Deep Learning 1, lecture 4
Convolutional Neural Network (CNN) architectures

Sadrtdinov Ildus, 17.10.22

LeNet

- **Convolutions** and **Subsampling** (i.e. **Pooling**) to extract features
- **Fully-connected network** as a classification head
- Works with fixed image size (!)

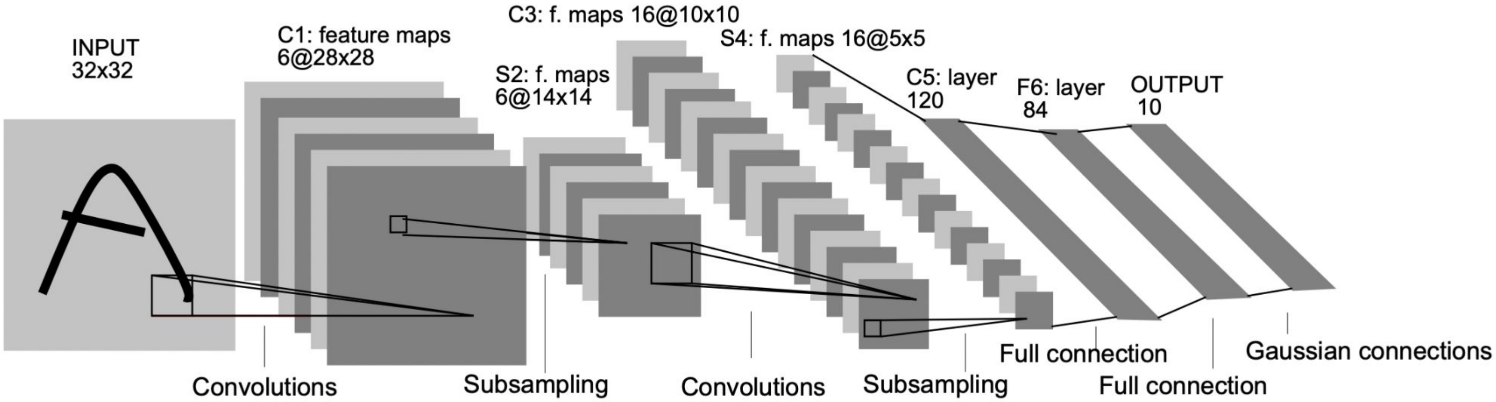


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

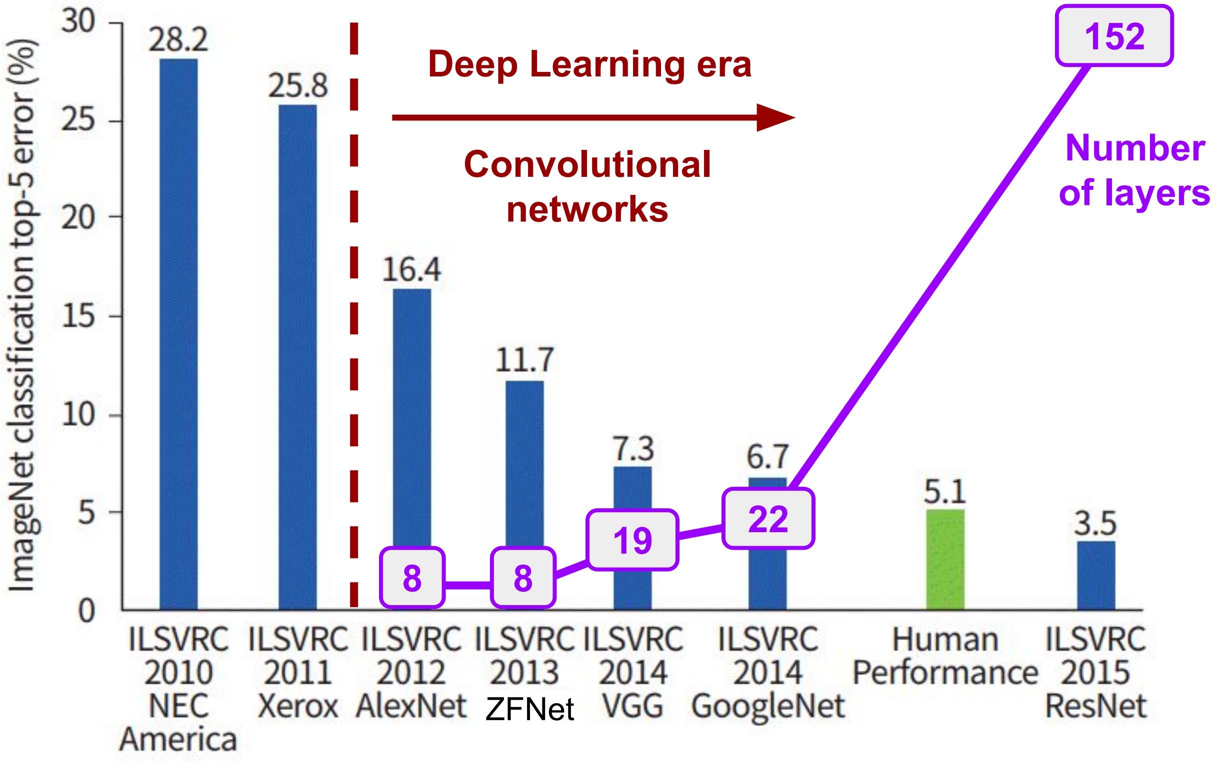
[Le et al., 1998]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- 1000 classes
- ILSVRC-2012: 1.2M images (~150Gb)
- Currently: >14M images in total
- Sped up development of deep learning

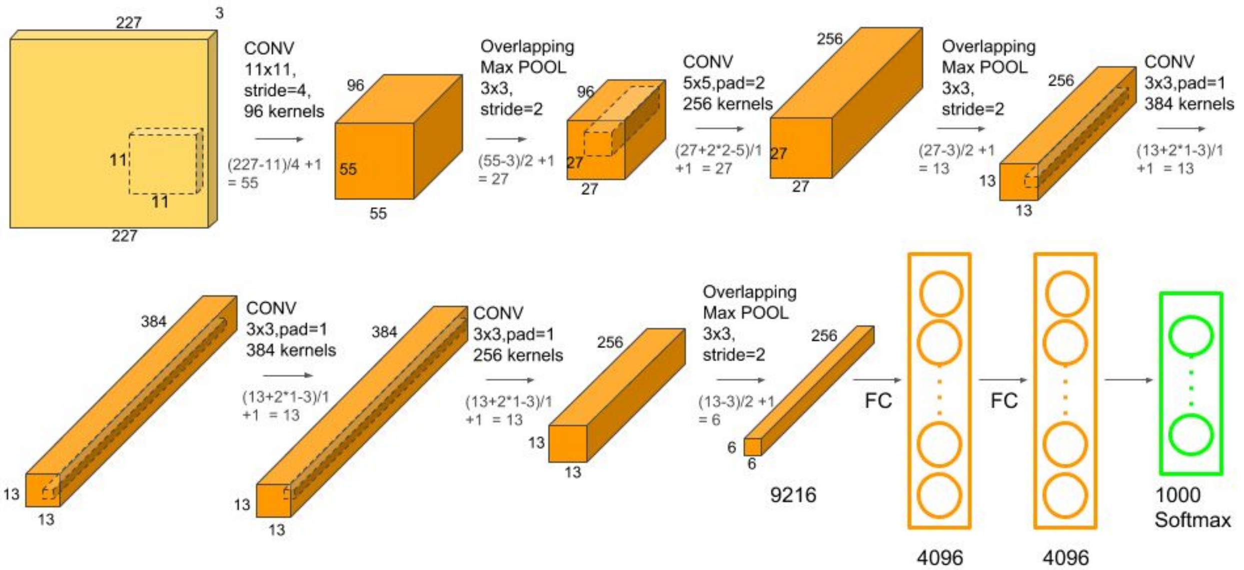
O.Russakovsky et al., 2015

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



AlexNet (2012)

- **Max pooling**, **ReLU** non-linearity
- **Dropout** and **Image augmentations** to reduce overfitting
- 5 – 6 days on 2 NVIDIA GTX 580 3GB

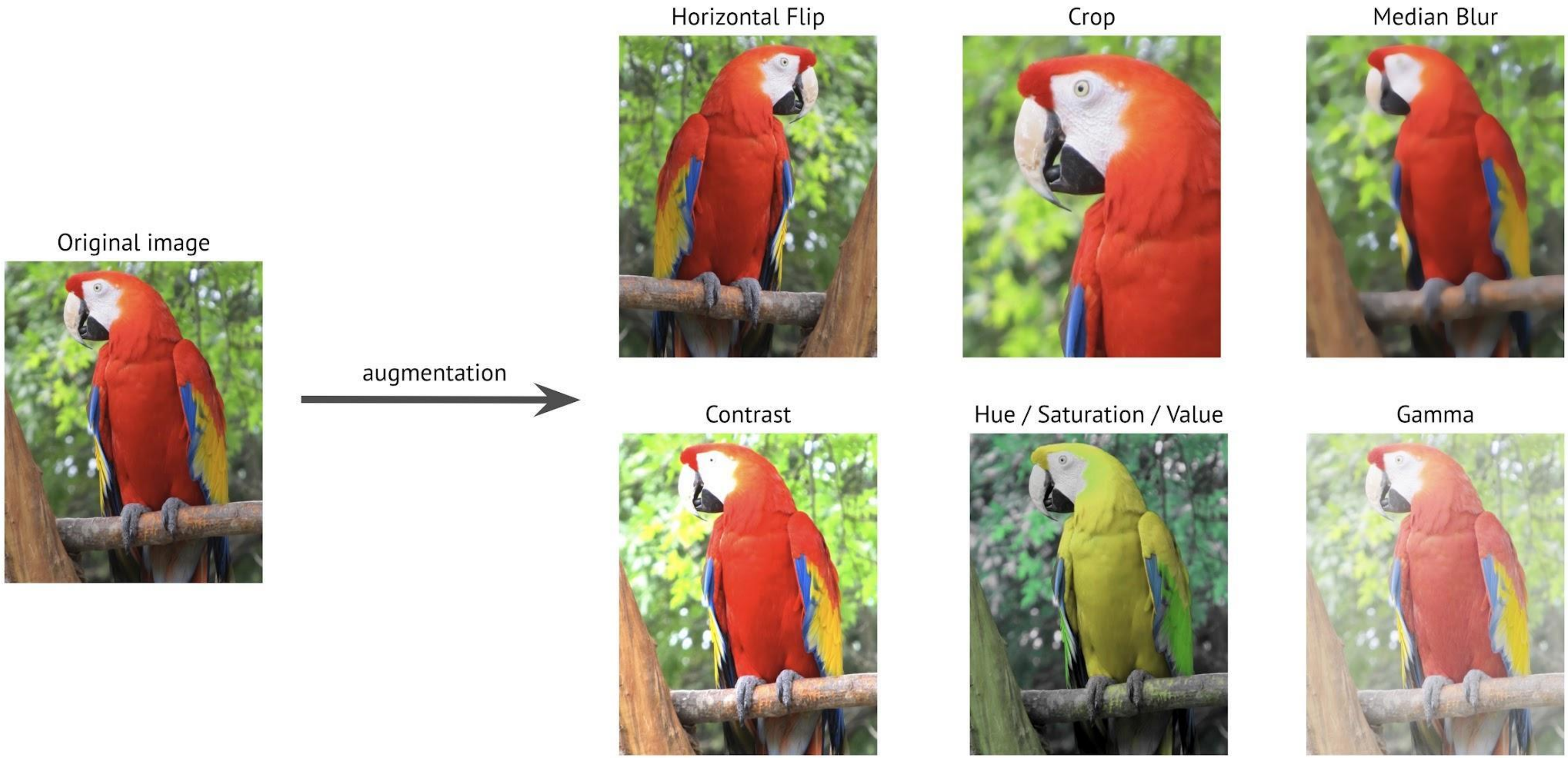


AlexNet		
Top-1 acc	**Top-5 acc**	**#params**
56.5	79.0	61.1M

**Tables taken from torchvision models*

[Krizhevsky et al., 2012](#)

Image augmentations



ImageNet augmentations

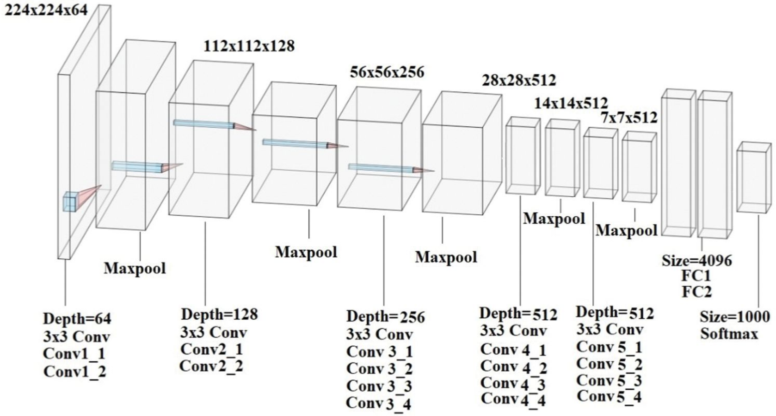
```
import torchvision.transforms as T

train_transform = T.Compose([
    T.RandomResizedCrop(224),
    T.RandomHorizontalFlip(),
    T.ToTensor(),
    T.Normalize(mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225])
])

test_transform = T.Compose([
    T.Resize(256),
    T.CenterCrop(224),
    T.ToTensor(),
    T.Normalize(mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225])
])
```

VGG (2014)

- **V**isual **G**eometry **G**roup
- Deeper than AlexNet (16 or 19 layers)
- Smaller convolutional kernels (3 x 3), more balanced computations
- Does not train end-to-end – vanishing gradients
- Trained in a few stages with increasing depth
- 2 – 3 weeks on 4 NVIDIA Titan Black GPU

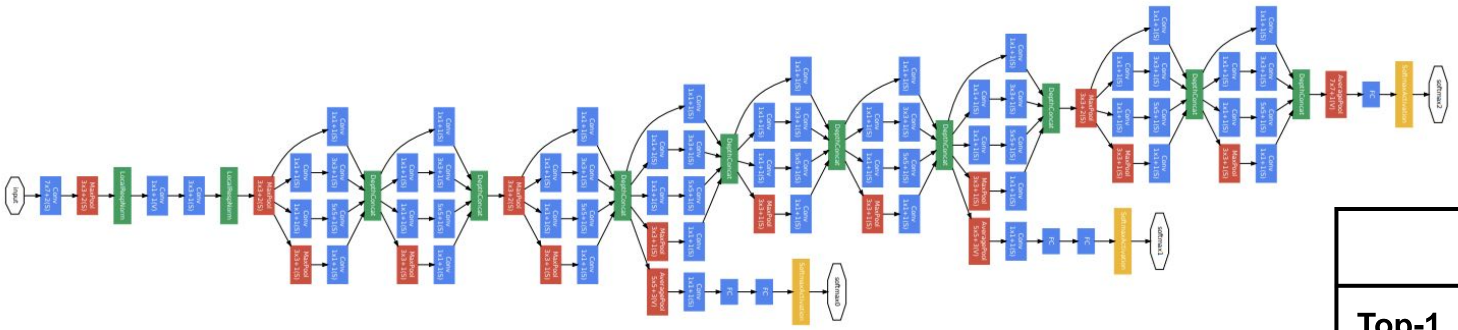


VGG-19		
Top-1 acc	**Top-5 acc**	**#params**
72.4	90.9	143.7M

[Simonyan and Zisserman, 2014](#)

GoogLeNet (a.k.a Inception, 2014)

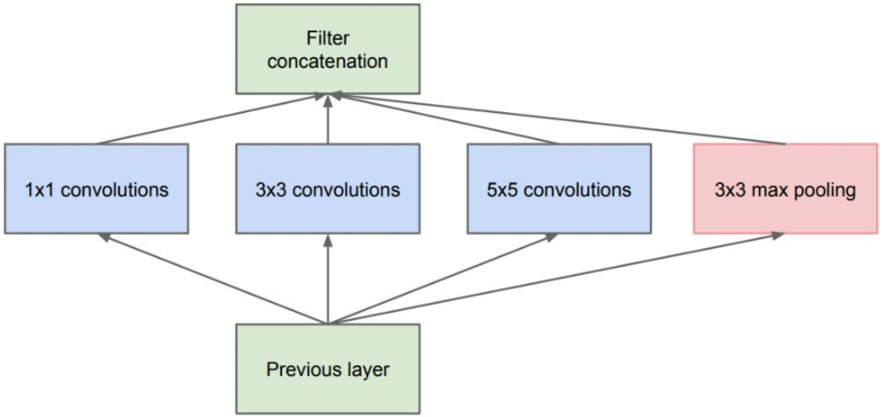
- Parallel computational blocks (architecture is **not** sequential)
- Utilizes convolutions with different kernel sizes



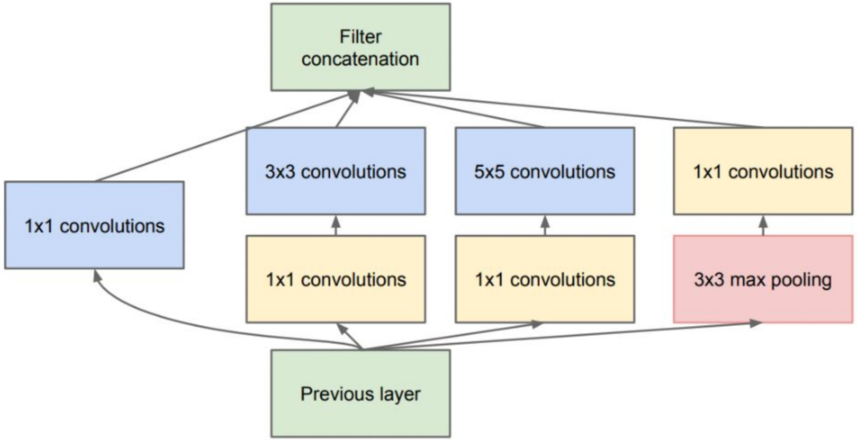
GoogLeNet		
Top-1 acc	**Top-5 acc**	**#params**
69.8	89.5	6.6M

Szegedy et al., 2014

Inception module



(a) Inception module, naïve version



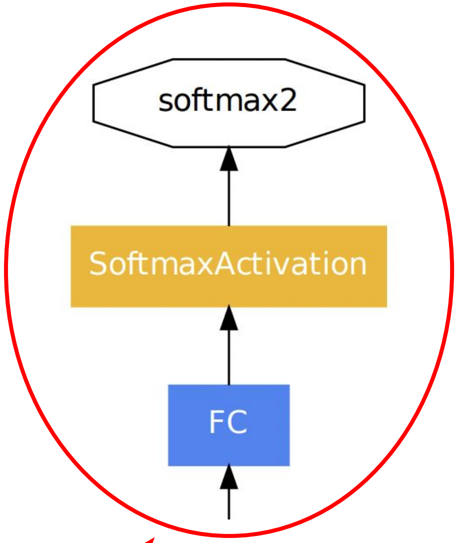
(b) Inception module with dimension reductions

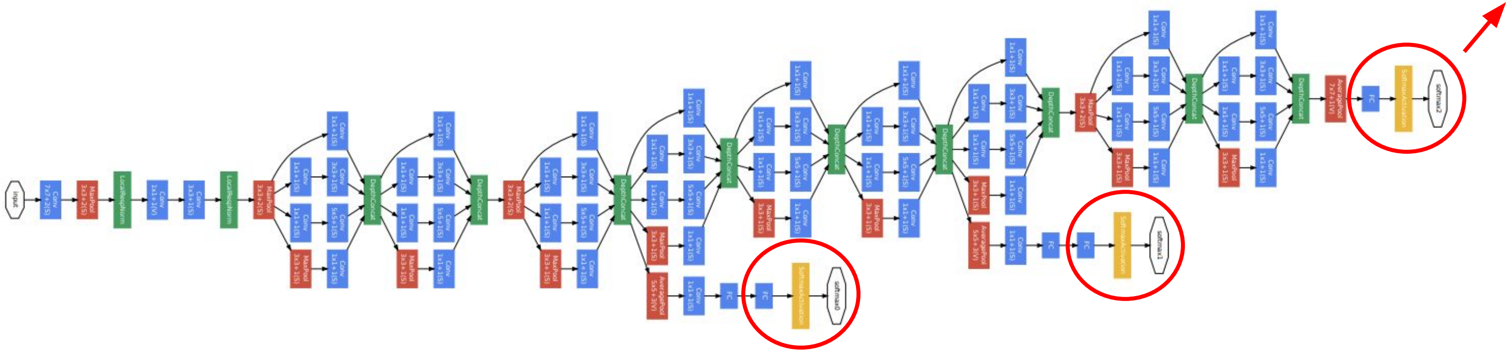
reduce number of channels with 1x1 convolutions =>
reduce computation time and memory consumption

Szegedy et al., 2014

GoogLeNet (a.k.a Inception, 2014)

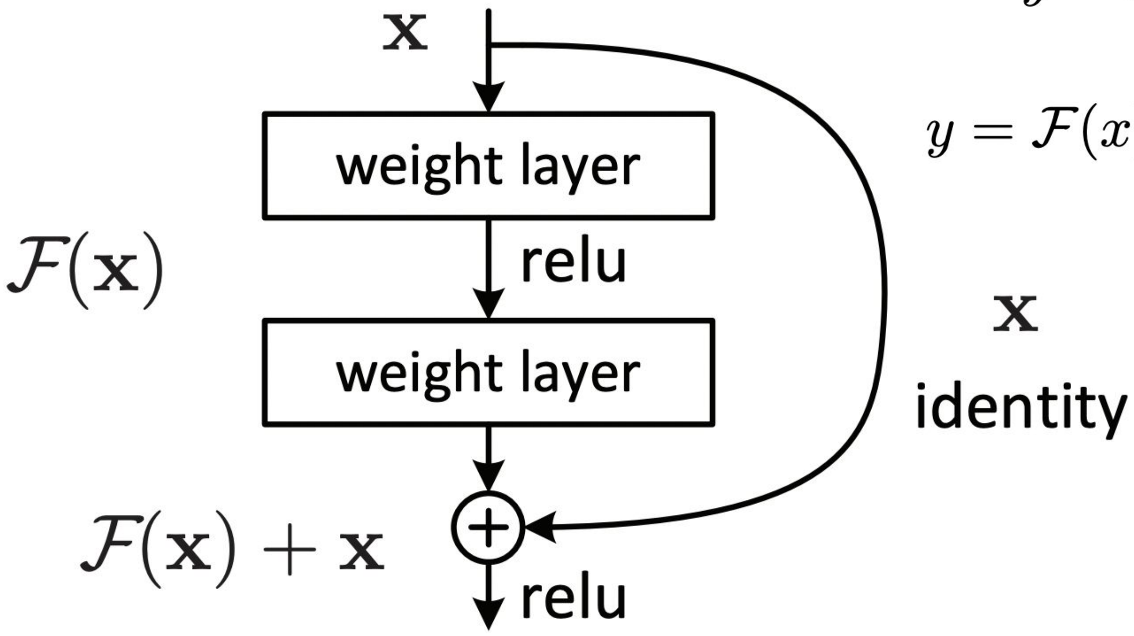
- Does not train end-to-end, needs **auxiliary classifiers** to propagate gradient





Szegedy et al., 2014

Residual (skip) connections

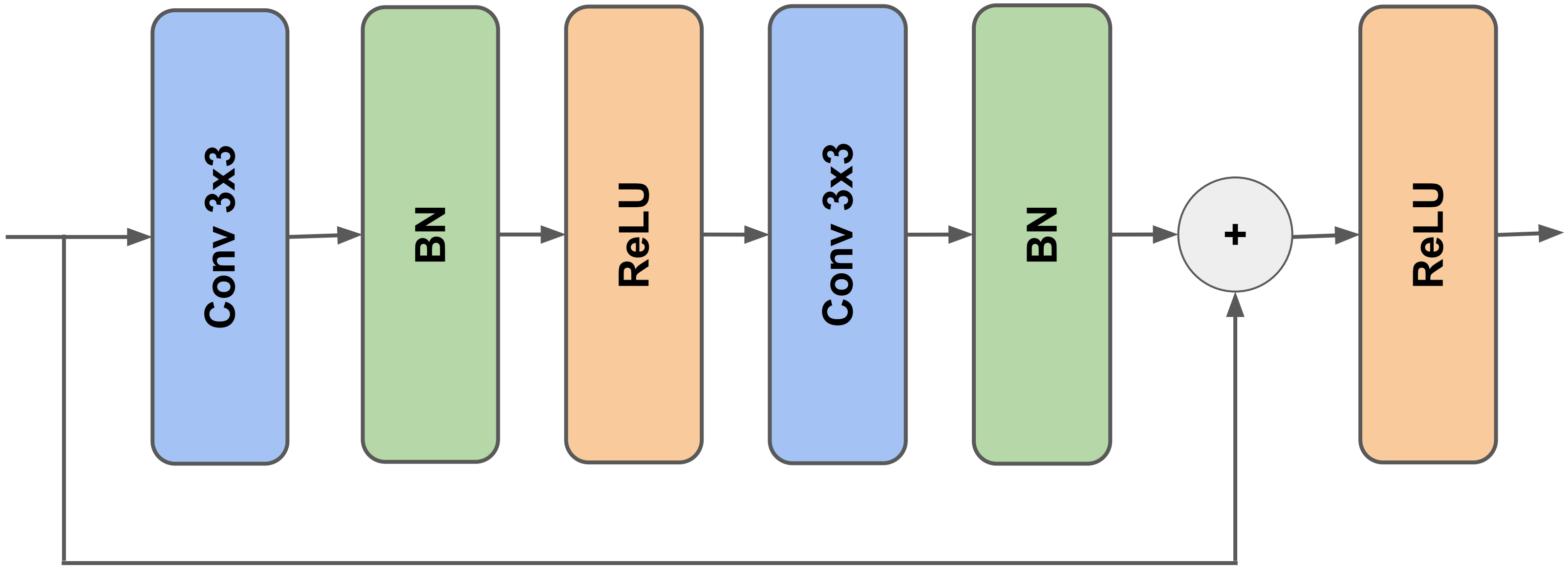


$$y = \mathcal{F}(x) \Rightarrow \frac{d\ell}{dx} = \frac{d\ell}{dy}\frac{d\mathcal{F}}{dx}$$

$$y = \mathcal{F}(x) + x \Rightarrow \frac{d\ell}{dx} = \frac{d\ell}{dy}\frac{d\mathcal{F}}{dx} + \frac{d\ell}{dy}$$

He et al., 2015

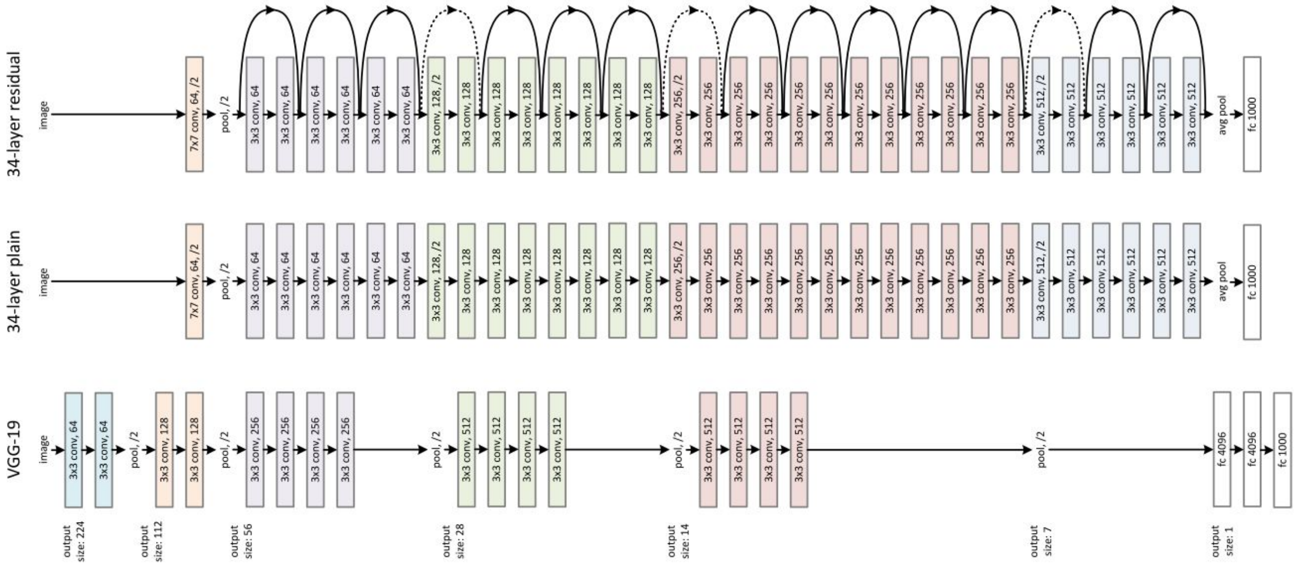
Residual block



He et al., 2015

ResNet (2015)

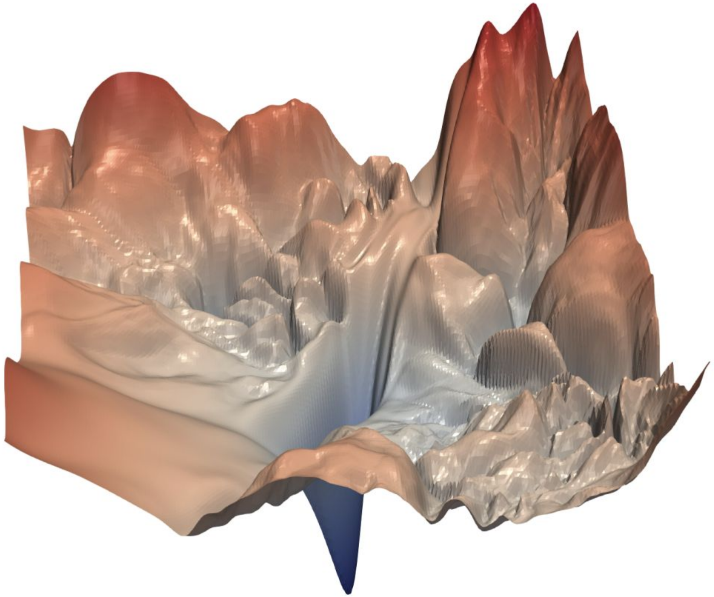
- Residual connections make gradient flow better => stack **MUCH MORE** layers
- **Batch Normalization** to stabilize training
- **No Max pooling** in blocks, strided convolutions instead
- **Global average pooling** of the resulting feature map => process arbitrary size images



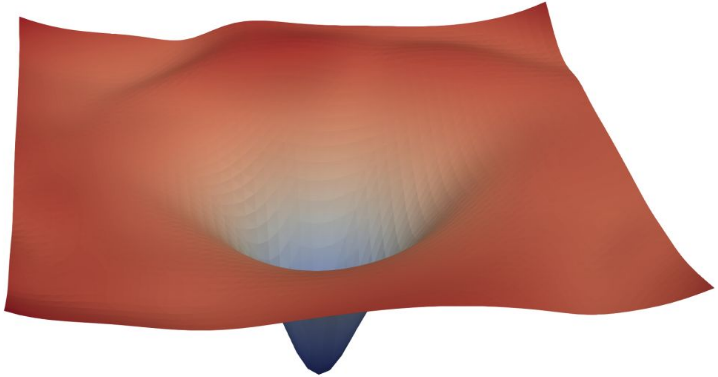
ResNet-152		
Top-1 acc	**Top-5 acc**	**#params**
78.3	94.0	60.2M

[He et al., 2015](#)

Why skip connections?



(a) without skip connections



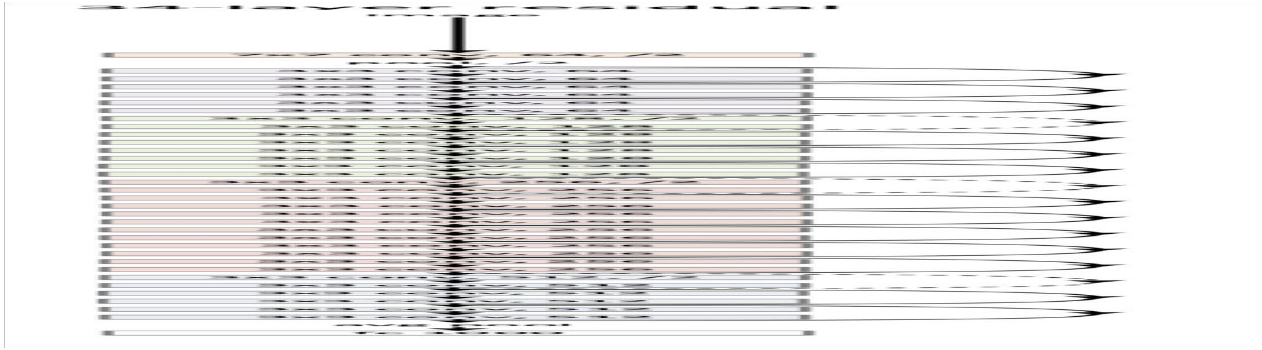
(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

[Li et al., 2017]

ResNet legacy

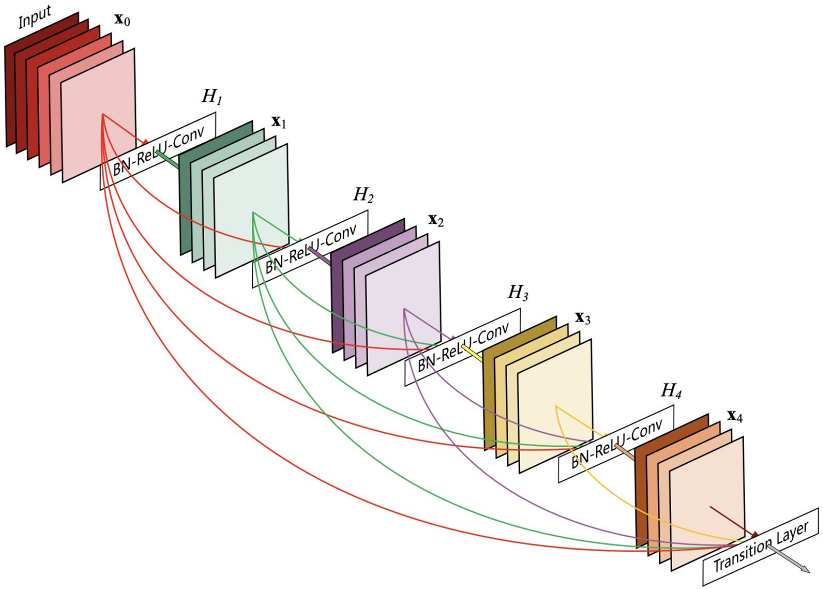
- All modern architectures use residual connections (even non-convolutional)
- ResNet direct successors:
  - **WideResNet** (Zagoruyko and Komodakis, 2016)
  - **ResNeXt** (Xie et al., 2016)
  - **Pre-activation ResNet** (He et al., 2016)
- ResNet-18/34/50 networks are the most frequent CNN baselines



He et al., 2015

DenseNet

- Any 2 layers are connected
- Channel-wise feature map concatenation
- Very narrow layers (i.e. small number of channels)
- Less parameters than in ResNet



DenseNet-201		
Top-1 acc	**Top-5 acc**	**#params**
76.9	93.4	20.0M

Huang et al., 2016

MobileNet

- Lightweight architecture to be used on mobile devices
- Uses combination of depthwise and pointwise convolution instead of a regular one

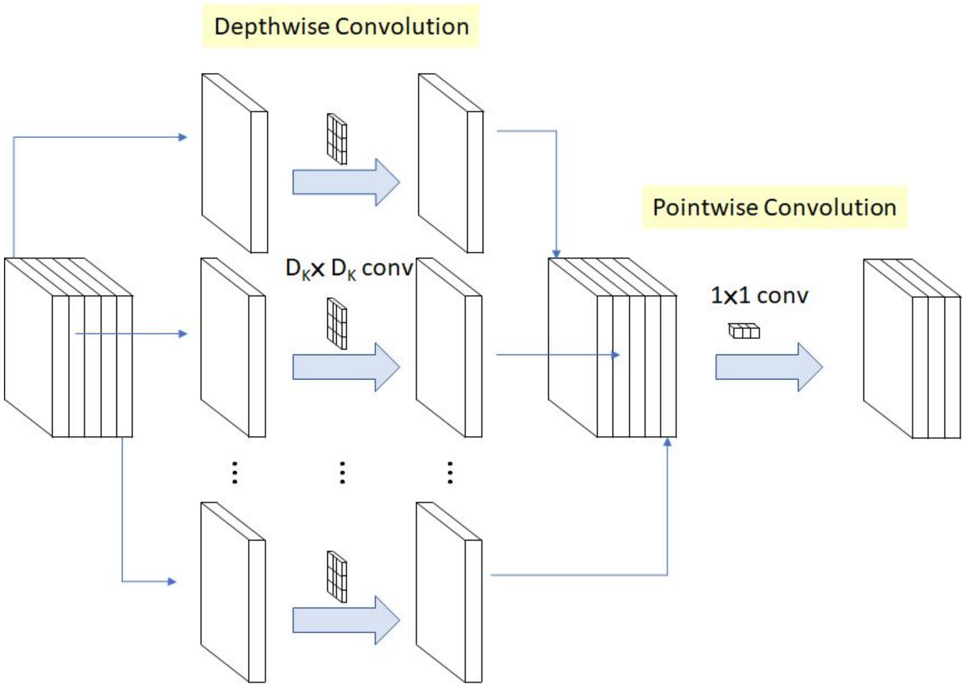


Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

MobileNet-v1		
Top-1 acc	**Top-5 acc**	**#params**
70.9	89.9	4.2M

**Table taken from tensorflow models*

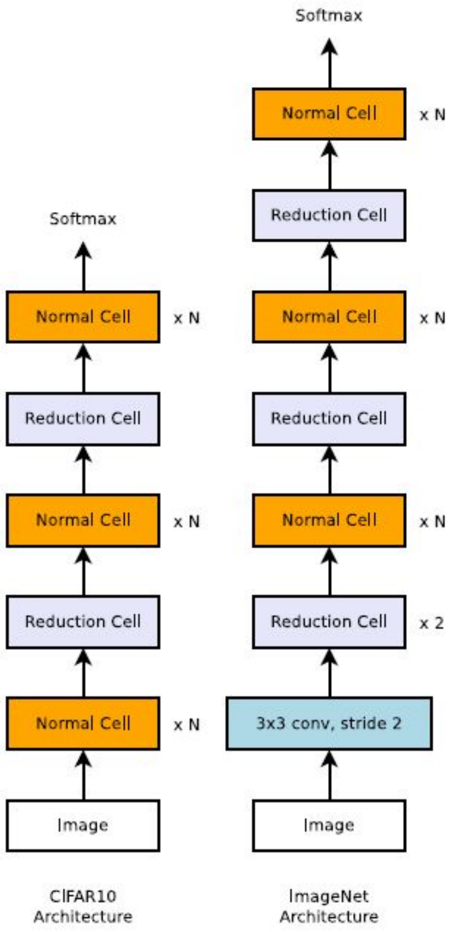
[Howard et al., 2017](#)

NASNet

- Neural Architecture Search (**NAS**) – meta-optimization of convolutional architecture
- Optimize architecture for small dataset (**CIFAR-10**), then transfer to large dataset (**ImageNet**)
- Search for optimal structure of **Normal cell** and **Reduction cell**
- Use so-called **Controller Network** to predict cells structure, trained using **Reinforcement Learning**

NasNet-A		
Top-1 acc	**Top-5 acc**	**#params**
74.0	91.6	5.3M

**Table taken from tensorflow models*

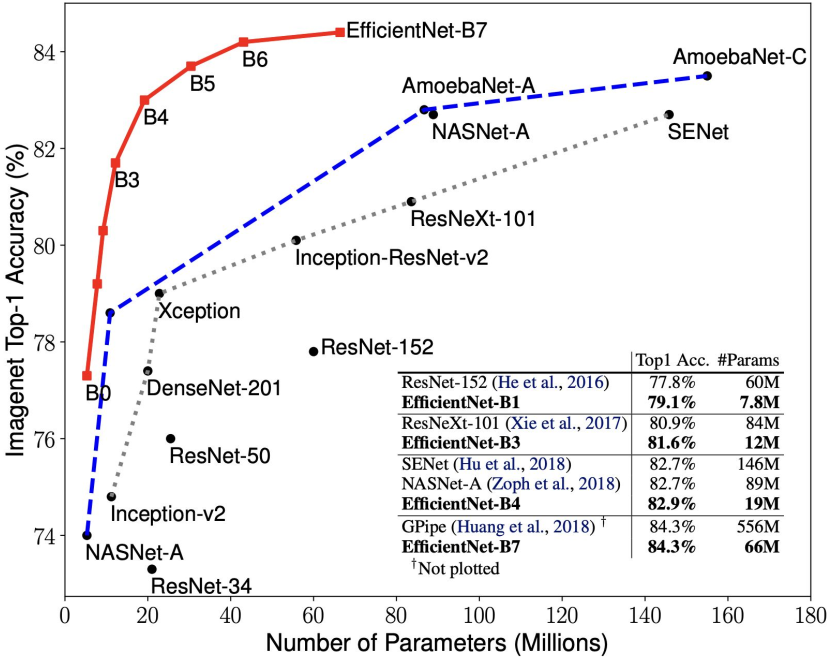


Zoph et al., 2017

EfficientNet

- Based on **MNAS** (mobile NAS)
- Finding optimal scaling of neural network **width**, **depth** and **image resolution** for different FLOPS budget

<table>
<tr><td colspan="3">EfficientNet-B0</td></tr>
<tr><td>Top-1 acc</td><td>Top-5 acc</td><td>#params</td></tr>
<tr><td>77.7</td><td>93.5</td><td>5.3M</td></tr>
</table>



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Tan and Le, 2019

Non-convolutional architectures

- Current **SOTA** (state-of-the-art) models are non-convolutional architectures
- Today's best models are usually **transformers** (to be discussed later)
- Convolutions are still frequently used
- CNNs are still popular, being faster and easier to train

What is next?

- **January, 2022** – new promising convolutional architecture, “A ConvNet for the 2020s”
- Main idea: mimic internal structure of transformer

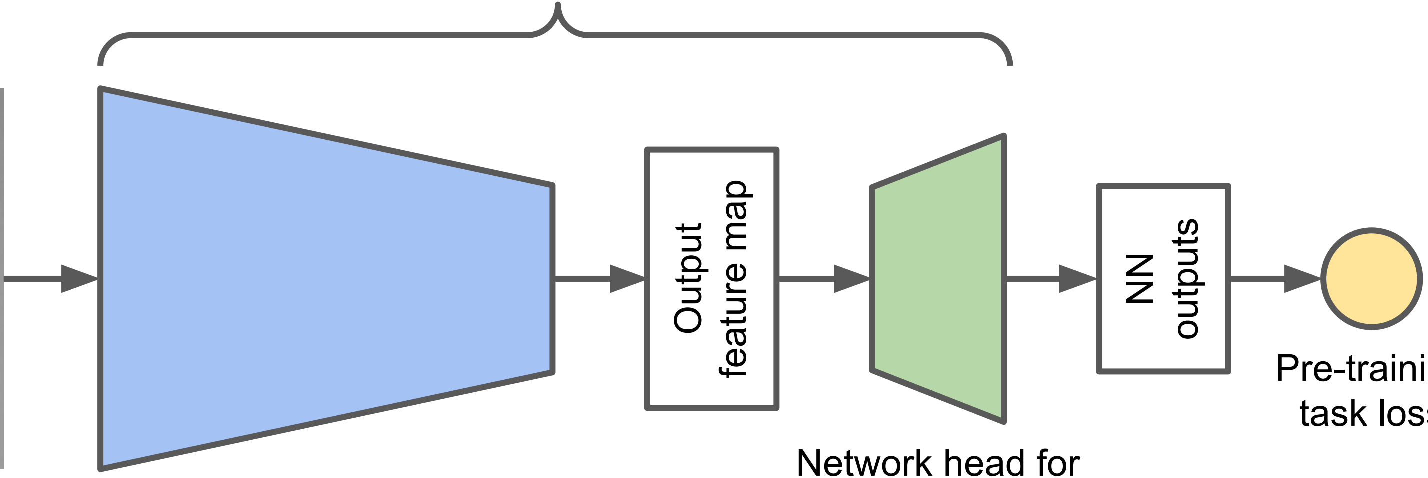
ConvNeXt-Base		
Top-1 acc	**Top-5 acc**	**#params**
84.0	96.9	88.6M

Liu et al., 2022

How to deal with small datasets?

Transfer learning: pre-training

Train both backbone and head



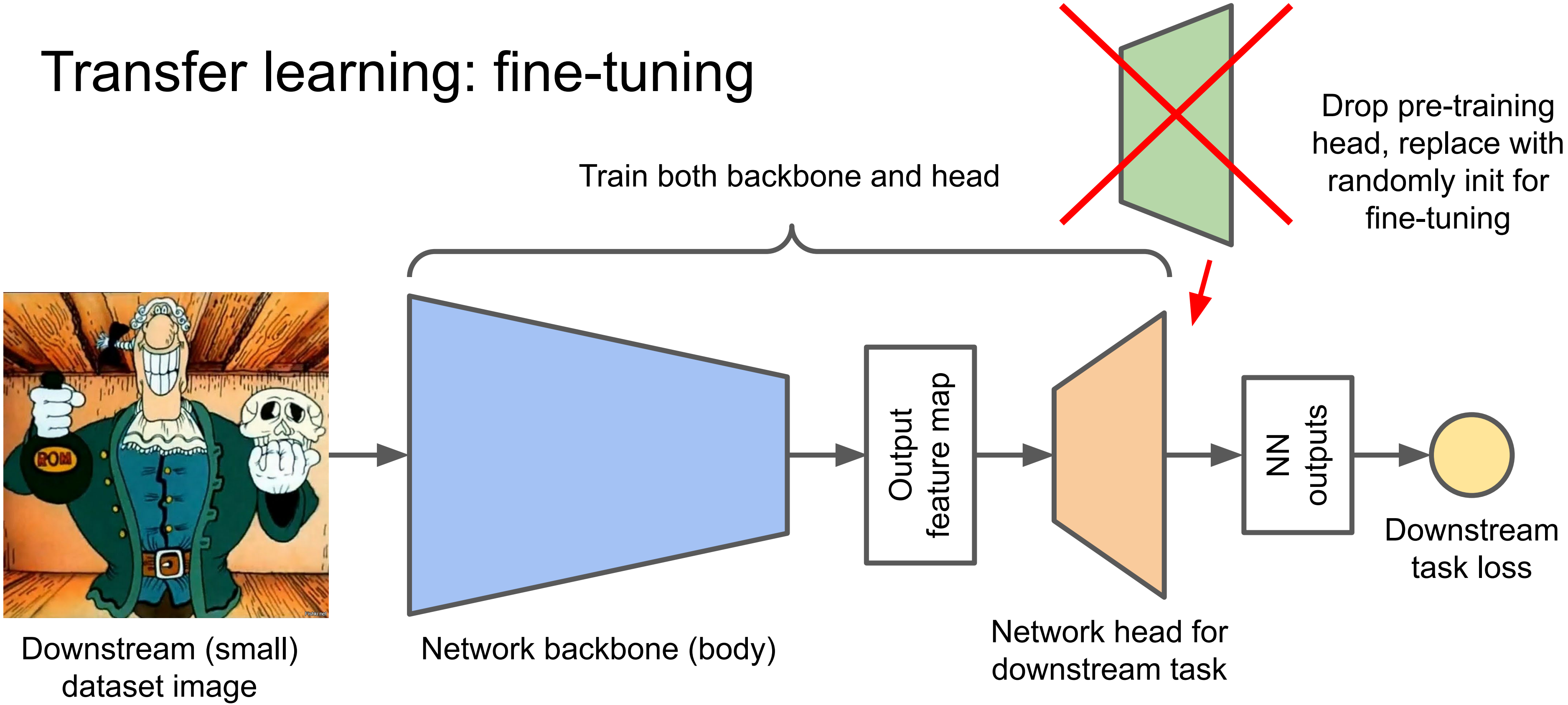
Pre-training (large) dataset image

Network backbone (body), convolutional encoder

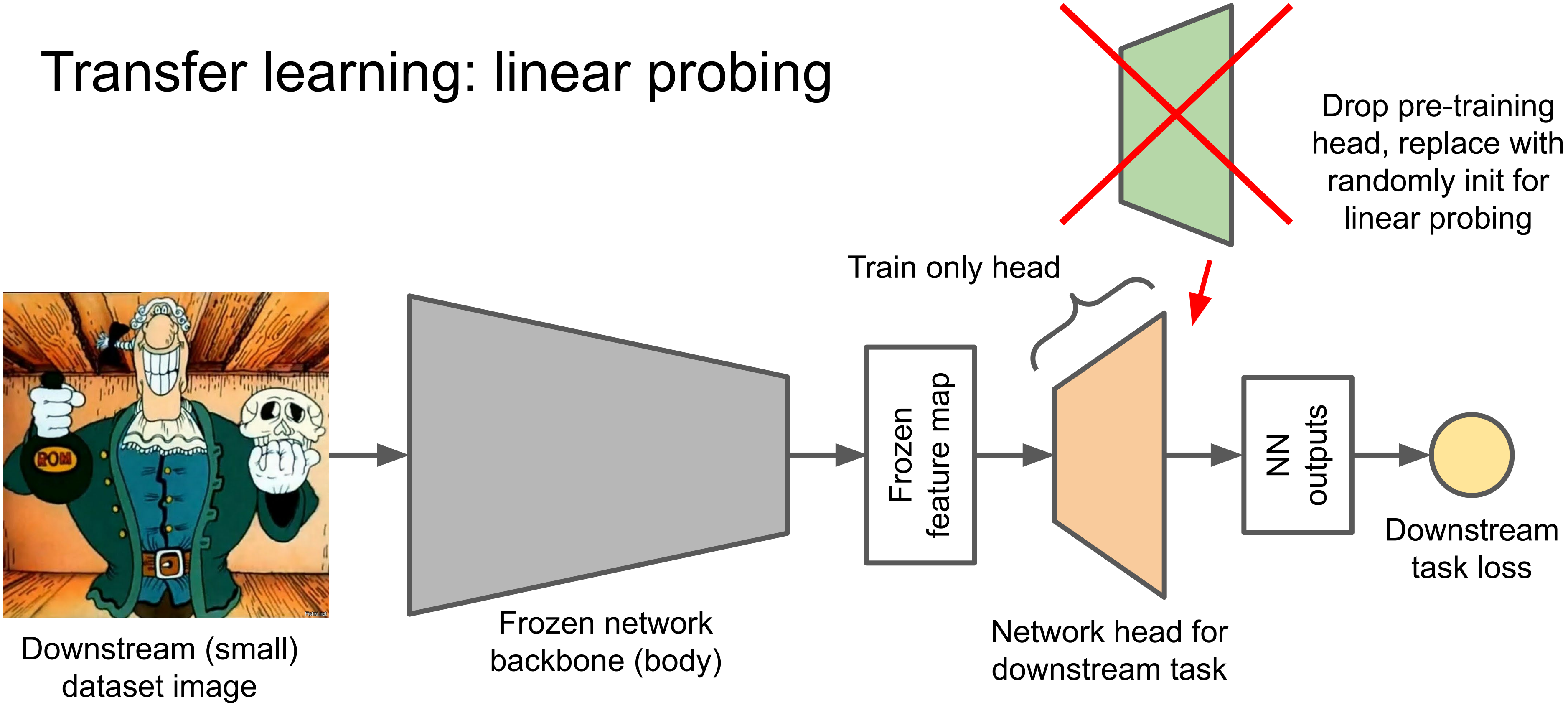
Network head for pre-training task, usually linear layer

Pre-training task loss

Transfer learning: fine-tuning



Transfer learning: linear probing



Transfer learning

- Main idea: re-use **NN weights** or **feature maps** from different datasets
- **Pre-training**: train full network on large dataset
- **Fine-tuning (FT)**: train both new head and the backbone
- **Linear probing (LP)**: train new head on top of the frozen backbone
- **LP** is faster (train only a linear model), **FT** is better (generally)
- It is possible to freeze part of backbone layers
- It is possible to pre-train without labels (self-supervised learning, to be discussed later)