Densely Connected Convolutional Networks

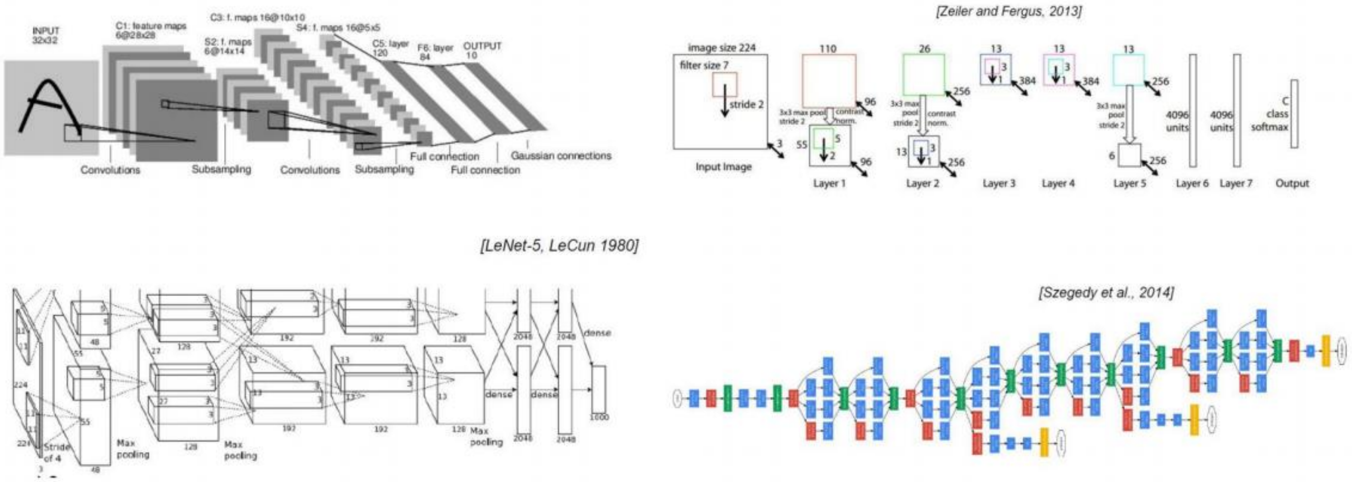
Gao Huang

Zhuang Liu

Laurens van der Maaten

Kilian Q.Weinberger

1. Introduction



[{(Convolution + Activation) x n + pooling} x m] + Fully connected layer

As CNNs become deeper the prediction is more accurate but
i) information flow weakens
ii) gradient vanishing when training ☹

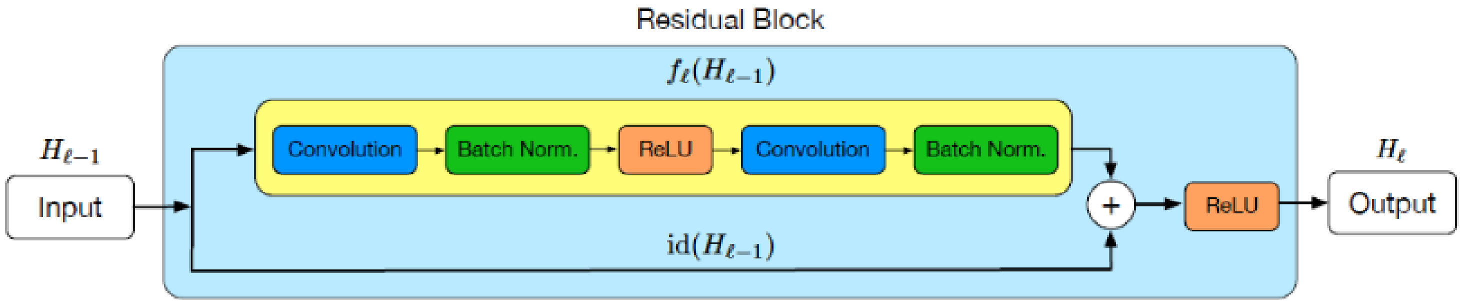
Need **shorter connections** between layers close to the input and those close to the output

2. ResNet

Identity connections

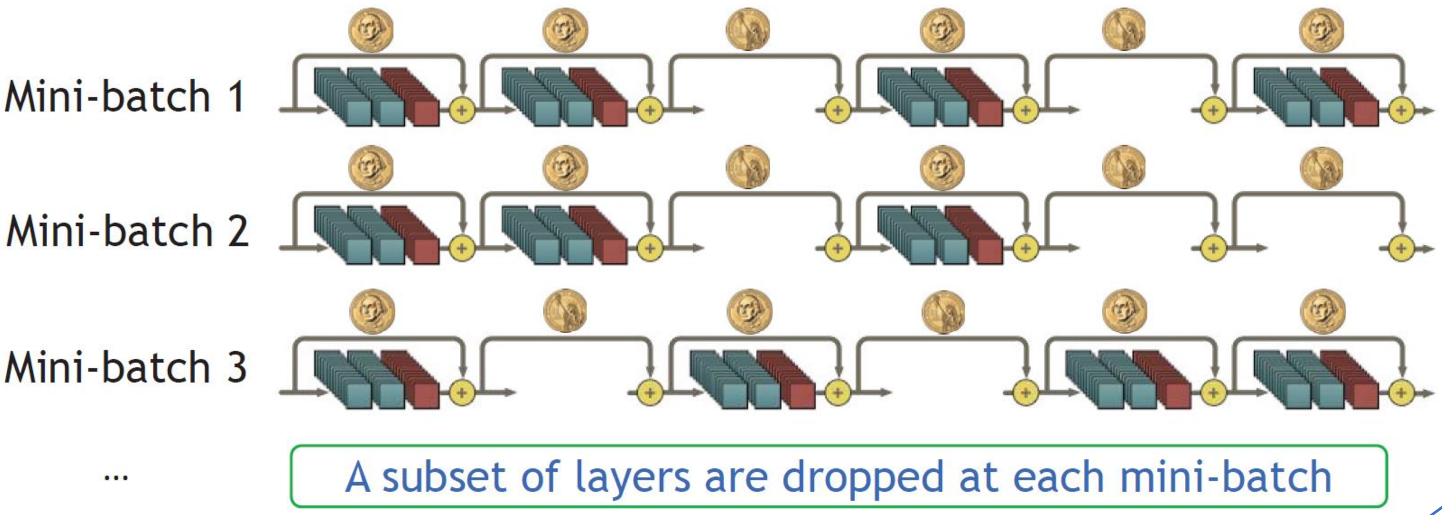
$$H_\ell = \mathrm{ReLU}(f_\ell(H_{\ell-1}) + \underline{\mathrm{id}(H_{\ell-1})})$$

add



One l-th Residual Block (ResBlock) in Original ResNet

2. ResNet



$$H_\ell = \texttt{ReLU}(b_\ell f_\ell(H_{\ell-1}) + \mathrm{id}(H_{\ell-1}))$$

Bernoulli random variable

$b_\ell \sim \texttt{Bernoulli}(p_\ell)$ with $p_\ell = (1 - \frac{\ell}{L}) \times 1 + \frac{\ell}{L} \times p_L$

Linear Decay Rule

Stochastic depth
→ Successfully trained 1202-layer ResNet

Identity connections
Stochastic depth

➔ Create **short path** from early layers to later layers

3. DenseNet Architecture

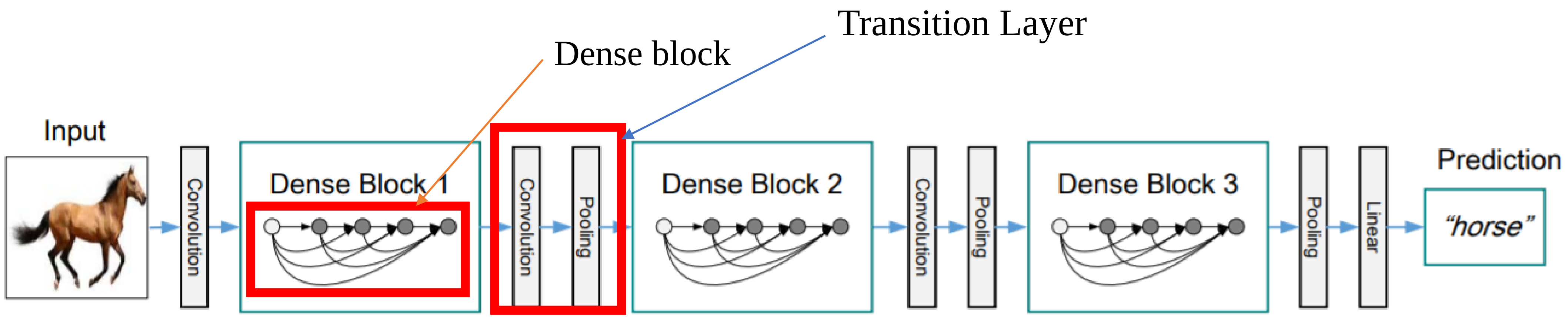


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Advantages

i) Alleviate vanishing-gradient problem
ii) Strengthen feature propagation
iii) Encourage feature reuse
iv) Easy train by improved flow of information and gradients
v) Reduce the number of parameters ☺

Properties

i) Compact feature representation → Reduce feature redundancy
ii) Implicit deep supervision

3. DenseNet Architecture

Dense Connectivity

concatenation

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

summation

$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}.$$

ResNet
earlier layer의 information이 잘 전달되지 않는다.
→ Impede information flo
→ w
Vs.
DenseNet
uncorrelated feature도 flow에 담긴다.

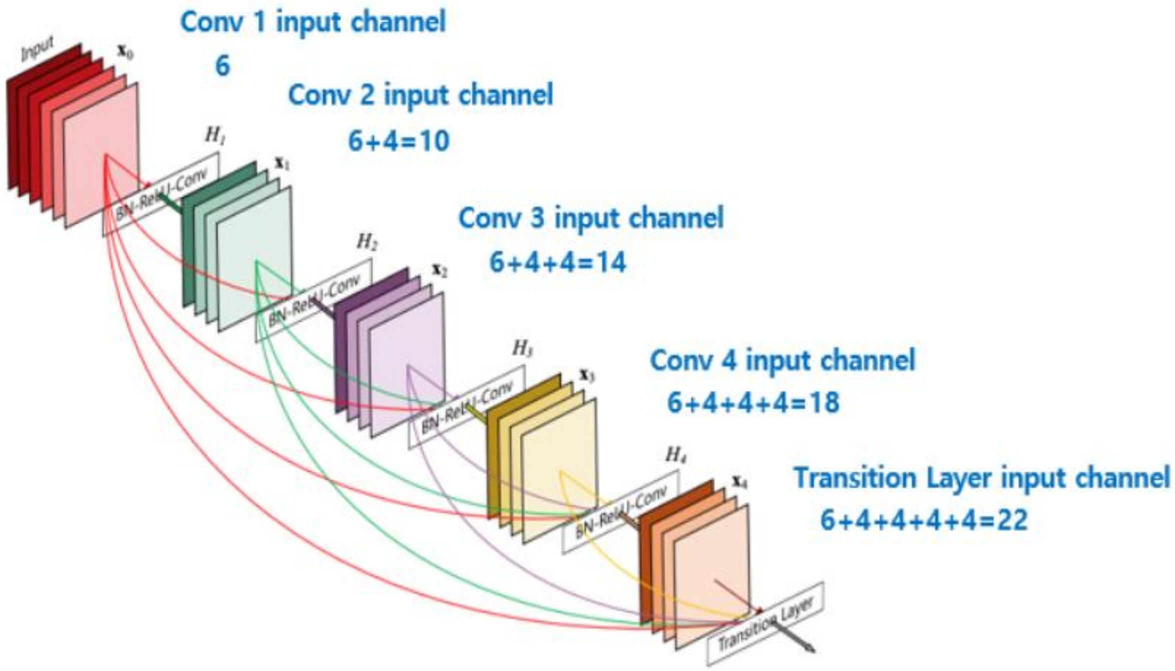


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

[Dense Connectivity]

3. DenseNet Architecture

Composite function

Composite function. Motivated by [12], we define $H_\ell(\cdot)$ as a composite function of three consecutive operations: batch normalization (BN) [14], followed by a rectified linear unit (ReLU) [6] and a 3×3 convolution (Conv).

Three consecutive operations

$H_\ell(\cdot)$

1. **Batch normalization (BN)**
2. **Rectified linear unit (ReLU)**
3. **3*3 convolution**

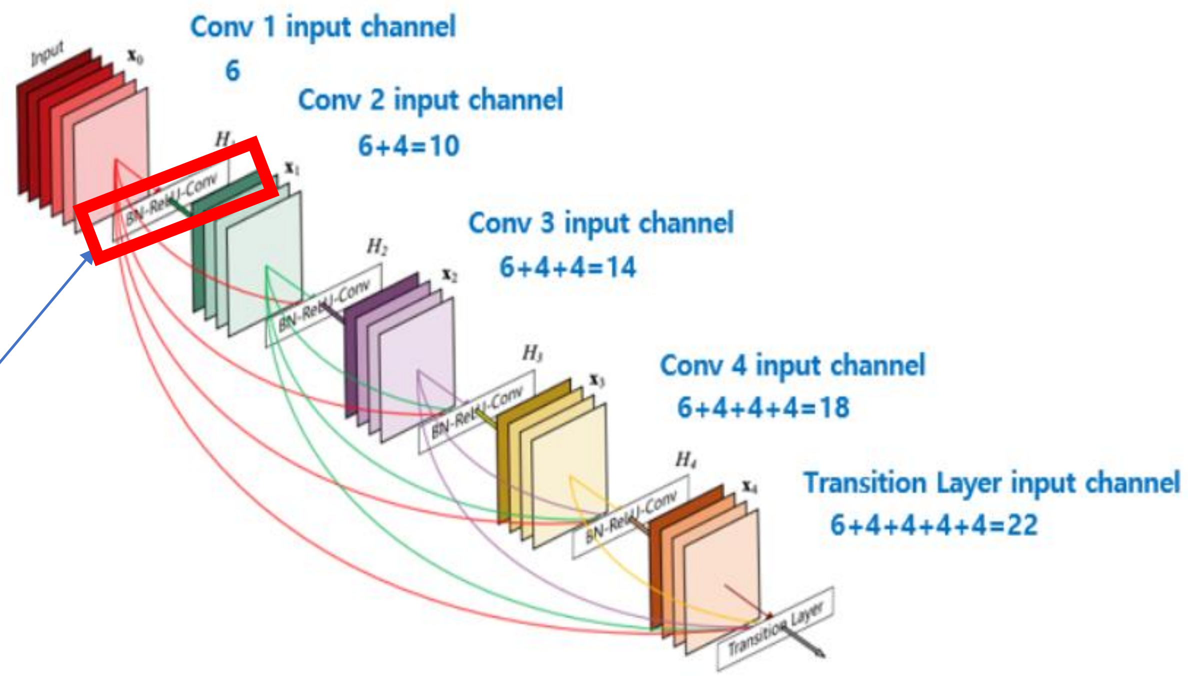


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

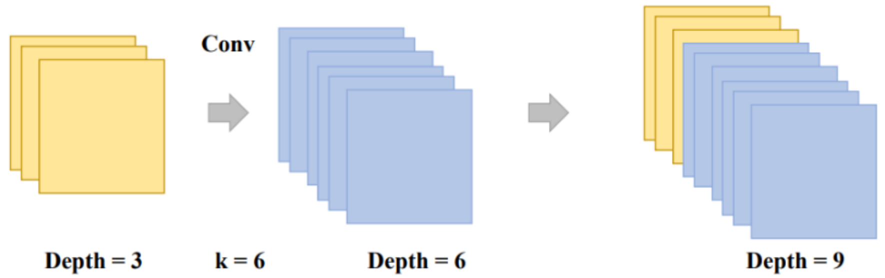
[Dense Connectivity]

3. DenseNet Architecture

Growth rate k (hyperparameter)

: 각 layer의 feature map에서의 channel 개수
Normally use small k, e.g., 12 but Why small k is sufficient?
→ "Collective knowledge by DENSE CONNECTIVITY"
→ Feature maps = global state of the network

k: regulates how much new information each layer contributes to the global state



K = 4

Input feature map of l th layer
$= k_0 + k(l-1)$

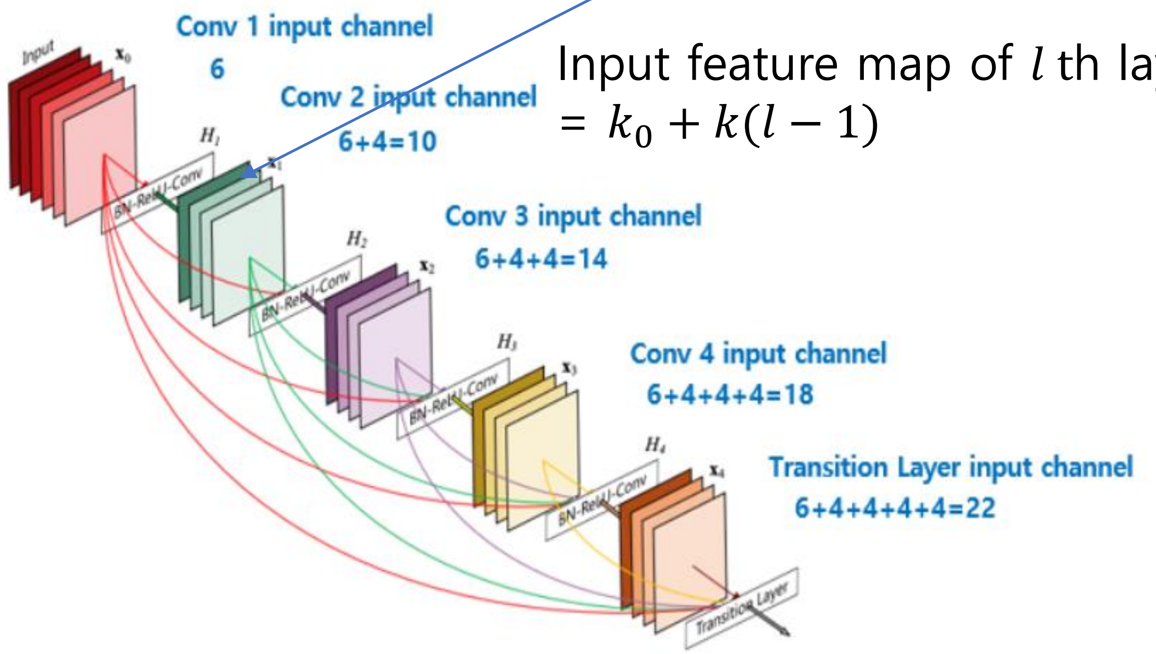
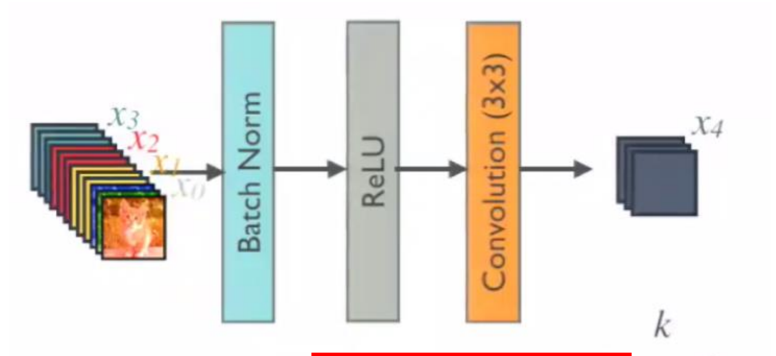


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

[Dense Connectivity]

3. DenseNet Architecture

Bottleneck Layer



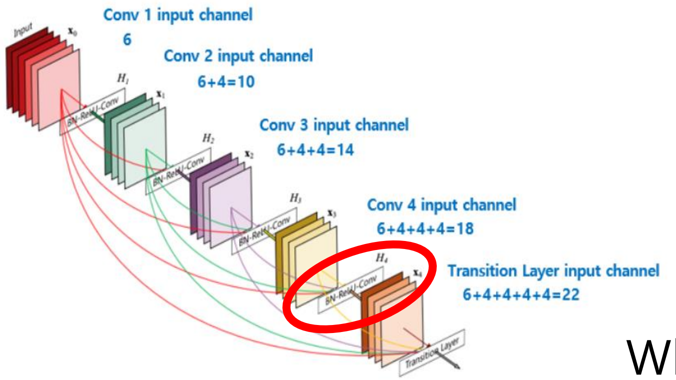
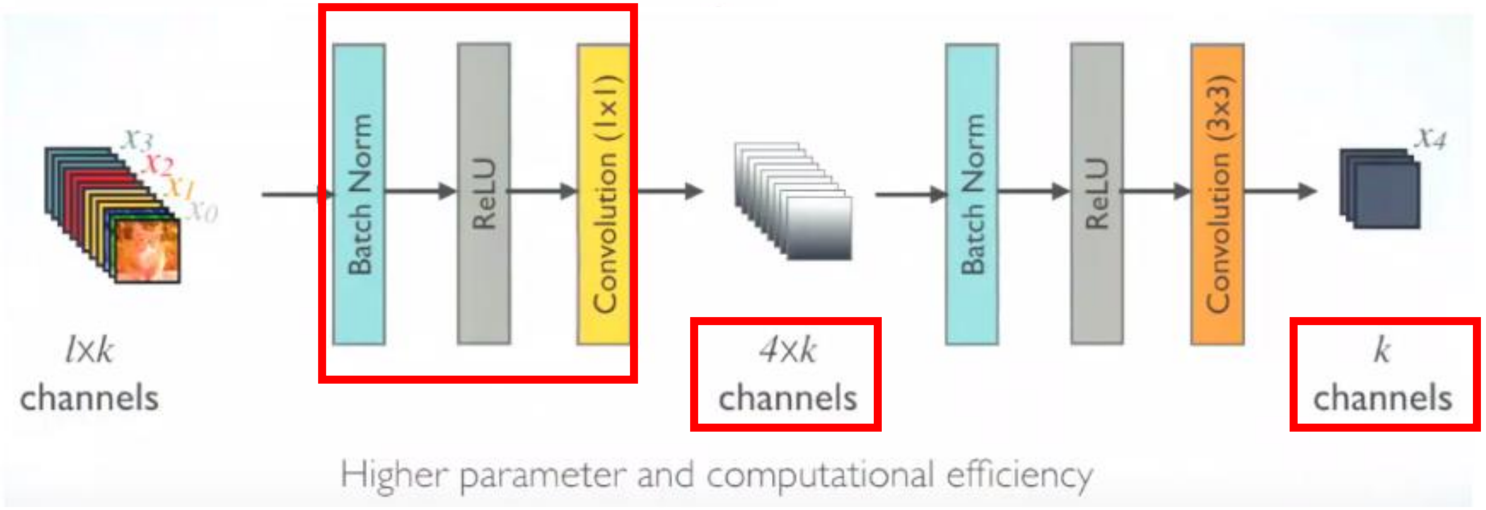


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

[Dense Connectivity]



While
The number of input= $k_0 + k(l-1)$
Output = should be k

➔ 3x3 conv 연산 전에
➔ 1x1 conv를 통해
➔ Input feature-maps의 숫자를 감소

➔ Reduce computational cost ☺
➔ **Computational efficiency**

3. DenseNet Architecture

Pooling layers

2 x 2 average pooling layer

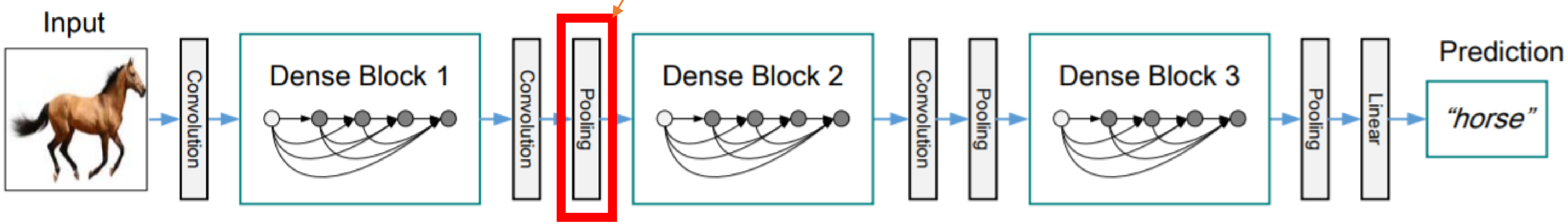


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Concatenation은 feature map size가 변하면 연산이 불가능하다.

Convolutional Network는 Pooling으로 feature map size를 줄여줘야 한다.

➔ 네트워크를 몇 개의 Dense Block으로 나눈 후

➔ 같은 feature map size를 가지는 layer를 dense block으로 묶는다.

➔ Dense block 사이를 transition layer로 연결하고 batch normalization, 1 x 1 conv, 2 x 2 average pool

3. DenseNet Architecture

Compression

Feature map 개수: m → $\lfloor \theta m \rfloor$개

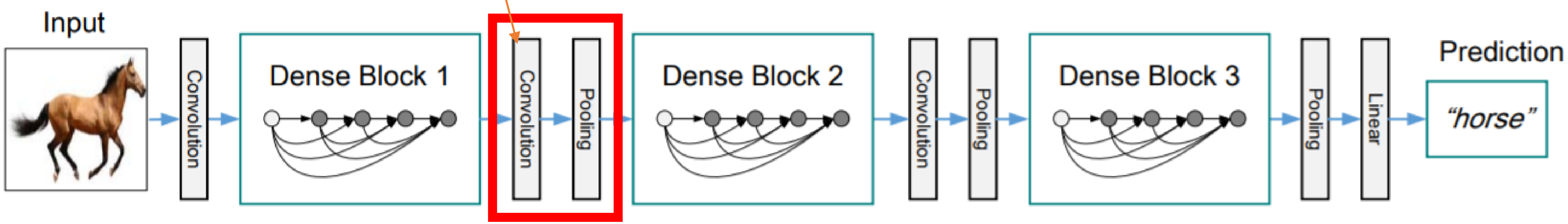


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

To improve model compactness,
Transition layer에서의 hyperparameter θ ($0 < \theta \leq 1$) 을 통해
output feature map의 개수를 적게 유지한다. ➔ θ = 0.5
If a dense block contains m feature maps,
We let the following transition layer generate $\lfloor \theta m \rfloor$ output feature maps

4. Experiments

Dataset(CIFAR-10, CIFAR-100, SVHN, ImageNet)

	CIFAR	SVHN	ImageNet
Optimization Method	SGD	SGD	SGD
Batch Size	64	64	256
Epoch	300	40	90
Initial Learning Rate	0.1	0.1	0.1
Initalization Method	He	He	He

4. Experiments

DenseNet Structure on ImageNet dataset

Input image size = 224 x 224

2k convolutions

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.

4. Experiments

Classification Results with ResNet variants

Increased representational power of bigger and deeper model

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Parameter Efficiency

Bottleneck + Compression

Accuracy

Table 2: Error rates (%) on CIFAR and SVHN datasets. k denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

4. Experiments

Classification Results

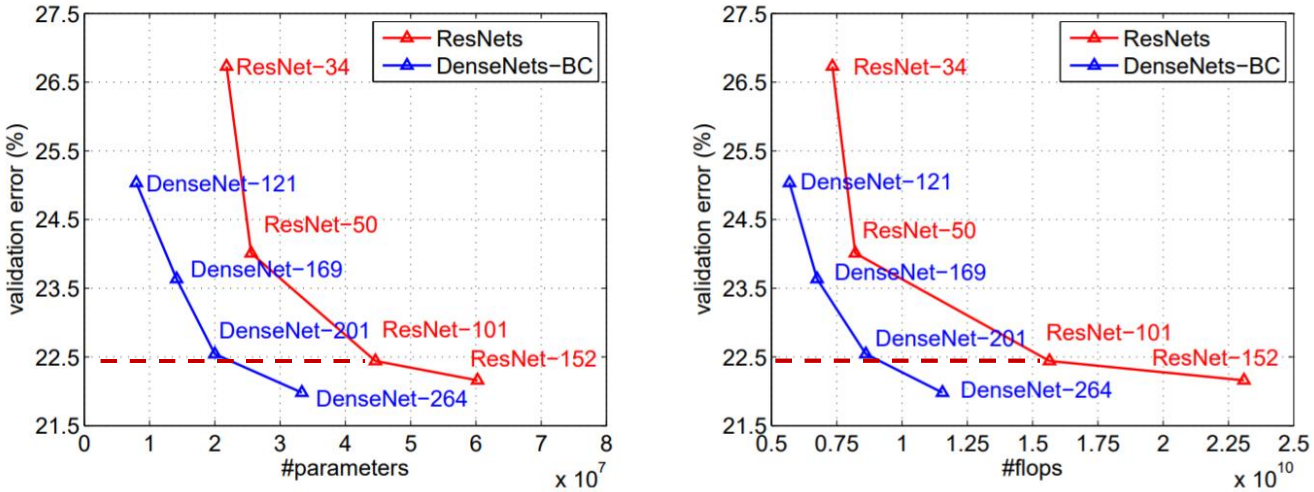


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

Hyperparameters are optimized for ResNets but not for DenseNets

4. Experiments

Model Compactness

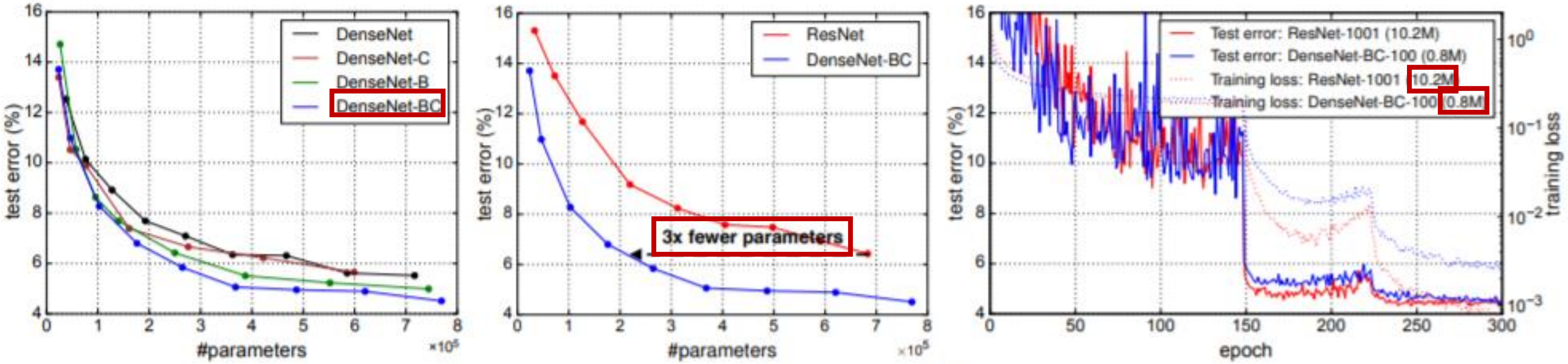


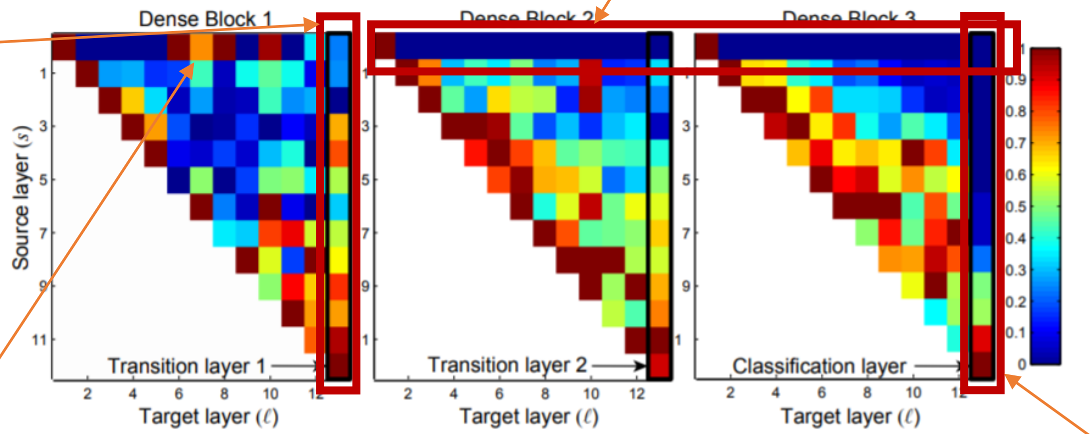
Figure 4: *Left:* Comparison of the parameter efficiency on C10+ between DenseNet variations. *Middle:* Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets. DenseNet-BC requires about 1/3 of the parameters as ResNet to achieve comparable accuracy. *Right:* Training and testing curves of the 1001-layer pre-activation ResNet [12] with more than 10M parameters and a 100-layer DenseNet with only 0.8M parameters.

4. Experiments

2번째와 3번째 dense block에서 각 block의 이전 transition layer는 비중 있는 weight을 차지하지 못한다
→ Transition layer가 redundant features를 내뱉는다.
→ Transition layer의 비중을 줄인 DenseNet-BC의 성능이 더 좋다

Feature Reuse

Transition layer는 이전 layer들의 weight을 골고루 받는다.



Final classification layer에서는 Final feature-maps로 weight가 집중되어있다.
→ high-level feature가 네트워크 후반부에 생성됨을 확인할 수 있다.

모든 layer는 dense block 내에서 weight를 골고루 퍼뜨린다.
특히 초반부 layer에서 extract된 features도 directly 사용된다.

Figure 5: The average absolute filter weights of convolutional layers in a trained DenseNet. The color of pixel (s, ℓ) encodes the average $L1$ norm (normalized by number of input feature-maps) of the weights connecting convolutional layer s to ℓ within a dense block. Three columns highlighted by black rectangles correspond to two transition layers and the classification layer. The first row encodes weights connected to the input layer of the dense block.

5. Conclusion

Dense Connectivity

Feature Reuse

Less Parameters

Less Computation

Compact Model

Reduce Feature Redundancy

Implicit Deep Supervision

DenseNet

이미지 출처: https://hoya012.github.io//blog/DenseNet-Tutorial-1/

Thank You ☺