Famous architectures

András Horváth

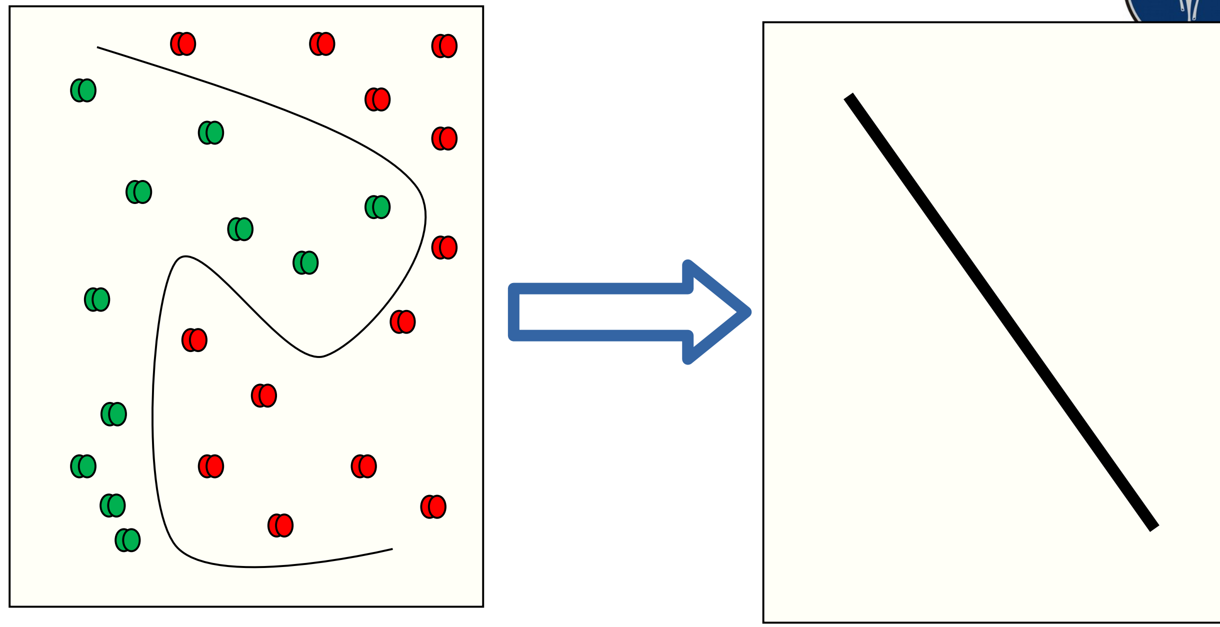
Budapest, 2018.12.03

Neural Networks

- Classification - decision
- FNN, SVM – linear classification

Is X larger than a limit? X>k?

- Finding a good feature representation:
  - Meaningful
  - Sparse - low dimensions
  - Ensures easy separation



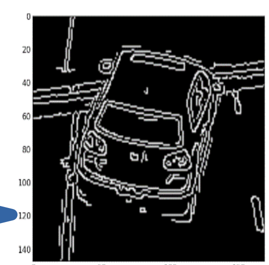
Input space *Feature space*

Finding the representation with the help of machine learning

Input Image

Convolution Kernel

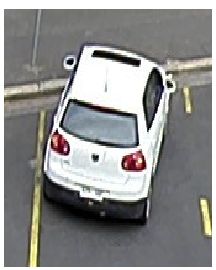
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



Feature Image

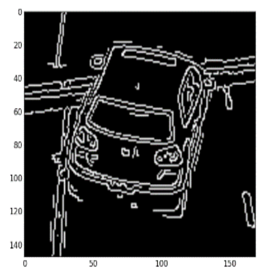
Convolutional neural networks

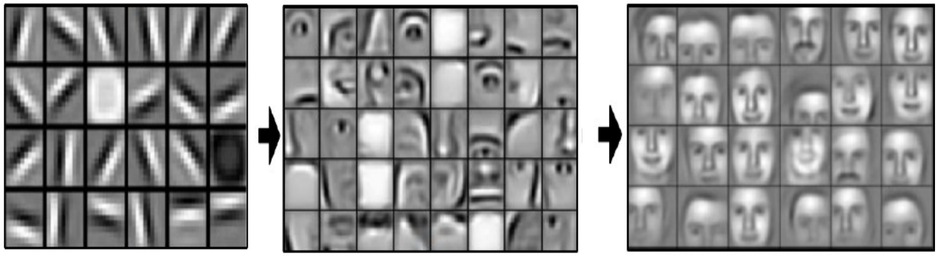
- A network of simple processing elements
  - Elements:
    - Convolution
    - ReLU
    - Pooling



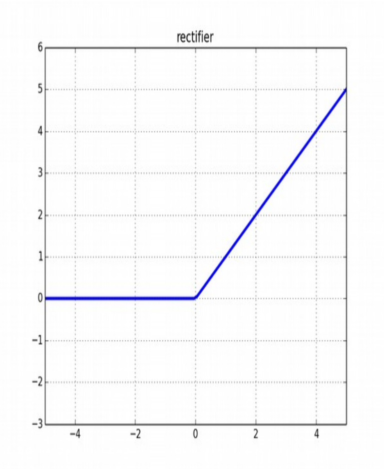
Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

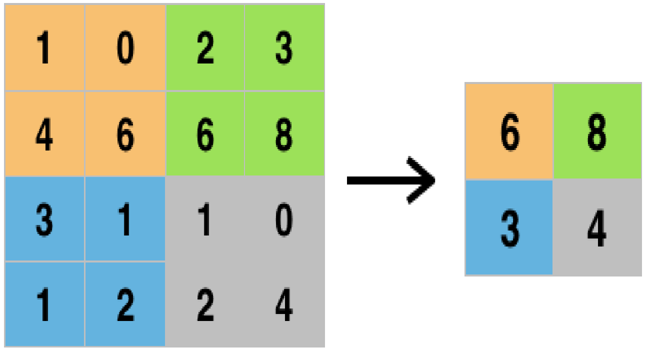




Low layers *Middle layers* *High layers*



Thresholding all values below zero



Selection of the maximal response in an area

Convolutional networks

Ok, but how many layers do we need?

How many features should be in each layer?

What should be the network architecture?

Convolutional networks

Ok, but how many layers do we need?

How many features should be in each layer?

What should be the network architecture?

These are called hyper-parameters:

Along whit: non-linearity type, batch-norm, dropout etc.

Convolutional networks

Ok, but how many layers do we need?

How many features should be in each layer?

What should be the network architecture?

These are called hyper-parameters:

Along whit: non-linearity type, batch-norm, dropout etc.

We can use a network which performed fairly well on an other dataset

It will probably work well on our task too

Large scale visual recognition challenge

Large scale visual recognition challenge

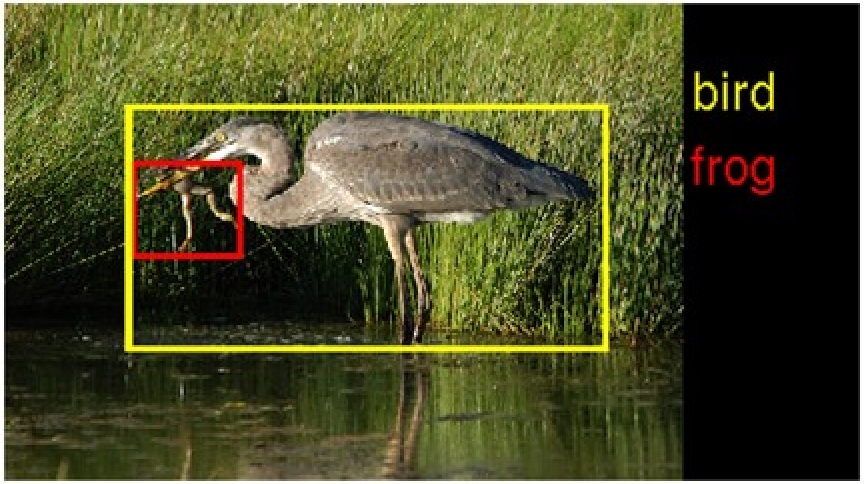
ImageNet:

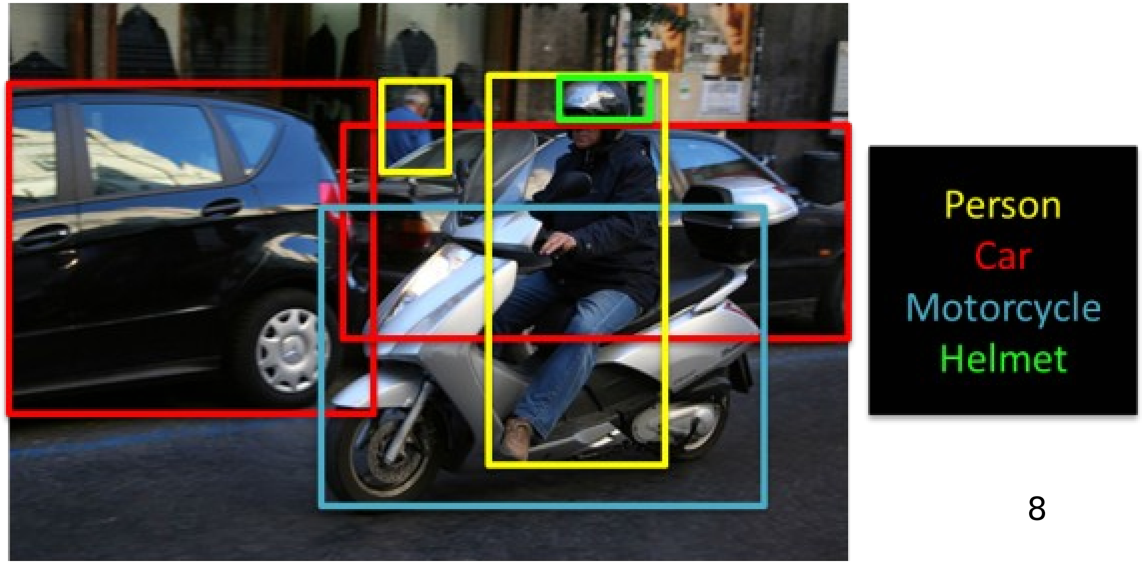
Over 15 million images, more than 22k categories

Detection and Classification

Detection for 200 fully labeled categories. 0.73 % mAP

Classification for 1000 categories. 0.97%





Large scale visual recognition challenge

Andrej Karpathy – „the human benchmark”

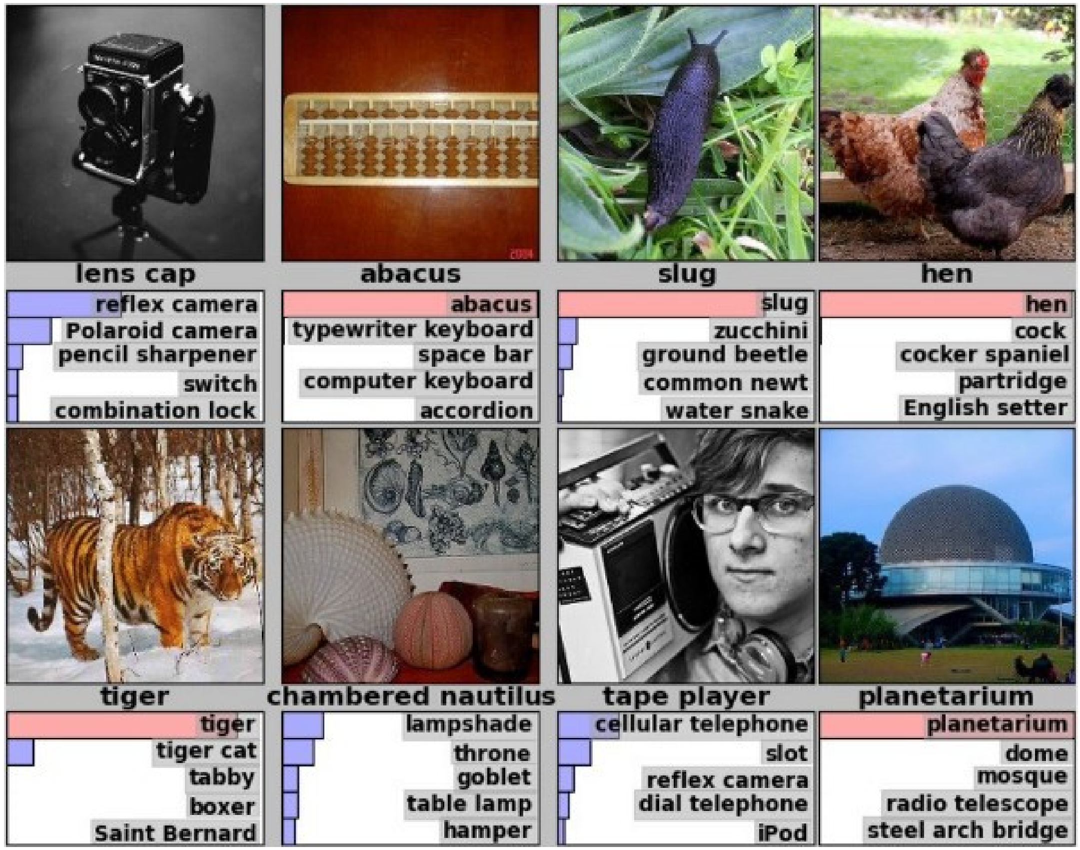
Shown 13 samples from each category

94.9% accuracy

Large scale visual recognition challenge

Andrej Karpathy

Shown 13 samples from each category



Large scale visual recognition challenge

Andrej Karpathy

Shown 14 samples from each category

Can be tried out online:

https://cs.stanford.edu/people/karpathy/ilsvrc/

rule, ruler	king crab, Alaska crab	sidewinder	saltshaker, salt shaker	reel	hatchet	schipperke
pencil box, pencil case	pizza, pizza pie	maze, labyrinth	pill bottle	stethoscope	vase	schipperke
rubber eraser, rubber	strawberry	gar, garfish	water bottle	whistle	pitcher, ewer	groenendael
ballpoint, ballpoint pen	orange	valley, vale	lotion	ice lolly, lolly	coffeepot	doormat, welcome mat
pencil sharpener	fig	hammerhead	hair spray	hair spray	mask	teddy, teddy bear
carpenter's kit, tool kit	ice cream, icecream	sea snake	beer bottle	maypole	cup	jigsaw puzzle

Alexnet

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (2012)

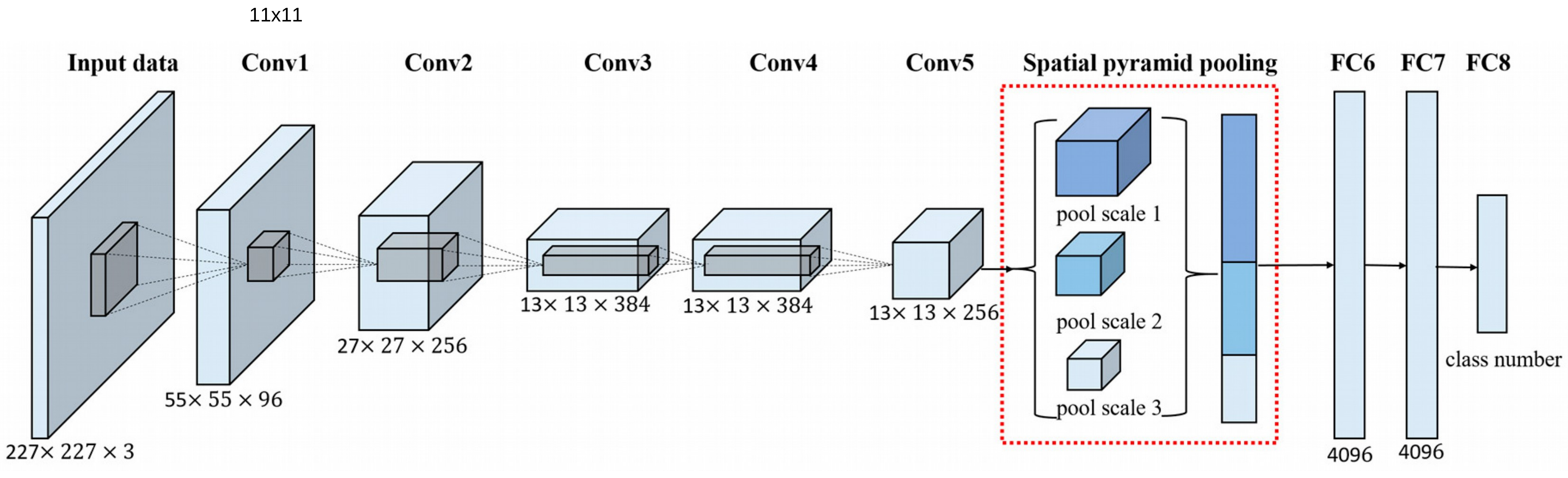
Trained whole ImageNet (15 million,22,000 categories)

Used data augmentation (image translations, horizontal reflections, and patch extractions)

Used ReLU for the nonlinearity functions (Decreased training time compared to tanh) - Trained on two GTX 580 GPUs for six days

Dropout layers

2012 marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4% (next best entry was with error of 26.2%)



VGG - 16/19

Karen Simonyan and Andrew Zisserman of the University of Oxford, 2014 Visual Geometry Group

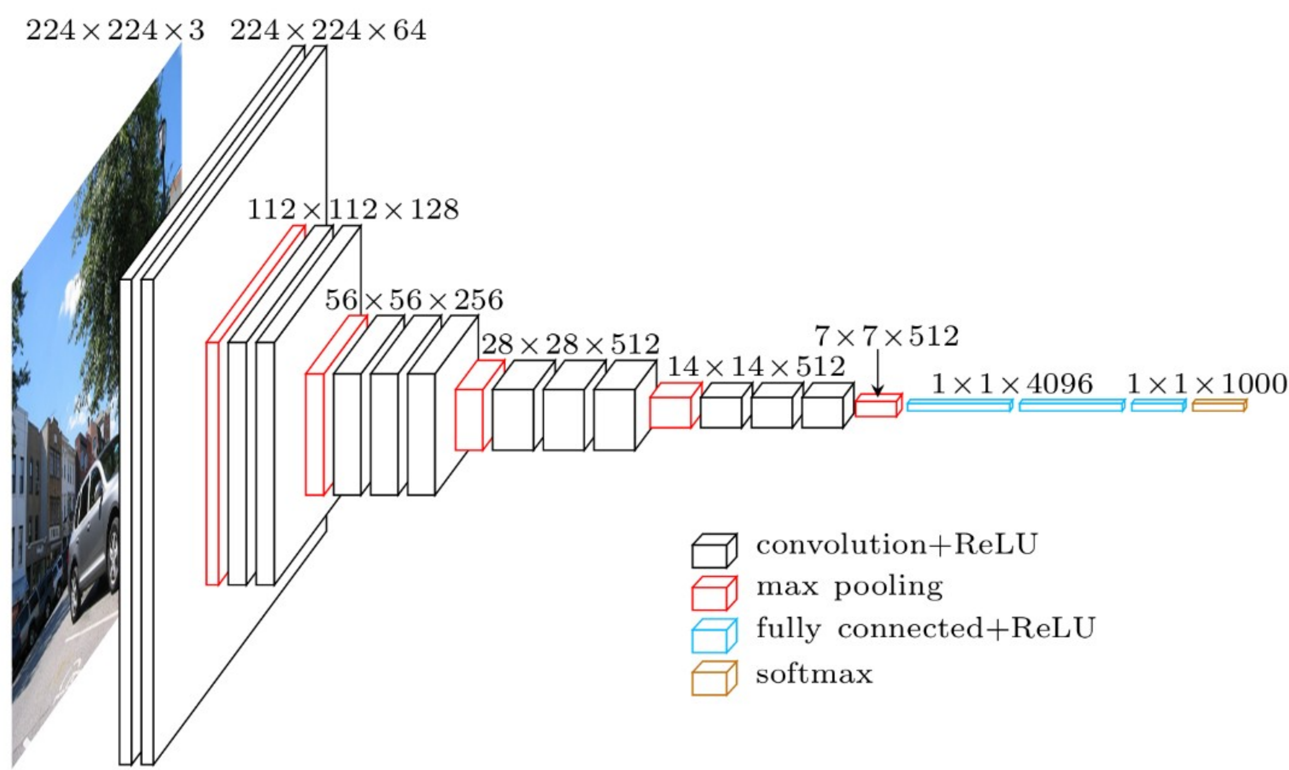
As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network.

Shrinking spatial dimensions but grwoing depth

3x3 filters with stride and pad of 1, along with 2x2 maxpooling layers with stride 2

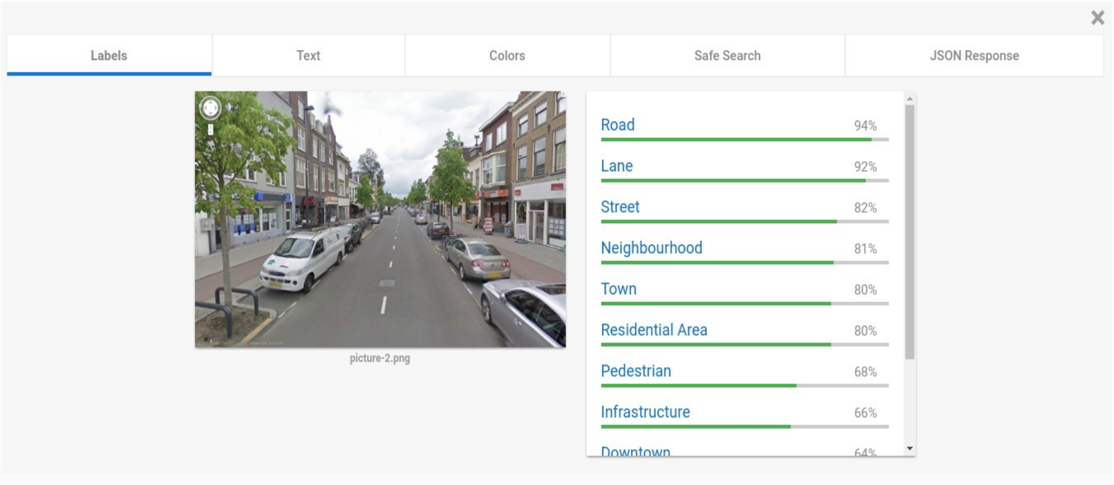
7.3% error rate

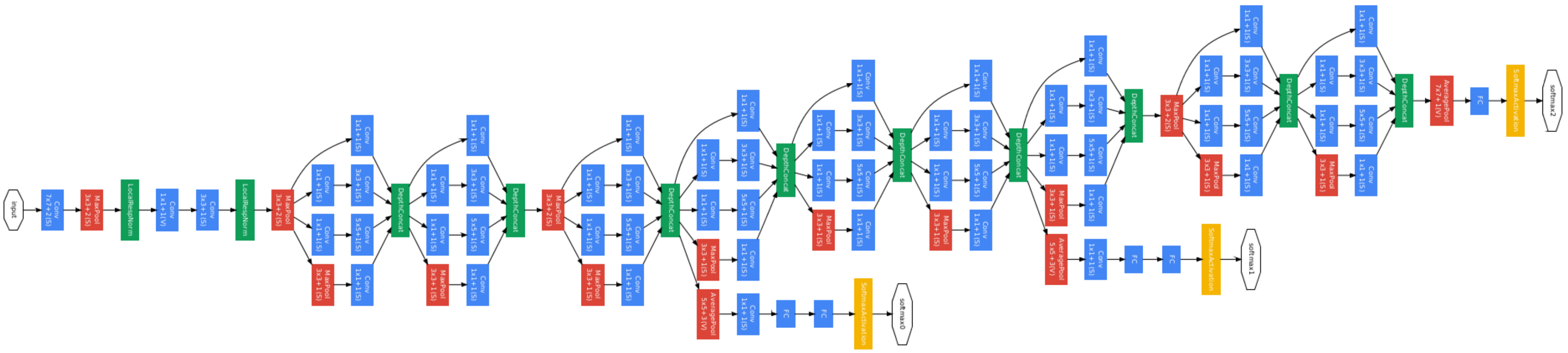
Simple architecture, still the swiss knife of deep learning



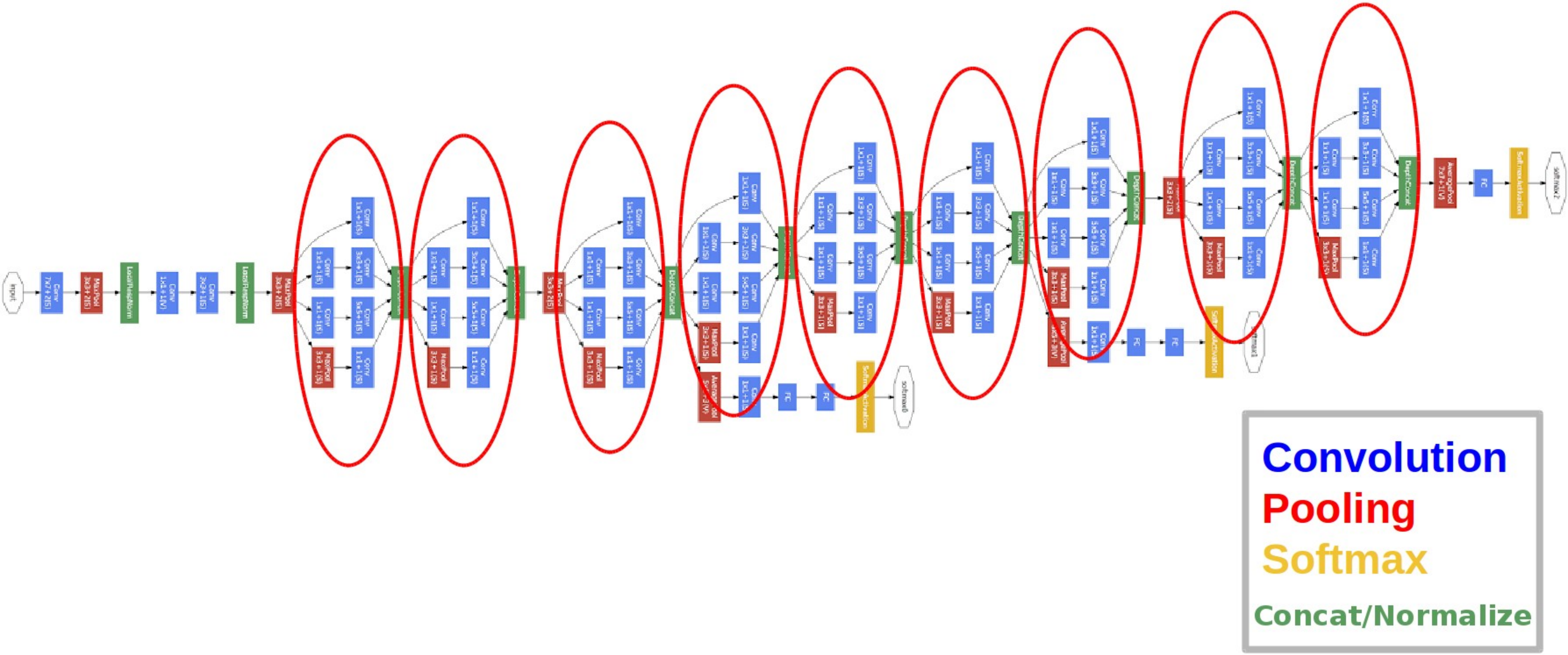
Google - Inception arhcitecture

- GoogLeNet:
22/42 layers (9 inception_v3 layers)
5 million free parameters
~1.5B operations/evaluations
Demo:https://cloud.google.com/vision/





Inception module



Inception

Google, Christian Szegedy

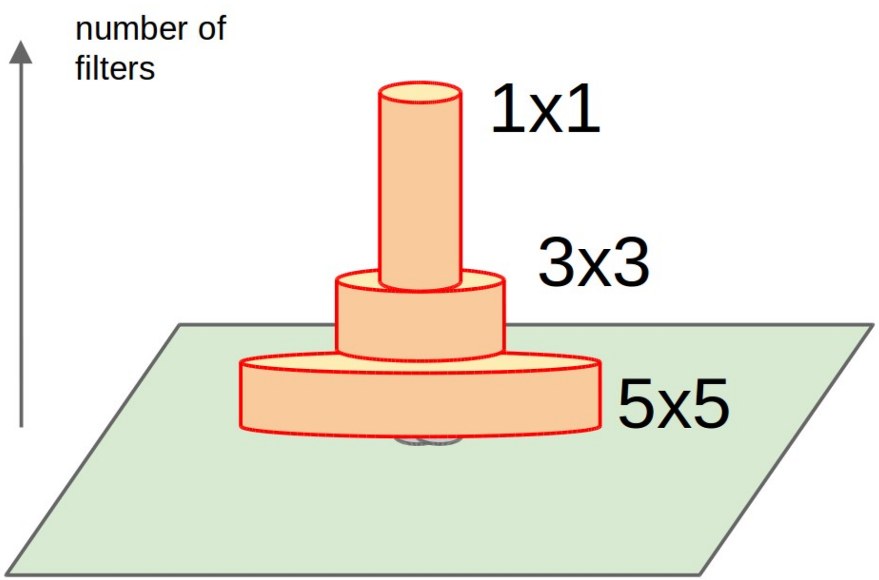
2014 with a top 5 error rate of 6.7%

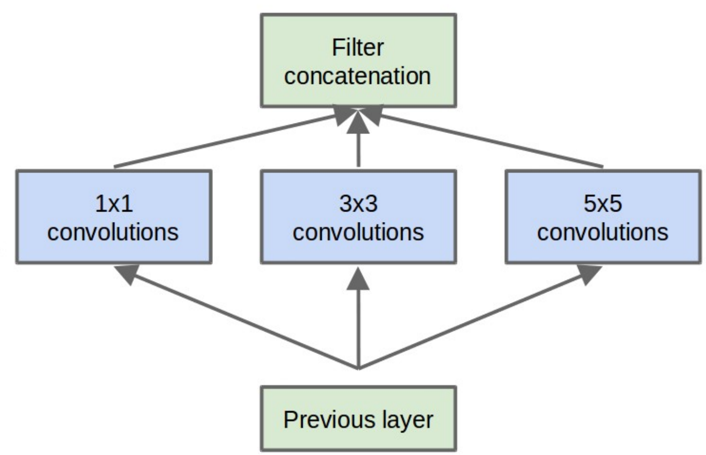
his can be thought of as a “pooling of features” because we are reducing the depth of the volume, similar to how we reduce the dimensions of height and width with normal maxpooling layers.

AlexNet: 60 million param

VGGNet: 1800 million

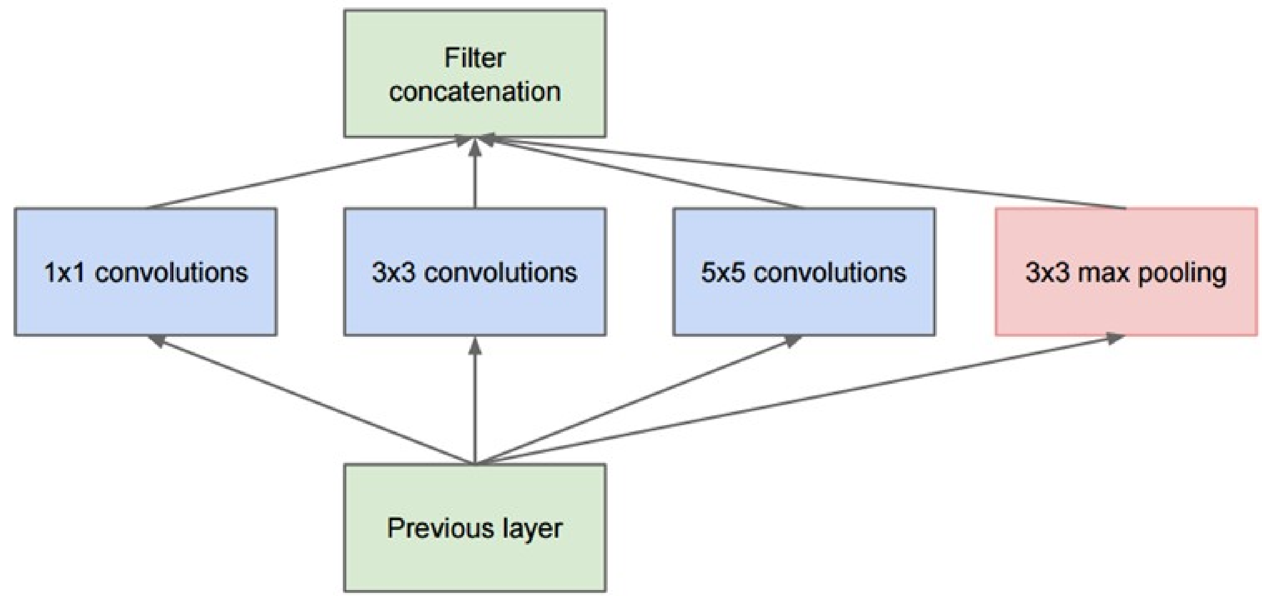
GoogLeNet / Inception-v1



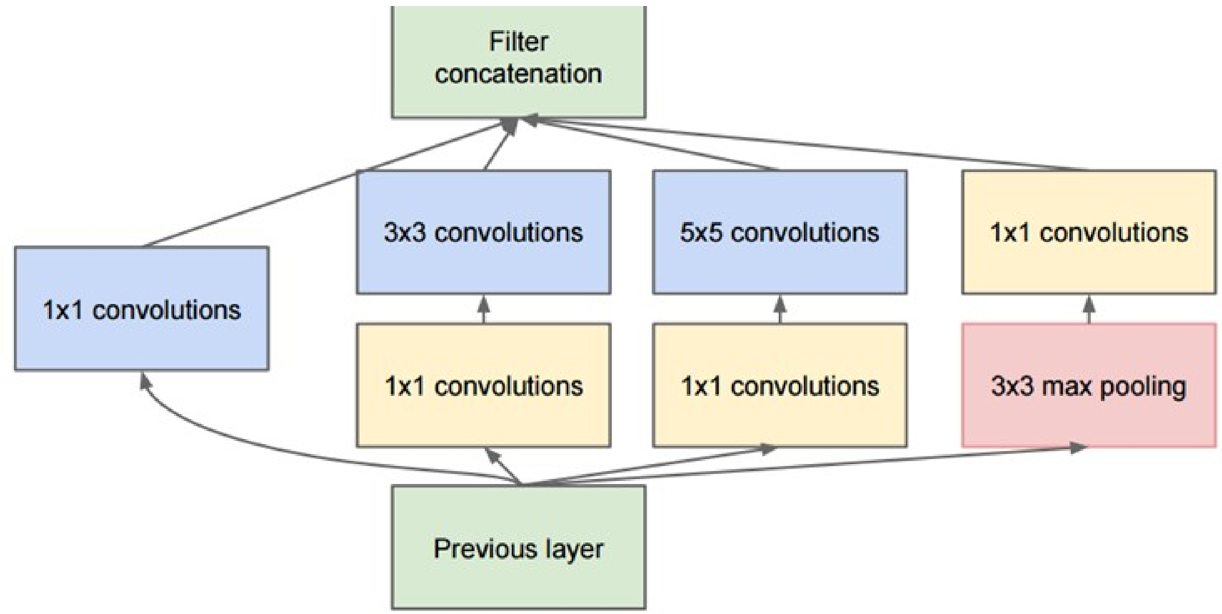


Rethinking Inception

Squeezing the number of channels for each kernel



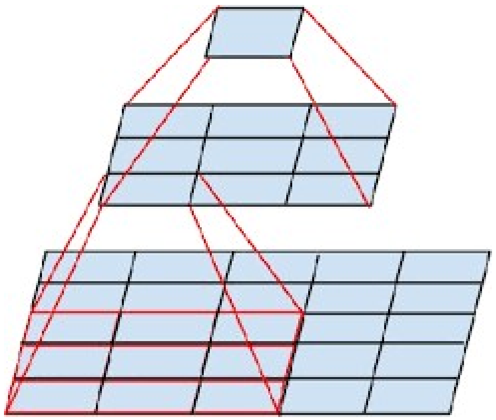
Naïve idea of an Inception module



Full Inception module

Rethinking Inception

Larger convolutions were substituted by series of 3x3 convolutions



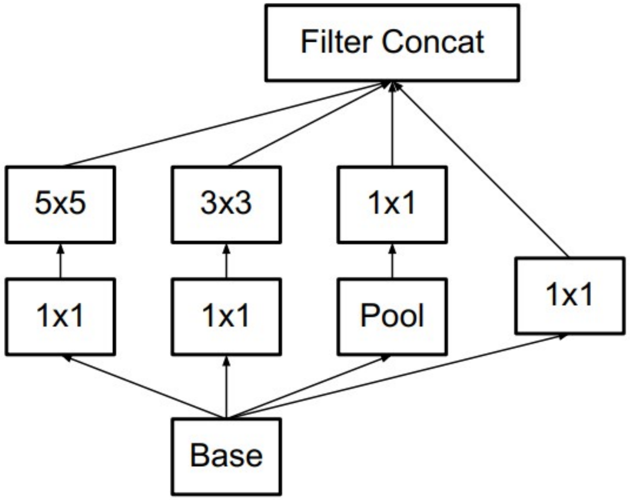
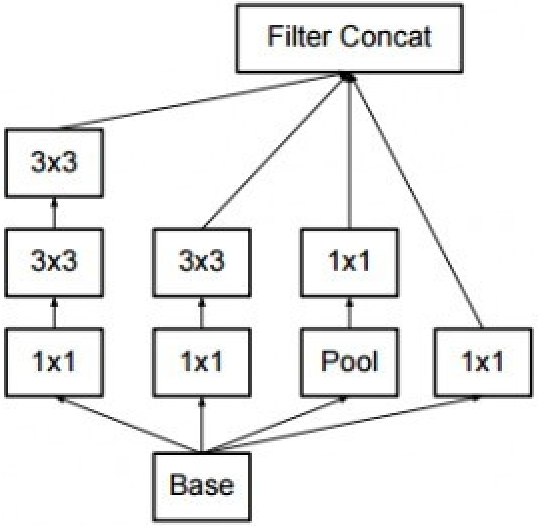


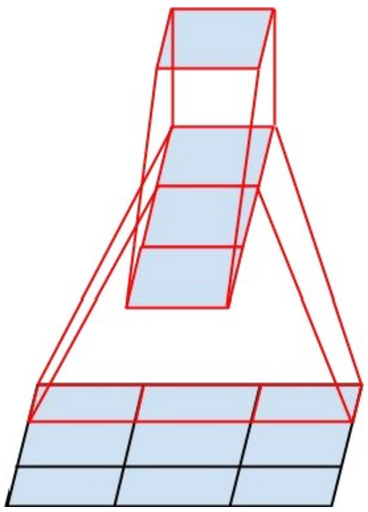
Figure 4. Original Inception module as described in [20].



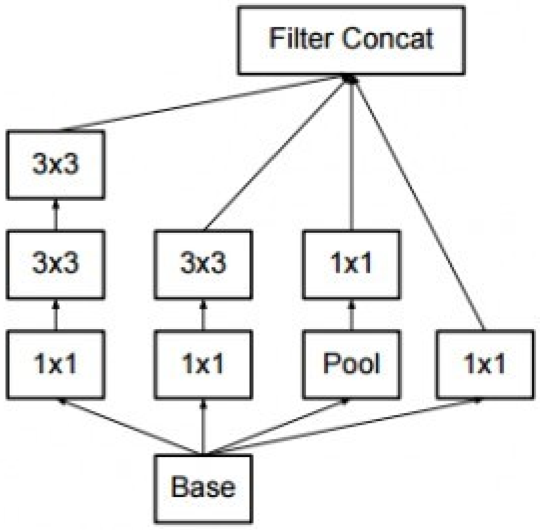
Rethinking Inception

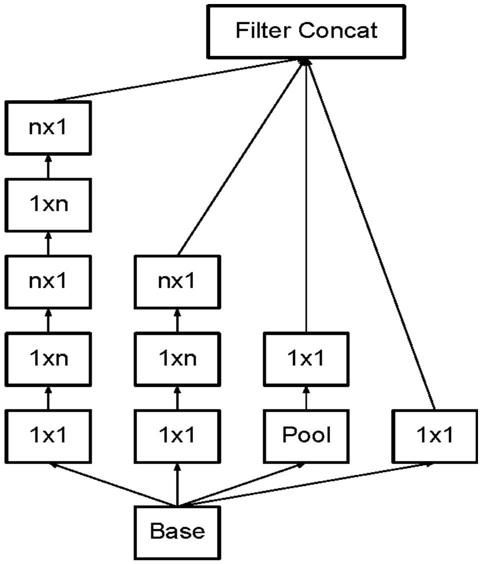
Larger convolutions were substituted by series of 3x3 convolutions

2D convolution were substituted by two 1D convolutions

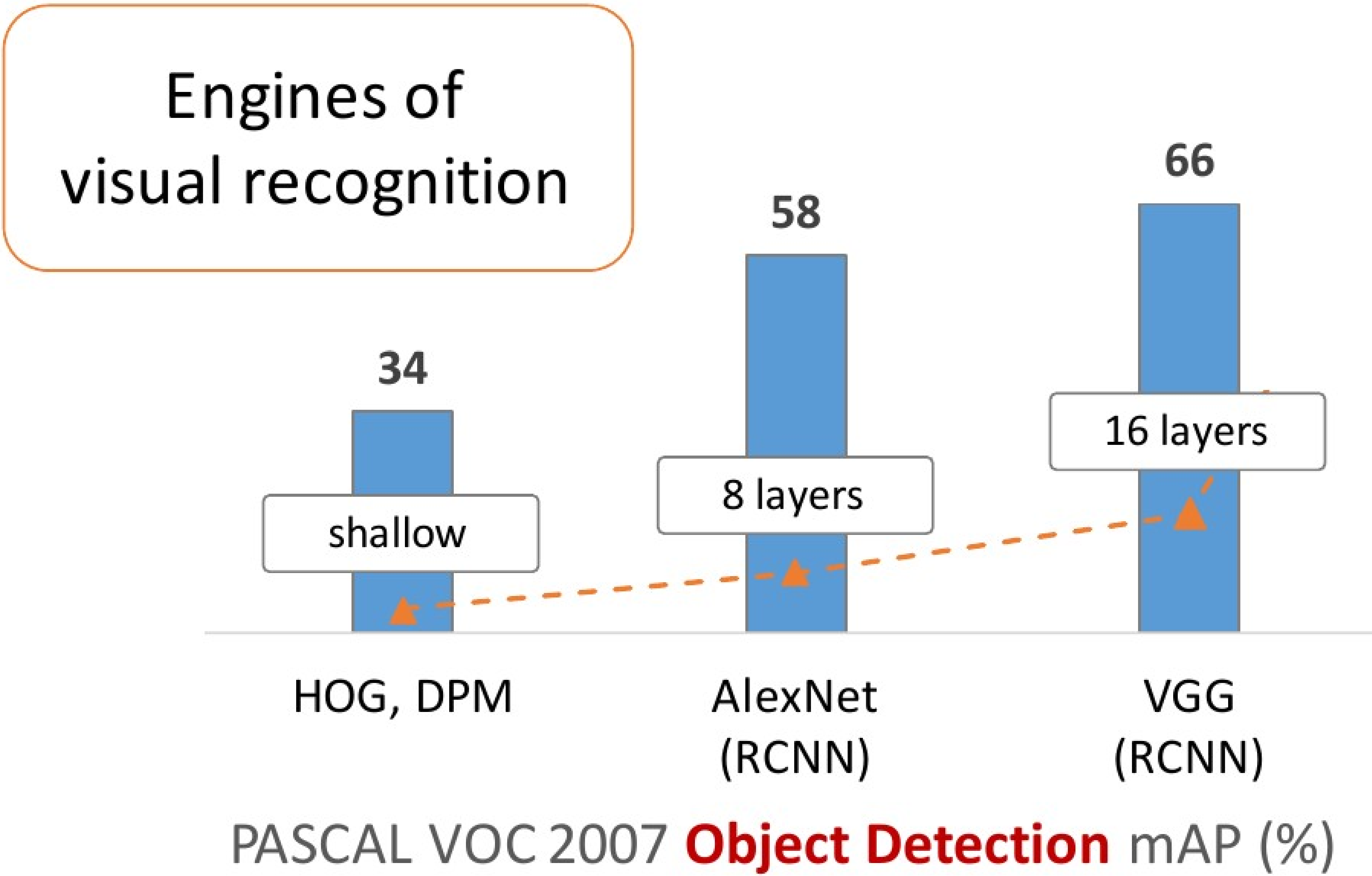


AlexNet: 60 million parameters
VGGNet :180 million parameters
GoogLeNet / Inception-v1: 7 million parameters





Revolution of Depth



How deep could/should a network be?

History of network depth

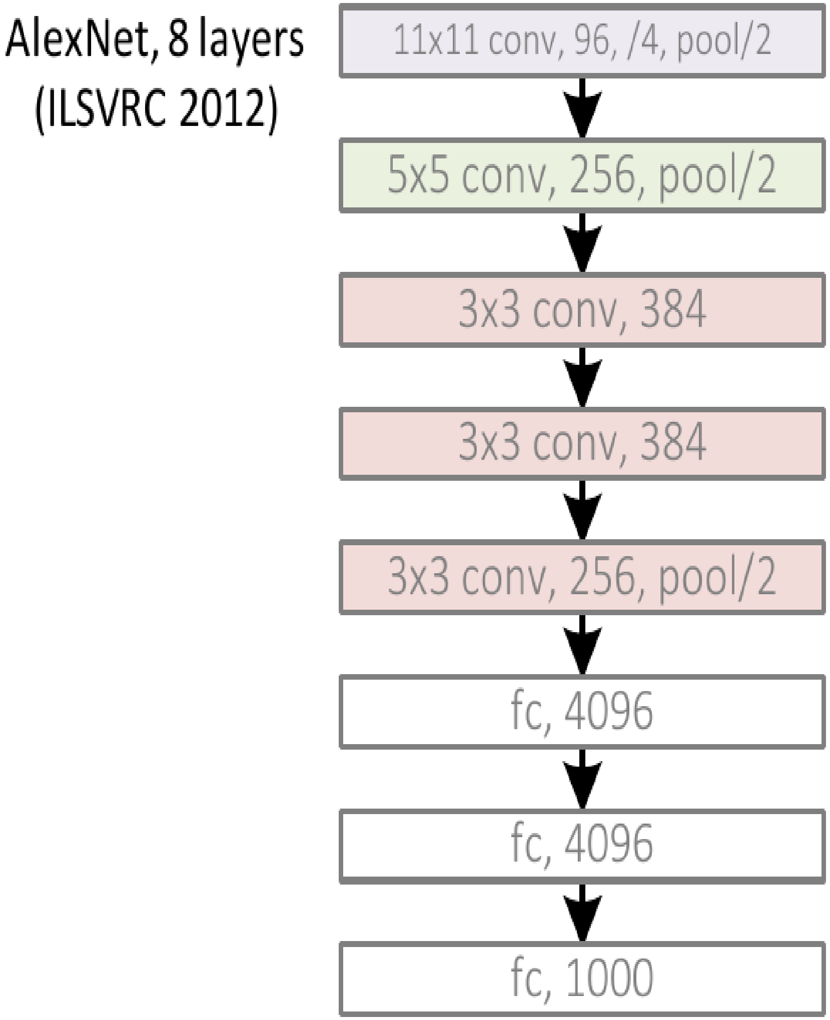
Before 2012: four layers

How deep could/should a network be?

History of network depth

Before 2012: four layers

2012: 8layers



How deep could/should a network be?

History of network depth

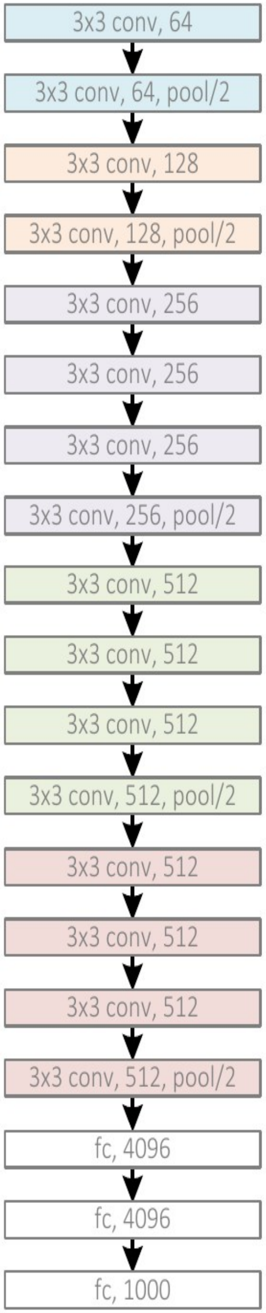
Before 2012: four layer

2012: 8layers

2016: 22 layers

VGG, 19 layers

(ILSVRC 2014)



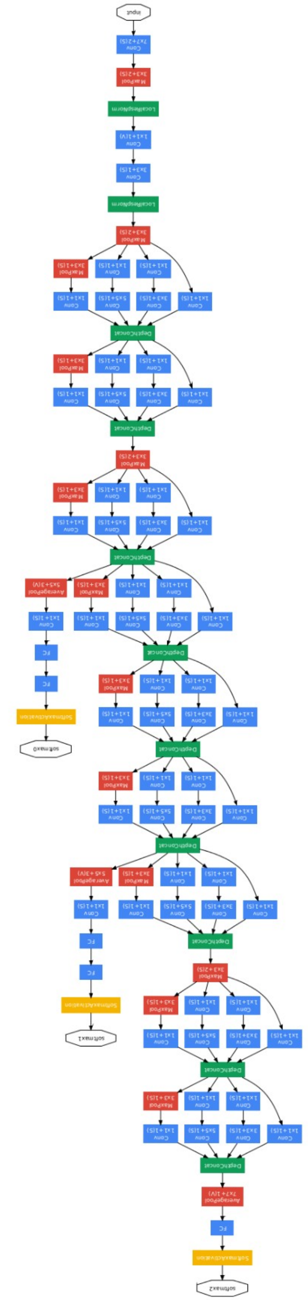
How deep could/should a network be?

History of network depth

Before 2012: four layer

2012: 8layers

2016: 19-22 layers



How deep could/should a network be?

History of network depth

Before 2012: four layer

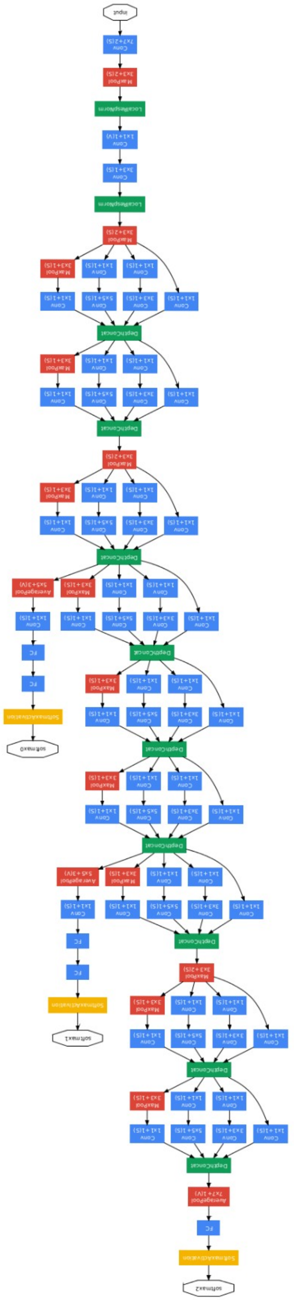
2012: 8layers

2016: 19-22 layers

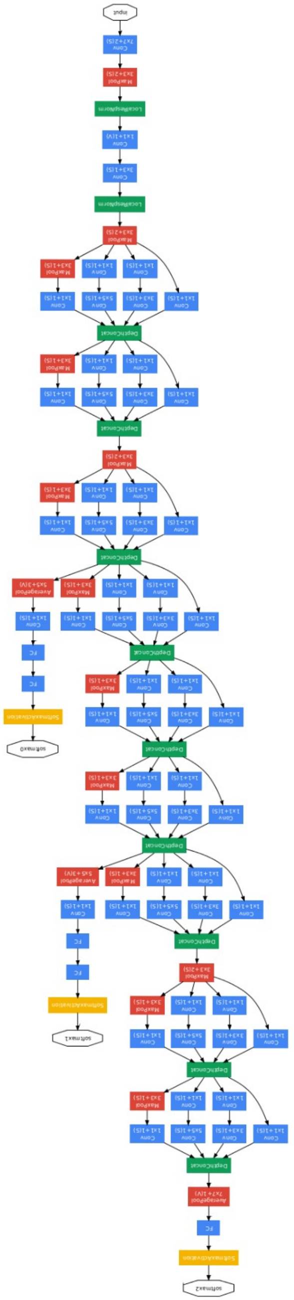
Deeper network:

Possibility to approximate more complex functions

Higher number of parameters



How deep could/should a network be?



History of network depth

Before 2012: four layer

2012: 8layers

2015: 19-22 layers

Deeper network:

Possibility to approximate more complex functions

Higher number of parameters

There are no convolutional networks with more than 30 layers. Why?

The amount of transfered data is decreased from layer to layer

Training becomes difficult

Is a deeper network always better?

A deeper network would have higher approximation power

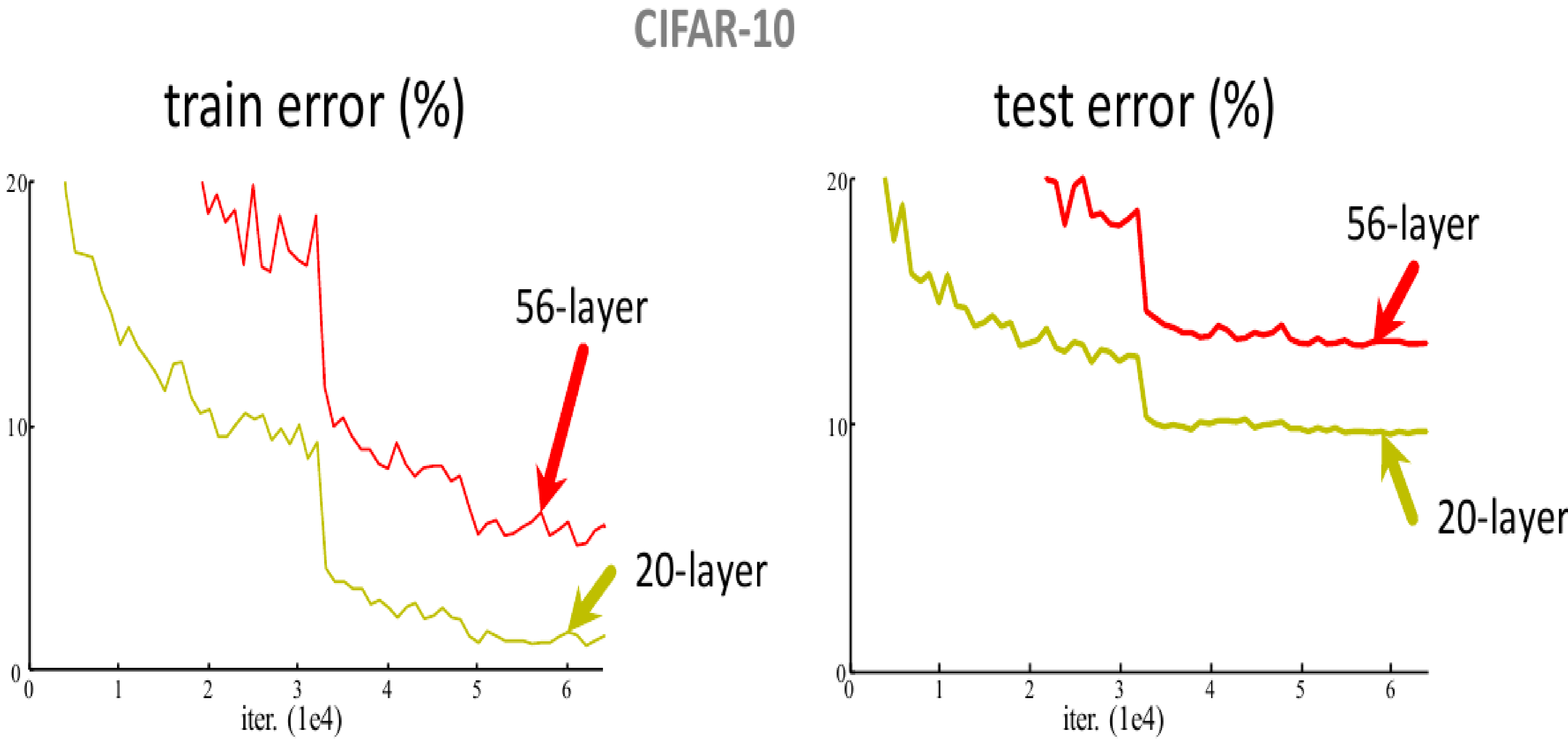
Higher number of parameters (both advantageous and disadvantageous)

Difficult to train the network

Is a deeper network always better?

A deeper network always has the potential to perform better, but training becomes difficult

After a given depth, the same network with the same training on the same data, usually performs worse

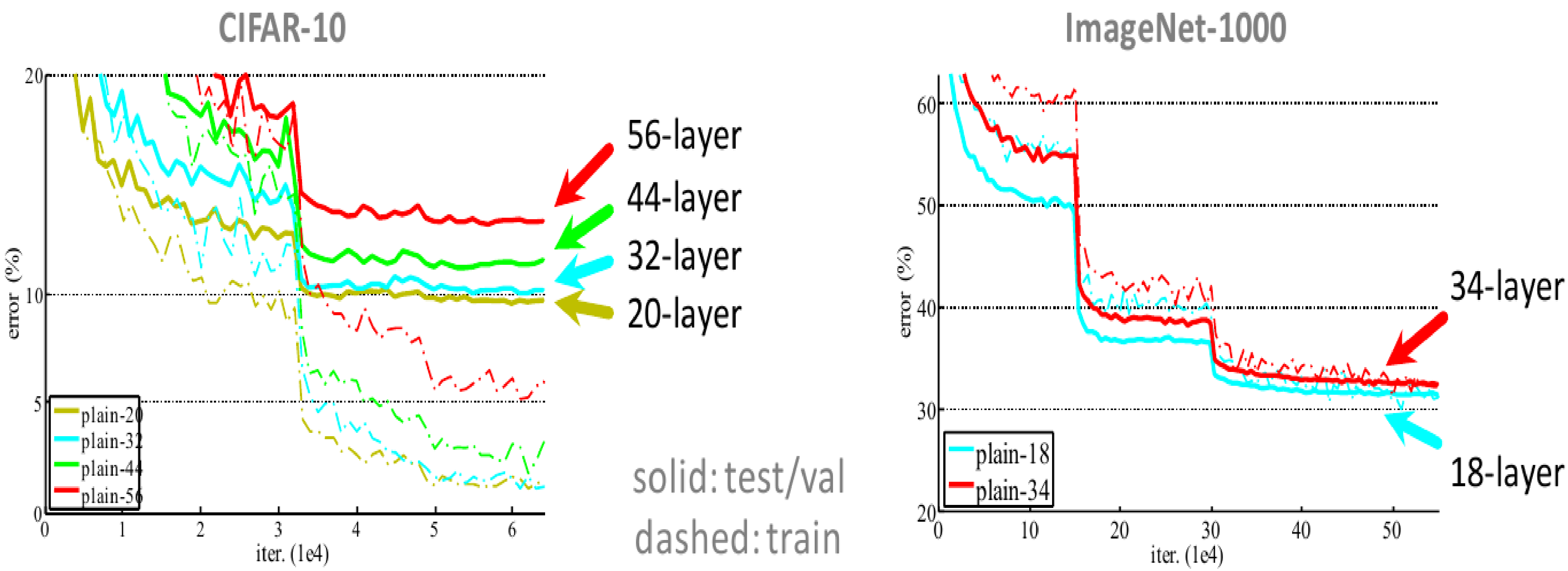


Is a deeper network always better?

A deeper network always have the potential to perform better, but training becomes difficult

We can not just simply stack convolutional layers to increase accuracy

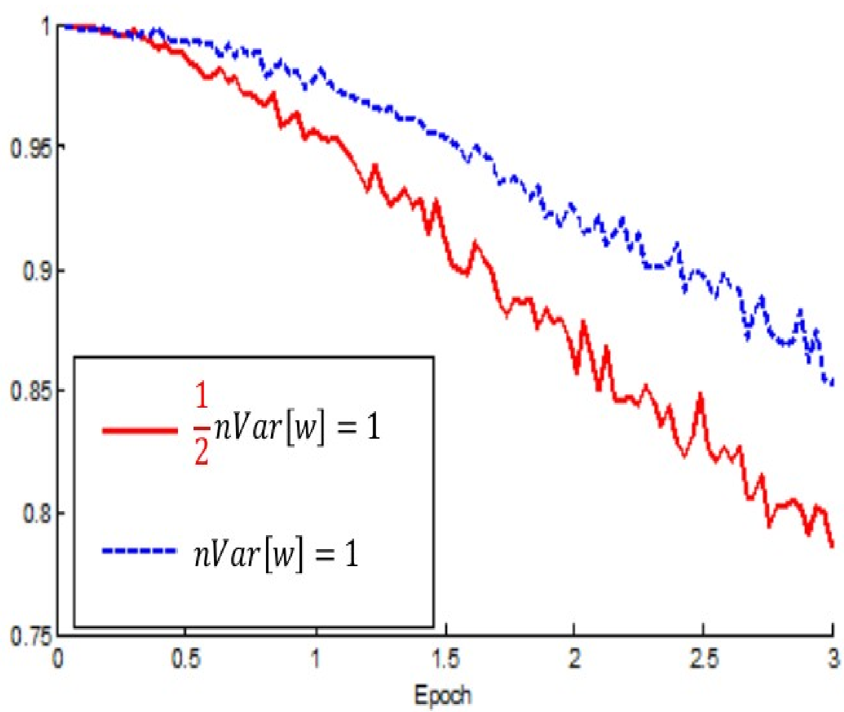
Example: stacking 3x3 convolution layers



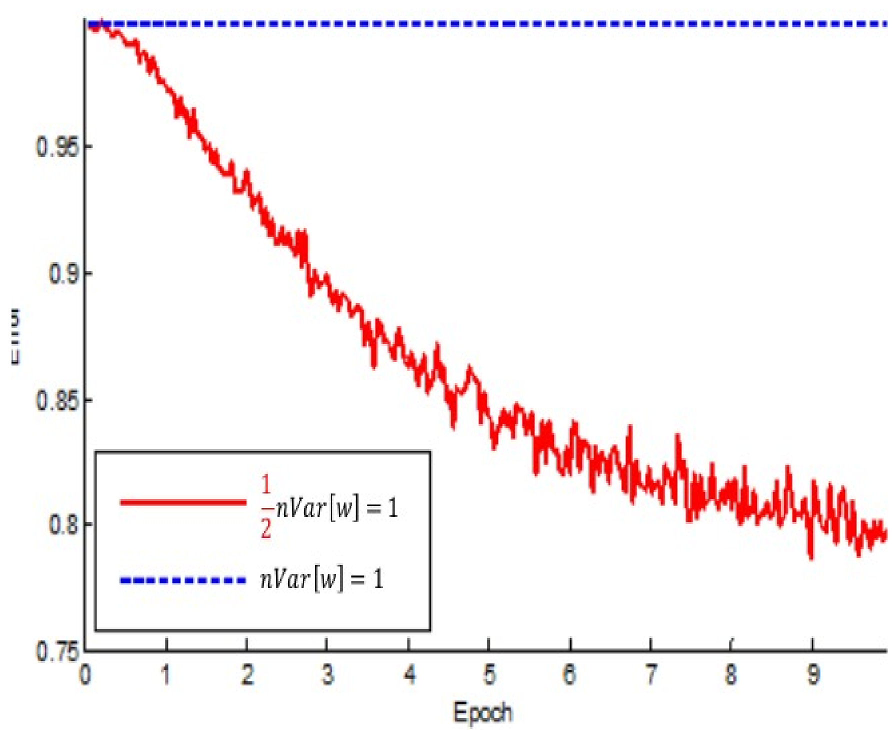
Is a deeper network always better?

A deeper network always have the potential to perform better, but training becomes difficult

22-layer ReLU net:
good init converges faster



30-layer ReLU net:
good init is able to converge



Is a deeper network always better?

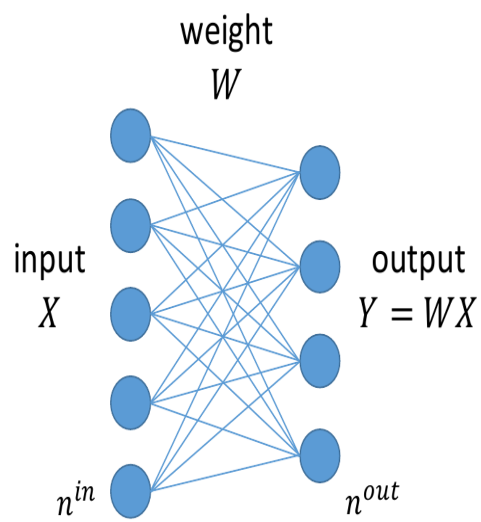
A deeper network would have higher approximation power

Higher number of parameters (both advantageous and disadvantageous)

Difficult to train the network:

Vanishing gradients

Initialization



If:

- Linear activation
- x, y, w: independent

Then:

1-layer:

$$Var[y] = (n^{in} Var[w]) Var[x]$$

Multi-layer:

$$Var[y] = (\prod_{d} n_d^{in} Var[w_d]) Var[x]$$

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

Is a deeper network always better?

A deeper network would have higher approximation power

Higher number of parameters (both advantageous and disadvantageous)

Difficult to train the network:

Vanishing gradients

The number of layers has an exponential impact

Initialization

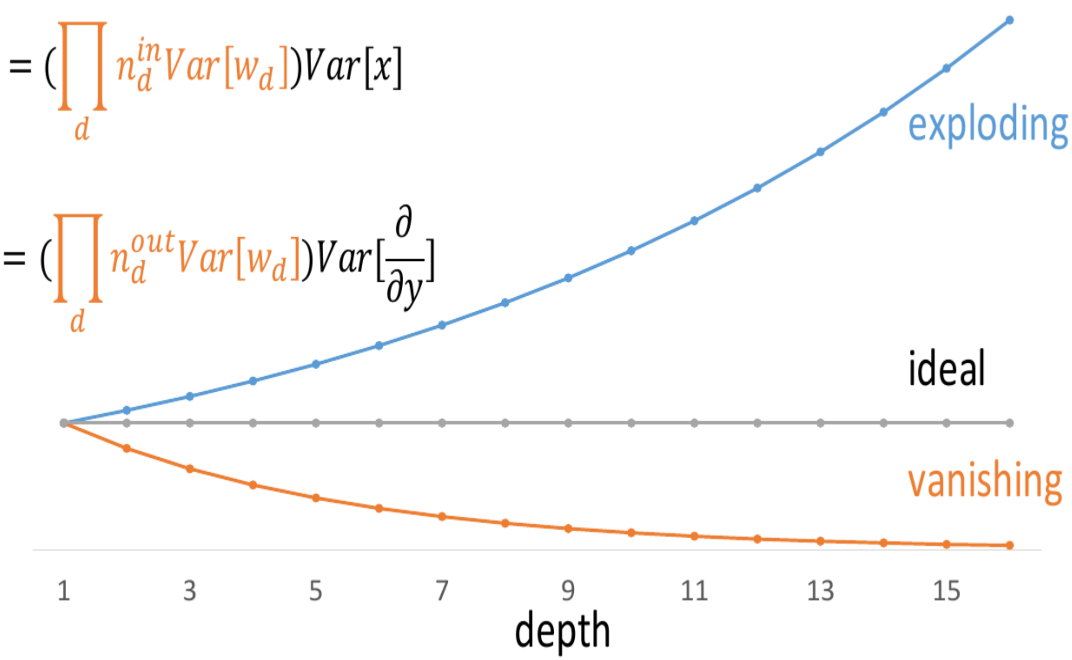
Both forward (response) and backward (gradient) signal can vanish/explode

Forward:

$$Var[y] = \left(\prod_d n_d^{in} Var[w_d]\right) Var[x]$$

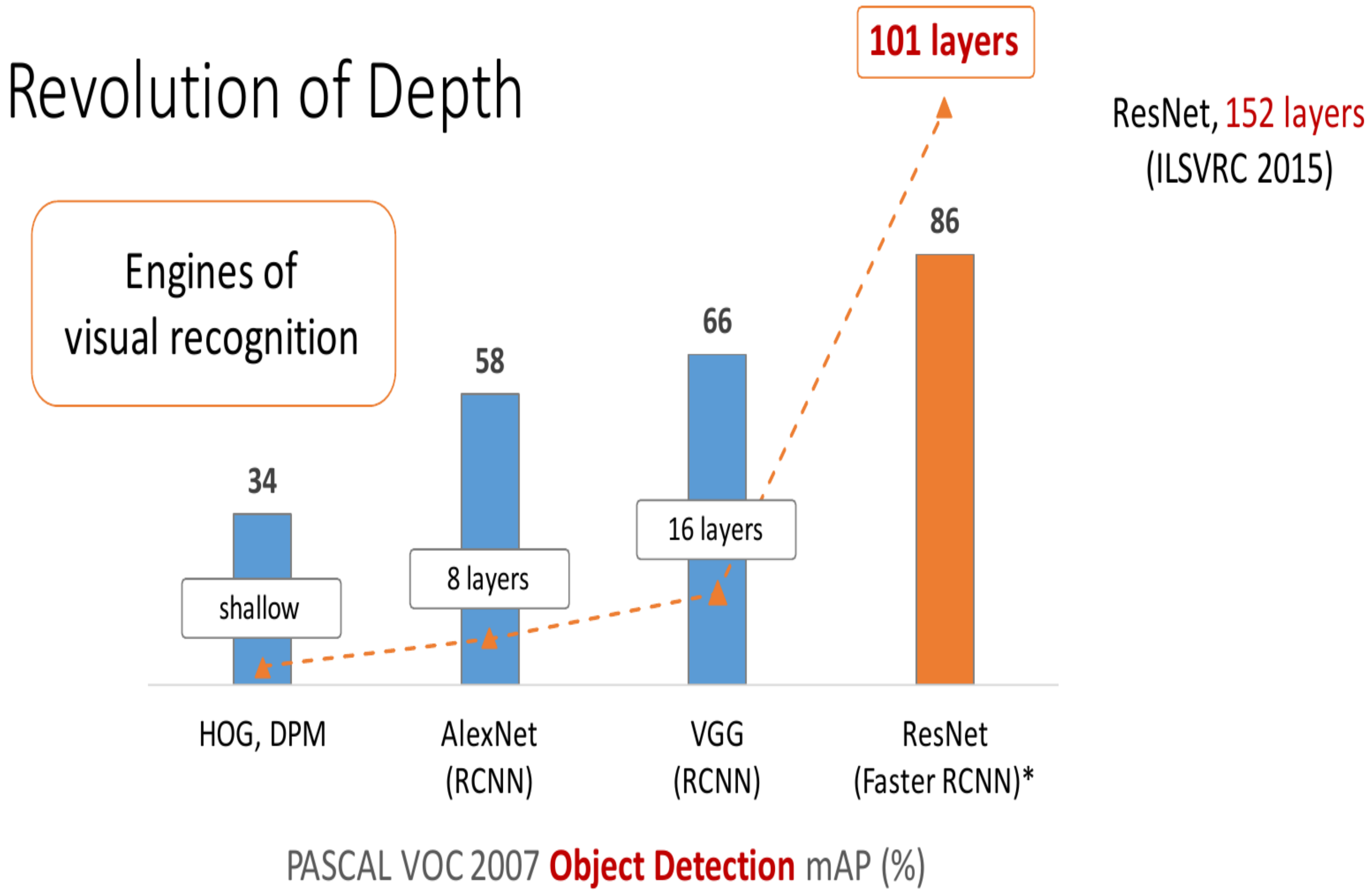
Backward:

$$Var\left[\frac{\partial}{\partial x}\right] = \left(\prod_d n_d^{out} Var[w_d]\right) Var\left[\frac{\partial}{\partial y}\right]$$



How deep could a network be?

Residual networks provide an answer to these questions

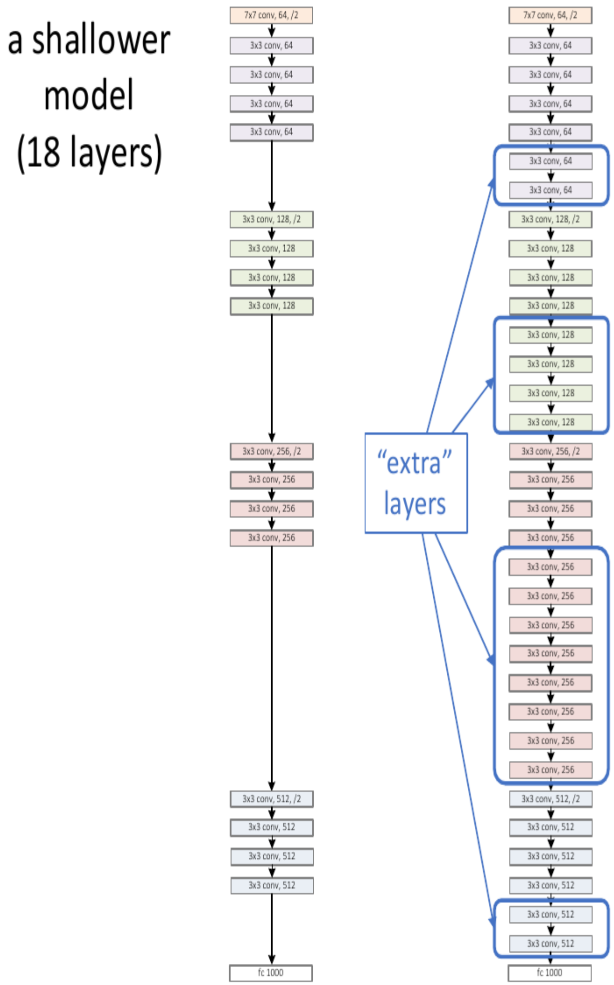


How could we create deeper networks?

A deeper network always have the potential to perform better, but training becomes difficult

How could we ensure that additional layers will not decrease accuracy (might even increase it)?

Let’s start with a shallow model (18 layers) and add some extra layers (which we hope could increase accuracy)

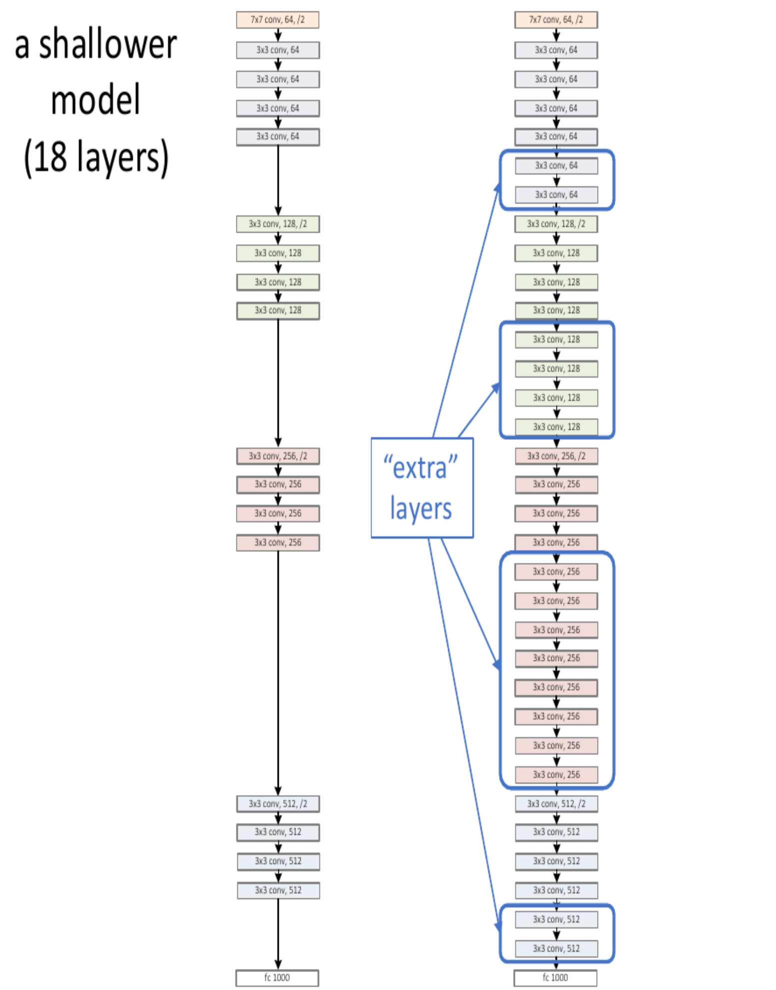


How could we create deeper networks?

A deeper network always have the potential to perform better, but training becomes difficult

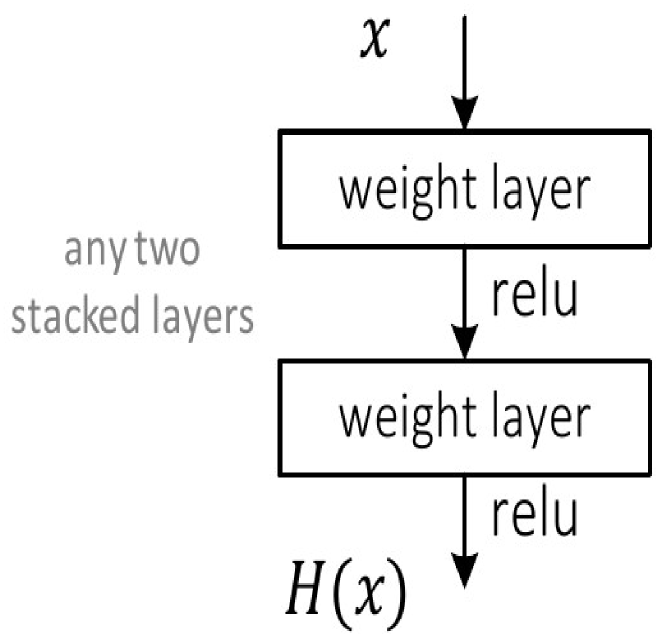
How could we ensure that additional layers will not decrease accuracy (might even increase it)?

Let’s start with a shallow model (18 layers) and add some extra layers (which we hope could increase accuracy)



Our aim is to add “useful” operations H(x)

The problem is that H(x) can ruin our accuracy because vanishing gradients, overfit - extra parameters



How could we create deeper networks?

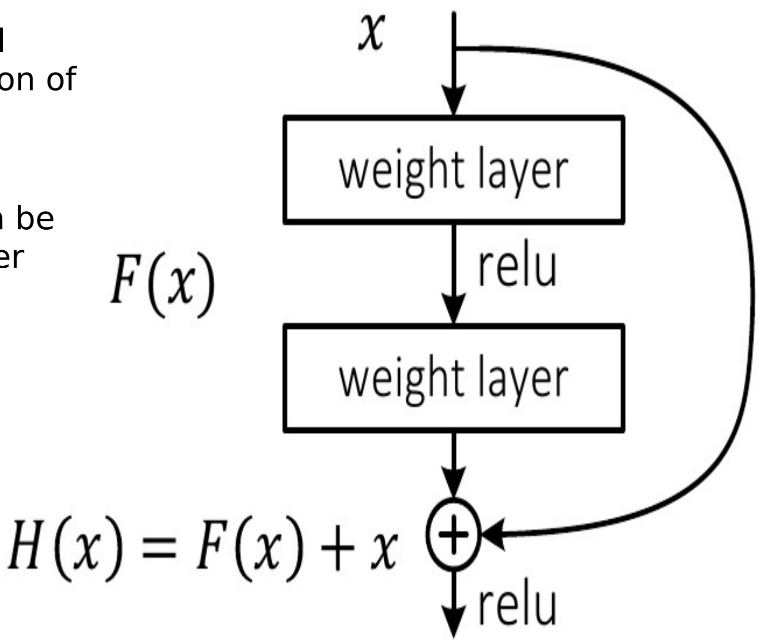
A deeper network always have the potential to perform better, but training becomes difficult

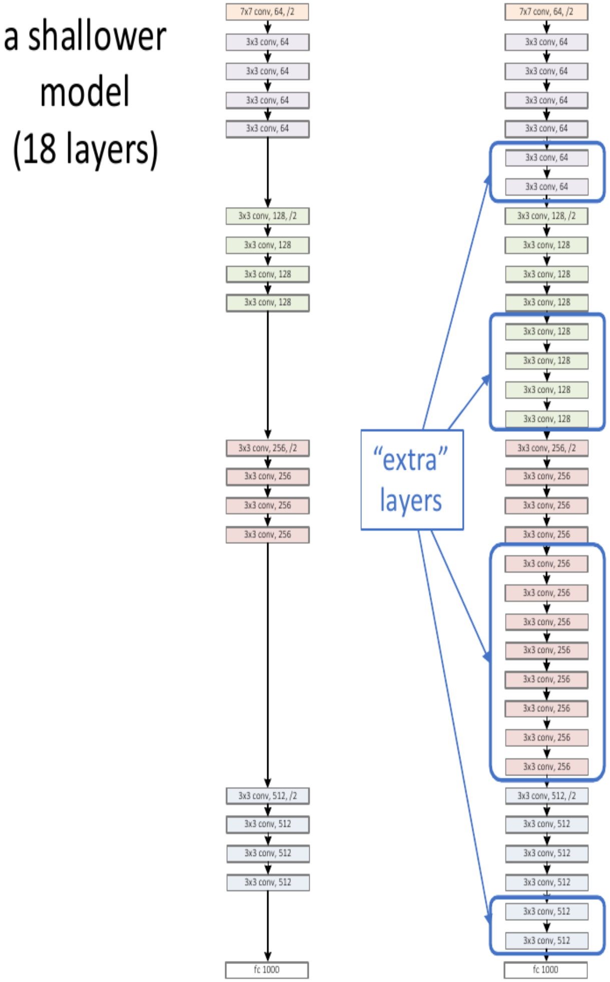
How could we ensure that additional layers will not decrease accuracy (might even increase it)?

The trick is to use residual connection and as a starting point F(x) could be zero, and H(x) becomes the dientity mapping

So H(X) will not change our performance, gradients will remain, because the addition of x

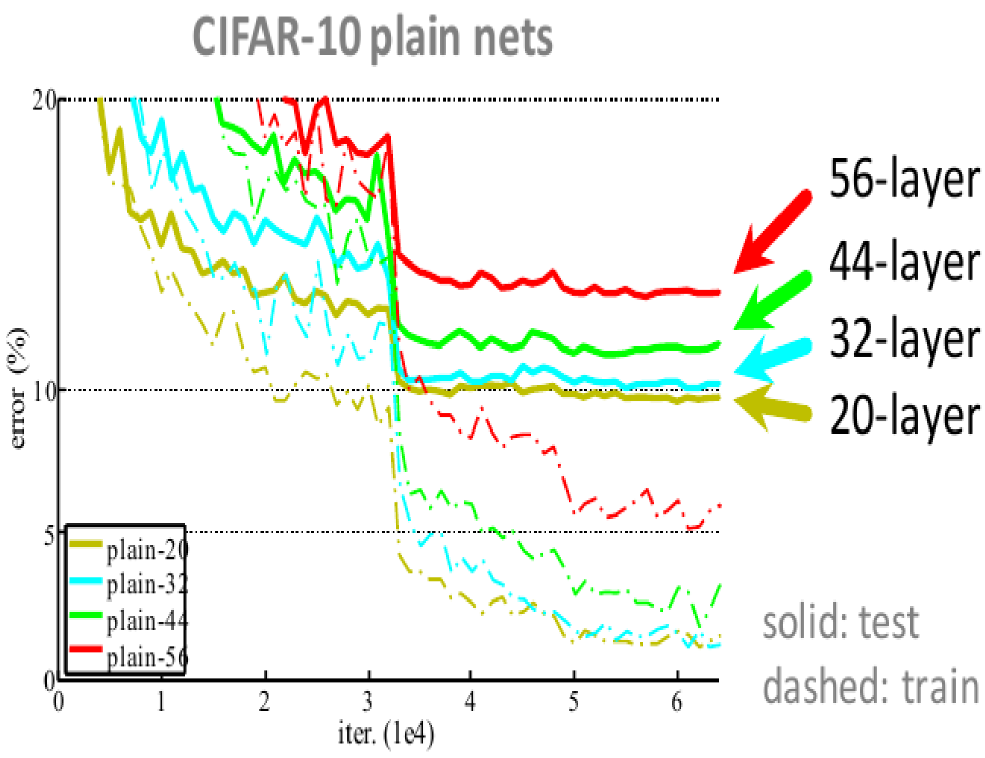
Our accuracy will no be decreased, and might even be increased if we find a proper F(x)

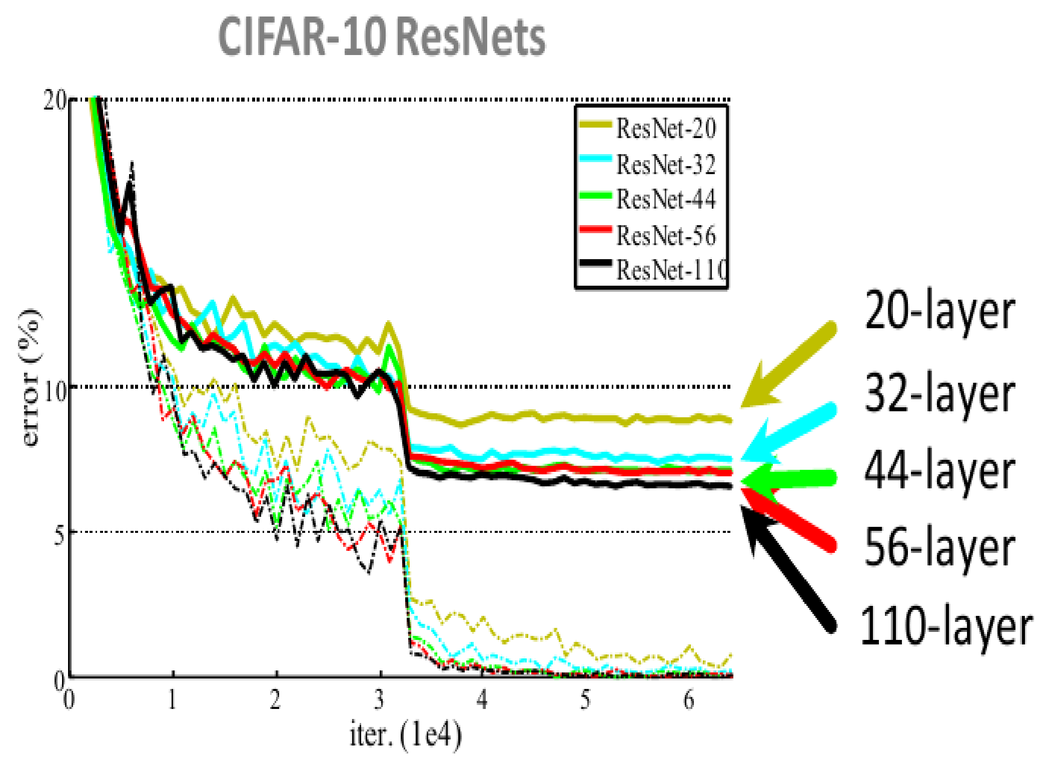




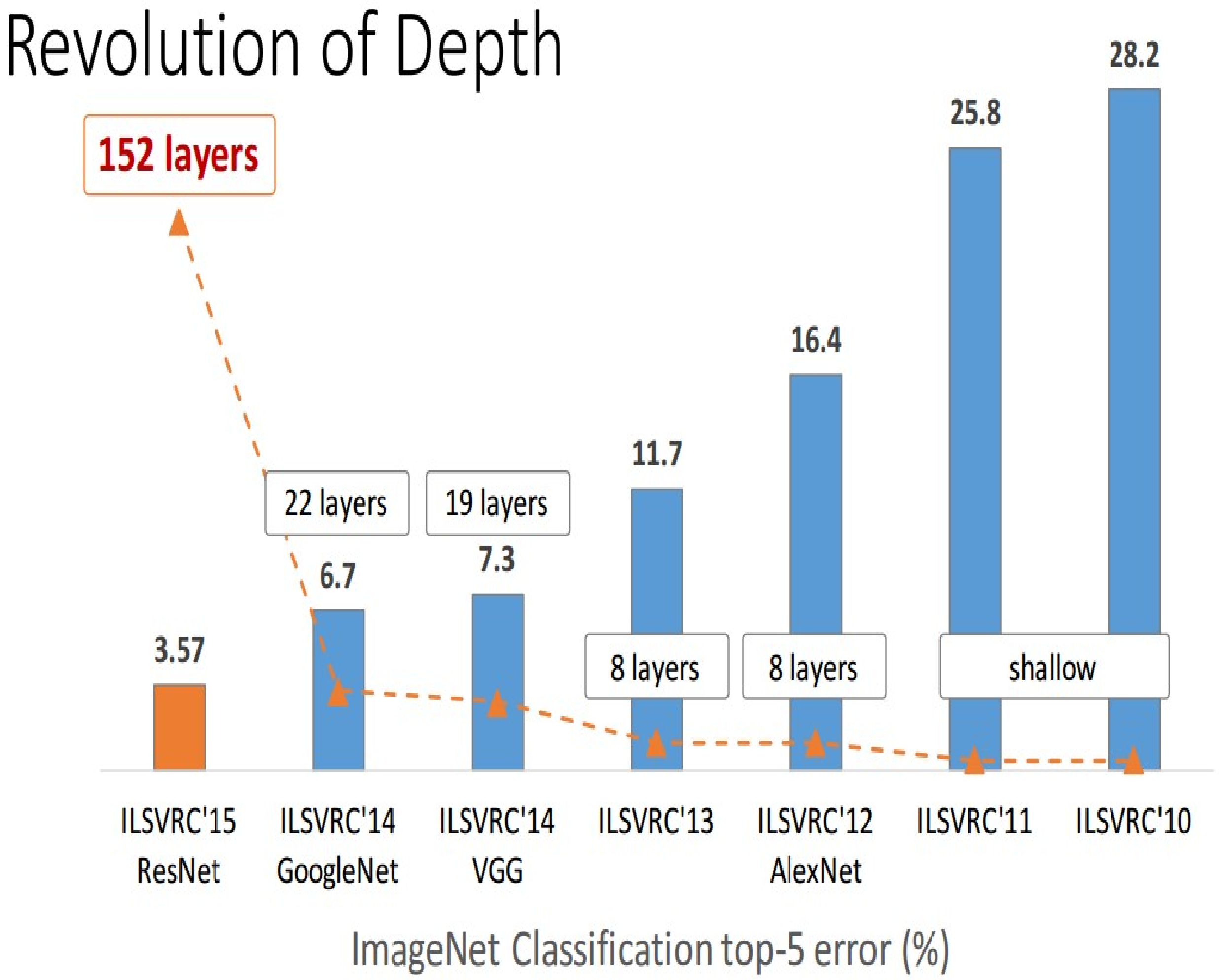
Residual networks

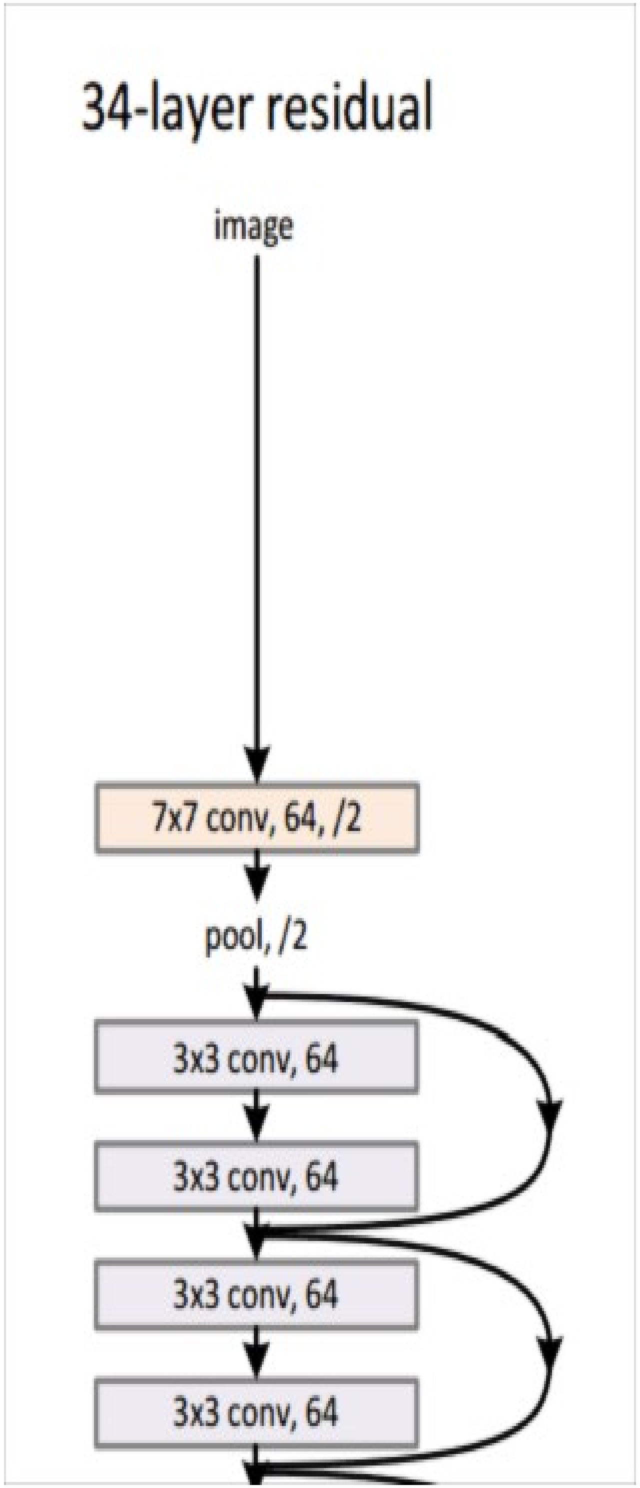
Results: Deeper residual networks result higher accuracy





Results with ResNets





Results with ResNets

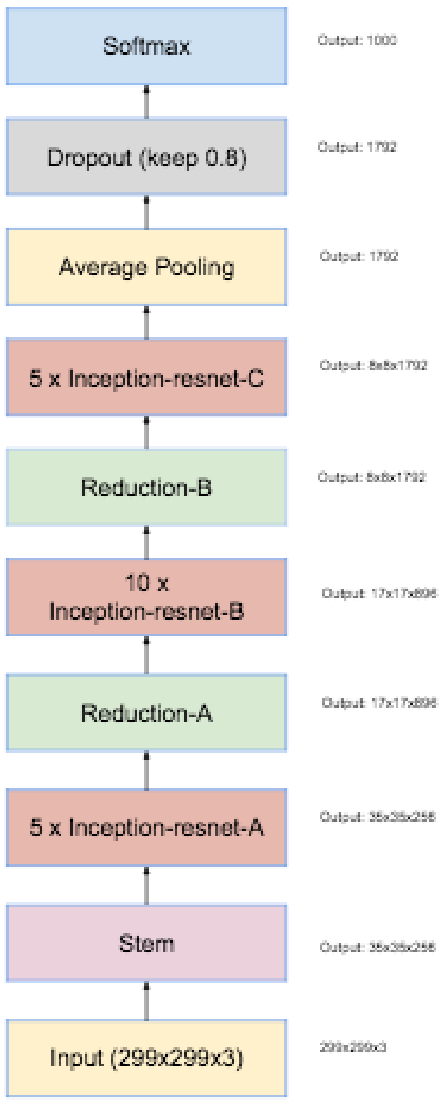
ResNets had the lowest error rate at most competitions since 2015

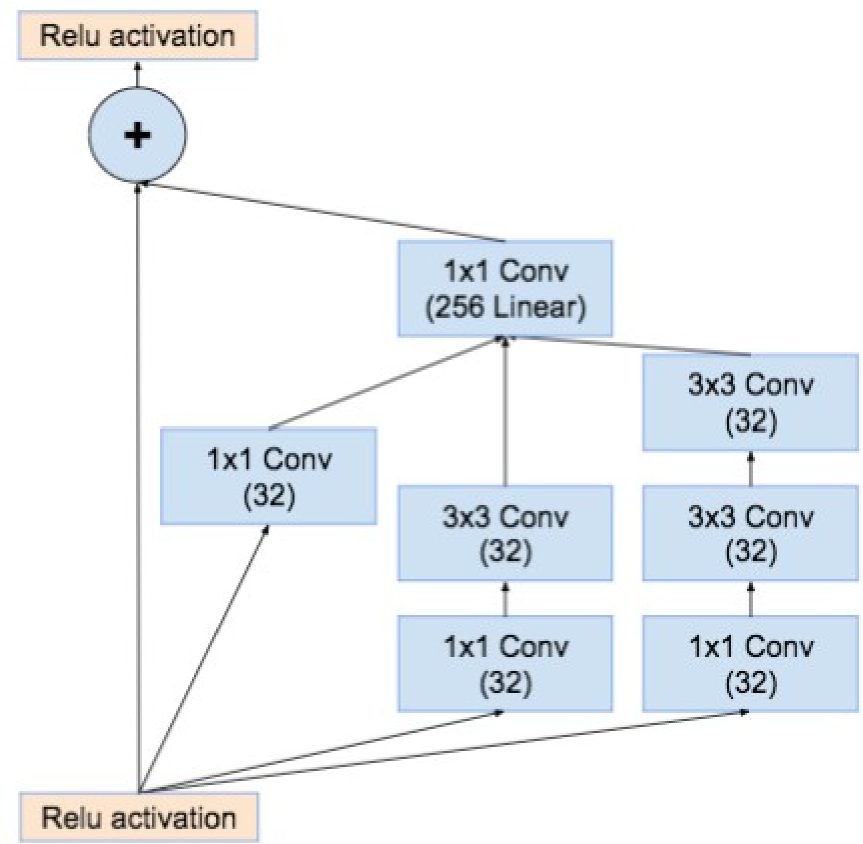
1st places in all five main tracks

- ImageNet Classification: "Ultra-deep" 152-layer nets
- ImageNet Detection: **16%** better than 2nd
- ImageNet Localization: **27%** better than 2nd
- COCO Detection: **11%** better than 2nd
- COCO Segmentation: **12%** better than 2nd

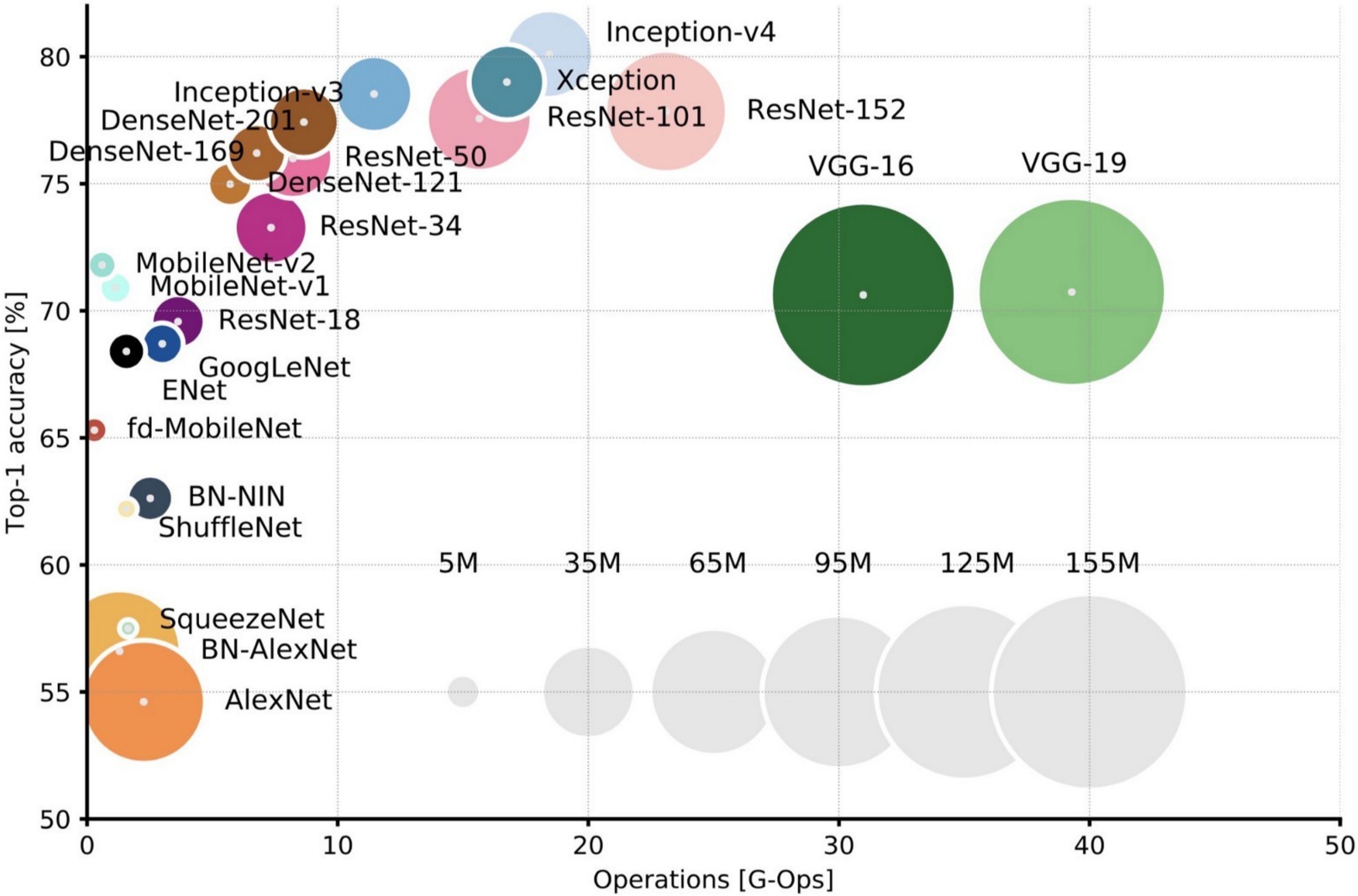
GoogleNet Inception v4

Inception architecture applied to residual networks

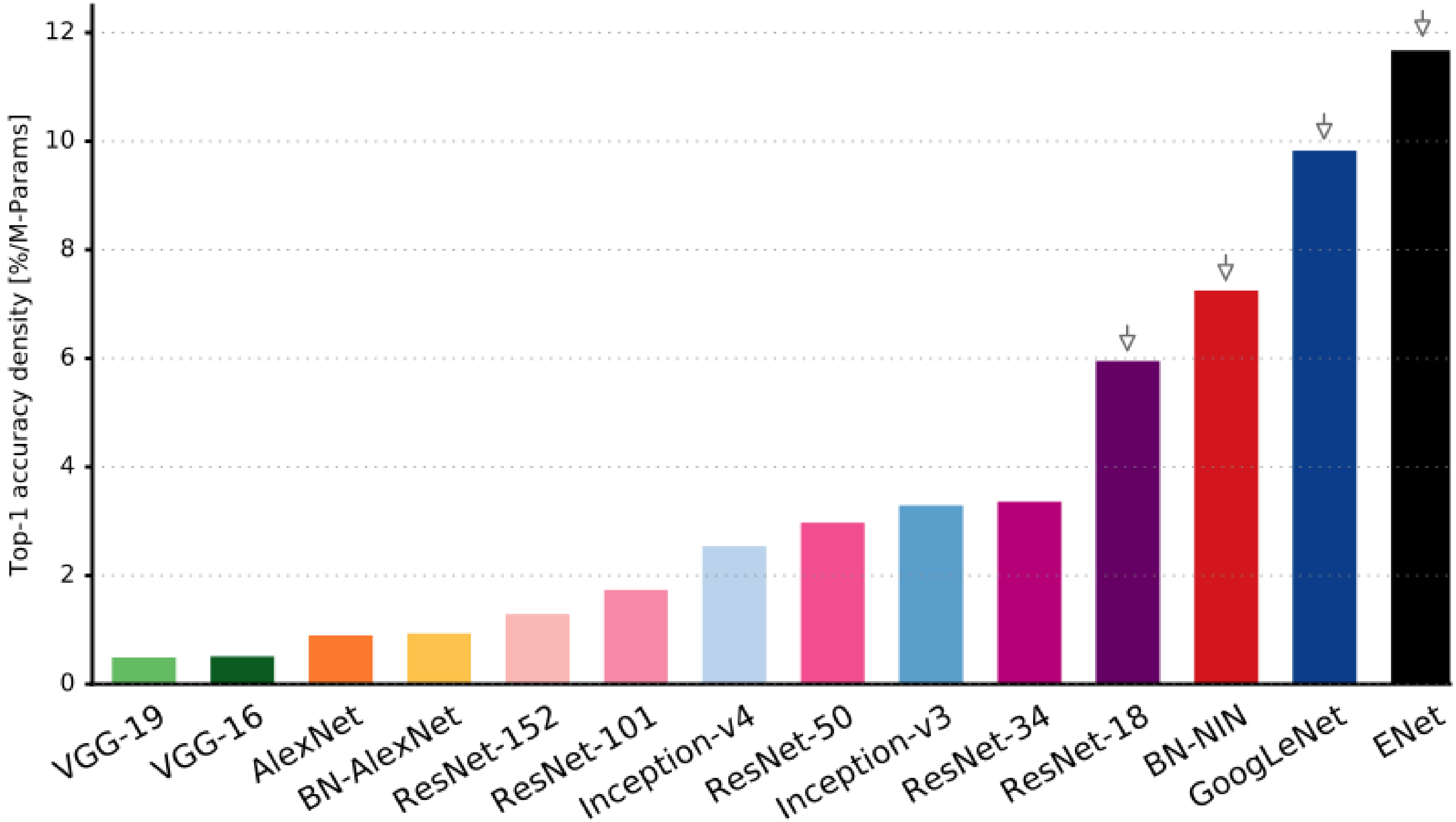




Efficiency of Neural Networks



Efficiency of Neural Networks

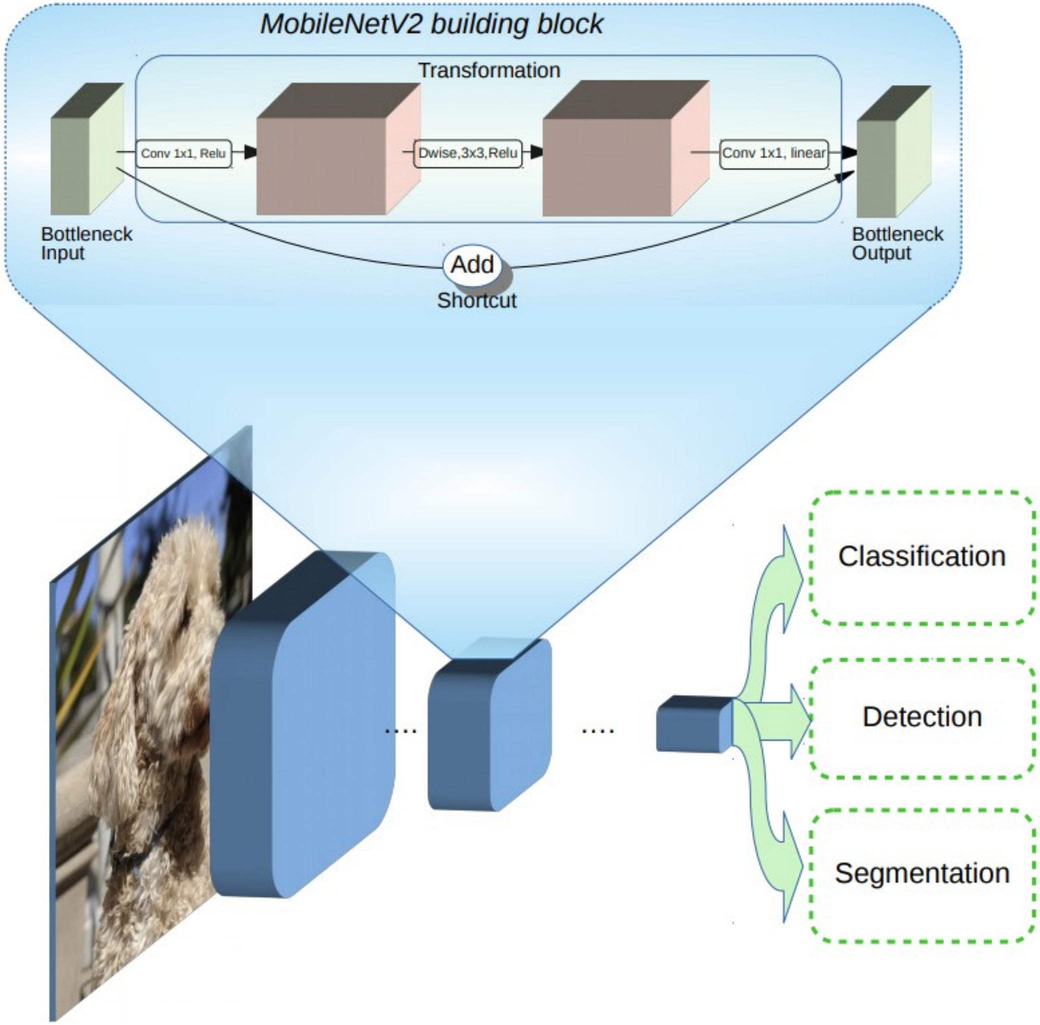


MobileNet

In this arhcitecture depths are squeezed before each operation

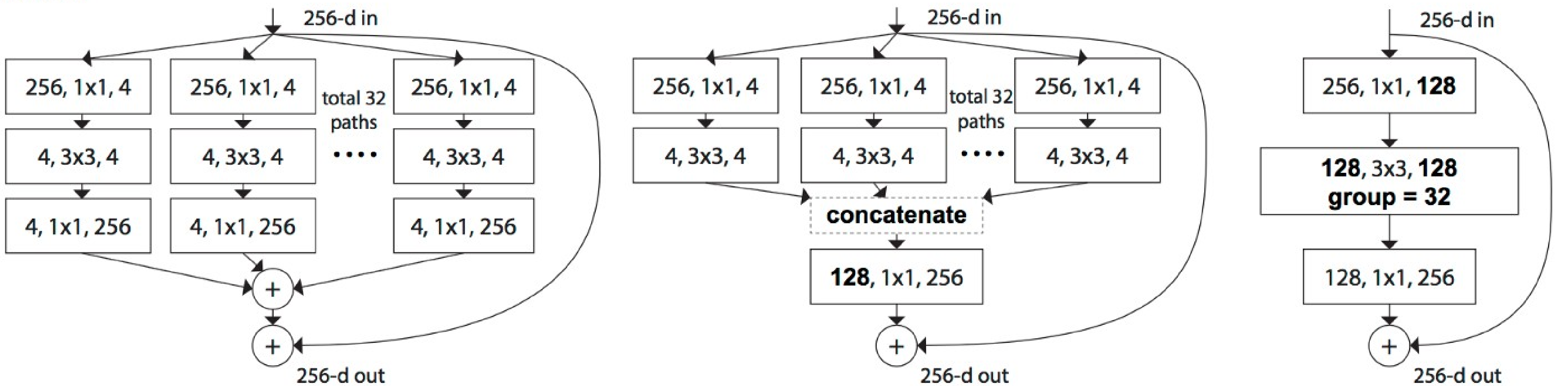
In a squeezed architecture we will use a linear approximation of 128 feature maps, using 16 independent feature maps And expanded by 1x1 convolution

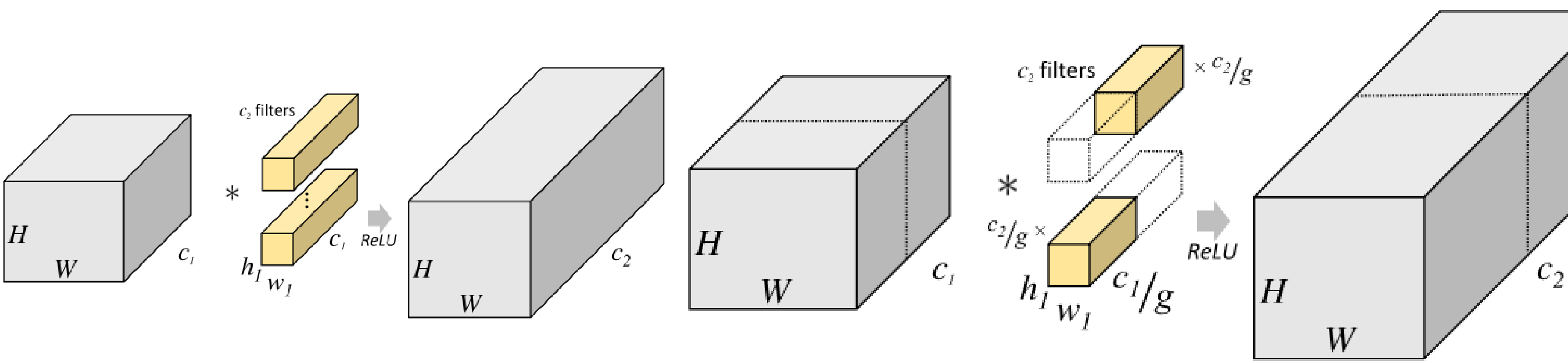
From the linear combination of these elements the new maps are created



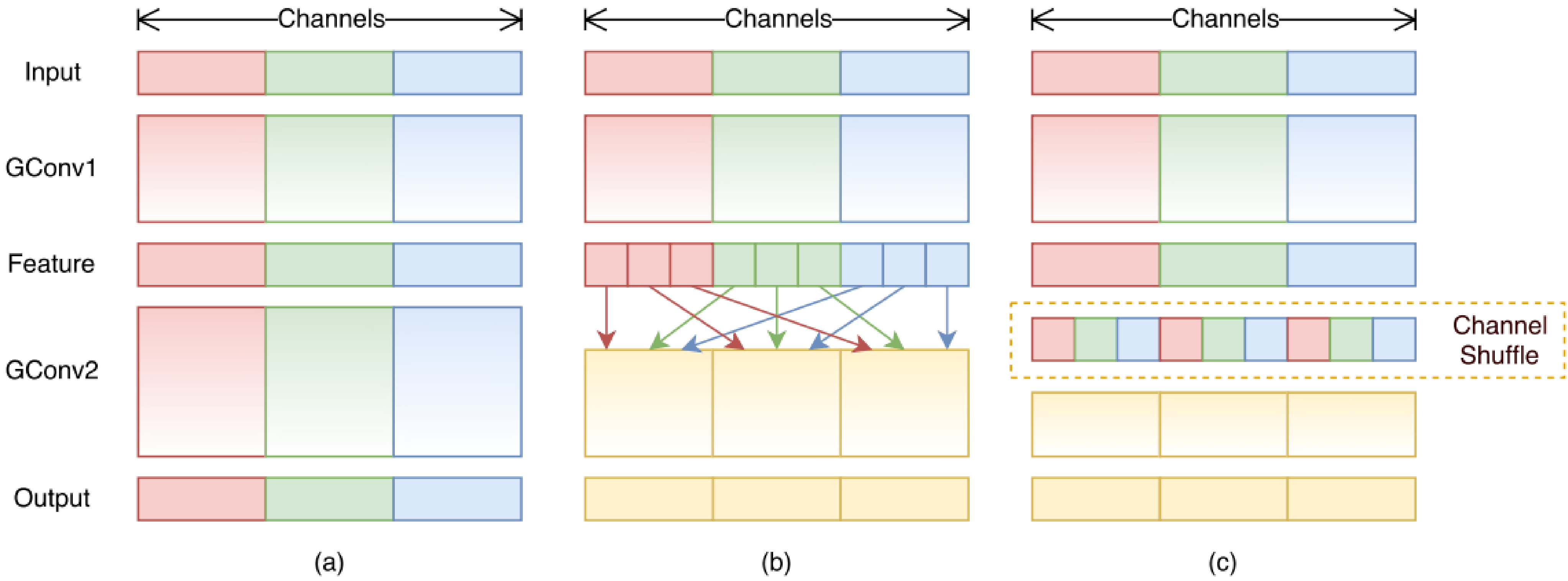
ResNext

equivalent



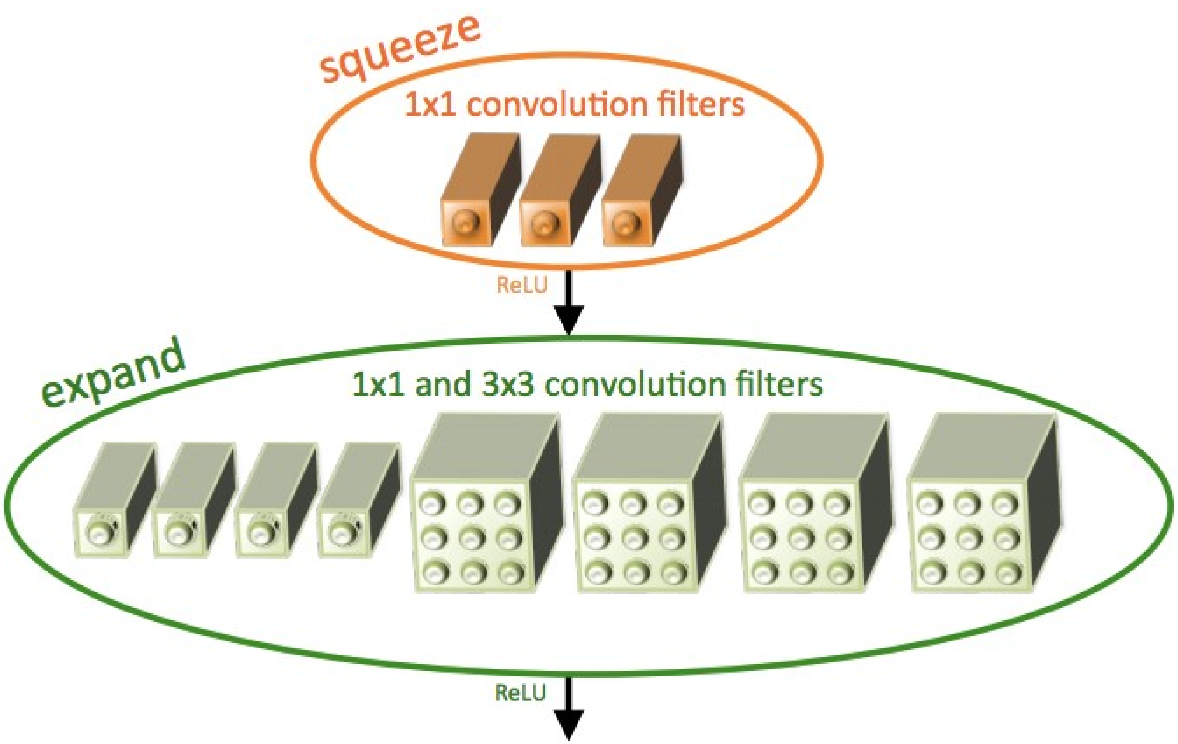


ShuffleNet


Channels
Channels
Channels
Input
GConv1
Feature
GConv2
Output
Channel Shuffle
(a)
(b)
(c)

SqueezeNet

In this arhcitecture depths are squeezed before each operation



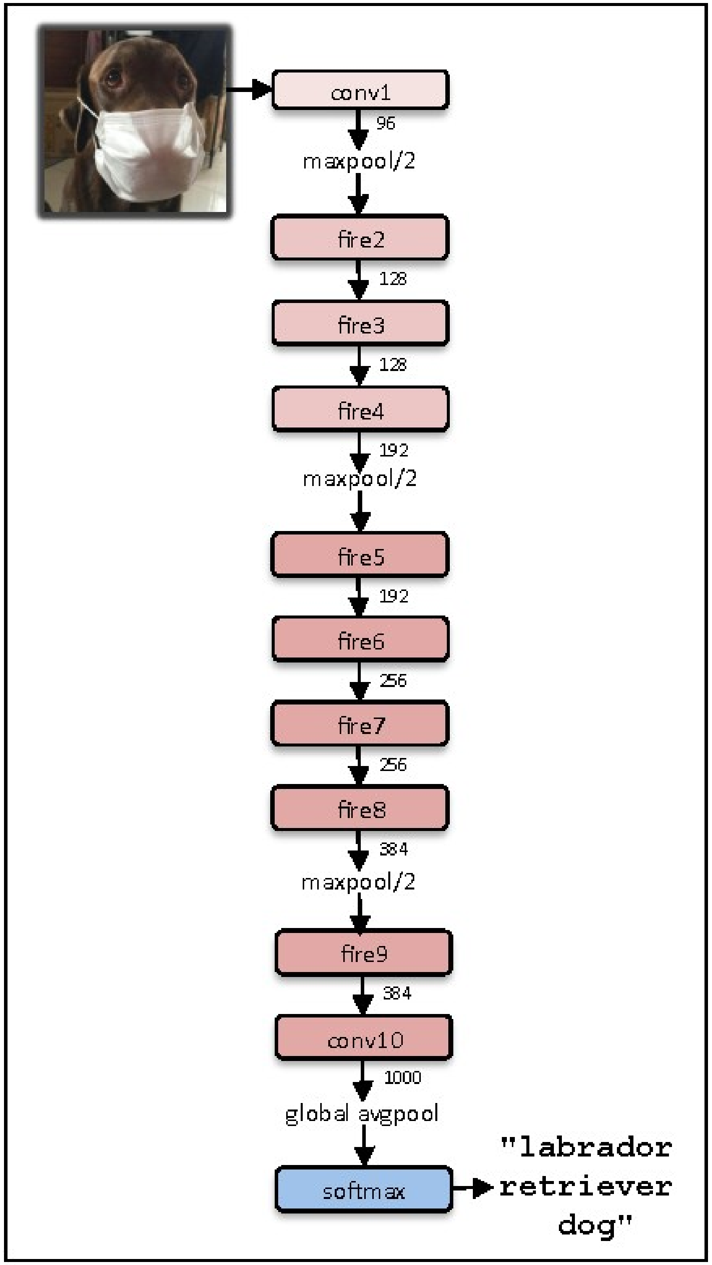
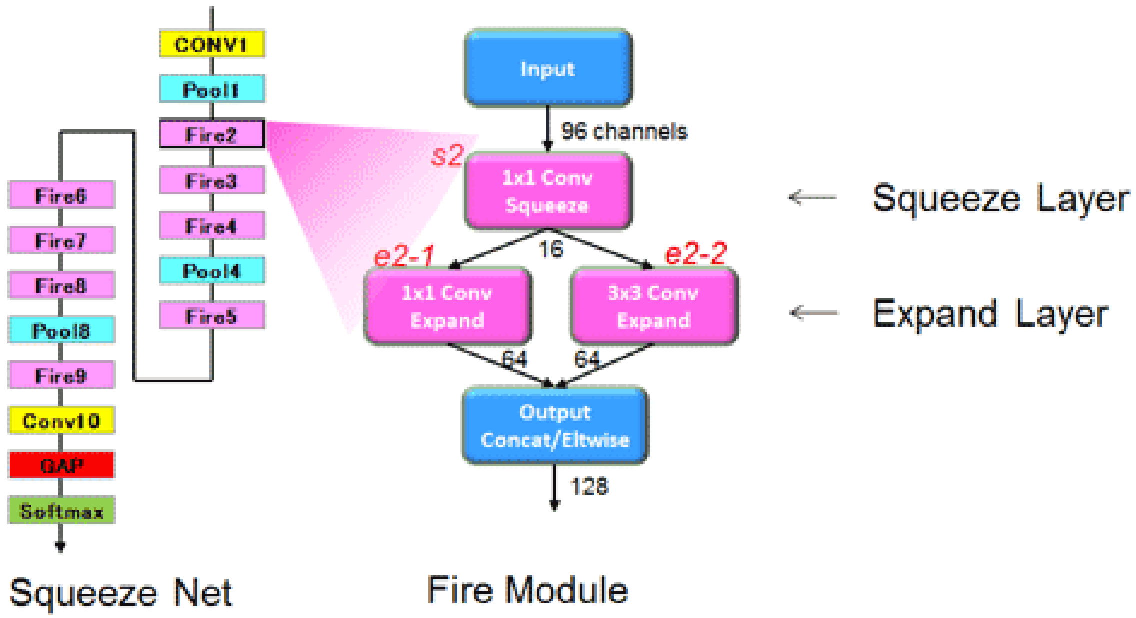


Figure 2. The SqueezeNet architecture

SqueezeNet

In this arhcitecture depths are squeezed before each operation



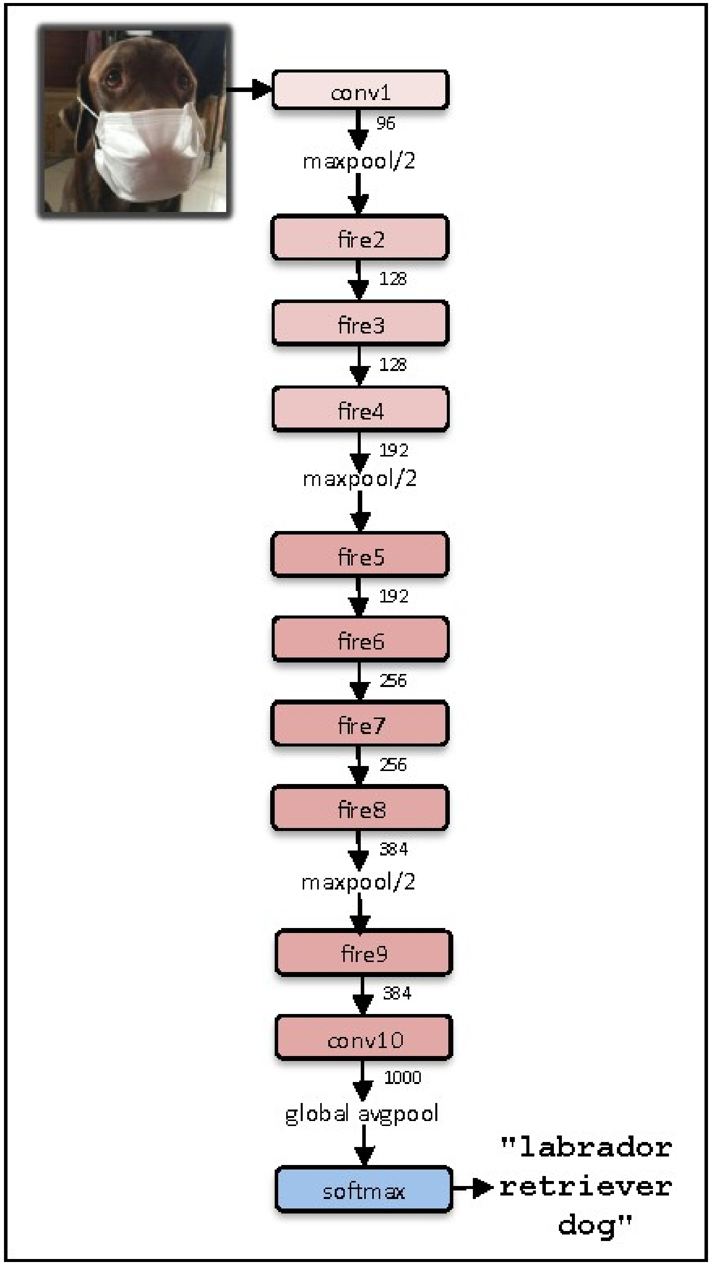


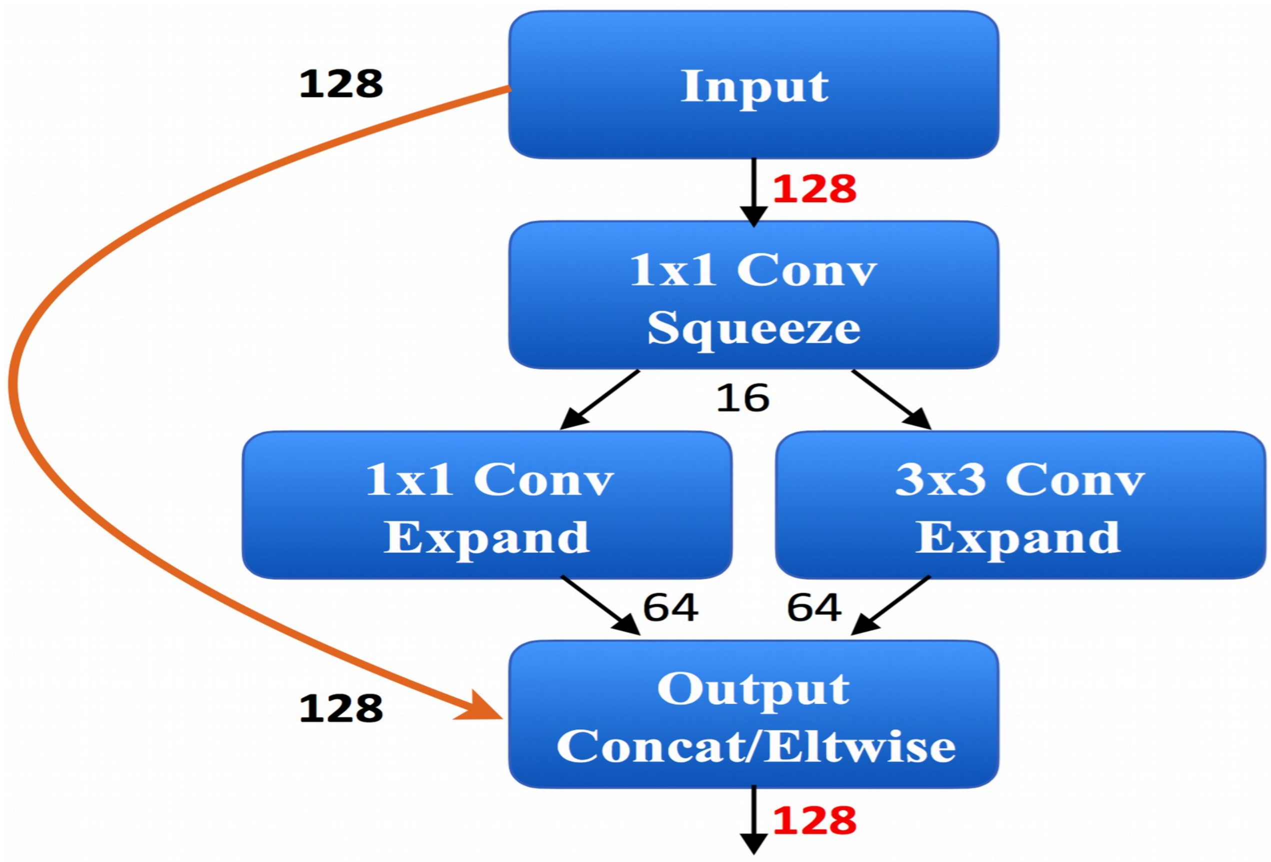
Figure 2. The SqueezeNet architecture

SqueezeNet

In this arhcitecture depths are squeezed before each operation

In a squeezenet architecture we will use a linear approximatino of 128 feature maps, using 16 independent feature maps

From the linear combination of these elements the new maps are created



Neural networks for regression

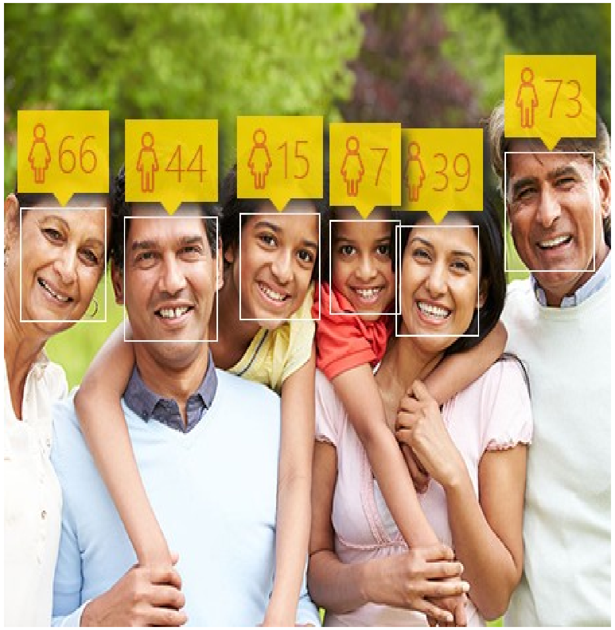
Age estimation

The output is not discreet classes or pixels, but continuous values

The network structure can remain the same but a different loss function and differently annotated dataset is needed.

Hard to interpret the error in common tasks.

E.G: Age estimation on images:

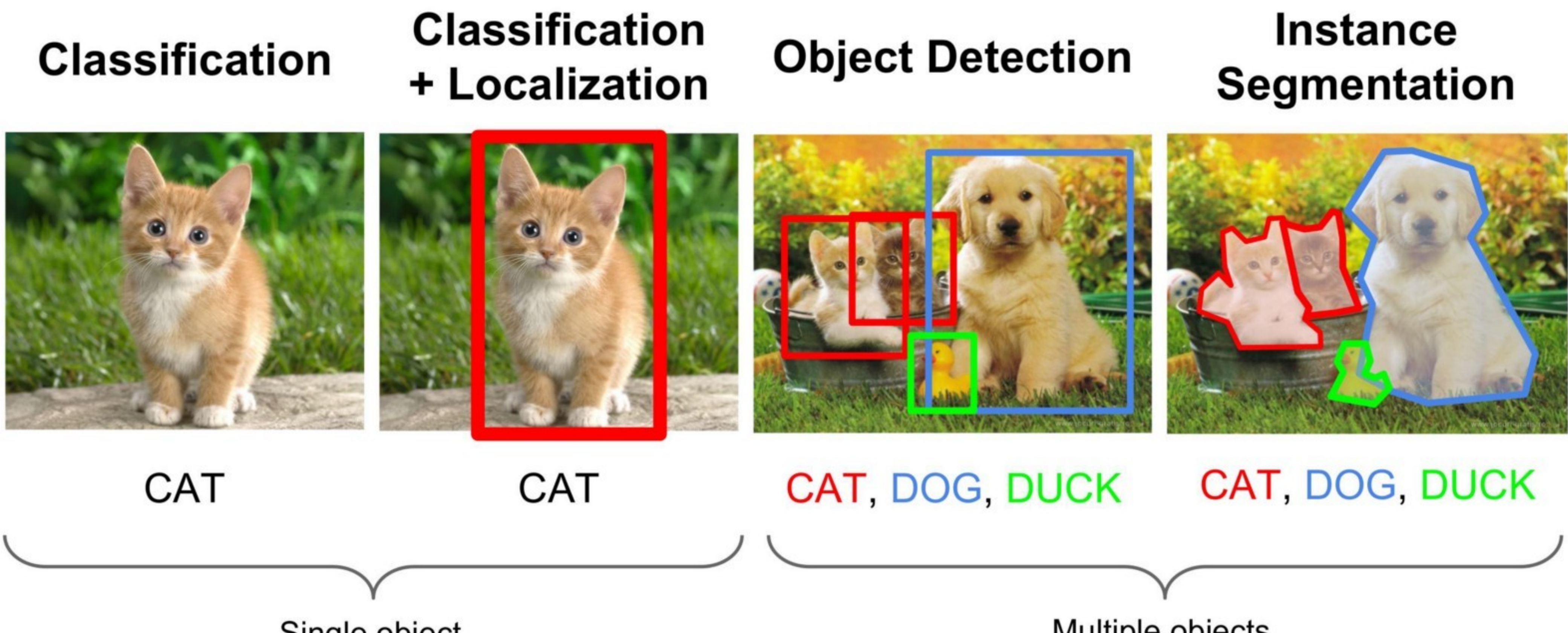


Neural networks for regression

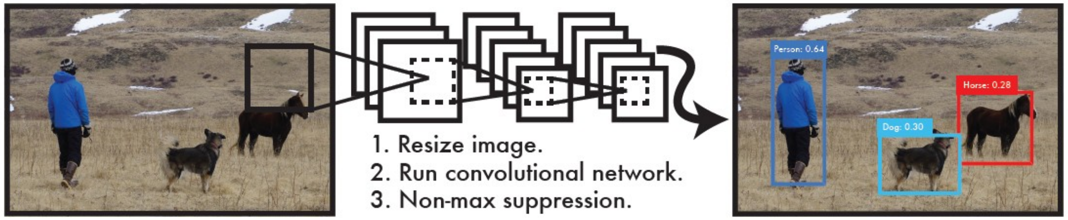
Multiple object detection on a single image

Classification is good for a single object (can be extended for k objects – top k candidates)

How could we detect objects in general, when the number of objects is unknow



Traditional method



Sliding window over the image

We might have objects in different scales

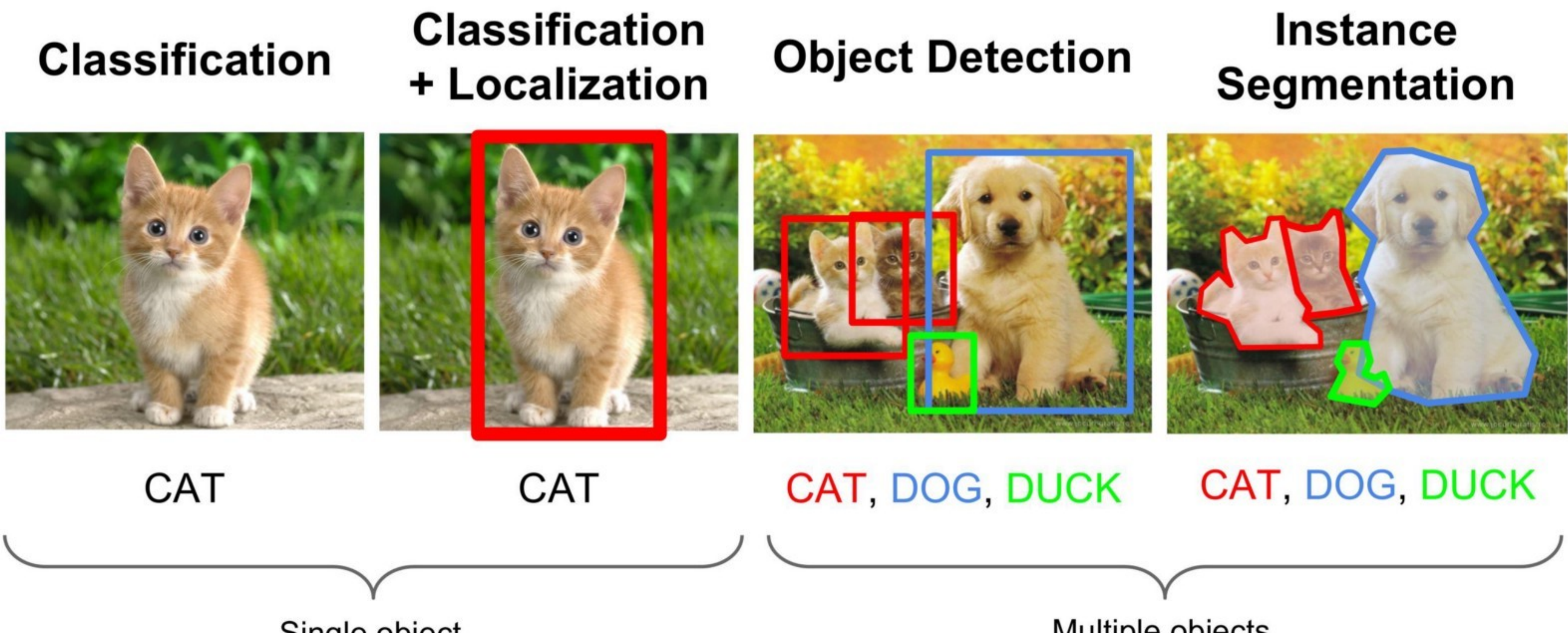
Slidign windowds in different scales, aspect ratios

Resutls a heat map → detect the objects: non-maximum suppression

Object detection as regression

Single shot object detector SSD (2016 March)

You Only Look Once YOLO (2016 May)

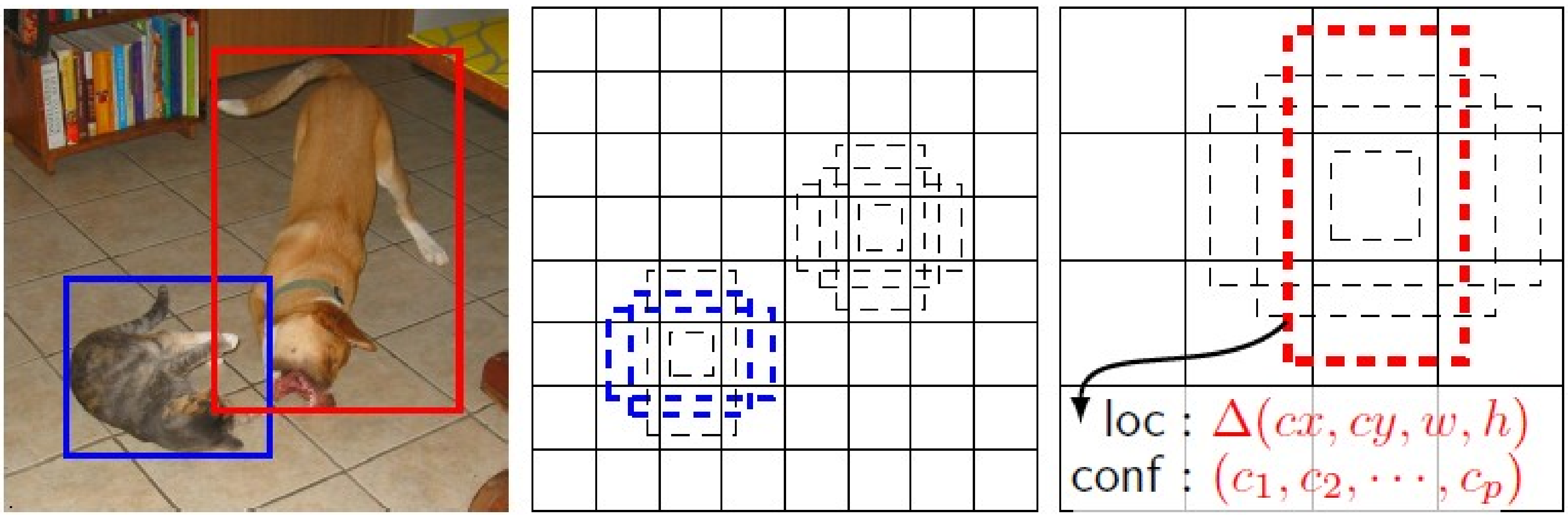


SSD

Single shot object detector SSD (2016 March)

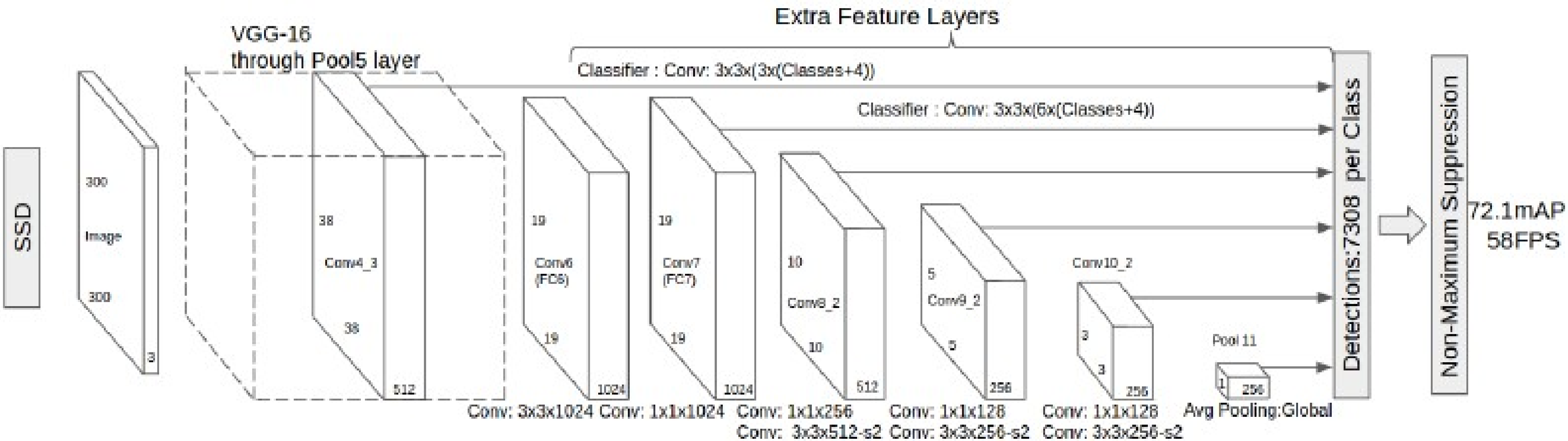
Has a fixed resultion and the last feature maps (with different scales) can be considered as maps of bounding boxes

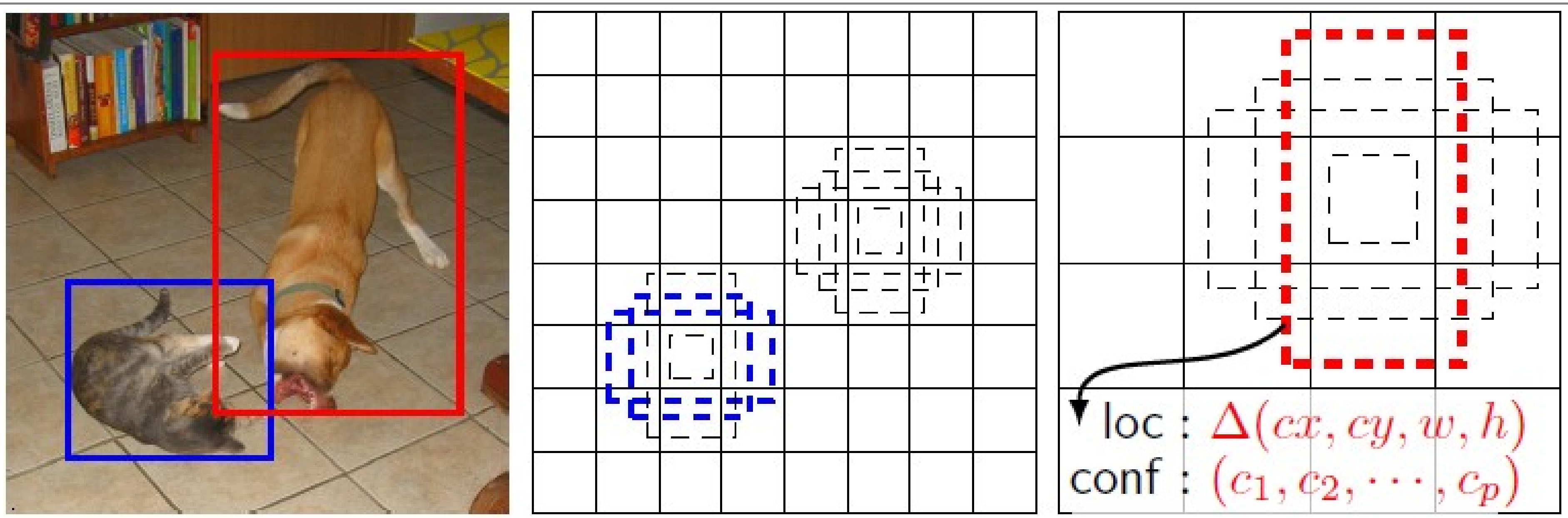
On these maps each pixel represent bounding boxes. A high pixel value represent high probability of the centerpoint of a detected object.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

SSD arhcitecture



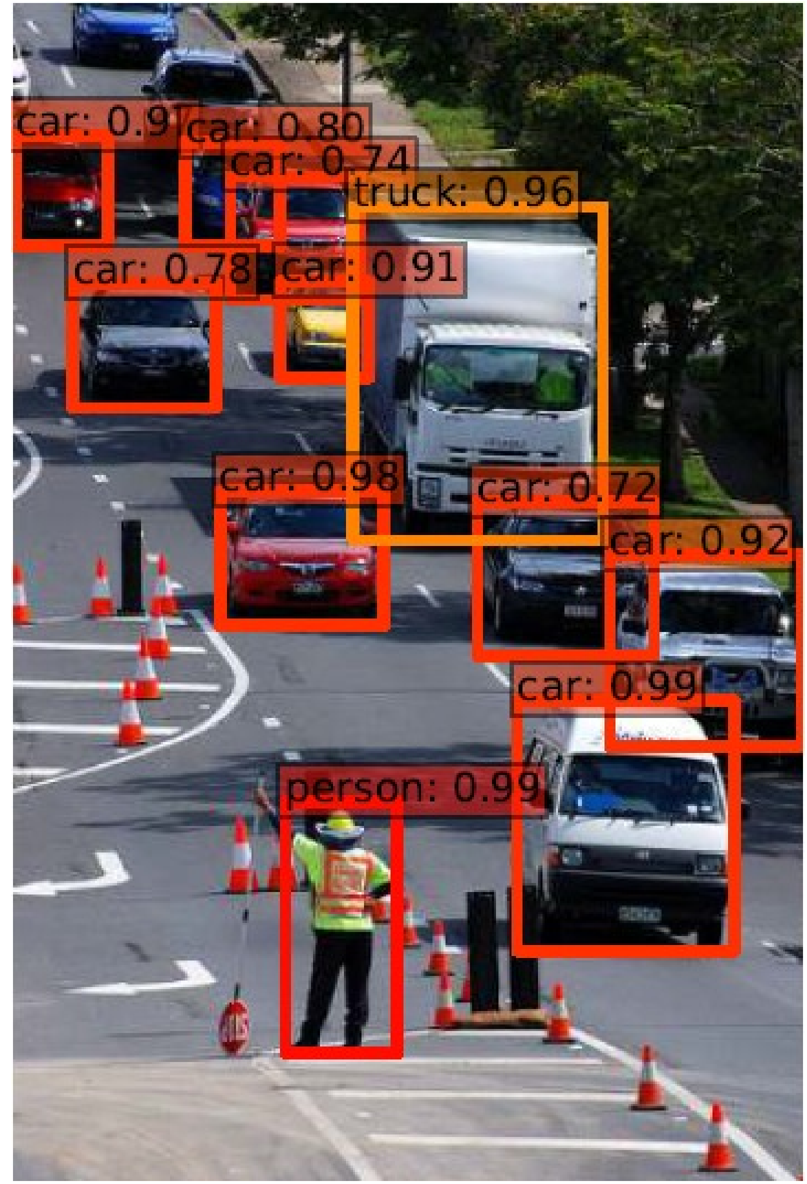


Loss function for bounding boxes

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

--The overall objective loss function is a weighted sum of the localization loss and the confidence loss(conf)
--N: the number of matched default boxes
--l: predicted boxes
--g: the ground truth box
--x=1 denotes some certain default box is matched to a ground truth box


truck: 0.60
person: 0.83
person: 0.86
dog: 1.00


car: 0.80
car: 0.74
truck: 0.96
car: 0.78
car: 0.91
car: 0.98
car: 0.72
car: 0.92
car: 0.99
person: 0.99

R-CNN

Region proposal CNN network

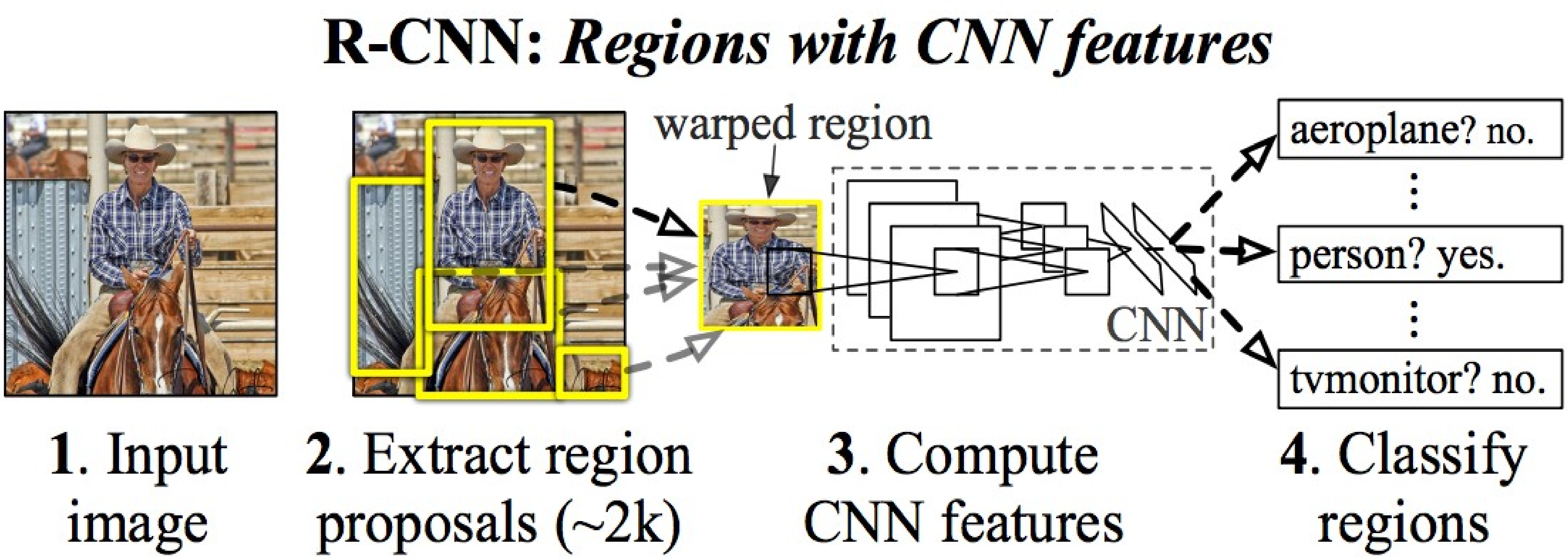
Separate the problem of object detection and calssification

It consists of three modules.

The first generates category-independent region proposals. These proposals define the set of candidate detection avail-able to detector.

· The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region.

The third module is a set of class- specific linear SVMs



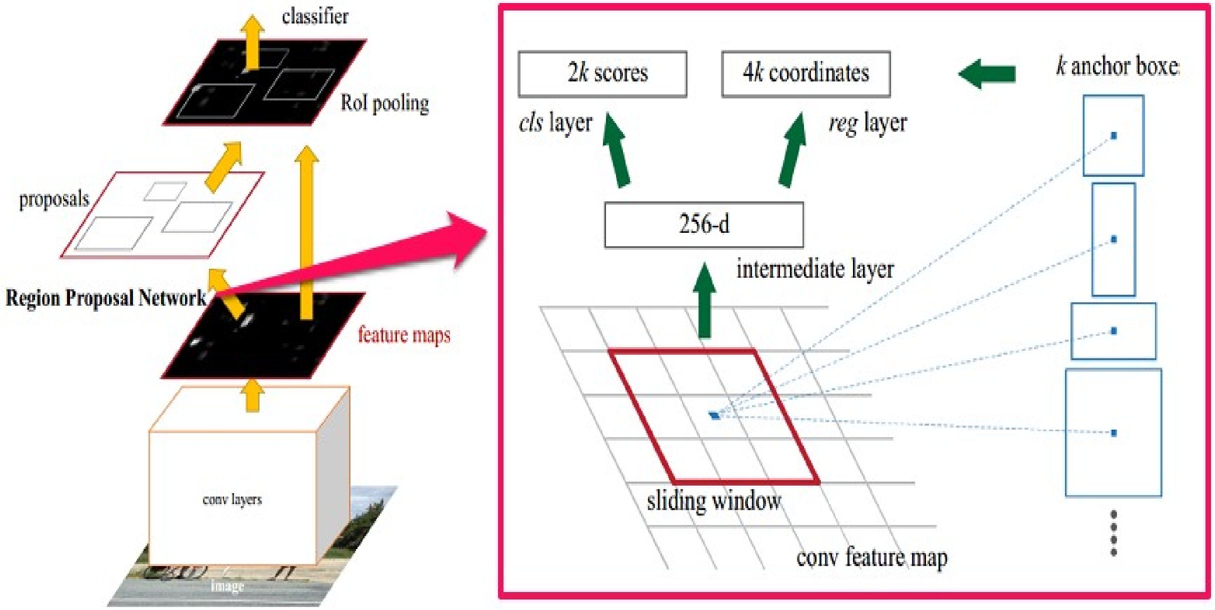
R-CNN

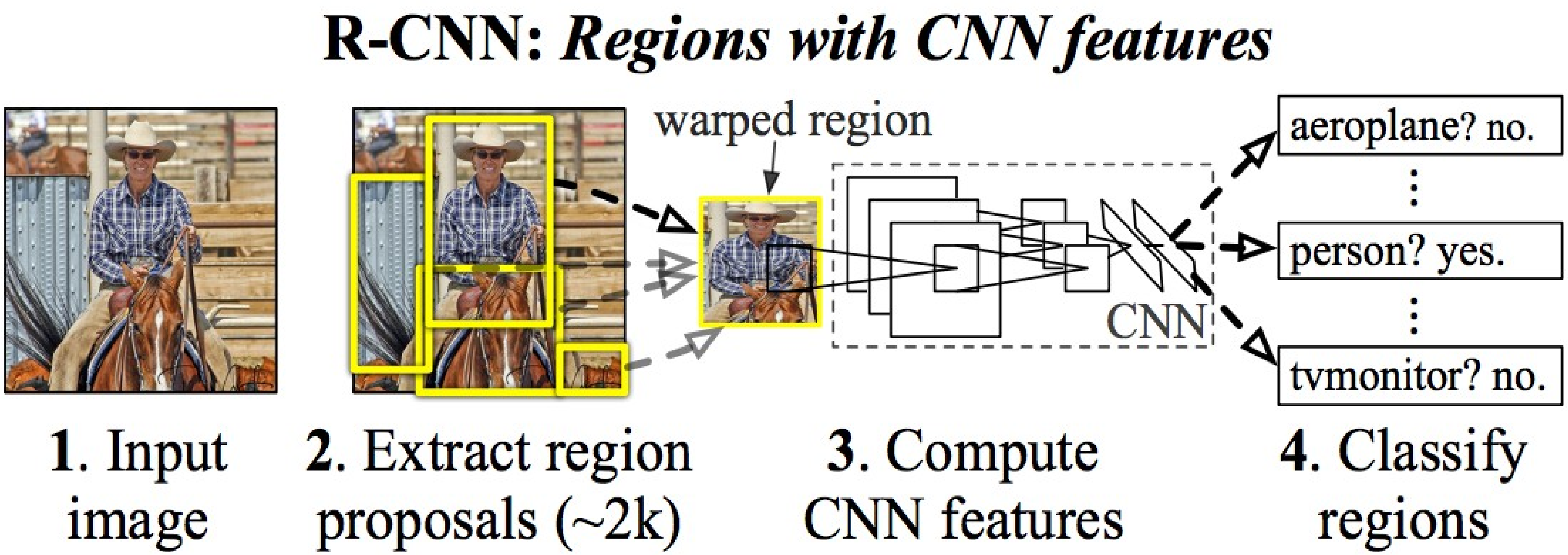
Region proposal from a network

Step 3 and 4 are standard CNN implementations

Extra layers for region proposals

Possible region refinement at the end



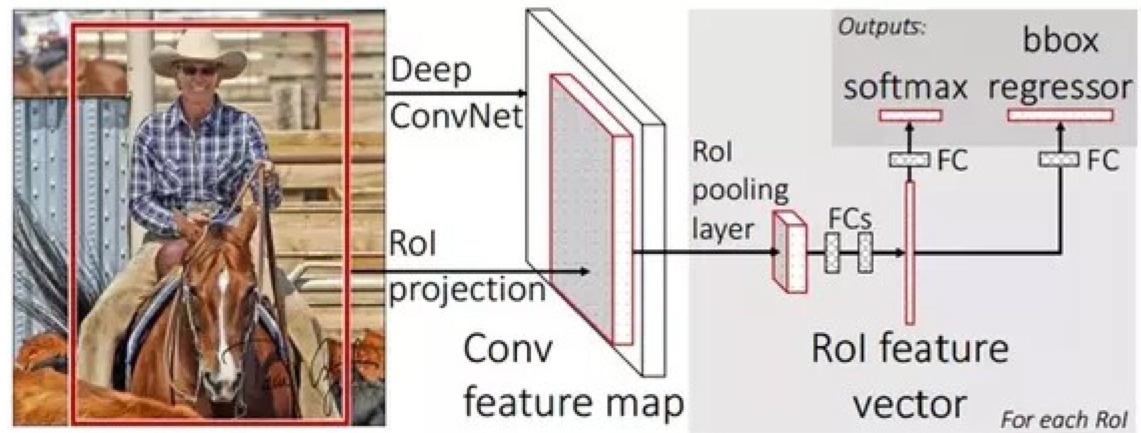


Fast R-CNN

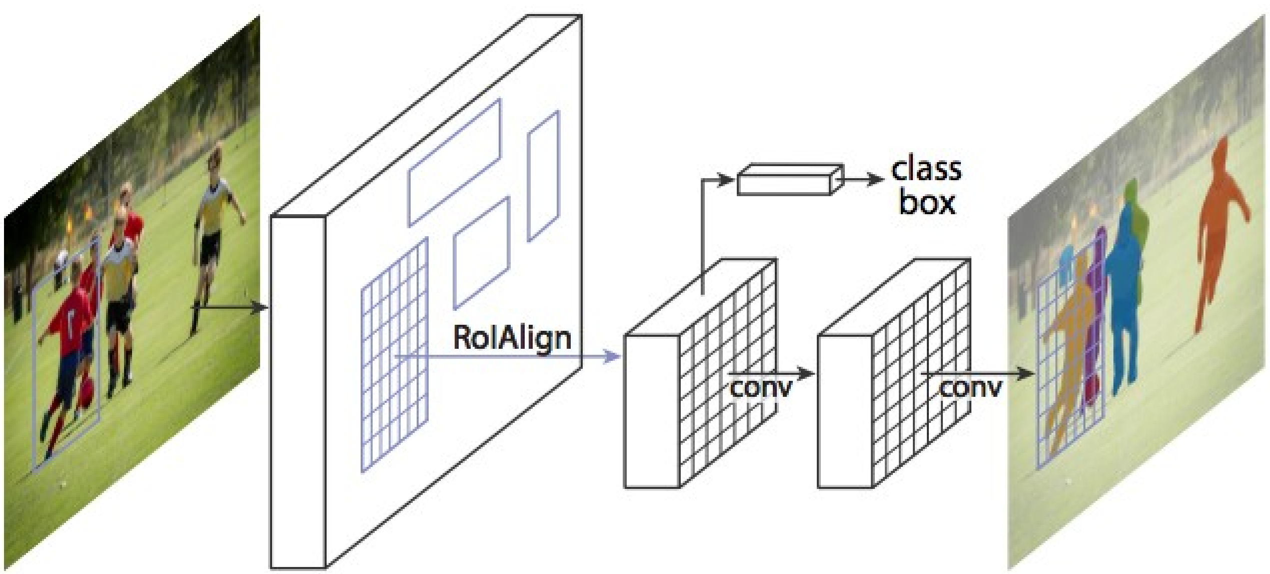
Fast R-CNN:

Speeding up by sharing computation of the conv layers between the proposal generation and classificaiton

In this model, the image is first fed through a ConvNet, features of the region proposals are obtained from the last feature map of the ConvNet and lastly we have fully connected layers as well as our regression and classification heads.



Fast R-CNN workflow



Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE i

YOLO, Detectnet

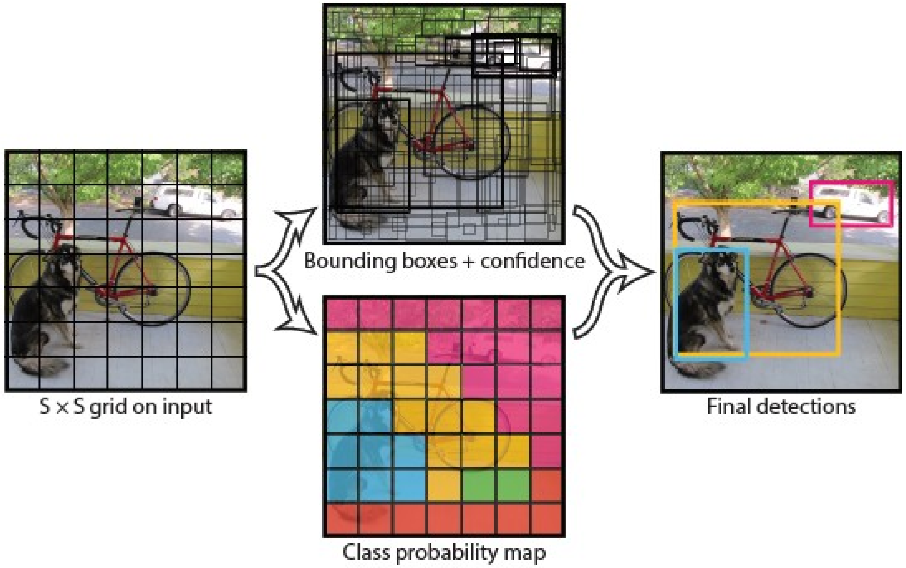
Models detection as a regression problem:

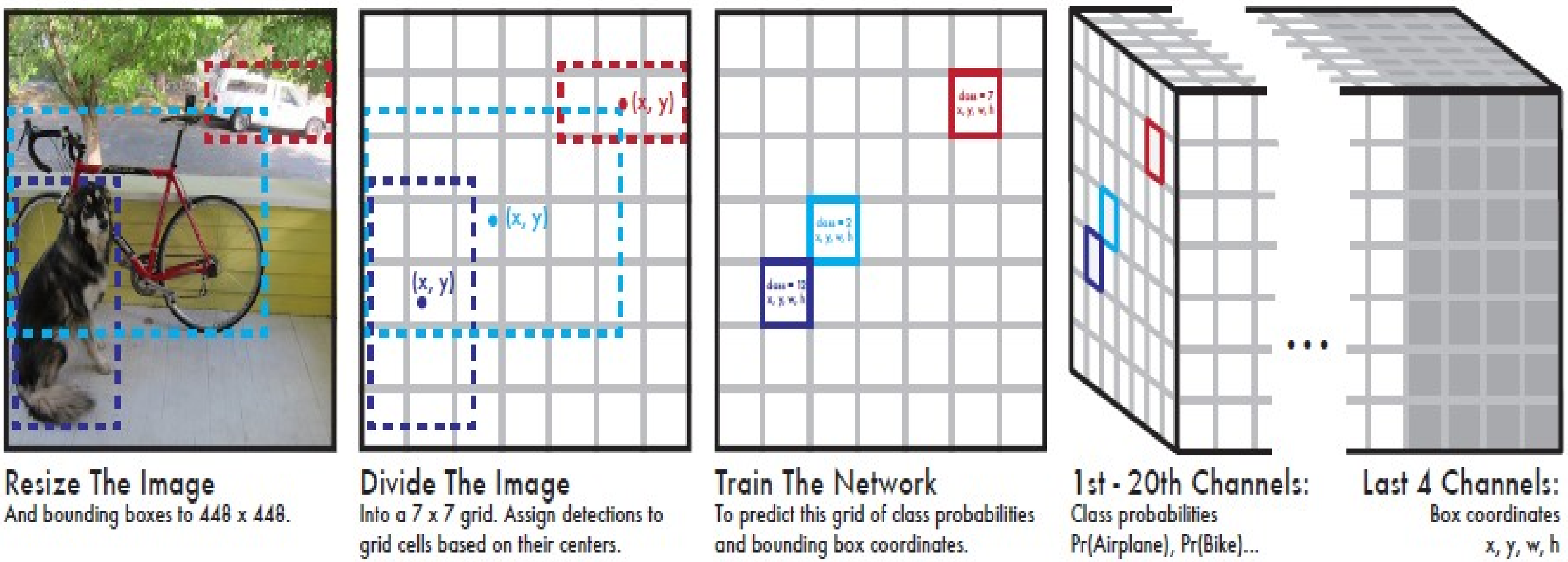
Divide the image into a grid and each cell can vote for the bounding box position of possible object.

Boxes can have arbitrary sizes

Non-suppression on the boxes

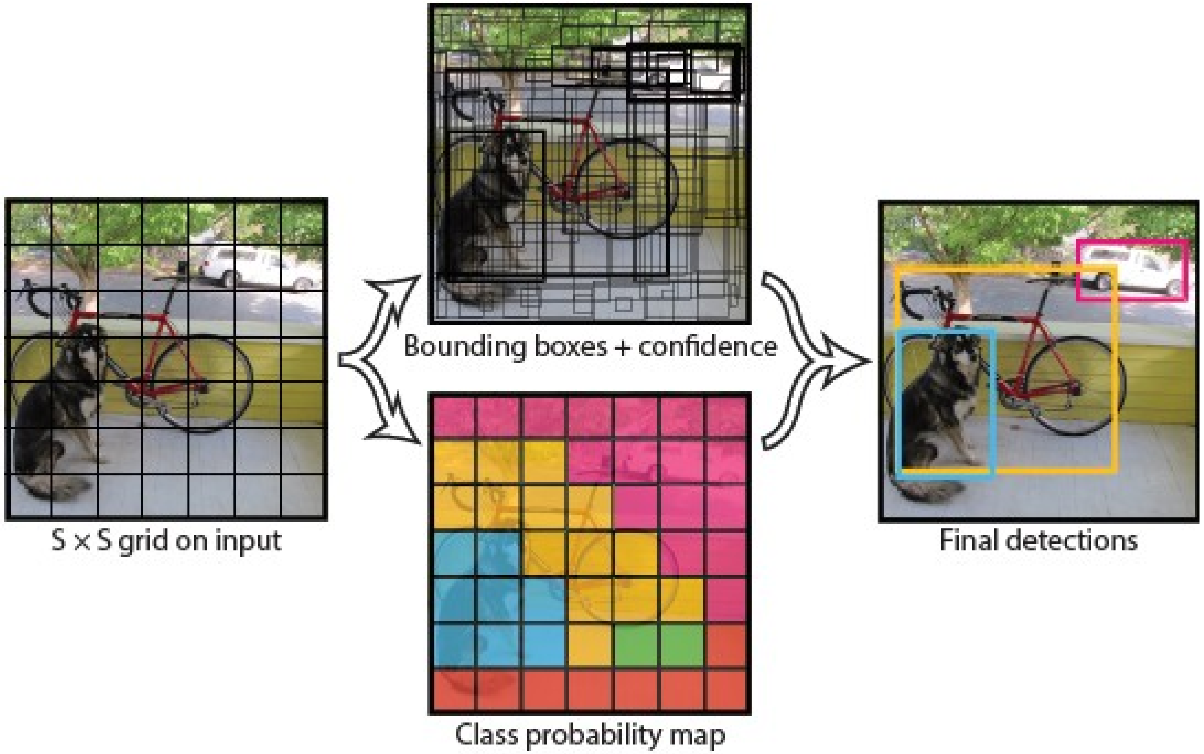
No need for scale search, the image is processed once and objects in different scales can be detected





Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

How unified detection works?



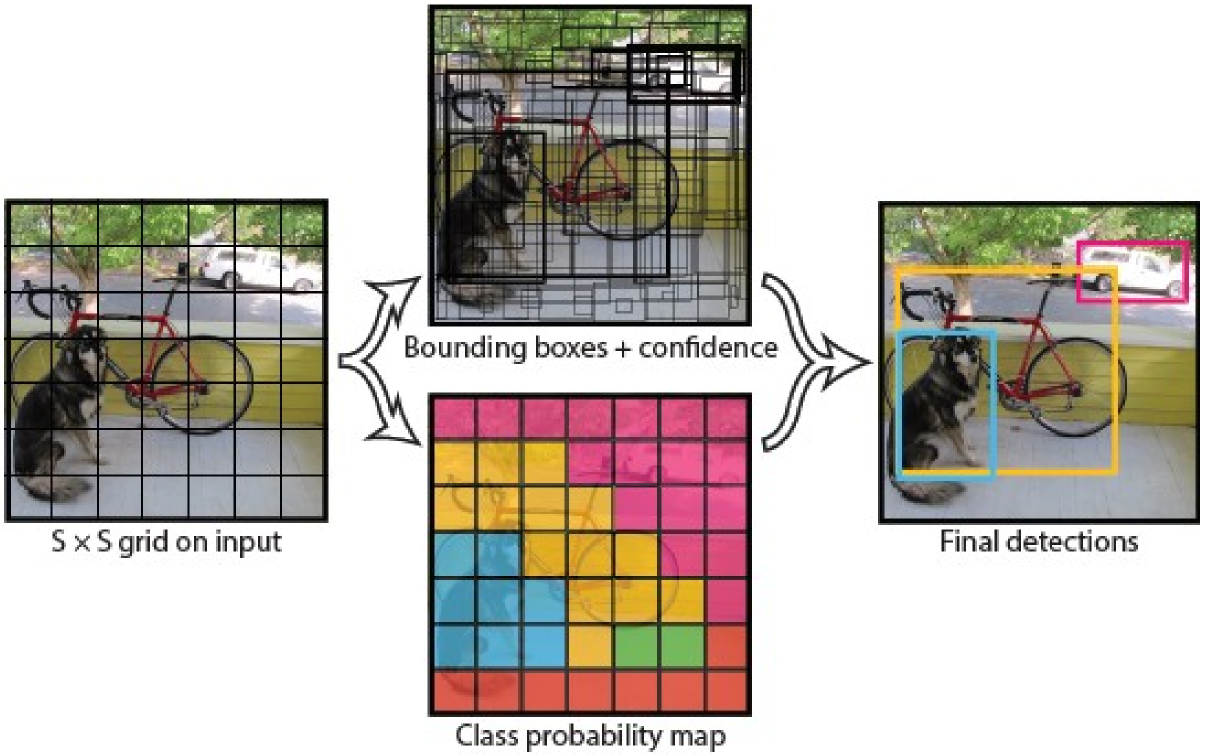
<u>confidence scores</u>: reflect how confident is that the box contains an object+how accurate the box is .

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

conditional class probabilities: conditioned on the grid cell containing an object

$$\Pr(\text{Class}_i|\text{Object})$$

How unified detection works?



$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- At test time, multiply the conditional class probabilities and the individual box confidence predictions
- giving class-specific confidence scores for each box
- Showing both the probability of that class appearing in the box and how well the predicted box fits the object

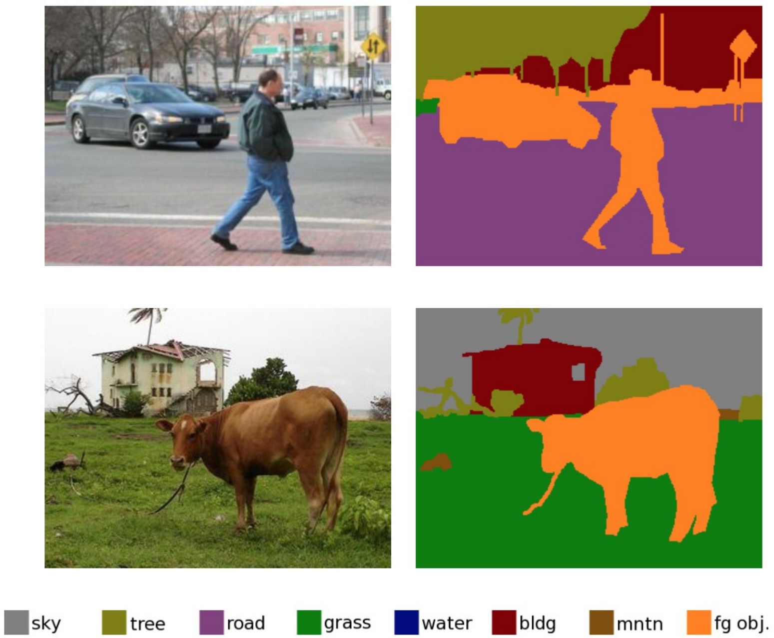
Pixel level segmentation

The expected output of the network is not a class, but a map representing the pixels belonging to a certain class.

Creation of a labeled dataset (handmade pixel level mask) is a tedious task

More complex architectures are needed (compared to classification)

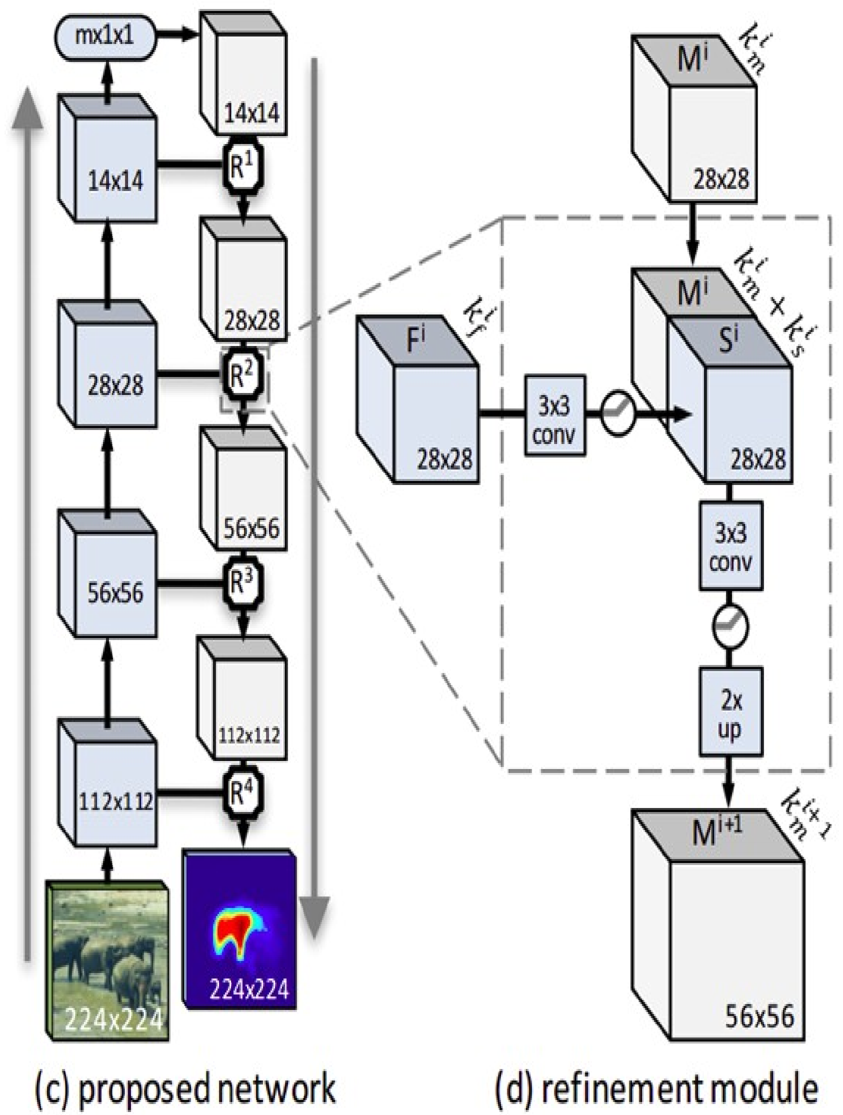
Popular architectures (Sharpmask, U-NET ...)



SharpMask: Learning to Refine Object Segments. Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, Piotr Dollàr (ECCV 2016)

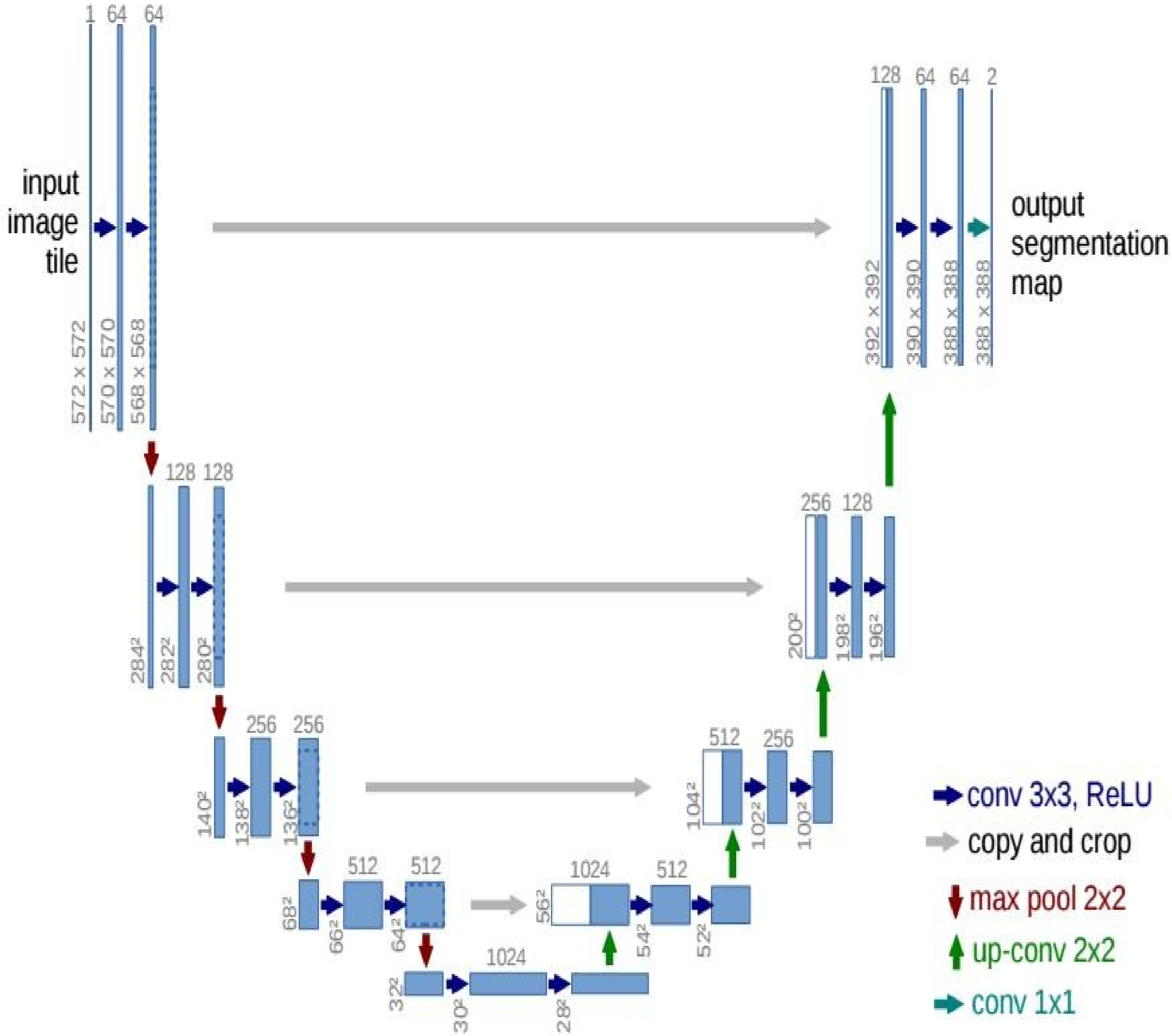
SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS Liang-Chieh Chen et al. ICLR 2015

Sharpmask



SharpMask network architecture

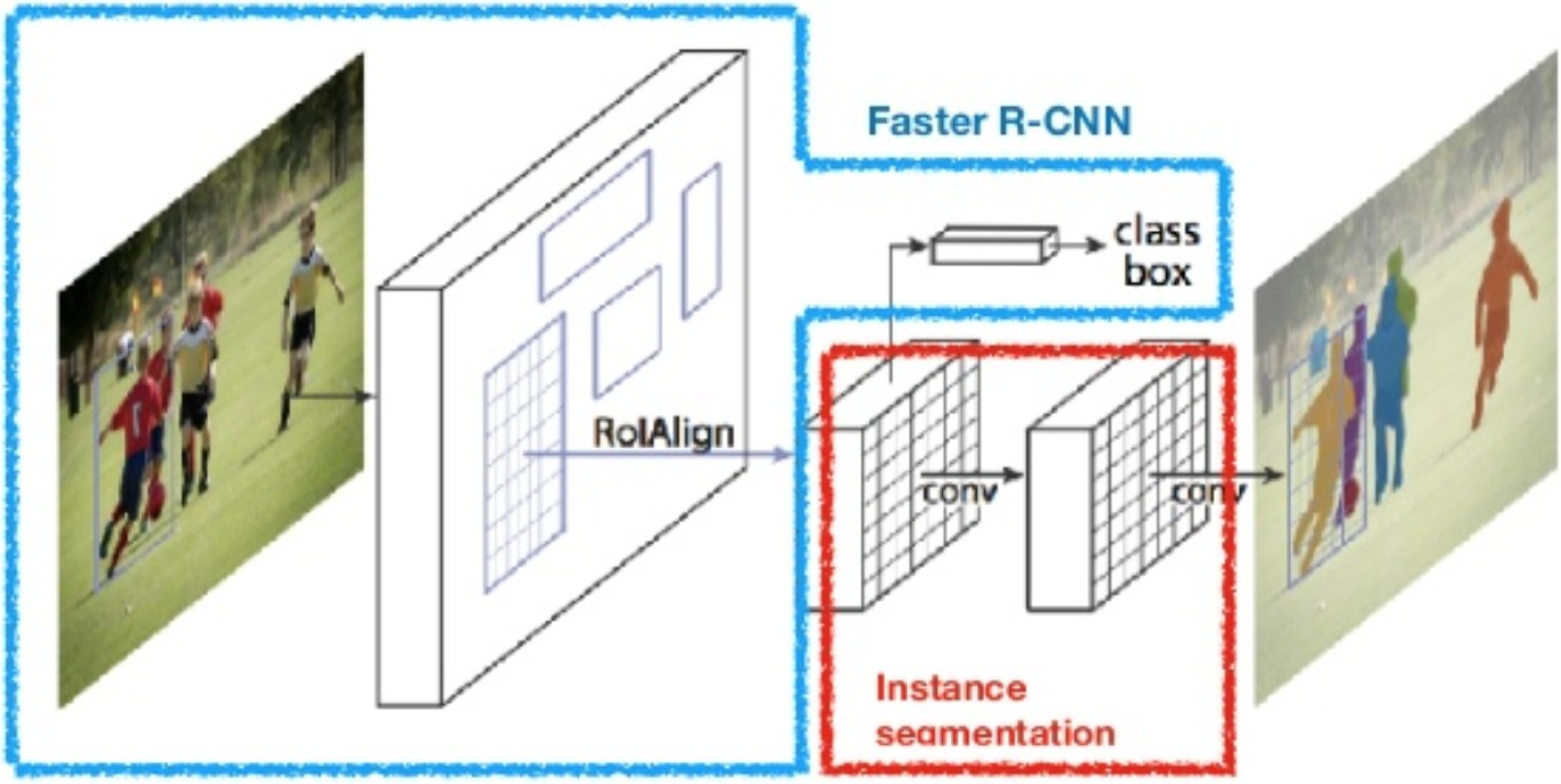
U-net


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

Mask RCNN, RetinaNet

These networks generate bounding boxes and sematnic segmentation maps simultanously

They can be trained on images having lables for only one or both types of output

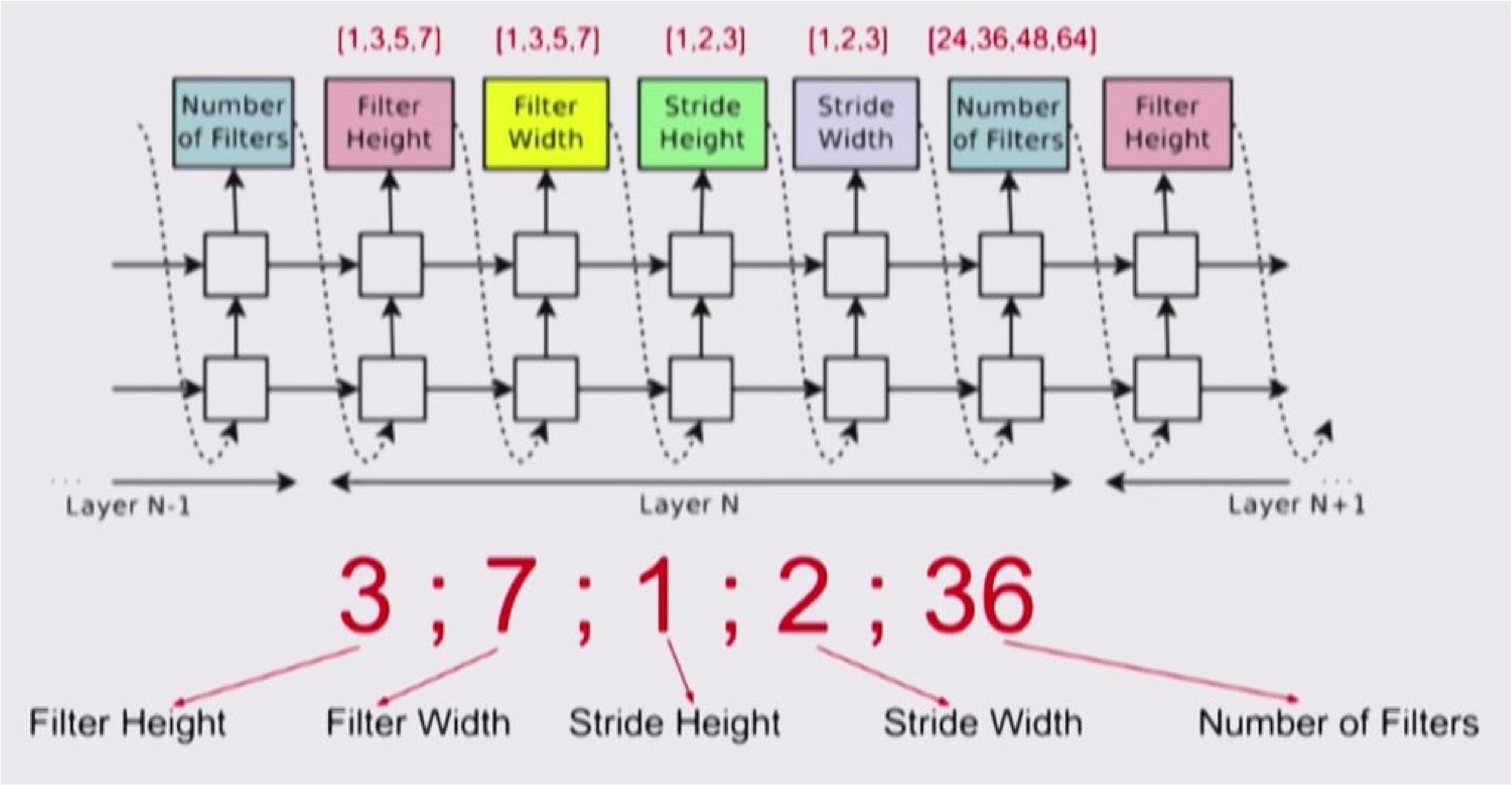


Mask RCNN, RetinaNet

These networks generate bounding boxes and sematnic segmentation maps simultanously

Starting from scratch

Neural architecture search:
Networks can be described as a series of operations
As series of words → text



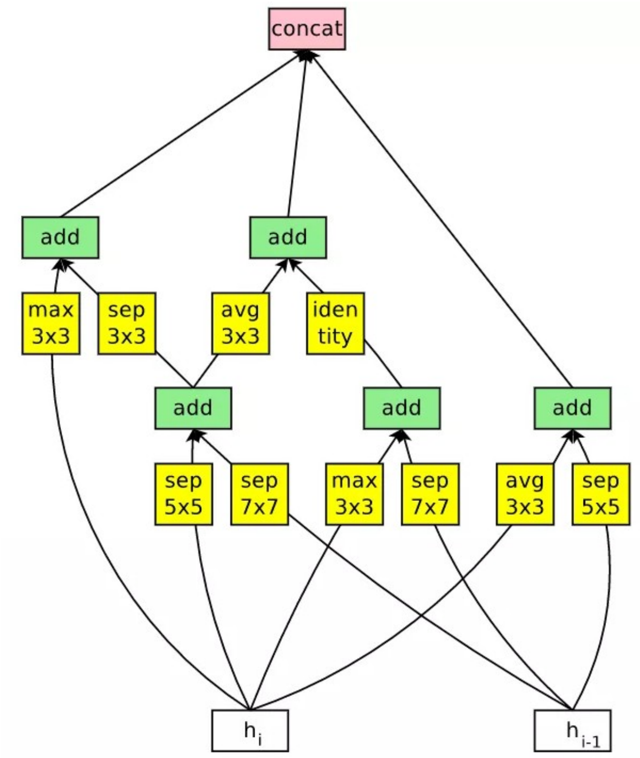
Starting from scratch

Neural architecture search:
Networks can be described as a series of operations
As series of words → text

The parameters of eahc layer can be described as numbers The input(s)/outputs(s) of the layer can be Ids

The whole network can be described as a graph

```
layers {
bottom: "conv1"
top: "conv1"
name: "relu0"
type: RELU
}
layers {
bottom: "conv1"
top: "cccp1"
name: "cccp1"
type: CONVOLUTION
blobs_lr: 1
blobs_lr: 2
convolution_param {
num_output: 96
kernel_size: 1
stride: 1
}
```



Starting from scratch

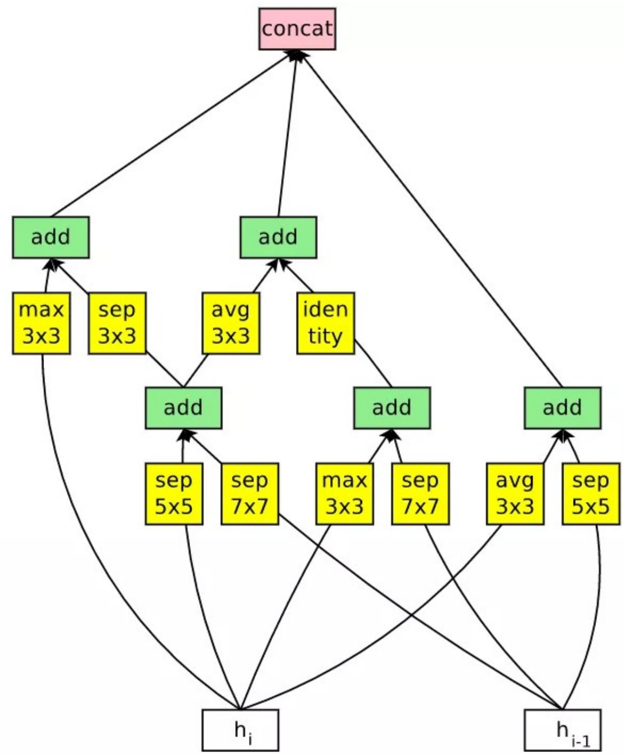
Neural architecture search:
Networks can be described as a series of operations
As series of words → text

The parameters of eahc layer can be described as numbers The input(s)/outputs(s) of the layer can be Ids

The whole network can be described as a graph

We have a problem space where we have text as an input and an accuracy number as an output

```
layers {
bottom: "conv1"
top: "conv1"
name: "relu0"
type: RELU
}
layers {
bottom: "conv1"
top: "cccp1"
name: "cccp1"
type: CONVOLUTION
blobs_lr: 1
blobs_lr: 2
convolution_param {
num_output: 96
kernel_size: 1
stride: 1
}
```



Starting from scratch

Neural architecture search:
Networks can be described as a series of operations
As series of words → text

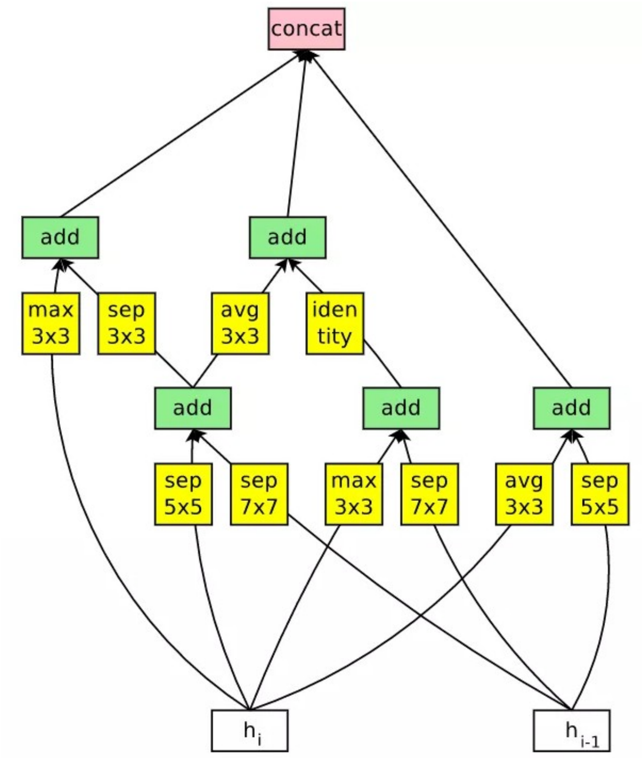
The parameters of eahc layer can be described as numbers The input(s)/outputs(s) of the layer can be Ids

The whole network can be described as a graph

We have a problem space where we have text as an input and an accuracy number as an output

We can train an RNN for regression, which approximates the accuracy of a given network

```
layers {
bottom: "conv1"
top: "conv1"
name: "relu0"
type: RELU
}
layers {
bottom: "conv1"
top: "cccp1"
name: "cccp1"
type: CONVOLUTION
blobs_lr: 1
blobs_lr: 2
convolution_param {
num_output: 96
kernel_size: 1
stride: 1
}
```



Starting from scratch

Neural architecture search:
Networks can be described as a series of operations
As series of words → text

We can turn the problem around:

A recurrent network can be trained with reinforcement learning which can train a network with predifined accuracy on a given dataset.

This recurrent network will understand the effect of the elements on this dataset

Test accuracy On CIFAR-10:
96.35%

Best pervious accuraccy:
96.26

This architecture os also 1.05 times faster (less computations)