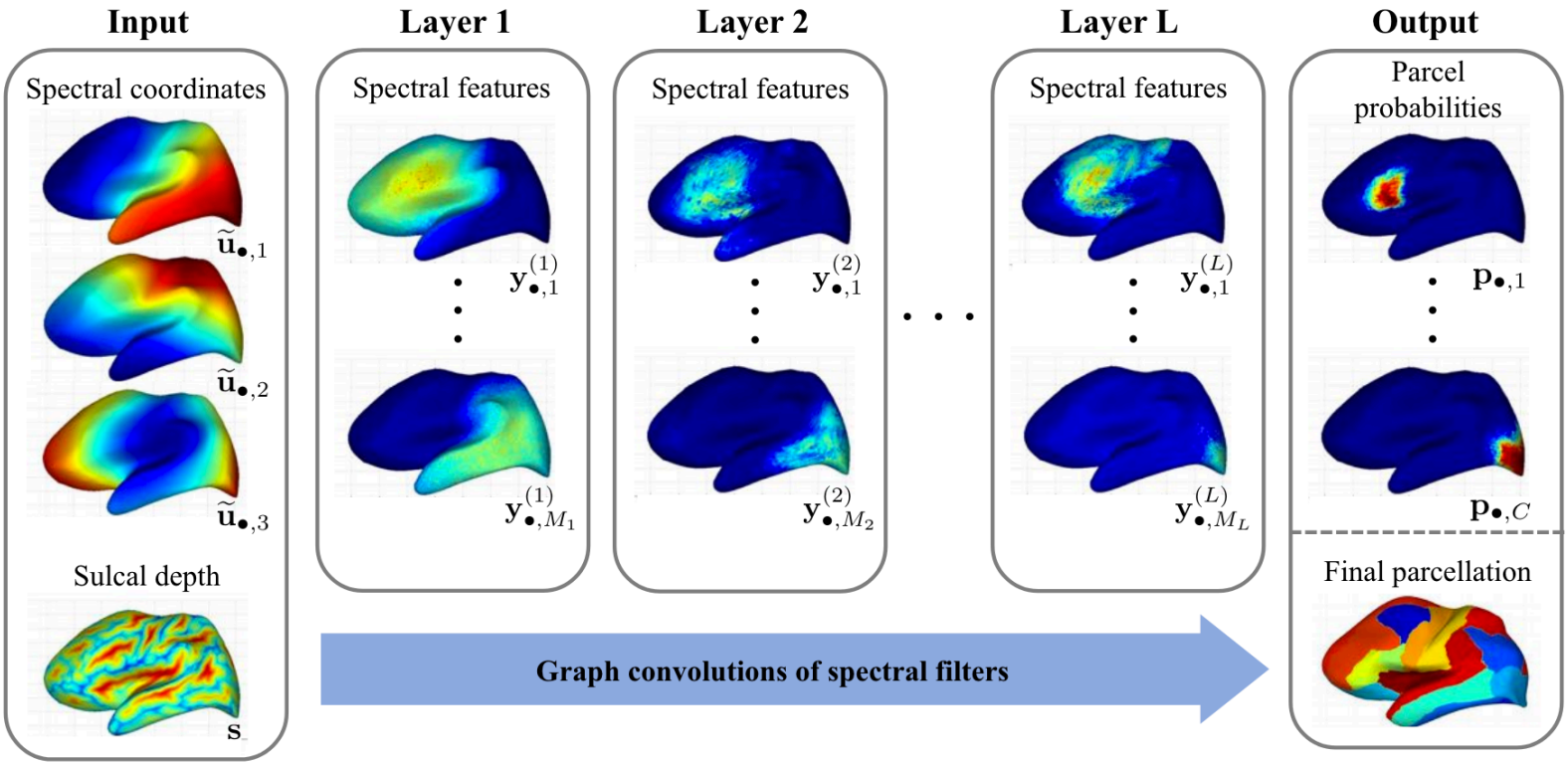
Parameters mu and sigma are learned



Image: Gopinath et al. "Adaptive Graph Convolution Pooling for Brain Surface Analysis. " IPMI, 2019 (to appear).

# Graph convolutions

**Cortical surface parcellation [Gopinath et al., 2019]**




What about classification ?

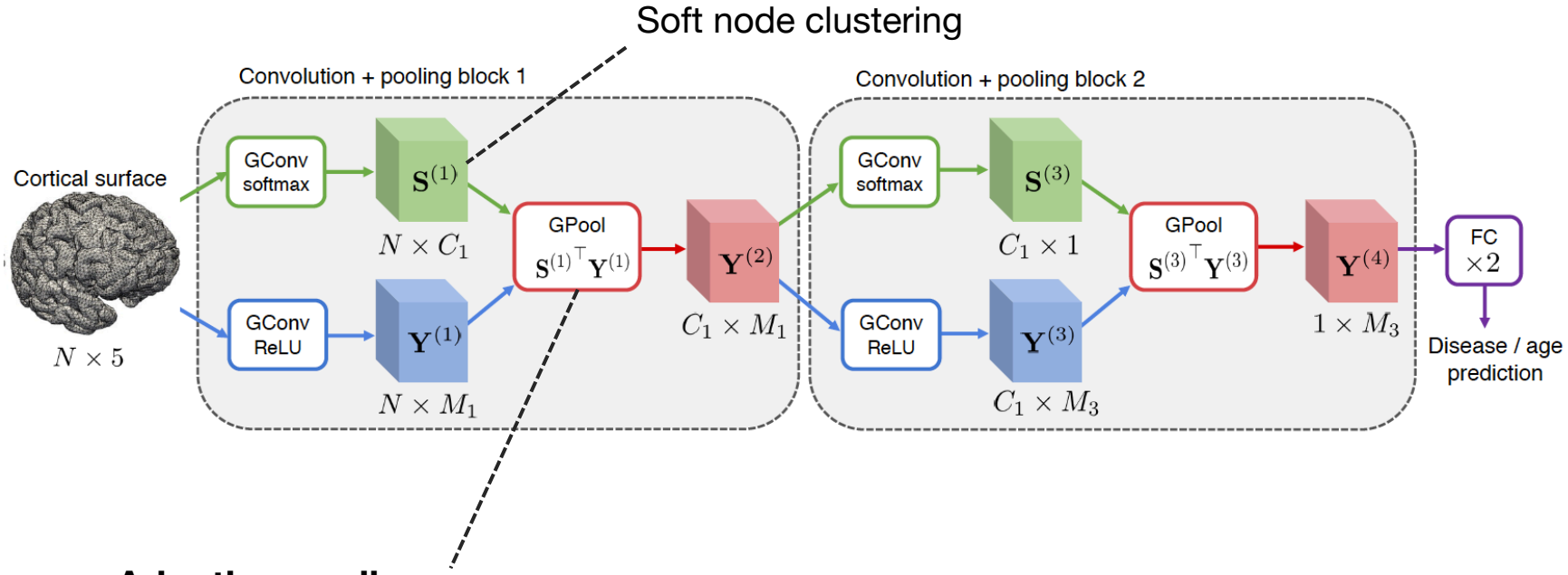
We need to do pooling on the graph



Image: Gopinath et al. "Graph Convolutions on Spectral Embeddings for Cortical Surface Parcellation. " Medical Image Analysis, 2019

# Graph convolutions

## Cortical surface classification [Gopinath et al., 2019]




**Adaptive pooling:**

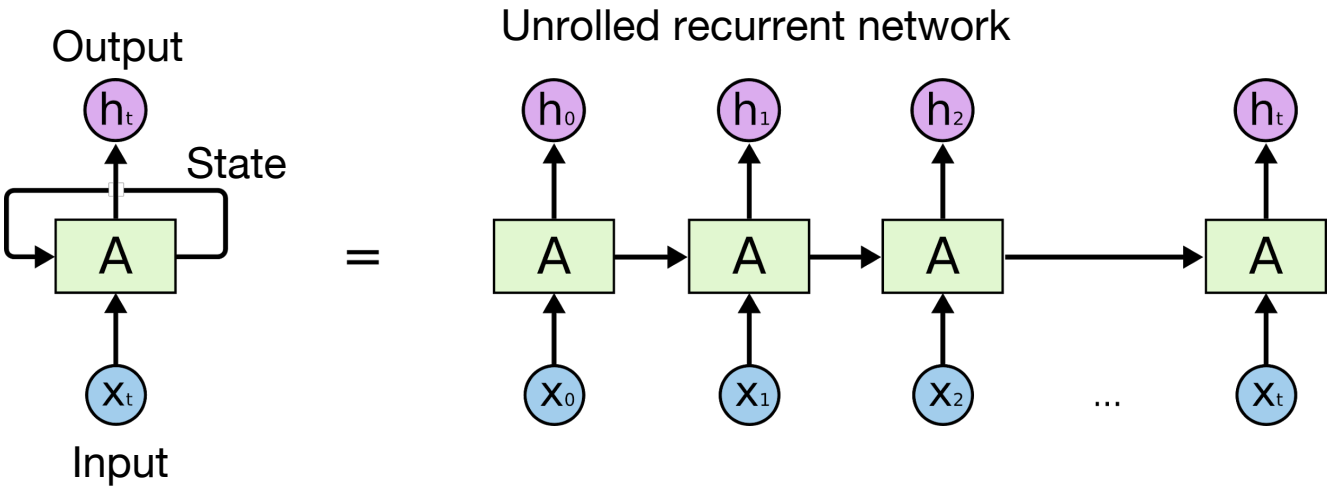
- Soft grouping of graph nodes into learned clusters
- Features are averaged within each cluster
- Fixed-size output, regardless of input size



Image: Gopinath et al. "Adaptive Graph Convolution Pooling for Brain Surface Analysis. " IPMI, 2019 (to appear).

# RECURRENT NEURAL NETWORKS

# Recurrent neural networks




**Key ideas:**

- Consider previous predictions (state) when doing a new prediction
- Can process input sequences of varied length
- Make network recurrent by unrolling loop (chaining multiple copies of the network)

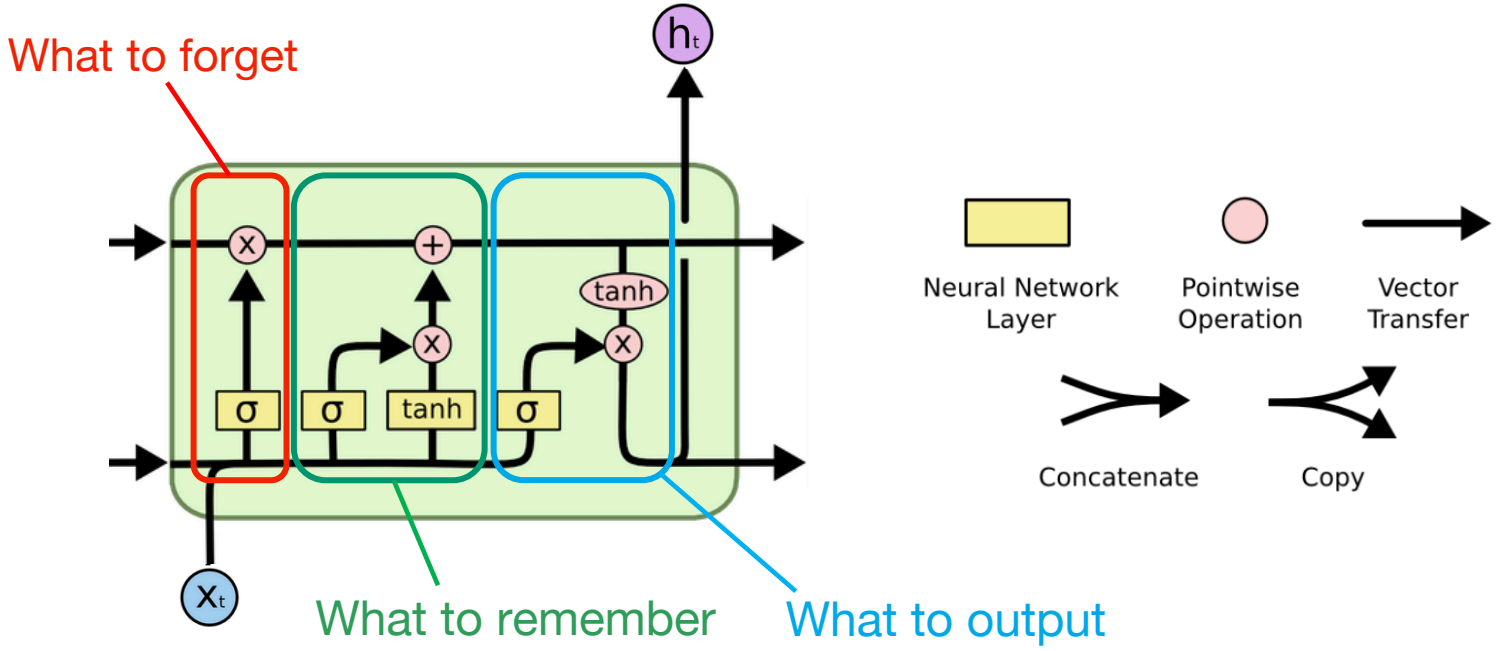
**Problem:** Can have problem learning long-term dependencies



Image: Colah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Recurrent neural networks

**Long Short Term Memory (LSTM) network**




**Key ideas:**

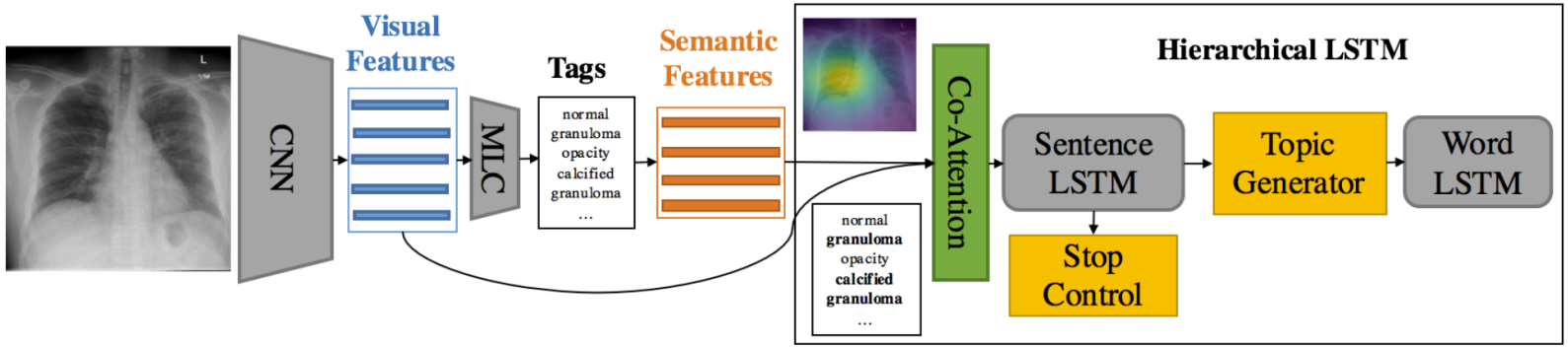
- Proposed in 1997 by Hochreiter and Schmidhuber
- Long-term dependencies are handled using a memory
- Decide dynamically what to remember and forget



Image: Colah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Recurrent neural networks

## Generation of Medical Imaging Reports [Zhang et al., 2017]




| | | | | |
|---|---|---|---|---|
| normal | normal; **calcified granuloma; granulomatous disease; granuloma;** scarring; opacity; degenerative change; sternotomy; thoracic aorta; nodule | **normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change;** sternotomy; thoracic aorta; **nodule** | **normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change;** sternotomy; thoracic aorta; **nodule** | **normal;** calcified granuloma; granulomatous disease; granuloma; scarring; opacity; **degenerative change; sternotomy;** thoracic aorta; nodule |
| | Right upper lobe infiltrate. | Lungs are clear . | Stable heart size and aortic contours. | No acute displaced rib fractures. |



Images: Jing et al. "On the automatic generation of medical imaging reports." arXiv, 2017.