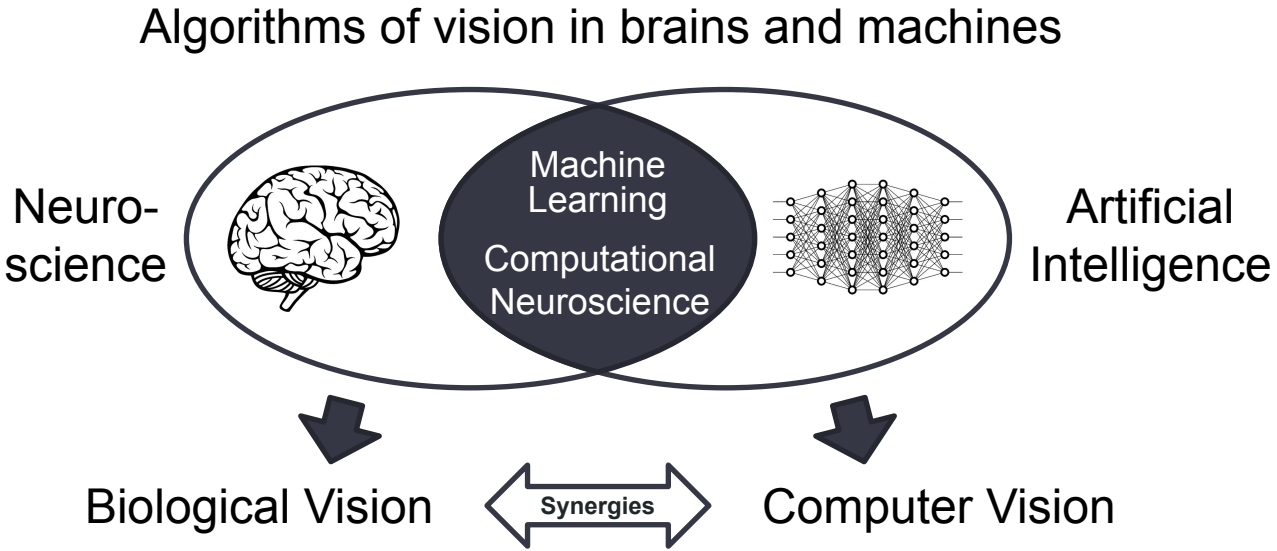
Modern CNNs & transfer learning

Alexander Ecker

Ecker Lab: Neural Data Science

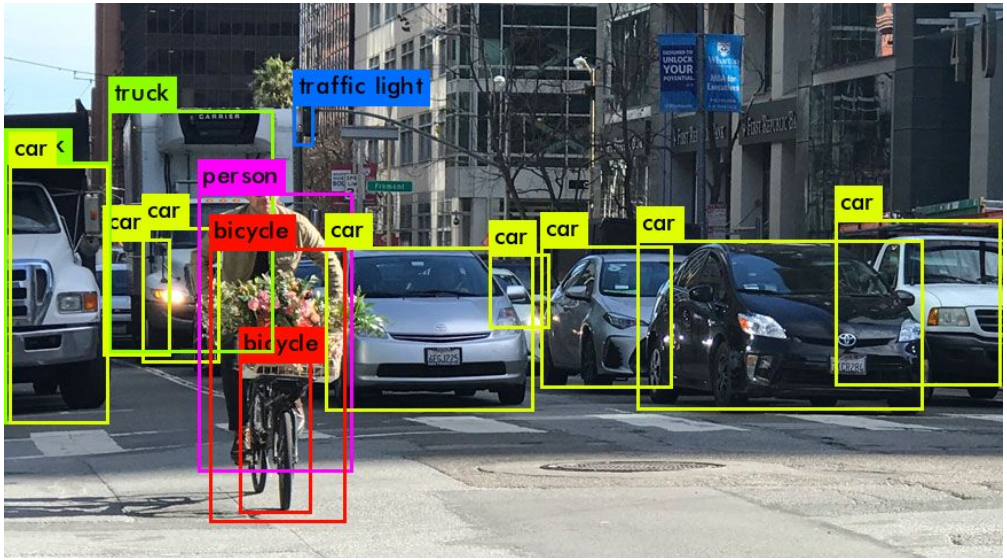
GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN



Yesterday: You learned to build a CNN

Convolution
Pooling/downsampling
Training
Data augmentation
Dropout

Today: CNNs “in the real world”



https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

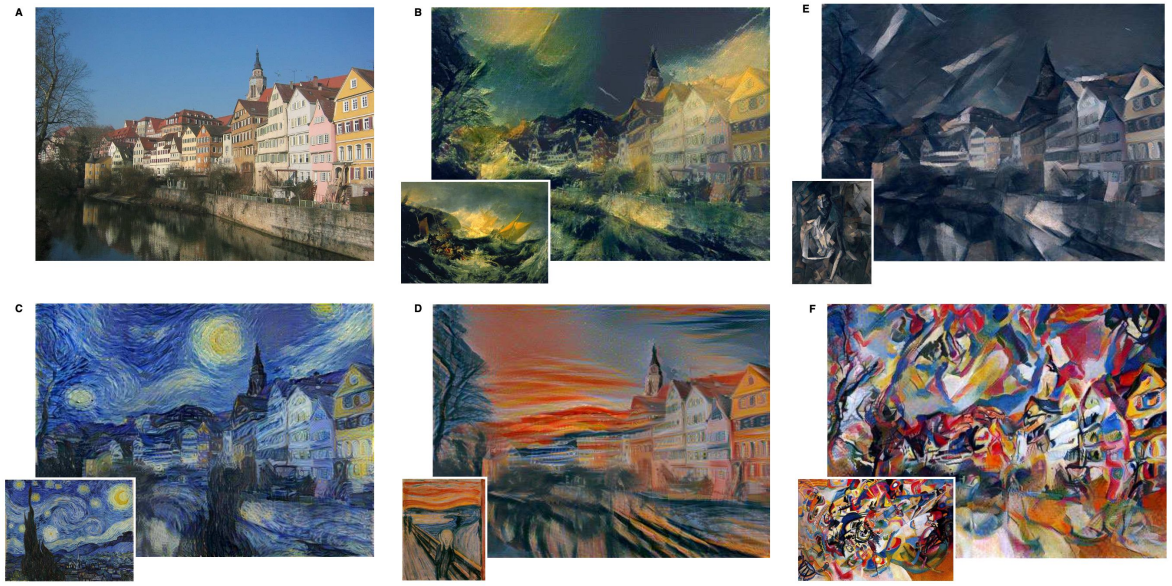
CNNs are everywhere: image segmentation

Source: Cityscapes dataset

CNNs are everywhere: pose estimation

Cao et al., IEEE PAMI 2019: OpenPose

CNNs are everywhere: image style transfer

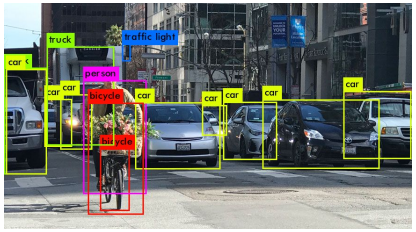


Gatys, Ecker, Bethge, CVPR 2016

Today: Large-scale CNNs & transfer learning

The success of deep learning for image recognition has been driven by two key factors:

1. Large-scale CNNs
2. Transfer learning

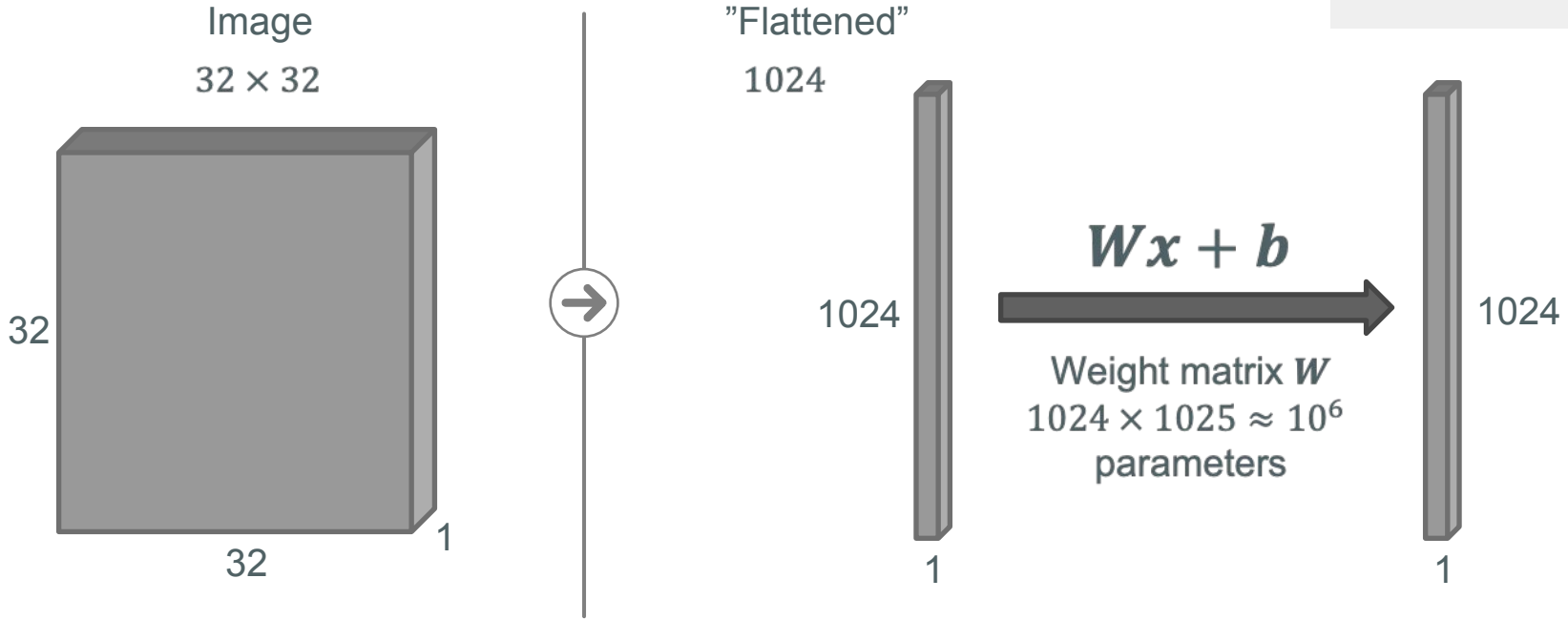


Agenda for today

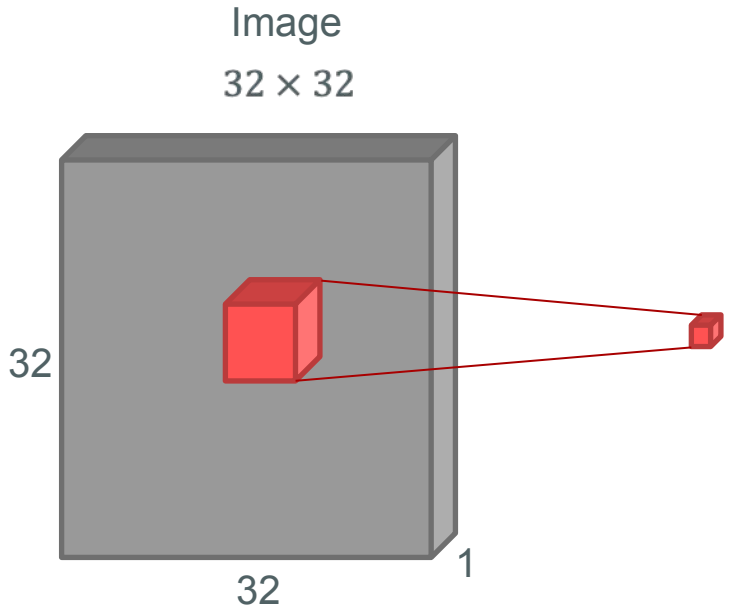
The history of CNNs
ImageNet: the breakthrough moment of deep learning
Going deeper and larger
Transfer learning
Face recognition

Recap: convolution

32 × 32 grayscale image: fully-connected layer



Convolution



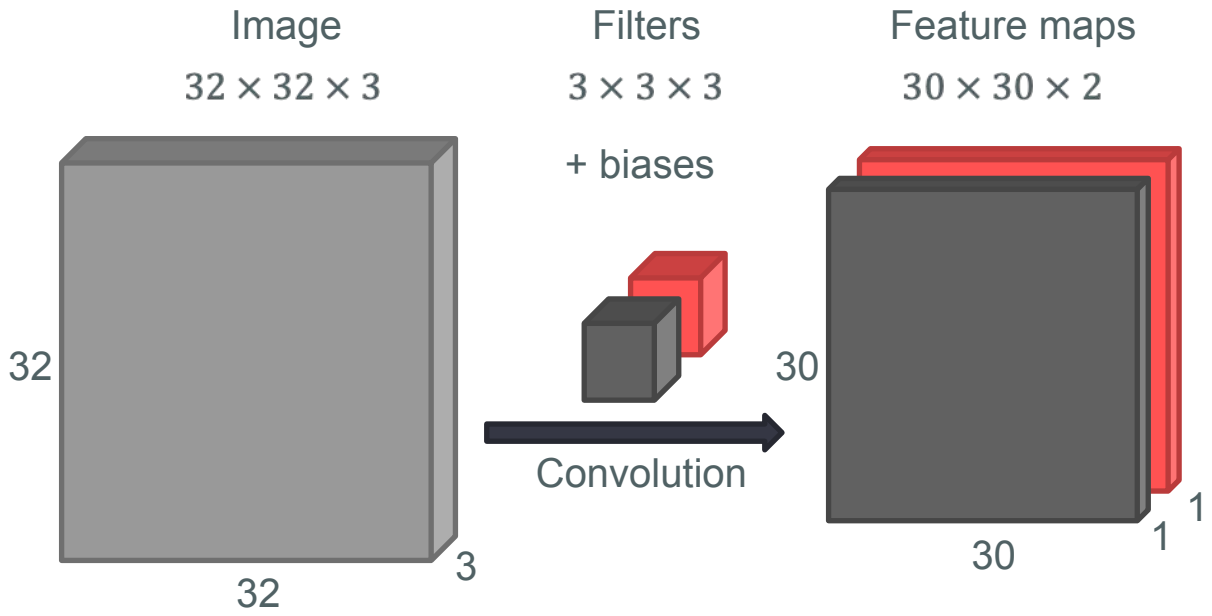
Convolve the filter with the image:
Slide over the image, computing dot products

1 number: $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

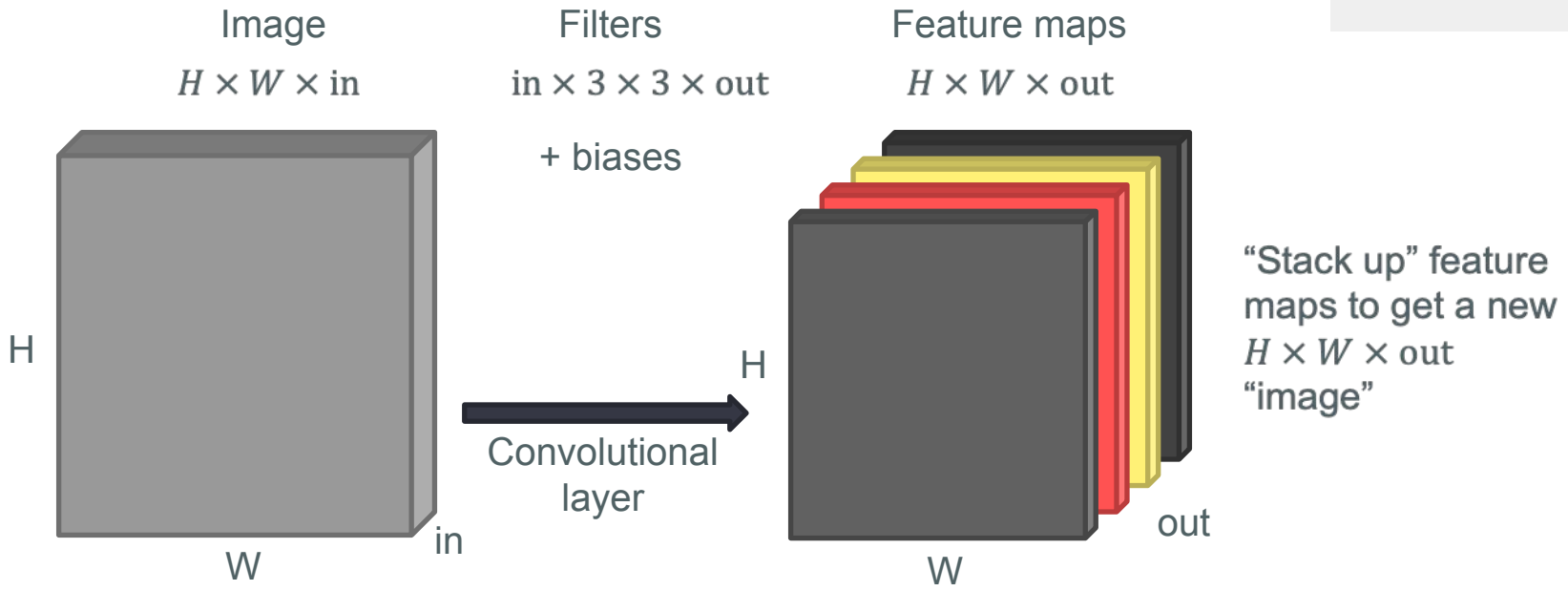
The result is the inner product of

- a local 3×3 chunk $\boldsymbol{x}$ of the image
- a 3×3 filter $\boldsymbol{w}$
- plus a bias b

Convolution: color image



Convolution: in general



Exercise

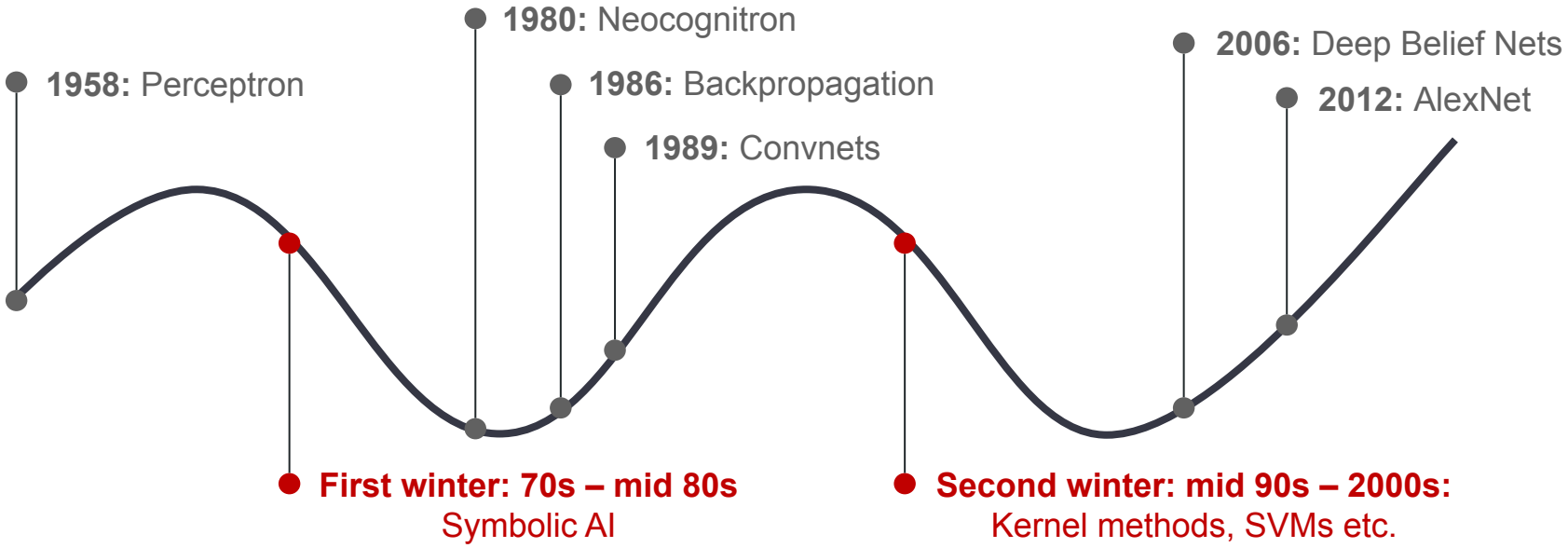
Recap how much convolutions reduce the number of parameters

Count parameters of a fully-connected and a convnet

History of convnets

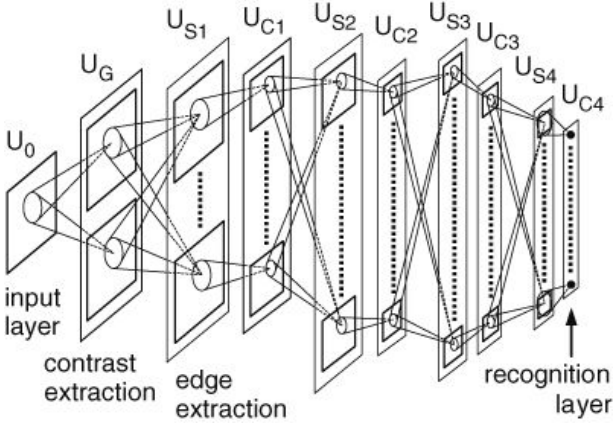
1980ies until now

Neural networks: summers & winters

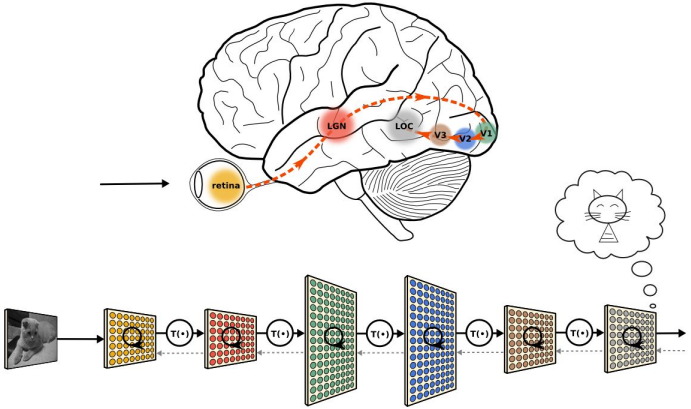


The first convnet: the Neocognitron (1980)

Fukushima 1980: Neocognitron

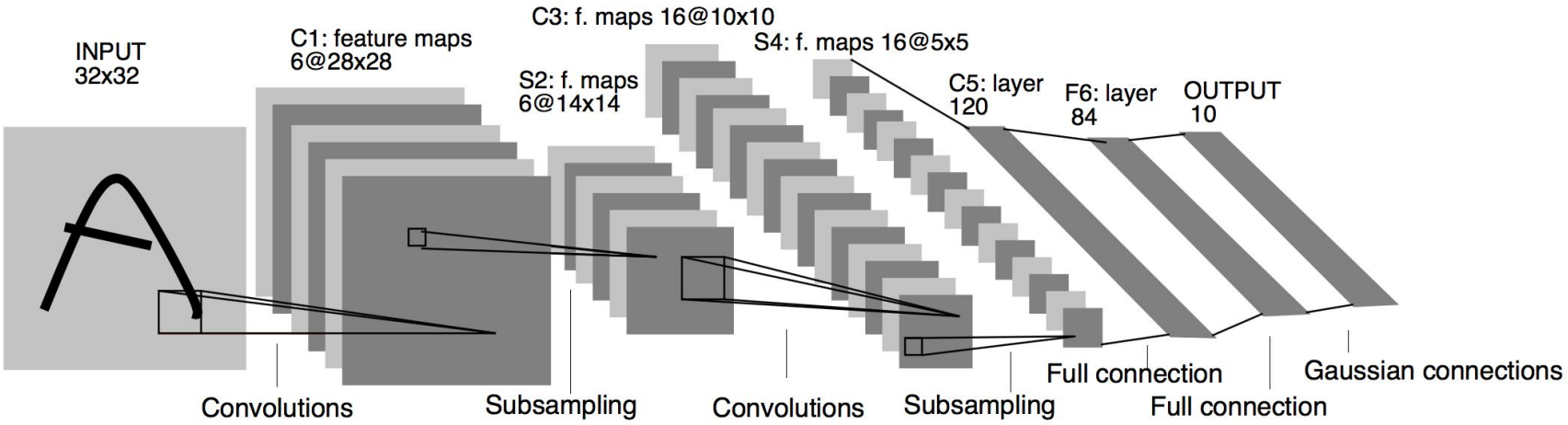


Proposed as a model of hierarchical information processing in the brain



K. Fukushima (1980): "Neocognitron: A self-organizing neural network model […]", *Biological Cybernetics*.

The late 80ies: character recognition



Denker et al. 1989, LeCun et al. 1989, 1998

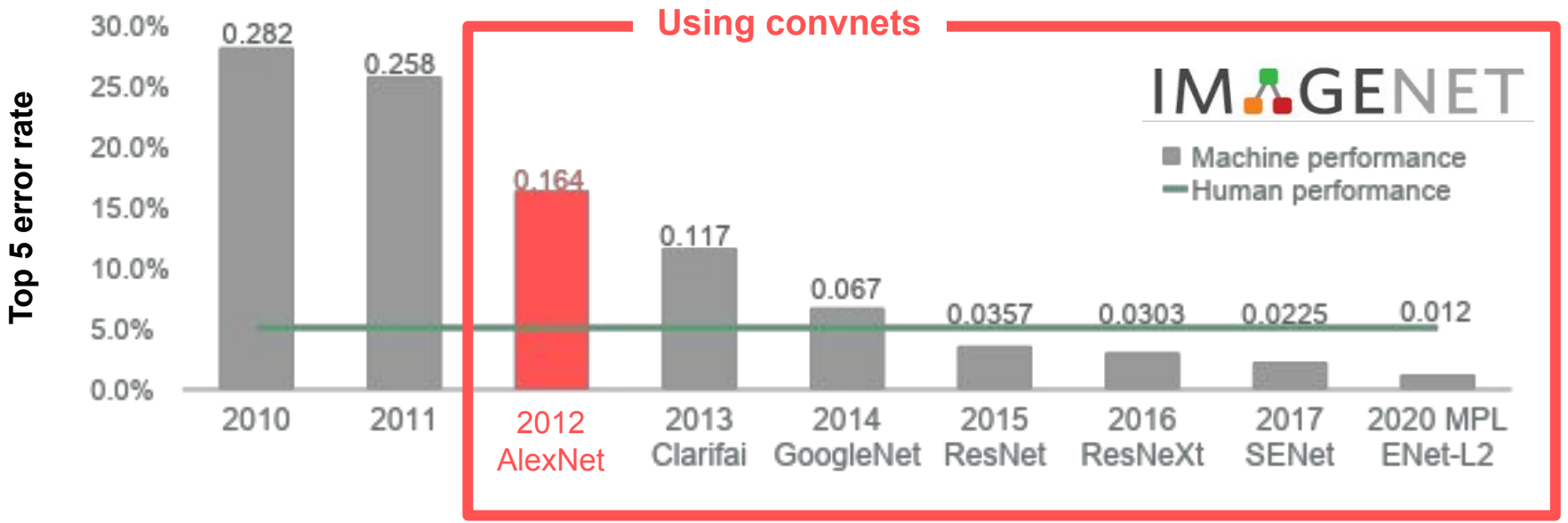
2009: IMAGENET

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): Dataset and benchmark on image classification

- 1 million images with ground truth class labels for training (hand-annotated)
- 1000 object categories

Deng et al., CVPR 2009

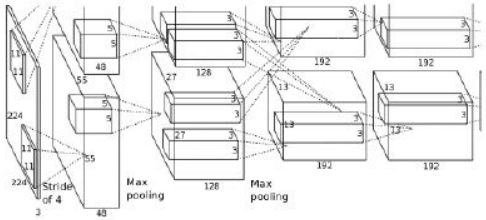
The breakthrough: “AlexNet” 2012



CNNs are old. Why did it work eventually?

Big Data: ImageNet

+



Deep Convolutional Neural Network

+

Backprop on GPU

+ **A number of small tweaks**
Sigmoid □ ReLU, batch normalization, dropout

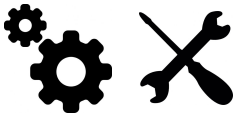


Image credit: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Exercise

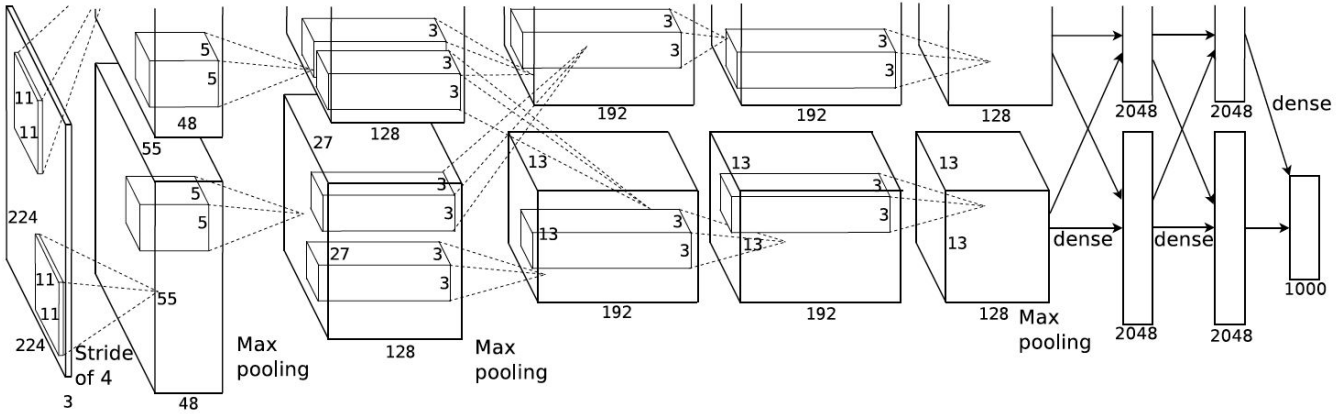
ImageNet performance has improved from 16% to 1.2% top-5 error since the introduction of AlexNet, the first 8-layer CNN trained on the dataset

Discuss with your pod:

- What challenges might researchers and engineers have faced when scaling up CNNs?

AlexNet & VGG

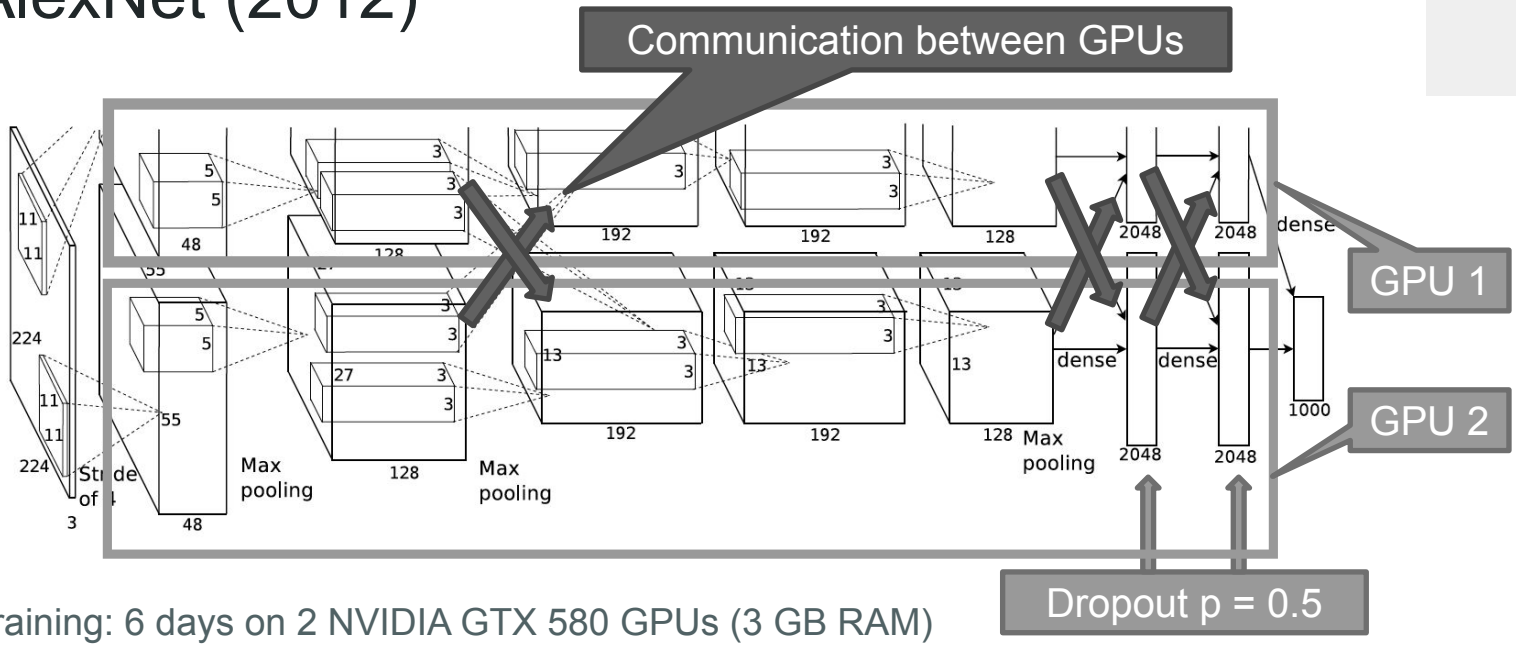
AlexNet (2012)



Alex Krizhevsky, Ilya Sutskeever, Geoffrey Hinton
"ImageNet classification with deep convolutional neural networks." NeurIPS 2012

Krizhevsky, Sutskever, Hinton 2012

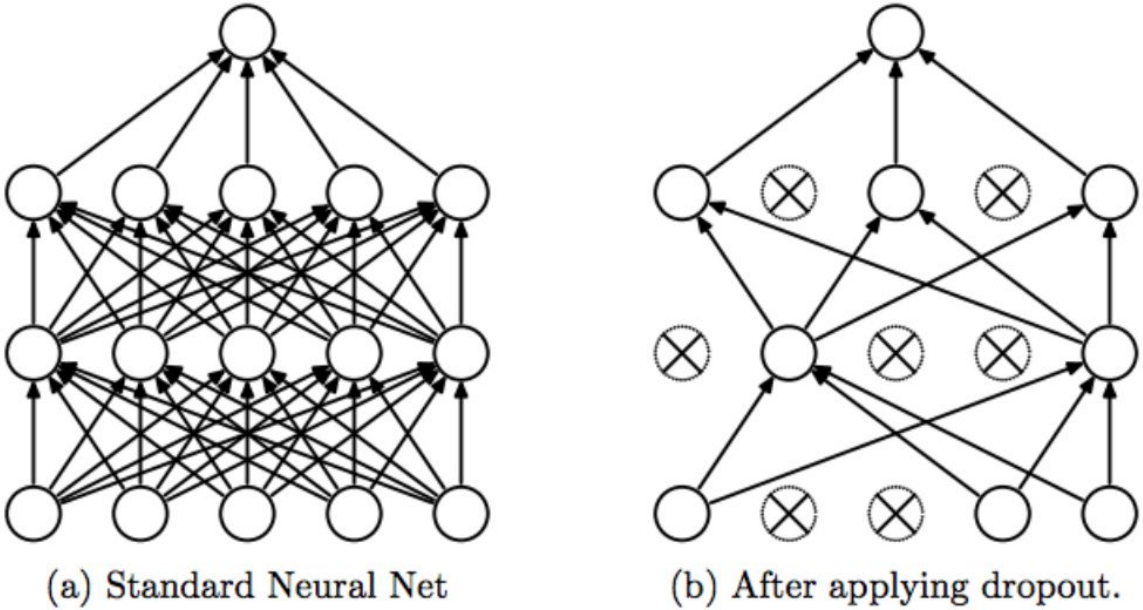
AlexNet (2012)



Training: 6 days on 2 NVIDIA GTX 580 GPUs (3 GB RAM)
8 layers / 60 million parameters

Krizhevsky, Sutskever, Hinton 2012

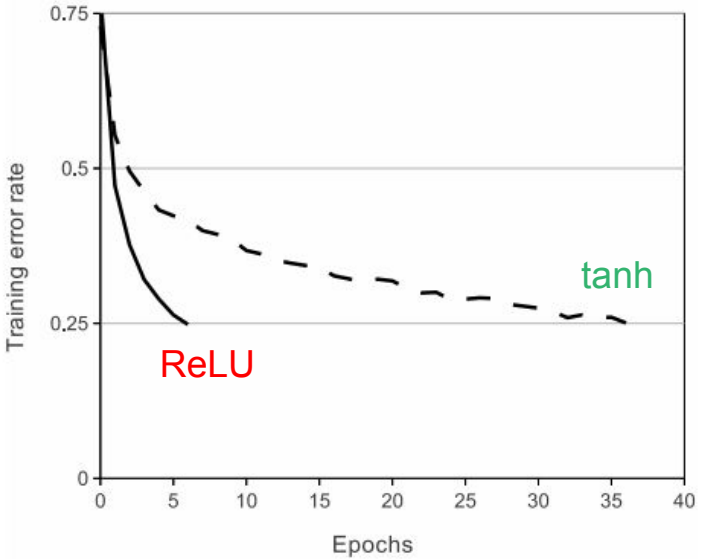
Dropout stochastically deactivates neurons

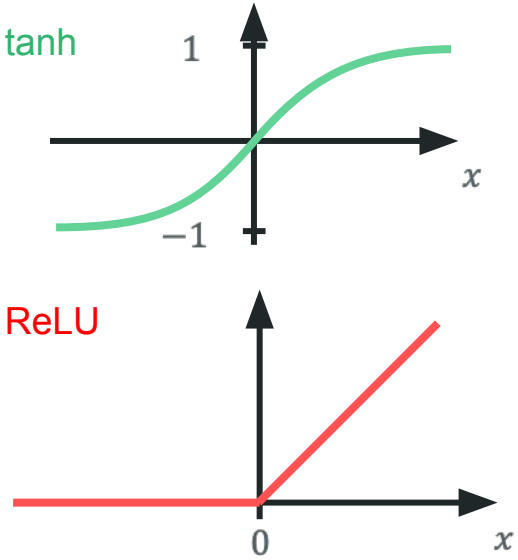


(a) Standard Neural Net

(b) After applying dropout.

ReLu trains faster than tanh





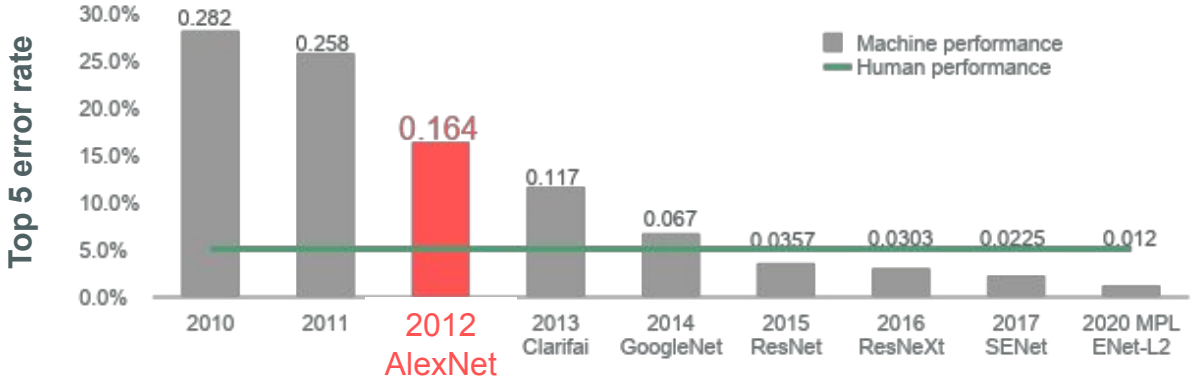
Krizhevsky, Sutskever, Hinton, NeurIPS 2012

AlexNet (2012): Results

Smoked the competition with 16% top-5 error (runner-up had 26%)

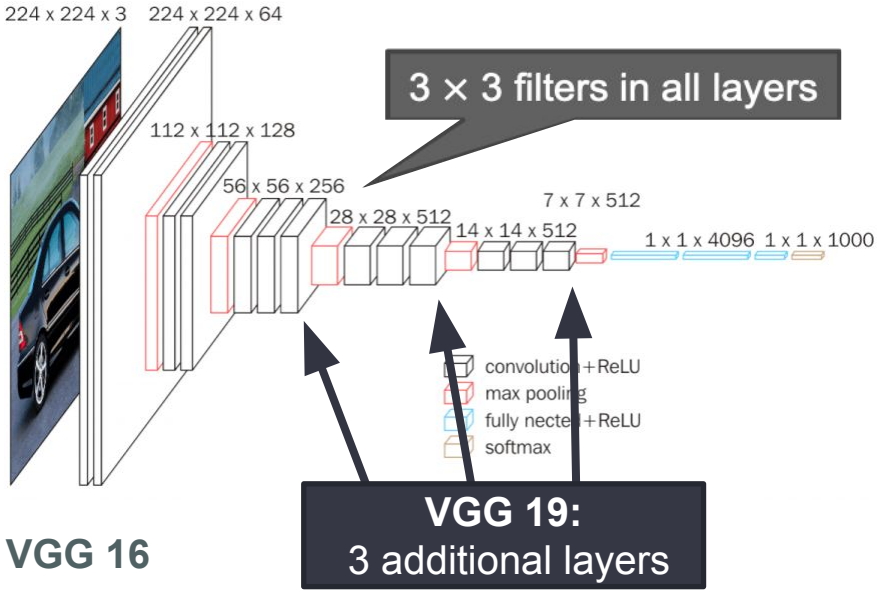
One of the first neural nets trained on a GPU with CUDA.

Paper cited over 80,000 times



VGG (2014)

```
torchvision.models.vgg16()
torchvision.models.vgg19()
torchvision.models.vgg19_bn()
…
```



By the Vision Geometry Group at Oxford

Only 3 × 3 filters and max pooling

Training: 3 weeks on 4 NVIDIA Titan Black GPUs, 6 GB RAM

First train smaller configurations, then inject layers in between:
11 □ 13 □ 16 □ 19 layers

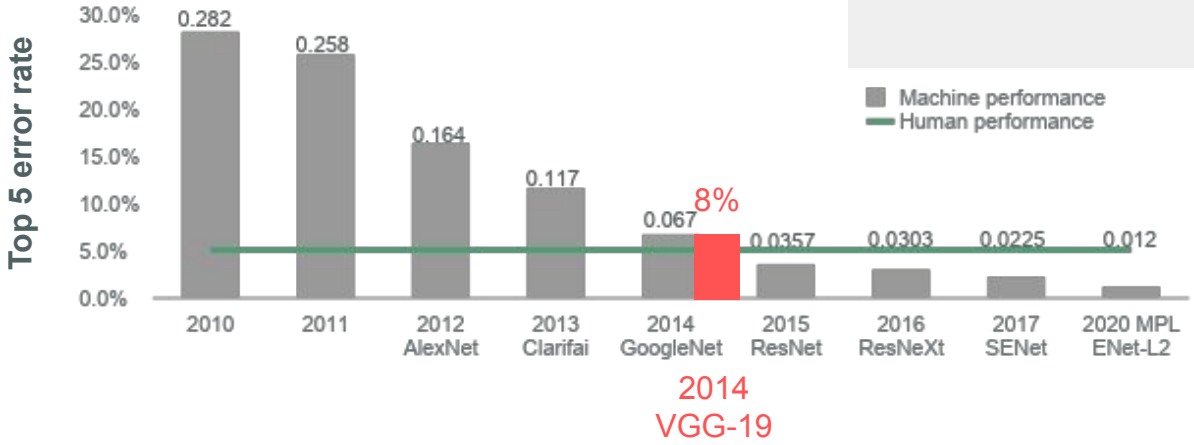
VGG-19: 138 million parameters / 500 MB

Simonyan & Zisserman, *ICLR* 2015

VGG (2014)

Has been very popular for a long time

- Particularly simple architecture
- Half the error rate of AlexNet
- Pre-trained net available

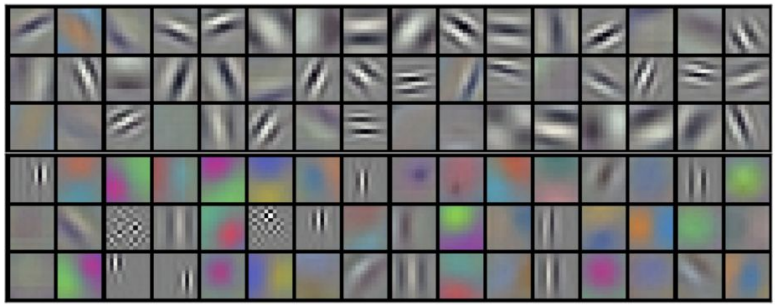


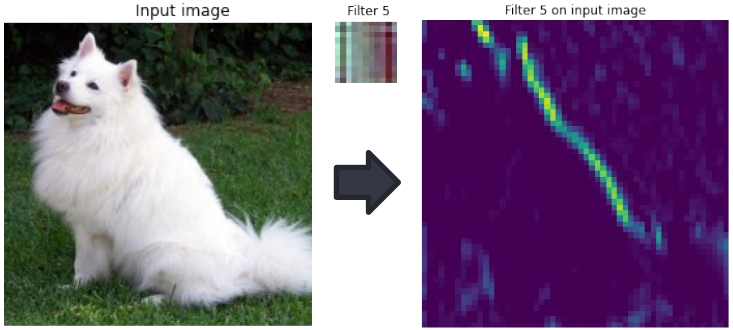
A couple of drawbacks:

- Slow and complex to train. Started with 11-layer version, then add additional layers, iterate until at 19
- Weights themselves are large: over 530 MB, which makes deploying it a tiresome task
- First couple of layers are very expensive to compute as they operate at full resolution

Exercise

Inspect what AlexNet has learned: filters and feature maps





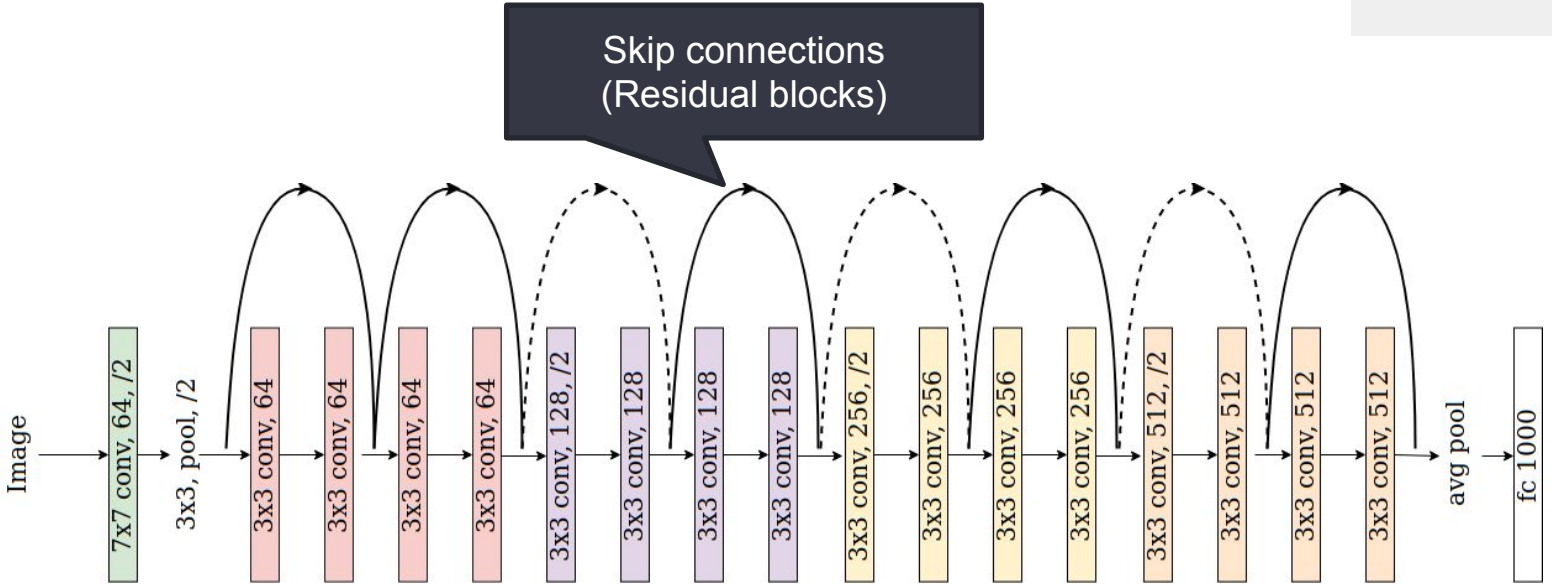
Going deeper

Residual Networks (ResNets)


WE NEED TO GO
DEEPER
troll-face.ru

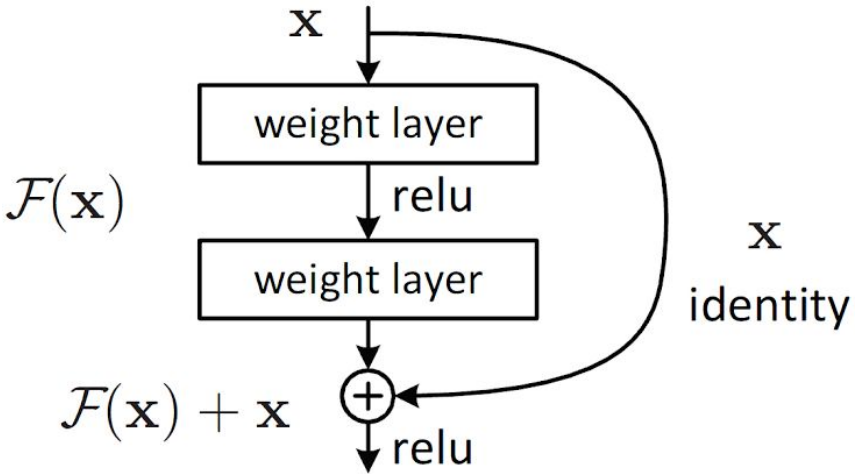
ResNet (2015)

```
torchvision.models.resnet18()
…
torchvision.models.resnet152()
```



He et al., *CVPR* 2016

Residual blocks (“skip connections”)



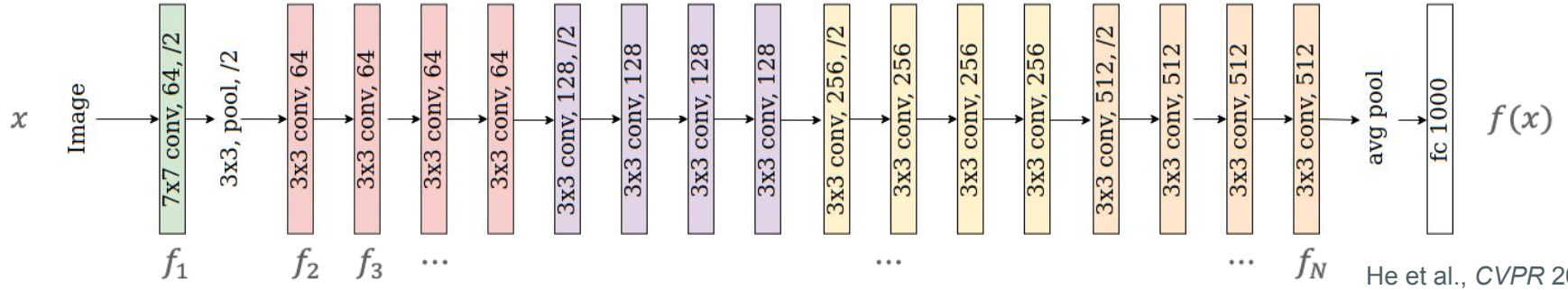
Each block learns to approximate a residual function

Better gradient flow because of skip connections

He et al., *CVPR* 2016

Skip connections avoid vanishing gradients

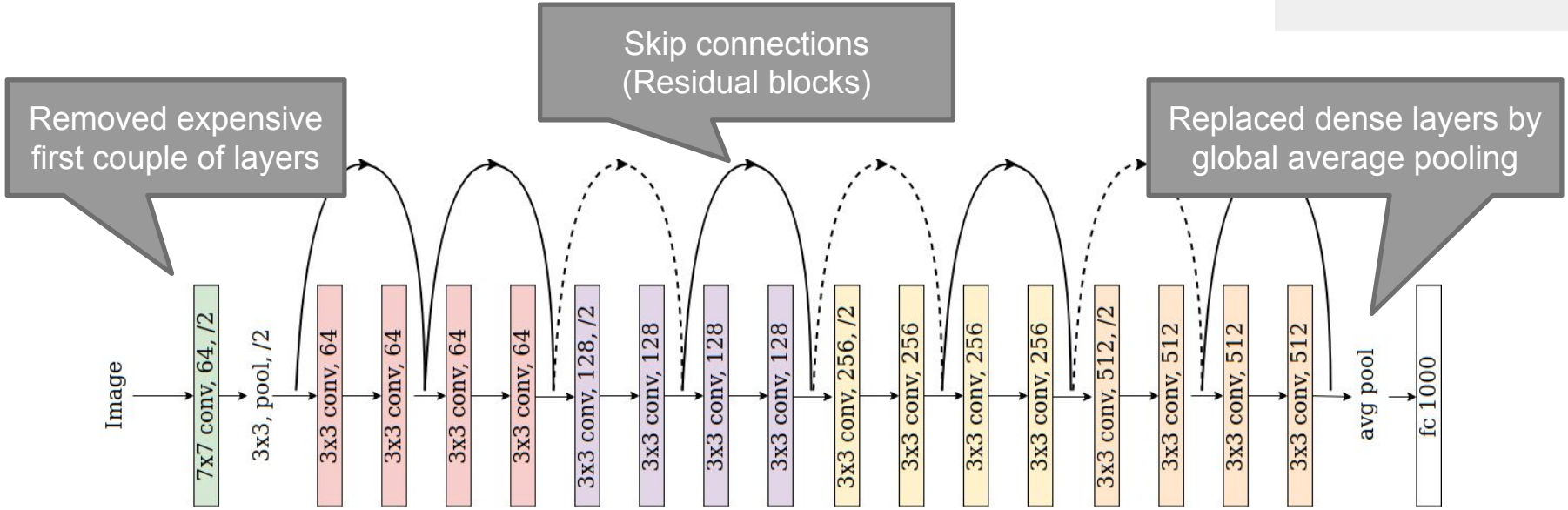
$$f(x) = f_1\left(f_2\left(\cdots f_N(x)\right)\right) \quad \Rightarrow \quad f'(x) = f_1'\left(f_2(\cdots)\right) \cdot f_2'(\cdots) \cdot \ldots \cdot f_N'(x) \qquad \textit{(chain rule)}$$



He et al., *CVPR* 2016

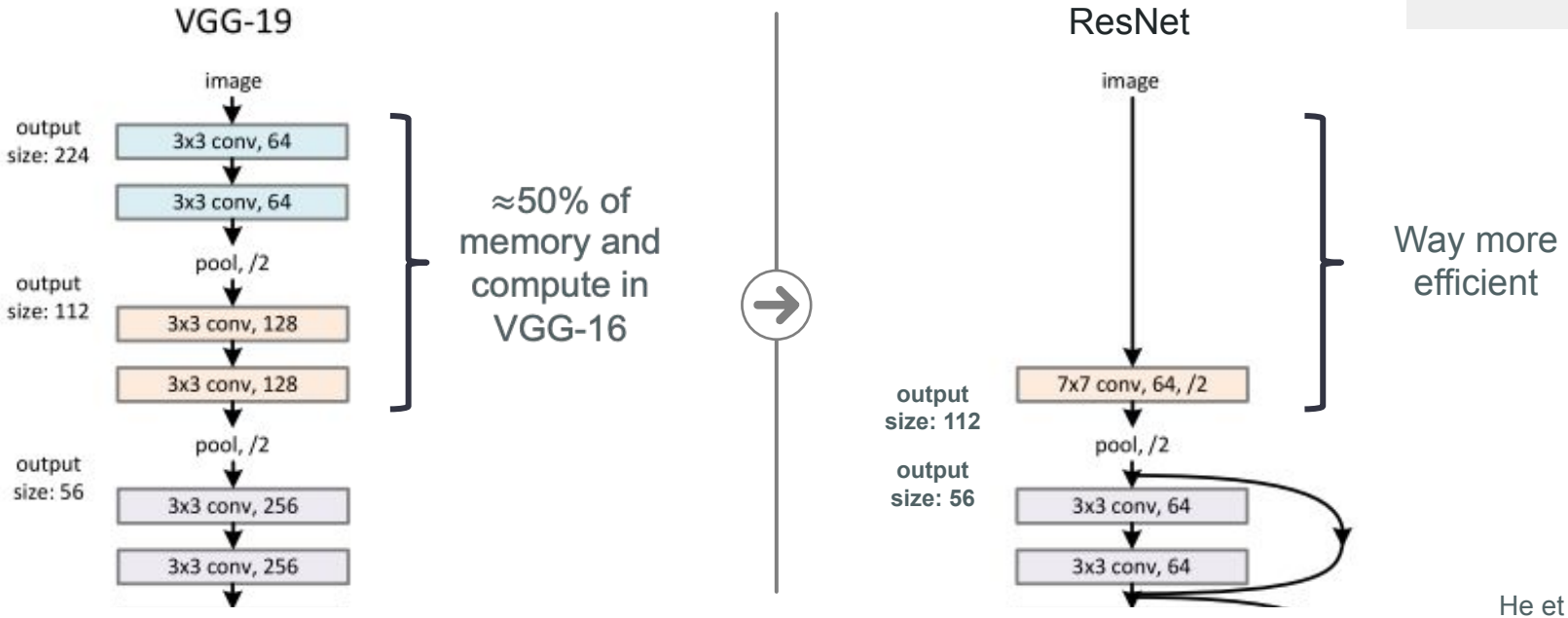
ResNet (2015)

```
torchvision.models.resnet18()
…
torchvision.models.resnet152()
```



He et al., *CVPR* 2016

ResNet: Remove expensive early layers



He et al., *CVPR* 2016

Exercise

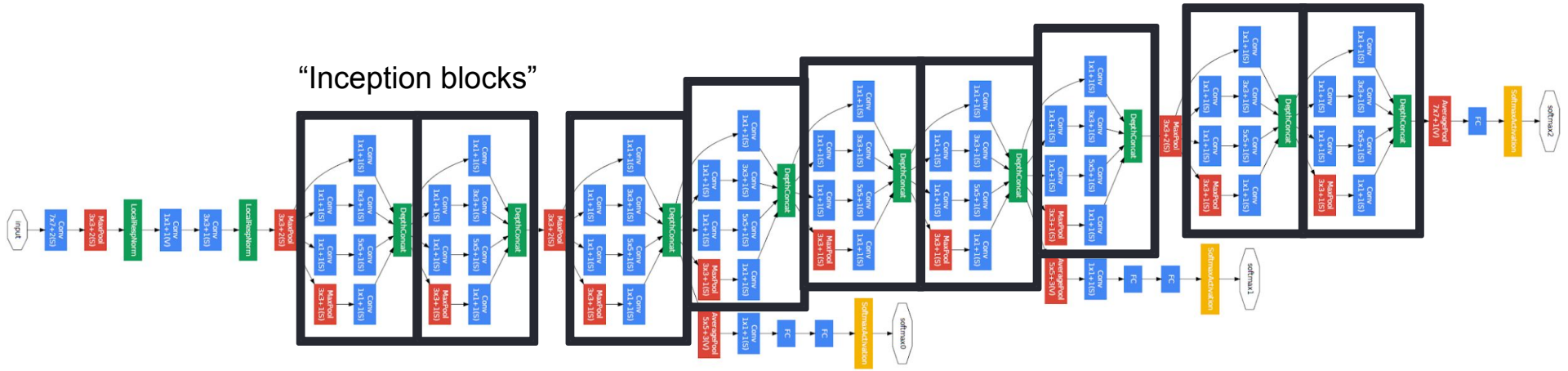
Load a pre-trained ResNet-18
Run a couple of images through the net
Inspect its predictions

How does it fare with non-photographic images?

Improving efficiency

Part 1: Inception + ResNeXt

Inception / GoogLeNet (2014/15)

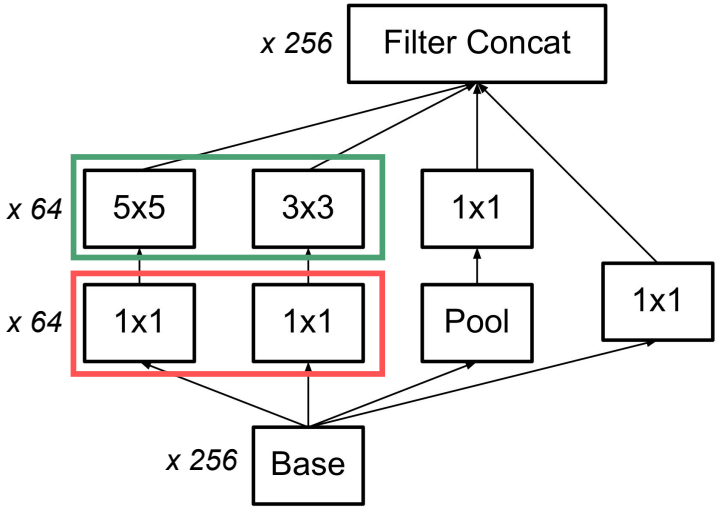


Inception v1: efficiency by 1 × 1 convolutions

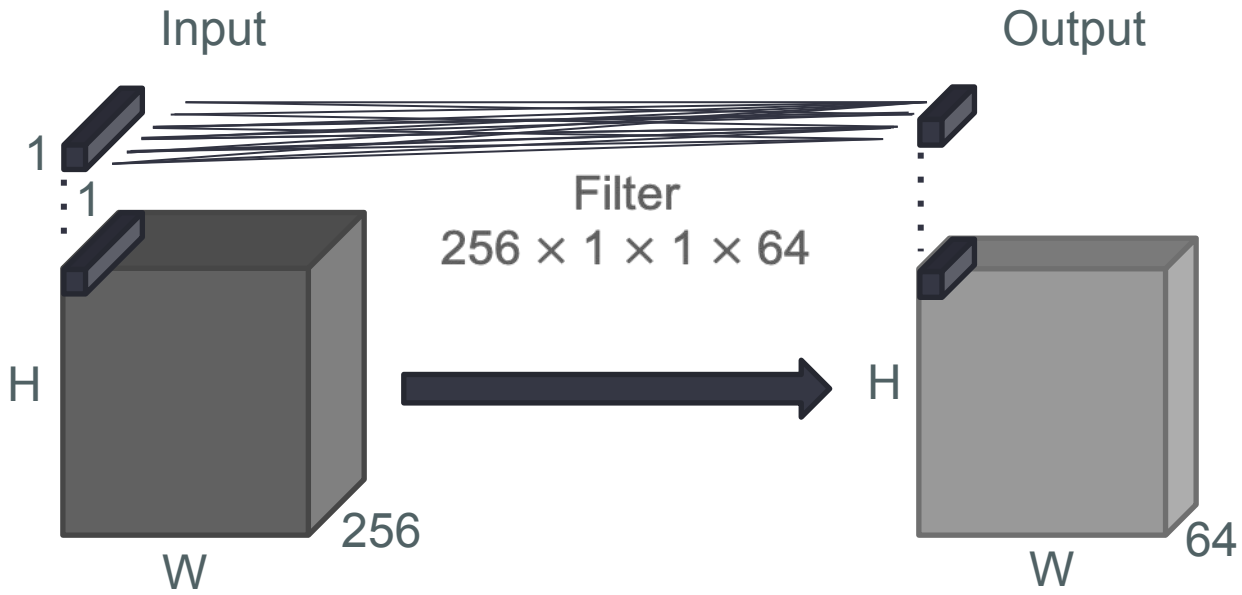
A single "Inception Module"

Spatial processing

Reducing number of feature maps



1 × 1 convolutions?



No spatial processing

Reduces number of channels

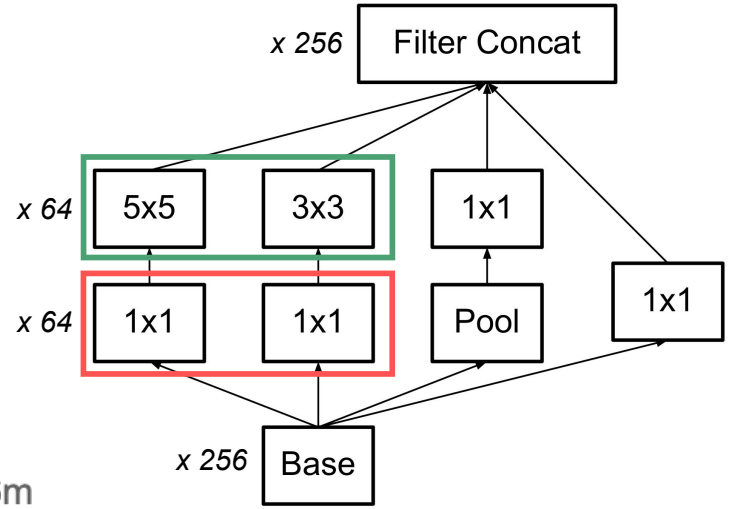
torchvision.models.googlenet()

Inception v1: efficiency by 1 × 1 convolutions

A single "Inception Module"

Spatial processing

Reducing number of feature maps

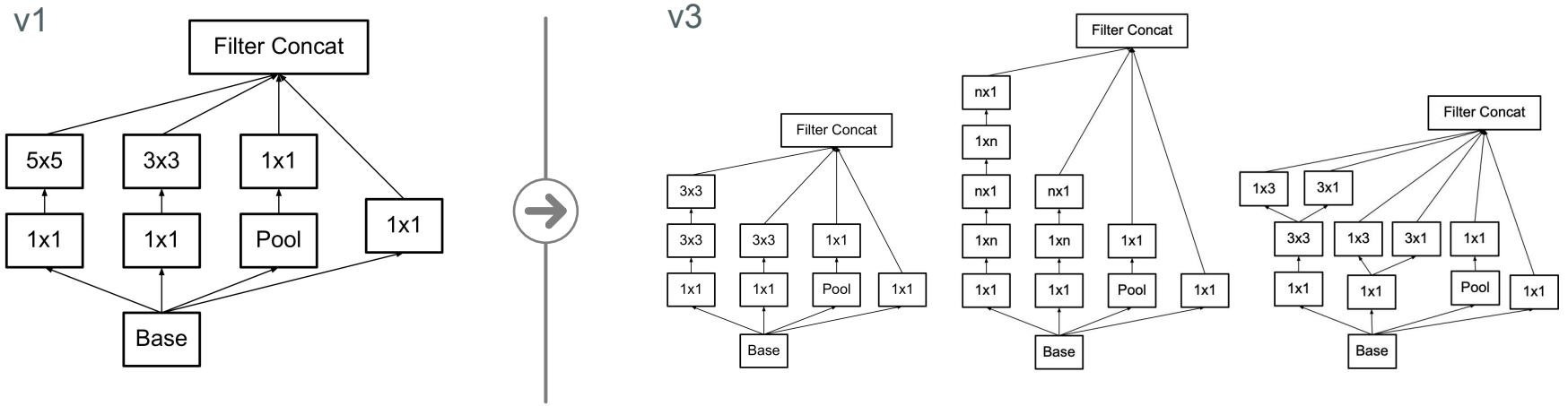


 # parameters

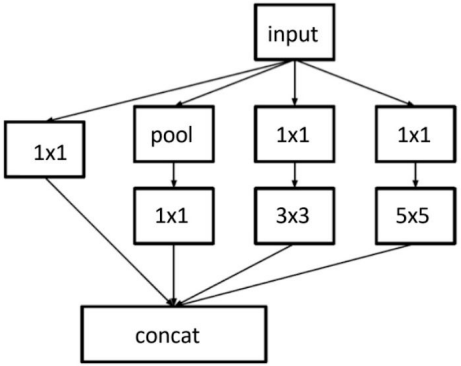
Standard convolution: 256 × 5 × 5 × 256 = 1.6m

Inception branch: 256 × 64 + 64 × 5 × 5 × 64 = 120k

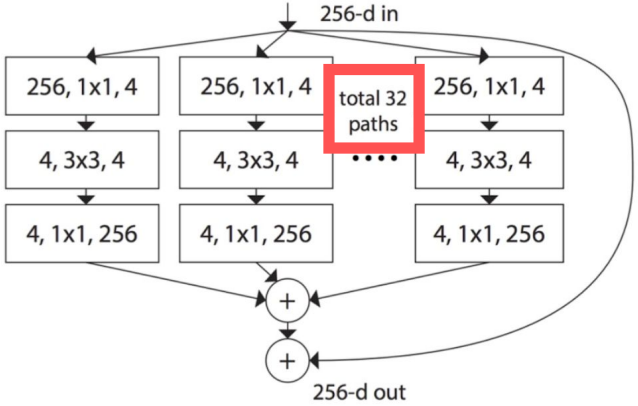
Inception v3: efficiency and bottlenecks



ResNeXt: combining Inception and ResNet



Inception:
heterogeneous multi-branch



ResNeXt:
uniform multi-branch

Compression

Spatial processing

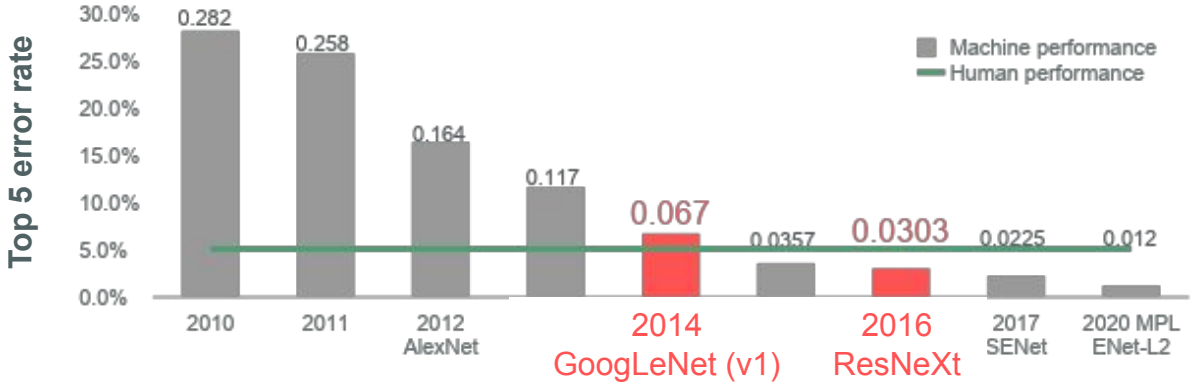
Channel mixing

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He.

Inception-v1 (2014) + ResNeXt (2016): Results

6.7% top-5 error rate in 2014, 3.0% in 2016!

Top-5 error rate of human expert (Andrej Karpathy) was ~5%



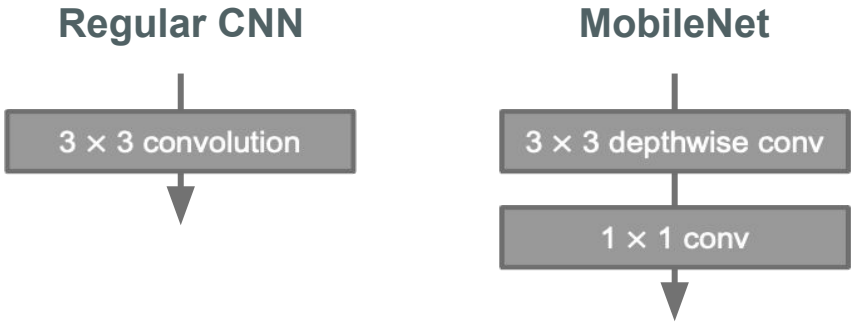
Exercise

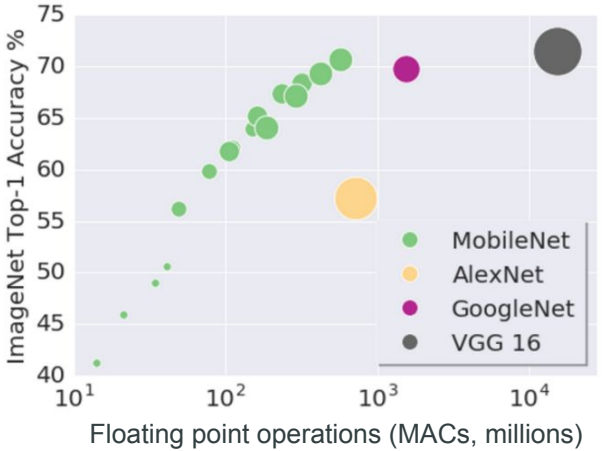
Explore how the multiple parallel paths of the ResNeXt architecture allow for increased network capacity without increasing the number of parameters.

Improving efficiency

Part 2: MobileNet / depthwise separable convolutions

MobileNet (2017): pushing efficiency





Same accuracy as VGG, >100 frames/sec on iPhone 7

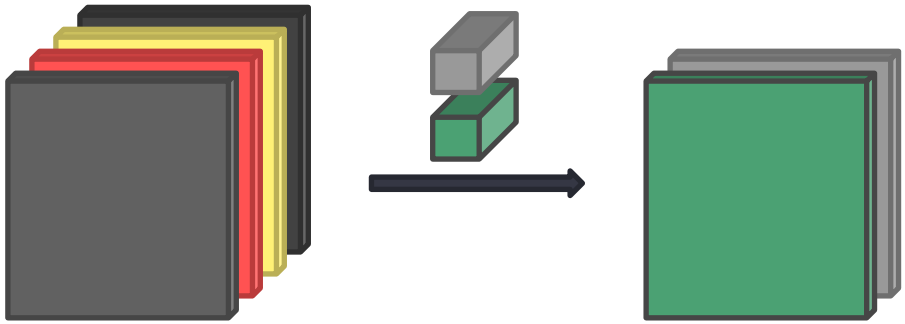
Howard et al., *CVPR* 2018

Recap: regular convolution layer

Input features
$H \times W \times 4$

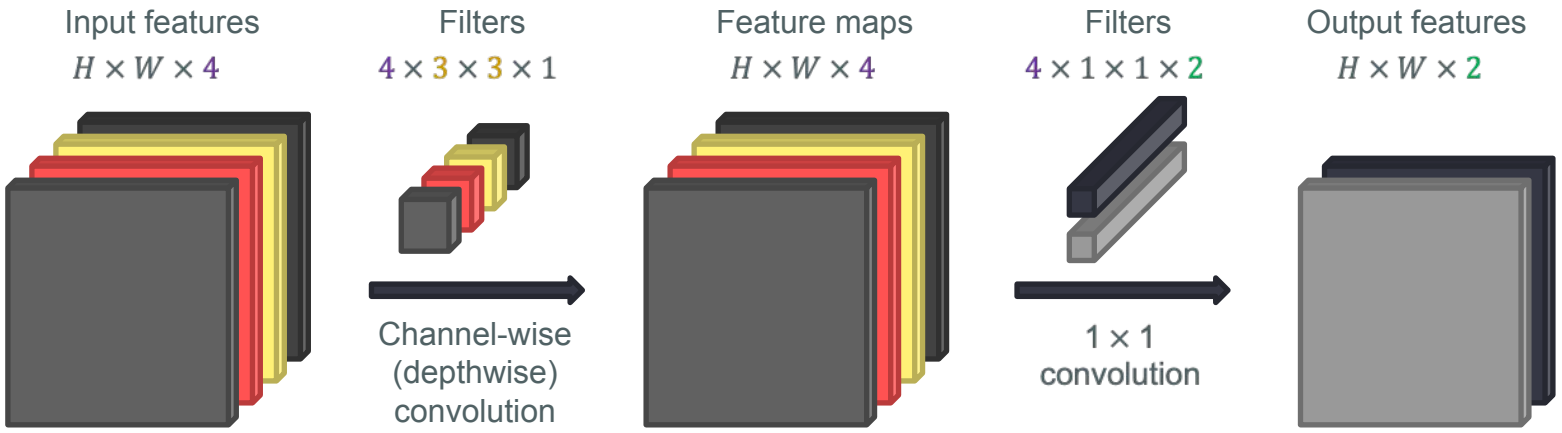
Filters
$4 \times 3 \times 3 \times 2$

Feature maps
$H \times W \times 2$



Regular convolution: $4 \cdot 3 \cdot 3 \cdot 2 = 72$ parameters

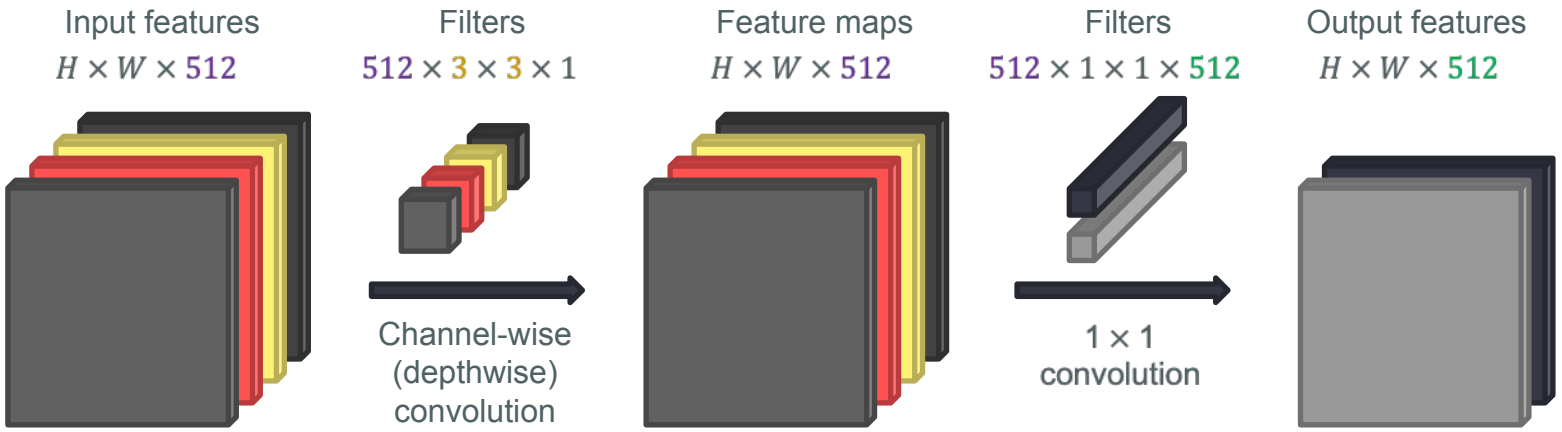
Depthwise separable convolution



Depthwise separable: $4 \cdot 3 \cdot 3 + 4 \cdot 2 = 44$ parameters $O(MK^2 + MN)$ parameters

Regular convolution: $4 \cdot 3 \cdot 3 \cdot 2 = 72$ parameters $O(MK^2N)$ parameters

Depthwise separable convolution



Depthwise separable: $512 \cdot 3 \cdot 3 + 512 \cdot 512 = 267\text{k}$ parameters

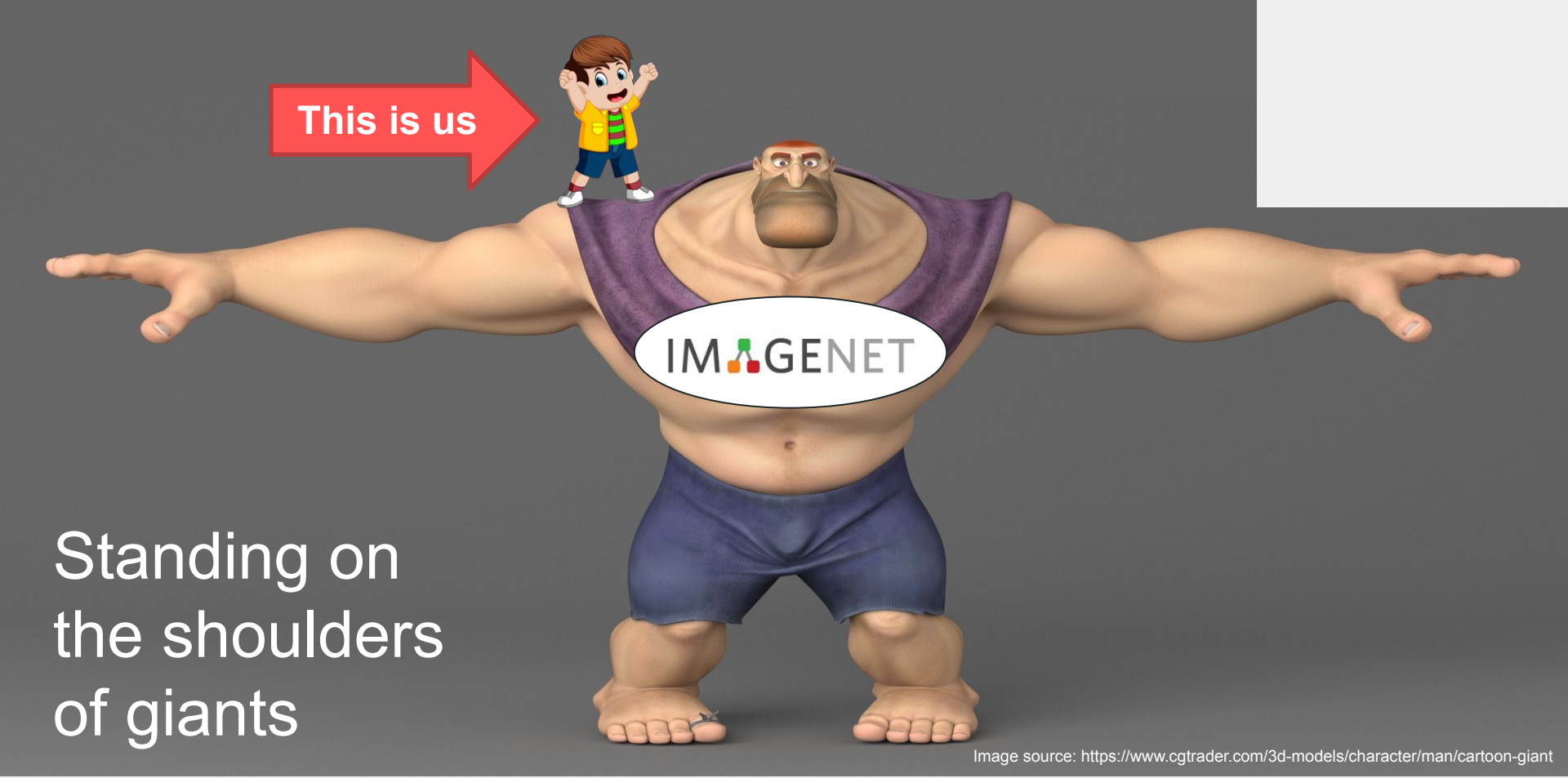
Regular convolution: $512 \cdot 3 \cdot 3 \cdot 512 = 2.4\text{m}$ parameters

Exercise

Understand how depthwise separable convolutions improve efficiency and save parameters in CNNs.

Transfer learning

Building upon pre-trained CNNs



Standing on the shoulders of giants

Image source: https://www.cgtrader.com/3d-models/character/man/cartoon-giant

Transfer learning: motivation

Suppose you want to build an app to recognize flowers…

Training a convolutional net from scratch won't work well

. . .

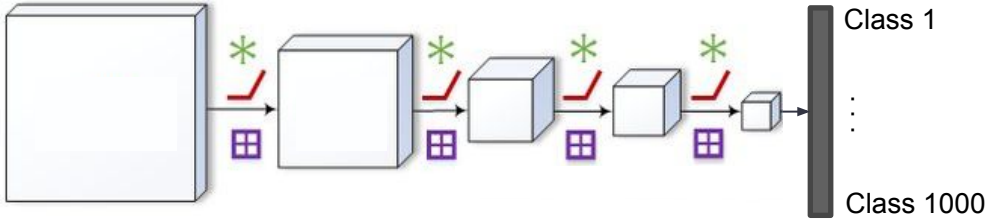
1000 images
100 classes

Image source: https://www.robots.ox.ac.uk/~vgg/data/flowers/102/

Transfer learning: the idea

1 Pre-training
(massive data)

IMAGENET
1.2 million images
1000 object classes

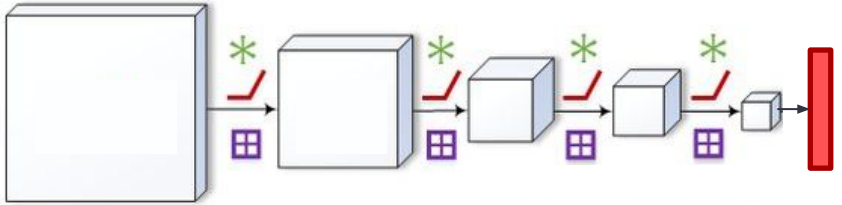


IMAGENET

Transfer weights

Replace classifier

2 Fine-tuning
(much less data)



Recognize flowers

Transfer learning: what should we train?

Option 1: Train only classification layer, freeze backbone

(sometimes referred to as the “linear evaluation protocol”)

- Fast & simple

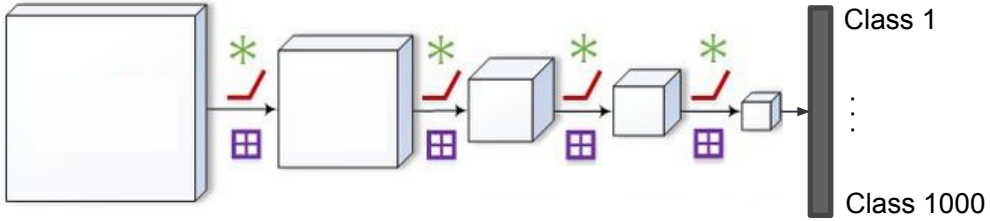
Option 2: Train classification layer, fine-tune backbone at the same time

- Slower, but can adapt feature extraction to dataset statistics

Transfer learning: beyond classification

1 Pre-training
(massive data)

IMAGENET
1.2 million images
1000 object classes

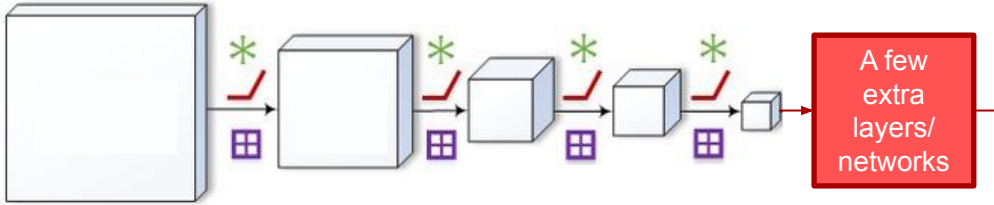


IMAGENET

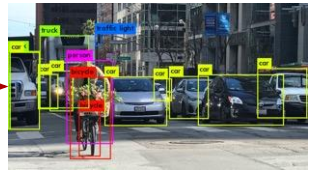
Transfer weights

Replace classifier

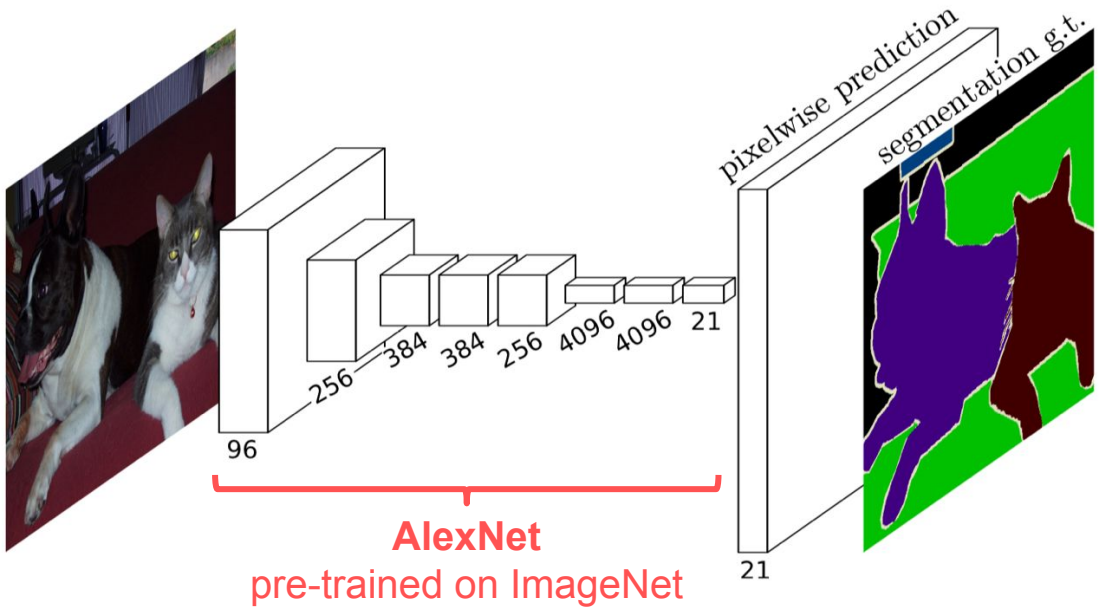
2 Fine-tuning
(much less data)



Object detection

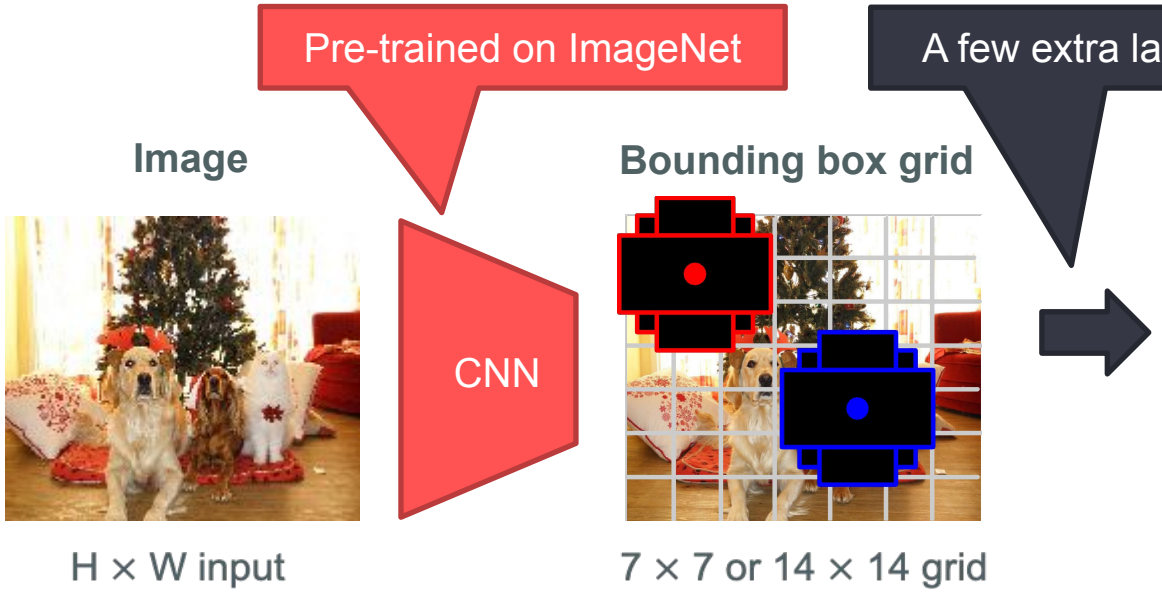


Example: semantic segmentation



Long, Shelhamer, Darrell, *CVPR* 2015

Single-stage object detectors: SSD, YOLO, …



For each grid cell

- Bounding box regression $(\Delta x, \Delta y, \Delta h, \Delta w)$
- Classification in C classes + background

Then output N boxes with highest confidence

Liu et al., *ECCV* 2016 | Redmon et al., *CVPR* 2016 | Lin et al., *CVPR* 2017

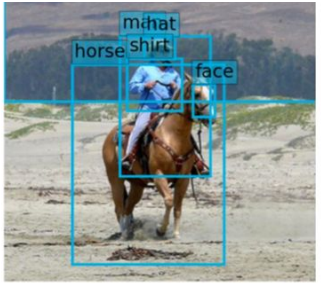
Pre-training on ImageNet is everywhere …

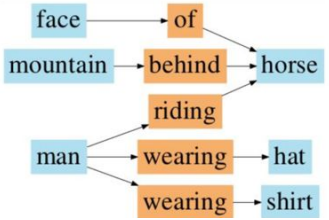
Pose estimation

Image captioning

"two young girls are playing with lego toy."

Scene graph prediction





. . .

Many, many more

Exercise

Implement transfer learning for Pokemon classification

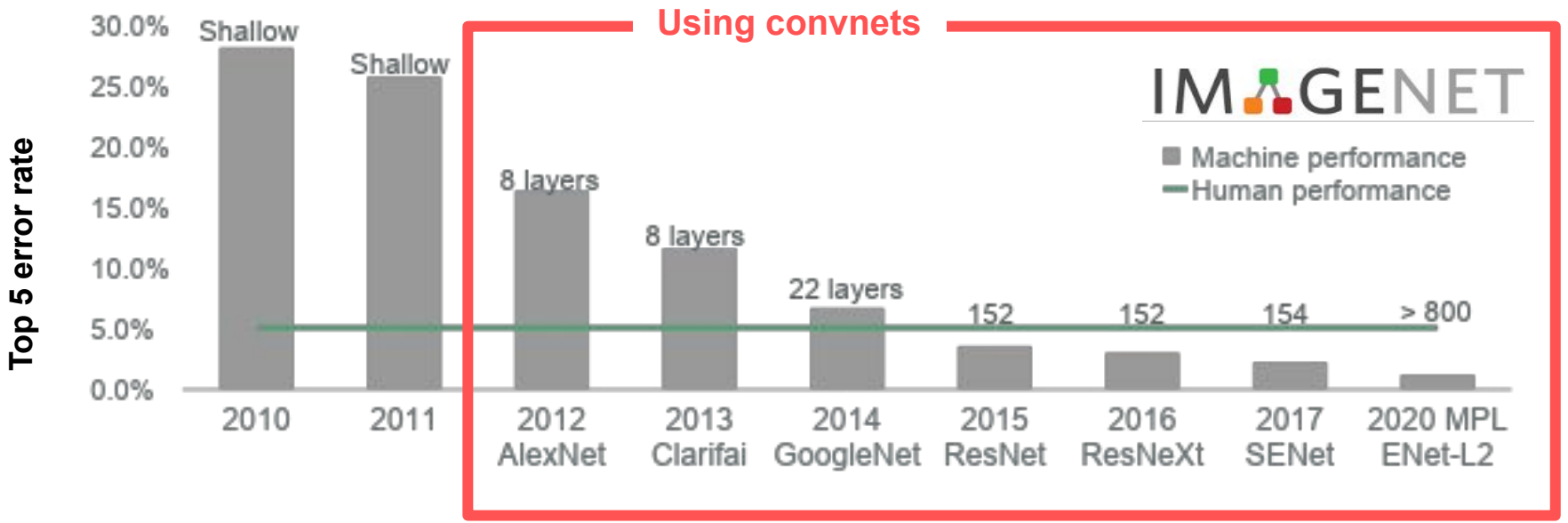
Which approach do you think will perform best?

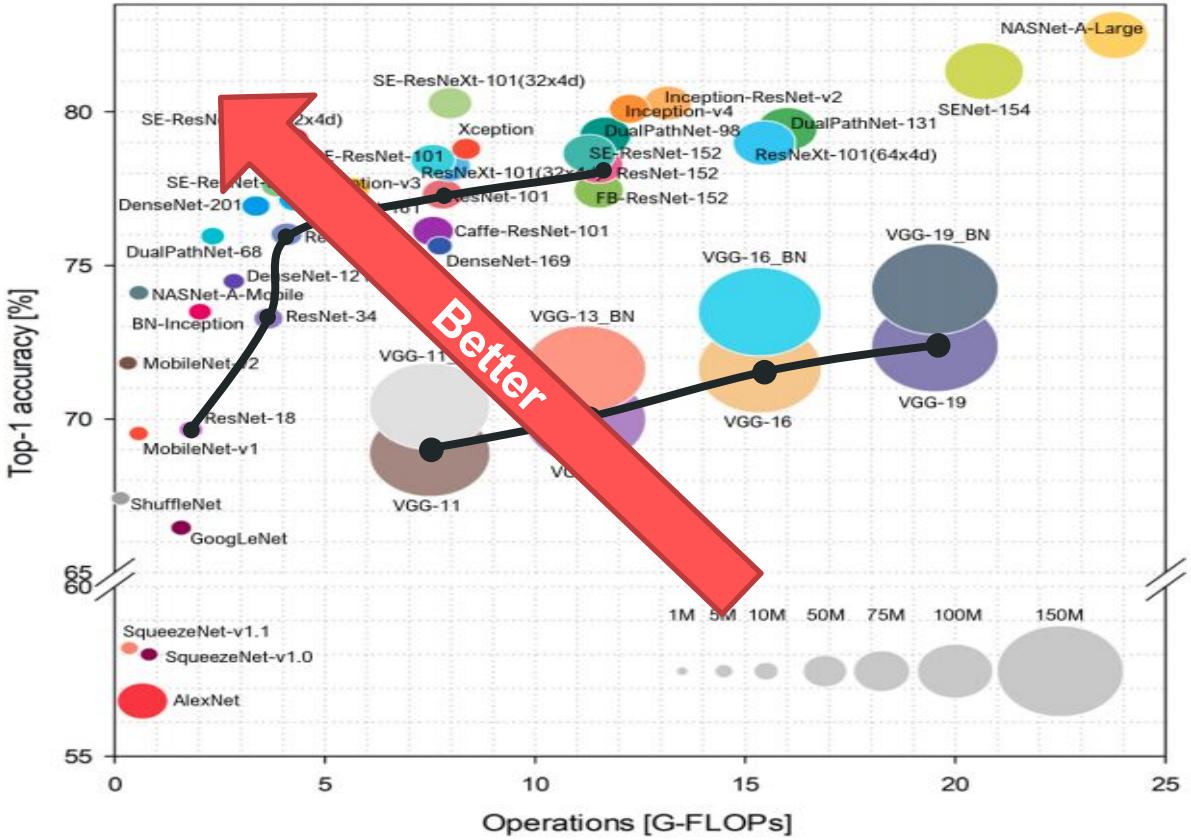
- Training from scratch
- Training only the classification layer
- Fine-tuning entire network

Speed-accuracy trade-off

How the choice of backbone affects results

Deeper = better?

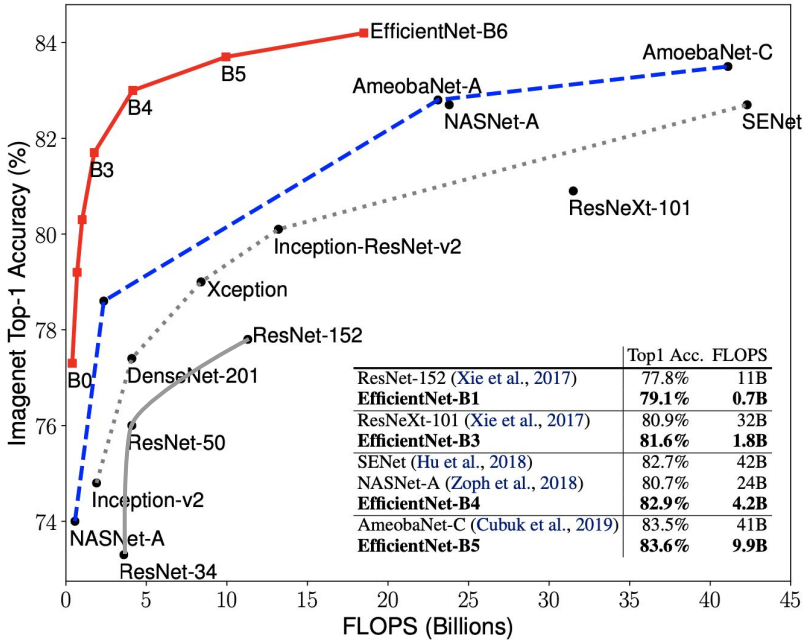




Many modern CNNs are actually families of CNNs

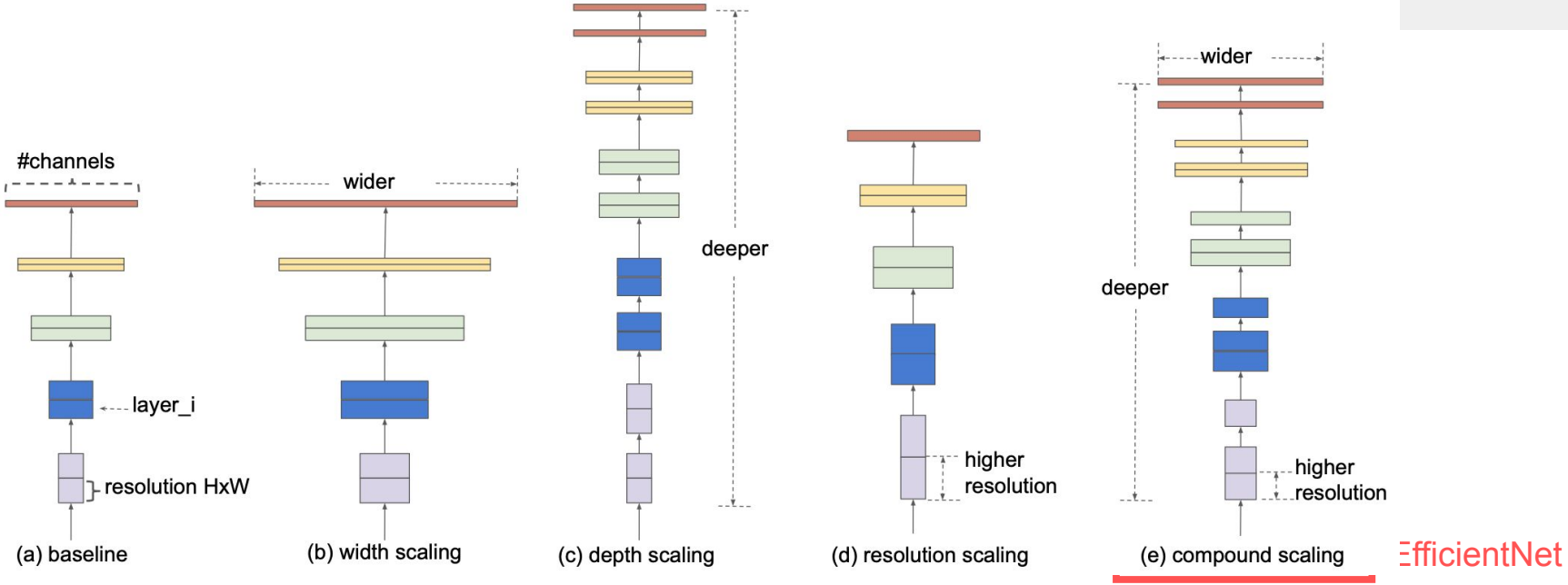
Deeper nets within a familiy perform better, but are more expensive

EfficientNet (2019)

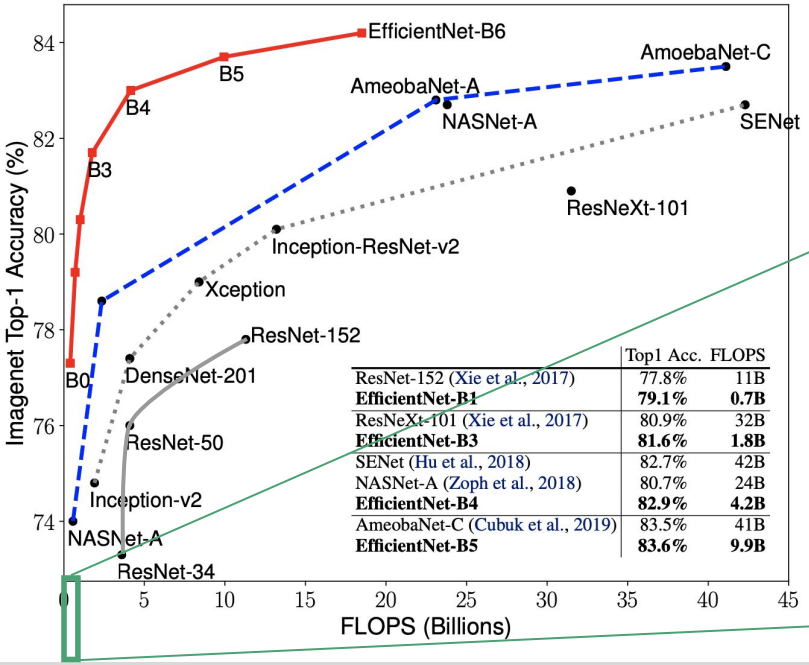


	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

EfficientNet (2019): how to scale networks?

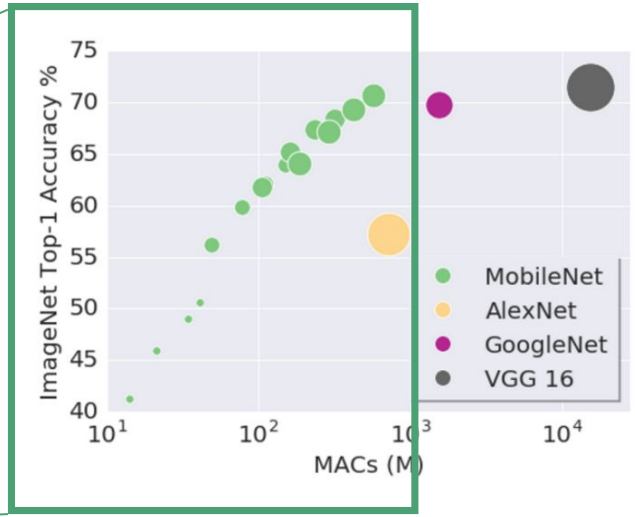


EfficientNet (2019): Results



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

MobileNet



Better ImageNet models lead to improvements on downstream tasks

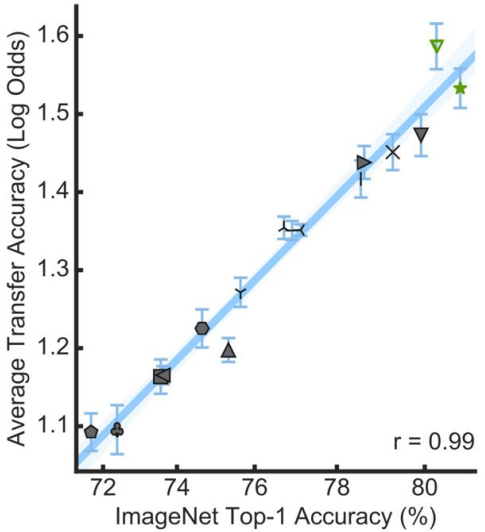
Do Better ImageNet Models Transfer Better?

Simon Kornblith and Quoc V. Le

{skornblith,shlens,qvl}@google.com

Yes!

Tested on 12 different image classification sets



How to choose a backbone for transfer learning?

Is speed or accuracy more important?

- Speed: MobileNet
- Accuracy: Use the best ImageNet model you can get your hands on

Availability of code and pre-trained weights (e.g. torchvision)?

Exercise

Explore speed-accuracy trade-off by using different backbones pre-trained on ImageNet for transfer learning.