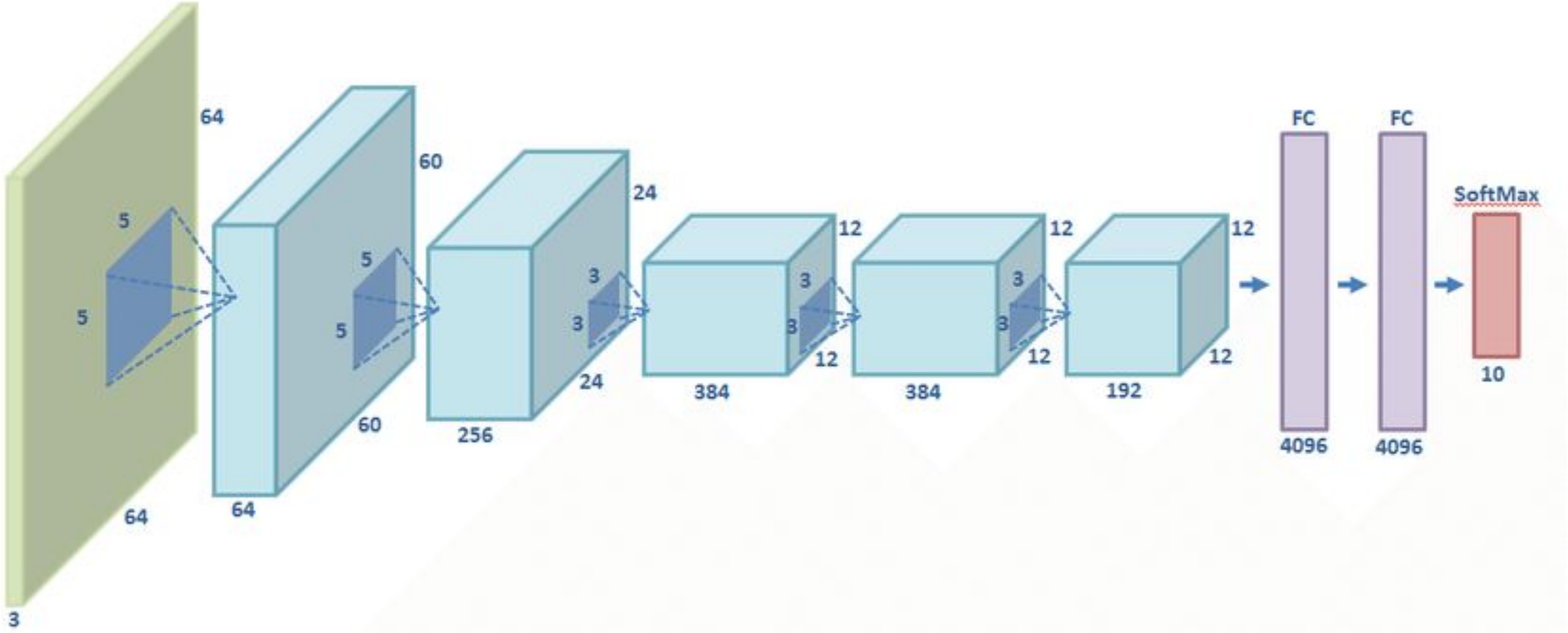
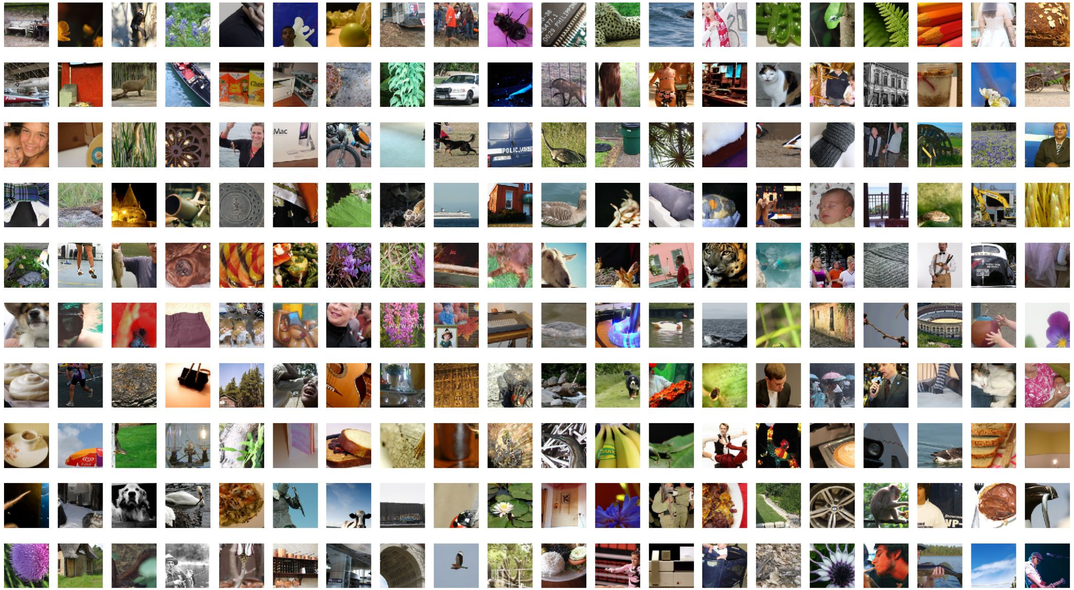
History of Convolutional Neural Networks



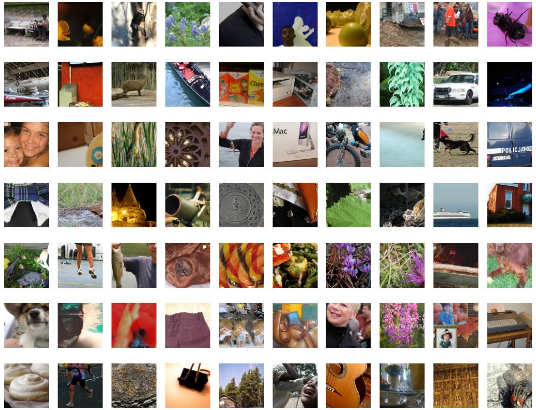
ImageNet dataset



Dataset size:
14,000,000 + images
30,000+ classes

Image size 224x224

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Dataset size:
20,000 images
1000 classes

Task:
Predict which class picture belongs to.

Evaluation types:

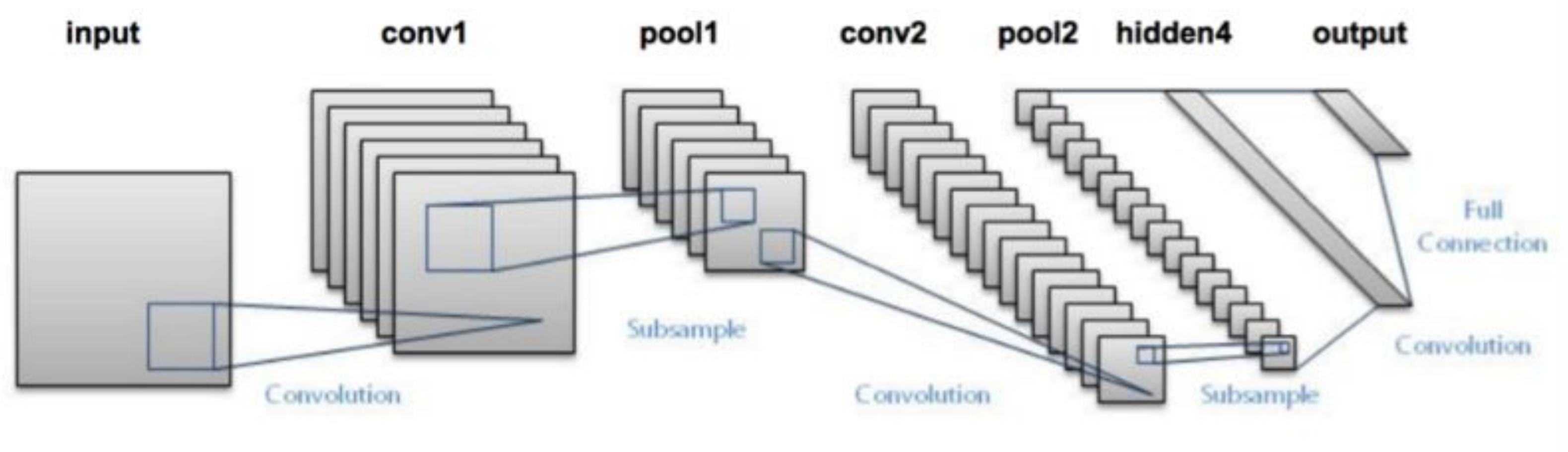
1. Top 5 error.

The percentage of the images that the classifier did not include the correct class among its top 5 guesses.

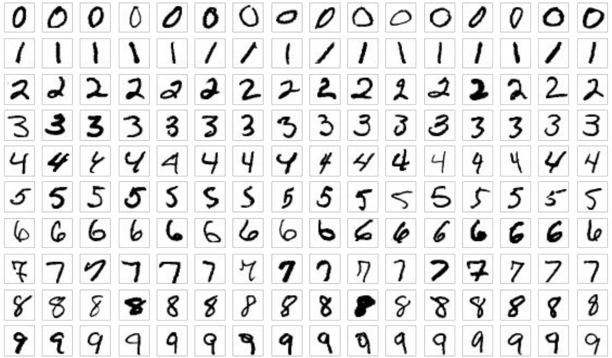
2. Top 1 error.

The percentage of the images that the classifier did not give the correct class the highest score.

LeNet-5 (1998)

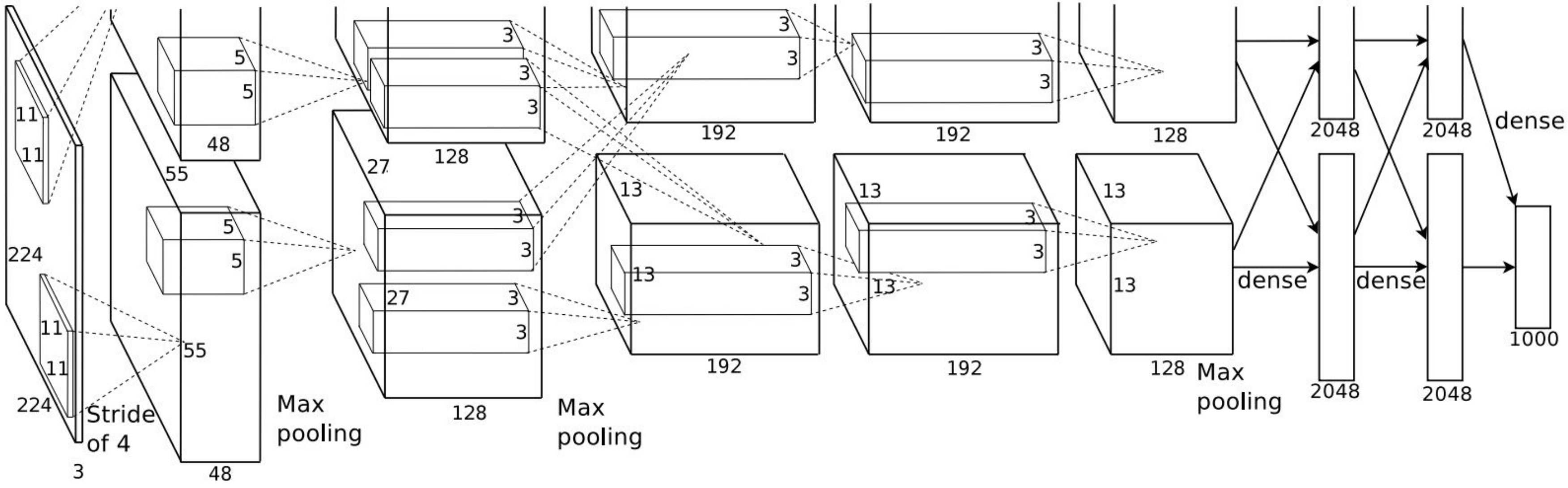

input
conv1
pool1
conv2
pool2
hidden4
output
Convolution
Subsample
Convolution
Subsample
Convolution
Full
Connection

LeNet-5 (1998)



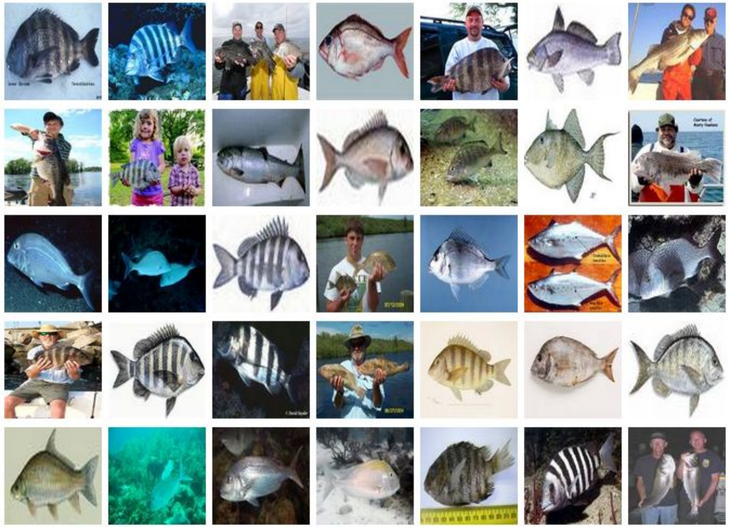
- Small 7 layers network developed by Yann Lecun.
- Used for recognizing 32x32 hand-written numbers.
- It required a lot of resources for training that were not available in 1998.

Alexnet (2012)

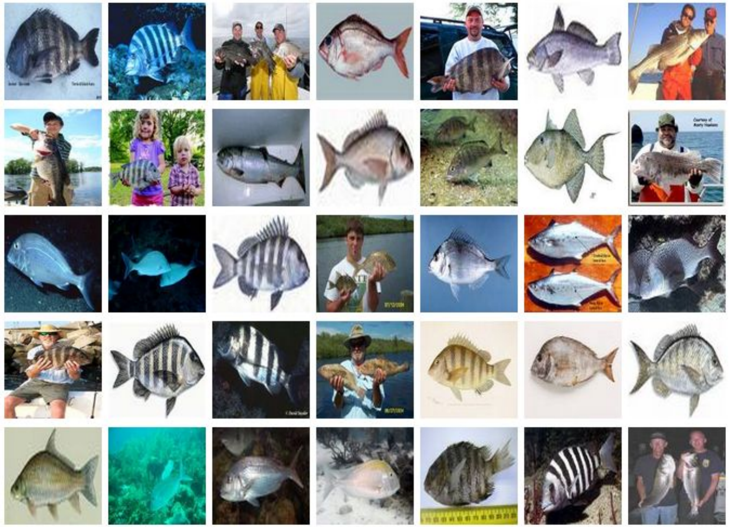


Alexnet (2012)

- Winner of ILSVRC 2012 with 15.3% of top 5 error.
- Deeper architecture with huge convolutions 11x11.
- Was trained on 2 GPU for 6 days.
- 60 million parameters.



Alexnet (2012)

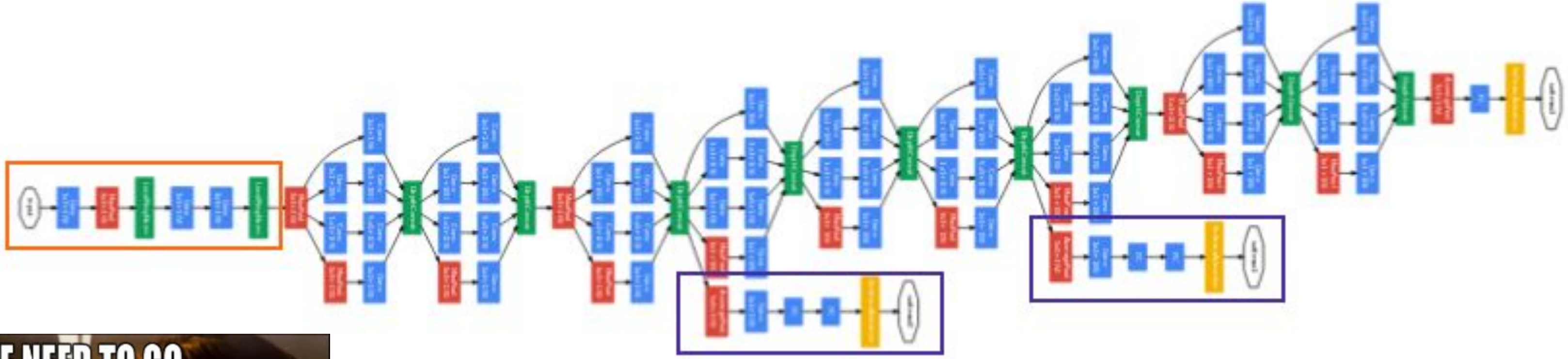


- Winner of ILSVRC 2012 with 15.3% of top 5 error.
- Deeper architecture with huge convolutions 11x11.
- Was trained on 2 GPU for 6 days.
- 60 million parameters.

Drawbacks:

- Huge convolutions with a lots of weights.

GoogLeNet or Inception v1 (2014)



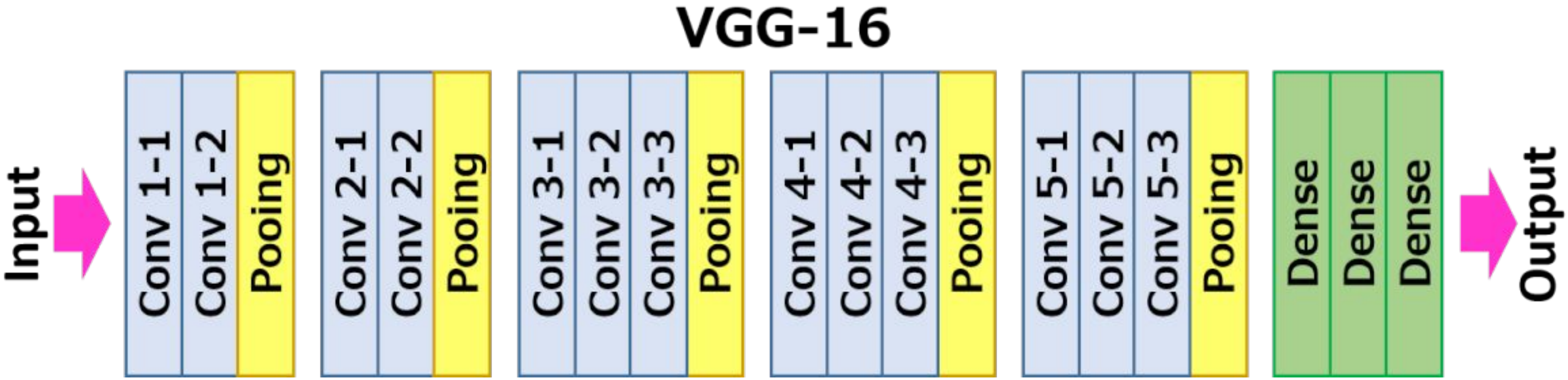
GoogLeNet or Inception v1 (2014)

- Winner of ILSVRC 2014 with 6.67% of top 5 error.
  (Right now it is better than human)
- Huge block structure with 22 layers.
- Small number of parameters. (4 million)

Drawbacks:

- Need a lot of time to reach a good quality.

VGG (2014)


VGG-16
Input
Conv 1-1
Conv 1-2
Pooing
Conv 2-1
Conv 2-2
Pooing
Conv 3-1
Conv 3-2
Conv 3-3
Pooing
Conv 4-1
Conv 4-2
Conv 4-3
Pooing
Conv 5-1
Conv 5-2
Conv 5-3
Pooing
Dense
Dense
Dense
Output

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 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

VGG (2014)

- 2d place on ILSVRC 2014 with 7.32% of top 5 error.
- Only 3x3 convolutions and poolings!
- There are a lot of different versions from small VGG7 to huge VGG19. (133 - 144 million parameters)

Drawbacks:

- All parameters in fully connected layers in the end of the network.
- Vanishing gradients.

Vanishing gradients

Reminder about weights update process

$$w_{new} = w_{old} - \alpha \frac{\partial L(w,x)}{\partial w}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{f_{out}} * \frac{\partial f_{out}}{W_N} * \ldots * \frac{\partial f_1}{W_1} X$$

< 1

Vanishing gradients

Reminder about weights update process

$$w_{new} = w_{old} - \alpha \frac{\partial L(w,x)}{\partial w}$$

< 1 < 1

$$\frac{\partial L}{\partial w} = \frac{\partial L}{f_{out}} * \frac{\partial f_{out}}{W_N} * \dots * \frac{\partial f_1}{W_1} X$$

Vanishing gradients

Reminder about weights update process

$$w_{new} = w_{old} - \alpha \frac{\partial L(w,x)}{\partial w}$$

< 1 $\quad$ < 1

$$\frac{\partial L}{\partial w} = \frac{\partial L}{f_{out}} * \frac{\partial f_{out}}{W_N} \underbrace{* \dots *}_{\text{A lot of parameters!}} \frac{\partial f_1}{W_1} X$$

≈ 0

Vanishing gradients

Reminder about weights update process

$$w_{new} = w_{old} - \alpha \frac{\partial L(w,x)}{\partial w}$$

$$\underbrace{\frac{\partial L}{\partial w}}_{\approx 0} = \frac{\partial L}{f_{out}} * \overbrace{\frac{\partial f_{out}}{W_N}}^{<1} \underbrace{* \ldots *}_{\text{A lot of parameters!}} \overbrace{\frac{\partial f_1}{W_1} X}^{<1}$$

Thus,

$$w_{new} = w_{old} - \alpha \cancelto{0}{\frac{\partial L(w,x)}{\partial w}}$$

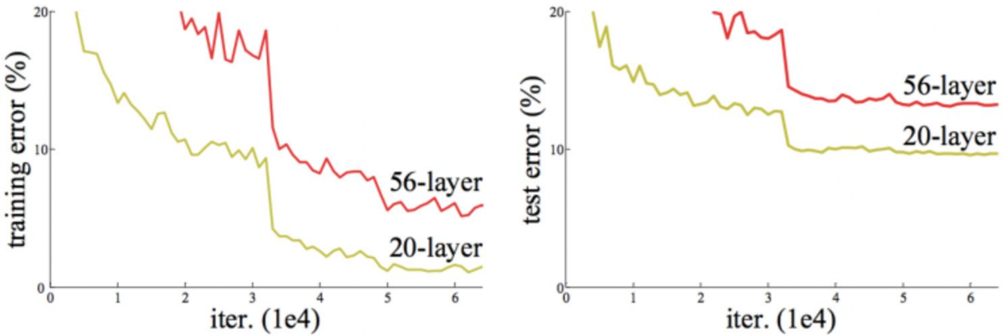
$$w_{new} = w_{old}$$ Almost no updates!

Vanishing gradients

Reminder about weights update process

$$w_{new} = w_{old} - \alpha \frac{\partial L(w,x)}{\partial w}$$

$$\underbrace{\frac{\partial L}{\partial w}}_{\approx 0} = \frac{\partial L}{f_{out}} * \overbrace{\frac{\partial f_{out}}{W_N}}^{<1} \underbrace{* \dots *}_{\text{A lot of parameters!}} \overbrace{\frac{\partial f_1}{W_1} X}^{<1}$$

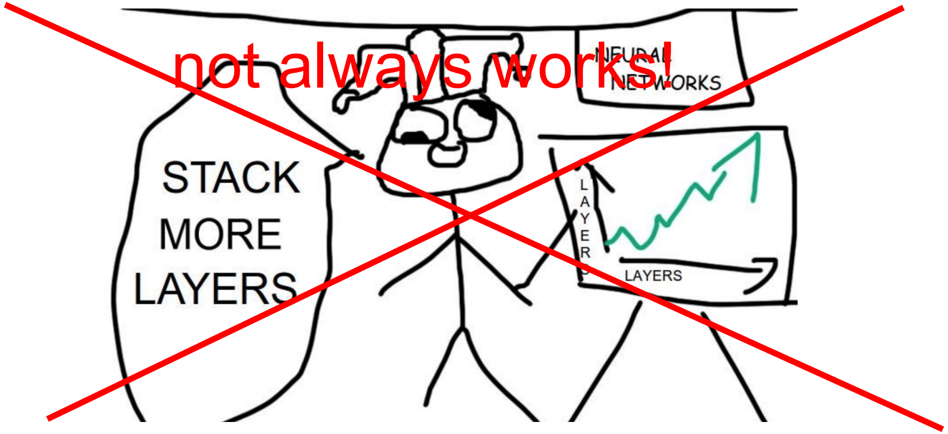


It means that networks with huge number of layers will be train very slowly.

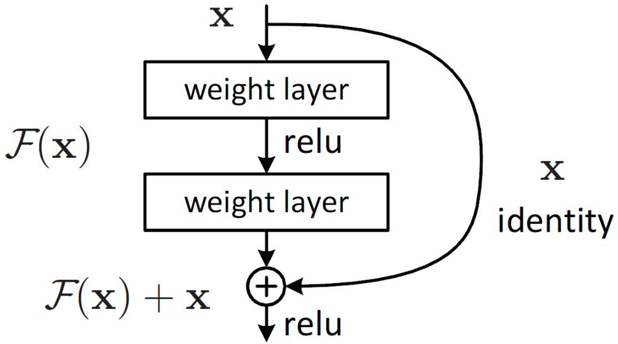
Thus,

$$w_{new} = w_{old} - \alpha \cancelto{0}{\frac{\partial L(w,x)}{\partial w}}$$

$$w_{new} = w_{old}$$ Almost no updates!



ResNet (2015)



$\mathbf{x}$
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
$\mathbf{x}$
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

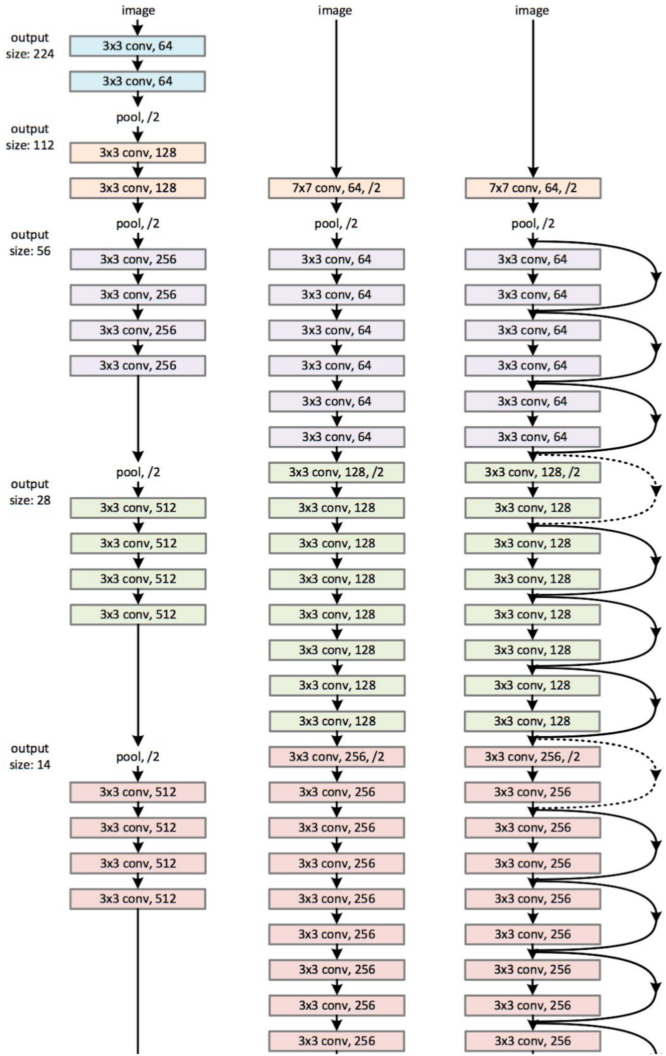
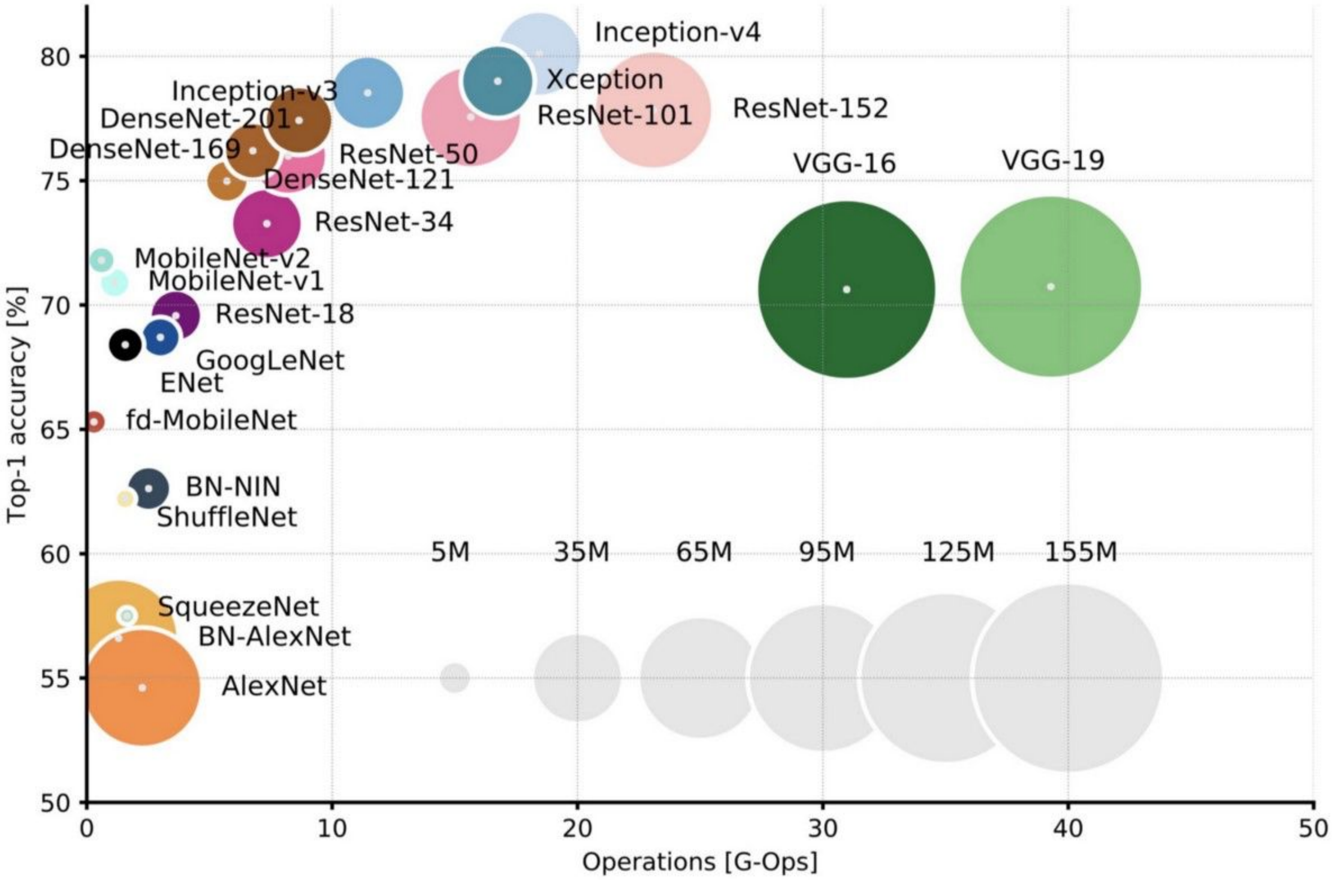


image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256

ResNet (2015)

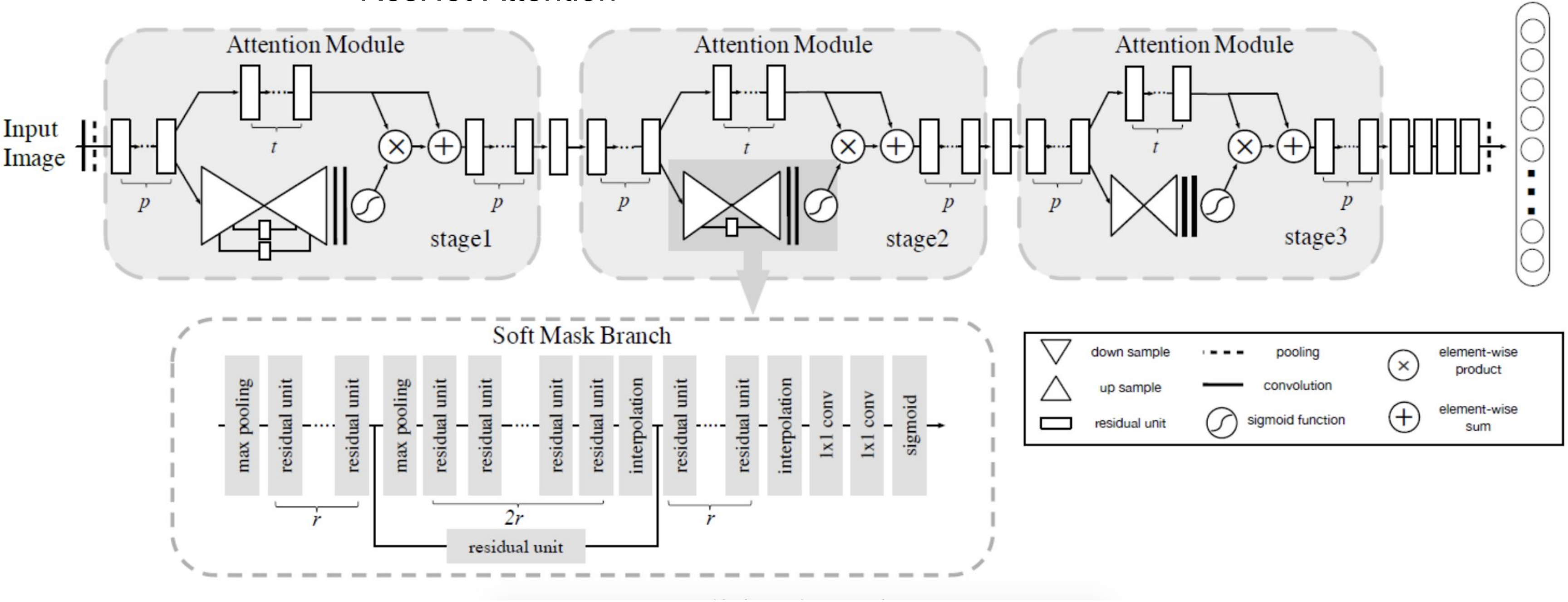
- Winner of ILSVRC 2015 with top 5 error 3.5%
- No vanishing gradients
- Really huge networks, started from ResNet18 to ResNet152.

A lots of other architectures here..

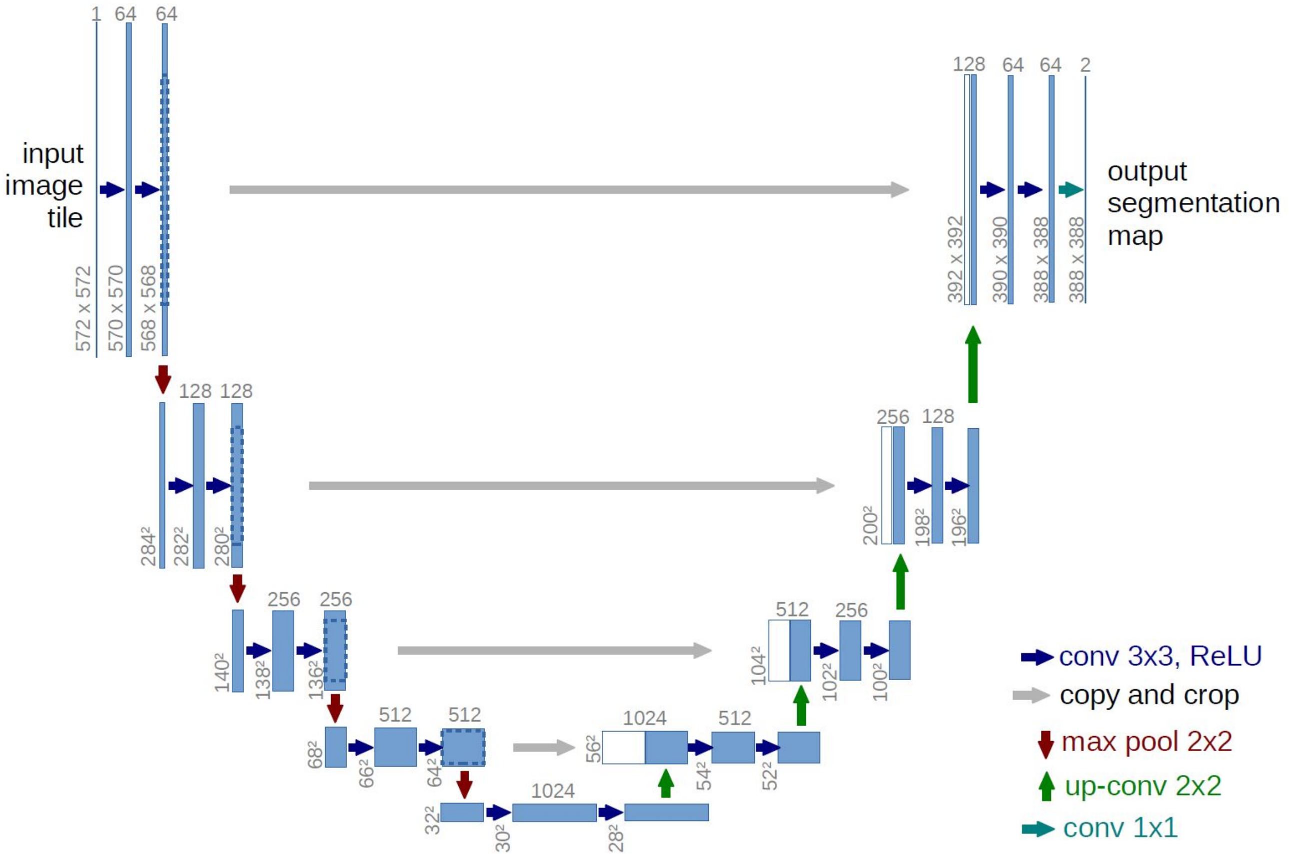


Attention models

ResNet-Attention



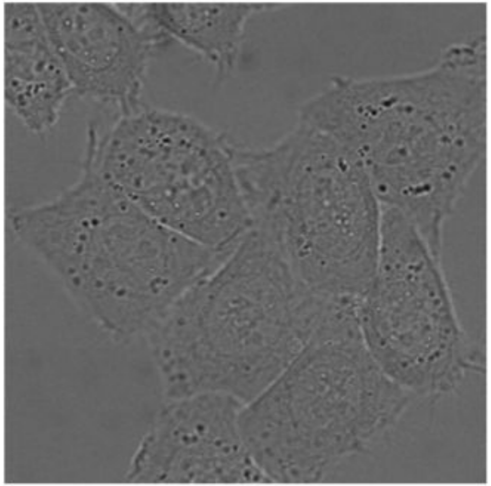
UNet (2015)

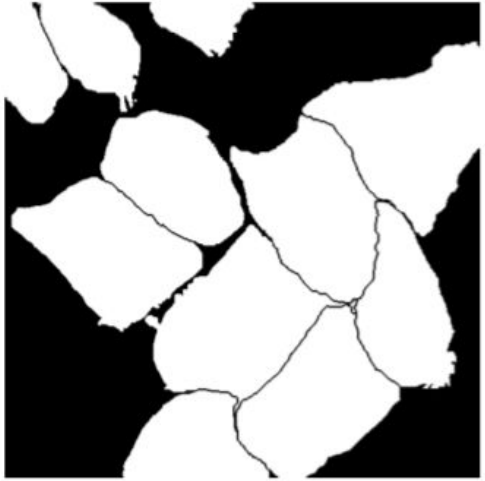


1
64
64
input
image
tile
572 x 572
570 x 570
568 x 568
128
128
284^2
282^2
280^2
256
256
140^2
138^2
136^2
512
512
68^2
66^2
64^2
1024
32^2
30^2
28^2
1024
512
56^2
54^2
52^2
512
256
104^2
102^2
100^2
256
128
200^2
198^2
196^2
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
output
segmentation
map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

UNet (2015)

Binary segmentation

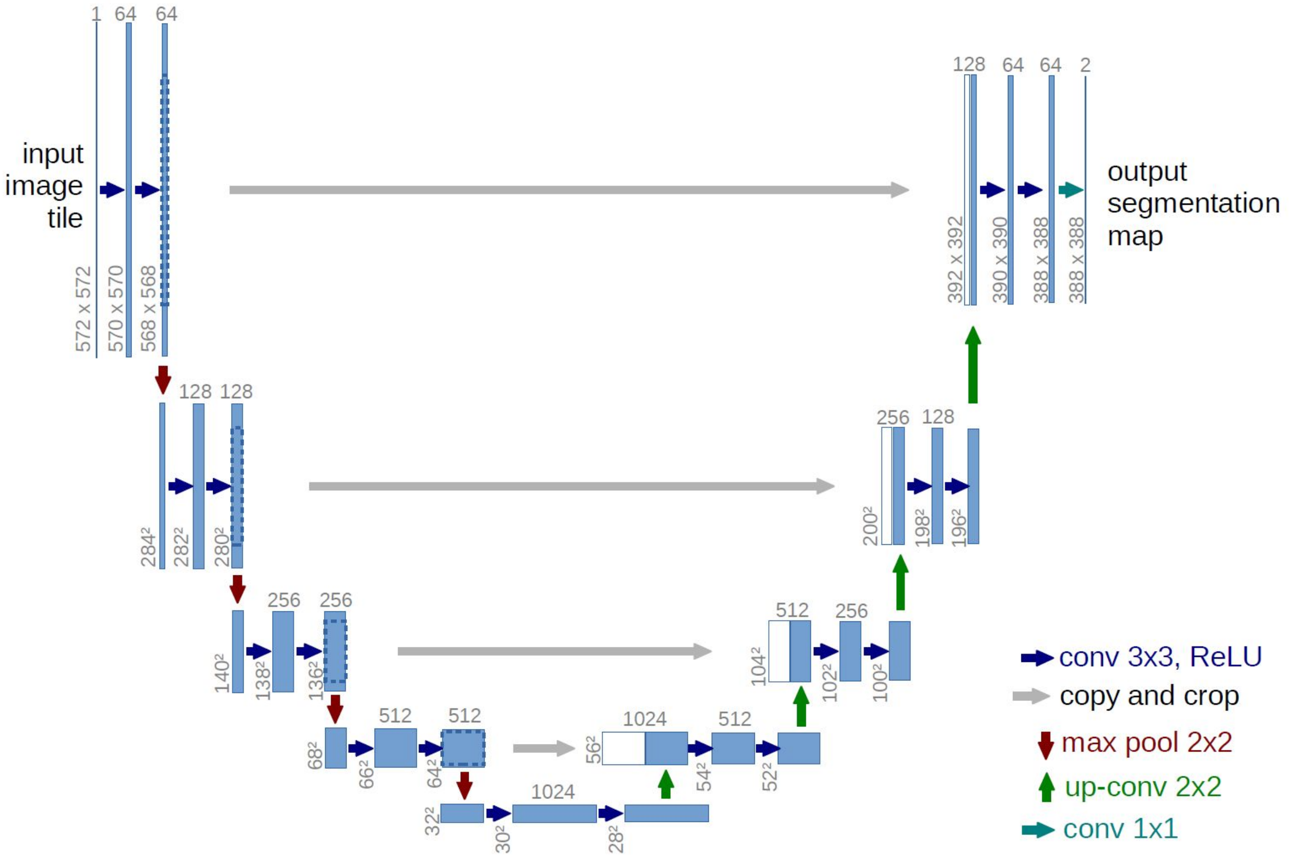




UNet (2015)

2015? To old!

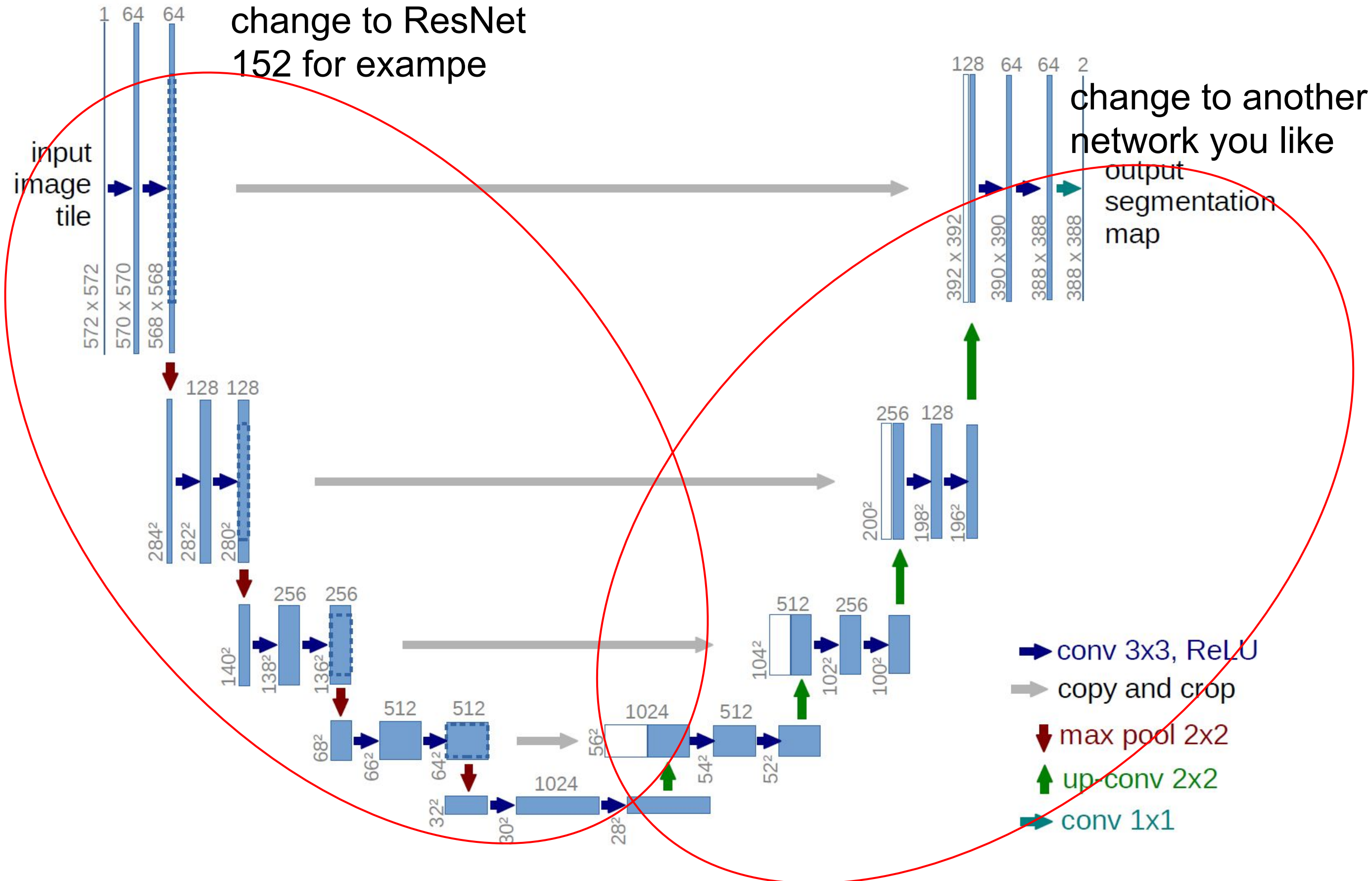
What should we do then?



UNet (2015)

2015? To old!

What should we do then?



What tasks we can solve by CNN?

- regression
- classification
- segmentation
- detection

What tasks we can solve by CNN?

- regression
- classification
- segmentation
- detection

What is the output of the network?

CNN with regression

CNN

Prediction

4

number of
dogs

CNN with classification

f :

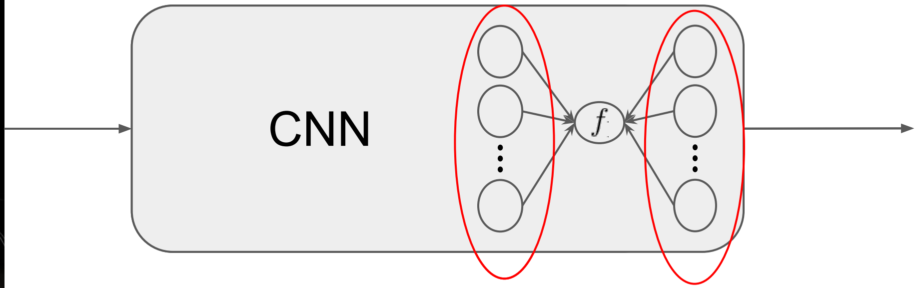
$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Softmax

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

Sigmoid

Probability for each class

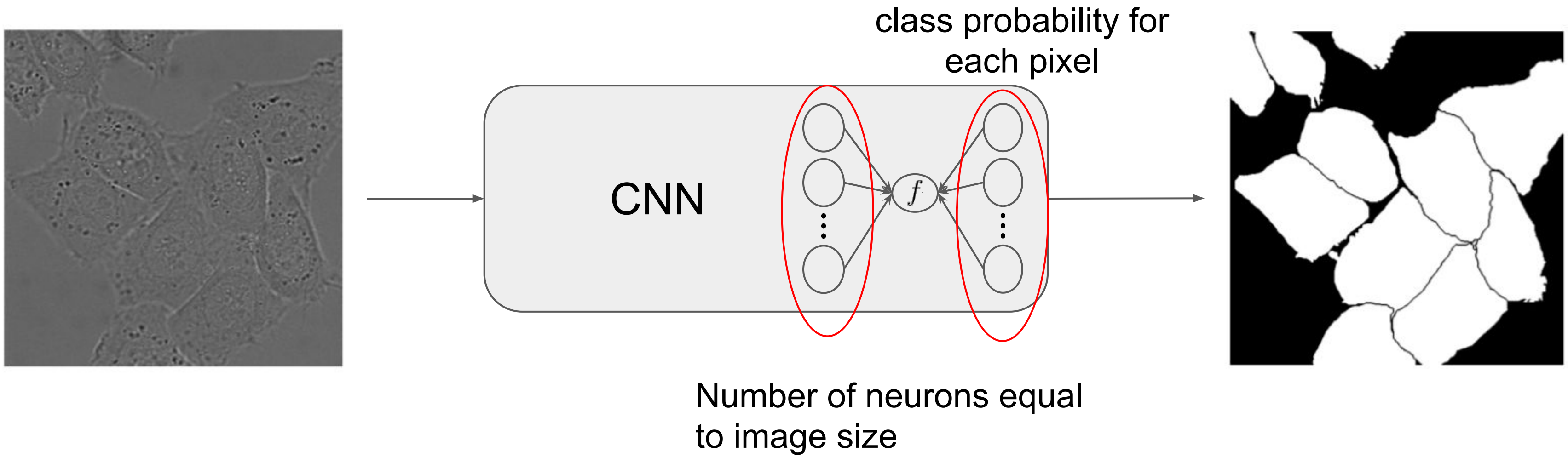


class (cat)

Number of neurons equal
to number of classes

CNN with classification

CNN with segmentation



CNN with detection one object

4 neurons:

- length of bb
- width of bb
- x
- y

CNN with detection for several objects

Number of neurons in output layer is 5xWxH.

CNN with detection for several objects

Number of neurons in output layer is WxHx(5+NumClass).

- confidence in this bb
- length of bb
- width of bb
- x
- y
- probability for each class