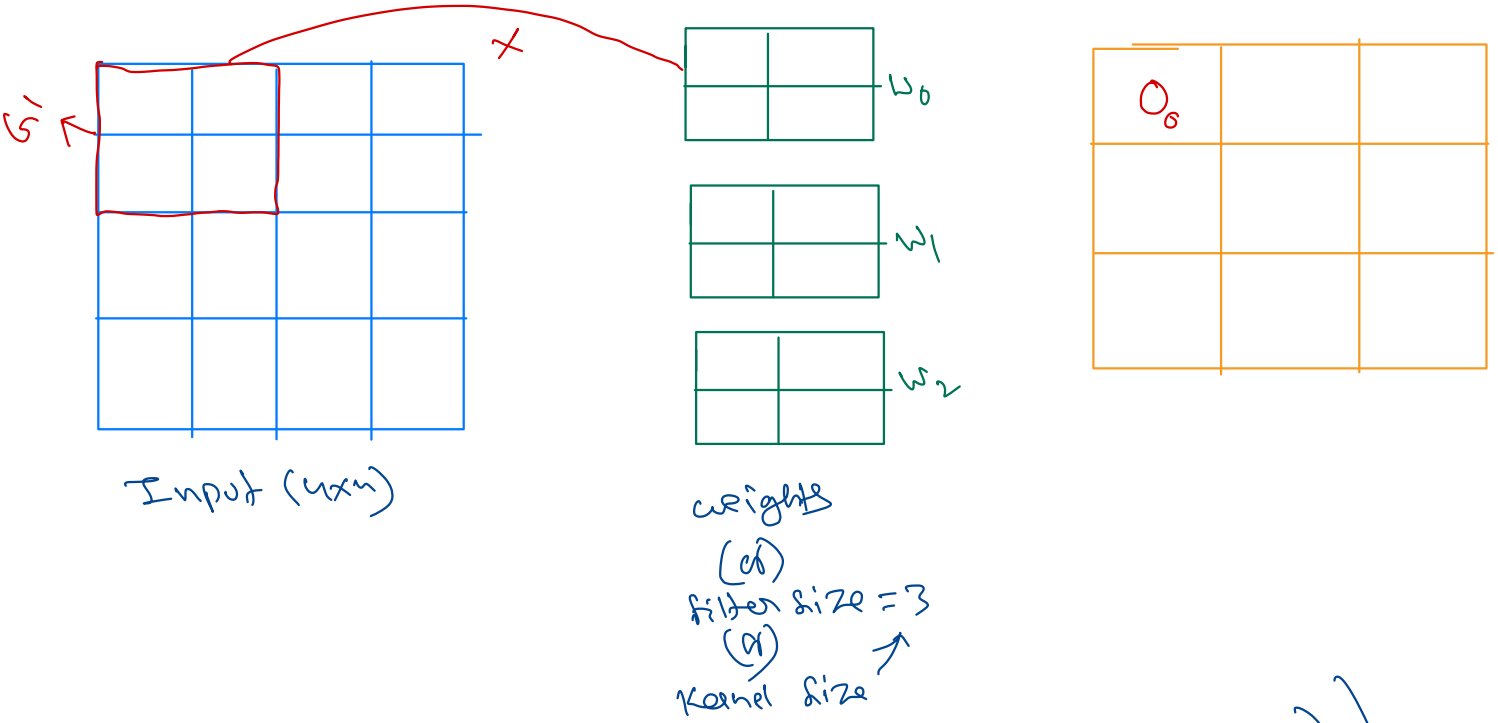
CNN's

Main Property: Sharing of weights

How do you calculate gradients when weights are shared?

Ans: Calculate individual gradients for each position then aggregate them together & update.

How do you convolve?



$O_0 = np.sum(np.multiply(G1 \times w_0))$

* All the activation maps can be computed parallely.

Convolution Neural Network

Explain CNN? Why is it called convolution?

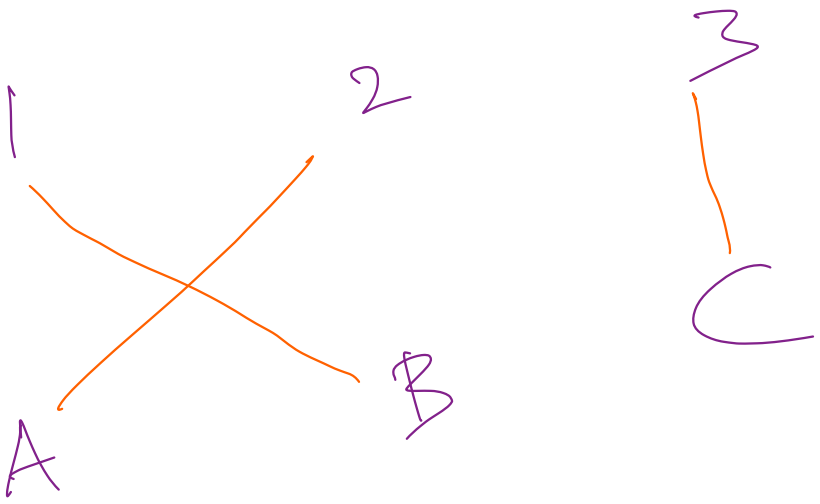
CNNs are neural networks that use convolution operation in place of general matrix multiplication.

Convolution is a special kind of linear operations.

$S(t) = (x * w)(t)$

x: input, w: kernel, *: convolution, t because kernel slides over the entire input.
S(t): output/feature map

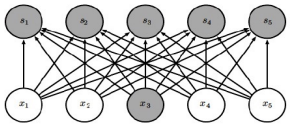
These types of networks are mainly used when the input is in grid fashion.



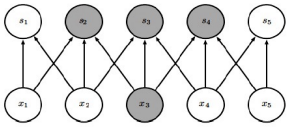
3- Important properties of CNN's :-

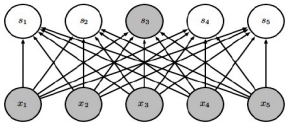
1. Local Connectivity

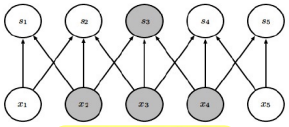
fully-connected (dense) layer



convolutional layer



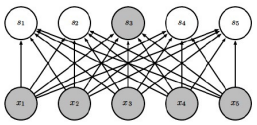




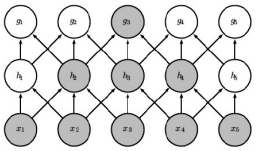
receptive field

1. Local Connectivity

1 fully-connected (dense) layer



2 convolutional layers

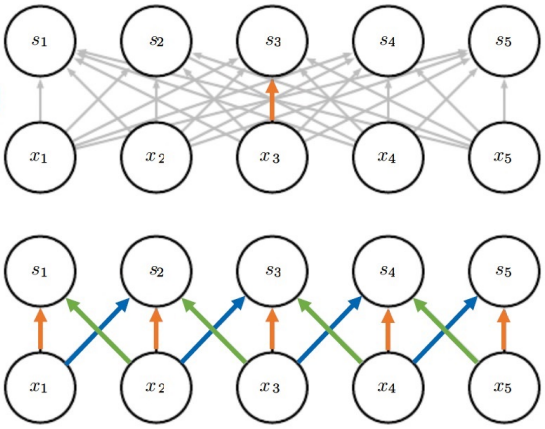


growing receptive field

2. Shared Weights

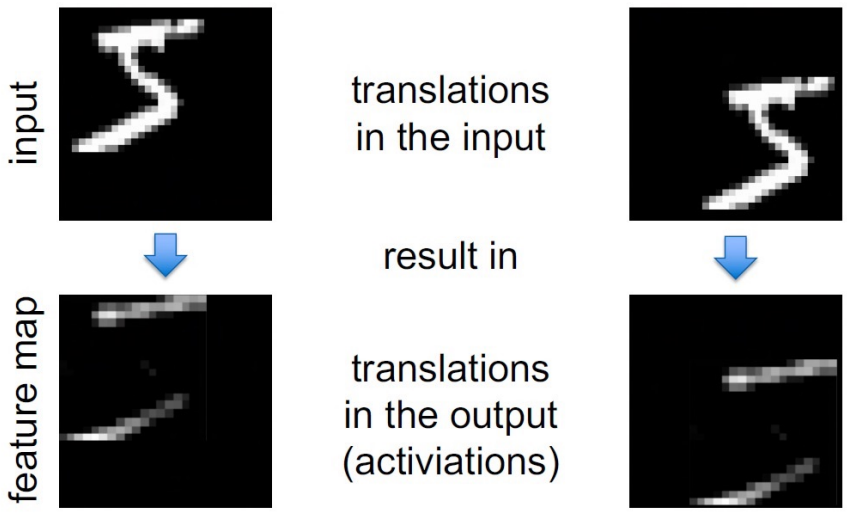
fully-connected

convolutional



3. Translation Equivariance

(If the inp changes the o/p changes in the same way).



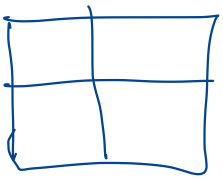
translate first and convolve (or) convolve first and translate. Both result same output.

This is NOT invariance!

You do something to your input the output doesn't change.

Filters: Since one filter weights are shared across, One filter is like a set of neurons that detect same feature at multiple positions.

e.g:

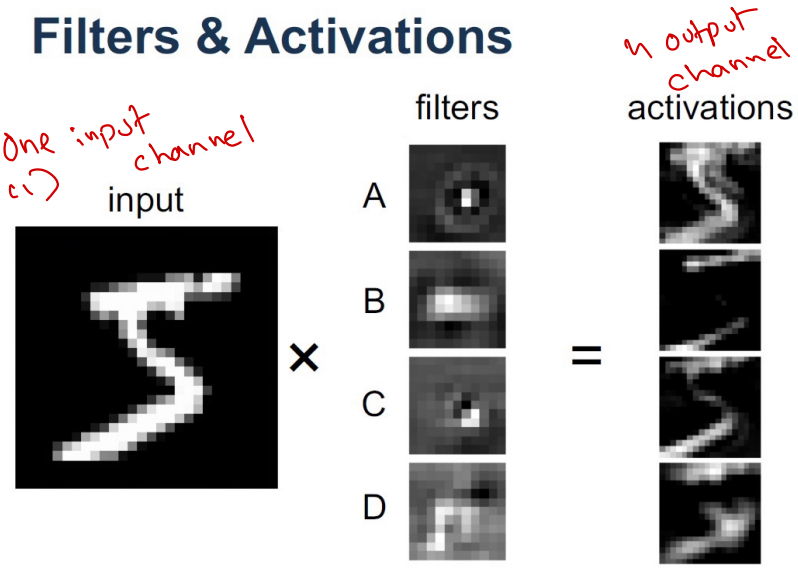


→ detect → Edge

4 Neurons

Multiple filters: detect multiple features in diff position.

Activation maps (or) feature maps doesn't necessarily to be the same in put size. As we convolve.



Pdame teg Calculation.

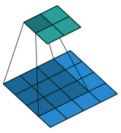
(inp-channel size) x (filter size) x (no. of filters).

1 x 2x2 x 4 = 16.

if bias is there the 16+4 = 20.

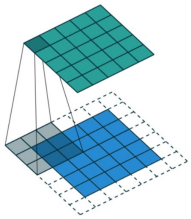
Padding in “valid” Mode

- rationale:
  only apply filters in actual (valid) data, i.e. no padding
- given a 1D-input with length *n* and a convolutional filter with length *k*, the resulting output size is *n*-*k*+1



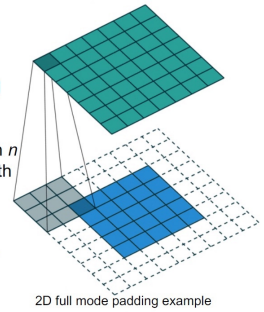
Padding in “same” Mode

- rationale:
  output has the same size as the input
- given a 1D-input with length *n* and a convolutional filter with length *k*, add (*k*-1) / 2 zeros at each end of the input



Padding in “full” Mode

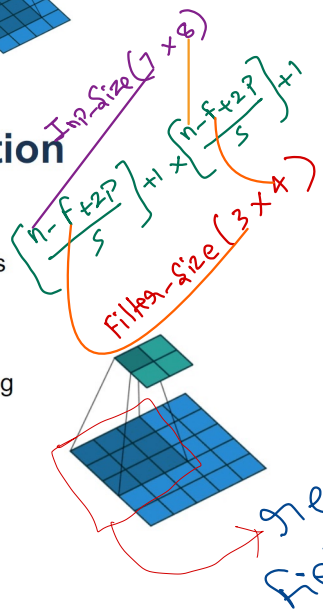
- rationale:
  consider every possible superimposition of filter and input
- given a 1D-input with length *n* and a convolutional filter with length *k*, add *k*-1 zeros at each end of the input
- size of output increased by *k*-1



2D full mode padding example

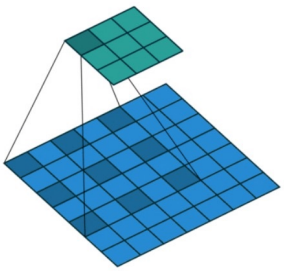
Strided Convolution

- rationale:
  decrease resolution (and thus dimensionality)
  reduce computation
- same effect as down-sampling



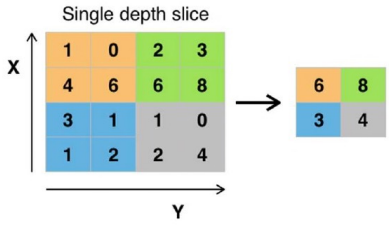
Dilated Convolution

- rationale:
  increase receptive field size
- “inflate” filter by inserting spaces between filter elements
- dilation rate *d* corresponds to *d*-1 spaces



2D dilated convolution example

Pooling



here: stride 2 => downsampling

reduces multiple values into one

introduces translation invariance

different aggregation functions:
- max pooling
- mean/avg pooling
- …

No trainable parameters!

Strong Priors

- CNN = “fully connected net with an infinitely strong prior [on weights]”
- only useful when the assumptions made by the prior are reasonably accurate
- convolution+pooling can cause underfitting

Convolution Properties

Which property does **not** apply to convolution?

- a) Sparse Interaction
- b) Parameter Sharing
- c) Translation ~~Invariance~~ Equivariance
- d) Local Connectivity

1. Local Connectivity

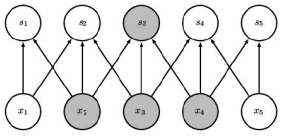
For 5 inputs and a fully-connected layer with 5 neurons, there are 25 connections.
How many connections are there in case of a convolutional layer with filter width 3? (no padding!)

- a) 12
- b) 13
- c) 14
- d) 15

2. Shared Weights

How many weight parameters does this convolutional layer have?

- a) 3
- b) 5
- c) 13
- d) 15



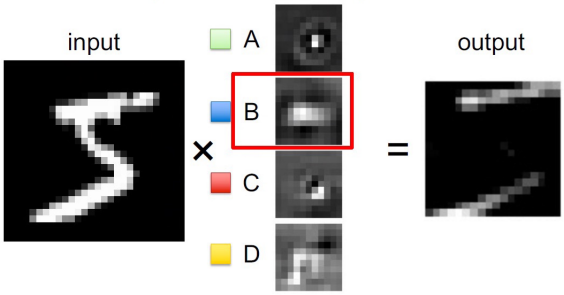
3. Translation Equivariance

- What is **equi**variance of a function f with respect to another function g?

- a) f(x) = g(f(x))
- ~~b) f(x) = f(g(x))~~ invariance w.r.t. g
- c) f(g(x)) = g(f(x))
- d) f(g(x)) = g(x) * f(x)

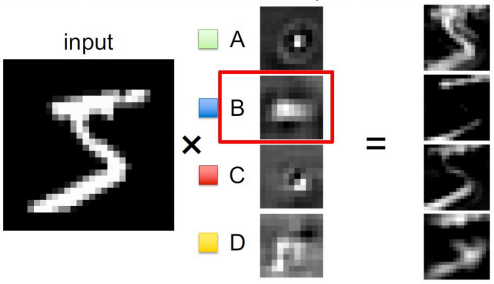
Filter-Activation Matching

Which filter produces the output?



Filter Usefulness

Which filter is most useful (for MNIST task)?

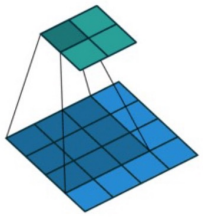


How can we encourage learning complex filters?

- a) penalize filter sparsity
- b) penalize activation sparsity
- c) encourage filter sparsity
- d) encourage activation sparsity

Padding in "valid" Mode
(no zero-padding)

Given a 1D-input with length *n* and a convolutional filter with length *k*, how long is the output?

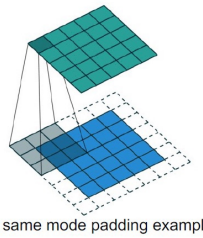


2D valid mode padding example

- a) n+k+1
- b) n+k-1
- [x] c) n-k+1
- d) n-k-1

Padding in "same" Mode
(output has the same length as the input)

Given a 1D-input with length *n* and a convolutional filter with length *k*, how much zero-padding has to be applied **at each end of the input**?

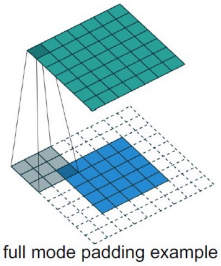


2D same mode padding example

- [x] a) (k-1) / 2
- b) k / 2
- c) (k+1) / 2
- d) k - 2

Padding in "full" Mode
(every possible superimposition of filter and input)

Given a 1D-input with length *n* and a convolutional filter with length *k*, how much zero-padding has to be applied **at each end of the input**?

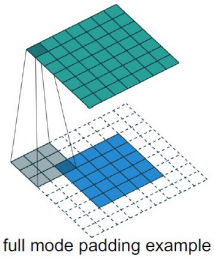


2D full mode padding example

- a) k / 2
- b) k / 2 + 1
- c) k / 2 - 1
- [x] d) k - 1

Padding in "full" Mode
(every possible superimposition of filter and input)

Given a 1D-input with length *n* and a convolutional filter with length *k*, how long is the output?



2D full mode padding example

- a) n + k / 2
- [x] b) n + k - 1
- c) n + k / 2 - 1
- d) n + k + 1

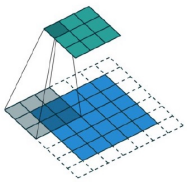
$(k-1)(d-1)+K$

$(7-1)\times(3-1)+7$

$12+7=19$

Strided Convolution
(same effect as down-sampling but less computation)

Given a 1D-input with length *5* and a convolutional filter with length *3*, padding *1* and stride *2*, how long is the output?



2D strided convolution example

- a) 2
- [x] b) 3
- c) 4
- d) 5

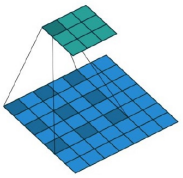
$\left(\frac{n-f+2p}{s}\right)+1$

$\frac{5-3+2}{2}+1=3$

Dilated Convolution

"Inflate" filter by inserting spaces between filter elements (dilation rate *d* corresponds to *d-1* spaces)

What is the effective filter size for a filter of length 7 and a dilation rate of 3?



2D dilated convolution example

- [x] a) 19
- b) 20
- c) 21
- d) 22

when dilation is there
First we calculate filter_size

$(k-1)(d-1)+K$

$\text{filter\_size} = (3-1)(2-1)+K = 5$

Pooling (3 Points)

(untrainable) aggregation (e.g. max/average) with similar hyper-params like convolution (width, stride, padding)

Which operations do **not** result in an output of length 3 for an input of length 7?

- a) conv(d=1, width=3, stride=1, pad=same)
  pool(width=2, stride=2, pad=valid)
- b) conv(d=1, width=3, stride=2, pad=valid)
  pool(width=3, stride=1, pad=same)
- c) conv(d=2, width=3, stride=1, pad=valid)
  pool(width=2, stride=1, pad=same)
- [x] d) conv(d=2, width=3, stride=1, pad=valid)
  pool(width=2, stride=1, pad=valid)

output size

$= (n-k)+1$

$= (7-5)+1 = 3$

after pooling

$= (n-k)+1$

$= (3-2)+1$

$= 2$

Pooling (3 Points)

(untrainable) aggregation (e.g. max/average) with similar hyper-params like convolution (width, stride, padding)

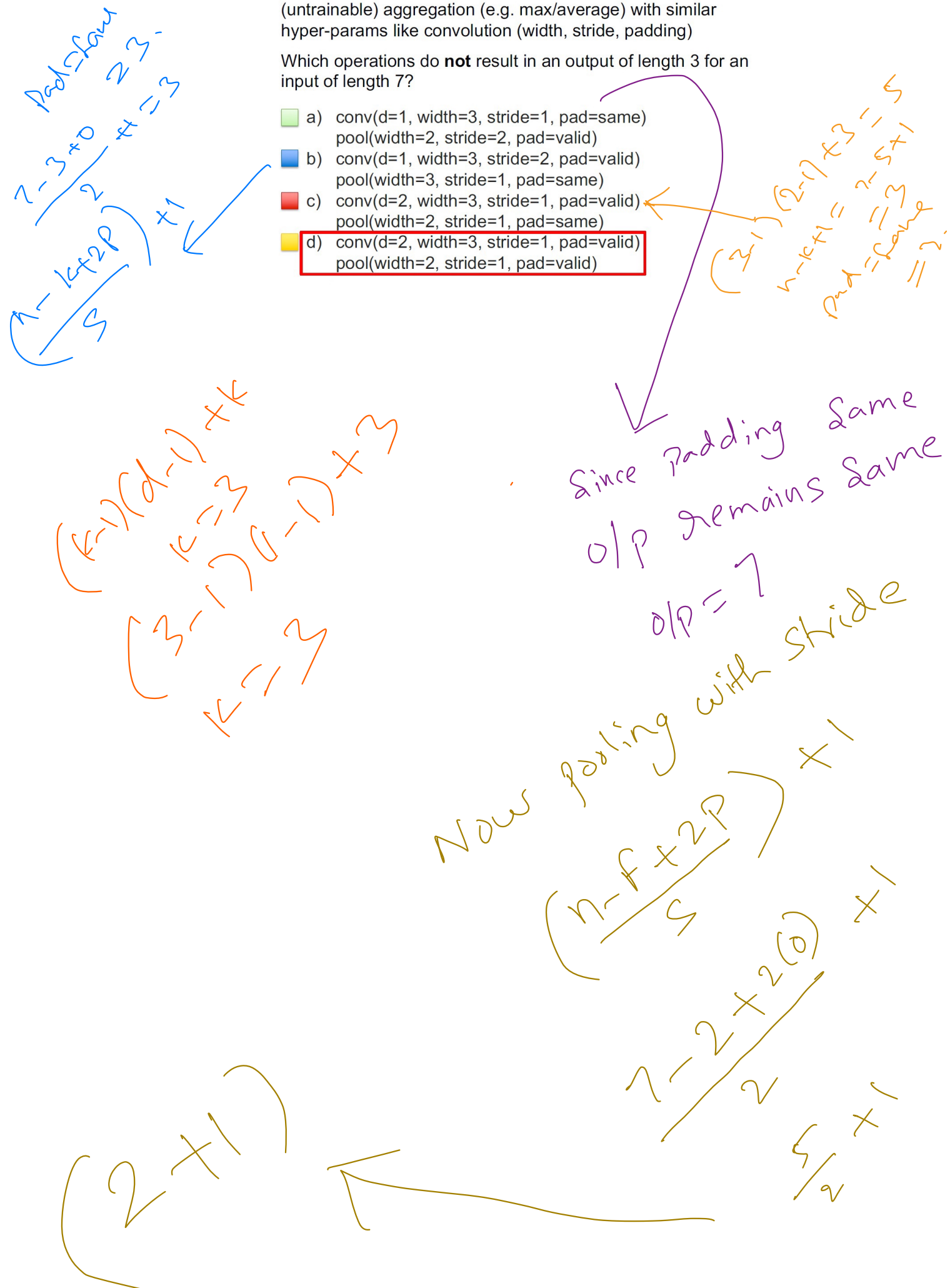
Which operations do **not** result in an output of length 3 for an input of length 7?

a) conv(d=1, width=3, stride=1, pad=same)
pool(width=2, stride=2, pad=valid)

b) conv(d=1, width=3, stride=2, pad=valid)
pool(width=3, stride=1, pad=same)

c) conv(d=2, width=3, stride=1, pad=valid)
pool(width=2, stride=1, pad=same)

d) conv(d=2, width=3, stride=1, pad=valid)
pool(width=2, stride=1, pad=valid)



What is meant by infinitely strong prior?

CNN can be thought of as FCN but with strong priors on weights. CNN can be represented as FCN where we have 0 weight values for no connections and non-zero weights for existing connections, making it sparse. Also weights of 1 hidden unit should be identical to the weights of neighbor but shifted in space.

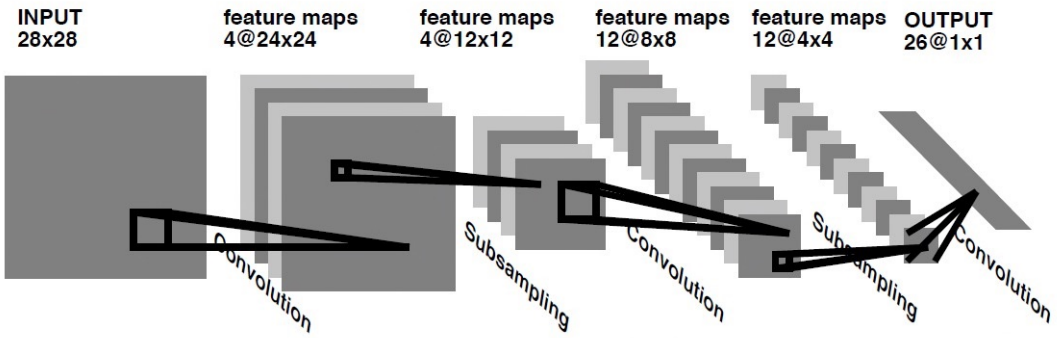
Convolution + pooling can cause underfitting because of the strong prior.

Strong Priors

- CNN = "fully connected net with an infinitely strong prior [on weights]"
- only useful when the assumptions made by the prior are reasonably accurate
- convolution+pooling can cause underfitting

LENET

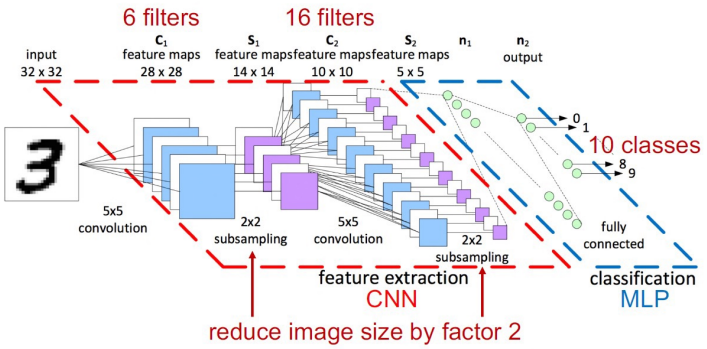
CNN from Y. LeCun and Y. Bengio: **Convolutional Networks for Images, Speech, and Time-Series**, in Arbib, M. A. (Eds), *The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.*



	shape 5x5 stride 1 mode "valid"	shape 2x2 stride 2 per-filter bias	shape 5x5 stride 1 mode "valid"	shape 2x2 stride 2 per-filter bias	= fully connected
receptive field:	5x5	6x6	14x14	16x16	28x28
# parameters:	4x1x5x5 = 100	4	12x4x5x5 = 1200	12	26x12x4x4 = 4992
	total: 6308				

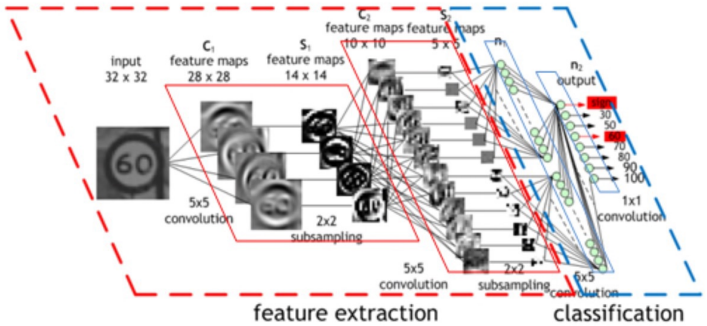
parameters for an MLP with a single fully-connected layer: 28x28x26 = 20384

Convolutional Neural Nets (CNNs)



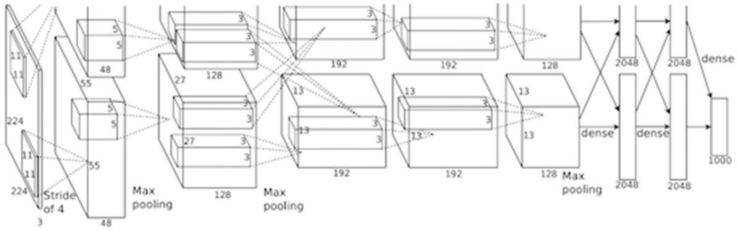
adapted from M. Peemen et al. "Speed sign detection and recognition by convolutional neural networks." *Proc. of 8th Int. Automotive Congress*. 2011.

Convolutional Neural Nets (CNNs)



https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/ Image by M. Peemen

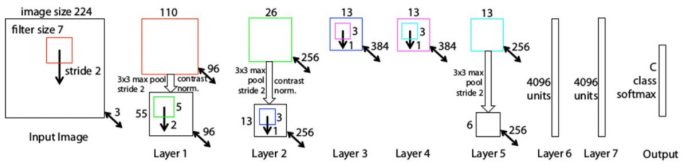
AlexNet (2012)



- distributed onto 2 GPUs
- 11x11 filters with stride 4
  - skipping a lot of relevant information, especially as this is the first conv layer

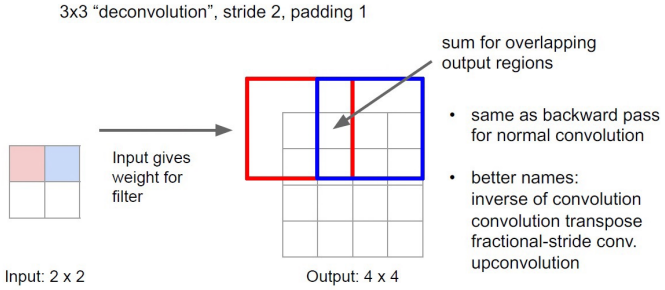
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

ZF-Net (2013)



- uses only 7x7 filters with stride 1
  - retain a lot of original pixel information in the input volume
- deconvolutional layers
  - visualize conv. filter by adding a path back to the input domain to each conv layer

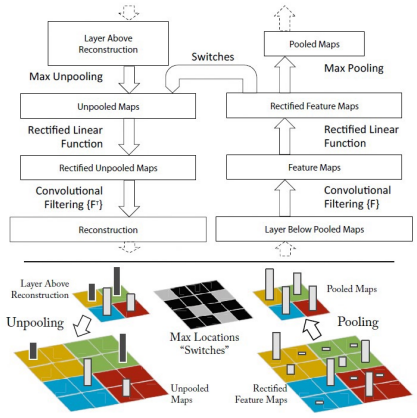
Deconvolution



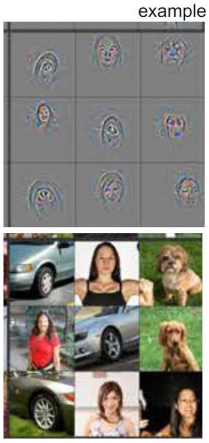
- same as backward pass for normal convolution
- better names:
  inverse of convolution
  convolution transpose
  fractional-stride conv.
  upconvolution

recommended reading: https://distill.pub/2016/deconv-checkerboard/
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

DeconvNet Layer

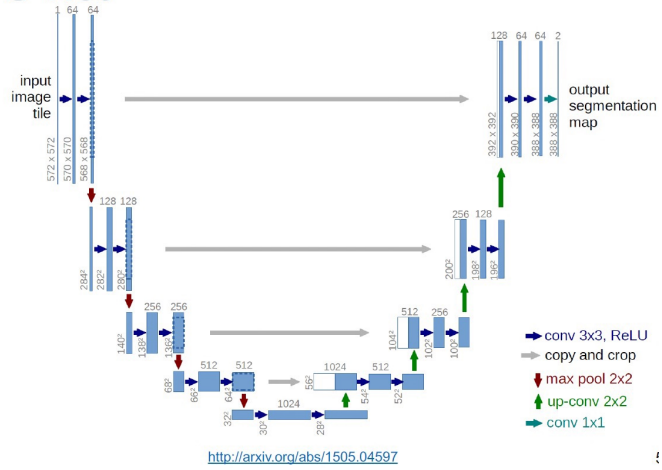


example



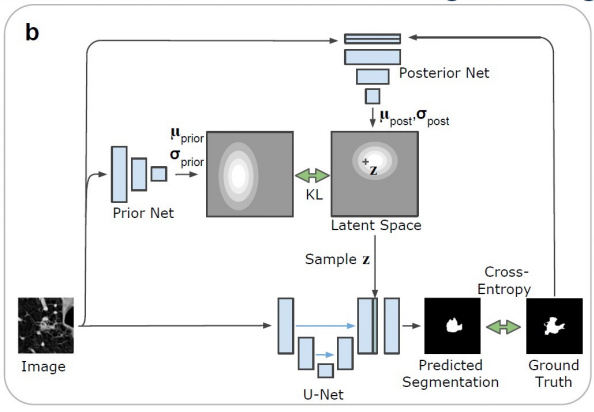
https://cs.nyu.edu/%7Efergus/papers/zeilerECCV2014.pdf

U-Net



http://arxiv.org/abs/1505.04597

Probabilistic U-Net for ambiguous images

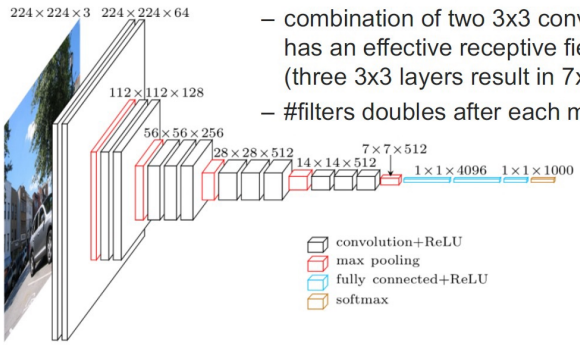


https://www.youtube.com/watch?v=-cfFxQWfFrA
http://papers.nips.cc/paper/by-source-2018-3455

VGG Net

simplicity and depth:

- only 3x3 filters with stride and pad 1
- only 2x2 maxpool with stride 2
- combination of two 3x3 conv layers has an effective receptive field of 5x5 (three 3x3 layers result in 7x7)
- #filters doubles after each maxpool



https://arxiv.org/abs/1409.1556

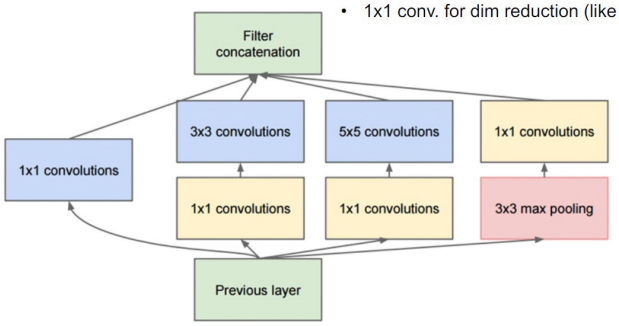
VGG Net

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					

The 6 different architecures of VGG Net. Configuration D produced the best results

Inception Module

- different parallel processing pathways (different filter sizes / pooling)
- output **concatenated**
- 1x1 conv. for dim reduction (like PCA)



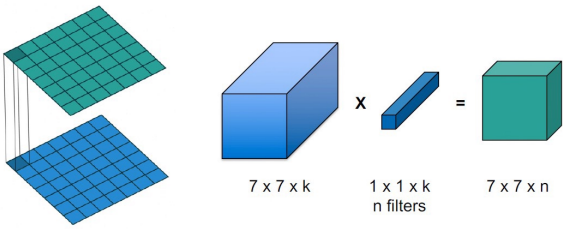
https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

1x1 Convolutions

"In Convolutional Nets, there is no such thing as 'fully-connected layers'. There are only convolution layers with 1x1 convolution kernels and a full connection table."

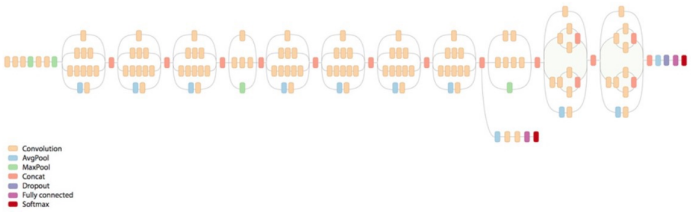
Yann LeCun

Dimensionality Reduction:



further reading: https://iamaaditya.github.io/2016/03/one-by-one-convolution/

GoogLeNet



- 9 Inception modules with over 100 layers total
- 12x fewer parameters than AlexNet
- average pool instead of fully-connected layer

https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

Inception v2 – Adding Batch Norm

- reducing internal covariance shift
- make normalization part of architecture

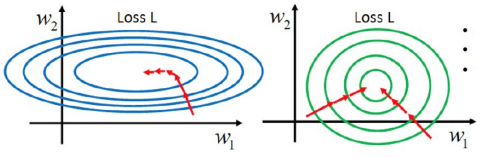
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$; Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



- allows higher LR
- less sensitive to init.
- eliminates need for Dropout

http://proceedings.mlr.press/v37/ioffe15.html

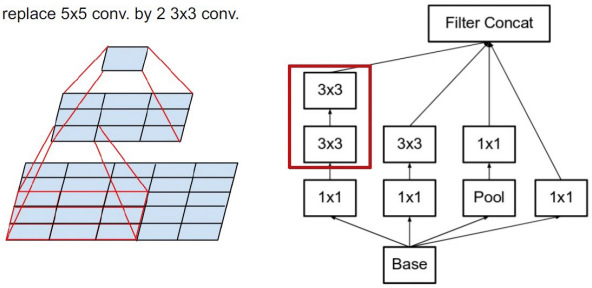
Inception v3

- General Design Principles:
  1. Avoid representational bottlenecks, especially early in the network
  2. Higher dimensional representations are easier to process locally within a network.
  3. Spatial aggregation can be done over lower dimensional embeddings without much or any loss in representational power.
  4. Balance width and depth of the network

https://arxiv.org/abs/1512.00567

Inception v3

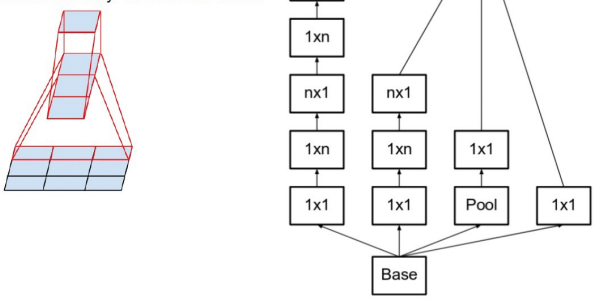
replace 5x5 conv. by 2 3x3 conv.



https://arxiv.org/abs/1512.00567

Inception v3

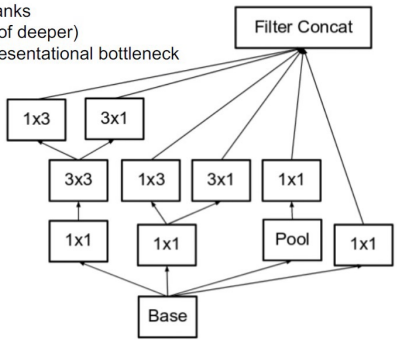
replace nxn conv. by 1xn and nx1 conv.



https://arxiv.org/abs/1512.00567

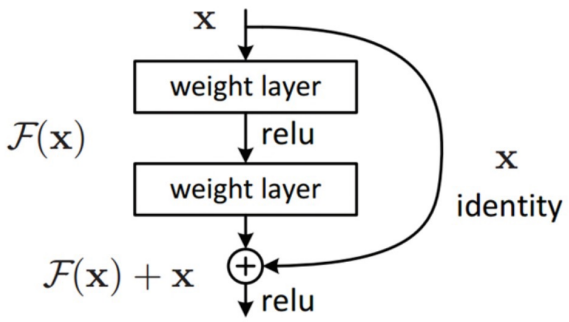
Inception v3

expand filter banks
(wider instead of deeper)
to remove representational bottleneck



https://arxiv.org/abs/1512.00567

Residual Block

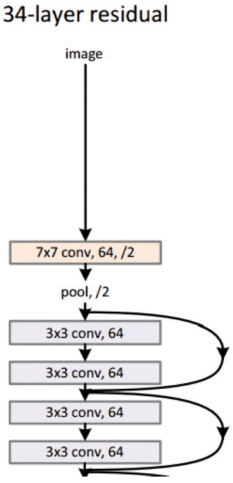


optionally multiple (= cardinality) parallel residual paths possible (ResNeXt)

http://arxiv.org/abs/1512.03385

ResNet

- 152 layers
- *first 2* layers compress input from 224x224 to 56x56 pixels
- 3.6% ILSVRC error rate

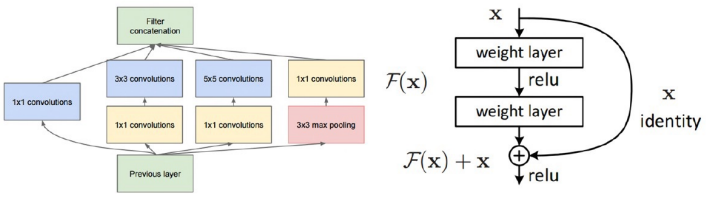


http://arxiv.org/abs/1512.03385

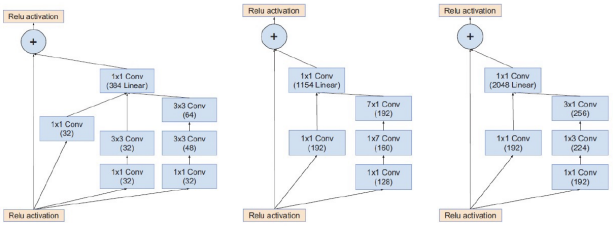
Inception vs. Residual Block

Concatenation

Element wise addition

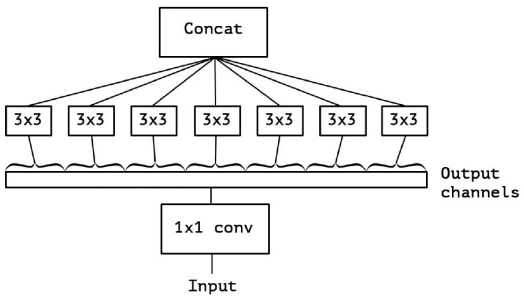


Inception ResNet v2



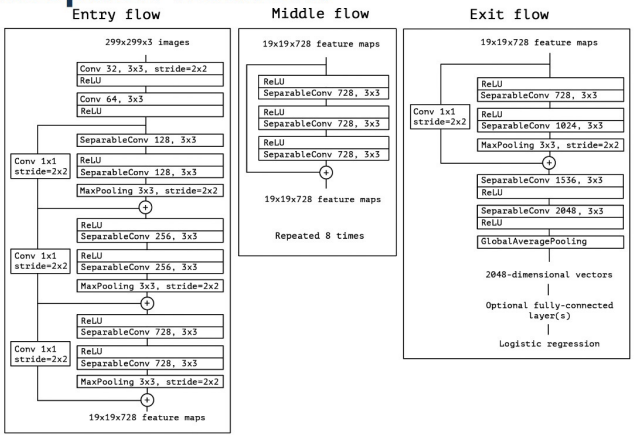
https://arxiv.org/abs/1602.07261

Xception Module



https://arxiv.org/abs/1610.02357

Xception Network



https://arxiv.org/abs/1610.02357

assume growth rate = 32, L = 3

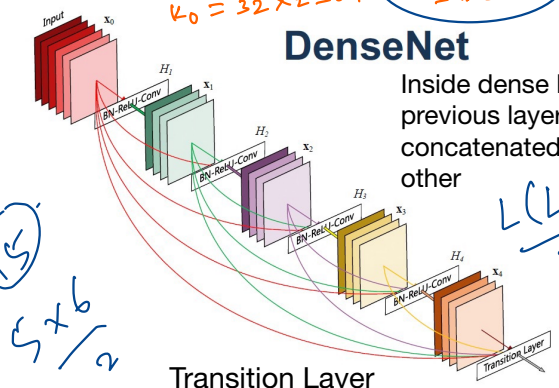
$k_0 + k \times (l-1) \rightarrow 64 + 32(3-1) = 128$

$k_0 = 32 \times 2 = 64$

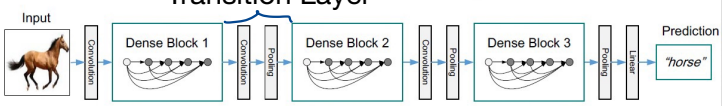
DenseNet

Inside dense block all previous layers are concatenated to each other

$\frac{L(L+1)}{2}$ connections

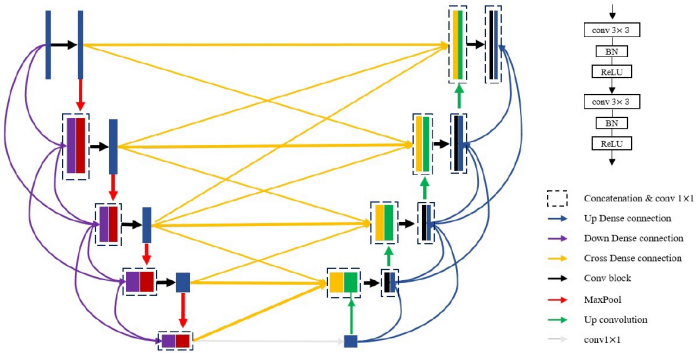


Transition Layer



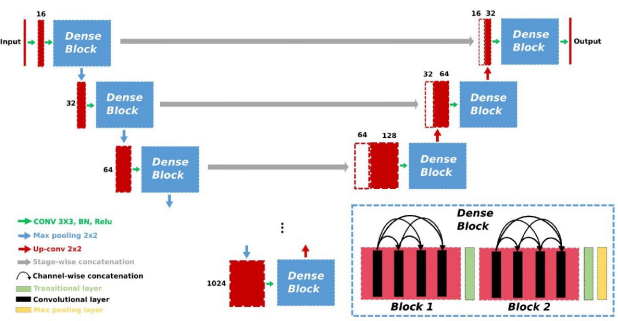
https://arxiv.org/abs/1608.06993

Multi-Scale Dense U-Net



https://arxiv.org/abs/1812.00352

Dense U-Nets for MR Images



N. Aldoj, F. Biavati, F. Michallek, S. Stober & M. Dewey: "**Automatic prostate and prostate zones segmentation of magnetic resonance images using convolutional neural networks**" (under review)

Wav2Letter

R. Collobert, C. Puhrsch and G. Synnaeve. **Wav2letter: an end-to-end convnet-based speech recognition system.** *arXiv:*1609.03193, 2016.
https://arxiv.org/abs/1609.03193