Convolutional Neural Networks I

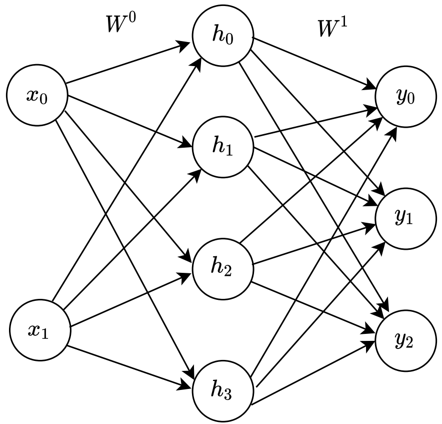
Lecture 10

Automatic Image Analysis

June 14, 2021

Technische Universität Berlin

Last lecture: a small MLP for computer vision

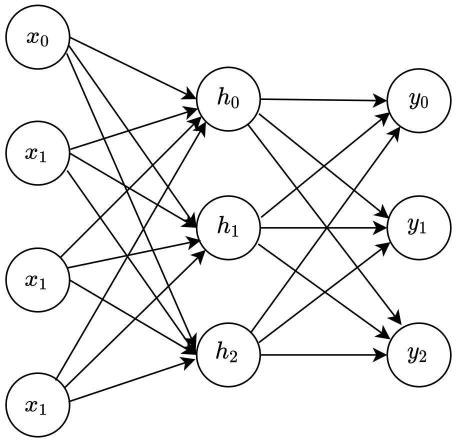


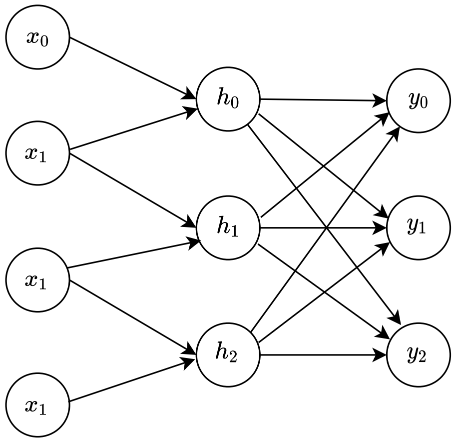
$$128 \times 128 \times 3 = 49152$$
$$16 \times 16 \times 36 = 9216$$
$$\rightarrow 49152 \cdot 9216 + 9216 \cdot 10 = 453076992 \approx 450 \cdot 10^6$$

- A small MLP with feature vector comparable to HOG.
- Input: an rgb image relatively low resolution
  $\rightarrow 128 \times 128 \times 3 = 49152$
- Hidden layer: comparable to HOG with 36 dim feature vector computed from 8×8 patches
  $\rightarrow 16 \times 16 \times 36 = 9216$
- Output neurons for e.g. 10 object classes
  $\rightarrow 49152 \cdot 9216 + 9216 \cdot 10 = 453076992 \approx 450 \textit{million}$ parameters

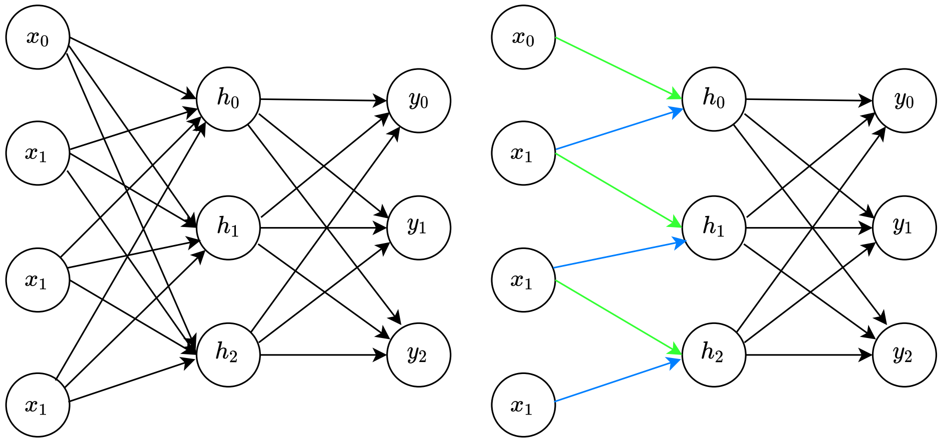
- Can we use knowledge about image statistics to reduce the number of connections?

Do we need to connect all the pixels?

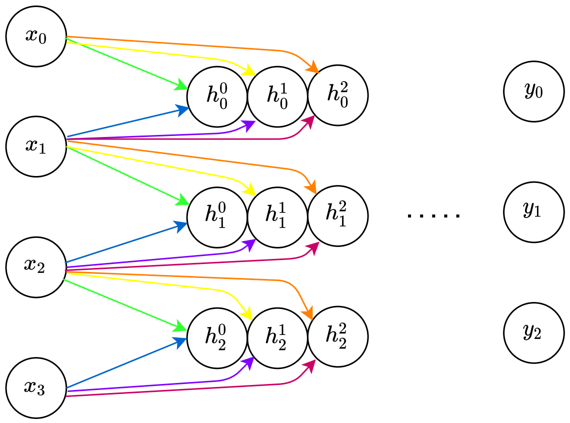




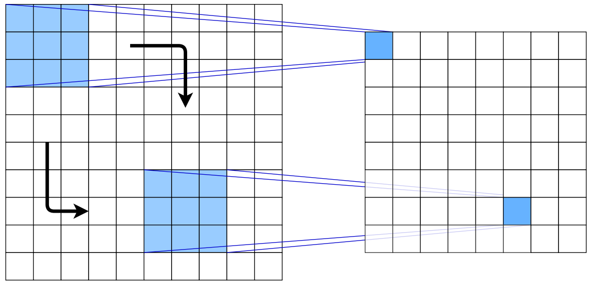
- Assumption: local regions to be processed together, regions far apart not related



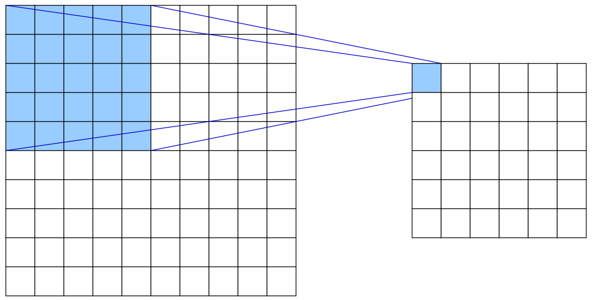
- Assumption: image processing should not vary with image region.



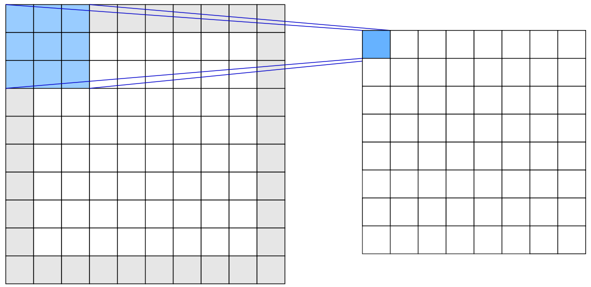
- Instead we can connect multiple neurons to every dimension of the input.



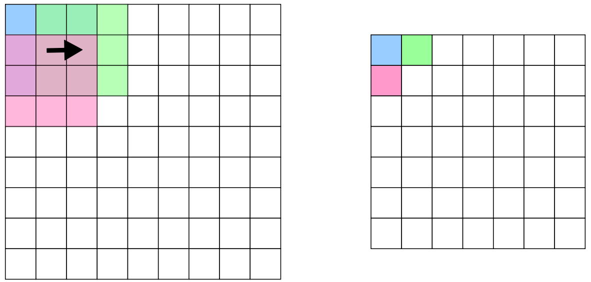
- A convolutional layer corresponds to a convolution with a filter kernel plus non linearity.



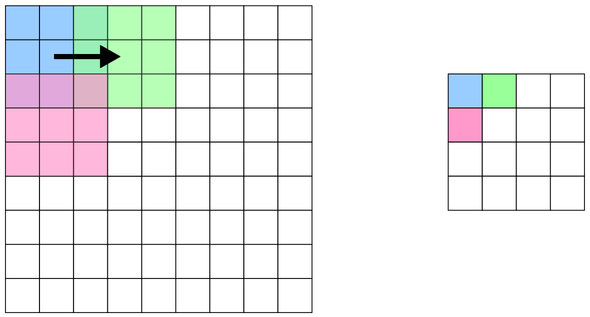
- We loose $\frac{1}{2}$ kernel size pixels at the image boarder.



- Usually we pad image boarder to keep image size.

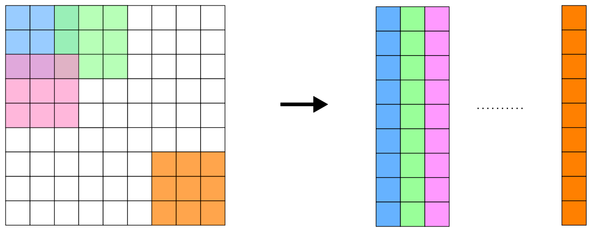


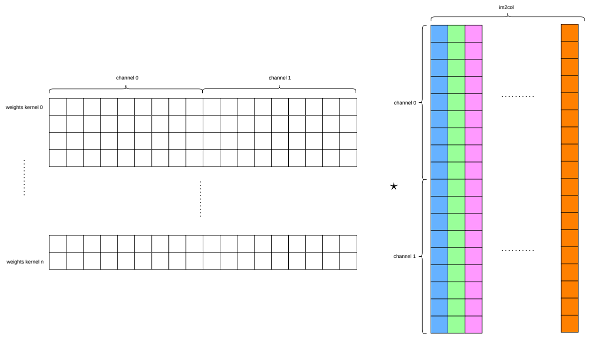
- The number of pixels in between neurons is called stride.



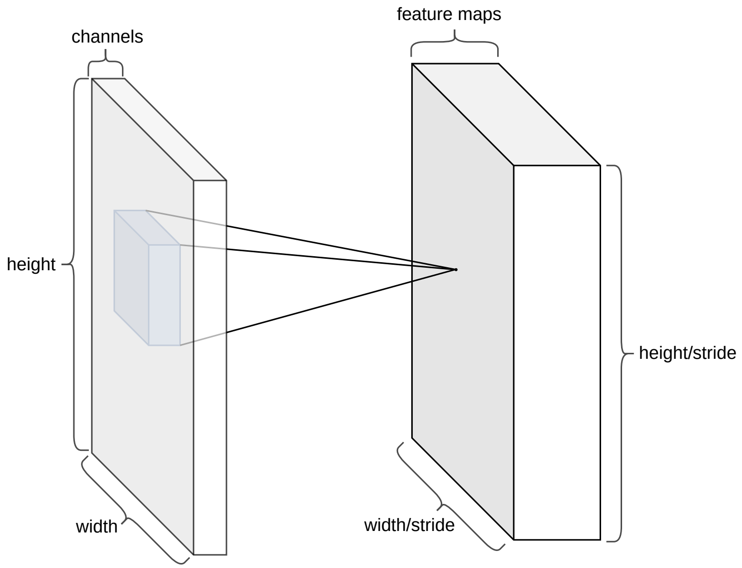
- With strides > 1 we can downsample the image to lower resolution.

- To formulate the convolutions as matrix operations, the image pixels are duplicated and rearranged.

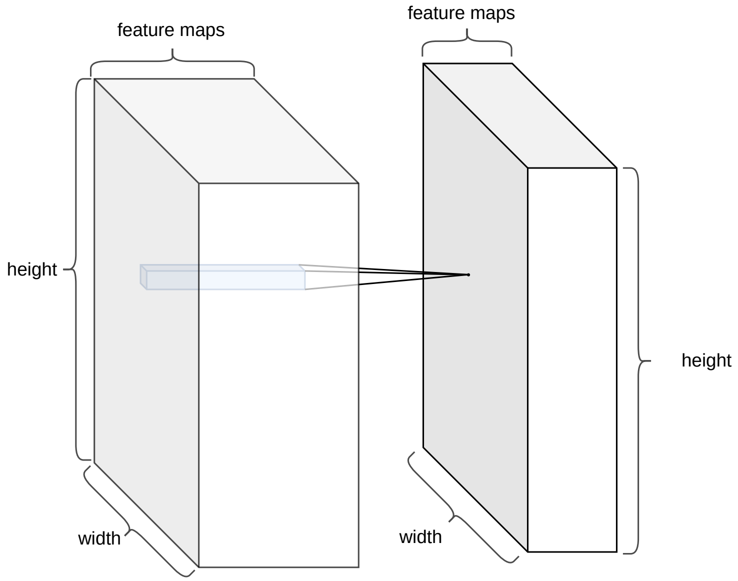




- Afterwards we multiply with a matrix that consists of the kernel values.
  $\rightarrow$ The computation of a forward/backward pass of convolutional layer becomes one big matrix operation.

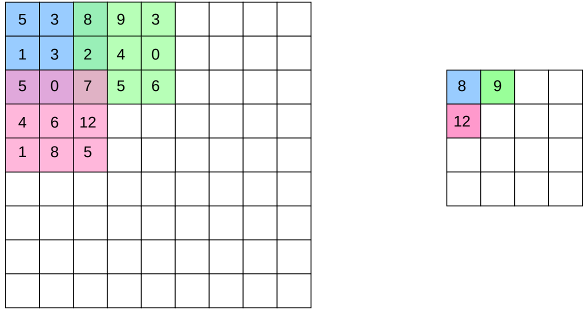


- The input to a convolutional layer is a tensor with $width \times height \times channels$
- The kernel is a four dimensional tensor with $nk \times ks \times ks \times c$, with number of kernels nk, the kernel size ks, and the number of channels c.
- The output is again a tensor $width` \times height` \times nk$, where the new width and height depend on padding and strides.
- The output channels are often referred to as feature maps.

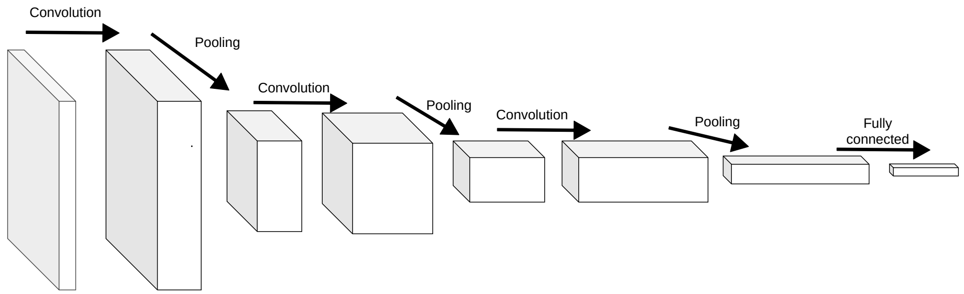


- Kernels with kernel size 1 can make sense, e.g. to reduce the number of feature maps.
- $\mathit{fmaps}' \times 1 \times 1 \times \mathit{fmaps}$ are called 1×1 convolutions.
- Network In Network, Lin et al, CVPR 2013

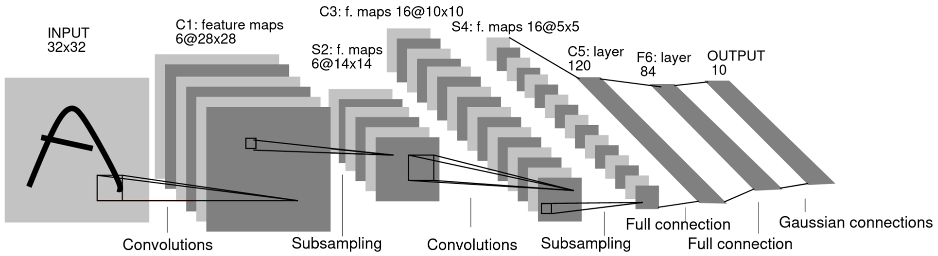
- Down-sampling with Max-Pooling with kernel size 3 and stride 2.
- Pooling is also done with the average instead of the max operation.



Summary



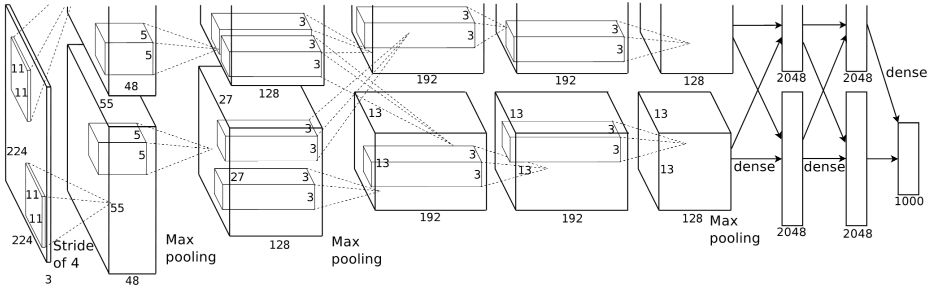
- Illustrated is the default architecture for image classification.
- Alternating convolution and pooling layers lead to constant memory footprint of activations and translation invariance.
- A fully connected final layer removes any spatial information.



- Gradient-based learning applied to document recognition, LeCun et al, 1998
- Classifies handwritten digits of the MNSIT dataset.

The big dataset: ImageNet 2009

- ImageNet is an image database organized according to the WordNet hierarchy (15 mio images).
- https://www.image-net.org/
- Widely used subset for ImageNet Large Scale Visual Recognition Challenge (ILSVRC): 1000 object classes, 1,281,167 training images, 50,000 validation images and 100,000 test images



- ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NeurIPS 2012
- Implements LeNet-like architecture on GPU (deeper and wider).
- ReLU activations, dropout regularization, max pooling.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

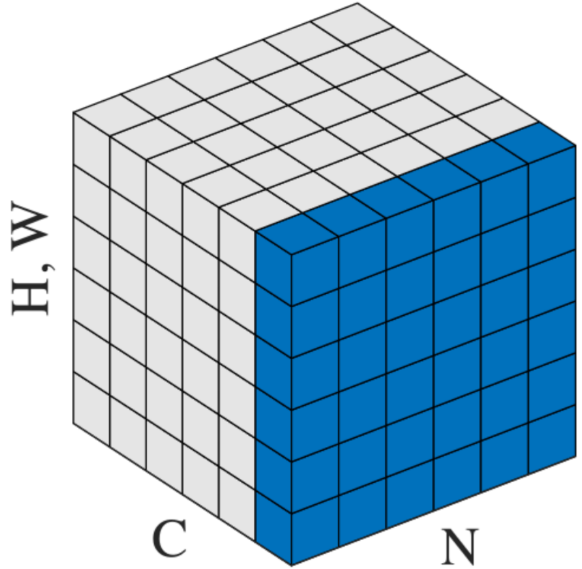
Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

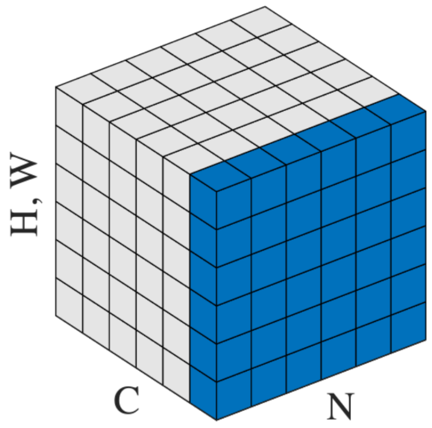
ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

- Very Deep Convolutional Networks for Large-Scale Image Recognition, Simonyan & Zisserman, ICLR 2015
- Visual Geometry Group → VGG
- Depth matters, small kernels with size 3 (less parameters, more non-linearities, same receptive field
- Still often used but really shouldn't.

- With deep networks and bounded activation functions gradients get very small.
- With unbounded activation functions gradients can explode.



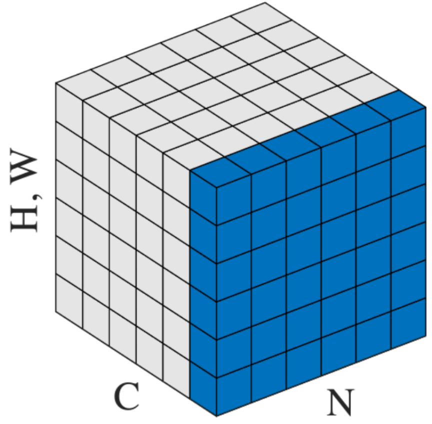
- Distribution of layer activations changes after every weight update! (Ioffe & Szegedy call this the internal covariate shift.)
- Lets normalize input to every layer!
- Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Ioffe & Szegedy, PLMR 2015
- Image from Group Normalization, Wu & He, Group normalization, 2018



$$x'_i = \frac{x_i - E[x_i]}{\sqrt{Var[x_i]}}$$

- It's as simple as the normalization of the input data. Almost ...
- What if mean and variance of activations matter?

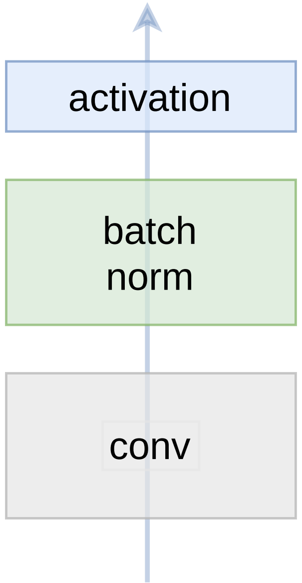
Batch normalization



$$x'_i = \frac{x_i - E[x_i]}{\sqrt{Var[x_i]}}$$

$$x'' = \gamma x' + \beta$$

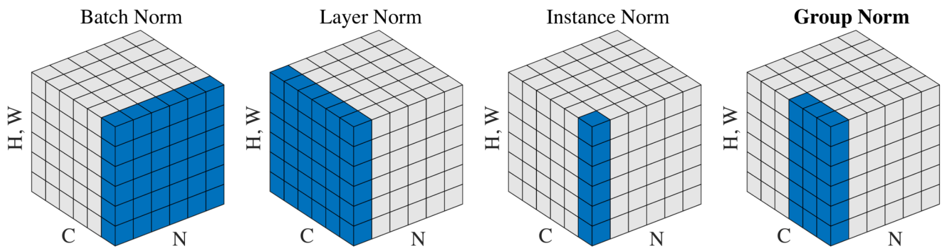
- It's as simple as the normalization of the input data. Almost ...
- What if mean and variance of activations matter?
  → Add learnable parameters to modulate mean and variance!



- Usually inserted before the non-linearity

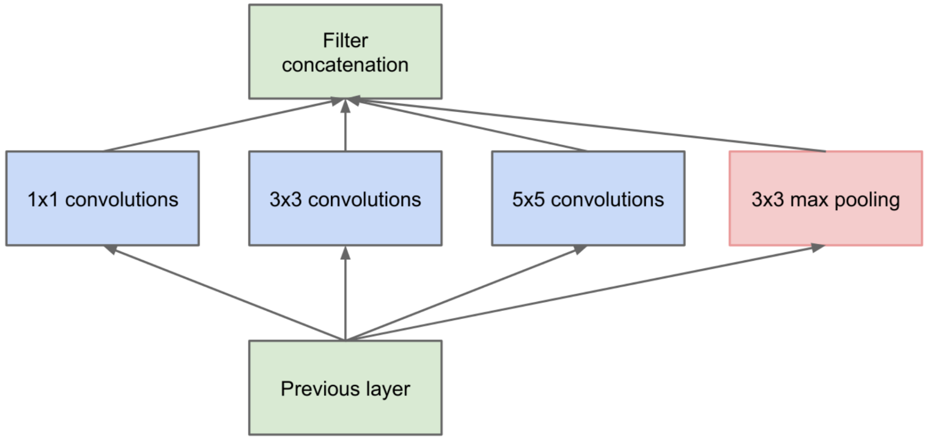
- Internal covariate shift is reduced
  $\rightarrow$ Training is more stable, higher learning rates possible
  $\rightarrow$ Contribution of samples in mini-batch to gradient harmonized
  $\rightarrow$ Input to non-linearity centered around zero
- Contribution to gradient of a sample depends on other samples in mini-batch
  $\rightarrow$ Regularization
  $\rightarrow$ In some cases detrimental

- Different behavior in training and test time
  $\rightarrow$ Often leads to bugs
- Adds a lot of complexity in recurrent networks
  $\rightarrow$ Every pass through a layer needs a dedicated batch-norm layer
- Depends on batch size (zero variance for single sample)

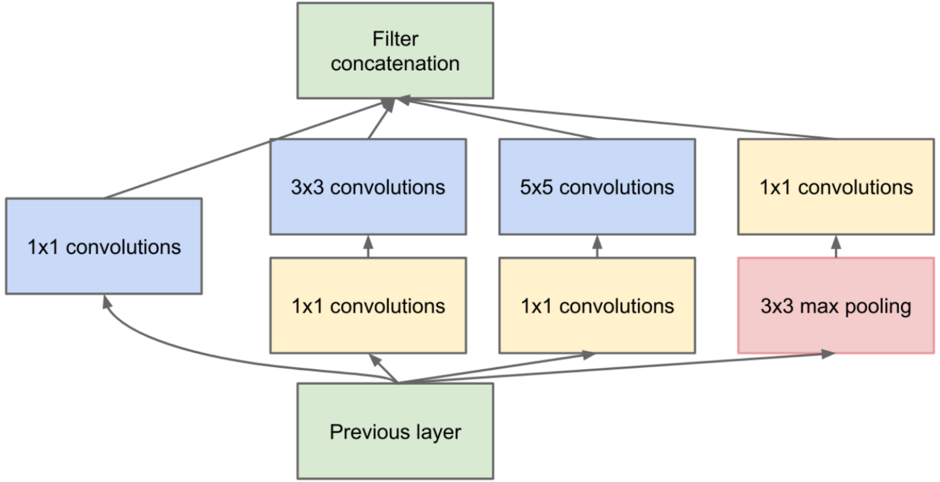


- Layer Normalization, Ba et al, 2016
- Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesi, Ulyanov et al, 2017
- Group Normalization, Wu & He, Group normalization, 2018
- Many more including combinations of these and weight normalization
- Image from Group Normalization, Wu & He, Group normalization, 2018

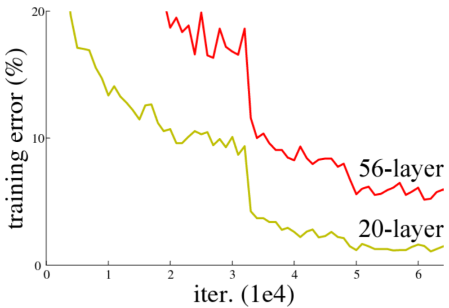
Codename Inception 2015

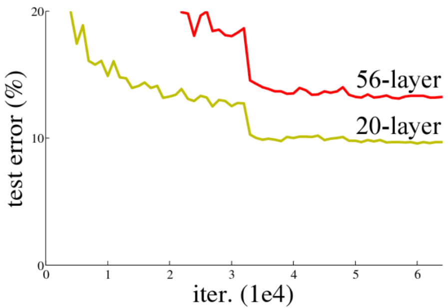


- Motivated by spatial sparsity.
- Reducing number of total weights per layer by combining filters with different sizes.
- Going Deeper with Convolutions, Szegedy et al, 2015

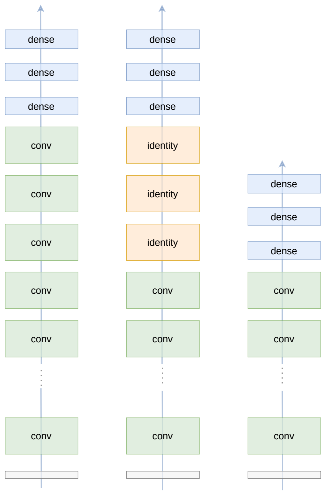


- Higher filter sizes and pooling layers still need a lot of resources.
- Use 1x1 convolutions to reduce the number of filter maps.
- 1x1 layers also have non-linearities leading to dual purpose layers.
- Going Deeper with Convolutions, Szegedy et al, 2015

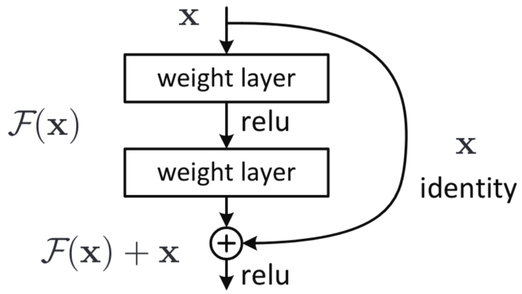




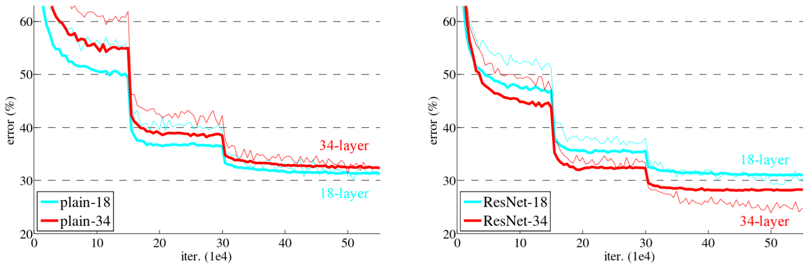
- VGG and others showed that accuracy increases with depth.
- Vanishing/exploding gradients are alleviated by normalization.
- But even with normalization, we can observe that training and test error start to increase again at a certain number of layers/depth.
- Deep Residual Learning for Image Recognition, He et al, CVPR 2016



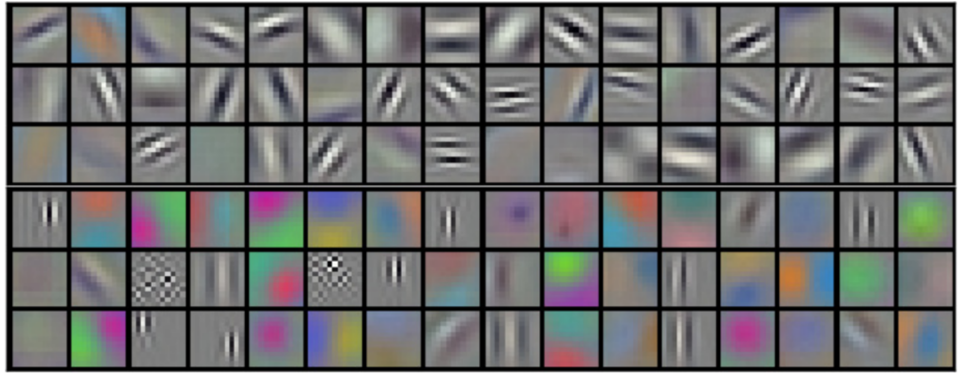
- But even with normalization, we can observe that training and test error start to increase again at a certain number of layers/depth.
- That's weird, because there is an obvious solution that is at least as good as the shallower network.
- However it seems, that finding this solution in deeper networks is more difficult.



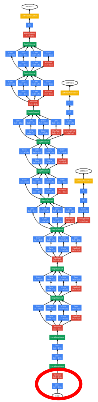
- Idea: Shortcut layers, so learning the identity is setting weights to zero, which should be easier as actually learning the identity.

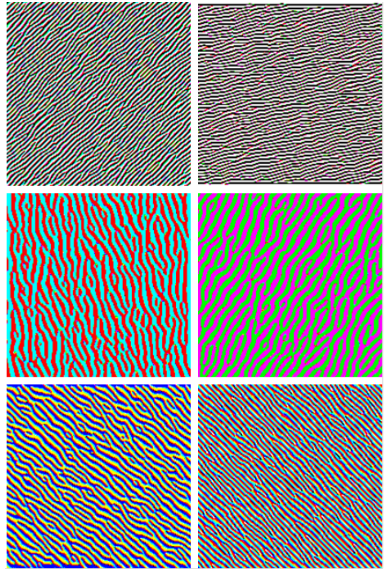


- Yay! Works even better!
- There is no limit to depth any more!
- Super human performance on ImageNet with a network with 152 layers.



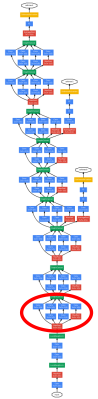
- Filters of the first convolutional layer in AlexNet.
- ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NeurIPS 2012

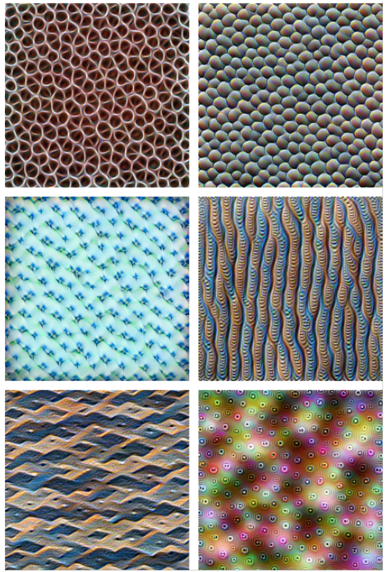




- Going deeper with convolutions, Szegedy et al, CVPR 2015
- Feature Visualization, Olah et al, https://distill.pub/2017/feature-visualization/

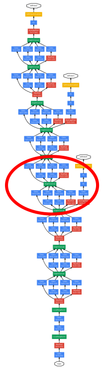
Visualization: Middle

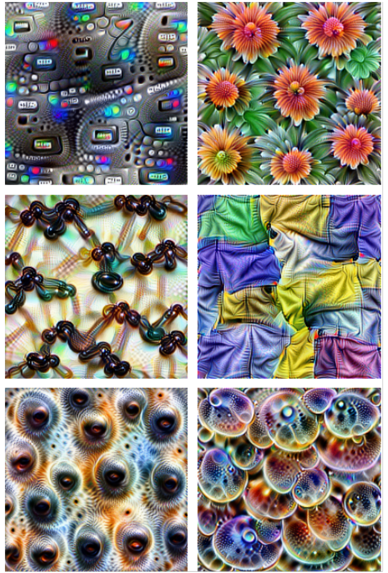




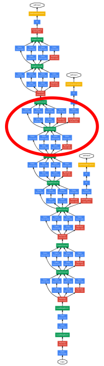
Visualization: Middle

Visualization: Middle

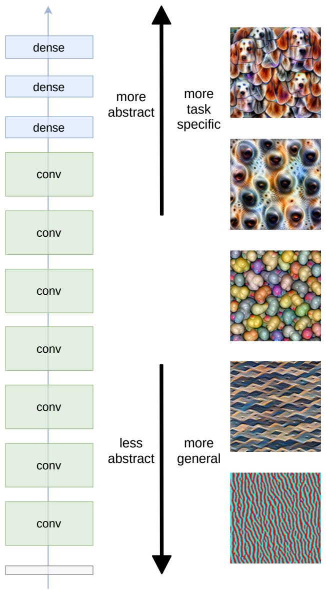




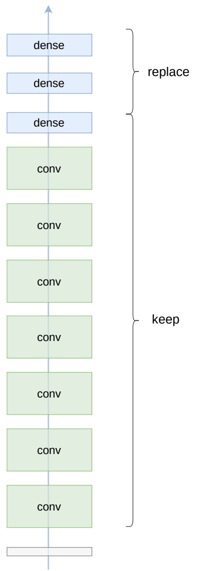
Visualization: Late



-
-

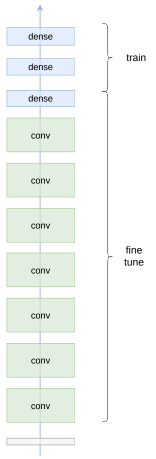


- Deep learning needs big data!
- But what if we use the more general abilities a network learned for another task?
- Images from Feature Visualization, Olah et al, https://distill.pub/2017/feature-visualization/



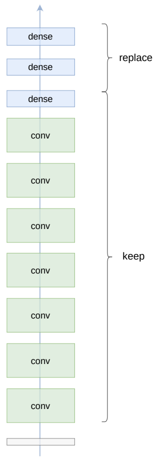
- We can transfer knowledge by replacing the specialized layers with randomly initialized layers and only train those!

Transfer Learning: fine tuning



- 1. Train the randomly initialized layers to convergence.
- 2. Unfreeze the some of the upper layers and continue training.

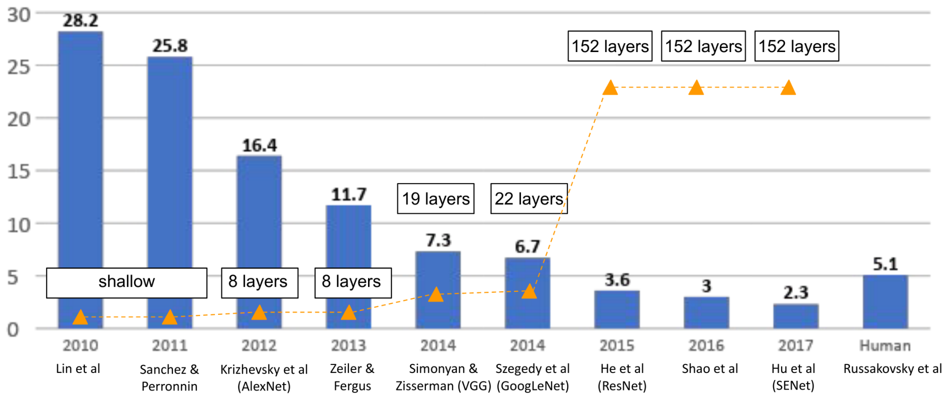
- Randomly initialized layers generate high gradients, which would destroy what was learned in the layers below.
- Fine tune with very small learning rate ($\approx .1\times$ original lr)



- The more data we have for the target domain
  $\rightarrow$ the more layers we can replace.
  $\rightarrow$ the more layers we can fine tune.
- The higher the distance between original and target task
  $\rightarrow$ the more layers we may want to replace.
  $\rightarrow$ the more layers we need to fine tune.

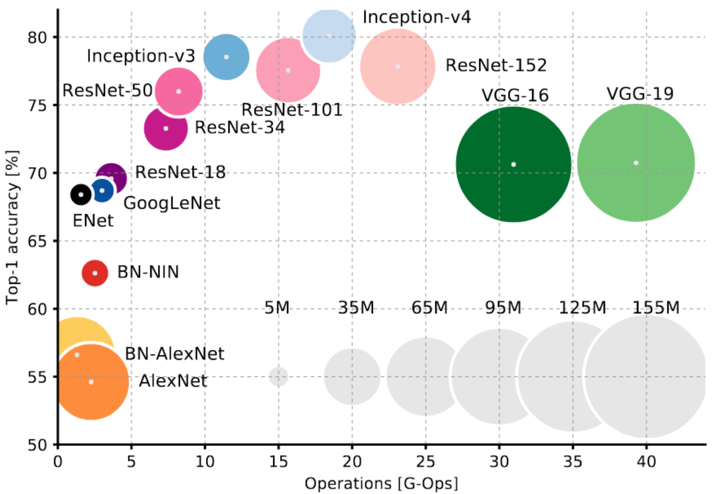
- Give it a try, it works surprisingly well.
- Transfer learning has become the default initialization.
  (In many frameworks, it's just an argument in a function call to initialize with weights trained on ImageNet.)
- Recent results show, that it is not always necessary.
  (Rethinking ImageNet Pre-training, He et al, ICCV 2019)

Winners ImageNet Large Scale Visual Recognition Challenge



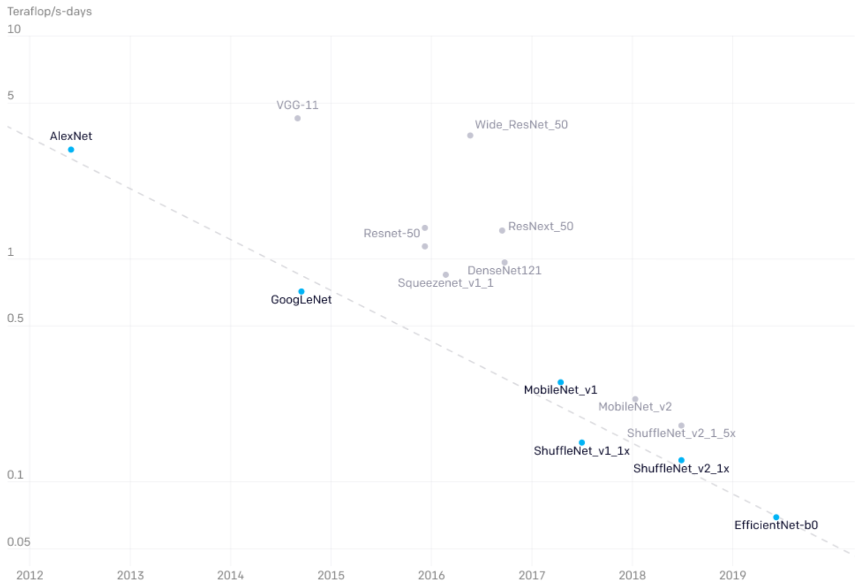
- Image from Stanford CS231n Lecture 9, Fei-Fei Li
  http://cs231n.stanford.edu/slides/2021/lecture_9.pdf

Accuracy ImageNet Ops/Params



- Image from An Analysis of Deep Neural Network Models for Practical Applications, Canziani et al, 2017

Efficiency



- Total amount of compute in teraflops/s-days used to train to AlexNet level performance. Lowest compute points at any given time shown in blue, all points measured shown in gray.
- Image from https://openai.com/blog/ai-and-efficiency/