2019.07.25

DenseNet Review

Youngtaek Hong, PhD

- 오늘 리뷰할 논문은 DenseNet으로 잘 알려져 있는 CNN architecture를 다룬 "Densely Connected Convolutional Networks" 이라는 논문입니다.

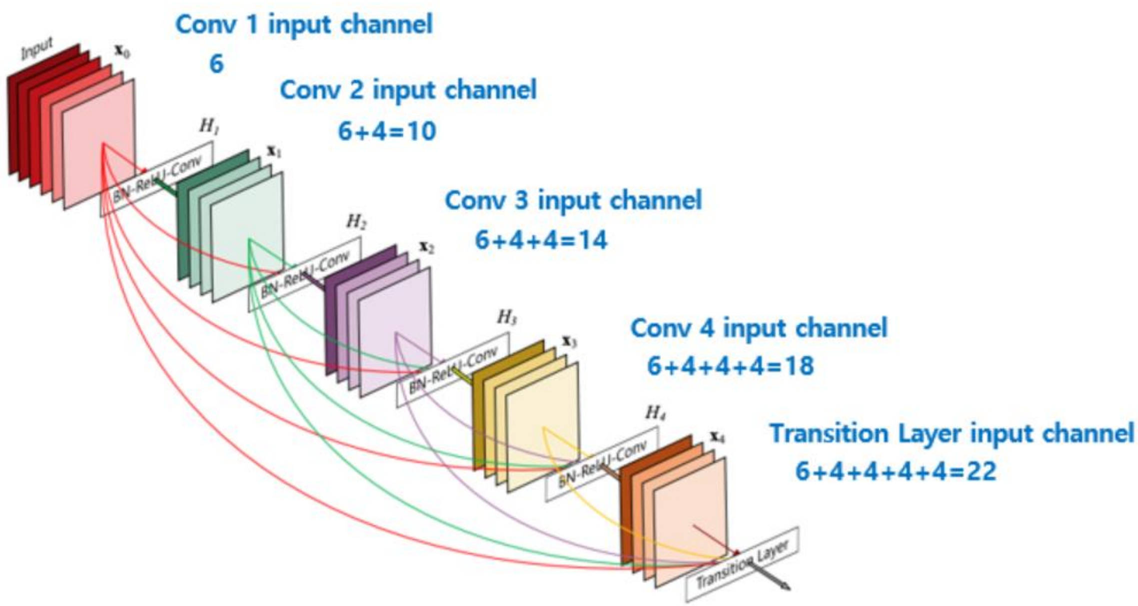


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

[Dense Connectivity]

- 이전 layer들의 feature map을 계속해서 다음 layer의 입력과 연결하는 방식이며 이러한 방식은 ResNet에서도 사용이 되었습니다. 다만 ResNet은 feature map 끼리 **더하기** 를 해주는 방식이었다면 DenseNet은 feature map끼리 **Concatenation** 을 시키는 것이 가장 큰 차이점입니다.

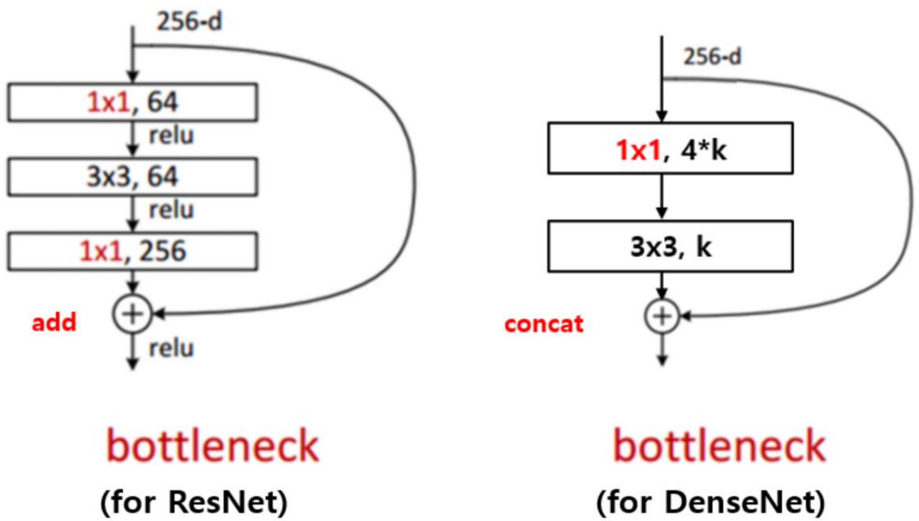
- Vanishing Gradient 개선
- Feature Propagation 강화
- Feature Reuse
- Parameter 수 절약

Growth Rate

- 각 feature map끼리 densely 연결이 되는 구조이다 보니 자칫 feature map의 channel 개수가 많은 경우 계속해서 channel-wise로 concat이 되면서 channel이 많아 질 수 있습니다.
- 그래서 DenseNet에서는 각 layer의 feature map의 channel 개수를 굉장히 작은 값을 사용하며, 이 때 각 layer의 feature map의 channel 개수를 **growth rate(k)** 이라 부릅니다.

Bottleneck Layer

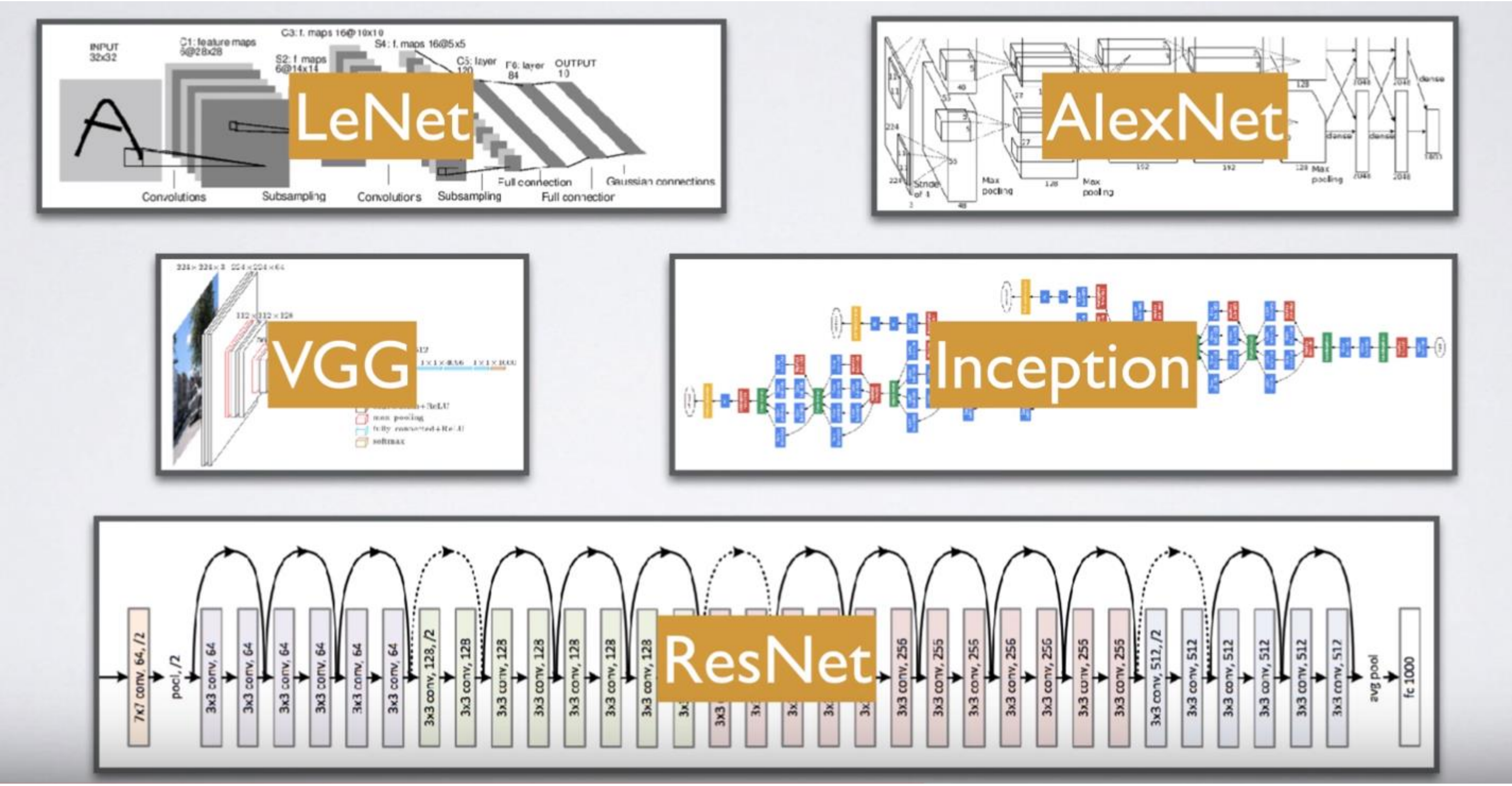
ResNet과 Inception 등에서 사용되는 bottleneck layer의 아이디어는 DenseNet에서도 찾아볼 수 있습니다.



[DenseNet bottleneck layer]

- 3x3 convolution 전에 1x1 convolution을 거쳐서 입력 feature map의 channel 개수를 줄이는 것 까지는 같은데,
- 그 뒤로 다시 입력 feature map의 channel 개수 만큼을 생성하는 대신 growth rate 만큼의 feature map을 생성하는 것이 차이 점
- 이를 통해 computational cost를 줄일 수 있다고 합니다.

Trends of CNNs



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

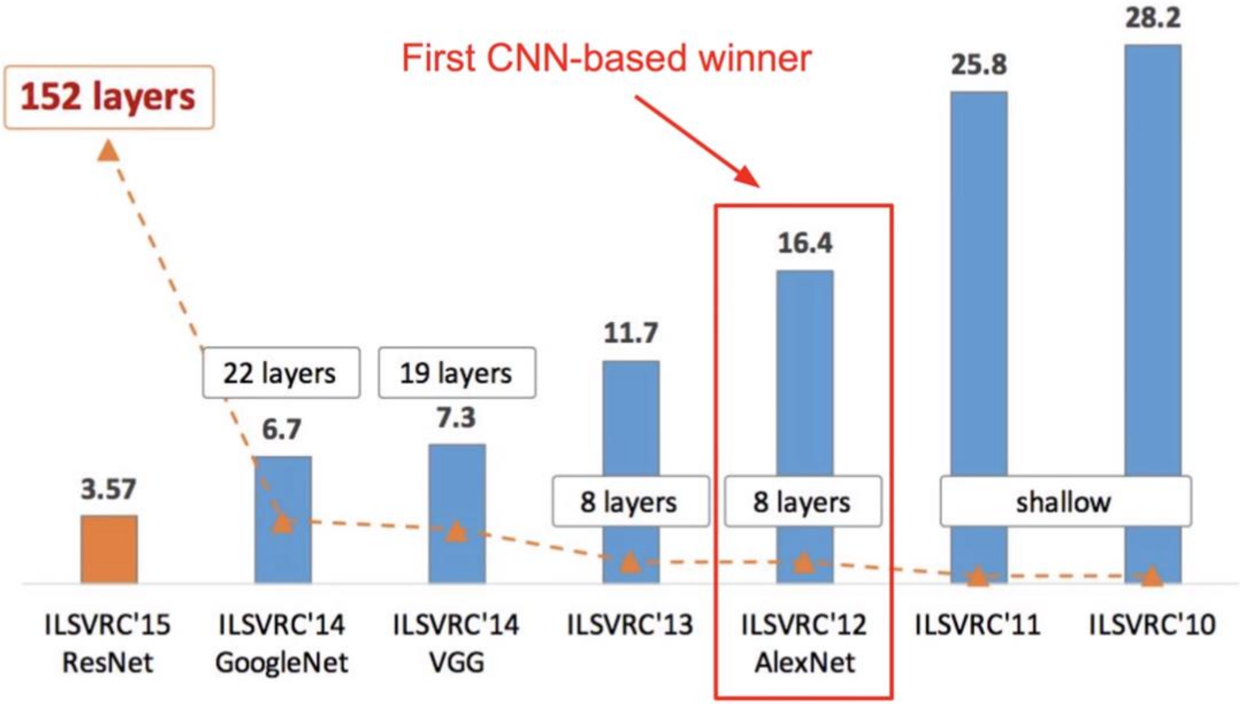
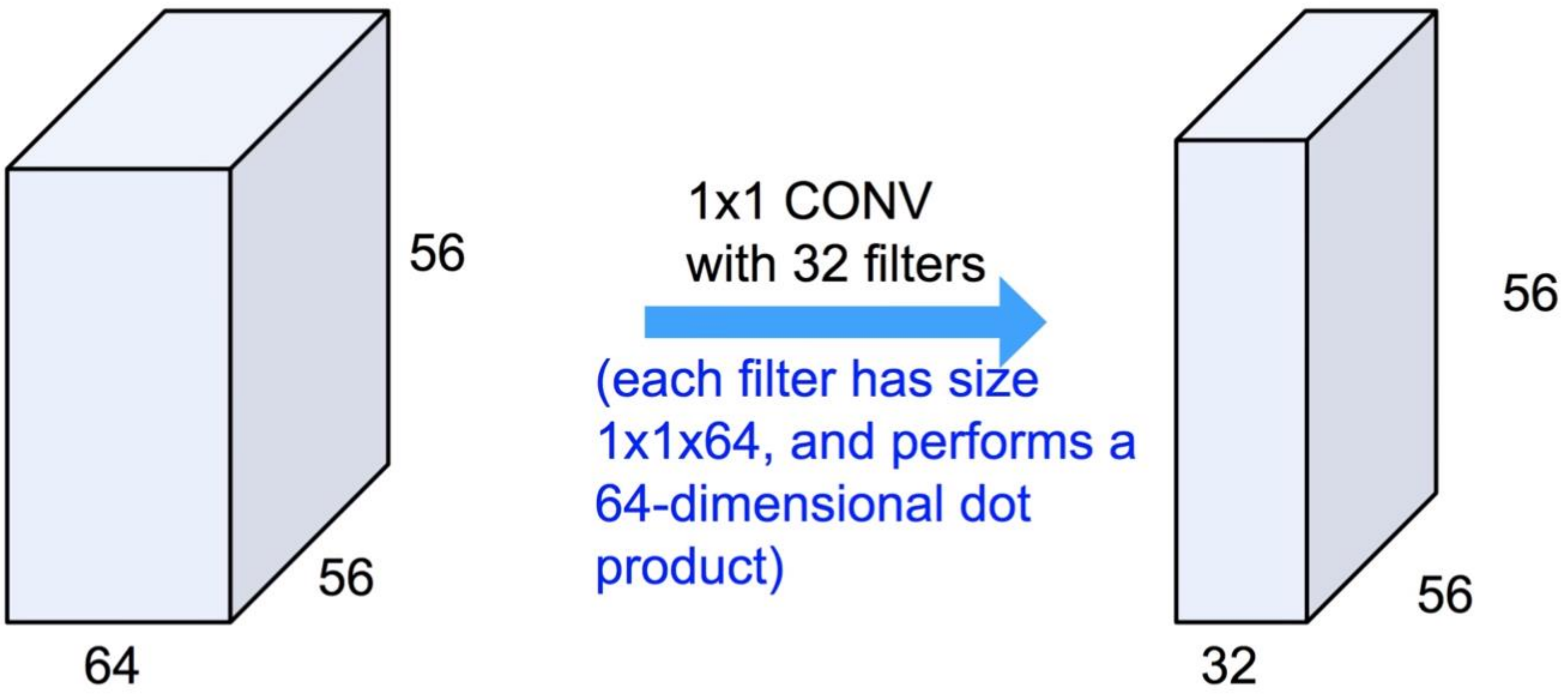


Figure copyright Kaiming He, 2016. Reproduced with permission.

1 X 1 Convolutions

Reminder: 1x1 convolutions

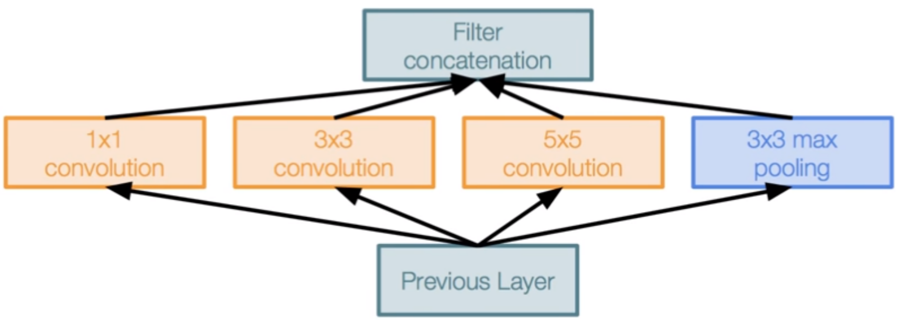


Google Inception to reduce computation cost

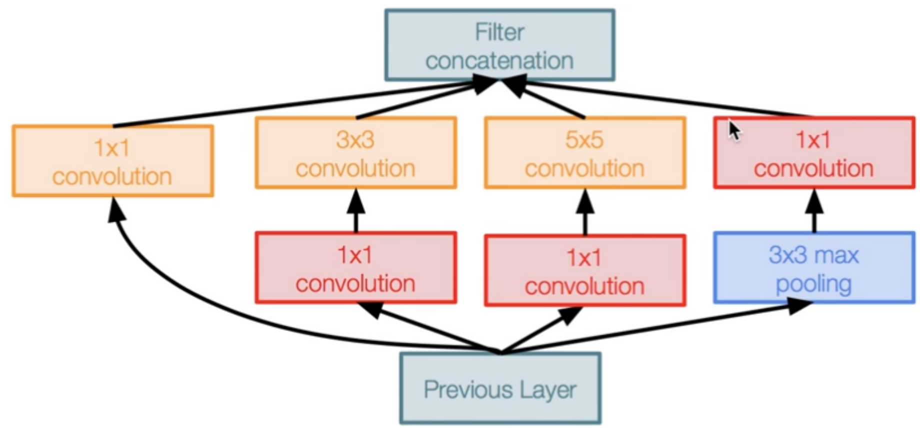
Case Study: GoogLeNet

[Szegedy et al., 2014]

1x1 conv “bottleneck” layers

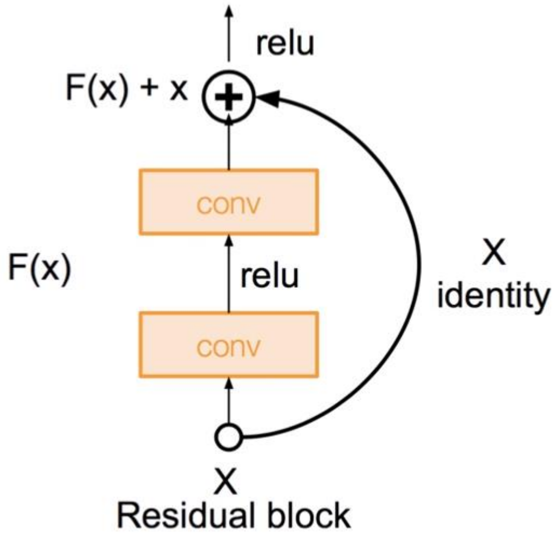


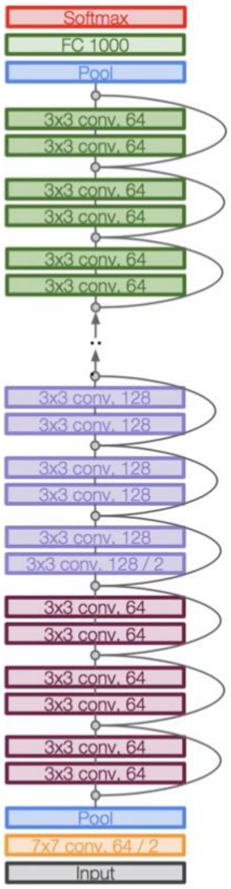
Naive Inception module



Inception module with dimension reduction

Very deep networks using residual connections



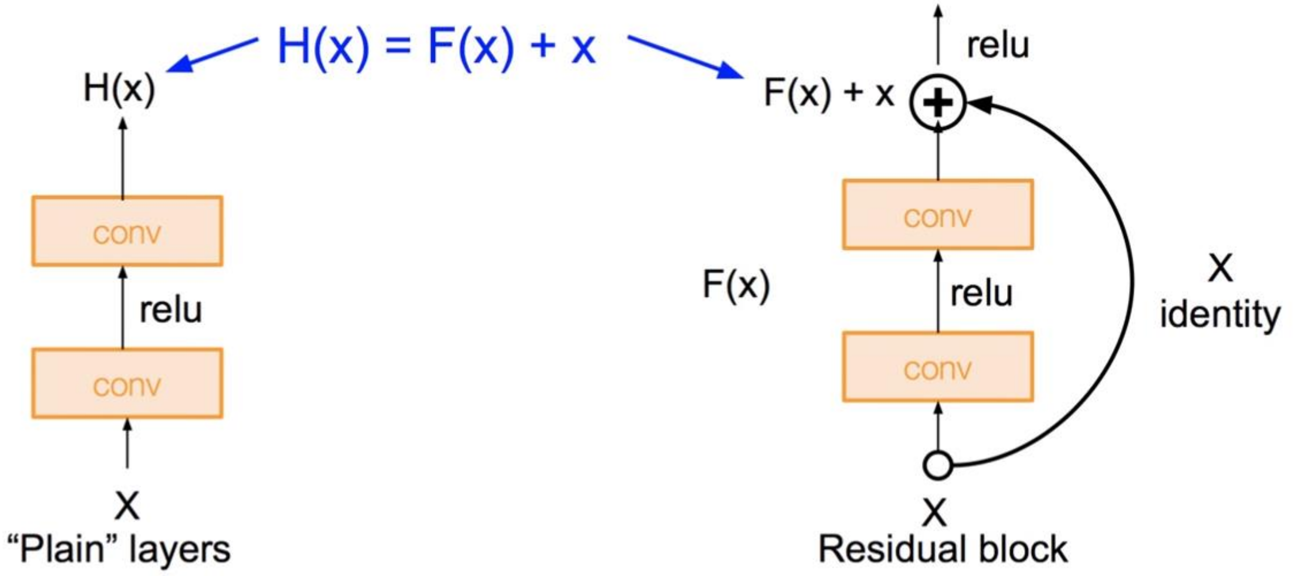


Residual

Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

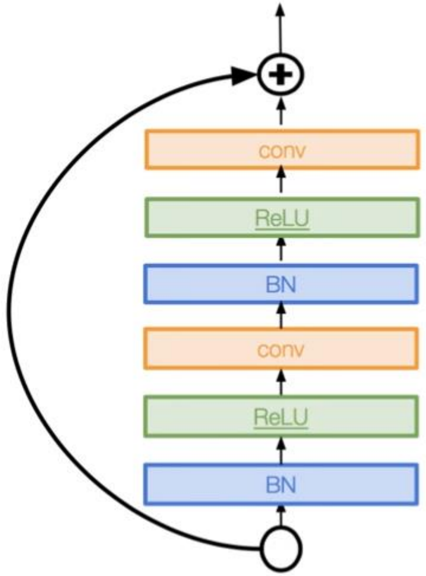


Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

Identity Mappings in Deep Residual Networks

[He et al. 2016]

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance



DenseNet

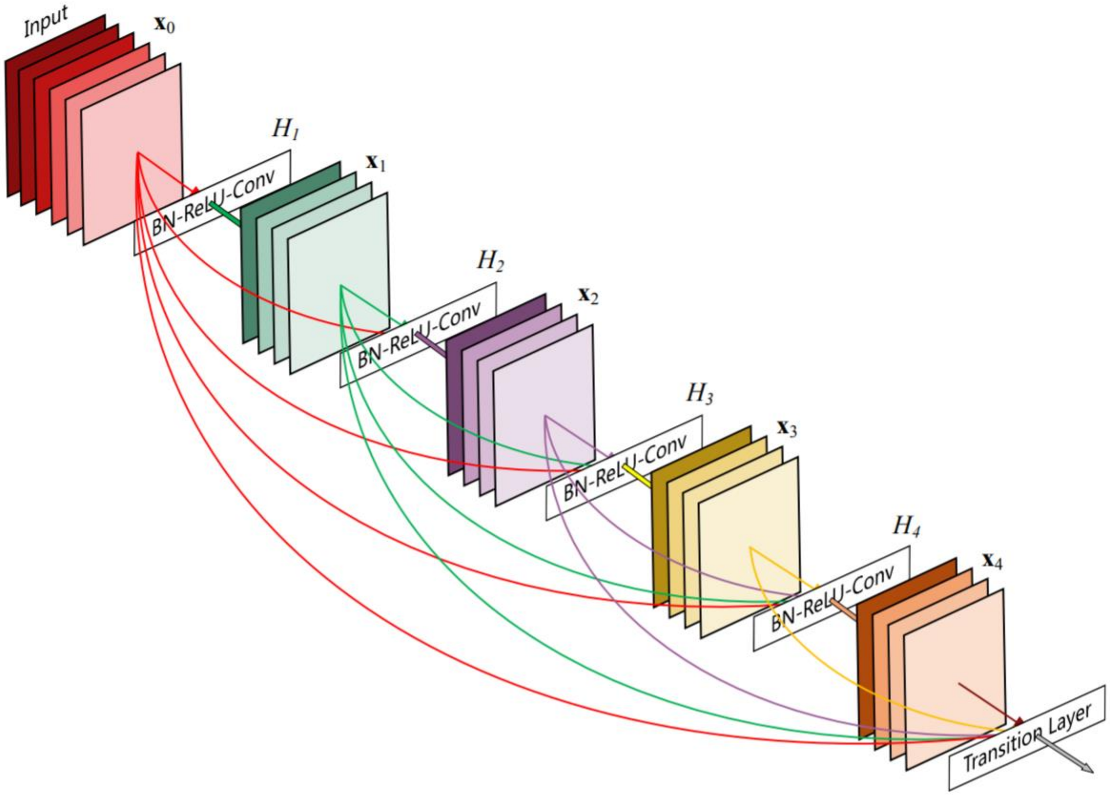


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

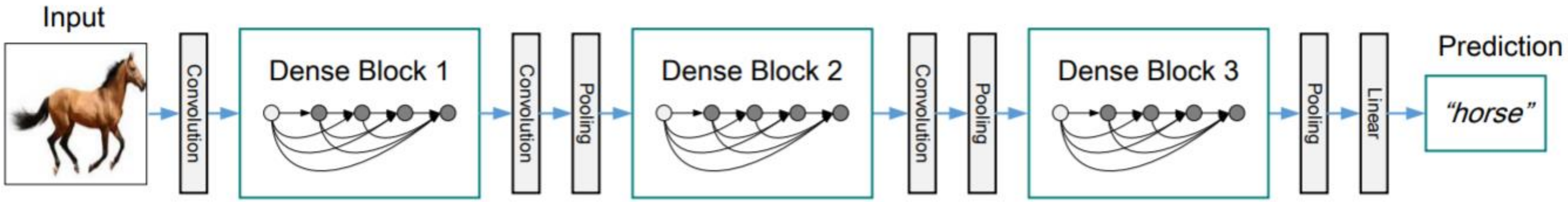
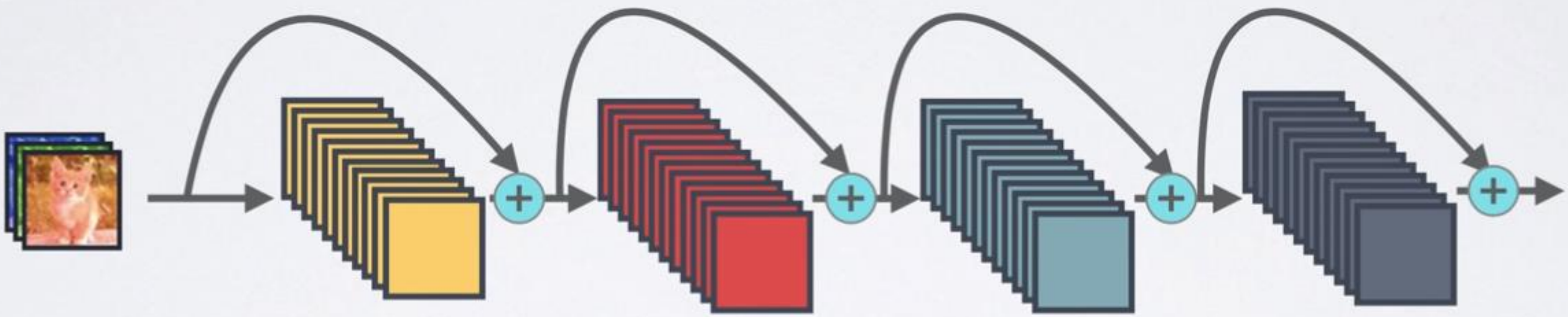


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

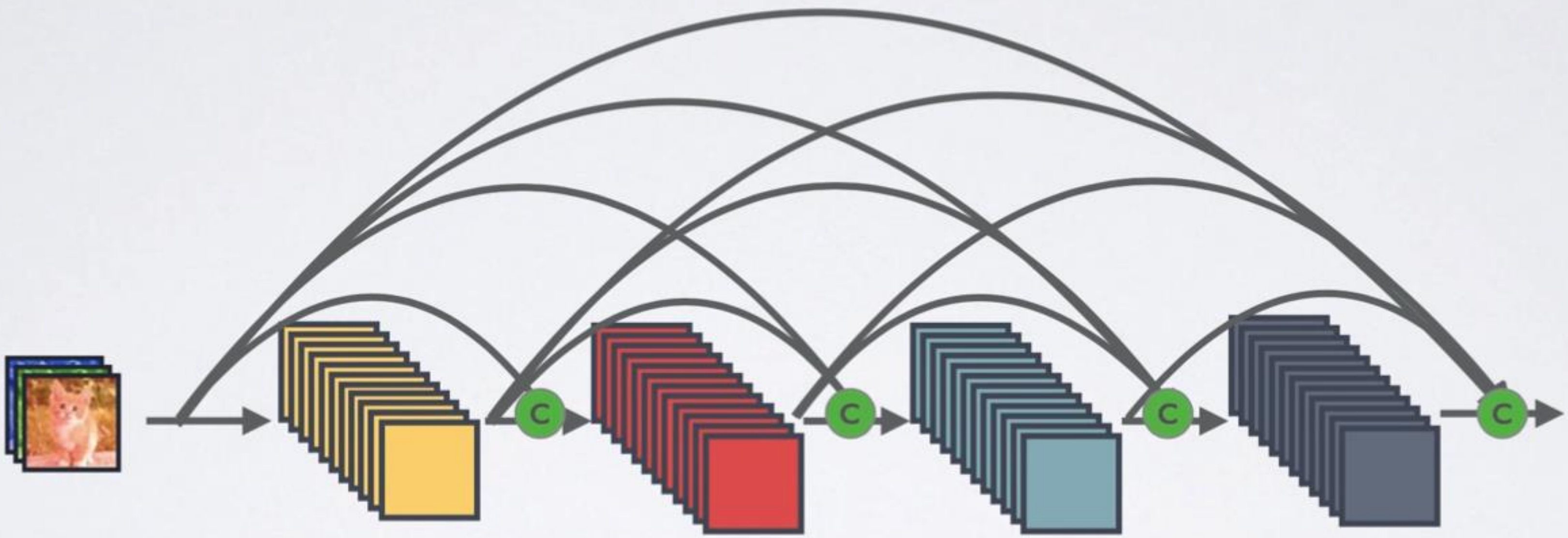
RESNET CONNECTIVITY

Identity mappings promote gradient propagation.



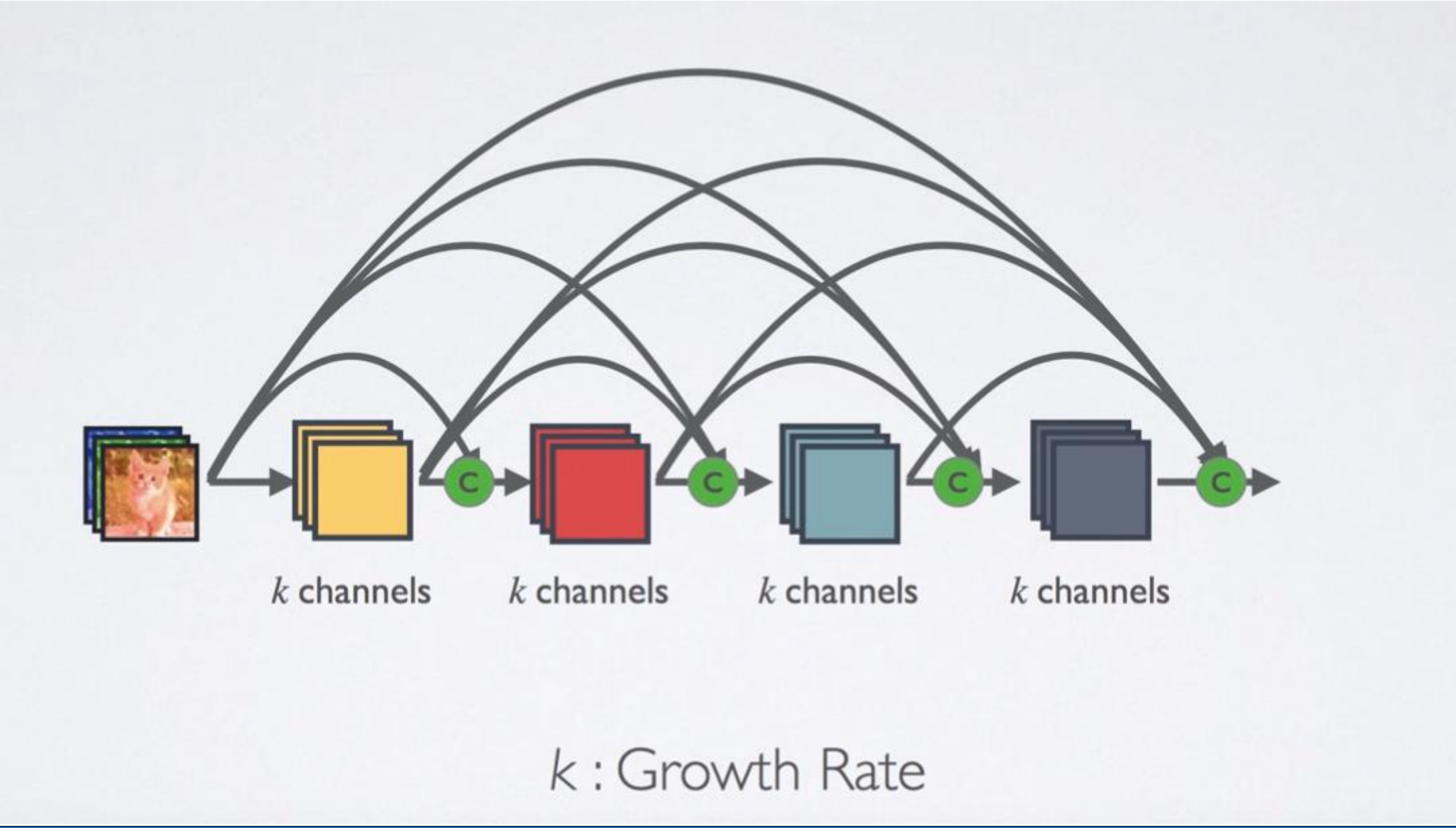
⊕ : Element-wise addition

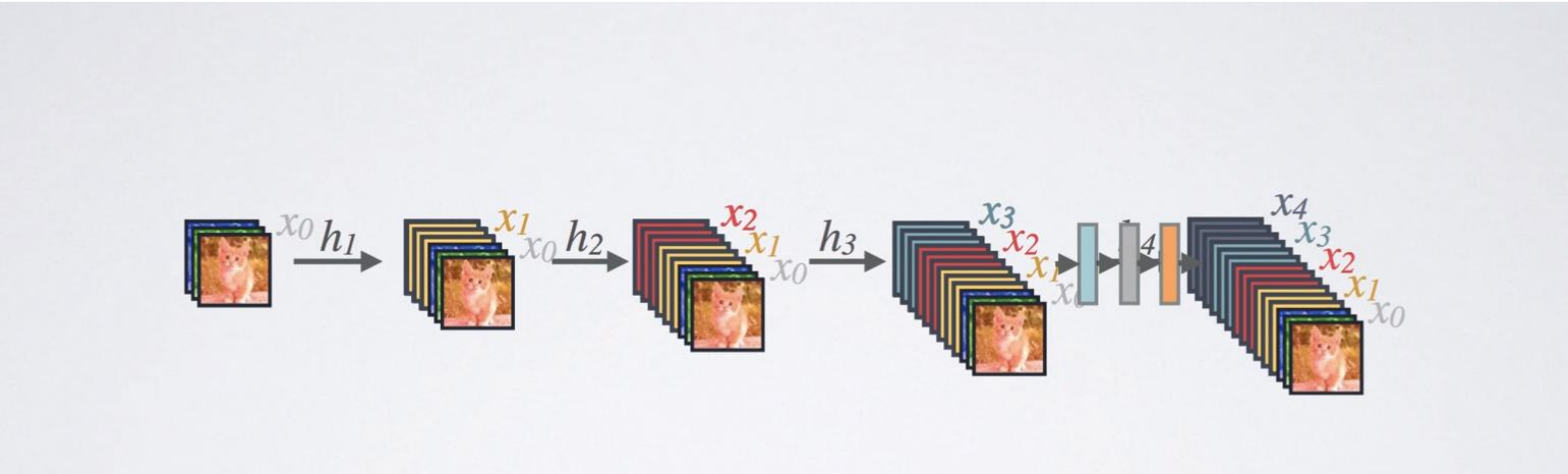
DENSE CONNECTIVITY



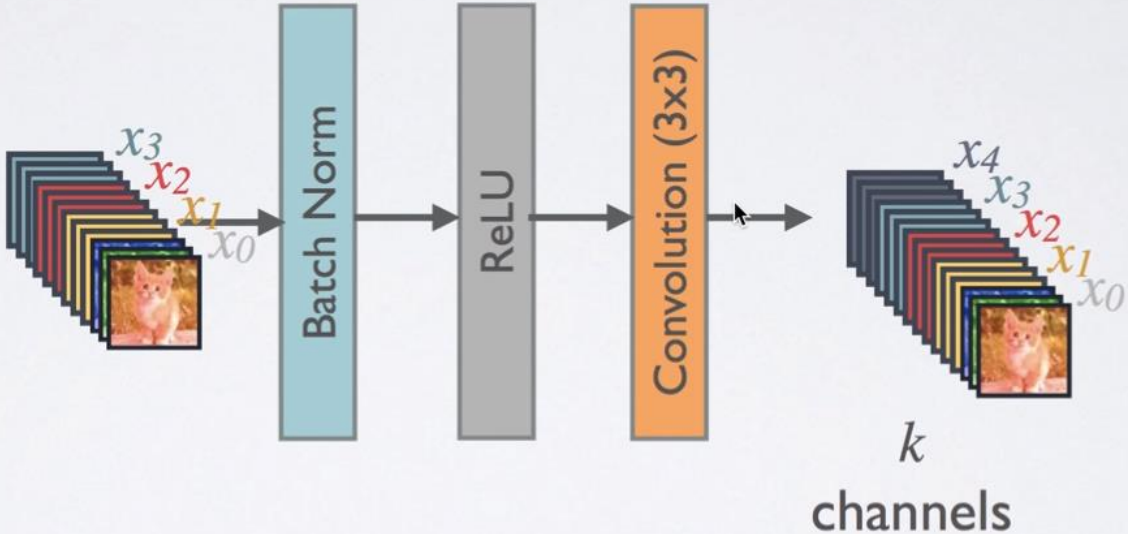
: Channel-wise concatenation

Dense and Slim with growth rate

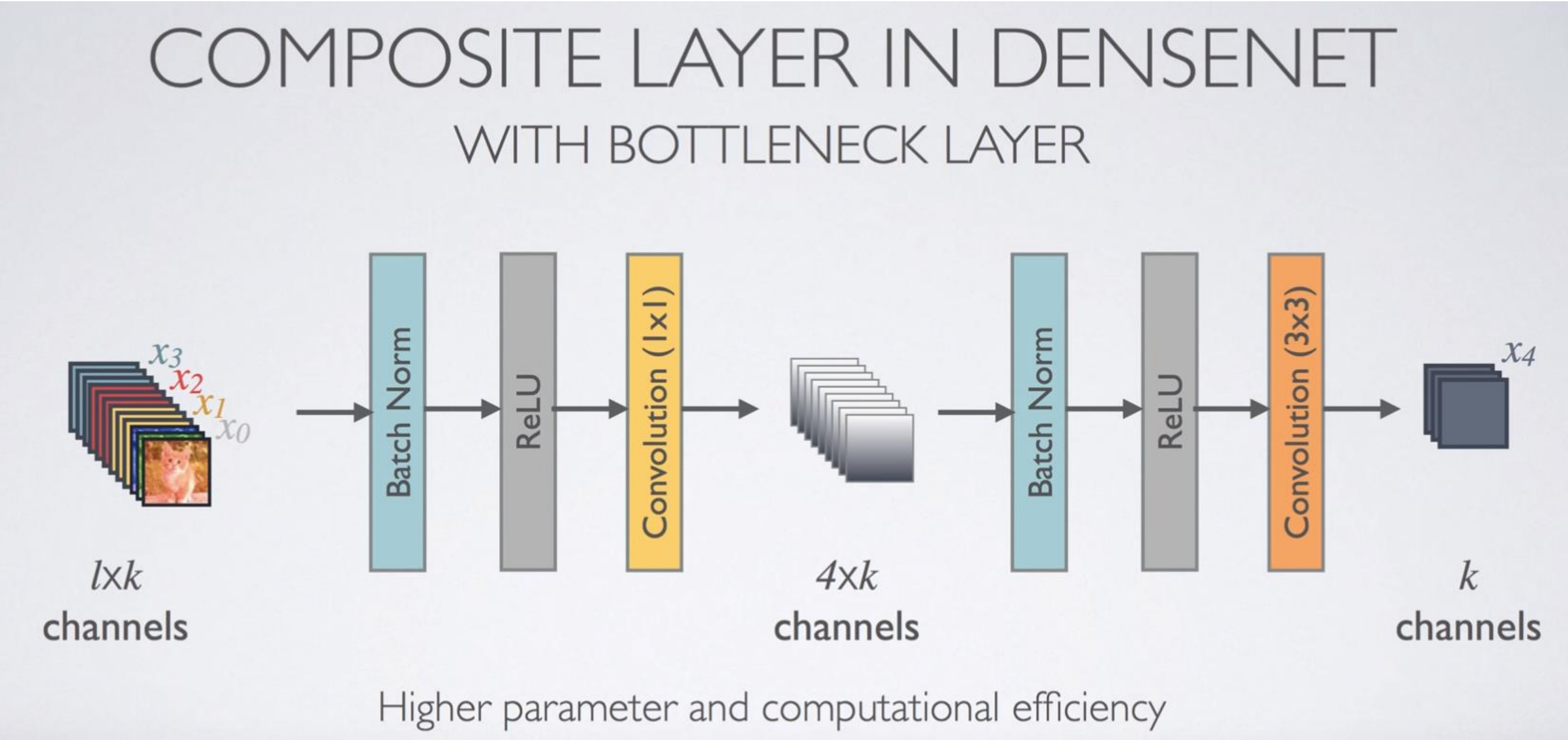


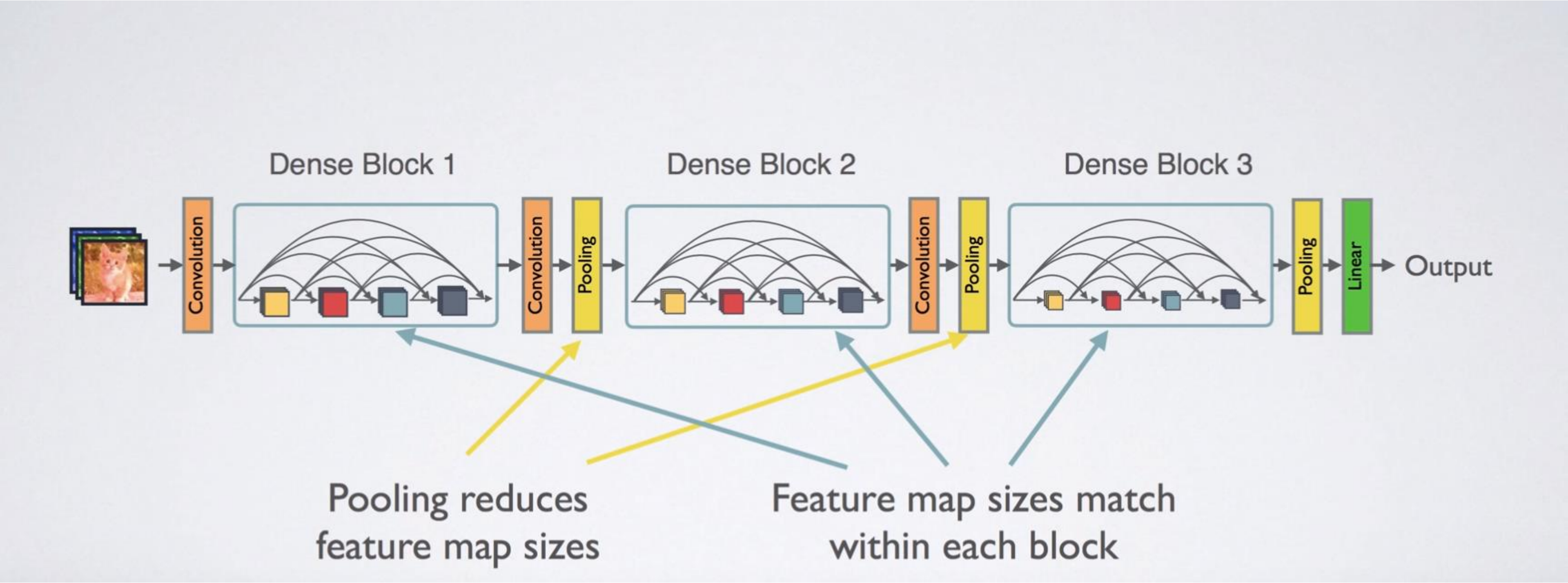

x0
h1
x1
x0
h2
x2
x1
x0
h3
x3
x2
x1
x4
x3
x2
x1
x0

COMPOSITE LAYER IN DENSENET



$$x_5 = h_5([x_0, \ldots, x_4])$$


COMPOSITE LAYER IN DENSENET
WITH BOTTLENECK LAYER
x3
x2
x1
x0
Batch Norm
ReLU
Convolution (1x1)
Batch Norm
ReLU
Convolution (3x3)
x4
lxk
channels
4xk
channels
k
channels
Higher parameter and computational efficiency


Dense Block 1
Dense Block 2
Dense Block 3
Convolution
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output
Pooling reduces
feature map sizes
Feature map sizes match
within each block

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			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.