Computer vision tasks

Computer vision

- Understanding visual data
  - Images
  - Video
  - Range images
- Inverse graphics

Computer vision tasks

- Global labeling
  - Label per image/video
- Sparse labeling
  - Spatial/temporal location used
- Dense labeling
  - Label per pixel

Global labeling tasks

- Image classification
- Action recognition
- Image retrieval
- Captioning / VQA
- …

Sparse labeling tasks

- Object detection
- Sparse tracking
- Pose estimation
- Face recognition
- Optical character recognition
- …

Dense labeling tasks

- Semantic segmentation
- Instance segmentation
- Depth estimation
- Intrinsic image decomposition
- Dense tracking
- …

Image classification

Image classification

- Assign a single label per image

car

apple

Christmas tree

broccoli

Datasets: CIFAR-10

- Low resolution: 32x32
- 10 classes
- 60,000 images
- Subset of MIT tiny images

Learning Multiple Layers of Features from Tiny Images, Krizhevsky & Hinton, Technical report, 2009.
80 million tiny images: A large data set for nonparametric object and scene recognition, Torralba et al., PAMI 2008

Datasets: ImageNet

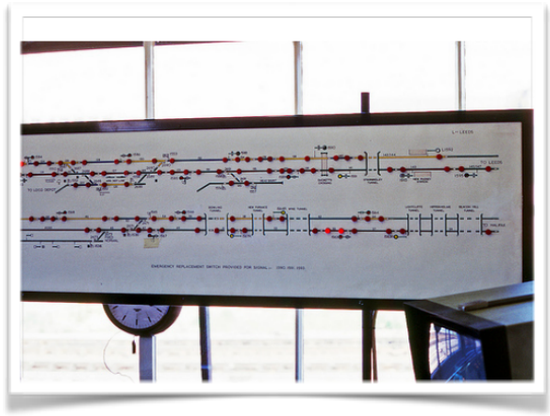
- 1000 classes
- 1.2M images
- Well balanced
- Well centered object

cheese golf ball

screwdriver harmonica

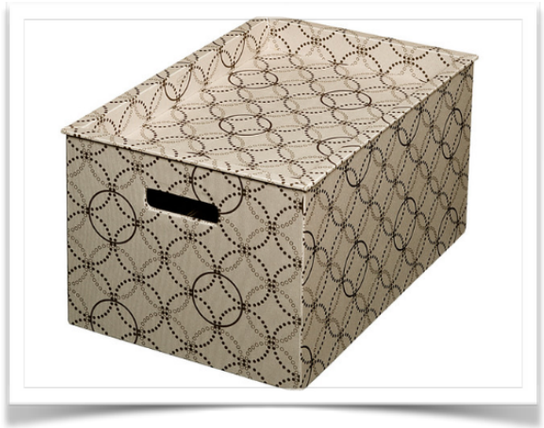
Datasets: Yahoo Flickr 100M

- >10K classes / tags
- 100M images
- noisy
- unbalanced



Datasets: Open Images

- 6,000 classes
- 9M images
- Creative commons



Datasets: MIT Places

- Scene categorization
- 400+ classes
- 10M images

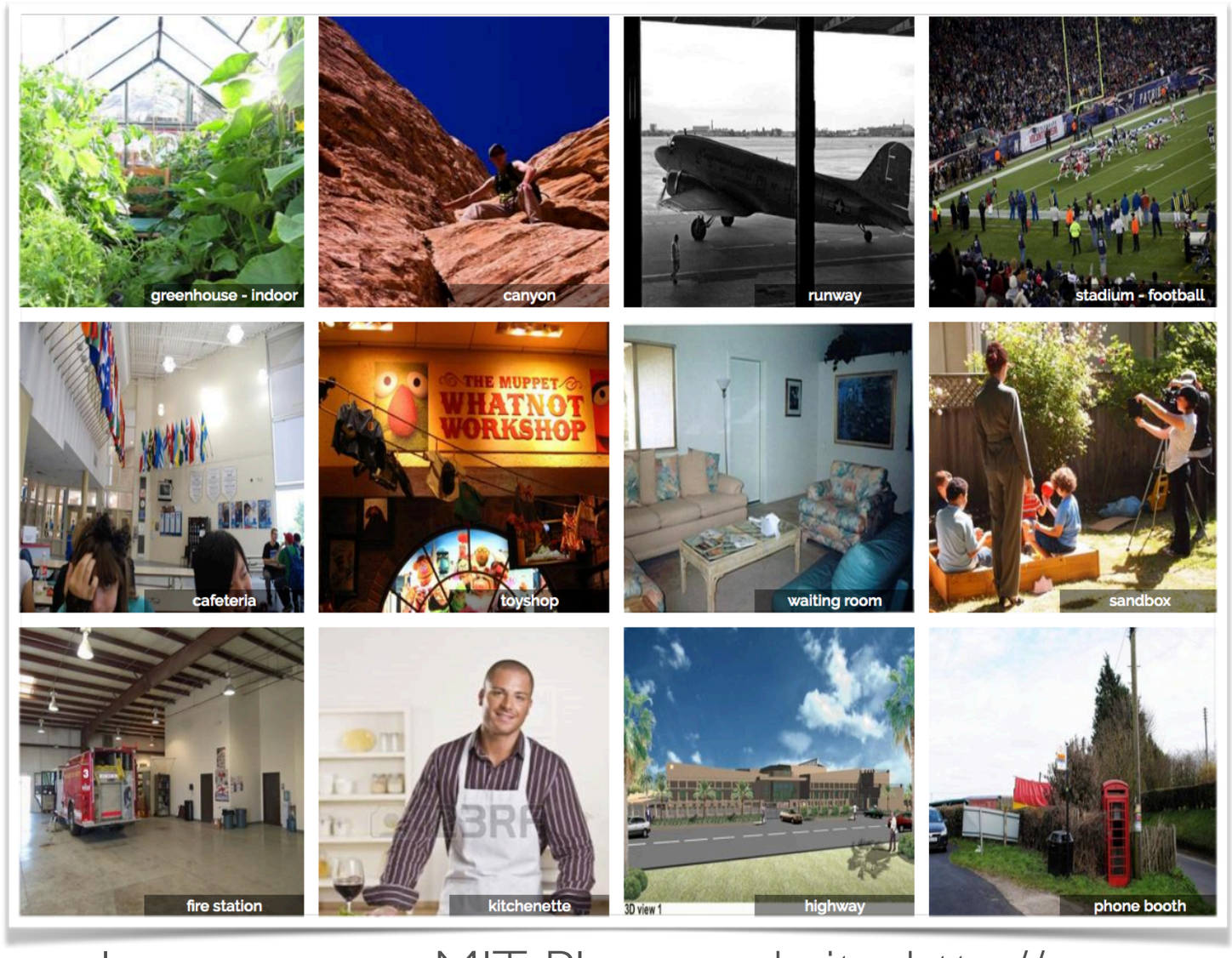
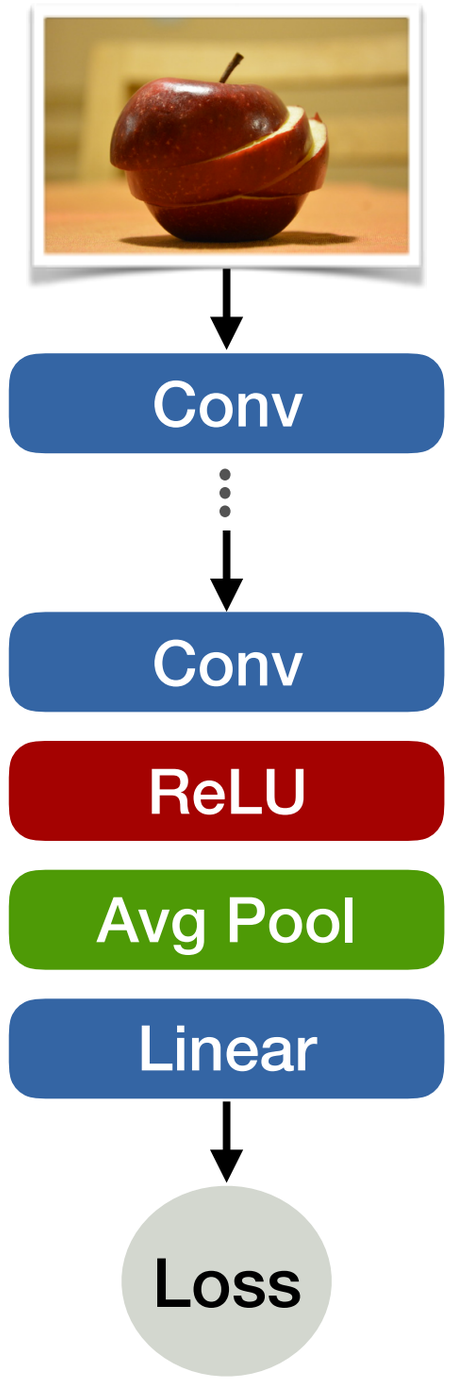


Image source: MIT Places website: http://places2.csail.mit.edu/explore.html

Image classification

- Input: Image
  - Fixed resolution
- Output: Class label
  - Cross entropy loss



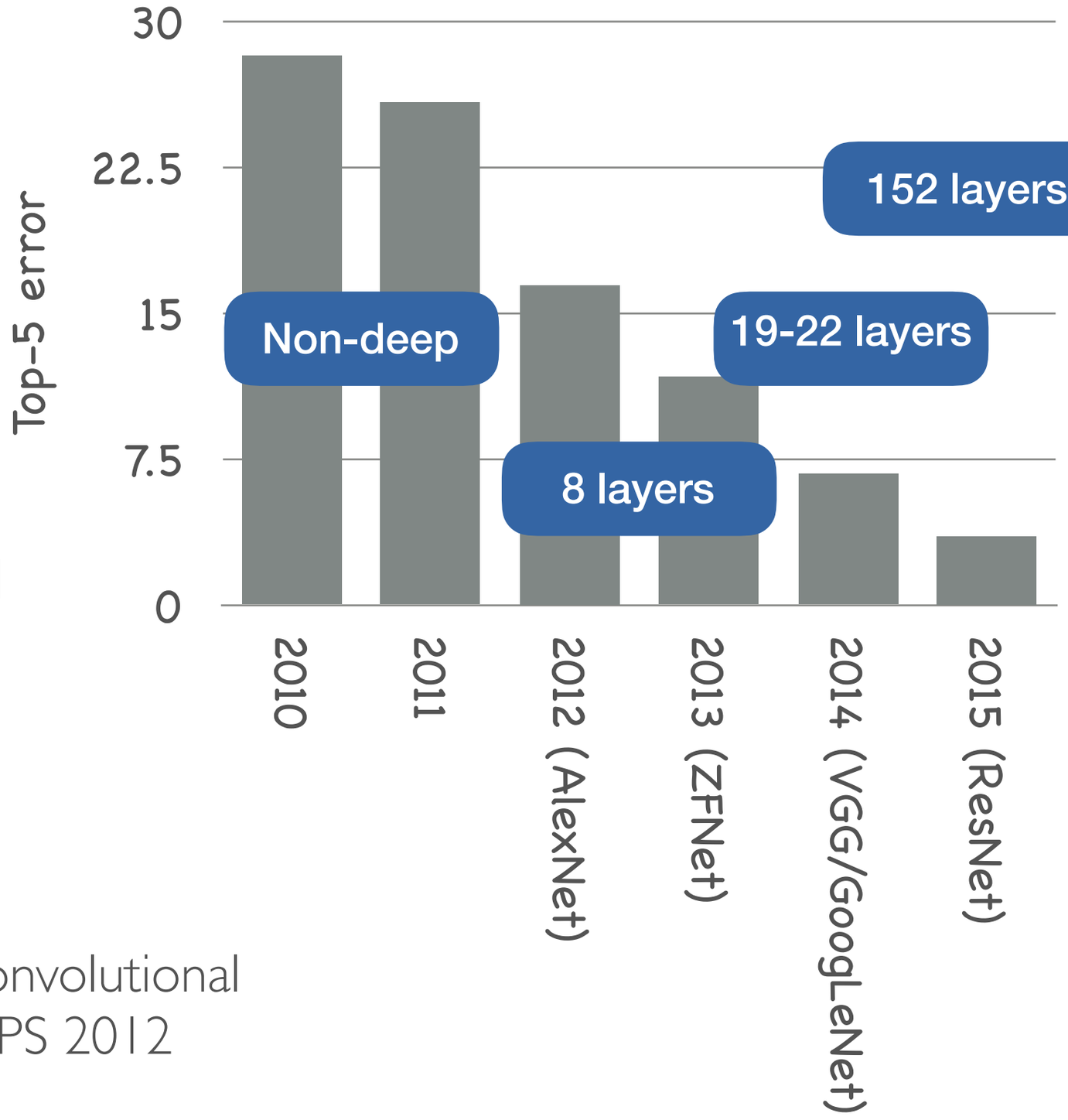
Applications

- Visual search
- Testbed for network design
    - Basis of many vision tasks

Case Study: AlexNet

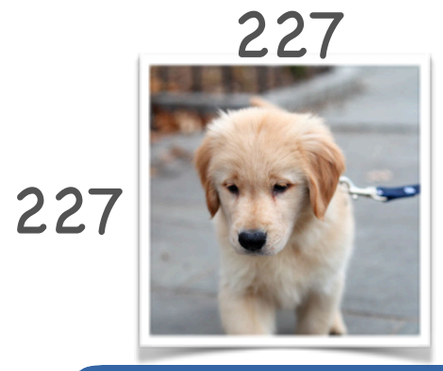
AlexNet

- Won the ImageNet competition 2012
- Start of deep learning revolution in computer vision

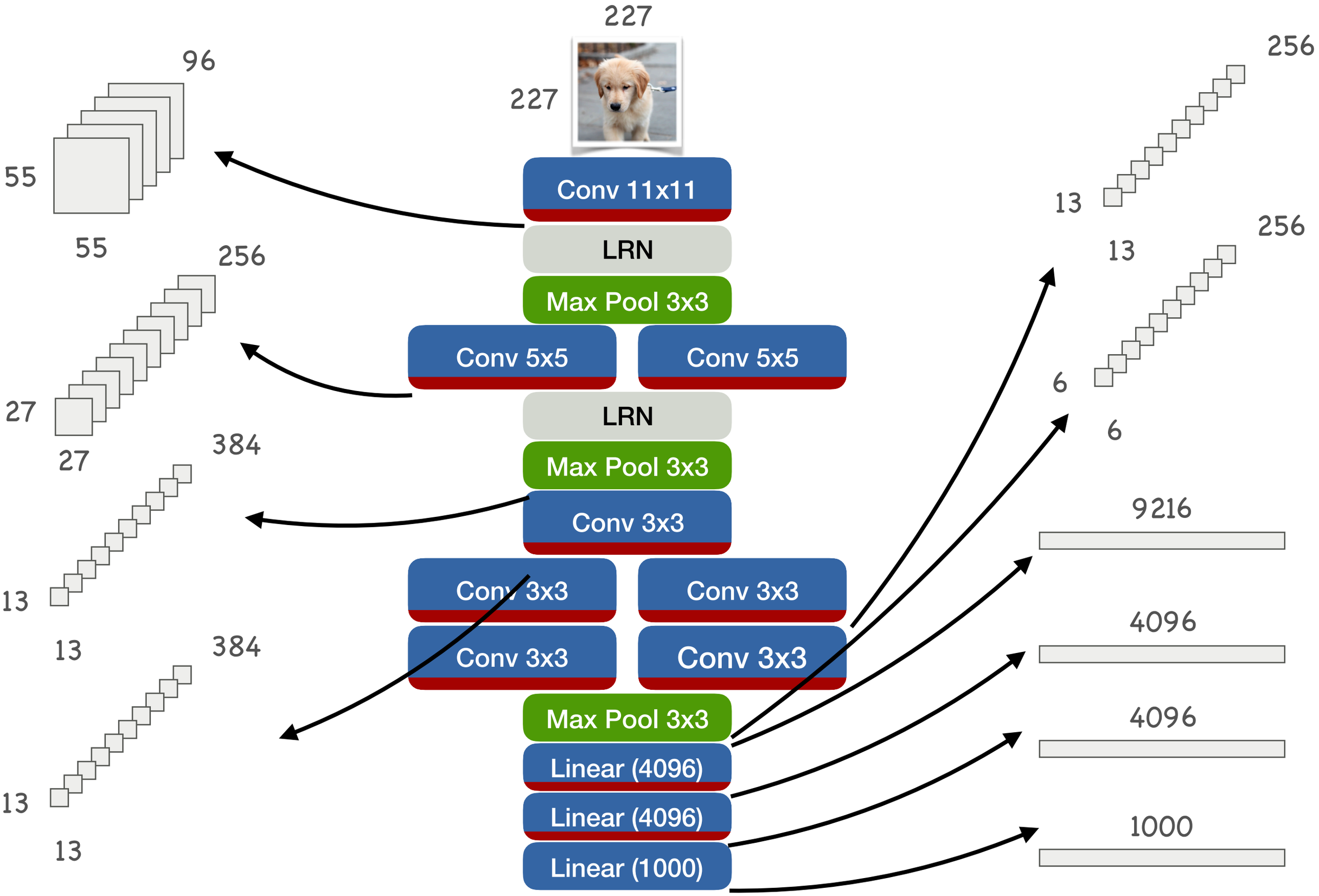


ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al., NIPS 2012

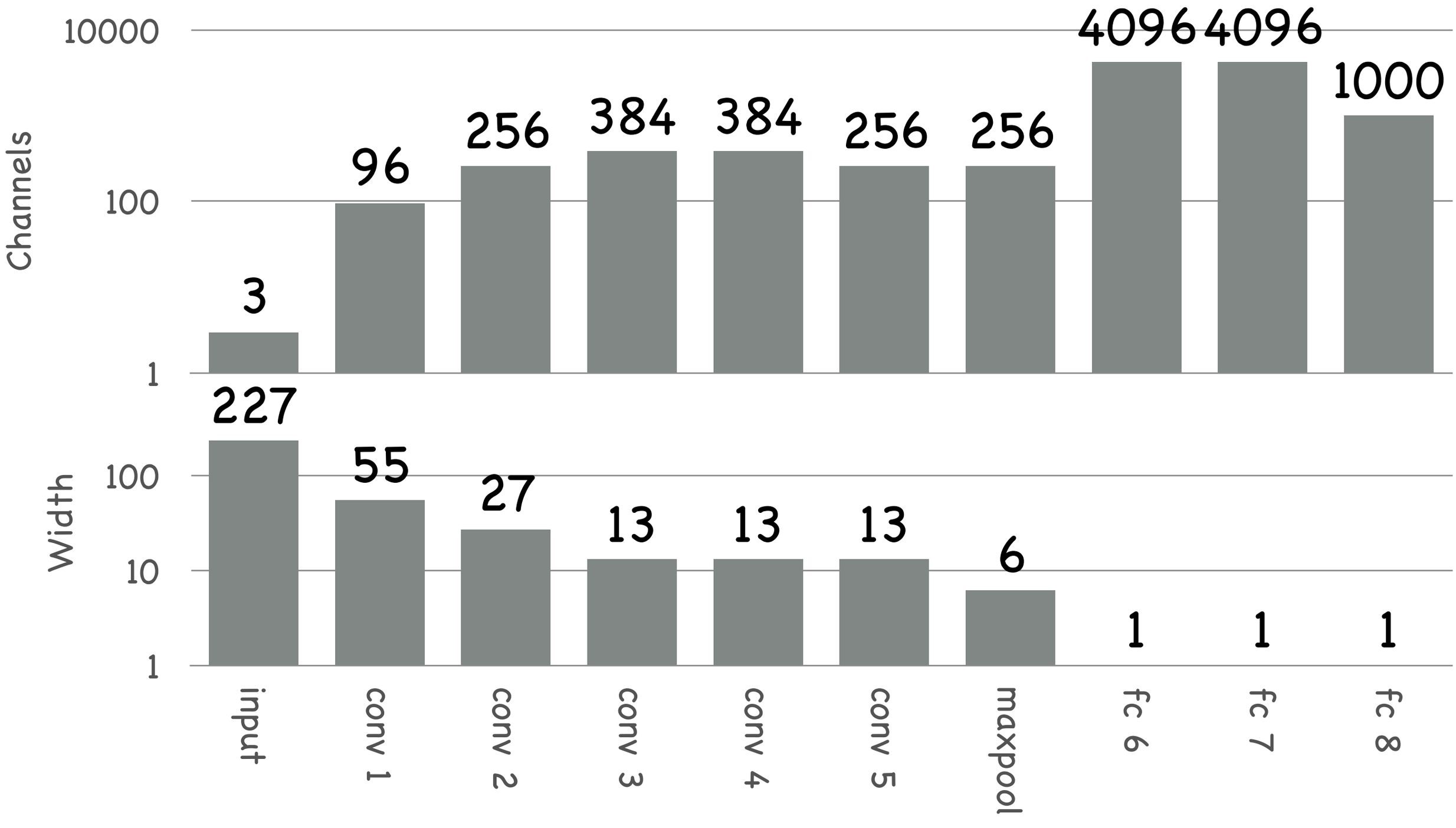
AlexNet


227
227
no pad
Conv 11x11
stride=4, C=96
LRN
no pad
Max Pool 3x3
stride=2
pad
Conv 5x5
Conv 5x5
C=128
LRN
Max Pool 3x3
pad=0, stride=2
Conv 3x3
C=384
Conv 3x3
Conv 3x3
C=192
Conv 3x3
Conv 3x3
C=128
Max Pool 3x3
pad=0, stride=2
Linear (4096)
Linear (4096)
Linear (1000)

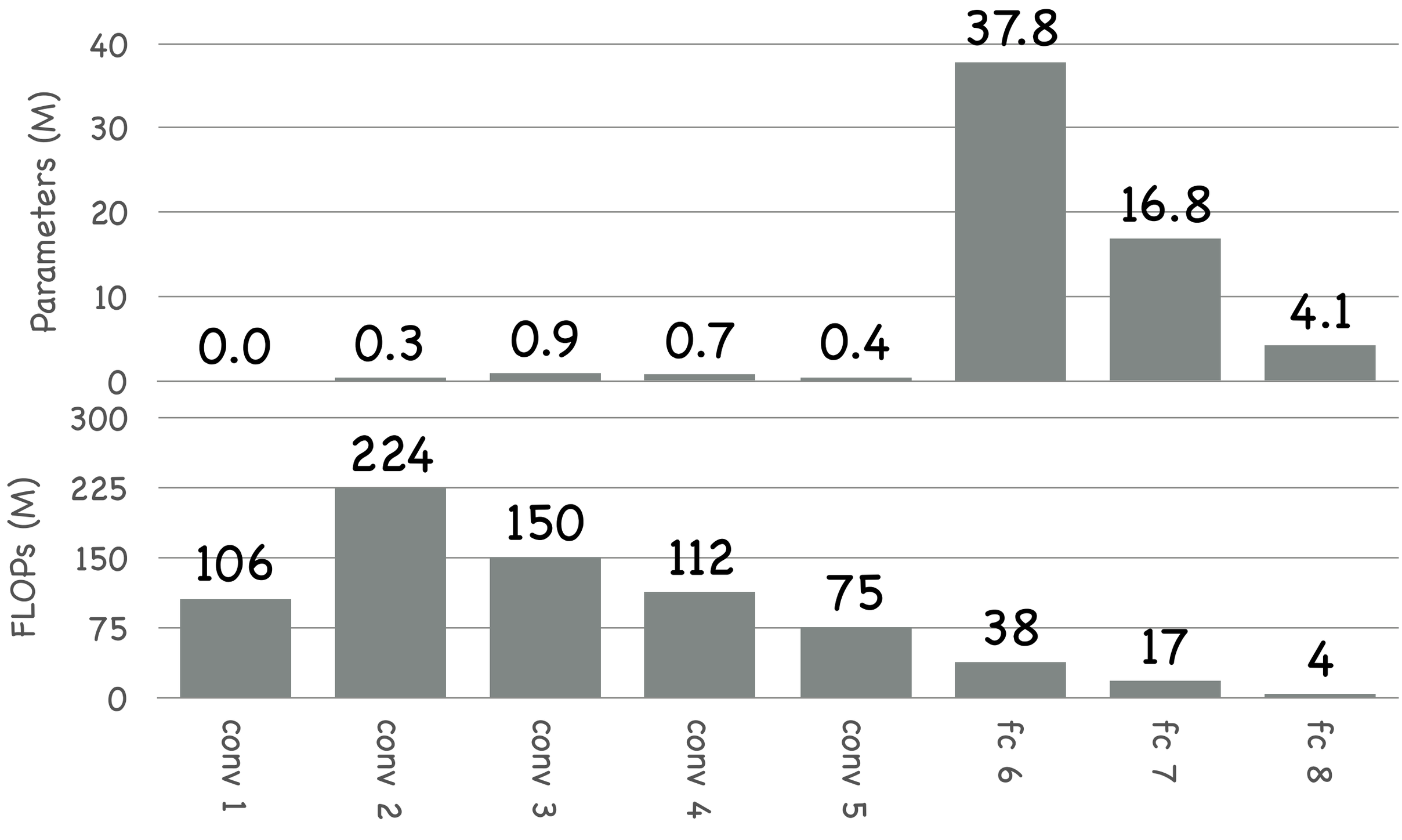
Activation maps in AlexNet



Activation maps in AlexNet

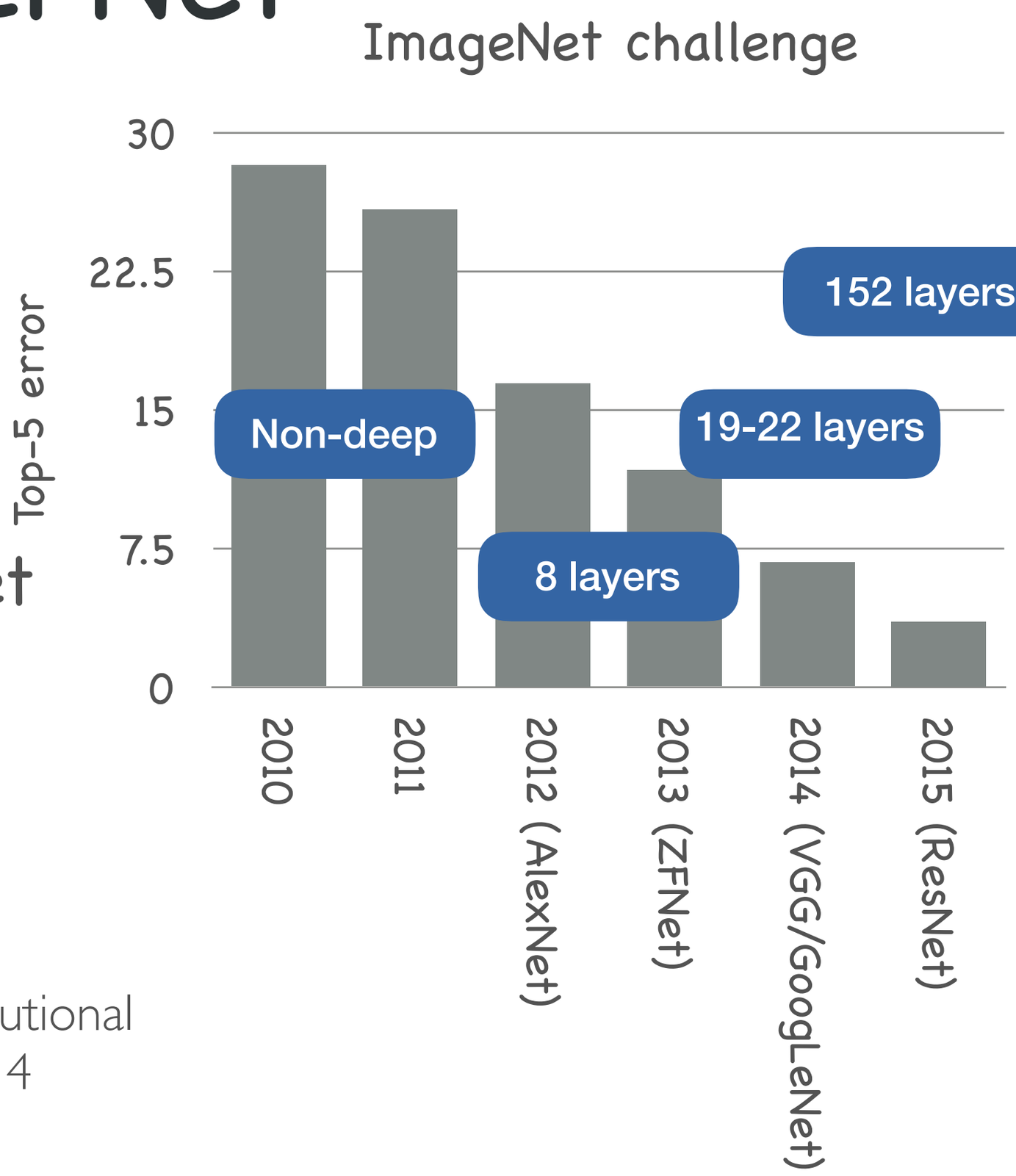


Parameters and computation



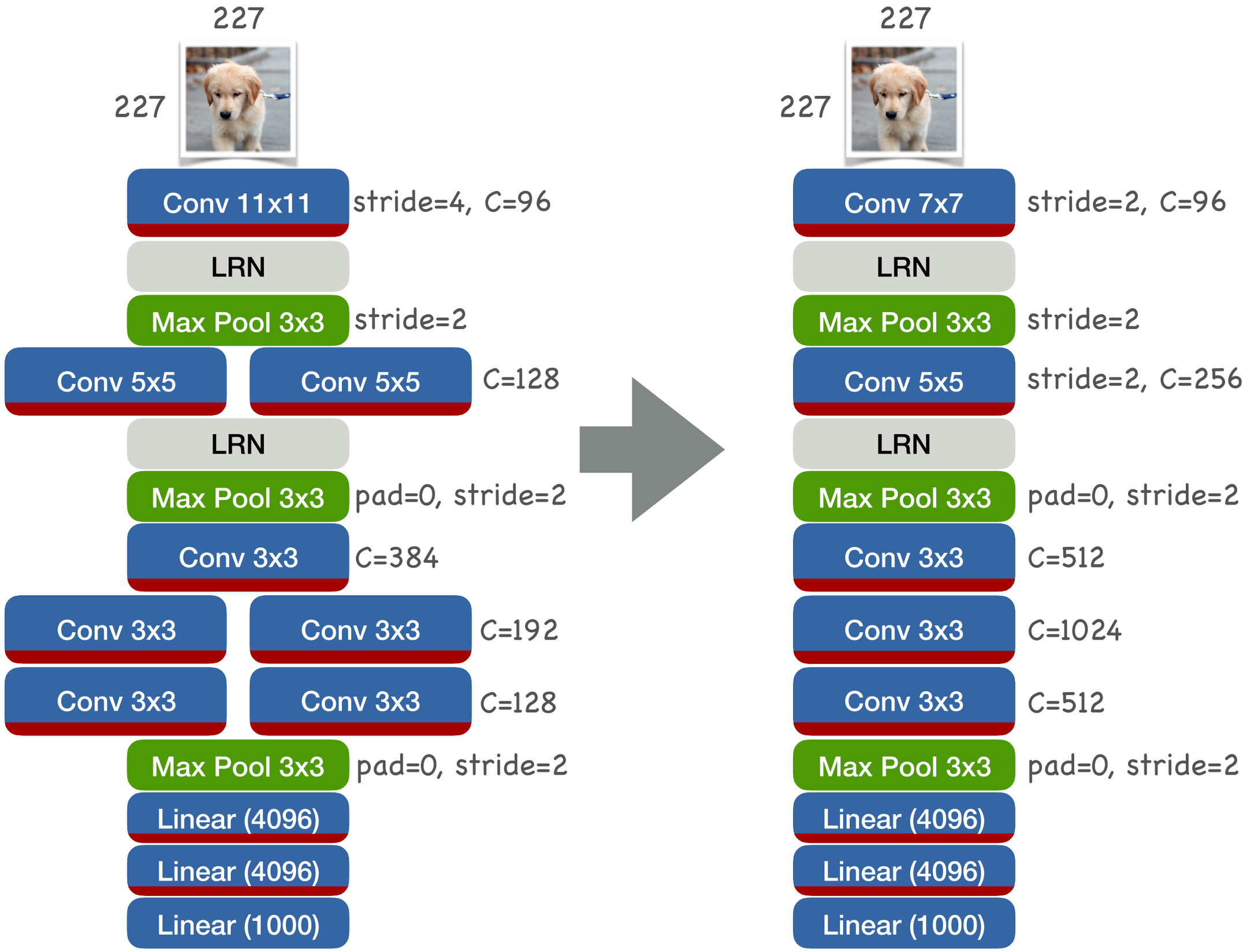
ZFNet

- Won the ImageNet competition 2013
- Nice analysis and visualization of AlexNet
- Introduced up-conv

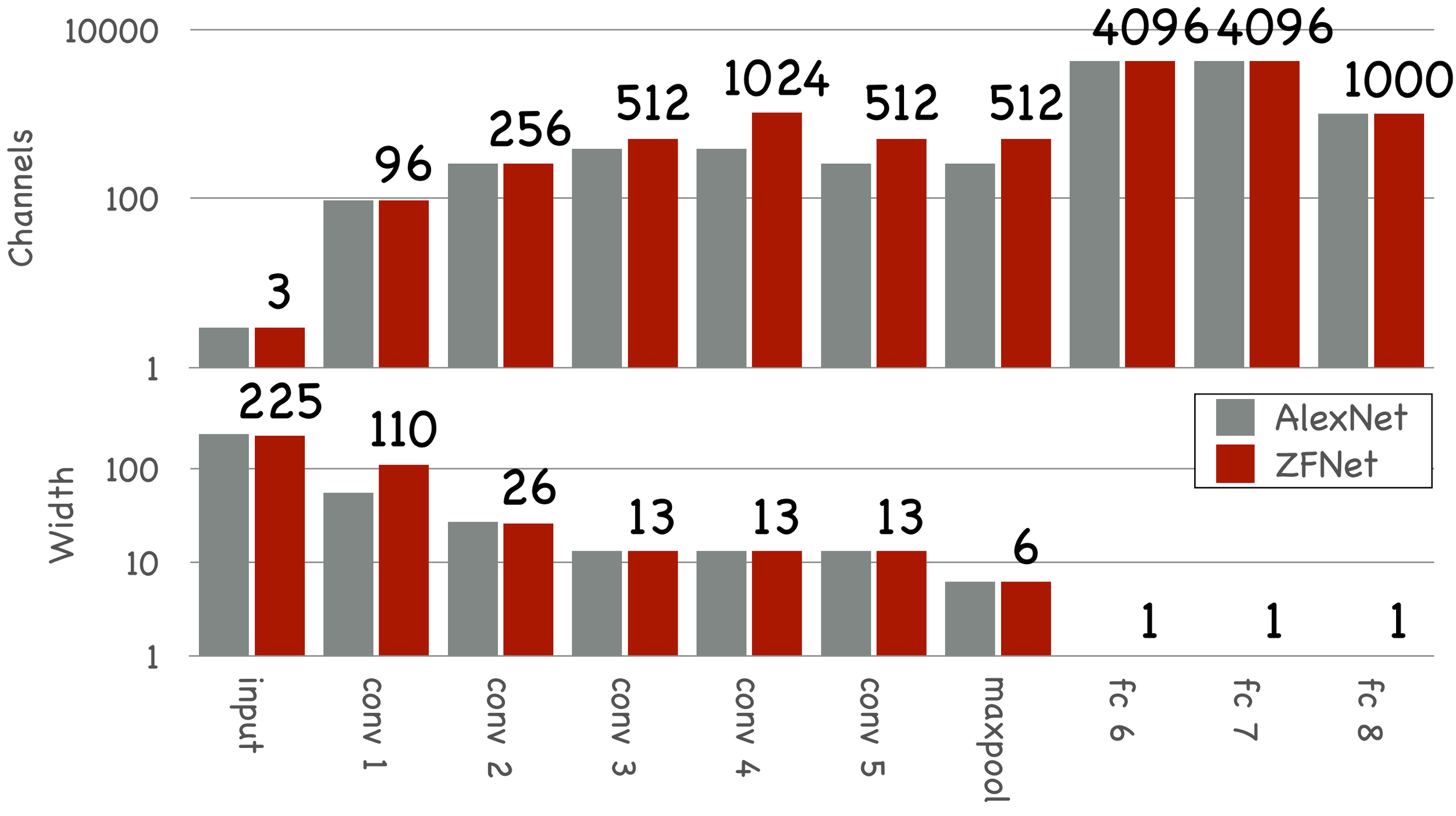


Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, ECCV 2014

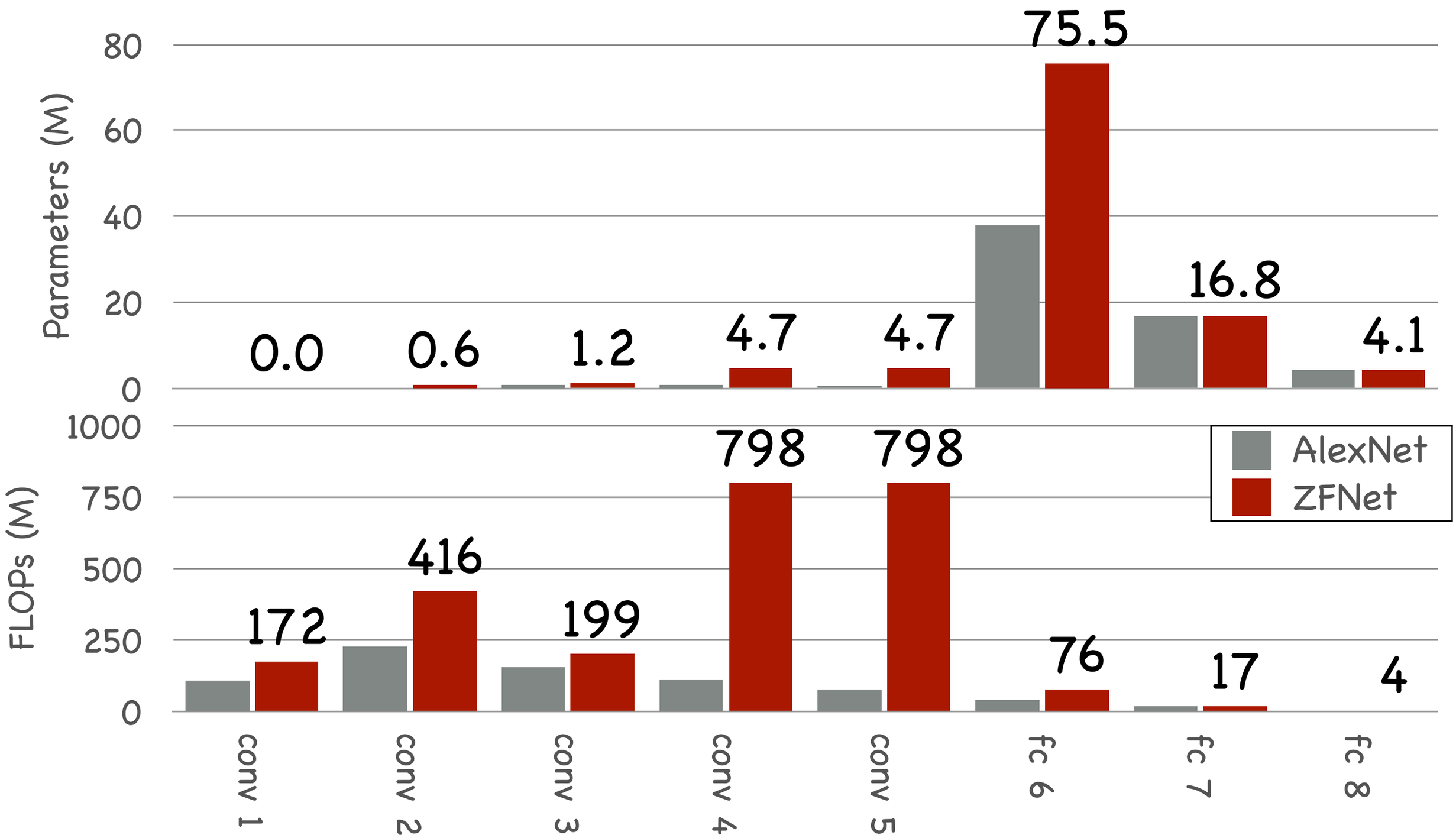
AlexNet to ZFNet



Activation maps in ZFNet



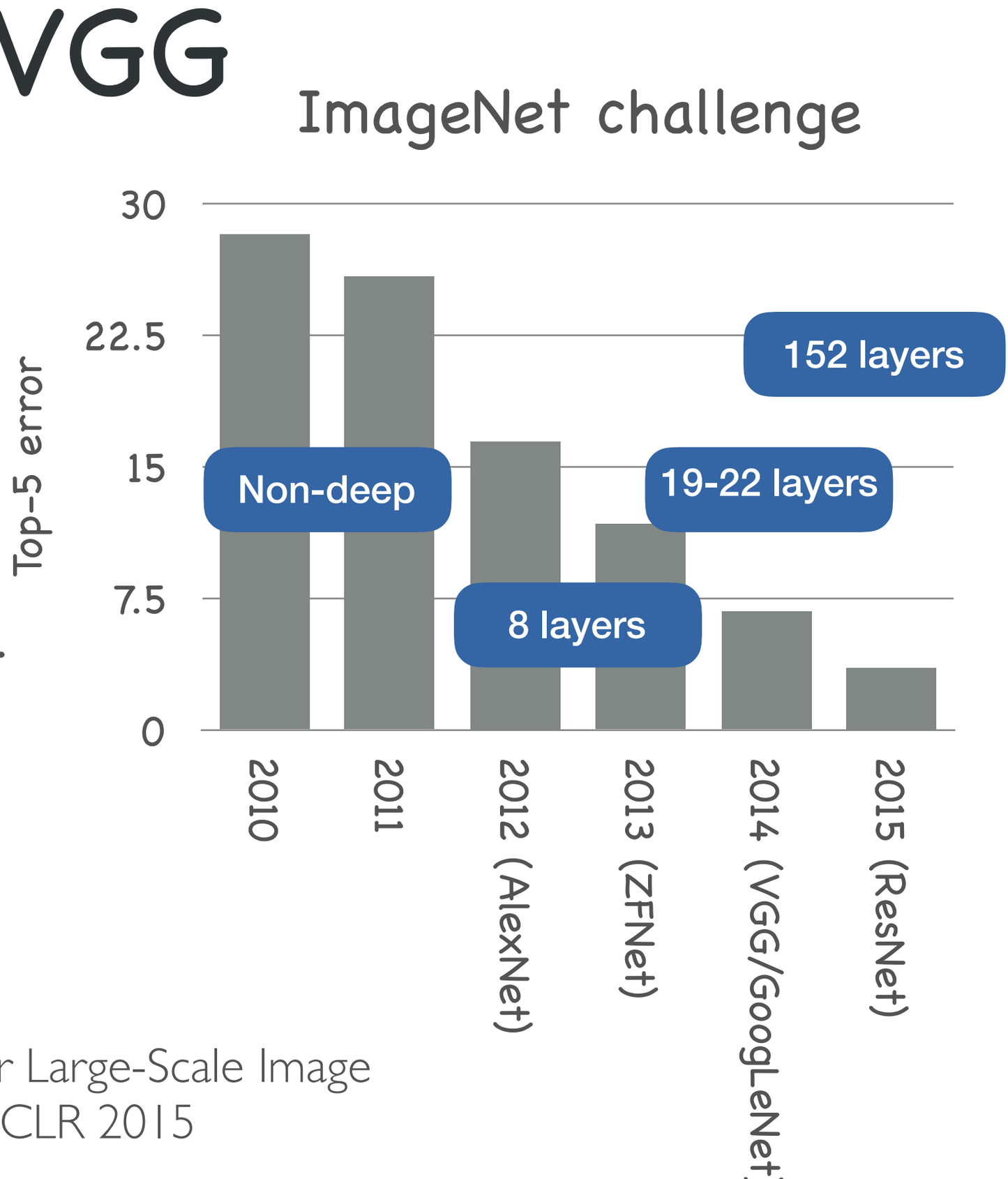
Parameters and computation



Case Study: VGG

VGG

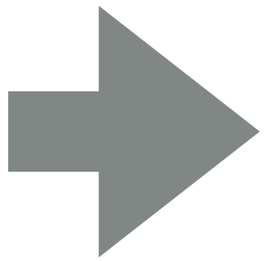
- Deeper AlexNet/ZFNet



Very Deep Convolutional Networks for Large-Scale Image Recognition, Simonyan and Zisserman, ICLR 2015

ZFNet to VGG

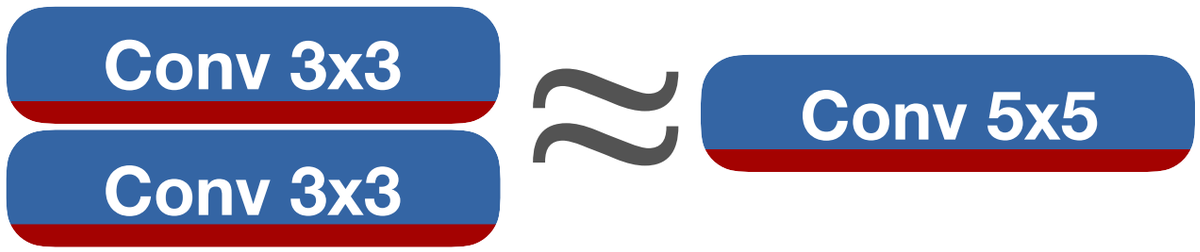
Conv 7x7	stride=2, C=96
LRN	
Max Pool 3x3	stride=2
Conv 5x5	stride=2, C=256
LRN	
Max Pool 3x3	pad=0, stride=2
Conv 3x3	C=512
Conv 3x3	C=1024
Conv 3x3	C=512
Max Pool 3x3	pad=0, stride=2
Linear (4096)	
Linear (4096)	
Linear (1000)	



Conv 3x3	C=64
Conv 3x3	C=64
Max Pool 2x2	stride=2
Conv 3x3	C=128
Conv 3x3	C=128
Max Pool 2x2	stride=2
Conv 3x3	C=256
Conv 3x3	C=256
Conv 3x3	C=256
Max Pool 2x2	stride=2
Conv 3x3	C=512
Conv 3x3	C=512
Conv 3x3	C=512
Max Pool 2x2	stride=2
Conv 3x3	C=512
Conv 3x3	C=512
Conv 3x3	C=512
Max Pool 2x2	stride=2
Linear (4096)	
Linear (4096)	
Linear (1000)	

Insights in VGG

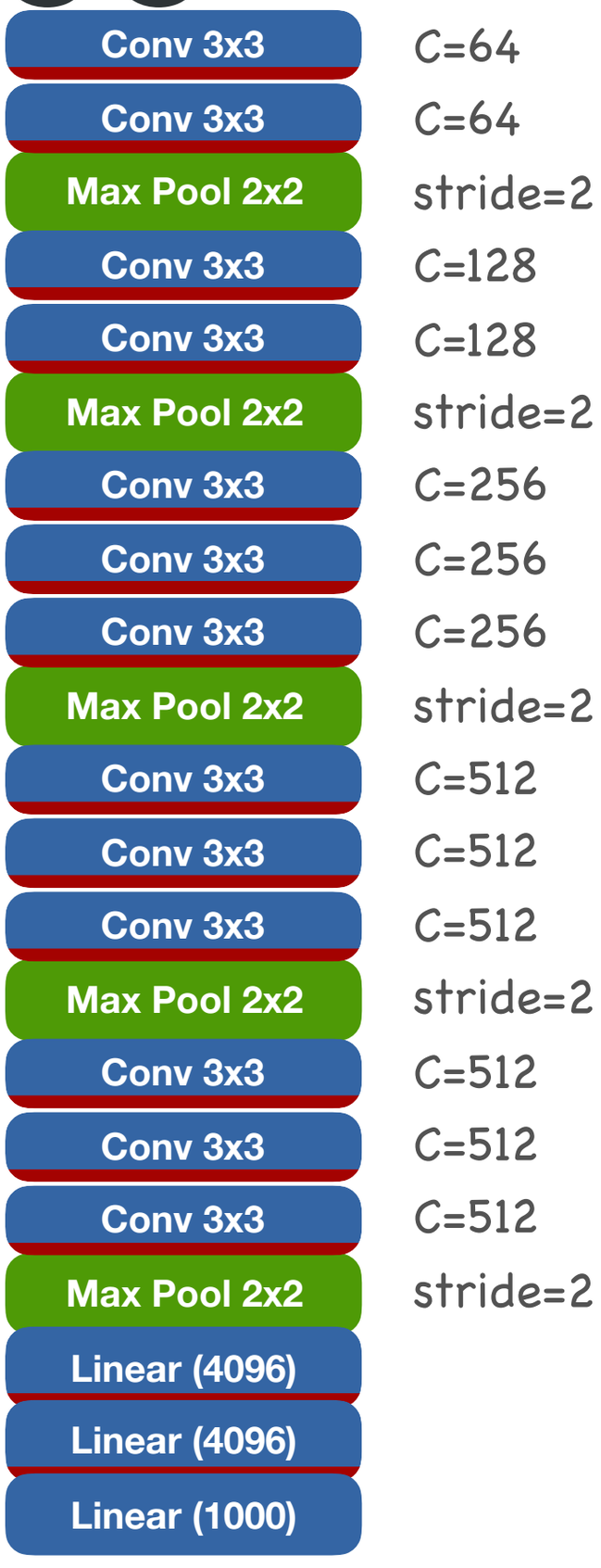
- Why use smaller filters?



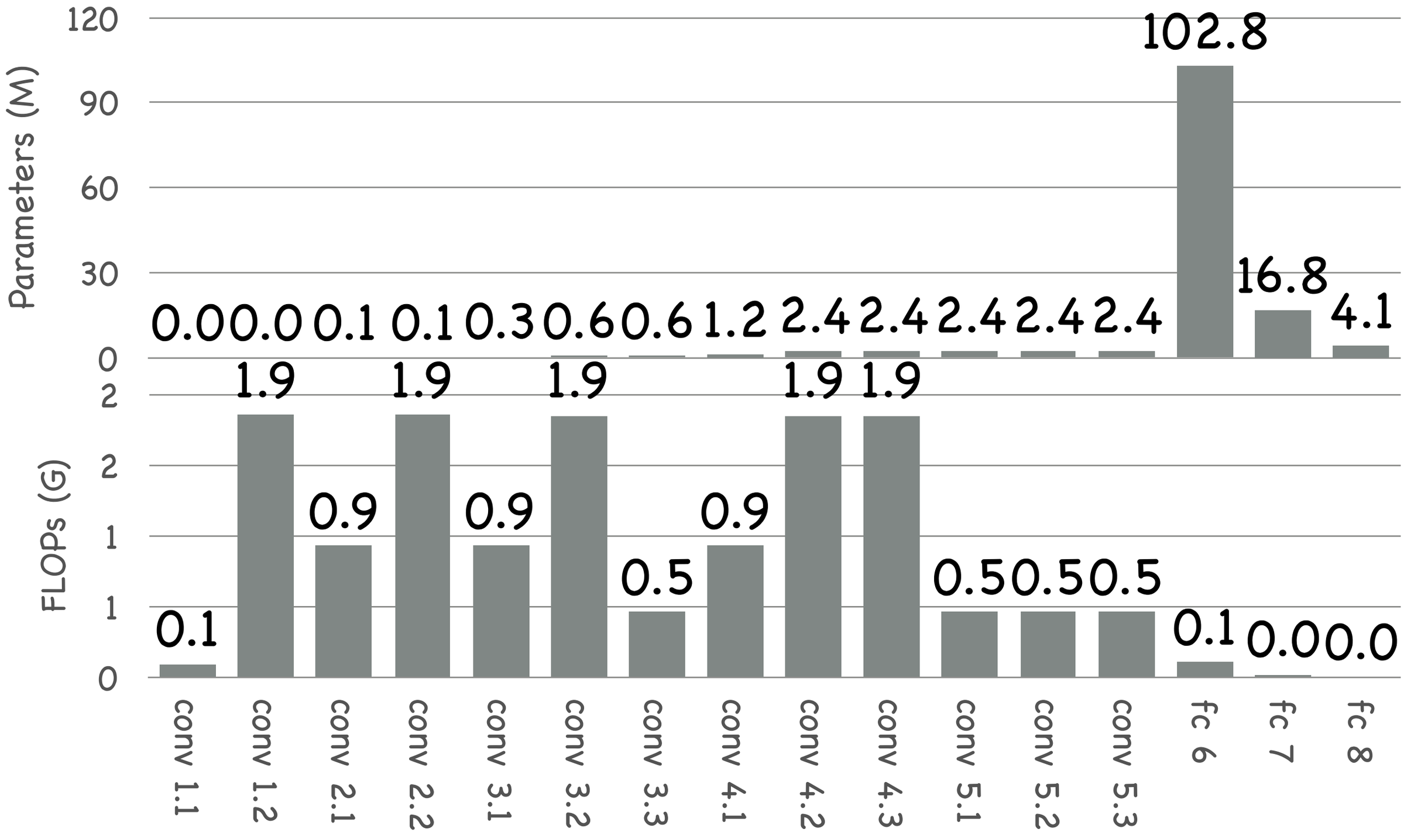
  - Factorization

Training VGG

- Vanishing gradients

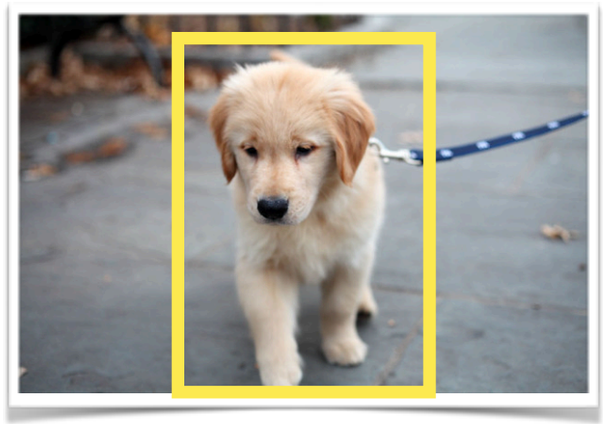


Parameters and computation



VGG

- Generalizes well to other tasks
  - Detection
  - Segmentation
  - Style-transfer (graphics)



Fully Convolutional Networks for Semantic Segmentation, Shelhamer et al., CVPR, 2015
Image style transfer using convolutional neural networks, Gatys et al., CVPR, 2016
Rich feature hierarchies for accurate object detection and semantic segmentation, Girshick et al., CVPR, 2014

1x1 convolutions and factorization

1x1 convolutions and factorization

- Convolutions are linear operators
- **Can we make them non-linear?**

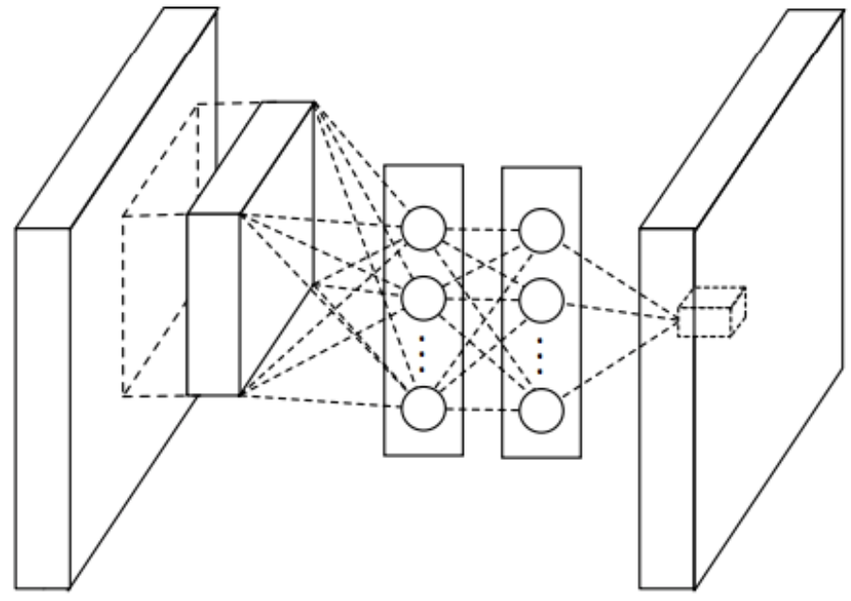
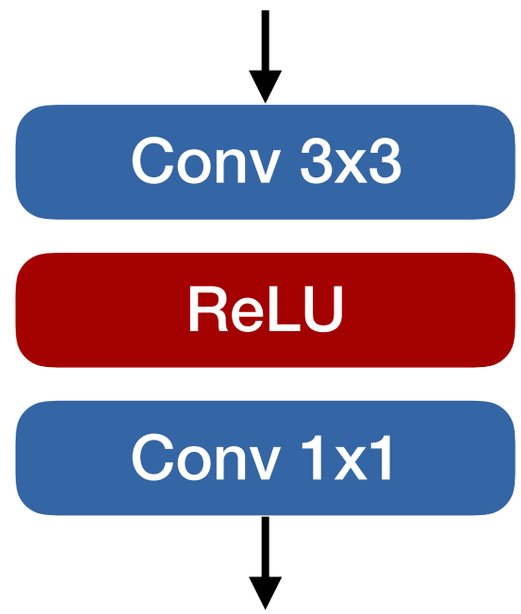


Figure source: "Network-in-Network", ICLR 2014 paper

Network-in-Network, Lin et al., ICLR 2014

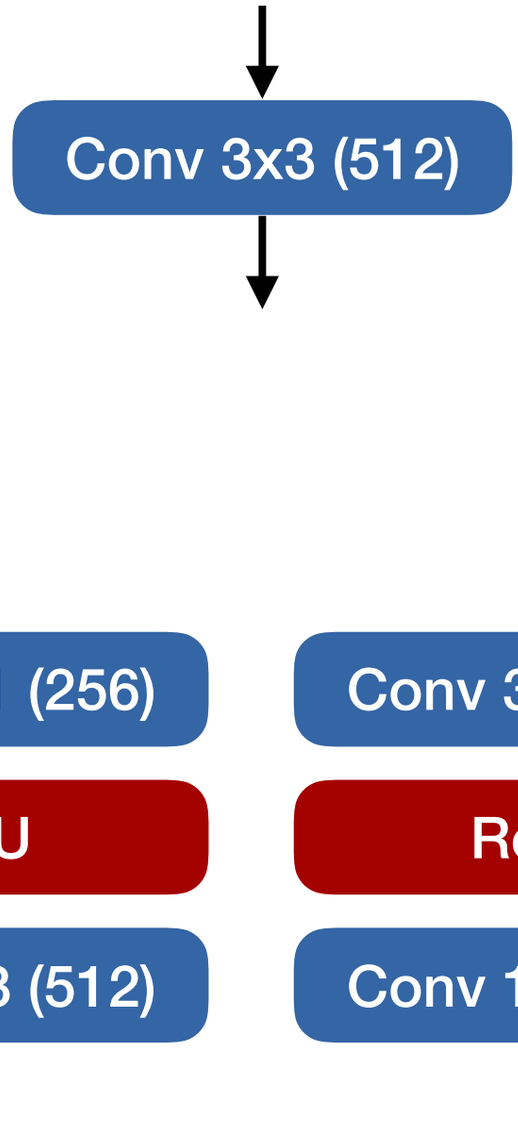
Factorized convolution

- Implements “non-linear” convolution



Why factorized convolution?

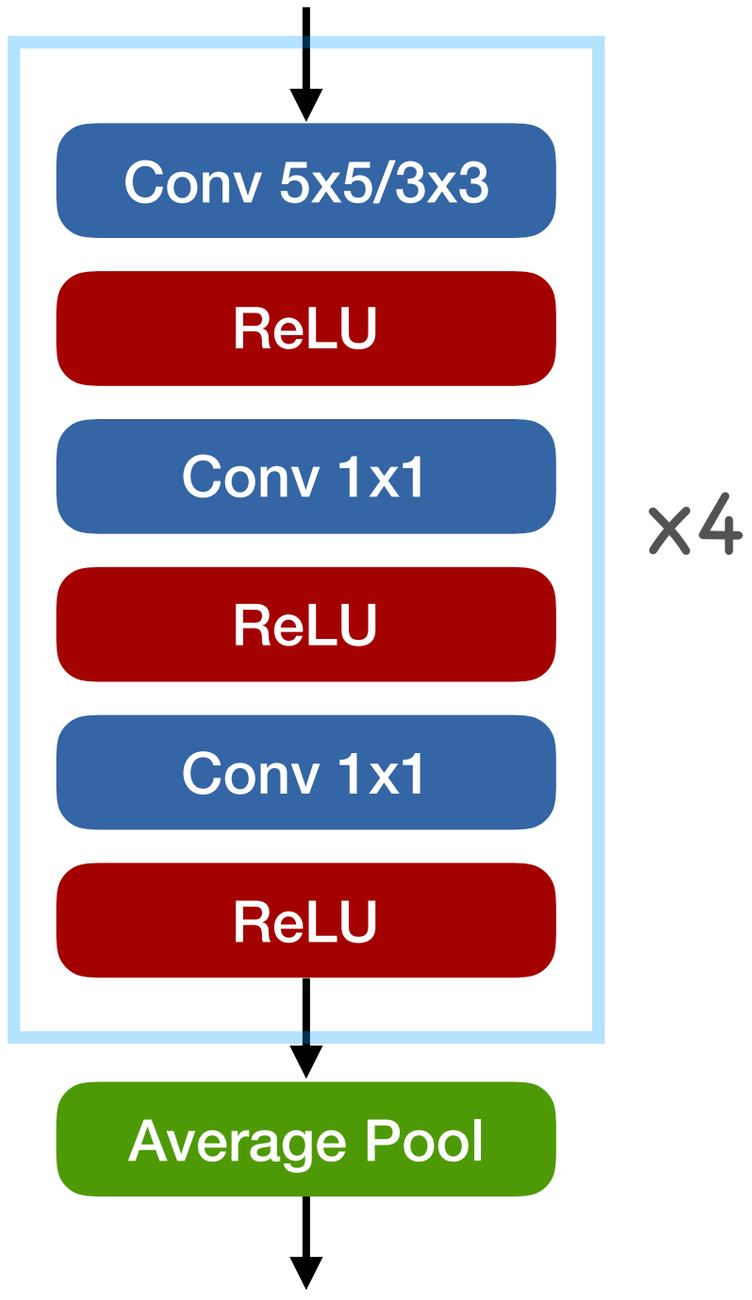
- Less computation
- Fewer parameters



Case Study: Network in Network

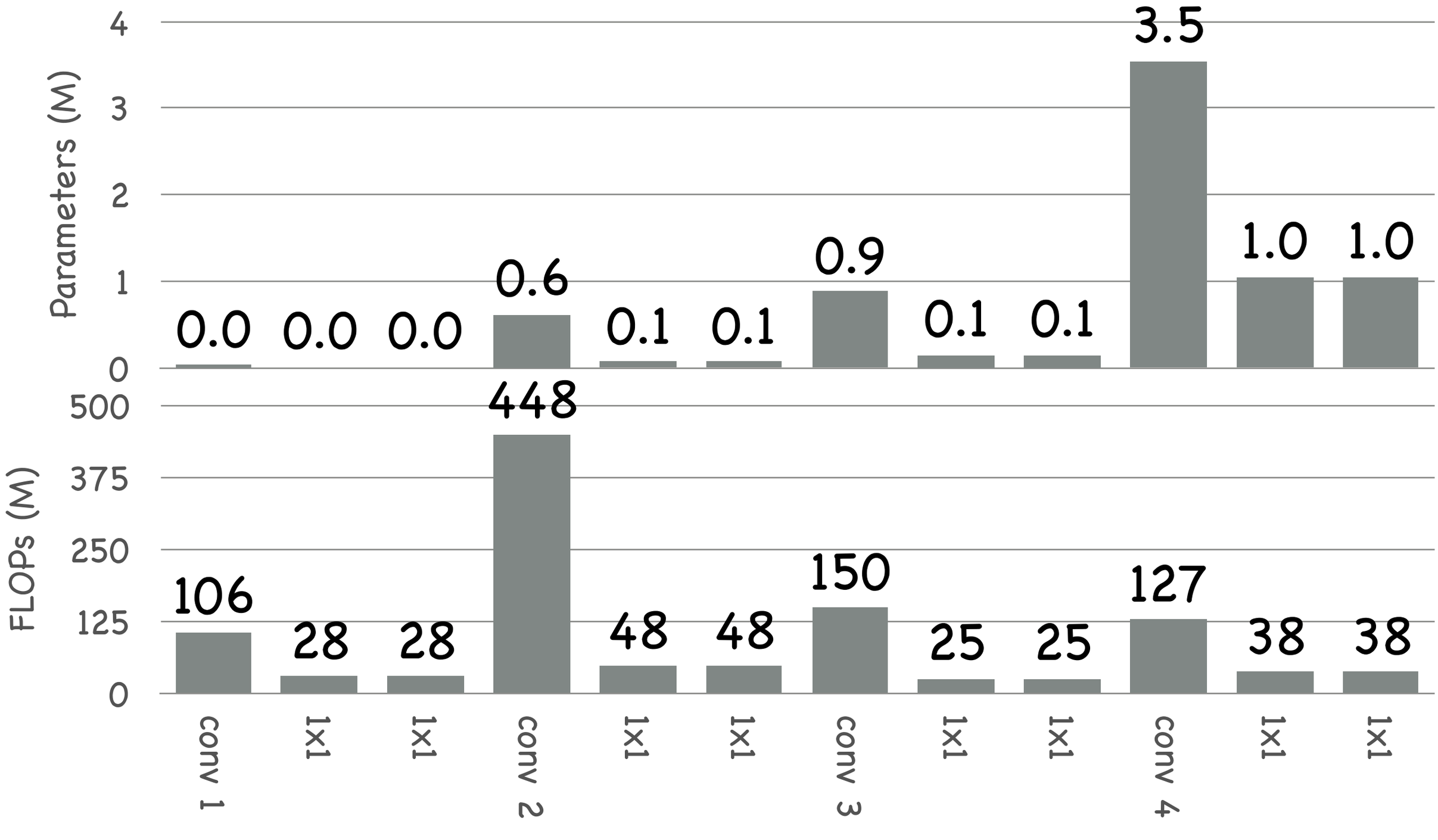
Network in Network

- Small and efficient network
- Factorized convolutions
- Global average pooling



Network-in-Network, Lin et al., ICLR 2014

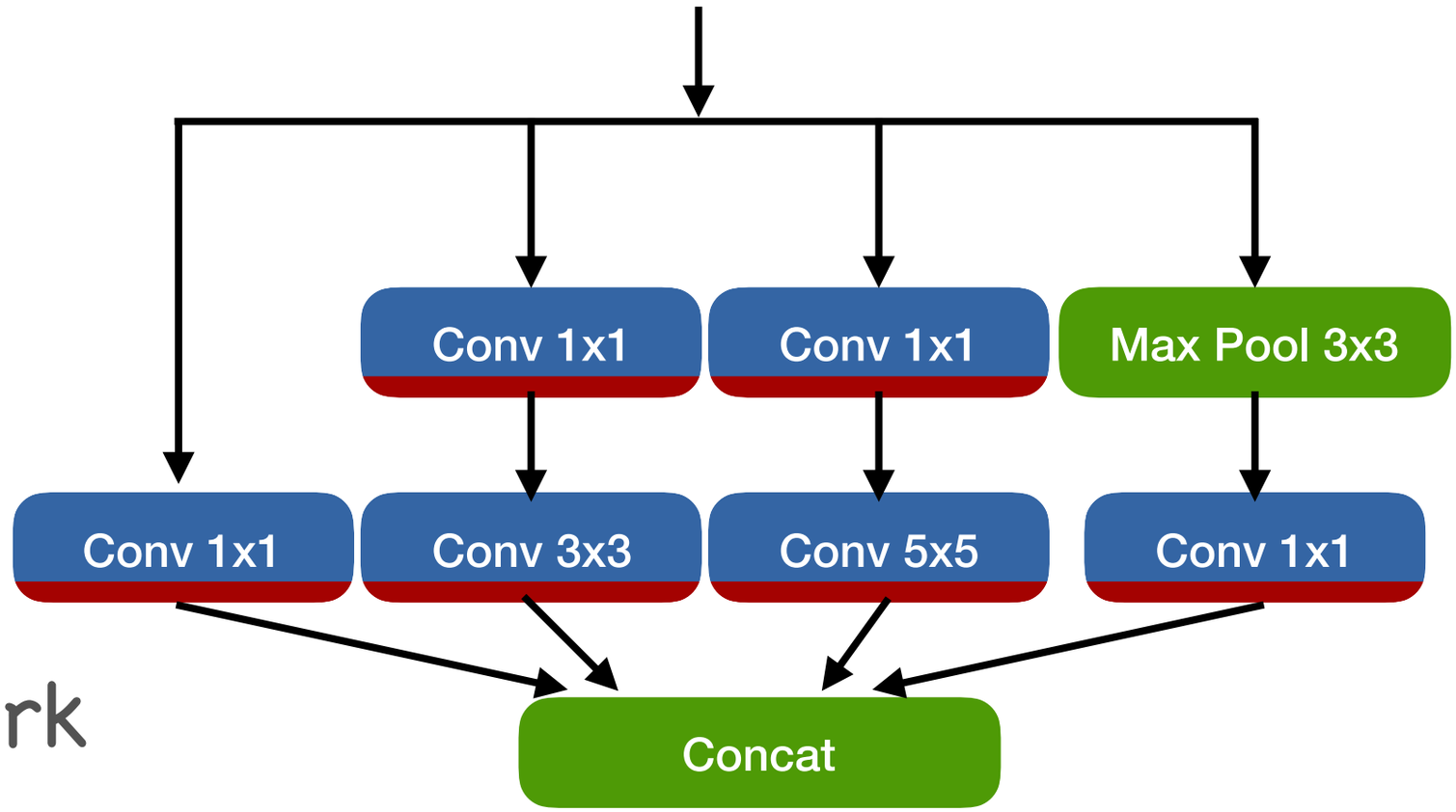
Parameters and computation



Case Study: Inception architecture

Inception architecture

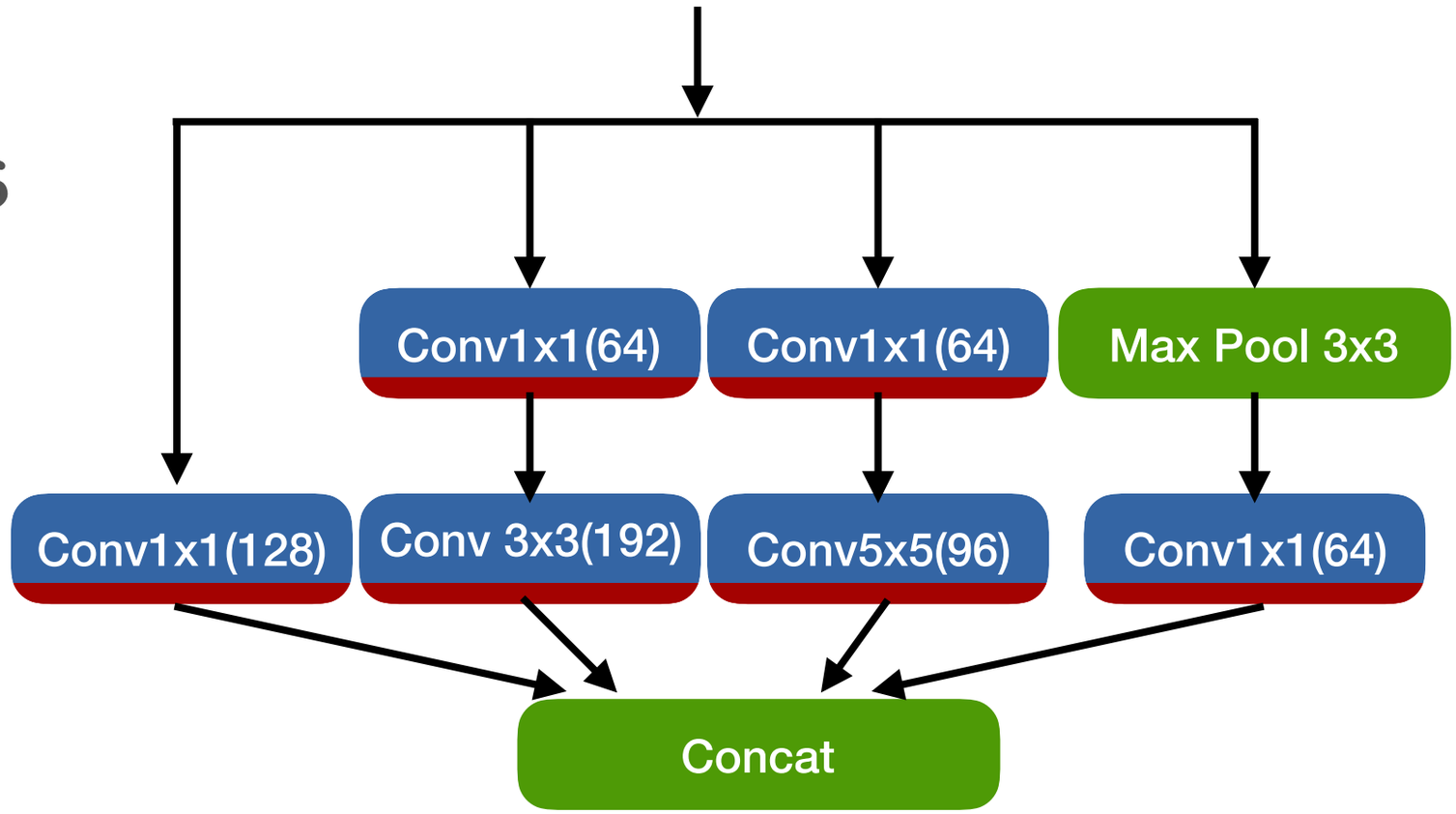
- GoogLeNet
  - Evolution of Network-in-Network



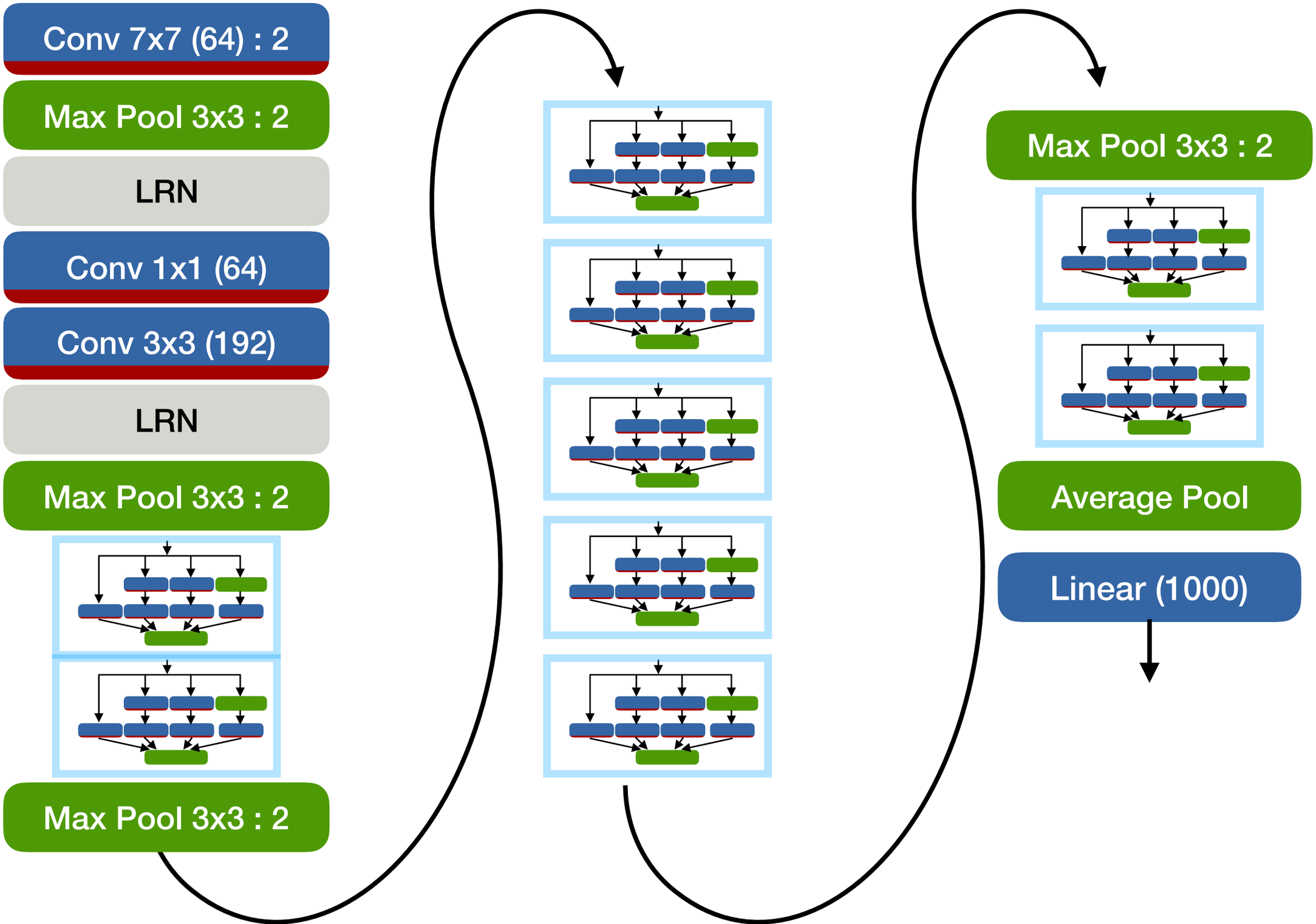
Going deeper with convolutions, Szegedy et al. CVPR 2015.

Inception module

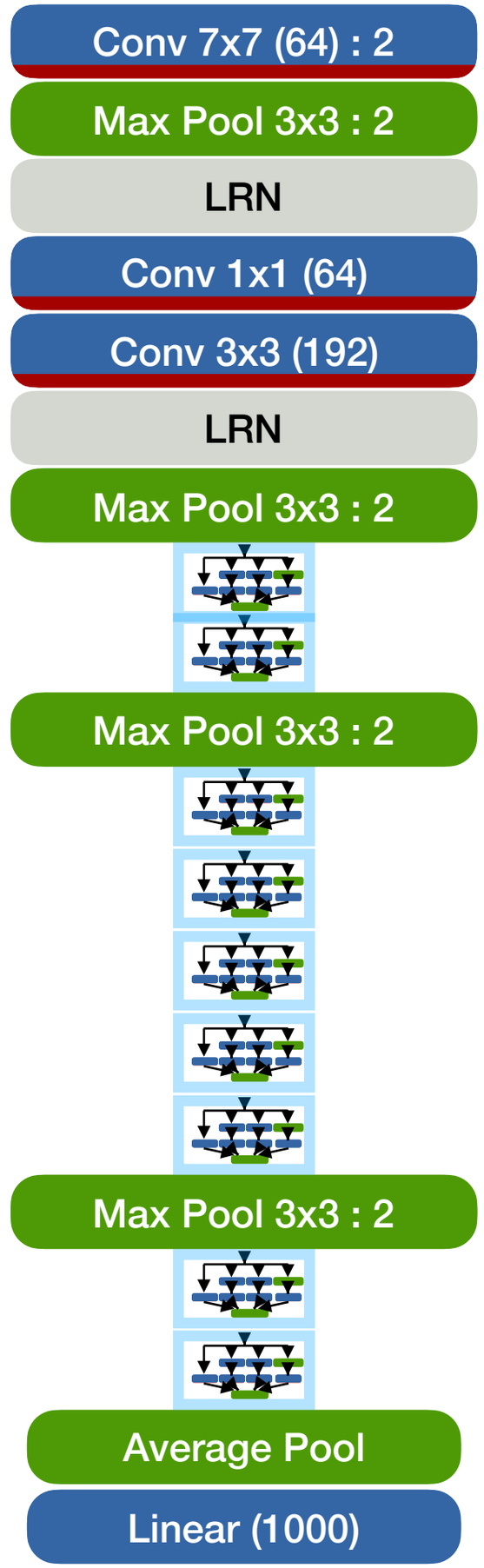
- Multiple kernel sizes
  - Handle small and large objects
- Factorization
  - Fewer parameters



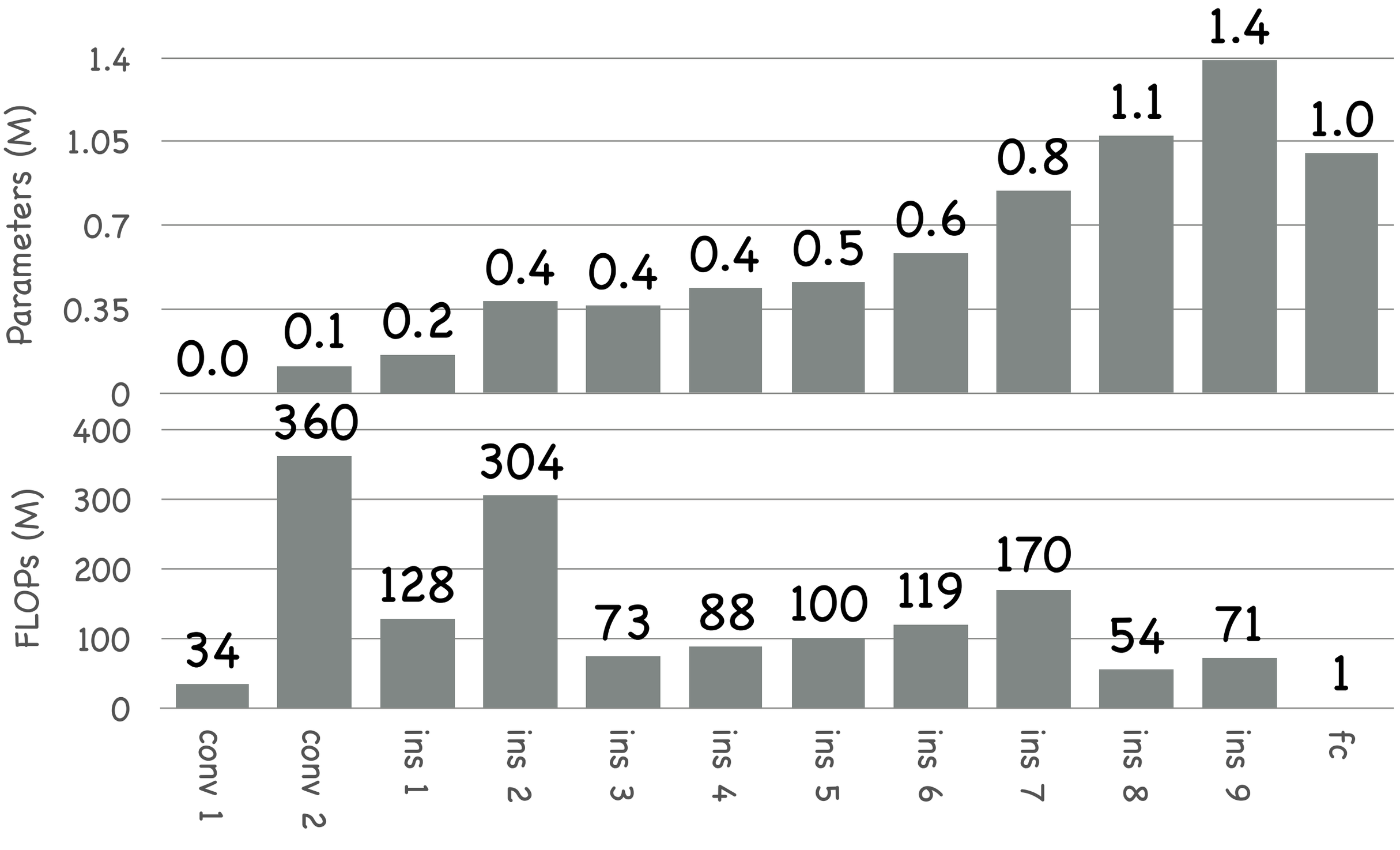
Inception architecture



Training inception



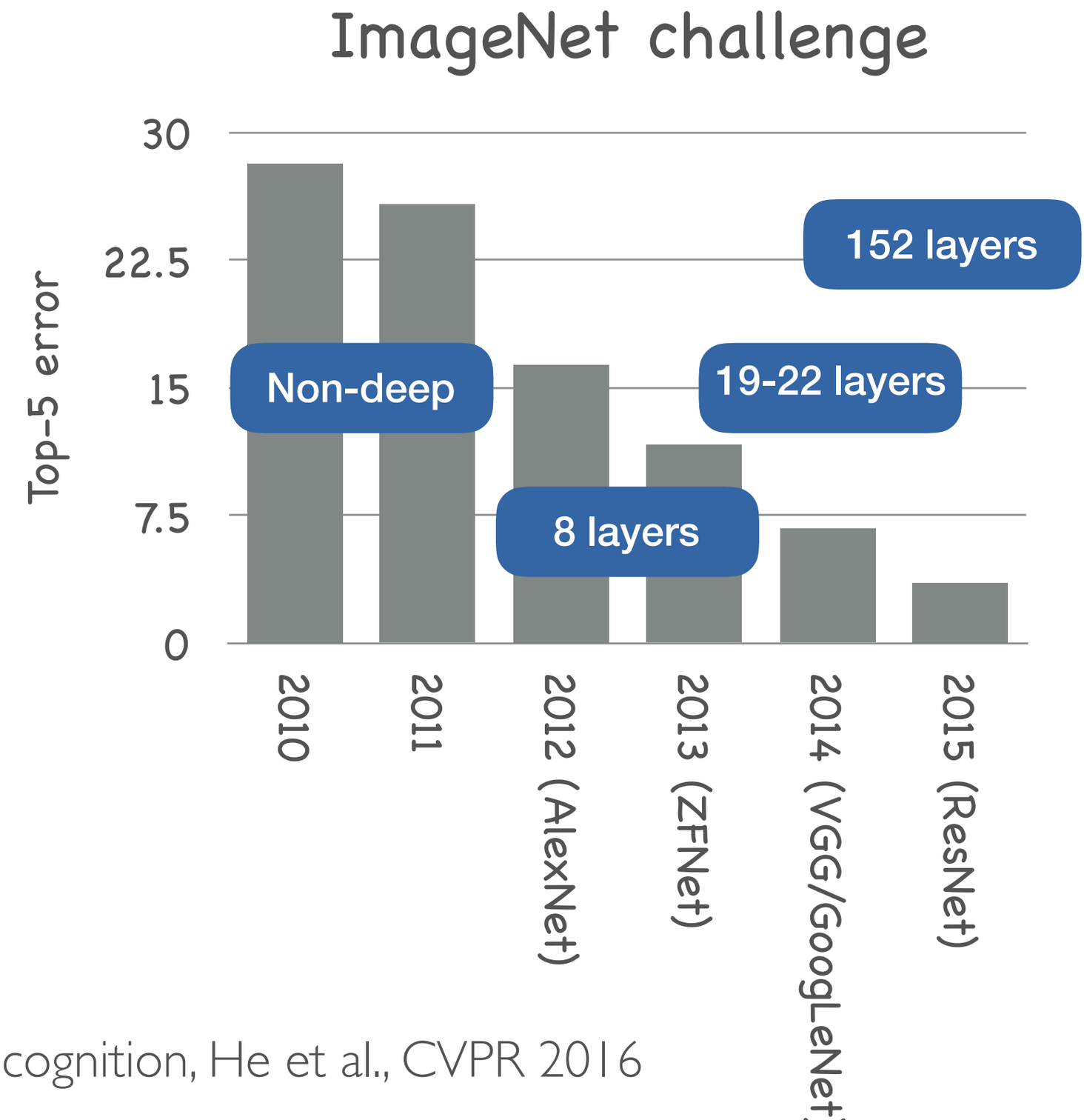
Parameters and computation



Case study: Residual Networks

Residual Networks

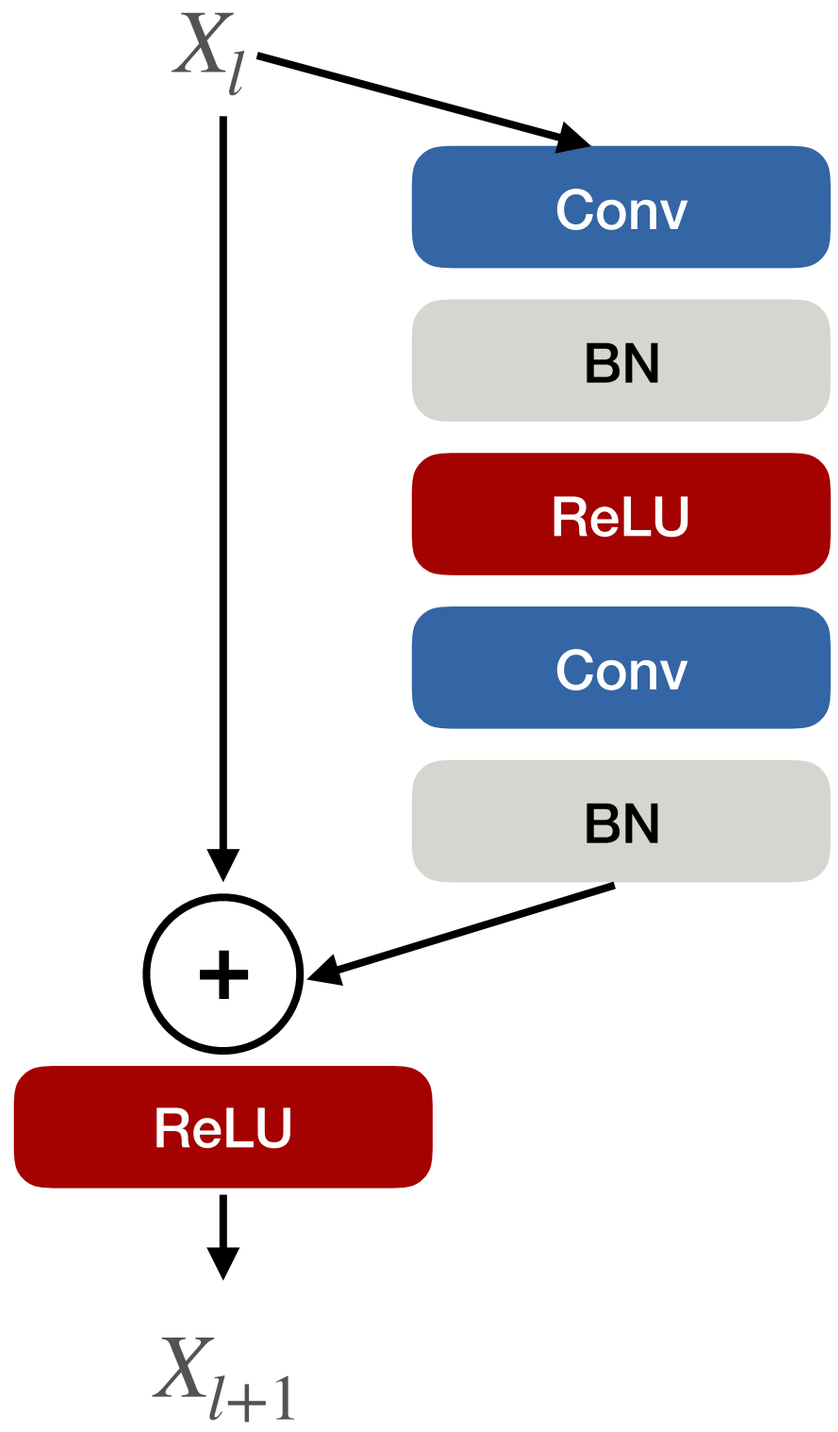
- Uses residual connections to build deeper networks
- Activation maps are additive



Deep Residual Learning for Image Recognition, He et al., CVPR 2016

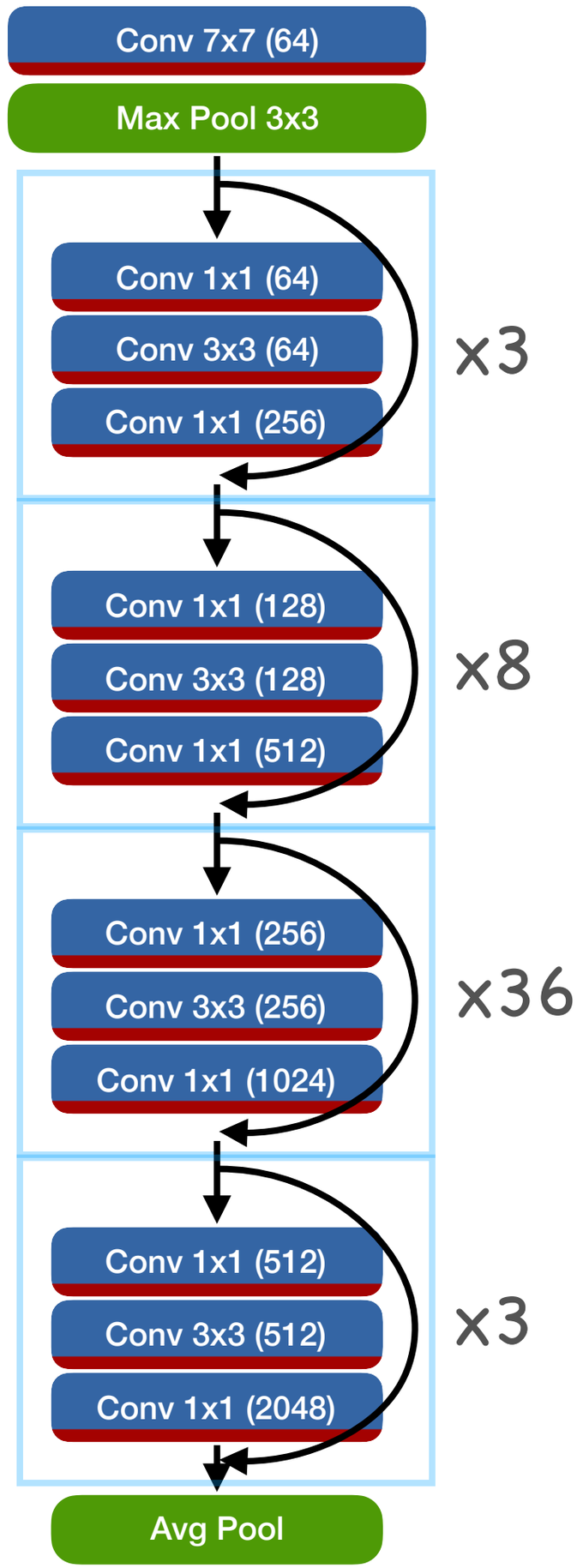
Residual blocks

- Add shortcut connections for gradients
- Identity
- Strided 1x1 convolution



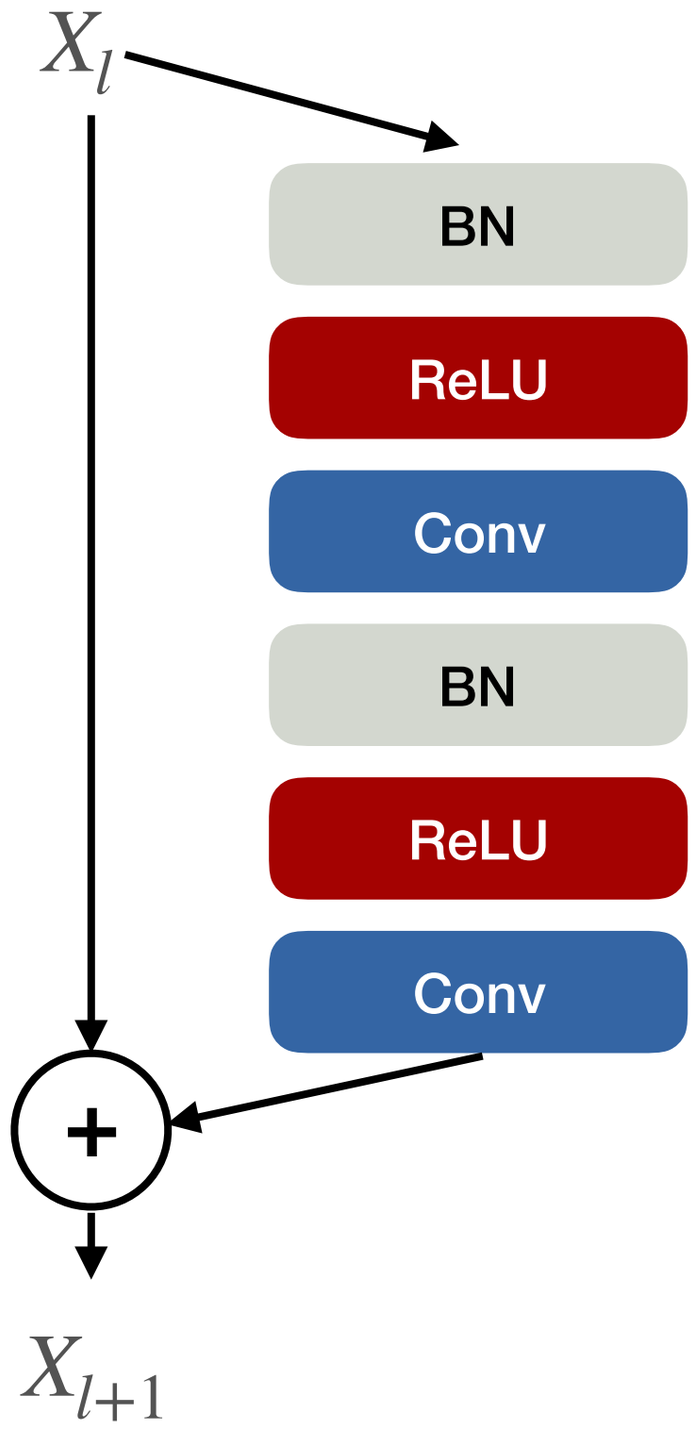
ResNet

- Multiple variants
- ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, ResNet-1001



Pre-activation block

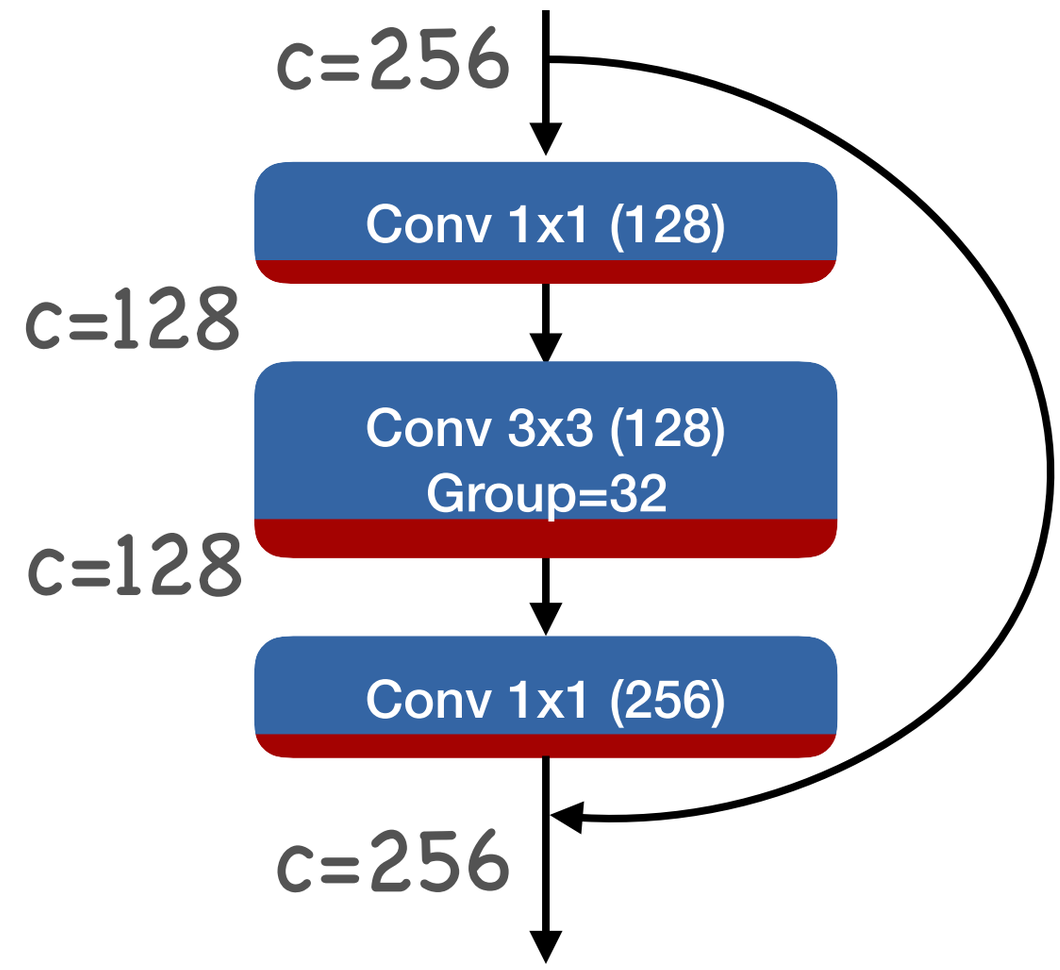
- Pure identity connections
- Allows for deeper networks
- Trains better



Identity Mappings in Deep Residual Networks, He et al., ECCV 2016

ResNeXt

- Group convolutions in residual blocks
  - More channels
  - Fewer parameters / computation

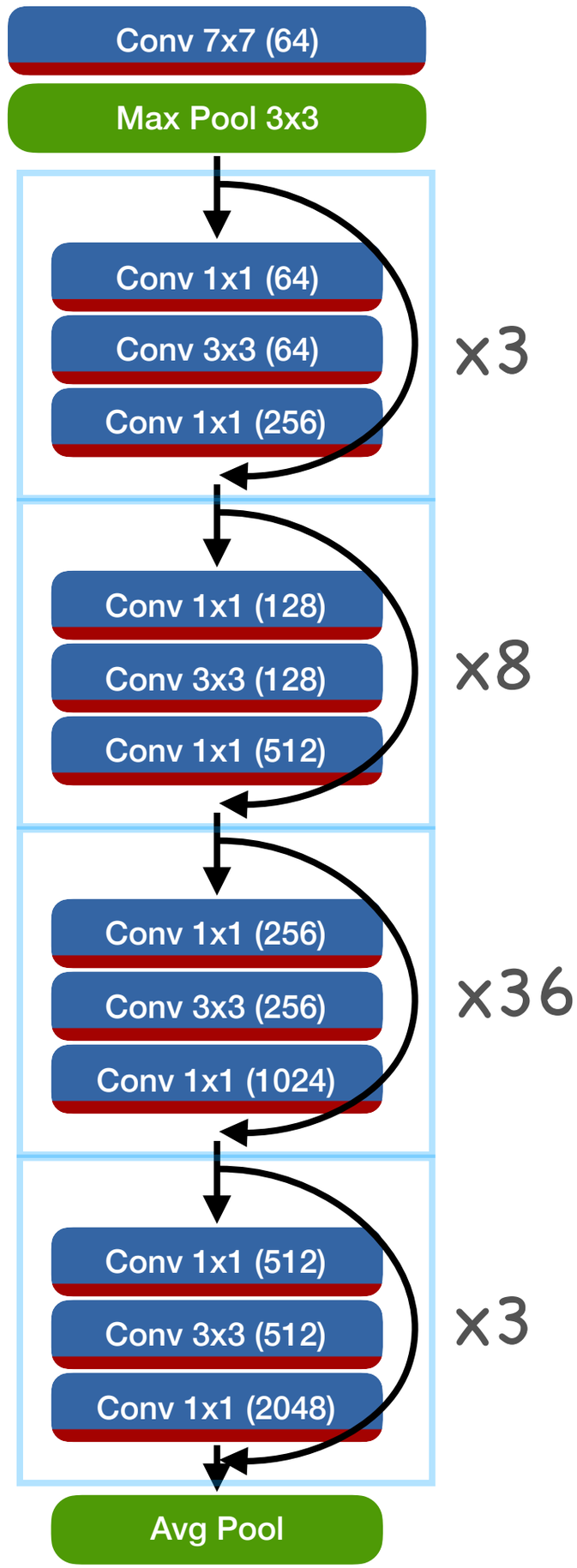


Aggregated Residual Transformations for Deep Neural Networks, Xie et al., CVPR 2017

Stochastic depth

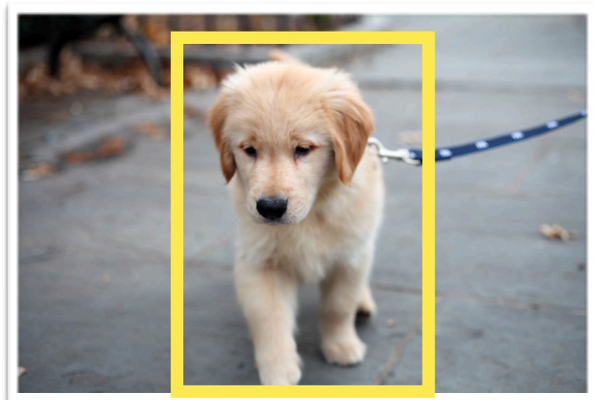
- Layer "dropout"
- Less overfitting

Deep Networks with Stochastic Depth, Huang et al., ECCV 2016



Uses of ResNet

- Anywhere in computer vision



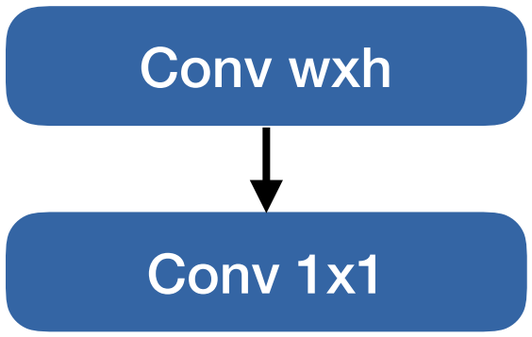
Factorization and light-weight networks

Factorization beyond 1x1 convolutions

- Parameters and computation of convolution
  - $\mathcal{O}(whC_1C_2)$
- Factorized 1x1 convolutions
  - $\mathcal{O}(whC_1C_2 + C_2C_3)$

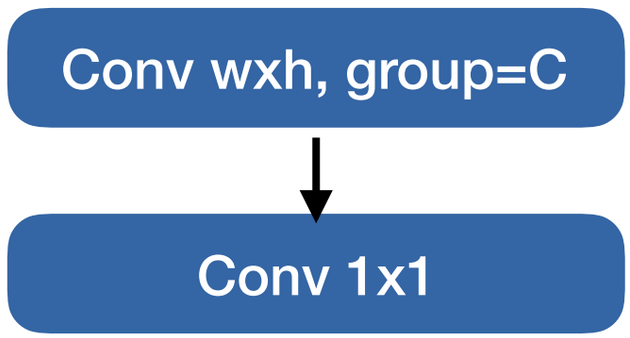
Regular convolution

Factorized 1x1 convolution



Factorization beyond 1x1 convolutions

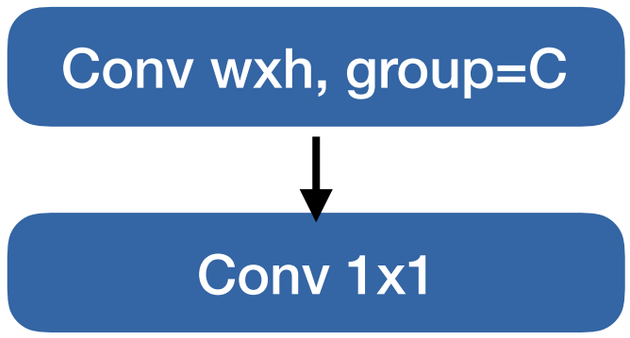
- Factorized depthwise convolutions
  - $\mathcal{O}(whC_1 + C_1C_2)$



MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al., arXiv 2017

Limitations

- Only learns a single spatial filter per channel



Case study: MobileNet

MobileNet

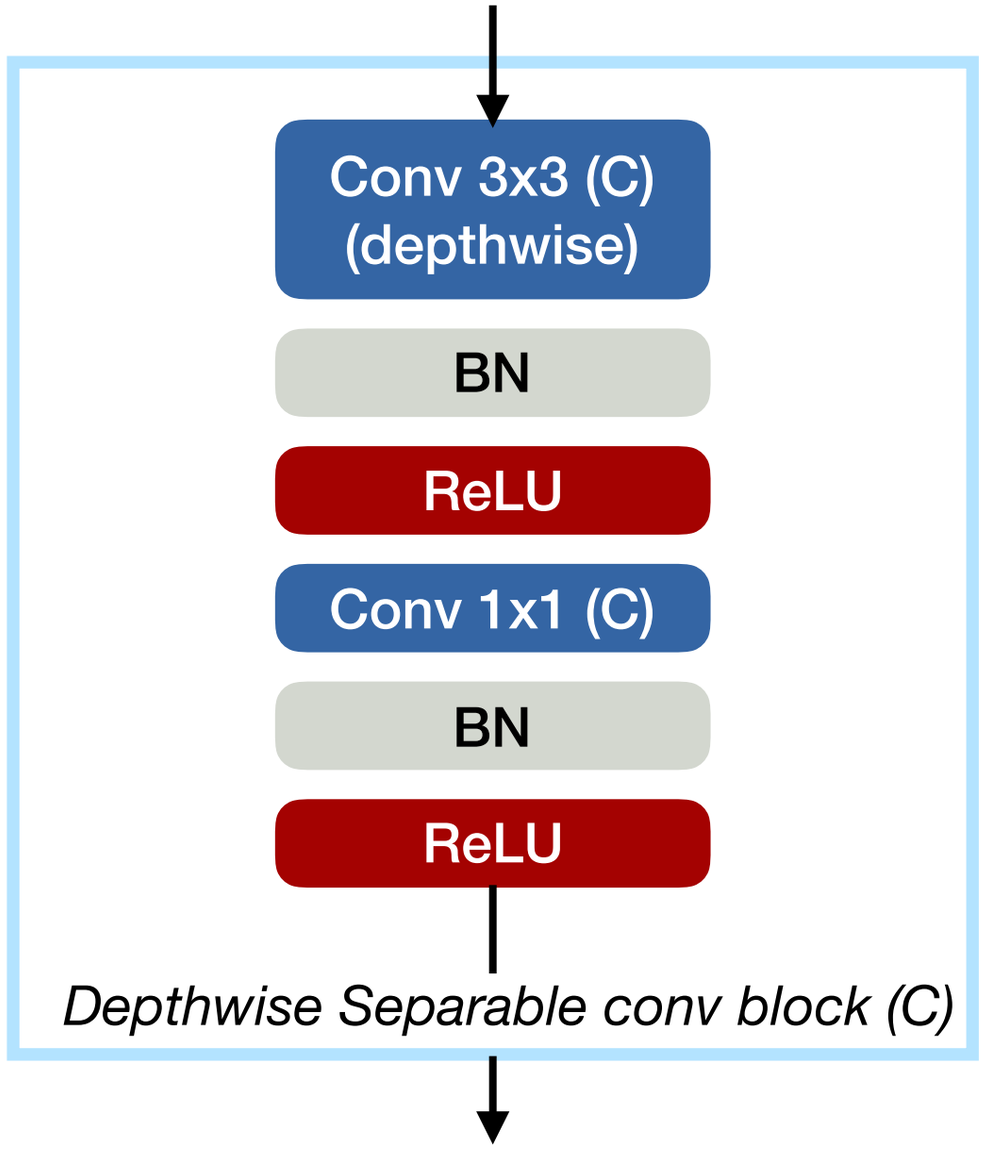
- Fast and small architecture for phones

- Heavily uses factorized depthwise convolutions

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al., arXiv 2017

Basic blocks in MobileNet building

- Factorized depthwise convolutions



MobileNet

Conv 7x7

Depthwise Sep Conv (32)

Depthwise Sep Conv (64)

Depthwise Sep Conv (128)

Depthwise Sep Conv (128)

Depthwise Sep Conv (256)

Depthwise Sep Conv (256)

Depthwise Sep Conv (512)

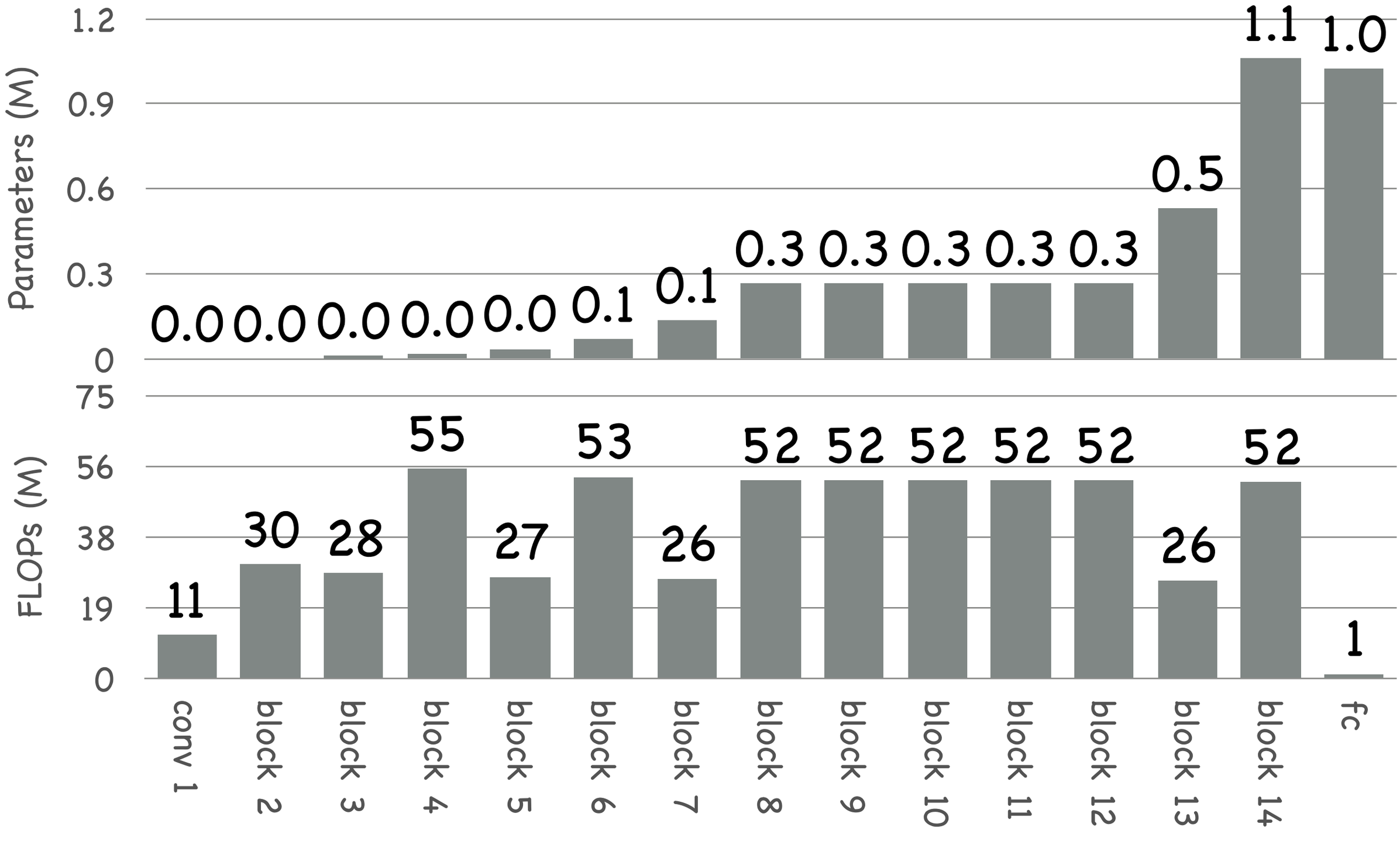
Depthwise Sep Conv (512)

Depthwise Sep Conv (1024)

Average Pool

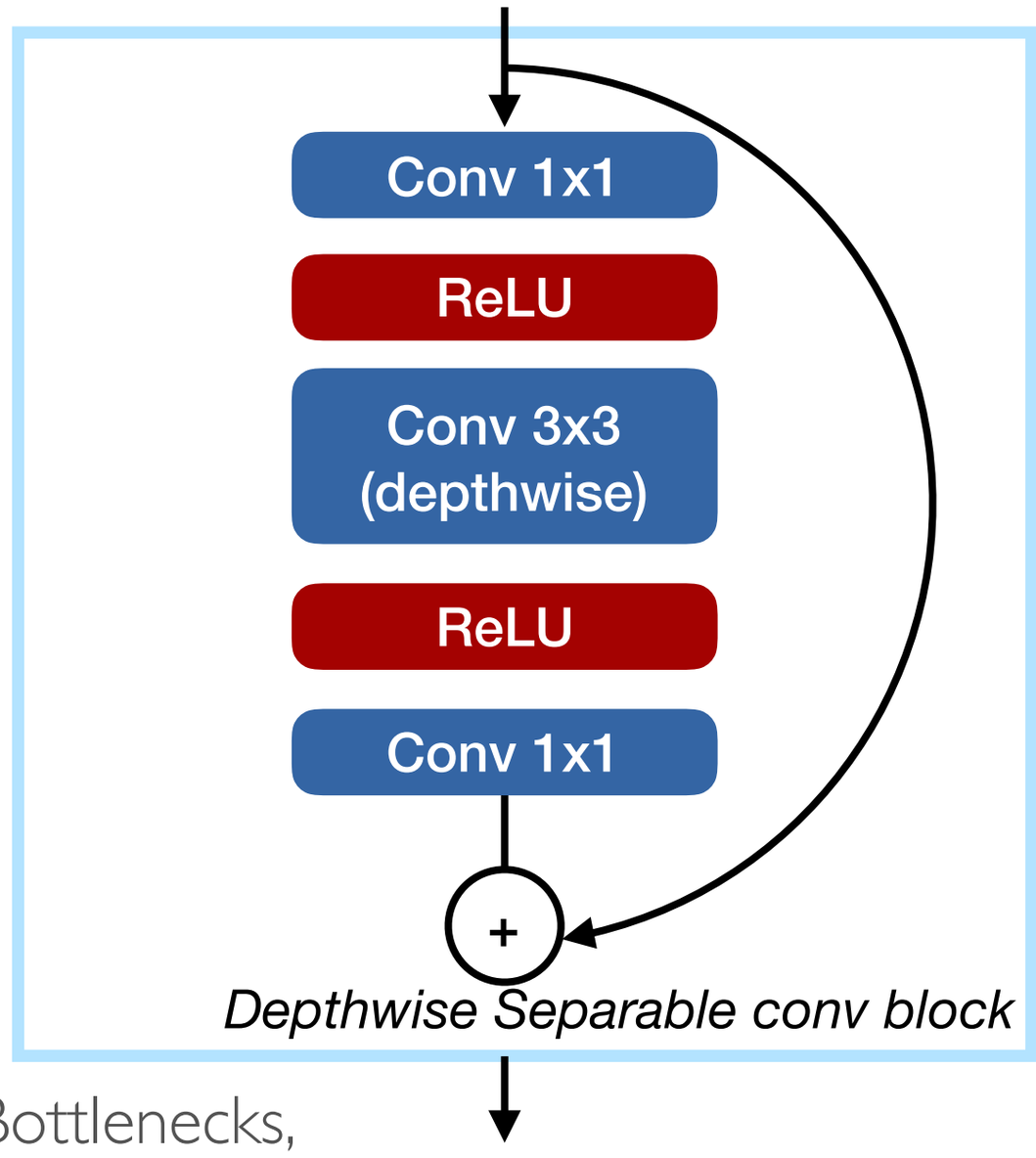
Linear (1000)

Parameters and computation



MobileNet v2

- Adds residual connections
  - Multiple spatial filter per channel
    - Adding additional 1x1 conv



MobileNetV2: Inverted Residuals and Linear Bottlenecks, Sandler et al., CVPR 2018

10

January 23, 2024

[1]:
```
%pylab inline
import torch
import torchvision
from PIL import Image
import json
class_idx = json.load(open("imagenet_class_index.json"))
I1 = Image.open('dog.jpg')
I2 = Image.open('cat.jpg')
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

[ ]:
```
model = torchvision.models.resnext101_32x8d(pretrained=True, progress=False)
# model = torchvision.models.mobilenet_v2(pretrained=True, progress=False)
```

[4]:
```
transform = torchvision.transforms.Compose([
    torchvision.transforms.Resize(224),
    torchvision.transforms.CenterCrop(224),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.
 ↪224, 0.225])])

model.eval()

p = model(transform(I1)[None])[0]
print( ' , '.join([class_idx[str(int(i))][1] for i in p.
 ↪argsort(descending=True)[:5]]) )
p = model(transform(I2)[None])[0]
print( ' , '.join([class_idx[str(int(i))][1] for i in p.
 ↪argsort(descending=True)[:5]]) )
```

```
wire-haired_fox_terrier , toy_terrier , Brabancon_griffon , standard_schnauzer ,
Lakeland_terrier
tiger_cat , tabby , Egyptian_cat , lynx , tiger
```

[5]:
```
model.fc = torch.nn.Linear(2048, 10)
```

[6]:
```
model(transform(I)[None])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 model(transform(I)[None])

NameError: name 'I' is not defined
```

[ ]:

Object Detection

Object Detection

- Find / label all objects in an image
  - Classify and put a box around them
- Most popular sparse labeling task
  - Basis of many computer vision tasks

Datasets: Pascal VOC

- 20 classes
- 11k images
  - 27k objects

The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, Everingham et al., 2007.

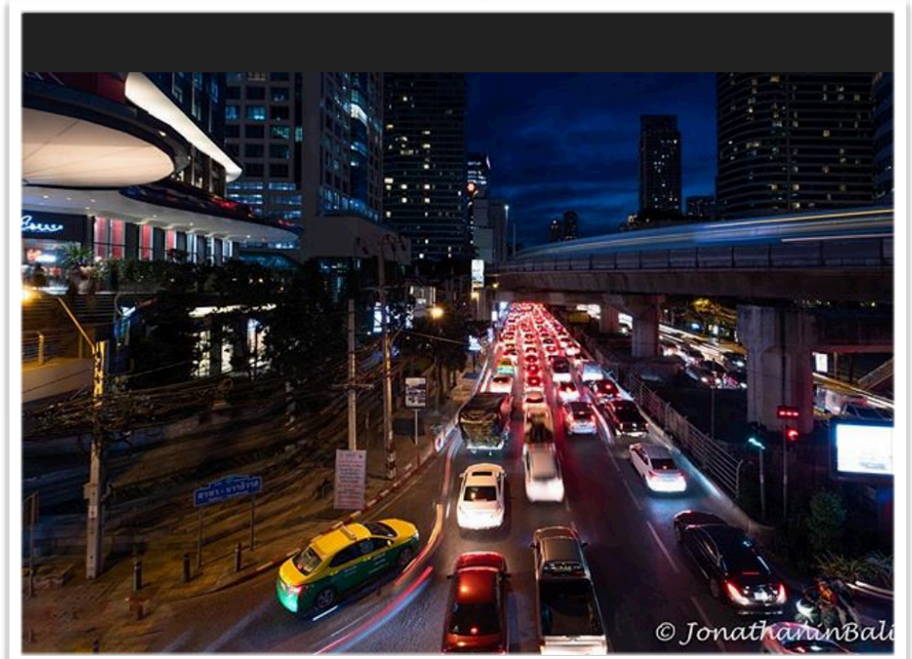
Datasets: MS COCO

- 80 classes
- 200k images
  - 1.5M objects
  - 250k people w. pose

Microsoft COCO: Common Objects in Context, Lin et al., ECCV 2014

Driving datasets

- 3d object detection and tracking
  - NuScenes
  - BDD100k
  - Mapillary
  - ApploScape

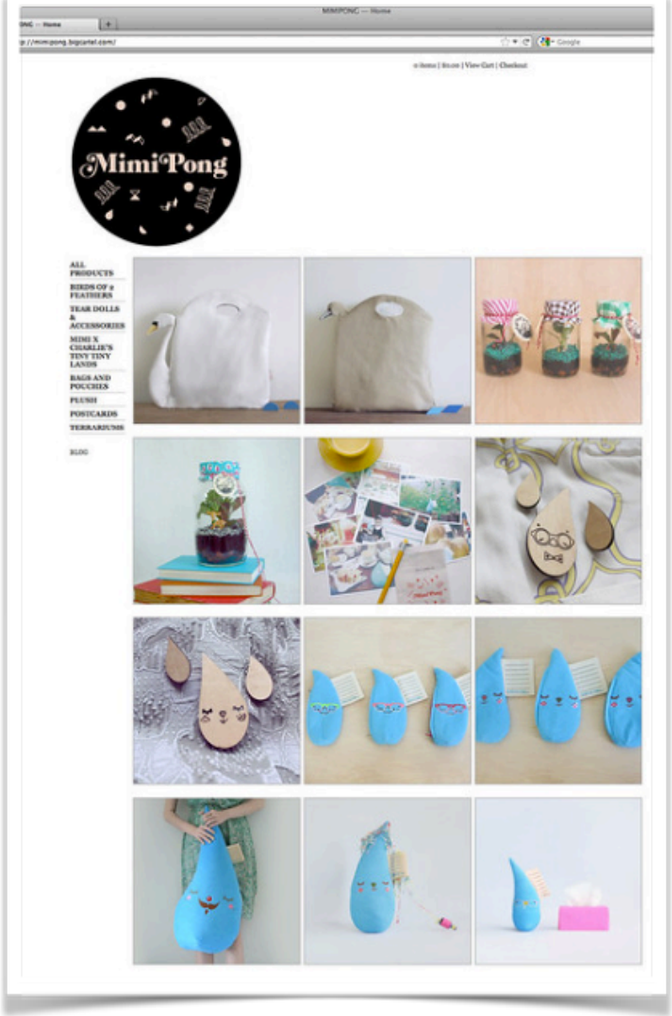


Simulators

- Dataset → Simulation
  - Infinite data
  - Free labels
  - Limited domain
- Examples
  - GTA V
  - Carla
  - Habitat

Applications

- Visual search
- Advertising
- Surveillance
  - 😀 Assisted living
  - 😭 State-sponsored
- Assisted driving



Case study: RCNN

Object Detection

- ConvNets classify images well

- How can we use ConvNets to detect objects?

  - e.g. find object locations?

RCNN

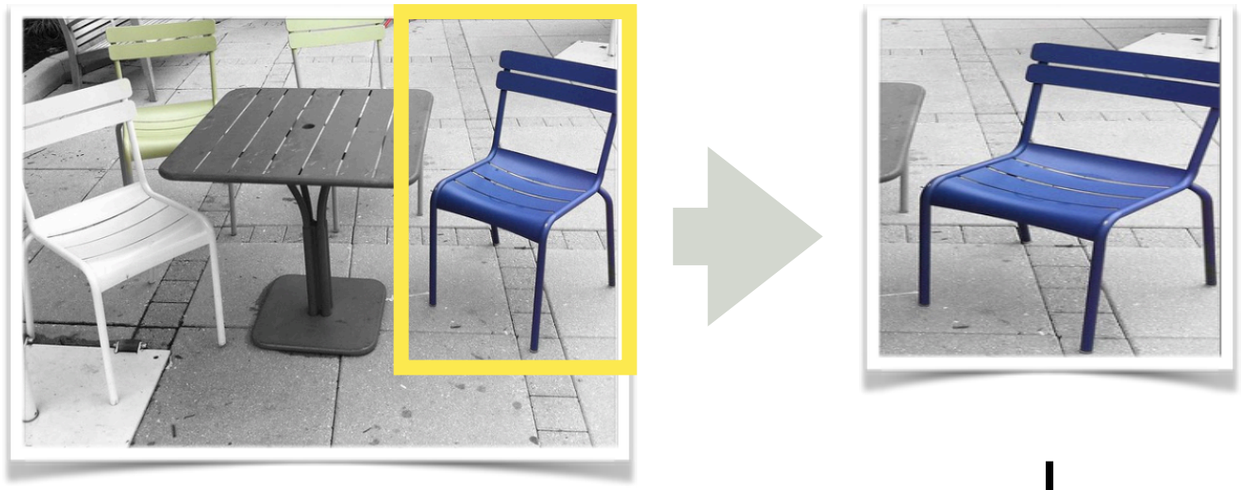
- Object detection as region classification

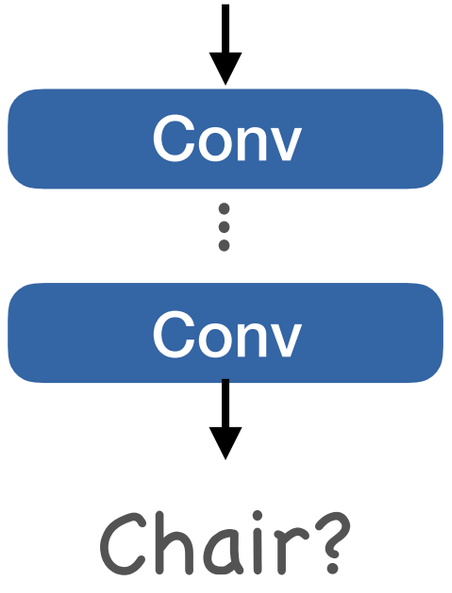
- Object or not?

Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation
Girshick et al., CVPR 2014

RCNN - basic idea

- Crop and resize image regions
- Classify all regions
  - Object or not?



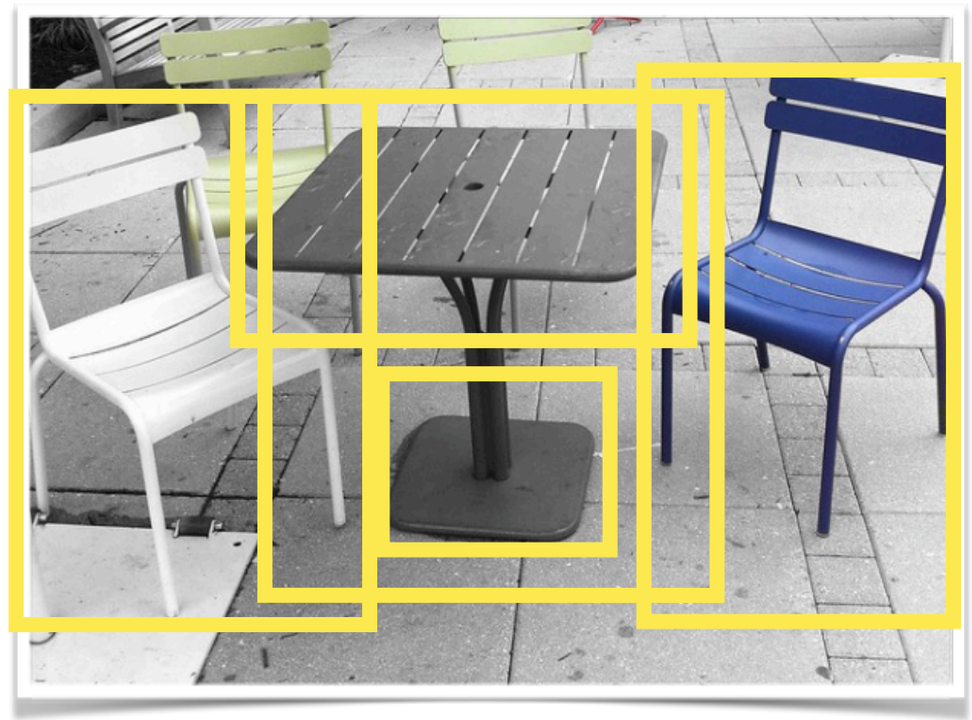


RCNN - Issues

- There are many potential regions

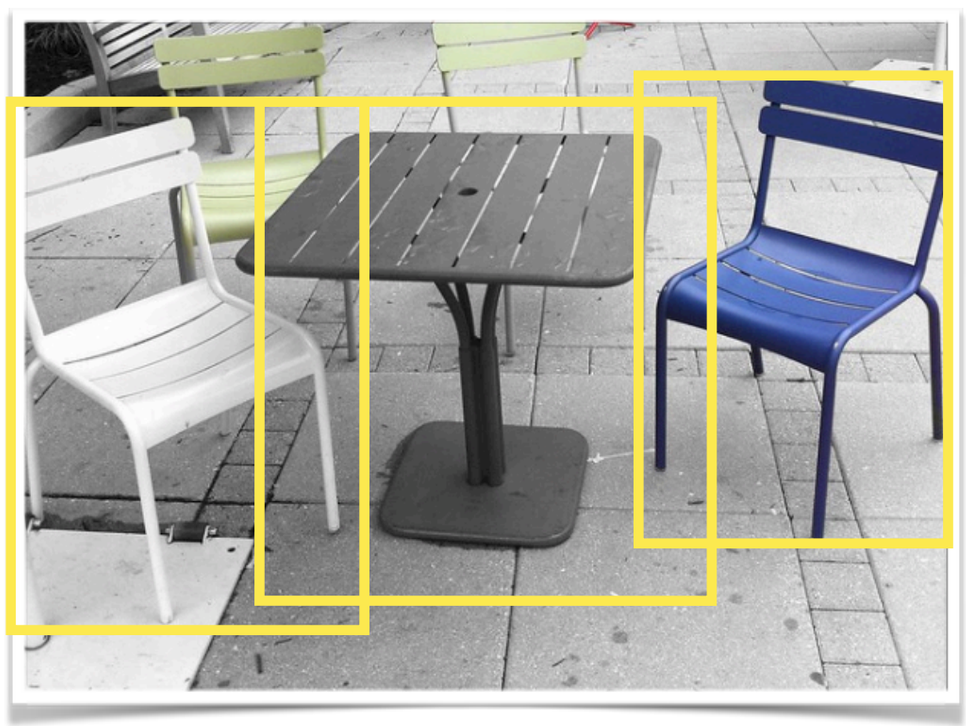
- Solution: region "proposals"

    - Fast low level object classifiers with high recall



RCNN - Issues

- What do we do with overlapping regions?
  - Solution: Non-maxima suppression
    - Filter our all detections close to a strong detection

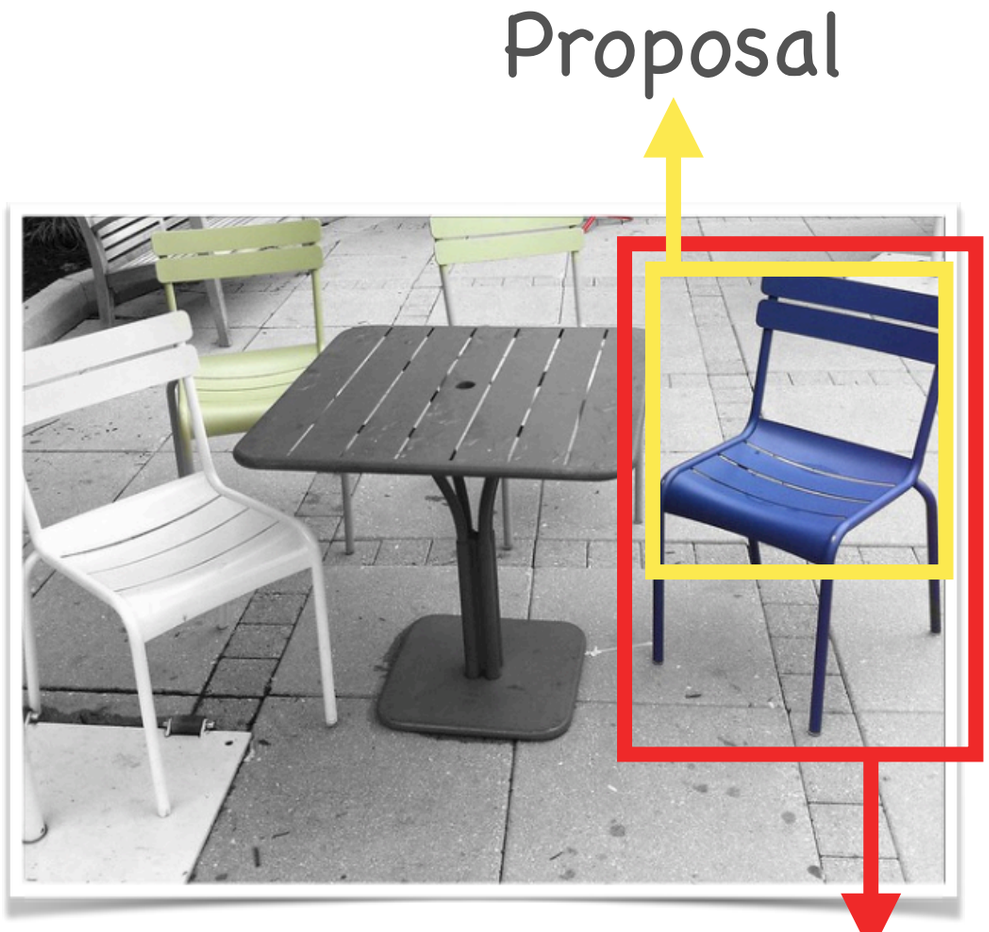


RCNN - Issues

- Proposals might not localize accurately enough

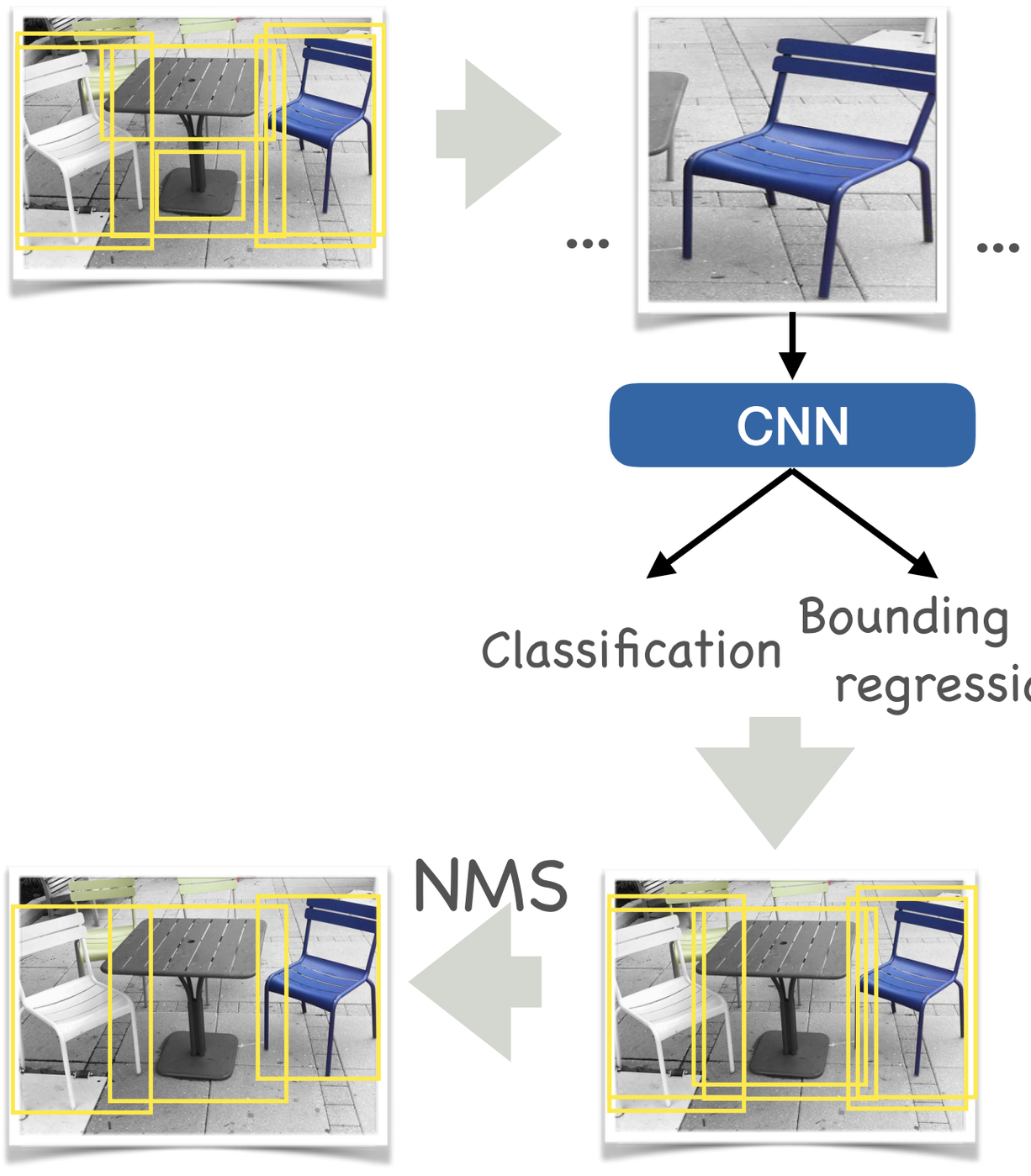
- Solution: Bounding box regression

  - Regress to precise object location



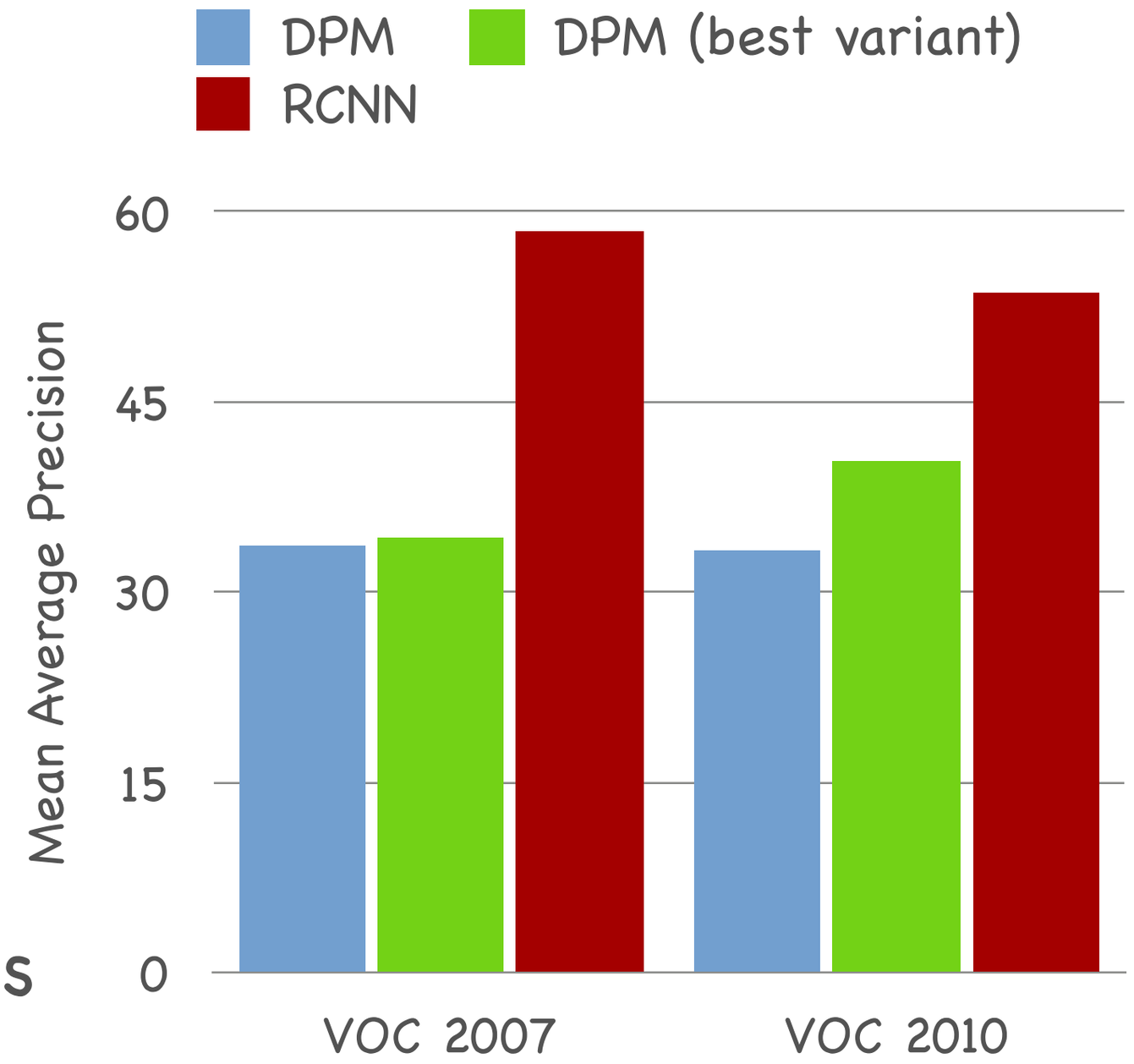
RCNN - Full algorithm

- Find region proposals
- Crop and resize regions
- Classify all regions
  - Object or not?
  - Regress exact location
- Extract detections using Non-maxima suppression



Summary

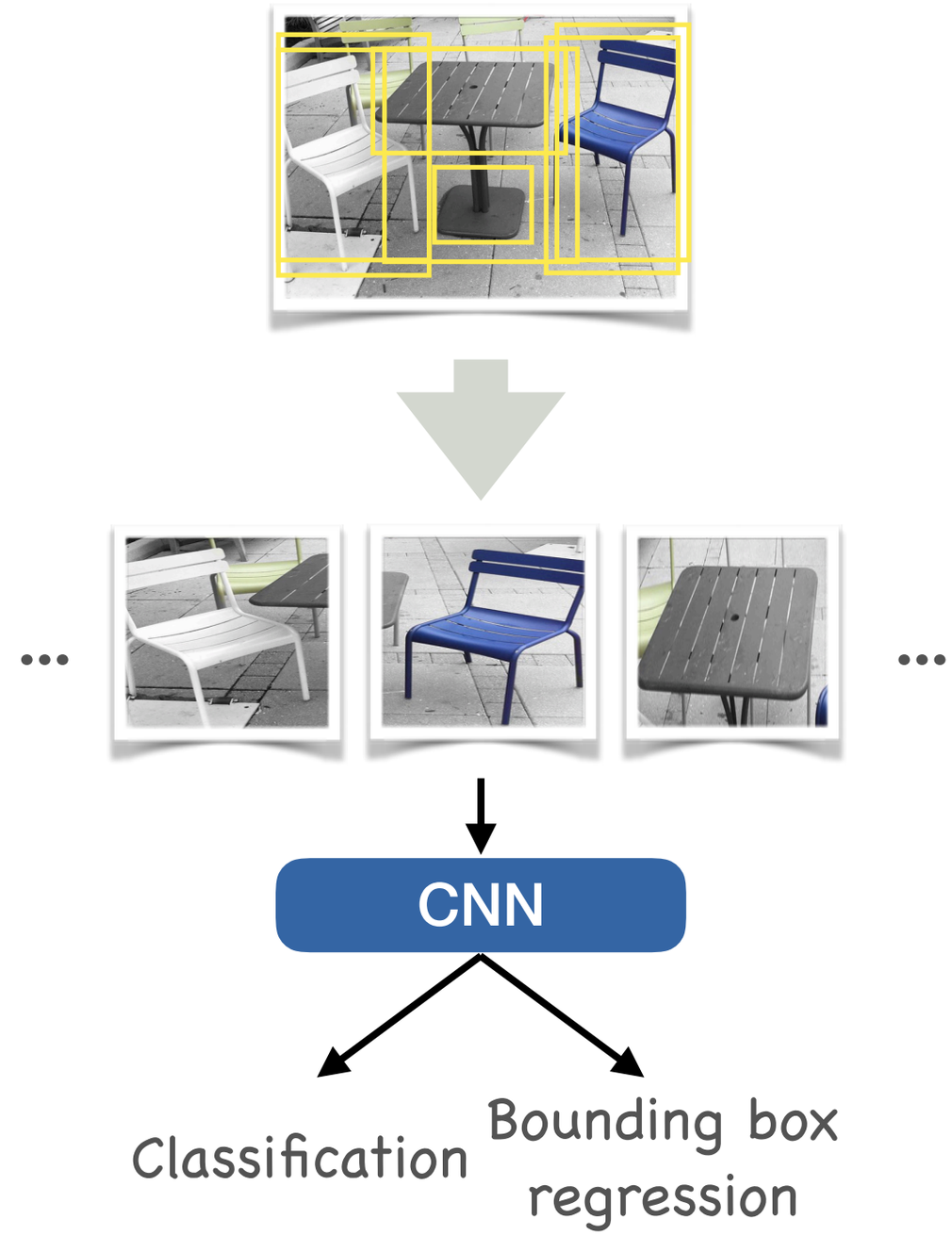
- State of the art performance 2014
- Slow
  - 1 min / image
  - Too many forward passes (1k+)
- Relies on region proposals



Case study: Faster RCNN

RCNN

- RCNN issues
  - Slow (1k forward passes)
  - Relies on good object proposals



Many forward passes

- FastRCNN
  - Forward the entire image through all conv layers
  - Crop activations
    - ROIPooling, ROIAlign

Fast R-CNN, Girshick, ICCV 2015

Region proposal network

- Use deep network for region proposals (RPN)
  - Region or not?
  - Try all spatial locations
    - Fully convolutional
  - Try multiple sizes
    - "Anchor boxes"

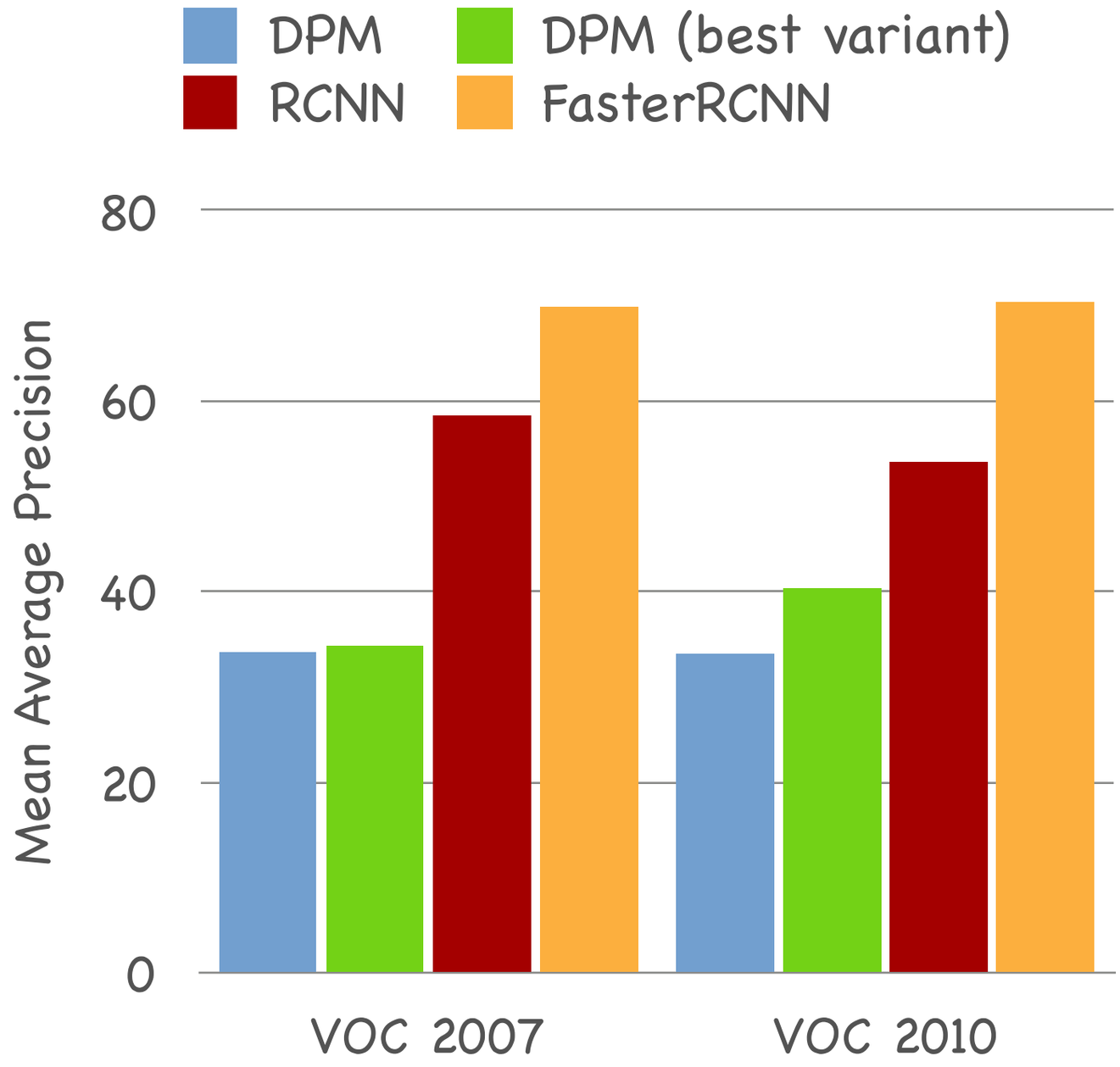
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren et al., NIPS 2015

FasterRCNN

- Run image through fully convolutional network
- Classify Region
- Crop activation (ROI Pooling)
- Classify activation
- Non-maxima suppression

Summary

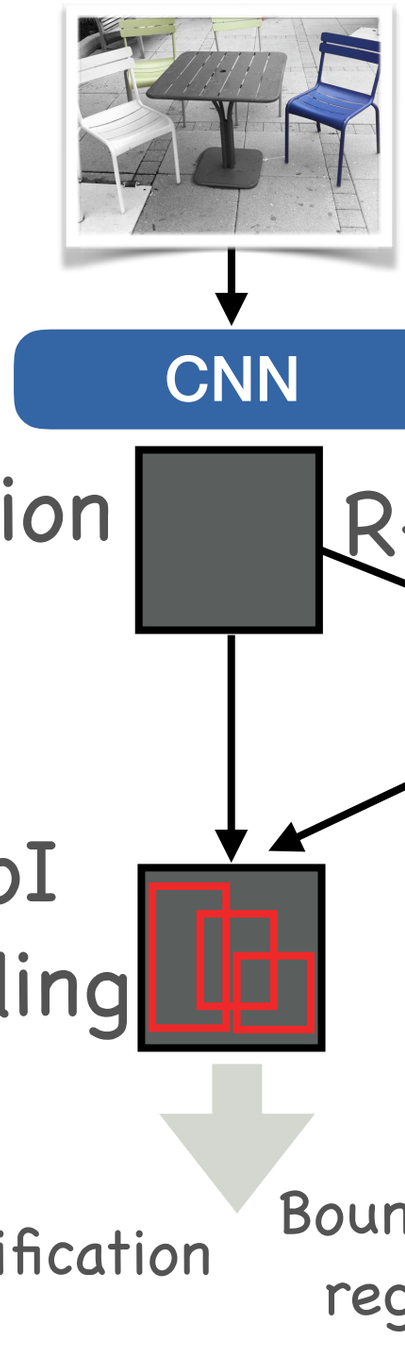
- Fast
  - 200ms / image
- Single network
  - Trained jointly
  - Assigning positive and negative boxes messy



Case study: RetinaNet

Single stage detection

- Object detection without cropping
- Use region proposal network for classification



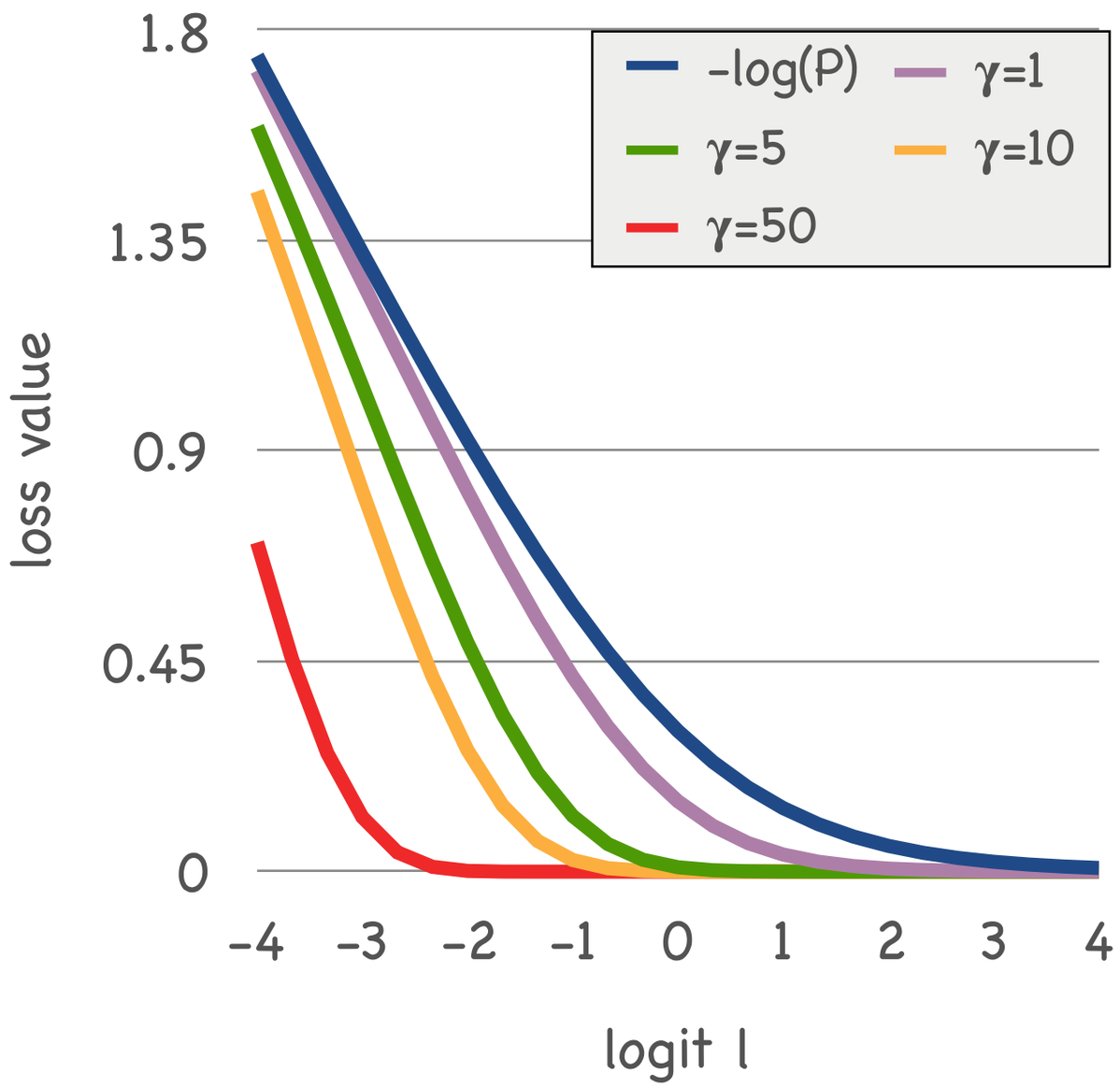
Focal Loss for Dense Object Detection, Lin et al., ICCV 2017
You Only Look Once: Unified, Real-Time Object Detection, Redmon et al., CVPR 2016

Single stage detection - issues

- End-to-end training
  - More negative examples than positives
- Solution: Weighted loss

RetinaNet

- Focal loss:
  - $-(1-p(y))^{\gamma}\log p(y)$
- With $p(y=1)=\dfrac{1}{1+\exp(-l)}$
- Allows for different weight on positives and negatives



Summary

- Focal loss
  - Used beyond object detection
    - Imbalanced training labels
- RetinaNet
  - Single stage
  - Faster than FasterRCNN

Segmentation

Image segmentation

- Group pixels
  - Same object
  - Same "stuff"
  - Same part

Semantic segmentation

- Group pixels according to their semantic class
- Does not distinguish identities
- Segments objects and stuff
- Easiest to train and evaluate

Instance segmentation

- Segment only objects
- Segment and label each instance
- Extension of object detection

Panoptic segmentation

- Segment instances and stuff
- Combines semantic and instance segmentation in a single task
- Evaluation tricky

Datasets: MS COCO

- 80 classes
- 200k images
- 1.5M objects
- Both instance and stuff labels

Microsoft coco: Common objects in context, Lin et al., ECCV 2014

Driving datasets

- Semantic segmention
  - Cityscapes
  - Mapillary

Image source : Cityscapes dataset

The Cityscapes Dataset for Semantic Urban Scene Understanding, Cordts et al., CVPR 2016
The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes, Neuhold et al., ICCV 2017

Datasets: Simulators

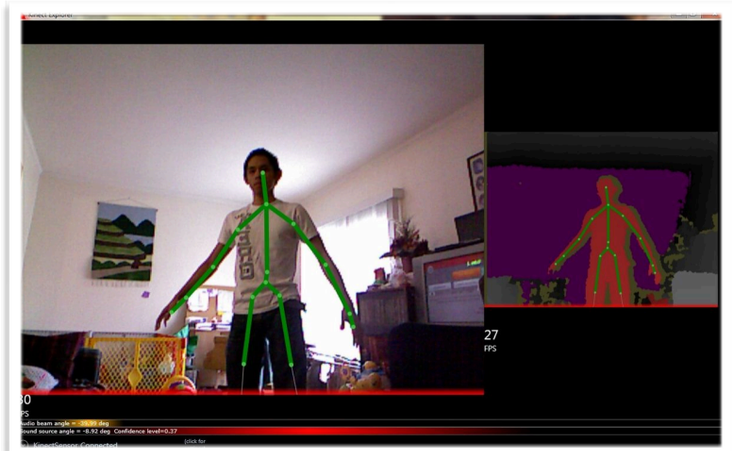
- Segmentation at no additional cost from rendering engine

- Examples
  - GTA V
  - Carla
  - Habitat

Playing for data: Ground truth from computer games, Richter et al., ECCV 2016
Free supervision from video games, Krähenbühl, CVPR 2018

Applications

- Assisted driving



- MS Kinect body tracking

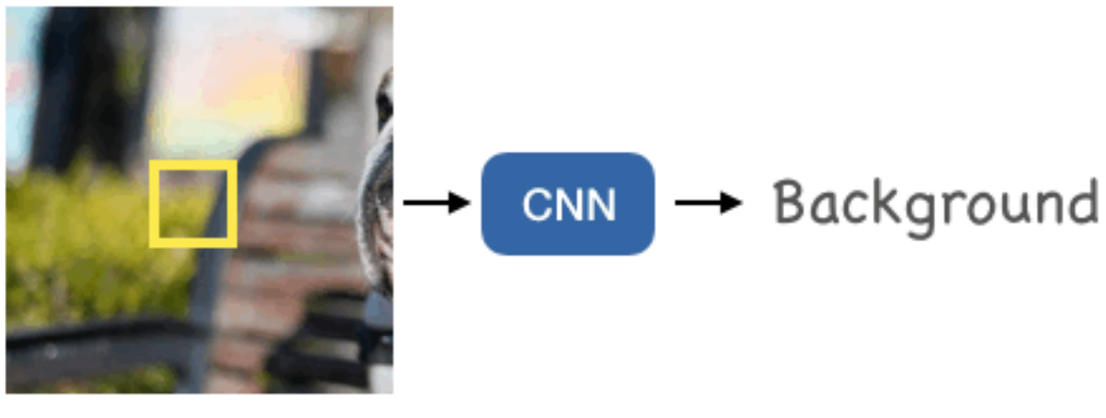
- Computer graphics (matting)

Image source: Deep Image Matting, Xu et al., https://arxiv.org/pdf/1703.03872.pdf

Case study: FCN

Fully convolutional networks

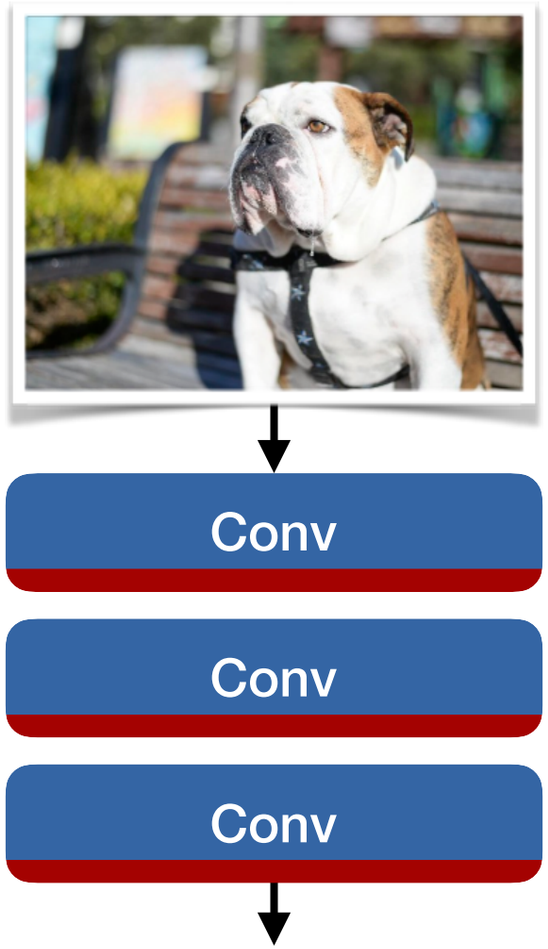
- Semantic segmentation by classifying each pixel
  - Fully convolutionally
  - Cross entropy loss at every pixel



Fully Convolutional Networks for Semantic Segmentation, Shelhamer et al., CVPR 2015

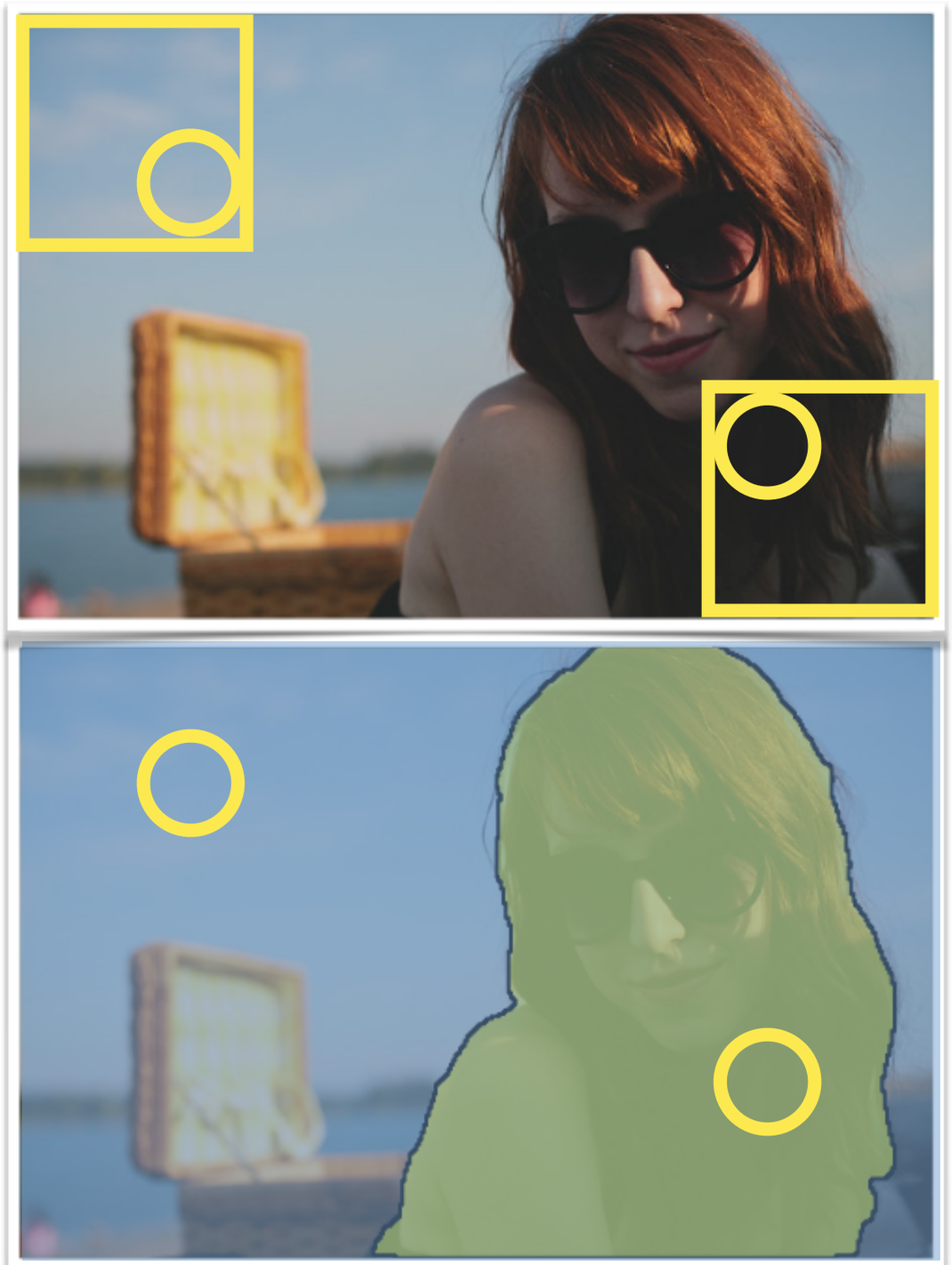
Output resolution

- Striding in network reduces output resolution
- Output segmentation 16-32x smaller
- Solution: Upsampling
  - Naive: Linear
  - Up-convolution



Practical considerations

- Output should be at center of receptive field
  - Do the math, or NaN test to verify



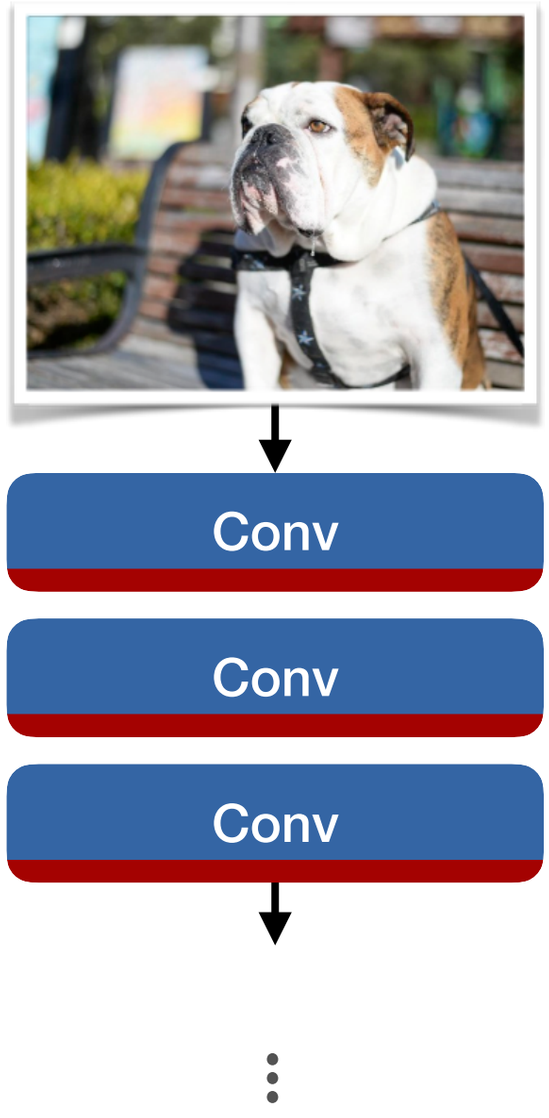
How to build an FCN

- Take a classification network
  - Convert linear layers to convolutions
  - Add up-sampling / up-conv
  - Add optional skip connections
- Train fully convolutionally
  - Image-net pre-training helps

Case study: Dilated convolutional networks

Dilated convolutional networks

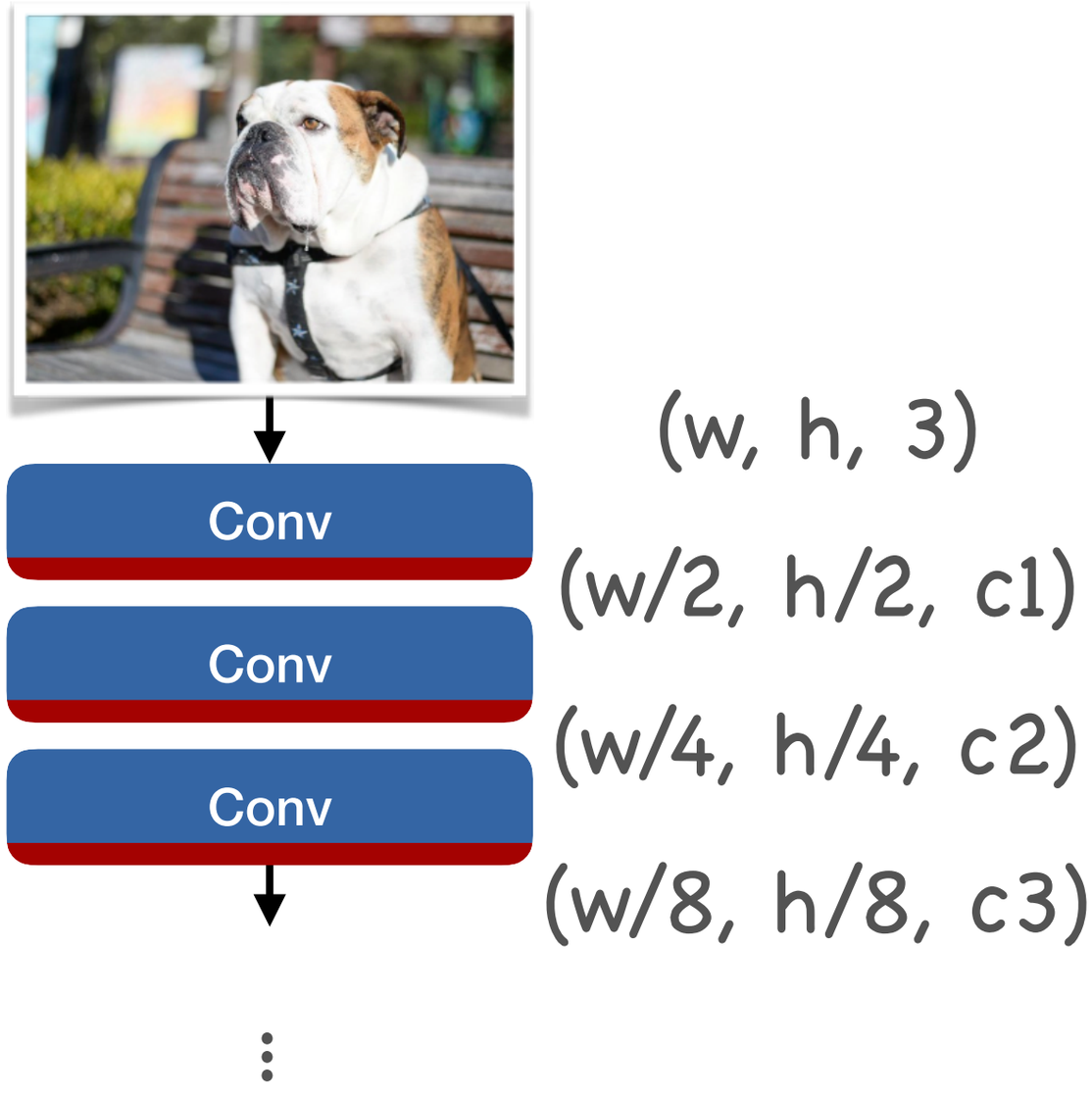
- FCNs have a fairly low resolution
- Solution: Replace strided convolutions with dilated convolutions



Semantic image segmentation with deep convolutional nets and fully connected CRFs, Chen etal. ICLR 2015

Extension: Context module

- Reason about class-wise interactions
  - Dilated layer on top of class output
- Alternatives: Graphical models



Multi-Scale Context Aggregation by Dilated Convolutions, Yu and Koltun, ICLR 2016

Extensions: Deformable convolutions

- Allow inputs to convolutions to be sampled irregularly

Deformable Convolutional Networks, Dai et al., ICCV 2017

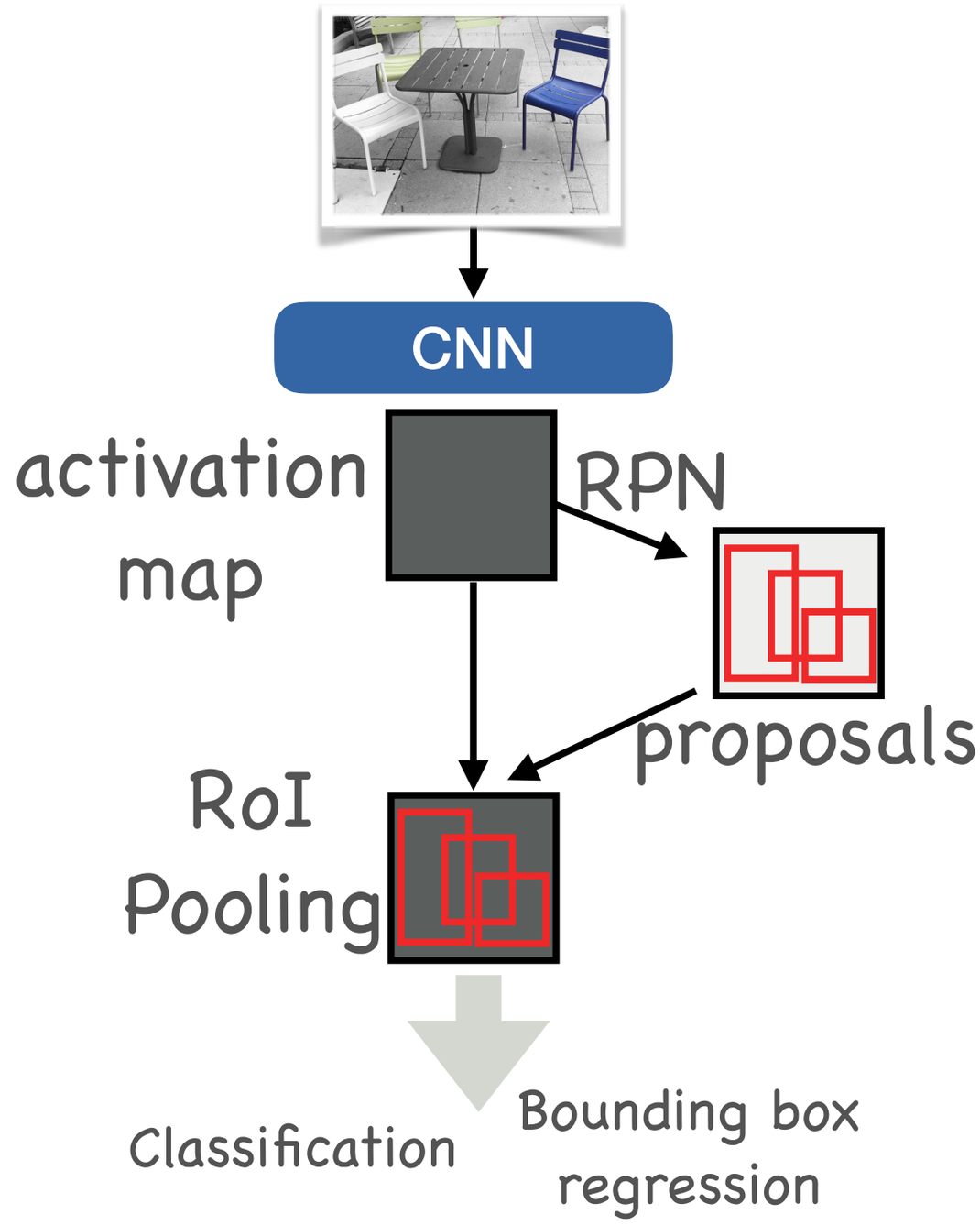
Extensions: Deformable convolutions

Image source: Deformable Convolutional Networks, Dai et al., https://arxiv.org/pdf/1703.06211.pdf

Case study: Mask RCNN

MaskRCNN

- Instance segmentation
- Segmentation by detection



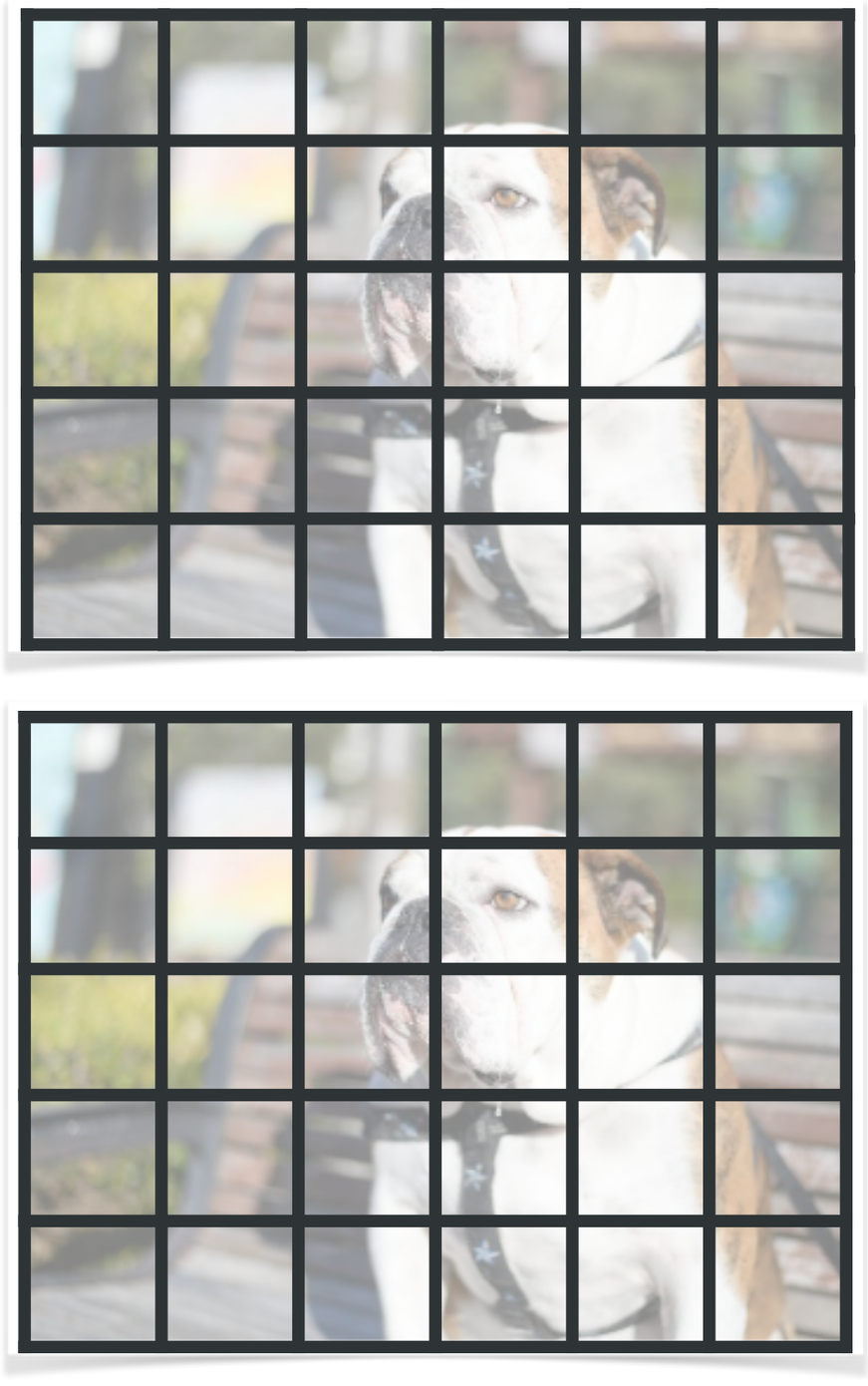
Mask R-CNN, He et al., ICCV 2017

MaskRCNN - trick 1

- ROIPooling blurs activations

- Solution: ROIAlign

    - ROIPooling with linear interpolation

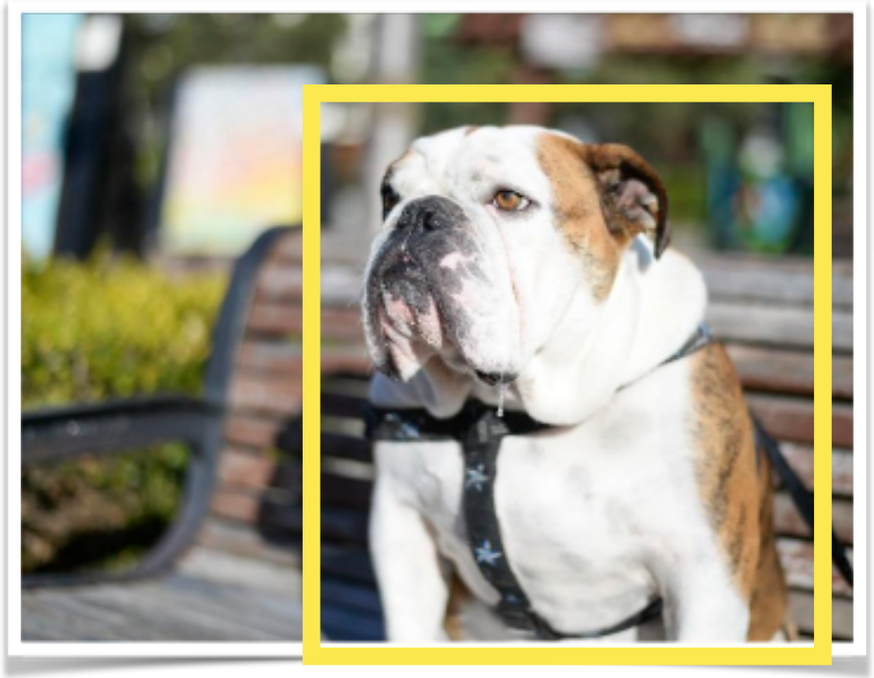


MaskRCNN - trick 2

- Masks do not need to be class specific

MaskRCNN - trick 3

- Masks are best predicted using an FCN



Summary

- MaskRCNN is basic building block for many instance level tasks
- Pose estimation
- 3D reconstruction

Open Problem: Object representations

How should we represent objects in our scenes?

- A box?
- A mask / group of pixels?
- Keypoints?
- A 3D model?

A box?

- Advantages:
  - Low dimensional (4 values)
  - Easy to annotate
- Disadvantages:
  - Distracted by occlusions
  - Too large
  - Not enough information

A mask / group of pixels?

- Advantages:
  - More detailed
  - Handles occlusions
- Disadvantages:
  - Harder to annotate
  - Not much more information than box
  - Not enough information

Mask R-CNN, He et al., ICCV 2017
Bottom-up Object Detection by Grouping Extreme and Center Points, Zhou et al., CVPR 2019

Keypoints?

- Advantages:
  - Low dimensional
  - More information
- Disadvantages:
  - Class specific
  - Ill-defined for many classes
  - Harder to annotate

A 3D model?

- Advantages:
  - Very detailed
- Disadvantages:
  - Too large
  - Very hard to annotate
    - Infeasible for non-human
  - Not much more information

How do we represent stuff?

- Ill-defined

Temporal models

Temporal tasks

- Action recognition/ detection
- Action prediction
- Tracking
- Monocular 3D estimation

Temporal models

- How do we feed videos into CNNs?

Idea 1 - Individual frames

- Treat videos as unordered collection of images
- Ignores time
- Very competitive baseline

Idea 2 - Individual frames + global model

- First parse frames, then model time (e.g. with sequence model)
- Overfits easily
- Hard to train
- Loses motion information

Idea 3 - extend convolutions through time

- Build a 3D convolutional network

- 2D spatial
- 1D temporal

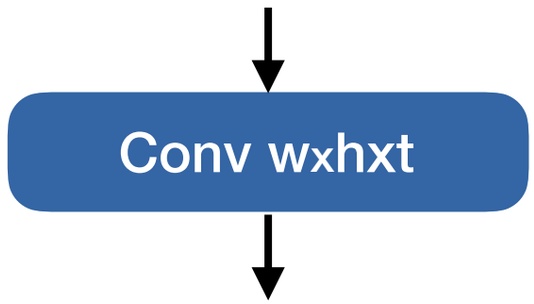
3D convolutions

3D convolutions

- Convolution across space and time
  - Input video is a 4D tensor
    - time
    - width, height
    - color channels
- 3D kernel
  - time, width, height

Formal definition

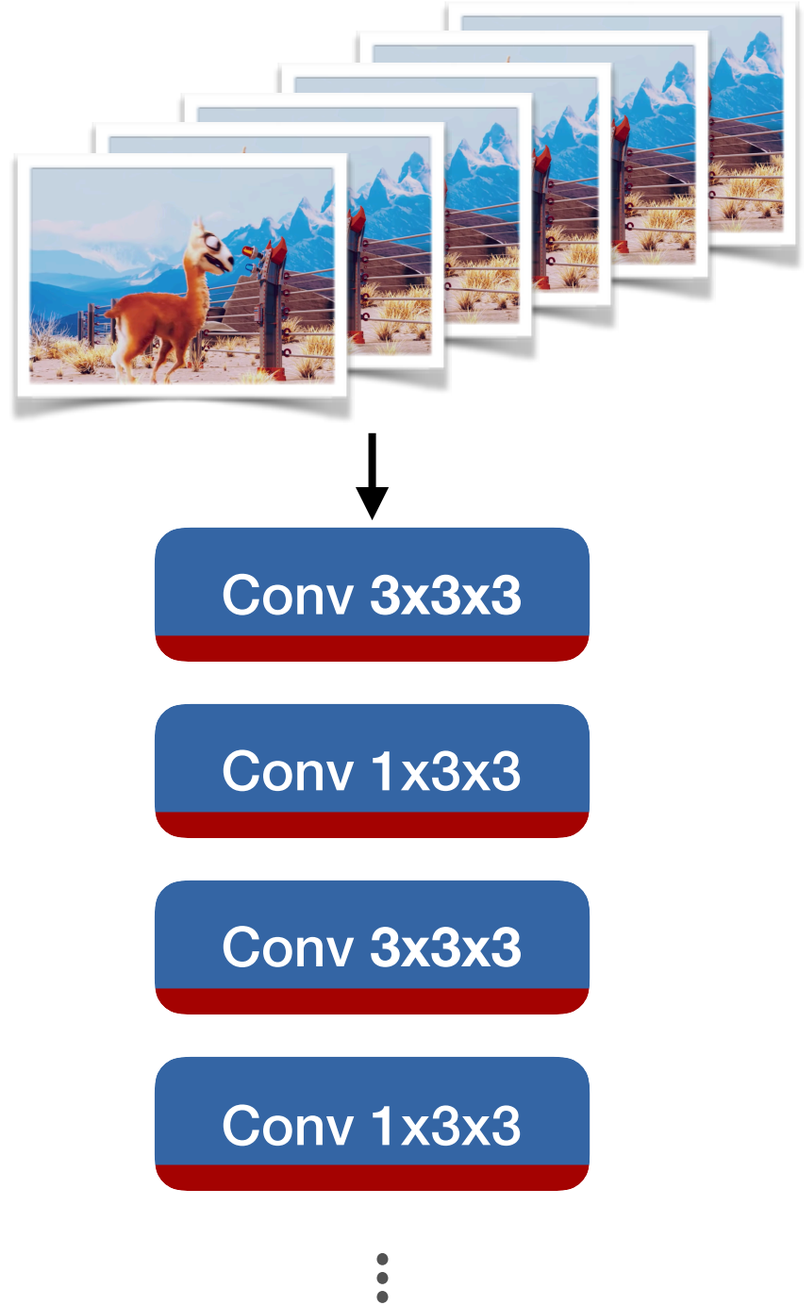
- Input: $\mathbf{X} \in \mathbb{R}^{T \times H \times W \times C_1}$
- Kernel: $\mathbf{w} \in \mathbb{R}^{t \times h \times w \times C_1 \times C_2}$
- Bias: $\mathbf{b} \in \mathbb{R}^{C_2}$
- Output: $\mathbf{Z} \in \mathbb{R}^{\left(\frac{T-t+2p_t}{s_t}+1\right) \times \left(\frac{H-h+2p_h}{s_h}+1\right) \times \left(\frac{W-w+2p_w}{s_w}+1\right) \times C_2}$



$$\mathbf{Z}_{d,a,b,c} = \mathbf{b}_c + \sum_{l=0}^{t} \sum_{i=0}^{h} \sum_{j=0}^{w} \sum_{k=0}^{C_1} \mathbf{X}_{d+l,a+i,b+j,c+k} \mathbf{w}_{l,i,j,k}$$

3D CNNs

- Take a image CNN
  - Replace some (not all) layers with 3D conv



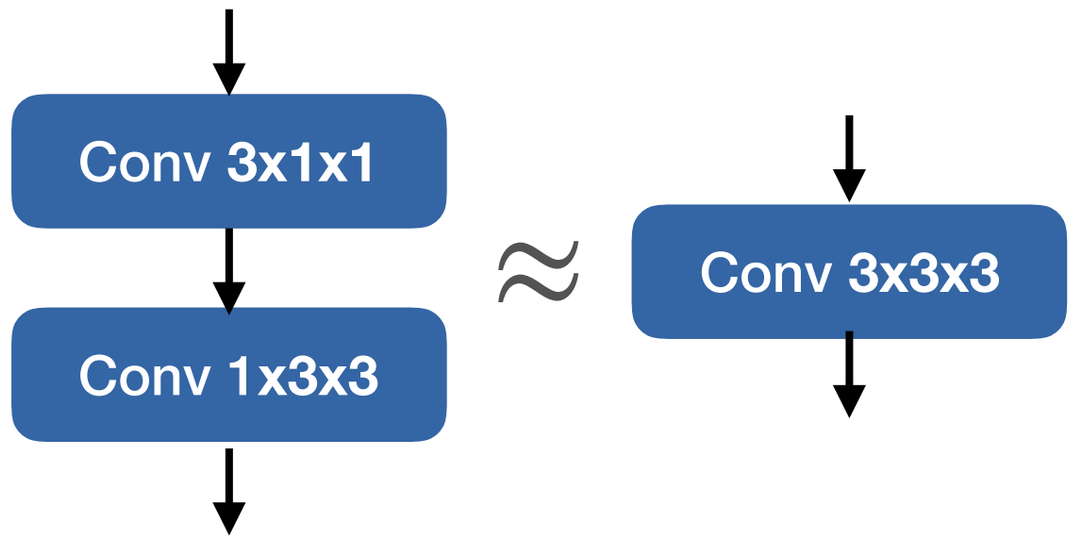
3D CNNs - Issues

- 3D convolutions are too large
- Slow
- Too many parameters

2+1D convolutions

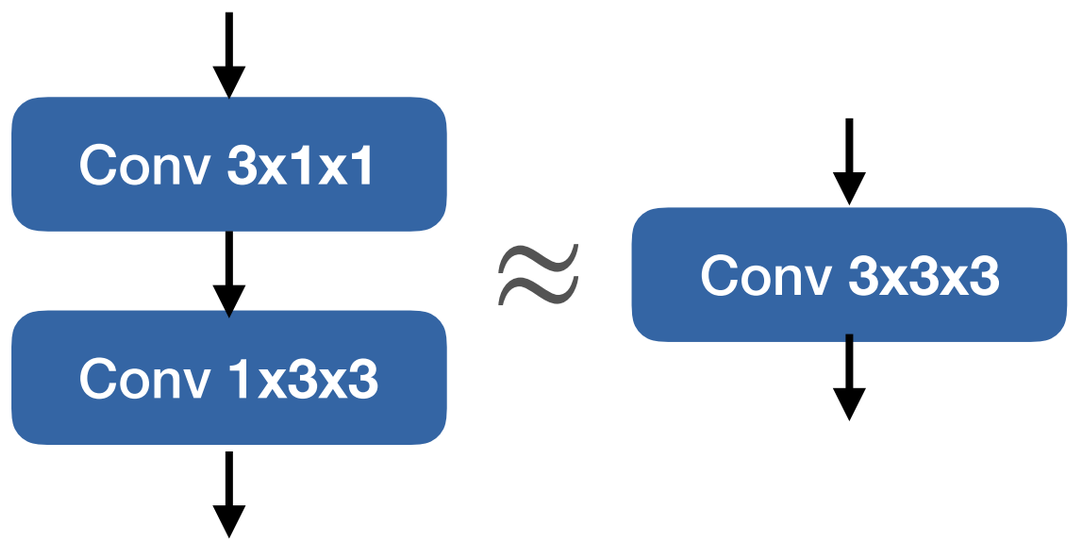
Factorized 3D convolutions

- Separate spatial and temporal convolutions
  - 2D convolution through space
  - 1D convolution through time



2+1D convolutions

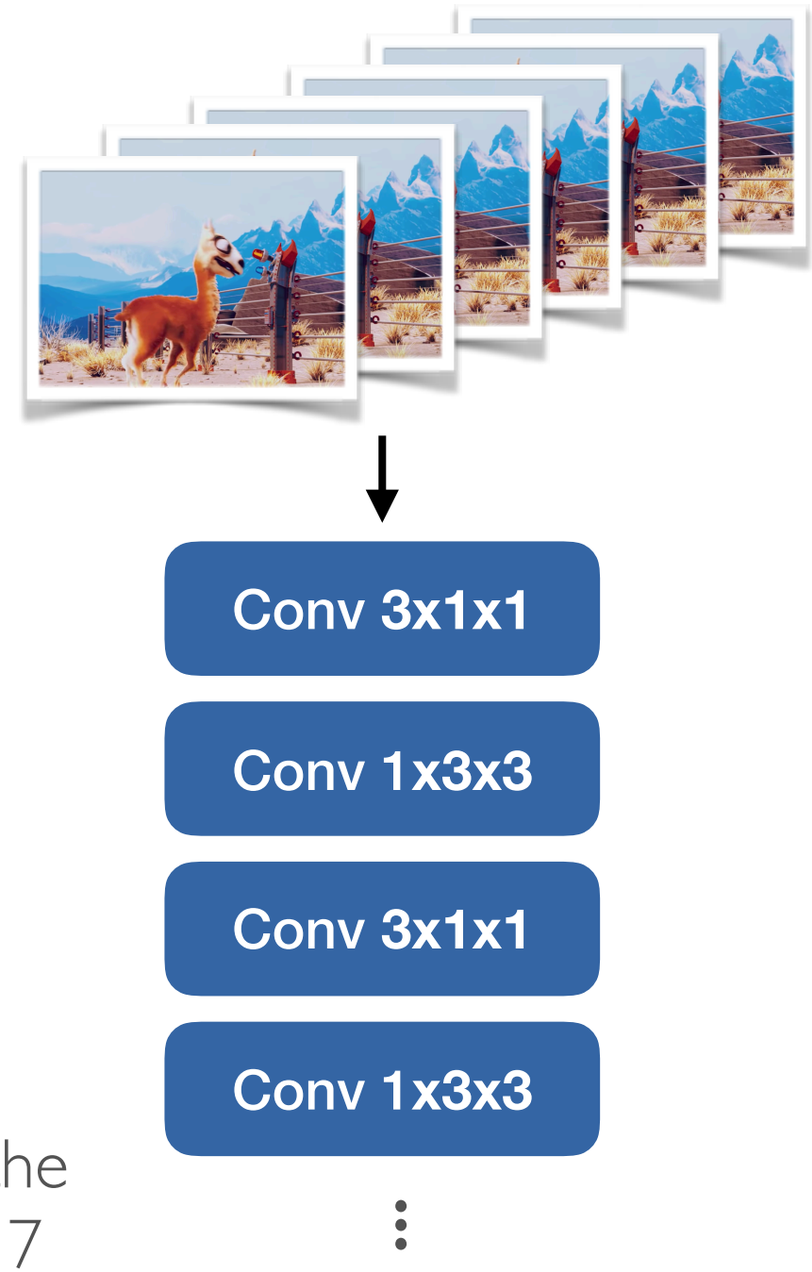
- Fewer parameters
- Less computation
  - $\mathcal{O}((WH + T)C^2)$ vs $\mathcal{O}(WHTC^2)$
  - Easier to train



Case study: I3D

How do we train large 3D CNNs?

- Hard to train from scratch
- Slow to forward and backprop
- No good pre-training task



Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset, Carreira and Zisserman, CVPR 2017

I3D

- Train image model first
  - on ImageNet
  - as 2D CNN
- then inflate to video model
  - Convert some 2d conv to 3d conv
  - Replicate weights in time

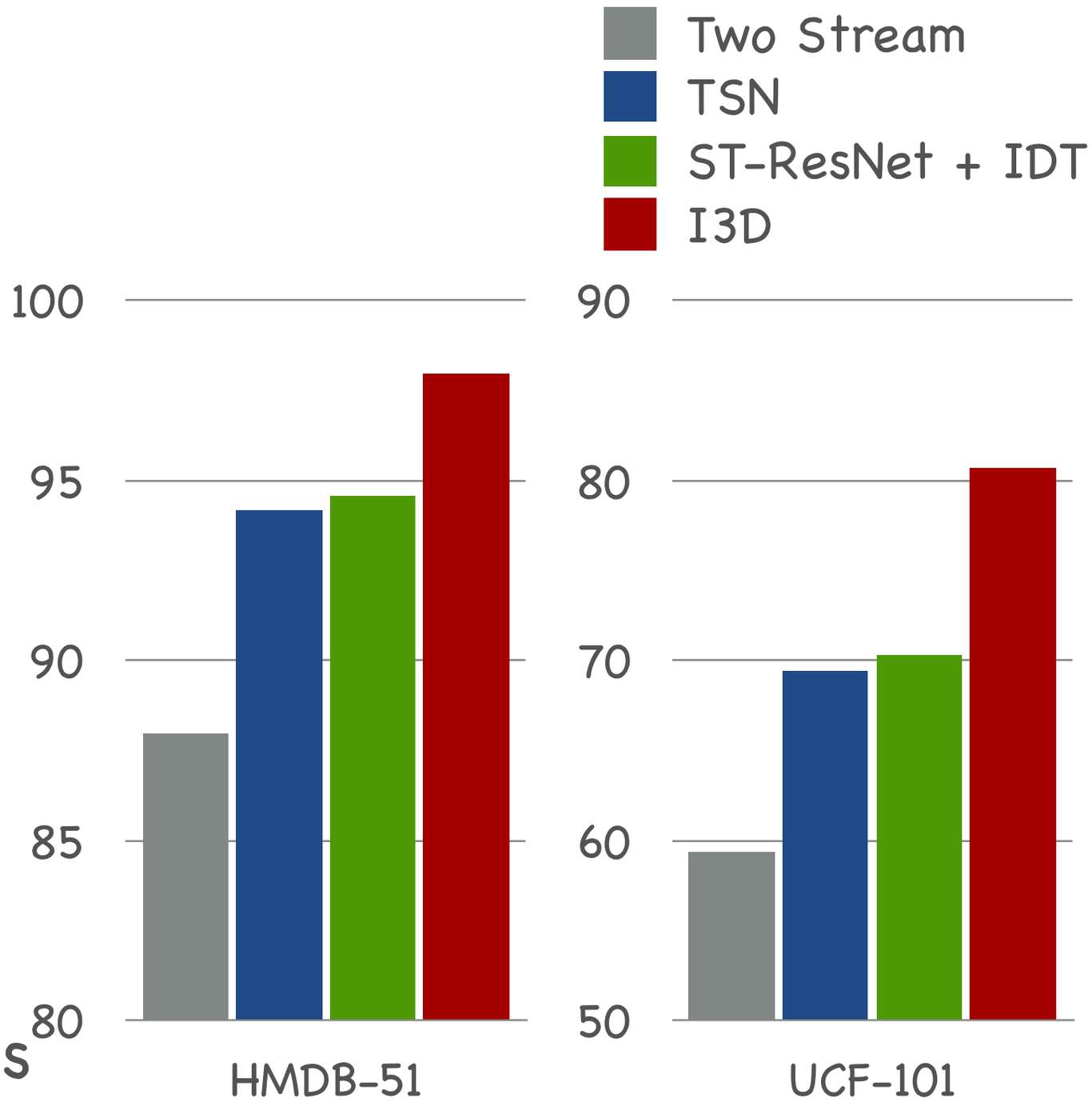



I2+1D

- Train image model first
  - on ImageNet
  - as 2D CNN
- then inflate to video model
  - Add temporal conv
  - Initialize with 1/T

I3D

- Much faster training
  - 2D CNN on ImageNet trains much faster
  - Inflation does not damage network
- Fine-tune on action recognition
  - many days on many GPUs



Open Problem: Effective temporal operations

Effective temporal operations

- 3D convolutions
  - Inefficient
- 2+1D convolutions
  - Do not care about time much
- Non-local operator
  - Orderless pooling

Non-local Neural Networks, Wang et al., CVPR 2018

Current approaches

- Either, completely ignore time
  - Orderless pooling
- Or apply a brute force hammer
  - 3D convolutions

How should we model time?

- Convolutions
  - Extract local motion
  - No long term tracking
- Motion adaptive convolutions
  - Hard to train or make work

Trajectory Convolution for Action Recognition, Zhao et al., NeurIPS 2018

Should we model time?

- Or is time simply a proxy for space
  - Through ego-motion
- How do we connect time and space?

Image source: EPIC-Kitchens dataset, Damen et al., https://arxiv.org/abs/1804.02748

Open problem: What should we infer or label?

Computer vision tasks

- Proxy for down stream applications
- Few are useful by themselves
- How good are our proxies?

Example: Assistive technologies

- Object detection
  - Goal: How many elderly people do you detect?

- Down stream goal
  - How quickly and accurately do you detect a dangerous situation?
  - How much more efficient do you make the caretaker?

Example: Autonomous driving

- Semantic segmentation
  - Goal: Label every pixel with a class
- Down stream task:
  - Get from A to B without crashing

What do computer vision tasks provide?

- An efficient proxy to design and tune models
- Pre-training
- A compact and generalizable representation

Summary

Computer vision tasks

- Global labeling
- Sparse labeling
- Dense labeling

dog

Global labeling tasks

- Tasks:
  - Image classification
  - Action recognition
  - Image retrieval
  - …
- Tool:
  - Residual networks

Sparse labeling tasks

- Tasks:
  - Object detection
  - Sparse tracking
  - Pose estimation
  - …
- Tool:
  - FasterRCNN / MaskRCNN

Dense labeling tasks

- Tasks:
  - Semantic segmentation
  - Instance segmentation
  - Depth estimation
  - …
- Tool:
  - FCN with bag of tricks

Videos

- Tasks:
  - Action recognition
  - Motion estimation
  - Tracking
  - …
- Tool:
  - I2+1D, or variants