Notes on Some Image Classification Methods After ResNet

1. ResNet

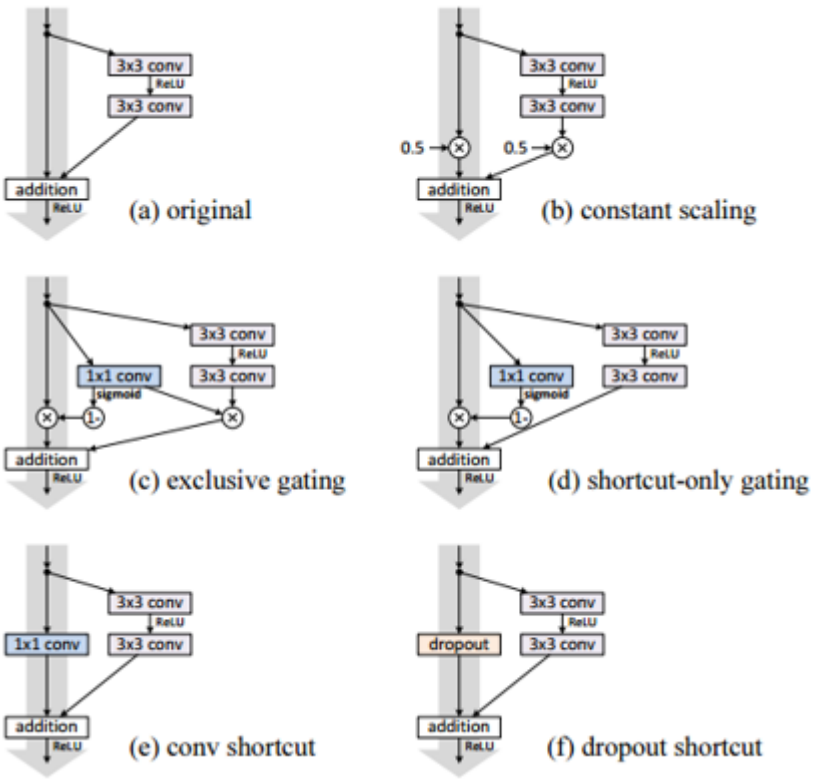
paper: [Deep Residual Learning for Image Recognition *CVPR2016*]

Nothing special to note except that the author consider two options of shortcut (i.e. the skip connections between layers): 1) identity mapping; 2) projection (done by 1×1 convolutions). Experiments shows that **projection shortcuts are not essential** because of its small differences in results from identity mapping.

2. Pre-activation ResNet

paper: [Identity Mappings in Deep Residual Networks *ECCV2016*]

2. 1 Various types of shortcut connections



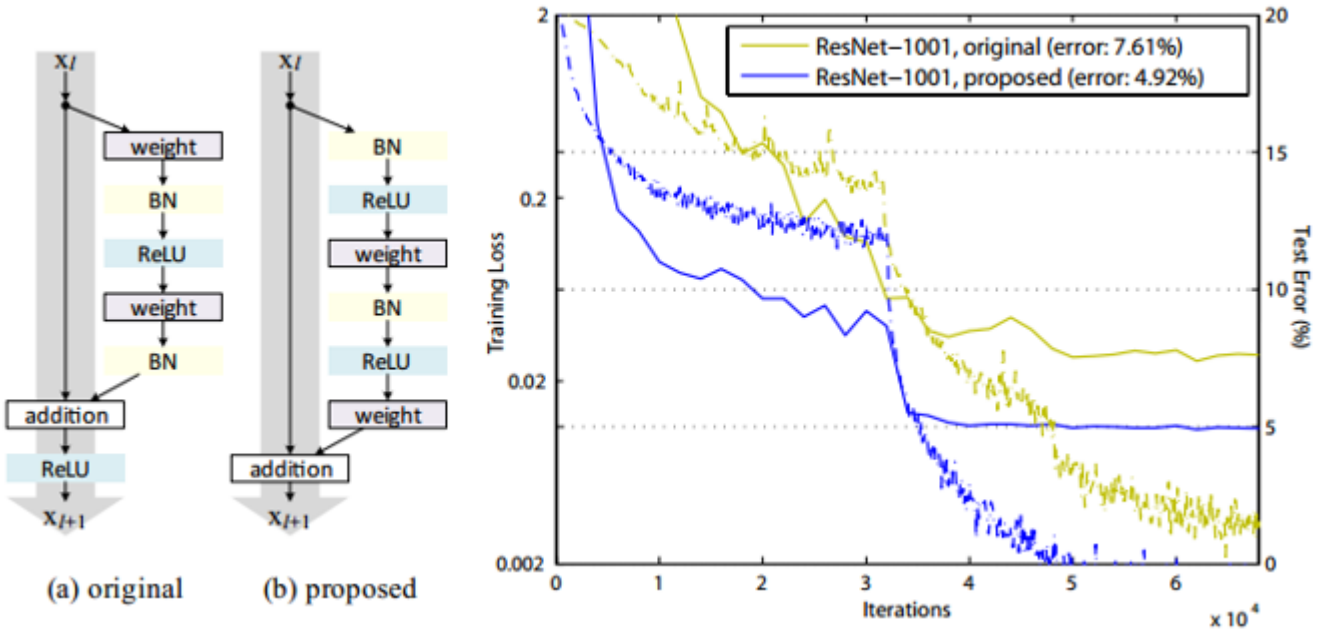
The gating function in (c) and (d) is:

$$g(\mathbf{x}) = \sigma\left(\mathbf{W}_g\mathbf{x} + b_g\right)$$

Results show that identity mapping performs best. These experiments suggest that **keeping a "clean" information path is helpful for easing optimization**. In contrast, multiplicative manipulations (scaling, gating, 1×1 convolutions, and dropout) on the shortcuts can hamper information propagation and lead to optimization problems.

2.2 Pre-activation

Moving BN and ReLu before weighted layers as the figure shows below. This figure also shows the training and testing loss curves of original and pre-act ResNet.



Experiments show that when BN and ReLU are both used as pre-activation, the results are improved by healthy margins. The reason is twofold:

1) Ease of optimization

The general function of residual units is:

$$\mathbf{y}_l = h\left(\mathbf{x}_l\right) + \mathcal{F}\left(\mathbf{x}_l, \mathcal{W}_l\right)$$
$$\mathbf{x}_{l+1} = f\left(\mathbf{y}_l\right)$$

In original ResNet, f is ReLu function that truncates the propagated signal. In pre-act version, f is an identity mapping so the signal can be propagated directly between any two units.

The impact of $f = \mathrm{ReLu}$ is especially prominent when ResNet is deep (e.g. 1001 layers).

2) Reducing overfitting

The author believes that this is caused by BN's regularization effect. In the original Residual Unit, the added signal (i.e. y_l) is not normalized while the inputs to all weight layers have been normalized in the pre-act version.

3. Xception

paper: [Xception: Deep Learning with Depthwise Separable Convolutions *CVPR2017*]

This paper modifies the Inception module with depth-wise convolution.

Depth-wise convolution: a spatial convolution performed independently over each channel of an input, followed by a pointwise convolution, i.e. a 1x1 convolution, projecting the channels output by the depth-wise convolution onto a new channel space.

An “extreme” version of an Inception module using depth-wise convolution can be derived from a simplified Inception module as follows.

Figure 1. A canonical Inception module (Inception V3).

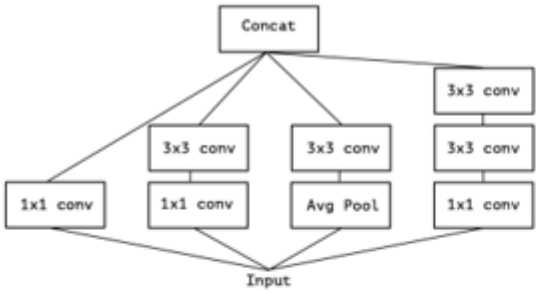


Figure 2. A simplified Inception module.

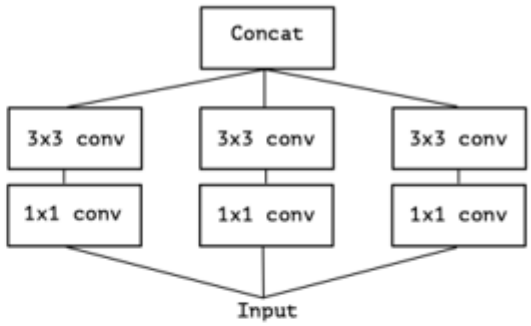


Figure 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

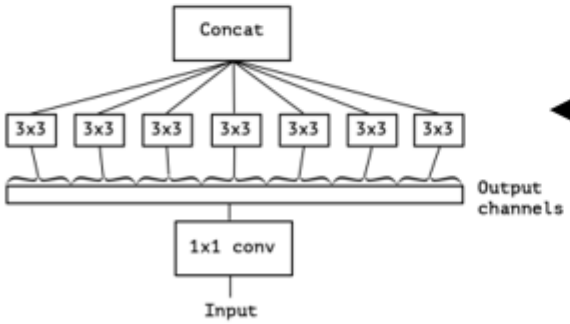
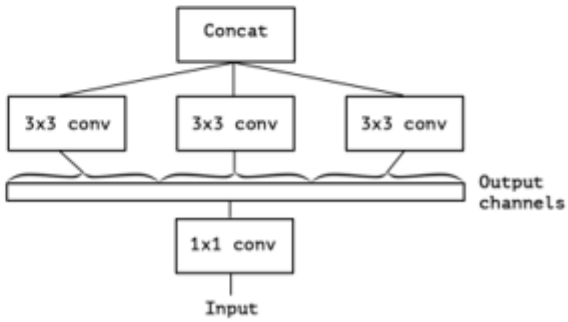
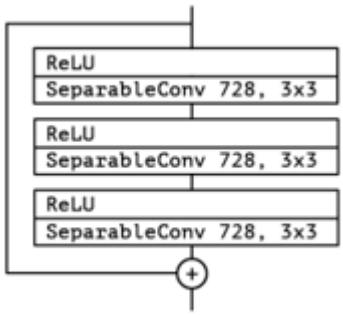


Figure 3. A strictly equivalent reformulation of the simplified Inception module.



The Xception architecture is a linear stack of depthwise separable convolution layers with residual connections.



SeparableConv denotes the operation that perform 1x1 convolution first, followed by a channel-wise spatial convolution. All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).

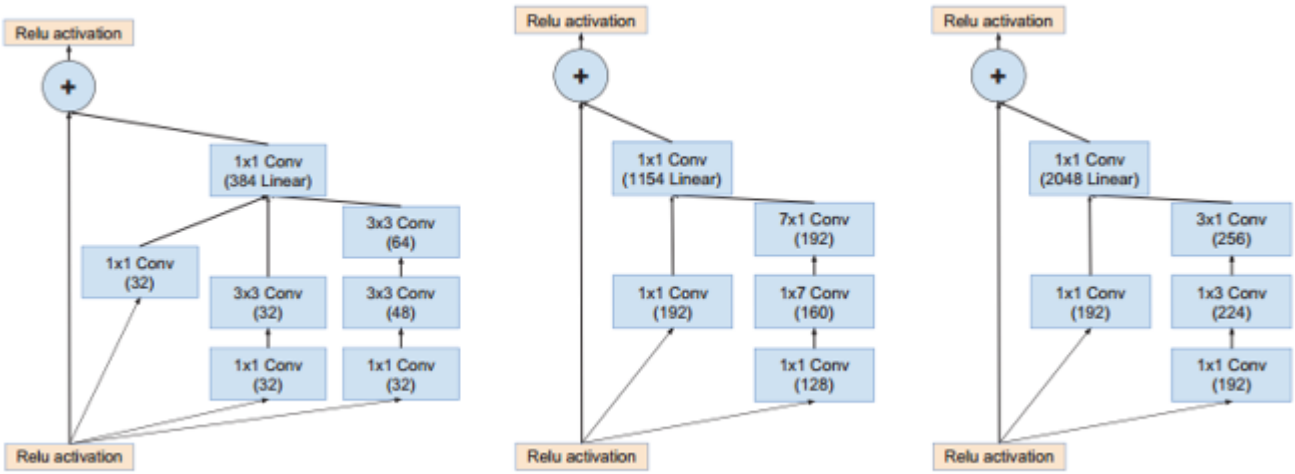
Experiment: mainly compared with Inception V3, the table below shows the results on ImageNet.

	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**

4. Inception-ResNet-v2

paper: [Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning](#) *AAAI2017*

Nothing special to note and the figure below shows the Inception-A, Inception-B and Inception-C in the Inception-ResNet-v2 network.



The table shows single-crop – single-model experimental results. Reported on the non-blacklisted subset of the validation set of ILSVRC 2012.

Network	Crops	Top-1 Error	Top-5 Error
ResNet (He et al. 2015)	10	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

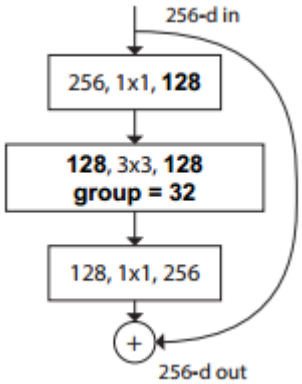
5. ResNeXt

paper: [Aggregated Residual Transformations for Deep Neural Networks](#)
CVPR2017

This paper uses **group convolution** to modify the original ResNet and introduces a new dimension called "cardinality" (i.e. the groups in group convolution).

Note that the depth-wise convolution in *Xception* can be seen as a special case of group convolution in which the number of groups is equal to the number of channels.

A building block of ResNeXt:



ResNeXt-50 architecture compared with ResNet-50, note that the width of the bottleneck (i.e. number of kernels or feature maps) is adjusted in group convolution to preserving complexity:

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	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Ablation experiments on ImageNet-1K:

	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**

Comparisons on ImageNet-1K where model complexity is increased by 2×:

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

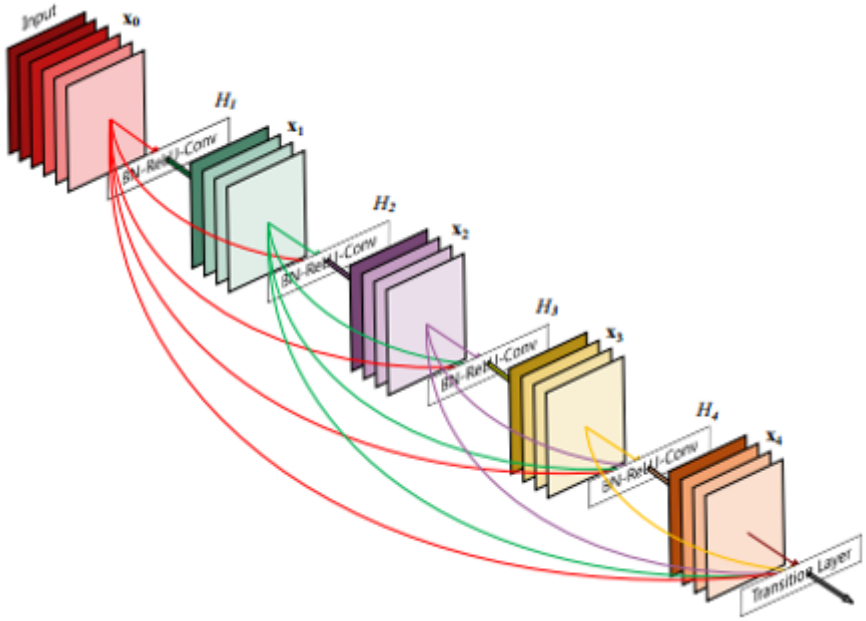
We can see **increasing cardinality C shows much better results than going deeper or wider**.

6. DenseNet

paper: [Densely Connected Convolutional Networks *CVPR2017*]

This paper is CVPR2017 oral. It proposes a new and efficient network, which is seen as the biggest breakthrough after ResNet.

DenseNet connects each layer to every other layer in a feed-forward fashion, hence the network has L(L+1)/2 direct connections totally. A 5-layer dense block is showed below:



DenseNet combines features by **concatenating** them, but not through summation. The major difference between DenseNet and ResNet can be clearly showed by the equations below:

DenseNet: $\mathbf{x}_\ell = H_\ell\left([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]\right)$

ResNet: $\mathbf{x}_\ell = H_\ell\left(\mathbf{x}_{\ell-1}\right) + \mathbf{x}_{\ell-1}$

DenseNet layers are very narrow (e.g., 12 filters per layer) and thus requires fewer parameters. The author believes that a relatively small filters per layer (called "growth rate" k in the paper) is sufficient to obtain good results. One explanation is that the feature maps can be viewed as the global state of the network. The growth rate regulates how much new information each layer contributes to the global state. **The global state,**

once written, can be accessed from everywhere within the network and, unlike in traditional network architectures, there is no need to replicate it from layer to layer.

In other words, each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision.

The figure below is the DenseNet architectures for ImageNet.

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

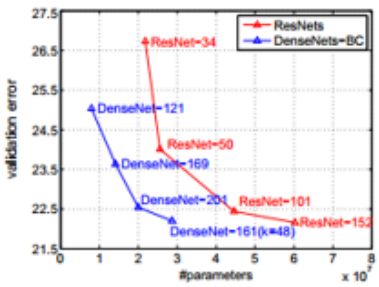
Some compositions of DenseNet are described as follows:

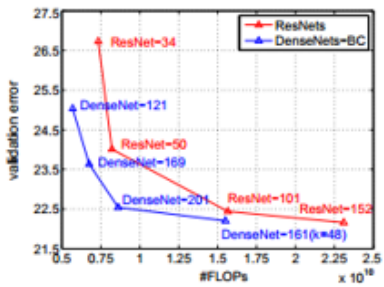
Transition layers: these layers consist of a batch normalization layer and an 1×1 convolutional layer followed by a 2×2 average pooling layer for down-sampling. The whole network is divided into multiple densely connected dense blocks to facilitate down-sampling.

Bottleneck layers: they are 1×1 convolution injected before each 3×3 convolution to reduce the large number of input feature-maps (because of dense connections).

Experiments: on ImageNet validation set

Model	top-1	top-5
DenseNet-121 ($k=32$)	25.02 (23.61)	7.71 (6.66)
DenseNet-169 ($k=32$)	23.80 (22.08)	6.85 (5.92)
DenseNet-201 ($k=32$)	22.58 (21.46)	6.34 (5.54)
DenseNet-161 ($k=48$)	22.33 (20.85)	6.15 (5.30)





It can be indicated that DenseNets utilize parameters more efficiently than ResNets. One positive side-effect of the more efficient use of parameters is a tendency of DenseNets to be less prone to overfitting.