📖

[논문 리뷰] DenseNet [2017]

Introduction

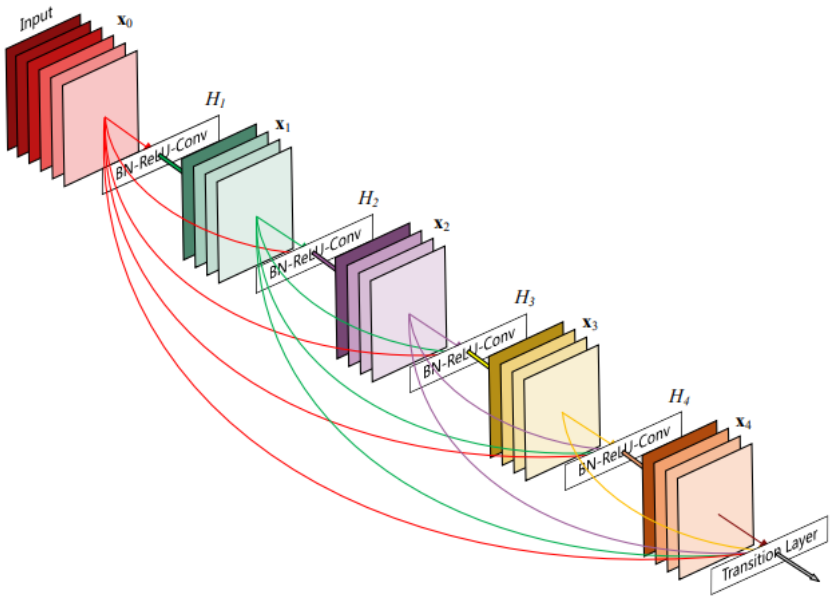
🌟 해당 논문은 이전의 Short Connection 개념에 집중하여 feed-forward fashion 방식에서 각 layer와 each layer를 Connect하는 방법을 사용한 **DenseNet Model을 소개**합니다.

DenseNet은 $\frac{L(L+1)}{2}$개의 Direct Connection을 활용하는 구조를 나타냅니다.

앞의 layer에서 얻은 모든 Feature map은 모든 preceding layer에 연결되는 방식입니다.

DenseNet Advantages

- Alleviate the Vanishing Gradient
- Strengthen the feature Propagation
- Encourage feature reuse
- substantially reduce the number of parameters



CNN 모델이 깊어질수록(Deep), input의 정보나 gradient가 layer를 거치면서, **Vanish & Washout** 문제가 발생합니다.

DenseNet은 모든 layer를 연결함으로서, Information을 최대한 전달하는 것에 목적을 두었습니다.

Feed-forward를 보존하기 위해, 각각 layer는 **모든 이전 layer들로부터 input을 추가로 받아 모든 layer의 feature map을 통과**시킵니다.

DenseNet layer는 narrow하고, 적은 양의 “Collective Knowledge(Feature-map)”를 사용함으로서, feature-map을 바꾸지 않게 합니다.

DenseNet의 장점은, **gradient와 information의 흐름이 개선**되었다는 것이다. 모든 layer와 선행된 다른 layer과의 연결이 있기 때문에, **loss function이나 input signal의 gradient 역시 직접적으로 접근하는 것이 가능**해졌습니다.

DenseNet은 **regularizing effict를 사용하기에, Overfitting문제로부터 상당히 자유롭다는 것**을 알 수 있습니다.

DenseNets

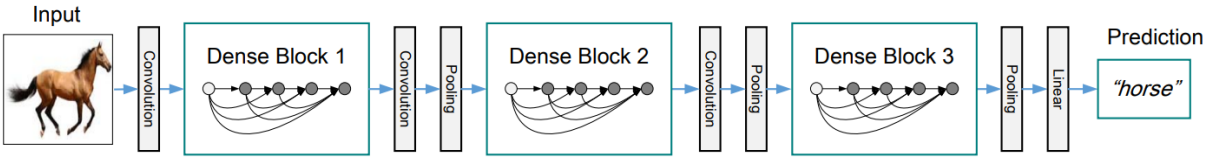
🌟 **Dense connectivity**

모든 layer 뒤에 존재하는 layer들을 연결해주는 새롭게 고안된 방법이다.

결과적으로, **l번째 layer는 이전에 존재하는 l개의 feature map(x까지 포함)을 입력값으로 받는다.**

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]), \quad (2)$$

where $[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]$ refers to the concatenation of the feature-maps produced in layers $0, \ldots, \ell-1$. Because of its dense connectivity we refer to this network architecture as *Dense Convolutional Network (DenseNet)*. For ease of implementation, we concatenate the multiple inputs of $H_\ell(\cdot)$ in eq. (2) into a single tensor.



Pooling layers

- feature-map의 사이즈가 달라지는 경우에 concatenation의 문제가 발생할 수 있습니다.
  (ResNet의 경우, indentity function을 사용하지만,
  ResNet의 torch의 경우, concatenate을 사용합니다.)
- 해당 부분을 해결하기 위해, down-sampling을 이용하여, feature map의 크기를 변경합니다.
- DenseNet의 실험에 사용된 **transition layer는 batch_normalization과 1x1 Conv layer, 2x2 Avg Pooling layer**로 이루어져 있습니다.

Growth rate

- 각 H의 함수에서는 k개의 feature map을 만듭니다. DenseNet은 다른 Network에 비해, layer가 좁다는 것이 특징입니다.

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.

Experiment

- DenseNet은 ResNet과 상당히 유사합니다. **특정 층에서 학습된 feature map들은 뒤의 모든 층에서 접근**이 가능하다.
- 따라서, 모든 **NetWork를 통해서 feature를 다시 사용가능하게 하여 Model의 Compact를 개선**하였습니다.

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.

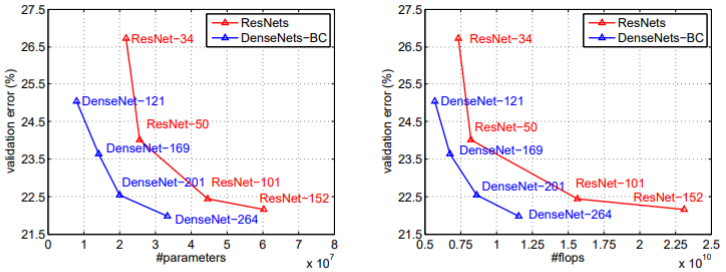


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

Conclusion

DensNet은 뛰어난 성능을 보이는 CNN모델이며, 적은 parameter와 적은 computation을 통해 좋은 성능을 얻었습니다.

DenseNet을 통하여, 특징적인 feature를 다른 모델에서 학습시킨 뒤, 새로운 예측을 진행하는 transfer learning방식에 대한 연구를 하면 더 좋은 성능을 낼 수 있을 것이라고 본다.

▼ **Model [Pytorch]**

```
from torchsummary import summary
import torch
import torch.nn as nn

class BottleNeck(nn.Module):
    def __init__(self, in_channels, growth_rate):
        super().__init__()
        inner_channels = 4 * growth_rate

        self.residual = nn.Sequential(
            nn.BatchNorm2d(in_channels),
            nn.ReLU(),
            nn.Conv2d(in_channels, inner_channels, 1, stride = 1, padding = 0, bias = False),
            nn.BatchNorm2d(inner_channels),
            nn.ReLU(),
            nn.Conv2d(inner_channels, growth_rate, 3, stride = 1, padding = 1, bias = False)
        )

        self.shortcut = nn.Sequential()

    def forward(self, x):
        return torch.cat([self.shortcut(x), self.residual(x)], 1)

class Transition(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()

        self.down_sample = nn.Sequential(
            nn.BatchNorm2d(in_channels),
            nn.ReLU(),
            nn.Conv2d(in_channels, out_channels, 1, stride = 1, padding = 0, bias = False),
            nn.AvgPool2d(2, stride = 2)
        )

    def forward(self, x):
        return self.down_sample(x)
```

```
class DenseNet(nn.Module):
    def __init__(self, nblocks, growth_rate = 12, reduction = 0.5, num_classes = 10):
        super().__init__()

        self.growth_rate = growth_rate
        inner_channels = 2 * growth_rate

        self.conv1 = nn.Sequential(
            nn.Conv2d(3, inner_channels, 7, stride = 2, padding = 3),
            nn.MaxPool2d(3, 2, padding = 1)
        )

        self.features = nn.Sequential()

        for i in range(len(nblocks) -1):
            self.features.add_module('dense_block_{}'.format(i), self._make_dense_block(nblocks[i], inner_channels))
            inner_channels += growth_rate * nblocks[i]
            out_channels = int(reduction * inner_channels)
```

```
            self.features.add_module('transition_layer_{}'.format(i), Transition(inner_channels, out_channels))
            inner_channels = out_channels

        self.features.add_module('dense_block_{}'.format(len(nblocks)-1), self._make_dense_block(nblocks[len(nblocks)-1], inner_ch
        inner_channels += growth_rate * nblocks[len(nblocks)-1]
        self.features.add_module('bn', nn.BatchNorm2d(inner_channels))
        self.features.add_module('relu', nn.ReLU())

        self.avg_pool = nn.AdaptiveAvgPool2d((1,1))
        self.linear = nn.Linear(inner_channels, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.features(x)
        x = self.avg_pool(x)
        x = x.view(x.size(0), -1)
        x = self.linear(x)
        return x

    def _make_dense_block(self, nblock, inner_channels):
        dense_block = nn.Sequential()
        for i in range(nblock):
            dense_block.add_module('bottle_neck_layer_{}'.format(i), BottleNeck(inner_channels, self.growth_rate))
            inner_channels += self.growth_rate
        return dense_block

def DenseNet_121():
    return DenseNet([6, 12, 24, 6])
```

Densely Connected Convolutional Networks

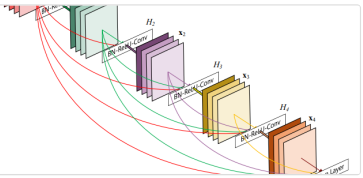
Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network

https://arxiv.org/abs/1608.06993

Dense Net(2018)논문 정리

안녕하세요, 블로그에 처음 공부 내용을 정리하게 되었습니다. 저도 아직 가야할 길이 멀게만 느껴지지만, 열심히 공부하는 누군가가 저의 글을 보고 함께 달릴 수 있으시길 바라며 글을 올립니다. arxiv.org/abs/1608.06993 Densely Connected Convolution Networks - Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q.

https://aijyh0725.tistory.com/2



DenseNet Tutorial [1] Paper Review & Implementation details

안녕하세요, 오늘은 오랜만에 Image Classification 분야의 논문을 리뷰하고,코드로 구현하는 과정을 설명드릴 예정입니다. 오늘 리뷰할 논문은 DenseNet으로 잘 알려져 있는 CNN architecture를 다룬 "Densely Connected Convolutional Networks" 이라는 논문입니다. 2017년 CVPR Best Paper Award를 받았으며 아이디어가 참신하고

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