Network Structures

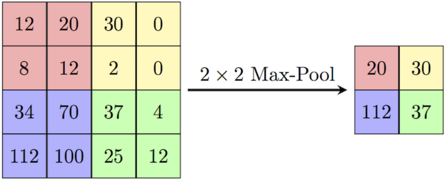
Xiaogang Wang

xgwang@ee.cuhk.edu.hk

February 18, 2019

Max pooling and strided convolution

- Both max pooling and strided convolution are constantly used to decrease spatial dimension of feature maps

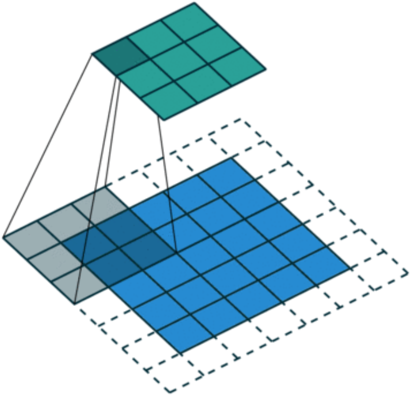


Max pooling with 2×2 kernel and stride 2

Max pooling and strided convolution

- Both max pooling and strided convolution are constantly used to decrease spatial dimension of feature maps

Strided convolution with 3×3 kernel and stride 2



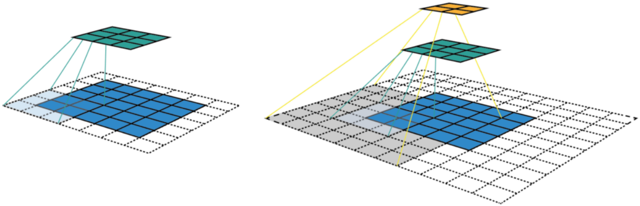
Feature map size and receptive field size

- Output feature maps can be calculated with the following formula

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

where n_{in} and n_{out} are the number of channels of the input and output feature maps, p is the padding size, s is the stride size, k is the convolution kernel size.

- The **receptive field** of a feature can be briefly defined as the region in the input image pixel space that the feature is calculated from



Two consecutive convolution with kernel size $k = 3 \times 3$, padding size $p = 1 \times 1$, stride $s = 2 \times 2$.

Feature map size and receptive field size

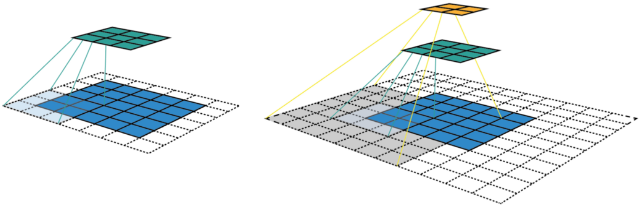
- Output feature maps can be calculated with the following formula

$$j_{out} = j_{in} \times s$$
$$r_{out} = r_{in} + (k-1) \times j_{in}$$

where j is the jump in the output feature map, r is the receptive field size

- For very first input to a network, we always have $r_0 = 1$ and $j_0 = 1$
- Given the previous example, we have

$$r_1 = r_0 + (k-1) \times j_0 = 1 + (3-1) \times 1 = 3, j_1 = j_0 \times 2 = 2$$
$$r_2 = r_1 + (k-1) \times j_1 = 3 + 2 \times 2 = 7, j_2 = j_1 \times 2 = 4$$



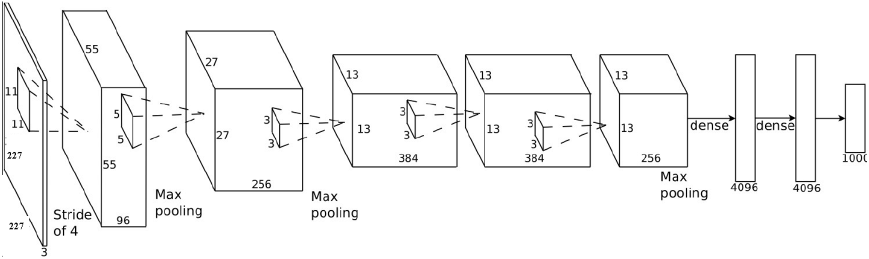
Two consecutive convolution with kernel size $k = 3 \times 3$, padding size $p = 1 \times 1$, stride $s = 2 \times 2$.

Different CNN structures for image classification

- AlexNet
- Clarifai
- Overfeat
- VGG
- Network-in-network
- GoogLeNet
- ResNet

Model architecture-AlexNet Krizhevsky 2012

- 5 convolutional layers and 2 fully connected layers for learning features.
- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000
- 650000 neurons, 60000000 parameters, and 630000000 connections



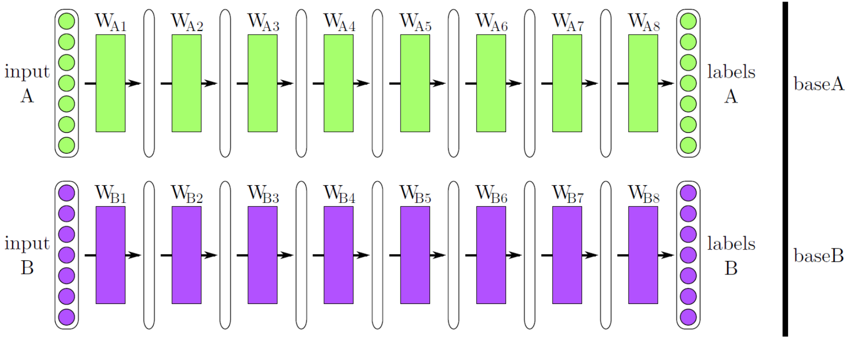
(Krizhevsky NIPS 2014)

How transferable are features in CNN networks?

- (Yosinski et al. NIPS'14) investigate transferability of features by CNNs
- The transferability of features by CNN is affected by
  - Higher layer neurons are more specific to original tasks
  - Layers within a CNN network might be fragilely co-adapted
- Initializing with transferred features can improve generalization after substantial fine-tuning on a new task

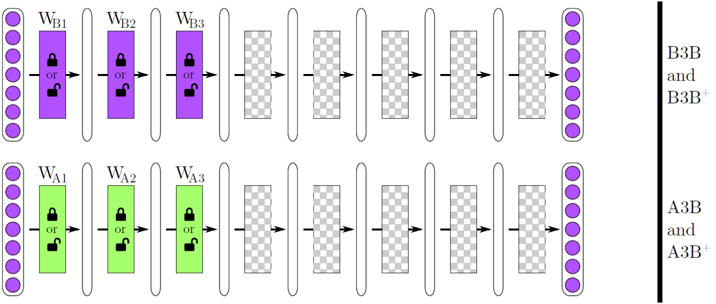
Base tasks

- ImageNet are divied into two groups of 500 classes, A and B
- Two 8-layer AlexNets, baseA and baseB, are trained on the two groups, respectively



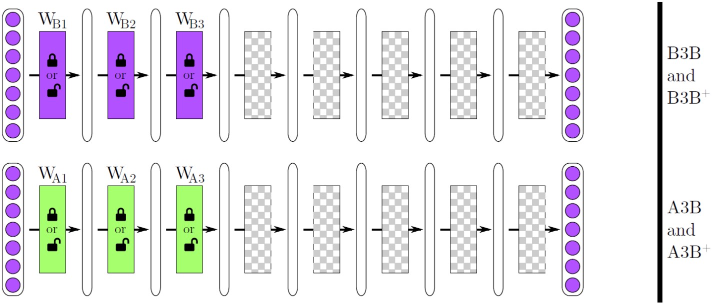
Transfer and selffer networks

- A *selffer* network BnB: the first n layers are copied from baseB and frozen. The other higher layers are initialized randomly and trained on dataset B. This is the control for transfer network
- A *transfer* network AnB: the first n layers are copied from baseA and frozen. The other higher layers are initialized randomly and trained toward dataset B

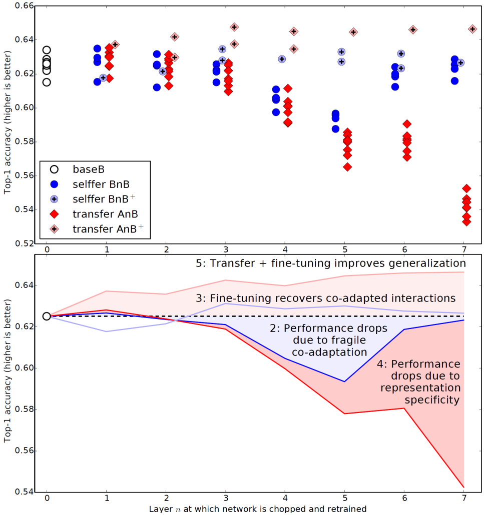


Transfer and selffer networks (cont'd)

- A *selffer* network BnB+: just like BnB, but where all layers learn
- A *transfer* network AnB+: just like AnB, but where all layers learn



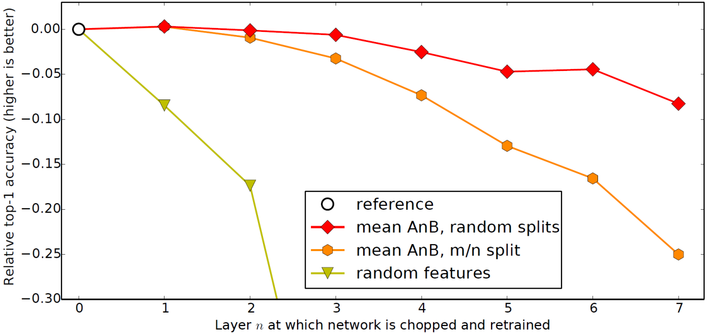
Results



Top-1 accuracy (higher is better)
0.66
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0 1 2 3 4 5 6 7
baseB
selffer BnB
selffer BnB$^+$
transfer AnB
transfer AnB$^+$
5: Transfer + fine-tuning improves generalization
3: Fine-tuning recovers co-adapted interactions
2: Performance drops due to fragile co-adaptation
4: Performance drops due to representation specificity
Layer n at which network is chopped and retrained

Dissimilar datasets

- Divide ImageNet into man-made objects A (449 classes) and natural objects B (551 classes)
- The transferability of features decreases as the distance between the base task and target task increases



Investigate components of CNNs

- Filter size
- Filter (channel) number
- Stride
- Dimensionality of fully connected layers
- Data augmentation
- Model averaging

Investigate components of CNNs (cont'd)

- (Chatfield et al. BMVC'14) pre-train on ImageNet and fine-tune on PASCAL VOC 2007
- Different architectures
  - mAP: CNN-S > (marginally) CNN-M > (∼%2.5) CNN-F
- Different data augmentation
  - No augmentation
  - Flipping (almost no improvement)
  - Smaller dimension downsized to 256, cropping 224 × 224 patches from the center and 4 corners, flipping (∼ 3% improvement)

Arch.	conv1	conv2	conv3	conv4	conv5	full6	full7	full8	
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max	Fast similar to AlexNet
CNN-M	96x7x7 st. 2, pad 0 LRN, x2 pool	256x5x5 st. 2, pad 1 LRN, x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max	Medium similar to Clarifai model
CNN-S	96x7x7 st. 2, pad 0 LRN, x3 pool	256x5x5 st. 1, pad 1 x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x3 pool	4096 drop- out	4096 drop- out	1000 soft- max	Slow similar to OverFeat Accurate model

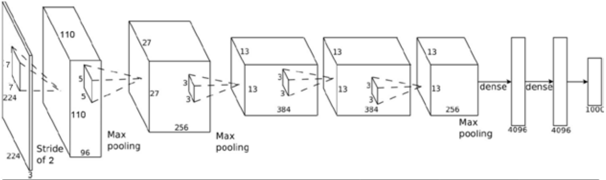
(Chatfield et al. BMVC 2014)

Investigate components of CNNs (cont'd)

- Gray-scale vs. color ($\sim$ 3% drop)
- Decrease the number of nodes in FC7
  - to 2048 (surprisingly, marginally better)
  - to 1024 (marginally better)
  - to 128 ($\sim$ 2% drop but 32x smaller feature)
- Change the softmax regression loss to ranking hinge loss
  - $w_c\phi(I_{pos}) > w_c\phi(I_{neg}) + 1 - \xi$ (ξ is a slack variable)
  - $\sim$ 2.7% improvement
  - Note, $\mathcal{L}_2$ normalising features account for $\sim$ 5% of accuracy for VOC 2007
- On ILSVRC-2012, the CNN-S achieved a top-5 error rate of 13.1%
  - CNN-F: 16.7%
  - CNN-M: 13.7%
  - AlexNet: 17%

Model architecture-Clarifai

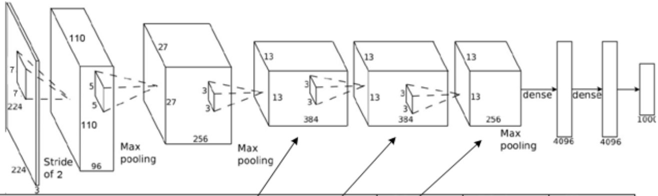
- Winner of ILSVRC 2013
- Max-pooling layers follow first, second, and fifth convolutional layers
- 11×11 to 7×7, stride 4 to 2 in 1st layer (increasing resolution of feature maps)
- Other settings are the same as AlexNet
- reduce the error by 2%.



Error %	Val Top-1	Val Top-5	Test Top-5
(Gunji et al., 2012)	-	-	26.2
(Krizhevsky et al., 2012), 1 convnet	40.7	18.2	−−
1 convnet for Clarifai	38.4	16.5	−−

Model architecture-Clarifai further investigation

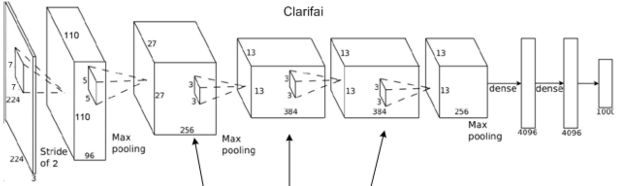
- More maps in the convolutional layers leads to small improvement.
- Model averaging leads to improvement (random initialization).



Error %	Val Top-1	Val Top-5	Test Top-5
(Gunji et al., 2012)	-	-	26.2
(Krizhevsky et al., 2012), 1 AlexNet	40.7	18.2	– –
1 convnet for Clarifai	38.4	16.5	– –
5 convnets for Clarifai (a)	36.7	15.3	15.3
1 convnet for Clarifai but with layers 3,4,5: 512,1024,512 maps – (b)	37.5	16.0	16.1
6 convnets, (a) & (b) combined	**36.0**	**14.7**	**14.8**

Model architecture-Overfeat

- Less pooling and more filters (384 => 512 for conv3 and 384=>1024 for conv4/5).



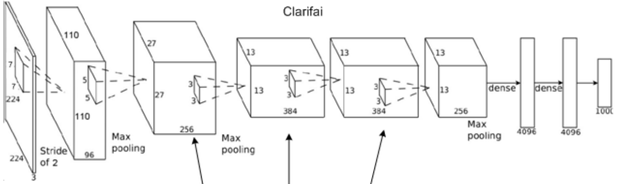
Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

Overfeat

	top-5 error (%)		
	Clarifai	Overfeat-5	Overfeat-7
Without data augmentation	16.5	16.97	14.18

Model architecture-Overfeat

- With data augmentation, more complex model has better performance.



Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

Overfeat

	top-5 error (%)		
	Clarifai	Overfeat-5	Overfeat-7
With data augmentation	14.76	13.52	11.97
Without data augmentation	16.5	16.97	14.18

Model architecture-the devil of details

- CNN-F: similar to AlexNet, but less channels in conv3-5.
- CNN-S: the most complex one.
- CNN-M 2048: replace the 4096 features in fc7 by 2048 features. Makes little difference.
- Data augmentation. The input image is downsized so that the smallest dimension is equal to 256 pixels. Then 224×224 crops are extracted from the four corners and the centre of the image.

ILSVRC-2012	(top-5 error)
(a) Clarifai 1 ConvNet	16.0
(b) CNN F	16.7
(c) CNN M	13.7
(d) CNN M 2048	13.5
(e) CNN S	**13.1**

Arch.	conv1	conv2	conv3	conv4	conv5	full6	full7	full8
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max
CNN-M	96x7x7 st. 2, pad 0 LRN, x2 pool	256x5x5 st. 2, pad 1 LRN, x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max
CNN-S	96x7x7 st. 2, pad 0 LRN, x3 pool	256x5x5 st. 1, pad 1 x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x3 pool	4096 drop- out	4096 drop- out	1000 soft- max
Clarifai	96x7x7 st. 2, LRN,x2 pool	256x5x5 st. 2, pad1 LRN,x2 pool	384x3x3 st. 1,pad1	384x3x3 st. 1,pad1	256x3x3 st. 1,pad1	4096 drop	4096 drop	4096 drop

Model architecture-very deep CNN

- The deep model VGG in 2014.
- Apply 3×3 filter for all layers.
- 11 layers (A) to 19 layers (E).

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					

Model architecture- very deep CNN

- The deep model VGG in 2014.
- Better to have deeper layers. 11 layers (A) => 16 layers (D).
- From 16 layers (D) to 19 layers (E), accuracy does not improve.

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

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**

Model architecture- very deep CNN

- Scale jittering at the training time.
- The crop size is fixed to 224×224.
- S: the smallest side of an isotropically-rescaled training image.
- Scale jittering at the training time: [256; 512]: randomly select S to be within [256 512].
- LRN: local response normalisation. A-LRN does not improve on A.

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

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**

Model architecture- very deep CNN

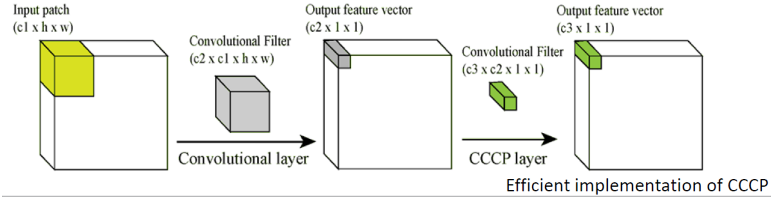
- Multi-scale averaging at the testing time.
- The crop size is fixed to 224×224.
- Q: the smallest side of an isotropically-rescaled testing image.
- Running a model over several rescaled versions of a test image (corresponding to different Q), followed by averaging the resulting class posteriors. Improves accuracy (25.5 => 24.8).

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

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

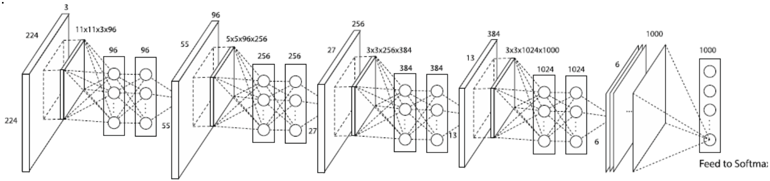
Model architecture- Network in Network

- Use 1×1 filters after each convolutional layer.



Model architecture- Network in Network

- Remove the two fully connected layers (fc6, fc7) of the AlexNet but add NIN into the AlexNet.



	Parameter Number	Performance	Time to train (GTX Titan)
AlexNet	60 Million (230 Megabytes)	40.7% (Top 1)	8 days
NIN	7.5 Million (29 Megabytes)	39.2% (Top 1)	4 days

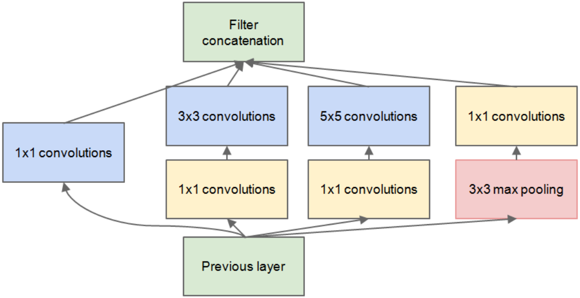
Model architecture- GoogleNet

- Inspired by the good performance of NIN.

Google™

Model architecture- GoogleNet

- Inception model.
- Variable filter sizes to capture different visual patterns of different sizes. Enforce sparse connection between previous layer and output.
- The 1×1 convolutions are used for reducing the number of maps from the previous layer.



Model architecture- GoogleNet

- Based on inception model.
- Cascade of inception models.
- Widths of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.

Model architecture- GoogleNet

- Parameters.

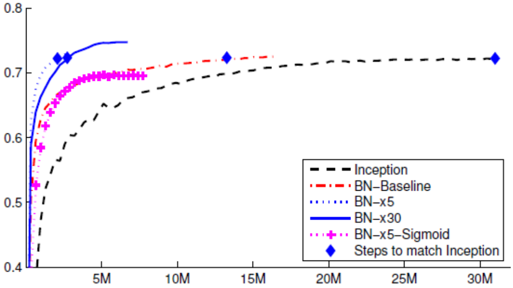
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

GoogleNet-v2/BN-Inception

- The advantages of Batch Normalization (BN) layer
  - Higher learning rate can be used.
  - The need for Dropout can be reduced.
- Main differences from GoogleNet-v1
  - 5×5 convolution layers are converted to two consecutive 3×3 convolution layers with up to 128 filters
  - Adopt the BN layer after each convolution layer.
  - During training, moving average is used to calculate the mean and variance of the BN layers
  - During testing, the mean and variance are calculated using the entire training set in a layer-by-layer manner

GoogleNet-v2/BN-Inception

- Inception vs. BN-Baseline: using BN can improve the training speed significantly
- BN-x5 & BN-x30: the initial learning rate can be increased largely to improve the training speed even better
- BN-x5-Sigmoid: saturation problem by Sigmoid can be a kind of removed

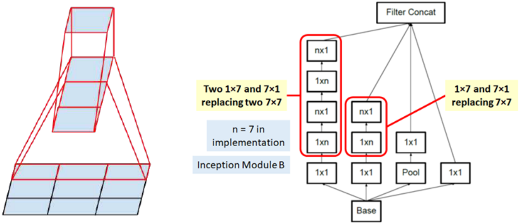


Performance on ImageNet

Inception: Inception-v1 without BN
BN-Baseline: Inception with BN
BN-x5: Initial learning rate is increased by a factor of 5 to 0.0075
BN-x30: Initial learning rate is increased by a factor of 30 to 0.045
BN-x5-Sigmoid: BN-x5 but with Sigmoid

GoogleNet-v3

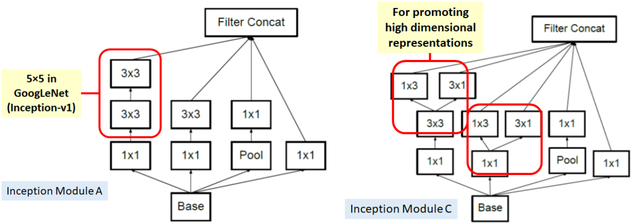
- Factorization was introduced in convolution layer as shown above to further reduce the dimensionality, so as to reduce the overfitting problem
- By using 3×3 filter, number of parameters = $3 \times 3 = 9$
- By using 3×1 and 1×3 filters, number of parameters = $3 \times 1 + 1 \times 3 = 6$
  Number of parameters is reduced by 33%



3×3 conv becomes 1×3 and 3×1 convs (Left), 7×7 conv becomes 1×7 and 7×1 convs (Right)

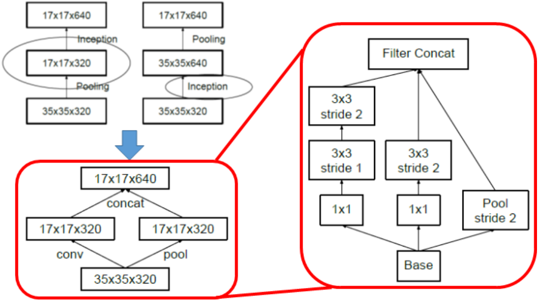
GoogleNet-v3

- Three types of inception modules (A, B, C)



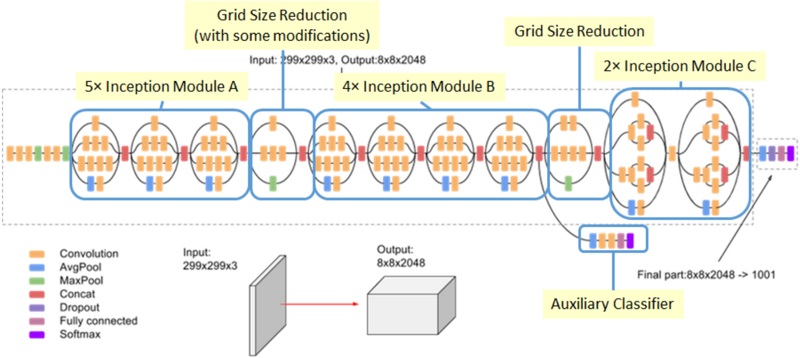
GoogleNet-v3

- Conventionally, such as AlexNet and VGGNet, the feature map downsizing is done by max pooling
- The drawback is either too greedy by max pooling followed by conv layer, or too expensive by conv layer followed by max pooling
- Half of feature maps are done by conv with stride 2. Half of feature maps are obtained by max pooling. These 2 sets of feature maps are concatenated



GoogleNet-v3

- GoogleNet-v3/Inception-v3 architecture

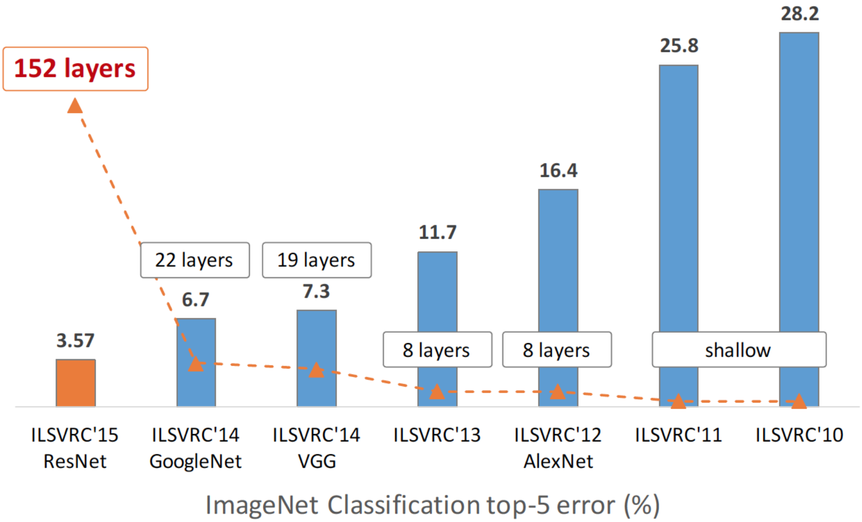


Inception-v3 Architecture (Batch Norm and ReLU are used after Conv)

ResNets @ ILSVRC & COCO 2015 Competitions

- 1st places in all five main tracks
  - ImageNet Classification: ‘Ultra-deep’ 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

Roadmap of Network Depth



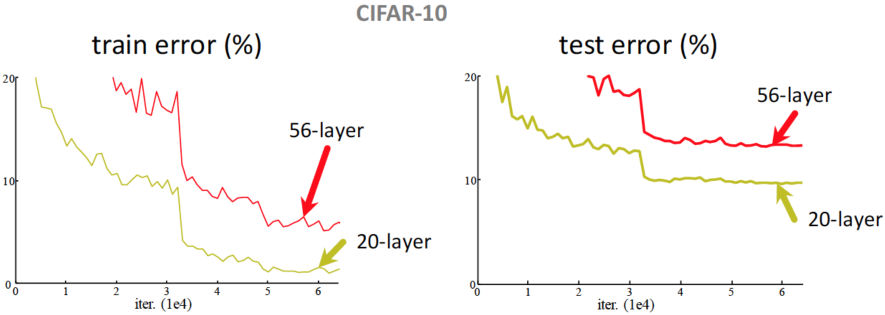
Going deeper

Bear the following in mind:

- Batch normalization. [Sergey Ioffe, Christian Szegedy. ICML 2015]

Is learning better networks as simple as stacking more layers?

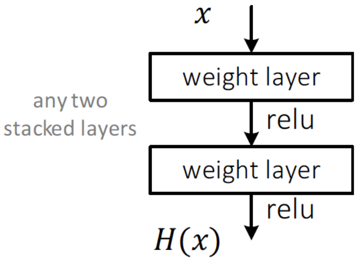
Simply stacking more layers



- *Plain* nets: stacking 3x3 conv layers.
- 56-layer net has **higher training error** and test error than 20-layer net.

Deep Residual Learning

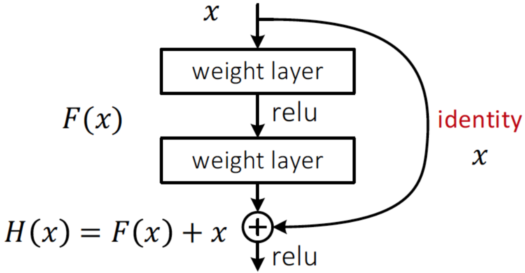
Plain net:



$H(x)$ is any desired mapping.
Let these two conv (weight) layers fit H(x).

Deep Residual Learning

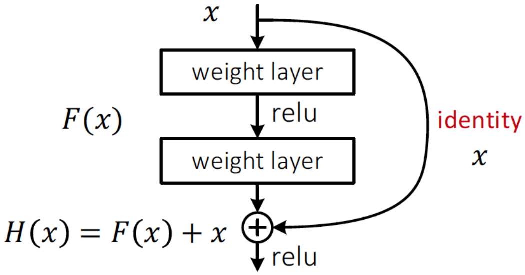
Residual net:



$H(x)$ is any desired mapping.
Let these two conv (weight) layers fit H(x).
Let these two conv (weight) layers fit F(x), where $F(x) = H(x) - x$.

Deep Residual Learning

Residual net:



$F(x)$ is a residual mapping w.r.t. identity.

- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

Network Structure

Basic design: VGG style

- all 3×3 conv
- no FC layer, no dropout

Training details:

- Trained from scratch
- Use batch normalization
- Standard hyper-parameters & augmentation

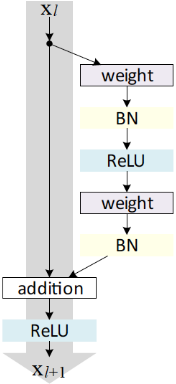
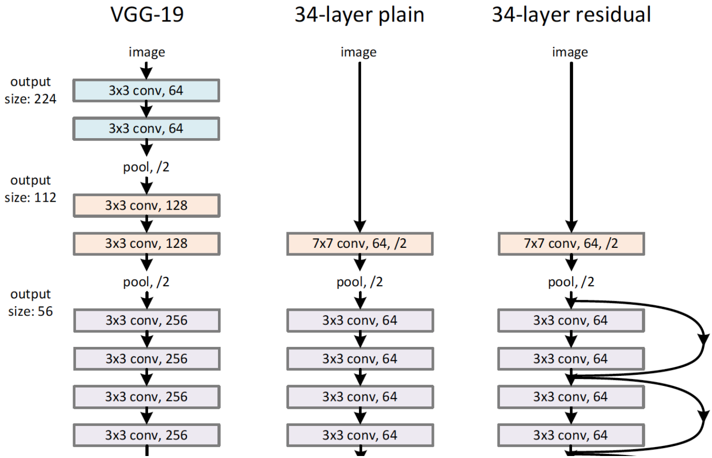


Figure: Basic residual block.

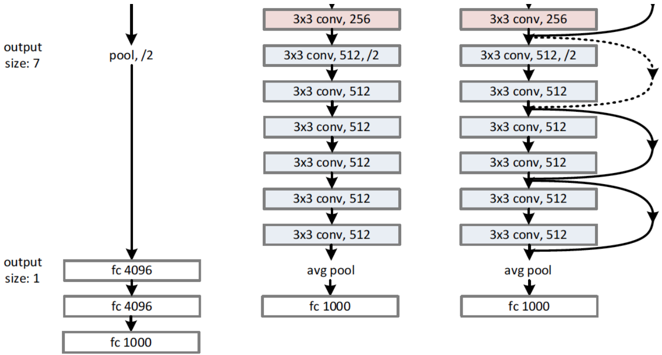
Network Structure

Detailed ResNet structure (rightmost) for ImageNet 2015 entry: (part1)



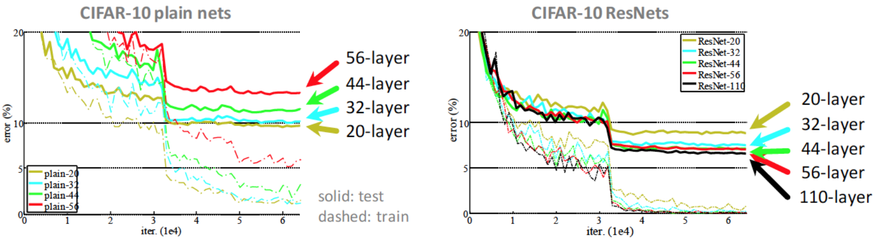
Network Structure

Detailed ResNet structure (rightmost) for ImageNet 2015 entry: (part2)



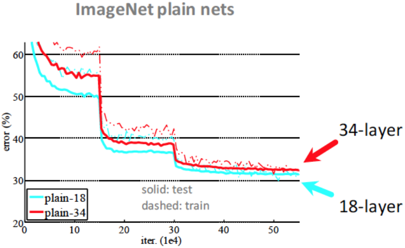
The dotted shortcuts increase channel dimensions.

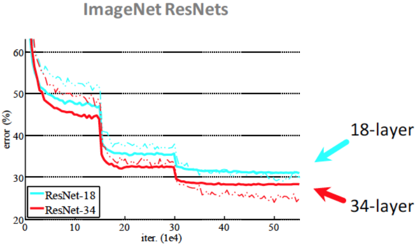
CIFAR-10 experiments



Deep ResNets can be trained without difficulties.
Deeper ResNets have **lower training error**, and also lower test error.

ImageNet experiments





Deep ResNets can be trained without difficulties.
Deeper ResNets have **lower training error**, and also lower test error.

Extension and Resource

- Residual Networks Behave Like Ensembles of Relatively Shallow Networks, NIPS 2016.
- Comparison among ResNet, Highway Network, DenseNet. A blog post here. Another one.
- ResNet code: [Model available] [Torch implementation]

Roadmap of Network Structure

arXiv 2016
Inception-ResNet model
C. Szegedy et al.
GoogleNet -> Inception -> Inception_v4
Top-5: 3.1

CVPR 2016
Residual Network
K. He et al.
VGG -> PReLU (20 layers) -> ResNet (269 layers)
Top-5: 4.49

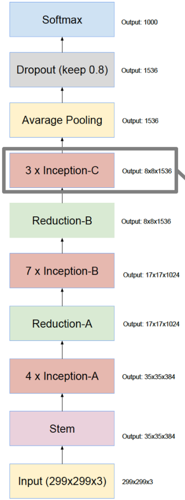
ICLR 2015, CVPR 2015
VGG, GoogLeNet
K. Simonyan; C. Szegedy
Classification top-5 error: 9.33, 9.15

2012 - 2014
Overfeat, Clarifi, Net-in-Net, etc.
Variants of AlexNet

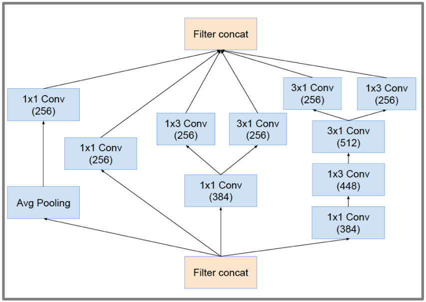
NIPS 2012
AlexNet
A. Krizhevsky et al.
Milestone. First time to apply CNN in large-scale dataset (top-5: 15.3).

Inception-v4 model

- A more uniform simplified architecture and more inception modules than Inception-v3



Inception-v4 network

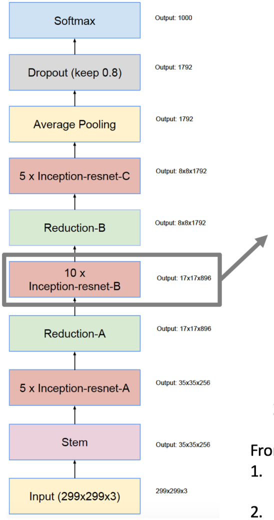


Zoom-in description of Inception-C block.

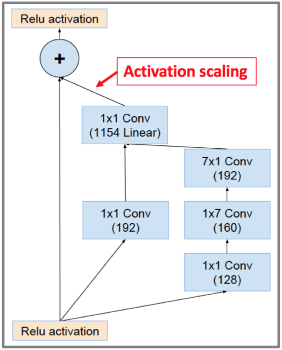
Compared with the original GoogleNet, it has more convolution outputs with smaller filter size before feature concatenation.

Inception-ResNet-v2 model

- A shortcut connection at the left of each module. Inception-ResNet-v2 was training much faster and reached slightly better final accuracy than Inception-v4.



Inception-Resnet v2



Zoom-in description of Inception-resnet-B block.

From empirical evidence:

1. Training with residual connections accelerates the training of Inception networks significantly;
2. Scaling down residuals before adding them to the subsequent layer's activation stabilizes training.

Experiment results

Single model evaluated on ILSVRC CLS 2012 validation set.

Network	**Top-1 Error**	**Top-5 Error**
BN-Inception [6]	25.2%	7.8%
Inception-v3 [15]	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet-151 [5]	dense	19.4%	4.5%
Inception-v3 [15]	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

DenseNet

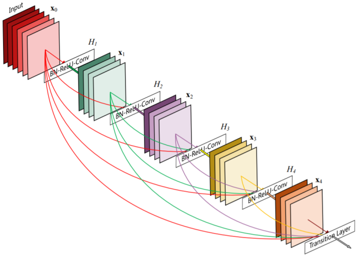
- ResNet solve the gradient vanishing problem by converting the feature mapping equation with **identity addition**

$$x_l = H_l(x_{l-1}) \quad \rightarrow \quad x_l = H_l(x_{l-1}) + x_{l-1}$$

- DenseNets do not sum the output feature maps of the layer with the incoming feature maps but **concatenate** them

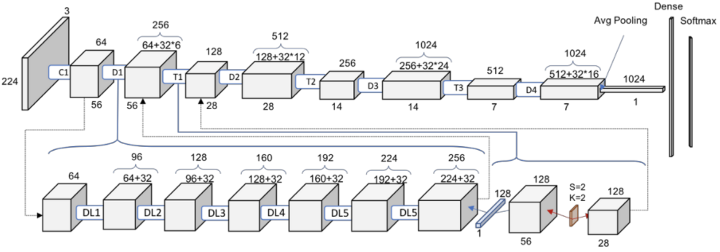
$$x_l = H_l([x_0, x_1, \cdots, x_{l-1}])$$

- Every layer has access to its preceding feature maps, and therefore, to the collective knowledge



DenseNet

- DenseNets are divided into **Dense Blocks**, where the spatial dimensions of the feature maps remains constant within a block, but the number of filters changes between them.
- The feature volume within a dense block remains constant
- There is a transition block follows every dense block, which has 1×1 convolution that halves the number of feature maps followed by a 2×2 pooling with a stride of 2
- The volume and the feature maps are halved after every transition block



Dense-121. Dx: Dense Block x. Tx: Transition Block x.

DenseNet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Different DenseNet structures

Reading materials

- A. Krizhevsky, L. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Proc. NIPS, 2012.
- M. Ranzato, “Neural Networks,” tutorial at CVPR 2013.
- K. Chatfield, K. Simonyan, A. Vadaldi, and A. Zisserman, “Return of the Devil in the Details: Delving Deep into Convolutional Networks,” BMVC 2014.
- P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” In Proc. Int’l Conf. Learning Representations, 2014.
- K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv:1409.1556, 2014.
- M. Lin, Q.. Chen, and S. Yan, “Network in network,” arXiv:1312.4400v3, 2013.
- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” arXiv:1409.4842, 2014.

Reading materials

- Deep Residual Learning for Image Recognition. K. He, et al. *CVPR 2016*. **Best paper.**
  - Highway and Residual Networks learn Unrolled Iterative Estimation, ICLR 2017.
  - Identity Mappings in Deep Residual Networks. K. He, et al. *ECCV 2016*. Extension discussion of ResNet.
  - Deep Networks with Stochastic Depth. G. Huang, et al. *ECCV 2016*
  - Unsupervised Domain Adaptation with Residual Transfer Networks. *NIPS 2016.*
  - Wide Residual Networks. *BMVC 2016.*
  - Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition. https://arxiv.org/abs/1701.03360.
  - . . .

Reading materials

- Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. https://arxiv.org/abs/1602.07261v2.
    - Rethinking the Inception Architecture for Computer Vision. https://arxiv.org/abs/1512.00567v3.
    - Wide-Residual-Inception Networks for Real-time Object Detection. https://arxiv.org/pdf/1702.01243v1.pdf
    - . . .