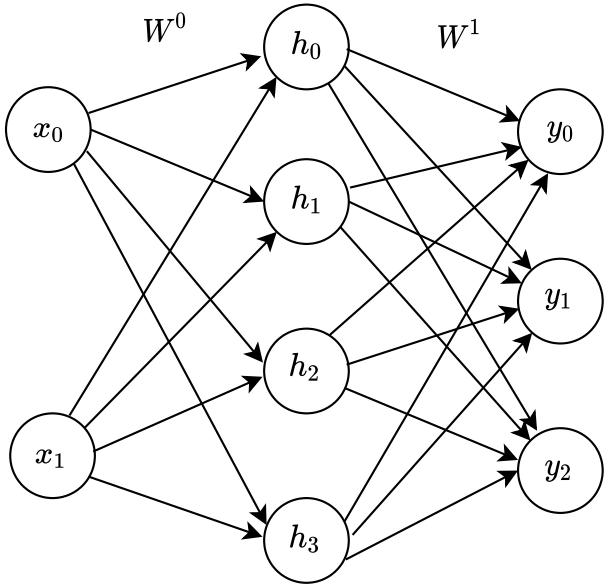
Neural Networks 2/3

Lecture 11

Computer Vision for Geosciences

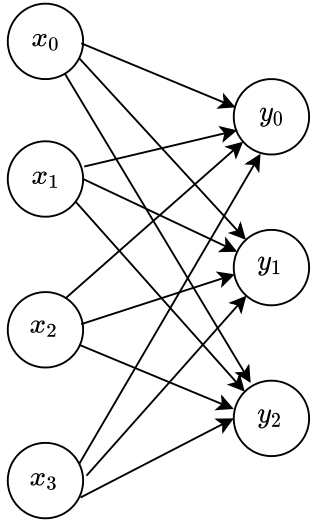
May 28, 2021

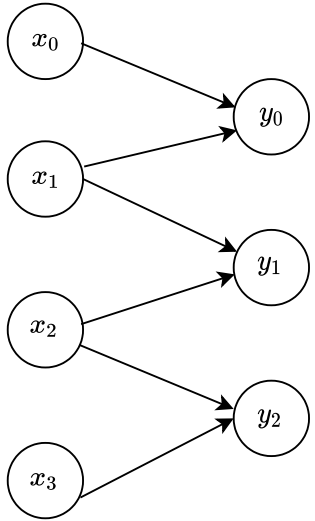


- Input: an rgb image relativly low resolution
  $\rightarrow 128 \times 128 \times 3 = 49152$
- Hidden layer: comparable to HOG with 36 dim feature vector computed from 8x8 patches
  $\rightarrow 16 \times 16 \times 36 = 9216$
- Output neurons for e.g. 10 object classes
  $\rightarrow 49152 \cdot 9216 + 9216 \cdot 10 = 453076992 \approx 450 \textit{million}$ parameters

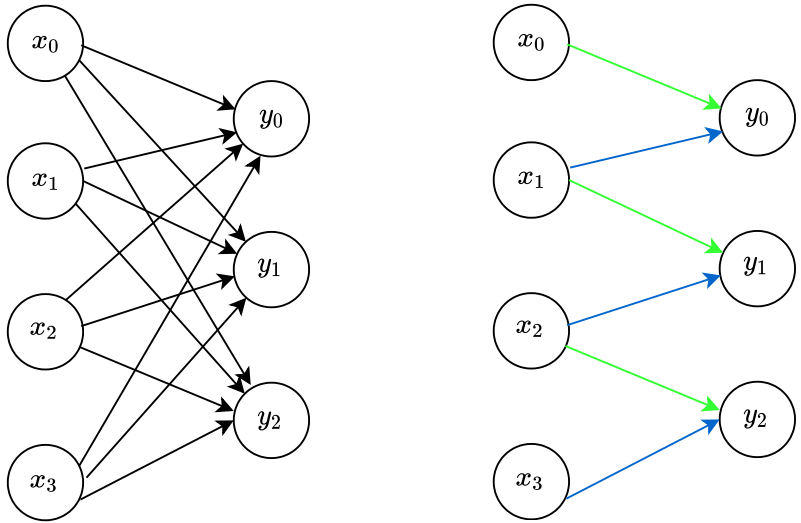
Do we need to connect all the pixels?

- Too many weights.
- Local image statistics.
- Weight sharing.

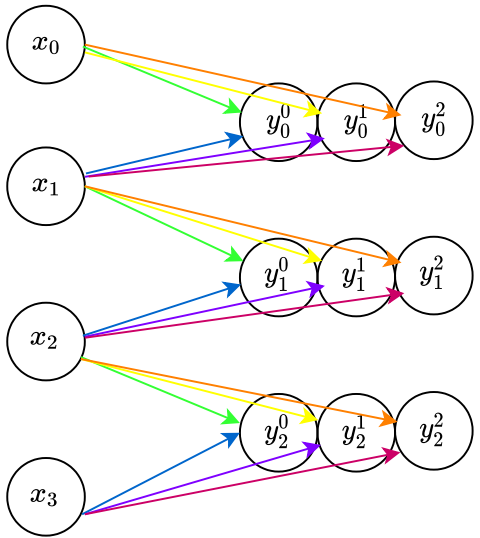




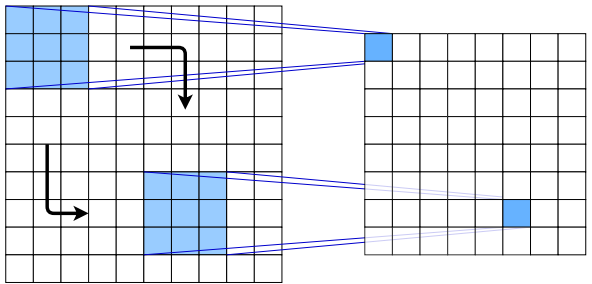
- Assumption: local regions to be processed together, regions far apart not related



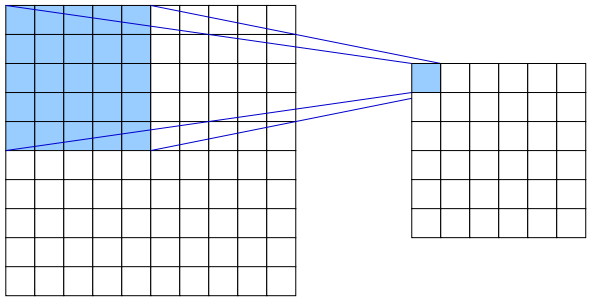
- Assumption: image processing should not vary with image region.



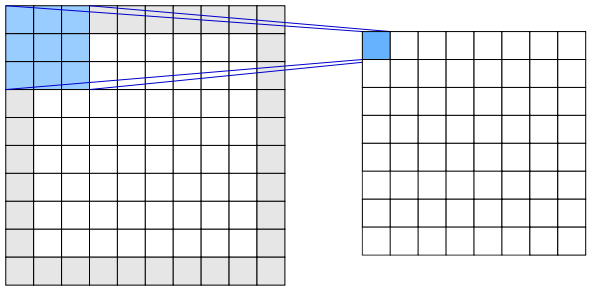
- Instead we can connect multiple neurons to every dimension of the input.



- The perceptron operation $\sigma(Wx)$ corresponds to a convolution with a filter kernel plus non linearity.

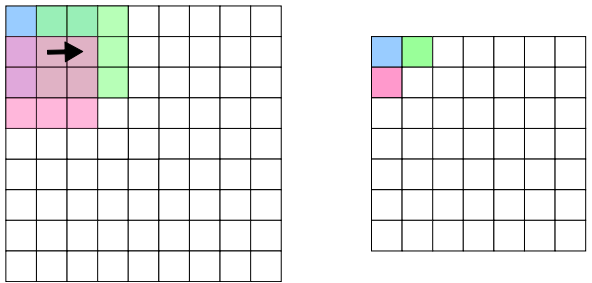


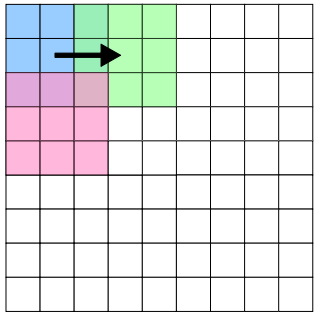
- We loose $\frac{1}{2}$ kernel size pixels at the image boarder.

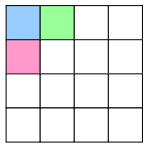


- Usually we pad image boarder to keep image size.

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

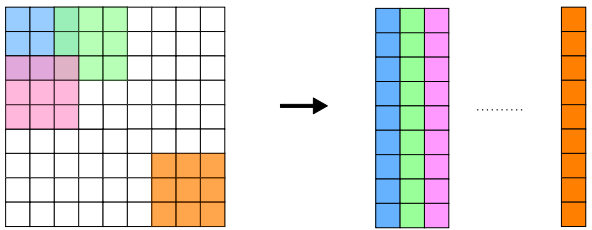


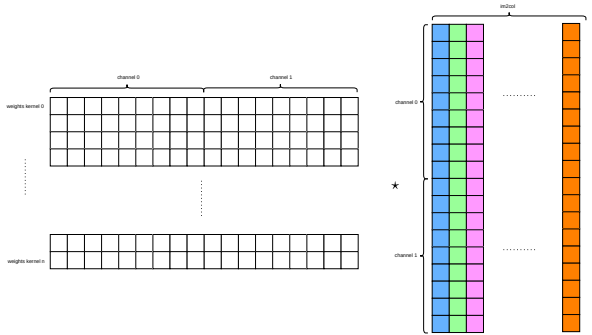




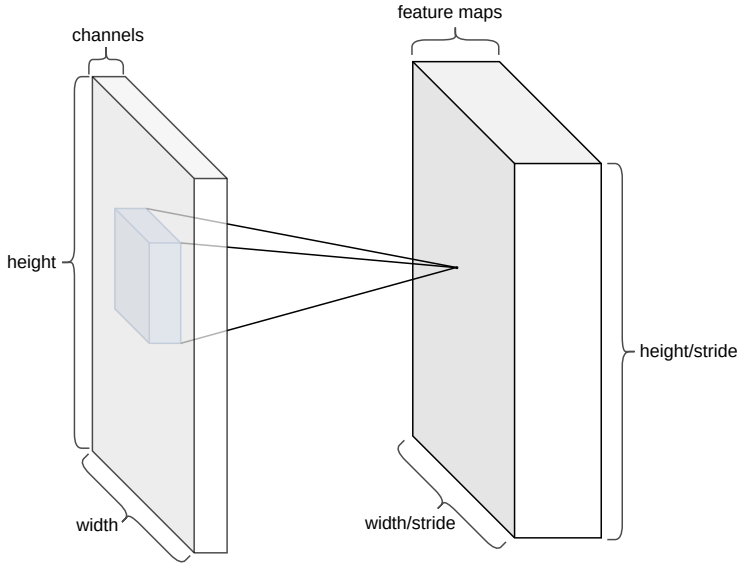
- With strides > 1 we can downsample the image to lower resolution.

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

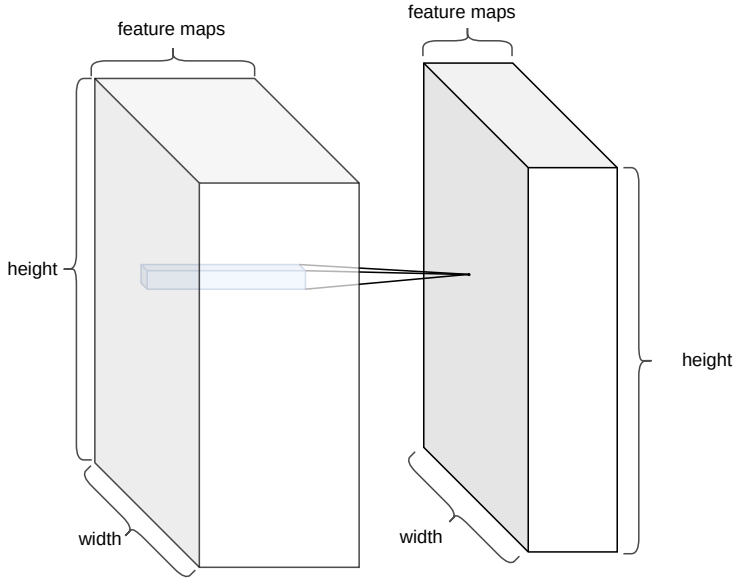




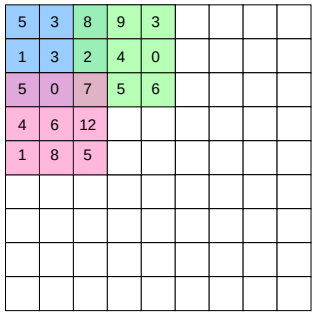
- Afterwards we multiply with a matrix that consists of the kernel values.
  → The computation of a forward/backward pass of convolutional layer becomes one big matrix operation.

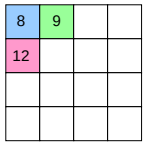


- The input to a convolutional layer is a tensor with $width \times height \times channels$
- The kernel is a four dimensional tensor with $nk \times ks \times ks \times c$,
  with number of kernels nk, the kernel size ks, and the number of channels c.
- The output is again a tensor $width' \times height'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

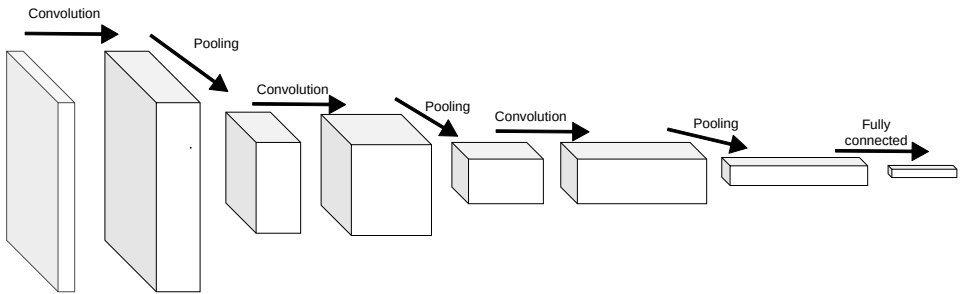


- Kernels with kernel size 1 can make sense, e.g. to reduce the number of feature maps.
- $fmaps' \times 1 \times 1 \times fmaps$ are called 1×1 convolutions.

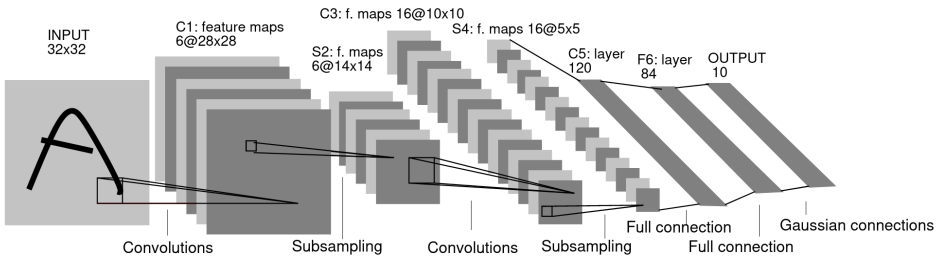

5 3 8 9 3
1 3 2 4 0
5 0 7 5 6
4 6 12
1 8 5


8 9
12

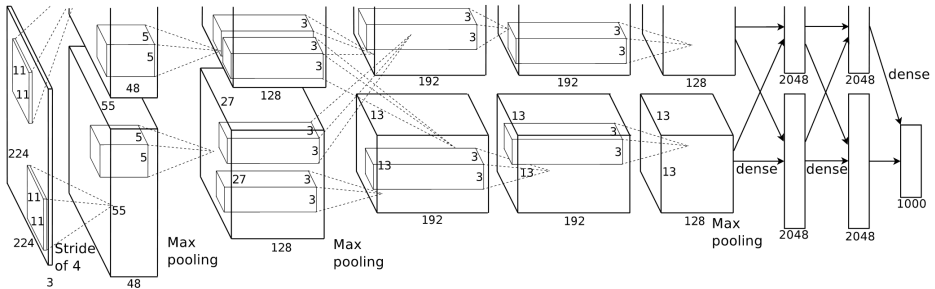
Summary


Convolution
Pooling
Convolution
Pooling
Convolution
Pooling
Fully
connected

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



- 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



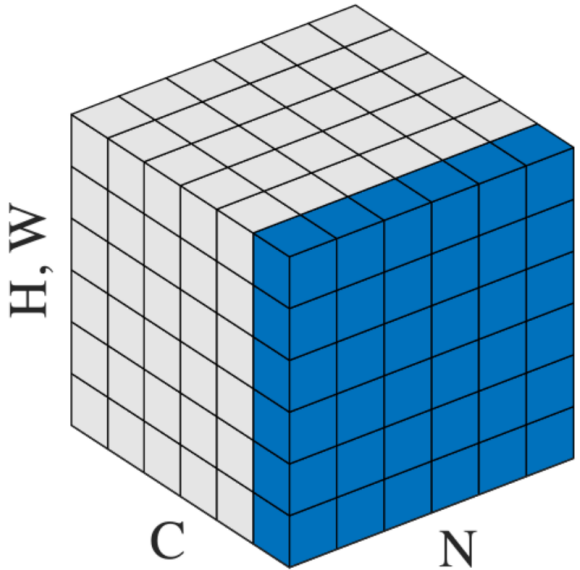
- ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NeurIPS 2012
- Implements LeNet-like architecture on GPU (deeper and wider).
- ReLU activations, dropout regularization, max pooling.

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					

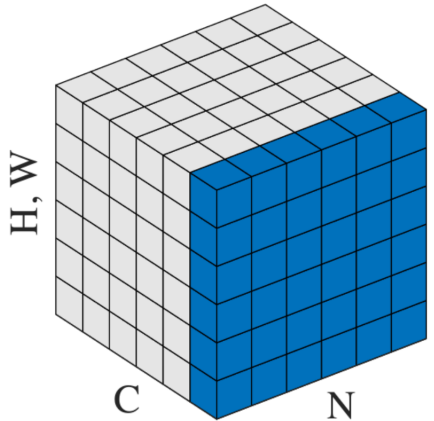
Table 2: **Number of parameters** (in millions)

- Very Deep Convolutional Networks for Large-Scale Image Recognition, Simonyan & Zisserman, ICLR 2015
- Visual Geometry Group $\rightarrow$ VGG
- Depth matters, small kernels with size 3
- 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.

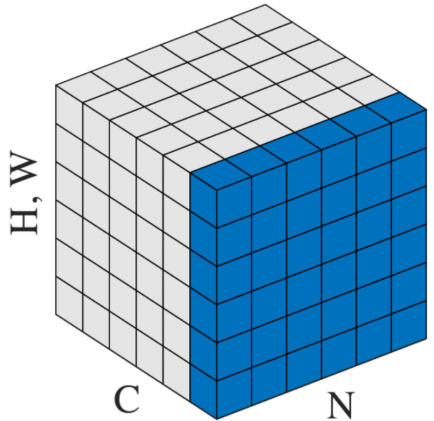


- Distribution of layer activations changes after every weight update!
  (Ioffe & Szegedy call this the internal covariate shift.)
- Lets normalize input to every layer!
- Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Ioffe & Szegedy, PLMR 2015
- Image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



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

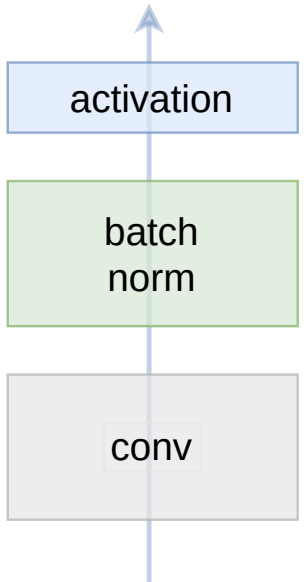
- It's as simple as the normalization of the input data. Almost ...
- What if mean and variance of activations matter?



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

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

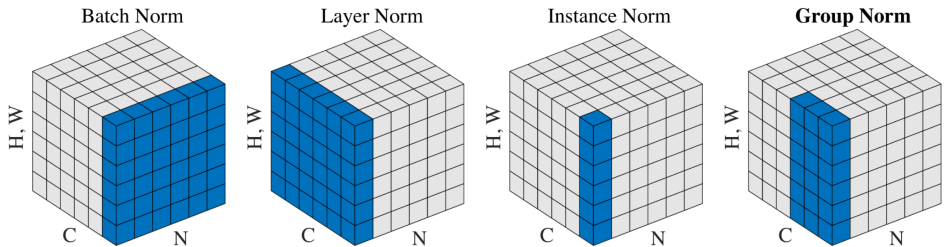
- It's as simple as the normalization of the input data. Almost ...
- What if mean and variance of activations matter?
  $\rightarrow$ Add learnable parameters to modulate mean and variance!



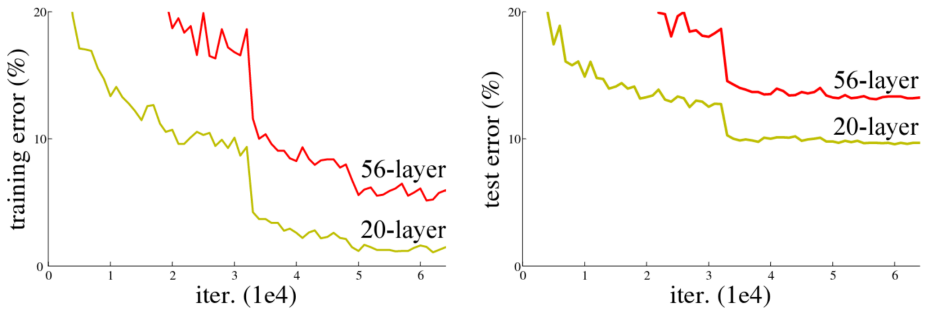
- Usually inserted before the non-linearity

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

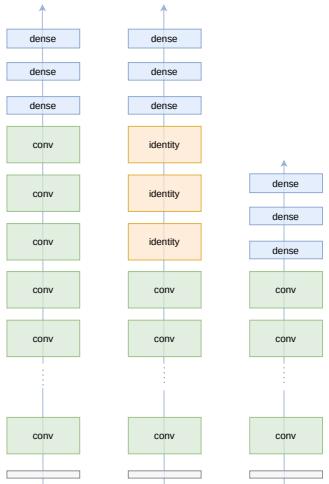
- Different behavior in training and test time
  $\rightarrow$ Often leads to bugs
- Adds a lot of complexity in recurrent networks
  $\rightarrow$ Every pass through a layer needs a dedicated batch-norm layer
- Depends on batch size (zero variance for single sample)



- 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

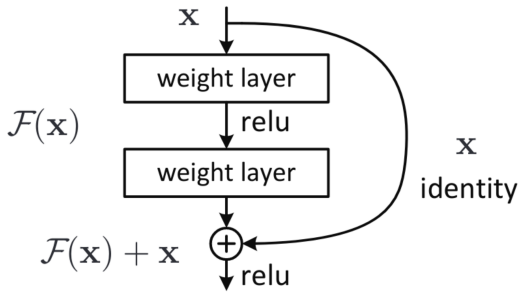


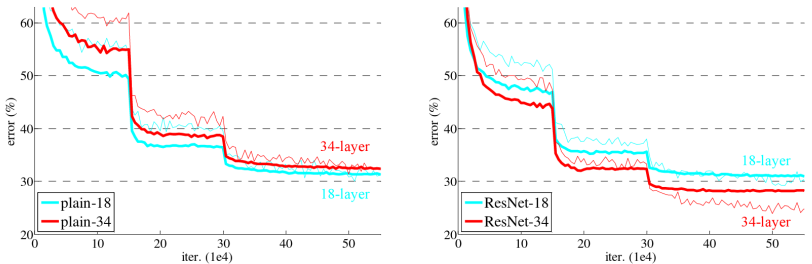
- VGG and others showed that accuracy increases with depth.
- Vanishing/exploding gradients are alleviated by normalization.
- But even with normalization, we can observe that training and test error start to increase again at a certain number of layers/depth.
- Deep Residual Learning for Image Recognition,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



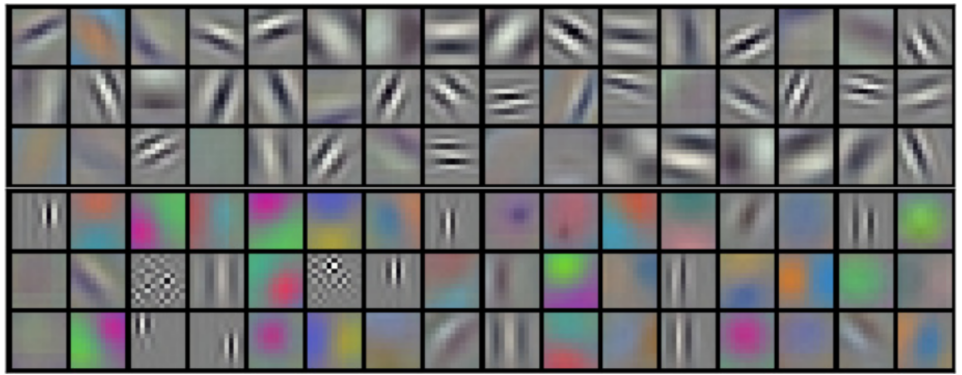
- But even with normalization, we can observe that training and test error start to increase again at a certain number of layers/depth.
- That's weird, because there is an obvious solution that is at least as good as the shallower network.
- However it seems, that finding this solution in deeper networks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

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

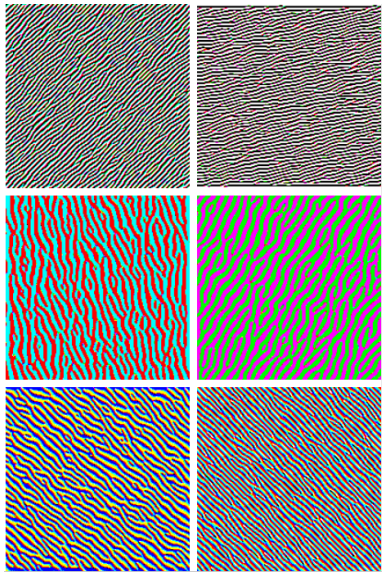




- Yay! Works even better!
- There is no limit to depth any more!
- Super human performance on ImageNet with a network with 152 layers.

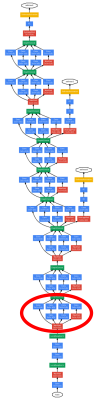


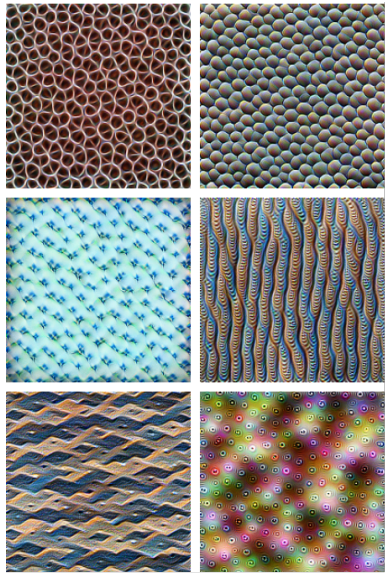
-
- ImageNet Classification with Deep ConvolutionalNeural Networks, Krizhevsky et al, NeurIPS 2012



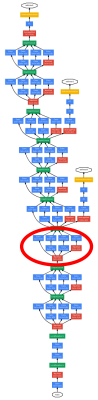
- Going deeper with convolutions, Szegedy et al, CVPR 2015
- Feature Visualization, Olah et al, https://distill.pub/2017/feature-visualization/

Visualization: Middle



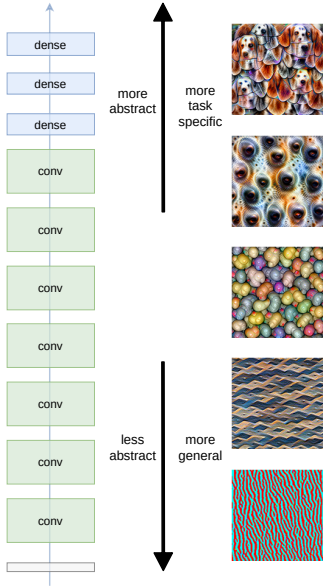


Visualization: Middle

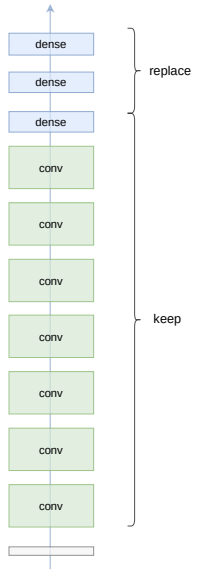


Visualization: Middle

Visualization: Late

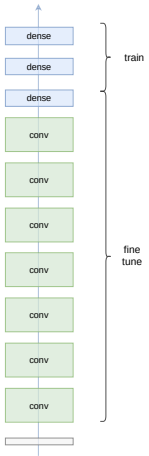


- Deep learning needs big data
- But what if we use the more general abilities a network learned for another task?
- Images from Feature Visualization, Olah et al, https://distill.pub/2017/feature-visualization/



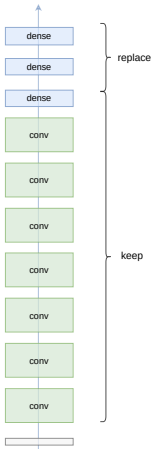
- We can transfer knowledge by replacing the specialized layers with randomly initialized layers and only train those!

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)