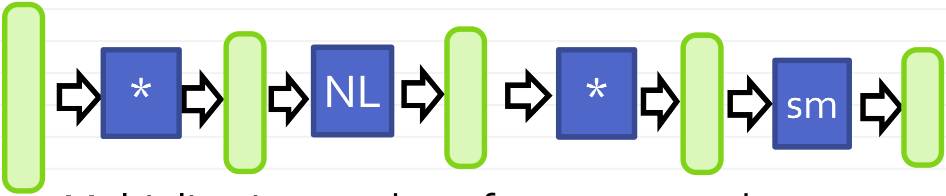
Lecture 3: Convolutional Networks

Neural network: parameter overdoze



Multiplicative number of parameters: the main problem. Solving the problem:

- Stop optimization early (always keep checking progress on validation set)
- Impose smoothness (weight decay)
- Bag multiple models

Main avenue: picking a less generic architecture with less parameters

Huber-Wiesel

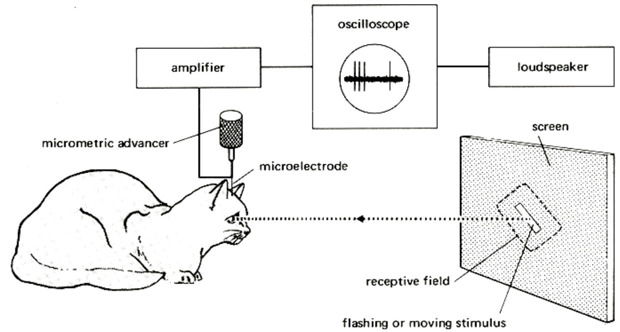
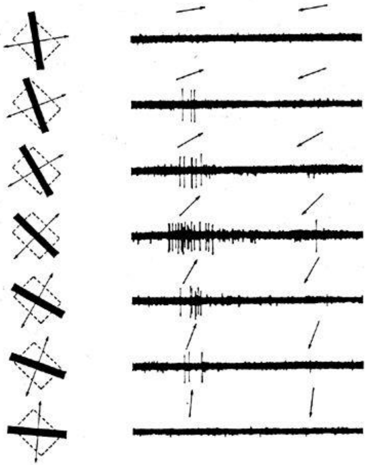


Fig. 7 *If a microelectrode is inserted into the brain and picks up impulses from a single nerve cell, the neuron's responses can be studied by projecting patterns of light in front of the stationary eyes.*

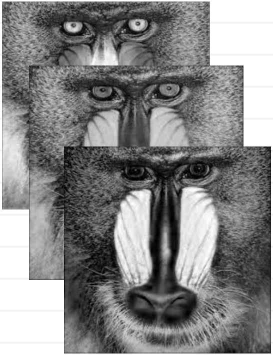
The reaction of a neuron is localized to a part of the visual field

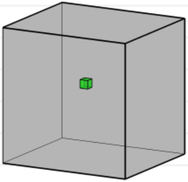
Huber and Wiesel 1968

V1 physiology:
direction
selectivity



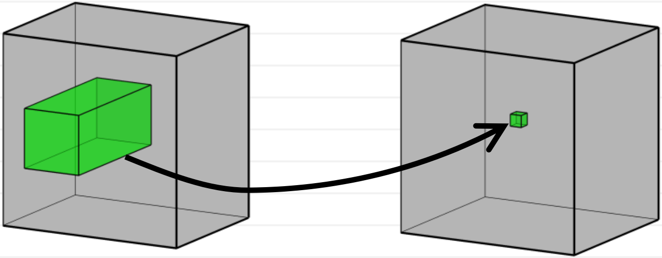
Idea 1: neuron maps





1. Organize neurons into *map stacks*
2. E.g. an image is a WxHx3 map stack

Idea 1: locally-connected layer

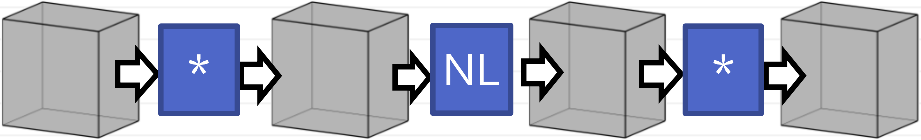


1. Organize neurons into *maps*
2. Limit the *receptive field* in the multiplicative layer:

$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K^{x,y,t}(i-x+\delta, j-y+\delta, s) \cdot U(i,j,s)$$

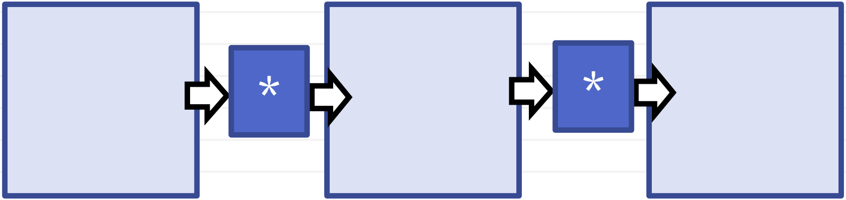
3. Huge parameter reduction by a factor of O($(W/2\delta)^2$) compared to fully-connected layers

Stacking layers



- The layers are stacked and interleaved with non-linearities (e.g. ReLU)

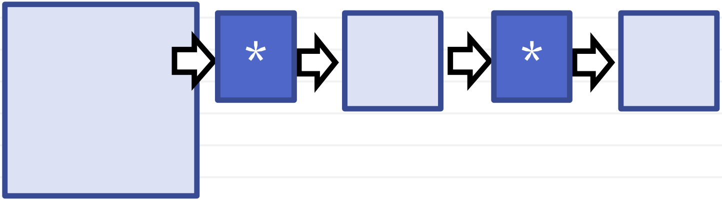
Growing receptive field



E.g. with 7x7 filters:

- Receptive field is 7x7 after 1 conv layers
- Receptive field is 13x13 after 2 conv layers

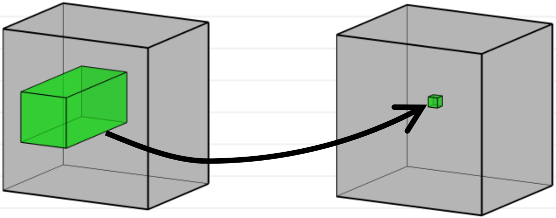
Accelerating growth: stride



- In the early layers strides are often > 1
- Strides > 1 downsample maps
- Strides > 1 increase the receptive fields

What is the receptive field after [7x7 stride=2] followed by [7x7 stride = 1] convolution?

Idea 2: tying together weights



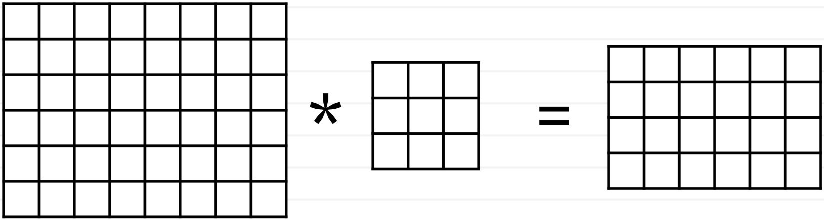
$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K^t(i-x+\delta, j-y+\delta, s) \cdot U(i,j,s)$$

Further dramatic reduction in the number of parameters by a factor O(W^2) compared to locally-connected layer:

$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t) \cdot U(i,j,s)$$

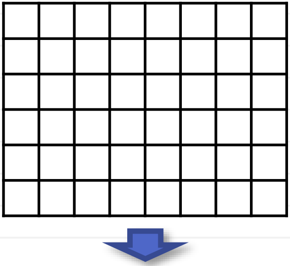
Boundary issues

"Valid" mode:



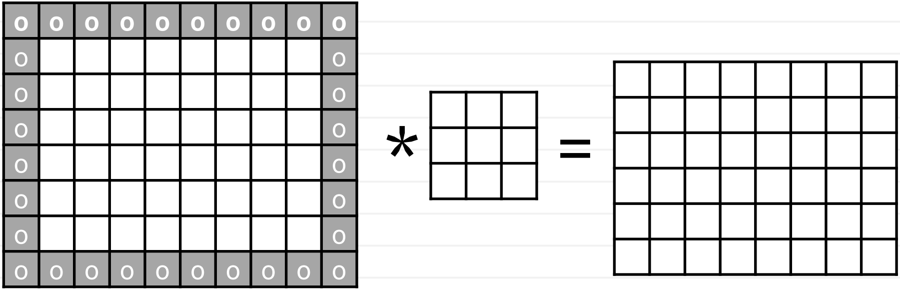
- Complicates implementation and reasoning
- Unequal contribution of elements

Boundary issues: padding with zeros



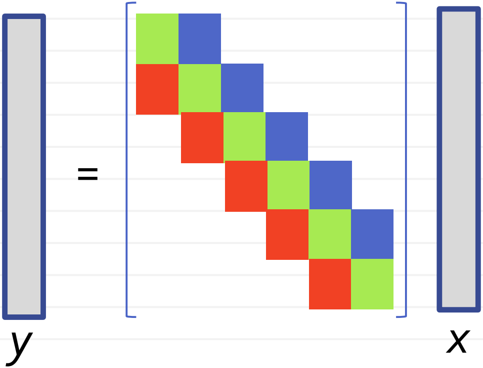
“Same” mode:

- Solves the problem
- Introduces “false” edges



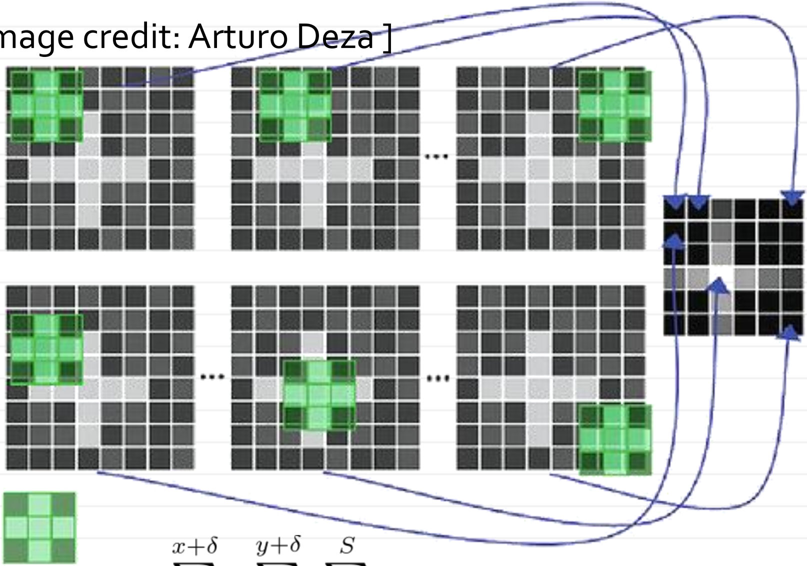
Conv. layer is still multiplicative

E.g. 1D correlation with 
is a multiplication over a banded matrix:



Interpretation: looking for patterns

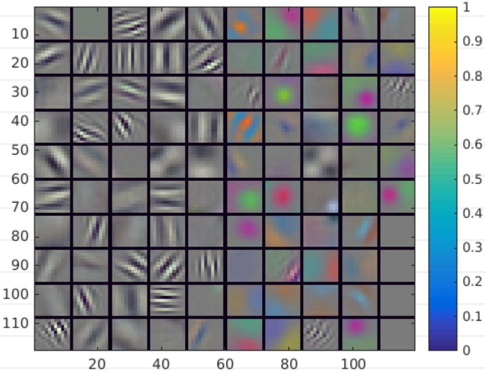
[Image credit: Arturo Deza]



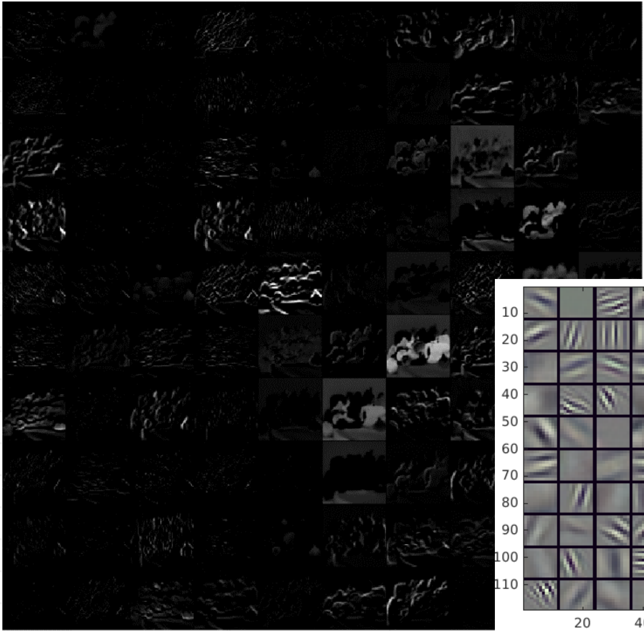
$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t) \cdot U(i,j,s)$$

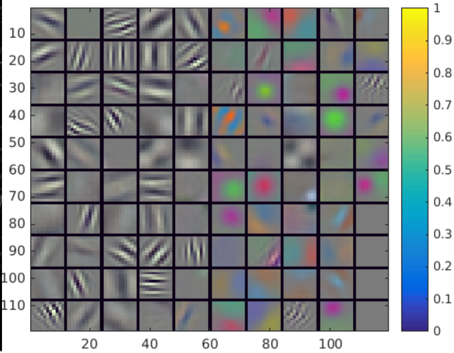
Interpretation: looking for patterns

AlexNet filters of the first layer:

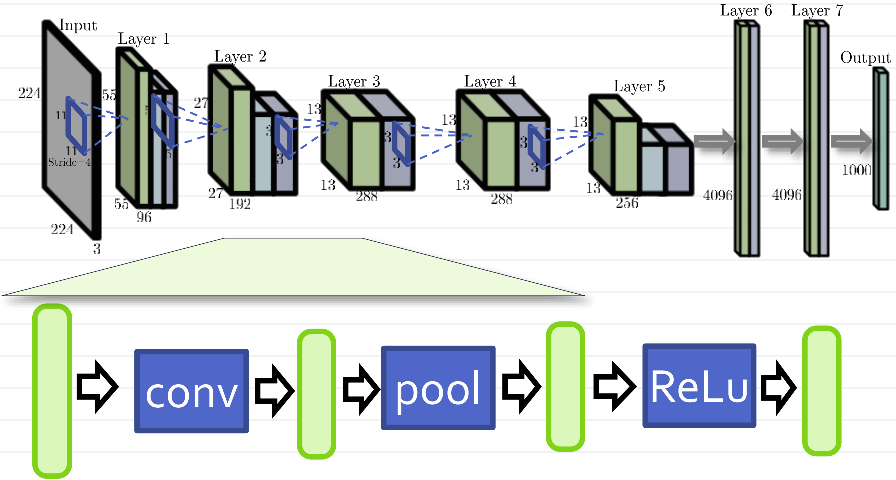


Responses in the first layer

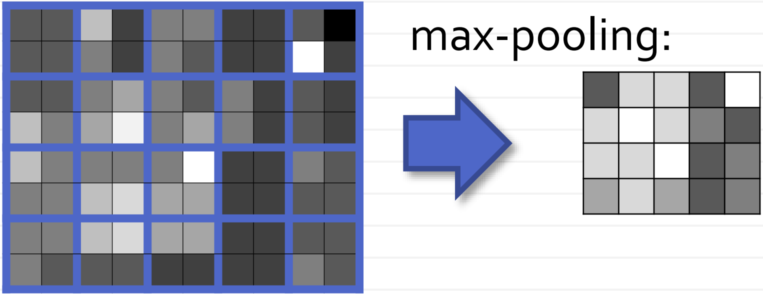




What are modern ConvNets made of



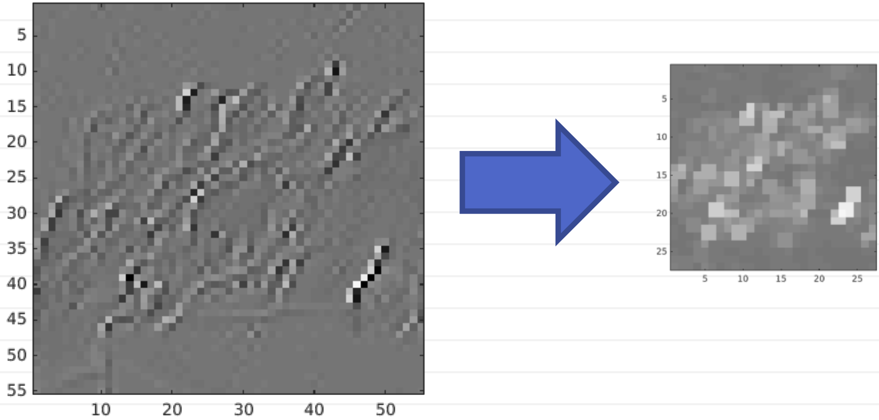
Third-component: pooling (+subsampling)



Pooling is almost always with subsampling

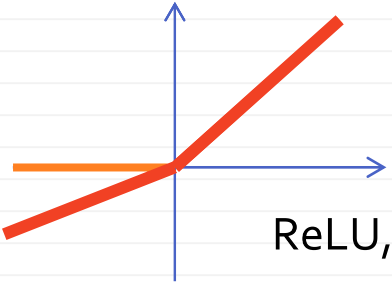
Alternatives: sum-pooling, average pooling,
Rapid decrease of map size
Parameter-free

Max-pooling and jitter-invariance



- Usual motivation: adding invariance to small shifts
- Several max-poolings can accumulate invariance to stronger shifts

Details: nonlinearity

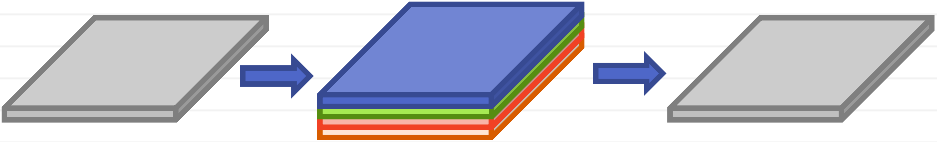


$$f(x;\alpha) = \max(x, \alpha x)$$

ReLU, leaky ReLu

Another formerly popular non-linearity: *maxout*

$$f(x) = \max(\alpha_1 x + \beta_1, \alpha_2 x + \beta_2, \ldots, \alpha_m x + \beta_m)$$



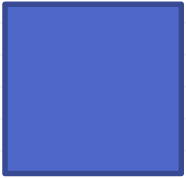
CNN applications

Pattern finding through convolution/correlation is ubiquitous:

- 2D images (and the like, e.g. speech)
- 1D signals (e.g. time series, speech)
- 3D images
- Videos
- Graphs (more generalized sense)

[Bruna et al. Spectral Networks and Locally Connected Networks on Graphs. ICLR 2014]

Reminder: layer abstraction



Each layer is defined by:

- forward performance: $y = f(x; w)$
- backward performance:

$$z(x) = z(f(x;\, w))$$

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy} \qquad\qquad \frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

Backprop equations: multipicative layer

$$z(x) = z(f(x;\, w))$$

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy} \qquad\qquad \frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

$$y = Wx$$

$$\frac{dy}{dx} = W$$

$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$

Backprop equations: convolutional layer

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy}$$

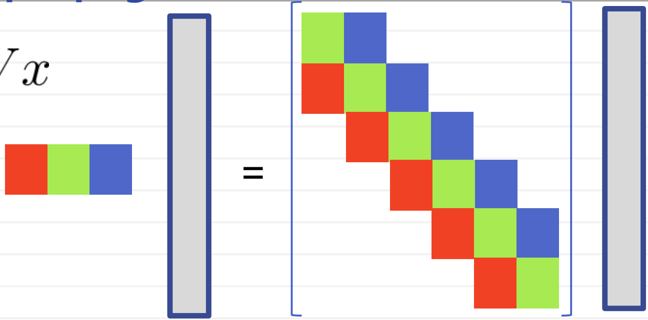
$$y = Wx$$ - still holds for conv layer

$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$ - also corresponds to some correlation?

Backpropagation via convolution

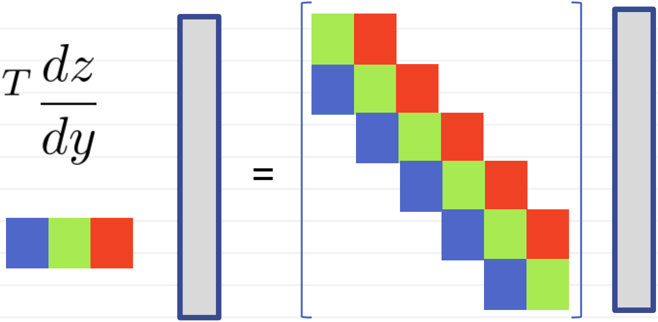
$$y = Wx$$

Corr with



$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$

Corr with



Backprop equations: convolutional layer

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy}$$

$$y = Wx$$ - still holds for conv layer

$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$ - also corresponds to conv

During backpass do the same correlations but with flipped kernels

$$K^{\text{back}}(i,j,t,s) = K(2\delta + 1 - i, 2\delta + 1 - j, s, t)$$

Backprop equations: multipicative layer

$$z(x) = z(f(x;\, w))$$

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy}$$

$$y = Wx$$

$$\frac{dy}{dx} = W$$

$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$

$$\frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

$$\frac{\partial z}{\partial w_{ij}} = \left(\frac{dy}{dw_{ij}}\right)^T \frac{dz}{dy} = x_j \frac{\partial z}{\partial y_i}$$

$$\frac{dz}{dW} = \frac{dz}{dy} \cdot x^T$$

Backprop equations: convolutional layer

$$\frac{dz}{dW} = \frac{dz}{dy} \cdot x^T \qquad \frac{\partial z}{\partial W_{ij}} = \frac{\partial z}{\partial y_i} \cdot x_j$$

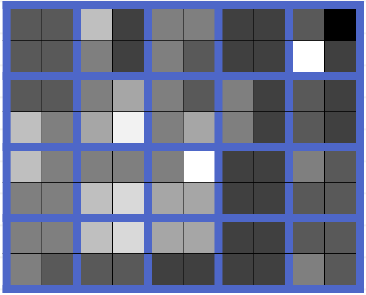
In conv. layer we tie together multiplicative weights corresponding to the same relative position of y_i and x_j.

So the formula becomes:

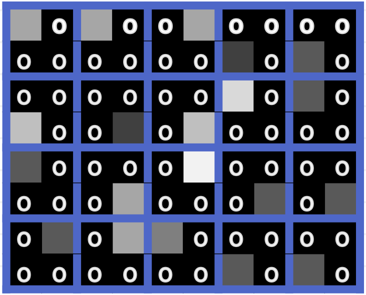
$$\frac{\partial z}{\partial K_{m,n,s,t}} = \sum_{k,l} \frac{\partial z}{\partial y_{k,l,t}} \cdot x_{k-m,l-n,s}$$

NB: this is also a convolution between dz/dy and x

Backpropagation: max-pooling

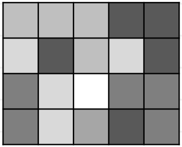


forward pass:

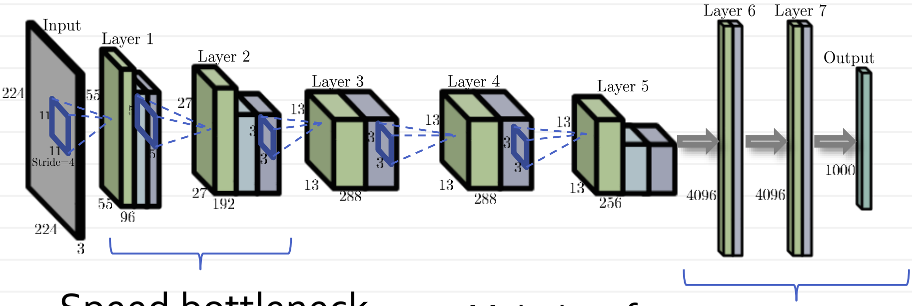


backward pass:

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy}$$



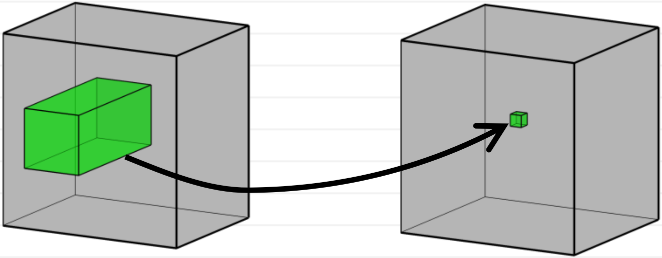
Bottlenecks



Speed bottleneck

Majority of parameters
Drop-out often applied here

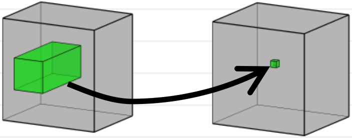
Efficient implementations: direct



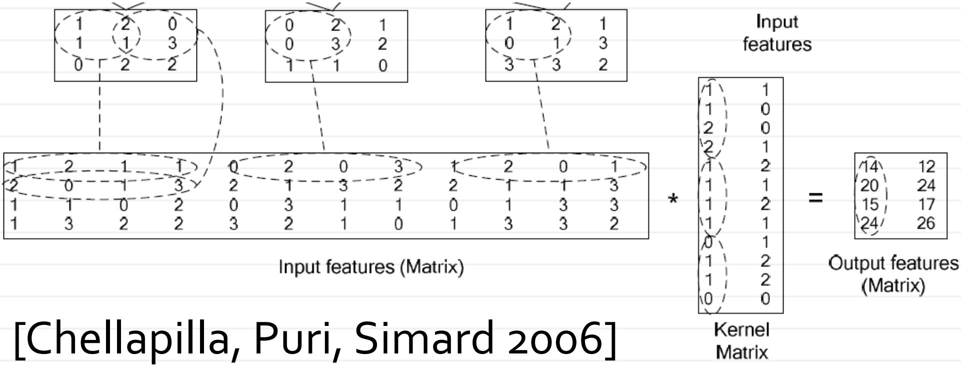
$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t) \cdot U(i,j,s)$$

- Loop ordering very important
- Data alignment very important
- NVIDIA cuDNN, Nervana kernels - efficient GPU implementations

Efficient implementations: im2col



Idea: reduce all *ST* convolutions to a single matrix multiplication



[Chellapilla, Puri, Simard 2006]

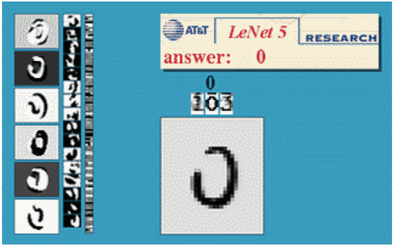
Efficient implementations: Fourier

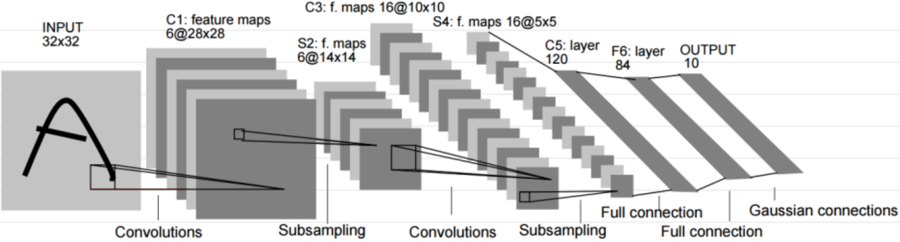
$$A * K = \mathcal{F}^{-1}(\mathcal{F}(A) \odot \mathcal{F}(K))$$

- Each map participates in many convolutions (hence FFT is reused)
- Maps are much smaller than images that most FFT codes are optimized for
- Careful implementation needed to get reasonable speed-up
- Memory hungry (why?) ☹
- **Does not help for very small filters**

[Fast Training of Convolutional Networks through FFTs
Michael Mathieu; Mikael Henaff; Yann LeCun, ICLR 2014]

History: LeNet





[LeCun 89, 98]

2012: Image-net

IMAGENET

www.image-net.org

14,197,122 images, 21841 synsets indexed

(all competitions on much smaller subset 1000x1000)

Statistics of high level categories

High level category	# synset (subcategories)	Avg # images per synset	Total # images
amphibian	94	591	56K
animal	3822	732	2799K
appliance	51	1164	59K
bird	856	949	812K
covering	946	819	774K
device	2385	675	1610K
fabric	262	690	181K
fish	566	494	280K
flower	462	735	339K
food	1495	670	1001K
fruit	309	607	188K

Image-net

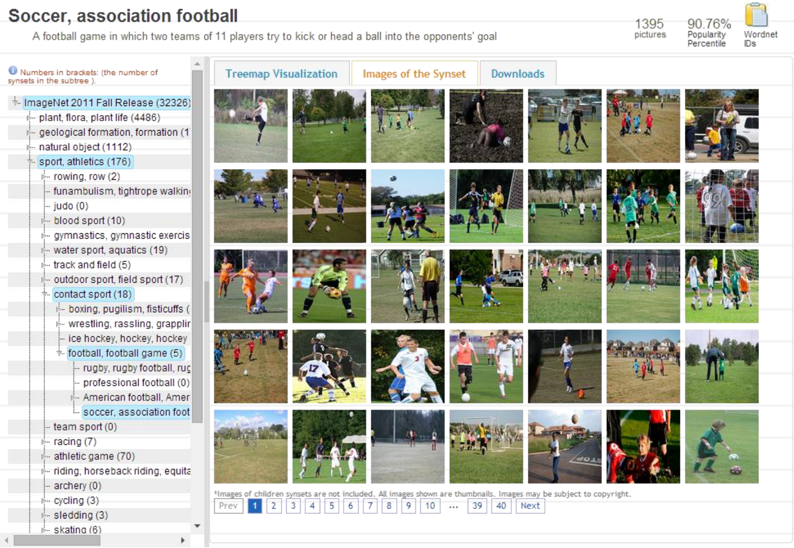
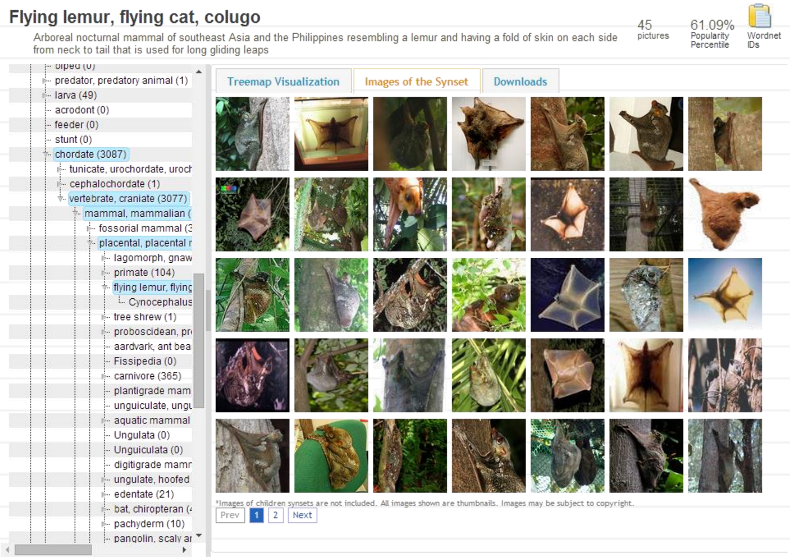
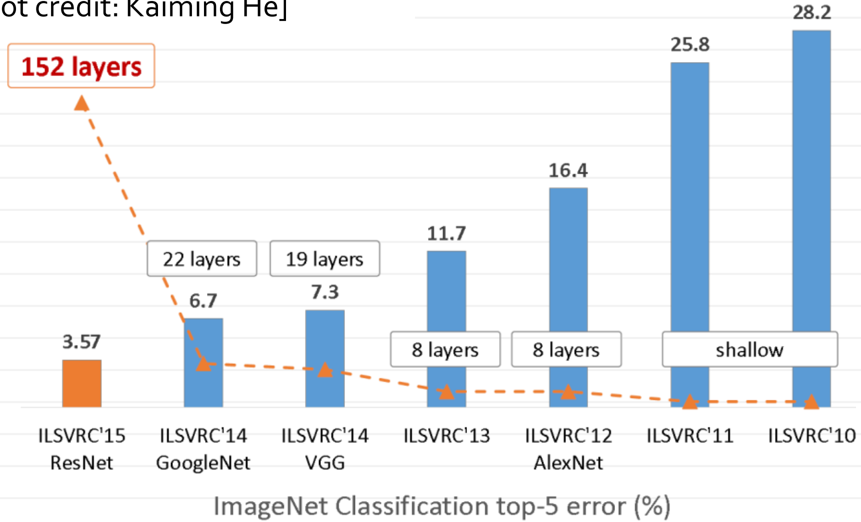

Soccer, association football
A football game in which two teams of 11 players try to kick or head a ball into the opponents' goal
1395
pictures
90.76%
Popularity
Percentile
Wordnet
IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (1
natural object (1112)
sport, athletics (176)
rowing, row (2)
funambulism, tightrope walkin
judo (0)
blood sport (10)
gymnastics, gymnastic exercis
water sport, aquatics (19)
track and field (5)
outdoor sport, field sport (17)
contact sport (18)
boxing, pugilism, fisticuffs (
wrestling, rassling, grapplir
ice hockey, hockey, hockey
football, football game (5)
rugby, rugby football, rug
professional football (0)
American football, Amer
soccer, association foot
team sport (0)
racing (7)
athletic game (70)
riding, horseback riding, equita
archery (0)
cycling (3)
sledding (3)
skating (6)
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev
1
2
3
4
5
6
7
8
9
10
...
39
40
Next

Image-net

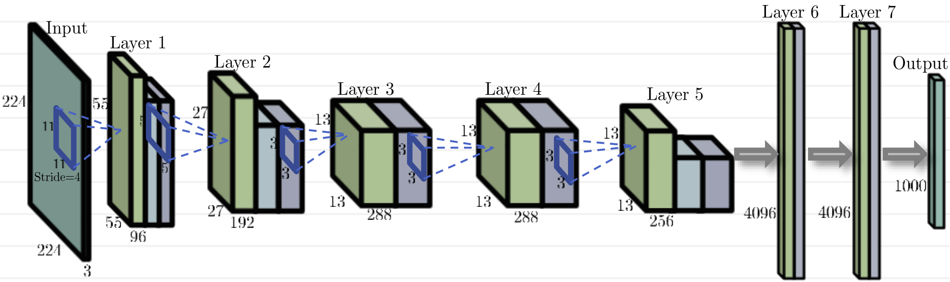

Flying lemur, flying cat, colugo
Arboreal nocturnal mammal of southeast Asia and the Philippines resembling a lemur and having a fold of skin on each side from neck to tail that is used for long gliding leaps
45 pictures
61.09% Popularity Percentile
Wordnet IDs
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, uroch
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (
fossorial mammal (3
placental, placental r
lagomorph, gnaw
primate (104)
flying lemur, flying
Cynocephalus
tree shrew (1)
proboscidean, pr
aardvark, ant bea
Fissipedia (0)
carnivore (365)
plantigrade mam
unguiculate, ungu
aquatic mammal
Ungulata (0)
Unguiculata (0)
digitigrade mamr
ungulate, hoofed
edentate (21)
bat, chiropteran (
pachyderm (10)
pangolin, scaly ar
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev
1
2
Next

Building the best network

[plot credit: Kaiming He]



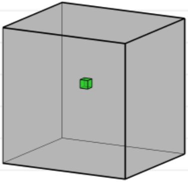
AlexNet (2012)

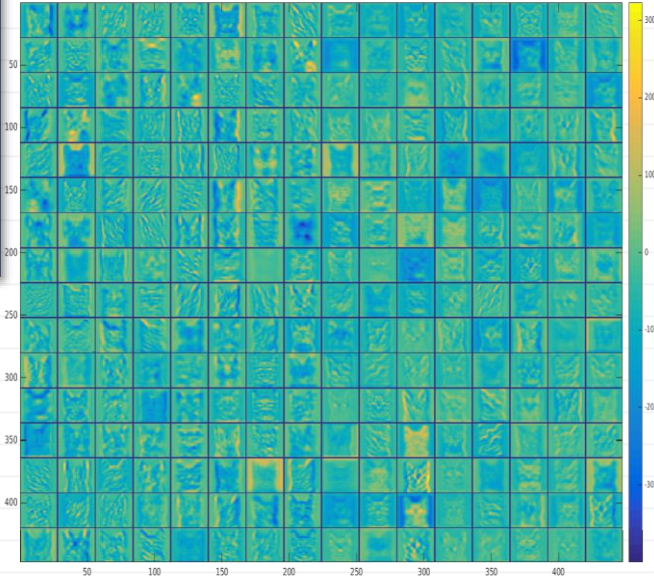


- .."CaffeNet" variant
- 5 conv layers (11x11,5x5,3x3,3x3,3x3)
- 60M parameters
- Learns in 3-5 days on a GPU
- Faster than subsequent architectures

[Krizhevsky et al. 2012]

Idea 1: neuron maps





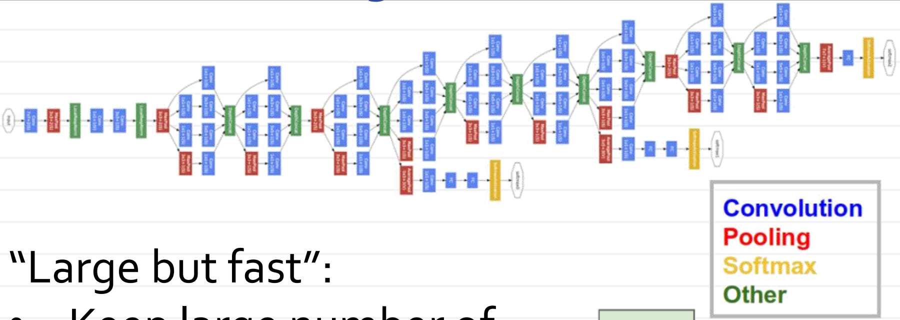
VGGNet (2014)

- upto 16 conv. layers
- All filters are 3x3
- balances load between layers
- ~140M params
- Stagewise training (before batch norms)
- The highest performance among chain-like models

[Simonyan & Zisserman, 2014]

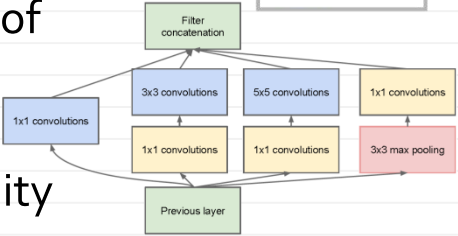
B	C	D	E
weight ayers	16 weight layers	16 weight layers	19 weight layers
(224 × 224 RGB image)			
nv3-64 **nv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool			
v3-128 **v3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool			
v3-256 v3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool			
v3-512 v3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool			
v3-512 v3-512	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			

GoogleNet (2014)



"Large but fast":

- Keep large number of maps
- apply convolutions only to dimensionality reduced stacks



[Szegedy et al. 2014]

ResNet (2015)

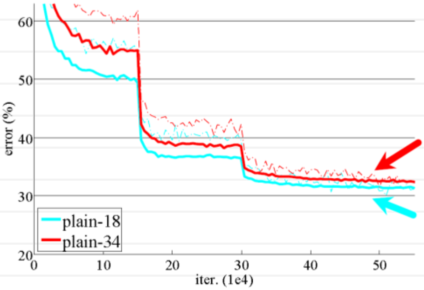
AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Simply deepening
does not work:

ImageNet-1000

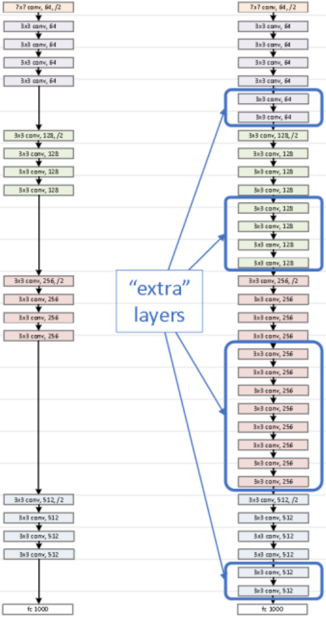


34-layer

18-layer

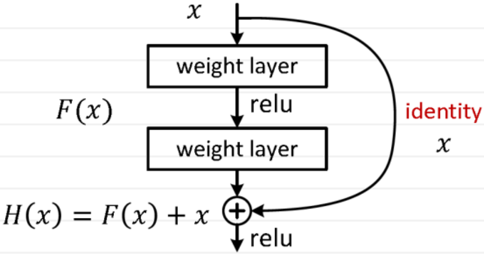
[He et al. 2015]

ResNet (2015)



Q: How to ensure that the training error does not go up?

A: shortcuts



Learn F(x) instead of H(x)

[He et al. 2015]

1x1 Convolution

A variant of convolutional layer with 1x1 spatial filter size that is very important in modern architectures:

$$V(x,y,t) = \sum_{s=1}^{S} K(1,1,s,t) \cdot U(x,y,s)$$

- The 4-dim kernel turns into a 2D-matrix
- No spatial propagation (has to be done by other layers)
- We can think that we apply the same small fully-connected layer at each location independently (originally 1x1 conv was called “network-in-network”)

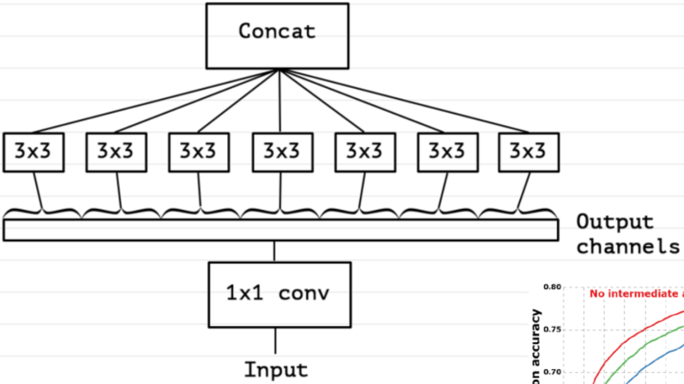
Depthwise-separate Convolution

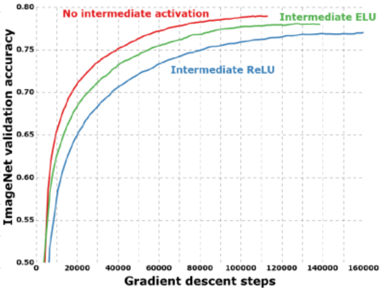
A variant of convolutional layer with 1x1 spatial filter size that is very important in modern architectures:

$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} K(i-x+\delta, j-y+\delta, t) \cdot U(i,j,t)$$

- No propagation across channels (has to be done by other layers)
- S times less operations and parameters than standard convolution

Xception (2016)

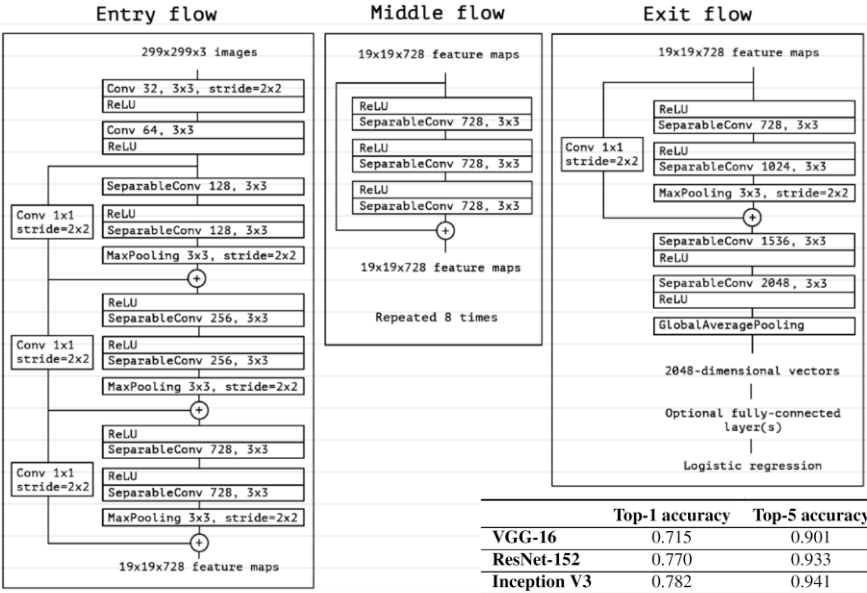




Idea: split spatial and cross-channel convolutions altogether (no non-linearity needed)

[Chollet CVPR17]

Xception (2016)



	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

[Chollet CVPR17]

MobileNet (2017)

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

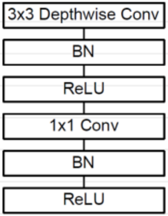


Table 2. Resource Per Layer Type

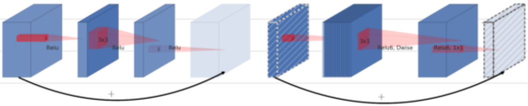
Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

[Howard et al. Arxiv17]

MobileNet v2 (2018)

- Expand the number of layers (to give more work to spatial convolutions)
- Do not use non-linearities in the expansion:

(a) Residual block (b) Inverted residual block



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

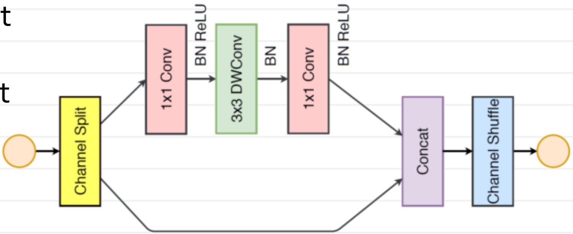
expansion repetitions

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

[Sandler et al. CVPR18]

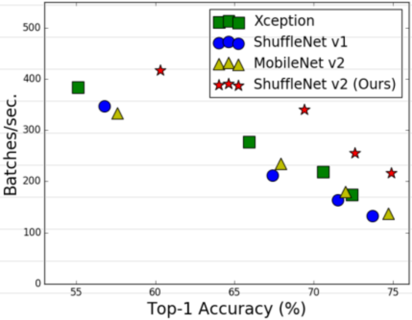
ShuffleNet v2 (2018)

- Process/mix only the first half of all channels
- Second half is kept intact
- Shuffle after each block



Layer	Output size	KSize	Stride	Repeat	Output channels 0.5×	1×	1.5×	2×
Image	224×224				3	3	3	3
Conv1 MaxPool	112×112 56×56	3×3 3×3	2 2	1	24	24	24	24
Stage2	28×28 28×28		2 1	1 3	48	116	176	244
Stage3	14×14 14×14		2 1	1 7	96	232	352	488
Stage4	7×7 7×7		2 1	1 3	192	464	704	976
Conv5	7×7	1×1	1	1	1024	1024	1024	2048
GlobalPool	1×1	7×7						
FC					1000	1000	1000	1000
FLOPs					41M	146M	299M	591M
# of Weights					1.4M	2.3M	3.5M	7.4M

4 variants



[Ma et al. ECCV18]

Recap

- Convolutional networks are the most popular and influential model in modern deep learning
- Convolutional layer is a special type of a multiplicative one with greatly reduced number of parameters
- Different ways to compute convolutions exist
- Good ConvNet architectures have been discovered

Bibliography

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard,Lawrence D. Jackel:
Handwritten Digit Recognition with a Back-Propagation Network. NIPS 1989: 396-404

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton:
ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012: 1106-1114

Joan Bruna, Wojciech Zaremba, Arthur Szlam, Yann LeCun:
Spectral Networks and Locally Connected Networks on Graphs. ICLR 2014

Kumar Chellapilla and Sidd Puri and Patrice Simard, High Performance Convolutional Neural Networks for Document Processing

Karen Simonyan, Andrea Vedaldi, Andrew Zisserman:
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.CoRR abs/1312.6034 (2013)

Bibliography

Karen Simonyan, Andrew Zisserman:
Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR abs/1409.1556 (2014)

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan,Vincent Vanhoucke, Andrew Rabinovich:
Going deeper with convolutions. CVPR 2015: 1-9

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun:
Deep Residual Learning for Image Recognition. CVPR 2016

François Chollet: Xception: Deep Learning with Depthwise Separable Convolutions. CVPR 2017: 1800-1807

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam:
MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR abs/1704.04861 (2017)

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen:
MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR 2018: 4510-4520

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun:
ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. ECCV (14) 2018: 122-138