DenseNet

0. Abstract

1. 입력에 가까운 층과 출력에 가까운 층의 short conn (resnet)

↓착안

feed forward 방식으로 각 layer를 다른 모든 층과 연결

2. L개 층

→ Resnet 이전 : $1 \times L = L$ (각 층마다 들어오는 edge 가 1개 뿐)

Resnet : $1 \times L$ + short conn

Densenet : $x_0 \to x_1$: 1개

$x_0, x_1 \to x_2$: 2개

⋮

$x_0, \cdots, x_{L-1} \to x$: L개

$\Rightarrow \frac{1+L}{2} \times L = \frac{L(L+1)}{2}$ "Dense connectivity"

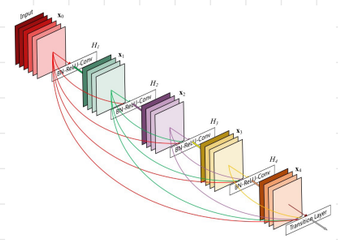


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

3. 각 이전 층들의 f.m을 입력으로

4. gradient vanishing 해결
   feature propagation↑
   feature reuse
   parameter ↓↓ (연산량↓)

1. Introduction

1.

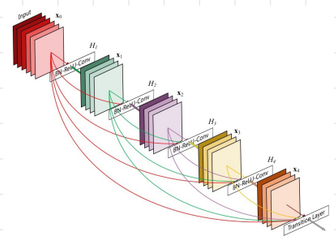


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

→ ResNet과 달리 합산해서 전달X : "concatenate"

↳ L번째 층은 L개를 입력 받음 (합산X)

→ 중복 feat 재학습X : "reuse" (파라미터수 ↓)

2. feed-forward

→ traditional: state 전달 시 $H(x) \approx x$ 를 직접 최적화하므로 보존/수정 정보 구분↓

→ ResNet: identity mapping, 파라미터↑

→ DenseNet: 보존/수정 정보 구분, 파라미터↓

↳ 각 층 필터↓ → 소량 피처맵만 추가

↳ "Implicit deep supervision": gradient 시 각 층 직적 gradient 식이 그대로 포함 (중간 auxiliary classifier 필요X)

" Regularization": dense-conn

2. Related work

1. ≈ Cascade: layer-by-layer 학습, but 소규모 데이터만

≈ ResNet, Highway Net: gating unit, bypass path

2. ResNet vs DenseNet

: feature reuse

3. DenseNets

1. x_0 → conv ··· → l번째 → ···

input image

Lm layers

: $H_l(\cdot)$

↳ BN
ReLU
pooling
conv

↓

출력: x_l (각 층의 f.m)

↳ traditional: $x_l = H_l(x_{l-1})$

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

정보흐름방해 가능성

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

Concatenate

2. DenseNets

1) Dense Connectivity

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

Concatenate

2) Composite function

→ $H_\ell(\cdot)$ = BN + ReLU + 3x3 conV

3) Pooling layers

Figure 1

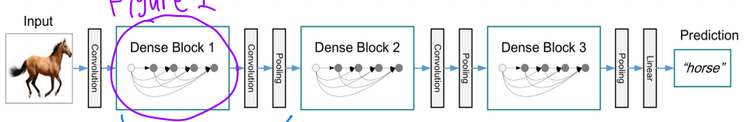


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

↳ feature 크기 맞춰야 Concatenate : downsampling

↳ Dense Block + transition layer (전이층)

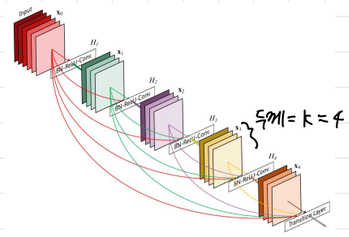
Dense Block: Conv

transition layer: Conv + pooling

(= BN + 1x1 conv + 2x2 avg pooling)

4) Growth rate

→



← 개수의 증가

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

H_ℓ 이 갖는 kernel map

→ ℓ번째층 입력 : $(\ell-1) \cdot k + k_0$

전체 입력 채널 수

→ "k" (growth rate) : 작은 값이어도 (작은 성장률만으로) 이전 fm(collective knowledge, global state)에 공헌

= 각 층 출력 channel 수

⇒

Transition : $k_0^{next} = \lfloor \theta \cdot C_{out} \rfloor$

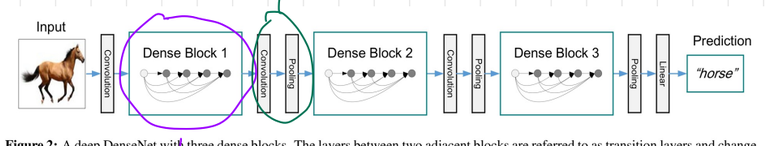


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

C_{in} : $k_0 + (\ell-1)k$

Concatenate

ℓ = 레이어 index (0~4)

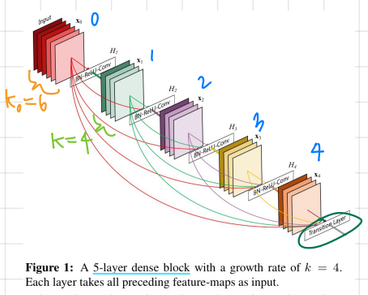


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

C_{out} : $k_0 + N \cdot k$

$= 6 + 5 \cdot 4$

$= 26$ (Transition이 적용되는 Dense Block의 최종 출력 ch)

N = layer 개수

5) Bottleneck layers

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	[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] × 64
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] × 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.

↳ Dense Block : 1x1 이후는 bottleneck 추가 (ResNet 참고) 효율↑

↳ DenseNet-B : H_ℓ = BN-ReLU-Conv(1x1) - BN-ReLU-Conv(3x3)

4k(4 배수) fm 생성 → 입력을 받아서 k개 fm 최종 생성

6) Compression

→ transition layer의 출력 fm 줄이기

Conv + pooling

(= BN + 1x1 conv + 2x2 avg pooling)

→ Dense block : m개 fm + transition layer : $\lfloor \theta m \rfloor$ fm

$0 < \theta \leq 1$: compression factor

(transition layer 내부의 1x1 conv 출력 fm의 ch는 입력 ch x θ로 설정)

θ=1 이면 그대로

↳ DenseNet-C : θ<1 (0.5로 설정)

↳ DenseNet-BC : Bottleneck + θ(=0.5)

7. Implement Details

→

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	[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] × 64
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] × 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.

→ 2k개 growth rate (ch)

k개

↳ DenseNet-BC

4개 DenseNet

4. Experiments

(논문 참고)

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**	-

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.

Model	top-1	top-5
DenseNet-121	25.02 / 23.61	7.71 / 6.66
DenseNet-169	23.80 / 22.08	6.85 / 5.92
DenseNet-201	22.58 / 21.46	6.34 / 5.54
DenseNet-264	22.15 / 20.80	6.12 / 5.29

Table 3: The top-1 and top-5 error rates on the ImageNet validation set, with single-crop / 10-crop testing.

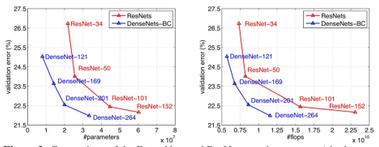


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

↳ ResNet hyper 동일 설정

→ Acc↑, Capacity(parameters), Parameter efficiency, overfitting↓)

5. Discussion

1. "Model Compactness"

→ f.m Concatenate (feature reuse)

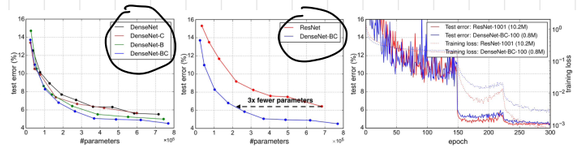


Figure 4: *Left:* Comparison of the parameter efficiency on C10+ between DenseNet variations. *Middle:* Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets. DenseNet-BC requires about 1/3 of the parameters as ResNet to achieve comparable accuracy. *Right:* Training 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.

2. "Implicit Deep Supervision"

→ 각 층 짧은 연결로 loss로부터 추가 지도 신호

→ gradient, loss 감소 완화

3. "Stochastic vs deterministic connection"

ResNet

→ ResNet dropout 시 주변 층들 간 직접 concatenate

↳ pooling 층은 drop X ≈ DenseNet

4. "Feature Reuse"

→

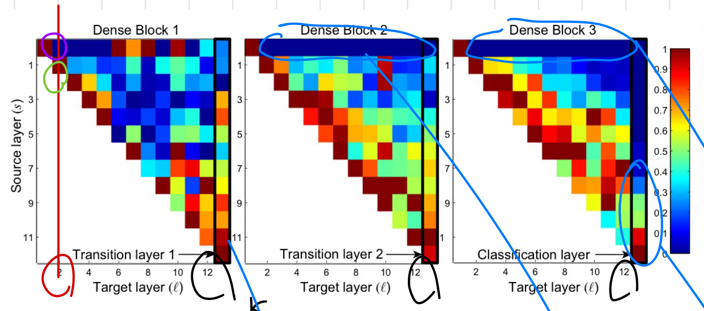


Figure 5: The average absolute filter weights of convolutional layers in a trained DenseNet. The color of pixel (s, ℓ) encodes the average $L1$ norm (normalized by number of input feature-maps) of the weights connecting convolutional layer s to ℓ within a dense block. Three columns highlighted by black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

ex)

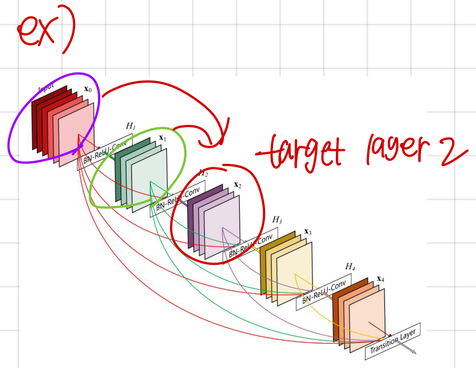


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

↳ (ℓ, s) : 빨갈수록 층 ℓ이 s층까지 생긴 f.m 사용↑ (가중치↑)
(ℓ: 현재 conv, s: 이전층)

↳ 1. All layers spread their weights over many inputs within the same block.

→ 같은 블록 안에서 대부분의 층이 여러 과거 층에 골고루 가중치를 분산한다.

→ 초반 층 저수준 특징도 깊은 층에서 직접 재사용

2. The weights of the transition layers also spread their weight across all layers within the preceding dense block,

→ Transition layer 역시 이전 블록의 여러 층 출력에 두루 가중치를 둔다

→ DenseNet의 맨 앞에서 맨 뒤까지 정보가 적은 간접 경유로 잘 흐름

3. The layers within the second and third dense block consistently assign the least weight to the outputs of the transition layer (the top row of the triangles),

→ 2.3번째 블록 내부 층들은 입력(= 직전 transition layer 출력 = heatmap 맨 윗줄)에 상대적으로 가장 적은 가중치를 둔다

→ 그래서 DenseNet-BC 가 transition 에서 θ (채널 압축)을 해도 성능 좋음 (중복이니까)

4. Although the final classification layer, shown on the very right, also uses weights across the entire dense block,

→ 맨 오른쪽 최종 분류 층도 블록 전반의 특징들에 두루 가중치 → but, 마지막 f.m쪽에 더 가중치↑ = "고수준 특징 생성"

6. Conclusion

최적화 난이도 X

파라미터 수 ↑ 해도 acc↑

과적합 x

파라미터 수 ↓

항등매핑, Deep super-vision, 다양한 깊이, Feature Reuse, Compact