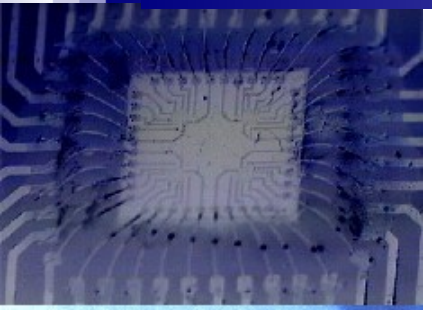
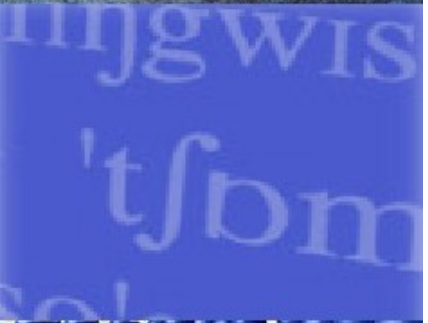
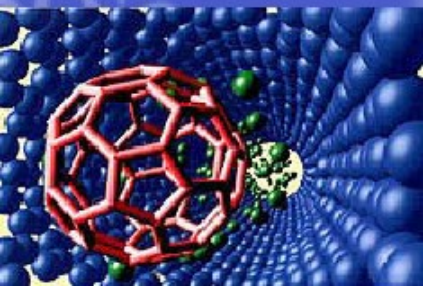


Neural Networks
(P-ITEEA-0011)

Famous architectures

András Horváth, *Ákos Zarándy*

Budapest, 2019.12.03

Administrative announcements

- Replacement paper-based test 17. 12. 9:00, Room 418
  - papíros pót ZH - dec. 17 9:00, 418-as terem
- Early exam 17. 12. 9:00, Room 419
  - The invited students will be emailed acknowledged this week

  Early exam - dec. 17 9:00, 419-es terem,
  érintettek a héten megtudják meg
- Project presentation - 17. 12. 11:00, Room 418

  Projekt bemutatás - dec. 17 11:00, 418-as terem
-
- Computer-based test - 19. 12. 9:00

  Géptermi ZH - dec. 19 9:00
-
- Computer-based replacement test TBA, early January

  Géptermi pót TBA, ~január eleje
-
- Oral Exams are already in the Neptun system

  Vizsgaidőpontok a Neptunban

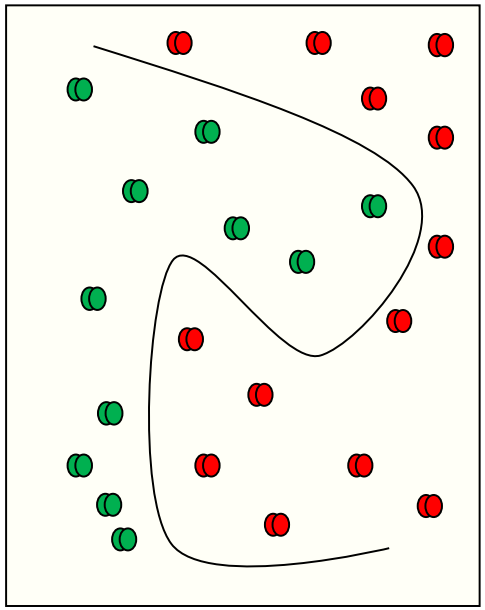
We are considering to create a list of the participants, to reduce waiting time for the oral exam.

Neural Networks

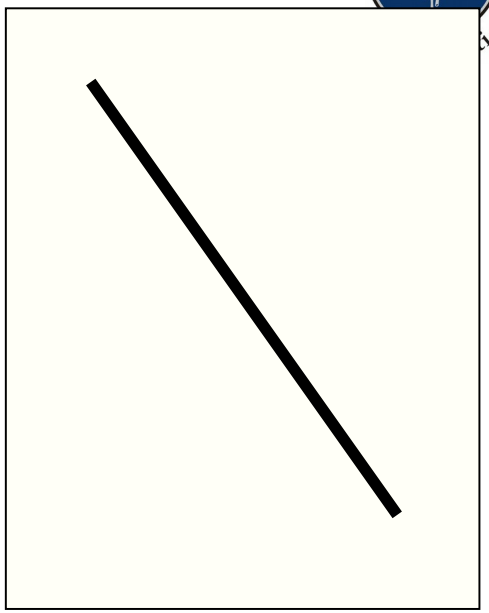
- Classification - decision
- FNN, SVM – linear classification
-

  Is X larger than a limit? X>k?

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



- Input *space*



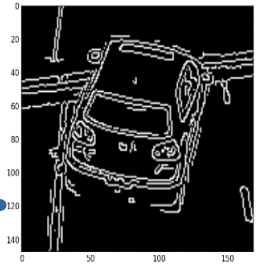
Feature space

- Finding the representation with the help of machine learning

Input Image

Convolution Kernel

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



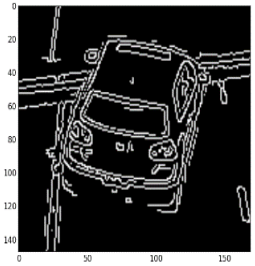
Feature Image

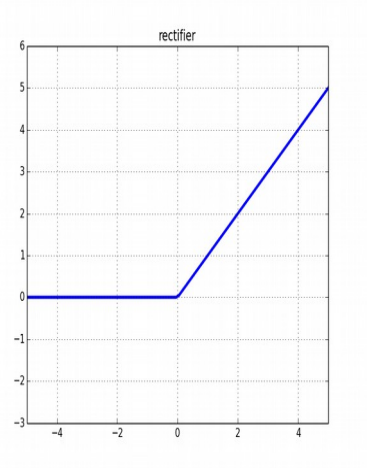
Convolutional neural networks

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

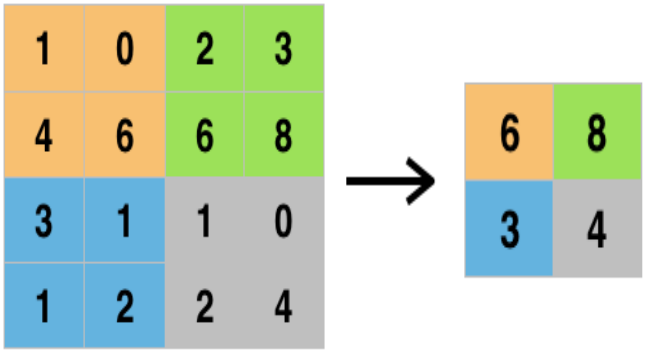
Convolution Kernel

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

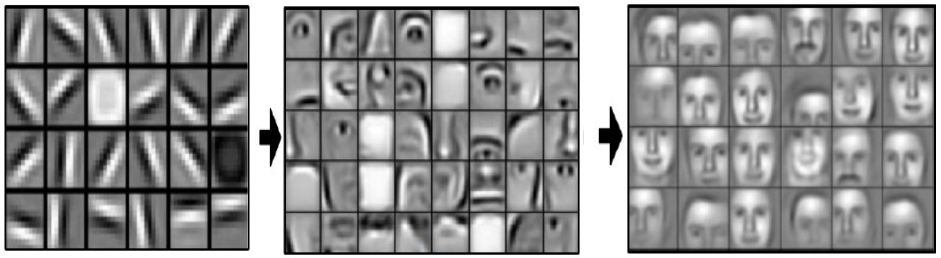




Thresholding all values below zero



Selection of the maximal response in an area



Low layers *Middle layers* *High layers*

Convolutional networks

Assume, I have a problem to solve.

Ok, but how many layers do we need?

How many features should be in each layer?

What should be the network architecture?

Convolutional networks

Assume, I have a problem to solve.

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 with: non-linearity type, batch-norm, dropout etc.

Convolutional networks

Assume, I have a problem to solve.

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 with: 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

Alexnet

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

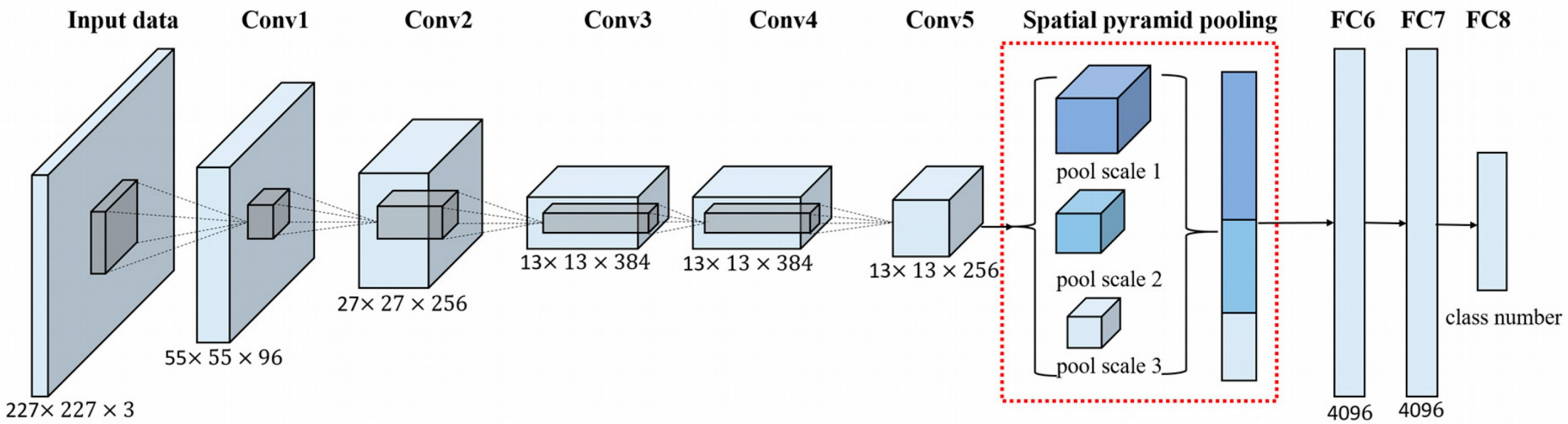
Trained whole ImageNet (15 million,22,000 categories)

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

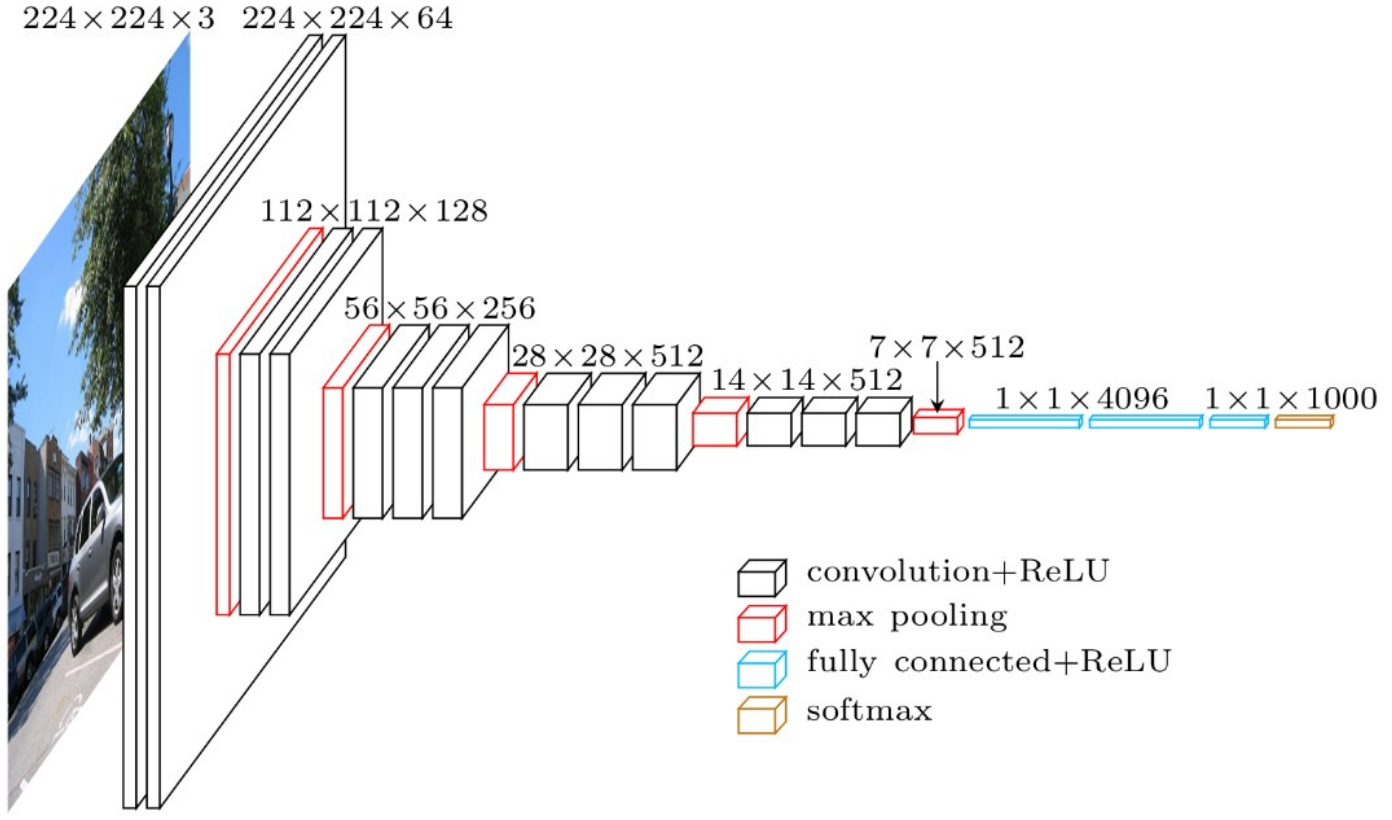
As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network.

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

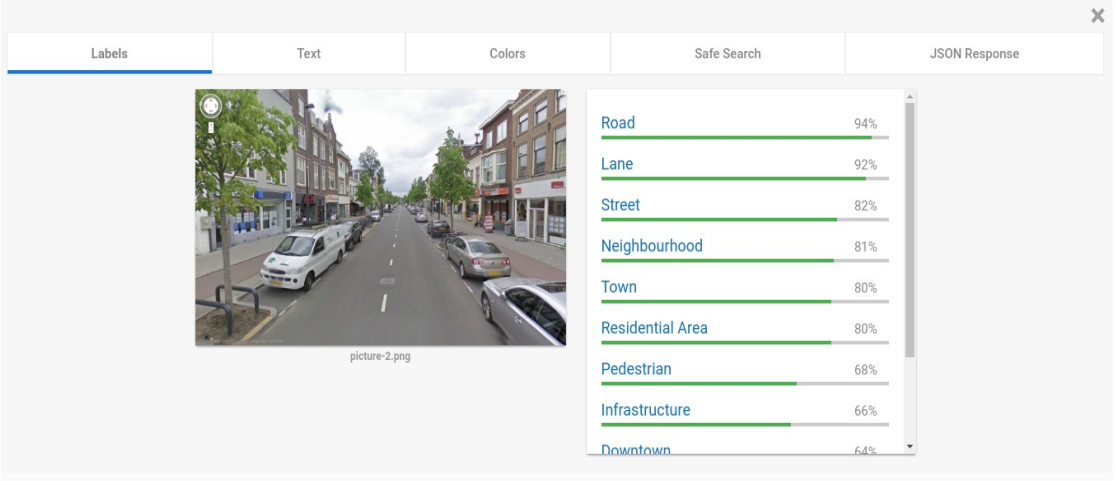
Simple architecture, still the swiss knife of deep learning

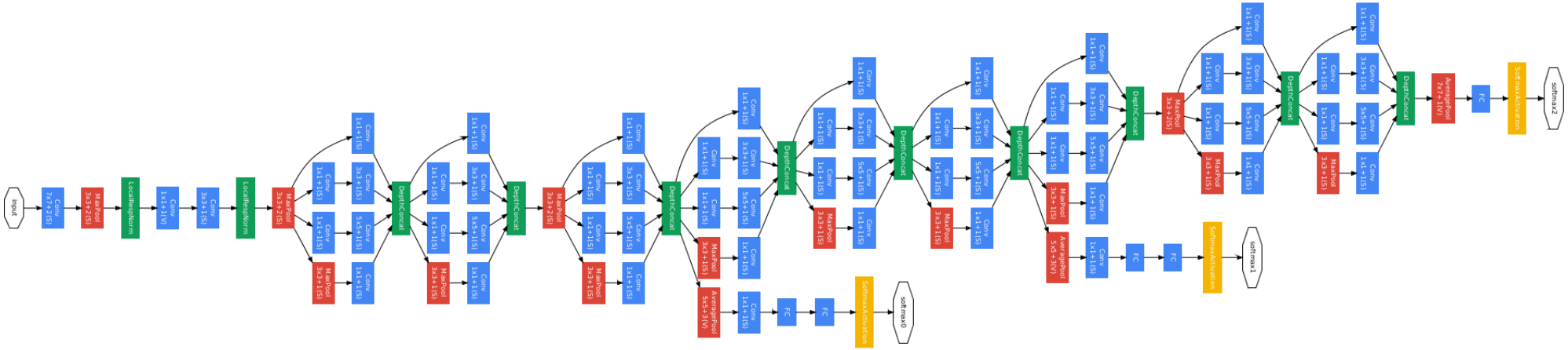


Google - Inception arhcitecture

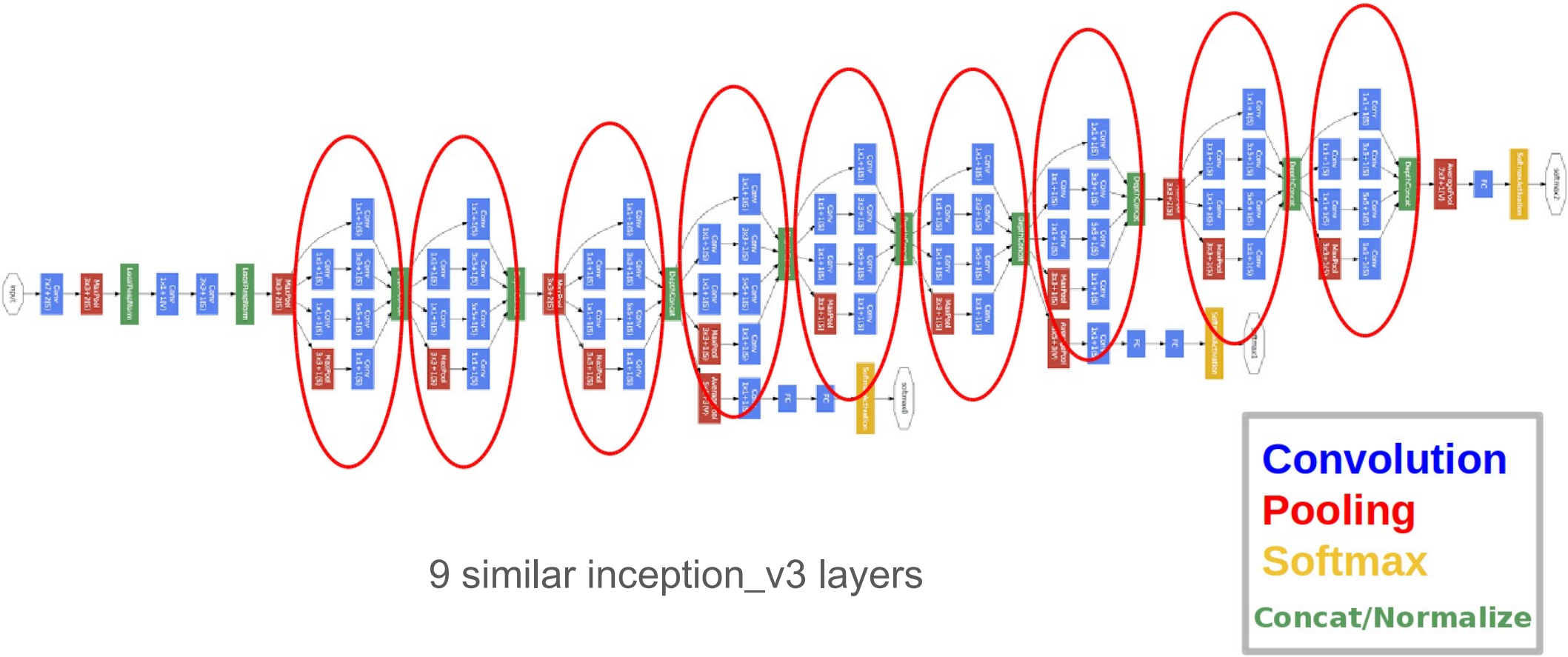
- GoogLeNet:

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





Inception module



9 similar inception_v3 layers

Inception

Google, Christian Szegedy

2014 with a top 5 error rate of 6.7%

This 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.

Idea:

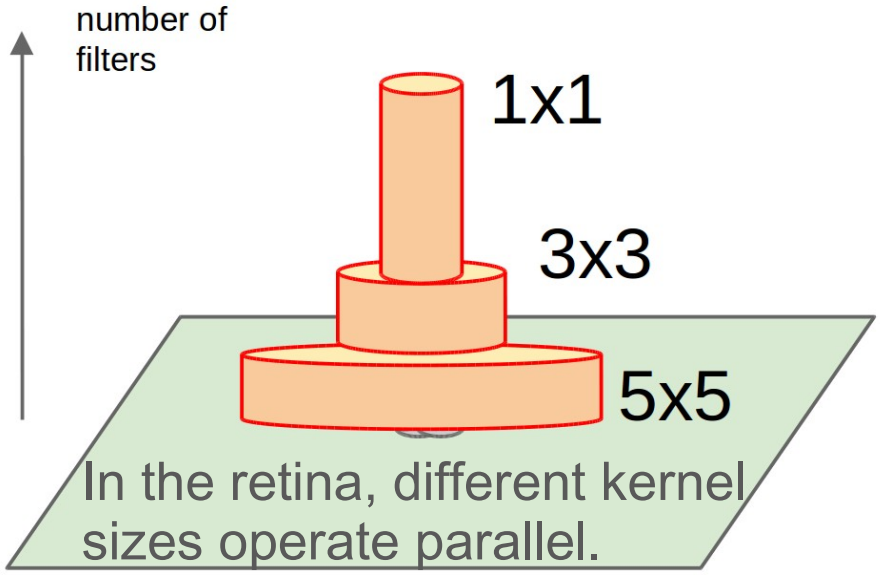
Not to introduce different size kernels in different layers, but introduce 1x1, 3x3, 5x5 in each layers, and let the Neural Net figure out, what representation is the most useful, and use that!

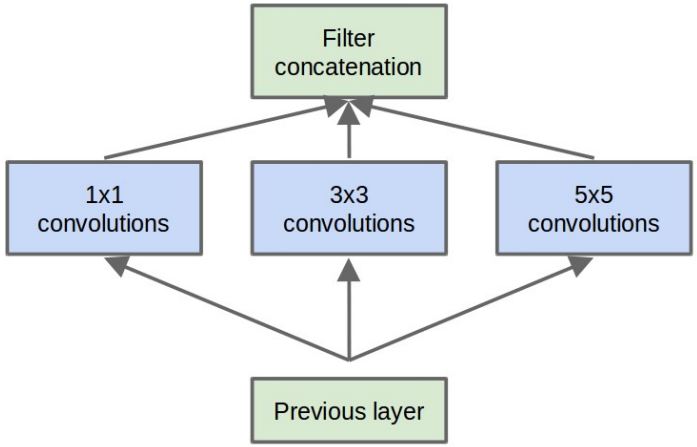
Parallel multi-scale approach.

AlexNet: 60 million param

VGGNet: 1800 million

GoogLeNet / Inception-v1





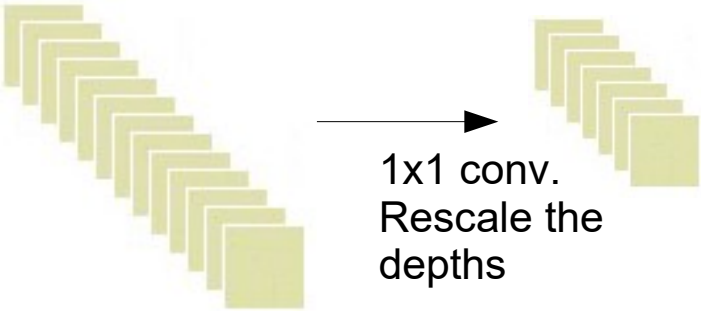
Rethinking Inception

Squeezing the number of channels for each kernel

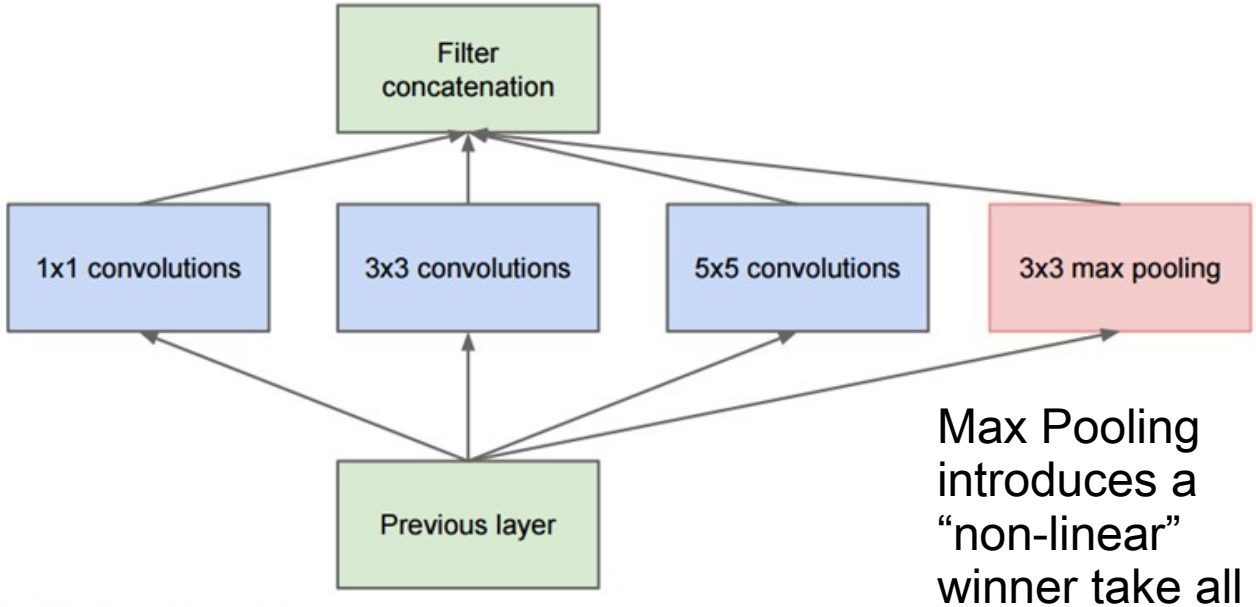
With the concatenations, the number of features increased in each layers, which introduced too many convolution.

To reduce these numbers, they introduced the 1x1 layer.

It can generate e.g. 16 feature maps from 64 feature maps

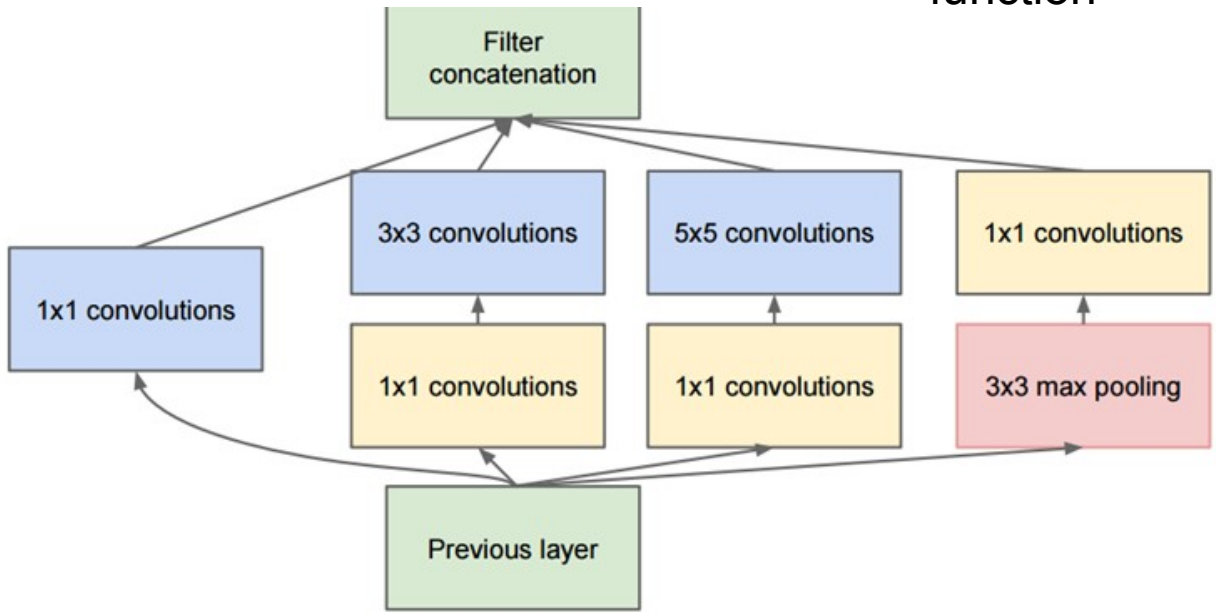


1x1 conv.
Rescale the depths



Naïve idea of an Inception module

Max Pooling introduces a "non-linear" winner take all function

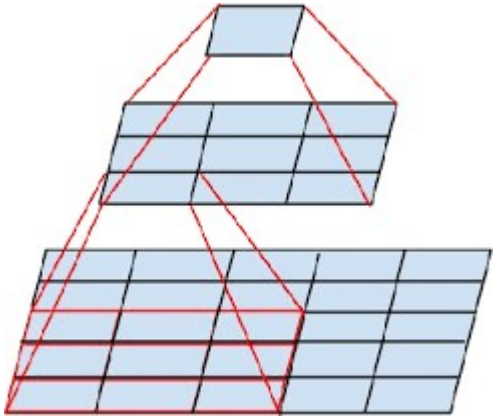


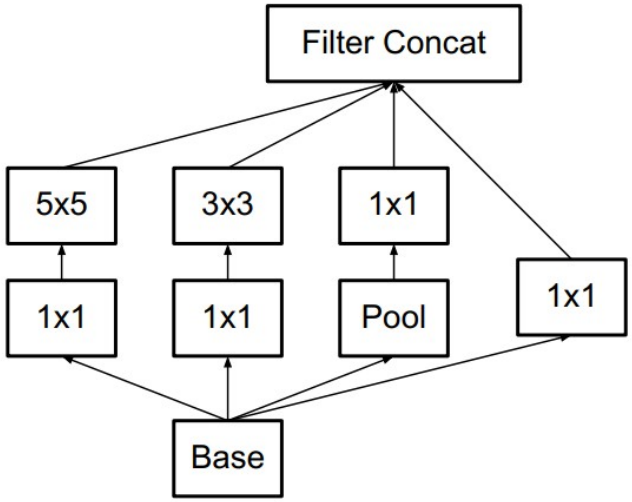
Full Inception module

Rethinking Inception

Larger (5x5) convolutions were substituted by series of 3x3 convolutions

Advantages:
1. Reduction of number of parameters,
2. Additional non-linearities (RELUs) can be introduced





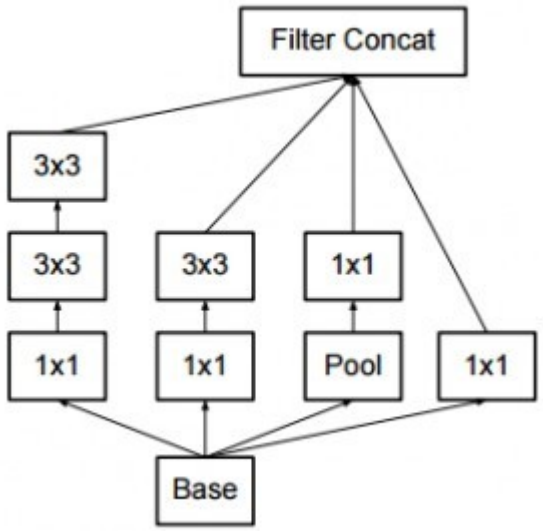
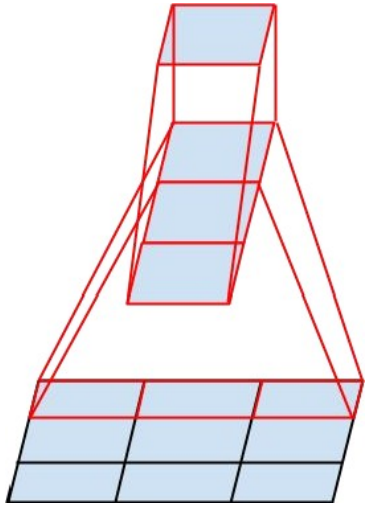


Figure 4. Original Inception module as described in [20].

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-v3: 7 million parameters

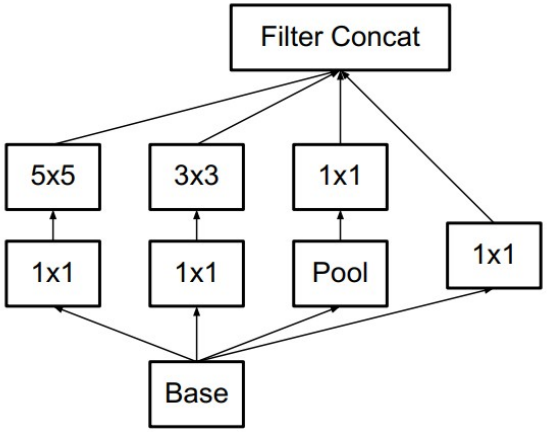
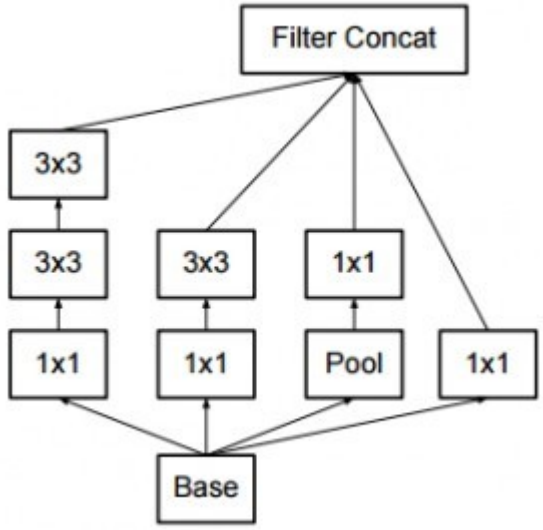
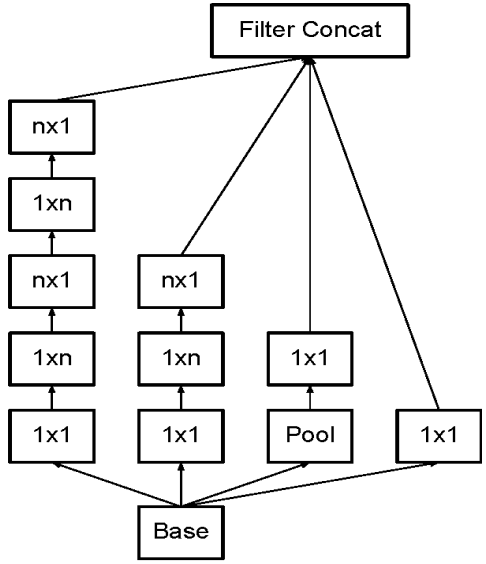


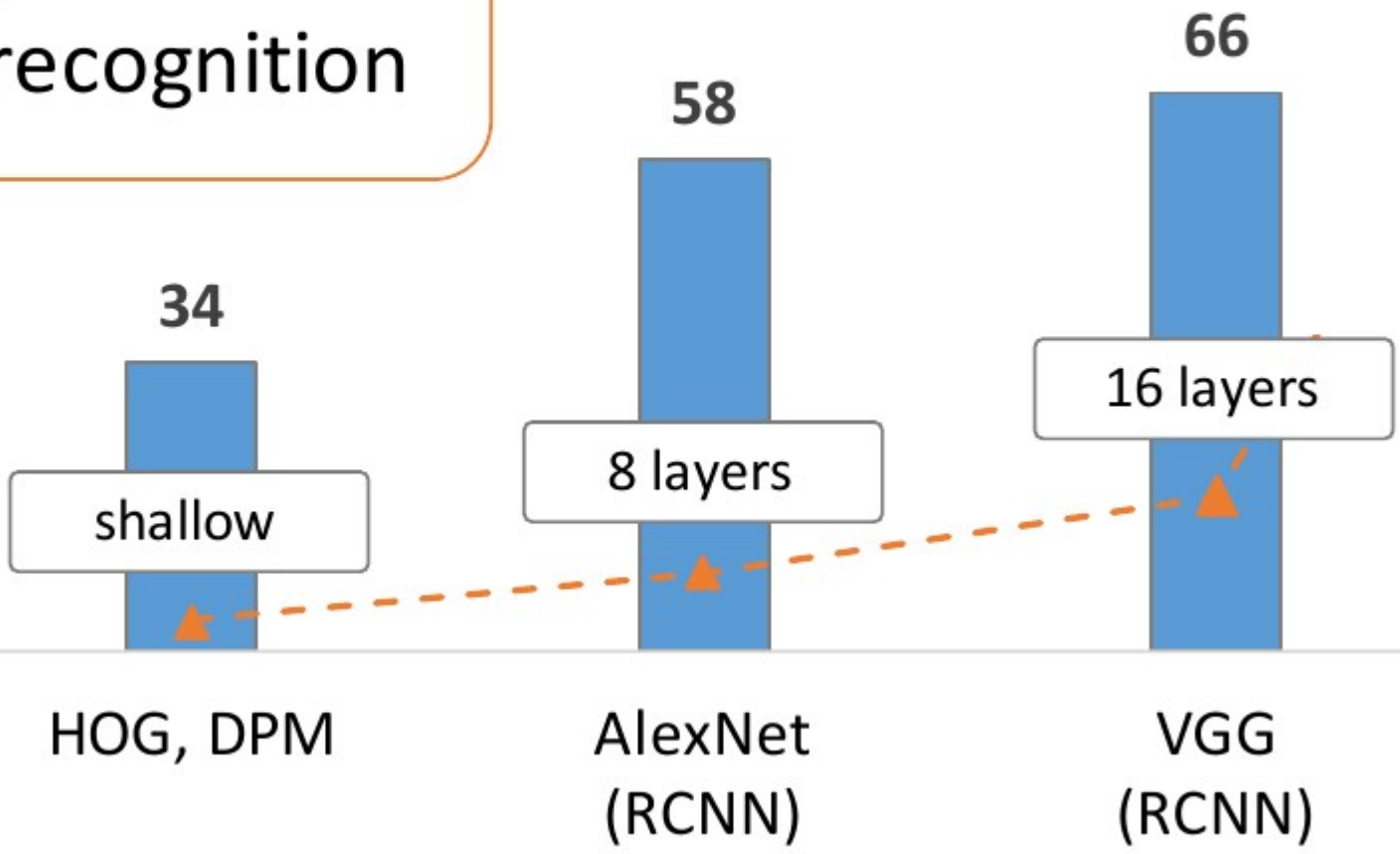
Figure 4. Original Inception module as described in [20].





Revolution of Depth

Engines of visual recognition



How deep could/should a network be?

History of network depth

Before 2012: four layers

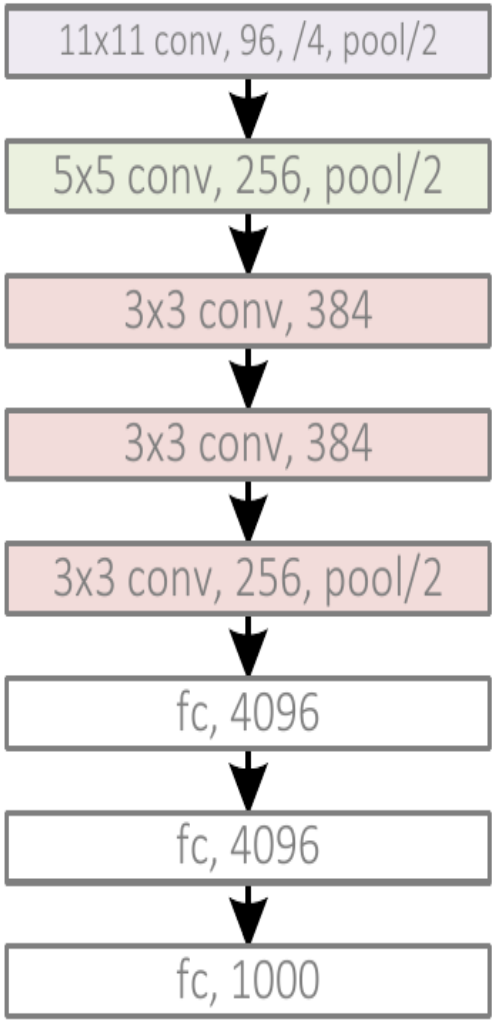
How deep could/should a network be?

History of network depth

Before 2012: four layers

2012: 8layers

AlexNet, 8 layers
(ILSVRC 2012)



How deep could/should a network be?

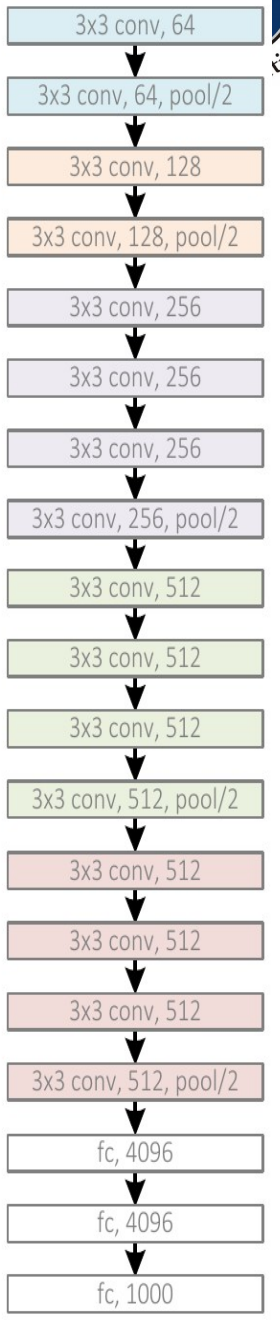
History of network depth

Before 2012: four layer

2012: 8layers

2014: 19 layers

VGG, 19 layers
(ILSVRC 2014)



How deep could/should a network be?

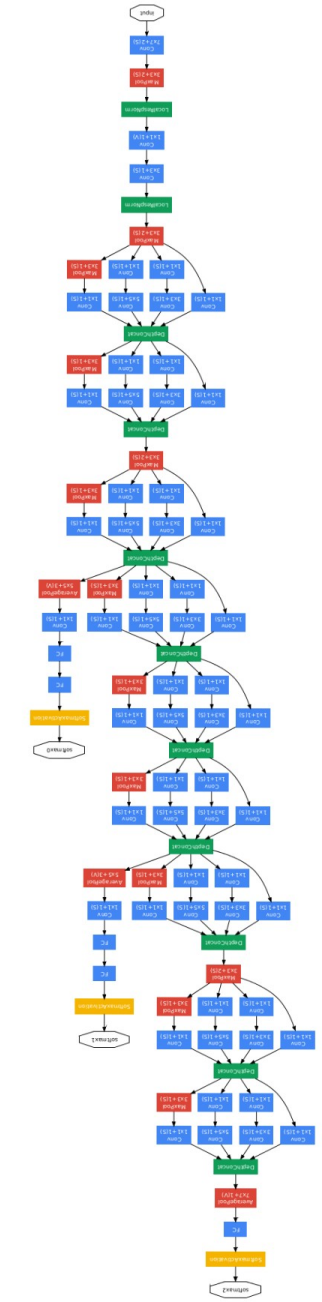
History of network depth

Before 2012: four layer

2012: 8layers

2014: 19 layers

2016: 19-22 layers



How deep could/should a network be?

History of network depth

Before 2012: four layer

2012: 8layers

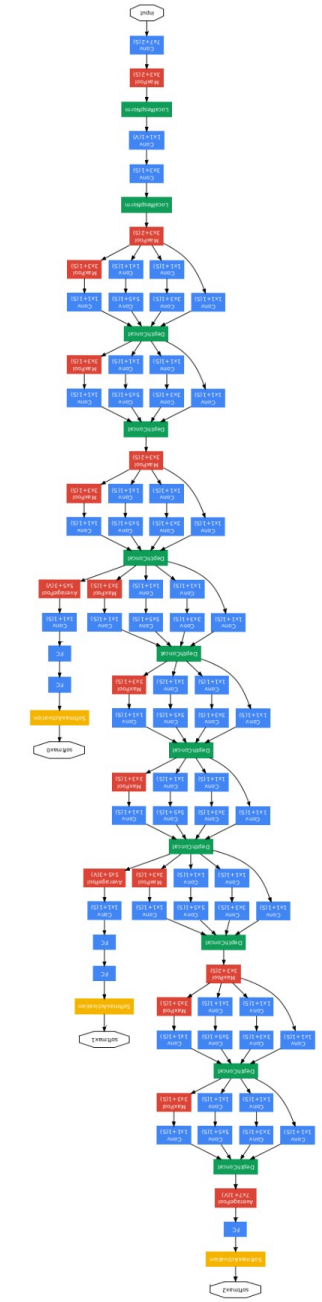
2014: 19 layers

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

2014: 19 layers

2016: 19-22 layers

Deeper network:

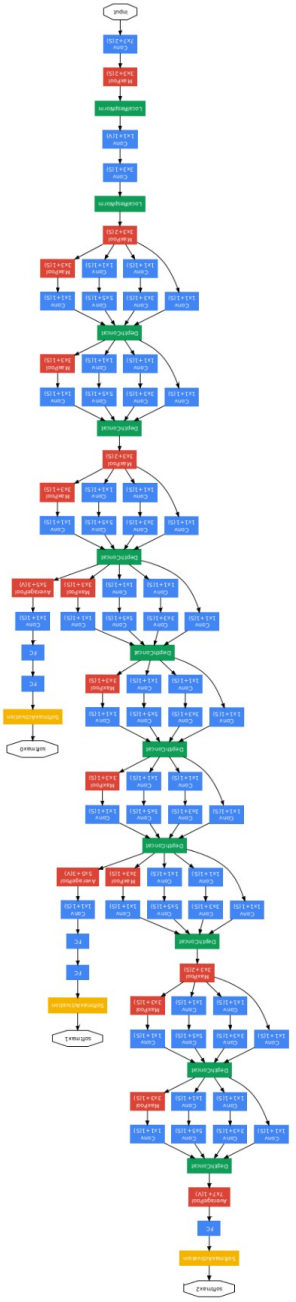
Possibility to approximate more complex functions

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

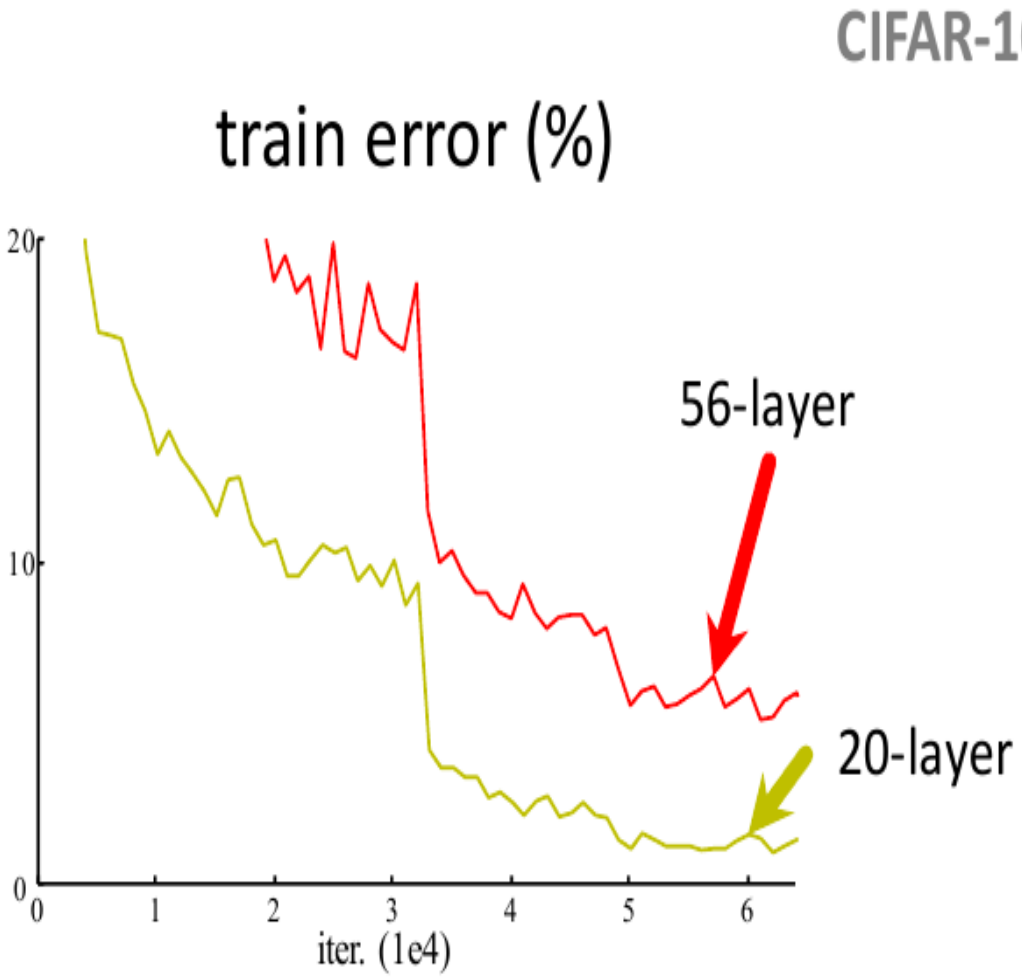
Higher number of parameters (both advantageous and disadvantageous)

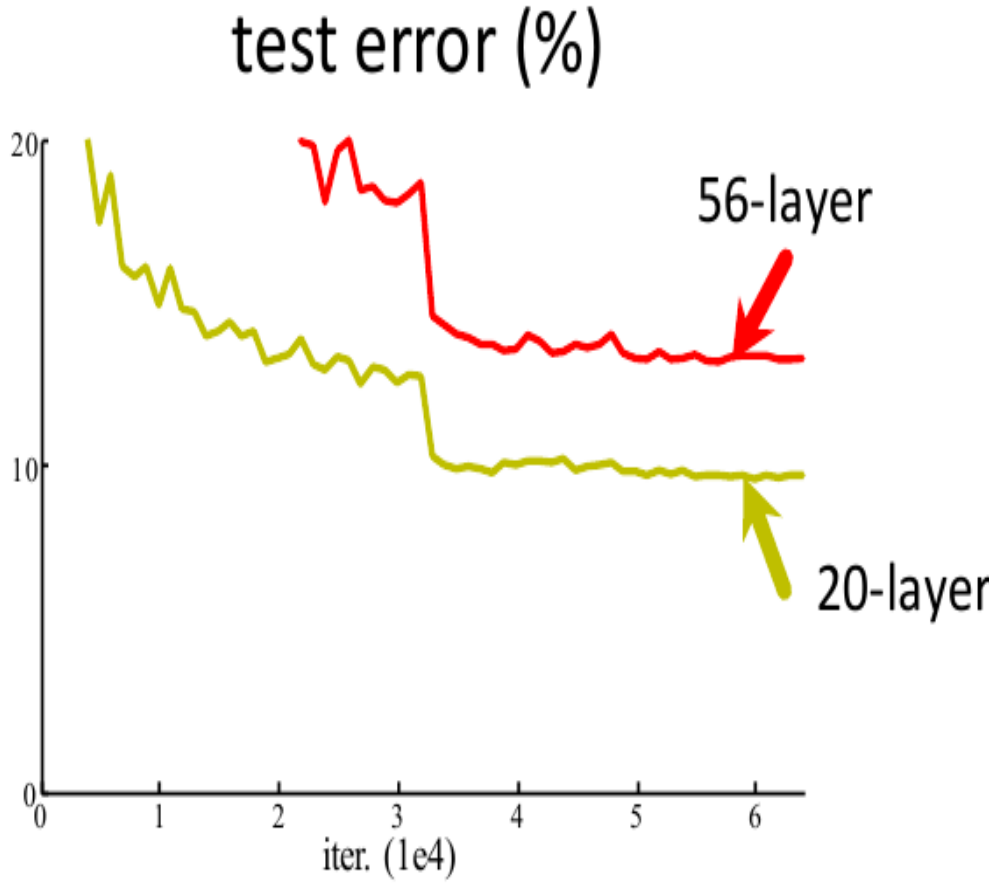
Difficult to train the network

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,





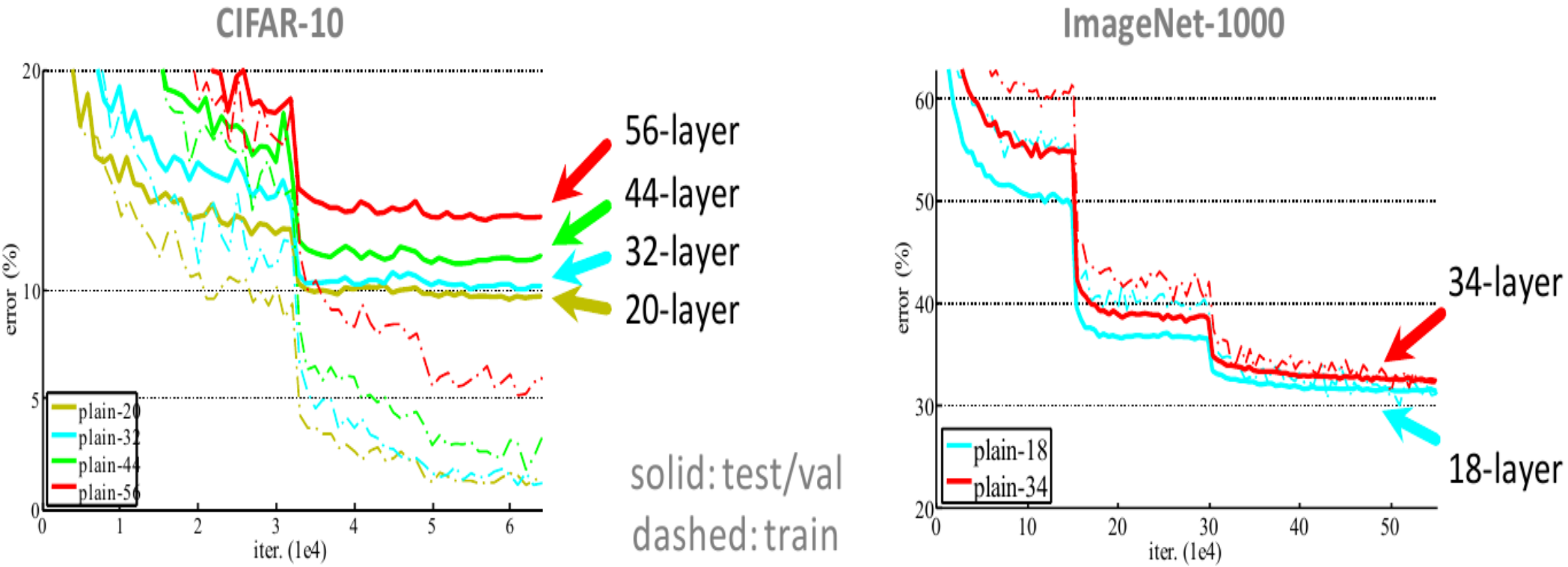
Is a deeper network always better?

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

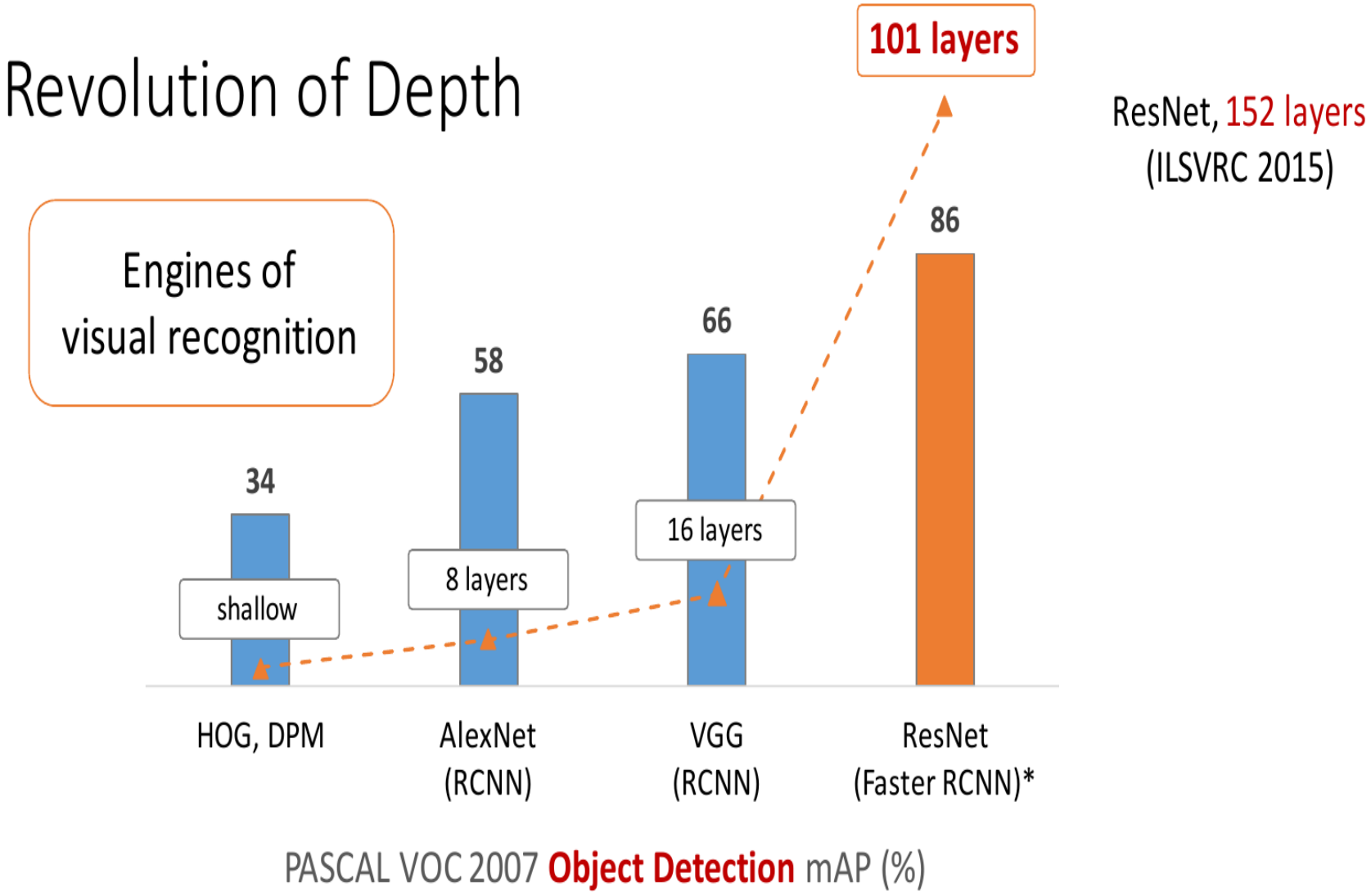
The backpropagated error will be smaller than the floating point accuracy limit.

The gradient will be disappear. The information will not pass the first layers, because there will be random noises on the weights, and they will not be trained.



How deep could a network be?

Residual networks provide an answer to these questions

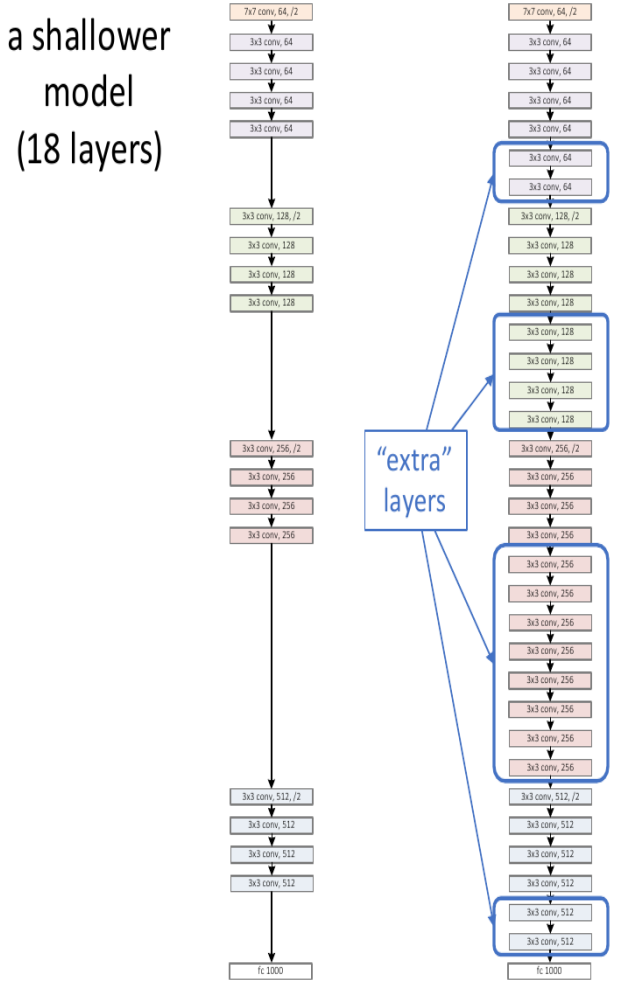


How could we create deeper networks?

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?

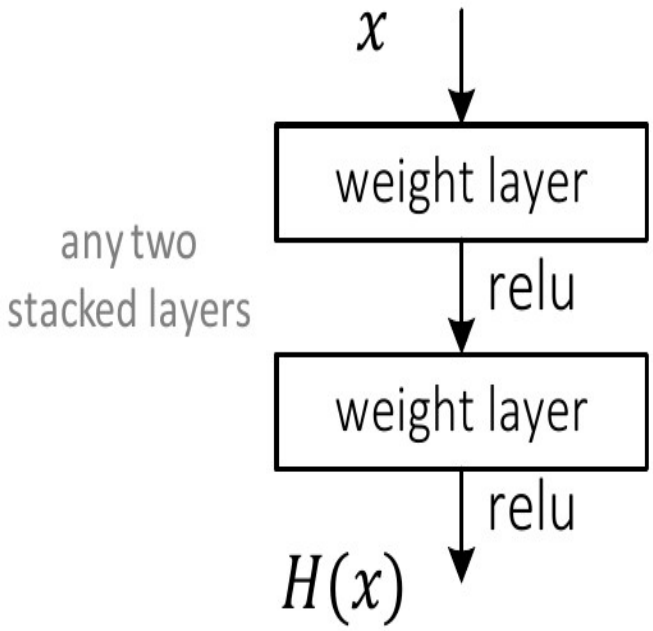
A deeper network always have the potential to perform better, but training becomes difficult

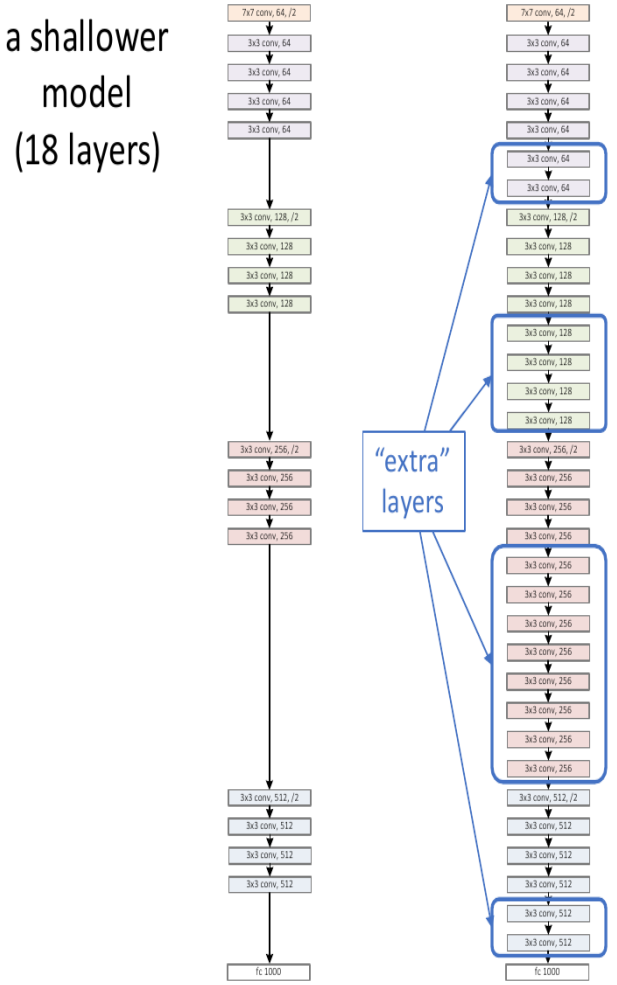
How could we ensure that additional layers will not decrease accuracy (might even increase it)?

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?

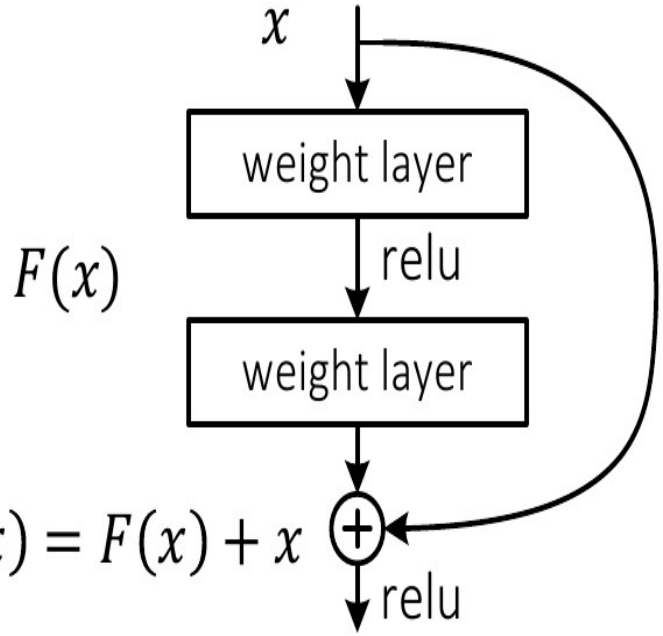
A deeper network always have the potential to perform better, but training becomes difficult

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

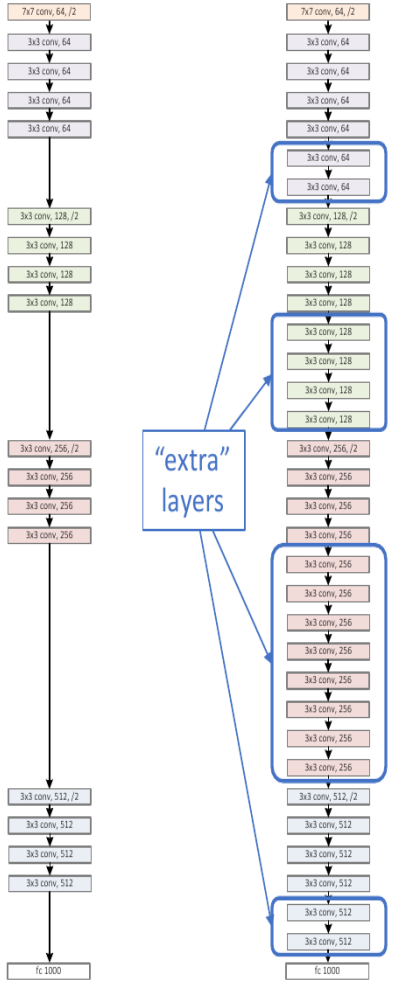
The trick is to use residual connection and as a starting point F(x) could be zero, and H(x) becomes the dientity mapping

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)

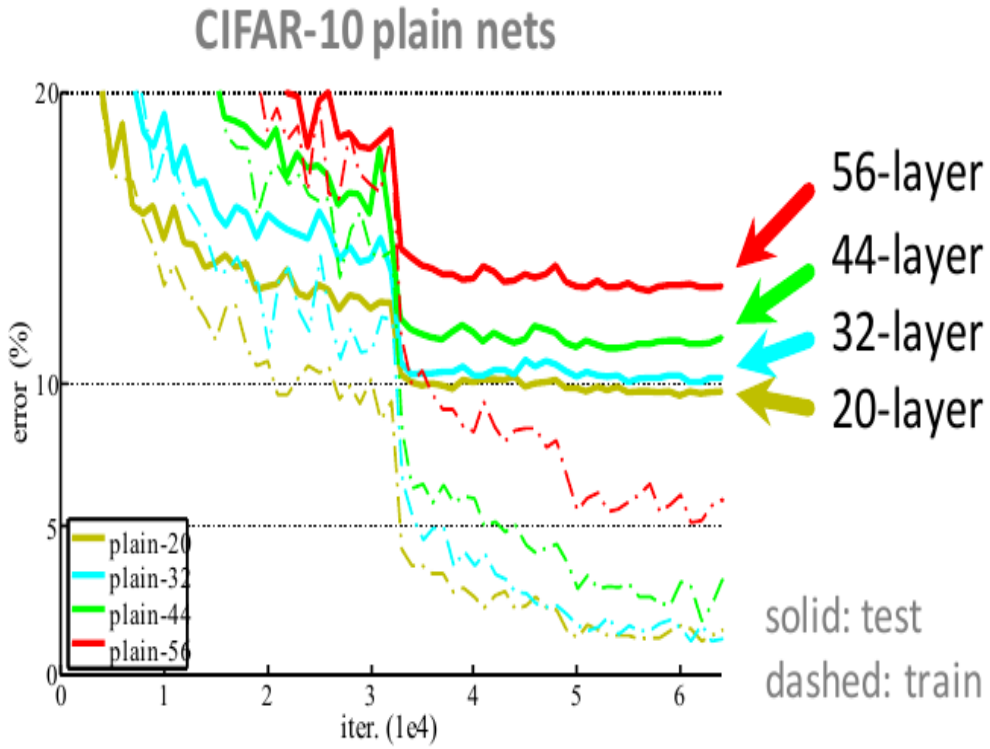


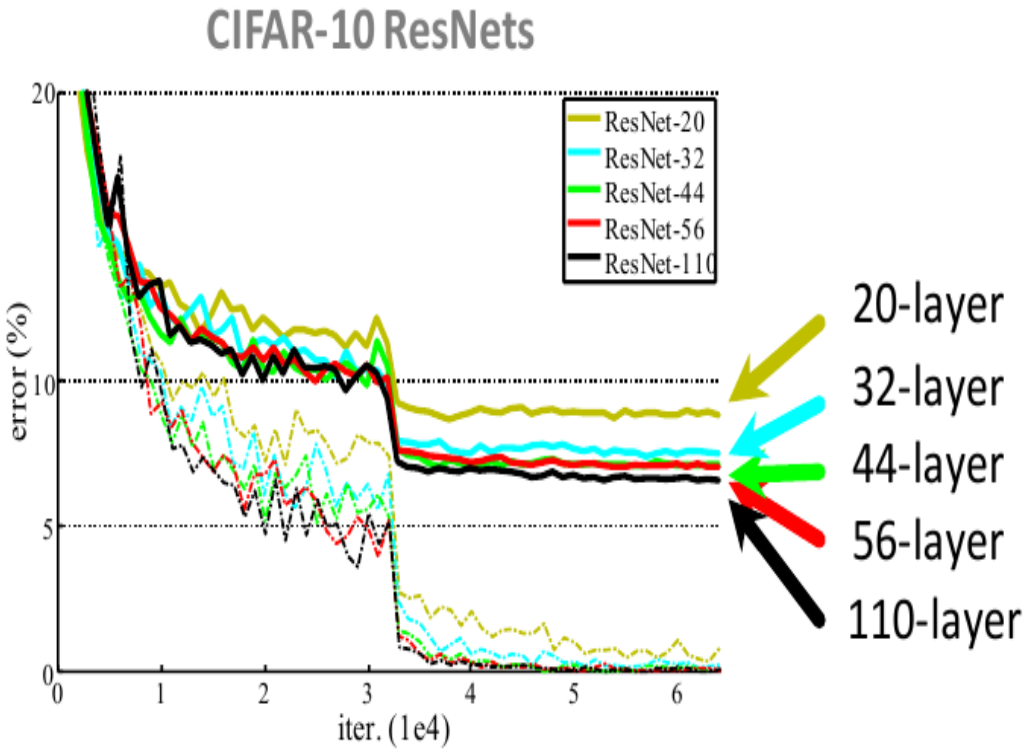
a shallower model (18 layers)



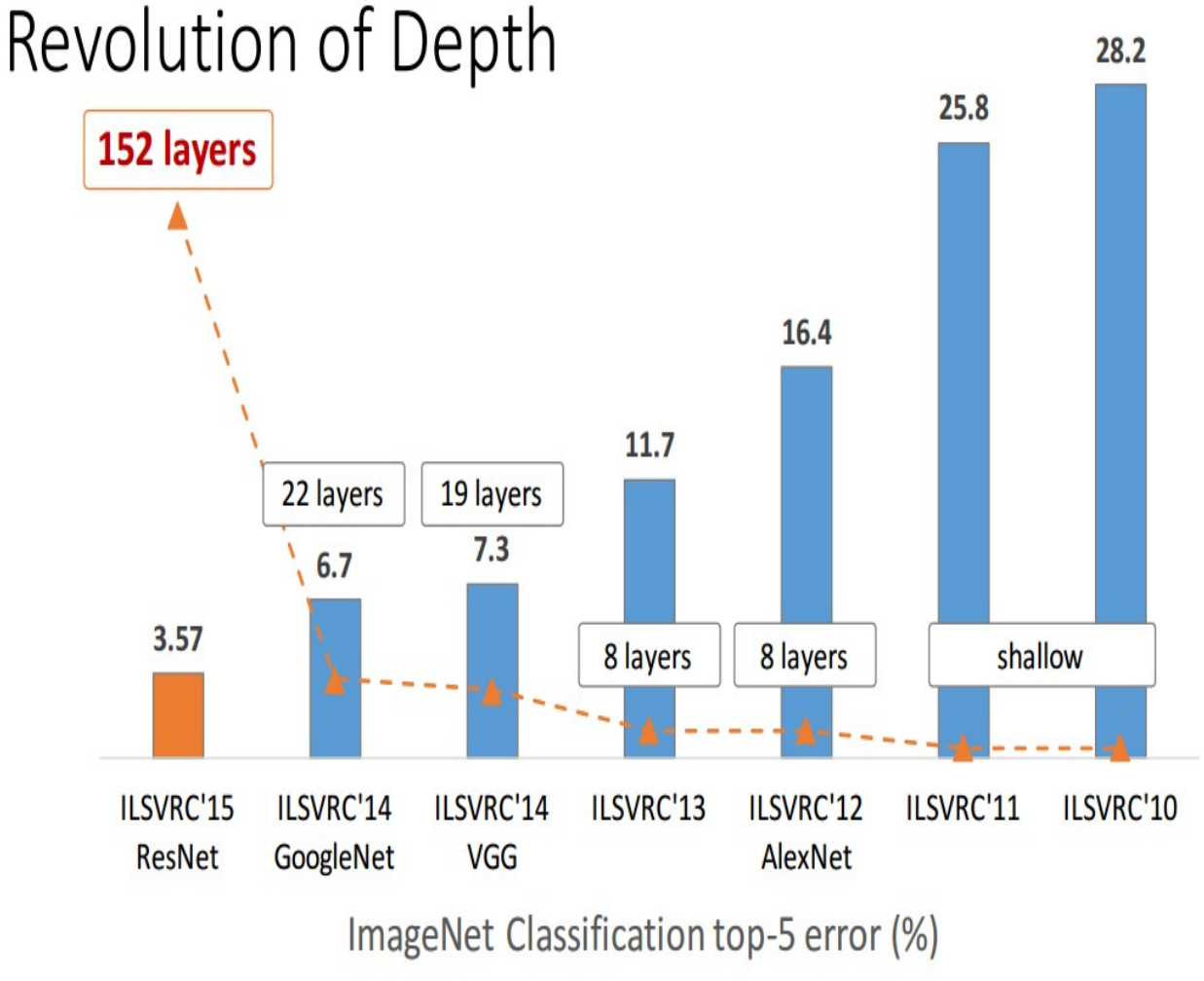
Residual networks

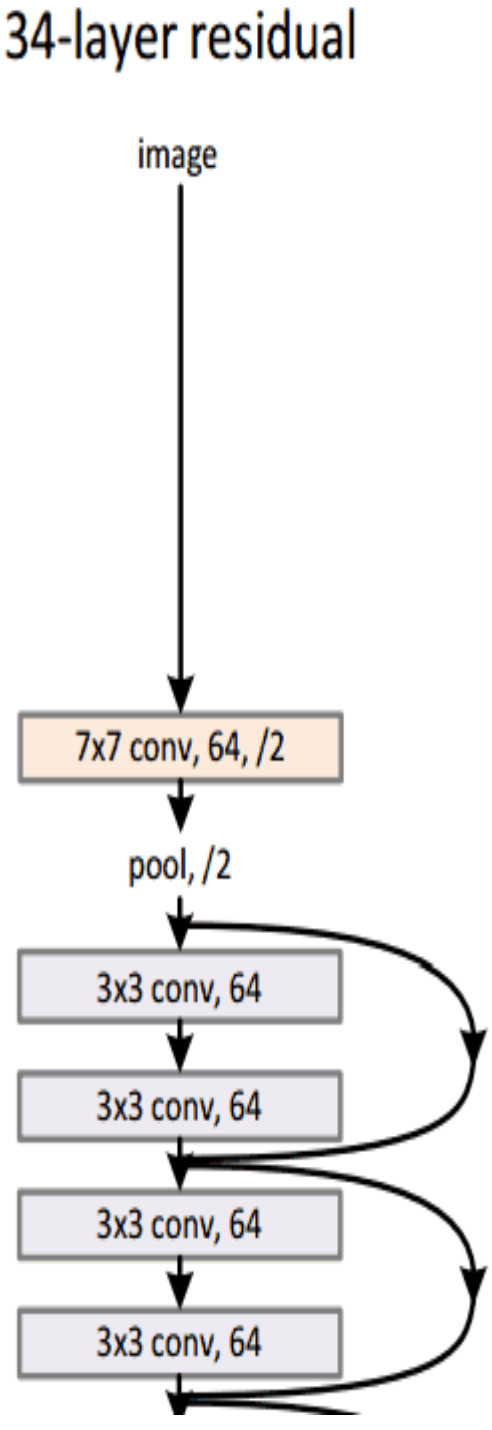
Results: Deeper residual networks result higher accuracy





Results with ResNets





Results with ResNets

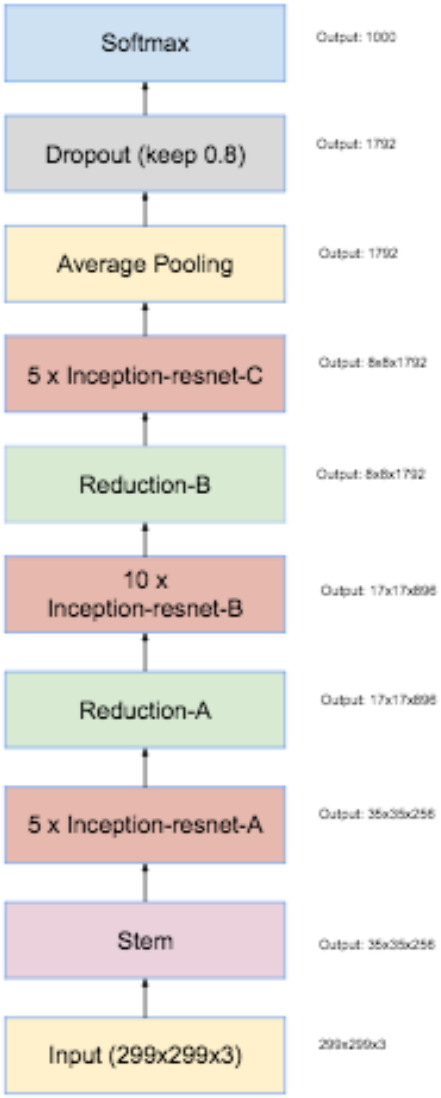
ResNets had the lowest error rate at most competitions since 2015

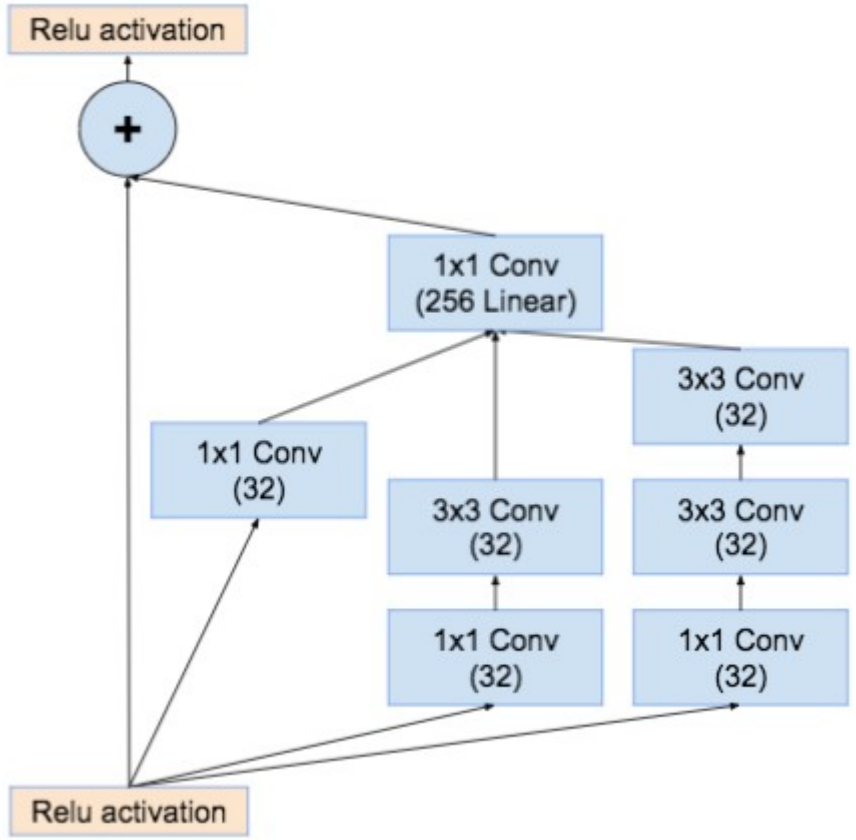
1st places in all five main tracks

- ImageNet Classification: “Ultra-deep” 152-layer nets
- ImageNet Detection: 16% better than2nd
- ImageNet Localization: 27% better than2nd
- COCO Detection: 11% better than2nd
- COCO Segmentation: 12% better than2nd

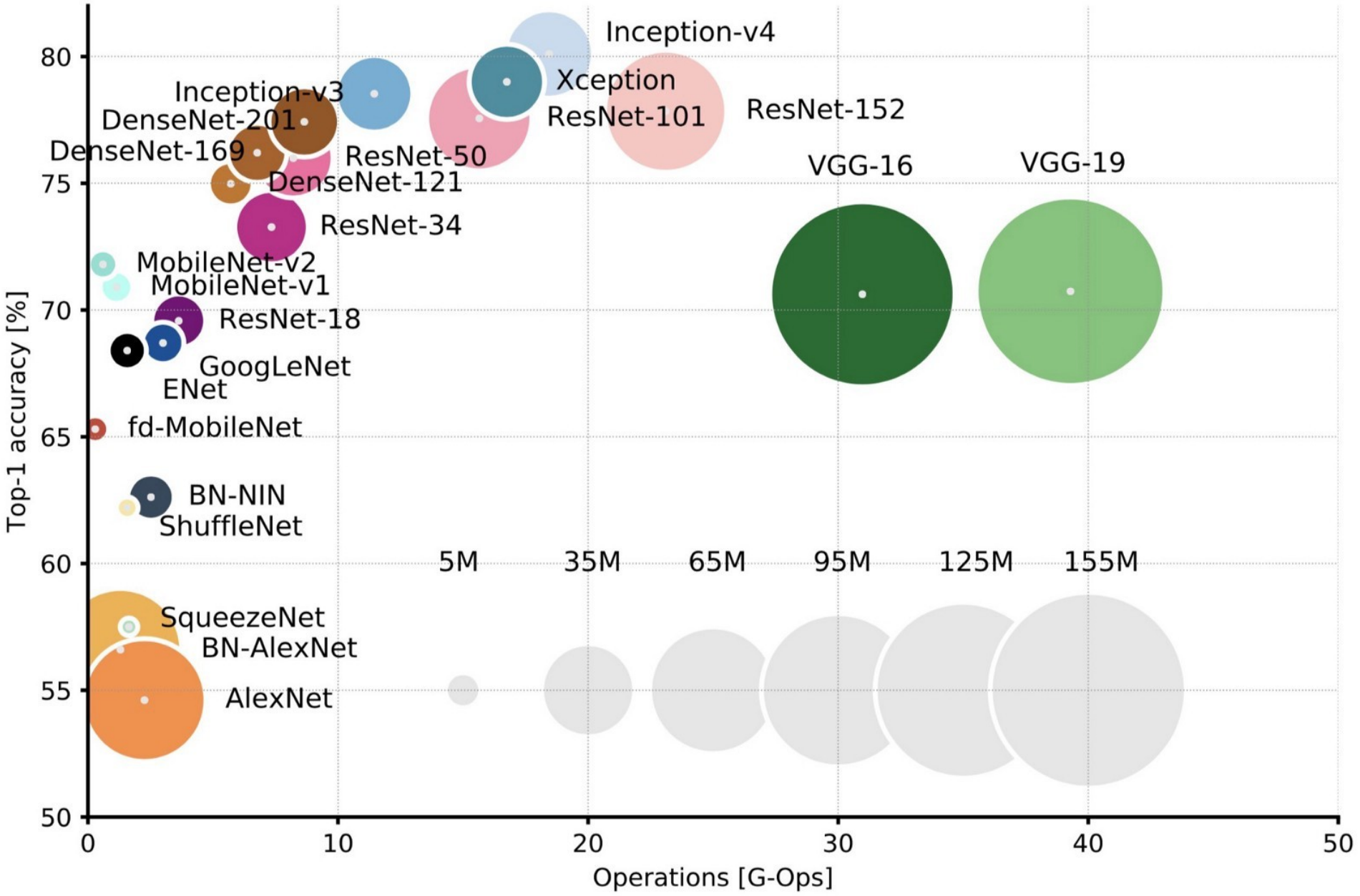
GoogleNet Inception v4

Inception architecture applied to residual networks





Efficiency of Neural Networks



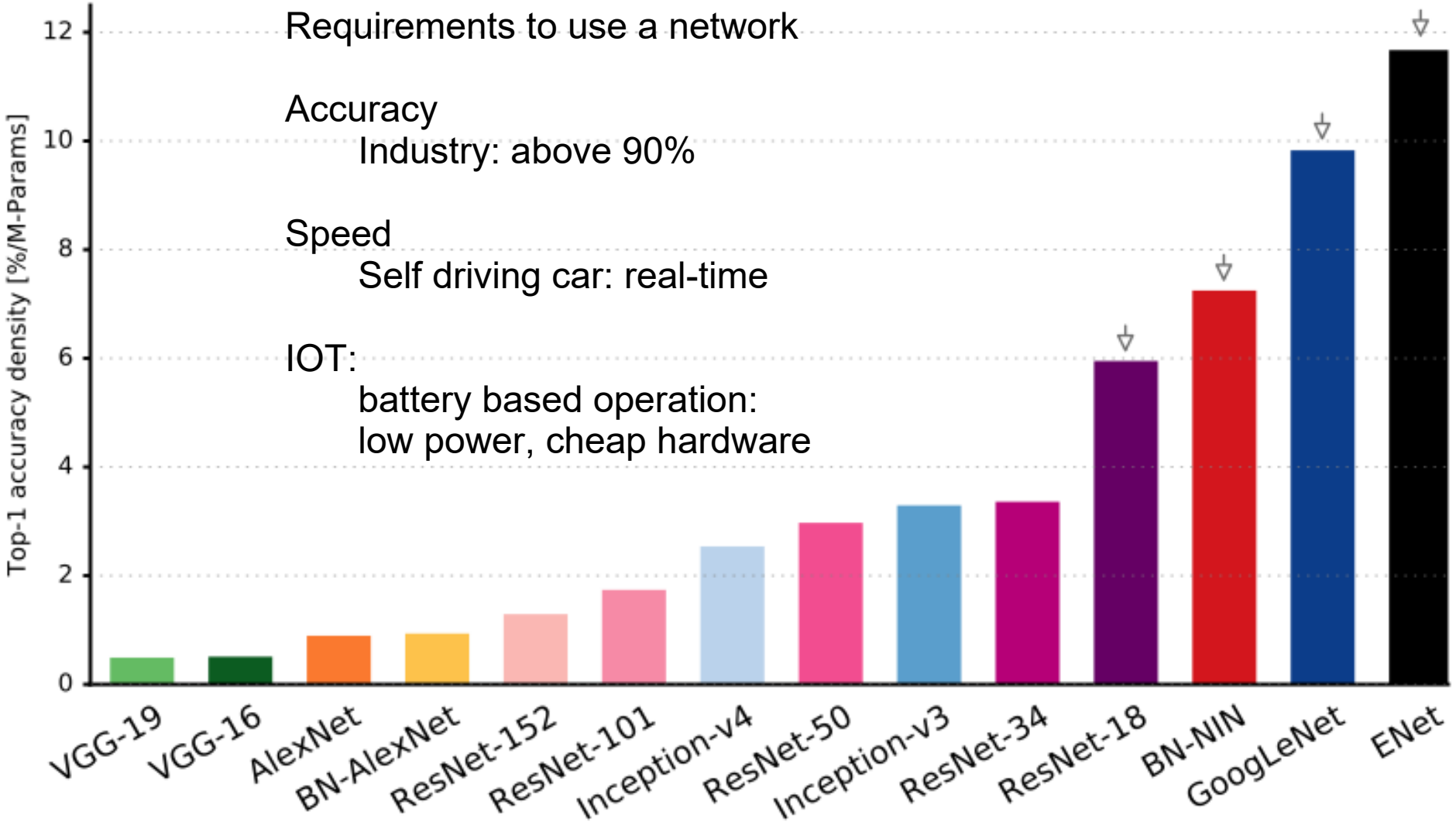
Efficiency of Neural Networks

Requirements to use a network

Accuracy
- Industry: above 90%

Speed
- Self driving car: real-time

IOT:
- battery based operation:
  low power, cheap hardware



MobileNet

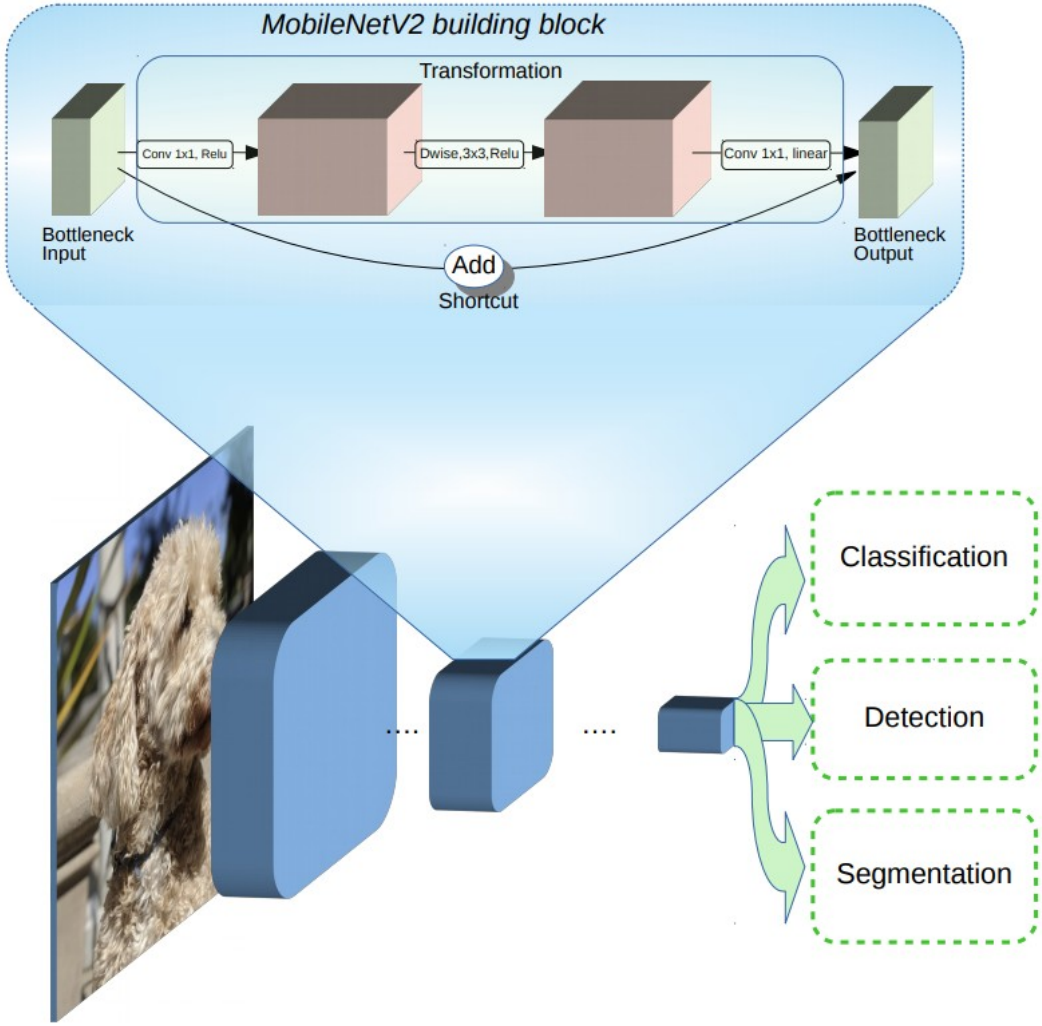
Scaling in feature map depths.

In this arhcitecture feature depths are squeezed before each operation

In a squeezed architecture we will use downscale the 128 feature maps to 16, using a linear combination (1x1 convolution)

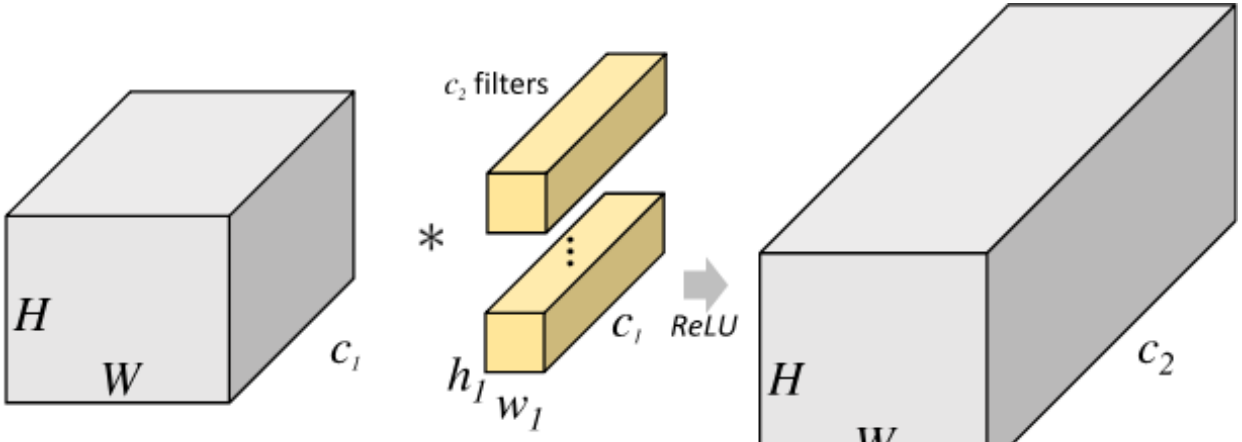
After the 3x3 covolutions, we expanded back to 128 layers by 1x1 convolution again

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



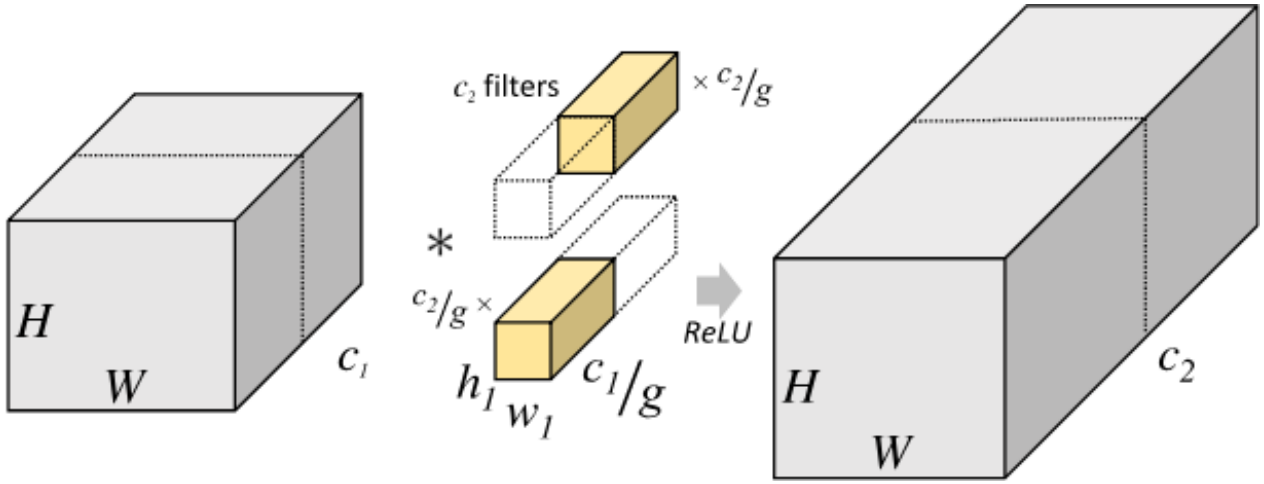
ResNext

- **Group convolution**:
  - Dividing the feature mapes into two groups, and apply the convolutions to each groups separately
  - The number of convolutions will be halved

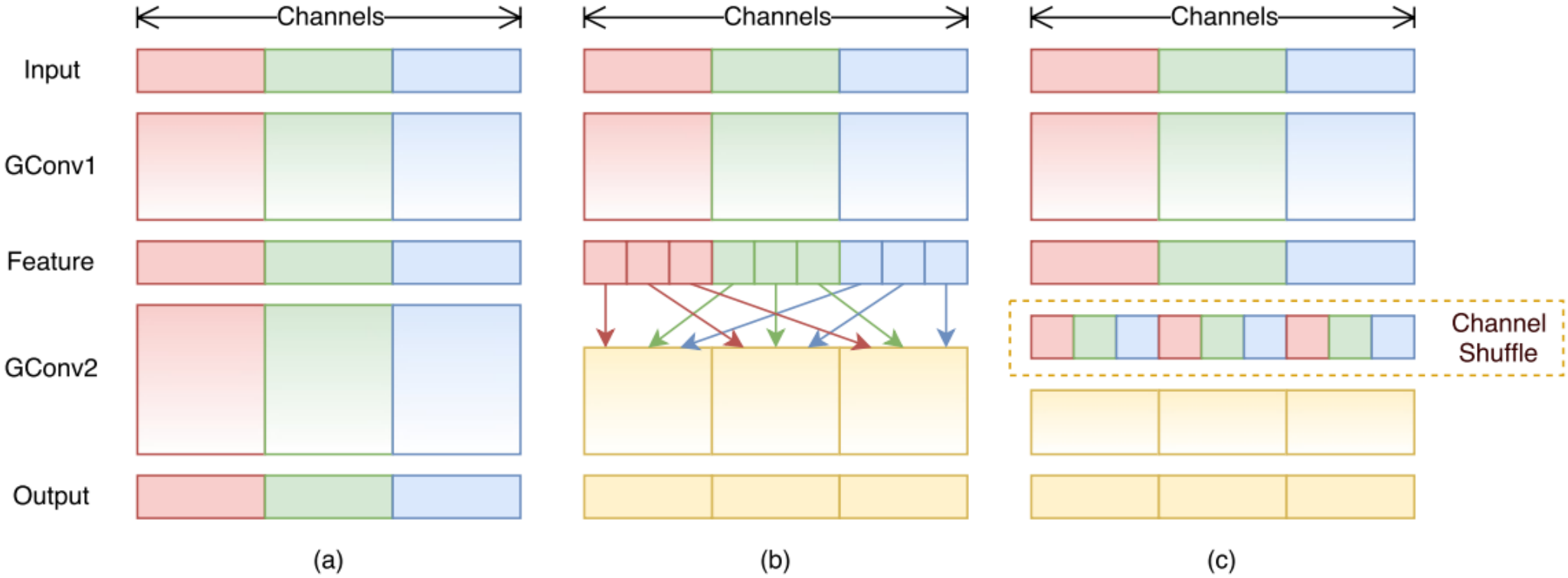


- normal convolution block:
-
- c_1 inputs, c_1 outputs
  - $c_1c_2$number of kernels

- group convolution block:
-
- 2x(c_1/2) inputs, 2x(c_2/2) output
  - 2x(c_1/2 c_2/2) = c_1c_2/2 number of kernels



ShuffleNet

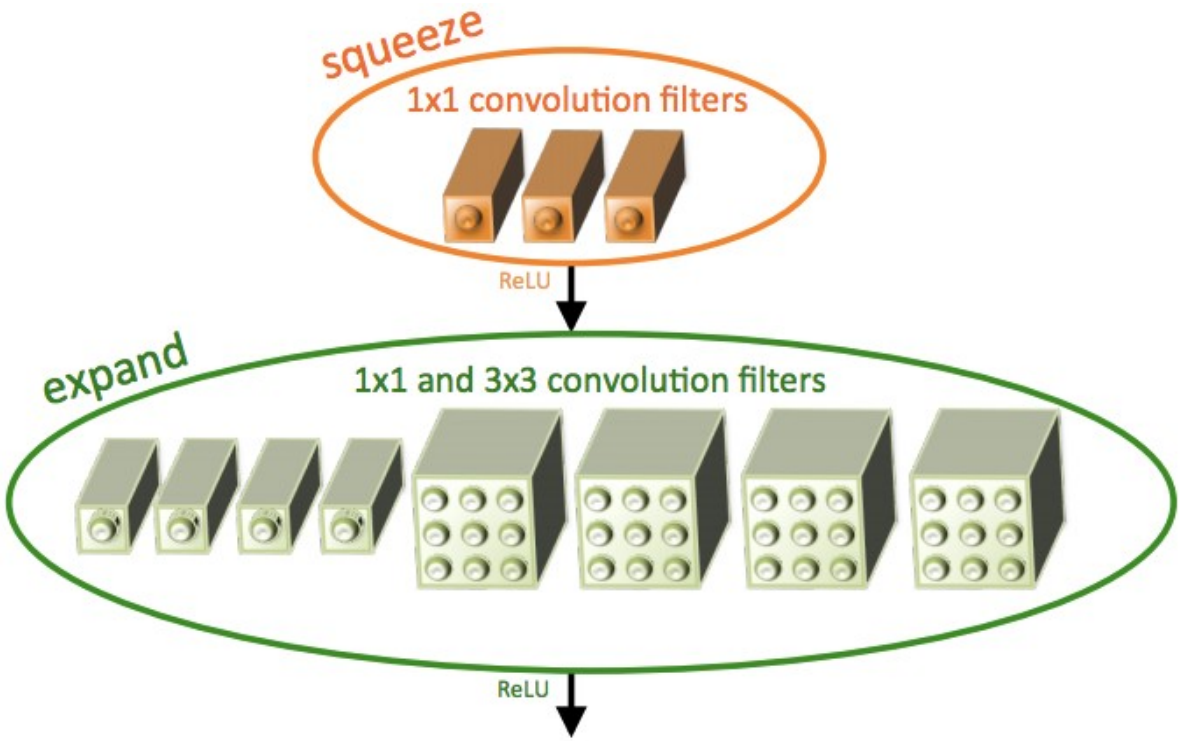

Channels
Channels
Channels
Input
GConv1
Feature
GConv2
Output
Channel Shuffle
(a)
(b)
(c)

SqueezeNet

In this arhcitecture depths are squeezed before each operation

The expand is done by the concatenation of the 1x1 and the 3x3 convolutions.

Advantage: the expand layer is saved.



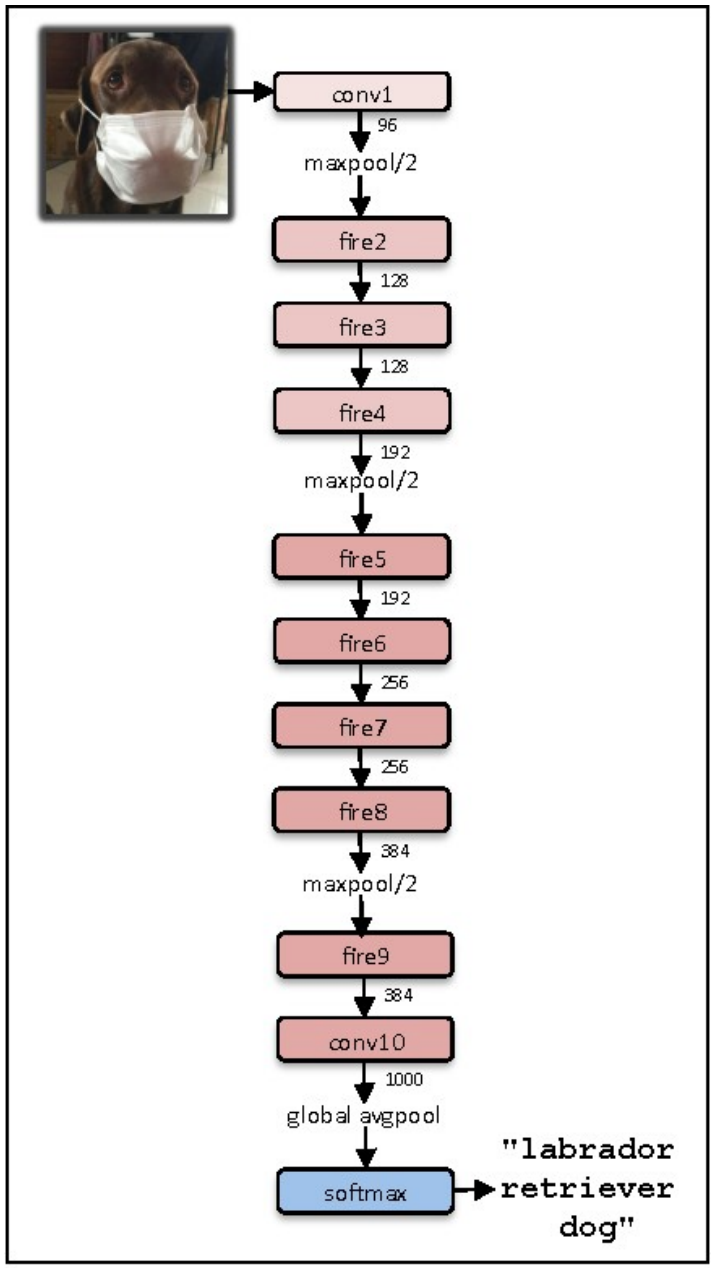
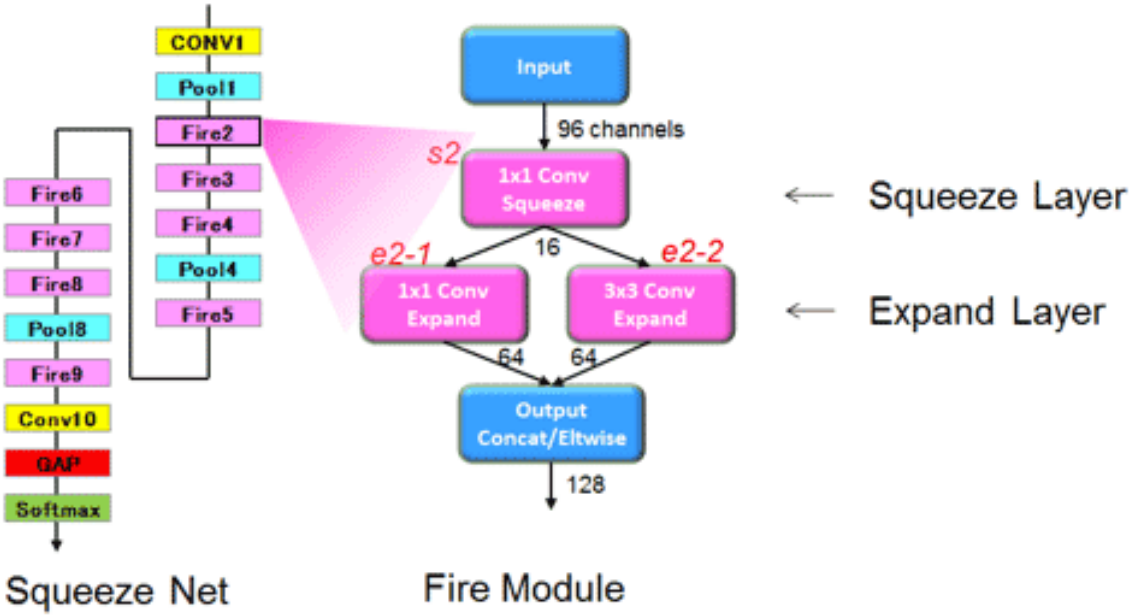


Figure 2. The SqueezeNet architecture

SqueezeNet

In this arhcitecture depths are squeezed before each operation



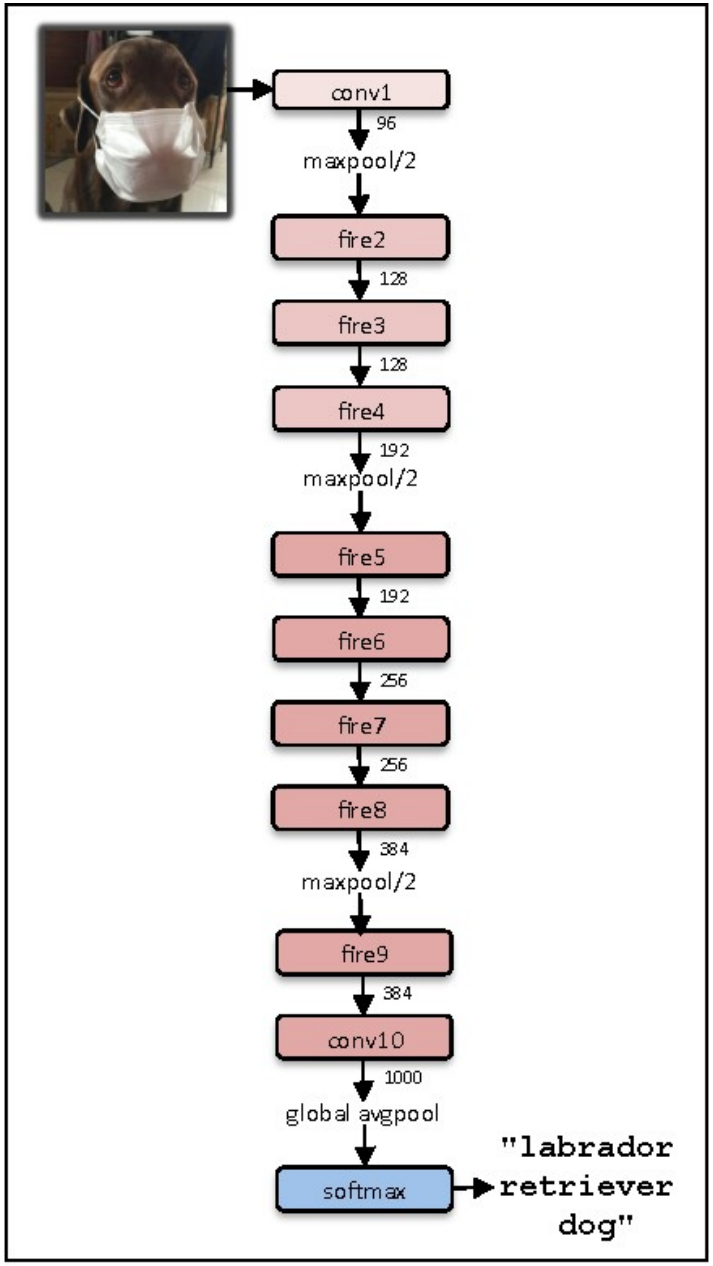


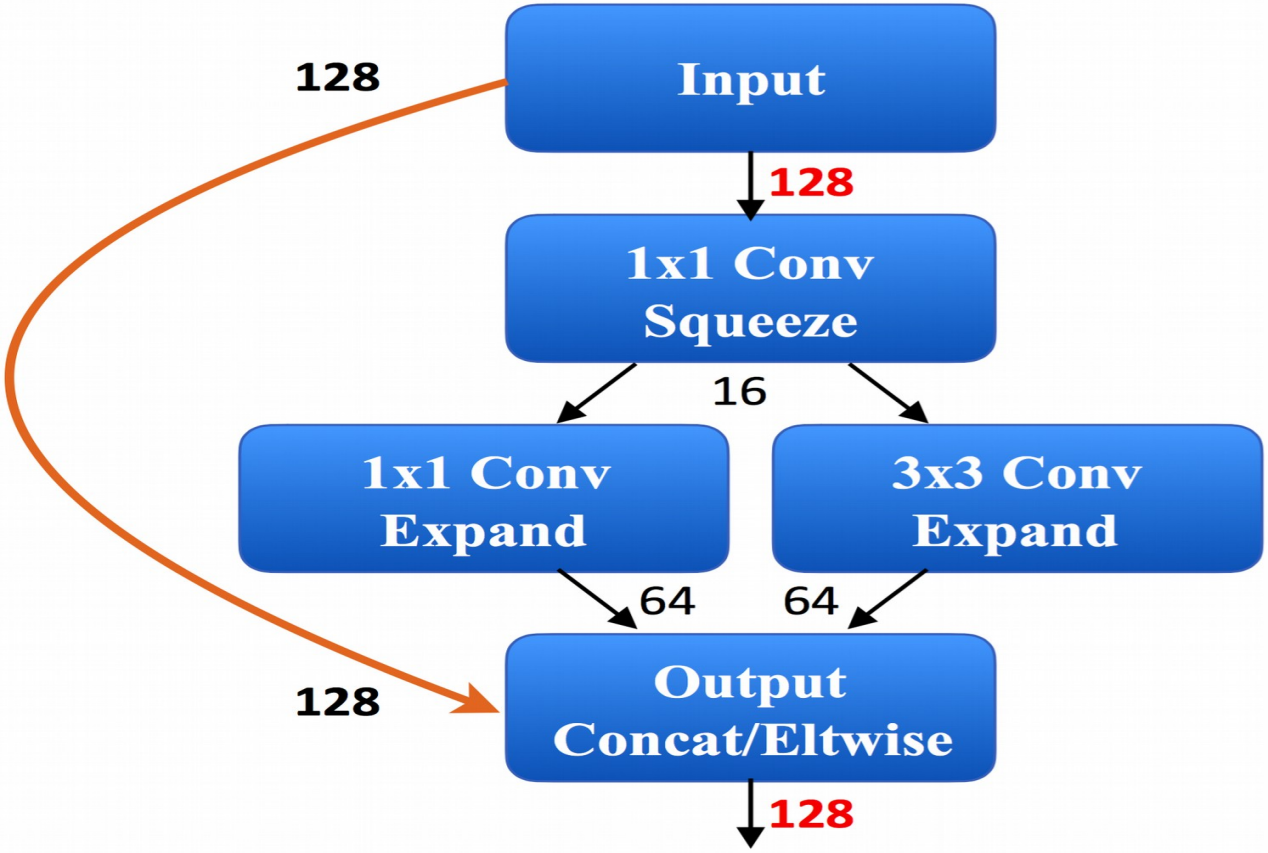
Figure 2. The SqueezeNet architecture

SqueezeNext

In this arhcitecture depths are squeezed before each operation

In a SqueezeNext architecture we will use a linear approximatine 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

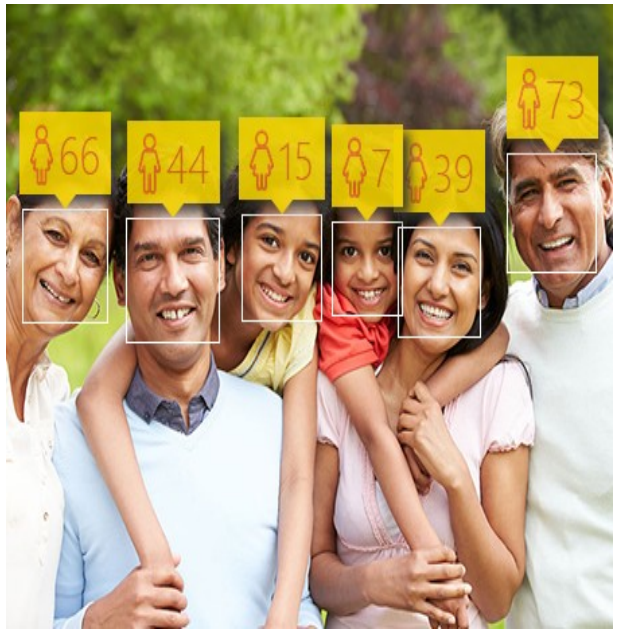
Age estimation

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:

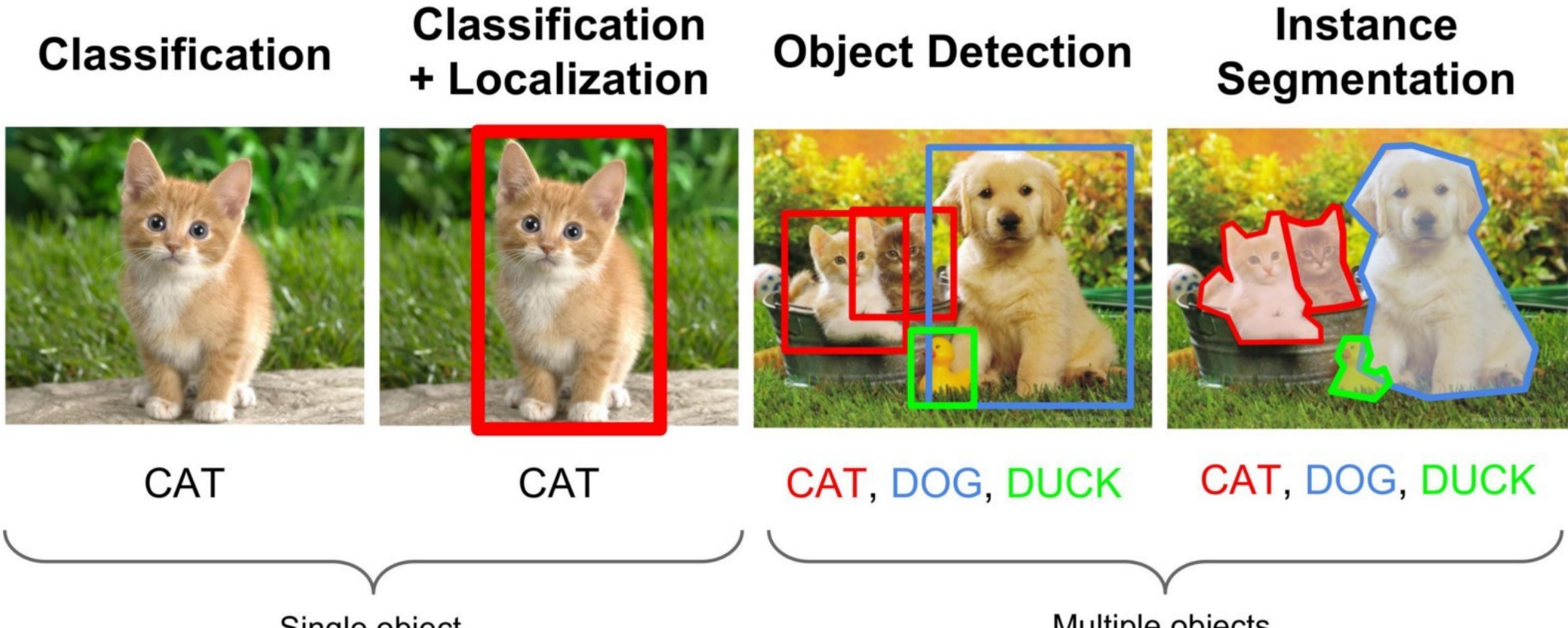


Neural networks for regression

Multiple object detection on a single image

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

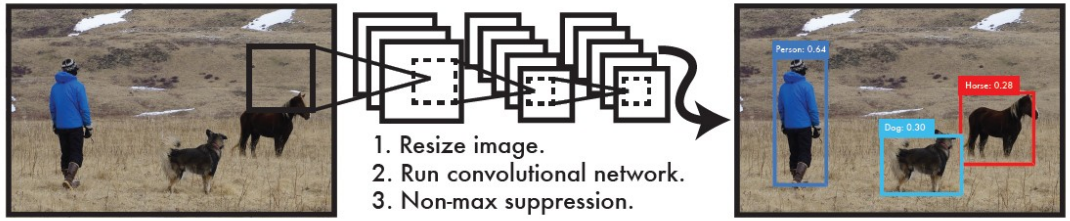
How could we detect objects in general, when the number of objects is unknow



Traditional method

Sliding window over the image

We might have objects in different scales



Slidign windowds in different scales, aspect ratios

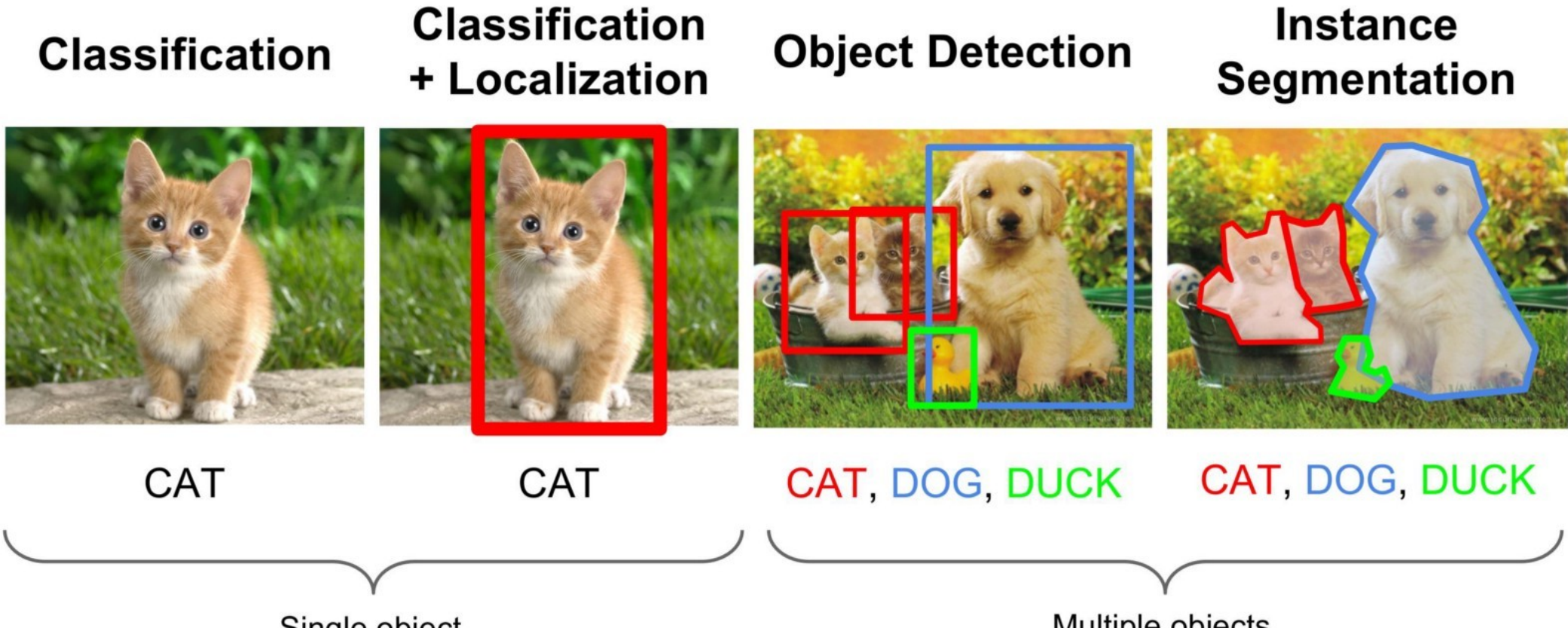
Resutls a heat map → detect the objects: non-maximum suppression

Object detection as regression

RCNN

Single Shot Object Detector (SSD) (2016 March)

You Only Look Once YOLO (2016 May)



R-CNN

Region proposal CNN network

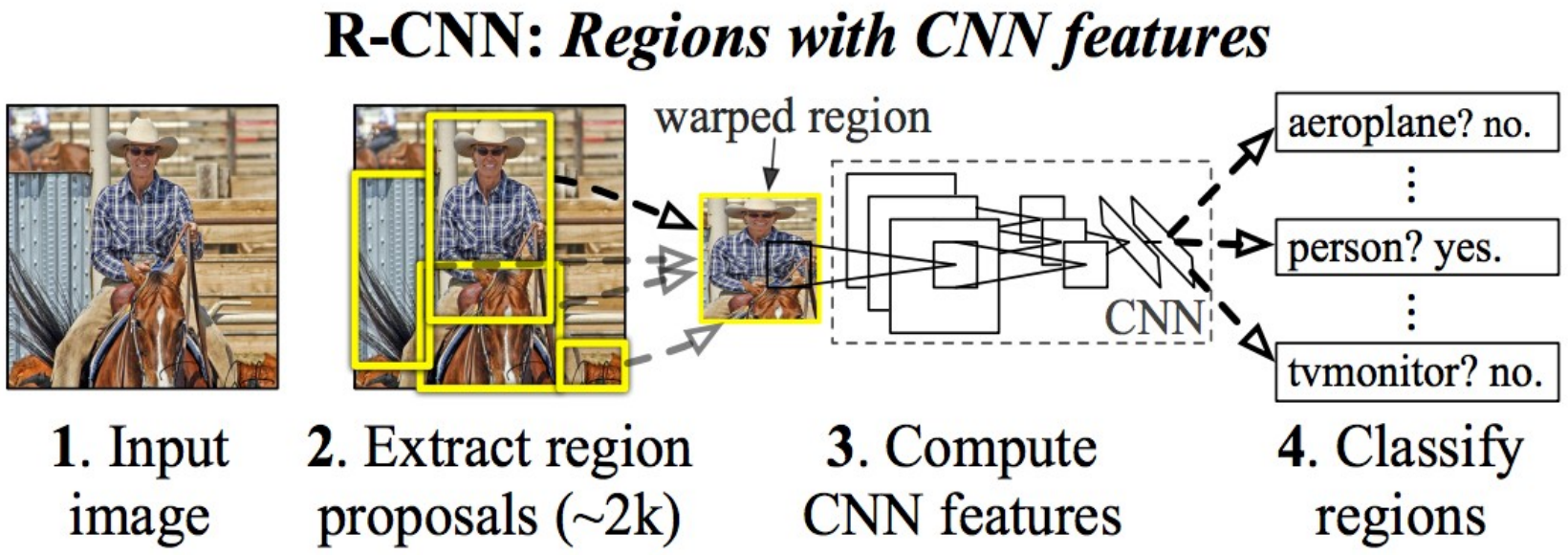
Separate the problem of object detection and calssification

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



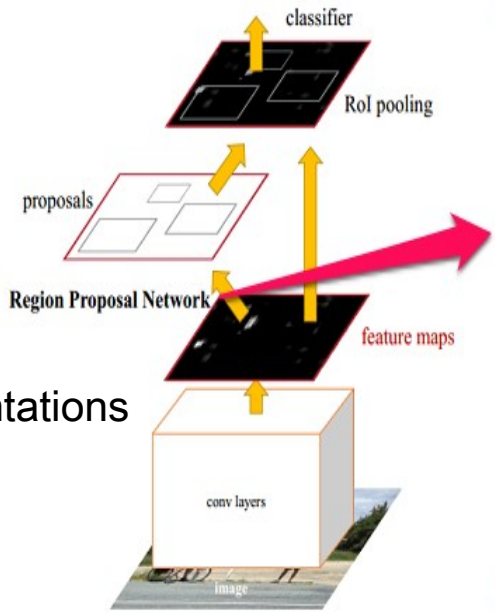
Faster R-CNN

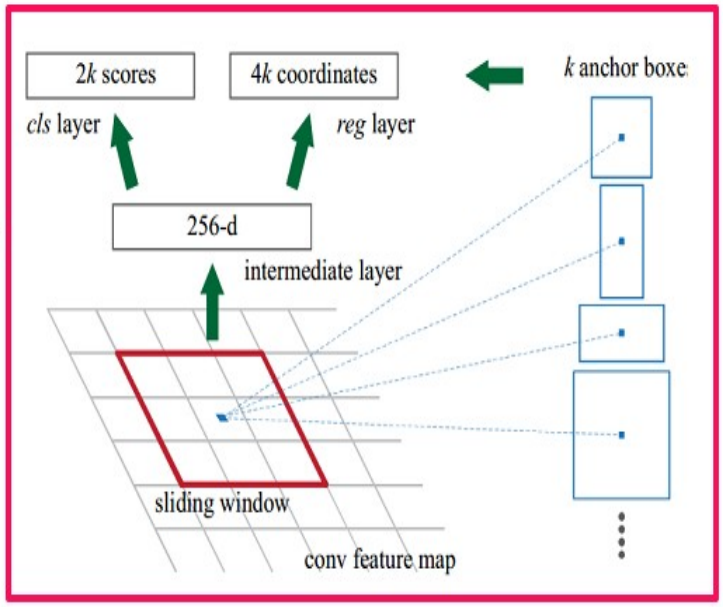
Region proposal from a network

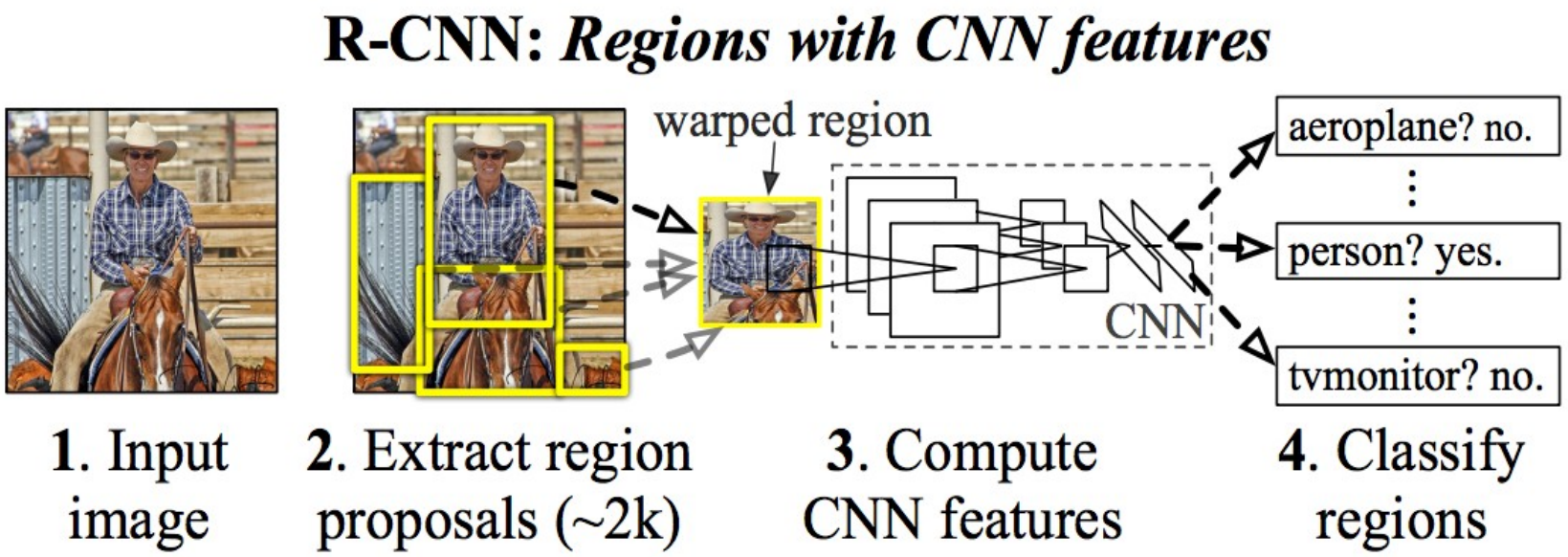
Step 3 and 4 are standard CNN implementations

Extra layers for region proposals

Possible region refinement at the end







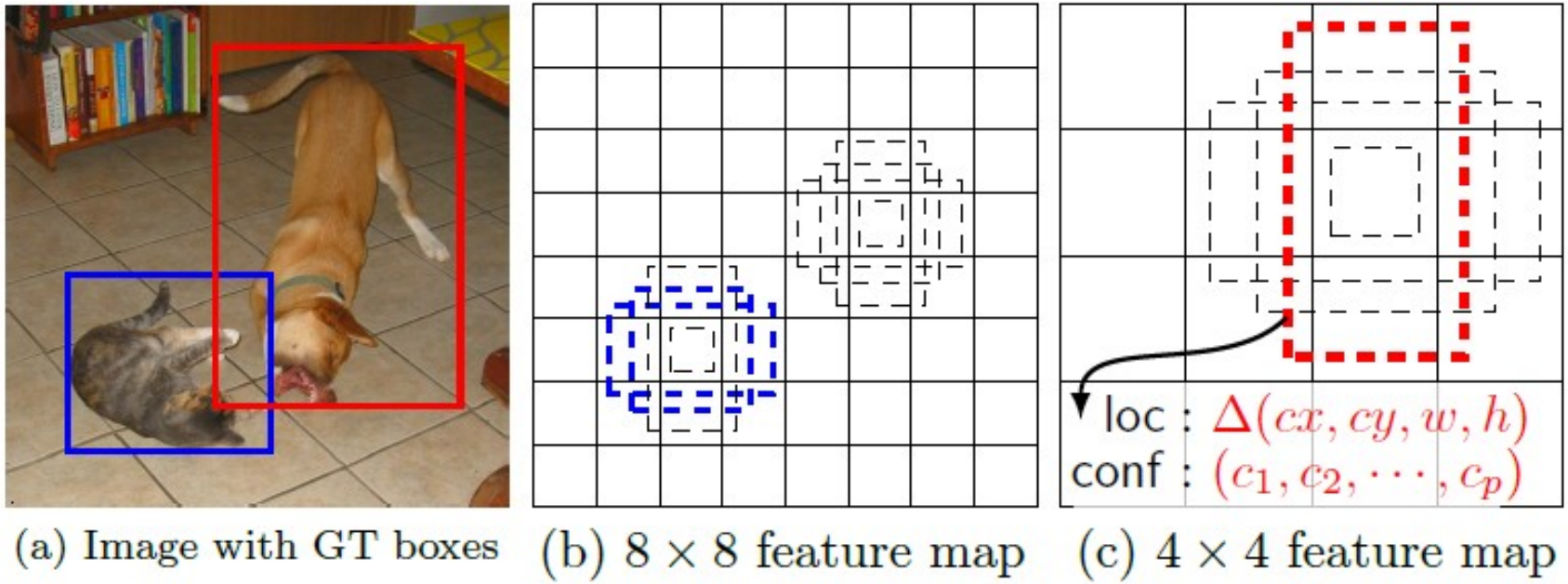
SSD

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 a fixed size bounding boxes. (Each feature map represents a certain box size.

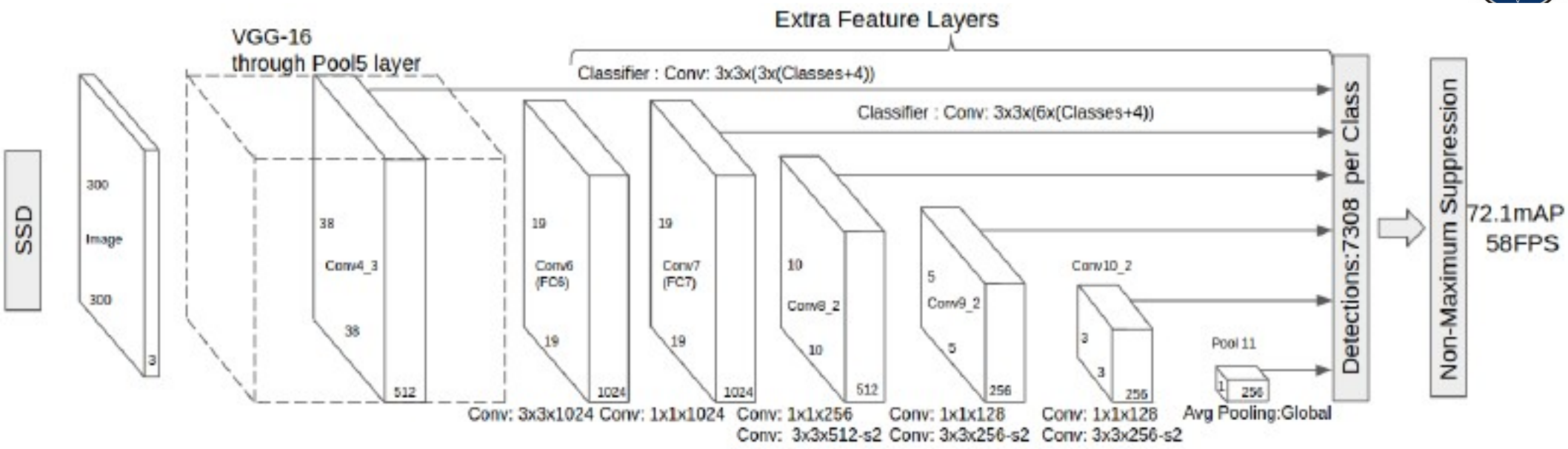
A high pixel value represent high probability of the centerpoint of a detected object.

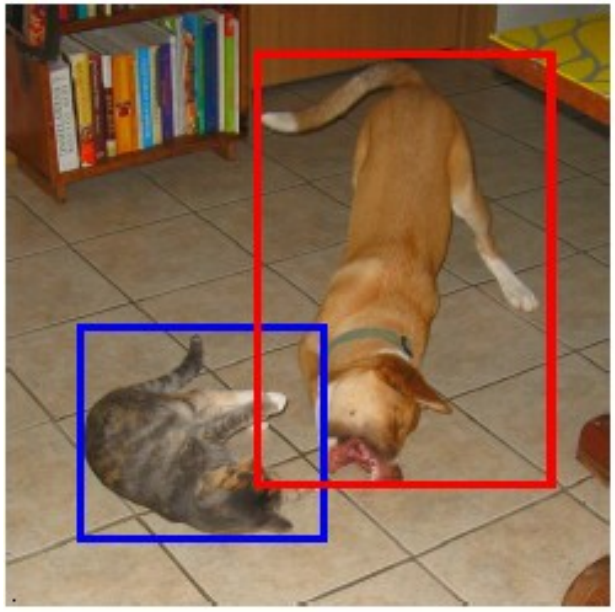


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

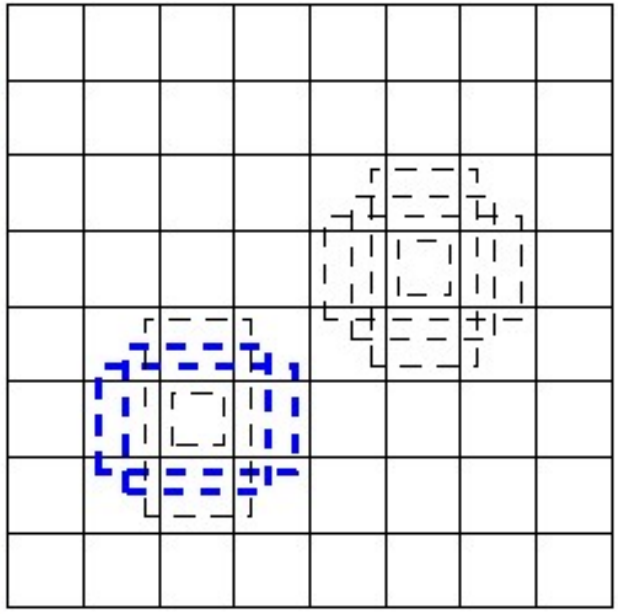
Problem: Unlike at R-CNN, the boundix boxes have fixed scale and positions, no fine turning in the last step.

SSD arhcitecture

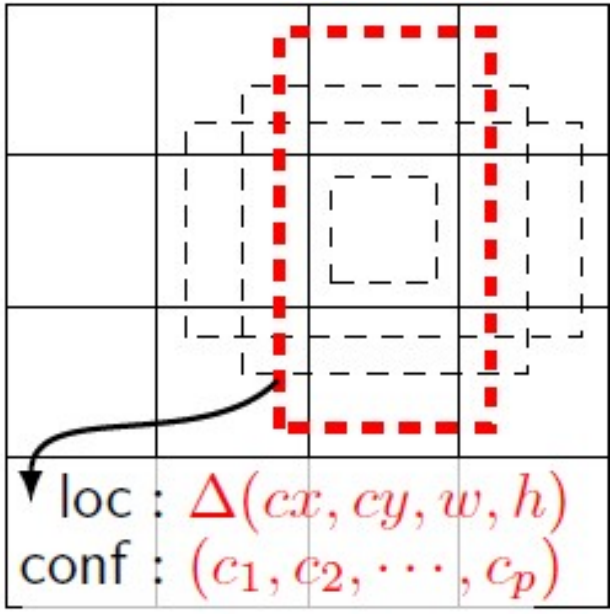




(a) Image with GT boxes

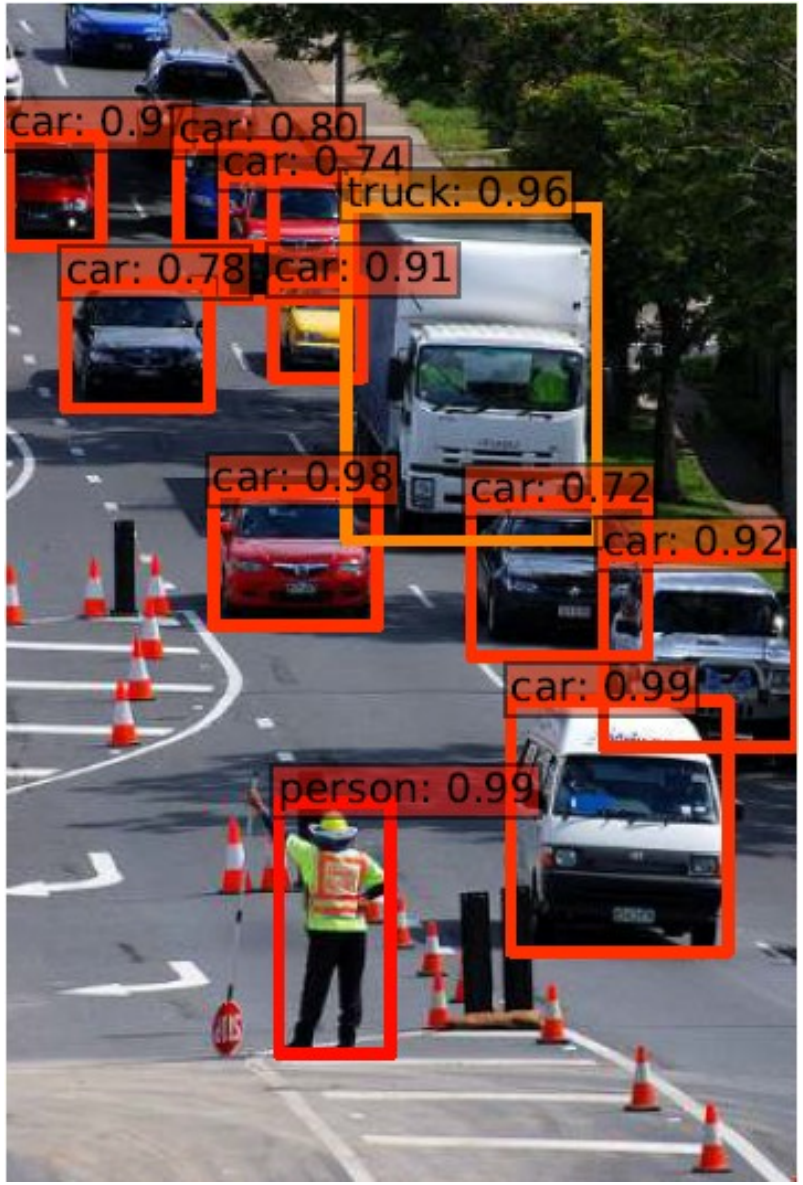


(b) 8×8 feature map



(c) 4×4 feature map


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


car: 0.91
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

YOLO, Detectnet

Models detection as a regression problem:

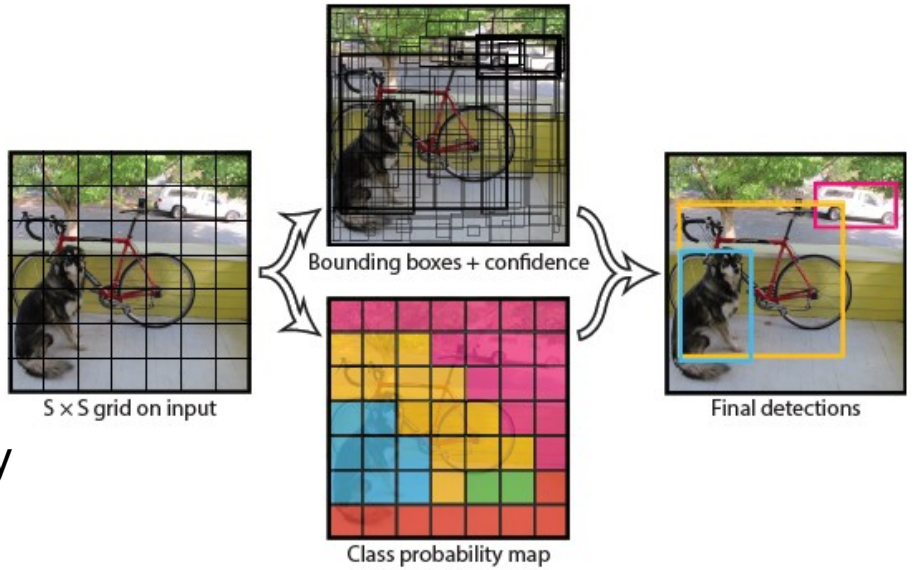
Divide the image into a grid and each cell can vote for the bounding box position of possible object. (Four output per cell for the corner positions.)

Boxes can have arbitrary sizes

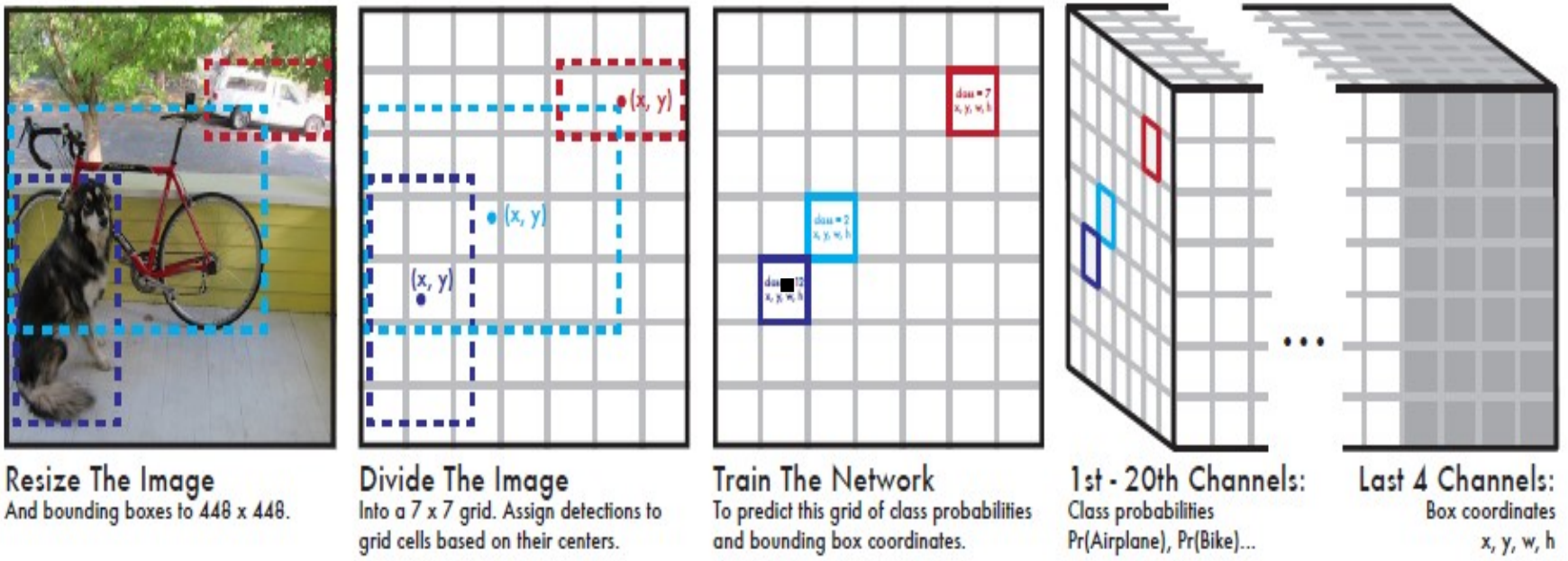
Each cell can proposes a bounding box one category (more layers, more categories per position).

Non-suppression on the boxes

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

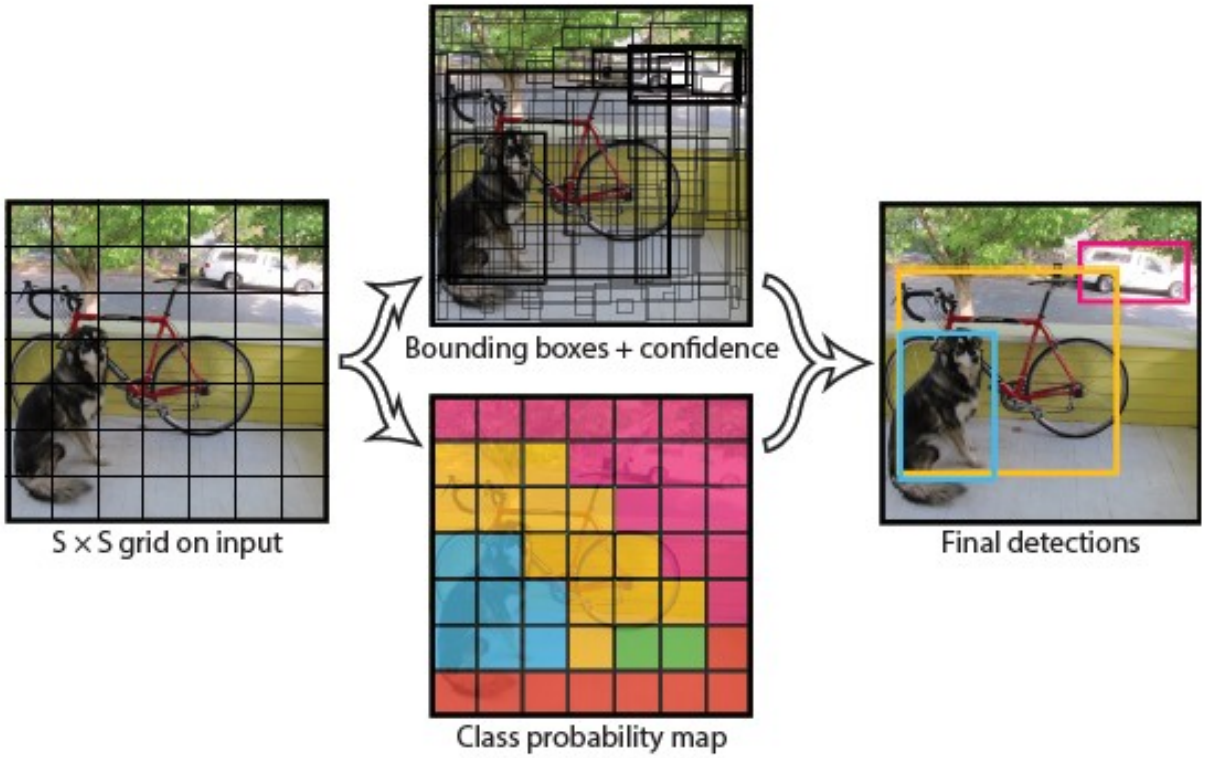


Handles oclusion



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?



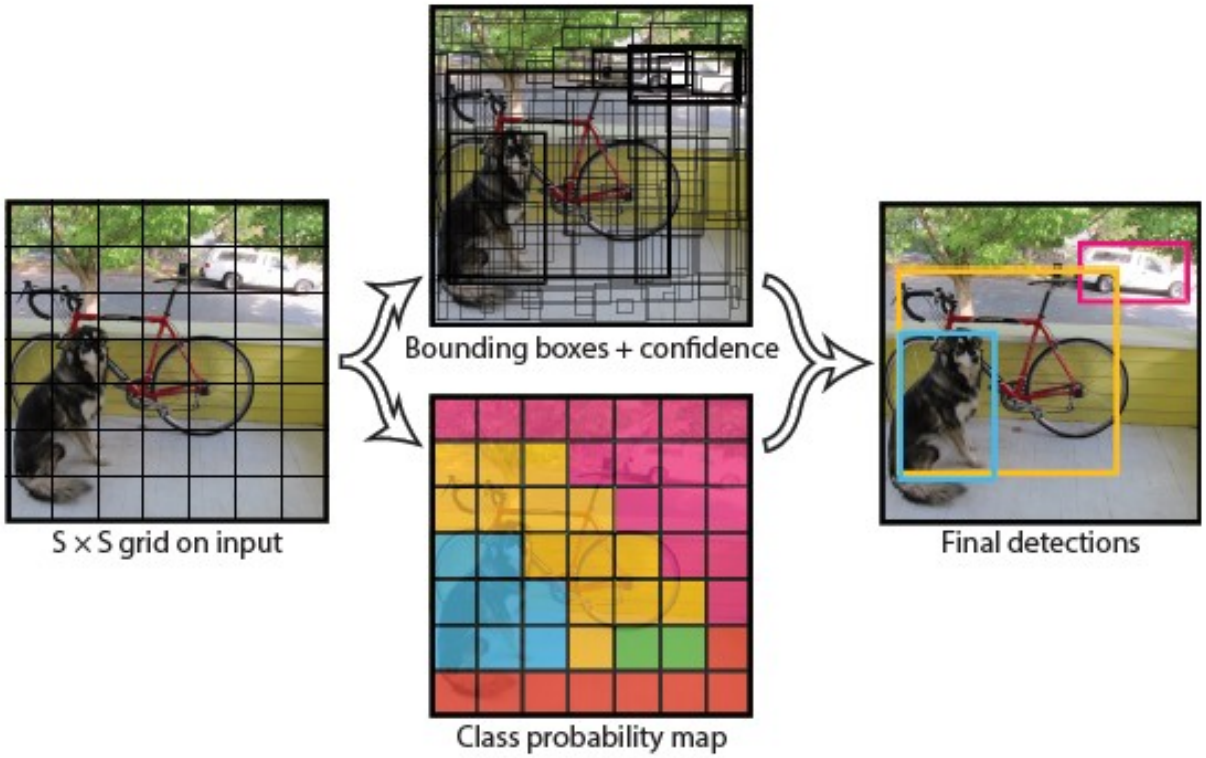
<u>confidence scores</u>: reflect how confident is that the box contains an object+how accurate the box is .

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

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

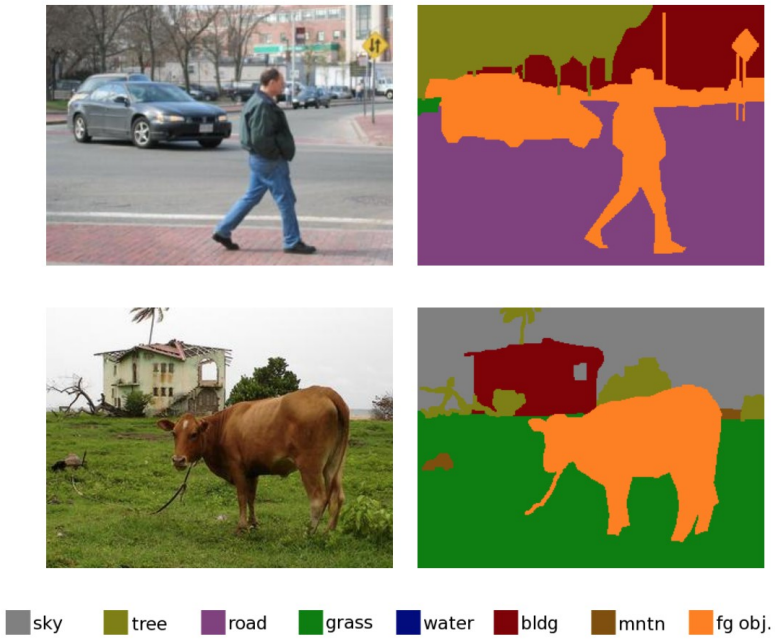
Pixel level segmentation

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)

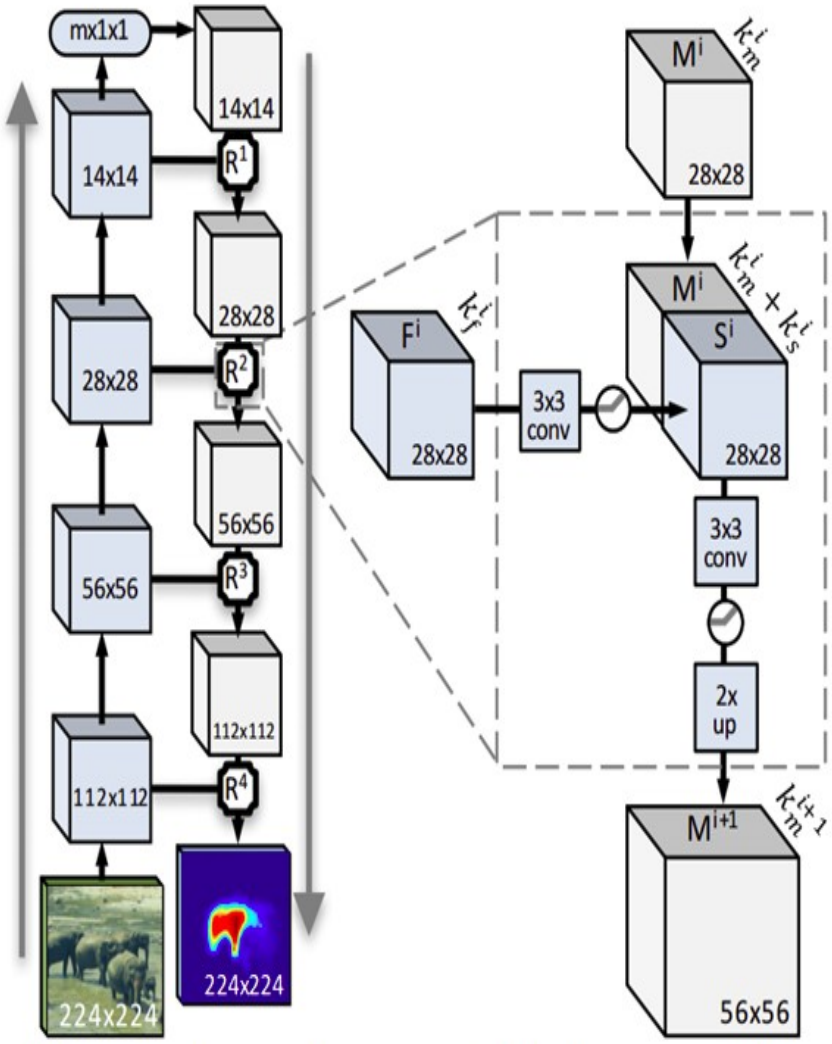
Popular architectures (Sharpmask, U-NET ...)



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

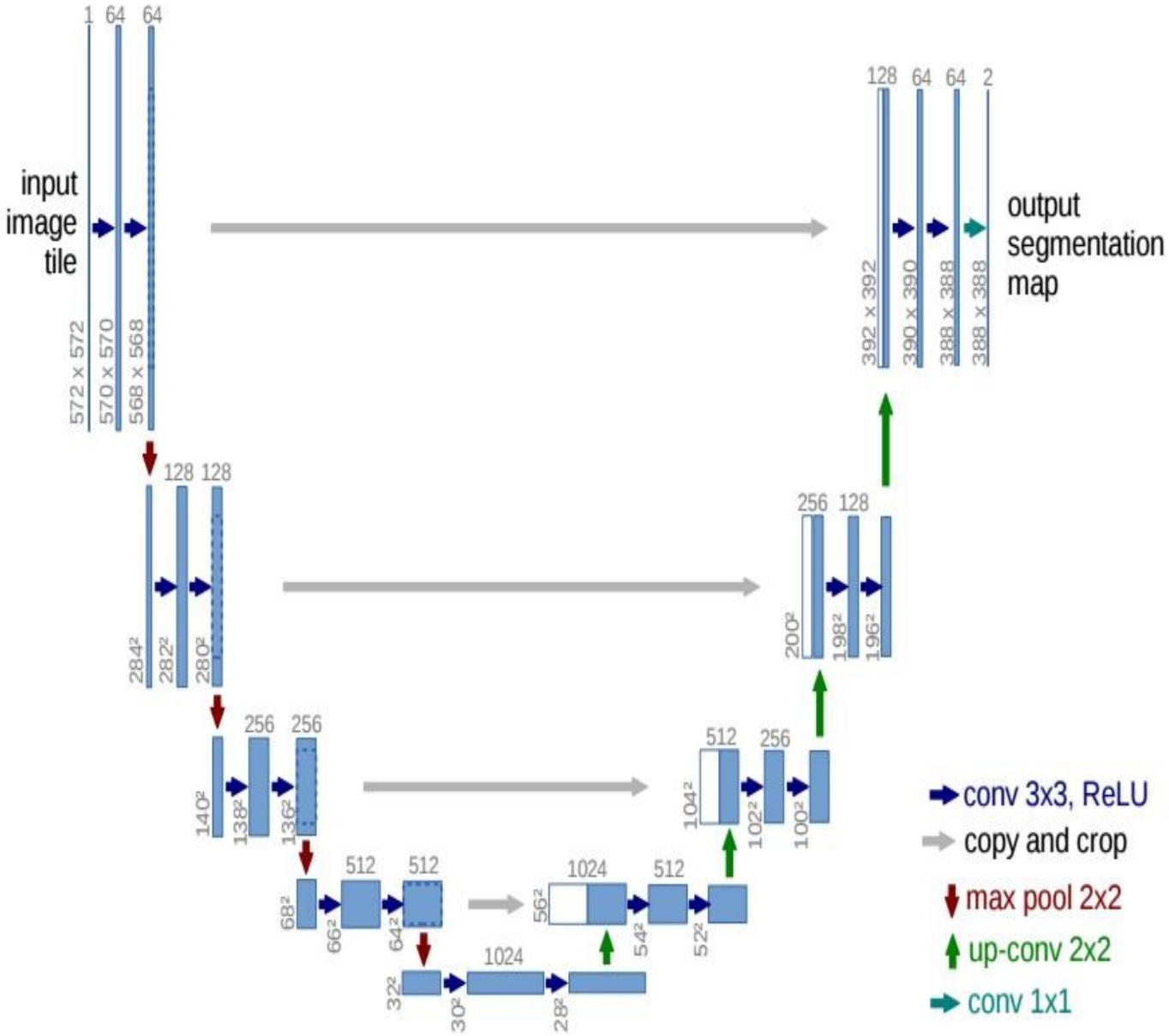
SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS Liang-Chieh Chen et al. ICLR 2015

Sharpmask



SharpMask network architecture

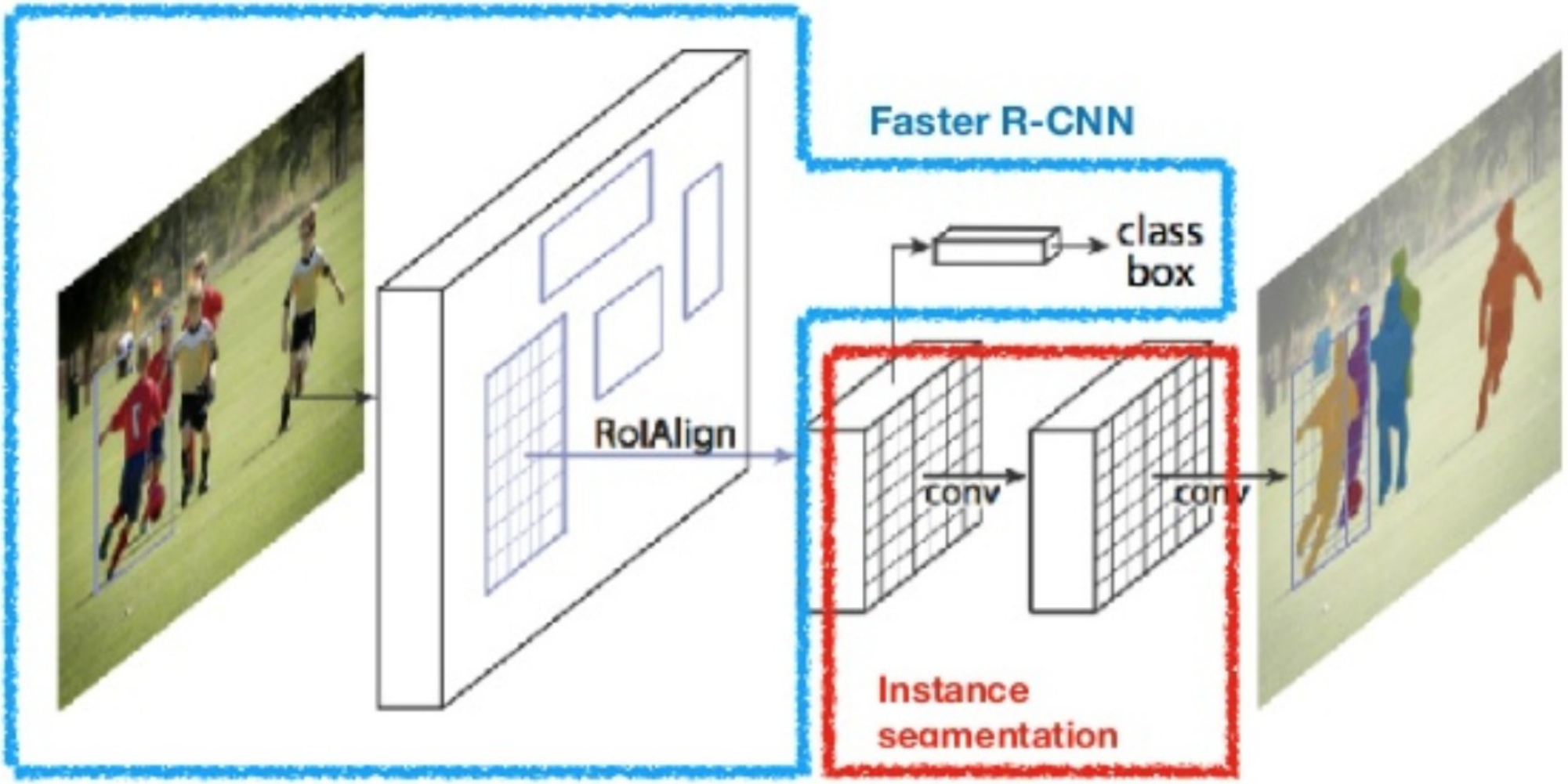
U-net


input image tile
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

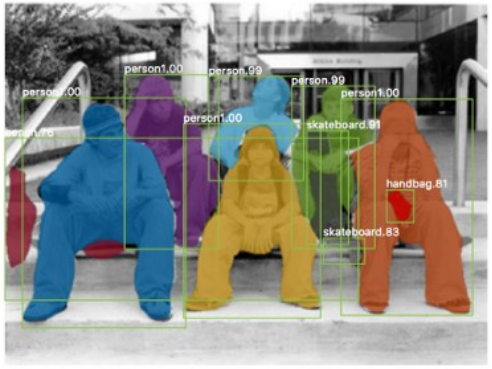
They can be trained on images having lables for only one or both types of output

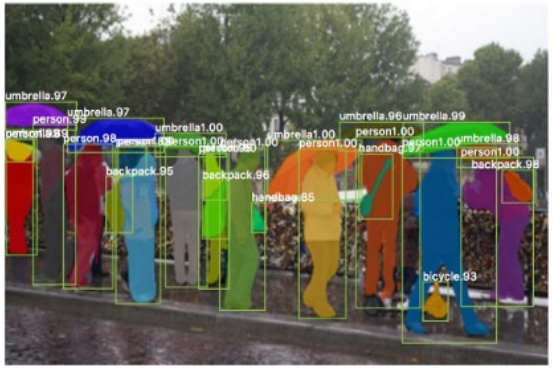


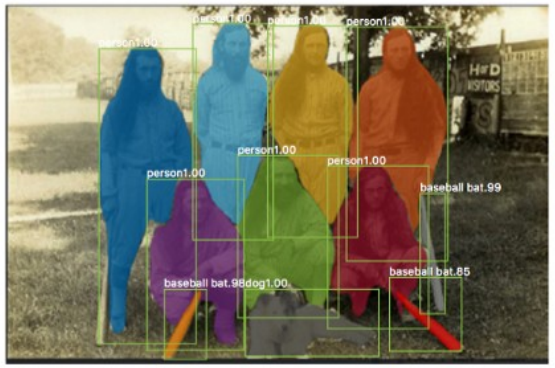
Mask RCNN, RetinaNet

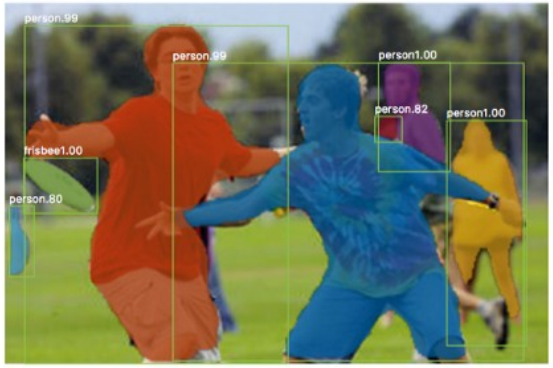
These networks generate bounding boxes and sematnic segmentation maps simultanously

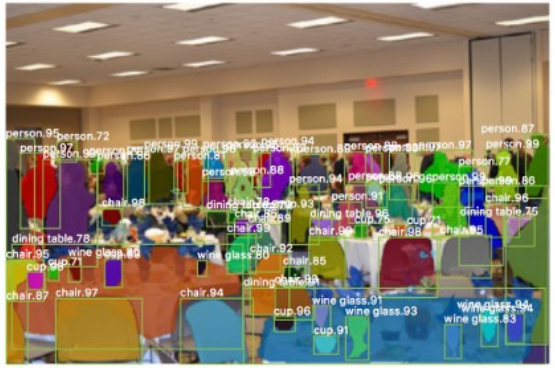
They can be trained on images having lables for only one or both types of output

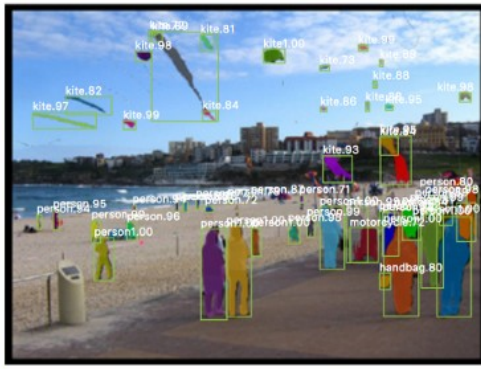


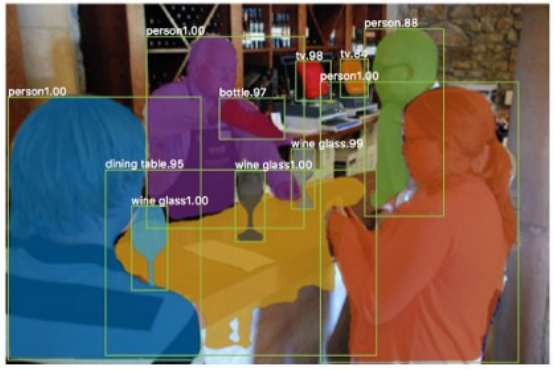


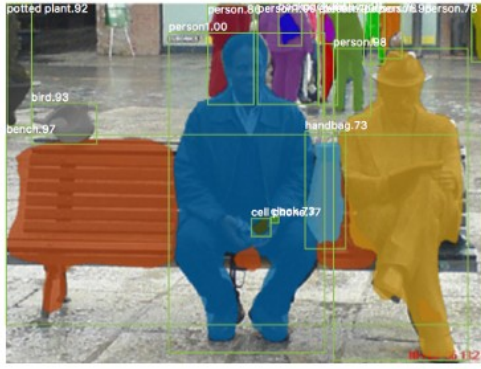












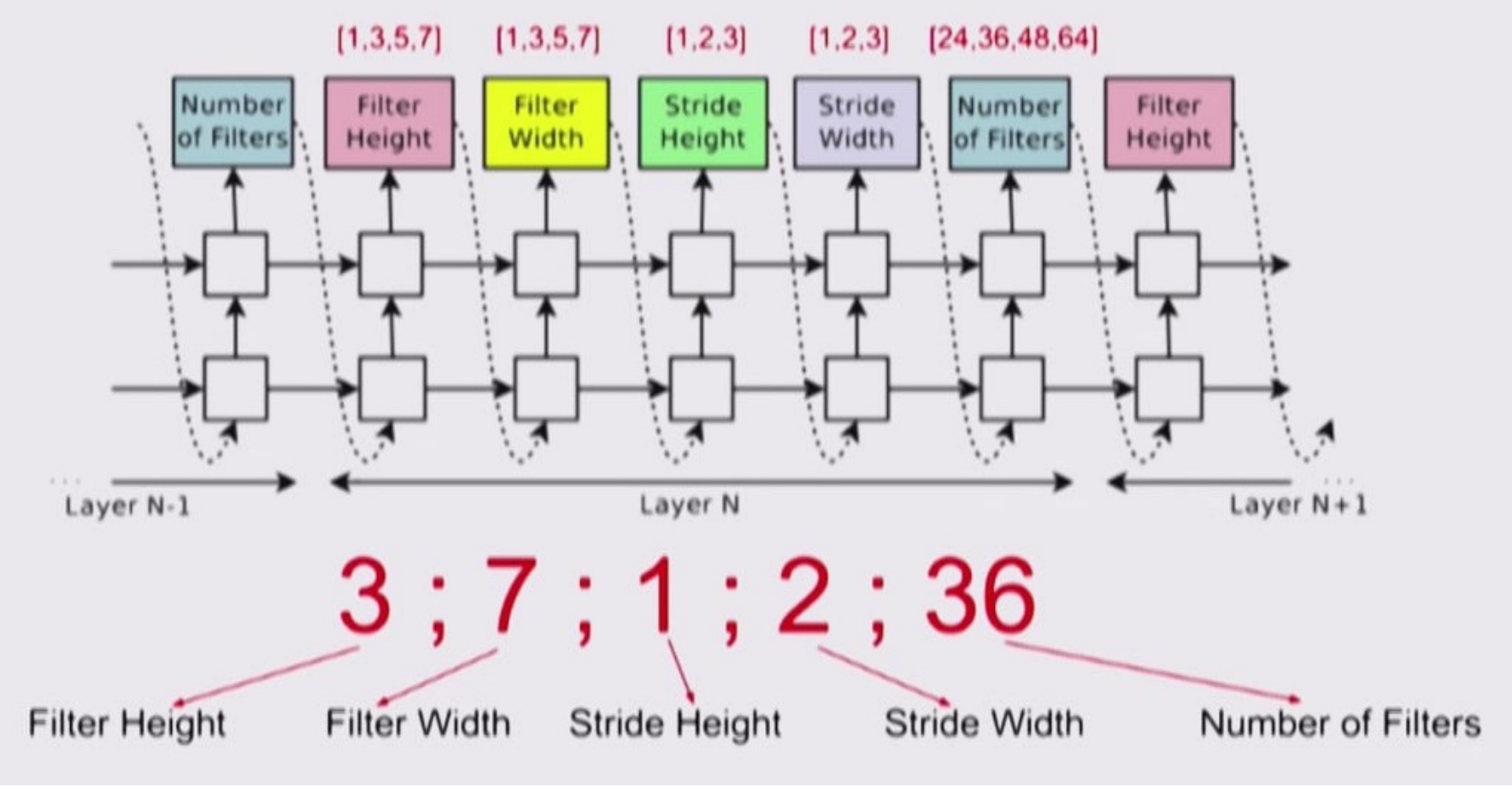
Starting from scratch (if you do not want to use one of the famous networks)

Neural architecture search:

Networks can be described as a series of operations

As series of words → text

We can feed a Recurrent network with this data series



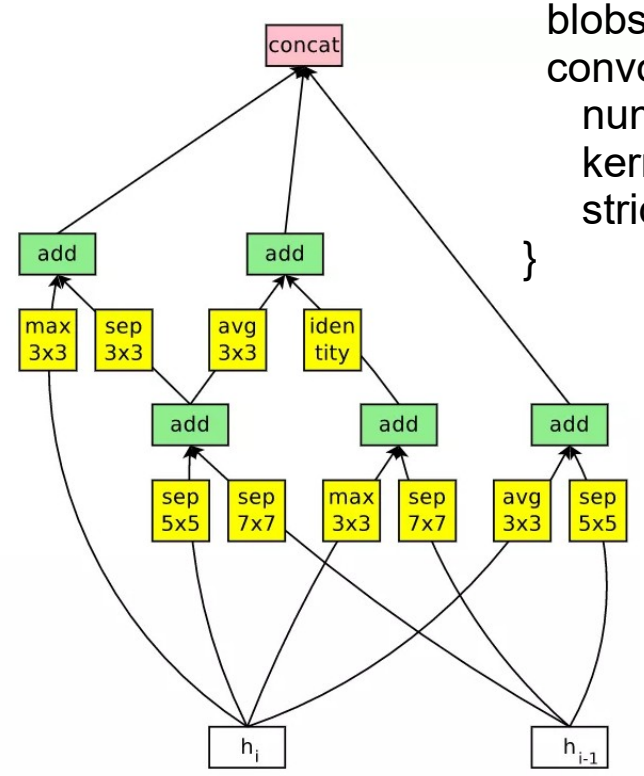
Starting from scratch

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

The parameters of each 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

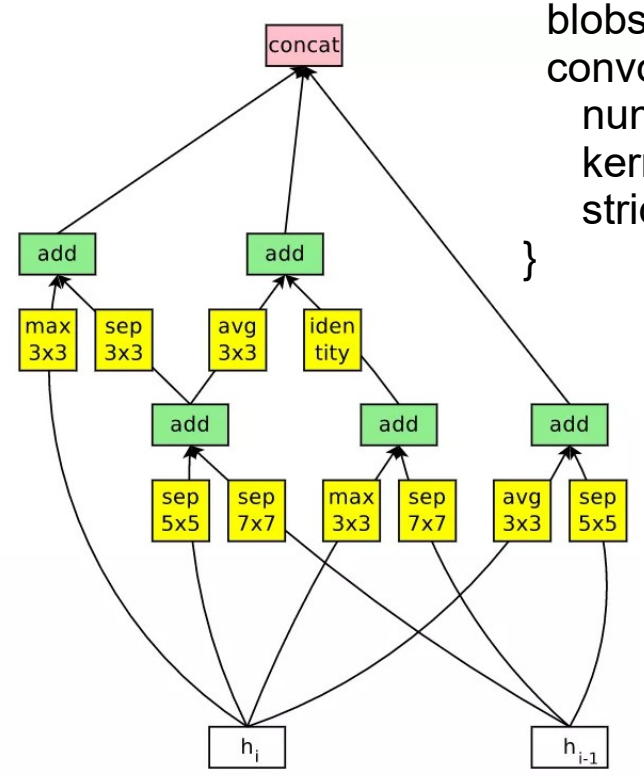
Neural architecture search:
Networks can be described as a series of operations
As series of words → text

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

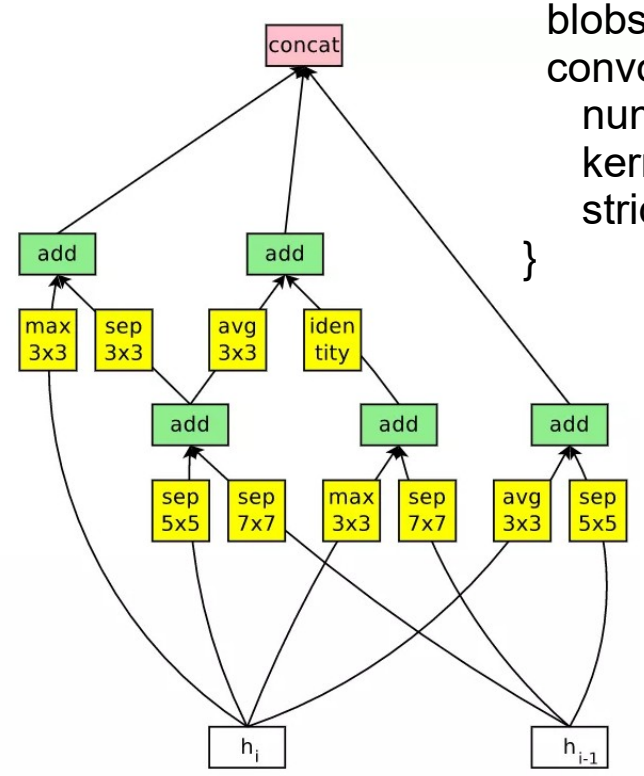
The parameters of eahc layer can be described as numbers The input(s)/outputs(s) of the layer can be Ids

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:

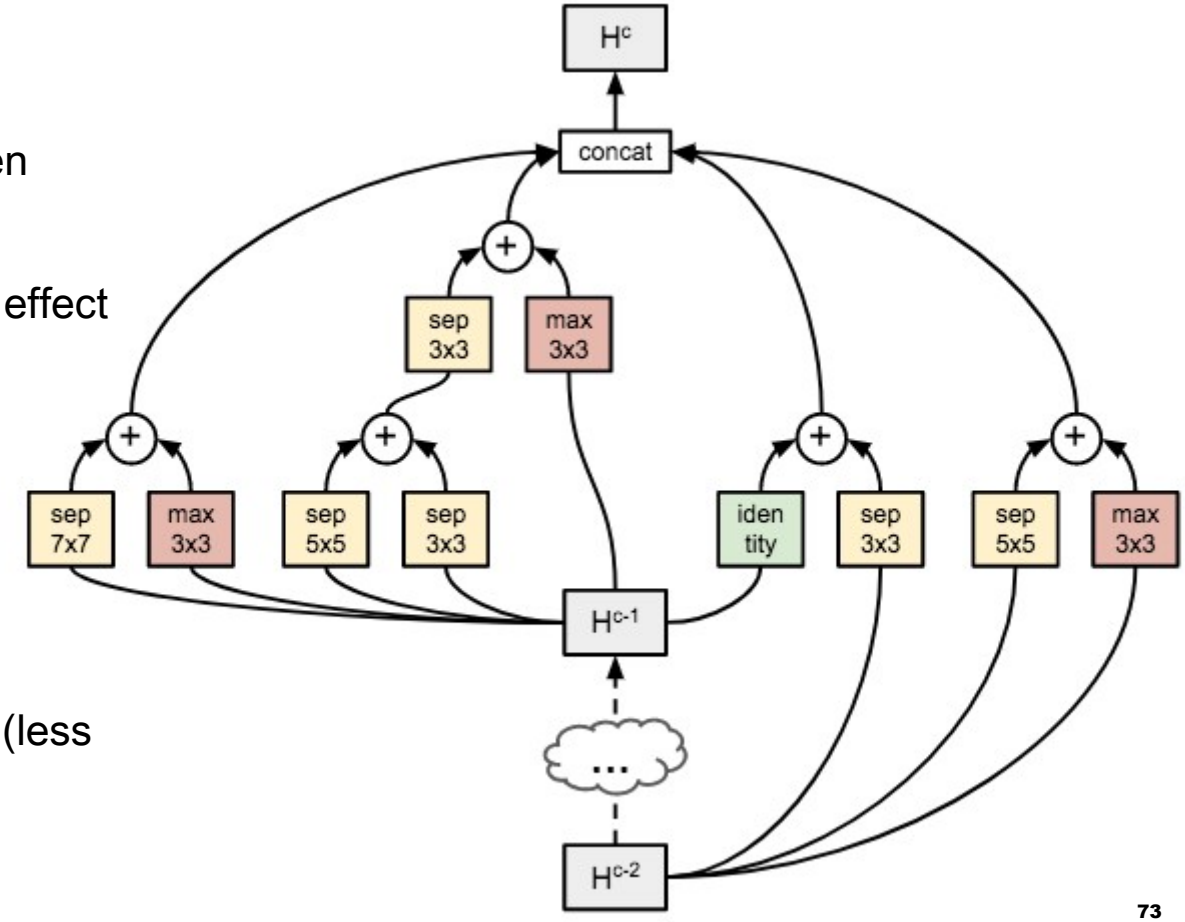
A recurrent network can be trained with reinforcement learning which can train a network with predifined accuracy on a given dataset.

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)



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.

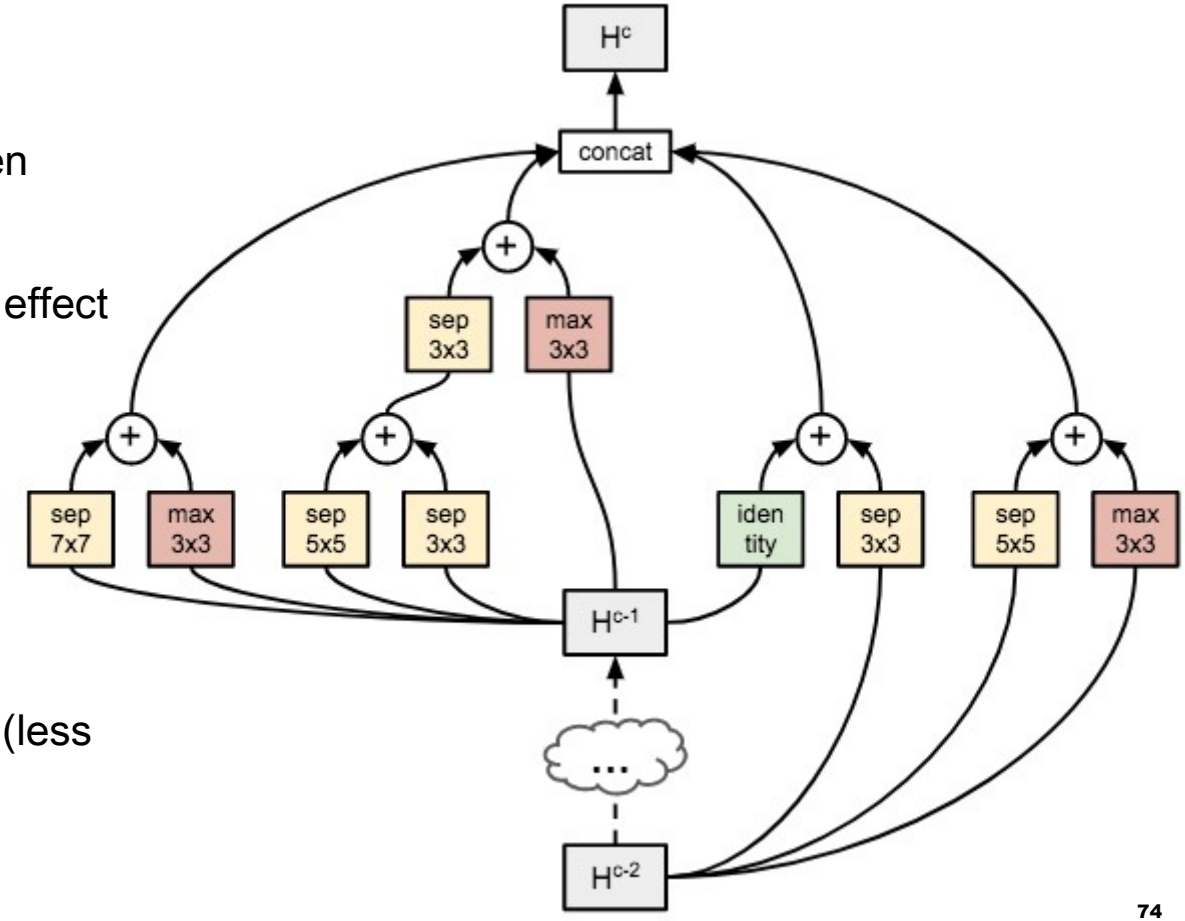
This recurrent network will understand the effect of the elements on this dataset

Test accuracy On CIFAR-10:
96.35%

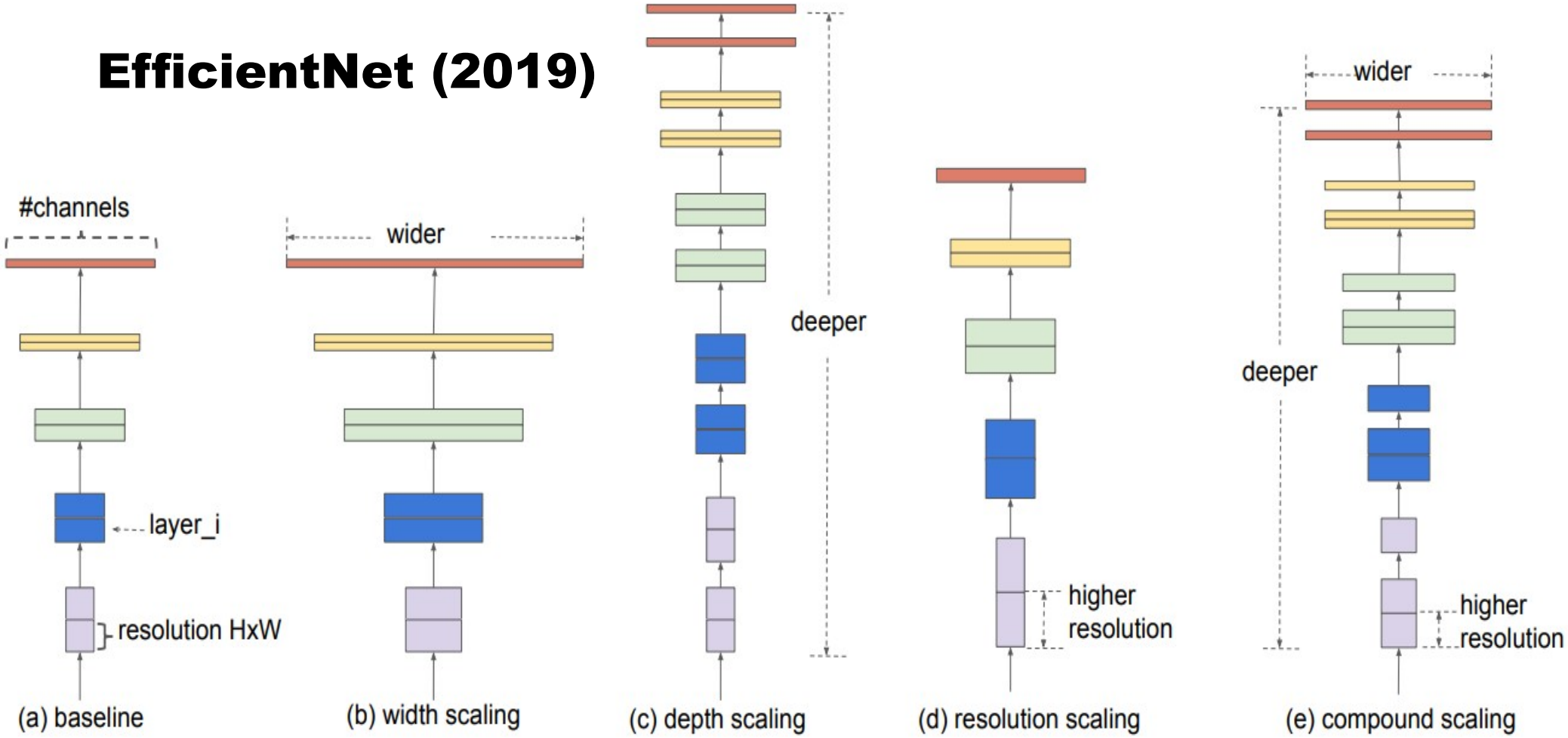
Best pervious accuraccy:
96.26

This architecture os also 1.05 times faster (less computations)

The important in this is that a network could design another network, and could reach as good performance as human.



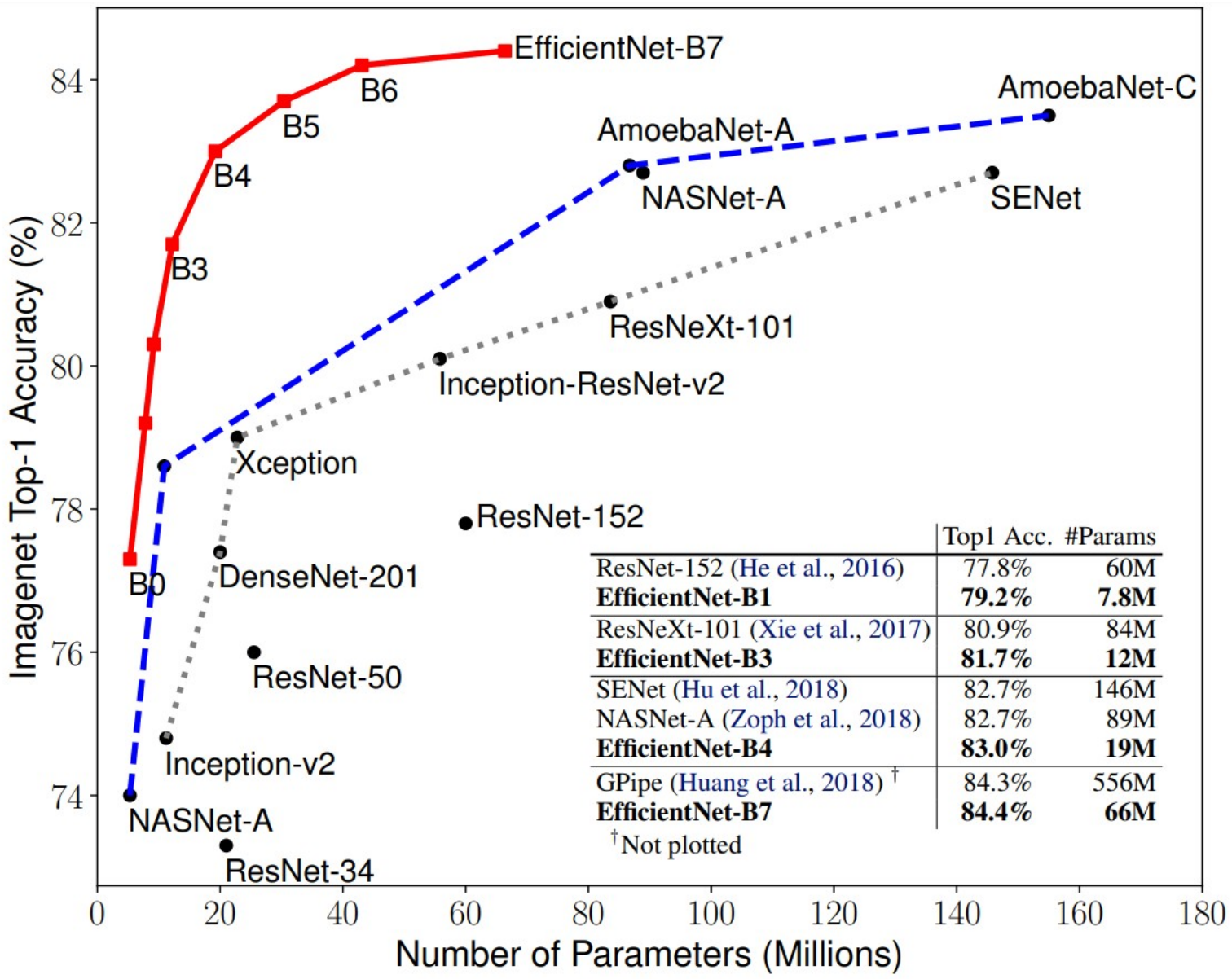
EfficientNet (2019)



- Scale the width, the depth, and the resolution uniformly!
- Can be used for any existing architecture, and the efficiency will be significantly better with the same performance
  - EfficientNet-B7 achieves stateof-the-art 84.4% top-1 / 97.1% top-5 accuracy on ImageNet, while being 8.4x smaller (number of parameters) and 6.1x faster on inference than the best existing ConvNet.
- Best performance can be reached by using NN to generate the optimal baseline ConvNet.

EfficientNet (2019)

7 different scaled version of EfficientNet. (B0, B1, … B7)



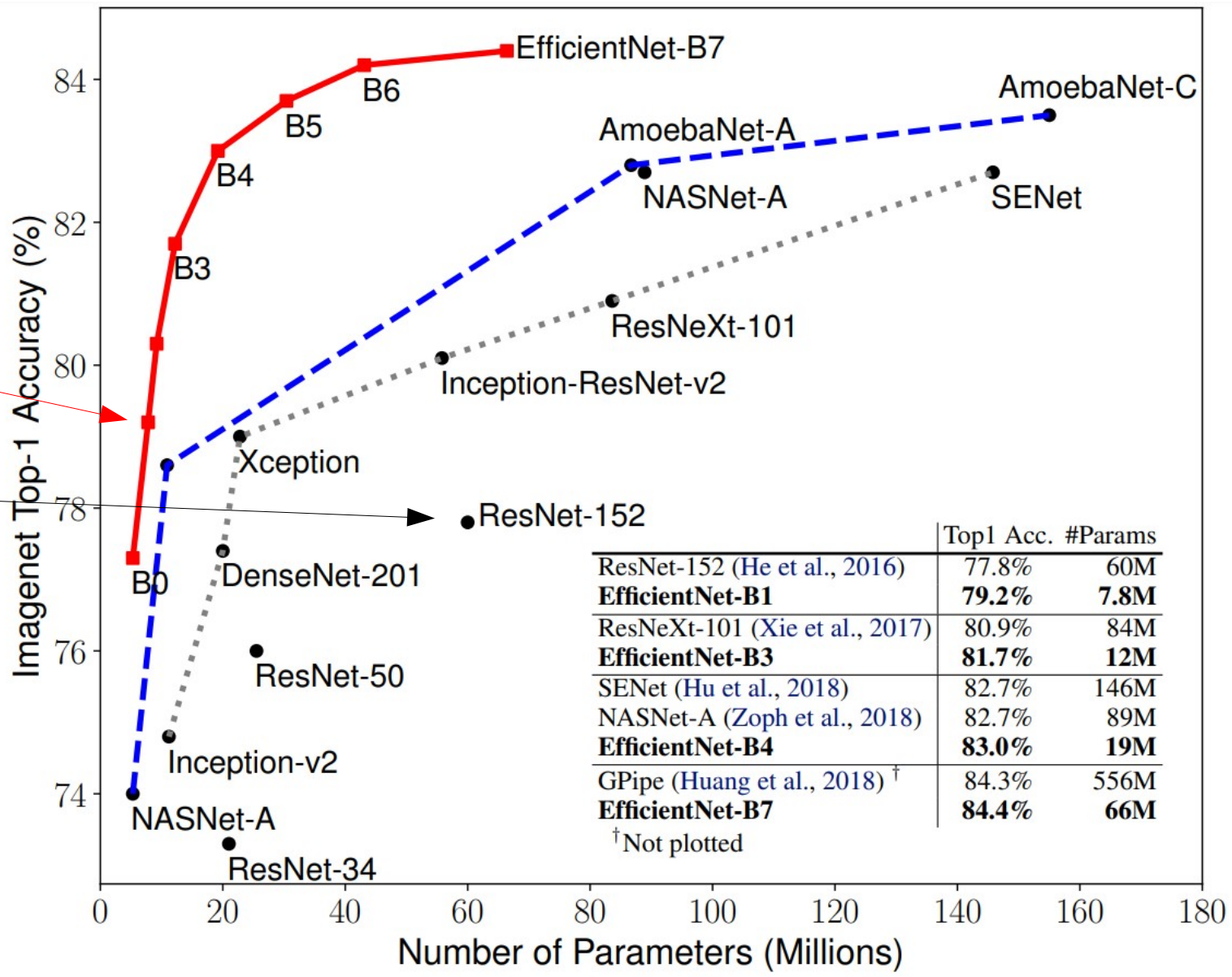
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

EfficientNet (2019)

7 different scaled version of EfficientNet. (B0, B1, … B7)

EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted