6. densely connected convolutional networks

Introduction

- CNN이 점차 깊은 구조를 갖게되면서, 여러 문제들이 대두되기 시작했는데 그 중 하나가 input이 수많은 Layer를 거치게 되면서, 네트워크 끝단에 다다를수록 정보가 소실된다는 것
- 모두 앞 Layer에서 뒷 Layer로의 Short Path를 사용한다는 공통점이 있음
- 본 논문에서는 간단한 아이디어의 Connectivity pattern을 하나 제안함 바로, 네트워크의 레이어들 사이의 maximum information flow를 보장하기 위해 모든 레이어들을 서로 연결하는 것

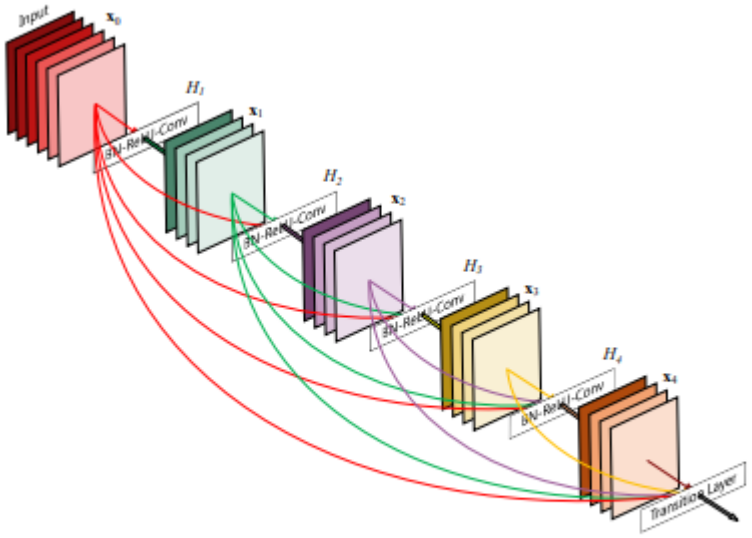


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

- 위 그림이 DenseNet의 구조를 가장 잘 나타냄
- ResNet과의 차이점 : ResNet이 Feature들이 Layer로 전달되기 전, summation을 통해 결합되는 반면 DenseNet은 Feature들을 Concatenation하여 결합함
    - ResNet의 경우 자기자신인 Identity를 더해 다음 layer의 input으로 부여하는 반면,
      DenseNet은 단순 덧셈이 아닌, feature들을 연결하여 부여함

    - 따라서 n번째 Layer에는 이전의 모든 convolution blocks들의 feature map으로 이루어진 n개의 input이 있음 이러한 feature mape들은 (L - n)개의 subsequent Layer를 지나게 되고,
      결국 L-layer network는 L(L + 1)/2 개의 Connection을 갖게 된다.
        - ResNet은 L개의 Connection을 갖게 됨
    - 모든 Layer들을 촘촘하게 연결하는 Dense Connectivity Pattern을 사용한 것이 바로 DenseNet

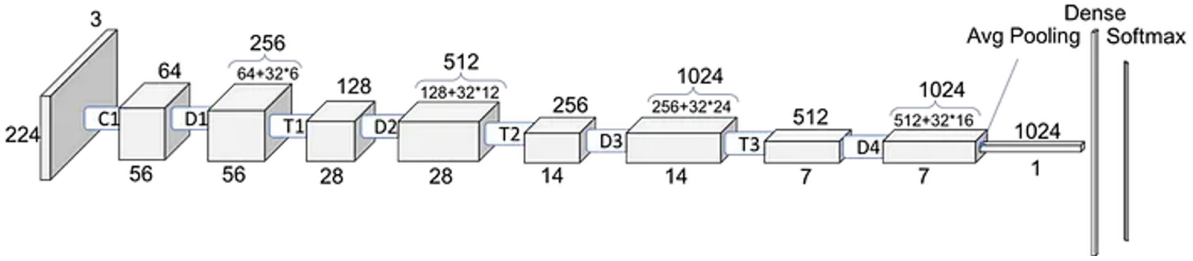
- DenseNet의 장점
    - Fewer Parameter
        - DenseNet은 모든 Layer를 연결한다는 점에서 기존 방식들보다 많은 Parameter가 필요할 것 같지만, 사실 DenseNet은 불필요한 feature map들을 재학습할 필요가 없기 때문에 더 작은 parameter를 가짐
    - Information Preserved
        - ResNet의 경우 identity transformation을 통해 정보를 보존하고 ResNet을 변형한 다양한 연구들의 경우 training 중 random하게 정보가 누락되기도 함
        - DenseNet은 네트워크에 추가되는 정보와 보존되는 정보를 명확히 구분하여 정보를 보존하며 DenseNet의 Layer들은 매우 좁은 형태를 갖는데(Layer 당 12개의 필터)네트워크의 ‘collective knowledge’에 작은 세트의 feature map만 더하며 나머지 feature map들은 변경되지 않은 상태로 유지, 마지막 classifier는 네트워크의 모든 feature-map에 기반하여 decision함
    - Improve Flow of Information and Gradient
        - DenseNet의 가장 큰 장점은 training에 용이하도록 정보와 gradient의 flow를 개선시켰다는 점
        - 각 Layer들은 Loss function과 original input signal의 gradient에 직접적으로 접근이 가능
          (모든 Layer들이 서로 연결되어 있기 때문에)
        - Overffiting을 방지하는 Regularization효과도 얻을 수 있음

REALTED WORK

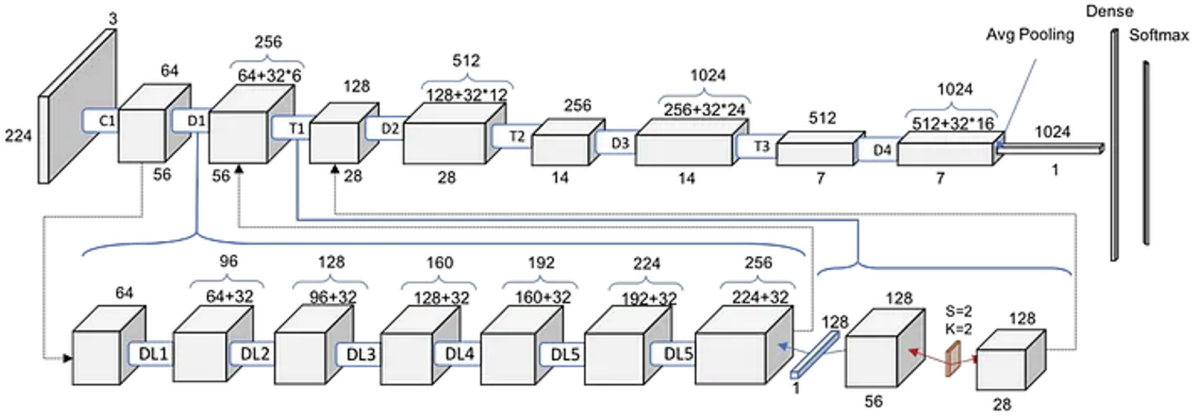
- Highway Network는 처음으로 100 Layer가 넘는 end - to - end 네트워크를 효과적으로 학습한 구조

- gate unit과 bypassing path를 이용해 최적화 했는데, bypassing path가 이 네트워크의 핵심
    - ResNet의 경우 pure identity mapping이 바로 ResNet의 bypassing path
    - GoogleNet의 경우 다양한 크기의 필터에 의해 생성된 feature map들을 concatenate한 'Inception Module'을 사용해 네트워크의 폭을 늘림
- DenseNet은 깊거나 넓은 구조를 이용해 성능을 높이는 것과 달리, feature 재사용을 통해 parameter 효율성을 높이고 훈련하기 쉬운 간단한 모델을 생성함
    - 서로 다른 Layer들에 의해 학습된 feature map들을 concatenating 함으로써, subsequent Layer들의 input 변화가 증가하고 효율성 또한 향상됨(ResNet은 비교적 깊고 넓은 구조인 반면, DenseNet은 좁은 구조)
    - ResNet뿐만 아니라, 다른 Layer들을 concatenate하는 Inception Networ의 경우와 비교했을 때에도 DenseNet은 더 단순하고 효율적은 구조를 가짐

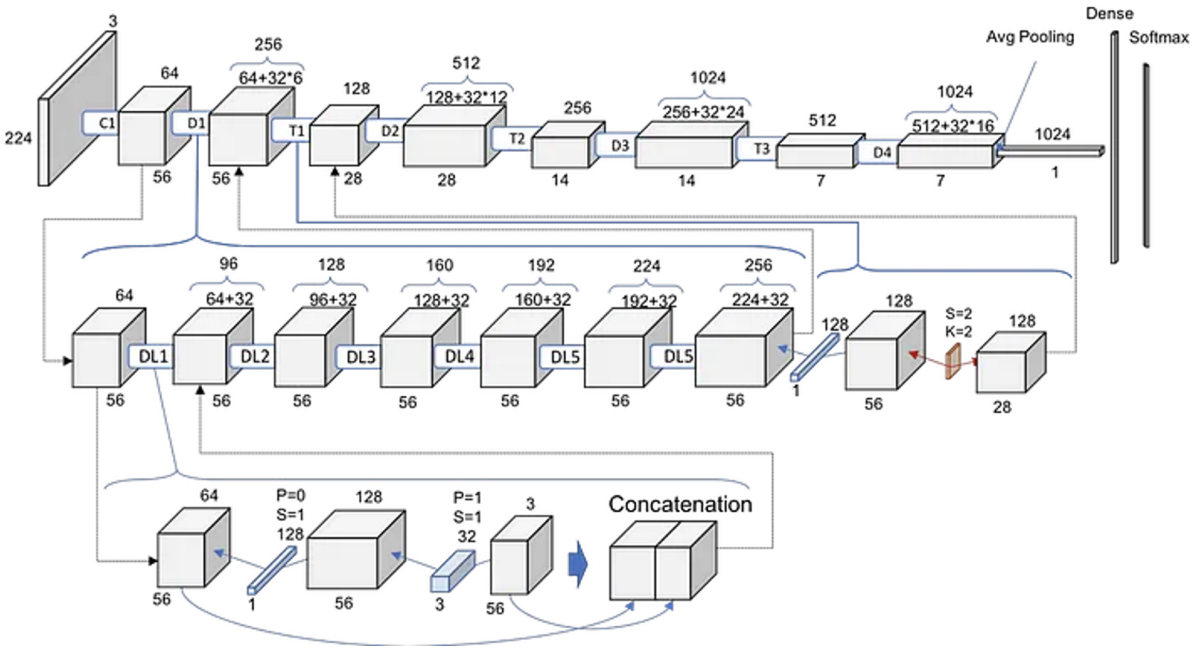
DenseNet



- One level deeper look at DenseNet-121. Dense Block and Transition Block. DLx: Dense Layer x



- One level deeper look at DenseNet-121. Dense Block and Transition Block. DLx: Dense Layer x



- 2 level deep. Full schematic representation of ResNet-121

- 단일 이미지 X0가 convolutional network를 통과할 때 네트워크는 L개의 Layer로 구성되어있고, 각 Layer에서는 non-linear transformation H(n)을 수행(n = Layer의 index)
  H는 Batch Normalization, ReLU, Pooling, Convolution과 같은 동작이며, n번째 Layer의 결과 = Xn

- ResNet
  - 기존의 Convolutional feed-forward network는
    n번째 Layer의 Output을 그다음 layer인 (n + 1)번째 Layer의 Input으로 연결
  - ResNet은 identity function을 통해 non-linear transformation을 건너뛰는 skip connection을 추가함
  - ResNet의 장점
    - indentity function을 통해 Later Layer에서 Earlier Layer로 gradient가 직접적으로 흐를 수 있다는 점
      - indentity function과 H의 output이 summation을 통해 합쳐지면서 네트워크의 information flow를 방해할 수 있음

- Dense Connectivity

- Layer들 간의 information flow를 향상시키기 위해, 다른 connectivity pattern을 제안함
  - 어떤 Layer든 subsequent Layer로 직접적으로 연결하는 모든 Layer를 연결하는 방식

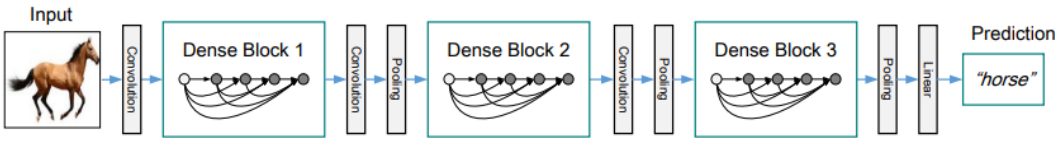


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

  - DenseNet의 구조를 도식화한 그림인데 결과적으로 n번째 Layer는 이전의 모든 Layer들의 feature map(x0, ..., xn -1)을 input으로 받게됨

$$x_l = H_l([x_0, x_1, ..., x_l - 1)$$

  - [X0, X1,...]은 각 Layer에서 만들어진 feature map들의 concatenation, 이러한 Dense connectivity로 인해서 DenseNet이라고 부름

- Composite Function
  - 다른 연구에서 영감을 받아 H(n)을 다음 3가지 operation의 composite function으로 정의함
    → Batch Normalization, ReLU, 3 × 3 Convolution

- Pooling Layers
  - 위 식에서 사용된 concatenation operation은 feature map의 사이즈가 바뀌면 사용할 수 없지만 Convolution Network의 down-sampling Layer를 통해 feature map의 사이즈를 바꿀 수 있음
  - down-sampling을 용이하게 하기위해, 네트워크를 서로 밀집하게 연결된 여러개의 dense block으로 나눔

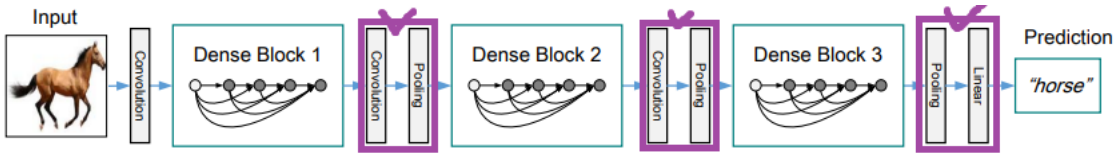


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

    - 각 Block 사이에서 Convolution과 Pooling을 수행하는 Layer(보라색 박스)를 Transition Layer라고 하며 본 논문에서의 Transition Layer는 Batch Normalization Layer, 1 × 1 Convolution Layer, 2 × 2 Pooling Layer로 구성되어 있음

- Growth Rate
    - 각각의 function H(n)이 K개의 feature map을 만든다고한다면, n번째 Layer의 feature map의 갯수는 K0 + K(n - 1)개가 됨(K0 = input Layer의 채널 수)
    - DenseNet과 기존 네트워크 구조들 간의 차이점은 DenseNet이 비교적 좁은 layer를 가졌다는 점(filter = 12), 여기서 filter(K)를 네트워크의 Growth Rate라고 함
        - Growth Rate란, 각 Layer에서 몇 개의 feature map을 뽑을지 결정하는 Parameter이며, 각 Layer가 전체 output에 어느정도 기여할지를 결정하는 Parameter이기도 함

- BottleNeck Layers
    - 각 Layer들은 K개의 output feature map을 생성하지만, 일반적으로 그에 비해 더 많은 input이 필요함
    - BottleNeck 구조는 1 x 1 convolution을 통해 3 x 3 convolution에서의 input을 줄여 computational efficiency를 높일 수 있음 → DenseNet에서도 이러한 BottleNeck구조를 사용

- Compression
    - 모델의 compactness를 향상시키기 위해, transition Layer의 feature map의 개수를 줄일 수 있음
    - 만약 Dense block이 m개의 feature map을 가지고 있다면 그 뒤에 transition Layer는 θm개의 output feature map을 반환 (θ = Compression factor) 단, (0 < θ ≤ 1)
    - 만약 θ < 1인 경우의 DneseNet은 DenseNet-C라고 부르며 논문에서의 실험은 θ = 0.5로 설정함
        - BottleNeck, Transition Layer 모두 θ < 1인 경우, DenseNet-BC라고 이야기

- Implementation
  - DenseNet은 모두 동일한 갯수의 Layer를 가진 3개의 dense Block으로 구성
  - 첫번째 Dense Block을 들어가기 전, input image에 대해 convolution with 16 output channel을 수행
  - feature map size를 고정하기 위해, kernel_size가 3 x 3인 convolution layer에서의 input은 모두 1pixel씩 zero-padding
  - 인접한 두 Dense Block 사이의 Transition Layer로는 1 x 1 Convolution, 2 × 2 average pooling을 사용
  - 마지막 Dense Block의 끝에는 global average pooling, softmax, classifier가 실행
  - 3개의 Dense Block의 feature map size는 각각 32 x 32, 16 × 16, 8 × 8
  - DenseNet-BC를 사용했으며, 224 * 224 이미지를 input, Dense Block의 갯수는 4개

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.

EXPERIMENTS

- DATASET
  - CIFAR, SVHN, ImageNet

- Training
  - Optimization : SGD

- Batch Size : 64(CIFAR & SVHN), 256(ImageNet)
- Epoch : 300(CIFAR), 40(SVHN), 90(ImageNet)
- Learning rate : 0.1
  - CIFAR & SVHN : divided by 10 at 50% and 75% of the total number of training epochs
  - ImageNet : lowered by 10 times at epoch 30 and 60
- Wight Decay : 0.0001 (Nesterov Momentum of 0.9)
- Dropout : 0.2 (C10, C100, SVHN)

- Classification Result on CIFAR and SVHN
  - 서로 다른 깊이(L) 및 Growth Rate(K(Filter))로 DenseNet 훈련을 진행
  - General trends를 나타내기 위해 기존 기술보다 높은 성능을 보인 결과는 굵은글씨
  - 전체적으로 최상의 결과를 보인 네트워크는 파란색으로 표기

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.

- Accuracy

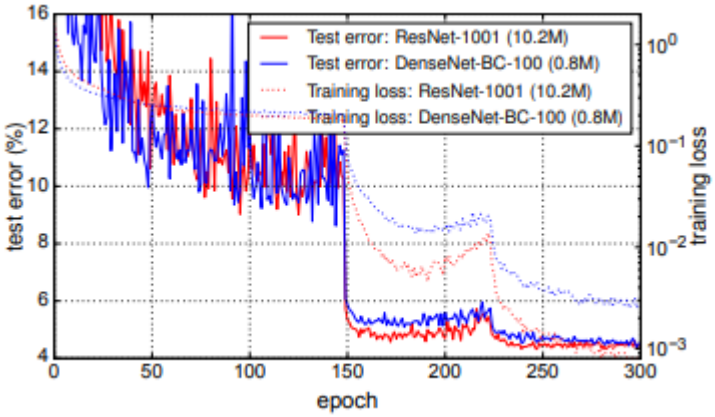
- DenseNet-BC, 그 중 특히 k = 40의 경우 CIFAR dataset에 대해 기존보다 훨씬 높은 성능을 보임
- C10+, C100+에서 각각 3.46, 17.18%의 error rate를 보임
  - wide ResNet 구조에서 달성한 Error Rate보다 훨씬 낮은 수치
- C10과 C100에 대한 결과에서 모두 drop-path regularization을 사용한 Fractal Net보다 30%가까이 낮음
- Dropout을 사용한 SVHN dataset에서는 k= 24인 DenseNet이 가장 좋은 결과를 보임
  250-Layer DenseNet-BC는 shortcut counterpart에 비해 더 좋은 성능을 내지 못함
  SVHN이 비교적 쉬운 dataset에 속하며, 너무 깊은 모델의 경우 training set에 overfit되기 쉽기 때문에 이러한 현상이 발생한것으로 보고 있음

- Capacity
  - Compression이나 BottleNeck Layer가 없는 DenseNet의 경우, L과 K가 증가할수록 높은 성능을 보임
  - Model Capacity의 Growth와 주로 일치하는 것으로 보였는데, C10+와 C100+에서 이를 확인할 수 있음
    - 사용한 CIFAR DATA : CIFAR-10(C10), CIFAR-100(C100)
    - C10+, C100+은 data augmentation을 거친 dataset의 이름에+를 붙였음
  - C10+의 경우, parameter의 개수가 증가할수록, Error가 점차 줄어들었음.
  - C100+ 또한 비슷한 양상을 보였음
  - DenseNet이 크고 깊은 모델의 representational power를 이용할 수 있으며, residual network에서 발생하는 overfitting이나 optimization문제가 발생하지 않음을 알 수 있음

- Parameter Efficiency
  - DensNet이 다른 구조들에 비해 Parameter를 더 효과적으로 사용할 수 있음을 알 수 있으며 BottleNeck 구조와 Transition Layer에서의 Dimension reduction을 수행한 DenseNet-BC가 특히 그러함
    - 예를들어 15.3M개의 parameter를 가진 250-Layer모델이 다른 30M개 이상의 parameter를 갖는 여러 모델들(FractalNet, Wide ResNet)보다 높은

성능을 보임

- 1001-Layer의 ResNet 모델과 비교했을 때, DenseNet-BC가 90% 정도 적은 parameter를 가졌음에도 비슷한 성능을 보임



- C10+의 2개의 네트워크에 대한 Training Loss와 Test Error를 보여주며 1001-Layer의 깊은 ResNet의 Training Loss값은 낮게 수렴하지만, Test Error는 비슷함

- Overfitting
  - parameter를 더 효율적으로 사용함에 따라 발생하는 긍정적인 부작용 중 하나는 DenseNet이 Overfitting 되는 경향이 적다는 것
  - 논문에서는 data augmentation을 하지 않은 dataset에서 DenseNet 구조의 improvement가 더 잘 두드러진다는 것을 확인했으며, single setting에서의 잠재적인 overfitting을 확인함
    - 이를 해결하기 위해 DenseNet-BC BottleNeck and Compression Layer가 효과적일 것으로 보임

- Classification Results on ImageNet
  - DenseNet-BC의 Depth와 Growth Rate를 바꿔가며 진행했으며, 기존 ResNet 구조와 비교함
  - 두 구조간의 공정한 비교를 위해, Torch implementation 된 ResNet을 사용함으로써, 데이터 전처리나 최적화 세팅 같은 다를 수 있는 요소들을 모두 제거함
  - 단순히 ResNet 모델을 DenseNet-BC 네트워크로 바꿨으며, 실험 환경은 ResNet과 동일하게 설정

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.

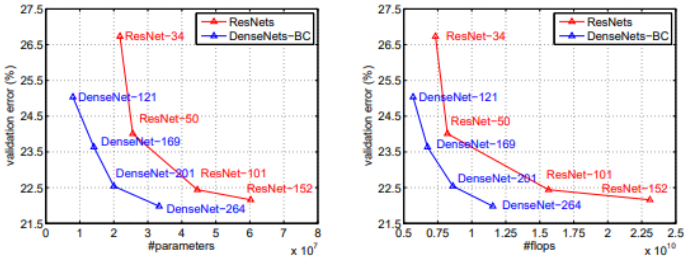


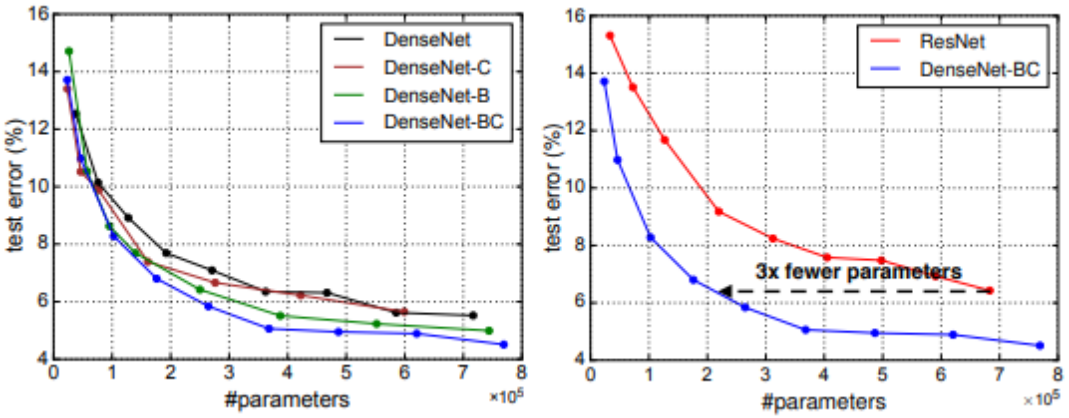
Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

- ImageNet에 DenseNet을 적용한 single-crop과 10-crop validation error결과는 위의 사진과 같음, DenseNet의 single-crop top1 validation error값과의 비교를 위한 그림이며 왼편은 ResNet의 parameter, 오른편은 FLOP에 대한 결과
- DenseNet의 parameter수와 computation이 훨씬 적음에도 불구하고, ResNet과 비슷한 성능을 내는 것을 확인할 수 있음
- 본 논문에서는 실험 설정이 ResNet에는 최적화 되었지만, DenseNet에는 최적화 되어 있지 않다는 점 또한 주목할 필요가 있고 만약 ImageNet에 맞는 DenseNet의 parameter를 보다 최적화한다면, 더 높은 성능을 낼 수 있을것으로 기대

DiSCUSSION

- 표면적으로 DenseNet의 구조는 ResNet과 유사하지만 두 구조간의 작은 차이는 실질적으로 두 네트워크 구조 간의 아주 큰 차이로 이어짐

- Model Compactness
  - input concatenation의 직접적인 결과로, DenseNet에서 학습된 feature map들은 모든 subsequent Layer에 접근이 가능함
  - 네트워크 전체의 feature 재사용성을 높이고, 보다 compact한 모델이 되도록 함



    - 왼쪽은 DenseNet의 parameter efficiency를 비교하는 그래프
    - 오른쪽은 ResNet 구조와 DenseNet을 비교하는 그래프
    - C10+에서 다양한 깊이의 여러 소규모 네트워크들을 train하고,
      network parameter들의 function으로 test 정확도를 그래프화함

    - AlexNet이나 VGGNet같은 여러 유명한 네트워크 구조와 비교했을 때,
      pre-activation된 ResNet은 적은 aprameter에도 일반적으로 더 나은 결과를 냄
    - DenseNet이 ResNet에 비해 parameter 대비 더 좋은 성능을 보임
        - DenseNet의 model이 더 compact하고 fewer parameter를 가짐

- Implicit Deep Supervision
    - Dense Convolution Network의 accuracy가 향상이 되는 이유는 각각의 Layer들이 Short Connection을 통해 Loss Function의 추가적인 Supervision을 받기 때문
        - DenseNet을 이용해서 Deep Supervision을 수행할 수 있음
    - DenseNet은 implicit한 방식으로 Deep Supervision을 수행함
    - 네트워크 상단의 single classifier는 꾀대 2~3개의 transition Layer를 통해 모든 Layer를 직접적으로 supervision
        - DenseNet의 loss function과 gradient의 경우, 모든 Layer에서 같은 loss function을 공유하기 때문에 훨씬 덜 복잡함

- Stochastic vs deterministic connection
    - Dense convolutional networ와 Residual network의 Layer들을 랜덤하게 Drop하여 주변 Layer들 간의 Direct Connection이 생성됨
    - Pooling Layer는 Drop하지 않기 때문에 네트워크는 DenseNet과 유사한 Connectivity Pattern을 가짐
    - 방법은 궁극적으로 다를지 몰라도, stochastic depth에 대한 DenseNet interpretation은 정규화의 성공에 대한 통찰력을 제공함

- Feature Reuse

- DenseNet은 현재 Layer가 이전의 모든 Layer들의 feature map에 접근할 수 있도록 설계되었음
- 논문에서 train된 Network가 이러한 opportunity를 잘 활용하는지 조사하기 위해서 실험을 수행함
- C10+, k = 12, L = 40 인 DensNet을 훈련
- 각 블록 내의 Convolutional Layer 'l'에 대해 Layer 's'와 연결에 할당된 average weight를 계산

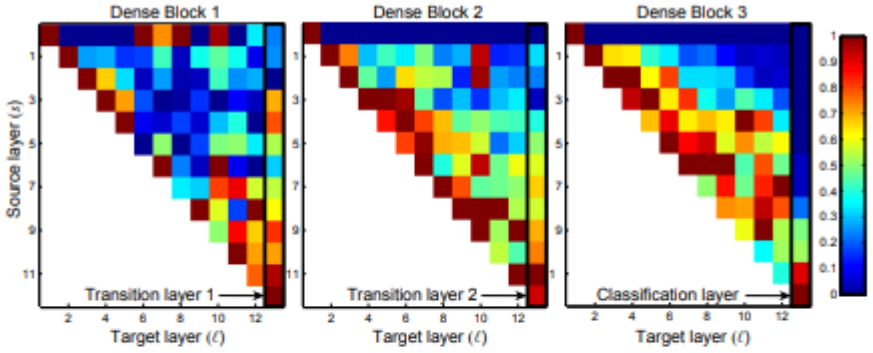


Figure 5: The average absolute filter weights of convolutional layers in a trained DenseNet. The color of pixel (s, ℓ) encodes the average $L1$ norm (normalized by number of input feature-maps) of the weights connecting convolutional layer s to ℓ within a dense block. Three columns highlighted by black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

- 3개의 Dense Block에 대한 Heat Map
  - average absolute weight = 이전 Layer에서 Convolution Layer에 대한 의존성 정도
  - 빨간색 점(l, s)은 Layer 'l'이 이전 Layer 's'에서 생성된 feature-map을 많이 사용하고 있음을 나타냄

CONCLUSION

- 일명 DenseNet이라고 불리는 새로운 Convolutional Network의 구조에 대해 설명함
- 같은 feature map size를 가진 어떤 2개의 layer들에 대한 direct connection에 대해 소개함
- optimization difficulty없이 Layer의 규모를 늘려나갈 수 있다는 것을 증명

- 성능 저하나 overfitting 없이 parameter의 갯수가 증가할수록 정확도가 향상되는걸 확인할 수 있음
- 다른 구조(모델)들에 비해 저은 parameter 갯수와 연산량으로 더 좋은 결과를 냄
- 모든 Layer들을 연결한다는 간단한 connectivity rule을 따르면서도 identity mapping, deep supervision, diversified depth 등의 특징을 모두 실현
- feature reuse를 더 용이하고, model을 더 compact하게 만듬
- DenseNet은 compact한 internal representations, feature redundacy 감소라는 장점을 통해 다양한 컴퓨터 비전 분야에서 훌륭한 feature extractor로 자리잡을 것임