ELEG5491: Introduction to Deep Learning

Network Architectures for Image Understanding I

Prof. LI Hongsheng

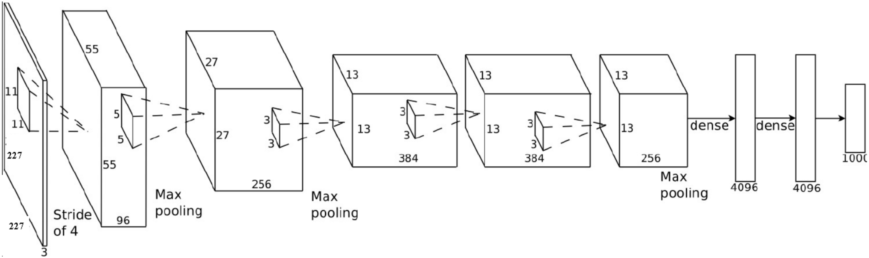
e-mail: hsli@ee.cuhk.edu.hk
Department of Electronic Engineering
The Chinese University of Hong Kong

Feb. 2023

Different CNN structures for image classification

- The evolution of network architectures is mostly driven by image and video understanding, and natural language processing
- We will cover some milestone architectures for image understanding since 2012 (but not all of them, obviously)
- AlexNet
- Clarifai
- Overfeat
- VGG
- Network-in-network
- GoogLeNet
- ResNet
- DenseNet
- ResNeXt
- MobileNet
- ...

- 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 $253,440$, $186,624$, $64,896$, $64,896$, $43,264$, $4,096$, $4,096$, $1,000$
- $650,000$ neurons, $60,000,000$ parameters, and $630,000,000$ connections



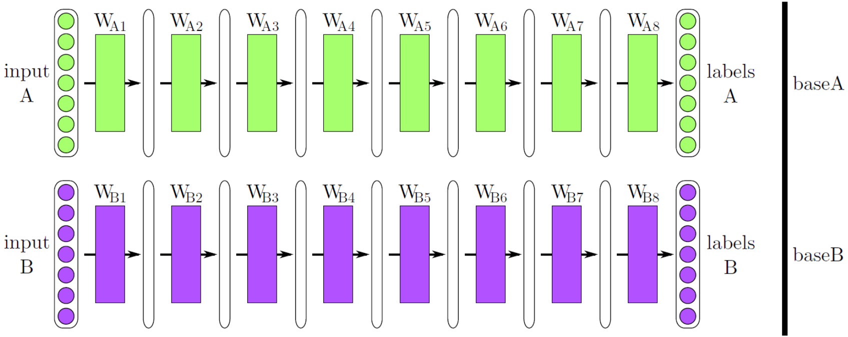
(Krizhevsky NIPS 2014)

How transferable are features in CNN networks?

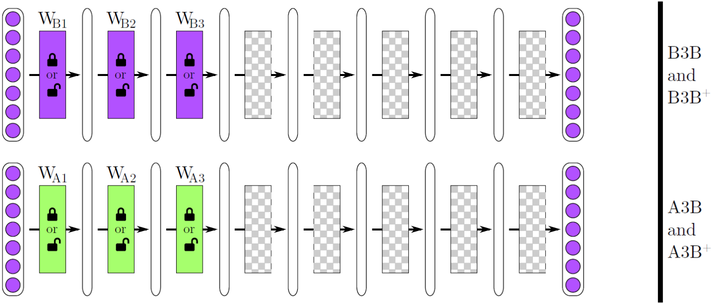
- (Yosinski et al. NIPS'14) investigate transferability of features by CNNs
- The transferability of features by CNN is affected by
  - Higher layer neurons are more specific to original tasks
  - Layers within a CNN network might be fragilely co-adapted
- Initializing with transferred features can improve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

Base tasks

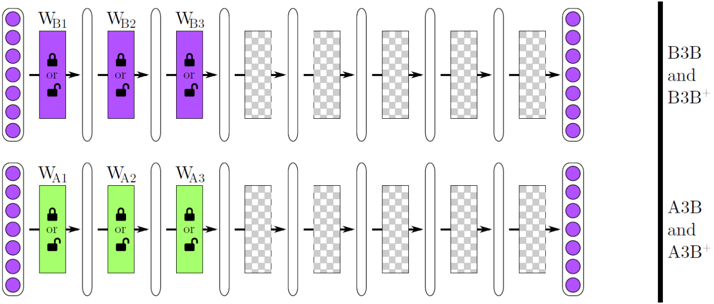
- ImageNet are divied into two groups of 500 classes, A and B
- Two 8-layer AlexNets, baseA and baseB, are trained on the two groups, respectively



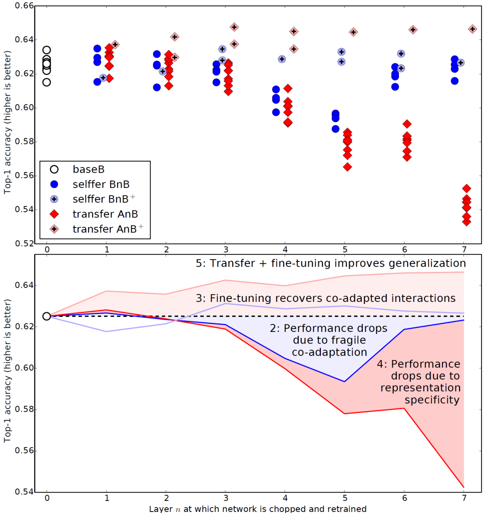
- A *selffer* network BnB: the first n layers are copied from baseB and frozen. The other higher layers are initialized randomly and trained on dataset B. This is the control for transfer network
- A *transfer* network AnB: the first n layers are copied from baseA and frozen. The other higher layers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



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



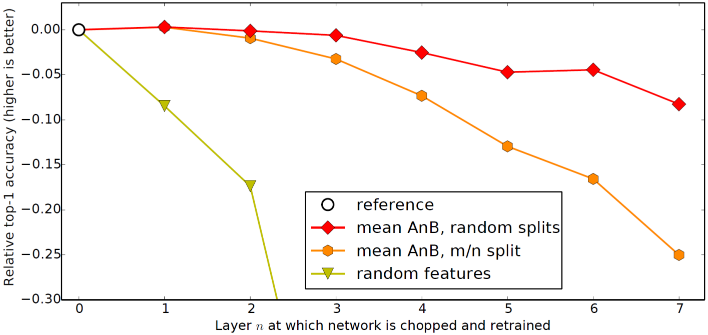
Results



Top-1 accuracy (higher is better)
0.66
0.64
0.62
0.60
0.58
0.56
0.54
0.52
baseB
selffer BnB
selffer BnB$^+$
transfer AnB
transfer AnB$^+$
0 1 2 3 4 5 6 7
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



- Nowadays, ImageNet pre-trained networks are widely used as weight initilization for finetuningthe networks on other tasks

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 ($\sim 3\%$ improvement)

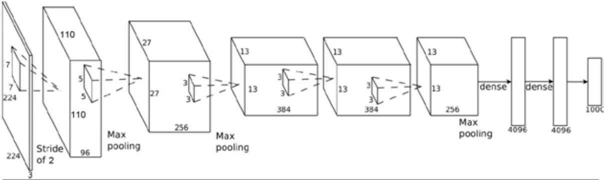
Arch.	conv1	conv2	conv3	conv4	conv5	full6	full7	full8	
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max	Fast similar to AlexNet
CNN-M	96x7x7 st. 2, pad 0 LRN,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

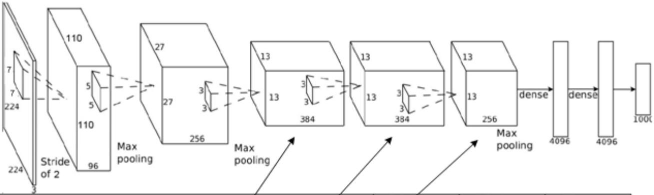
- Gray-scale vs. color ($\sim 3\%$ drop)
- Decrease the number of nodes in FC7
  - to 2048 (surprisingly, marginally better)
  - to 1024 (marginally better)
  - to 128 ($\sim 2\%$ drop but 32x smaller feature)
- Change the softmax regression loss to ranking hinge loss
  - $w_c \phi(I_{pos}) > w_c \phi(I_{neg}) + 1 - \xi$ (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%

- 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	−−

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

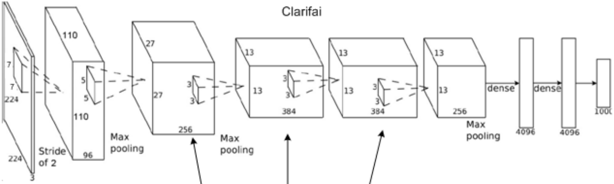


Error %	Val Top-1	Val Top-5	Test Top-5
(Gunji et al., 2012)	-	-	26.2
(Krizhevsky et al., 2012), 1 AlexNet	40.7	18.2	−−
1 convnet for Clarifai	38.4	16.5	−−
5 convnets for Clarifai (a)	36.7	15.3	15.3
1 convnet for Clarifai but with layers 3,4,5: 512,1024,512 maps – (b)	37.5	16.0	16.1
6 convnets, (a) &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

Model architecture-Overfeat

- Less pooling and more filters ($384 \geq 512$ for conv3 and $384 \geq 1024$ for conv4/5).

Clarifai

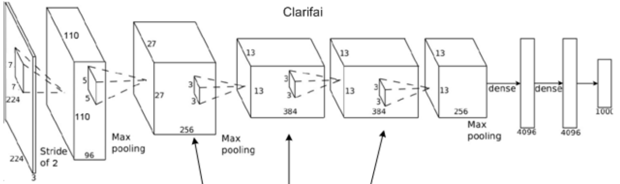


Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding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

- 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

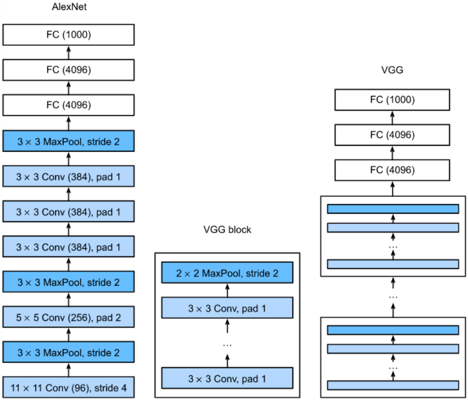
- CNN-F: similar to AlexNet, but less channels in conv3-5.
- CNN-S: the most complex one.
- CNN-M 2048: replace the 4096 features in fc7 by 2048 features. Makes little difference.
- Data augmentation: the input image is downsized so that the smallest dimension is equal to 256 pixels. Then 224×224 crops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

- The deep network VGG was proposed in 2014
- Apply 3×3 filters for all layers
- Introduction of **modular design**: conv blocks only responsible for convolutions and downsampling layers/blocks only responsible for feature map downsampling



Model architecture - VGG Network

- Better to have deeper layers. 11 layers (A) $\rightarrow$ 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 - VGG Network

- 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: randomly select S to be within $[256, 512]$
- LRN (obsolete): 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

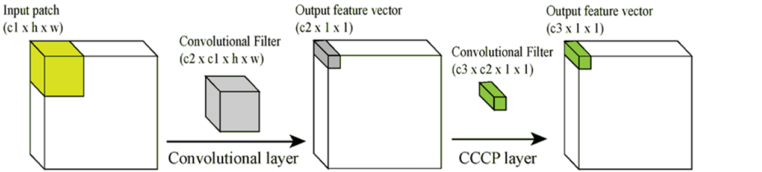
- Multi-scale averaging at the testing time.
- The crop size is fixed to 224×224.
- Q: the smallest side of an isotropically-rescaled testing image.
- Running a model over several rescaled versions of a test image (corresponding to different Q), followed by averaging the resulting class posteriors. Improves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

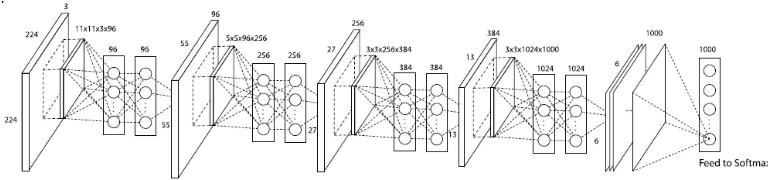
- Use 1×1 filters after each convolutional layer.



Efficient implementation of CCCP

Model architecture - Network in Network

- Remove the two fully connected layers (fc6, fc7) of the AlexNet but add NIN into the AlexNet.
- NIN are just 1×1 (pointwise) convolutions



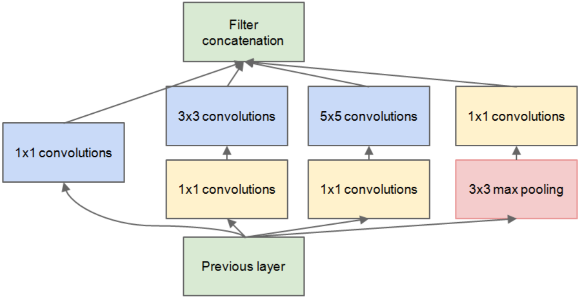
	Parameter Number	Performance	Time to train (GTX Titan)
AlexNet	60 Million (230 Megabytes)	40.7% (Top 1)	8 days
NIN	7.5 Million (29 Megabytes)	39.2% (Top 1)	4 days

- Inspired by the good performance of NIN.

Google™

Model architecture - GoogleNet / Inception-v1

- Inception module is the basic operation module in GoogleNet / Inception-v1
- The 1×1 convolutions are used for reducing the number of feature dimension before the computationally expensive 3×3 and 5×5 convolution
- 1×1, 3×3, 5×5 convolutions and 3×3 max pooling are used to encode different types of features before concatenation



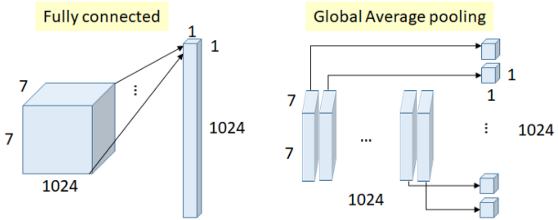
- Previously, fully connected layer are used at the end of network

$$\text{Number of weights (connections)} = 7 \times 7 \times 1024 \times 1024 = 51.3M$$

- In GoogleNet, global average pooling is used nearly at the end of network by averaging each feature map from 7×7 to 1×1

$$\text{Number of weights (connections)} = 0$$

- It is found to improve ImageNet classification accuracy by 0.6% and is less prone to overfit

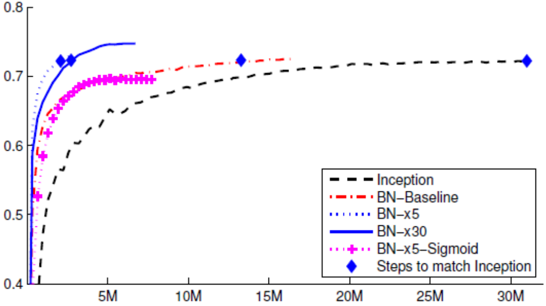


- Based on inception module
- Cascade of inception modules
- Widths of inception modules ranges from 256 filters in bottom modules to 1024 in top inception modules
- There are auxiliary classifiers, which are modeled as intermediate softmax branches for training. Each branch consists of 5×5 global average pooling, $1 \times 1 \times 128$ convolutoin, 128×1024 FC, and 1024×1000 FC, Softmax function
- Weights of the auxiliary classifier: 0.3 and 0.6

- Parameters

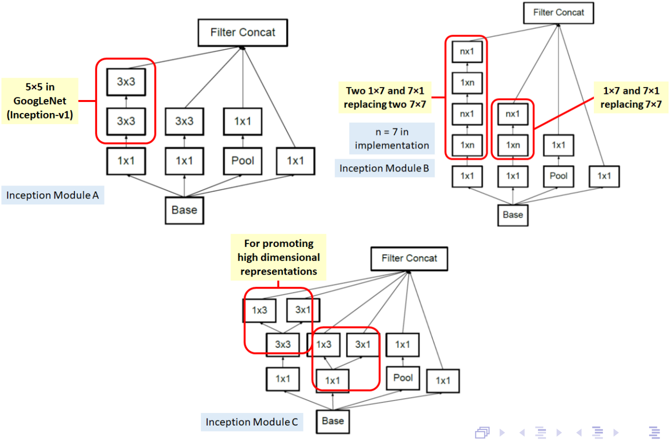
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	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								

- Batch normalize is introduced into Inception-v2
- 5×5 convolution is replaced by two 3×3 convolution for parameter reduction while maintaining the size of the receptive



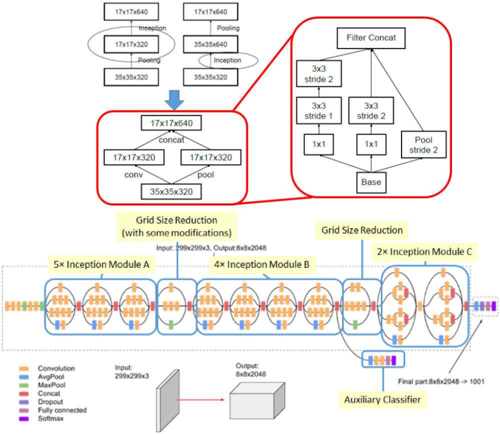
- Inception: Inception-v1 without BN
- BN-Baseline: Inception with BN
- BN-x5: Initial learning rate is increased by a factor of 5 to 0.0075
- BN-x30: Initial learning rate is increased by a factor of 30 to 0.045
- BN-x5-Sigmoid: BN-x5 but with Sigmoid

- **Factorization** was introduced in convolution layer
  - Using 3×1 and 1×3 filters to approximate 3×3 filters, number of parameters decreases from 9 to 6 (33% fewer)
  - Using 7×1 and 1×7 filters to approximate, number of parameters decreases from 49 to 14 (71% fewer)
- Inception A, B, C modules

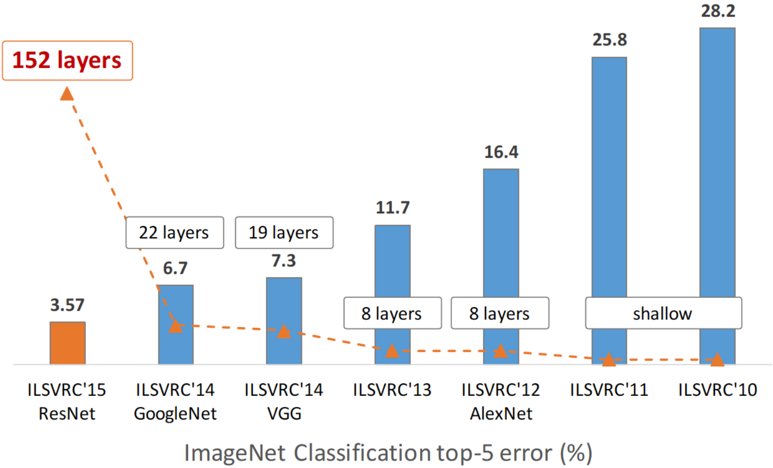


Model architecture - Inception-v3

- Conventionally, in AlexNet and VGGNet, the drawback of downsampling is either too greedy by max pooling followed by convolution layer, or too expensive by convolution layer followed by max pooling
- Efficient grid size reduction in v3: half feature channels are obtained via convolution with a stride 2 and half feature channels are obtained via max pooling



Roadmap of Network Depth



ResNets @ ILSVRC & COCO 2015 Competitions

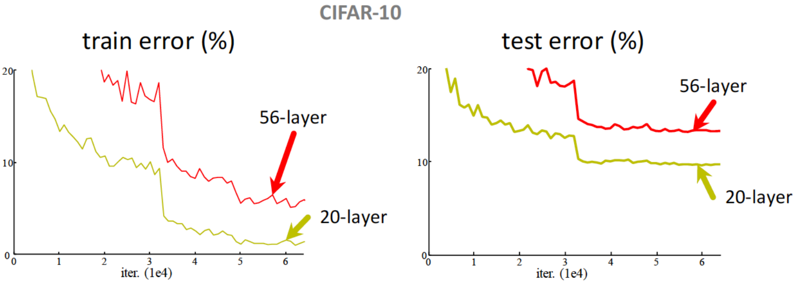
- The milestone network architecture ResNet was introduced in 2015
- It won 1st places in all five main tracks of the ImageNet Challenge
    - ImageNet Classification: 'Ultra-deep' 152-layer nets
    - ImageNet Detection: 16% better than the 2nd
    - ImageNet Localization: 27% better than the 2nd
    - COCO Detection: 11% better than the 2nd
    - COCO Segmentation: 12% better than the 2nd

Bear the following in mind:

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

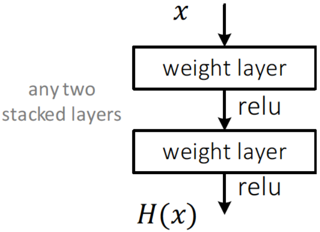
Is learning better networks as simple as stacking more layers?

Simply stacking more layers



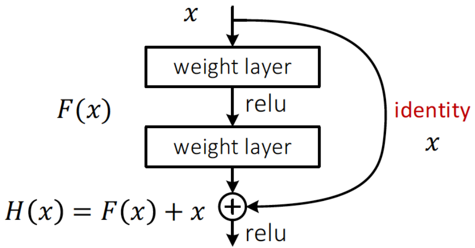
- *Plain* nets: stacking 3x3 conv layers.
- 56-layer net has **higher training error** and test error than 20-layer net.

Plain net:



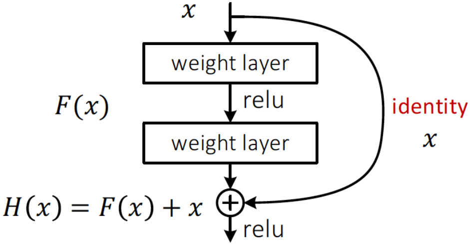
- $H(x)$ is any desired mapping for any two layers
- The learning process generally makes these two convolution (weight) layers fit the mapping $H(x)$

Residual learning block (naive version):



- $H(x)$ is any desired mapping
- Instead of letting the two layers fit $H(x)$, ResNet makes these two conv (weight) layers fit the residual $F(x)$, where $F(x) = H(x) - x$

Residual learning block (naive version):



$F(x)$ 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
- With the identity residual connection, the gradients are very easy to back-propagated back to bottom network layers

Basic design: VGG modular style

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

Training details:

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

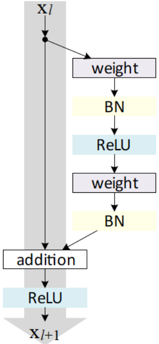
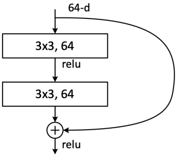
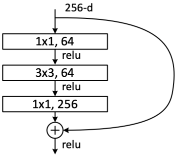


Figure: Orignal residual block in CVPR'16 paper

Building block of ResNet

- Two types of basic residual blocks are used

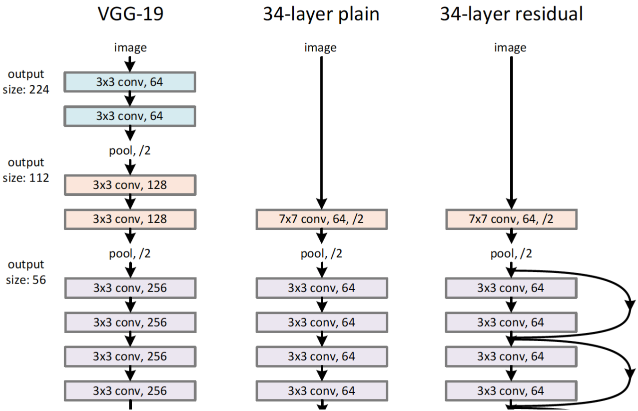




- A shortcut undergoes a 1×1 convolution when the output dimension increases
- Downsampling is achieved by convolution layers that have a stride of 2

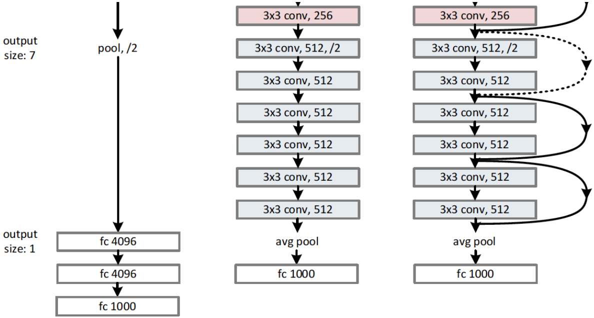
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

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



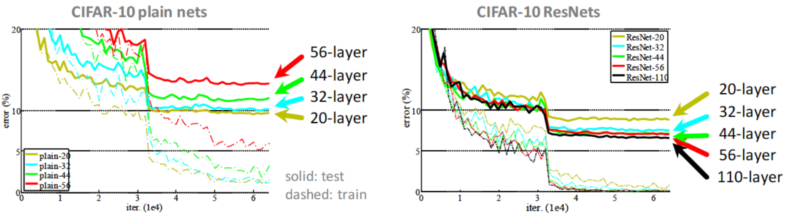
Network Structure

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



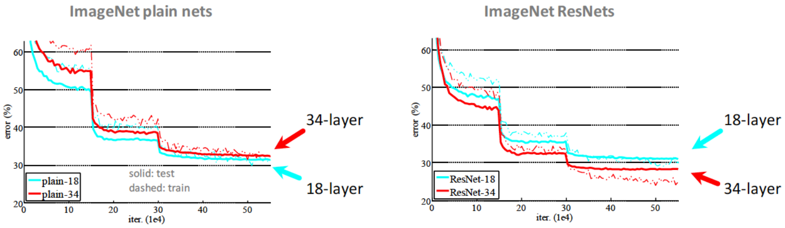
The dotted shortcuts increase channel dimensions.

CIFAR-10 experiments



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

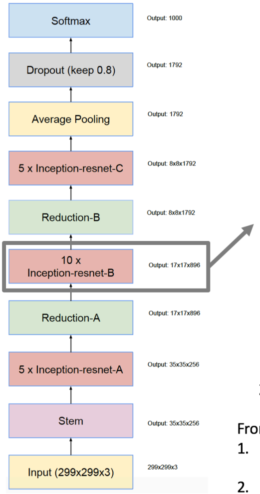
ImageNet experiments



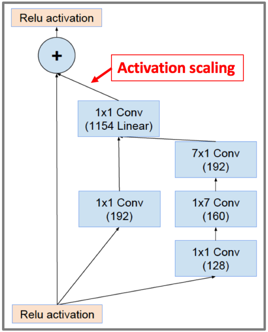
Deep ResNets can be trained without difficulties.
Deeper ResNets have **lower training error**, and also lower test error.

- Three slightly different blocks are tested
- A - The shortcut has identity mapping with extra zero entried padded if the feature dimension increases
- B - A shortcut undergoes a 1×1 convolution when the dimension increases
- C - All shortcuts undergo 1×1 convolutions
- After this inveistigation, the authors decided to make all other ResNet use the B option

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**



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

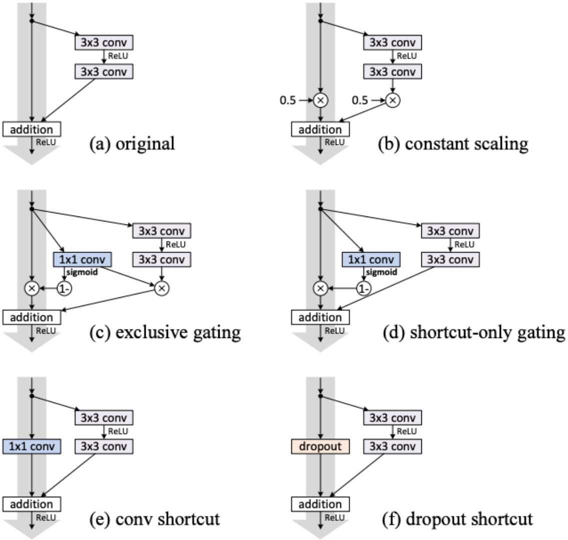
Single model evaluated on ILSVRC CLS 2012 validation set.

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%

Further investigation on residual block design

- Investigation on the function format of the shortcut connections



Further investigation on residual block design

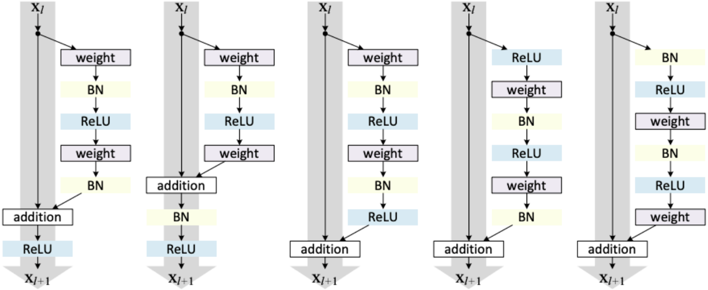
case	Fig.	on shortcut	on $\mathcal{F}$	error (%)	remark
original [1]	Fig. 2(a)	1	1	**6.61**	
constant scaling	Fig. 2(b)	0	1	fail	This is a plain net
		0.5	1	fail	
		0.5	0.5	12.35	frozen gating
exclusive gating	Fig. 2(c)	$1 - g(\mathbf{x})$	$g(\mathbf{x})$	fail	init b_g=0 to -5
		$1 - g(\mathbf{x})$	$g(\mathbf{x})$	8.70	init b_g=-6
		$1 - g(\mathbf{x})$	$g(\mathbf{x})$	9.81	init b_g=-7
shortcut-only gating	Fig. 2(d)	$1 - g(\mathbf{x})$	1	12.86	init b_g=0
		$1 - g(\mathbf{x})$	1	6.91	init b_g=-6
1×1 conv shortcut	Fig. 2(e)	1×1 conv	1	12.22	
dropout shortcut	Fig. 2(f)	dropout 0.5	1	fail	

Figure: CIFAR-10 test set using ResNet-101

- The plain shortcut connections are the most direct paths for the information to effective propagate
- All tested multiplicative manipulations (scaling, gating, 1×1 convolution, and dropout) on the shortcus hamper information propagation

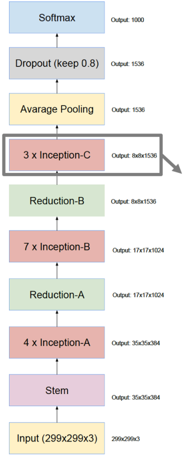
- Investigation on the usage of activation functions

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

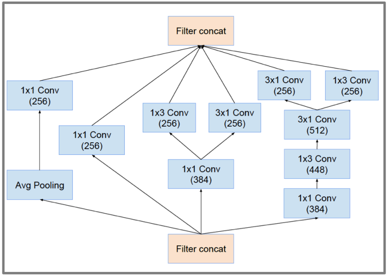


(a) original (b) BN after addition (c) ReLU before addition (d) ReLU-only pre-activation (e) **full pre-activation**

- **BN after activation:** The BN layer alters the signal that passes through the shortcut and impedes information propagation
- **ReLU before addition:** Only non-negative output from $\mathcal{F}(x)$, while a good residual function should take values in $(-\infty, \infty)$
- **Post-activation or pre-activation?** Activation only affects the $\mathcal{F}$ path.
  - Optimization is further eased because f is an identity mapping
  - Including BN in pre-activation improves regularization of the models
  - Pre-activation reduces overfitting (larger training loss but less test error). Presumably caused by BN's regularization effect. In the original design, although BN normalizes the information, it is soon added to the shortcut and the merged signal is not normalized
- However, the searched design in ECCV paper was not widely used. People find it have marginal influence to the final performance
- However, it is an important information that the position of normalization and normalization type actually affects the networks' final performances



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.

Roadmap of Network Structure

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

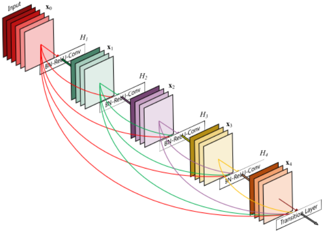
CVPR 2016
Residual Network
K. He et al.
VGG -> PReLU (20 layers) -> ResNet (269 layers)
Top-5: 4.49

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

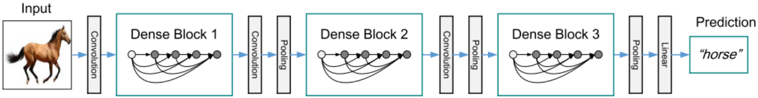
2012 - 2014
Overfeat, Clarifi, Net-in-Net, etc.
Variants of AlexNet

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

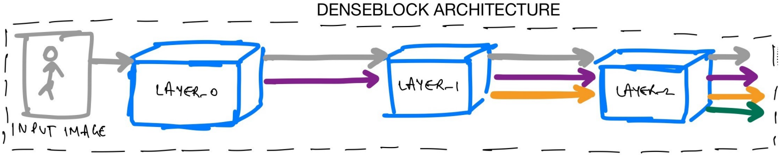
- We know that the residual network uses skip connection to model residual learning
$$\mathbf{x}_l = F_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1}$$
- In **DenseNet** architecture, the key idea is that the connectivity can be represented by concatenation of different features from different layers
$$\mathbf{x}_l = H_l(\mathrm{concat}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{l-1}))$$
- Limitation: it would not be possible to concatenate the feature maps if the size of feature maps is different
- To be able to perform the concatenation operation, we need to make sure that the size of the feature maps that we are concatenating is the same
- To design the DenseNet, only feature maps of the same size are densely connected with concatenation



- The network is divided into multiple densely connected blocks (dense blocks). Inside dense blocks, the feature map size remains the same



- **Convolution + Pooling** outside dense blocks: a bath normalization layer, 1×1 convolution, 2×2 averge pooling (stride 2)
- Within each dense block, each layer's output is connected to all follow-up layers' input



- For ImageNet classification, DenseNet architectures are generally divided into 4 dense blocks

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			

- The DenseNet-121 has $[6, 12, 24, 16]$ layers in the four dense blocks whereas DenseNet-169 has $[6, 12, 32, 32]$ layers in the four blocks

- ResNeXt block

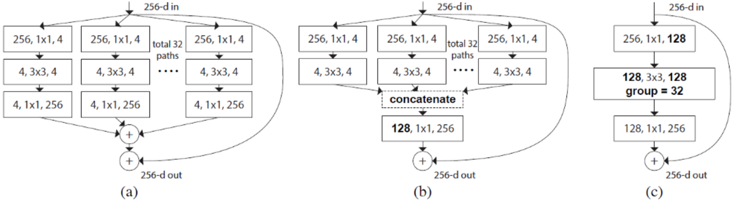


Figure: (a) ResNeXt block. (b) Inception-ResNet block. (c) Residual + Grouped Convolution.

- **Splitting:** the input feature maps are transformed to a series of low-dimensional feature maps with 1×1 convolutions
- **Transforming:** the low-dimensional representation is transformed with efficient 3×3 convolutions to capture spatial context
- **Aggregating:** Convert back to high-dimensional feature maps with 1×1 convolutions and perform feature addition

Model architecture - ResNeXt

- Results of ImageNet classification

Detailed Architecture of ResNet-50 and ResNeXt-50 (32x4d)

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times10^6$	**25.0**$\times10^6$
FLOPs		**4.1**$\times10^9$	**4.2**$\times10^9$

Number of Parameters (Proportional to FLOPs)

$$C\cdot(256\cdot d+3\cdot3\cdot d\cdot d+d\cdot256)$$

Different settings to maintain similar complexity

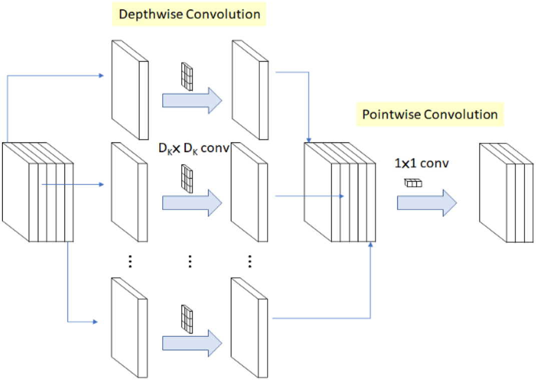
cardinality C	1	2	4	8	32
width of bottleneck d	64	40	24	14	4
width of group conv.	64	80	96	112	128

Comparisons under similar complexity

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

Model architecture - MobileNet-v1

- All previous networks focus on improving classification accuracy. There are another direction of reseasrch that focuses on maximizing the efficiency
- **Depthwise separable convolution:** a depthwise convolution followed by a pointwise (1×1) convolution



- There are 5 input feature dimensions. We will have 5 $D_k \times D_k$ spatial convolutions
- The follow-up 1×1 convolution change the output feature dimension

- M - Input feature channels
- N - Output feature channels
- D_k - Kernel size (side length)
- D_f - Feature map size (side length)
- The computation cost of standard convolution is

$$D_k \cdot D_k \cdot M \cdot N \cdot D_F \cdot D_F$$

- The computational cost of depthwise convolution is

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

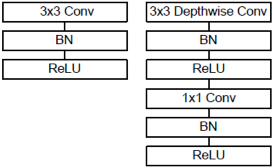
- The computational cost reduction is

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$
$$= \frac{1}{N} + \frac{1}{D_K^2}$$

- When $D_k \times D_k$ is 3×3, 8-9 times less computation can be achieved

Model architecture - MobileNet-v1

- MobileNet block



- MobileNet-v1

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	$1 \times 1 \times 1000$

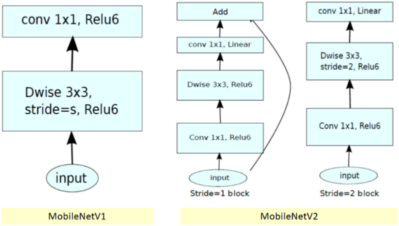
- MobileNet only got 1% loss in accuracy, but the Mult-Adds and parameters can be reduced tremendously

Table 4. Depthwise Separable vs Full Convolution MobileNet

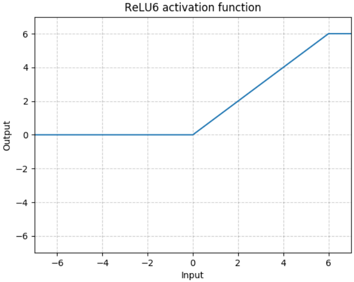
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Model architecture - MobileNet-v2

- The change of basic conv block



- ReLU6 is introduced as $\min(\max(x, 0), 6)$



- Network architecture

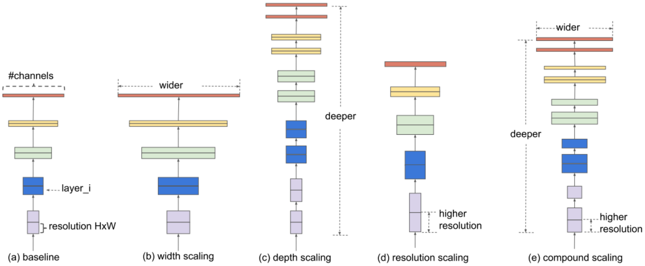
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck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

- Results

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**

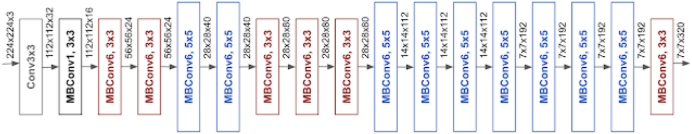
EfficientNet

- Before the EfficientNets came along, the most common way to scale up ConvNets was either by one of three dimensions - depth (number of layers), width (number of channels) or image resolution (image size)
- EfficientNets perform Compound Scaling - that is, balance all the dimensions of the network (width, depth and resolution) by uniformly scaling each one of them using a constant ratio
- Scale all three dimensions while maintaining a balance between all dimensions of the network
- Actually, Compound Scaling only works on existing MobileNet and ResNet



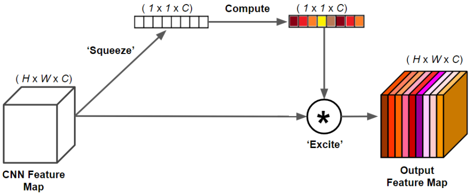
EfficientNet

- it is critical to have a good baseline network. The authors designed a mobile-size baseline network called **EfficientNet-B0**, that works by using a multi-objective neural architecture that optimizes accuracy and FLOPS. The model was inspired by Mnas-Net (an automatical neural architecture search method) and has the following architecture

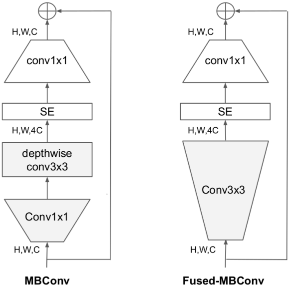


- The building block of this architecture is the mobile inverted bottleneck MBConv that is also called inverted residual block with an additional SE (Squeeze and Excitation) block.

- Sigmoid function is used to scale different channels differently



- EfficientNetv2 uses Fused-MBConv in early stages



- Step 1: Fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ according to the network performance. In particular, we find the best ratios for EfficientNet-B0 are $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$
- FLOPs of a regular convolution operator is proportional to d, w^2, r^2 (dominating in CNNs)
- Constrain $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ such that for any ϕ, the total FLOPs will approximately increase by 2^ϕ

$$
\begin{aligned}
\text{depth: } & d = \alpha^\phi \\
\text{width: } & w = \beta^\phi \\
\text{resolution: } & r = \gamma^\phi \\
\text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
& \alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}
$$

- Step 2: We then fix α, β, γ as constants and scale up baseline network with different ϕ, to obtain EfficientNet-B1 to B7

EfficientNet Results

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

- 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.

References

- 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.
- Szegedy, Christian, et al. Inception-v4, inception-resnet and the impact of residual connections on learning. The AAAI Conference on Artificial Intelligence, 2017.
- Sandler, Mark, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. The IEEE conference on computer vision and pattern recognition, 2018.