

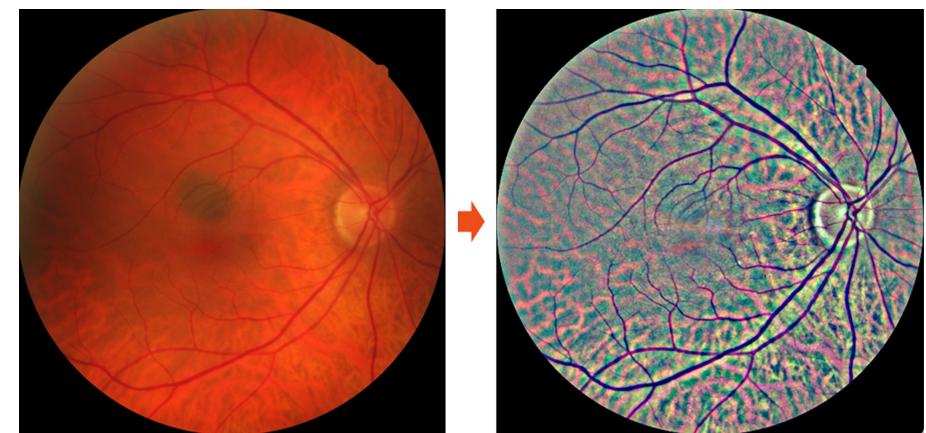
# Deep Learning II

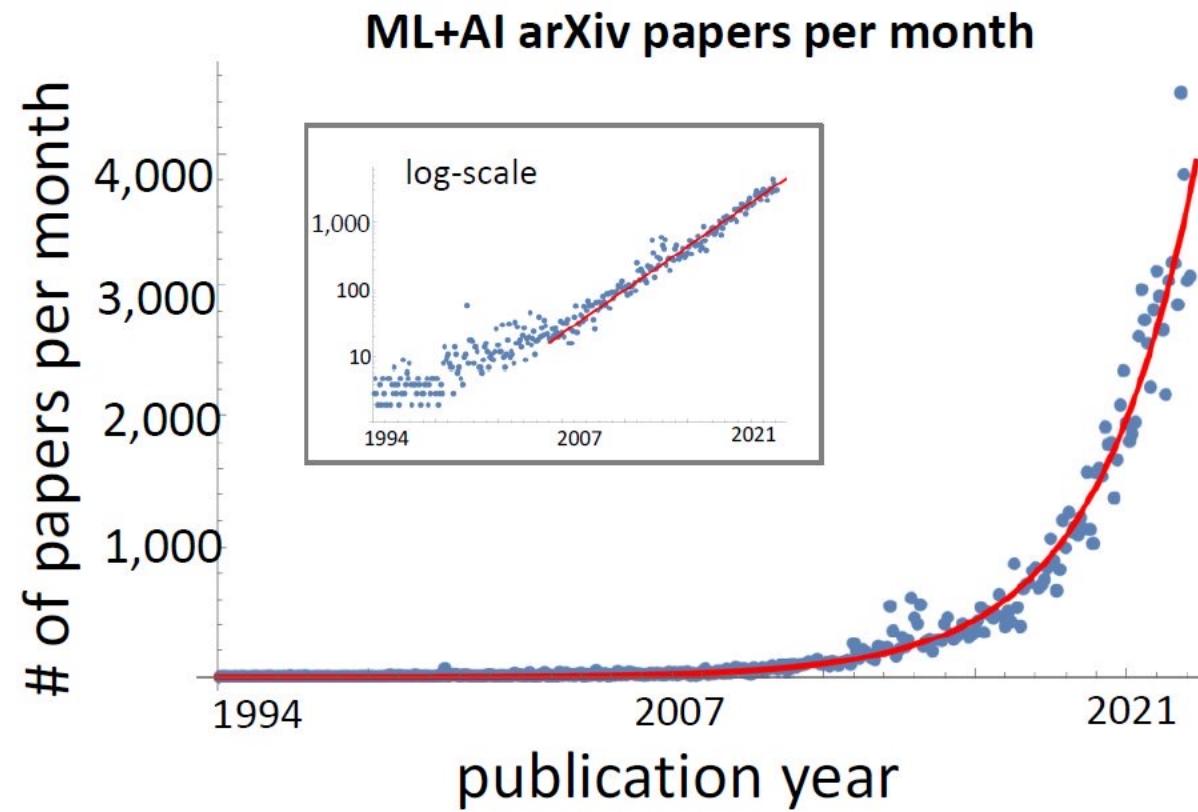
8DM50 Machine Learning in Medical Imaging and Biology

Mitko Veta

Acknowledgements: slides by Jelmer Wolterink, UTwente

| #  | △   | Team                           | Members | Score   | Entries | Last | Code |
|----|-----|--------------------------------|---------|---------|---------|------|------|
| 1  | —   | Min-Pooling                    |         | 0.84957 | 37      | 7Y   |      |
| 2  | —   | o_O                            |         | 0.84478 | 61      | 7Y   |      |
| 3  | —   | Reformed Gamblers              |         | 0.83936 | 58      | 7Y   |      |
| 4  | —   | Julian de Wit & Daniel Hammack |         | 0.83625 | 49      | 7Y   |      |
| 5  | —   | Jeffrey De Fauw                |         | 0.82898 | 133     | 7Y   |      |
| 6  | —   | DeepSense.io                   |         | 0.82854 | 88      | 7Y   |      |
| 7  | —   | Zhanpeng Zhang                 |         | 0.82499 | 148     | 7Y   |      |
| 8  | ▲ 1 | Plankton                       |         | 0.82222 | 14      | 7Y   |      |
| 9  | ▼ 1 | Halla Yang                     |         | 0.82165 | 69      | 7Y   |      |
| 10 | —   | Sungbin Choi                   |         | 0.81405 | 41      | 7Y   |      |
| 11 | —   | [RU.nl] AI for an Eye          | +2      | 0.80536 | 32      | 7Y   |      |
| 12 | ▲ 2 | Ryan Munion                    |         | 0.79638 | 29      | 7Y   |      |
| 13 | —   | Dan Nuffer                     |         | 0.79536 | 20      | 7Y   |      |
| 14 | ▼ 2 | Tim Hochberg                   |         | 0.78607 | 122     | 7Y   |      |
| 15 | ▲ 2 | Fusion Systems                 |         | 0.78572 | 5       | 7Y   |      |
| 16 | —   | brainsignals                   |         | 0.77767 | 13      | 7Y   |      |
| 17 | ▼ 2 | Tu/e                           |         | 0.77496 | 13      | 7Y   |      |





# **Learning goals for today**

**Have a good overview of neural networks architectures for (medical) image analysis**

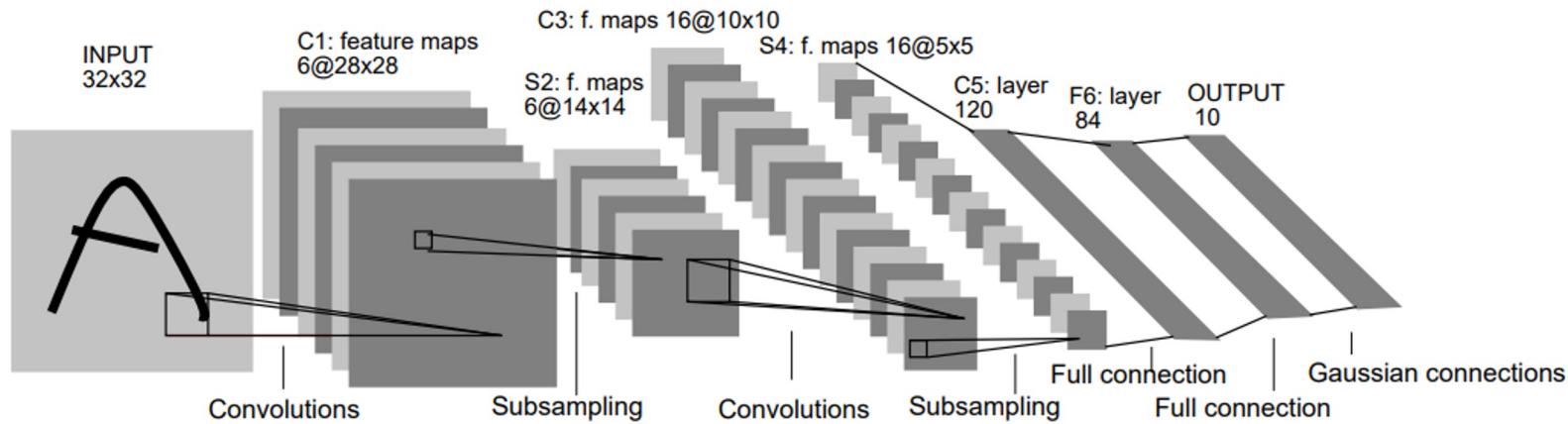
- Including architectures for sequential data**

Obtain a general understanding of interpretability of deep learning models

Obtain a general understanding of generative adversarial networks

- Only one example of generative models, but arguably the most successful one**

# Recap: Convolutional neural networks



[Demo](#)

- A standard convolutional network consists of
- **Convolutional layers** transform input into feature maps
  - **Subsampling operations** reduce size of feature maps, e.g. max pooling
  - **Fully-connected layers** perform classification (multi-layered perceptron)



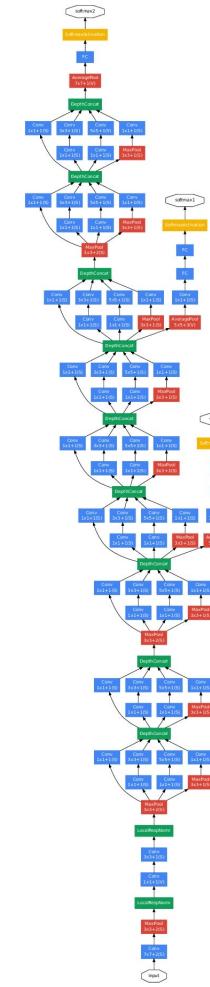
# What happened between 1998 and 2012?



Compute



Data



Algorithms

# Data: ImageNet challenge

Benchmark for image classification/object detection

## Data

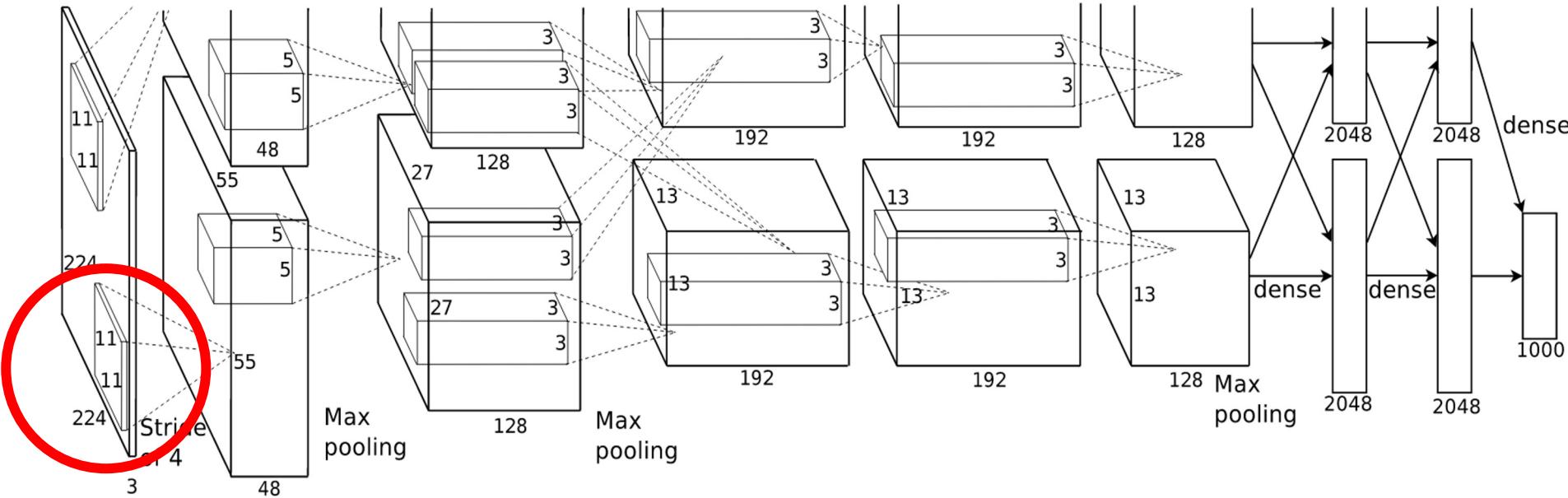
- > 1,200,000 RGB images
- Images show one of **1000** classes

## Task

- Detect label of image
- Top-1\top-5 accuracy



# AlexNet



- Substantially outperformed 'conventional' methods in 2012
- Convolutional + subsampling + fully connected layers
- Trained in parallel on two GPUs
- Training time
  - **2012:** 5 to 6 days (2 x GTX 580 3GB GPU)
  - **2017:** 24 minutes (supercomputer 32,000 cores)
- Large **11 x 11** convolution kernels

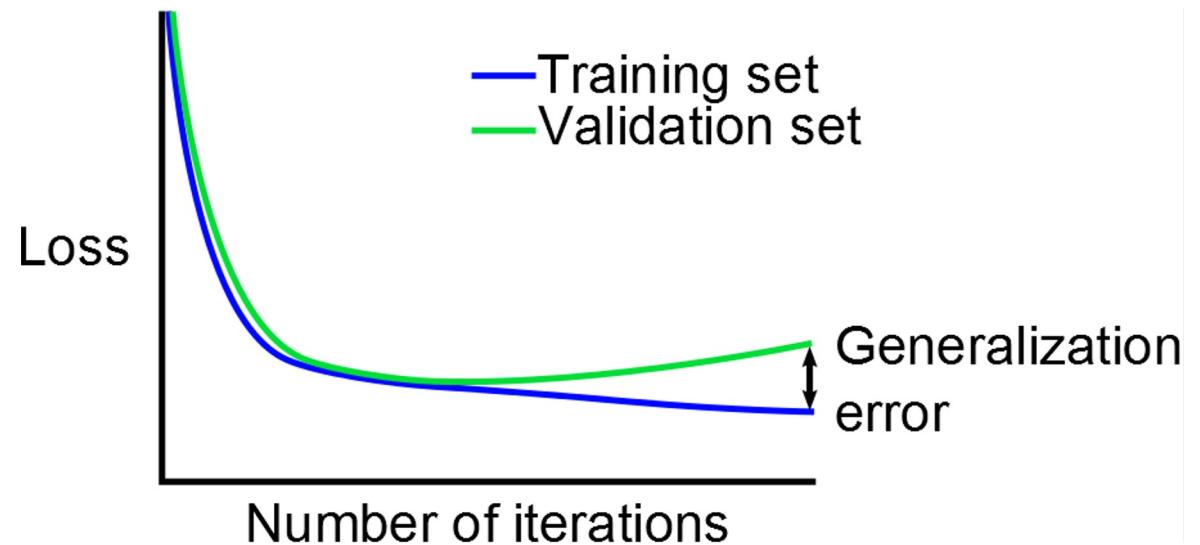
# Overfitting

## Reasons

- Too many parameters
- Not enough data

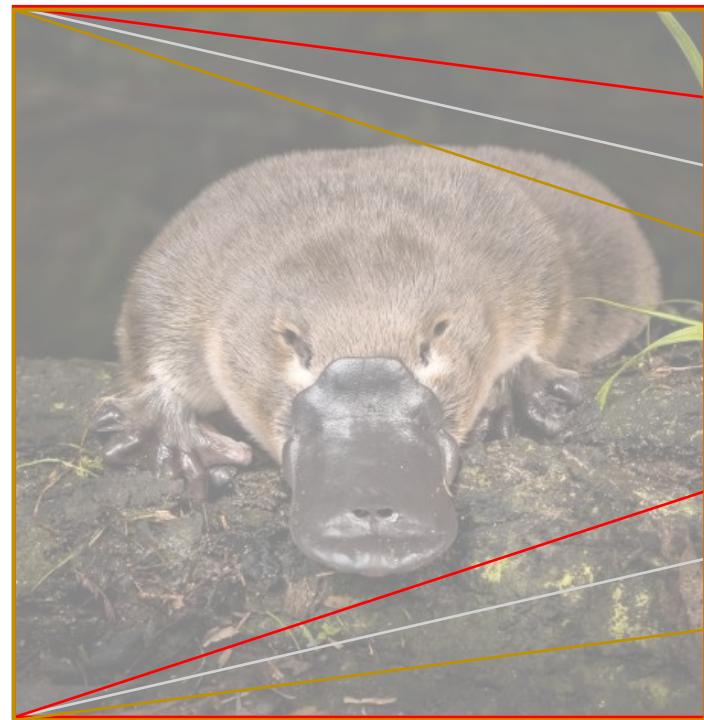
## One solution

- Reduce number of parameters



# Kernel size

100



**10,000 weights**

100

Input

Hidden

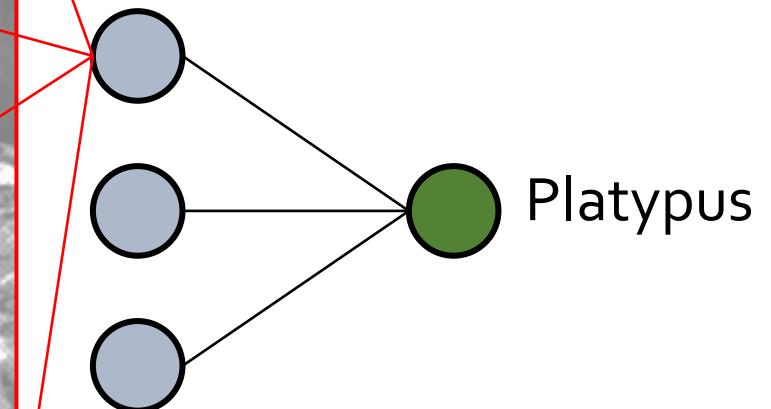
Output

Platypus

# Kernel size

100

1 weight



Input

Hidden

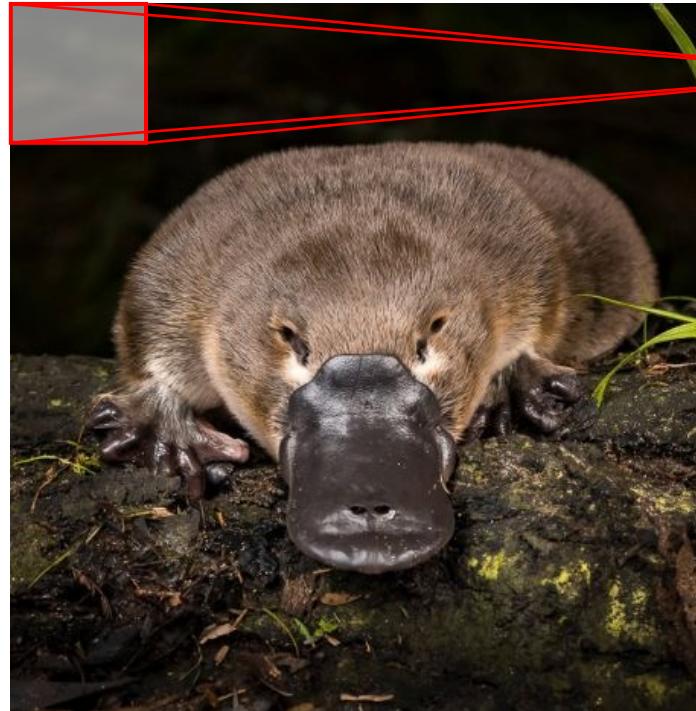
Hidden

Output

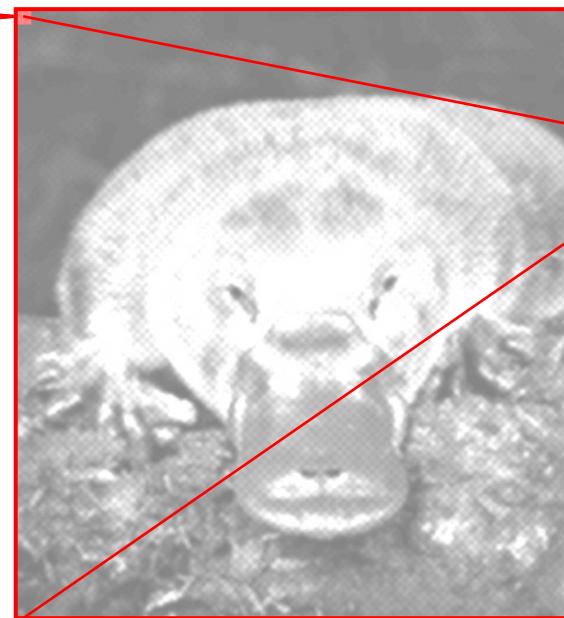
# Kernel size

100

**11 x 11 = 121 weights**



100

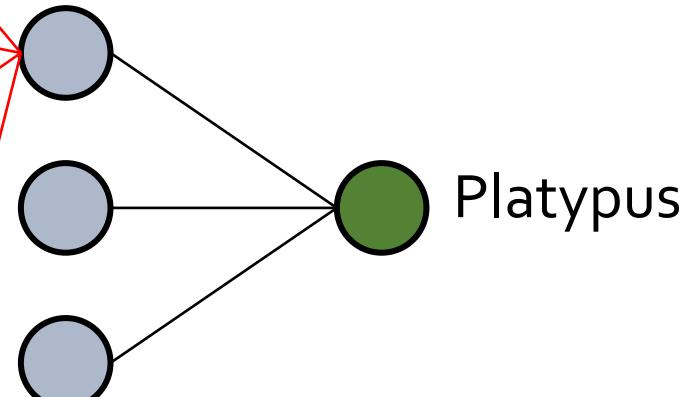


Input

Hidden

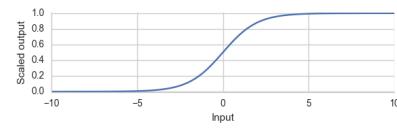
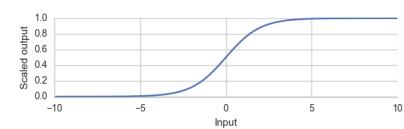
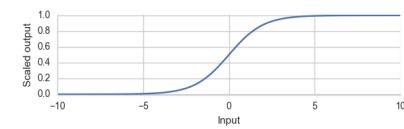
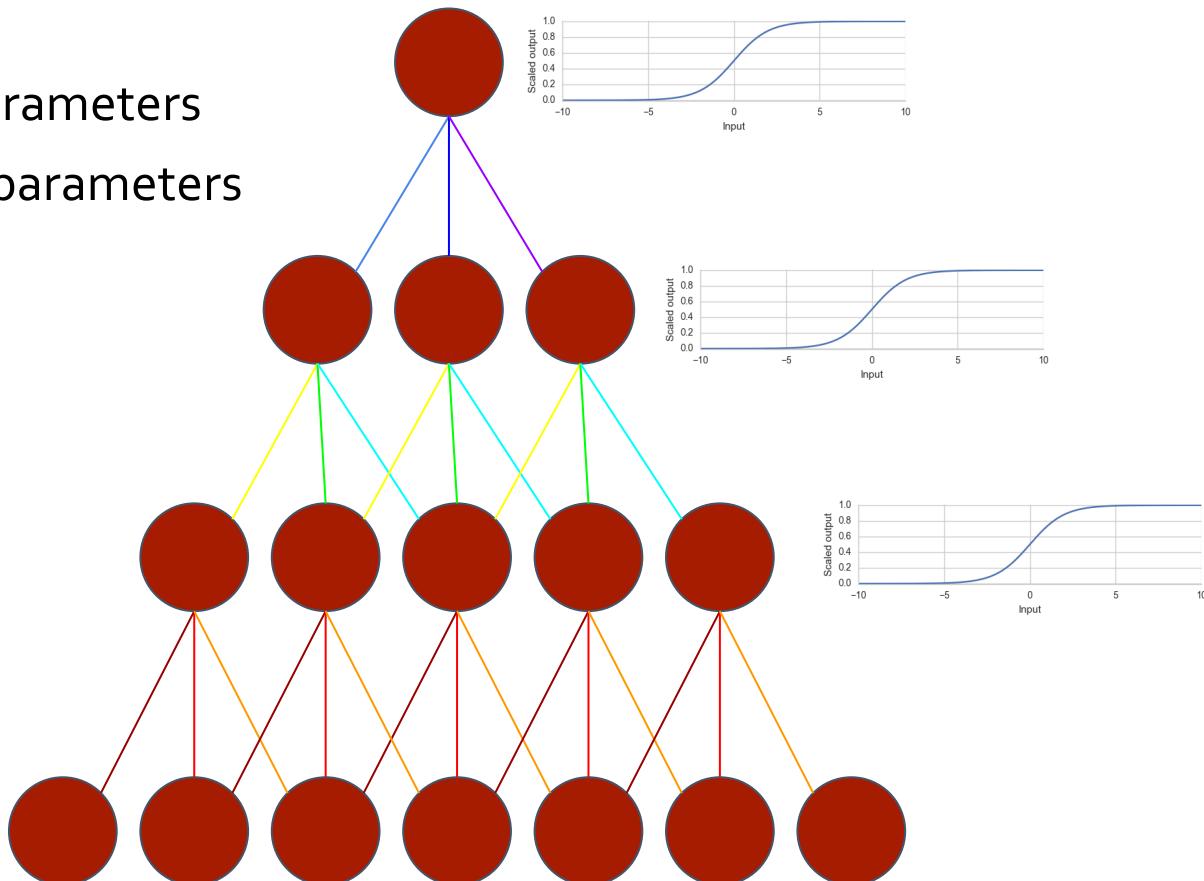
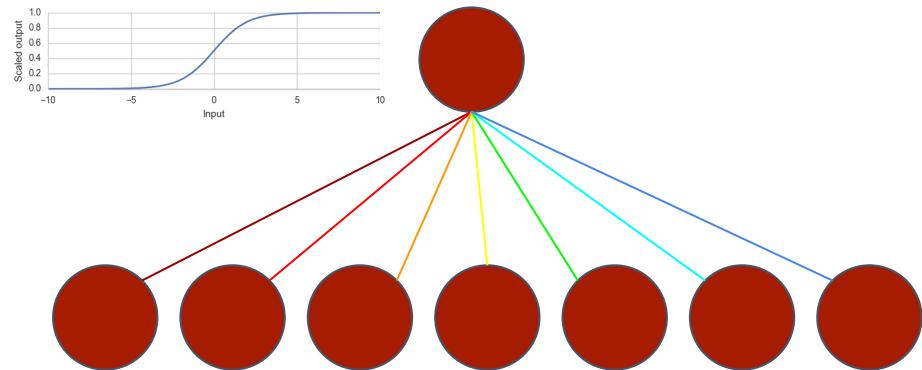
Hidden

Output

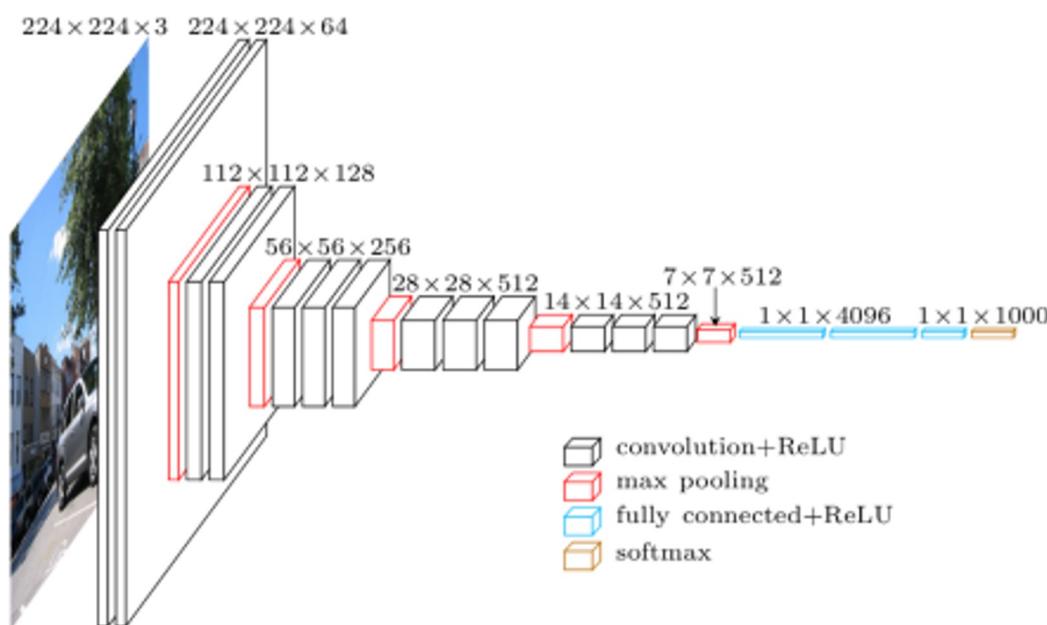


# Using smaller kernels

- Large kernels have many parameters:  $7 \times 7 = 49$  parameters
- Smaller kernels reduce parameters:  $3 \times (3 \times 3) = 27$  parameters
- More nonlinearities means more abstraction



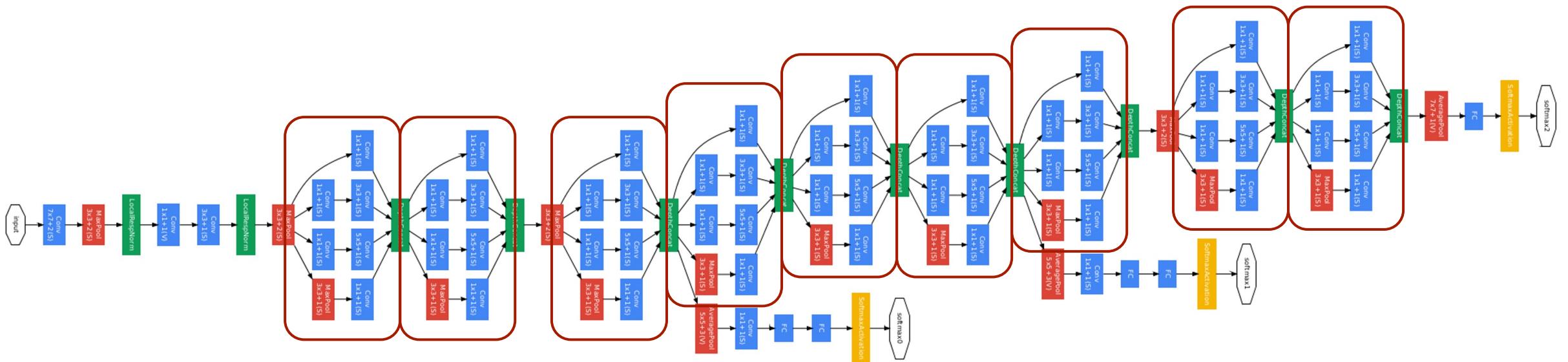
# VGG-Net



| ConvNet Configuration       |                        |                               |                                            |                                            |                                                         |
|-----------------------------|------------------------|-------------------------------|--------------------------------------------|--------------------------------------------|---------------------------------------------------------|
| A                           | A-LRN                  | B                             | C                                          | D                                          | E                                                       |
| 11 weight layers            | 11 weight layers       | 13 weight layers              | 16 weight layers                           | 16 weight layers                           | 19 weight layers                                        |
| input (224 × 224 RGB image) |                        |                               |                                            |                                            |                                                         |
| conv3-64                    | conv3-64<br>LRN        | conv3-64<br><b>conv3-64</b>   | conv3-64<br>conv3-64                       | conv3-64<br>conv3-64                       | conv3-64<br>conv3-64                                    |
| maxpool                     |                        |                               |                                            |                                            |                                                         |
| conv3-128                   | conv3-128              | conv3-128<br><b>conv3-128</b> | conv3-128<br>conv3-128                     | conv3-128<br>conv3-128                     | conv3-128<br>conv3-128                                  |
| maxpool                     |                        |                               |                                            |                                            |                                                         |
| conv3-256<br>conv3-256      | conv3-256<br>conv3-256 | conv3-256<br>conv3-256        | conv3-256<br>conv3-256<br><b>conv1-256</b> | conv3-256<br>conv3-256<br><b>conv3-256</b> | conv3-256<br>conv3-256<br>conv3-256<br><b>conv3-256</b> |
| maxpool                     |                        |                               |                                            |                                            |                                                         |
| conv3-512<br>conv3-512      | conv3-512<br>conv3-512 | conv3-512<br>conv3-512        | conv3-512<br>conv3-512<br><b>conv1-512</b> | conv3-512<br>conv3-512<br><b>conv3-512</b> | conv3-512<br>conv3-512<br>conv3-512<br><b>conv3-512</b> |
| maxpool                     |                        |                               |                                            |                                            |                                                         |
| conv3-512<br>conv3-512      | conv3-512<br>conv3-512 | conv3-512<br>conv3-512        | conv3-512<br>conv3-512<br><b>conv1-512</b> | conv3-512<br>conv3-512<br><b>conv3-512</b> | conv3-512<br>conv3-512<br>conv3-512<br><b>conv3-512</b> |
| maxpool                     |                        |                               |                                            |                                            |                                                         |
| FC-4096                     |                        |                               |                                            |                                            |                                                         |
| FC-4096                     |                        |                               |                                            |                                            |                                                         |
| FC-1000                     |                        |                               |                                            |                                            |                                                         |
| soft-max                    |                        |                               |                                            |                                            |                                                         |

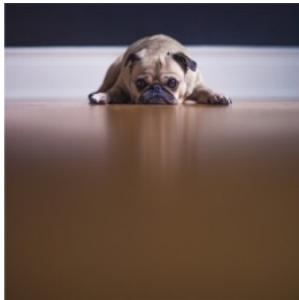
# GoogLeNet (Inception v1)

- 22 layer-network
- Very deep compared to LeNet/AlexNet
- SOTA on ImageNet (when published)



# Inception module

- Combine parallel multi-scale convolutions
- Let the model pick best filter size



**1x1**

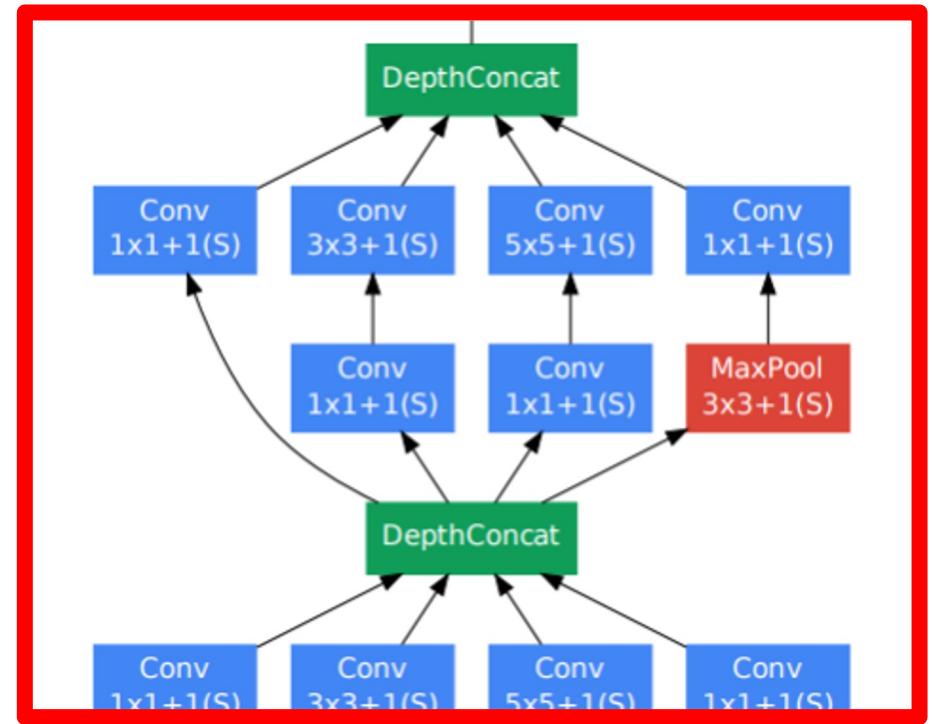


**3x3**



**5x5**

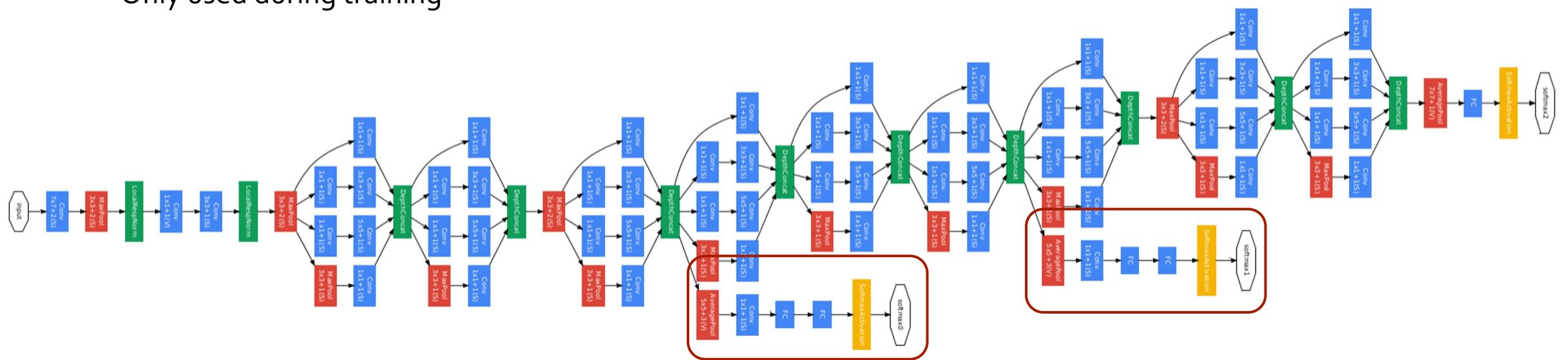
- Bottleneck layers:  $1 \times 1$  convolutions
  - Aggregate feature maps
  - Prevent explosion in number of parameters



# Auxiliary classifiers

Auxiliary classifiers provide extra supervision

- Vanishing gradients
  - Enforce useful features at intermediate layers
  - Only used during training



# Residual connections

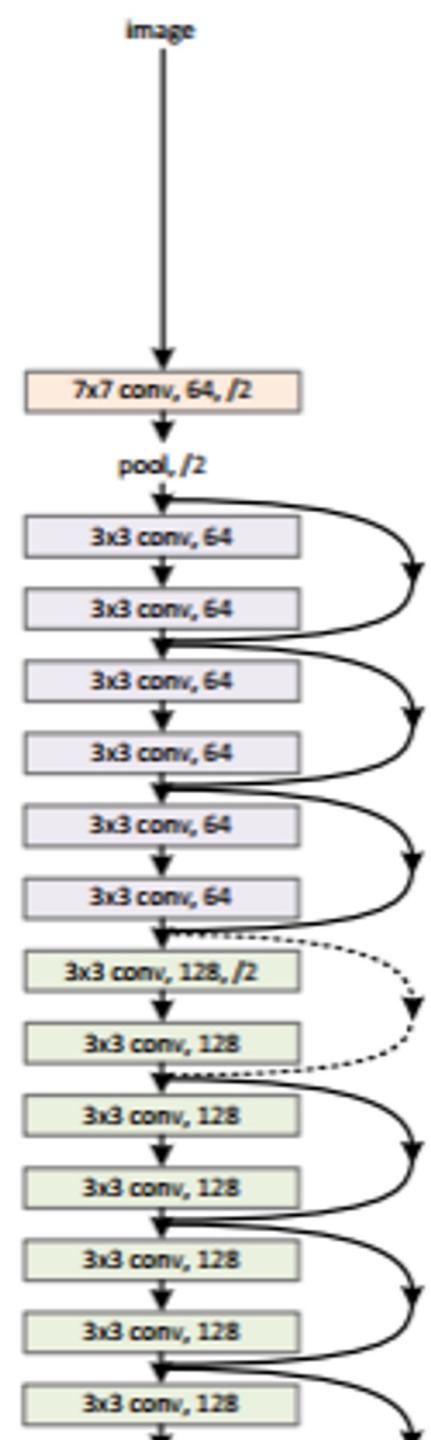
Very deep networks

- allow learning of better representations
- are difficult to optimize due to vanishing gradients

Residual connections can skip layers  $H(x) = x + F(x)$

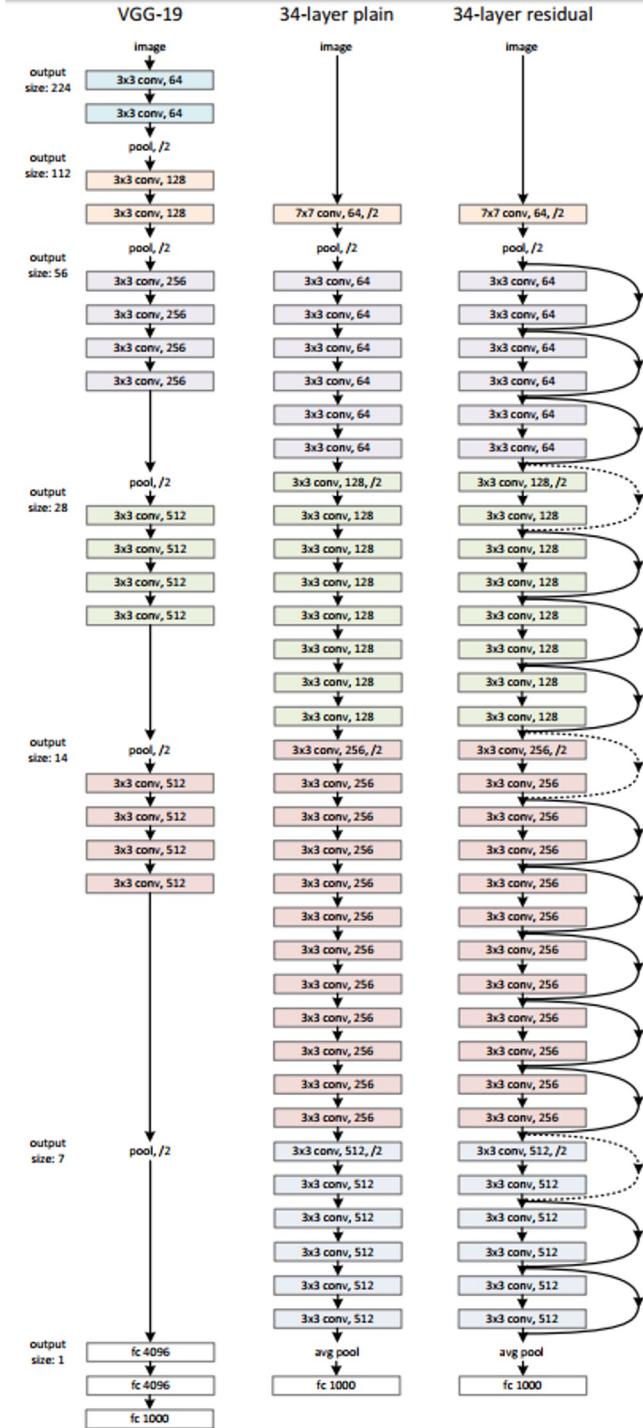
A deep network is at least as strong as its shallower variant

- If adding layers doesn't help, just use the skip connection

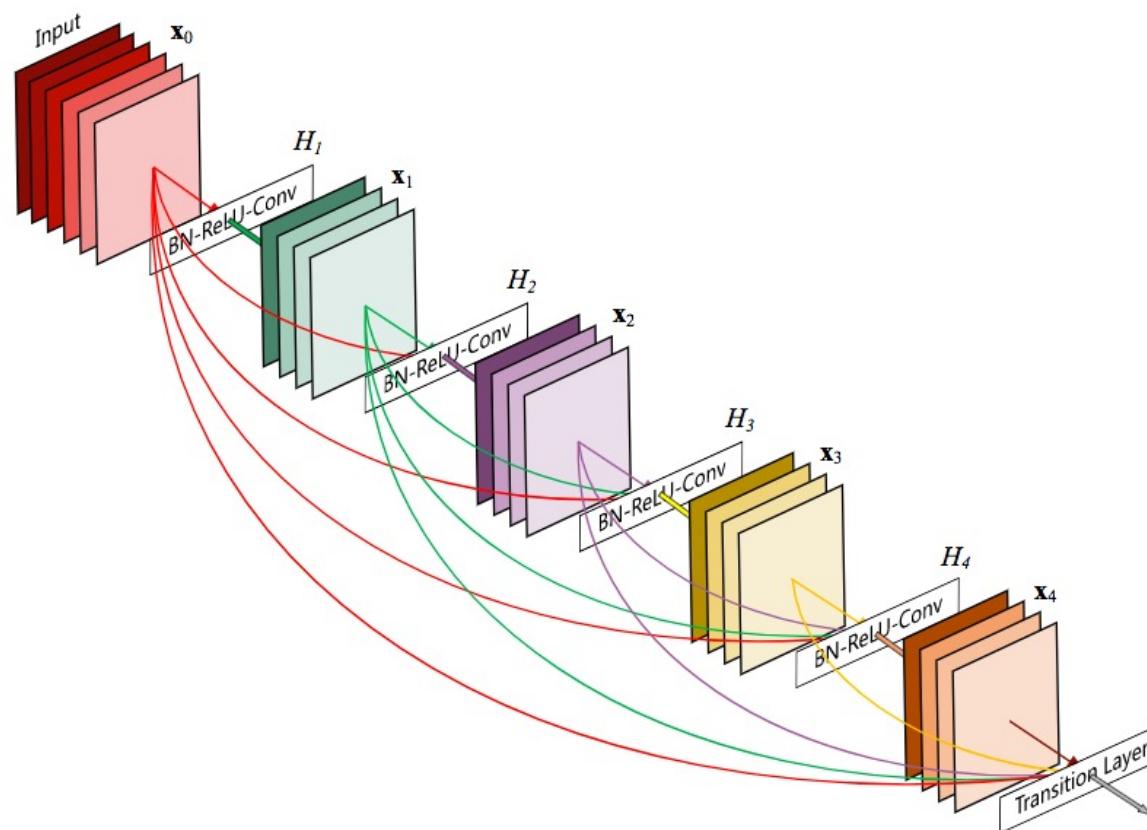


# Residual network (ResNet)

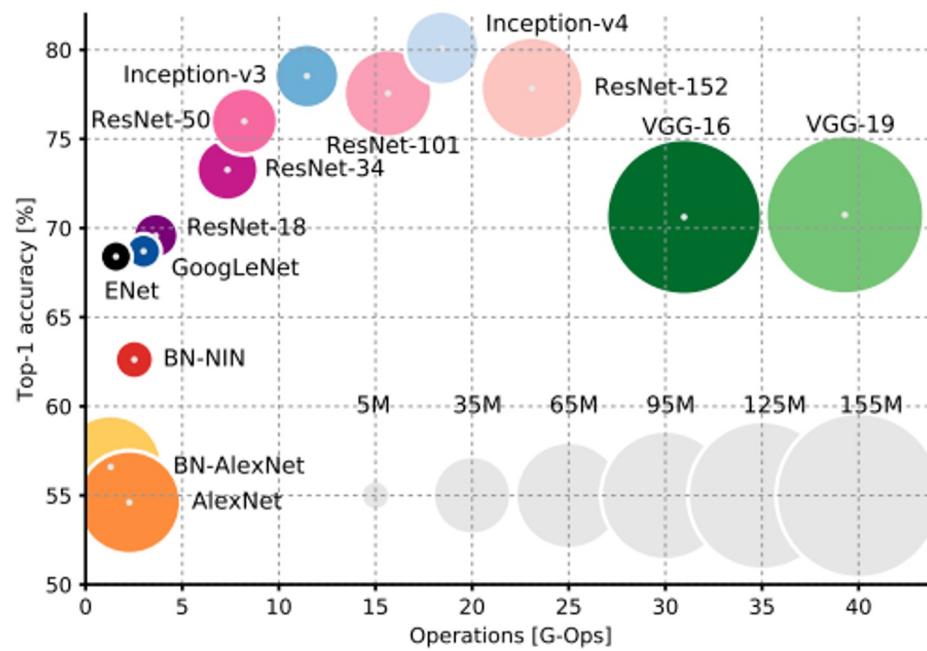
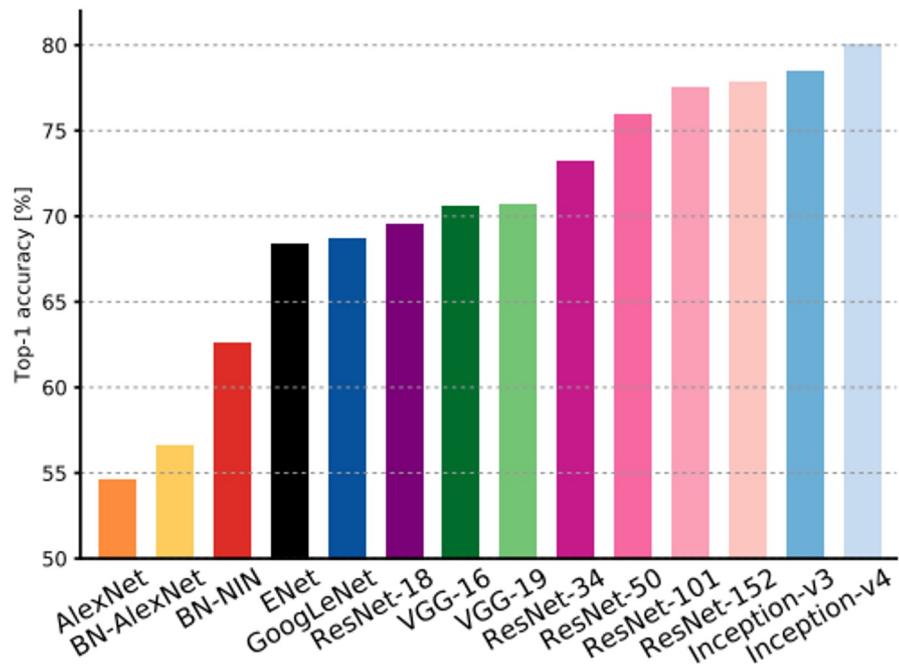
- Organize layers in blocks
- Use bottleneck layers
- Residual connections barely add computational complexity
- SOTA on ImageNet (when published)
- Inspired
  - Wide residual nets (50-layer wide ResNet > 152-layer ResNet)
  - DenseNets: get identity mapping from all previous layers



# Dense networks

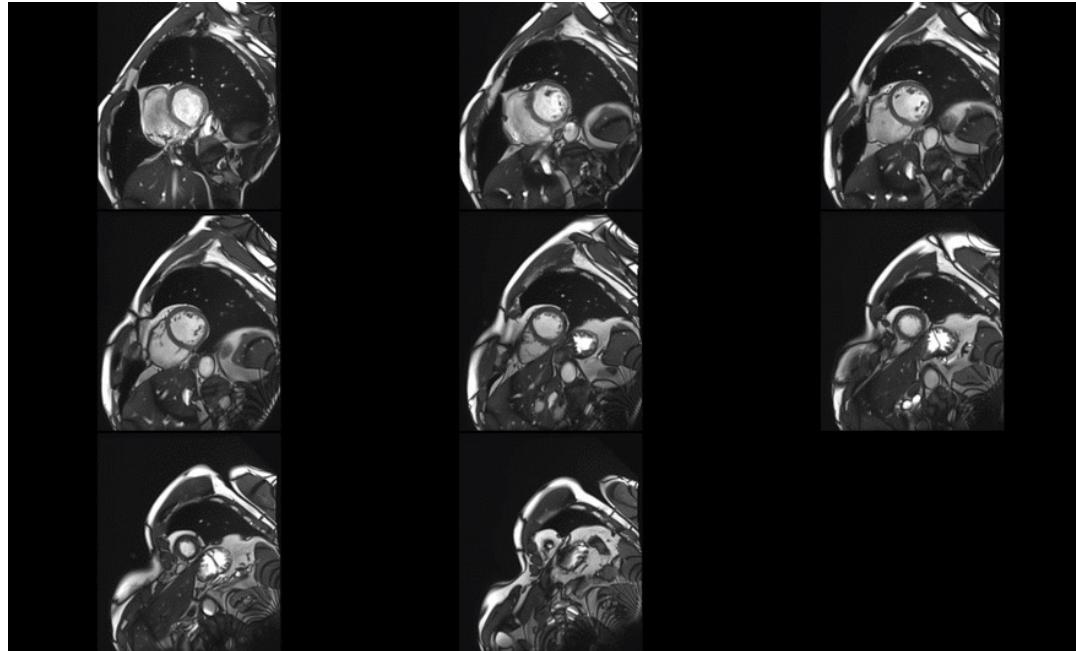


# Complexity vs. accuracy



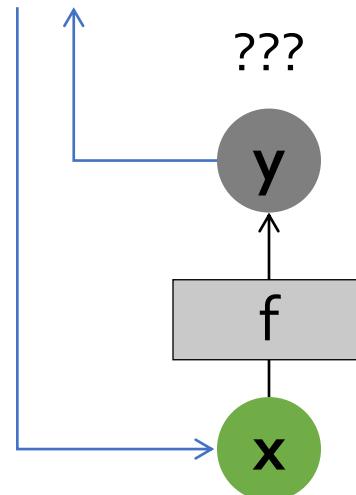
# Sequences

- A lot of data is sequential
- E.g. videos, audio, text, ECG, medical images, ...
- Can we use this in our neural network?



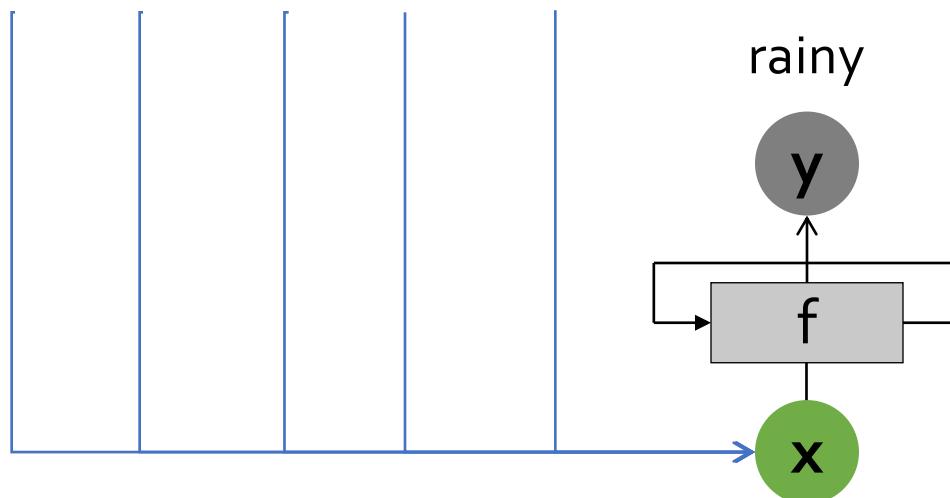
# Recurrent neural networks (RNNs)

- It would be good to use information that came before
- A feedforward neural network has no ‘memory’
- Consider training a neural network to predict the next word
  - “It’s September, the weather is ...”

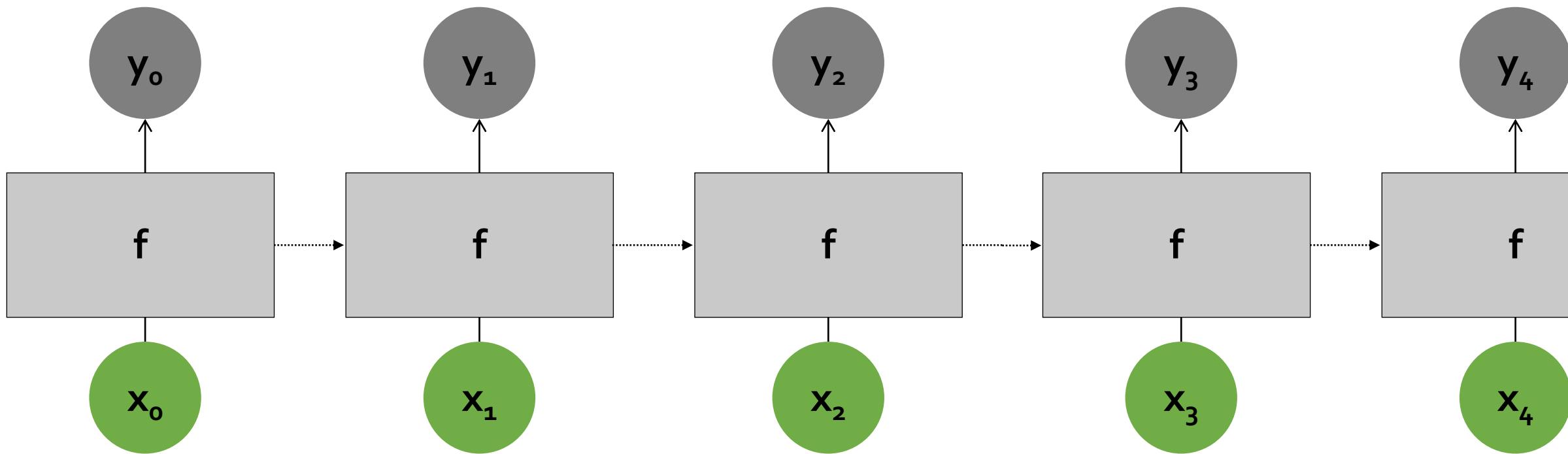


# Recurrent neural networks (RNNs)

- It would be good to use information that came before
- A feedforward neural network has no ‘memory’
- Consider training a neural network to predict the next word
  - “It’s September, the weather is ...”

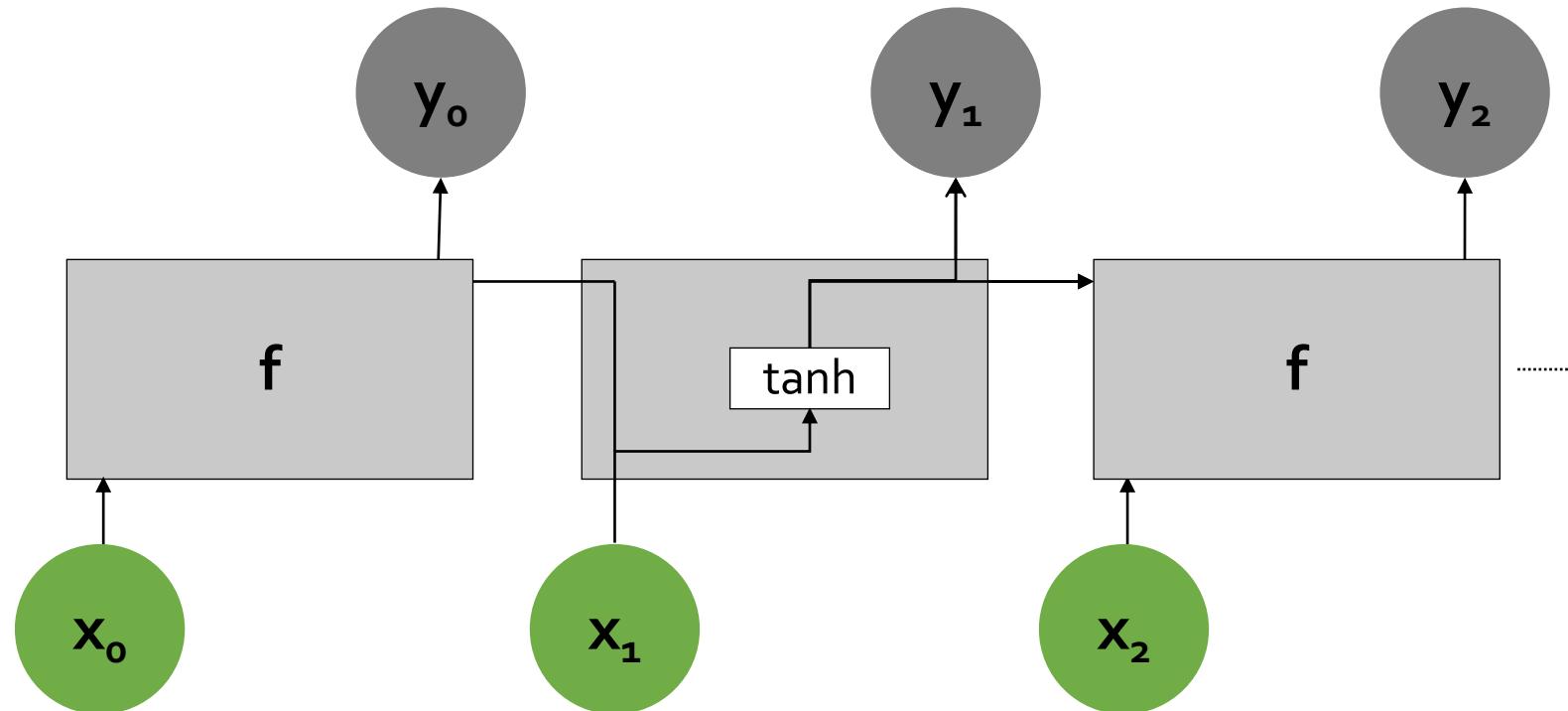


# Unrolling a recurrent neural network



# Unrolling a recurrent neural network

- Traditional RNNs have poor memory
- Previous outputs will get overwritten

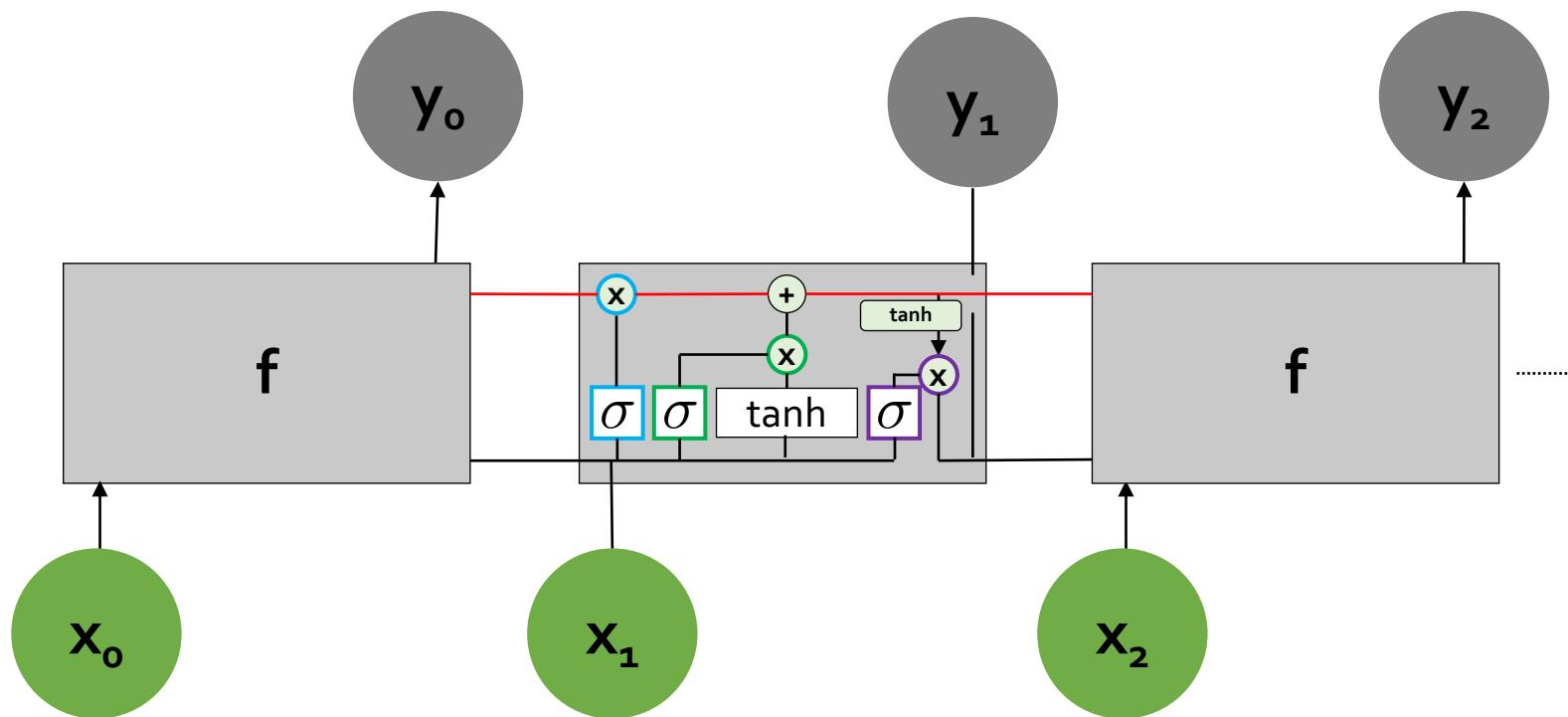


# Long short-term memory (LSTM)

## 1. Cell state

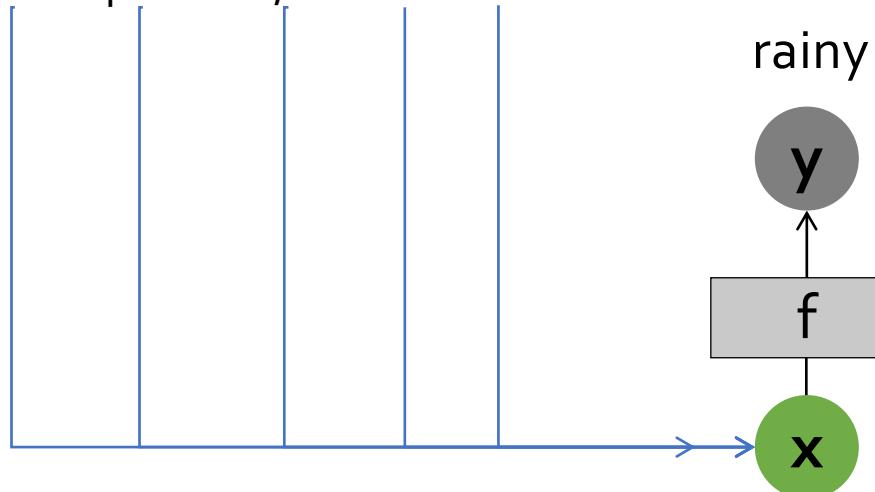
## 2. Gates

- Forget gate
- Input gate
- Output gate



# Recurrent vs. feedforward

- Recurrent networks are intuitively appealing, but
  - feedforward networks are faster (parallel), simpler and they often very competitive
  - “It’s September, the weather is ...”

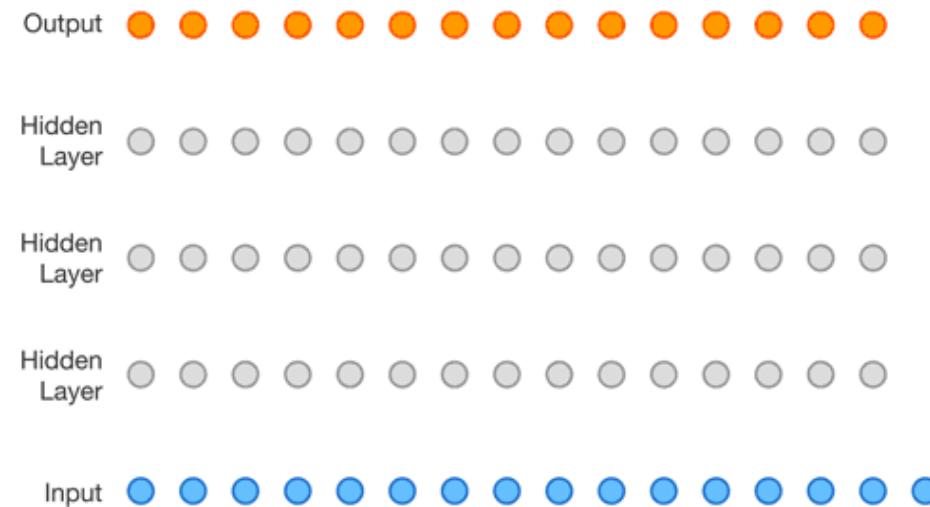


- One way to deal with large contexts in feedforward networks
  - dilated convolutions

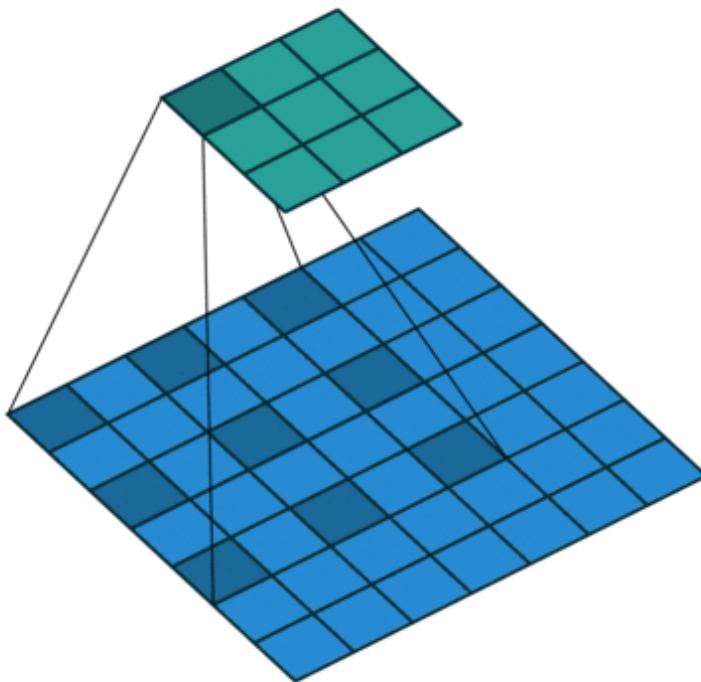
# Dilated convolutions

In each layer, add more spacing between elements

- increase receptive field
- prevent explosion in number of parameters



# Dilated convolutions

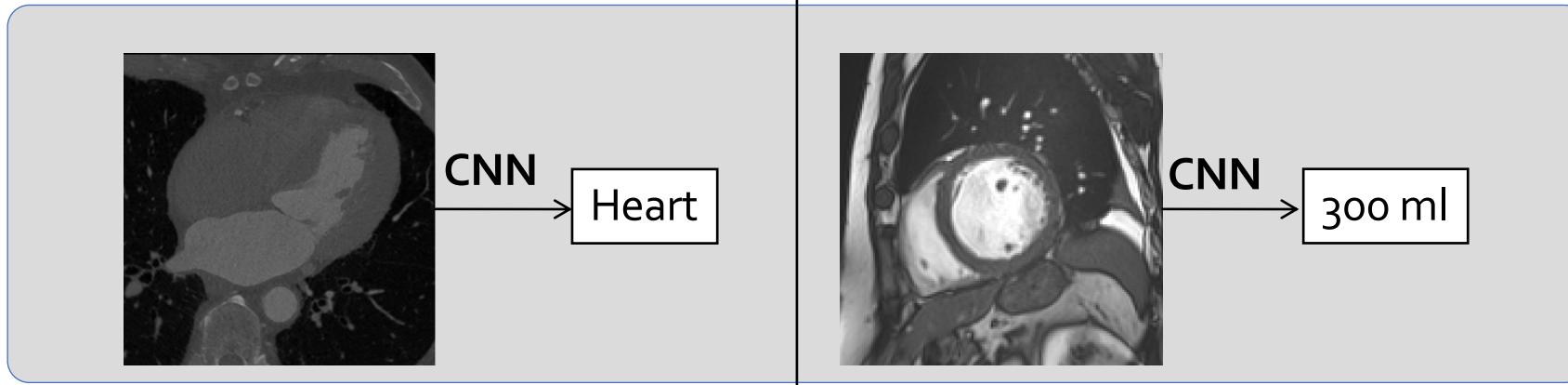


# Back to images

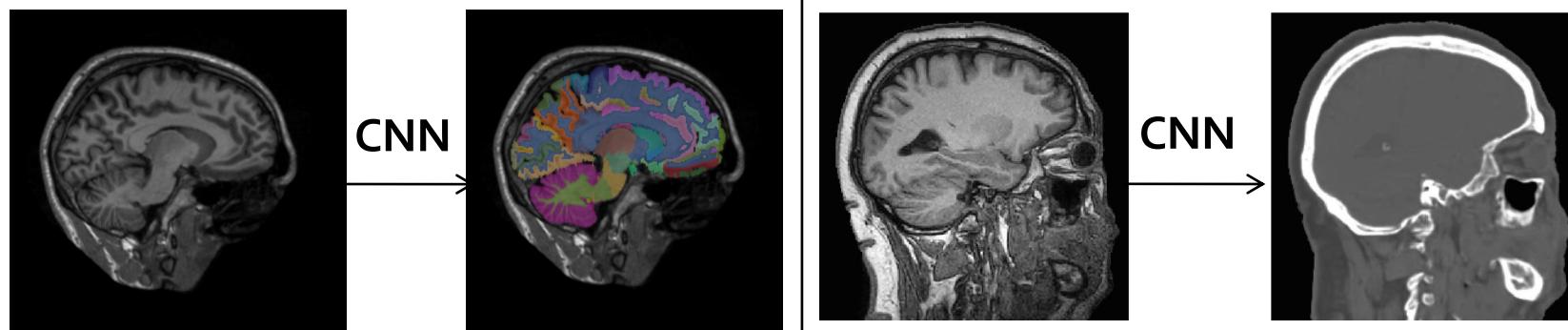
## Classification

## Regression

Image



Voxel



de Vos, Bob D., et al. "ConvNet-based localization of anatomical structures in 3-D medical images." *IEEE Trans Med Imaging* 36.7 (2017): 1470-1481.

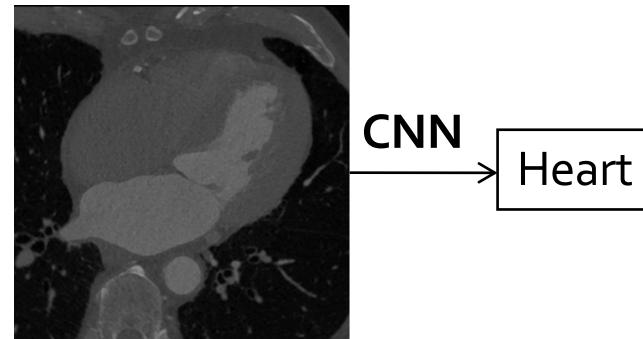
Luo, Gongning, et al. "Multi-views fusion CNN for left ventricular volumes estimation on cardiac MR images." *IEEE Transactions on Biomedical Engineering* 65.9 (2018): 1924-1934.

Moeskops, Pim, et al. "Automatic segmentation of MR brain images with a convolutional neural network." *IEEE transactions on medical imaging* 35.5 (2016): 1252-1261.

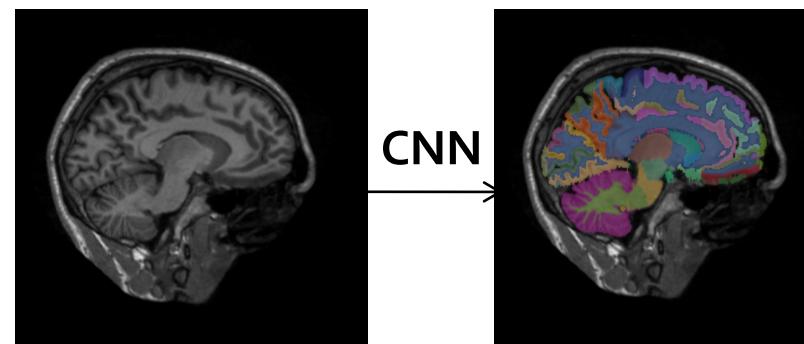
Wolterink, Jelmer M., et al. "Deep MR to CT synthesis using unpaired data." *International Workshop on Simulation and Synthesis in Medical Imaging*. Springer, Cham, 2017.

# CNNs for pixelwise prediction

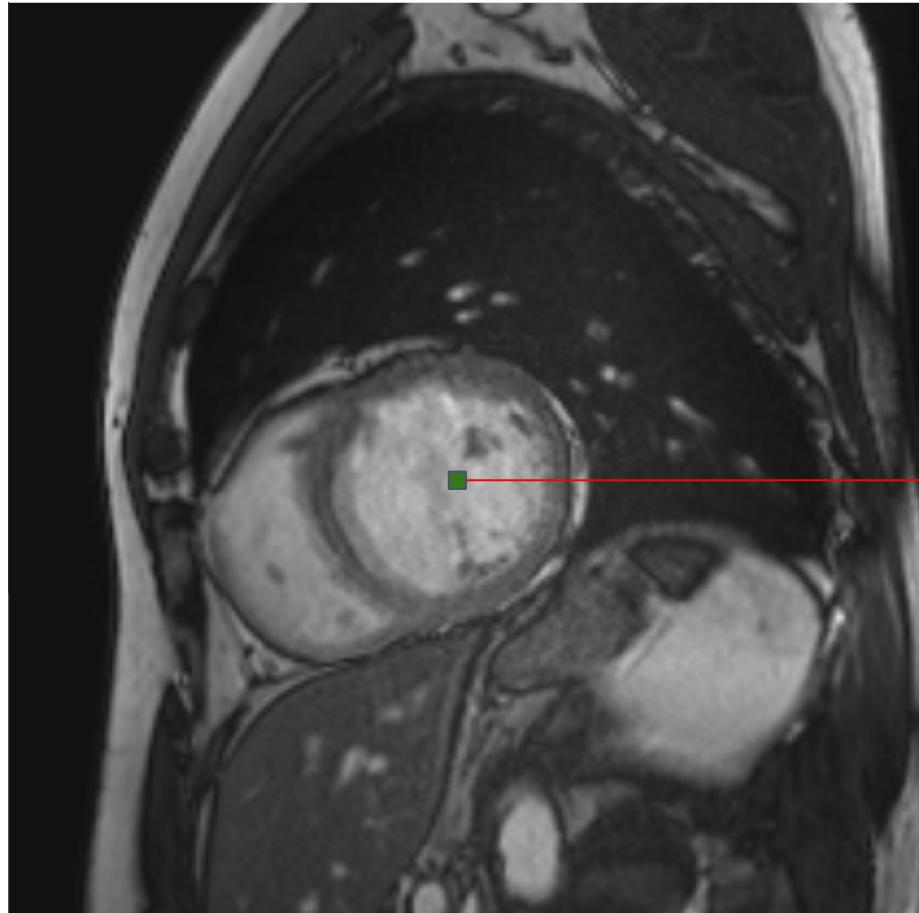
LeNet, AlexNet, VGG-Net, GoogLeNet all predict one value per **image**



Often, we want to predict one value per **pixel/voxel**



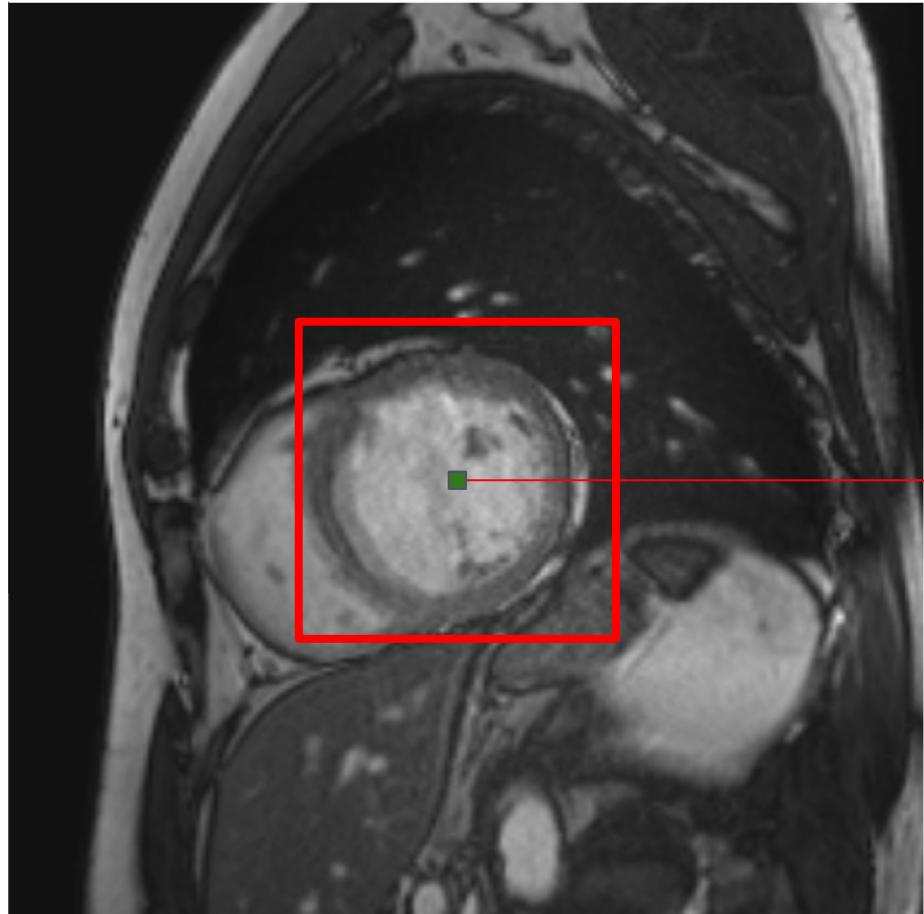
# Sliding window



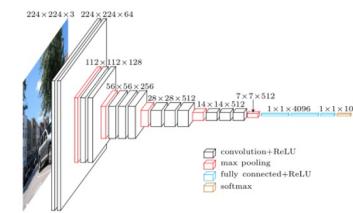
- **Image** = patch centered at voxel
- **Label** = class of center voxel

- Background
- Left ventricle
- Myocardium
- Right ventricle

# Sliding window



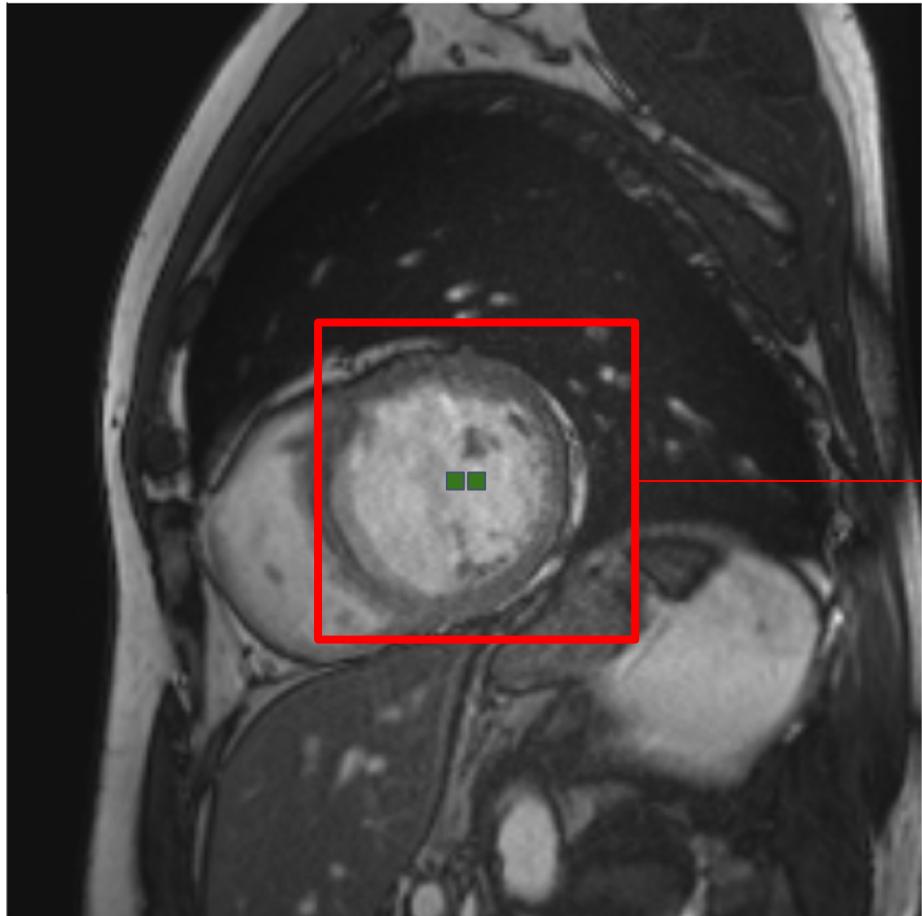
- **Image** = patch centered at voxel
- **Label** = class of center voxel



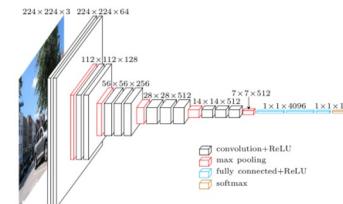
- Background
- Left ventricle
- Myocardium
- Right ventricle

# Sliding window

Combination of thousands of image classification tasks



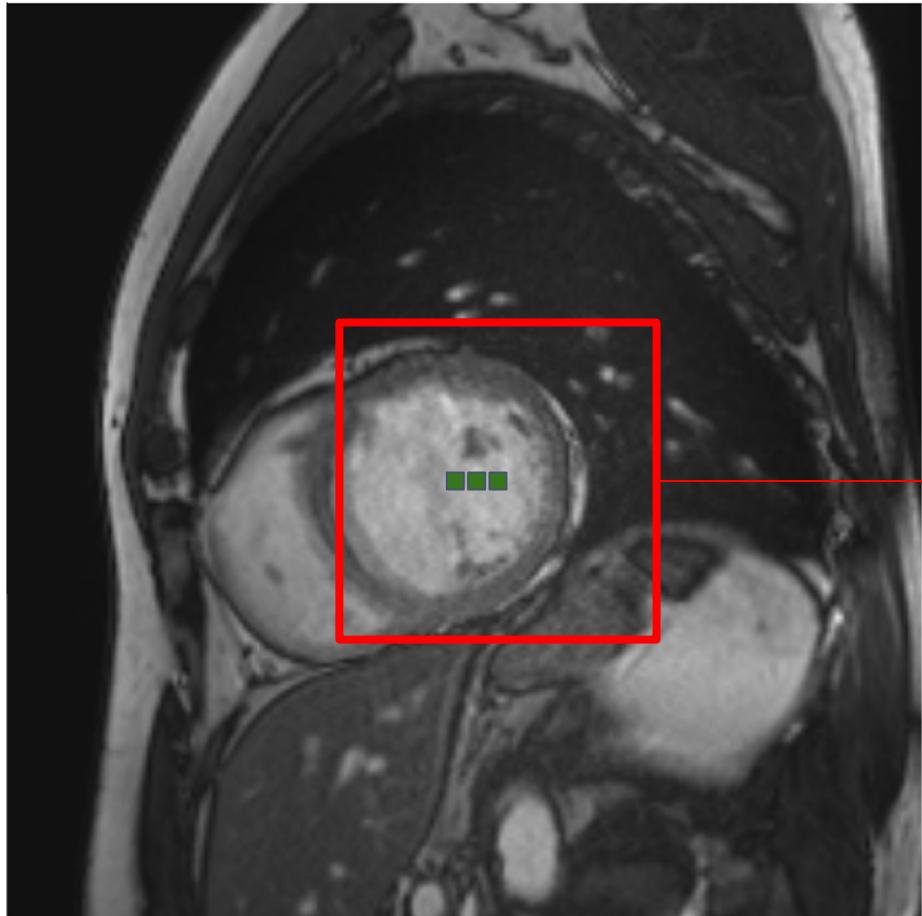
- **Image** = patch centered at voxel
- **Label** = class of center voxel



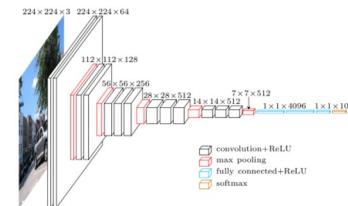
- Background
- Left ventricle
- Myocardium
- Right ventricle

# Sliding window

Combination of thousands of image classification tasks

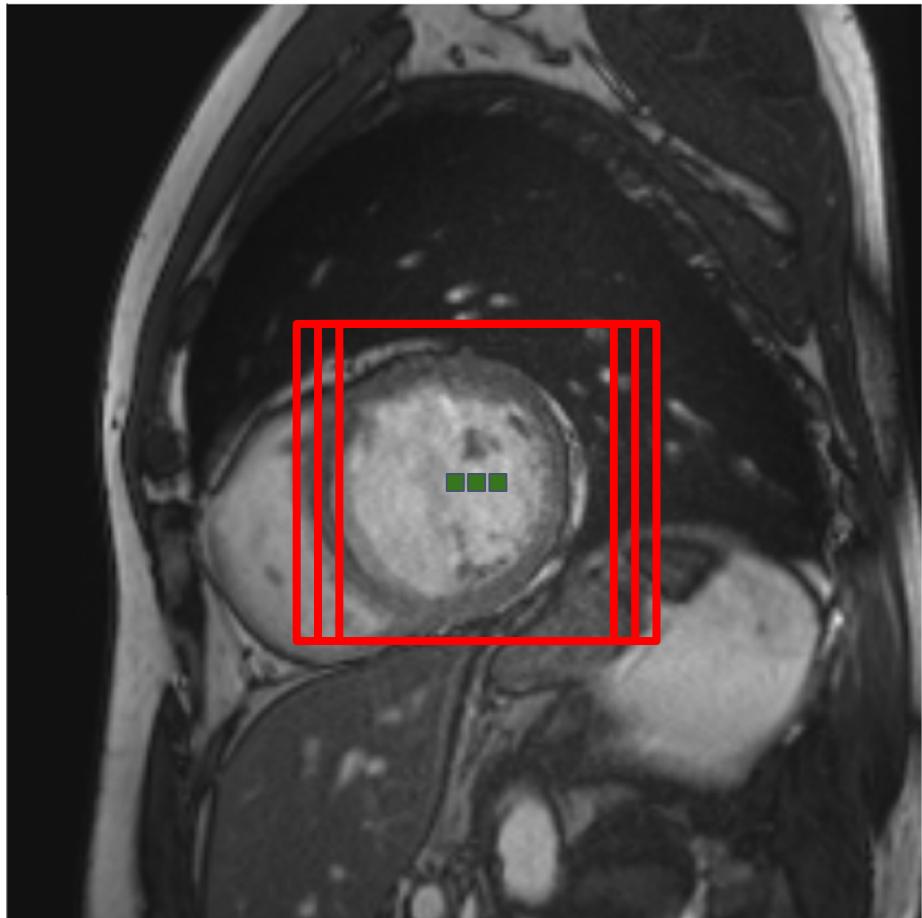


- **Image** = patch centered at voxel
- **Label** = class of center voxel



- Background
- Left ventricle
- Myocardium
- Right ventricle

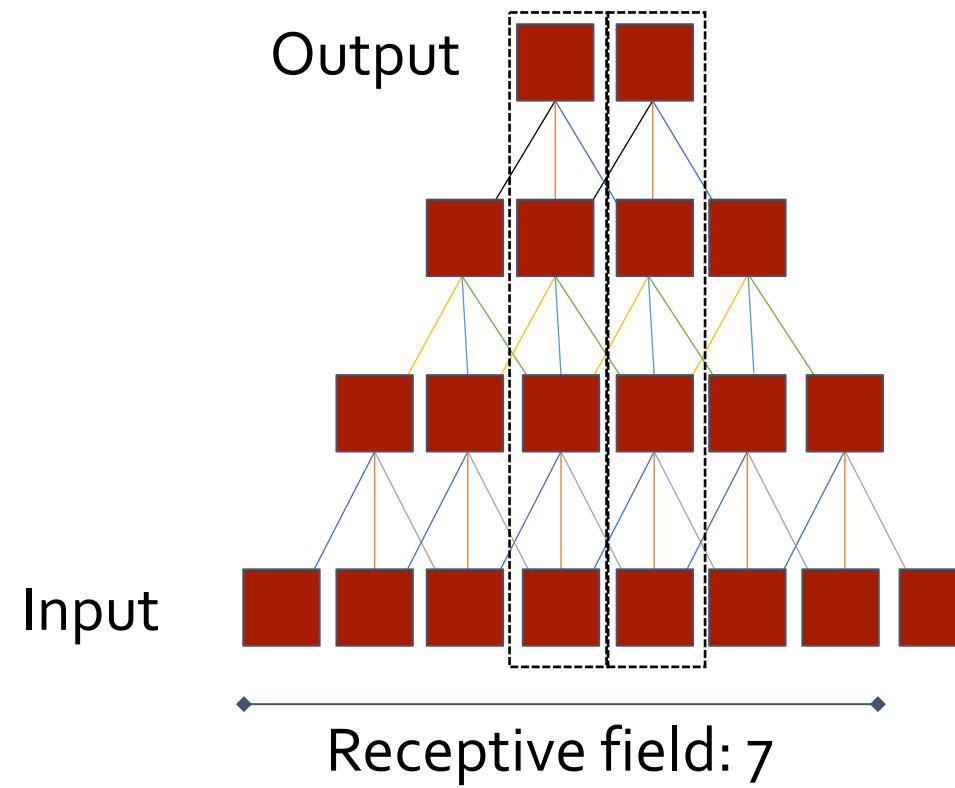
# Sliding window



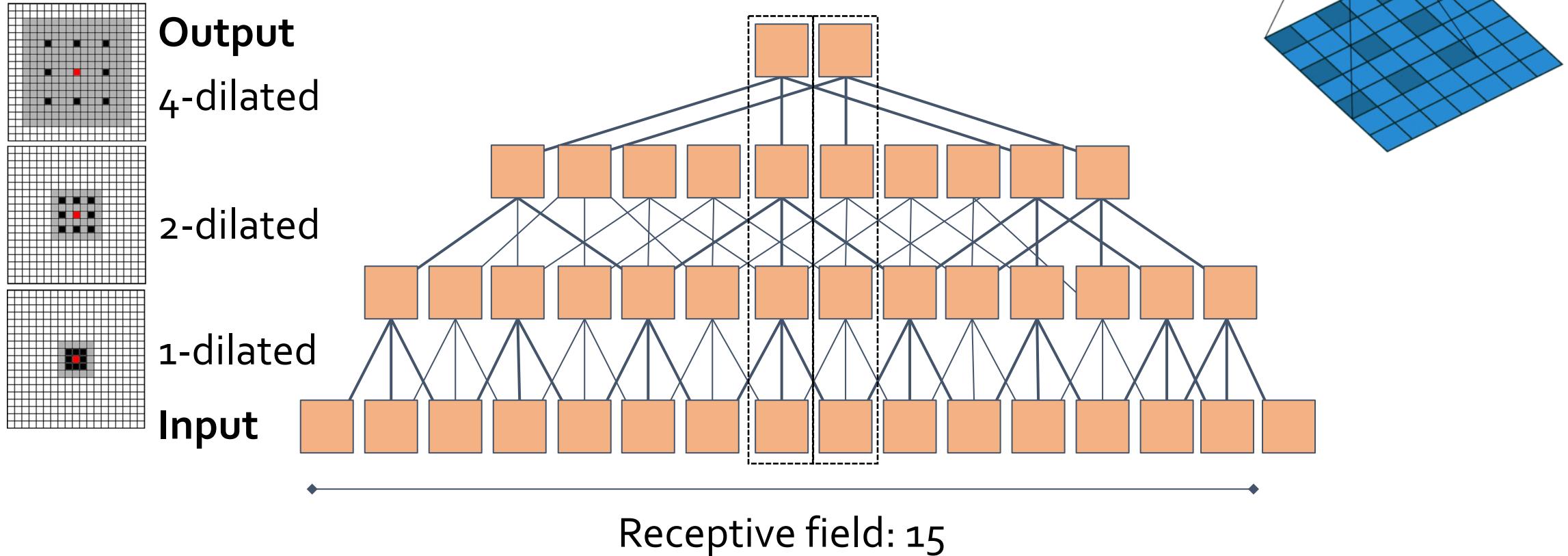
**Sliding window approaches are inefficient**

- Each patch is processed separately
- Lots of redundant operations
- We would like to re-use/share operations

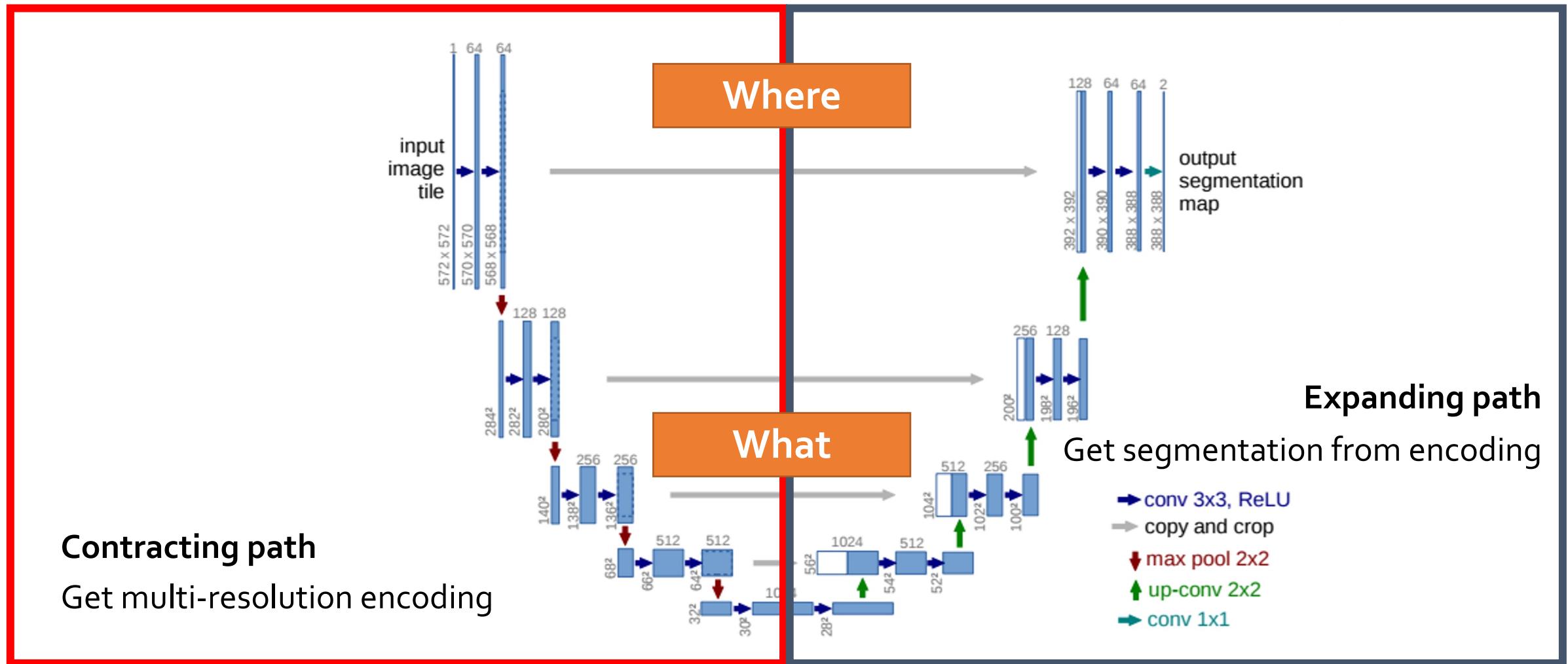
# All-convolutional network



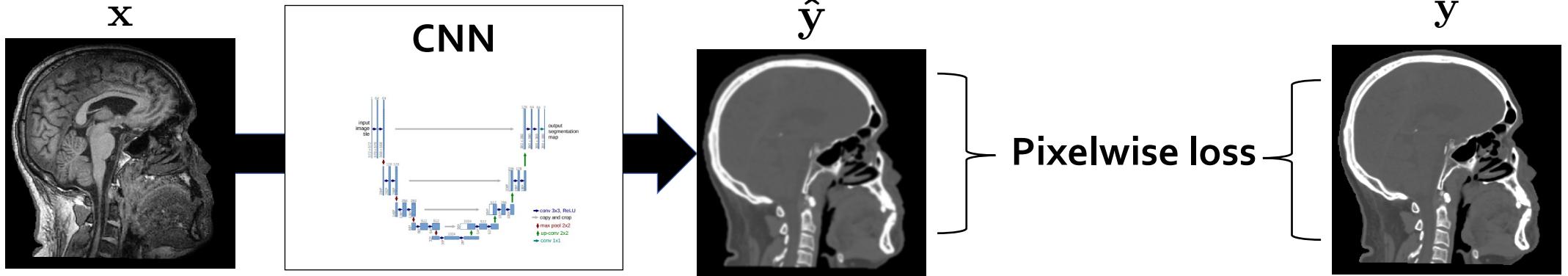
# Dilated convolutions



# Encoder-decoder architecture: U-Net

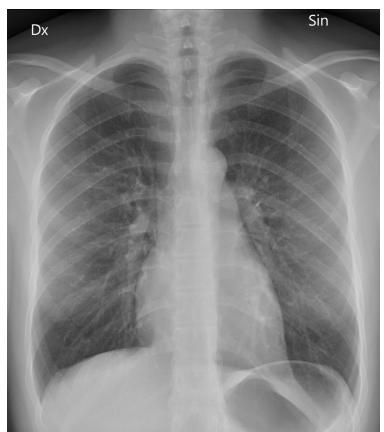
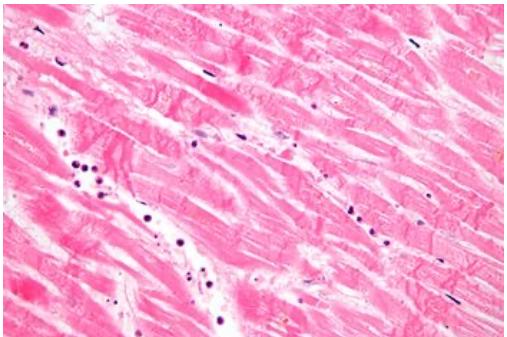


# Training

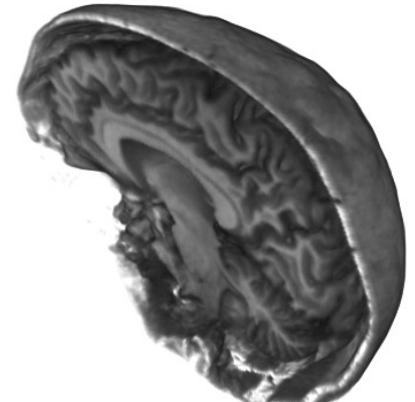
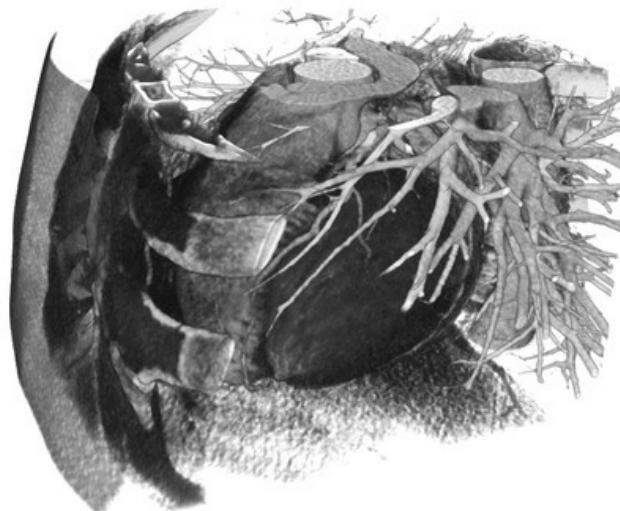


# 2D or 3D images

2D data



3D data



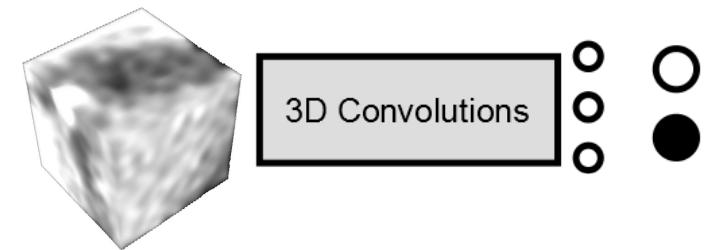
# 3D networks

Many medical images are 3D instead of 2D

- MR images
- CT images

Can we just use 3D layers instead of 2D layers? Sure!

- 3D convolution layers in Keras, TensorFlow, PyTorch, etc.
- 3D network architectures (e.g. U-Net, V-Net)



But

- Is your data really 3D (think about acquisition)? Isotropy?
- Increase in memory consumption + operations + parameters

# Summary

Advanced architectures

- AlexNet, GoogleNet, VGG-Net, ResNet
- Deeper, larger, better + some tricks

Recurrent neural networks

- RNNs + LSTMs

Per image prediction != per voxel prediction

- All-convolutional networks
- Encoder-decoder architectures

2D/3D data

- Most 2D neural networks extend to 3D

# **Learning goals for today**

Have a good overview of neural networks architectures for (medical) image analysis

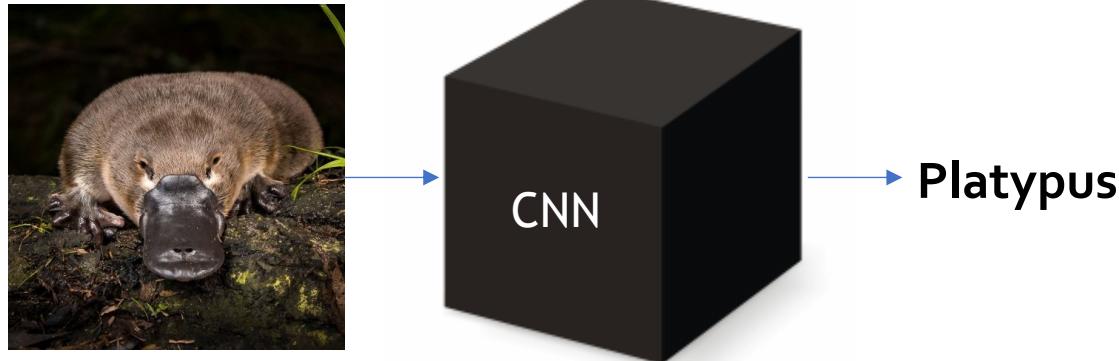
- Including architectures for sequential data

**Obtain a general understanding of interpretability of deep learning models**

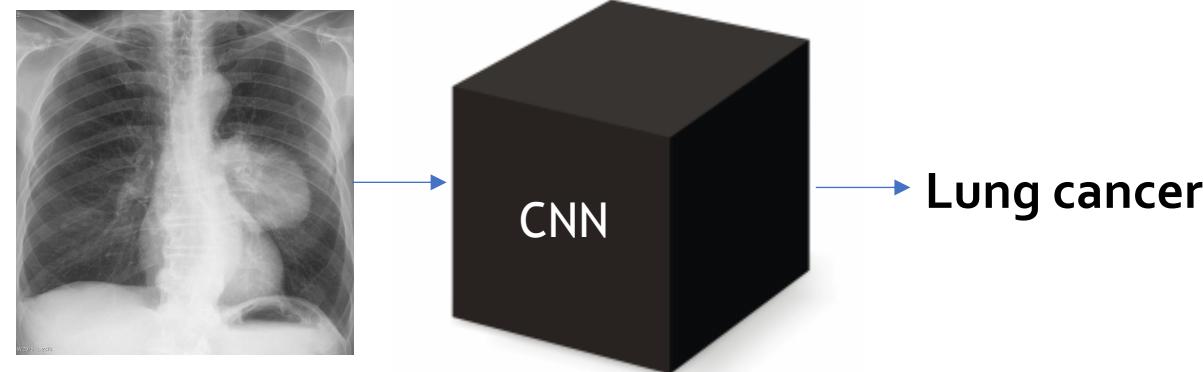
Obtain a general understanding of generative adversarial networks

- Only one example of generative models, but arguably the most successful one

# Machine learning as a black box



# Machine learning as a black box



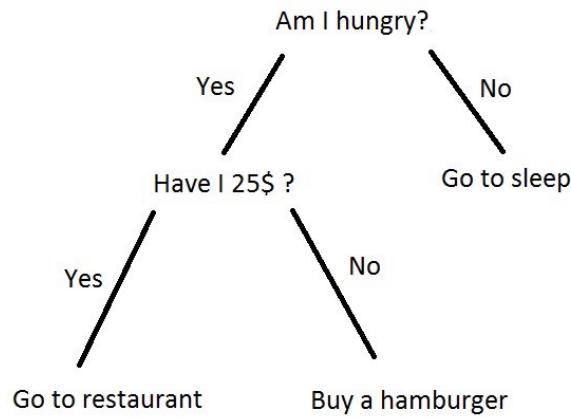
**Interpretability:** Can we predict what the model will do if the input changes?

**Explainability:** Do we know exactly what the model is doing?

**Uncertainty:** How sure is the network of its prediction?

Important questions for ethical use and liability

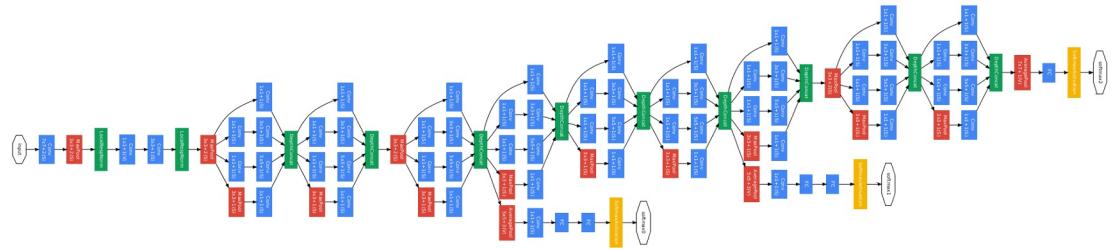
# Complexity/performance trade-off



## Decision tree

Explainable/interpretable

Performance



## CNN

Explainable/interpretable

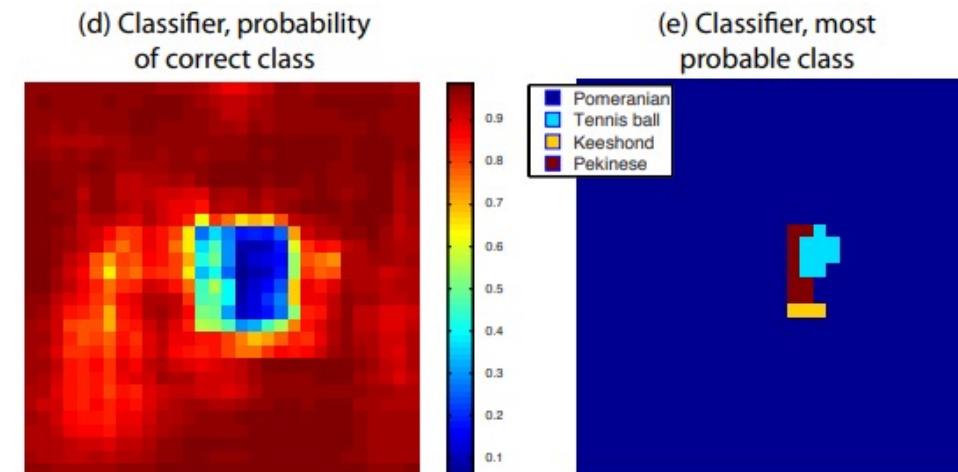
Performance

# Interpreting CNNs: saliency maps

- Millions of parameters, impossible to know in detail what each parameter does
- Substitute: what part of the input is associated with a CNN feature or output?
- Popular approaches
  - Occlusion
  - Deconvolution
  - Class activation mapping

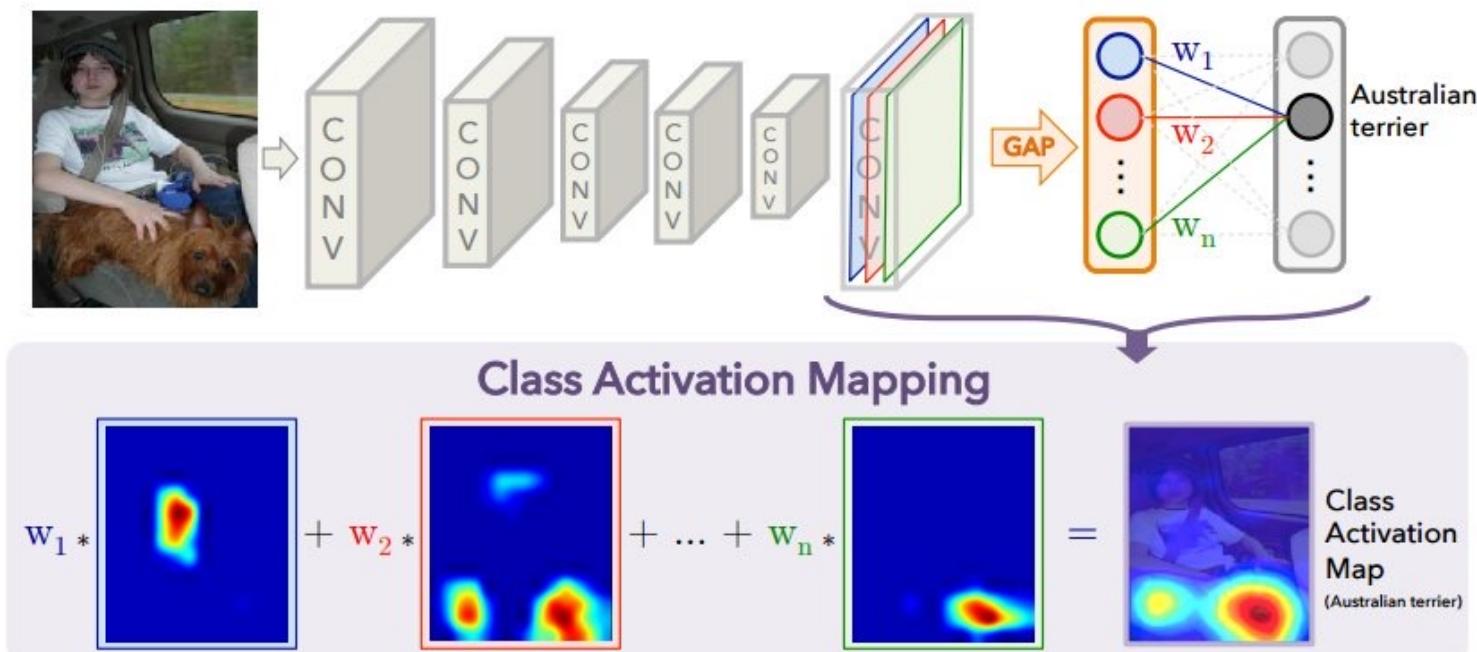
# Occlusion

1. Move grey square over image (a)
2. See how classifier output for correct class changes (d)



# Class activation maps (CAM)

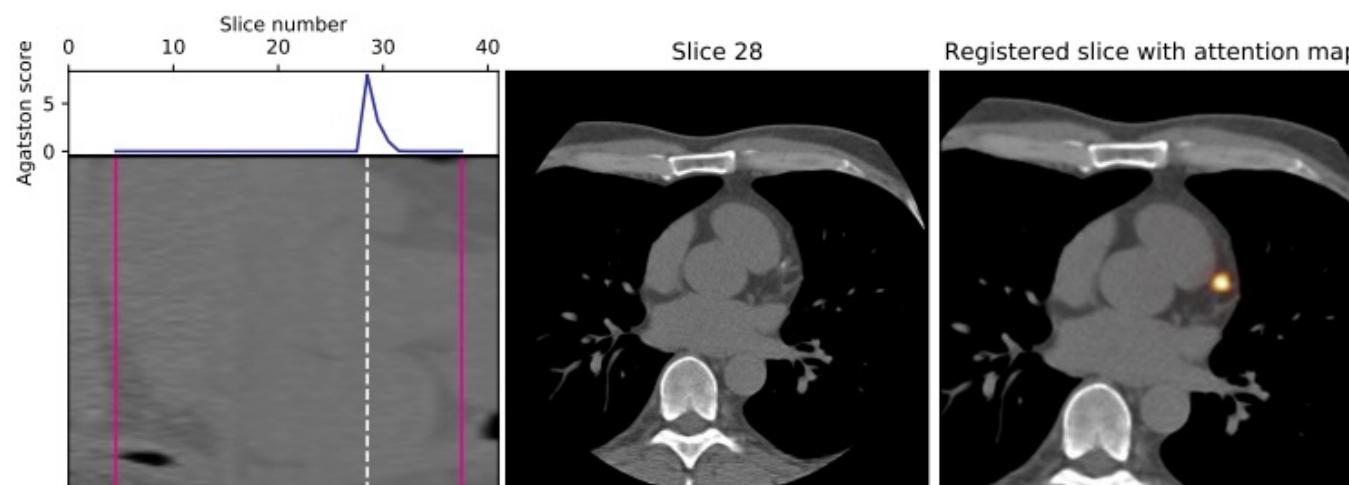
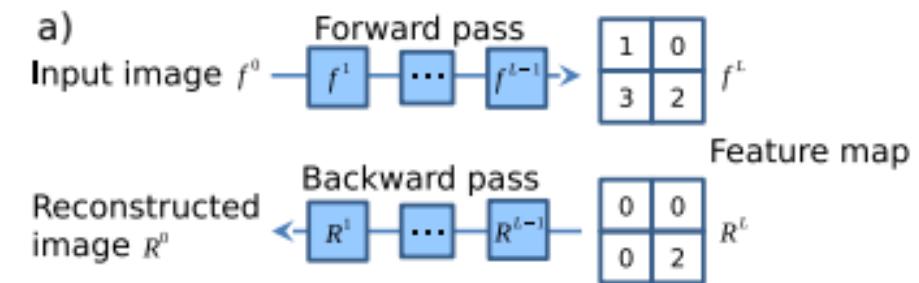
- Global average pooling: simply compute average of whole feature map
- Feature maps represent heat maps, weighting gives evidence



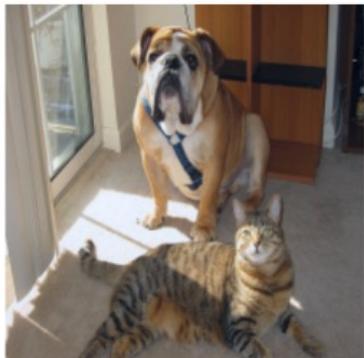
# Deconvolution

Inversion: find input corresponding to a particular feature

1. Push input through CNN
2. Fix value for one ‘neuron’, set all others to zero
3. Perform inverse of convolution, pooling, activation functions



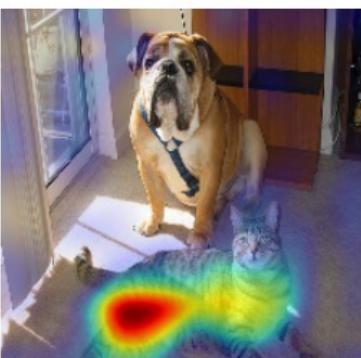
# Comparison



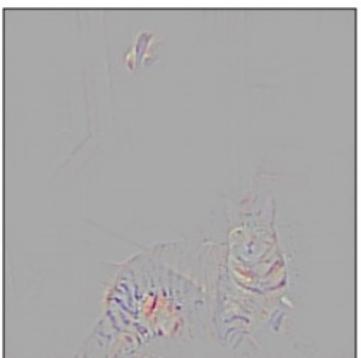
(a) Original Image



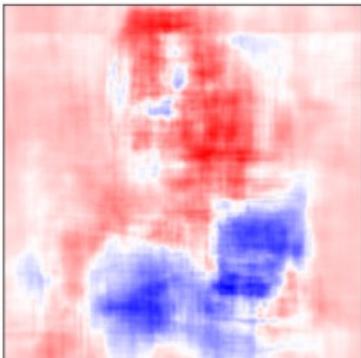
(b) Guided Backprop ‘Cat’



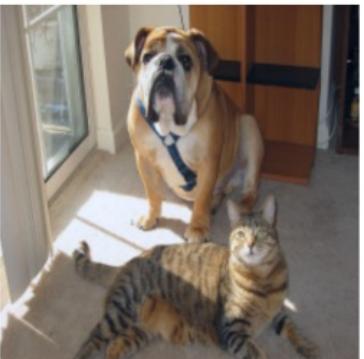
(c) Grad-CAM ‘Cat’



(d) Guided Grad-CAM ‘Cat’



(e) Occlusion map for ‘Cat’



(g) Original Image



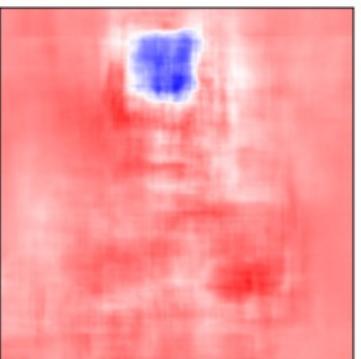
(h) Guided Backprop ‘Dog’



(i) Grad-CAM ‘Dog’



(j) Guided Grad-CAM ‘Dog’



(k) Occlusion map for ‘Dog’ (

# **Learning goals for today**

Have a good overview of neural networks architectures for (medical) image analysis

- Including architectures for sequential data

Obtain a general understanding of interpretability of deep learning models

**Obtain a general understanding of generative adversarial networks**

- Only one example of generative models, but arguably the most successful one

# Discriminative vs. generative models

## Discriminative models

- Go from high-dimensional sample to low-dimensional prediction
- E.g. from an image to the label of that image



Container ship

## Generative models

- Go from low-dimensional input to high-dimensional sample
- E.g. from an image label to an image sample

Container ship



# **But why would we do this?**

- **Unsupervised** learning: we don't need labels for our training data
- Learning more about your data set
- Synthesizing new data for discriminative models
- Advanced image manipulation

# Generative adversarial networks (GANs)

Instead of **one** neural network, we train **two** neural networks

1. A **generator** network generates samples of images
2. A **discriminator** network distinguishes generated samples from real samples

The **generator** network tries to fool the discriminator



# Generative adversarial networks (GANs)

1. A **generator** network generates samples of images
2. A **discriminator** network distinguishes generated samples from real samples



**Generator**  
Counterfeiter



**Discriminator**  
Detective

# Generative adversarial networks (GANs)

1. A **generator** network generates samples of images
2. A **discriminator** network distinguishes generated samples from real samples



**Generator**  
Counterfeiter



**Discriminator**  
Detective



# Generative adversarial networks (GANs)

1. A **generator** network generates samples of images
2. A **discriminator** network distinguishes generated samples from real samples



**Generator**  
Counterfeiter



**Discriminator**  
Detective



# Generative adversarial networks (GANs)

1. A **generator** network generates samples of images
2. A **discriminator** network distinguishes generated samples from real samples



**Generator**  
Counterfeiter



**Discriminator**  
Detective



# Generative adversarial networks (GANs)

A bit more formally...

We are given samples  $x_1, \dots, x_n$  from a data distribution  $p_{data}$

We want to generate new samples similar to  $p_{data}$

We can sample from  $p_{data}$  efficiently – examples are available



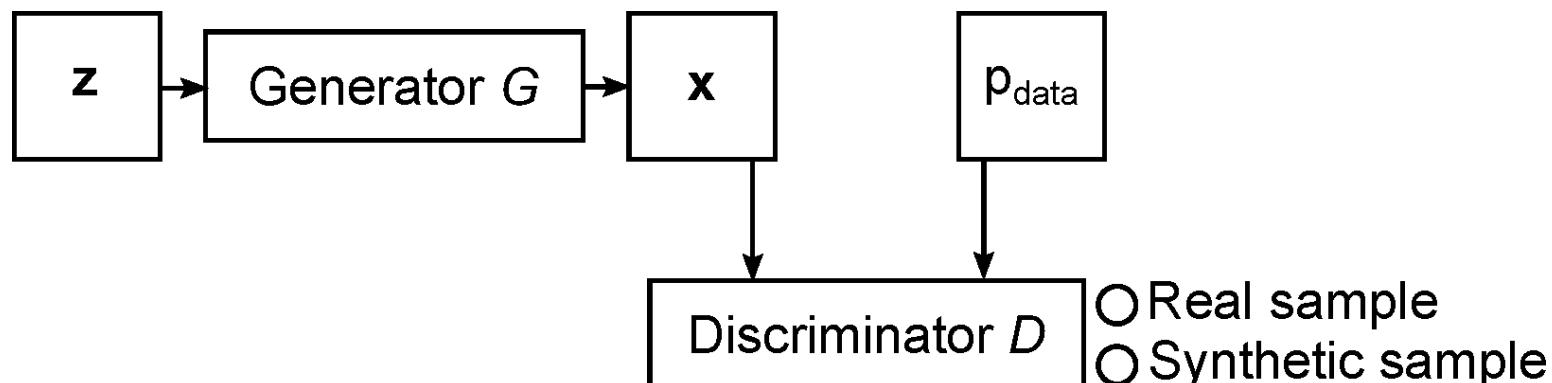
# Generative adversarial networks (GANs)

**Aim:** Find a sample distribution  $p_{gen}$  that closely resembles  $p_{data}$

Generator  $G$  maps points from a normal/uniform distribution  $z$  to samples  $x$  in  $p_{gen}$

Discriminator  $D$  tells samples from  $p_{data}$  and  $p_{gen}$  apart

Generators  $G$  and  $D$  are networks with trainable parameters  $\vartheta_G$  and  $\vartheta_D$



# Generative adversarial networks (GANs)

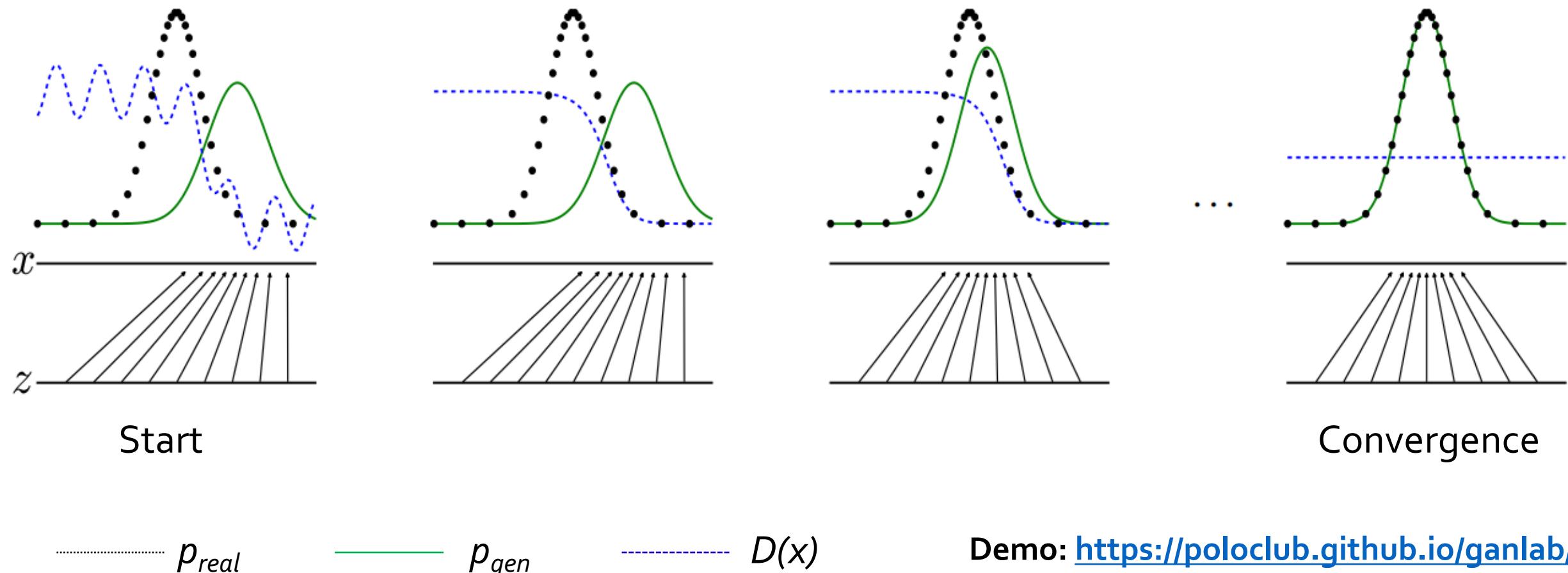
Discriminator  $D$  maximizes an **objective function** by assigning a high **probability to real samples** and a low **probability to synthetic samples**

$$V^{(D)}(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log (1 - D(G(z)))]$$

At the same time,  $G$  tries to minimize this **objective function** by generating samples that have a high **probability of being real**

$$V^{(G)}(D, G) = \mathbb{E}_{z \sim p_z} [\log (1 - D(G(z)))]$$

# GANs in theory



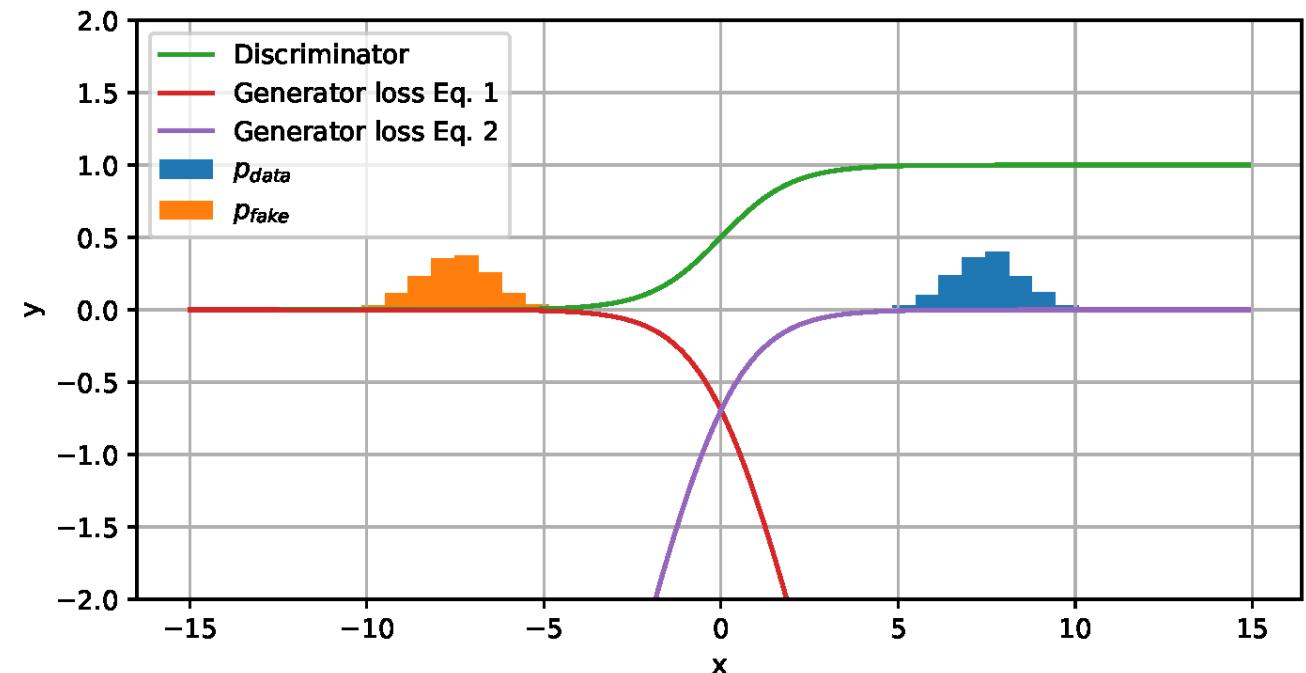
# Challenges when training GANs

Poor gradients from discriminator  $D$  to generator  $G$

Especially when the discriminator  $D$  can easily distinguish fake and real samples

$$1) \quad V^{(G)}(D, G) = \mathbb{E}_{z \sim p_z} [\log (1 - D(G(z)))]$$

$$2) \quad V^{(G)}(D, G) = - \mathbb{E}_{z \sim p_z} [\log (D(G(z)))]$$



# Challenges when training GANs

1. Poor gradients from discriminator  $D$  to generator  $G$

- Especially when the discriminator  $D$  can easily distinguish fake and real samples

2. Mode collapse

- Generator  $G$  maps all noise vectors to similar samples  $G(z)$
- E.g. generator only synthesizes 1 in MNIST
- Generator and discriminator chase each other from mode to mode

3. Unclear when training is finished

# Synthesizing MNIST

Generator

$z$  10

256

512

1024

Discriminator

$x$   $28 \times 28 = 784$

1024

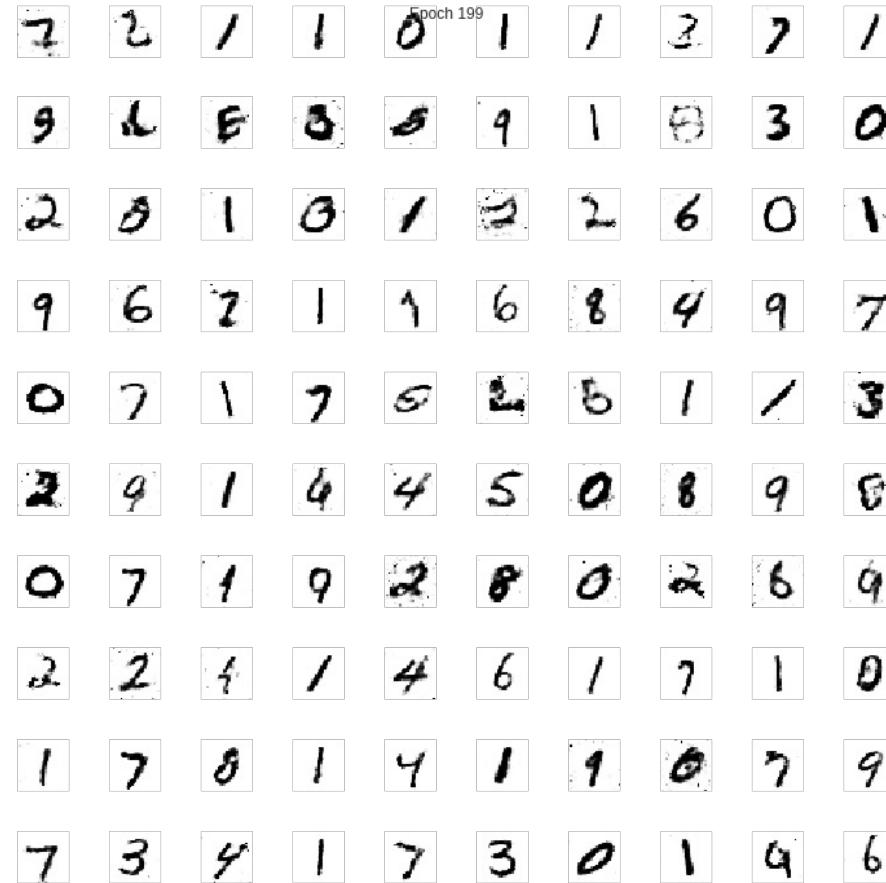
512

256

$D(x)$

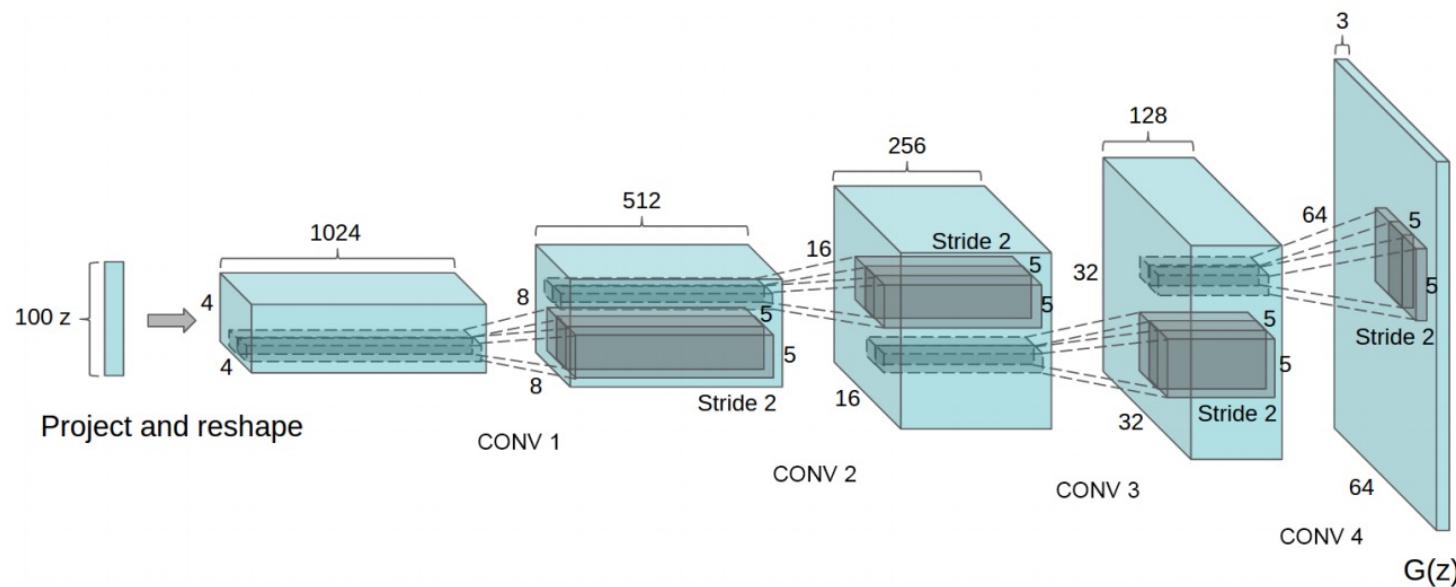
1

# Synthesizing MNIST

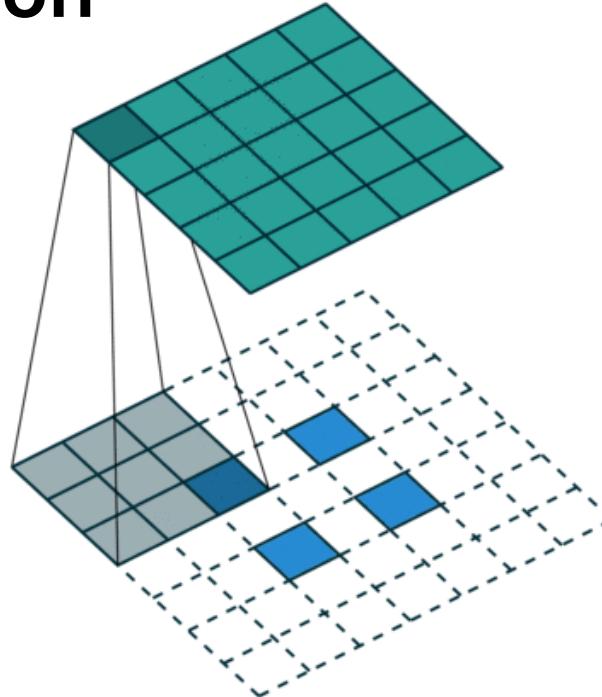


# Synthesizing images

- Image synthesis requires convolutional generators and discriminators
- We know how to reduce the image size
- Now we need some way to increase the image size



# Transposed convolution



- Deconvolution
- Fractionally-strided convolution
- Transposed convolution



2014



2015



2016



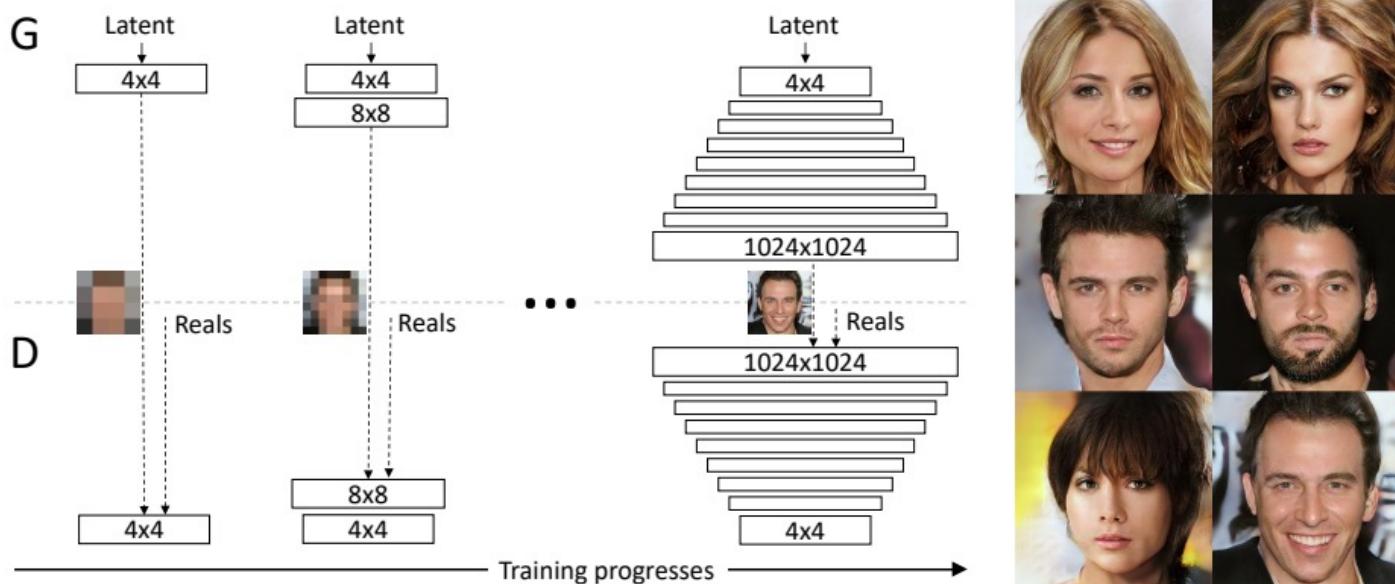
2017



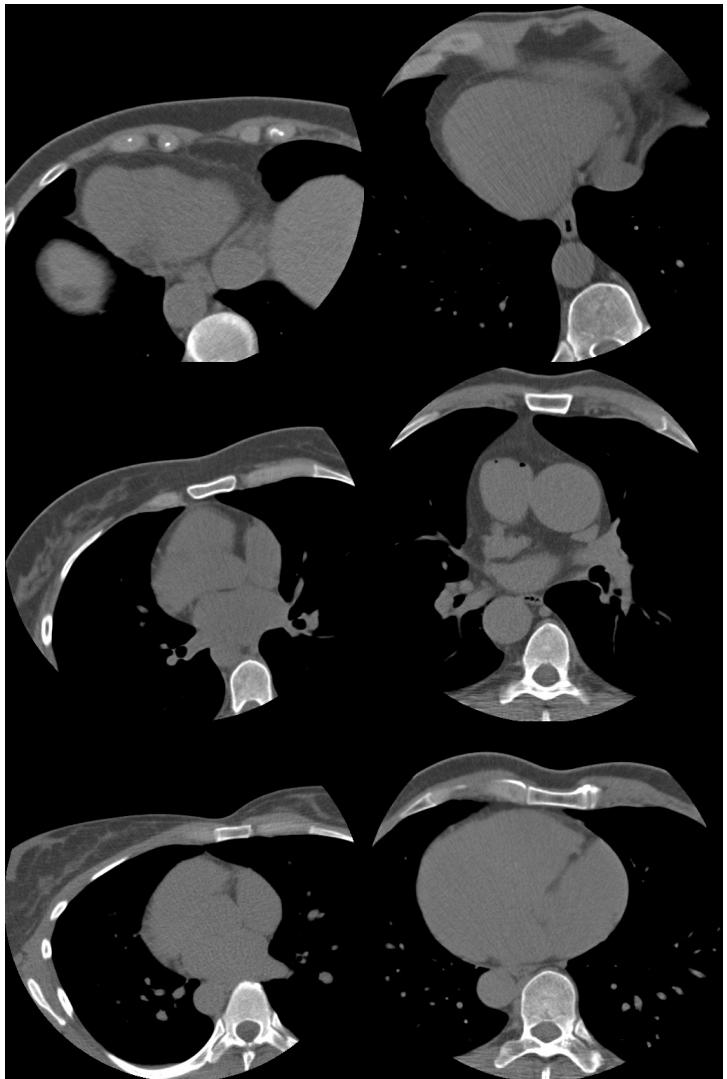
2018

# Progressive GAN

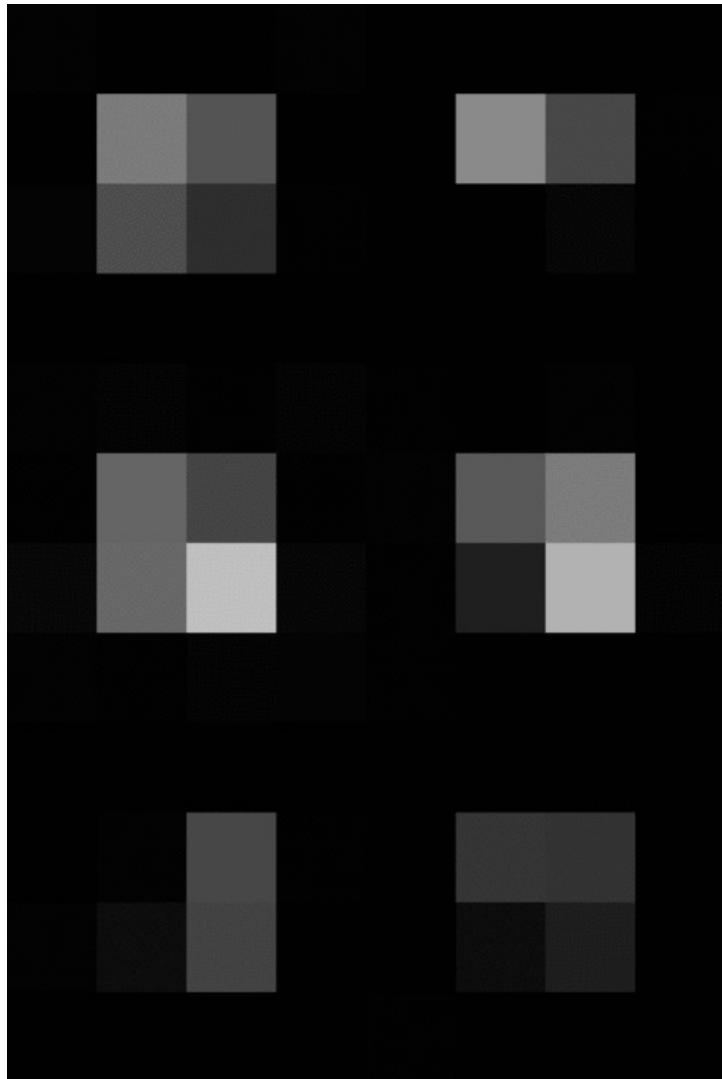
- Goal: high-resolution image synthesis ( $1024 \times 1024$  pixels)
- Problem: optimizing large and deep networks can lead to stability issues
- Trick: periodically increase resolution in  $G$  and  $D$



**Real**



**Fake**



# StyleGAN

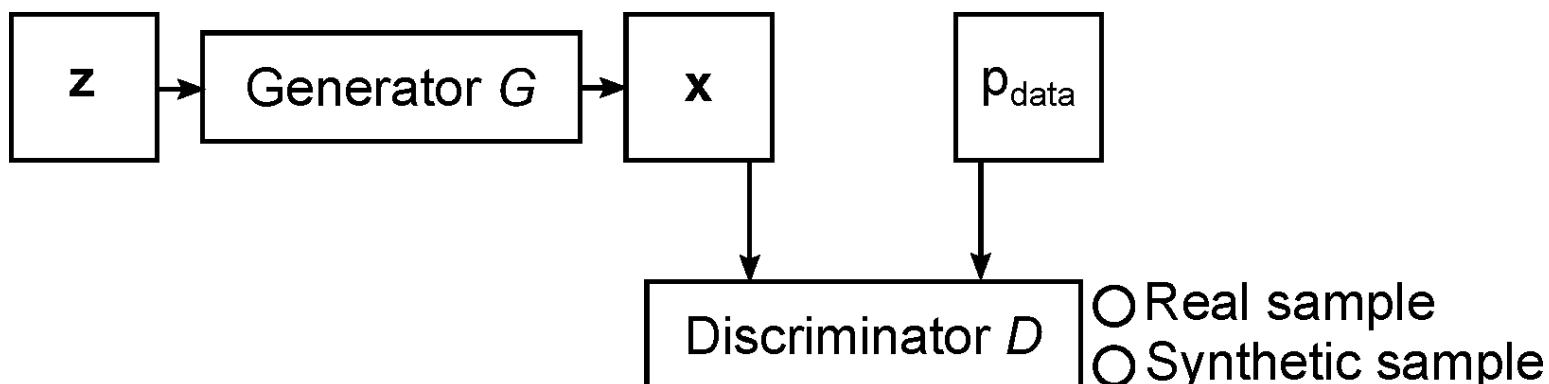


# Spaces

- GANs can help us learn more about our data
- Different latent space points correspond to different sample space points
- The latent space is structured
- Interpolation in the latent space leads to smooth transitions

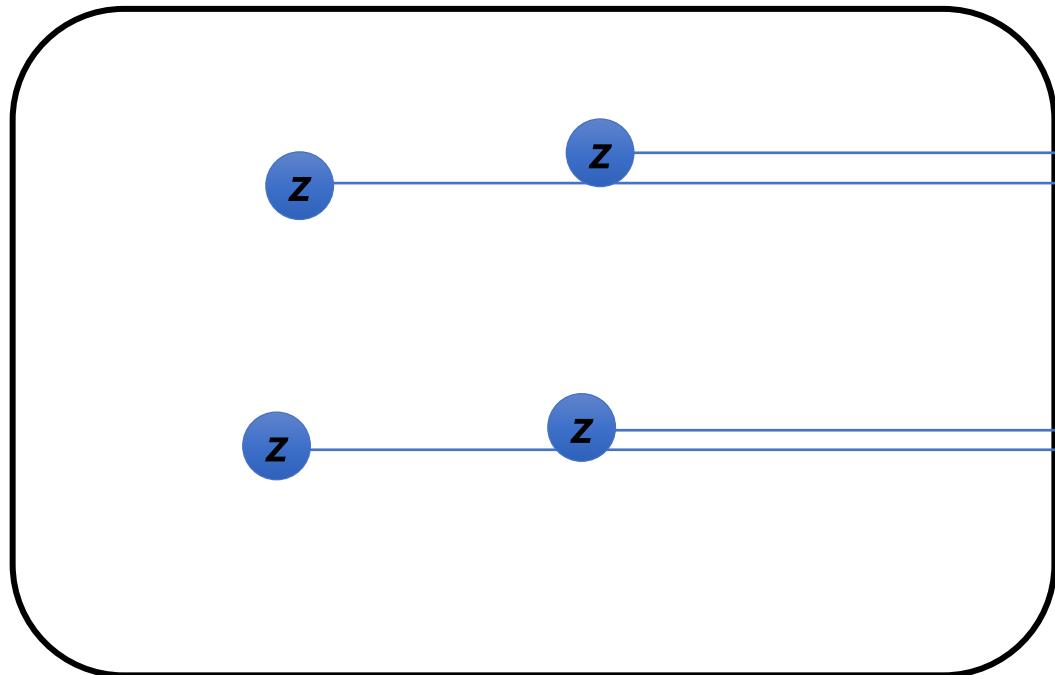
The latent space

The sample space

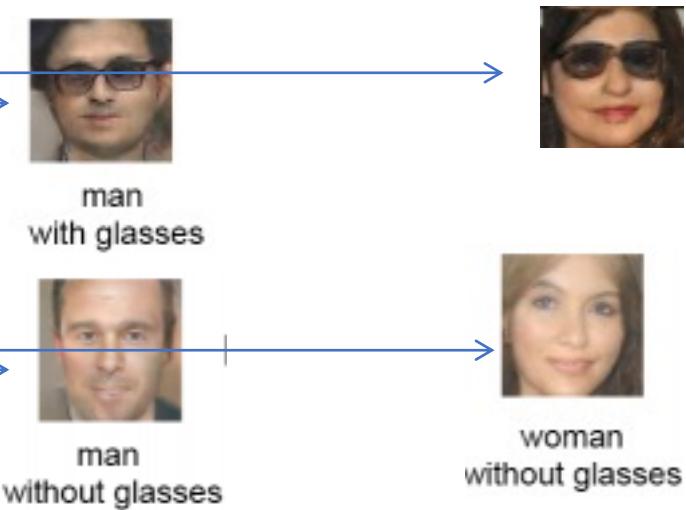


# Spaces

## The latent space

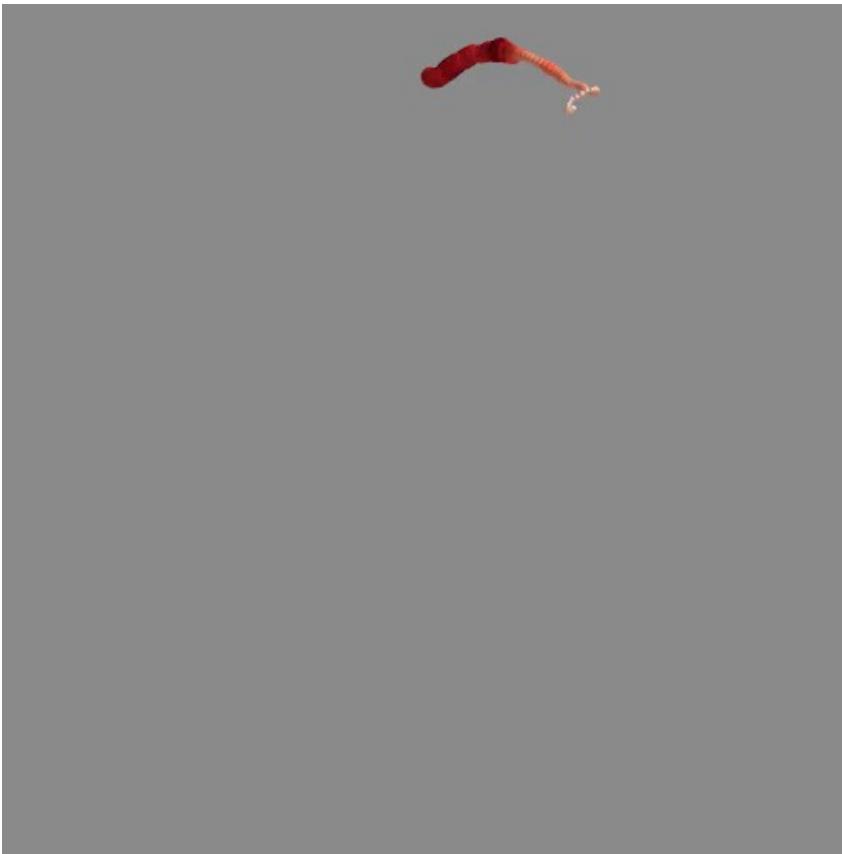


## The sample space

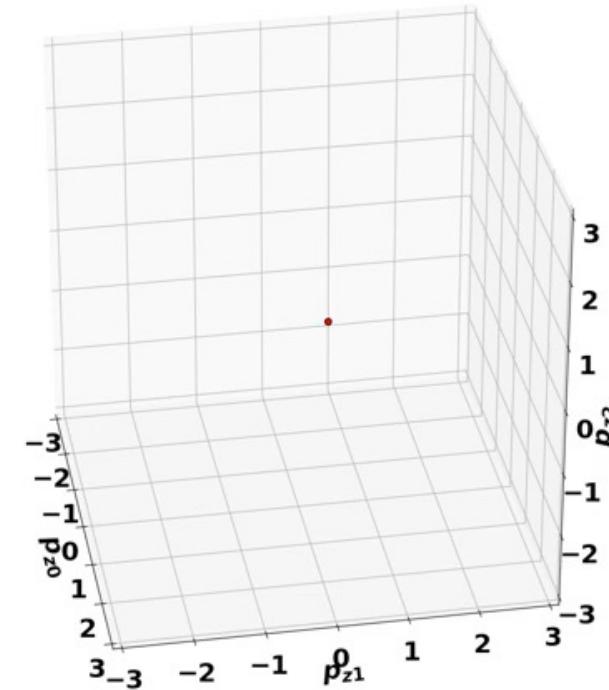


# Example: coronary artery synthesis

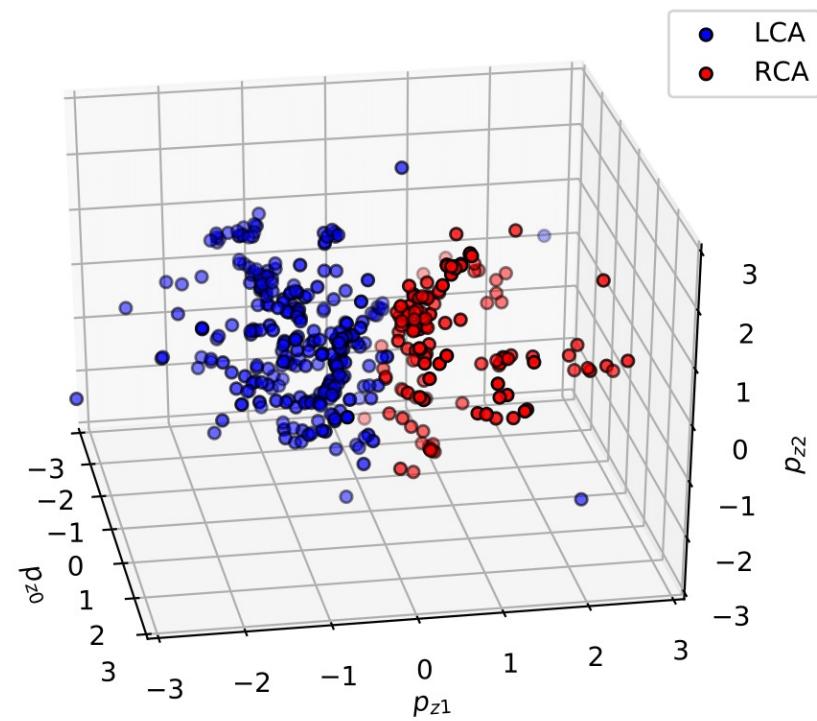
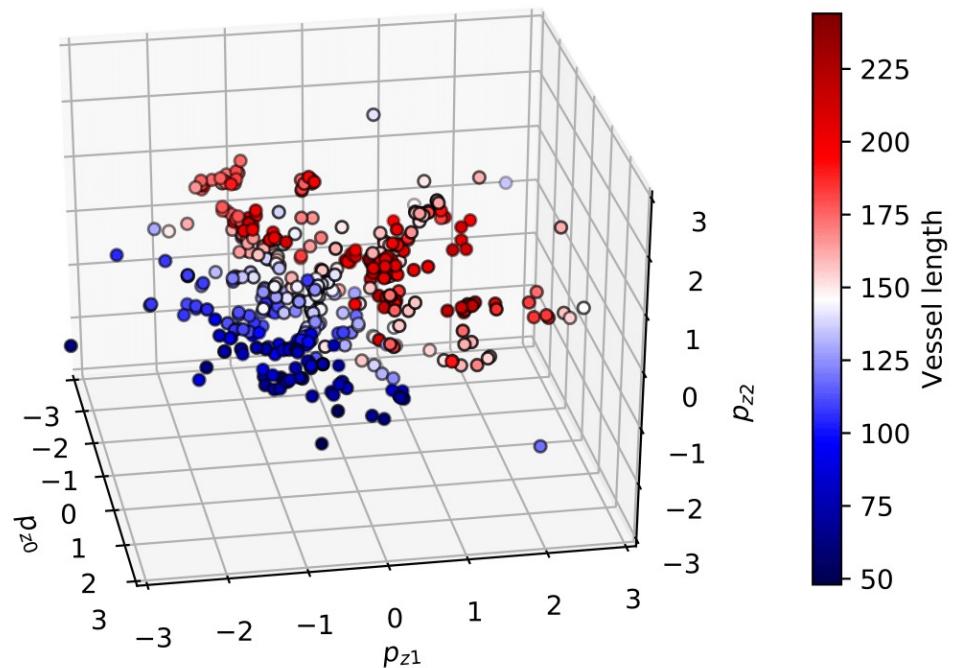
The sample space



The latent space

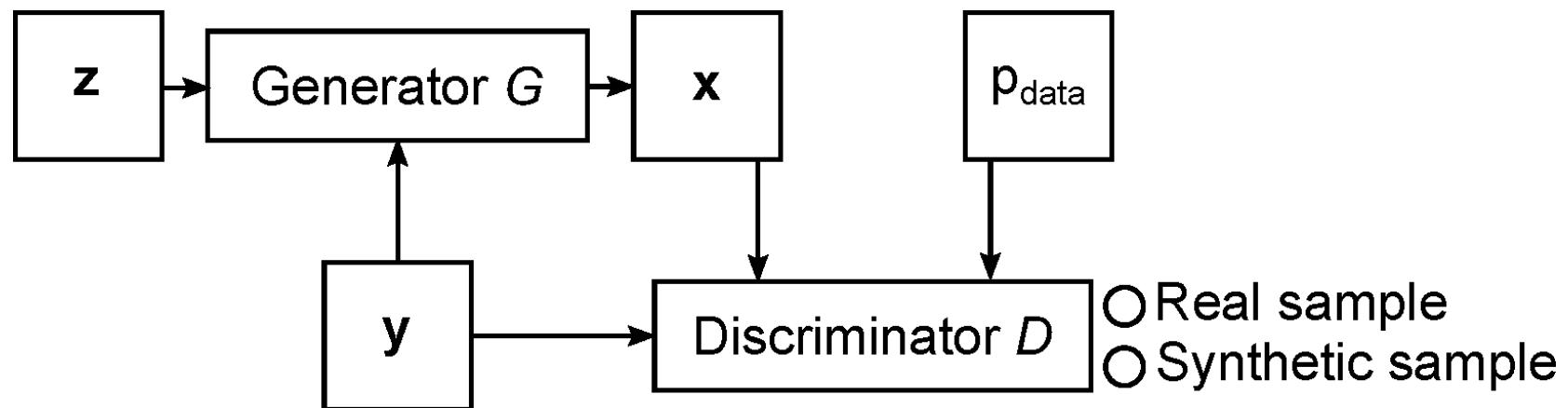


# Example: coronary artery synthesis



# Conditional GAN (cGAN)

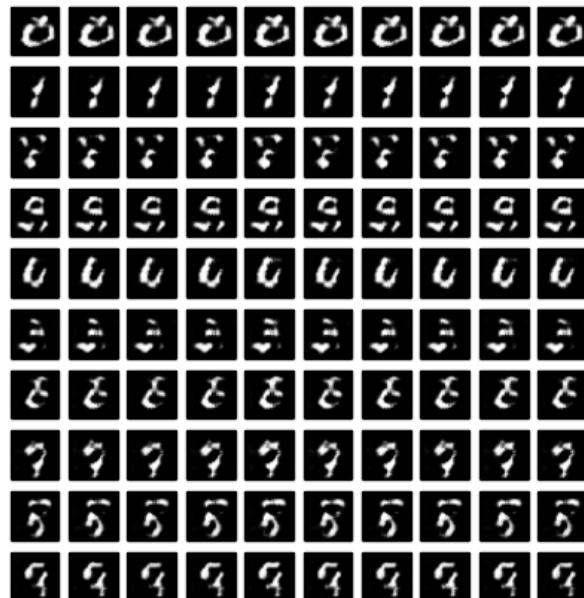
- GAN: difficult to control what is generated
- Condition GAN on extra input  $y$



# Conditional GAN (cGAN)

The generator and discriminator optimize an **objective function** based on the predictions for **real samples** and **synthetic samples**, given **extra information**

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x|y)] + \mathbb{E}_{z \sim p_z} [\log (1 - D(G(z|y)|y))]$$



# **Remaining slides: self study**

BigGANs

CycleGANs

GANs for medical image analysis

+ watch the video lecture from last year given by Jelmer Wolterink

# BigGAN

- State of the art in conditional GAN training
- Trained on ImageNet (~million images)
- Get any class you want



# In medical imaging

We typically have some image to condition on, ‘image-to-image’ applications

- Segmentation
- Synthesis
- Image quality enhancement

It's not always easy to determine a loss function that captures what we want

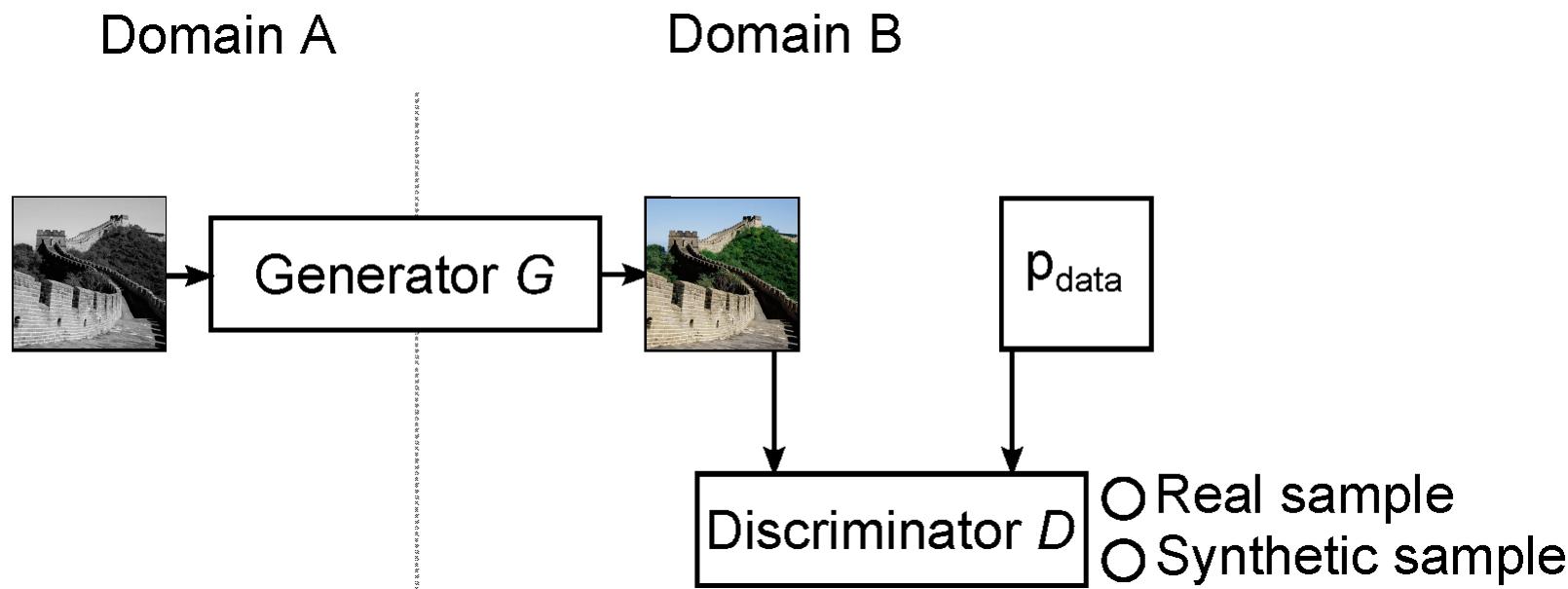
- Noise-free segmentations?
- Sharp images?

Use a conditional GAN where  $y$  is the input image

# Image-to-image translation

Domain A: input image  $y$

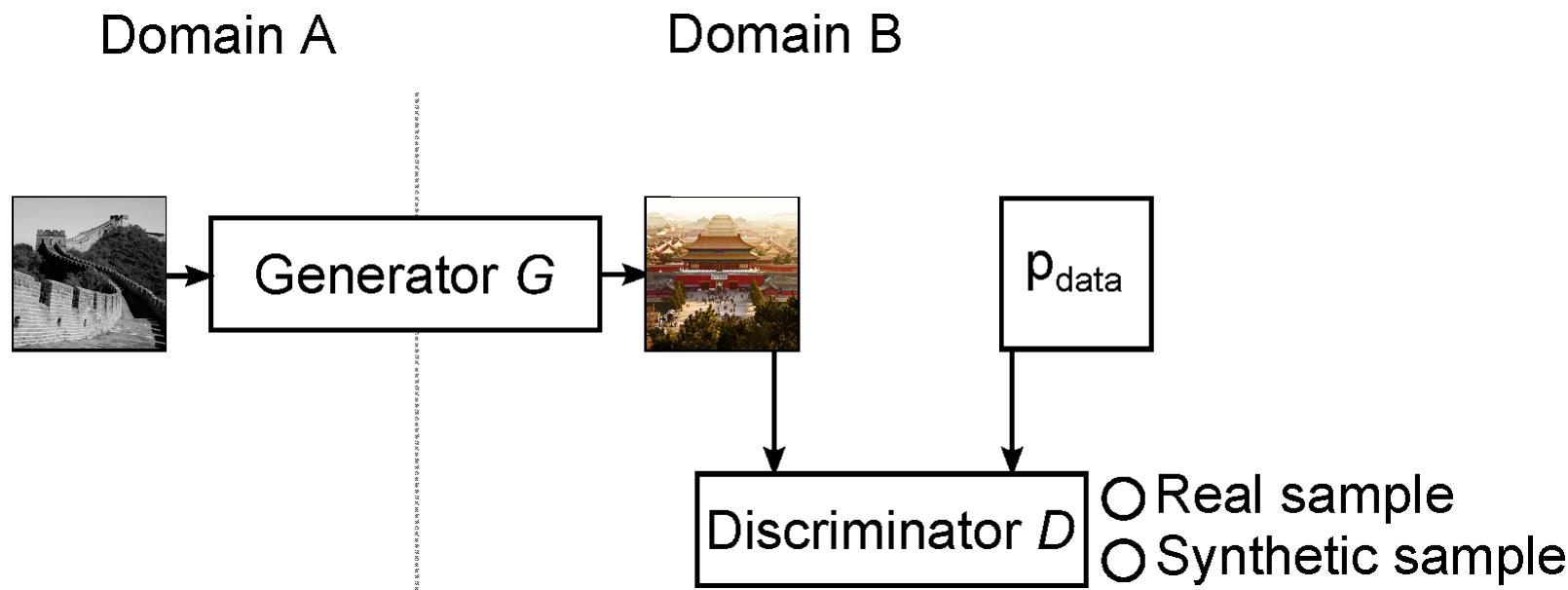
Domain B: output image  $x$



# Image-to-image translation

The output image  $x$  could be very realistic but 'wrong'

Regularize to make sure that it corresponds to the input image  $y$



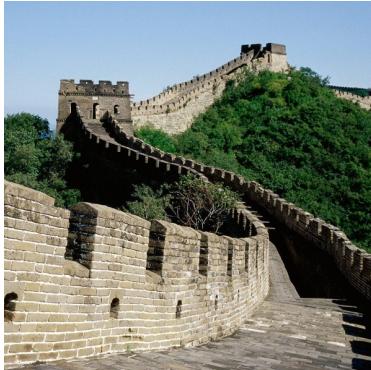
# Paired vs. unpaired training data

Paired

Domain A



Domain B



Unpaired

Domain A



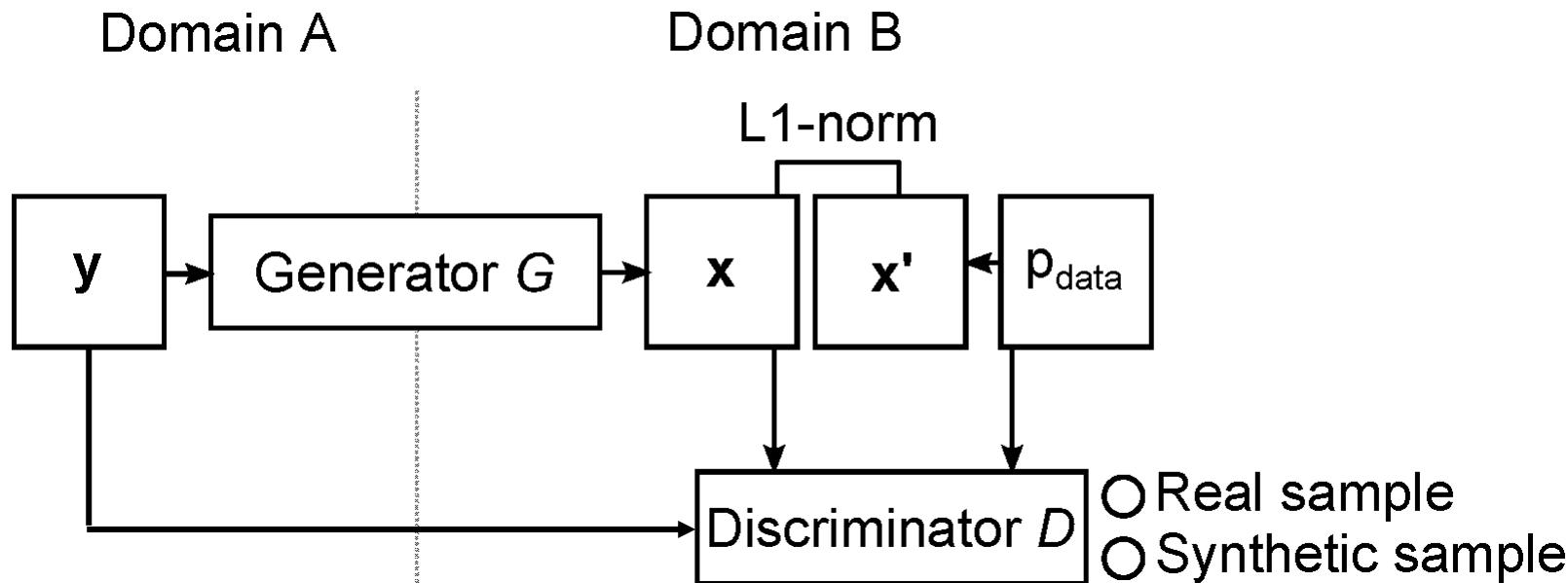
Domain B



# Paired image-to-image translation

The discriminator and generator optimize an **objective function** based on predictions for **real** and **synthetic** images in domain B, given input images in domain A while minimizing the **difference between real and synthetic images in domain B**

$$\min_G \max_D V(D, G) = \mathbb{E}_{x,y \sim p_{data_{B,A}}} [\log D(x, y)] + \mathbb{E}_{y \sim p_{data_A}} [\log (1 - D(G(y), y))] + \lambda \mathbb{E}_{x,y \sim p_{data_{B,A}}} [|x - G(y)|]_1$$



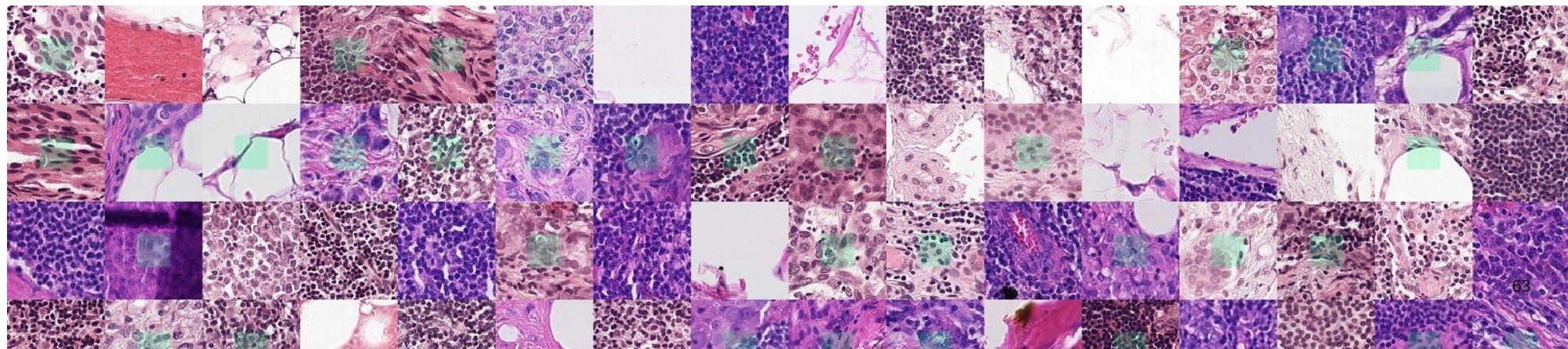
# Summary

- Interpretability + explainability important for practical ML use
- Generative adversarial networks are SOTA for image synthesis
- Conditional GANs synthesize samples with particular characteristics
- Adversarial networks can define loss functions that we cannot
- Many applications in medical image analysis
- **Self-study : unpaired image-to-image translation (Cycle GANs)**

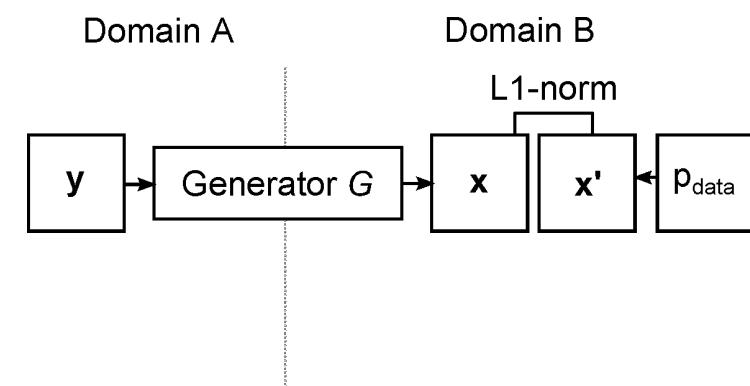
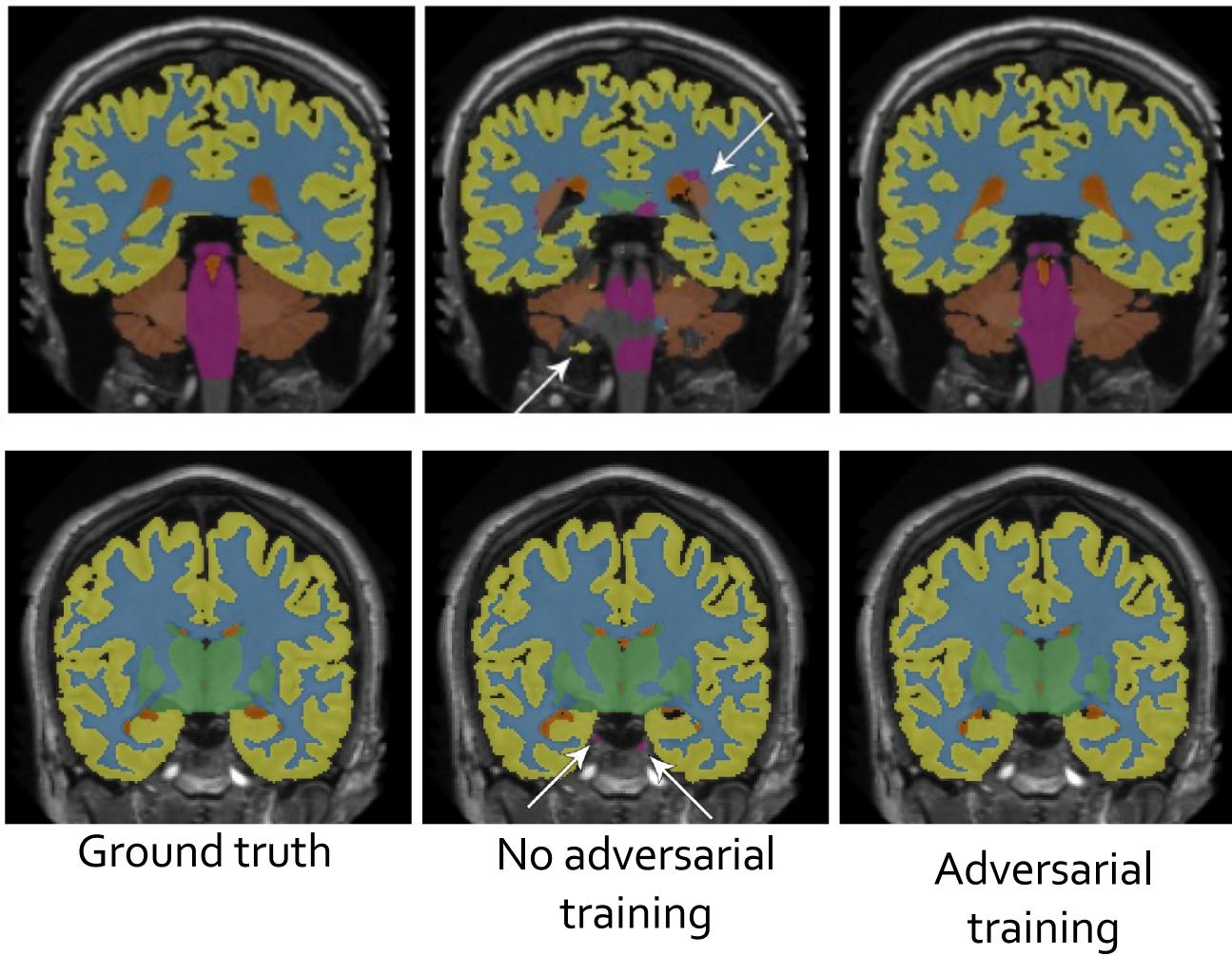
# Practical assignment

Train your own GAN! <https://tinyurl.com/capitaselectacolab>

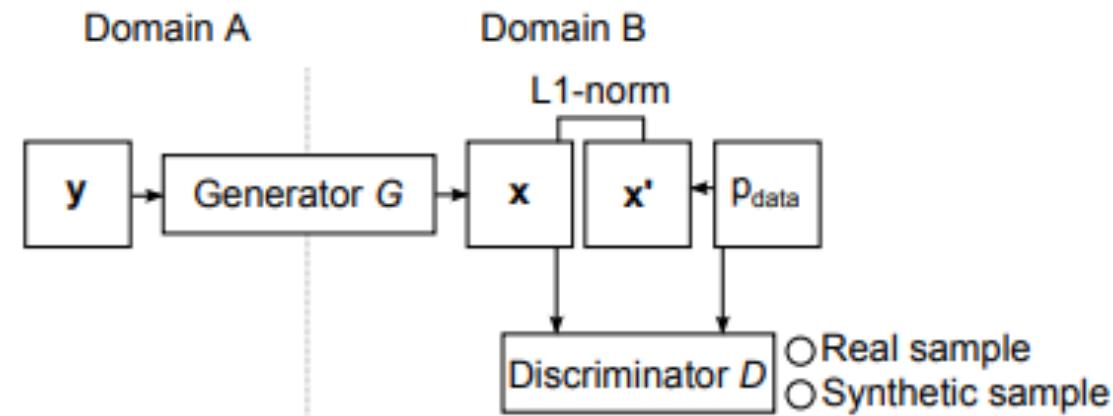
- Match a normal distribution
- MNIST digits (unconditional + conditional)
- Histopathology images: <https://github.com/basveeling/pcam>
- Run with free GPU: in Playground (no saving) or make copy in Google Drive (log in)
- Experiments with BigGAN <https://tinyurl.com/y8yuqyqv>



# Brain MR segmentation



# Unpaired training: self-regularization



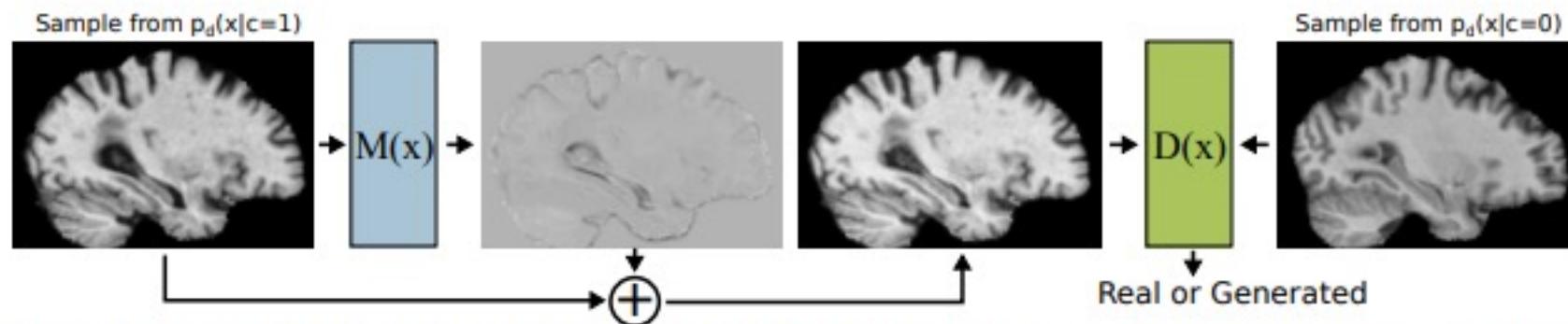
# Unpaired training: self-regularization

**Domain A**

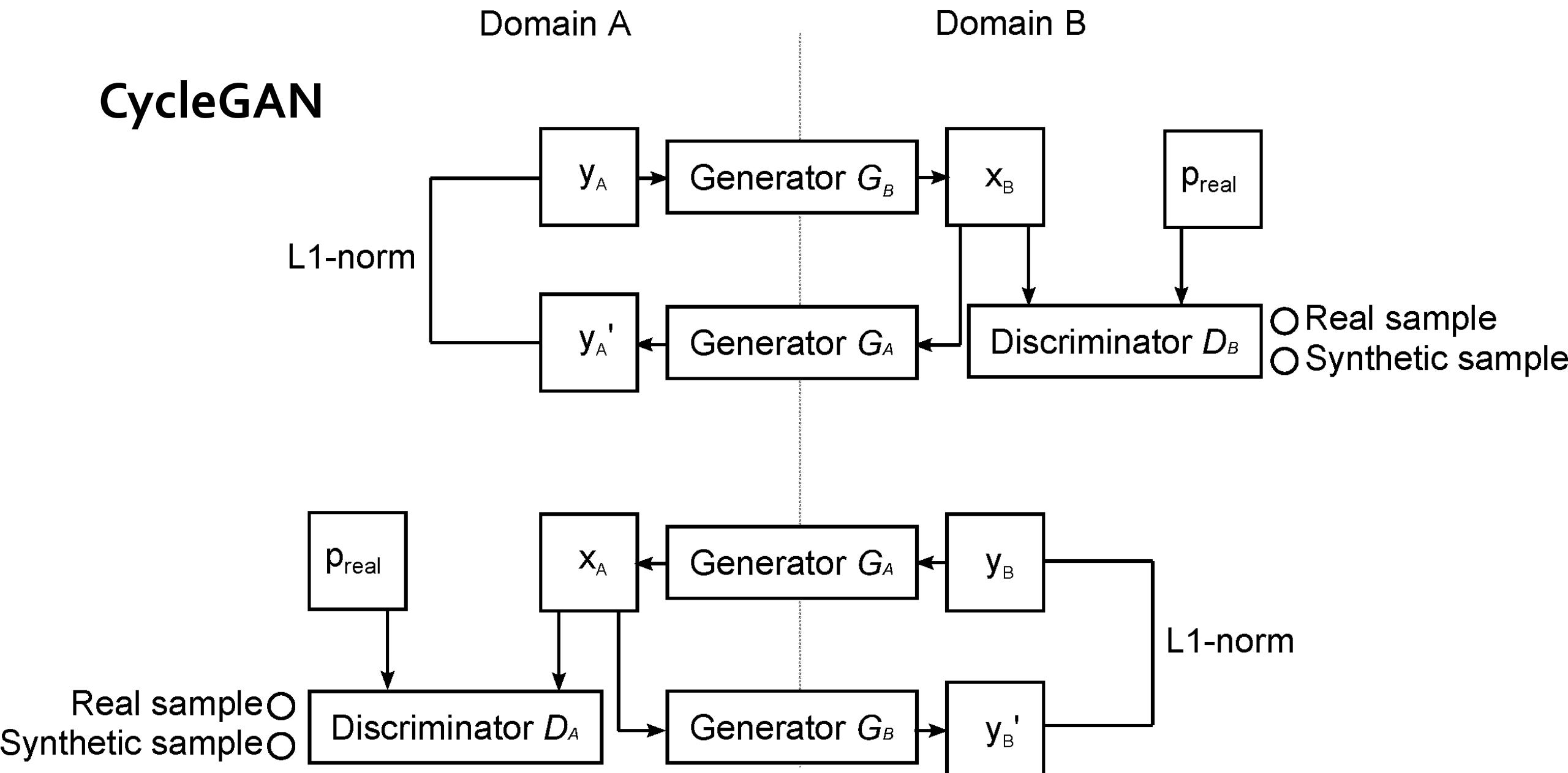
Brain MR images of patients with Alzheimer's

**Domain B**

Brain MR images of healthy patients



# CycleGAN



# CycleGAN

GAN objective functions in domain B and domain A and a cycle consistency loss

$$V_B(D_B, G_B) = \mathbb{E}_{x \sim p_{data_B}} [\log D_B(x)] + \mathbb{E}_{y \sim p_{data_A}} [\log (1 - D_B(G_B(y)))]$$

$$V_A(D_A, G_A) = \mathbb{E}_{y \sim p_{data_A}} [\log D_A(y)] + \mathbb{E}_{x \sim p_{data_B}} [\log (1 - D_A(G_A(x)))]$$

$$V_{Cycle}(G_A, G_B) = \mathbb{E}_{y \sim p_{data_A}} [||G_A(G_B(y)) - y||_1] + \mathbb{E}_{x \sim p_{data_B}} [||G_B(G_A(x)) - x||_1]$$

$$\min_{G_A, G_B} \max_{D_A, D_B} V(G_A, G_B, D_A, D_B) = V_B(D_B, G_B) + V_A(D_A, G_A) + \lambda V_{Cycle}(G_A, G_B)$$

# CycleGAN

**Domain A**



**Domain B**

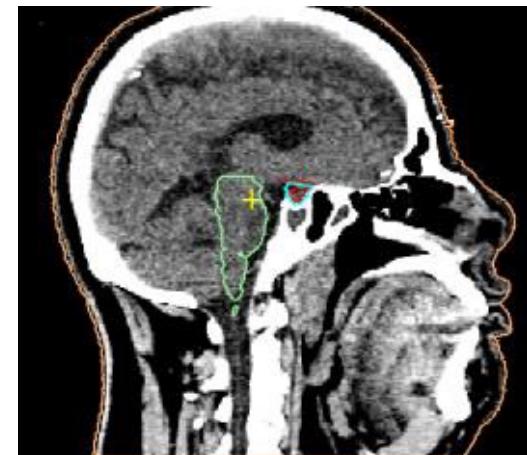


# Example: MR to CT synthesis

- Radiotherapy treatment planning requires
  - MR volume
    - Soft tissue contrast
    - Tissue delineation
  - CT volume
    - Electron density
    - Dose calculation
- Acquisition of both volumes leads to
  - Increase in time and money
  - Decrease in patient comfort

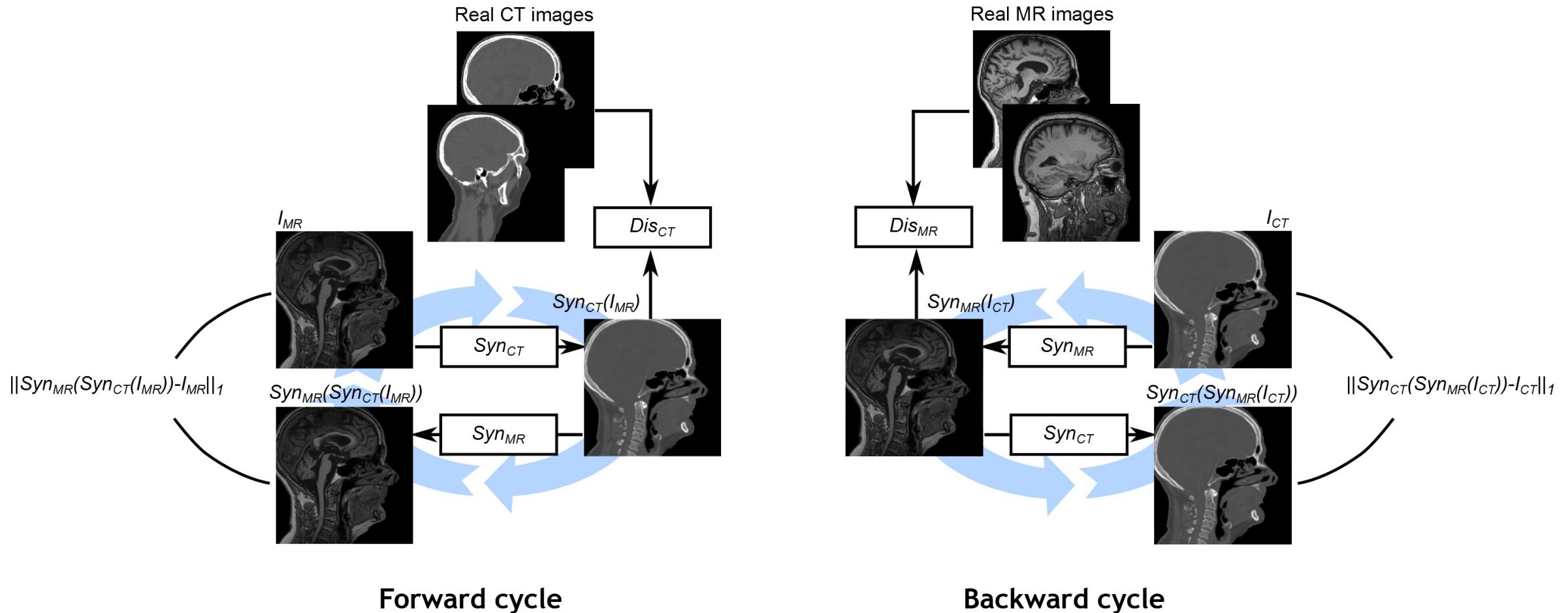


MR



CT

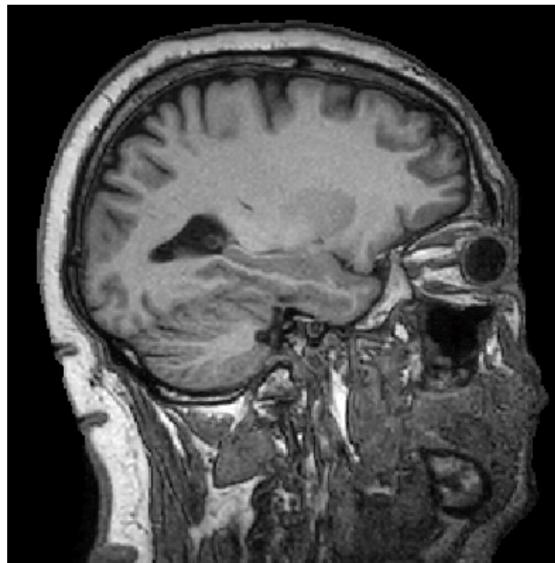
# Example: MR to CT synthesis



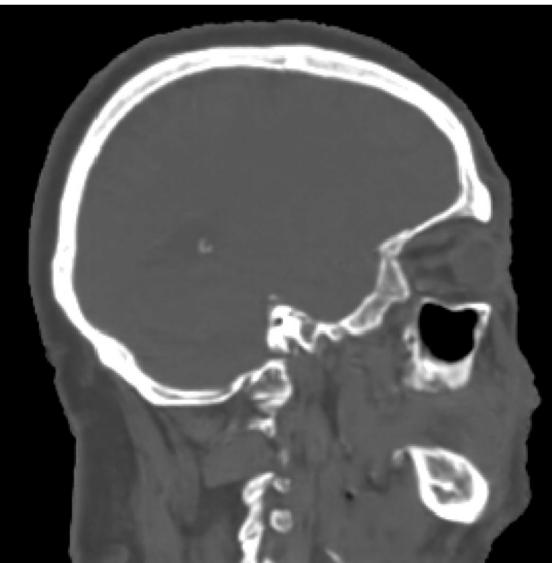
Forward cycle

Backward cycle

# Example: MR to CT synthesis



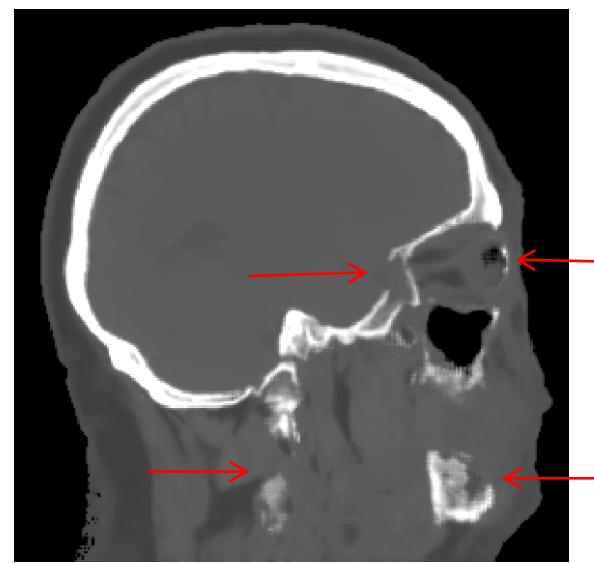
$I_{MR}$



$Syn_{CT}(I_{MR})$



$I_{CT}$



*Regression*

## Example: MR to CT synthesis



$I_{MR}$

$Syn_{CT}(I_{MR})$

$I_{CT}$

$|Syn_{CT}(I_{MR}) - I_{CT}|$

# CycleGAN

Domain A



Domain B

