

Pytorch를 활용한 딥러닝 학습 환경 구축 및 실습

- 3일차 -

강 수 명

smgang.kmu@gmail.com

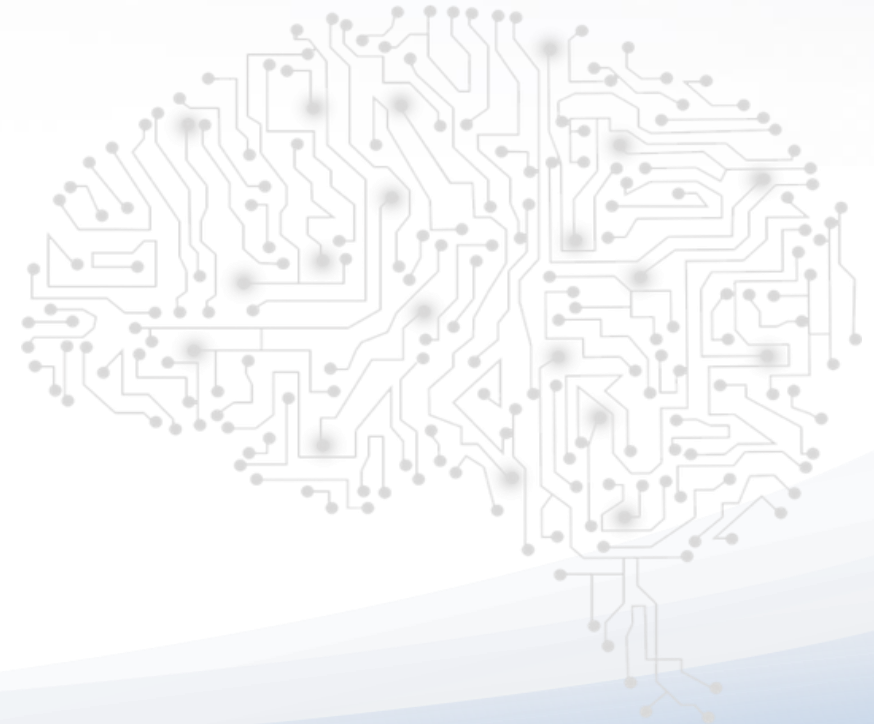
C

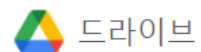
ONTENTS

I _ResNet

II _Transfer Learning

III _GAN





드라이브

🔍 드라이브에서 검색



새로 만들기



내 드라이브



컴퓨터



공유 문서함



최근 문서함



중요 문서함



휴지통



저장용량

15GB 중 1.48GB 사용

저장용량 구매

내 드라이브 ▾




바로가기로 간단하게 내 드라이브 사용하기

앞으로 몇 주 내에 두 개 이상의 폴더에 보관된 항목이 바로가기로 대체됩니다. 파일 및 폴더 액세스 권한은 변경되지 않습니다. [자세히 알아보기](#)



추천



pytorchStudy_GitClone.ipynb

오늘 수정함

chapter4.ipynb

참고 코드

- https://github.com/ayooshkathuria/YOLO_v3_t...
- https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/PyTorch/object_detection
- https://github.com/FackelPublishing/PyTorch-Computer-Vision-Cookbook
- https://github.com/robertdodder/darknet

chapter4.ipynb

강수영님이 지난주에 수정함

ML/DL for Everyone Season2


with PYTORCH

PyTorch Basic Tensor Manipulation I



Lab-01-1 Tensor Manipula...

지난달에 열어봄



명성대정탐6 11화(上).srt

지난달에 열어봄



Lifelong GAN: Continual Learning for Conditional Image Generation

2019. 11. 12

20191112_최최정.pdf

지난달에 열어봄

이름 ↑

소유자

마지막으로 수정한 날짜

파일 크기

data_lmdb_release.zip	Jeonghun Baek	2020. 4. 26. Jeonghun Baek	18.78GB
pytorchStudy_GitClone.ipynb	나	오후 10:56 나	1KB



pytorchStudy_GitClone.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

파일



- ..
- gdrive
 - MyDrive
 - data_lmdb_release.zip
 - pytorchStudy_GitClone.ipynb
 - sample_data
 - README.md
 - anscombe.json
 - california_housing_test.csv
 - california_housing_train.csv
 - mnist_test.csv
 - mnist_train_small.csv
 - adc.json

+ 코드 + 텍스트

✓ 2초 #구글 드라이브와 Colab 연동

```
from google.colab import auth
auth.authenticate_user()
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).

✓ 0초 [10] `cd /content/gdrive/MyDrive/`

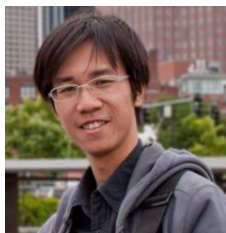
/content/gdrive/MyDrive

✓ 4초 [14] `!git clone https://github.com/smgang/pytorch_Study.git`

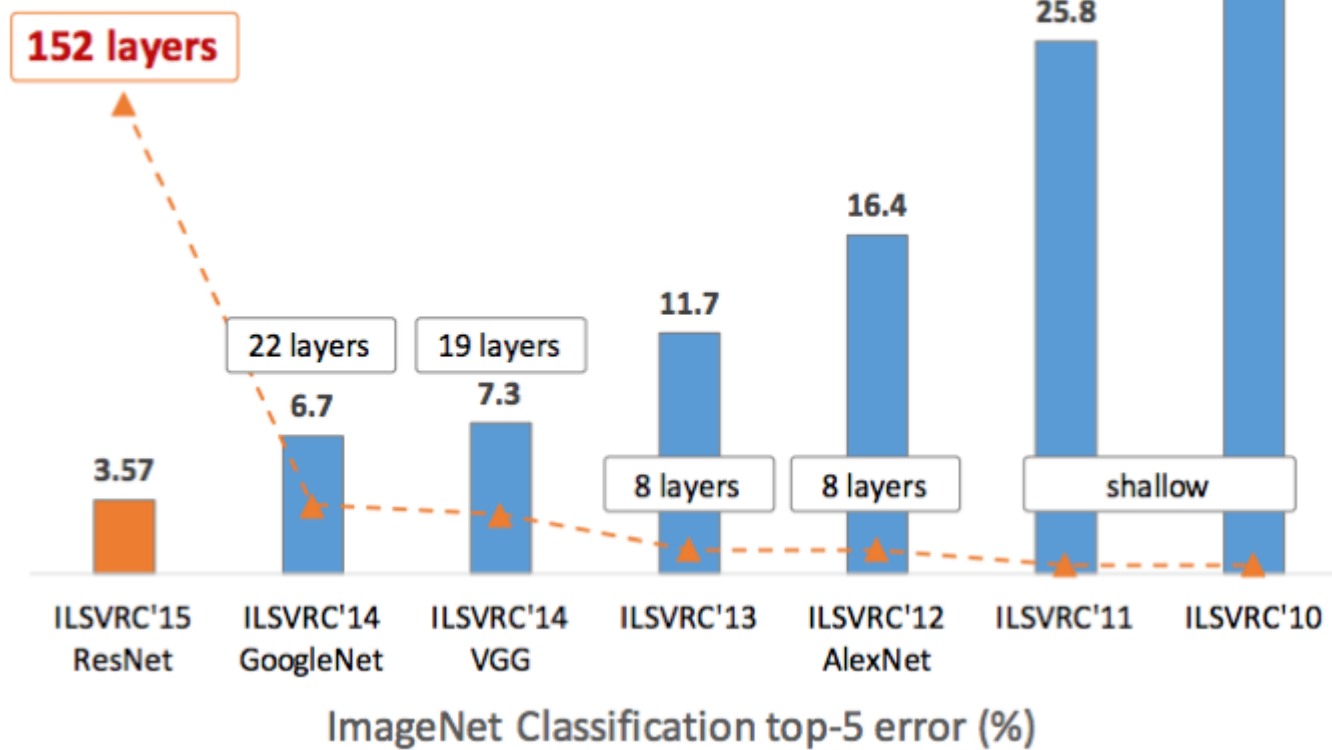
```
Cloning into 'pytorch_Study'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 22 (delta 1), reused 19 (delta 1), pack-reused 0
Unpacking objects: 100% (22/22), done.
```

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
 Microsoft Research
 {kahe, v-xiangz, v-shren, jiansun}@microsoft.com



Revolution of Depth



본격적으로 인간 인식률(94.90%)를 추월

◆ 망을 깊게 하면 결과가 더 좋아질까? : Degration

질문의 배경은 VGG로 인한 좋은 결과로 인해서..

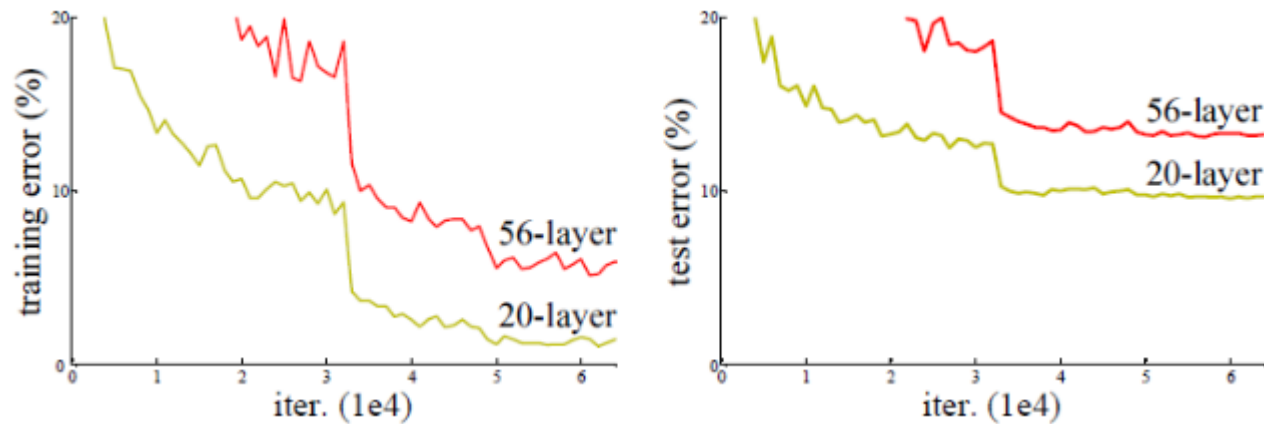
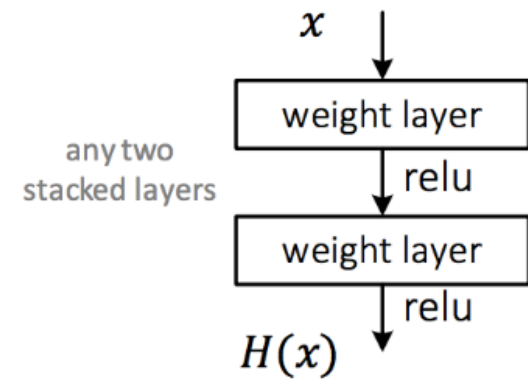


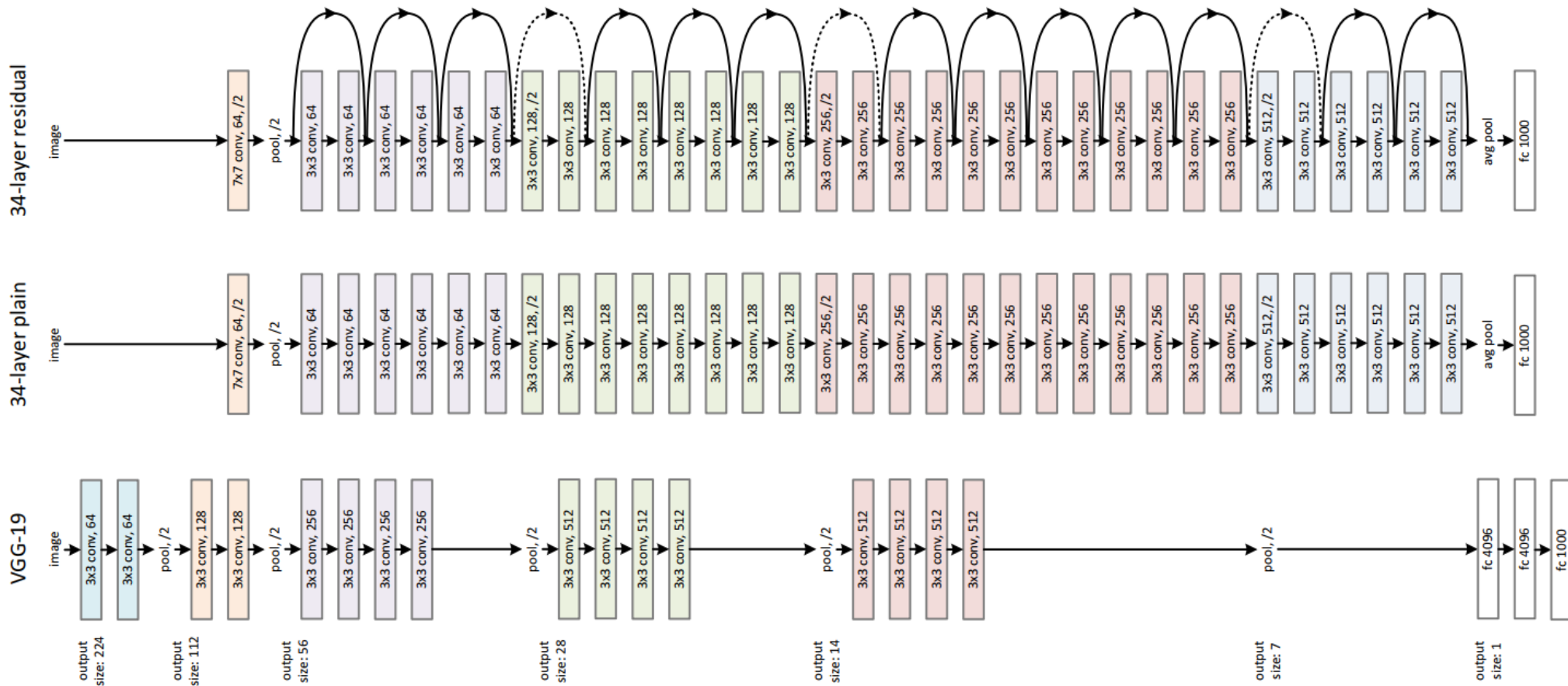
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

• Plain net

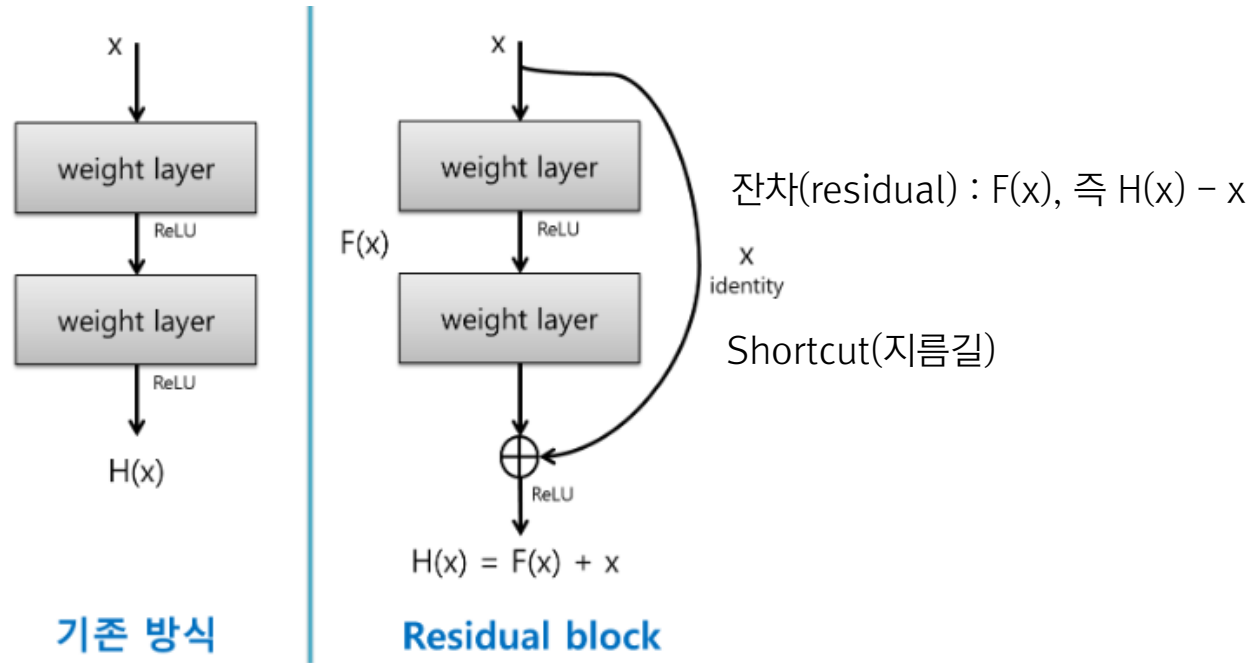


Plain Net이란 단순히 layer를 쌓은 네트워크를 의미 (ex : VGG19)

◆ 망을 깊게 하면 결과가 더 좋아질까? : Degration



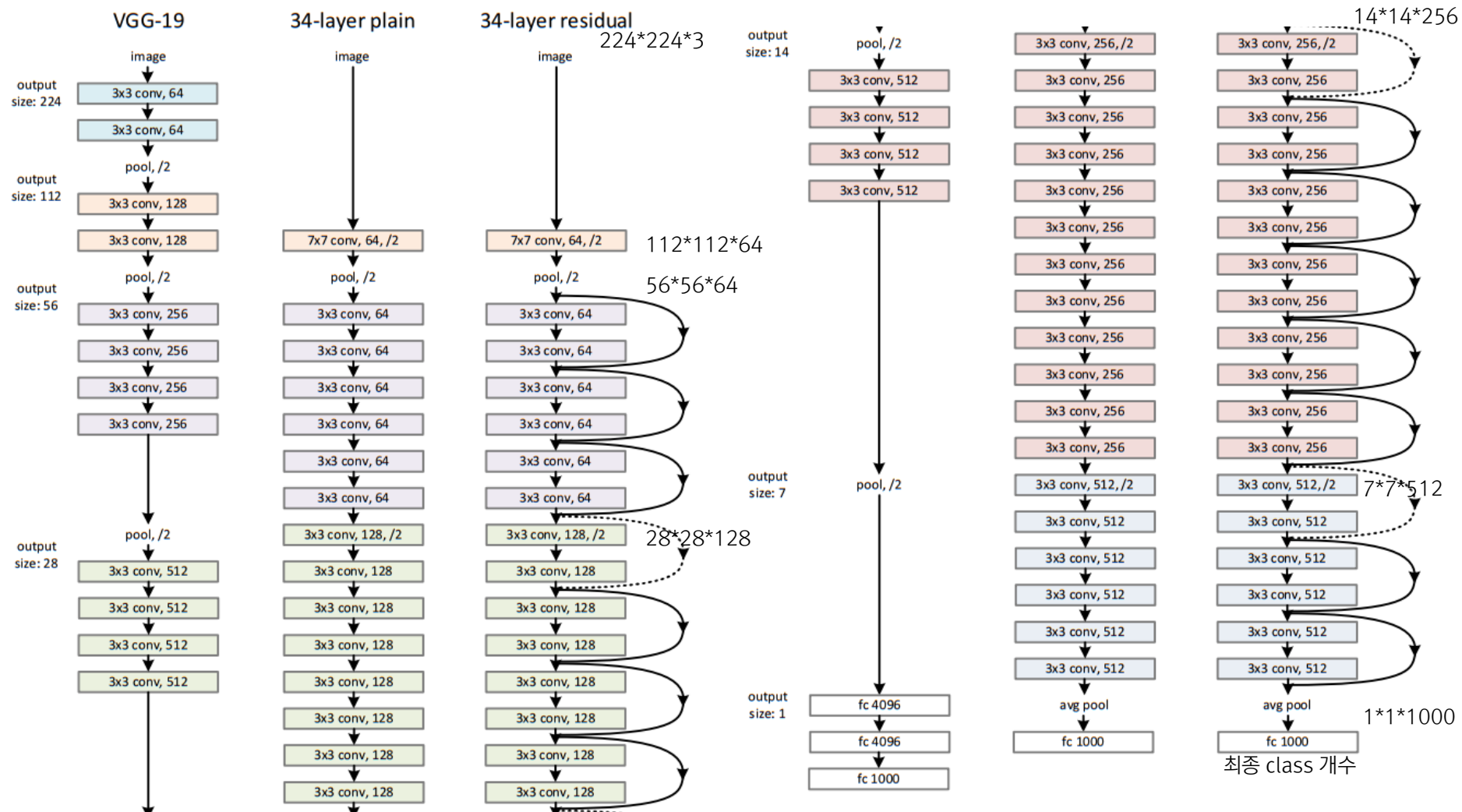
◆ 잔차를 학습하는 방법



- 기존의 신경망은 입력값 x 를 타겟값 y 로 매핑하는 함수 $H(x)$ 를 얻는 것이 목적
- ResNet은 $F(x) + x$ 를 최소화하는 것을 목적
- x 는 현시점에서 변할 수 없는 값이므로 $F(x)$ 를 0에 가깝게 만드는 것이 목적
- $F(x)$ 가 0이 되면 출력과 입력이 모두 x 로 같아짐
- $F(x) = H(x) - x$ 이므로 $F(x)$ 를 최소로 해준다는 것은 $H(x) - x$ 를 최소로 해주는 것과 동일
- 입력으로 들어온 “이전 레이어의 출력”과, “다음 레이어의 출력”이 동일하도록 함
- 즉, 앞의 기억을 잊지 않는 “잔차”레이어를 만드는 것이 목적

◆ 모델 구조

기본구조 (ImageNet-ILSVRC) ($224 \times 224 \times 3$)



◆ 모델 구조

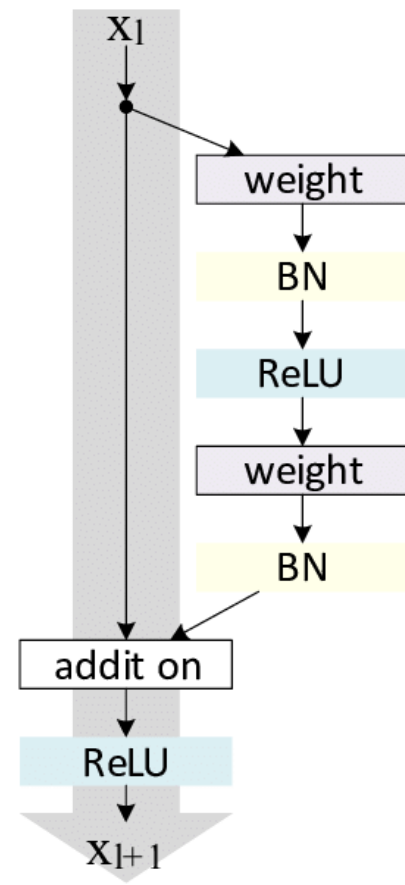
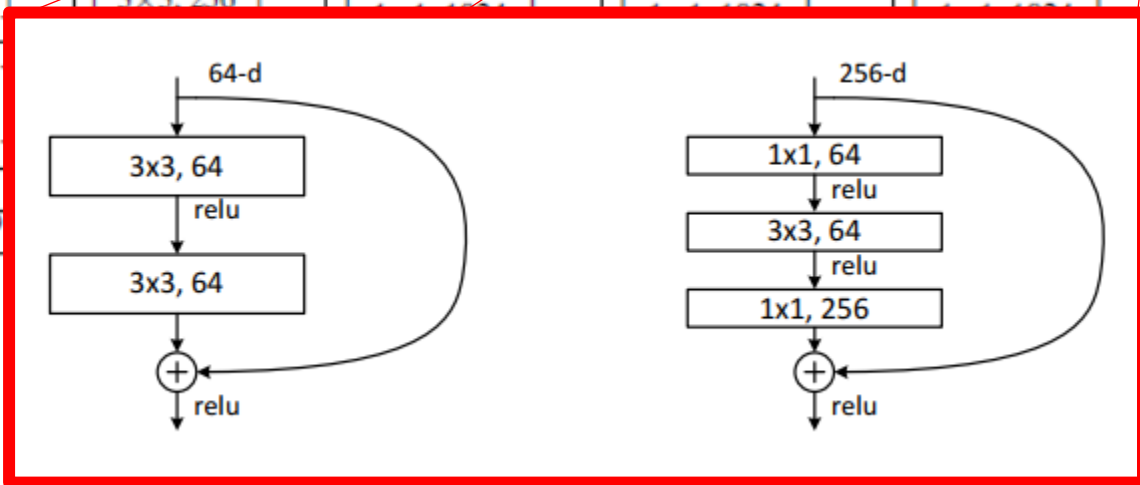
기본구조 (ImageNet-ILSVRC) (224*224*3)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

◆ 모델 구조

Bottleneck?

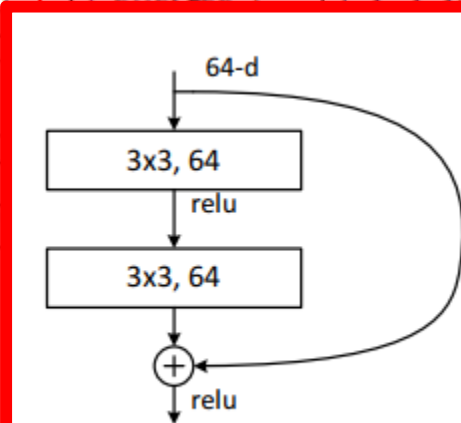
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 256 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 256 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$				
	1×1					
FLOPs		1.8×10^9				



◆ 모델 구조

Bottleneck?

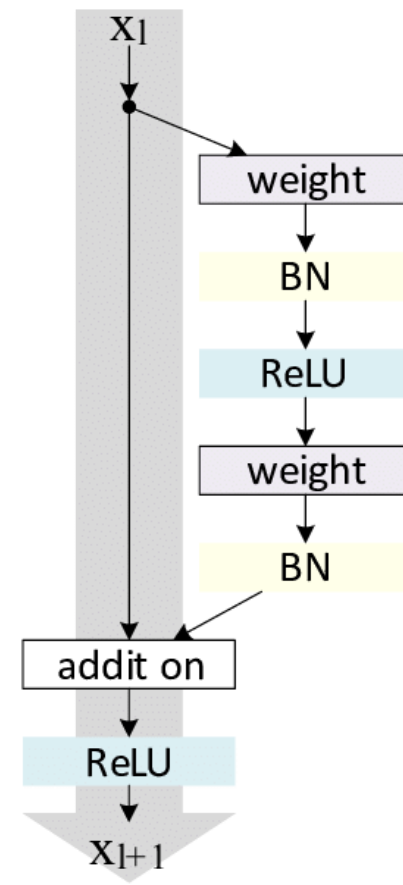
layer name	output size	18-layer	34-layer	50-layer
conv1	112×112	7×7, 64		
conv2_x	56×56	3×3 max pool		
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \end{bmatrix}$
	1×1			
FLOPs		1.8×10^9		



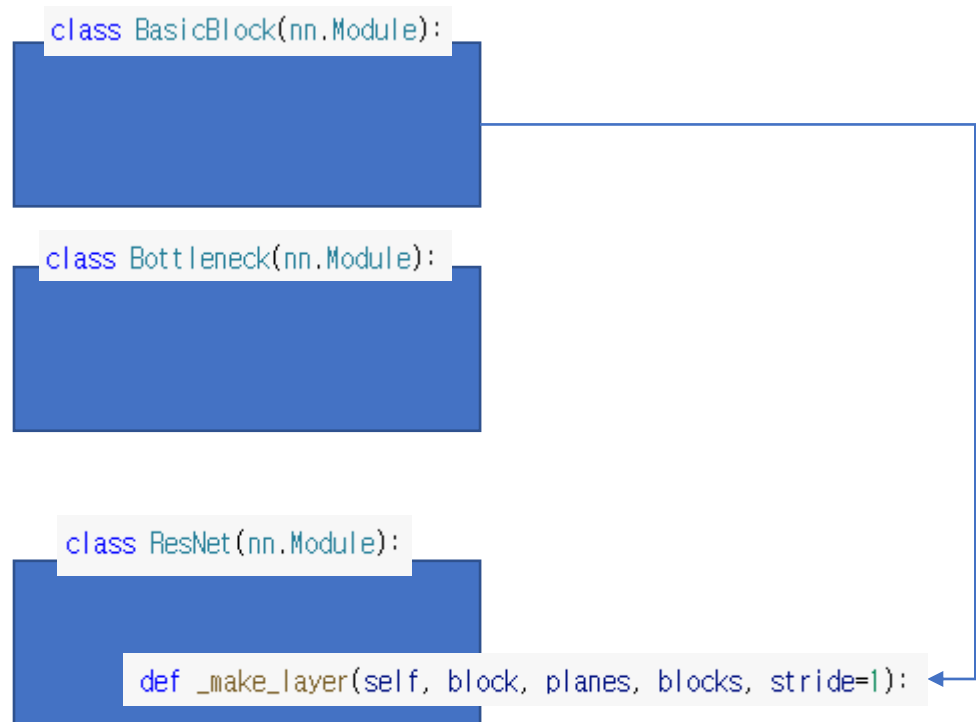
3.4. Implementation

Our implementation for ImageNet follows the practice in [21, 41]. The image is resized with its shorter side randomly sampled in [256, 480] for scale augmentation [41]. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [21]. The standard color augmentation in [21] is used. We adopt batch normalization (BN) [16] right after each convolution and before activation, following [16]. We initialize the weights as in [13] and train all plain/residual nets from scratch. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. We do not use dropout [14], following the practice in [16].

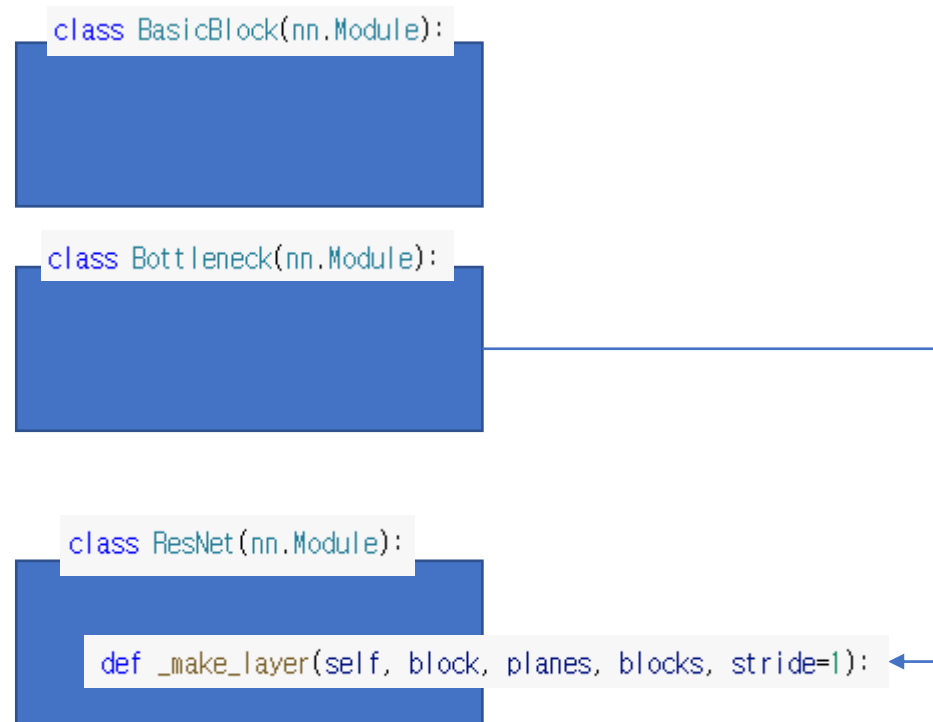
In testing, for comparison studies we adopt the standard 10-crop testing [21]. For best results, we adopt the fully-convolutional form as in [41, 13], and average the scores at multiple scales (images are resized such that the shorter side is in {224, 256, 384, 480, 640}).



◆ 18 Layer



◆ 50 Layer



◆ 18 Layer

```
class ResNet(nn.Module):
```

```
def forward(self, x): #실제 ResNet 모델 구조
```

```
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)
```

```
    x = self.layer1(x)    self.layer1 = self._make_layer(block, 64, layers[0]) #_make_layer
    x = self.layer2(x)    self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
    x = self.layer3(x)    self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
    x = self.layer4(x)    self.layer4 = self._make_layer(block, 512, layers[3], stride=2)
```

```
    x = self.avgpool(x)
    x = x.view(x.size(0), -1) #뷰(View) - 원소
    x = self.fc(x)
```

```
    return x
```

```
class ResNet(nn.Module):
```

```
def _make_layer(self, block, planes, blocks, stride=1):
    downsample = None
    #ResNet18 1번레이어일때
    #파라미터 (stride=1, inplanes = 64, plane=64, block.expansion=1)
    #즉, 아래의 if문 구절에 해당되지 않을 따라서 downsample은 None
    #ResNet18 2번레이어일때는 특징맵을 줄여서 만들어야 하므로
    #downsample이 만들어짐(각 파라미터 => 2, 64, 128, 1)
    if stride != 1 or self.inplanes != planes * block.expansion:
        downsample = nn.Sequential(
            nn.Conv2d(self.inplanes, planes * block.expansion,
                      kernel_size=1, stride=stride, bias=False),
            nn.BatchNorm2d(planes * block.expansion),
        )

    layers = []
    # 빈 레이어 리스트 안에 block 객체 생성해서 append
    layers.append(block(self.inplanes, planes, stride, downsample))

    #ResNet18 1번레이어일때
    #self.inplanes가 64로 변경 (ResNet 50이상은 달라짐)
    self.inplanes = planes * block.expansion
    #block은 모델 구조 layers로 부터 가지고옴. 여기서는 2
    #range(1,2) => i는 1까지
    for i in range(1, blocks):
        layers.append(block(self.inplanes, planes))

    #총 append 되는 개수 -> 2개

    return nn.Sequential(*layers)
```


◆ 18 Layer

```
class ResNet(nn.Module):
```

```
def forward(self, x): #실제 f
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    x = x.view(x.size(0), -1)
    x = self.fc(x)

    return x
```

```
class ResNet(nn.Module):
```

```
def _make_layer(self, block, planes, blocks, stride=1):
    downsample = None
    #ResNet18 1번레이어일때
    #파라미터 (stride=1, inplanes = 64, plane=64, block.expansion=1)
    #즉, 아래의 if문 구절에 해당되지 않을 따라서 downsample은 None
    #ResNet18 2번레이어일때는 특징맵을 줄여서 만들어야 하므로
    #downsample이 만들어짐(각 파라미터 => 2, 64, 128, 1)
    if stride != 1 or self.inplanes != planes * block.expansion:
        downsample = nn.Sequential(
            nn.Conv2d(self.inplanes, planes * block.expansion,
                      kernel_size=1, stride=stride, bias=False),
            nn.BatchNorm2d(planes * block.expansion),
        )

    layers = []
    # 빈 레이어 리스트 안에 block 객체 생성해서 append
    layers.append(block(self.inplanes, planes, stride, downsample))

    #ResNet18 1번레이어일때
    #self.inplanes가 64로 변경 (ResNet 50이상은 달라짐)
    self.inplanes = planes * block.expansion
    #block은 모델 구조 layers로 부터 가지고옴. 여기서는 2
    #range(1,2) => i는 1까지
    for i in range(1, blocks):
        layers.append(block(self.inplanes, planes))

    #총 append 되는 개수 -> 2개

    return nn.Sequential(*layers)
```

```
self.layer1 = self._make_layer(block, 64, layers[0]) #_make_layer
self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
self.layer4 = self._make_layer(block, 512, layers[3], stride=2)
```

```
class BasicBlock(nn.Module):
```

```
def forward(self, x):
    residual = x

    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)

    out = self.conv2(out)
    out = self.bn2(out)

    if self.downsample is not None:
        residual = self.downsample(x) #앞선

    out += residual
    out = self.relu(out)

    return out
```

◆ 18 Layer

```
class ResNet(nn.Module):
```

```
def forward(self, x): #실제 f
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    x = x.view(x.size(0), -1)
    x = self.fc(x)

    return x
```

```
self.layer1 = self._make_layer(block, 64, layers[0]) #_make_layer
self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
self.layer4 = self._make_layer(block, 512, layers[3], stride=2)
```

```
class ResNet(nn.Module):
```

```
def _make_layer(self, block, planes, blocks, stride=1):
    downsample = None
    #ResNet18 1번레이어일때
    #파라미터 (stride=1, inplanes = 64, plane=64, block.expansion=1)
    #즉, 아래의 if문 구절에 해당되지 않을 따라서 downsample은 None
    #ResNet18 2번레이어일때는 특징맵을 줄여서 만들어야 하므로
    #downsample이 만들어짐(각 파라미터 => 2, 64, 128, 1)
    if stride != 1 or self.inplanes != planes * block.expansion:
        downsample = nn.Sequential(
            nn.Conv2d(self.inplanes, planes * block.expansion,
                      kernel_size=1, stride=stride, bias=False),
            nn.BatchNorm2d(planes * block.expansion),
        )

    layers = []
    # 빈 레이어 리스트 안에 block 객체 생성해서 append
    layers.append(block(self.inplanes, planes, stride, downsample))

    #ResNet18 1번레이어일때
    #self.inplanes가 64로 변경 (ResNet 50이상은 달라짐)
    self.inplanes = planes * block.expansion
    #block은 모델 구조 layers로 부터 가지고옴. 여기서는 2
    #range(1,2) => i는 1까지
    for i in range(1, blocks):
        layers.append(block(self.inplanes, planes))

    #총 append 되는 개수 -> 2개

    return nn.Sequential(*layers)
```

```
class Bottleneck(nn.Module):
```

```
def forward(self, x):
    residual = x

    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)

    out = self.conv2(out)
    out = self.bn2(out)
    out = self.relu(out)

    out = self.conv3(out)
    out = self.bn3(out)

    if self.downsample is not None:
        residual = self.downsample(x) #앞스

    out += residual
    out = self.relu(out)

    return out
```


◆ 모델 구조

기본구조 (ImageNet-ILSVRC) (224*224*3)

layer name	output size	18-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
	1×1	average pool, 1000-d fc, softmax
FLOPs		1.8×10^9

7×7, 64, stride 2
3×3 max pool, stride 2

$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$

average pool, 1000-d fc, softmax

```
from torchsummary import summary
modelSummary = model.cuda()
summary(model, (3, 224, 224))
```

Layer (type)

Output Shape

Param #

Conv2d-1

[-1, 64, 112, 112]

9,408

BatchNorm2d-2

[-1, 64, 112, 112]

128

ReLU-3

[-1, 64, 112, 112]

0

MaxPool2d-4

[-1, 64, 56, 56]

0

◆ 모델 구조

기본구조 (ImageNet-ILSVRC) (224*224*3)

layer name	output size	18-layer			
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2		
		3×3 max pool, stride 2	3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$	Conv2d-5 BatchNorm2d-6 ReLU-7	$\begin{bmatrix} -1, 64, 56, 56 \\ -1, 64, 56, 56 \\ -1, 64, 56, 56 \end{bmatrix}$ 36,864 128 0
			$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$	Conv2d-8 BatchNorm2d-9 ReLU-10	$\begin{bmatrix} -1, 64, 56, 56 \\ -1, 64, 56, 56 \\ -1, 64, 56, 56 \end{bmatrix}$ 36,864 128 0
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$	BasicBlock-11	$\begin{bmatrix} -1, 64, 56, 56 \\ -1, 64, 56, 56 \end{bmatrix}$ 0
			$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$	Conv2d-12 BatchNorm2d-13 ReLU-14	$\begin{bmatrix} -1, 64, 56, 56 \\ -1, 64, 56, 56 \\ -1, 64, 56, 56 \end{bmatrix}$ 36,864 128 0
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$	Conv2d-15 BatchNorm2d-16 ReLU-17	$\begin{bmatrix} -1, 64, 56, 56 \\ -1, 64, 56, 56 \\ -1, 64, 56, 56 \end{bmatrix}$ 36,864 128 0
			$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$	BasicBlock-18	$\begin{bmatrix} -1, 64, 56, 56 \\ -1, 64, 56, 56 \end{bmatrix}$ 0
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$		
			$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$		
	1×1	average pool, 1000-d fc, softmax	average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9			

◆ 모델 구조

기본구조 (ImageNet-ILSVRC) (224*224*3)

layer name	output size	18-layer			
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2		
conv2_x	56×56	3×3 max pool, stride 2			
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$	Conv2d-19	[-1, 128, 28, 28] 73,728
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$	BatchNorm2d-20	[-1, 128, 28, 28] 256
			$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$	ReLU-21	[-1, 128, 28, 28] 0
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$	Conv2d-22	[-1, 128, 28, 28] 147,456
			$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$	BatchNorm2d-23	[-1, 128, 28, 28] 256
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$	Conv2d-24	[-1, 128, 28, 28] 8,192
			$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$	BatchNorm2d-25	[-1, 128, 28, 28] 256
			$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$	ReLU-26	[-1, 128, 28, 28] 0
				BasicBlock-27	[-1, 128, 28, 28] 0
				Conv2d-28	[-1, 128, 28, 28] 147,456
				BatchNorm2d-29	[-1, 128, 28, 28] 256
				ReLU-30	[-1, 128, 28, 28] 0
				Conv2d-31	[-1, 128, 28, 28] 147,456
				BatchNorm2d-32	[-1, 128, 28, 28] 256
				ReLU-33	[-1, 128, 28, 28] 0
				BasicBlock-34	[-1, 128, 28, 28] 0
	1×1	average pool, 1000-d fc, softmax	average pool, 1000-d fc, softmax		
FLOPs		1.8×10 ⁹			

이전 56*56 레이어 붙이기 위해
사이즈 줄여주는 계층

◆ 모델 구조

기본구조 (ImageNet-ILSVRC) (224*224*3)

layer name	output size	18-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
		$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
		$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
		$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$
	1×1	average pool, 1000-d fc, softmax
FLOPs		1.8×10^9

7×7, 64, stride 2
3×3 max pool, stride 2

$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$
 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$
 $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$
 $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$
 $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$

average pool, 1000-d fc, softmax

Conv2d-35	[-1, 256, 14, 14]	294,912
BatchNorm2d-36	[-1, 256, 14, 14]	512
ReLU-37	[-1, 256, 14, 14]	0
Conv2d-38	[-1, 256, 14, 14]	589,824
BatchNorm2d-39	[-1, 256, 14, 14]	512
Conv2d-40	[-1, 256, 14, 14]	32,768
BatchNorm2d-41	[-1, 256, 14, 14]	512
ReLU-42	[-1, 256, 14, 14]	0
BasicBlock-43	[-1, 256, 14, 14]	0
Conv2d-44	[-1, 256, 14, 14]	589,824
BatchNorm2d-45	[-1, 256, 14, 14]	512
ReLU-46	[-1, 256, 14, 14]	0
Conv2d-47	[-1, 256, 14, 14]	589,824
BatchNorm2d-48	[-1, 256, 14, 14]	512
ReLU-49	[-1, 256, 14, 14]	0
BasicBlock-50	[-1, 256, 14, 14]	0

◆ 모델 구조

기본구조 (ImageNet-ILSVRC) (224*224*3)

layer name	output size	18-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
		$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
		$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
		$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$
	1×1	average pool, 1000-d fc, softmax
FLOPs		1.8×10^9

7×7, 64, stride 2
3×3 max pool, stride 2

$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$

$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$

Conv2d-51	[-1, 512, 7, 7]	1,179,648
BatchNorm2d-52	[-1, 512, 7, 7]	1,024
ReLU-53	[-1, 512, 7, 7]	0
Conv2d-54	[-1, 512, 7, 7]	2,359,296
BatchNorm2d-55	[-1, 512, 7, 7]	1,024
Conv2d-56	[-1, 512, 7, 7]	131,072
BatchNorm2d-57	[-1, 512, 7, 7]	1,024
ReLU-58	[-1, 512, 7, 7]	0
BasicBlock-59	[-1, 512, 7, 7]	0
Conv2d-60	[-1, 512, 7, 7]	2,359,296
BatchNorm2d-61	[-1, 512, 7, 7]	1,024
ReLU-62	[-1, 512, 7, 7]	0
Conv2d-63	[-1, 512, 7, 7]	2,359,296
BatchNorm2d-64	[-1, 512, 7, 7]	1,024
ReLU-65	[-1, 512, 7, 7]	0
BasicBlock-66	[-1, 512, 7, 7]	0

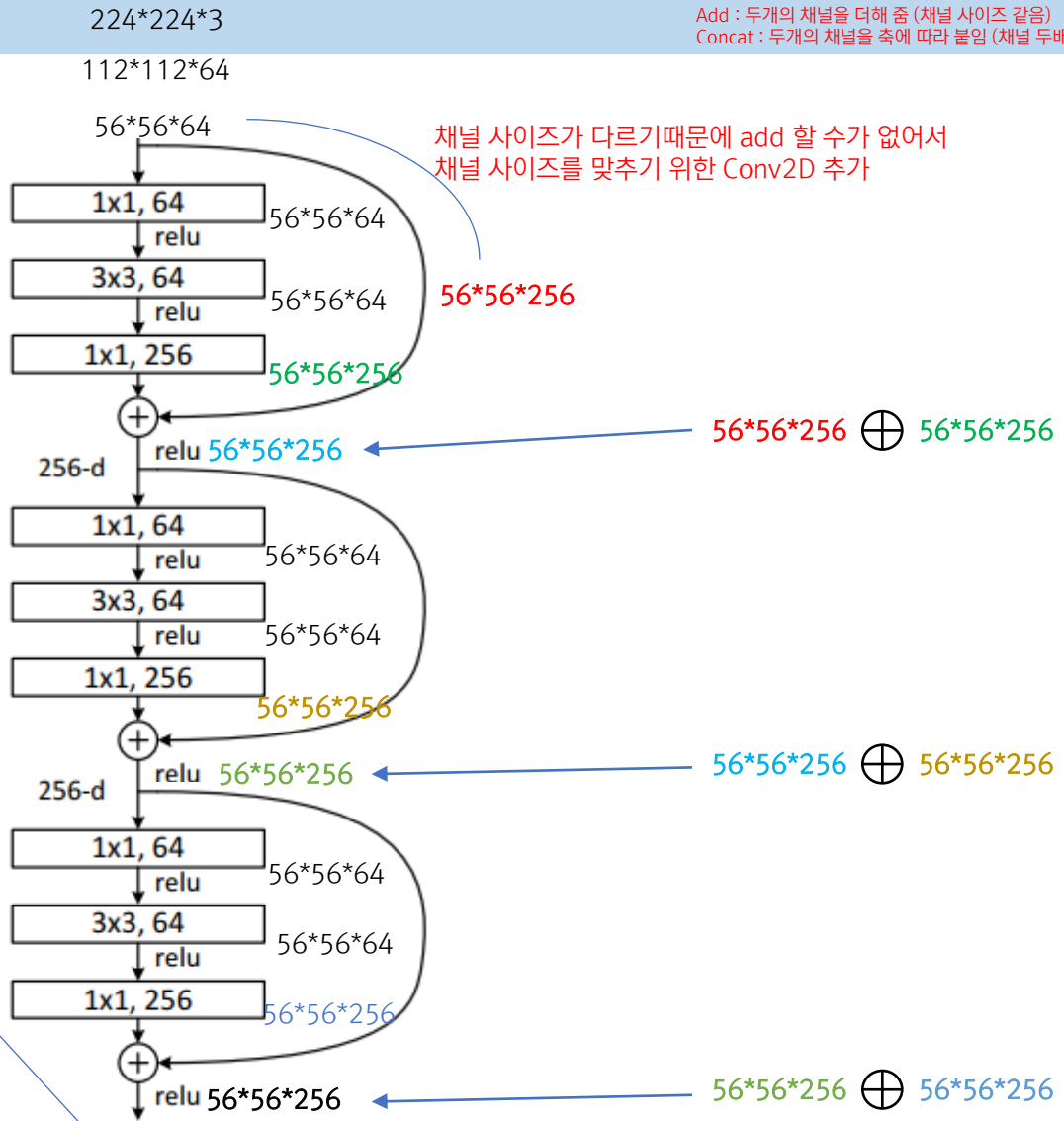
◆ 모델 구조

152-layer	
$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	}
$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$	
$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$	
$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	
11.3 × 10 ⁹	

50, 101, 152 layer에서만
Bottleneck 사용 (차원을 줄였다가 늘림-병목)
연산시간을 줄이는 것에 대한 목적

Bottleneck?

$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$
$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$

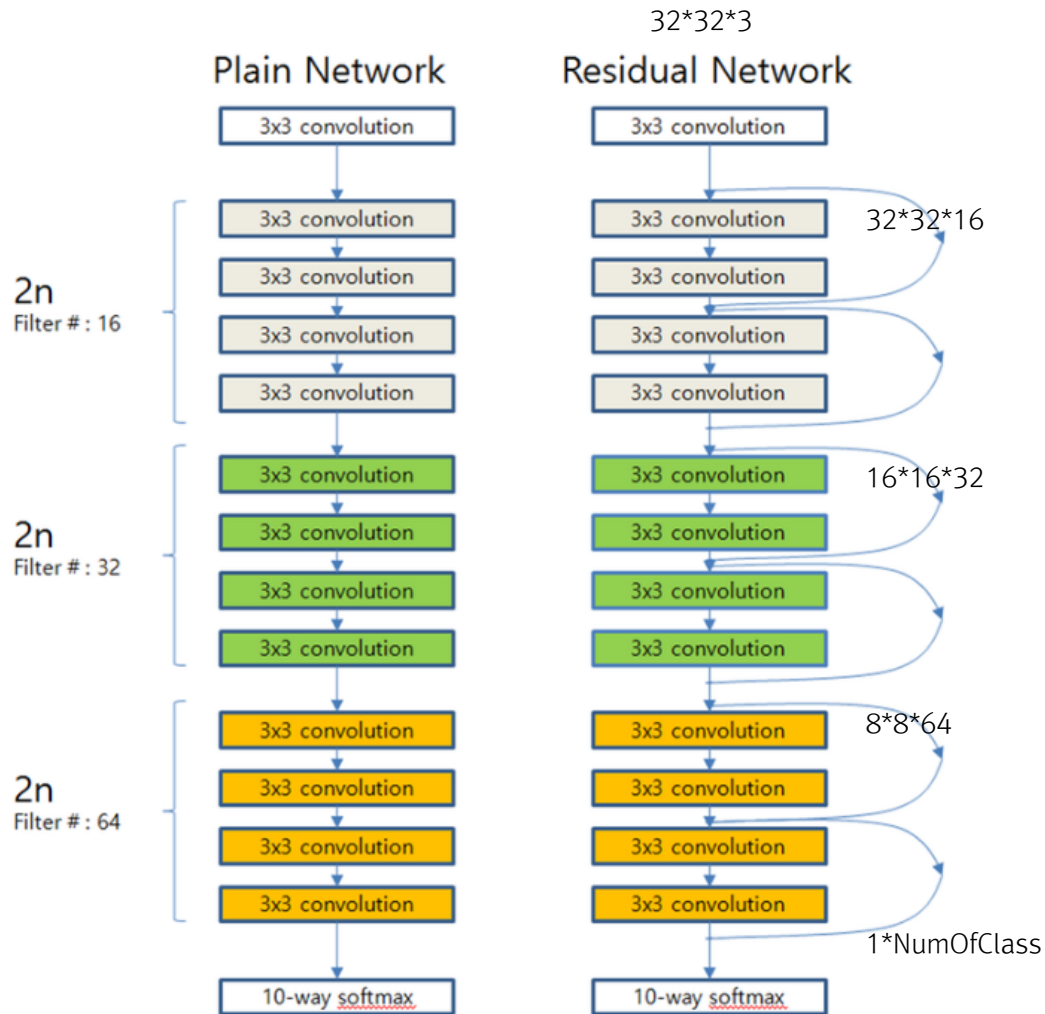


◆ 모델 구조

Cifar10 ($32 \times 32 \times 3$)을 위한 모델

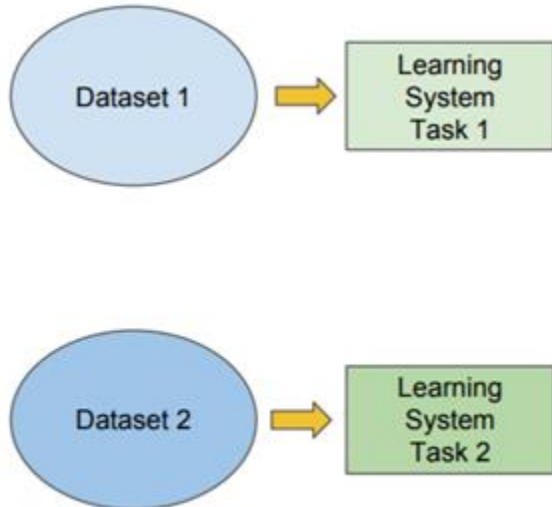
출처 :

<https://blog.naver.com/PostView.nhn?blogId=laonple&logNo=220770760126>



Traditional ML

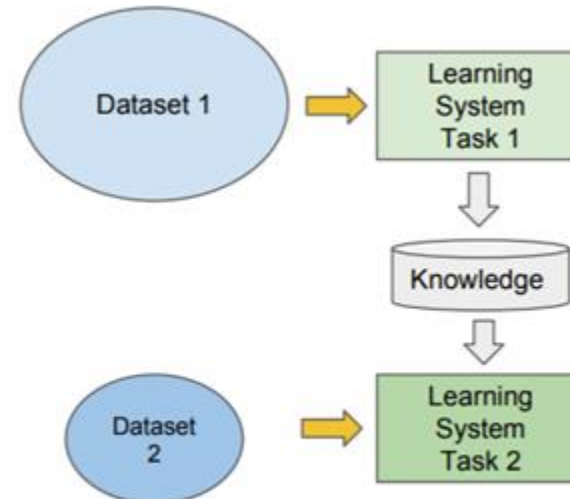
- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



vs

Transfer Learning

- Learning of a new task relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data



두 가지 타입의 Transfer learning

1) Fine-tuning

pre-trained된 모델로 시작하여 새로운 task에 대한 model의 모든 parameter를 업데이트

본질적으로 전체 model을 retraining

2) Feature extraction

pre-trained된 모델로 시작하여 prediction을 도출하는 마지막 레이어의 weight만 업데이트

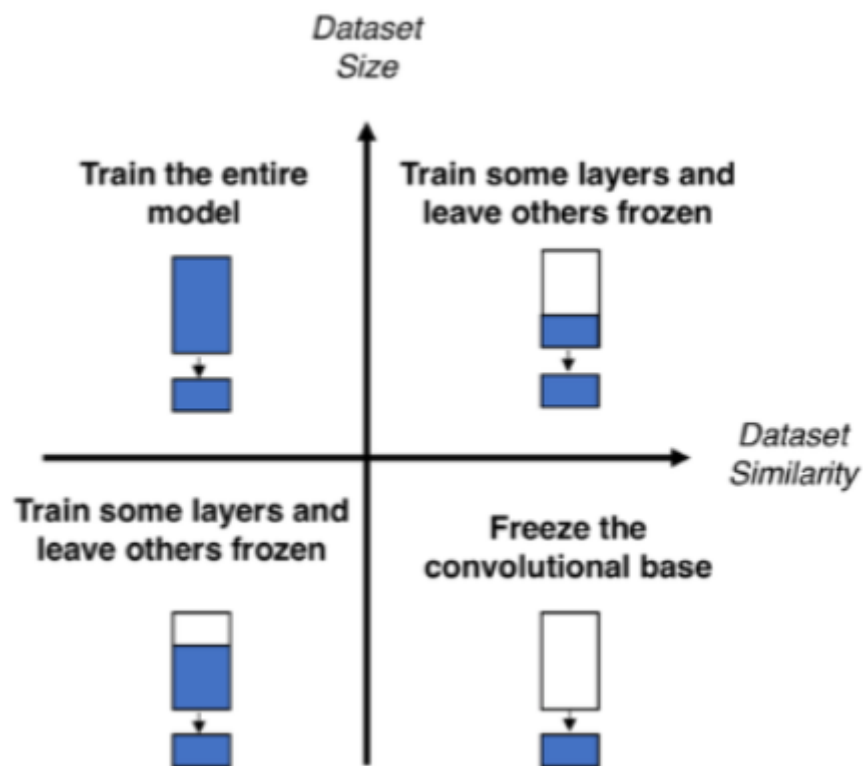
feature extraction이라고 불리는 것은 pretrained CNN을 고정된 feature-extractor로 이용하고,

오직 마지막 레이어만 바꾸기 때문

The screenshot shows the PyTorch documentation page for `torchvision.models`. The page title is `TORCHVISION.MODELS`. The main content area describes the models subpackage and lists various model architectures for image classification. A callout box on the right side of the page lists the following model architectures:

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNetV2
- MobileNetV3
- ResNeXt
- Wide ResNet
- MNASNet
- EfficientNet
- RegNet

<https://pytorch.org/vision/stable/models.html>



1. Large & 다른 Dataset

Overfitting이 발생하지 않을 정도로 큰 데이터셋과 pre-trained model의 학습에 사용된 dataset과 많이 다른(유사하지 않은) dataset을 사용하는 경우입니다.

이 경우, 기존 학습에 사용된 learning-rate(이후, lr)의 1/10으로 lr을 적용하여 Conv. layer와 FC layer에 대해 학습을 진행합니다.

lr을 너무 크게 조정하면 모델이 완전히 새로 학습되어 fine-tuning을 사용하는 의미가 없어지기 때문입니다.

2. Large & 유사 Dataset

Conv. layer는 input에 가까운 계층일수록 선과 색과 같은 일반적인 특징을 학습합니다.

output과 가까운 계층일수록 class 분류와 관련된 세세한 특징들을 학습합니다.

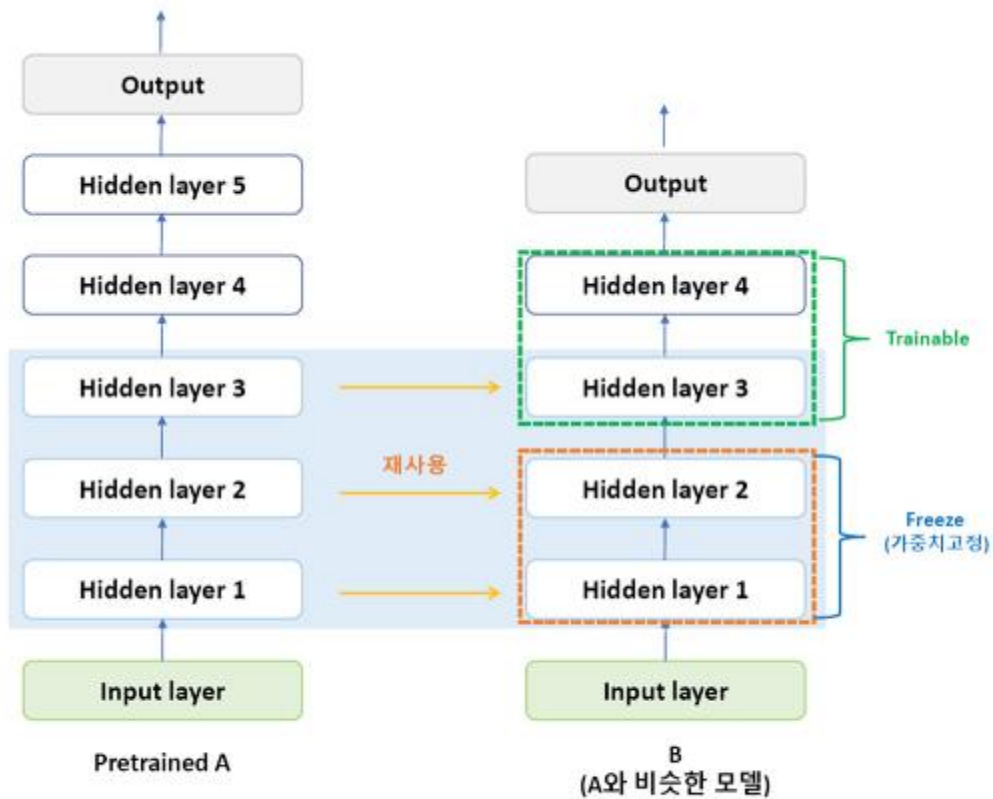
유사한 dataset을 사용한다면 시간절약을 위해 Conv. layer의 후반 계층과 FC layer만 학습을 진행해줍니다. 학습을 하지 않는 부분은 lr=0, 학습을 하는 부분은 lr = 기존의 1/10 으로 지정해줍니다.

3. Small & 다른 Dataset

가장 Challenge한 경우입니다. 데이터가 적기때문에 overfitting의 위험성이 있어 Conv. layer를 전부 학습시키는건 피해야합니다. Conv. layer에서 어느정도 계층을 학습할지 잘 정해서 학습시켜야 합니다. 역시 FC layer는 모두 학습합니다. 이 경우 data augmentation을 통해 데이터양을 늘려주기도 합니다.

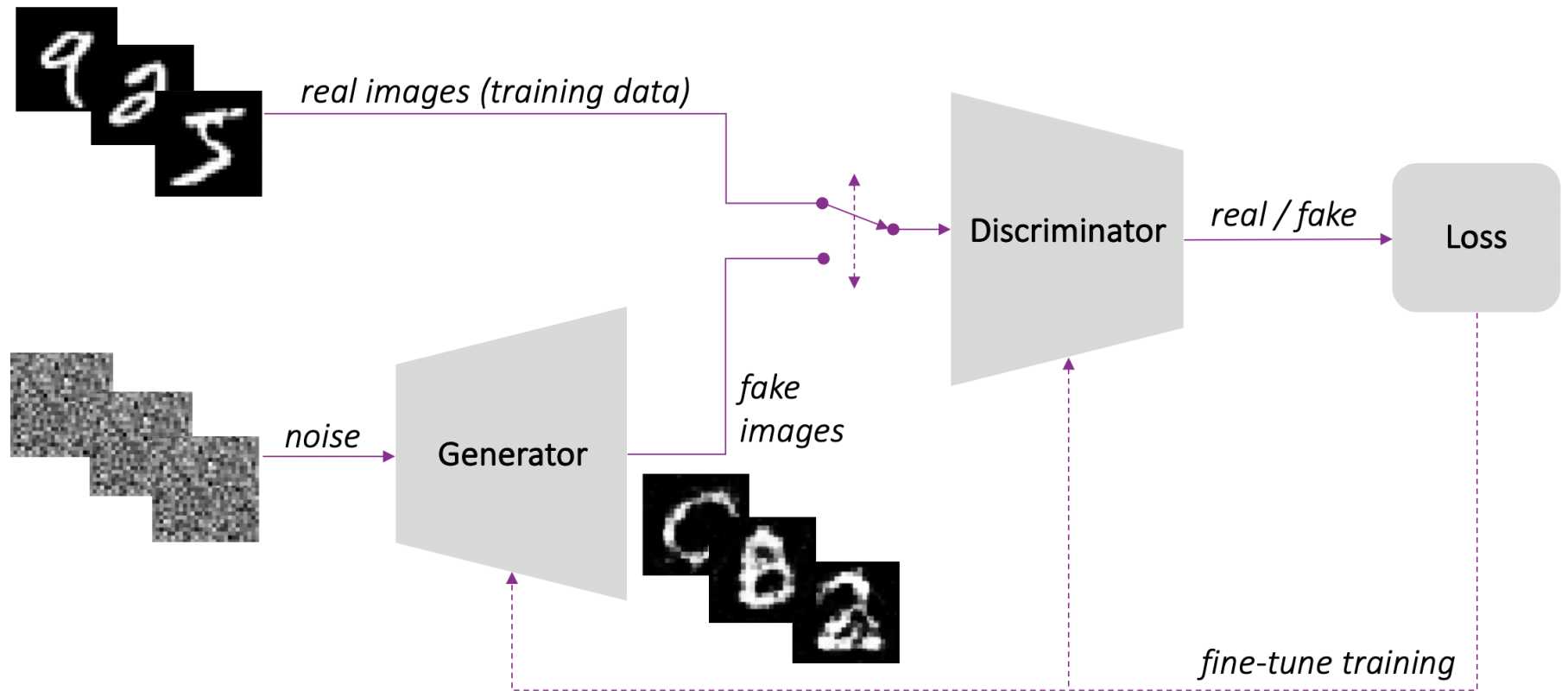
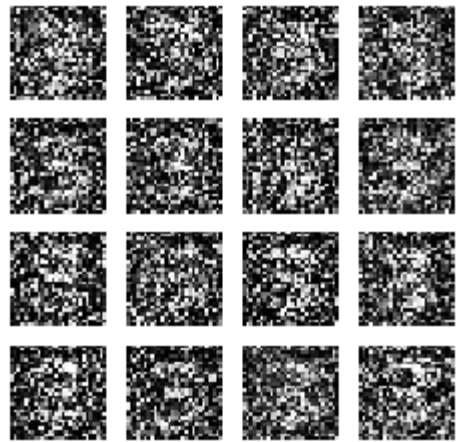
4. Small & 유사 Dataset

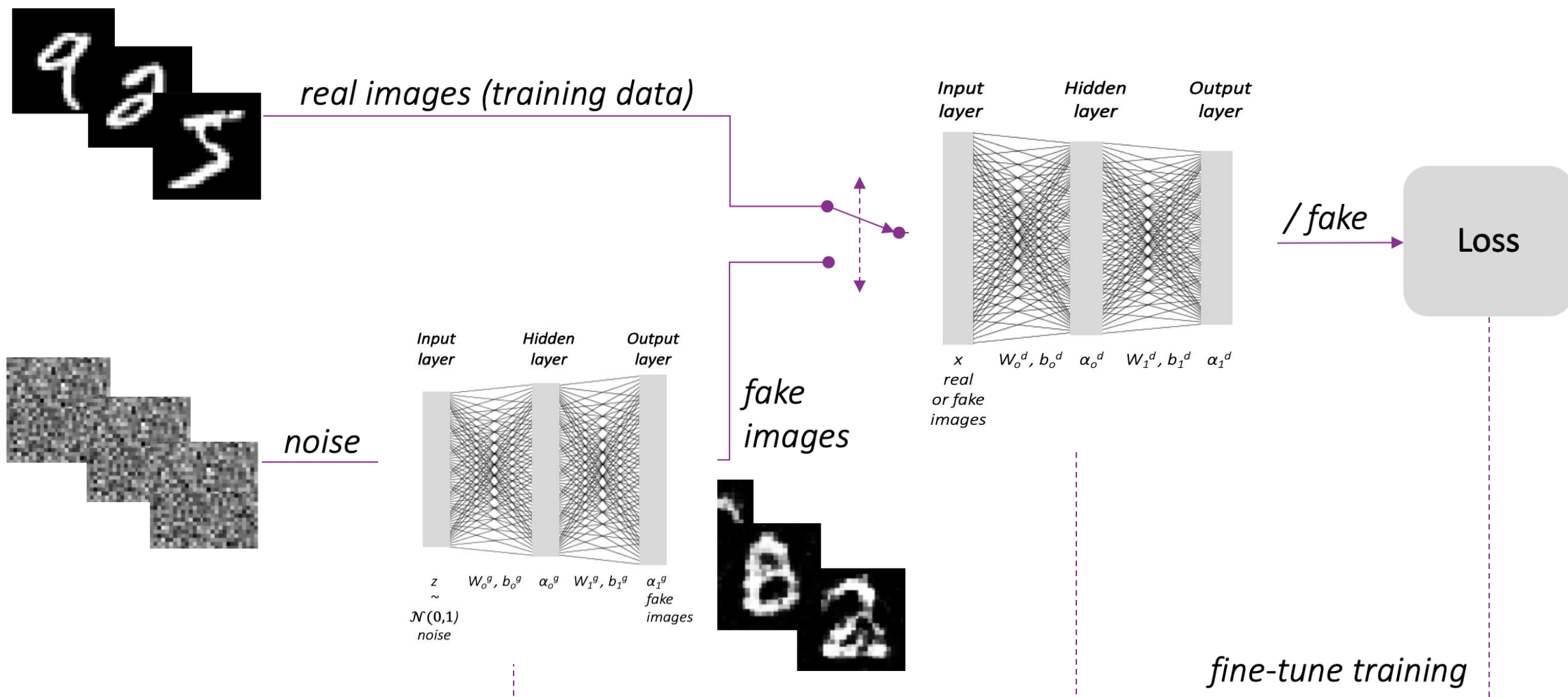
Dataset이 작기때문에 Conv.layer는 학습하지 않고 FC layer만 학습합니다.

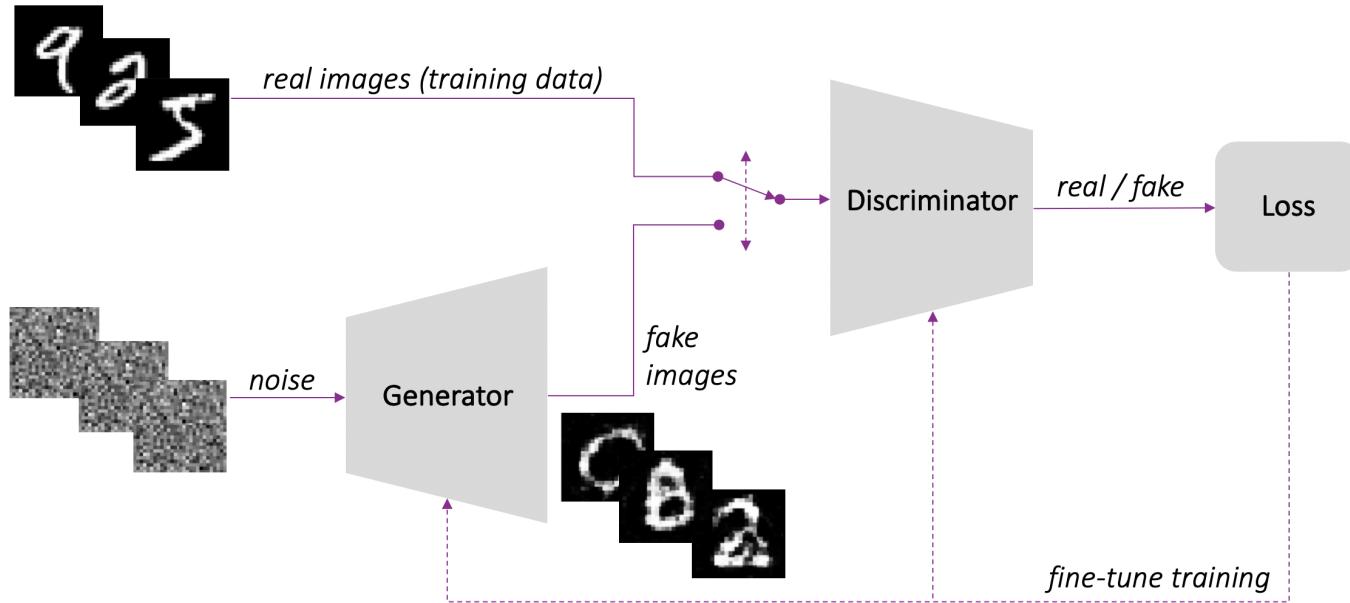


[전이학습의 개념]

```
set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```







- GAN은 Generator(G, 생성모델/생성기)와 Discriminator(D, 판별모델/판별기)라는 서로 다른 2개의 네트워크로 이루어져 있으며 이 두 네트워크를 적대적으로 학습시키며 목적을 달성
- 생성모델(G)의 목적은 진짜 분포에 가까운 가짜분포를 생성하는 것
- 판별모델(D)의 목적은 표본이 가짜분포에 속하는지 진짜분포에 속하는지를 결정
- GAN의 궁극적인 목적은 "실제 데이터의 분포"에 가까운 데이터를 생성하는 것
- 판별모델(D)는 진짜인지 가짜인지를 한 쪽으로 판단하지 못하는 경계(가짜와 진짜를 0과 1로 보았을 때 0.5의 값)를 가져야함

NAVER

팀뷰어



통합검색

웹사이트

블로그

지식iN

이미지

지식백과

뉴스

포스트

더보기

검색음


[TeamViewer - 원격지원, 원격접속, 서비스 데스크, 온....](http://www.teamviewer.com) www.teamviewer.com

[다운로드](#) · [구매](#) · [제품소개](#) · [연락처&의견](#) · [소개](#) · [자격증명](#)

TeamViewer의 원격 데스크탑 접속 솔루션: 원격 컴퓨터에 연결하고 원격 지원 및 온라인 협업을 제공할 수...

연관채널 트위터 페이스북 앱스토어 구글플레이



 TeamViewer

제품 솔루션 리소스 통합 회사정보 파트너

↓ 다운로드

지금 구매하기 >

곧 TeamViewer 무료 다운로드

TeamViewer 무료 버전이 다운로드 폴더에 저장됩니다. Mac 사용자는 [여기](#)를 클릭하십시오.

다운로드가 시작되지 않으면, > 직접 다운로드

다운로드받는 동안

1,000명 규모의 회사에서>일하시나요?

Enterprise용 원격 연결의 세계 기준과 **TeamViewer Tensor™ SaaS** 플랫폼을 통해 Corporate 로그인과 통합된 새로운 수준의 보안을 경험해보세요.

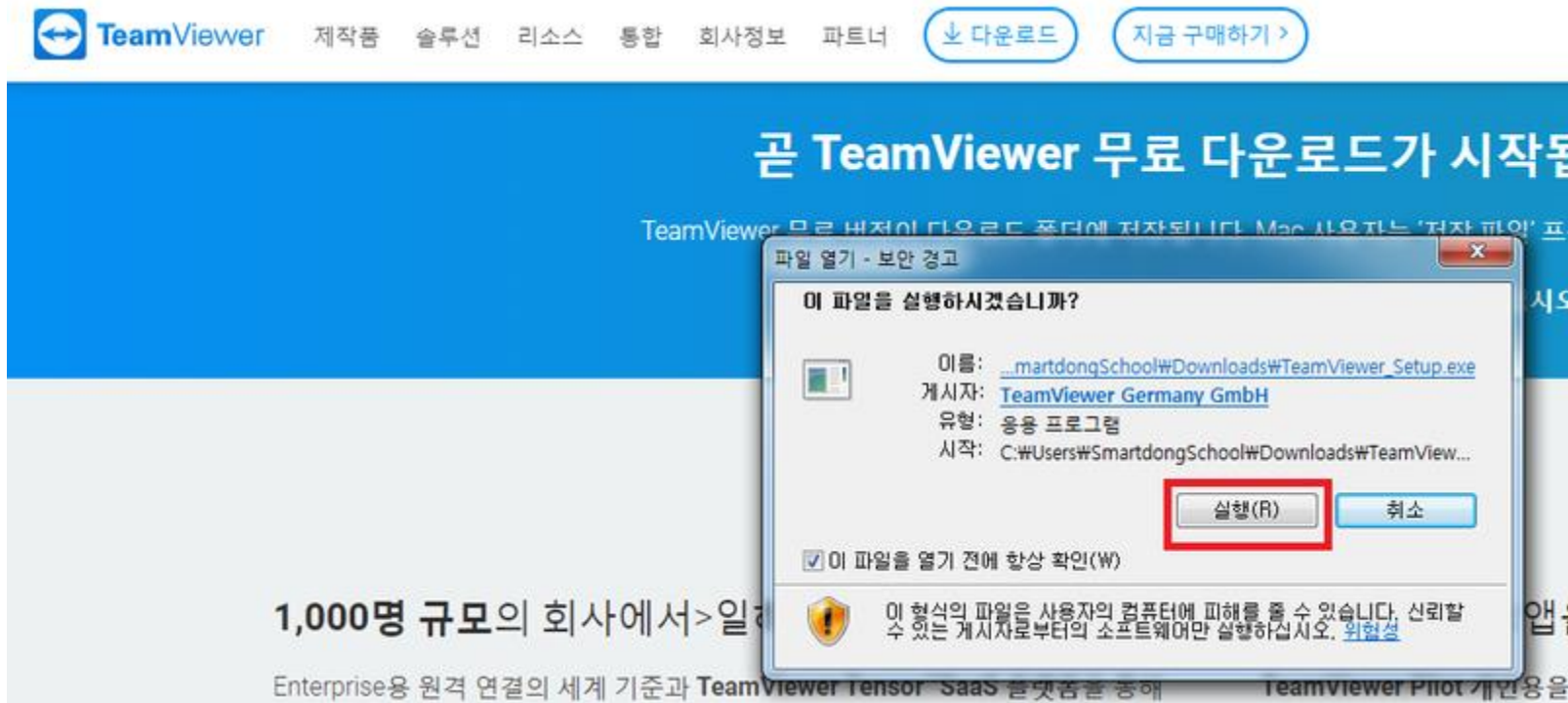
증강 현실

TeamViewer의 증강 현실 기능을 사용하여 원격으로 문제를 해결할 수 있습니다.

Cookie Consent

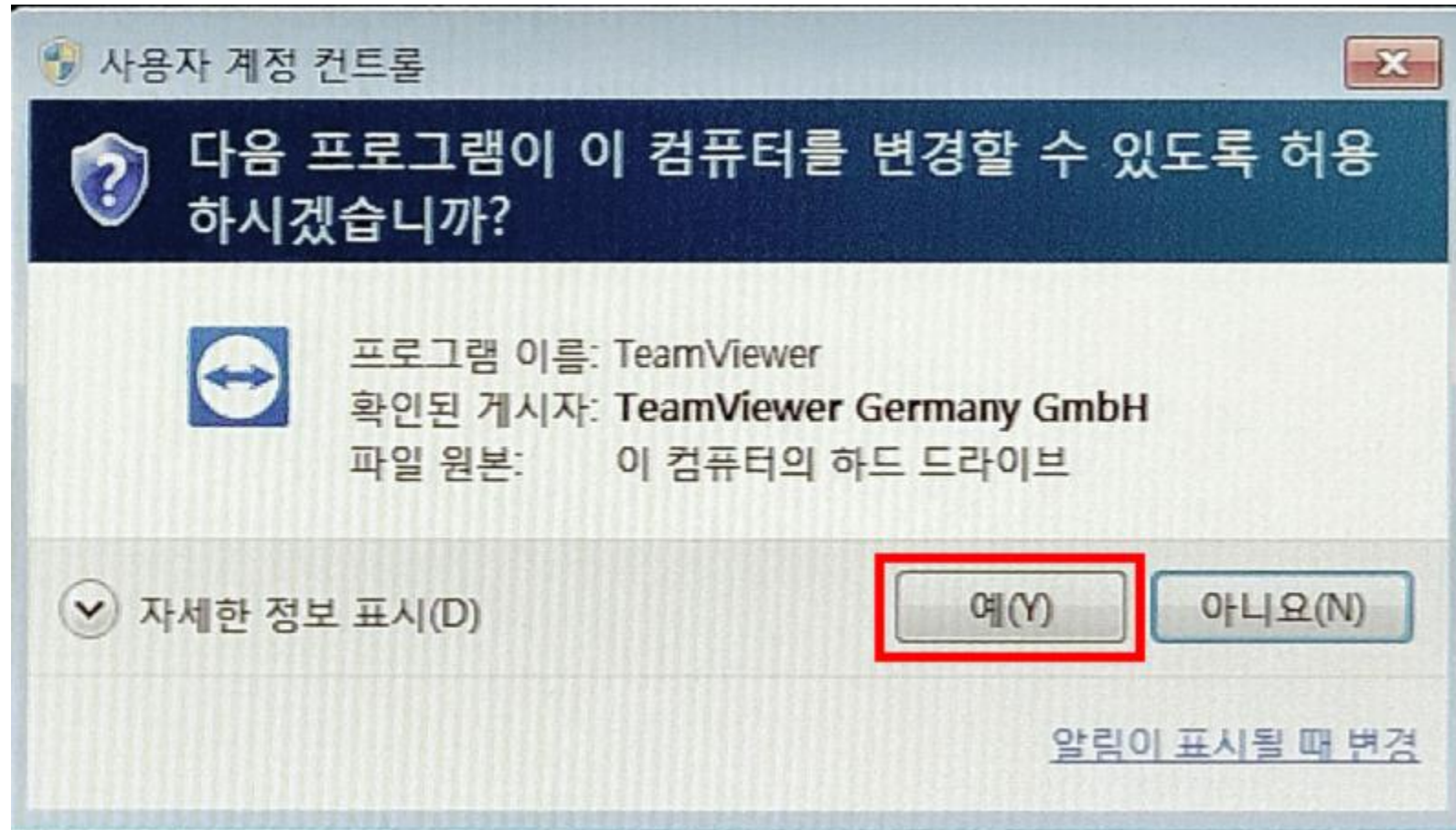
To improve your browsing experience on this website, TeamViewer and its partners would like to place cookies and similar technologies ("Cookies") on your device. This enables us to analyze website usage and optimize our marketing efforts for the best possible user experience. By clicking "Accept Cookies" you agree to Cookies being used for the purposes of website analytics, marketing and respective use, as well as the subsequent processing of the collected data for the purposes of personalized advertising and marketing. Detailed information about the exact purposes, third-party recipients, Cookie lifetime, and more can be found in our [Cookie Policy](#) and [Privacy Policy](#). You can always change your Cookie Settings in the footer of this website.

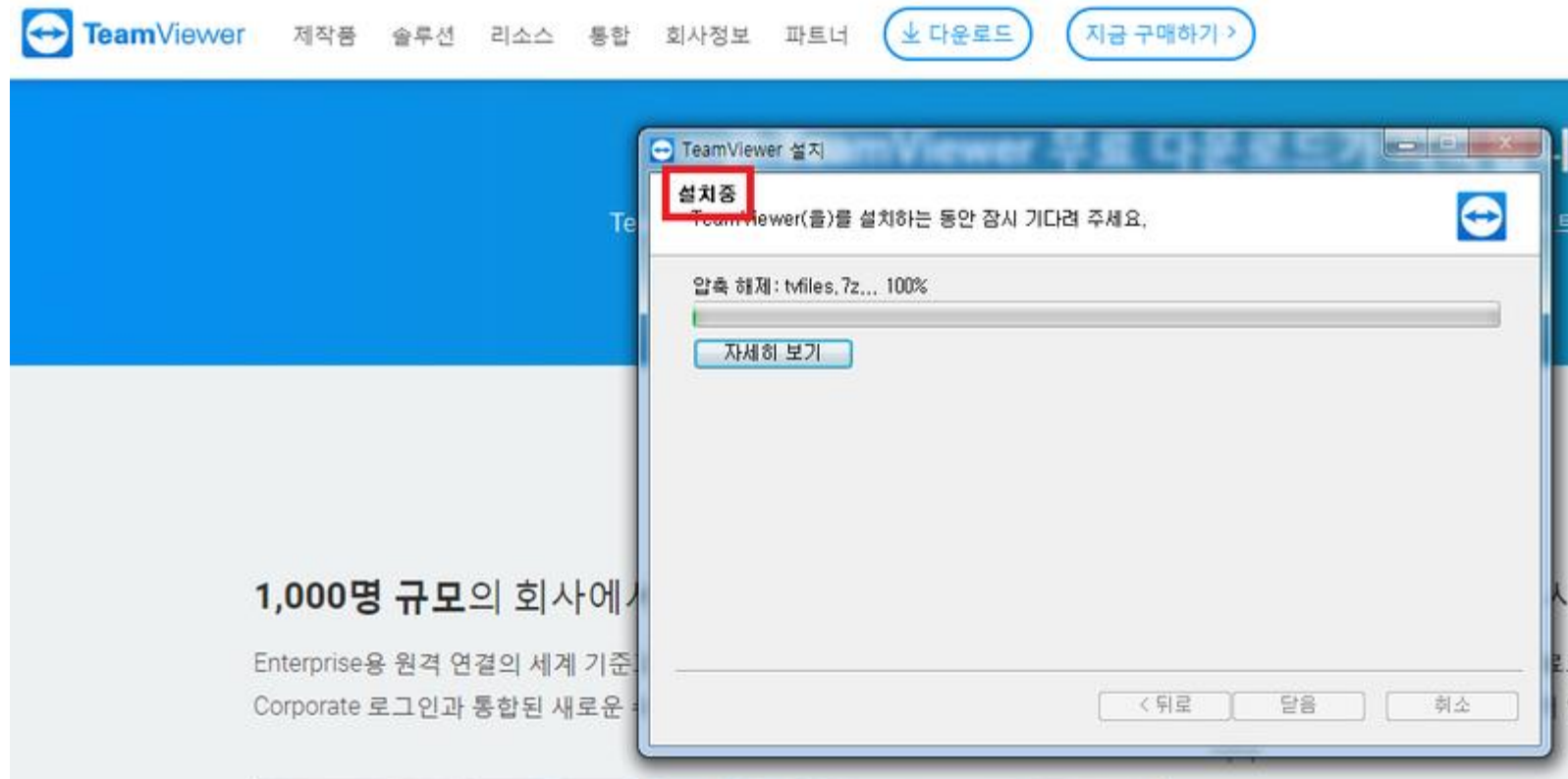
 TeamViewer_Setup.exe ^



The screenshot shows the Korean homepage of the TeamViewer website. At the top, there is a navigation bar with the TeamViewer logo and links for '제품' (Products), '솔루션' (Solutions), '리소스' (Resources), '통합' (Integration), '회사정보' (Company Information), and '파트너' (Partners). Two buttons are visible: '다운로드' (Download) and '지금 구매하기 >' (Buy Now >). The main banner features the text '곧 TeamViewer 무료 다운로드가 시작됨' (TeamViewer free download starts soon). Below this, a Windows security warning dialog box is open, titled '파일 열기 - 보안 경고' (File Open - Security Warning). The dialog asks '이 파일을 실행하시겠습니까?' (Do you want to run this file?). It displays the file name '...martdongSchool\Downloads\TeamViewer_Setup.exe', the publisher 'TeamViewer Germany GmbH', the type '응용 프로그램' (Application), and the location 'C:\Users\SmartdongSchool\Downloads\TeamView...'. There are two buttons at the bottom: '실행(R)' (Run) and '취소' (Cancel). The '실행(R)' button is highlighted with a red rectangle. At the bottom of the dialog, there is a checkbox '이 파일을 열기 전에 항상 확인(W)' (Always check before opening this file) which is checked. Below the checkbox, a warning icon and text state: '이 형식의 파일은 사용자의 컴퓨터에 피해를 줄 수 있습니다. 신뢰할 수 있는 게시자로부터의 소프트웨어만 실행하십시오. 위험성' (This type of file may harm your computer. Only run software from trusted publishers. Risk).





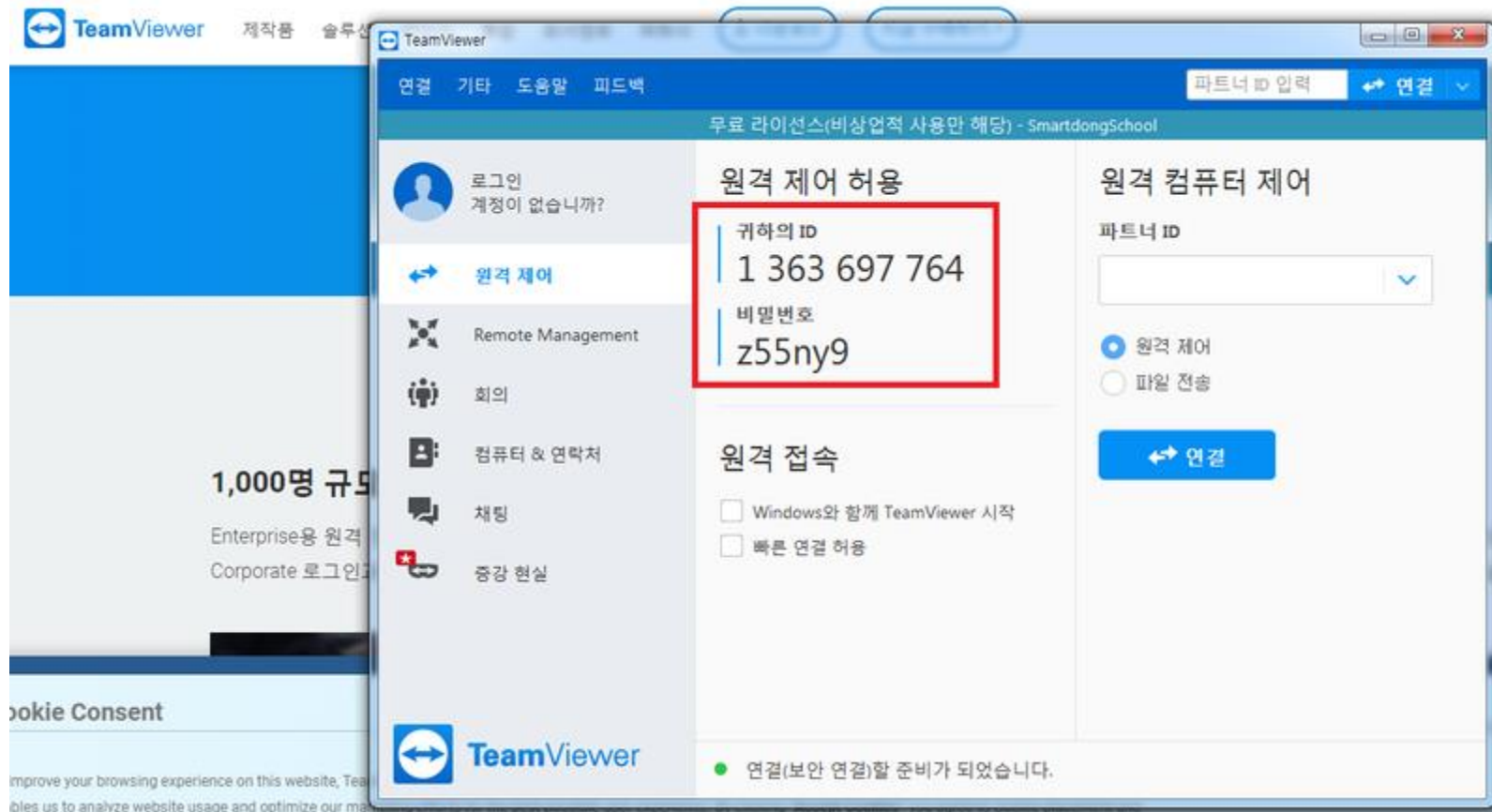




설치가 완료되면 자동으로 프로그램이 실행되거나

컴퓨터 화면 또는 작업 표시줄에 팀뷰어 아이콘이 뜹니다

팀뷰어 아이콘을 클릭



ID와 비밀번호를 메모

1) 원격데스크톱 세팅

<https://mpjamong.tistory.com/123>

2) 포트변경 (이때 포트는 443)

<https://mpjamong.tistory.com/133>

감사합니다

