Pytorch를 활용한 딥러닝 학습 환경 구축 및 실습

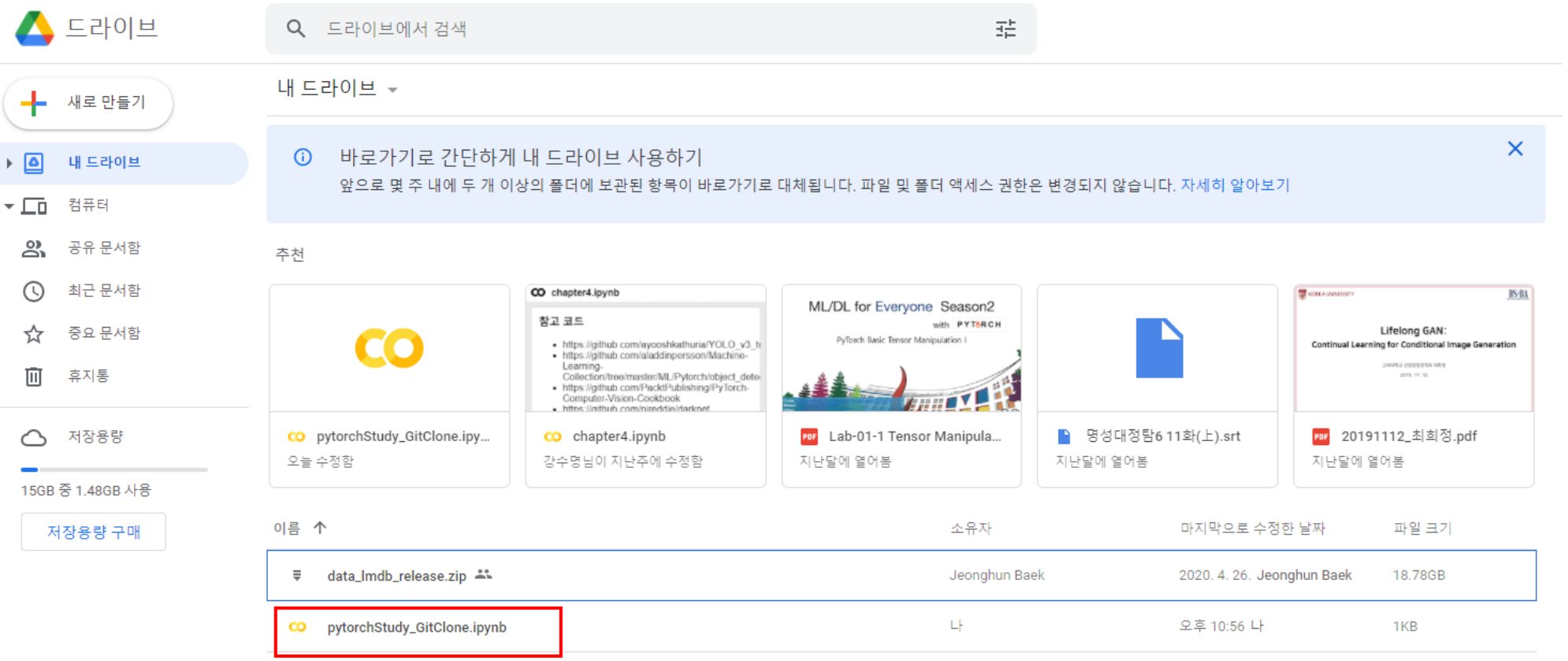
- 3일차 -

강 수 명

smgang.kmu@gmail.com

CONTENTS

00 Data, 학습자료


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.ipy...
오늘 수정함
chapter4.ipynb
참고 코드
chapter4.ipynb
강수명님이 지난주에 수정함
ML/DL for Everyone Season2
with PYTORCH
Lab-01-1 Tensor Manipula...
지난달에 열어봄
명성대정탐6 11화(上).srt
지난달에 열어봄
Lifelong GAN:
Continual Learning for Conditional Image Generation
20191112_최희정.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

00 Data, 학습자료

pytorchStudy_GitClone.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

파일

..
gdrive
MyDrive
data_Imdb_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
```

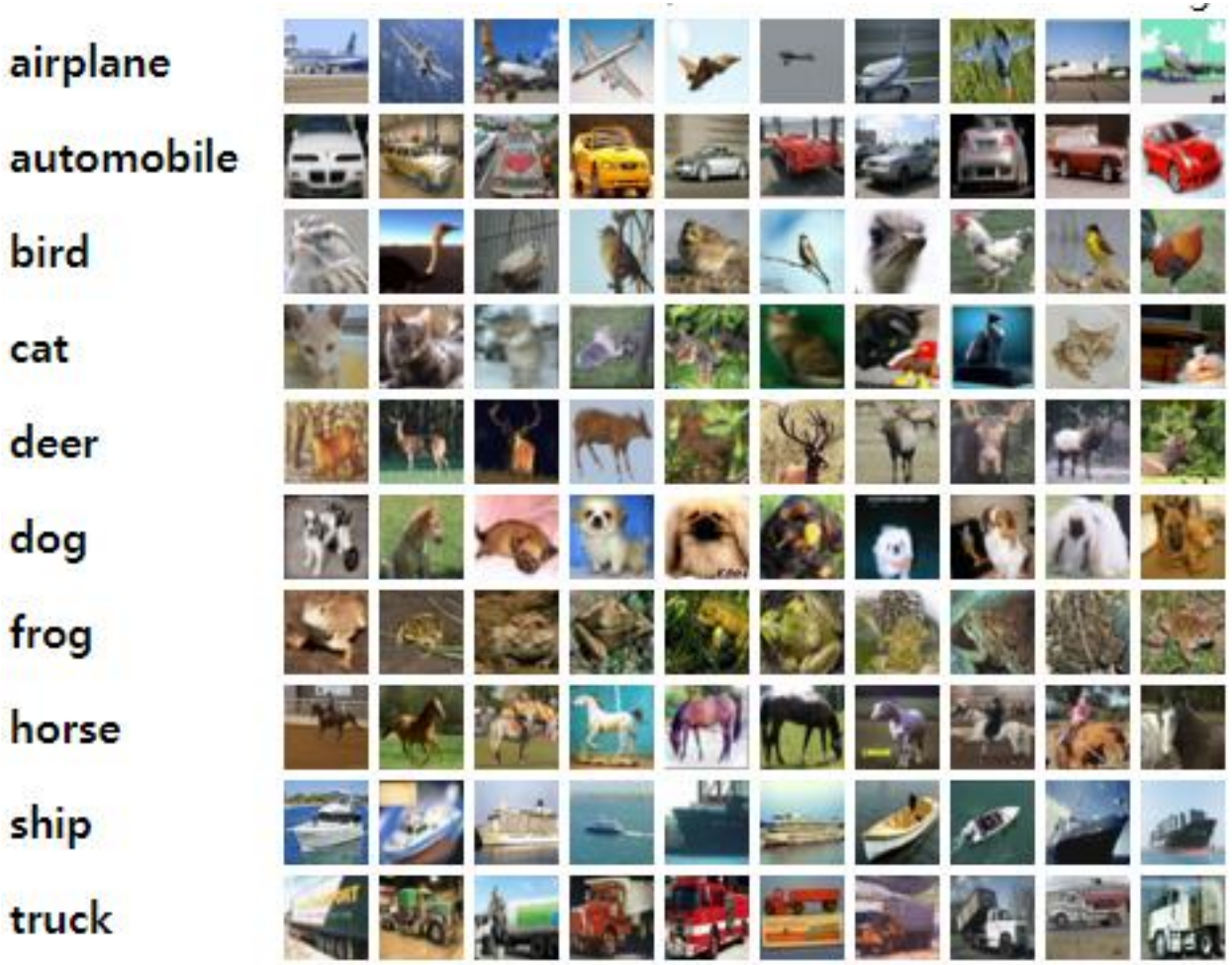
4초
```
[14] !git clone https://github.com/smgang/pytorch_Study.git
```

```
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

CNN(Convolution Neural Network)

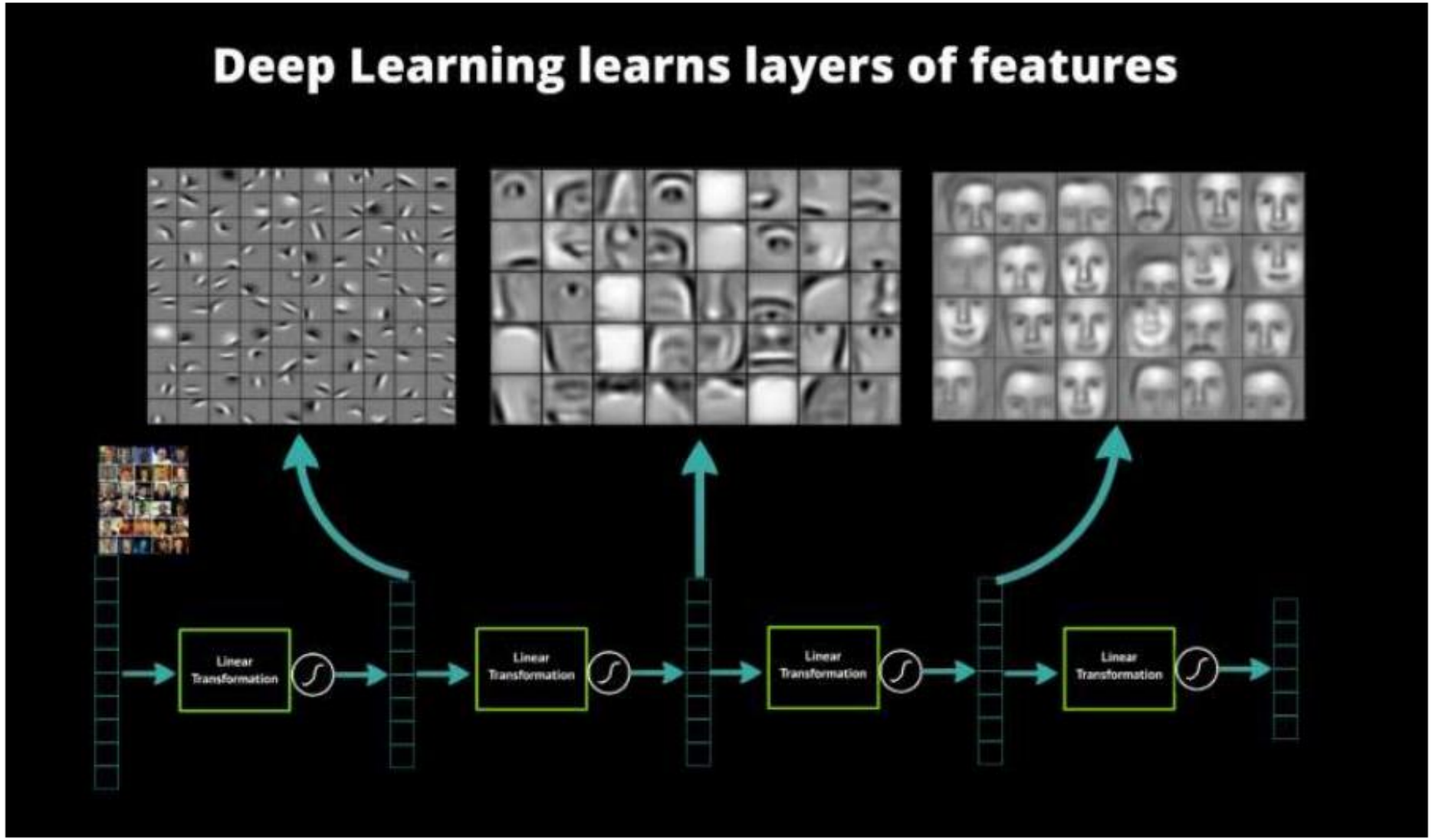


The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

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

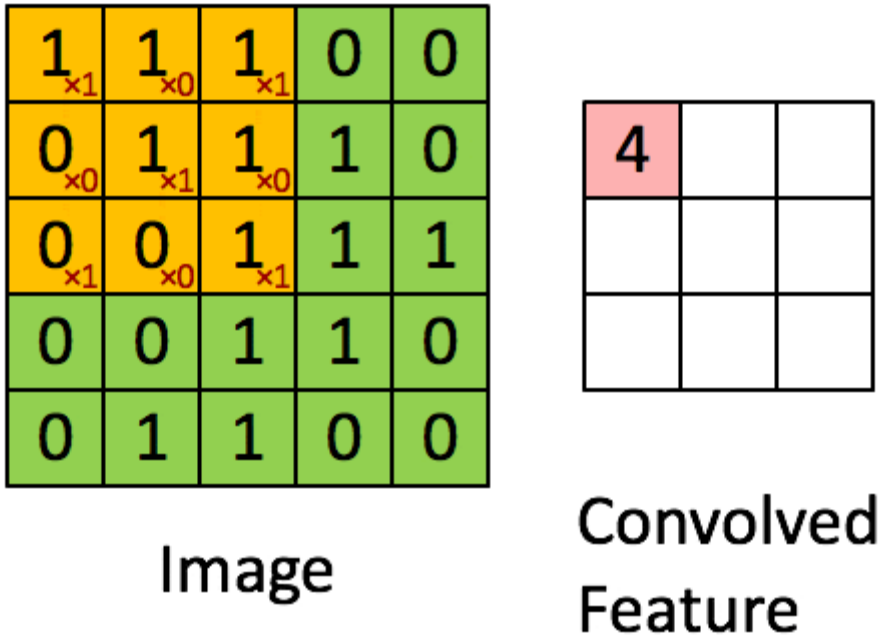
https://github.com/YoongiKim/CIFAR-10-images

CNN(Convolution Neural Network)



CNN(Convolution Neural Network)

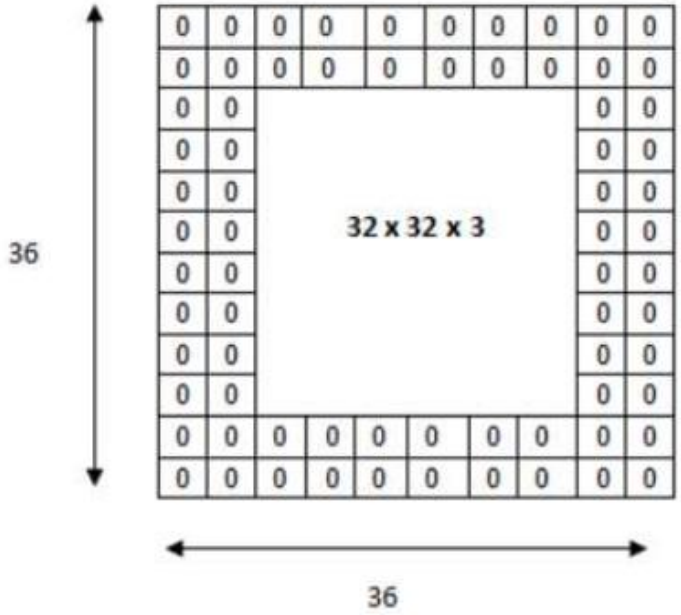
컨볼루션이 뭔데요?



출처 : https://m.blog.naver.com/win0k/221603529605

CNN(Convolution Neural Network)

영상이 작아져버렸네.. 왼쪽 끝이랑 오른쪽 끝 픽셀은 안보나요?

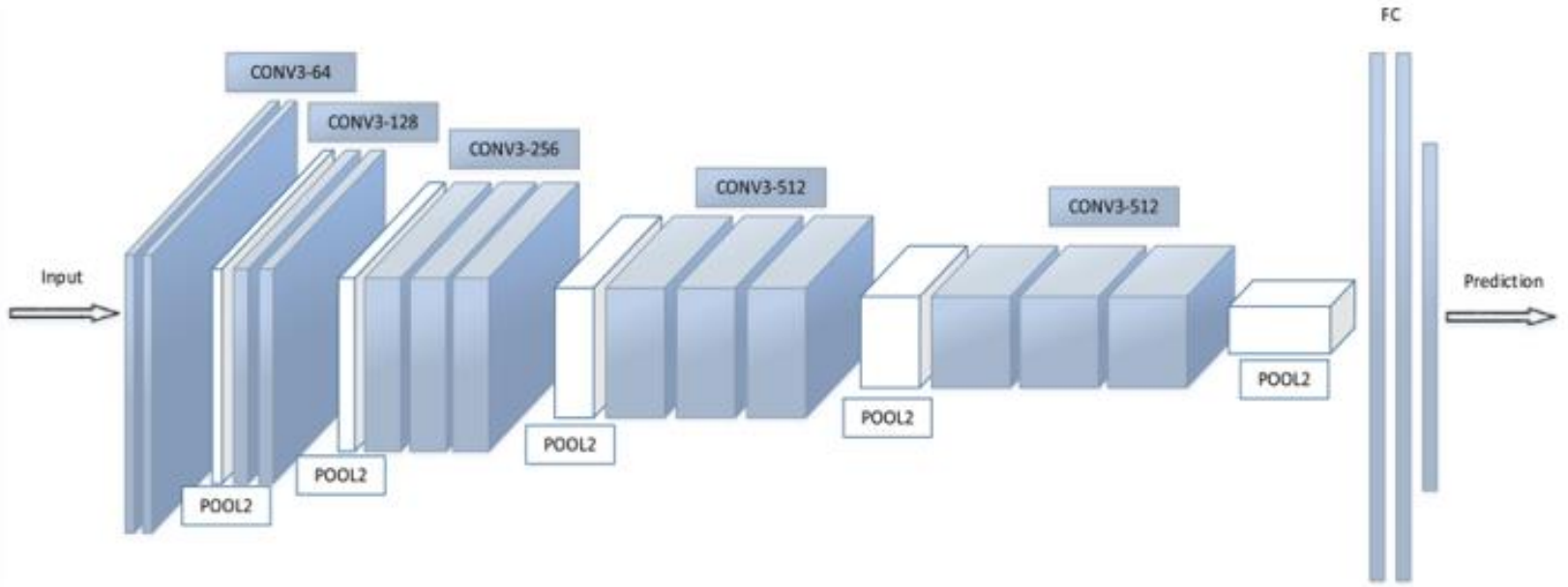


The input volume is 32 x 32 x 3. If we imagine two borders of zeros around the volume, this gives us a 36 x 36 x 3 volume. Then, when we apply our conv layer with our three 5 x 5 x 3 filters and a stride of 1, then we will also get a 32 x 32 x3 output volume.

볼수도 있고, 안볼수도 있습니다.

CNN(Convolution Neural Network)

VGGNET의 구조

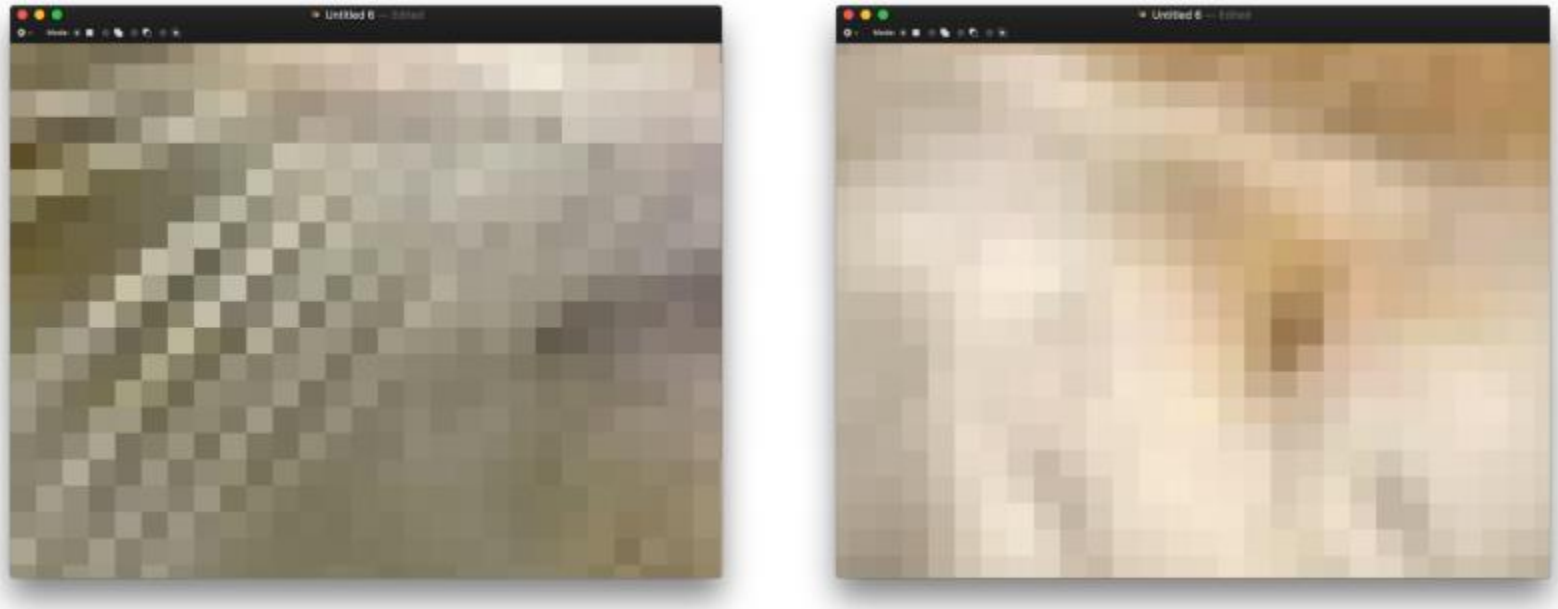


이해를 하기 위해, 우리가 그림을 본다고 상상해봅시다.

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

그림을 눈앞 1cm거리에서 본다고 생각해보자.

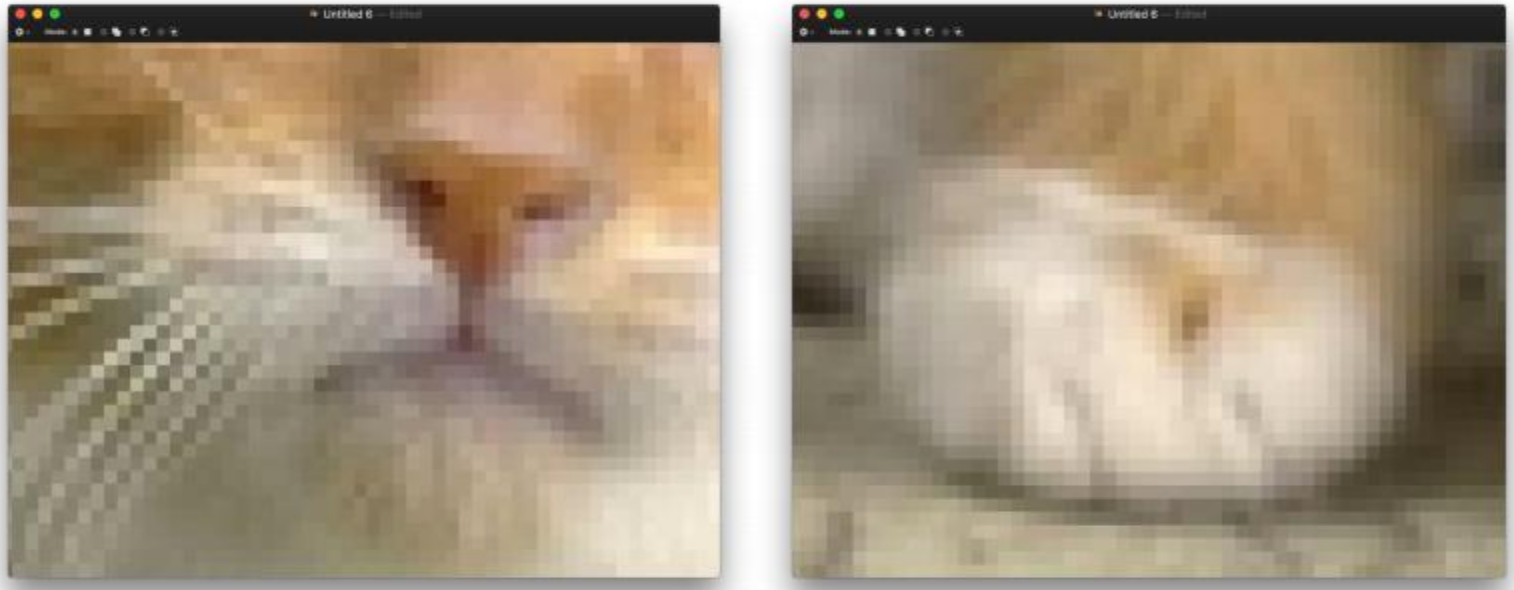


처음에는 점과 선, 이상한 질감 몇개 밖에 모르겠다.

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

점과 선, 질감을 충분히 배우고, 조금 떨어져서 보자.



점과 선이 질감이 합쳐져 삼각형, 동그라미, 북실함이 보인다.

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

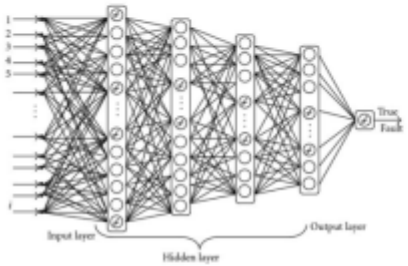
더 멀리서 보니, 그것들이 모아져있다. 이것은?

고양이!!

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

계산량 측면에서도!



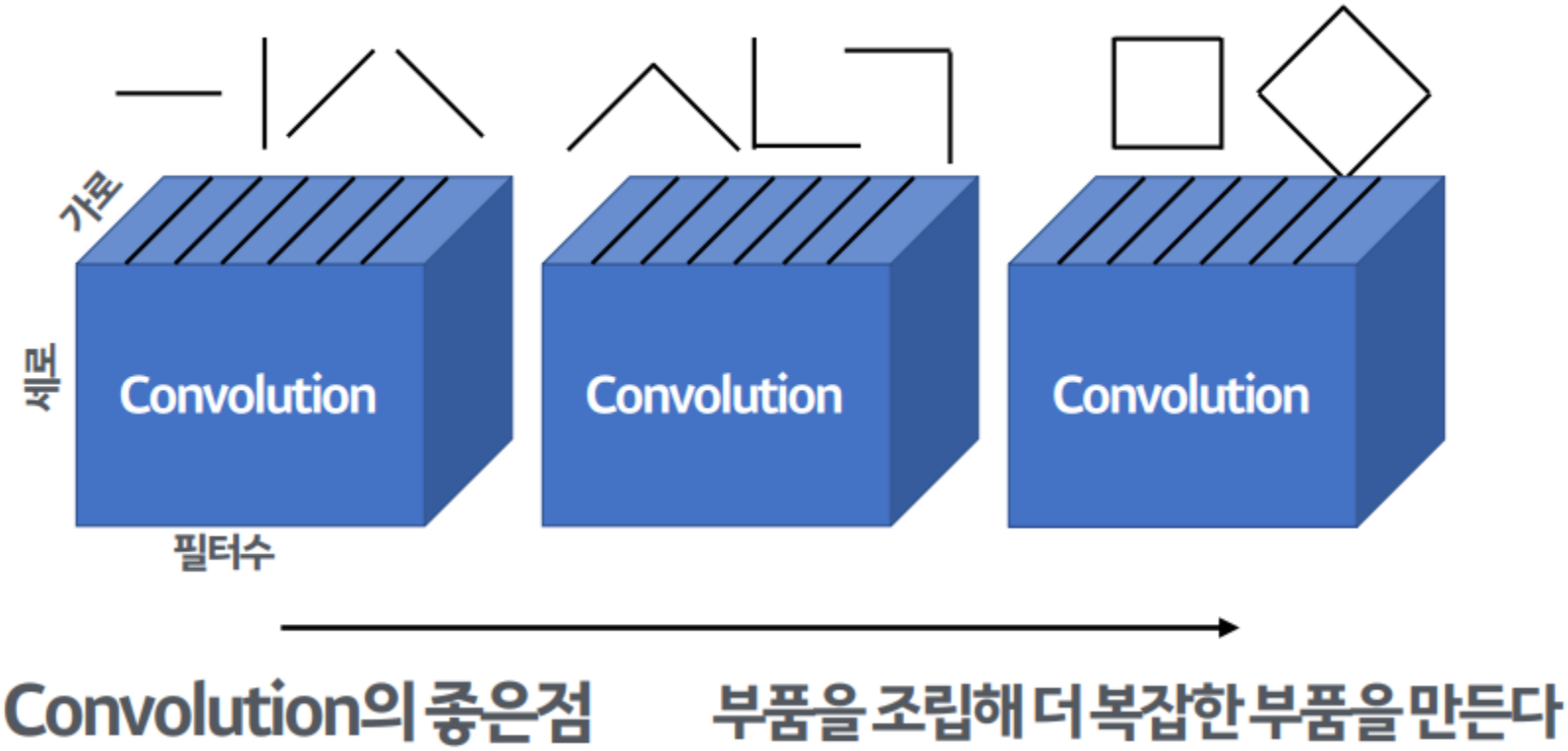
이런생각을해봤다.
보통 뉴럴넷은 서로가 서로에게 전부다 연결되어 있는데
이러다보니 맞춰야할weight들도 많아..

이미지에서는 인근 픽셀끼리만 상관있지 않나?

가까운 것들끼리만 묶어서 계산하면 의미도 있고
계산량도 줄겠는데?

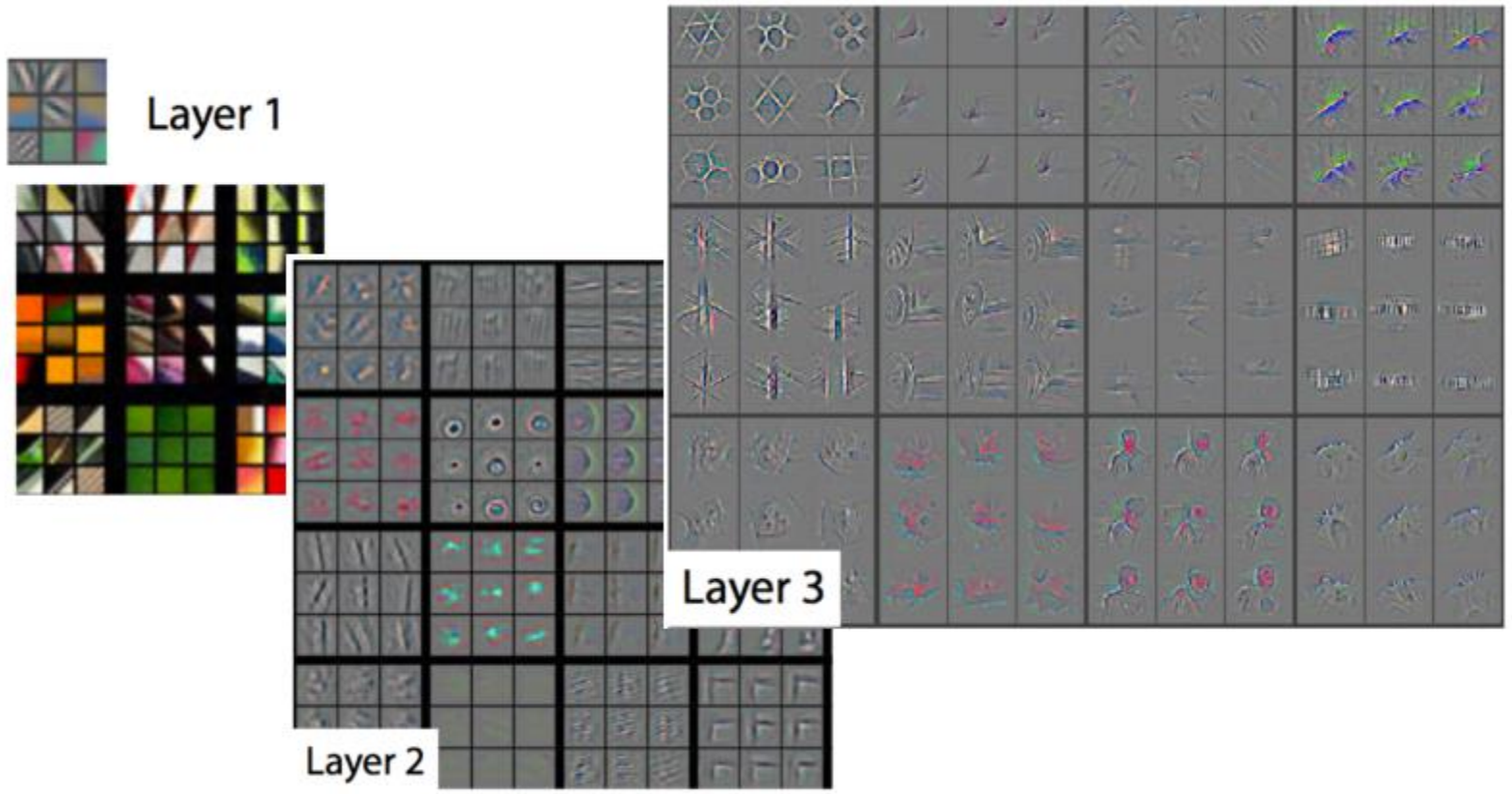
출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)



출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)



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

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

점점 더 멀리서 보는 법?

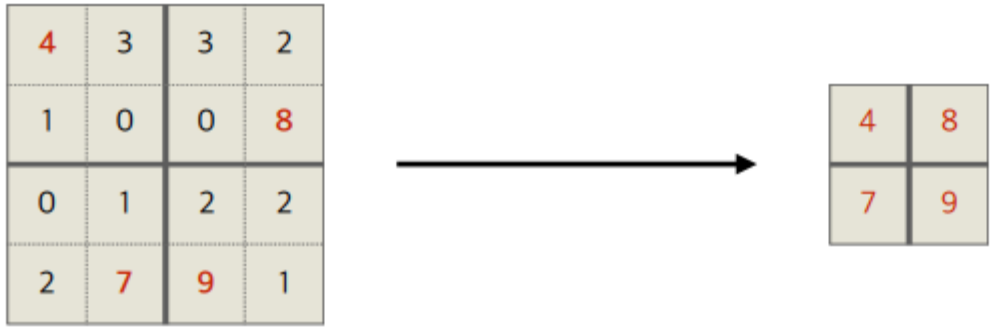
우리가 멀어져도 되지만

그림을 줄여도 되겠구나?

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

사이즈를 점진적으로 줄이는 법 MaxPooling



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

보통 2x2로 화면 전역에 적용한다

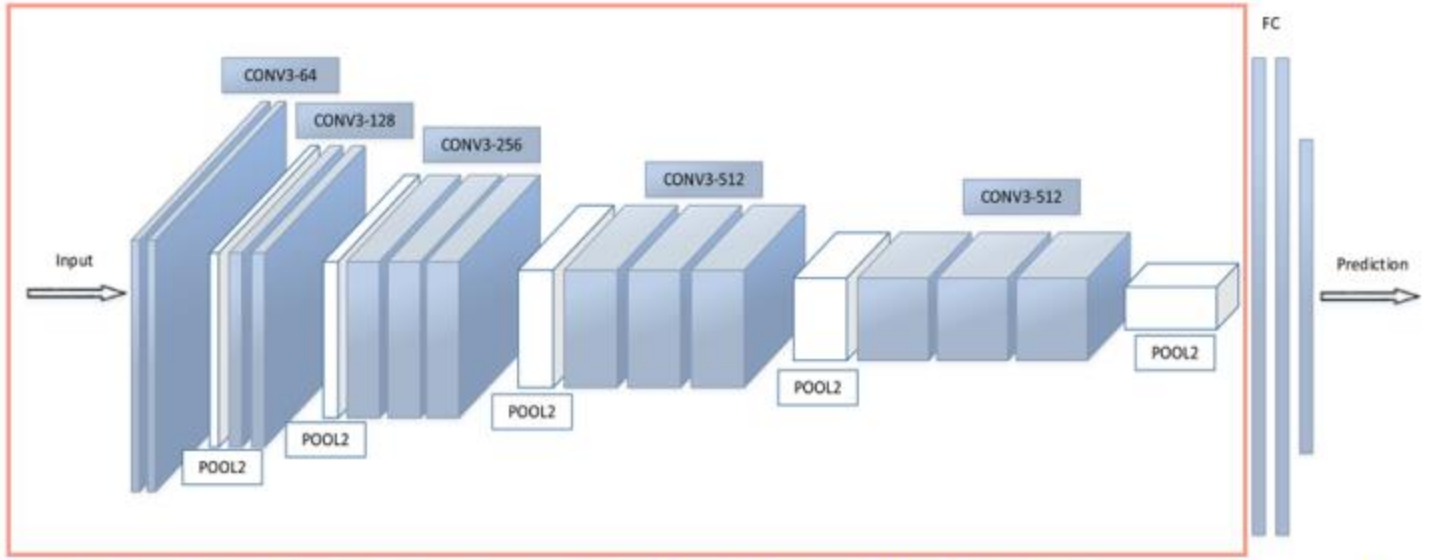
stride라고 해서
좌우로 몇칸씩 뛸지 설정. 보통(2x2)

그러면 절반짜리
이미지가 완성!

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

패턴들을 쌓아가며 점차 복잡한 패턴을 인식한다(conv)

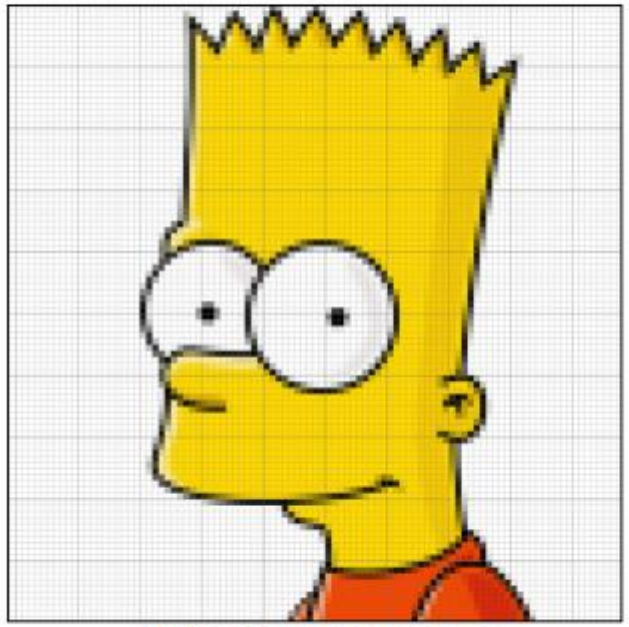


사이즈를 줄여가며, 더욱 추상화 해나간다(maxpooling)

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

후반부에는 추상화 부품으로 남는다.



시작은 이렇게 256*256
픽셀을 다 보았어야 해도

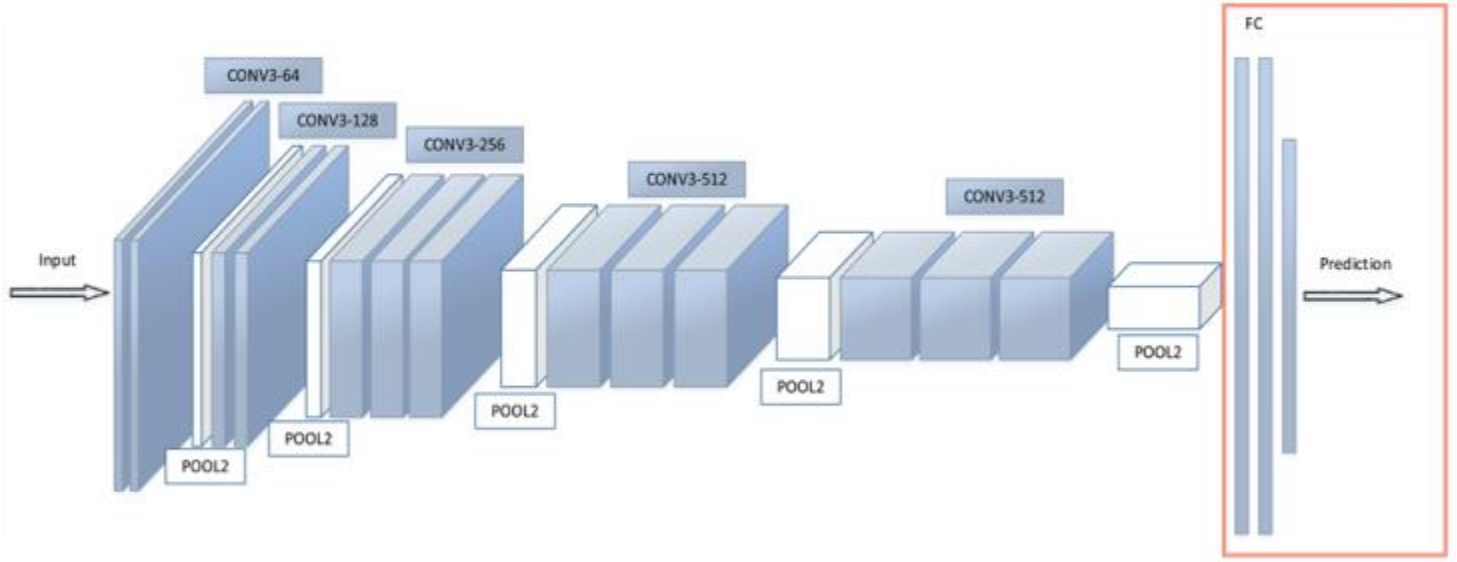
Conv와
MaxPooling
의 반복

우리는 궁극적으로
이런 녀석을 가지게 된다.

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

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

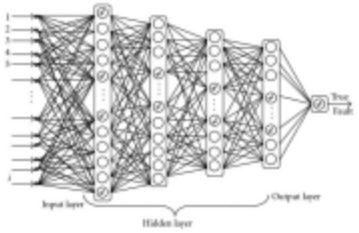


출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

최종판단은 Fully Connected Layer에게 먹여서 하게 한다.

눈과 코와 귀가 있고
티를 입고 있으니



"개" X

"고양이" X

"사람" O

"말" X

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

뉴럿넷에게 답을 회신받는 3가지 방법

Value	O/X	Category
이게 얼마가 될거 같니?	기냐? 아니냐?	종류중에 요건 뭐냐?
output을 그냥 받는다.	output에 sigmoid를 먹인다.	output에 softmax를 먹인다.

출처 : 하용호, 백날 자습해도 이해 안가던 딥러닝 머리속에 인스톨 시켜드립니다

CNN(Convolution Neural Network)

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

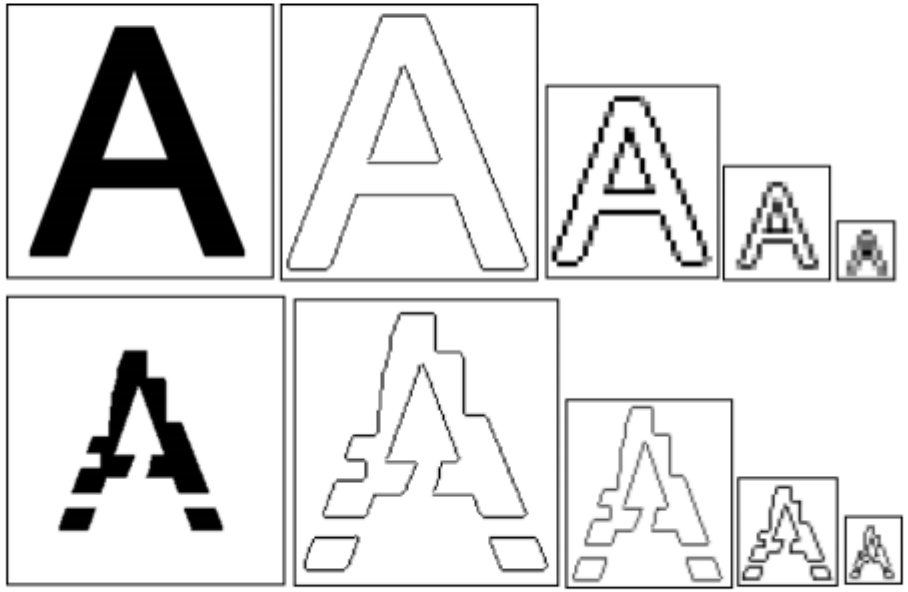
0	−1	0
−1	5	−1
0	−1	0

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

CNN(Convolution Neural Network)

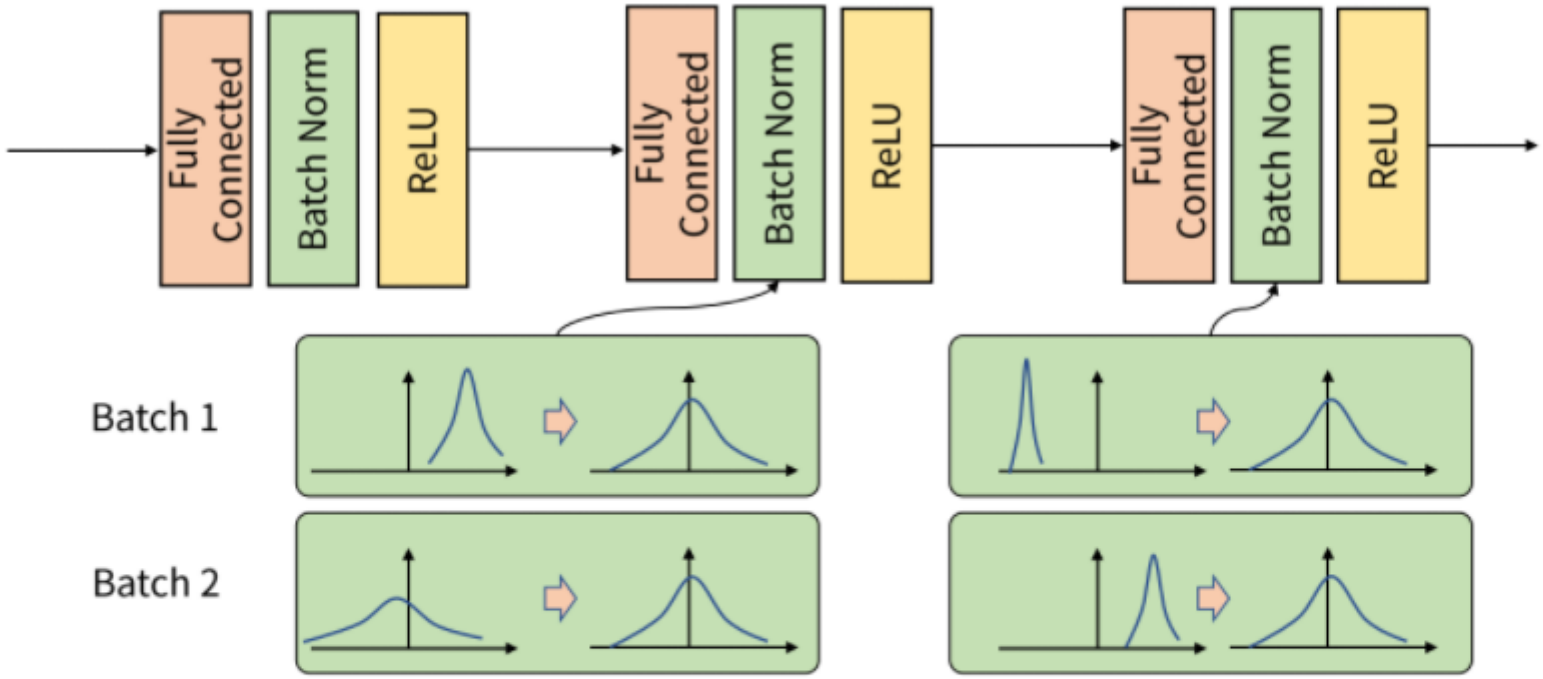
Subsampling(Pooling)의 효과

- 이미지의 변형에 대한 불변성 확보



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

BatchNorm

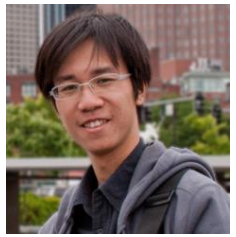


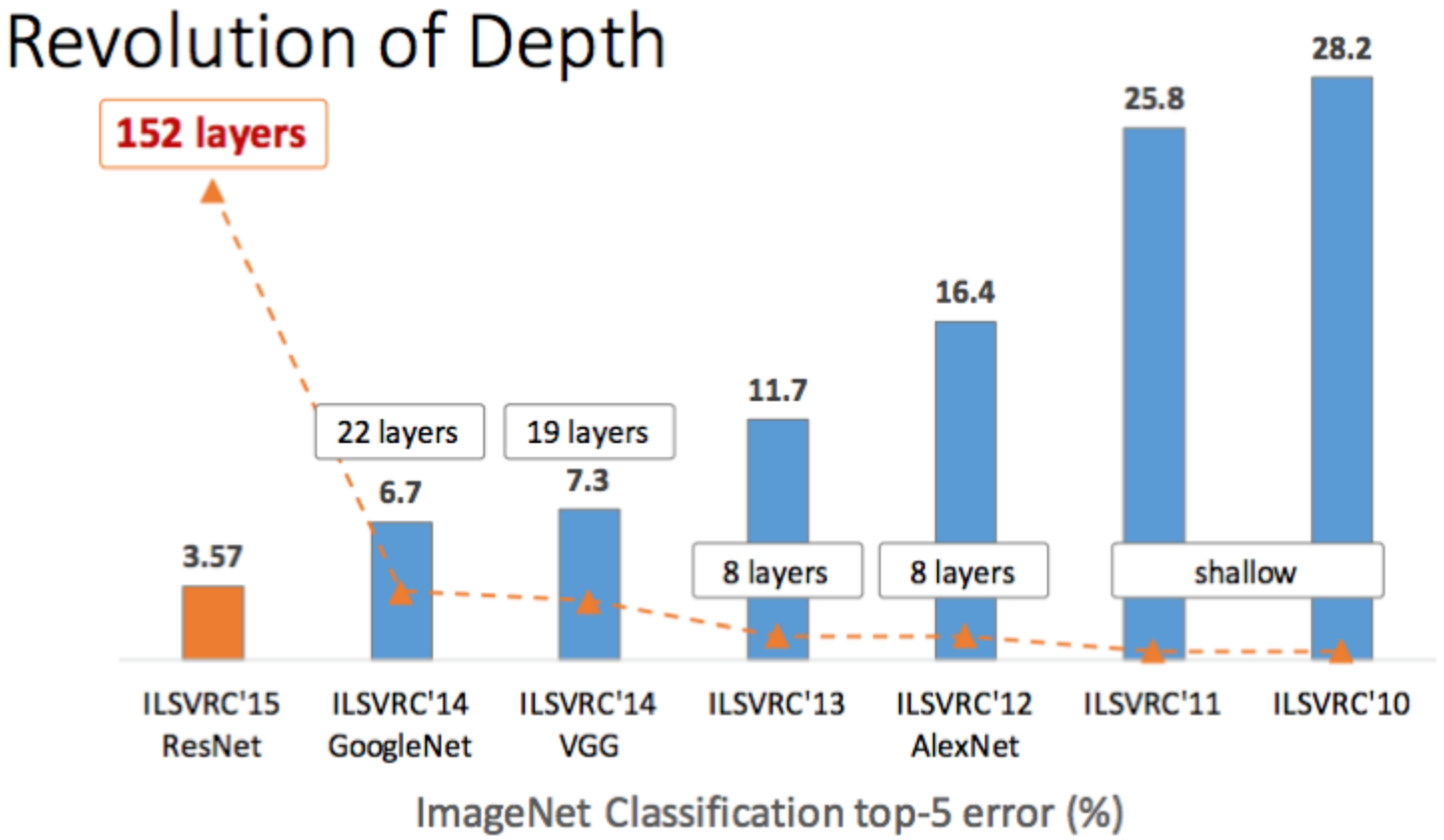
- batch normalization은 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 **각 배치별로 평균과 분산을 이용해 정규화**하는 것을 뜻합니다.
- 위 그림을 보면 batch 단위나 layer에 따라서 입력 값의 분포가 모두 다르지만 정규화를 통하여 분포를 zero mean gaussian 형태로 만듭니다.
- 그러면 평균은 0, 표준 편차는 1로 데이터의 분포를 조정할 수 있습니다.

https://gaussian37.github.io/dl-concept-batchnorm/

Deep Residual Learning for Image Recognition

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





본격적으로 인간 인식률(94.90%)를 추월

◆ 망을 깊게 하면 결과가 더 좋아질까? : Degration

질문의 배경은 VGG로 인한 좋은 결과로 인해서..

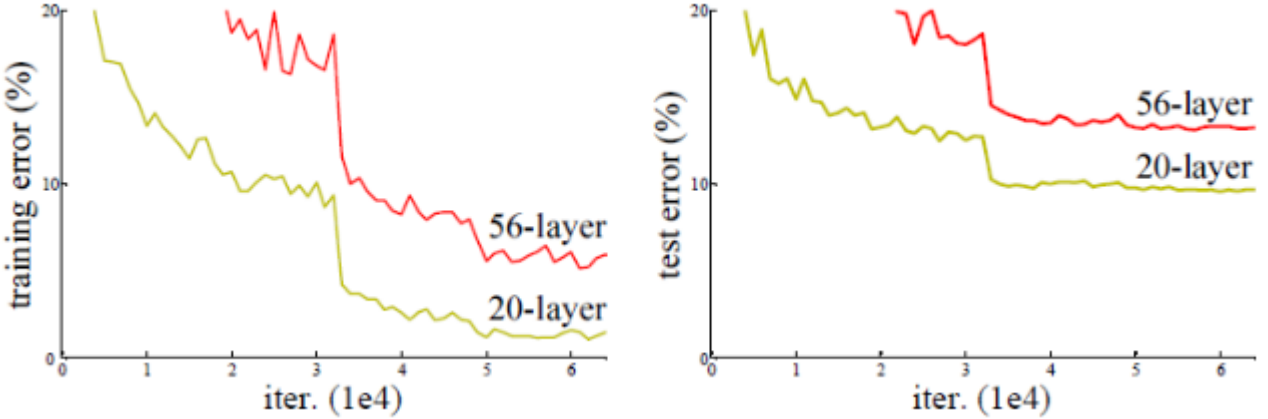
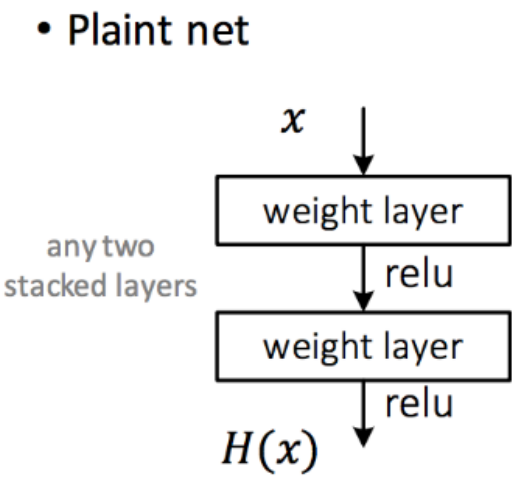
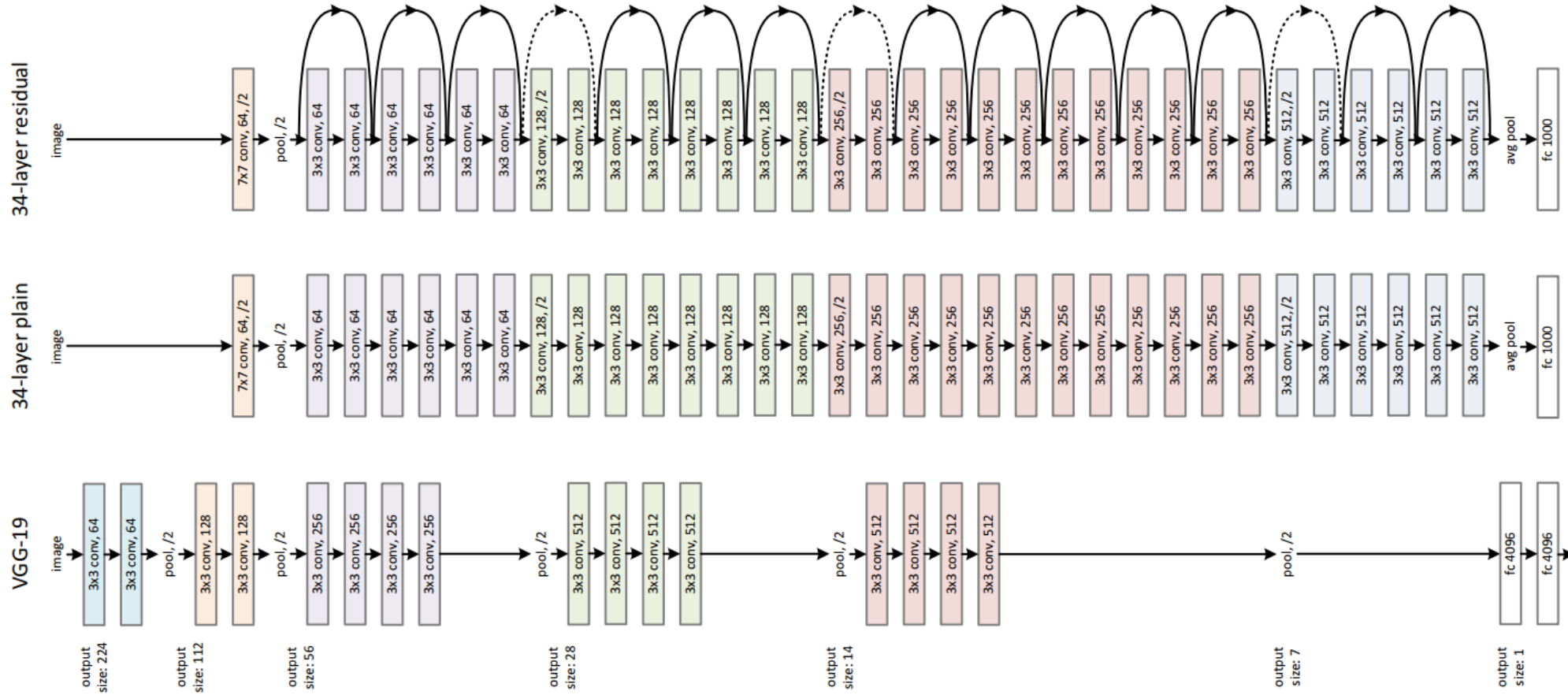


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

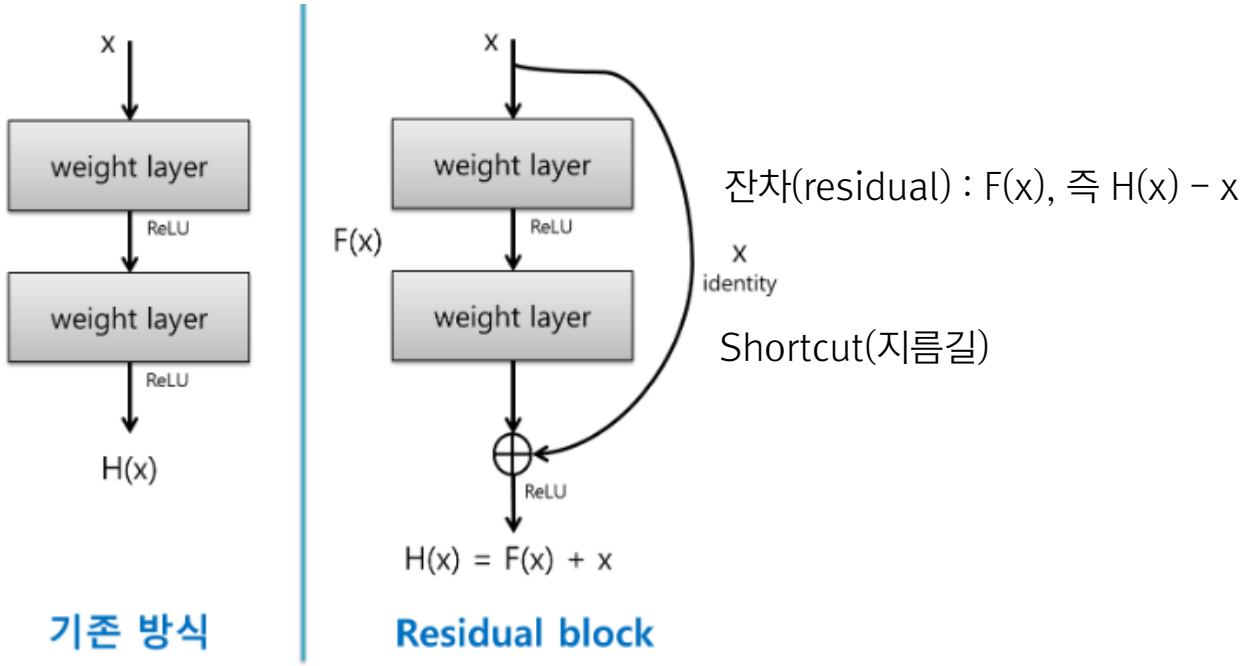


Plain Net이란 단순히 layer를 쌓은 네트워크를 의미 (ex : VGG19)

◆ 망을 깊게 하면 결과가 더 좋아질까? : Degration



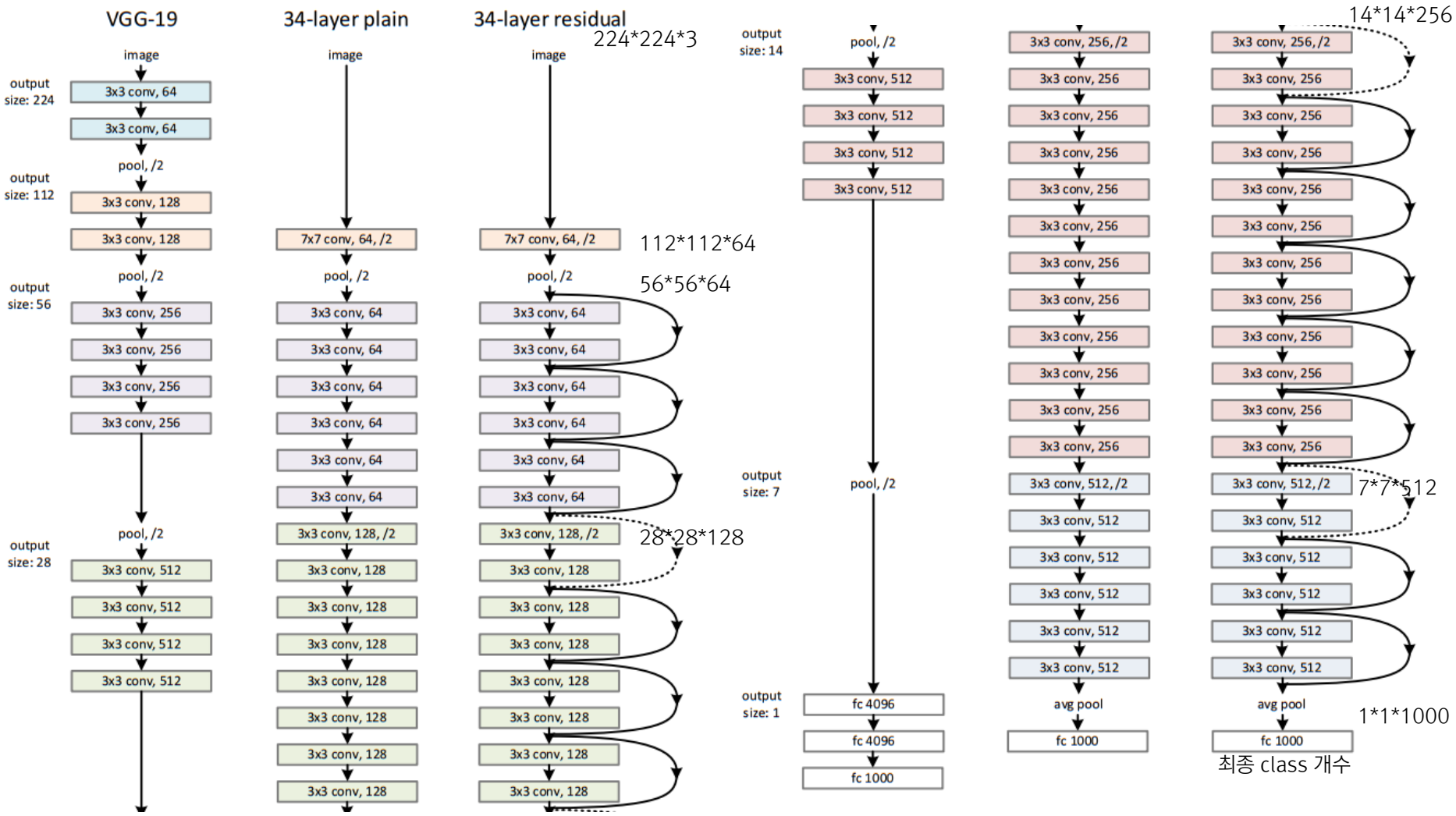
◆ 잔차를 학습하는 방법



- 기존의 신경망은 입력값 x를 타겟값 y로 매핑하는 함수 H(x)를 얻는 것이 목적
- ResNet은 **F(x) + x를 최소화하는 것**을 목적
- x는 현시점에서 변할 수 없는 값이므로 F(x)를 0에 가깝게 만드는 것이 목적
- F(x)가 0이 되면 출력과 입력이 모두 x로 같아짐
- F(x) = H(x) – x이므로 F(x)를 최소로 해준다는 것은 H(x) – x를 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*224*3)



◆ 모델 구조

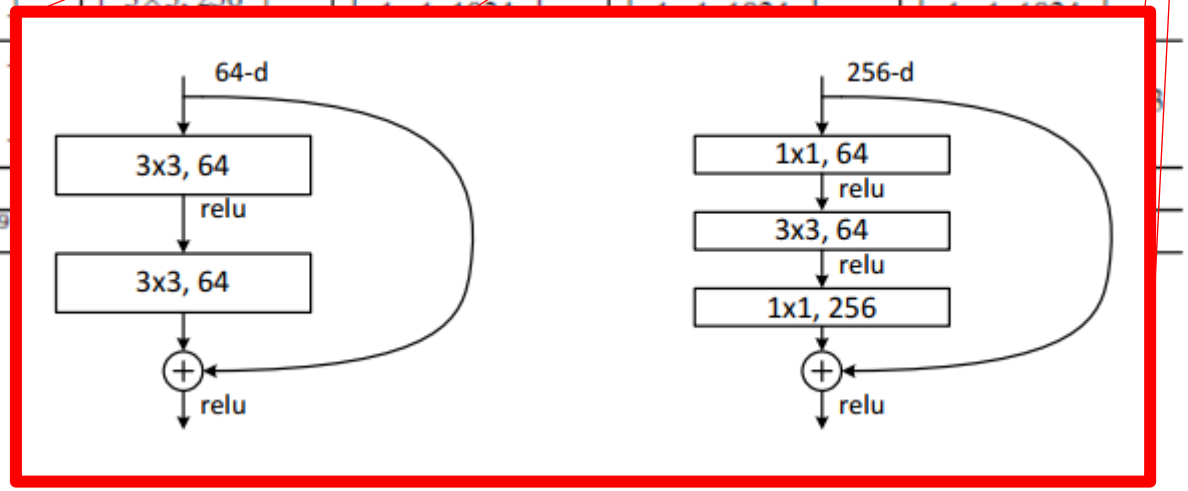
기본구조 (ImageNet-ILSVRC) (224*224*3)

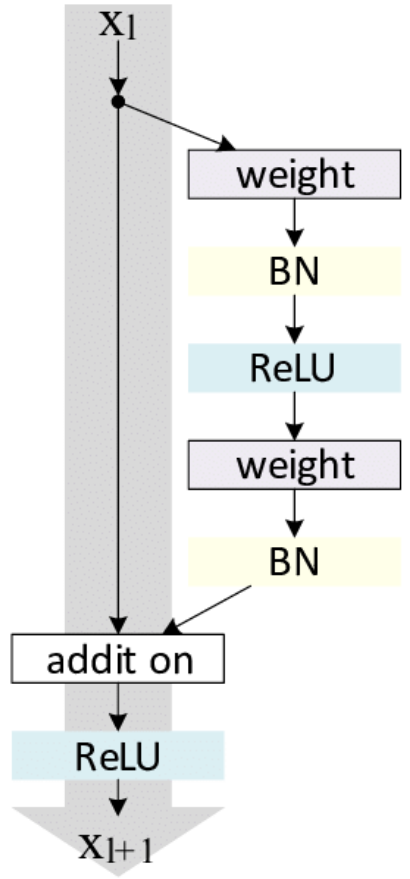
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	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				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256] ×23	[1×1, 256; 3×3, 256] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512]				
	1×1					
FLOPs		1.8×10^9				





◆ 모델 구조

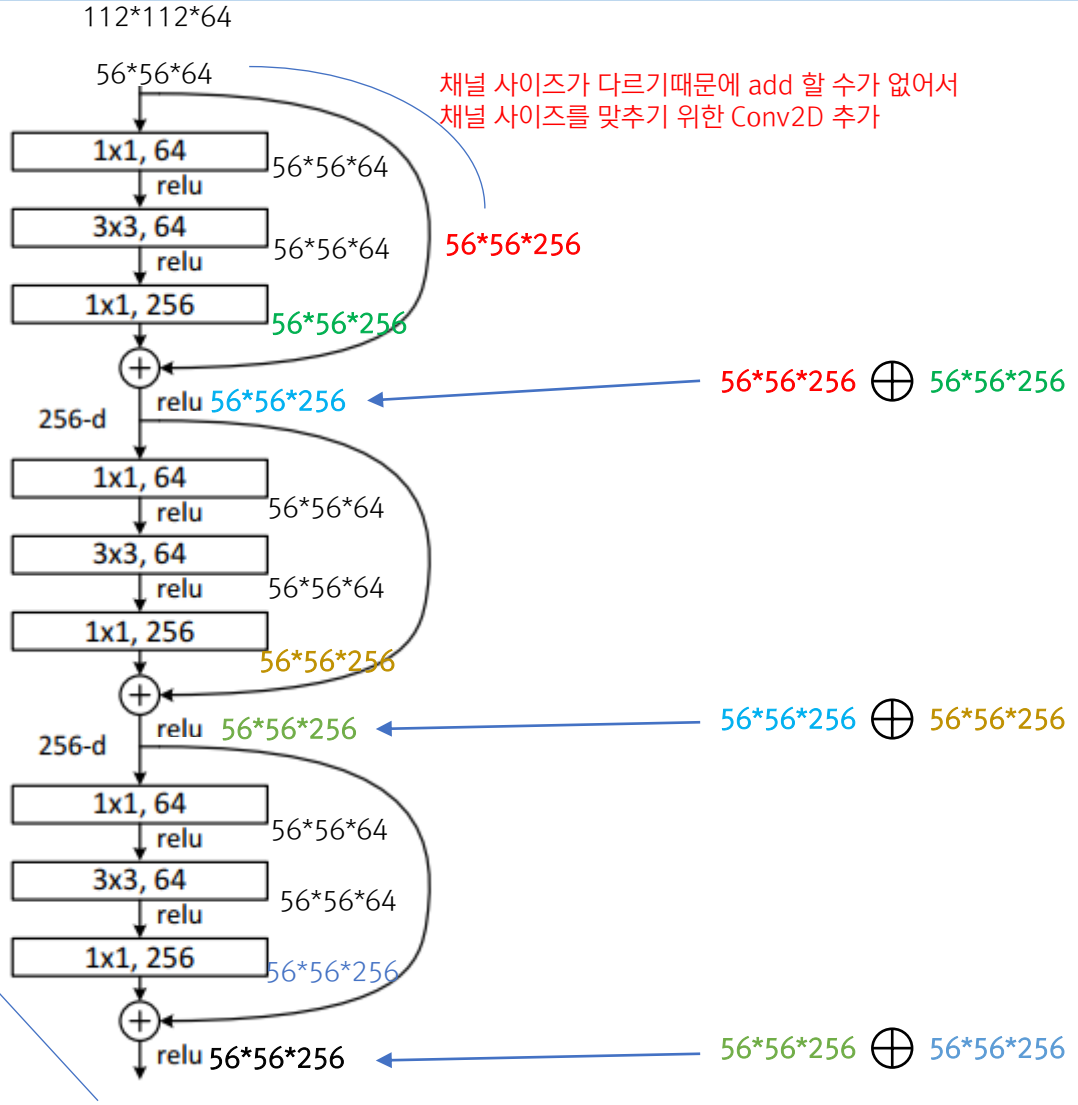
Add : 두개의 채널을 더해 줌 (채널 사이즈 같음)
Concat : 두개의 채널을 축에 따라 붙임 (채널 두배)

Bottleneck?

152-layer	
[1×1, 64; 3×3, 64; 1×1, 256]	×3
[1×1, 128; 3×3, 128; 1×1, 512]	×8
[1×1, 256; 3×3, 256; 1×1, 1024]	×36
[1×1, 512; 3×3, 512; 1×1, 2048]	×3
11.3×10^9	

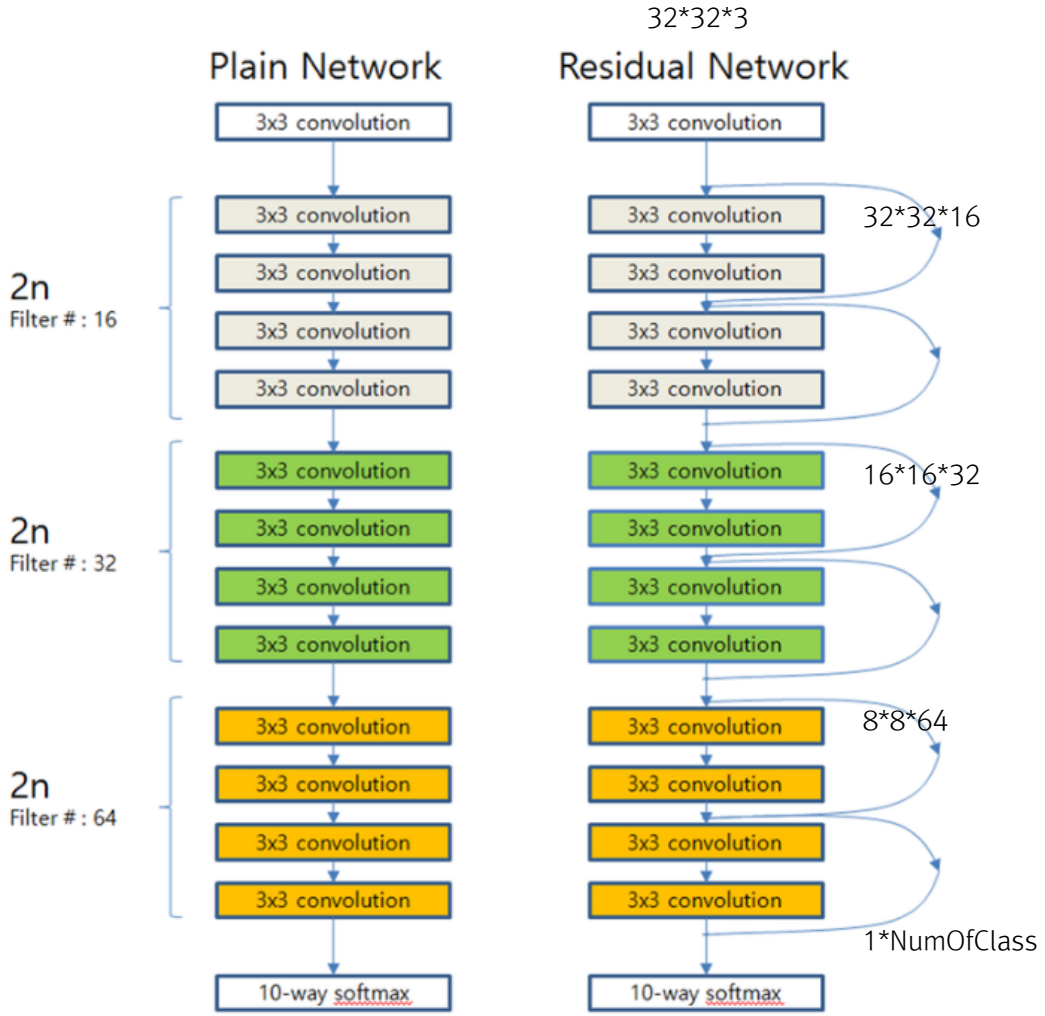
[1×1, 64; 3×3, 64; 1×1, 256] : [1×1, 64; 3×3, 64; 1×1, 256] : [1×1, 64; 3×3, 64; 1×1, 256]

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



◆ 모델 구조

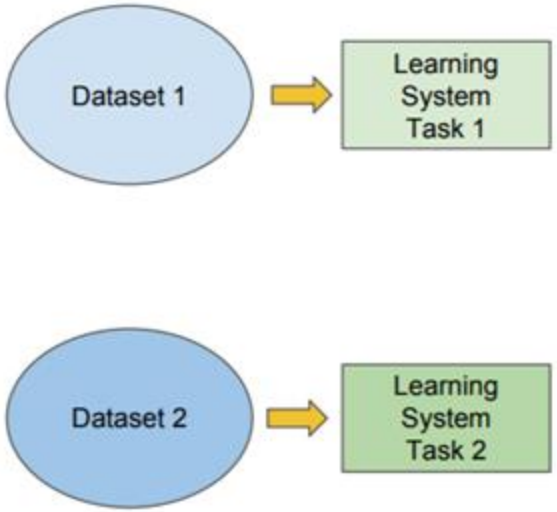
Cifar10 (32*32*3)을 위한 모델



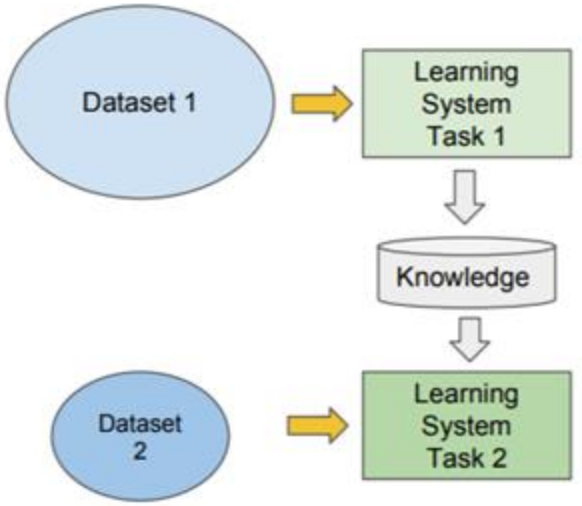
출처 :
https://blog.naver.com/PostView.nhn?blogId=laonple&logNo=220770760226

Traditional ML vs Transfer Learning

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



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



https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

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

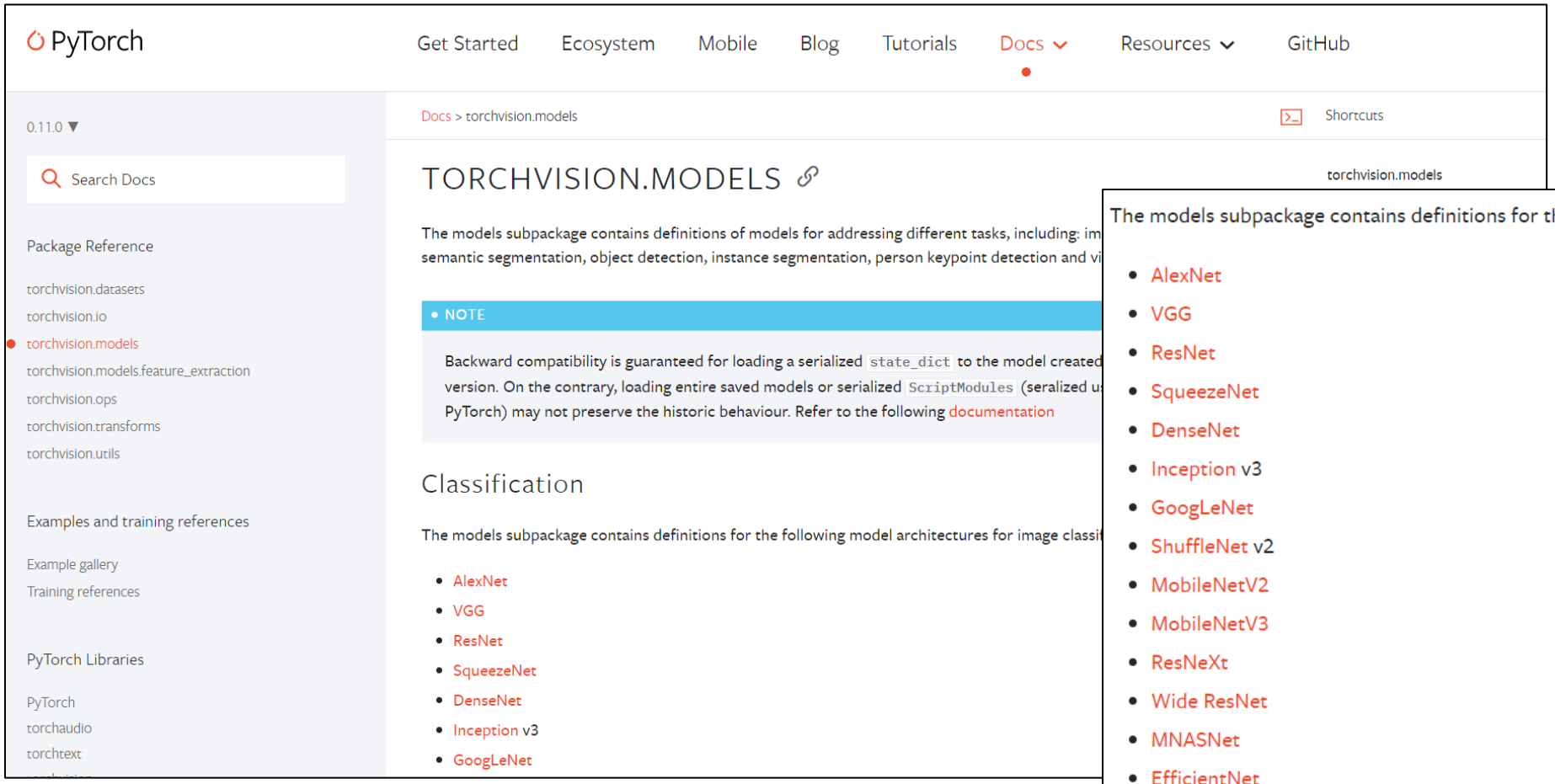
pre-trained된 모델로 시작하여 prediction을 도출하는 마지막 레이어의 weight만 업데이트

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

오직 마지막 레이어만 바꾸기 때문

https://better-tomorrow.tistory.com/entry/TorchVision-model-funetuning

Backbone Model

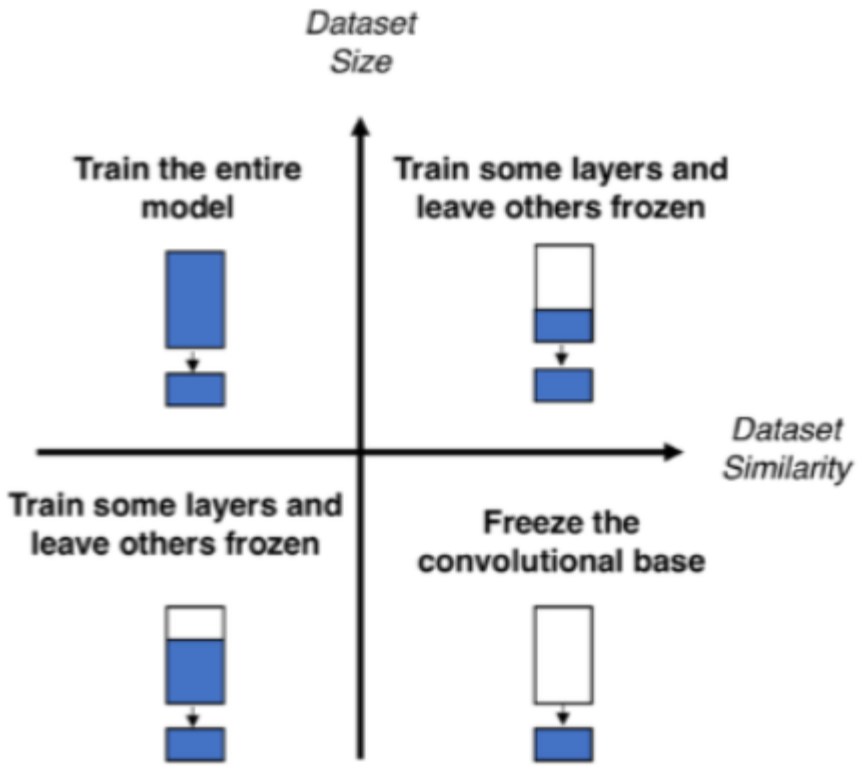


The models subpackage contains definitions for the following model architectures for image classification:

- 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

전이학습의 학습 방법 고민하기



https://velog.io/@garam/Fine-tuning-vs-Transfer-learning-vs-Backbone

1. Large & 다른 Dataset

Overfitting이 발생하지 않을정도로 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 시간절약을위해 Conv. layer의 후반 계층과 FC layer만 학습을 진행해줍니다. 학습을 하지 않는 부분은 lr=0, 학습을 하는 부분은 lr = 기존의 1/10 으로 지정해줍니다.

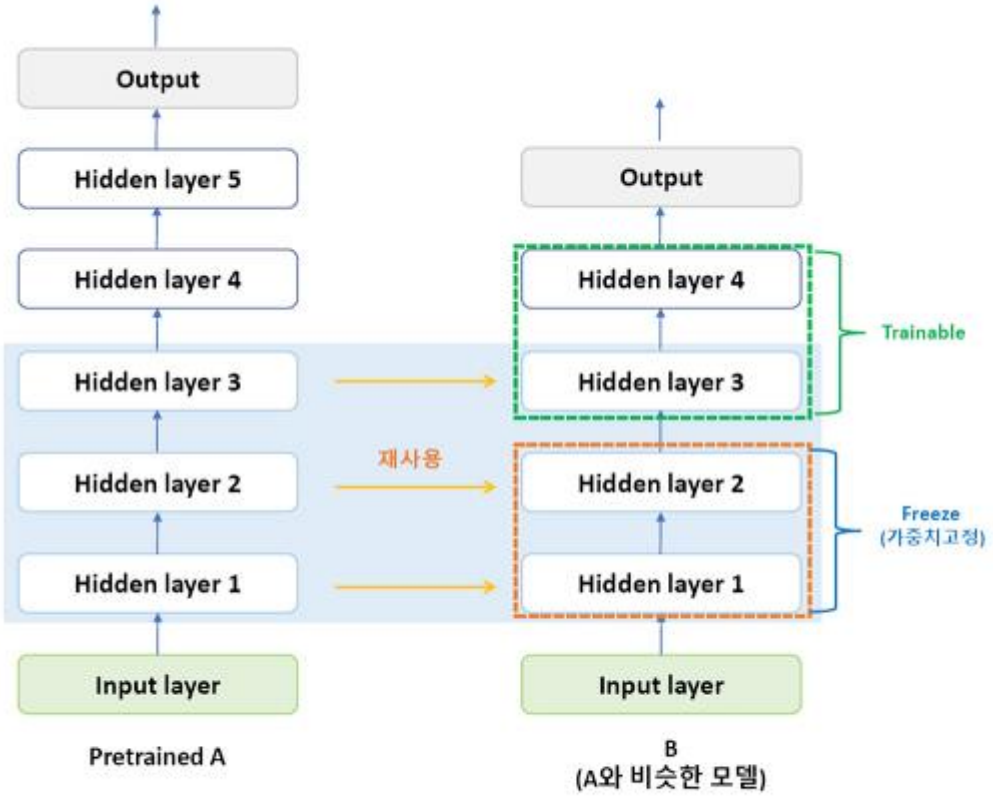
3. Small & 다른 Dataset

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.

레이어 가중치를 고정하기



[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
```

감 사 합 니 다