

# Pytorch를 활용한 딥러닝 학습 환경 구축 및 실습

- 3일차 -

강 수 명

smgang.kmu@gmail.com

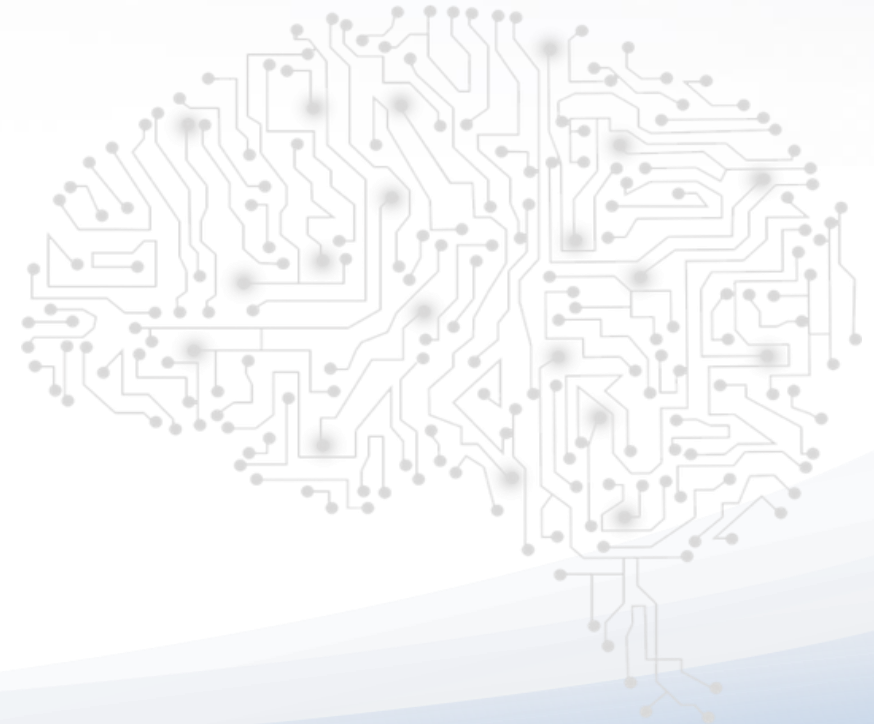
# C

## ONTENTS

I \_CNN 기초

II \_ResNet

III \_Transfer Learning





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/facktpublishing/PyTorch-Computer-Vision-Cookbook
- https://github.com/nineddie/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

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.
```

## #구글 드라이브와 Colab 연동

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

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

```
cd /content/gdrive/MyDrive/
!git clone https://github.com/smgang/pytorch_Study.git
```

[https://github.com/smgang/pytorch\\_Study](https://github.com/smgang/pytorch_Study)

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



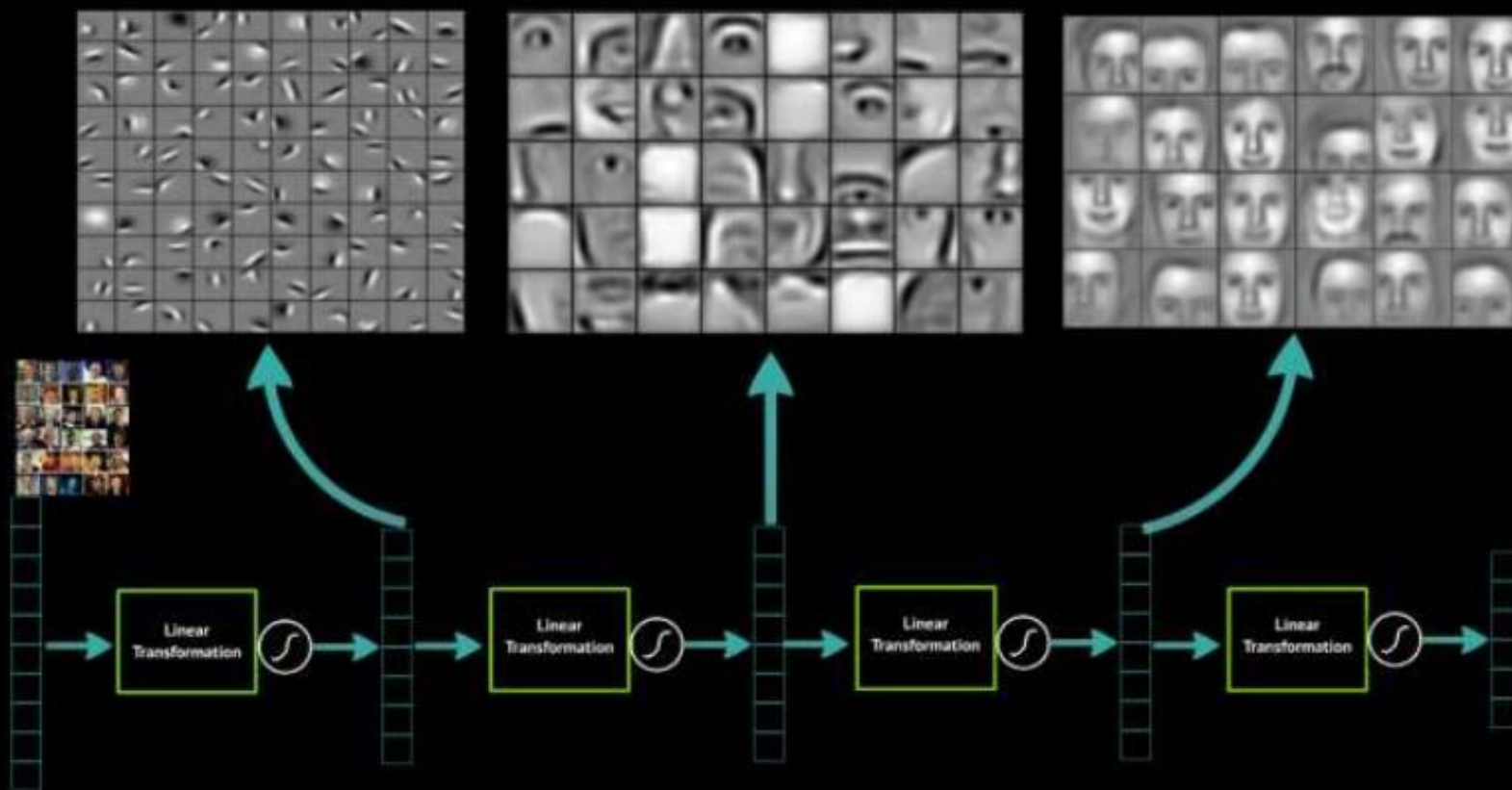
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

<https://www.cs.toronto.edu/~kriz/cifar.html>

<https://github.com/YoongiKim/CIFAR-10-images>



### Deep Learning learns layers of features



컨볼루션이 뭔데요?

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

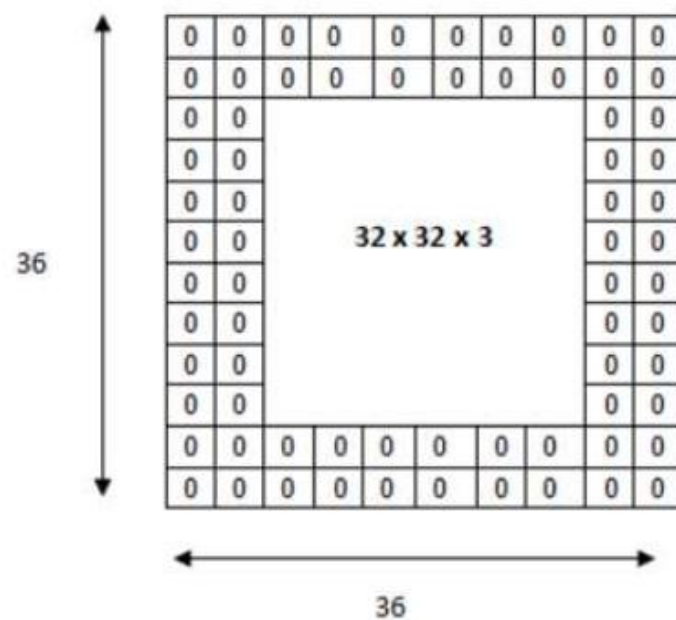
Image

4		

Convolved  
Feature



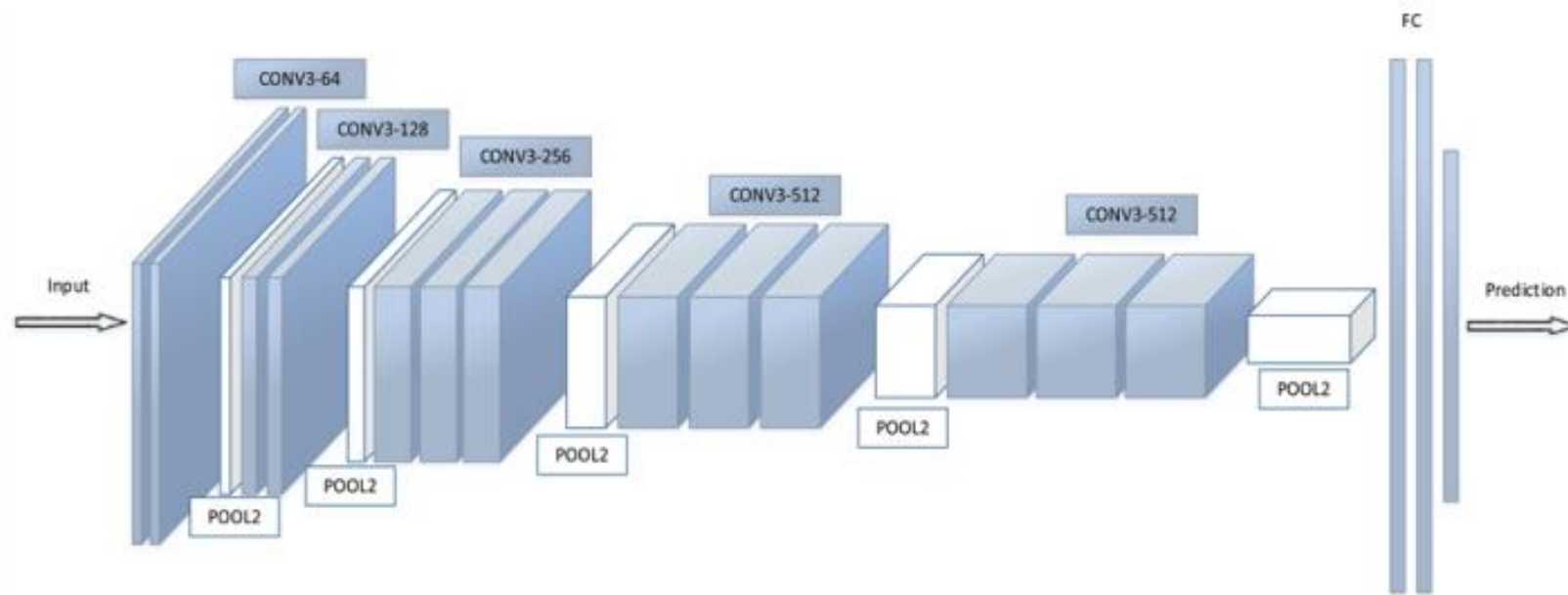
영상이 작아져버렸네.. 왼쪽 끝이랑 오른쪽 끝 픽셀은 안보나요?



The input volume is 32 x 32 x 3. If we imagine two borders of zeros around the volume, this gives us a 36 x 36 x 3 volume. Then, when we apply our conv layer with our three 5 x 5 x 3 filters and a stride of 1, then we will also get a 32 x 32 x 3 output volume.

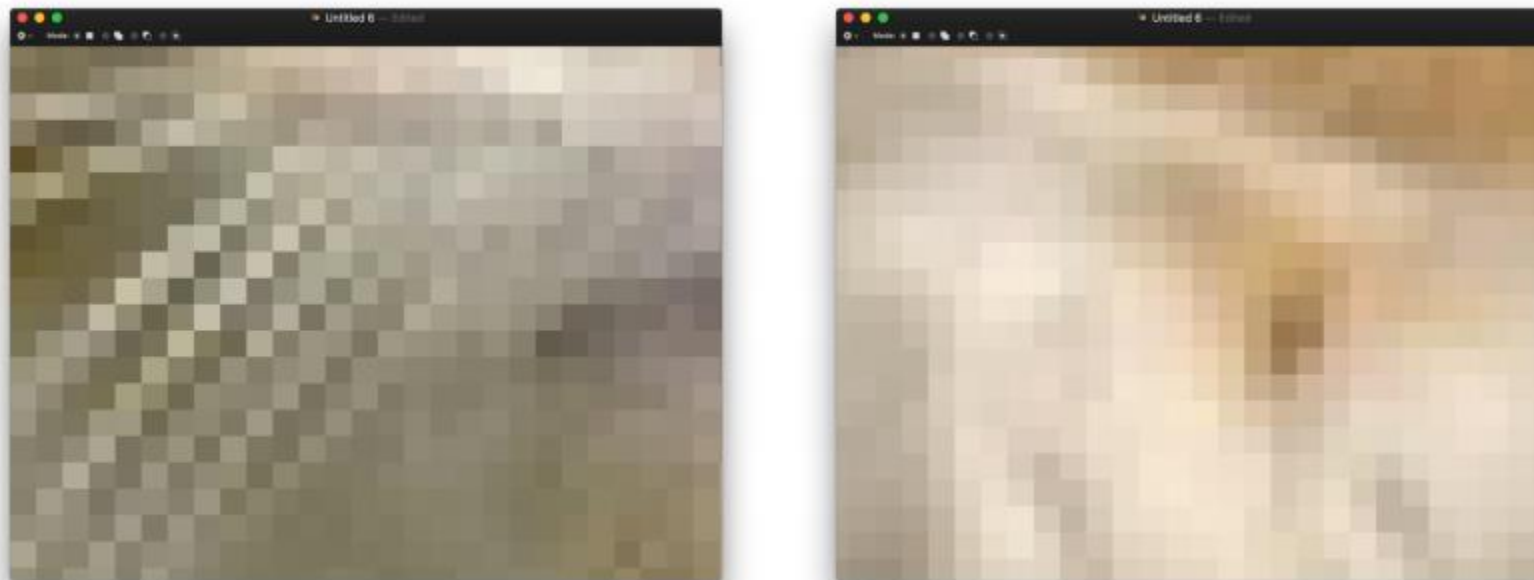
볼수도 있고, 안볼수도 있습니다.

# VGGNET의 구조



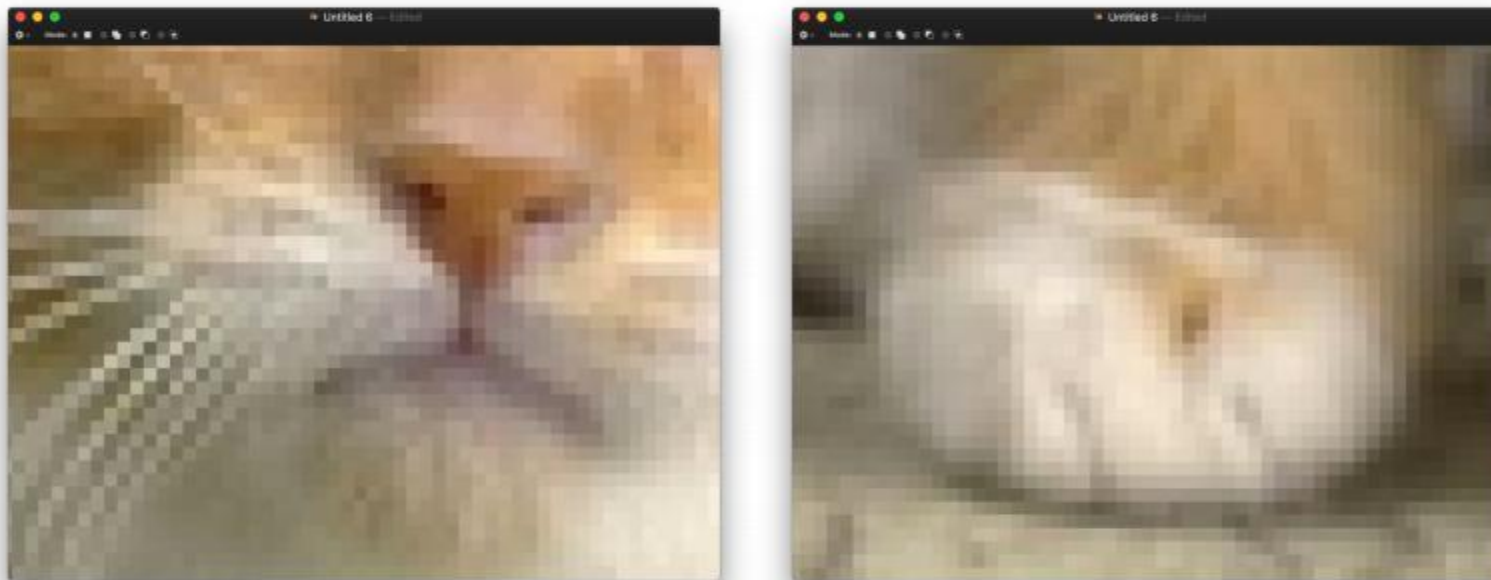
이해를 하기 위해,우리가 그림을 본다고 상상해봅시다.

**그림을 눈앞 1cm거리에서 본다고 생각해보자.**



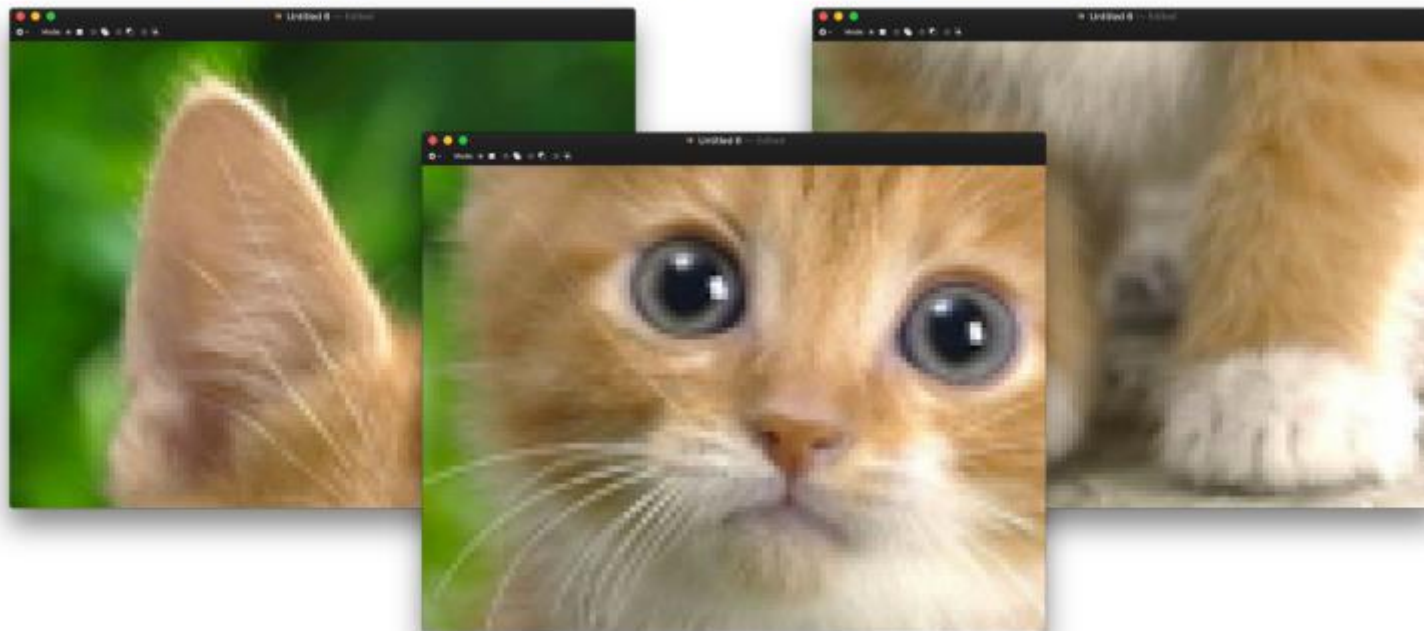
**처음에는 점과 선, 이상한 질감 몇개 밖에 모르겠다.**

**점과 선, 질감을 충분히 배우고, 조금 떨어져서 보자.**



**점과 선이 질감이 합쳐져 삼각형, 동그라미, 북실함이 보인다.**

**삼각형, 원, 사각형, 복실함등을 조합해서 보니**



**뽀족귀와 땡그란눈과 복실한 발을 배웠다.**

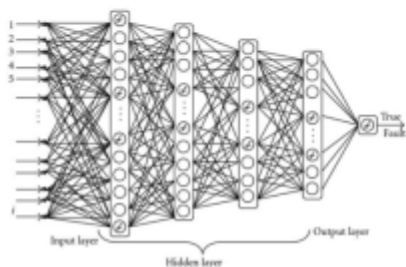


더 멀리서 보니, 그것들이 모아져있다. 이것은?

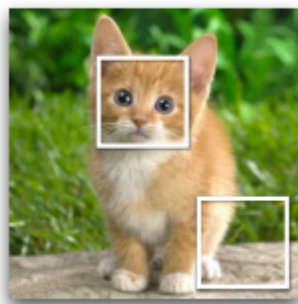


고양이!!

### 계산량 측면에서도!

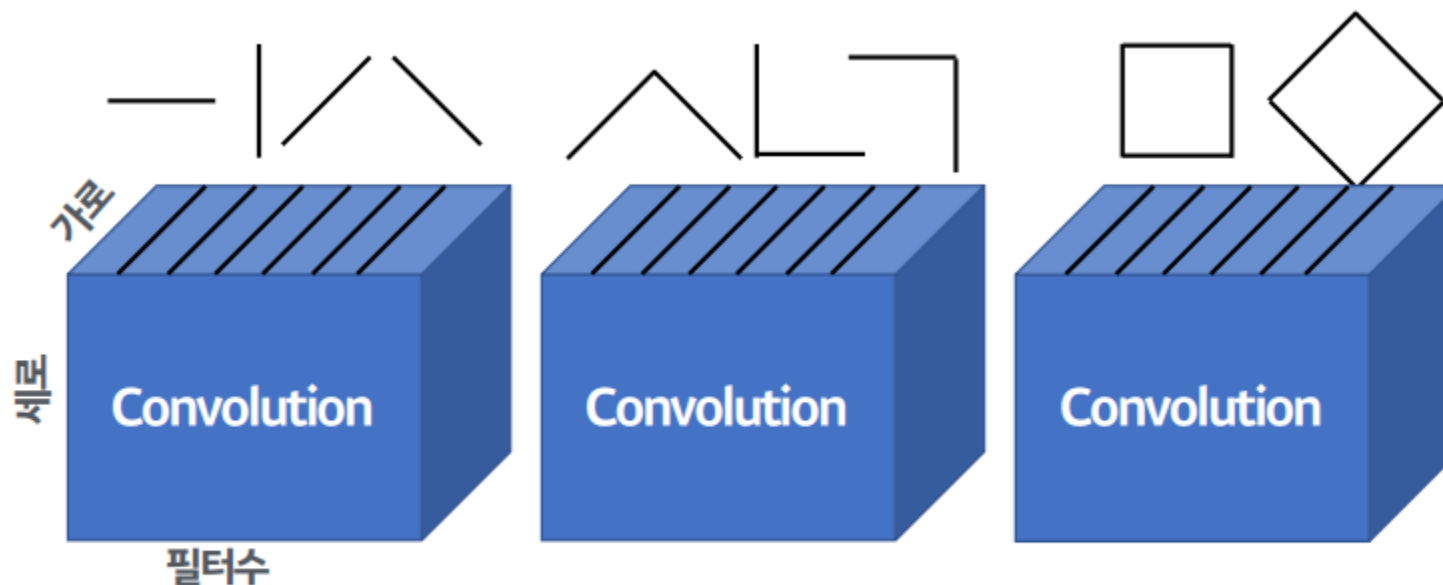


이런생각을해봤다.  
보통 뉴럴넷은 서로가 서로에게 전부다 연결되어 있는데  
이러다보니 맞춰야 할 weight들도 많아..



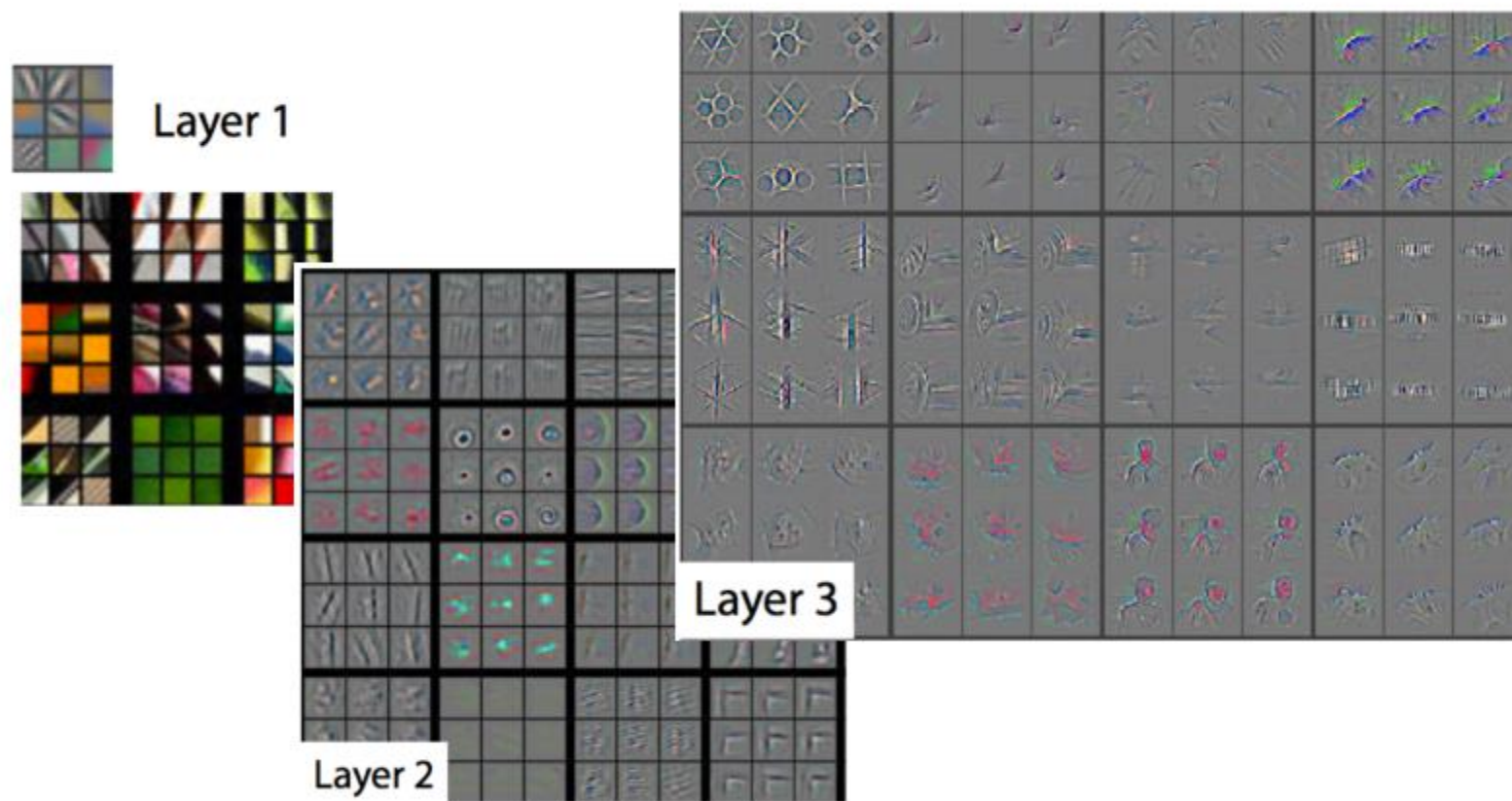
이미지에서는 인근 픽셀끼리만 상관있지 않나?  
가까운 것들끼리만 묶어서 계산하면 의미도 있고  
계산량도 줄겠는데?





Convolution의 좋은점

부품을 조립해 더 복잡한 부품을 만든다

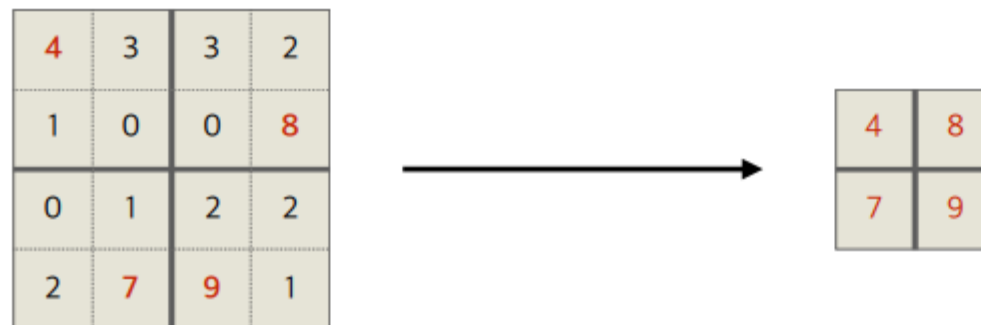


<https://arxiv.org/pdf/1311.2901.pdf>

점점 더 멀리서 보는 법?

우리가 멀어져도 되지만  
그림을 줄여도 되겠구나?

### 사이즈를 점진적으로 줄이는 법 MaxPooling



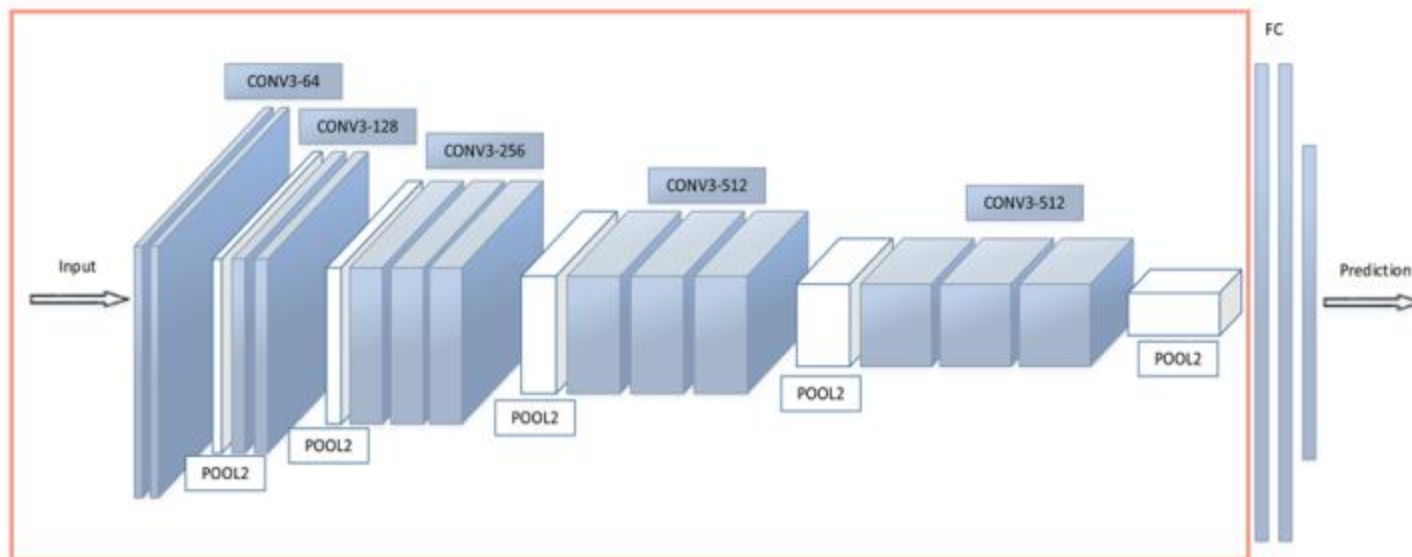
$n \times n$ (Pool)을 중요한 정보(Max)한개로 줄인다.  
 선명한 정보만 남겨서, 판단과 학습이 쉬워지고  
 노이즈가 줄면서, 덤으로 융통성도 확보된다.

stride라고 해서

보통 2x2로 하면 전역에 적용한다 좌우로 몇칸씩 뺄지 설정. 보통(2x2)

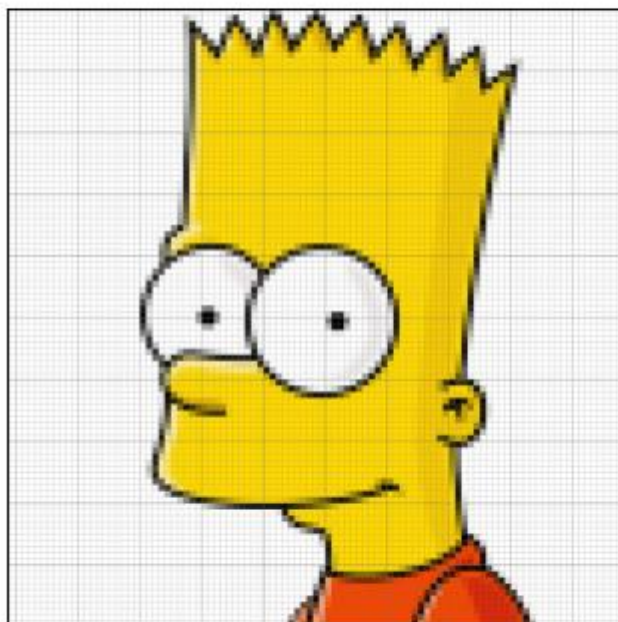
**그러면 절반짜리  
이미지가 완성!**

패턴들을 쌓아가며 점차 복잡한 패턴을 인식한다(conv)



사이즈를 줄여가며, 더욱 추상화 해나간다(maxpooling)

후반부에는 추상화 부품으로 남는다.



시작은 이렇게 256\*256  
픽셀을 다 보았어야 해도



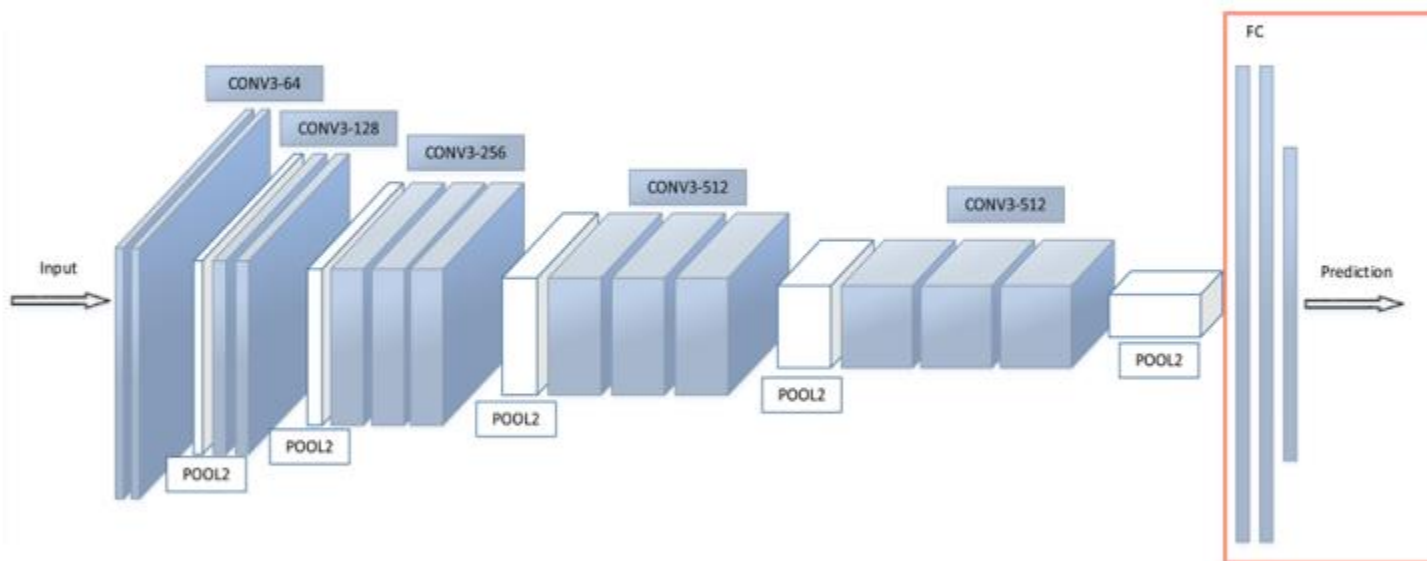
Conv와  
MaxPooling  
의 반복



우리는 궁극적으로  
이런 녀석을 가지게 된다.



막판, 추상화가 끝난 데이터를 FC에 넣어 판단한다  
(fully connected layer)

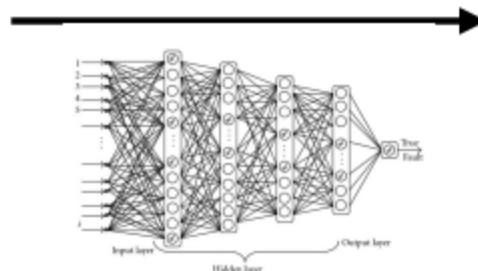




최종판단은 Fully Connected Layer에게 먹여서 하게 한다.



눈과 코와 귀가 있고  
티를 입고 있으니



“개”	X
“고양이”	X
“사람”	O
“말”	X

### 뉴넷넷에게 답을 회신받는 3가지 방법

Value

이게 얼마가 될거같니?

**output을  
그냥 받는다.**

O/X

기냐? 아니냐?

**output에  
sigmoid를 먹인다.**

Category

종류중에 요건 뭐냐?

**output에  
softmax를 먹인다.**

### 컨볼루션 필터(커널)의 효과

■ 있는 그대로

0	0	0
0	1	0
0	0	0



■ 흐려짐

1	1	1
1	1	1
1	1	1



■ 윤곽 탐지

0	1	0
1	-4	1
0	1	0



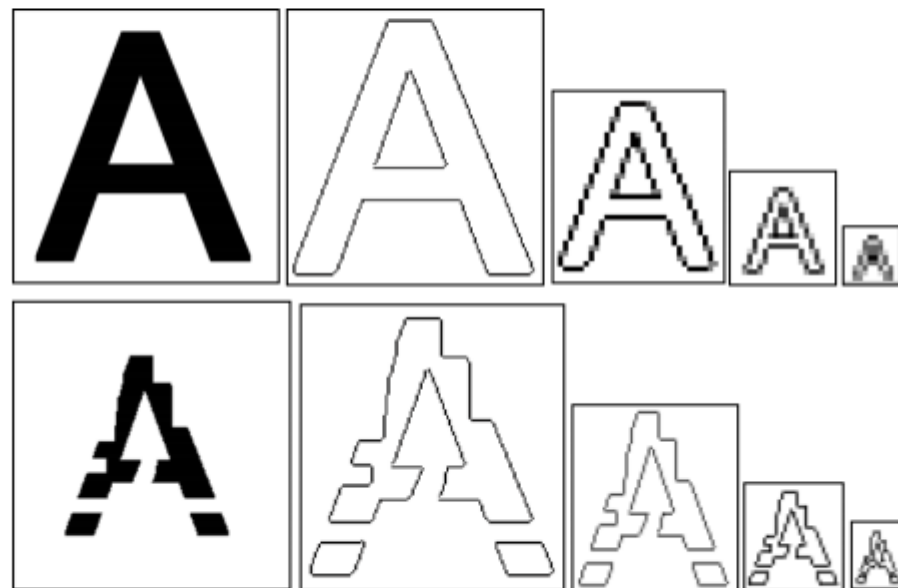
■ 선명하게

0	-1	0
-1	5	-1
0	-1	0



### Subsampling(Pooling)의 효과

- 이미지의 변형에 대한 불변성 확보



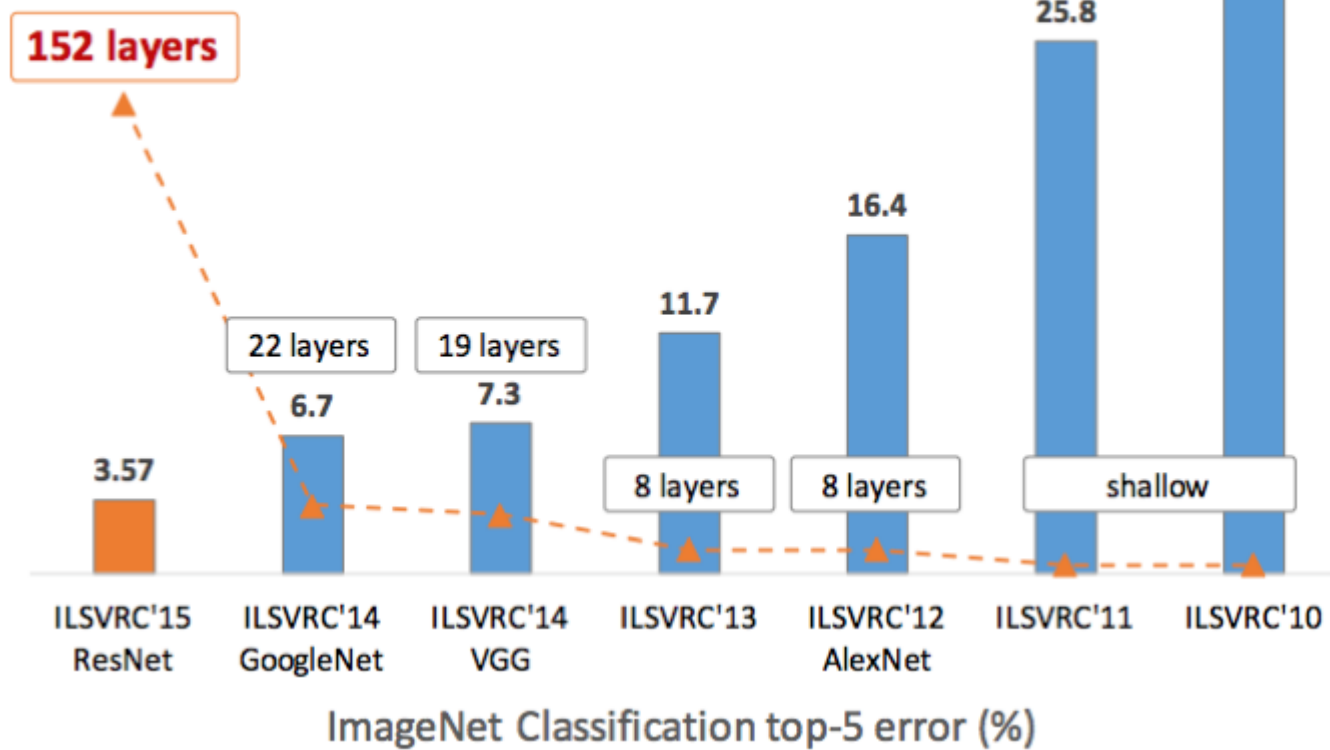
출처 : 딥러닝 인공지능의 과거, 현재 그리고 미래 9기 차세대 에너지 리더 과정

## 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..

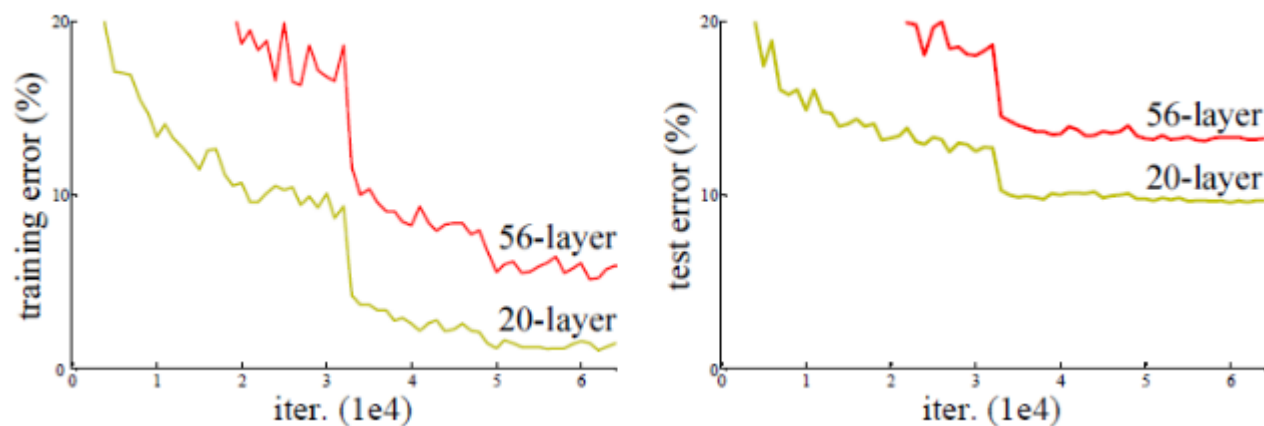
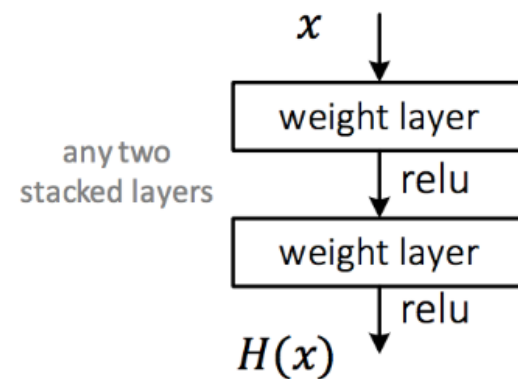


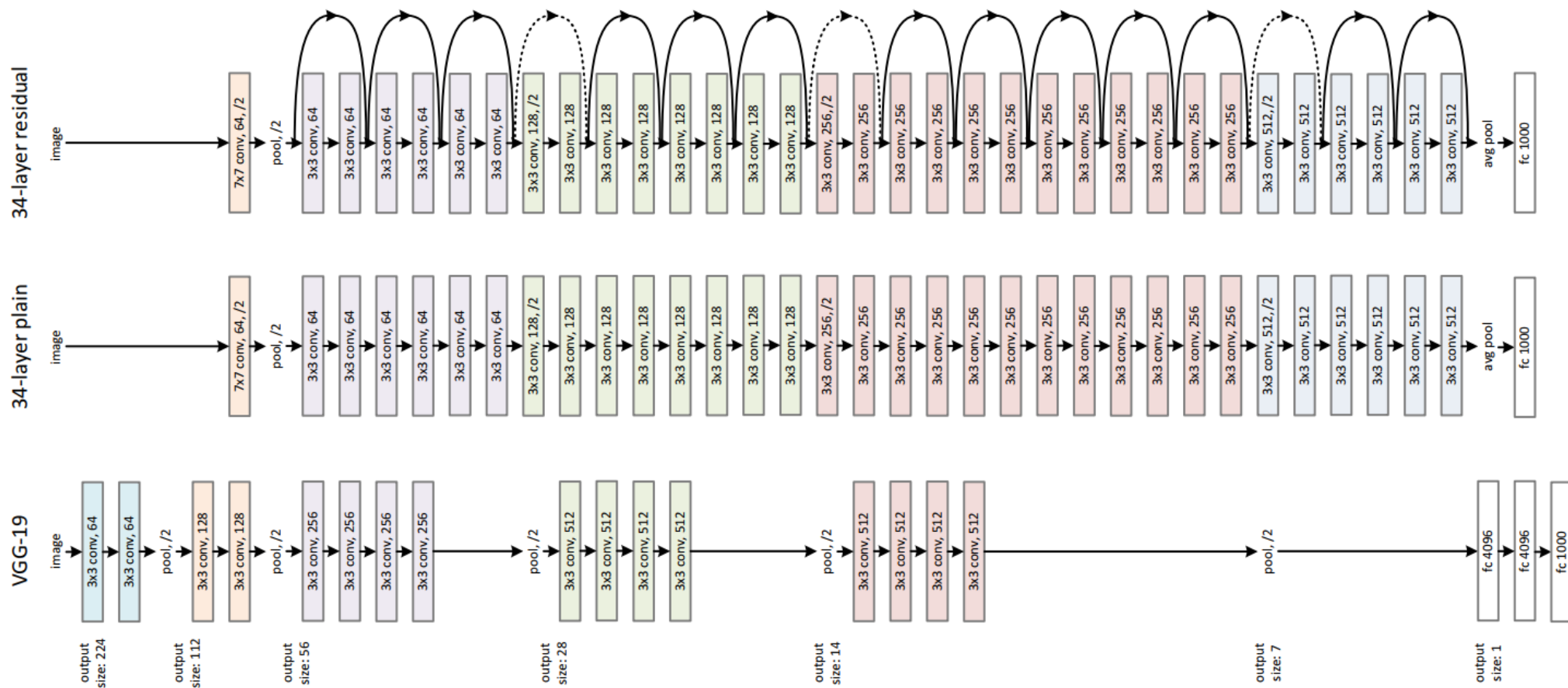
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

### • Plain net



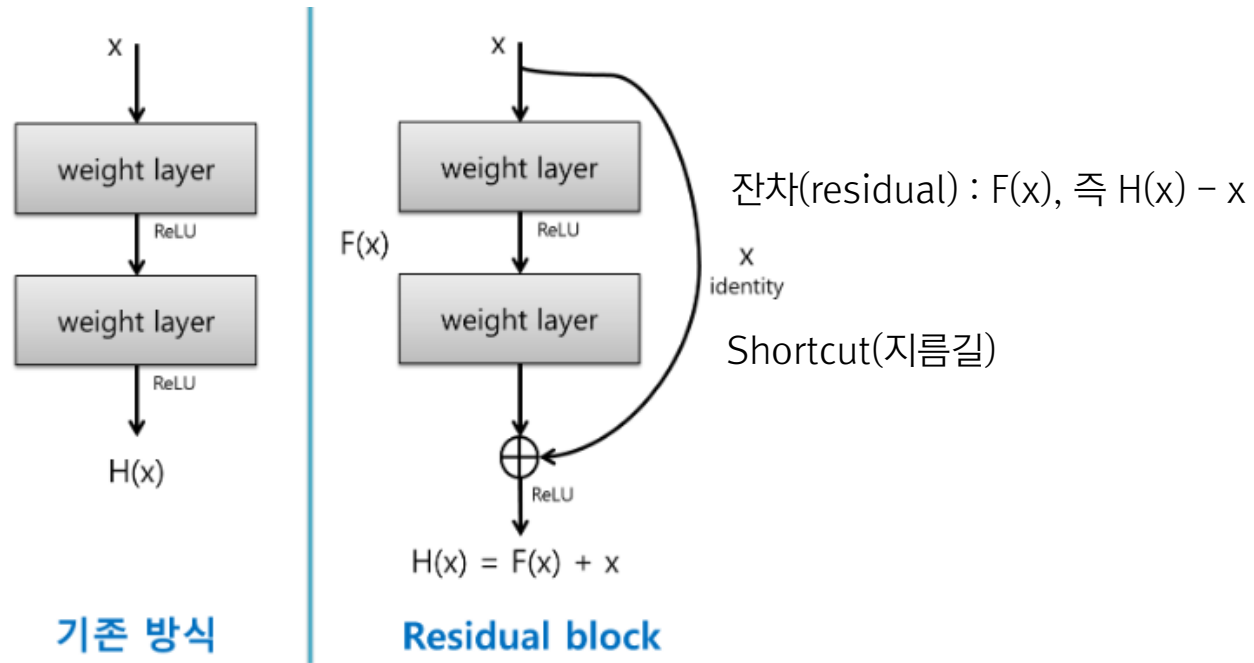
Plain Net이란 단순히 layer를 쌓은 네트워크를 의미 (ex : VGG19)

## ◆ 망을 깊게 하면 결과가 더 좋아질까? : Degration





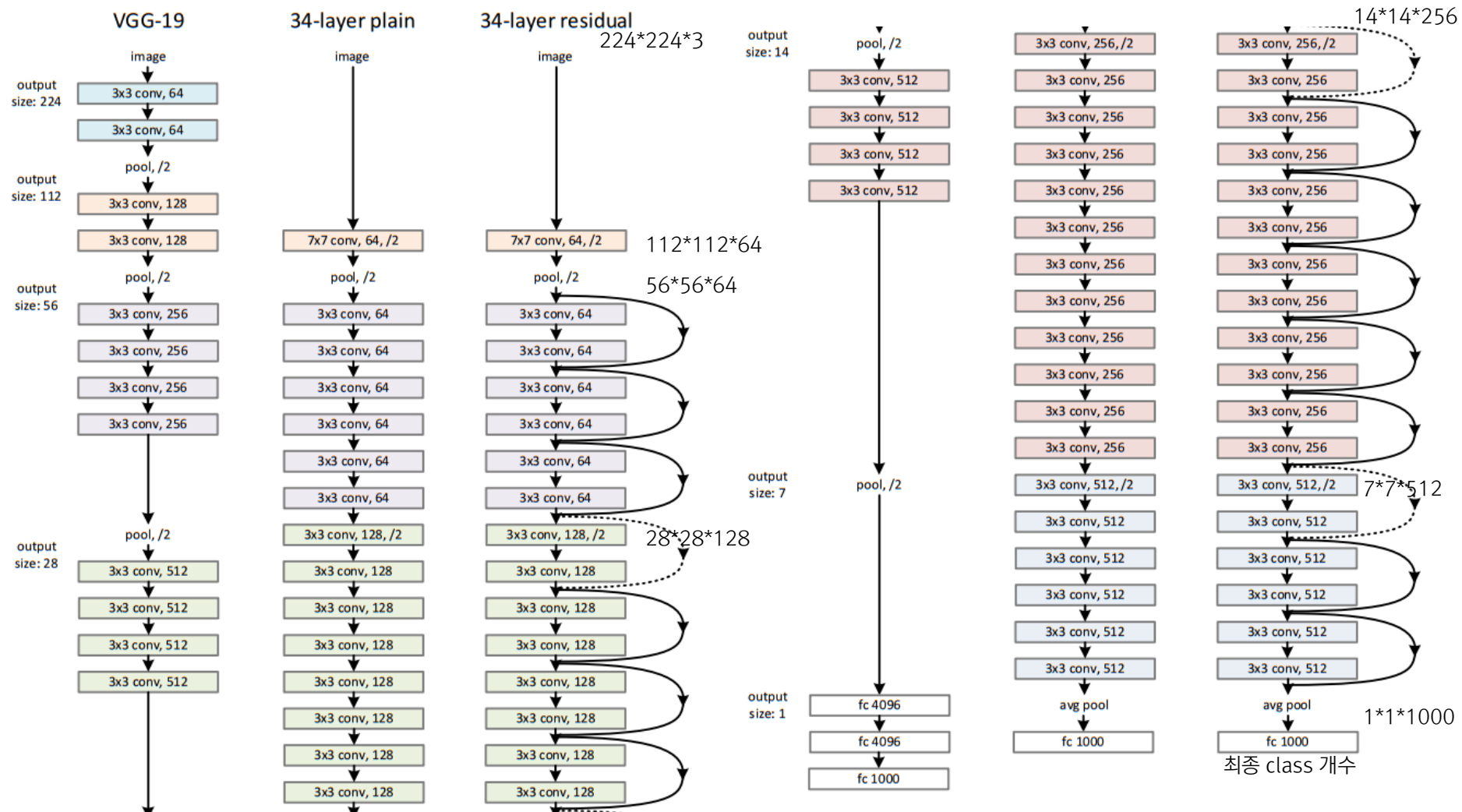
## ◆ 잔차를 학습하는 방법



- 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

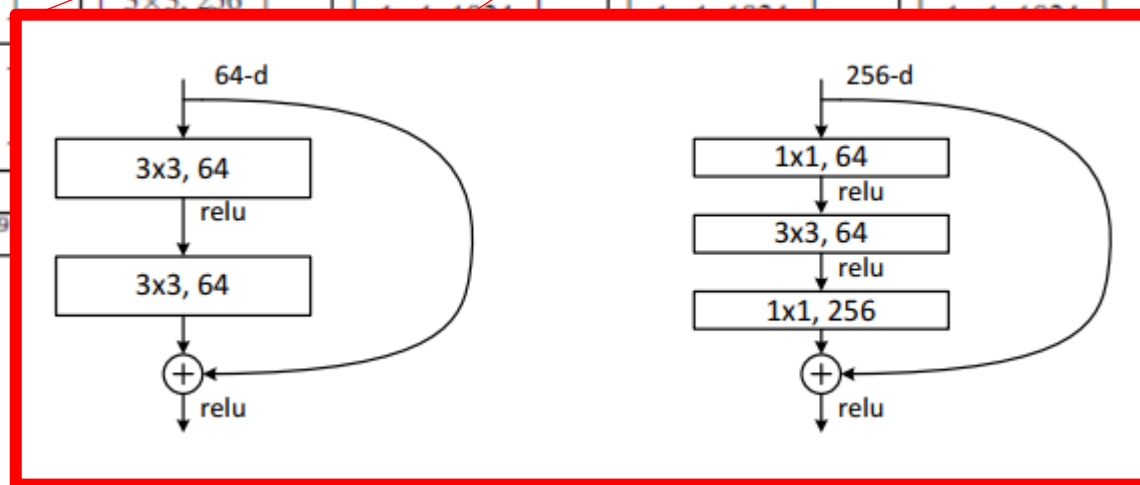
기본구조 (ImageNet-ILSVRC) (224\*224\*3)

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		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

## ◆ 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				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
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	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$				
	1×1					
FLOPs		$1.8 \times 10^9$				



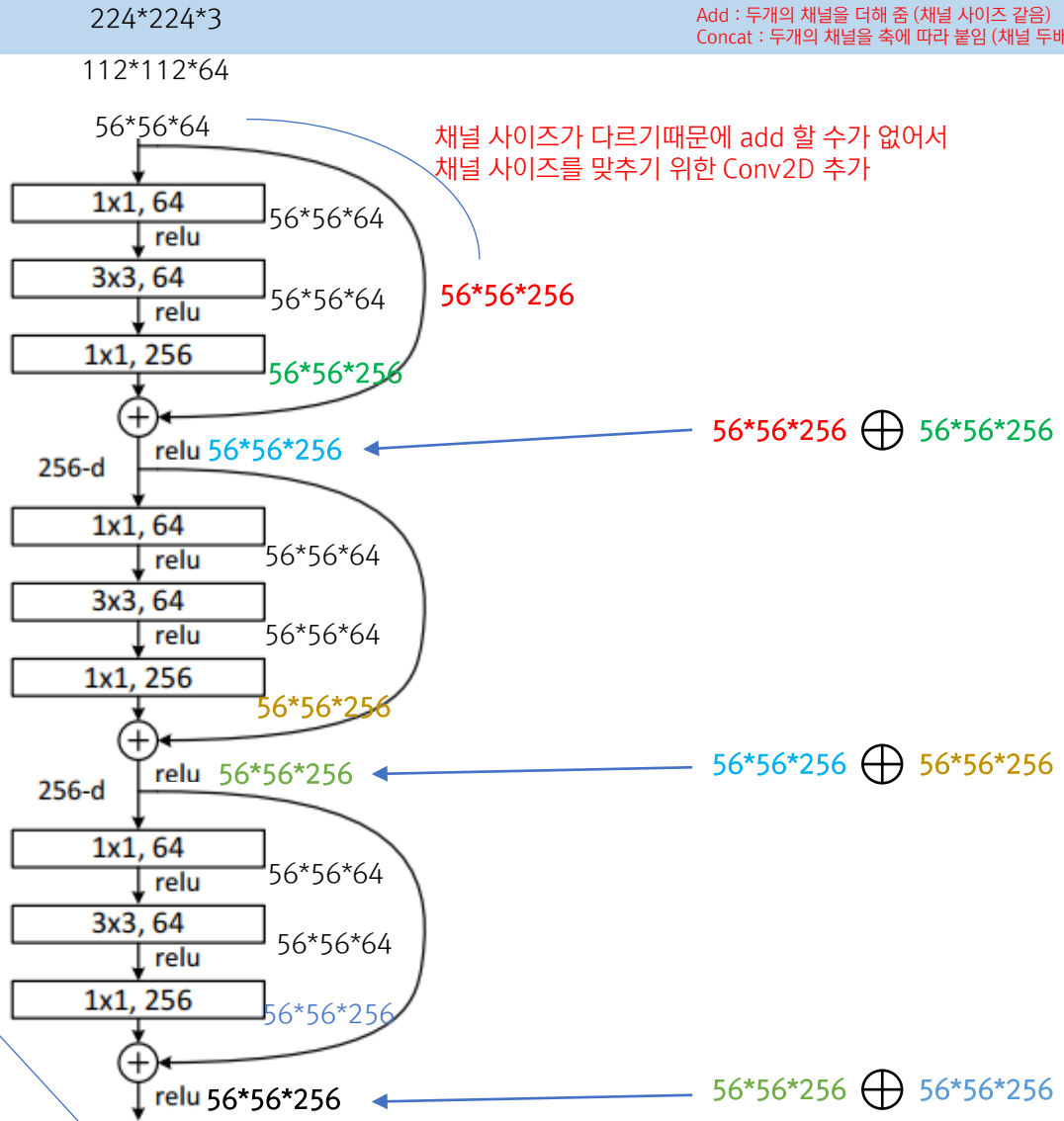
## ◆ 모델 구조

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

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

Bottleneck?

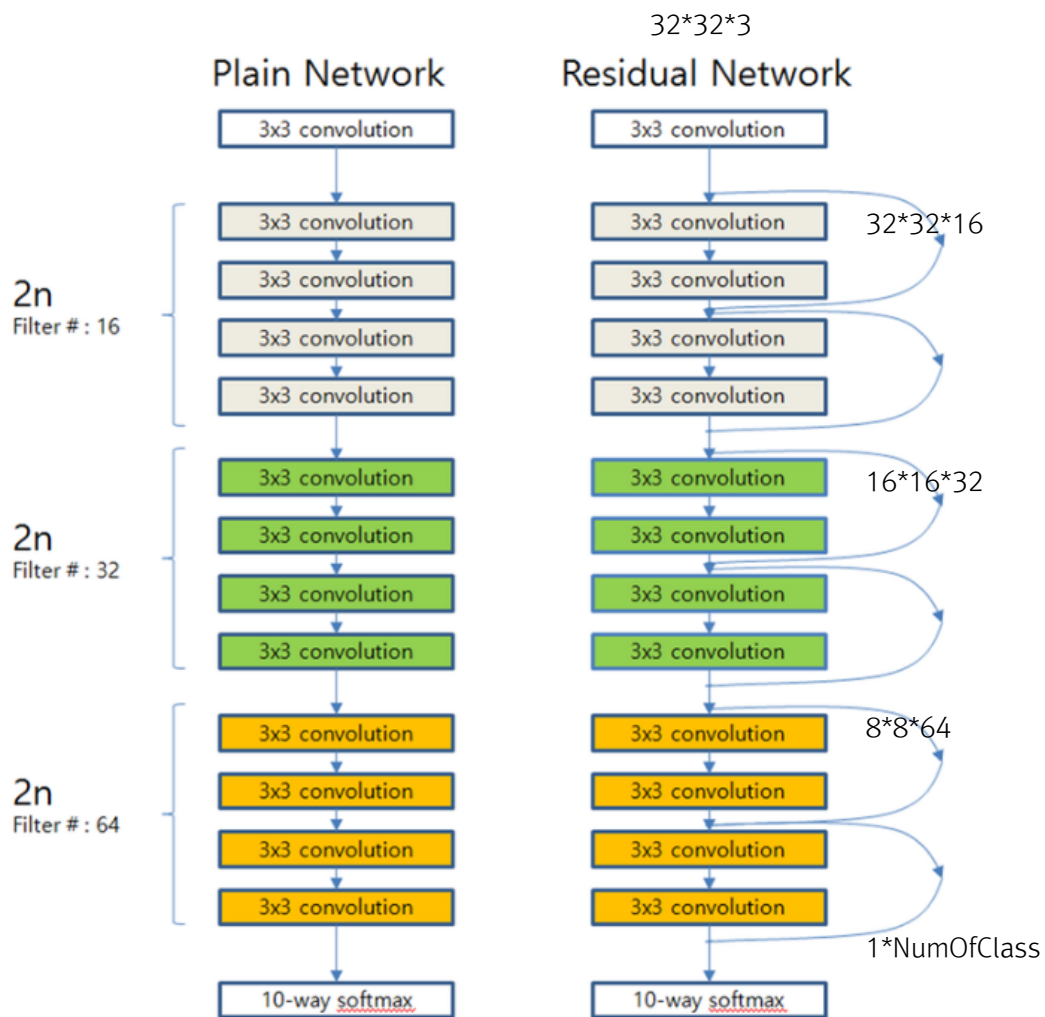
$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$
$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3,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



## ◆ 모델 구조

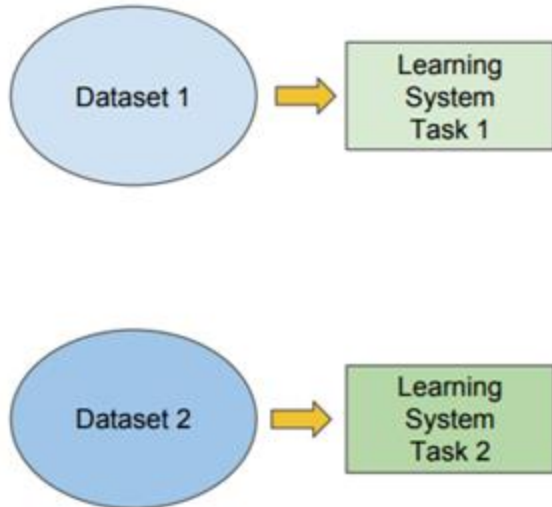
Cifar10 ( $32 \times 32 \times 3$ )을 위한 모델

출처 :

<https://blog.naver.com/PostView.nhn?blogId=laonple&logNo=220770760226>


## Traditional ML

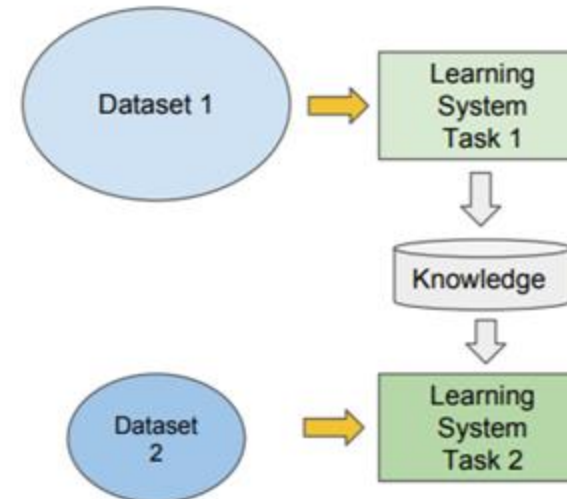
- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



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`. A callout box on the right lists the model architectures supported by the `models` subpackage for image classification.

**PyTorch** | Get Started | Ecosystem | Mobile | Blog | Tutorials | Docs | Resources | GitHub

0.11.0 ▼

Search Docs

Package Reference

- `torchvision.datasets`
- `torchvision.io`
- `torchvision.models`**
- `torchvision.models.feature_extraction`
- `torchvision.ops`
- `torchvision.transforms`
- `torchvision.utils`

Examples and training references

- Example gallery
- Training references

PyTorch Libraries

- PyTorch
- `torchaudio`
- `torchtext`
- `torchvision`

Docs > `torchvision.models`

### TORCHVISION.MODELS

The `models` subpackage contains definitions of models for addressing different tasks, including: image classification, semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

**NOTE**

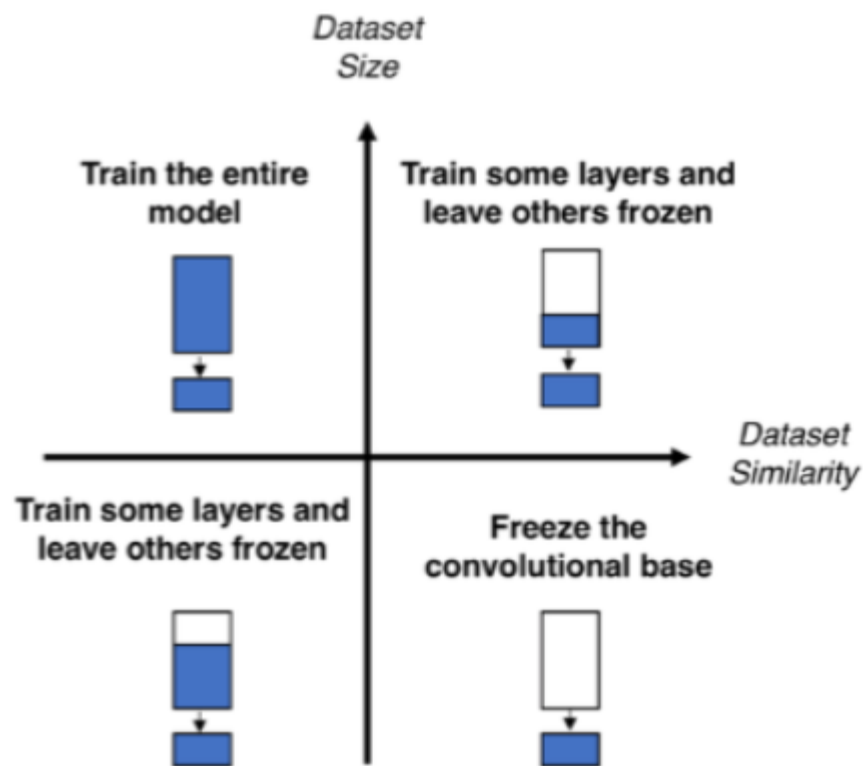
Backward compatibility is guaranteed for loading a serialized `state_dict` to the model created in version 0.11.0. On the contrary, loading entire saved models or serialized `ScriptModules` (serialized using `torch.jit.trace` or `torch.jit.trace_module`) may not preserve the historic behaviour. Refer to the following [documentation](#).

### Classification

The `models` subpackage contains definitions for the following model architectures for image classification:

- 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.

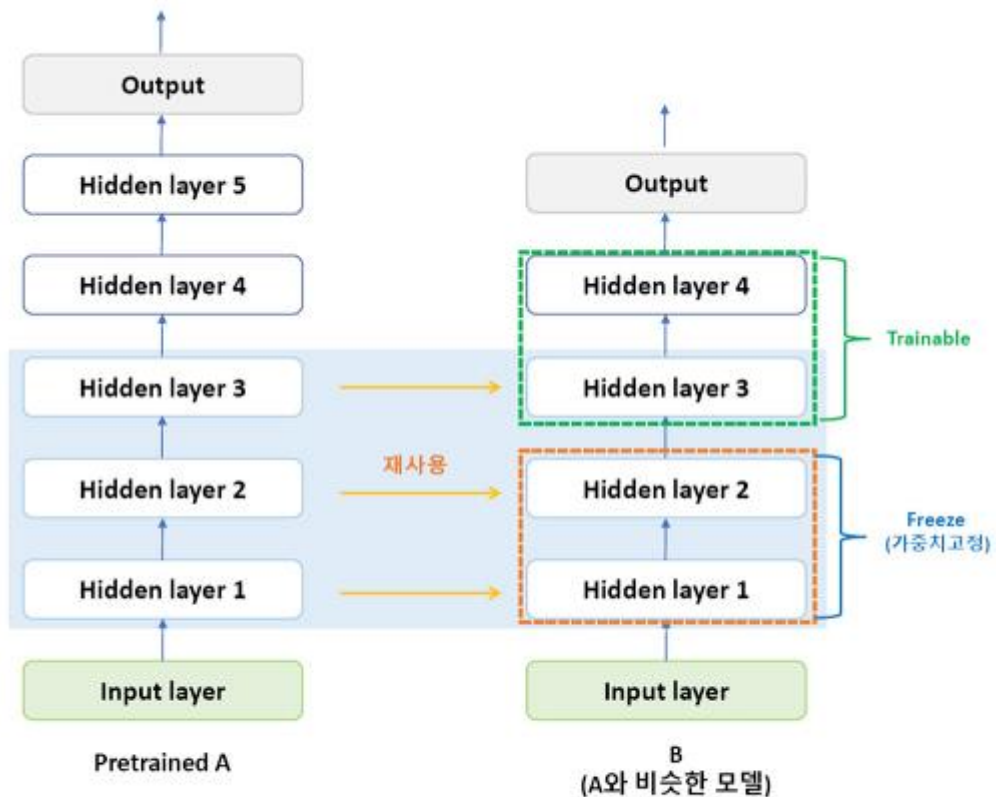
유사한 dataset을 사용한다면 시간절약을 위해 Conv. layer의 후반 계층과 FC layer만 학습을 진행해줍니다. 학습을 하지 않는 부분은 lr=0, 학습을 하는 부분은 lr = 기존의 1/10 으로 지정해줍니다.

### 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
```

# 감사합니다

