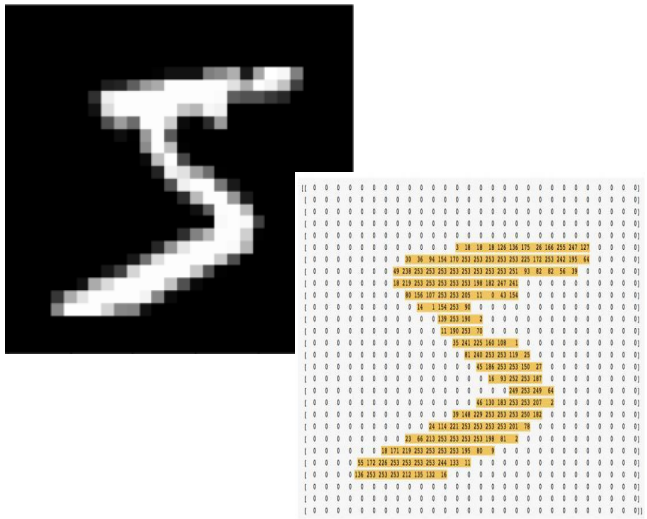
딥러닝 AI 기초

Hyungmin Jun, Ph. D.

Multiphysics Systems Design Laboratory
Department of Mechanical System Engineering
Jeonbuk National University

Image Data – Grayscale vs RGB



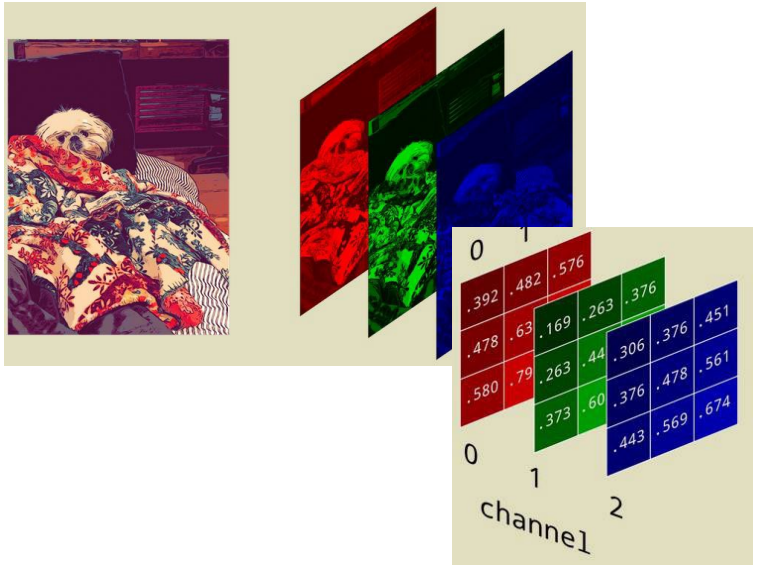


Image Data – 정형데이터 vs. Grayscale 이미지 vs. RGB 이미지

정형데이터

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0
5	0	3	Allen, Mr. William Henry	male	35.0	0	0

2차원 데이터

Grayscale 이미지

2차원 데이터

RGB 이미지

3차원 데이터

https://www.kaggle.com/code/subinium/subinium-tutorial-titanic-beginner/notebook
https://www.flipkart.com/pintura-canvas-painting-uv-print-digital-lion-wooden-frame-art-best-gifting-exclusive-designs-high-quality-12-inch-x-18/p/itm77568d659fbfb

Image Data – 정형데이터 vs. Grayscale 이미지 vs. RGB 이미지

정형데이터

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0
5	0	3	Allen, Mr. William Henry	male	35.0	0	0

2차원 데이터

Grayscale 이미지

3차원 데이터

RGB 이미지

4차원 데이터

https://www.kaggle.com/code/subinium/subinium-tutorial-titanic-beginner/notebook
https://www.flipkart.com/pintura-canvas-painting-uv-print-digital-lion-wooden-frame-art-best-gifting-exclusive-designs-high-quality-12-inch-x-18/p/itm77568d659fbfb1.2

Dataset – AI Hub

- 2020년 “디지털 뉴딜 사업”에 의해 인공지능 학습용 데이터를 범국가적으로 모이는 곳
- 국내 기업, 연구소, 개인 등이 자체적으로 확보하기 어려운 학습용 데이터를 공개

Dataset – PASCAL VOC, MS COCO, Google Open Images

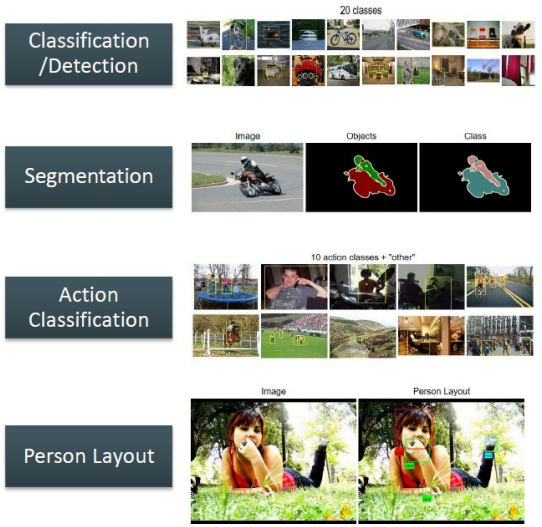
많은 딥러닝 Object Detection 패키지는 위의 데이터 셋을 가지고 학습된 형태로 배포

- MINIST
- FashionMINIST
- BCCD
- KITTY
- PASCAL VOC
- MS COCO
- Google Open Images
- Etc..

Dataset – PASCAL Visual Objects Classes

Visual Object Classes Challenge 2012 (VOC2012)

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

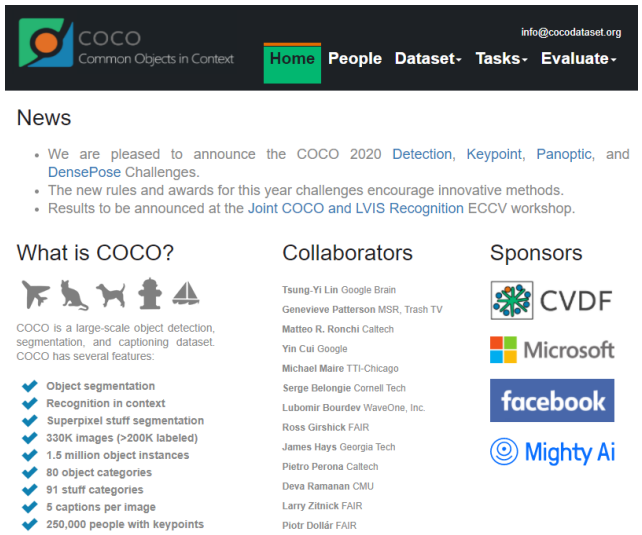


- PASCAL VOC Challenge(2005 ~ 2012)에서 쓰이던 데이터셋
- PASCAL VOC 2007과 2012 데이터셋이 벤치마크 데이터셋으로 자주 활용
- AP(Average Precision)을 기반으로 평가방식 채택

PASCAL VOC – PASCAL Visual Objects Classes

- 이미지 개수: 11k, 이미지당 평균 오브젝트 수: 2.4개
- IoU Threshold: 0.5
- 현재는 벤치마크용으로만 사용
- Annotation 파일: XML 포멧
- http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

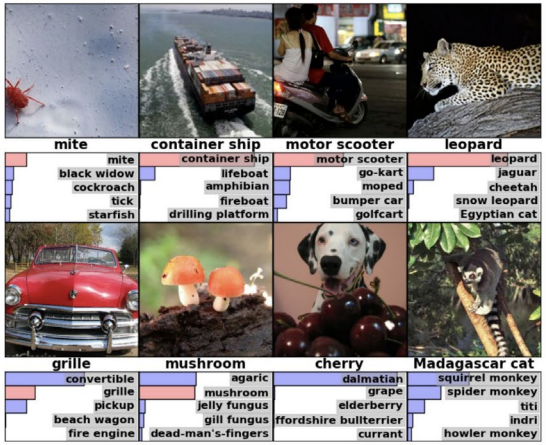
MS COCO – MS COCO (Microsoft's Common Objects in Context)



ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

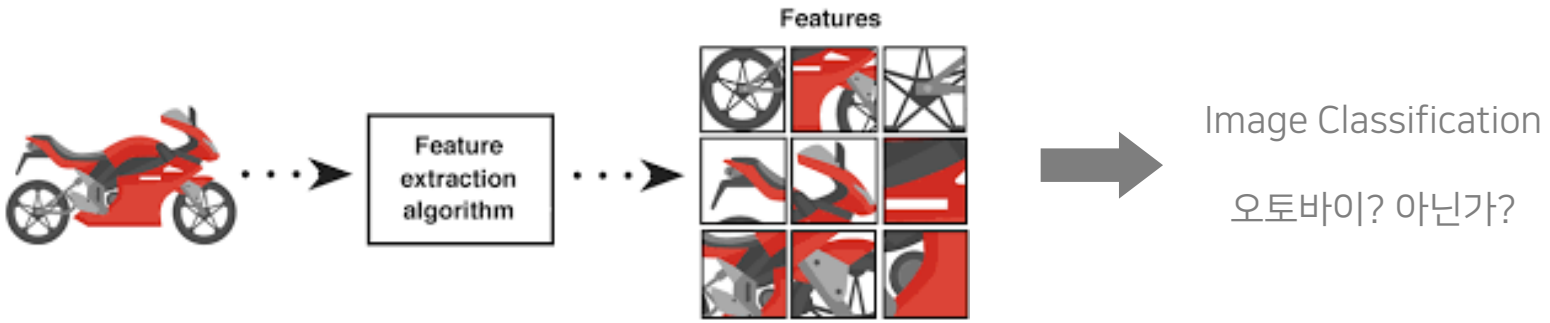


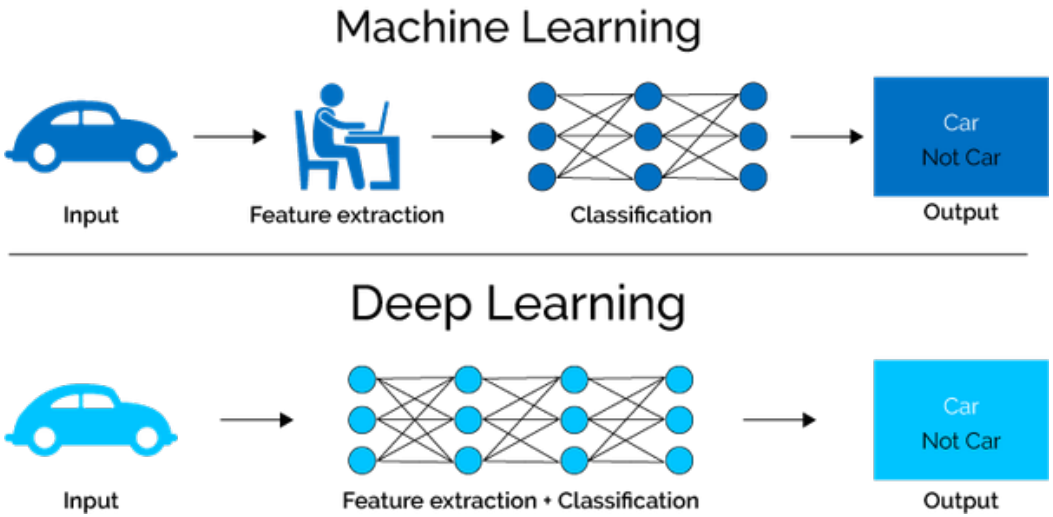
- 2014년 ImageNet, PASCAL VOC 데이터셋의 문제점을 해결하기 위해 제안
  - ✓ 이미지 안에 Object가 크고, 대부분 중앙에 위치
  - ✓ 이미지당 Object 수가 적음
- MS COCO는 AMT(Amazon Mechanical Turk)라는 웹 기반 데이터 클라우드 소싱 방식을 이용해 저렴한 비용으로 데이터셋을 제작
- AMT를 통해 고용된 노동자들이 데이터셋을 레이블 ➔ 전문가 평가

이미지 분류

(Image Classification)

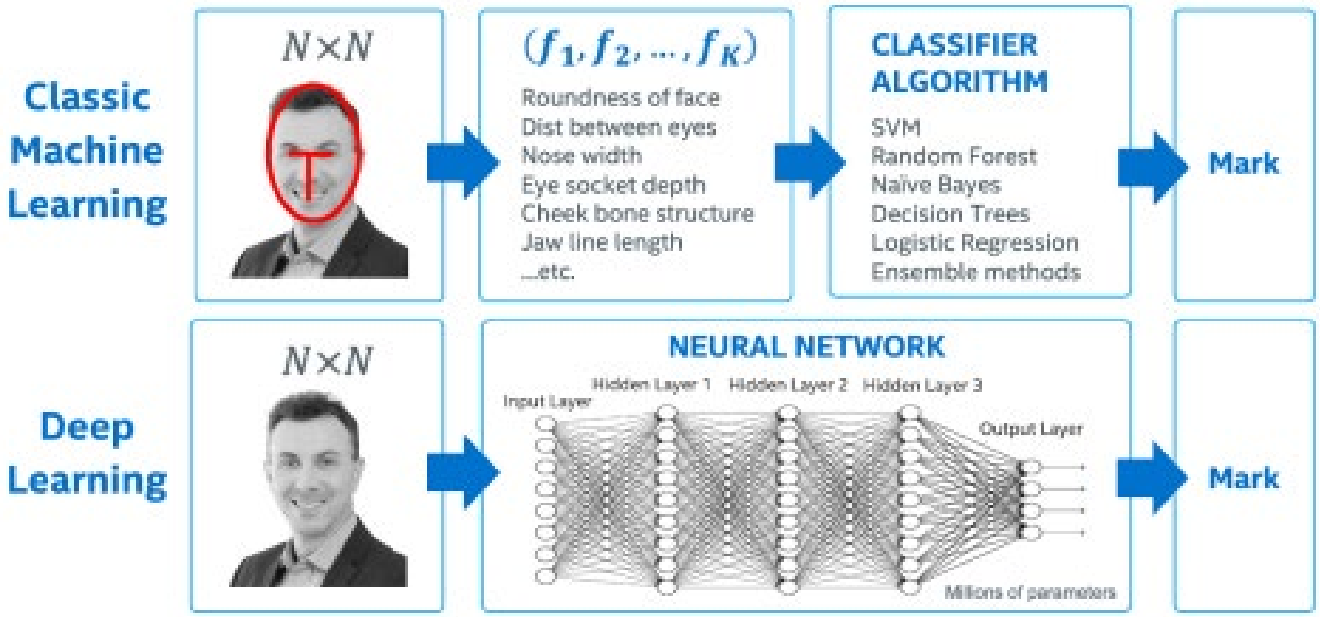
Image Classification – Classical ML vs. Deep Learning





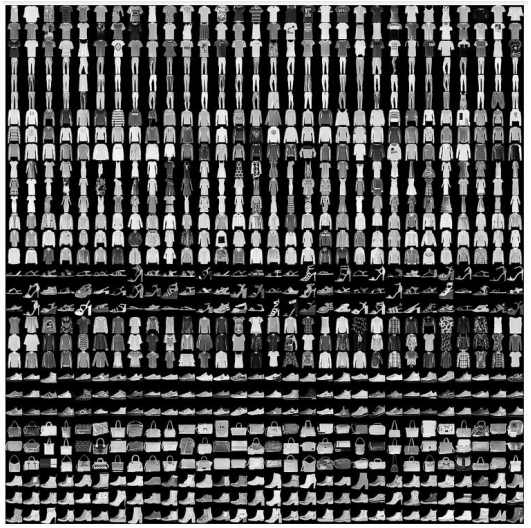
https://freecontent.manning.com/the-computer-vision-pipeline-part-4-feature-extraction/
https://levity.ai/blog/difference-machine-learning-deep-learning

Image Classification – Classical ML vs. Deep Learning

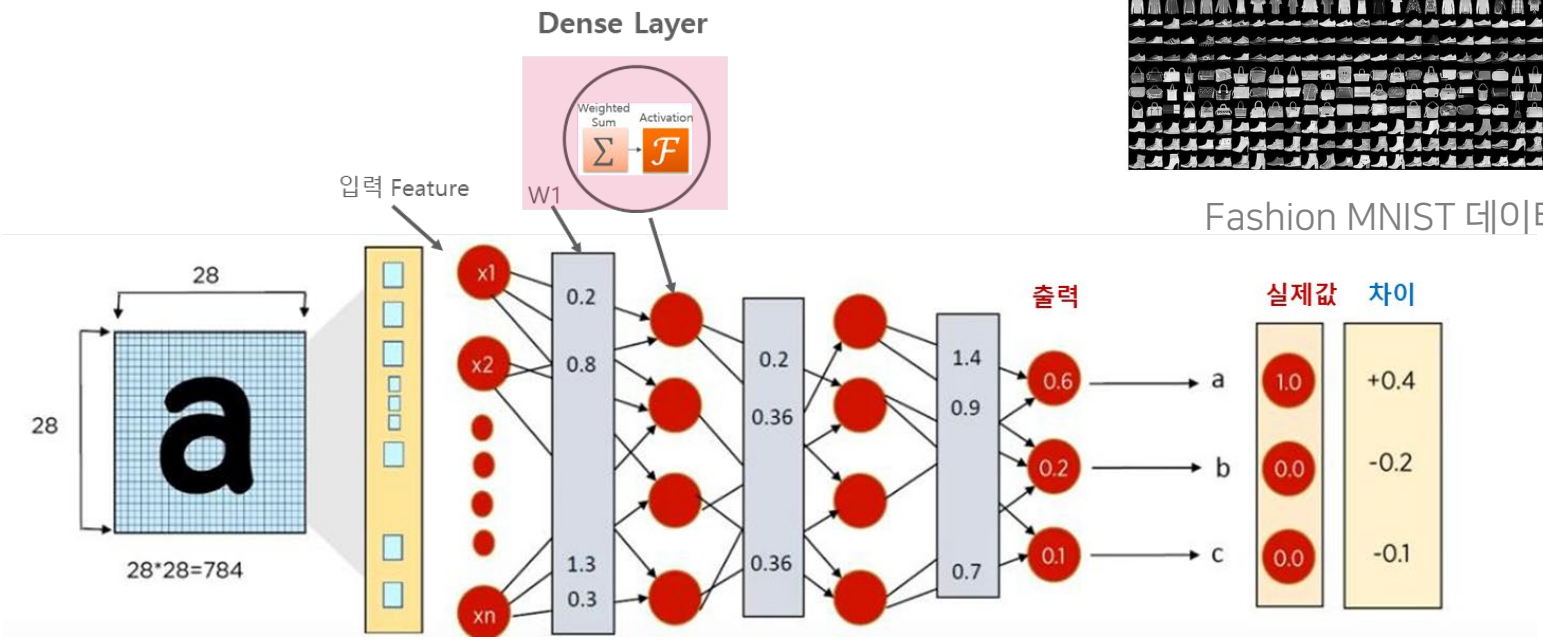


https://post.naver.com/viewer/postView.naver?volumeNo=32833704&memberNo=52806467

Image Classification – Dense Layer를 활용한 이미지 분류



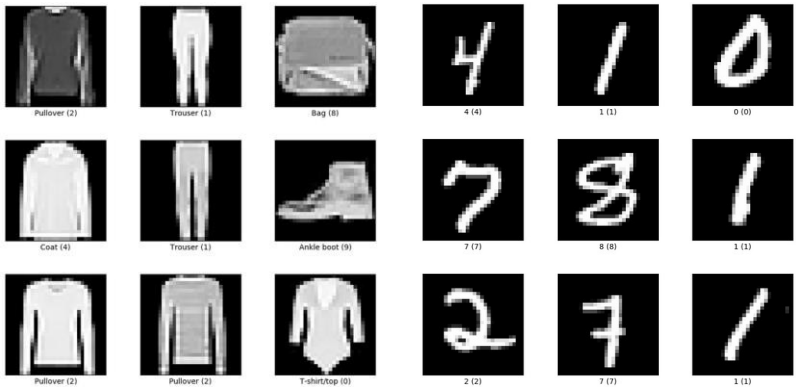
Fashion MNIST 데이터



Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).
https://gayathri-siva.medium.com/deep-learning-f52ae2ddef2

Image Classification – Dense Layer issue #1

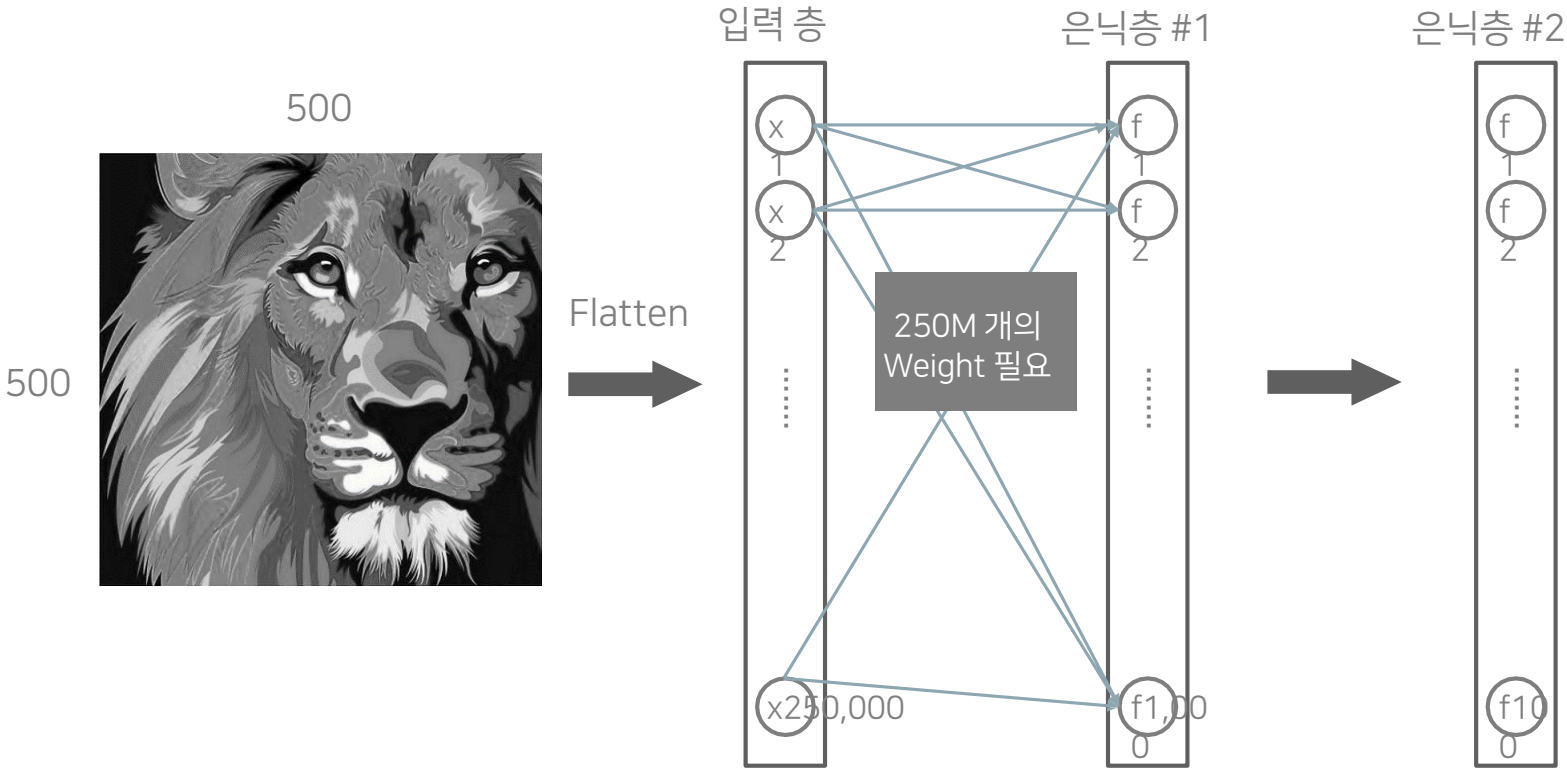
분류 대상이 이미지 중앙에 위치



분류 대상이 이미지 중앙에 위치하지 않음

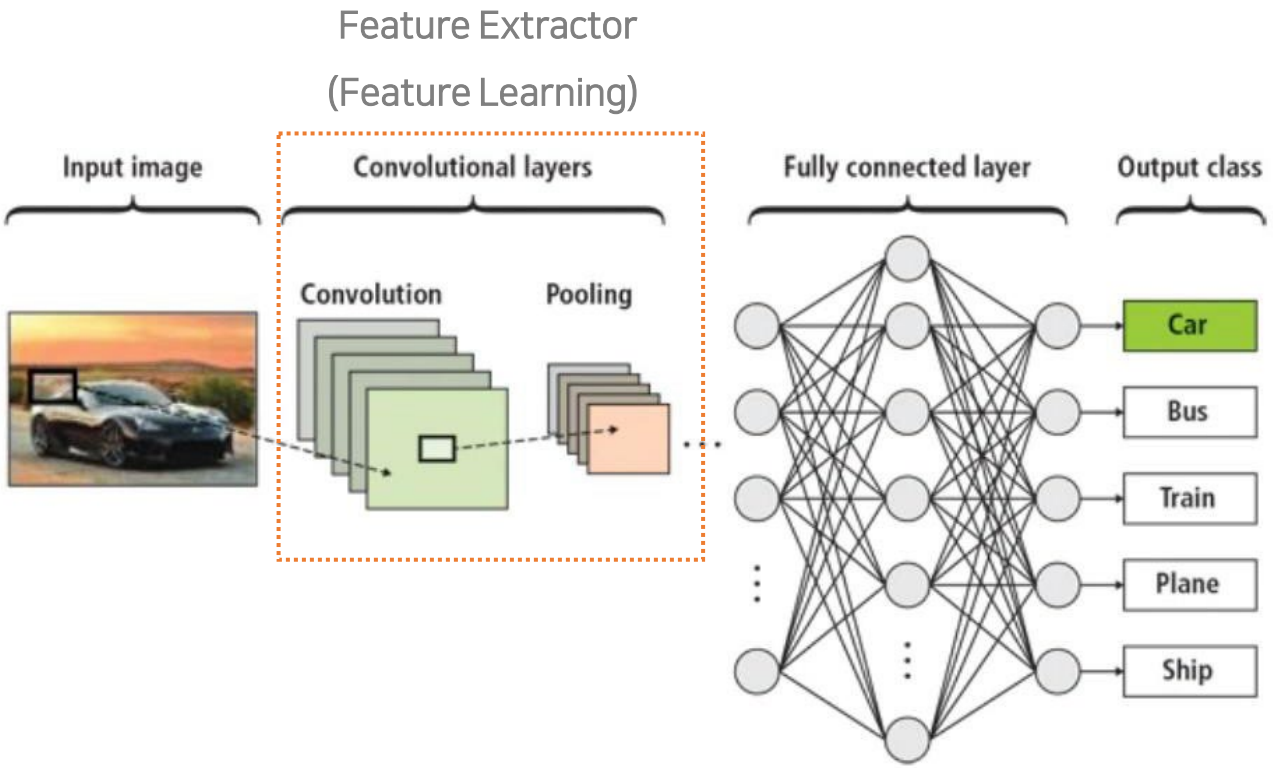
https://www.tensorflow.org/datasets/api_docs/python/tfds/visualization/show_examples
https://hanzhaoml.github.io/courses/cs598-haz/lectures/presentation_11_05_3.pdf
https://github.com/RaghavPrabhu/Deep-Learning/tree/master/dogs_breed_classification

Image Classification – Dense Layer Issue; Image Size



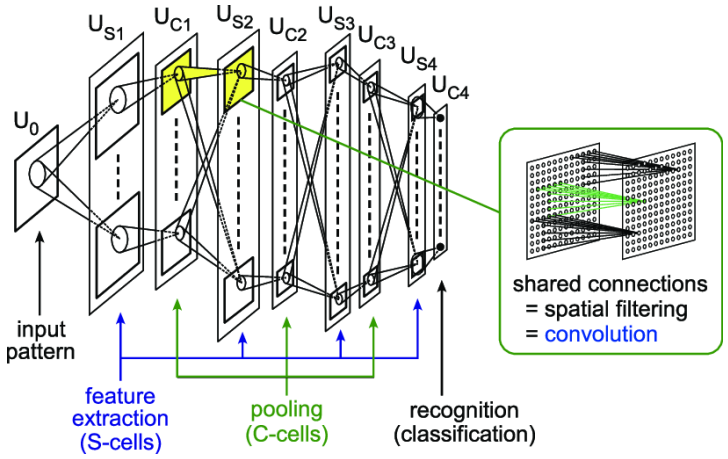
➔ 이미지의크기가커질수록너무많은Weight가필요

Convolutional Neural Network – for Image Classification



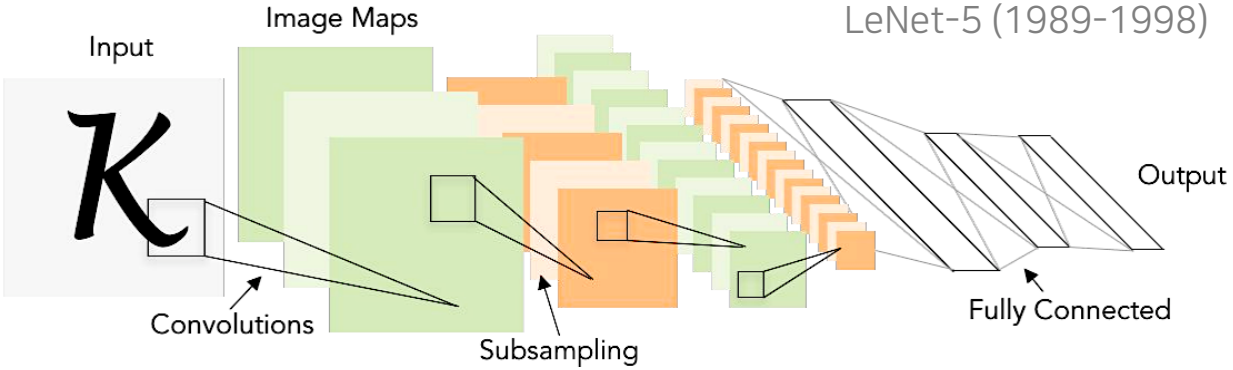
https://www.vision-systems.com/home/article/16736100/deep-learning-brings-a-new-dimension-to-machine-vision

Convolutional Neural Network – Neocognitron(1980), LeNet-5(1989-1998)



Neocognitron (1980)

Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." Competition and cooperation in neural nets. Springer, Berlin, Heidelberg, 1982. 267-285.

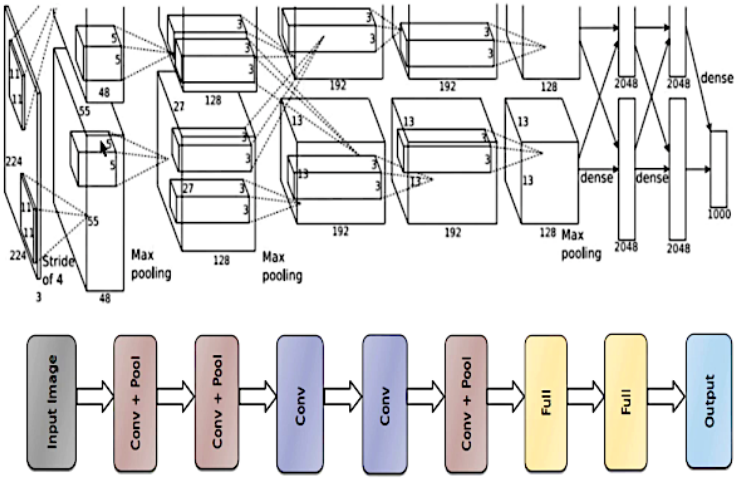


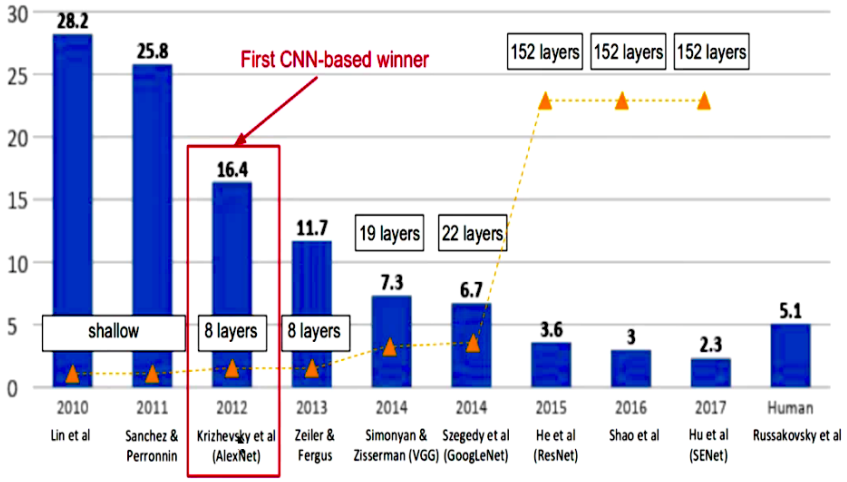
LeNet-5 (1989-1998)

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

Convolutional Neural Network – AlexNet(2012)

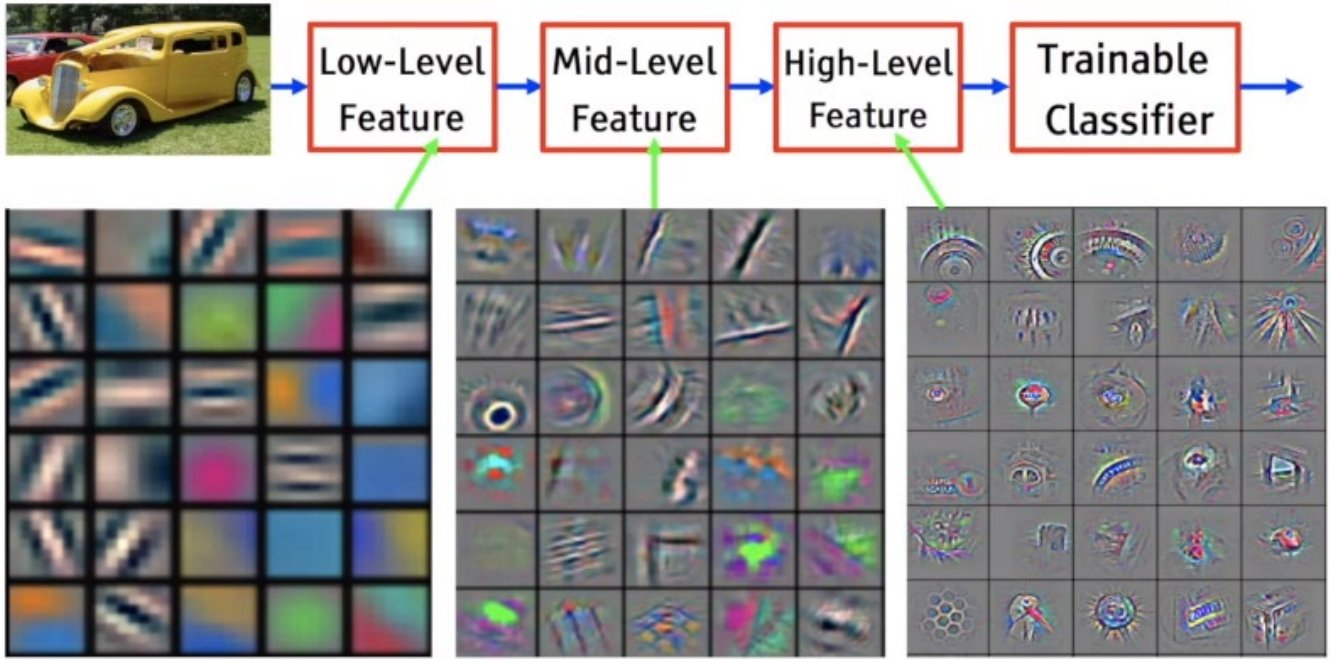
AlexNet (2012)





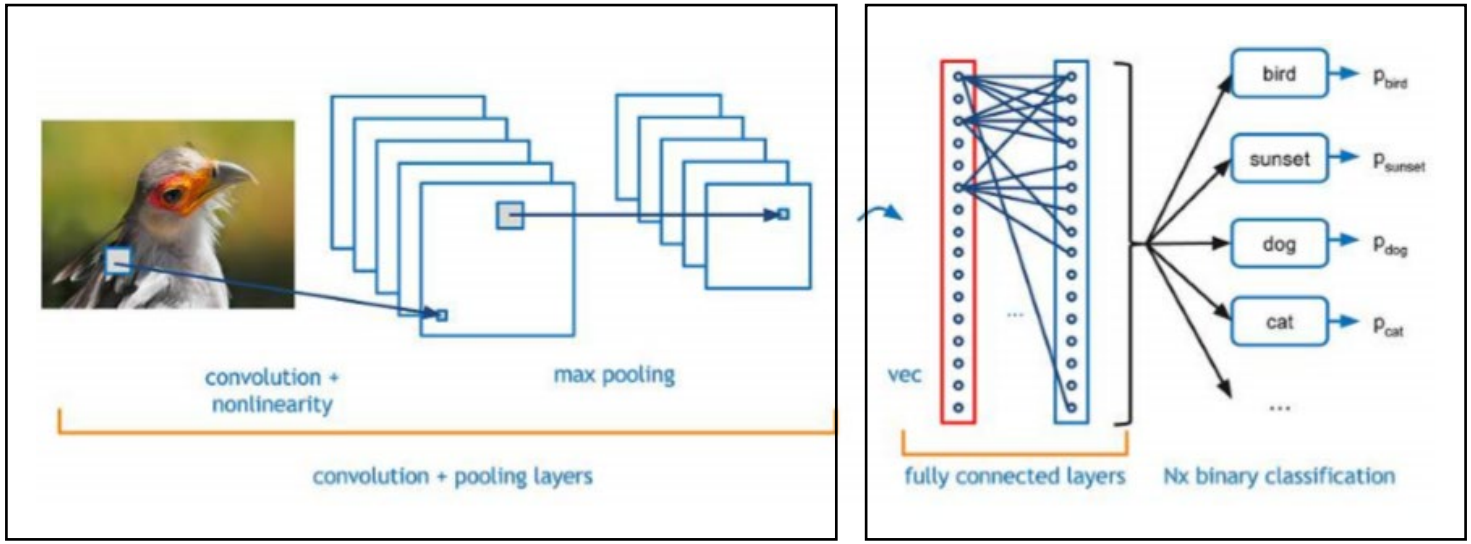
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

Convolutional Neural Network – CNN Layer 단계 Feature 학습



https://www.cs.cmu.edu/~epxing/Class/10708-17/slides/lecture20-DL.pdf

Convolutional Neural Network(CNN) – 전체 구조



Feature Extractor
(CNN + Activation + Polling 등)

Classifier

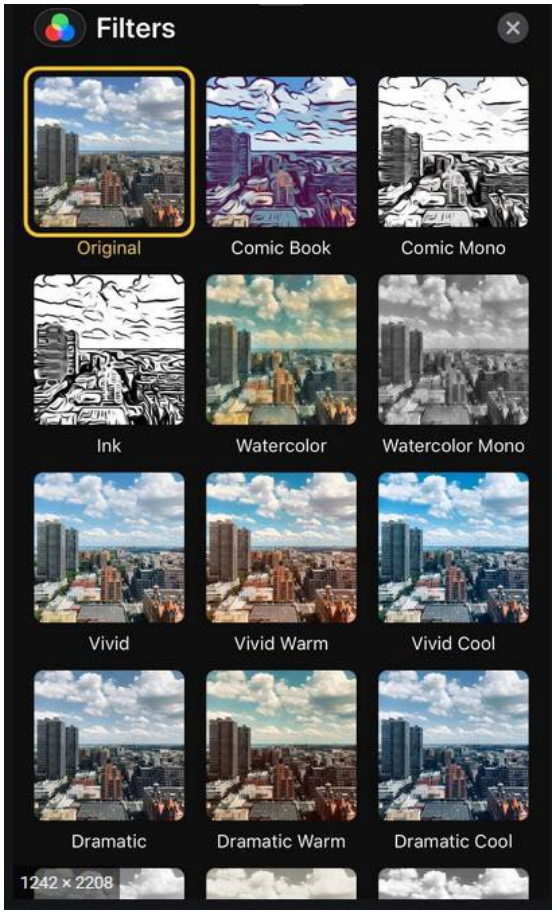
- Classification에 맞는 **최적의 Feature** 추출
- 최적의 Feature 추출을 위한 **최적 Weight값** 계산
- 최적 Feature 추출을 위한 **필터(필터 Weight) 값** 계산

https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

컨볼루션

(Convolution)

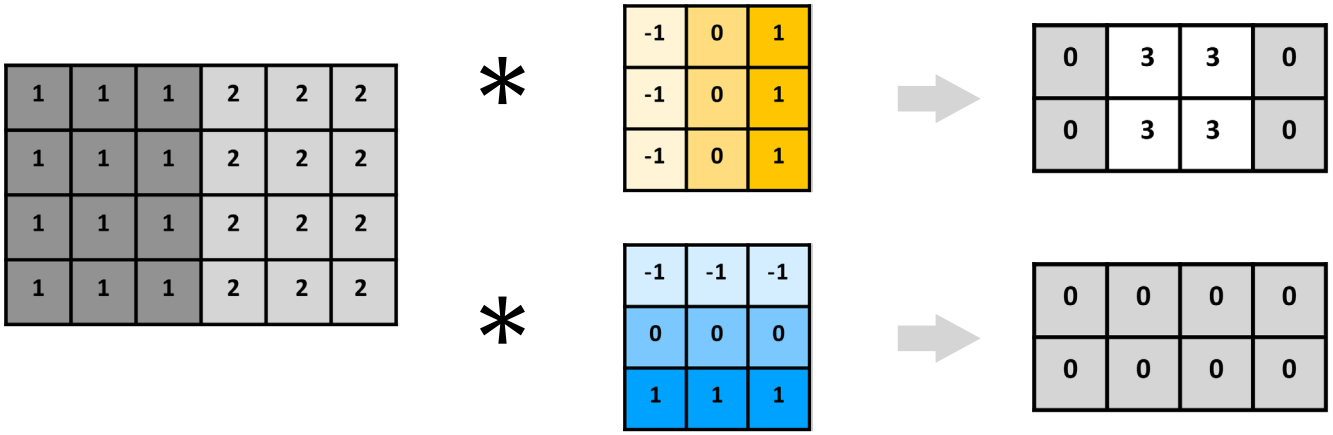
Convolution – Image Filtering

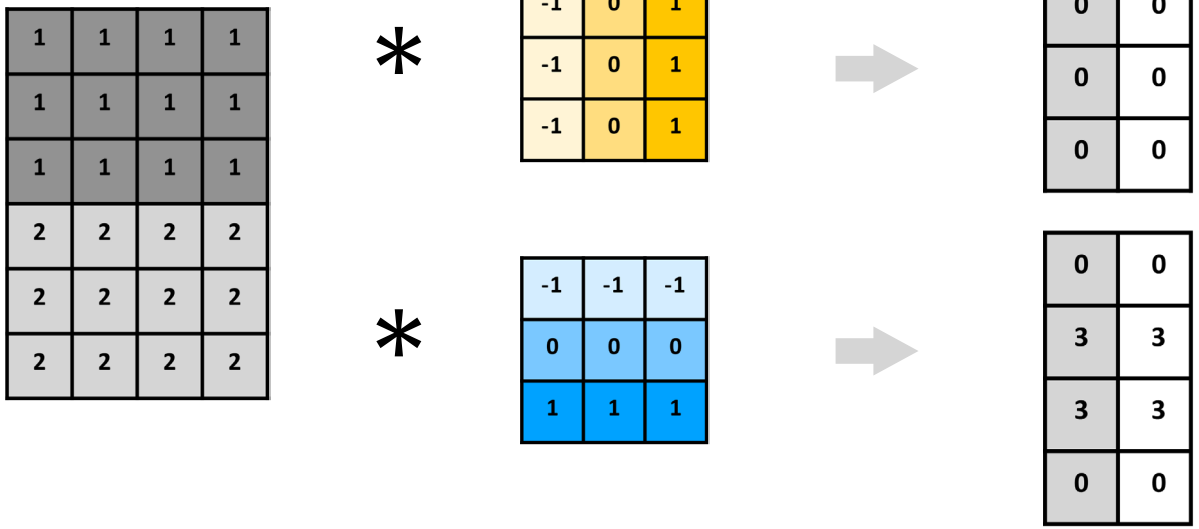


간단한 메트릭스 연산을 적용하여 이미지를 이루고 있는

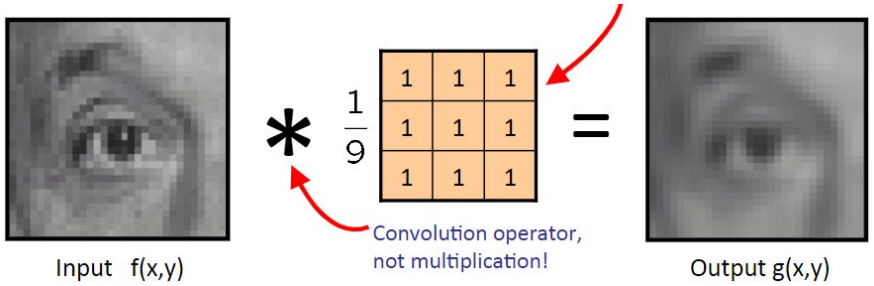
픽셀의 배열을 변경 ➔ 이미지를 변형

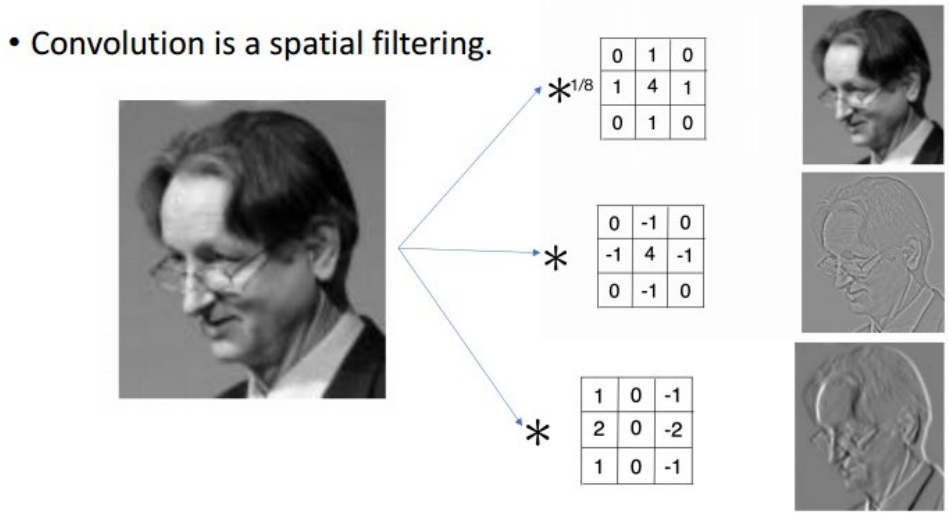
Convolution – Image Filtering





Convolution – Image Filtering

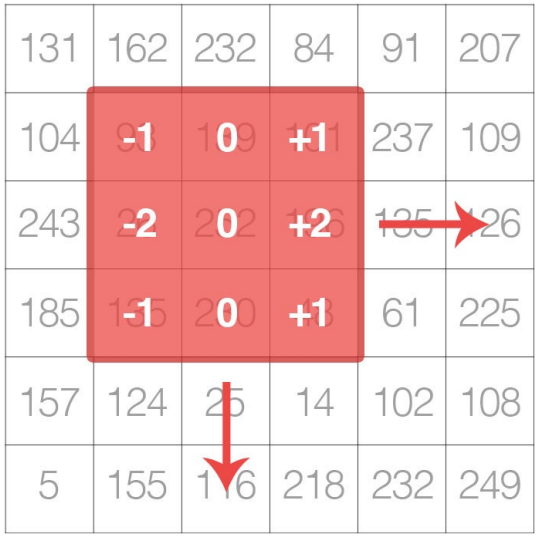


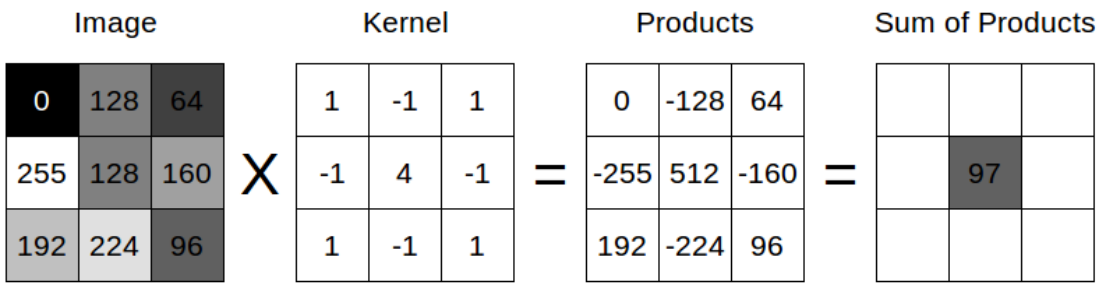


정방 행렬(Filter)을 원본 이미지에 순차적으로 슬라이딩 (Convolution 연산) ➔ 새로운 픽셀값

https://medium.com/@boelsmaxence/introduction-to-image-processing-filters-179607f9824a

Convolution – Operation

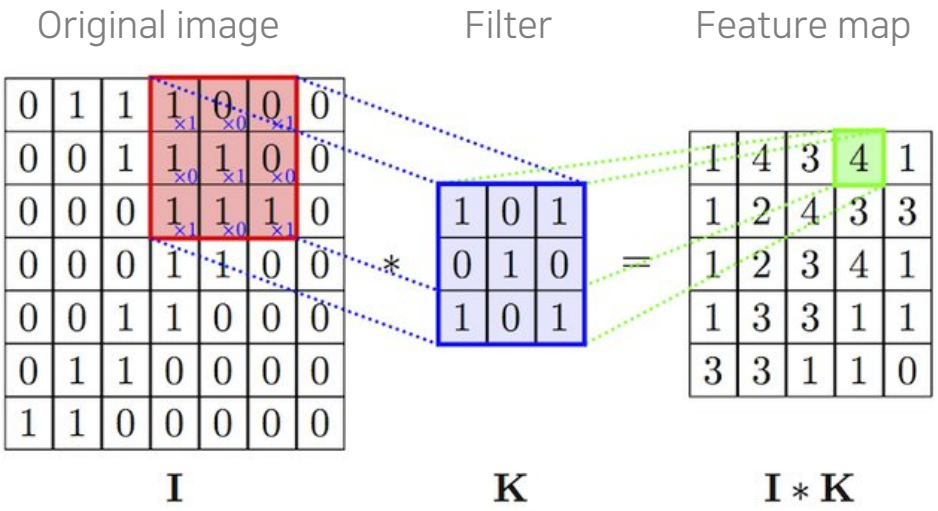


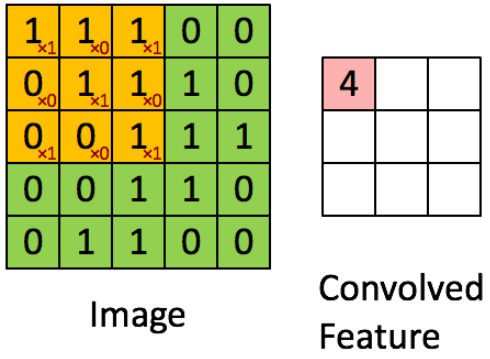


❖ 이미지의 상단 좌측부터 Sliding 하면서 Convolution 연산을 순차적으로 수행

https://pyimagesearch.com/2021/05/14/convolution-and-cross-correlation-in-neural-networks/

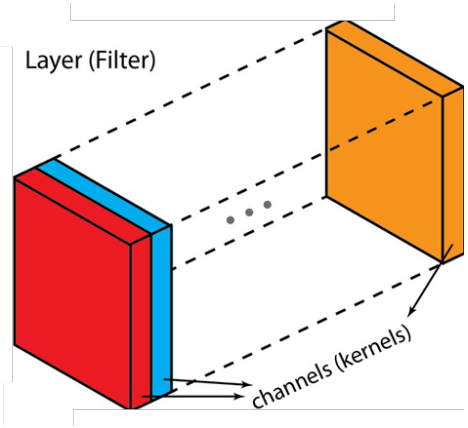
Convolution – Operation



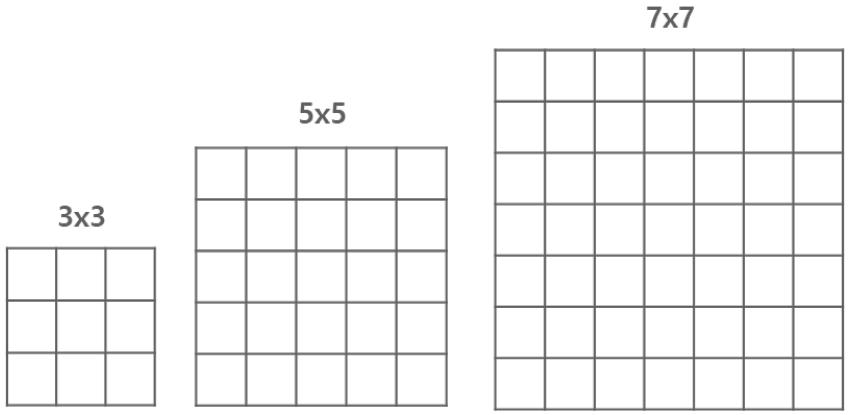


https://laptrinhx.com/implementing-kernel-filter-convolutional-in-your-own-1511826395/
https://developer.nvidia.com/discover/convolution

Convolution – Kernel vs. Filter, Kernel Size



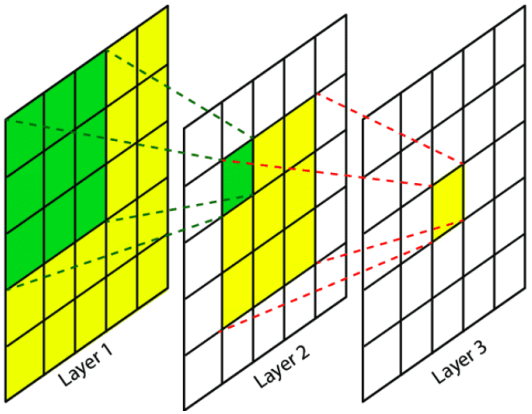
- 필터(Filter)는 여러 개의 커널(Kernel)로 구성
- 개별 커널은 필터내에서 서로 다른 값을 가질 수 있음



- Kernel 크기는 보통 같은 크기를 가짐
- Kernel 크기가 커지면 Feature Map은 많은 정보를 가짐
- 큰 Kernel은 많은 계산(많은 파라미터) 요구

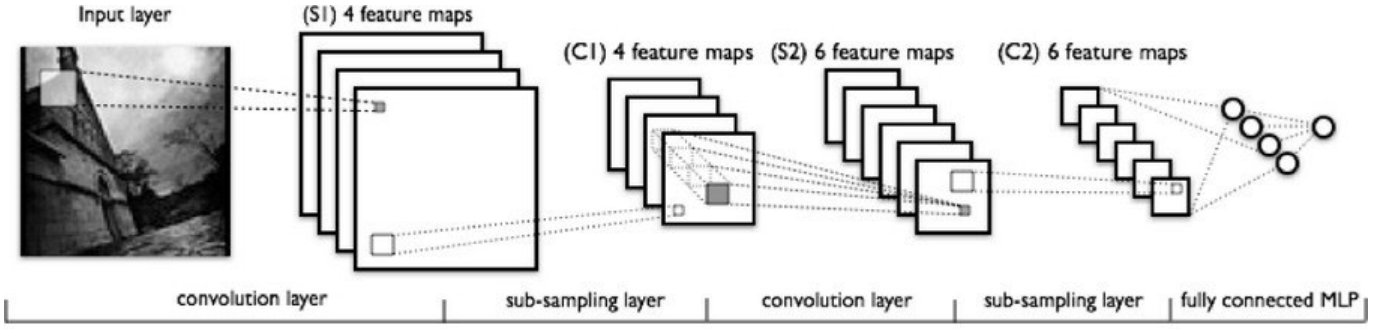
https://theaisummer.com/medical-image-deep-learning/1.2

Convolution – Kernel Size와 Receptive Field

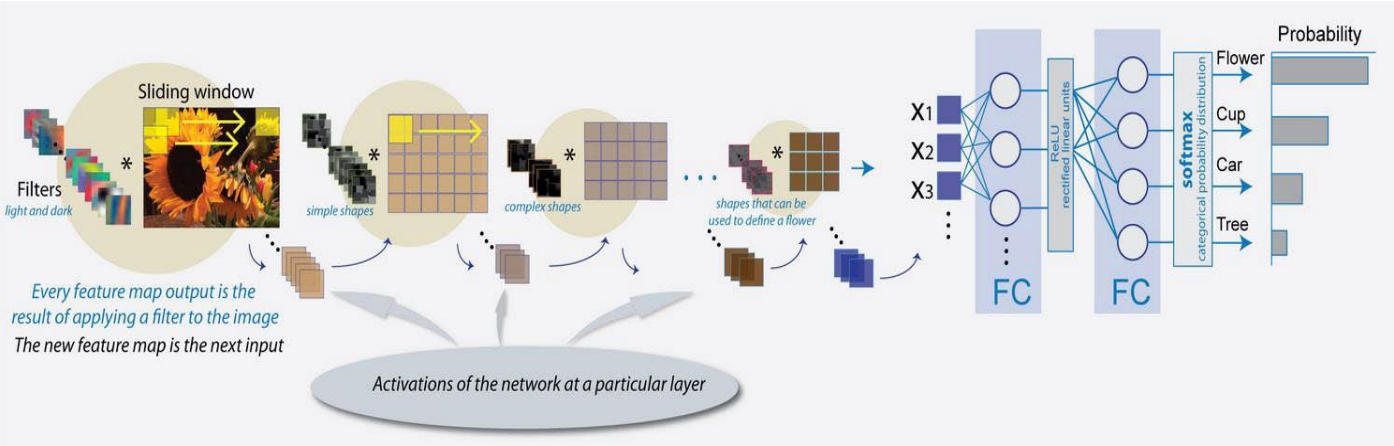


Receptive Field

➔ 입력(Image 또는 Feature map)에서 Feature를 만드는 영역의 기본 크기



Convolution – CNN에서 Filter 값이란?



Computer Vision에서는 사용자 목적에 맞게 특정 필터를 선택하여 이미지에 적용 ➔ 이미지 변환

Deep Learning CNN에서의 필터란?

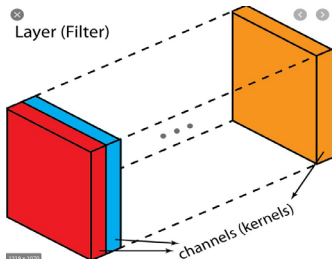
- ❖사용자가 특정 필터를 선택하지 않음
- ❖**Deep Learning Network는 목적에 맞는 Filter 값을 스스로 학습하여 최적 값을 찾음**

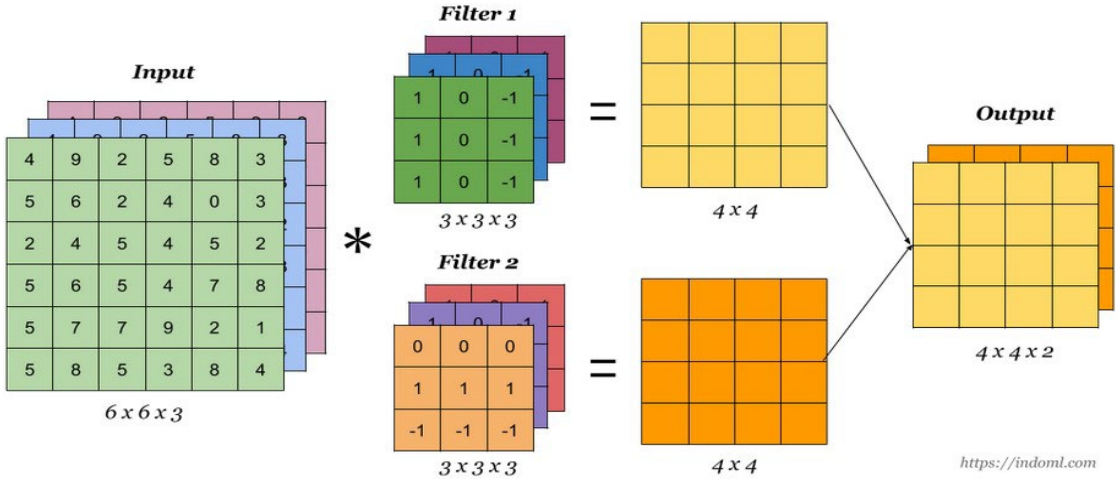
https://kr.mathworks.com/help/deeplearning/ug/introduction-to-convolutional-neural-networks.html

Convolution – Multi-filters

입력 이미지는 3차원 피처맵도 3차원

단일 필터도 3차원



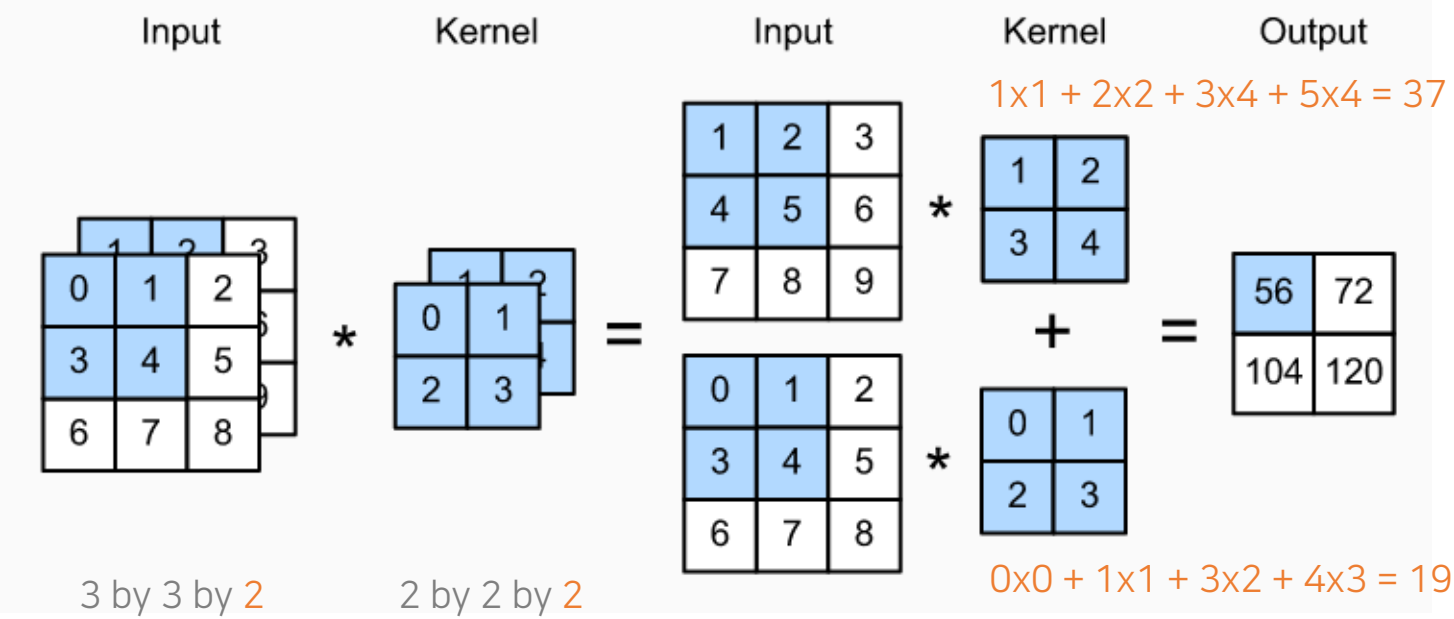


- Filter는 3차원
- CNN에서는 3차원 Filter 여러 개를 개별 Feature Map에 적용

Filter 채널 수와 입력과 출력 Feature map의 채널 수 관계

- Conv 연산을 적용할 Filter의 채널 수 == 입력 Feature Map의 채널 수
- Conv 연산을 적용한 Filter의 개수 == 출력 Feature Map의 채널 수

Convolution – One Filter Case



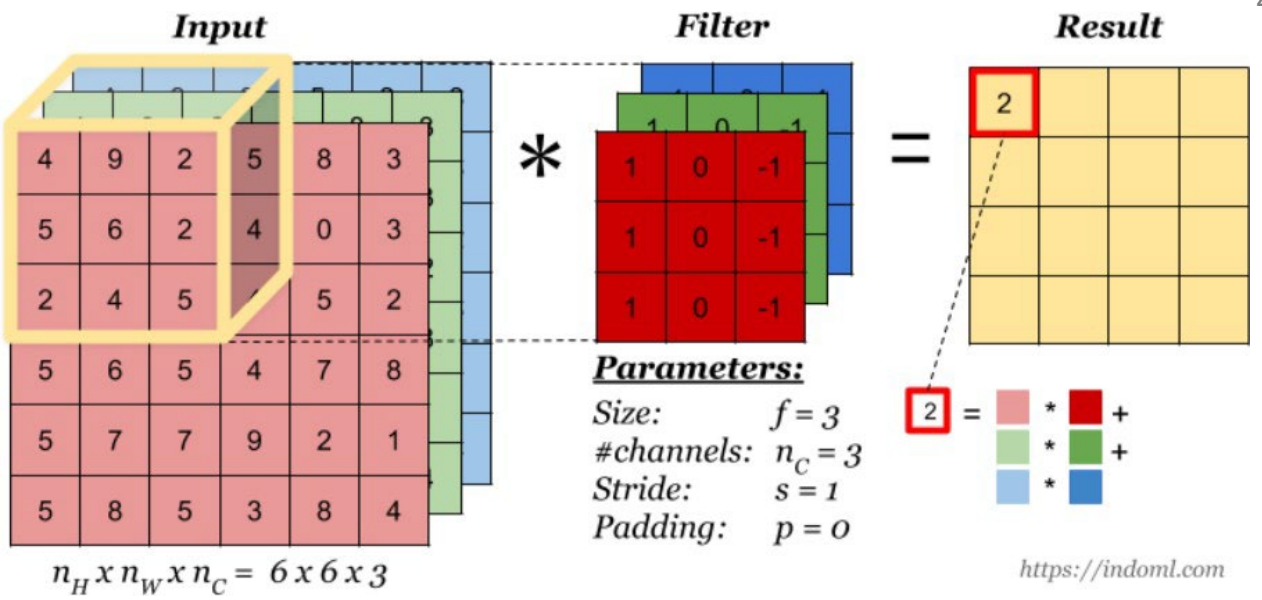
Input의 채널 수 = Filter의 채널 수

출력 Feature Map 크기 = 2 x 2 x 1

https://stackoverflow.com/questions/62493987/how-do-filters-run-across-an-rgb-image-in-first-layer-of-a-cnn

Convolution – One Filter Case

출력 Feature Map
4 x 4 x 1

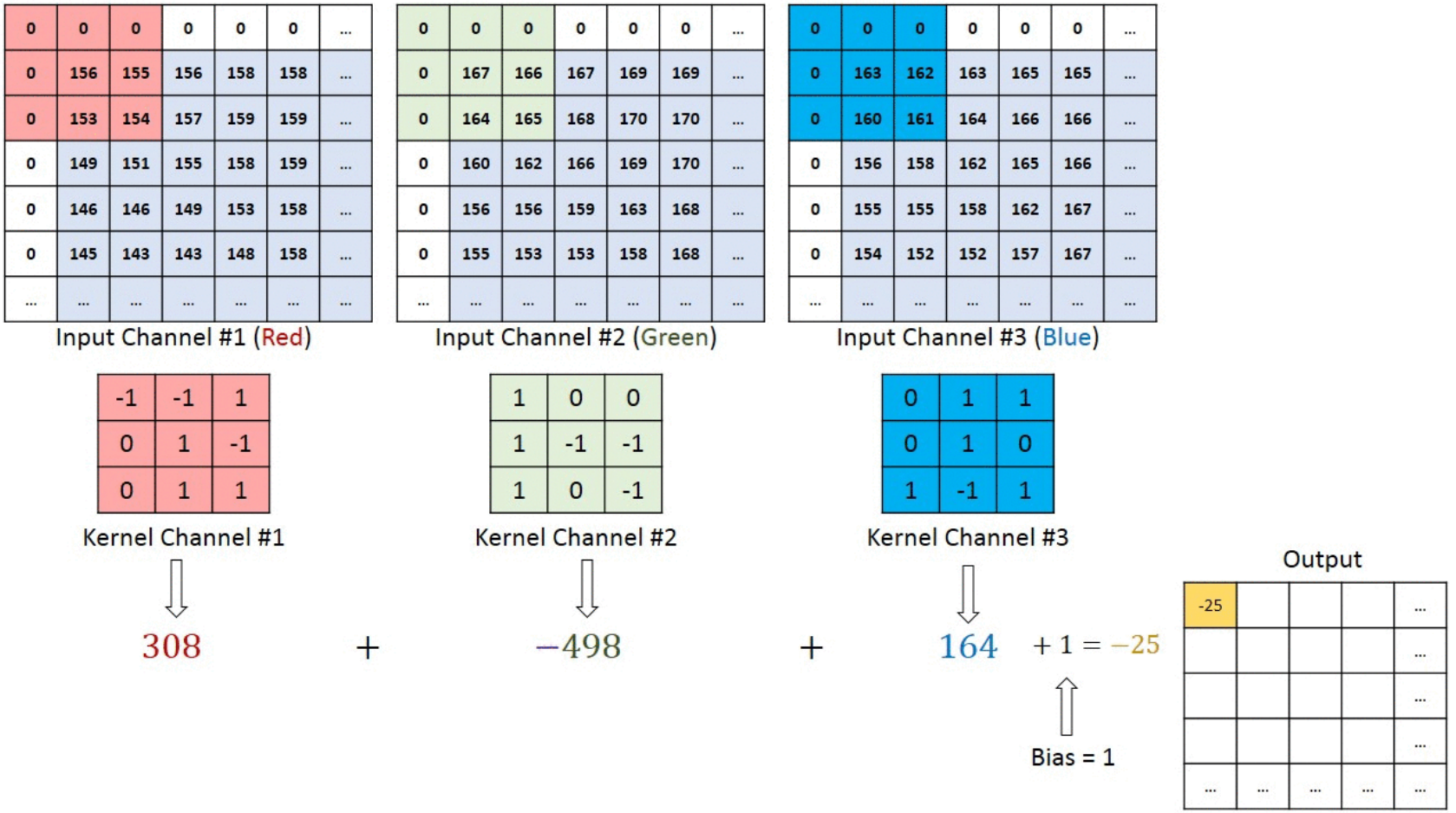


$n_H \times n_W \times n_C = 6 \times 6 \times 3$

입력 Feature Map = 6 x 6 x 3
(3개의 채널로 구성)

3개의 채널로
구성된 1개의 필터

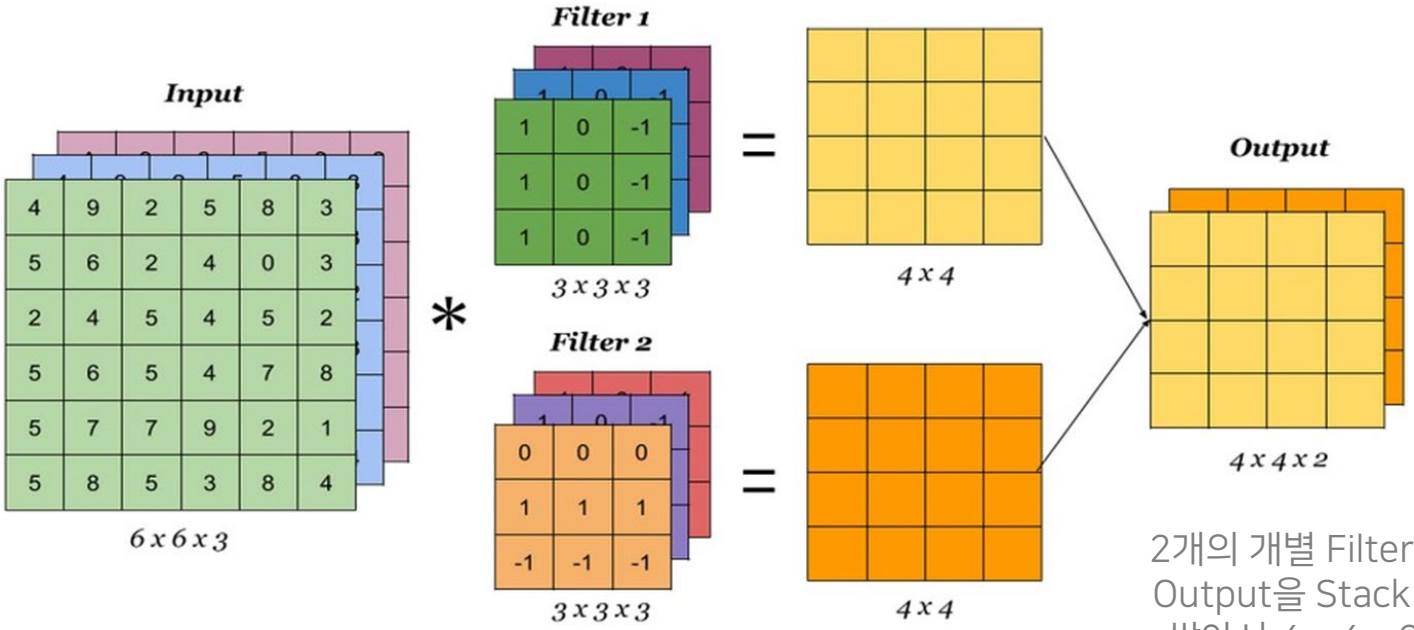
Convolution – One Filter Case



https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Convolution – Multi-filters Case

2개의 Filter 각각을 Input과 Conv 적용
2개의 4 x 4 Output 생성



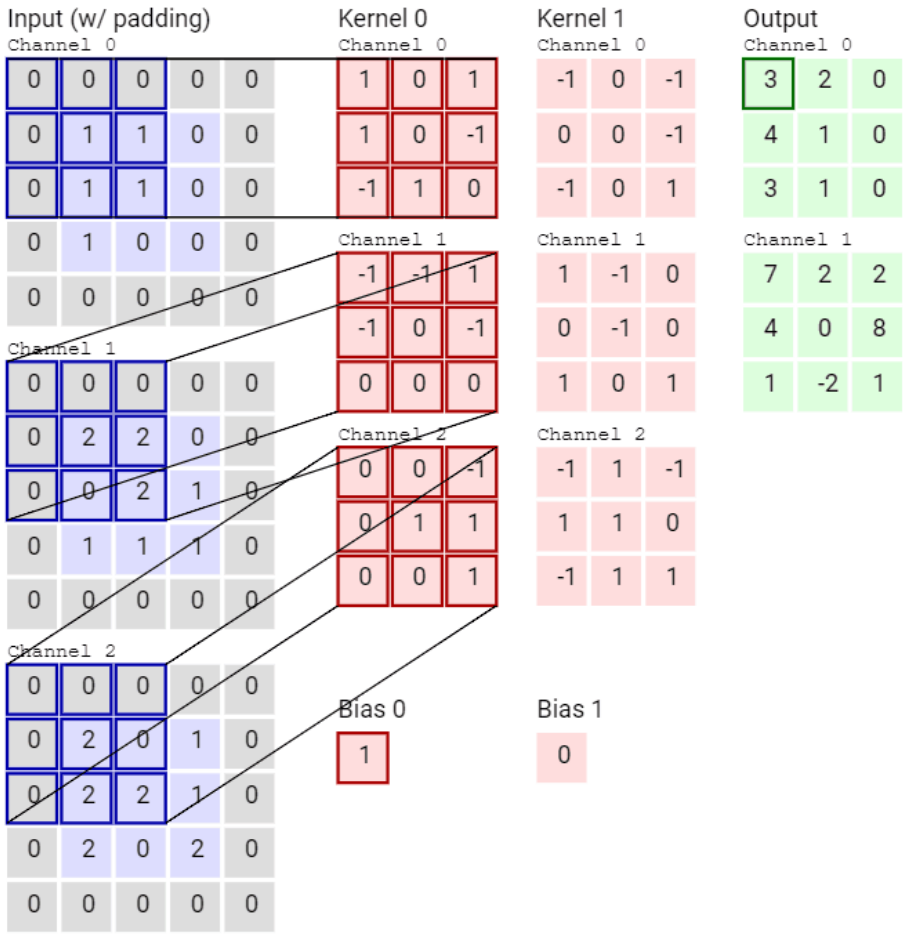
Filter 개수는 2개
개별 Filter들의 차원은 동일

2개의 개별 Filter로 생성된
Output을 Stack 형식으로
쌓아서 4 x 4 x 2 의 출력
Feature Map 생성

출력 Feature Map의 채널 수 = Conv를 적용한 Filter의 개수

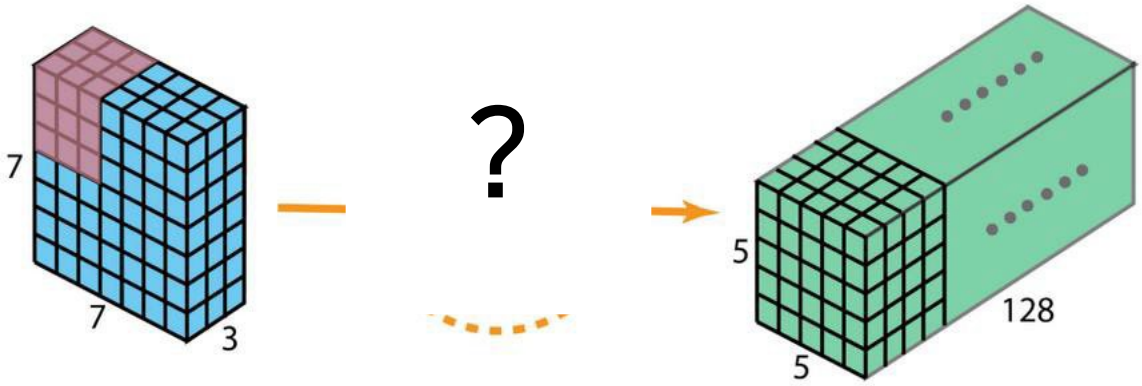
http://indoml.com

Convolution – Multi-filters Case



https://stats.stackexchange.com/questions/154798/difference-between-kernel-and-filter-in-cnn

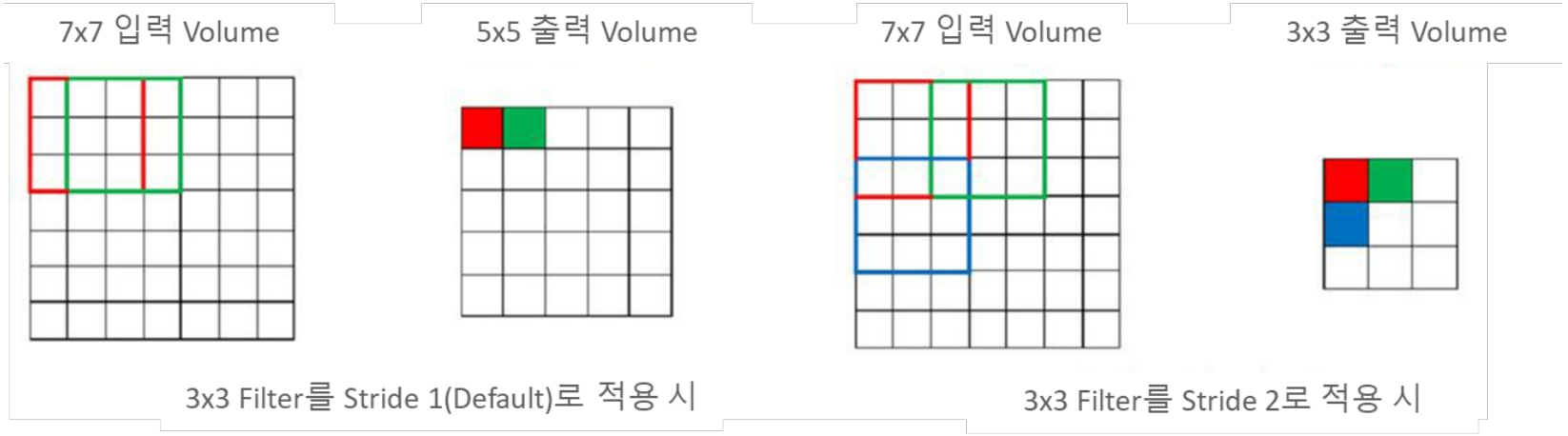
Convolution – Multi-filters Case



- ❖ Filter의 개수는?
- ❖ Kernel의 크기(Size)는?
- ❖ Filter의 Channel수는?
- ❖ 출력 Feature Map의 Channel수는?

https://developpaper.com/take-you-to-know-9-kinds-of-convolutional-neural-networks/

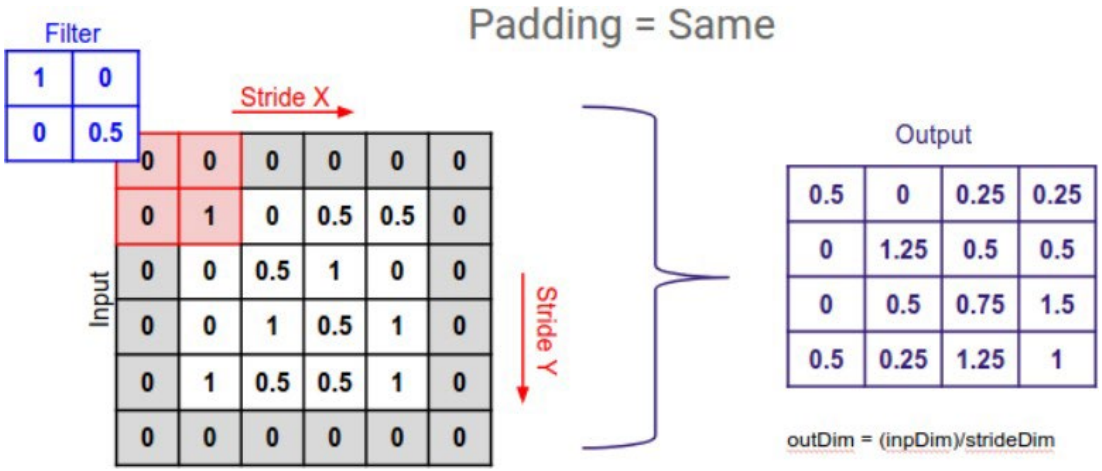
Convolution – Stride



- Stride는 필터가 이동하는 간격
- Stride == 2 ➔ Feature map의 크기는 절반
  - Stride를 키우면 공간적인 Feature 특성을 손실할 가능성이 높아짐
  - 하지만, **중요 Feature들이 손실되는 것을 의미하지 않음**
  - **Convolution 연산 속도 향상**

https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

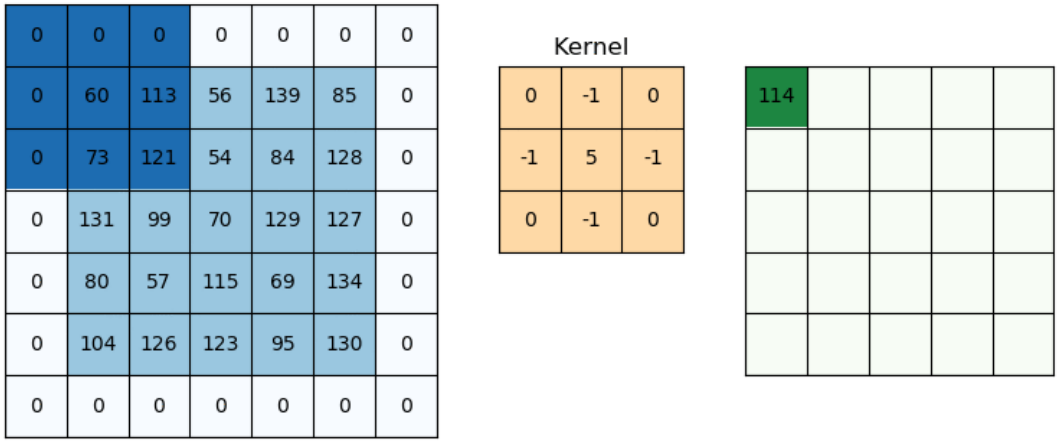
Convolution – Padding



- Convolution 연산 수행 시 출력 Feature Map이 지속적으로 작아지는 것을 방지
- Feature Map의 좌우 끝과 상하 끝에 각각 열과 행을 추가 한 뒤 , 0 값을 채워, 입력 Feature map 사이즈를 증가

https://ayeshmanthaperera.medium.com/what-is-padding-in-cnns-71b21fb0dd7

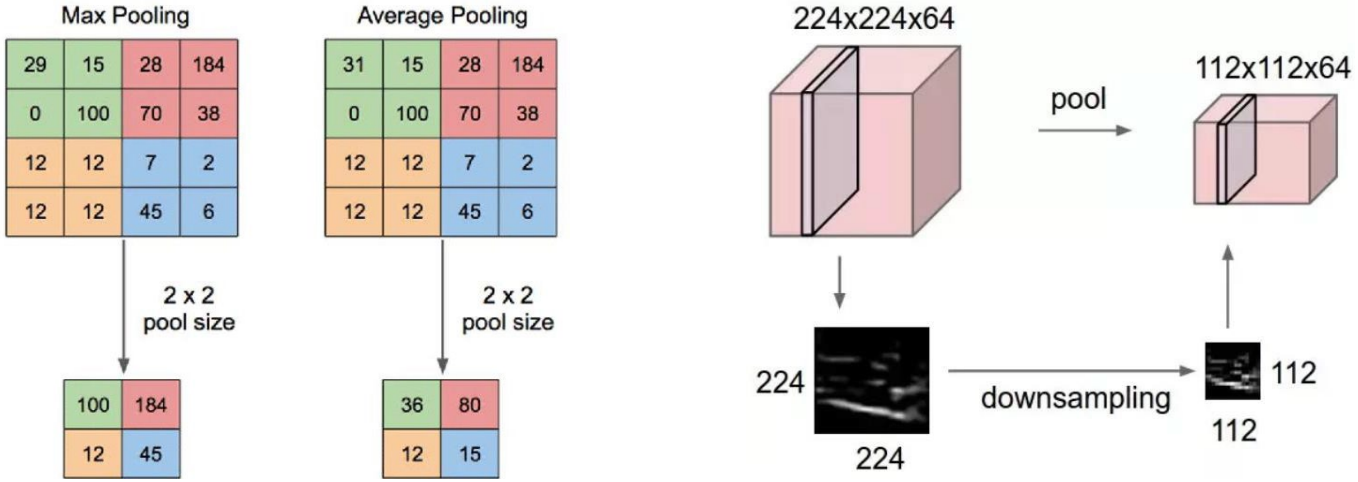
Convolution – Padding



- ❖ 모서리 주변(좌상단, 우상단, 좌하단, 우하단)의 Convolution 연산 횟수가 증가
  ➔ 모서리 주변 Feature 특징을 보다 강화
- ❖ Zero padding의 영향으로 모서리 주변에 0값이 입력되어 Noise가 약간 증가
  ➔ 성능에는 큰 영향이 없음

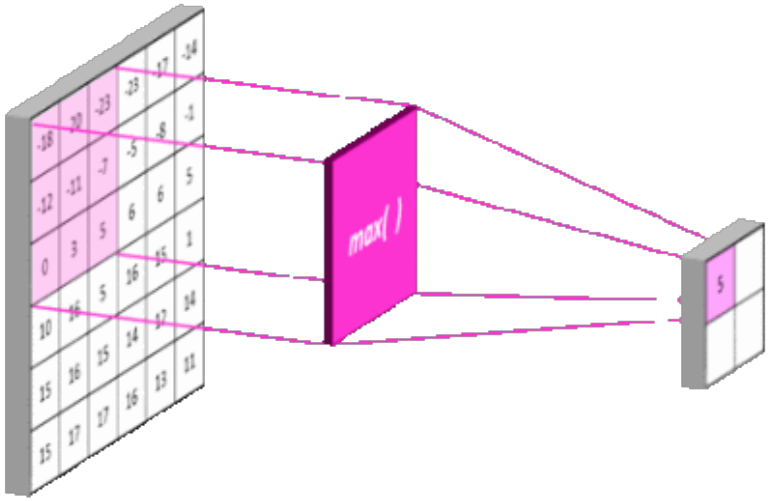
https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99

Convolution – Polling (Max. vs Average Polling)



- 일정 영역에서 가장 큰 값 또는 평균 값을 추출
  ➔ 위치의 변화에 따른 Feature 값의 변화를 일정 수준 중화
- Feature map의 사이즈를 줄임(Sub sampling = down sampling)
- 일반적으로 Convolution ➔ ReLU ➔ Pooling 순서로 적용

Convolution – Polling (Max. Polling)



- ❖ Feature Map의 크기 감소 ➔ Computation 계산속도 향상
- ❖ Max. Pooling은 Sharp한 Feature를 추출, Average Pooling은 Smooth한 Feature를 추출
- ❖ 일반적으로 Max. Pooling이 많이 사용됨

https://mlnotebook.github.io/post/CNN1/

Pooling and Stride

- Feature Map 크기를 줄이기 위한 방법 ➔ **Stride** 증가 vs **Pooling** 사용
  ➔ 위치 변화에 따른 Feature 값의 영향도 감소 (Spatial Invariance)
  ➔ Overfitting 감소

- Pooling의 경우 **특정 위치의 feature값이 손실되는 이슈**

- 과거 LeNet, AlexNet, VGG의 경우는 CNN(Stride/Padding) ➔ Activation ➔ Pooling 의 구조로 네트워크를 구성

- 최근 연구결과: Stride로 Feature Map 크기를 줄이는 것이 Pooling 보다 더 나은 성능 향상

- ResNet부터 이어지는 최근 CNN은 **최대한 Pooling을 자제, Stride를 이용한 네트워크 구성**

특징지도

(Feature Map)

Feature Map

Dog(3 by 32 by 32)

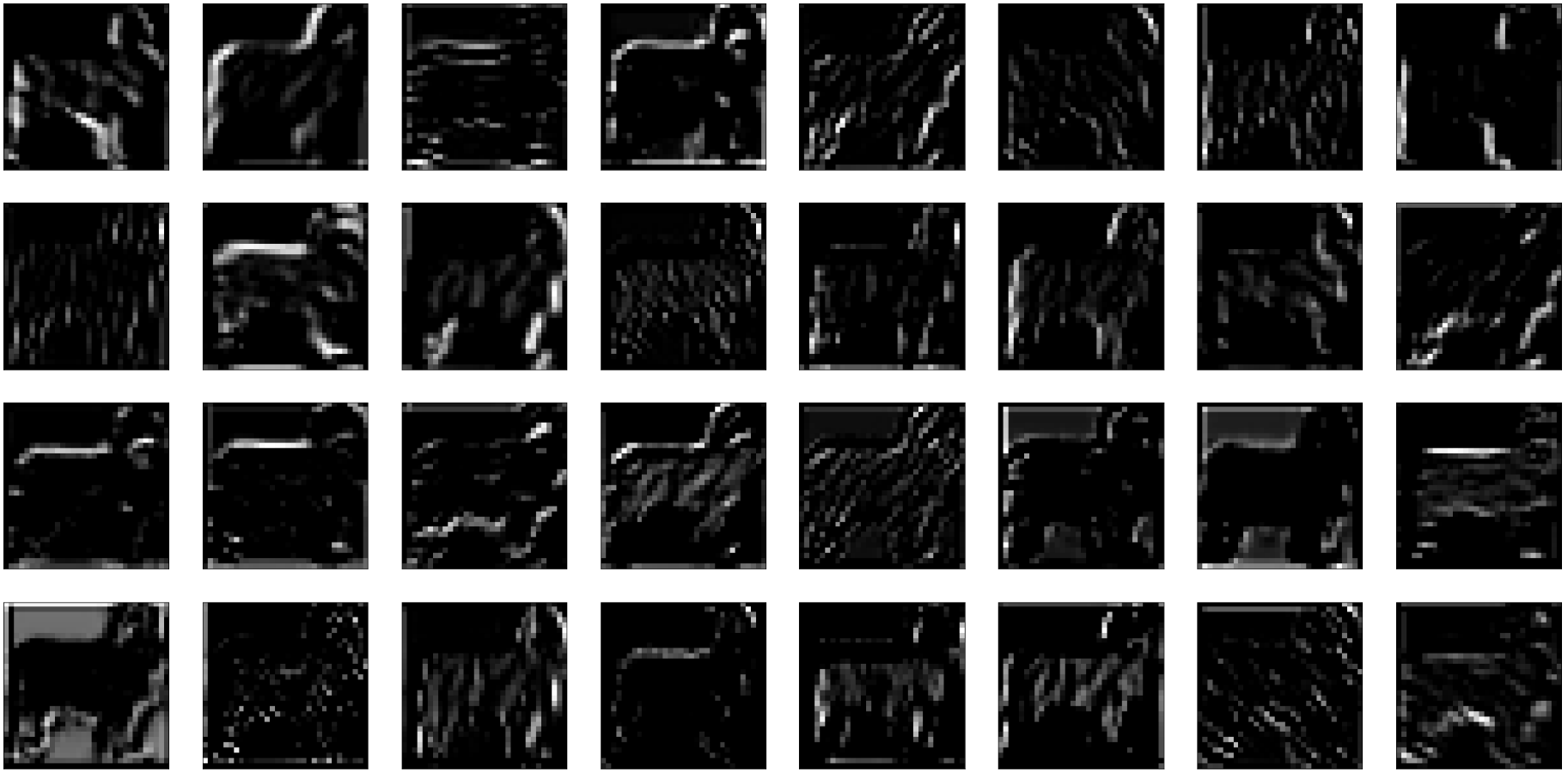
```
class CNNmodel_deep(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv_block1 = nn.Sequential(nn.Conv2d(3,32,3,padding=1),
                                         nn.BatchNorm2d(32),
                                         nn.ReLU(),
                                         nn.Conv2d(32,32,3,padding=1),
                                         nn.BatchNorm2d(32),
                                         nn.ReLU())
        self.Maxpool1 = nn.MaxPool2d(2)
        self.conv_block2 = nn.Sequential(nn.Conv2d(32,64,3,padding=1),
                                         nn.BatchNorm2d(64),
                                         nn.ReLU(),
                                         nn.Conv2d(64,64,3,padding=1),
                                         nn.BatchNorm2d(64),
                                         nn.ReLU(),
                                         nn.Conv2d(64,64,3,padding=1),
                                         nn.BatchNorm2d(64),
                                         nn.ReLU())
        self.Maxpool2 = nn.MaxPool2d(2)

        self.conv_block3 = nn.Sequential(nn.Conv2d(64,128,3,padding=1),
                                         nn.BatchNorm2d(128),
                                         nn.ReLU(),
                                         nn.Conv2d(128,128,3,padding=1),
                                         nn.BatchNorm2d(128),
                                         nn.ReLU(),
                                         nn.Conv2d(128,128,3,padding=1),
                                         nn.BatchNorm2d(128),
                                         nn.ReLU())
        self.Maxpool3 = nn.MaxPool2d(2)
        self.fc_block = nn.Sequential(nn.Linear(4*4*128,512),
                                      nn.Linear(512,10))
```

Test Accuracy: 83.4% in 10 classes

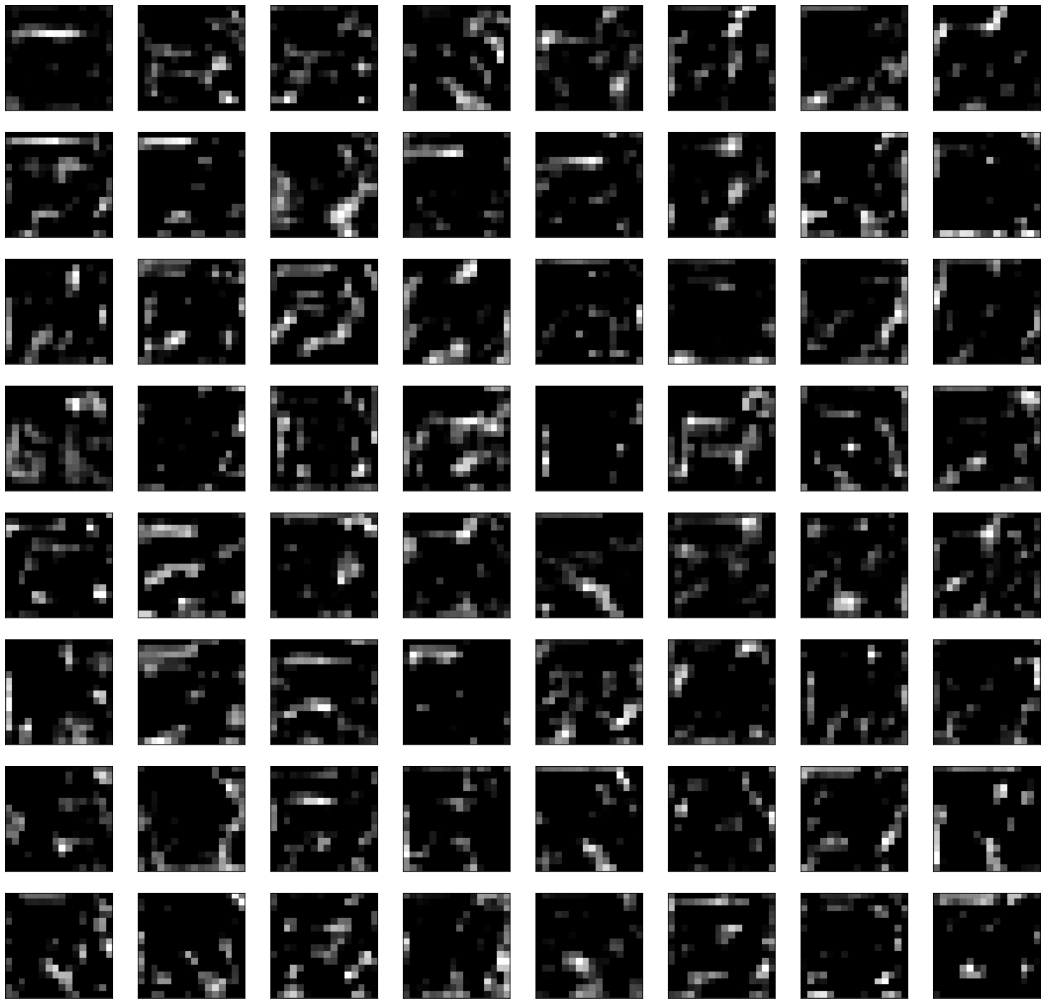
Feature Map

Conv Block 1 통과 (32개 채널)



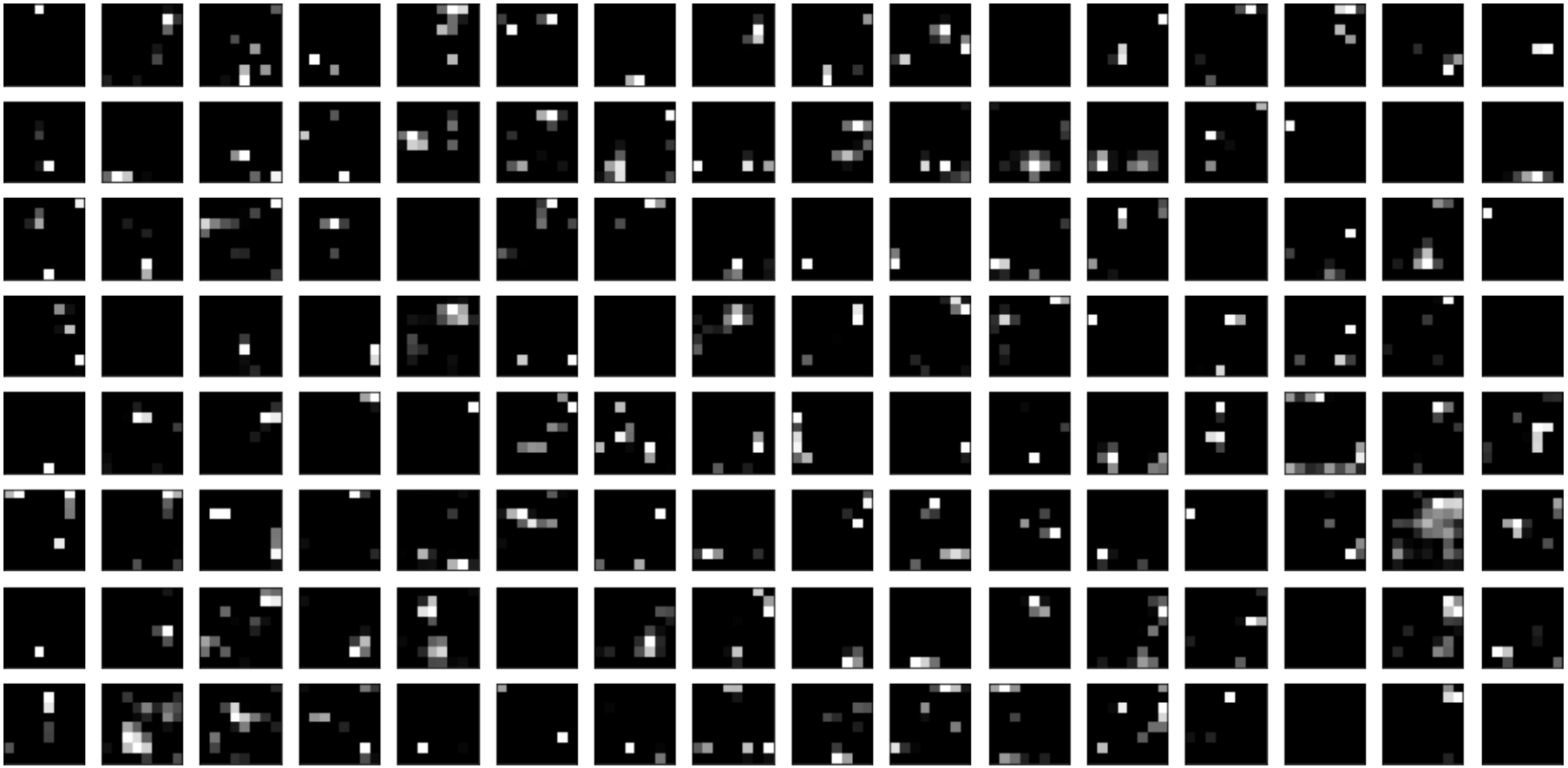
Feature Map

Conv Block 2 통과 (64개 채널)

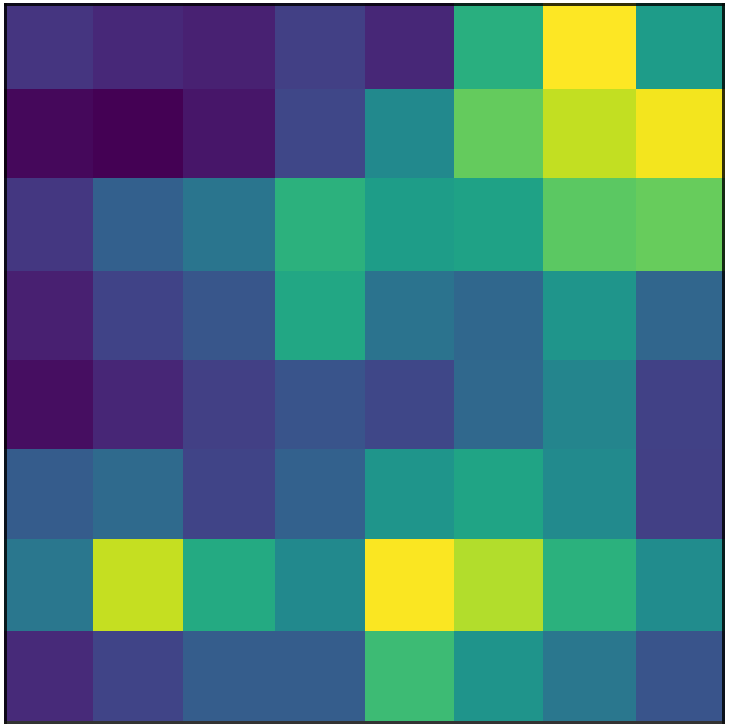


Feature Map

Conv Block 3 통과 (128개 채널)



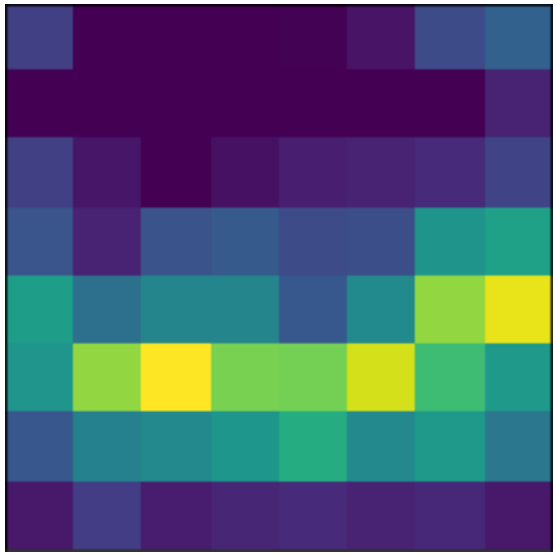
Feature Map

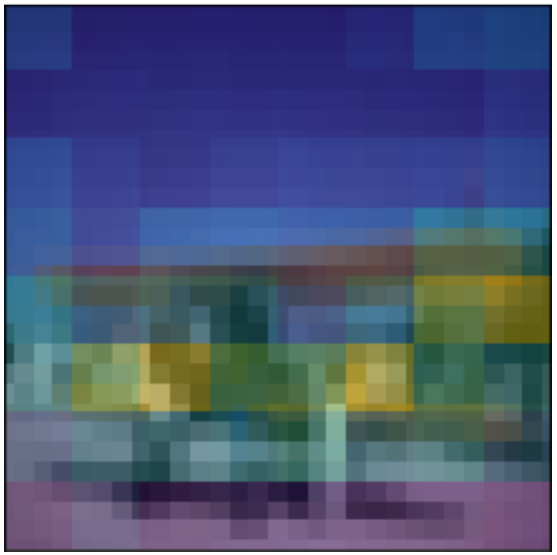


Feature Map

dog (cat)

Feature Map

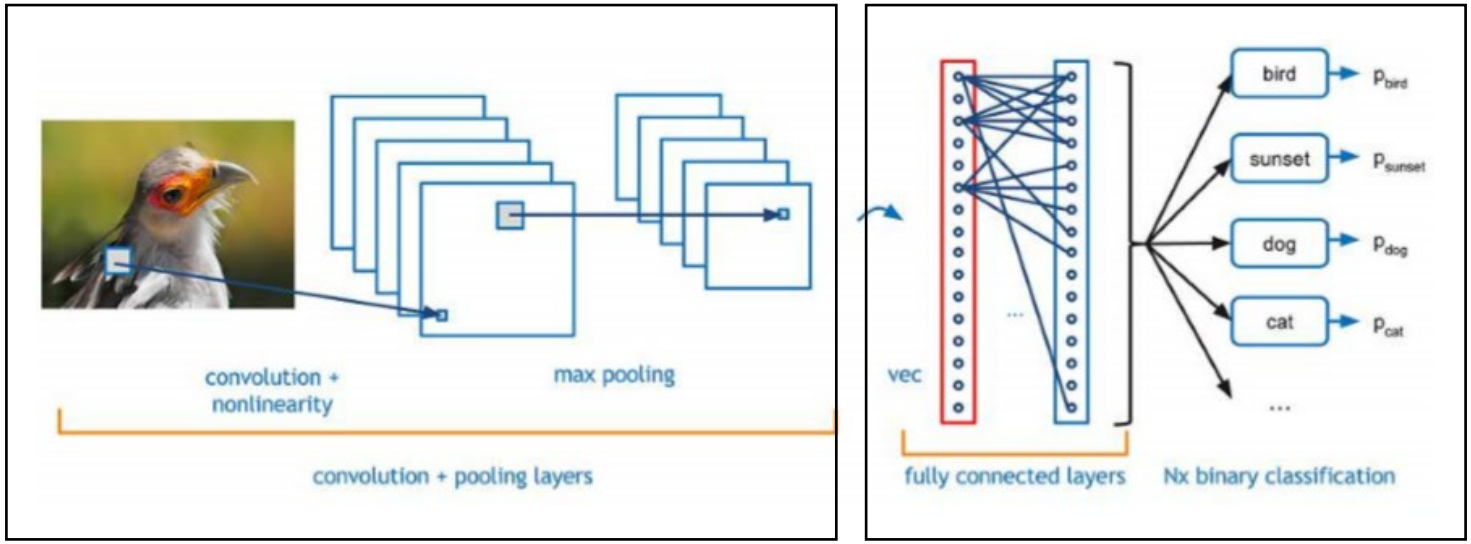




컨볼루션 신경망

(Convolutional Neural Network)

Convolutional Neural Network(CNN) – 전체 구조



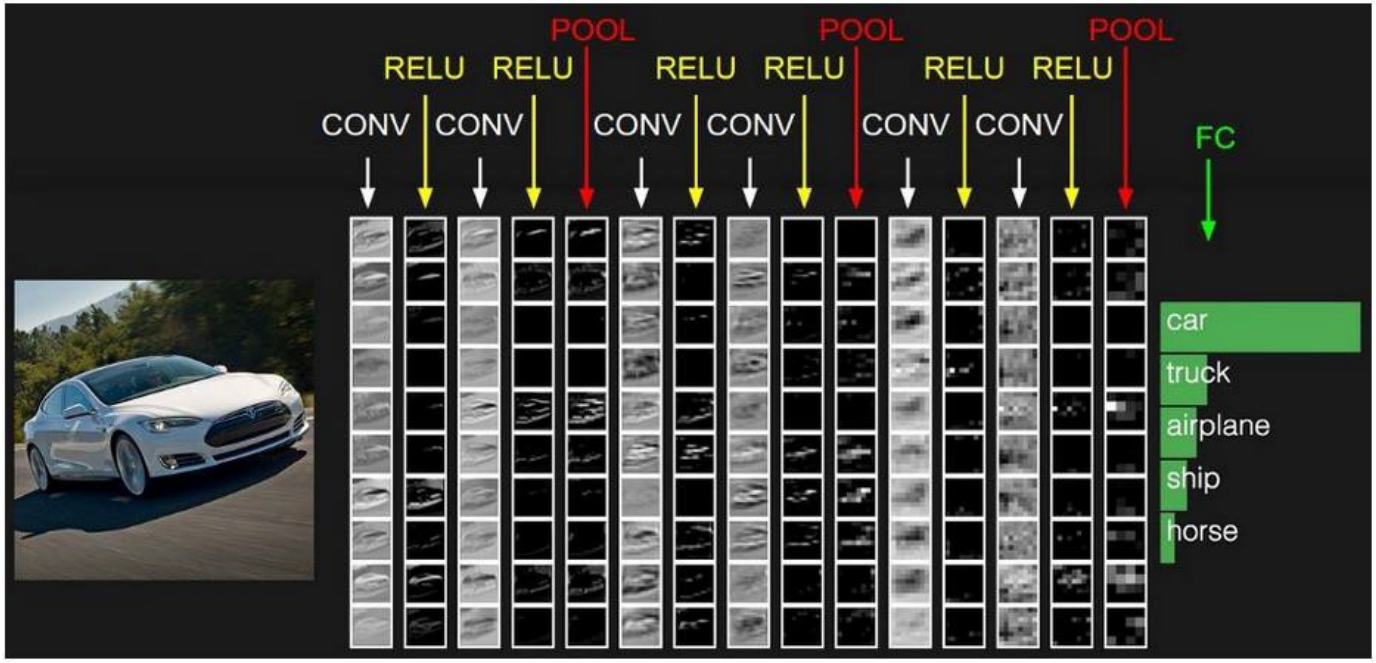
Feature Extractor
(CNN + Activation + Polling 등)

Classifier

- Classification에 맞는 **최적의 Feature** 추출
- 최적의 Feature 추출을 위한 **최적 Weight값** 계산
- 최적 Feature 추출을 위한 **필터(필터 Weight) 값** 계산

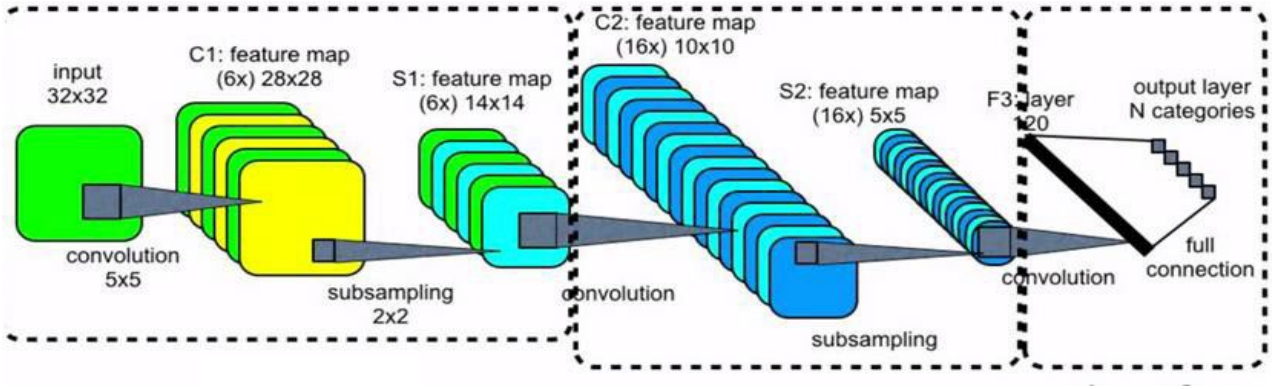
https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Convolution Neural Network – 전통적 구조



https://cs231n.github.io/convolutional-networks/

Convolution Neural Network – 전통적 구조

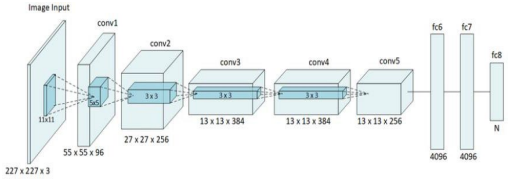


http://deeplearning.net/tutorial/lenet.html

CNN Architectures – Conventional CNN Models

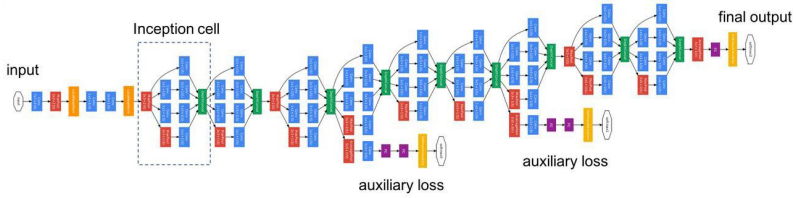
- AlexNet
- VGGNet
- GooLeNet
- ResNet
- Xception
- EfficientNet

AlexNet

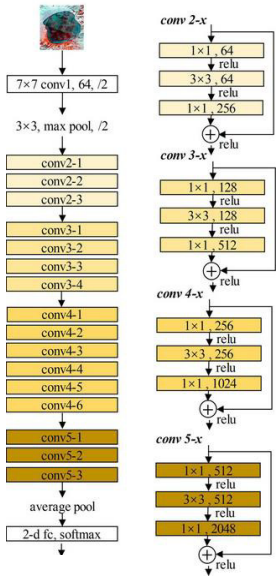


VGGNet

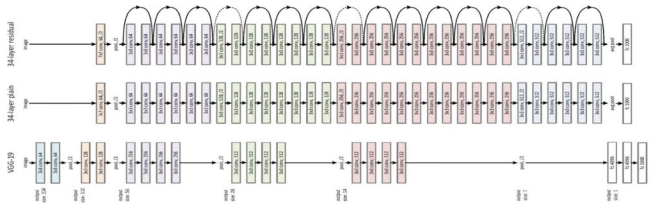
Inception



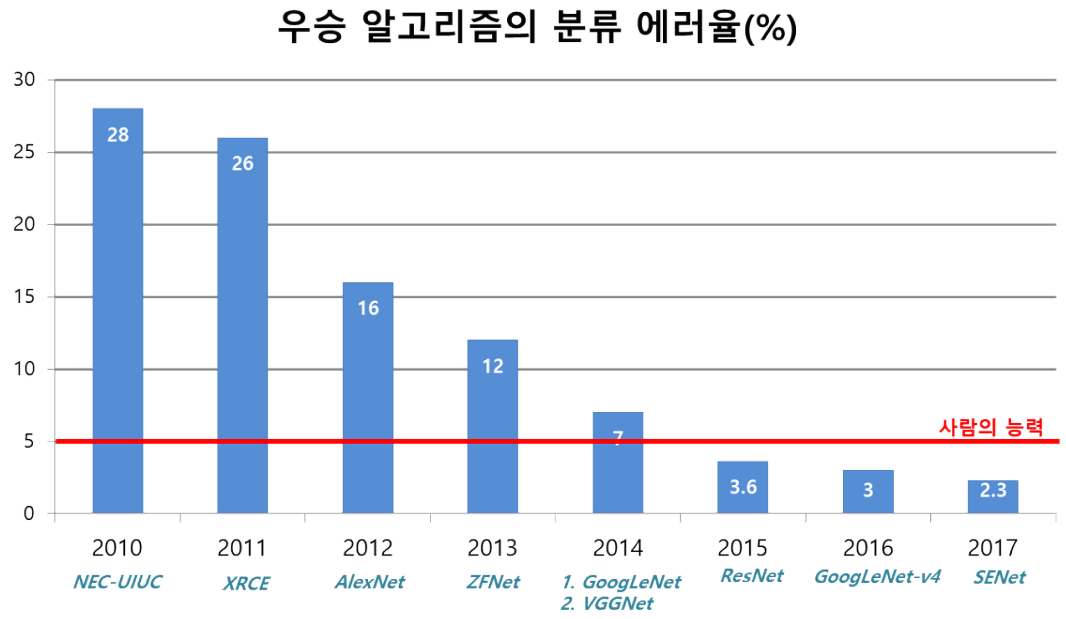
ResNet (50 Layers)



ResNet (34 Layers)

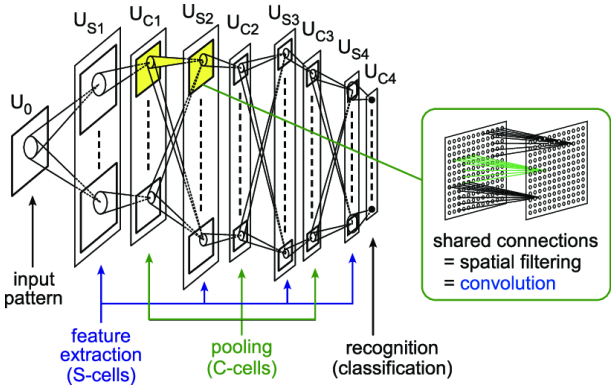


EfficientNet



- 이미지 넷(Image Net)의 사물 인식 대회(Large Scale Visual Recognition Challenge)
- ILSVRC는 이미지넷에서 제공하는 1,000여 카테고리로 분류된 100만개의 이미지를 인식하여 그 정확도를 겨루는 대회
- ILSVRC 대회 역대 우승 알고리즘

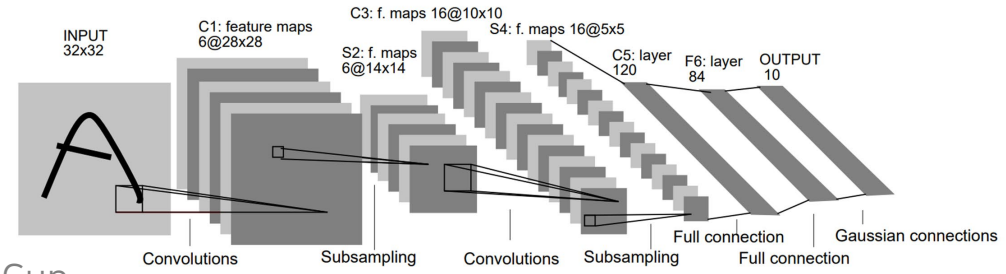
Neocognitron(1980), LeNet-5(1989-1998)



❖ Neocognitron (1982)

✓ 합성곱 레이어(convolution layer)와 다운 샘플링 레이어(down sampling)인 풀링(pooling) 개념이 제시됨

Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." Competition and cooperation in neural nets. Springer, Berlin, Heidelberg, 1982. 267-285.



❖ LeNet-5 (1998) by Yann LeCun

- 1998년 Yann LeCun 등에 의해 소개된 CNN 아키텍처
- 이미지 분류 및 인식 작업을 위해 설계된 최초의 딥러닝 모델 중 하나, **"역전파(back-propagation)"을 적용시킴**

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

AlexNet - Introduction

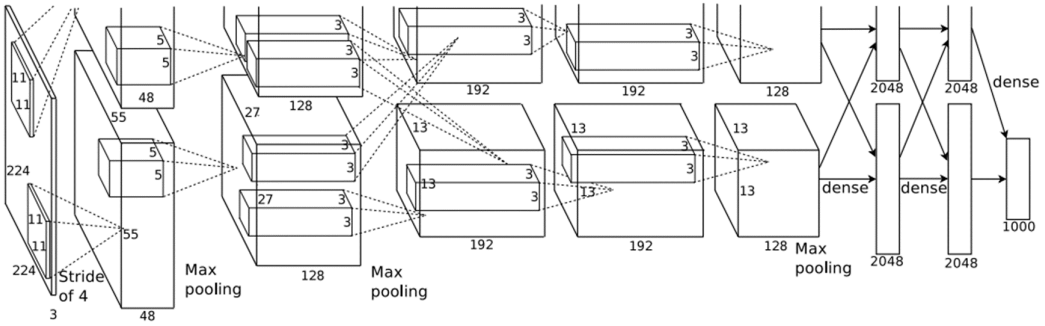


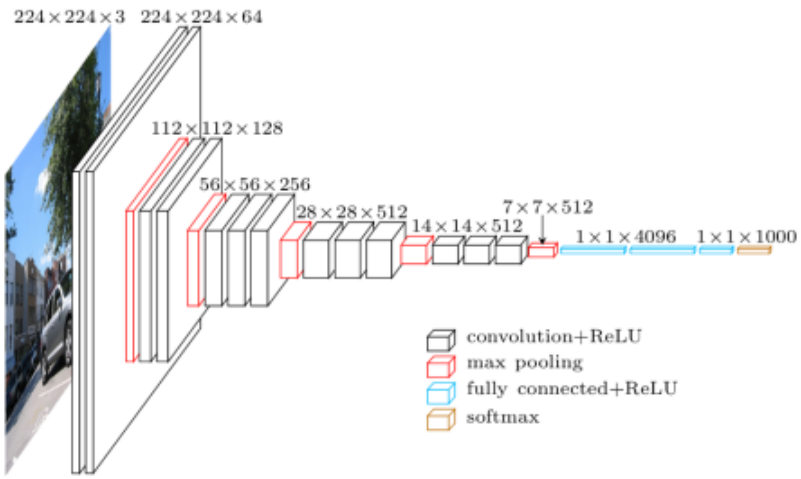
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

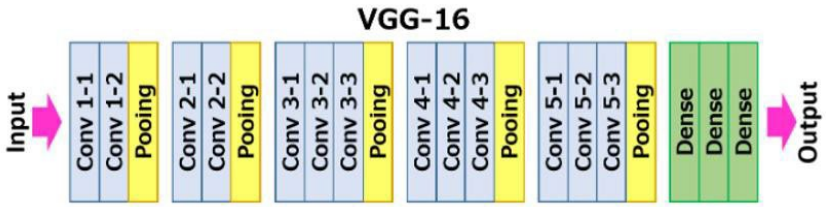
AlexNet – 2012년

- ❖ 대규모 벤치마크인 ImageNet 데이터 세트에서 상당한 성능 향상을 달성
- ❖ LeNet 비슷, 하지만 CNN을 보다 더 깊게 쌓음(5개 Convolution, 3개 Fully-connected 층)
- ❖ ReLU 활성함수, 데이터증강(Data Augmentation), Dropout 적용

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.

VGGNet – Visual Geometry Group

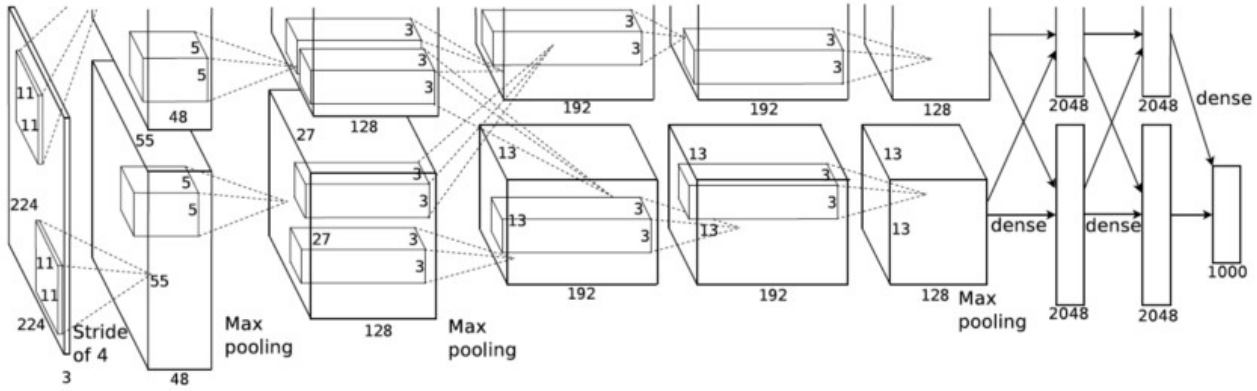




- VGGNet(Oxford 대학 팀)은 2014년 ImageNet Challenge에서 2위(1위 - GoogLeNet)
- ImageNet Challenge에서 Top-5 테스트 정확도 92.7%
- VGGNet-16
  - ✓ 13 Conv layers + 3 FC
  - ✓ 138 Million Weights and 500MB of Storage Space in Total
  - ✓ 변형: VGGNet-19

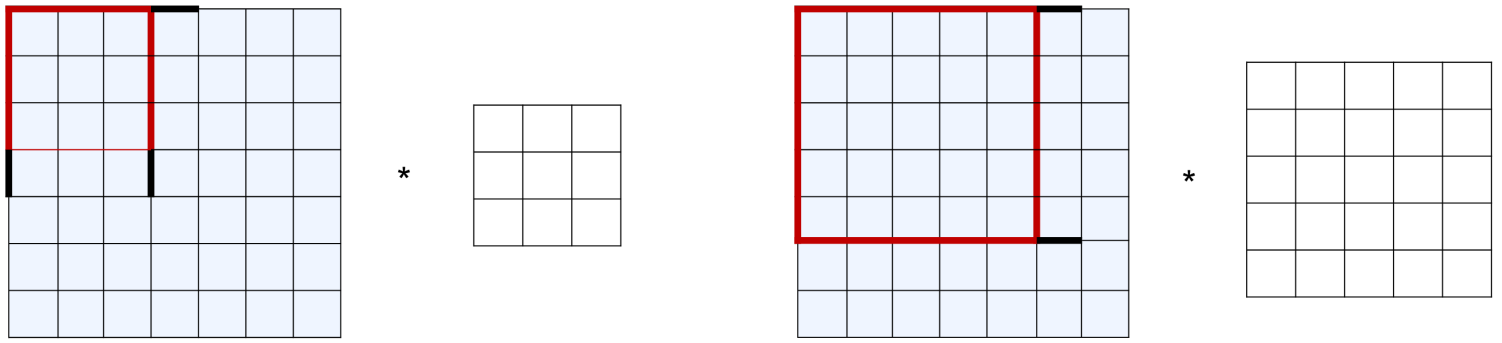
Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

VGGNet – AlexNet 리뷰



- VGGNet 이전에 나왔던 AlexNet의 경우에는 첫번째 Conv layer에 11x11 필터를 사용
- VGGNet의 가장 큰 특징!
  - ✓ 3x3 필터만 사용해도 충분하다는 것을 보여줌

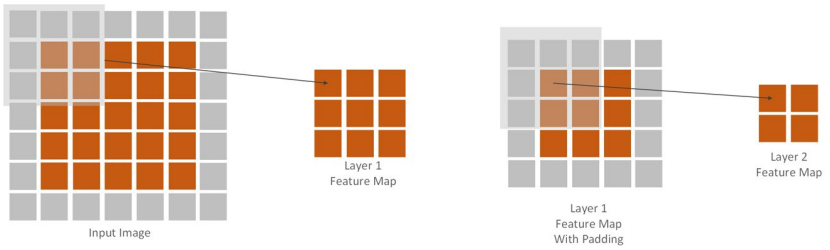
VGGNet



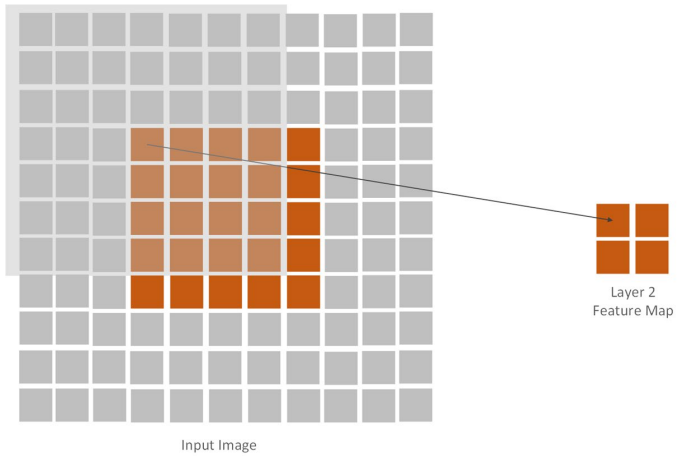
❖ Local Connectivity

- ✓ Receptive Field(RF)
- ✓ Kernel이 Convolution 연산을 수행할 때 이미지의 얼마나 많은 영역을 수용할 수 있는가?

VGGNet



$$r_i = r_{i-1} + (k-1)\prod_{k=1}^{i-1} s_k$$



- Local Connectivity
  - ✓ Receptive Field(RF)
    - Convolution 연산을 수행할 수록 RF가 더 커짐

https://rubikscode.net/2021/11/15/receptive-field-arithmetic-for-convolutional-neural-networks/

VGGNet – 7x7 필터 vs 3x3 필터 3번 사용

$$r_i = r_{i-1} + (k-1)\prod_{k=1}^{i-1} s_k$$

	7x7 필터한번	3x3 필터세번
ReceptiveField	$R_1 = k = 7$	$R_1 = k = 3$ $R_2 = R_1+3-1 = 3+2=5$ $R_3 = R_2+3-1 = 5+2=7$

- 7x7 커널을 하나만 사용한 경우 vs 3x3 커널을 3번 사용한 경우
  - ✓ 동일한 효과를 갖는다
  - ✓ Conv 연산의 횟수가 증가하는데 사용하는 메모리가 늘어나는 단점이 있지 않을까?

VGGNet - 5x5 필터 vs 3x3 필터 2번 사용

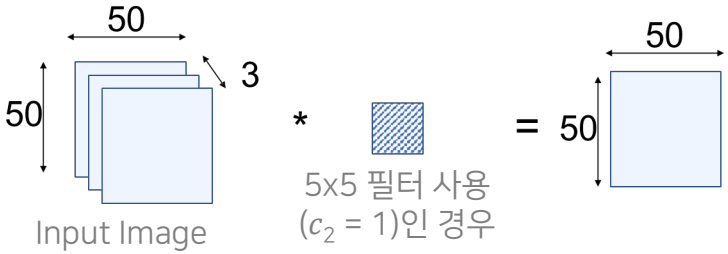
❖ Layer i의 가중치(weights) 수

= 학습을 통해 업데이트 되는 파라미터의 수

= $(k \times k \times c_{i-1}) \times c_i$

- k : filter size
- c_{i-1} : 이전 레이어의 채널의 수
- c_i : 현재 레이어의 채널의 수

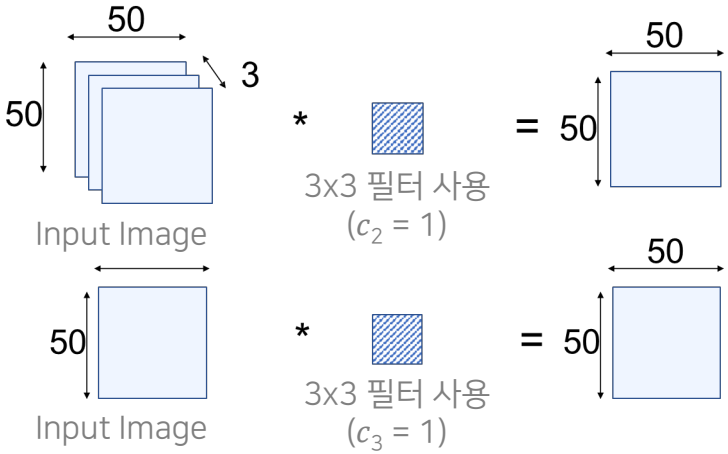
❖ 5x5 필터를 사용하는 경우



2번째 레이어의 Weight 수

$=(k \times k \times c_1) \times c_2 = (5 \times 5 \times 3) \times 1 = 75$

❖ 3x3 필터를 2개 사용하는 경우



2번째 레이어의 Weight 수

$=(k \times k \times c_1) \times c_2 = (3 \times 3 \times 3) \times 1 = 27$

3번째 레이어의 Weight 수

$=(k \times k \times c_2) \times c_3 = (3 \times 3 \times 1) \times 1 = 9$

총 weight의 수 = 36

VGGNet - 5x5 필터 vs 3x3 필터 2번 사용

	5x5 필터한번	3x3 필터두번
총 weight의 수	75	36
Activation Function	1	2
Receptive Filed	$R_1 = k = 5$	$R_1 = k = 3$ $R_2 = R_1 + 3 - 1 = 3 + 2 = 5$

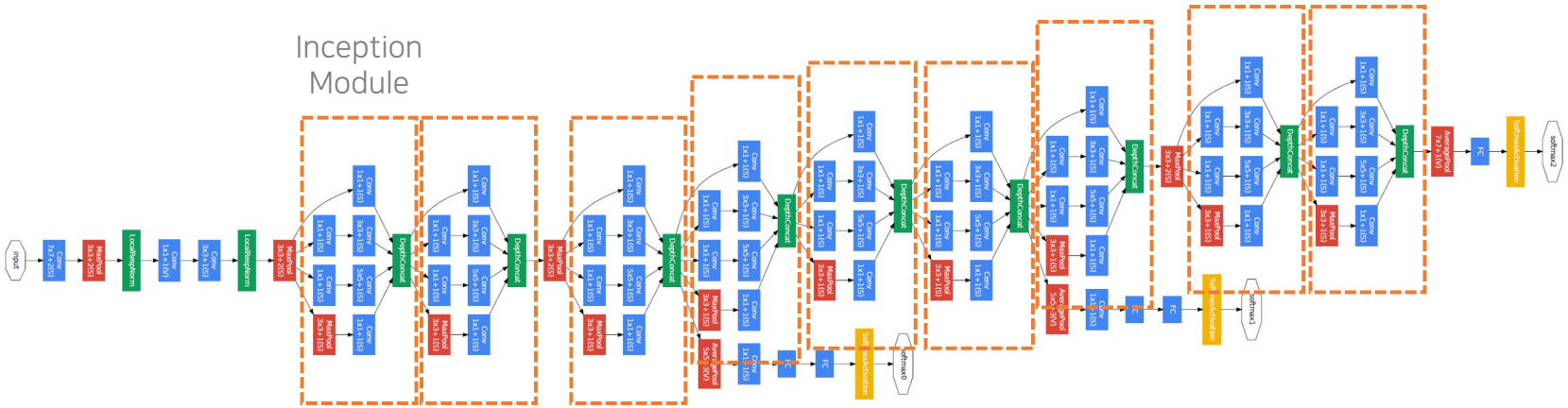
- 동일한 Receptive Filed!
- Weight의 수는 줄어든다 ➔ 메모리 사용량이 감소함
- 더 많은 Activation Function을 통과한다 ➔ 비선형성이 증가함

VGGNet - Conclusions

- 3x3 Convolution 연산을 여러 번 사용했을 때의 효과
  - ✓ Receptive Field를 늘리는 효과
  - ✓ 큰 사이즈의 Kernel을 사용했을 때 보다 Weight의 수는 줄어든다:

    메모리 사용량이 감 소함
  - ✓ 더 많은 Activation Function을 통과한다: 비선형성이 증가함

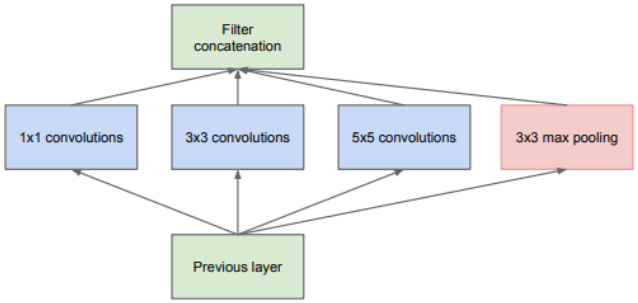
GoogLeNet (Inception v1)



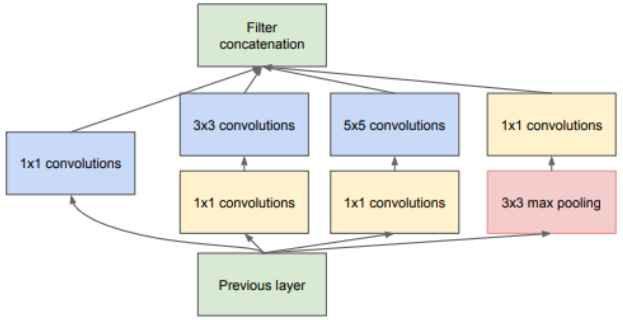
- 2014년 ILSVRC 1위 모델
- 22층 신경망, Inception Module 사용
- 22 Layers
  - ✓ 9개의 Inception Module(개별적인 Layer는 100개 이상)
  - ✓ 총 5 Million Weights

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

GoogLeNet – Inception Module



(a) Inception module, naïve version



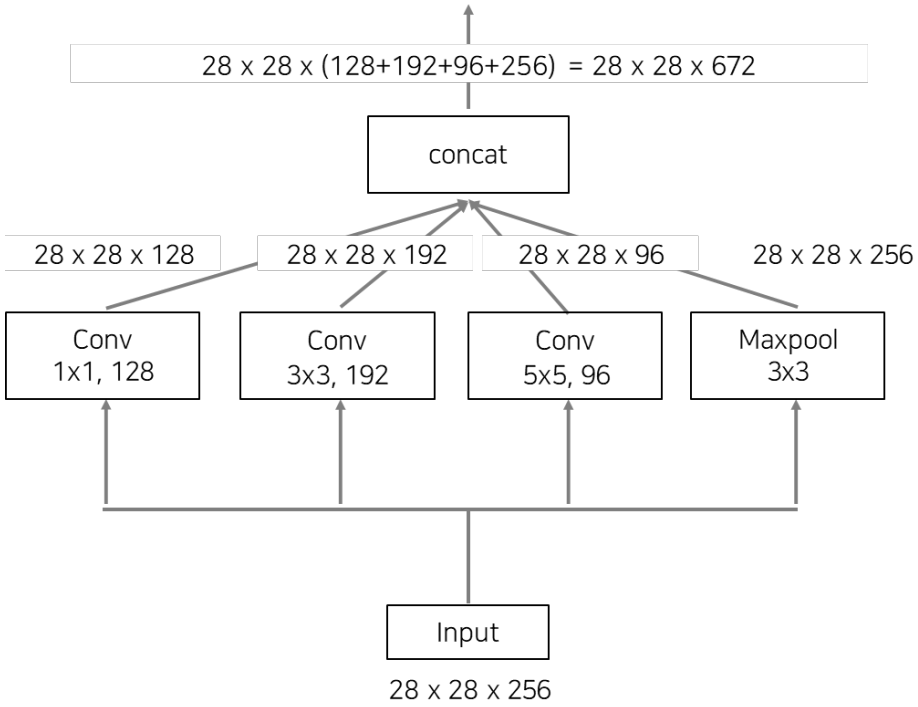
(b) Inception module with dimension reductions

- Split-transform-merge Strategy
  - ✓ 다양한 크기의 필터를 사용해서 다양한 특징을 추출
  - ✓ 1×1, 3×3, 5×5 Kernels
- Bottleneck Structure
  - ✓ 1x1 Conv: 필터의 크기가 1
  - ✓ Dimensionality Reduction 효과: 입력의 필터 크기 보다 Conv의 필터 수가 작을 때 연산 량을 줄이고, 신경망을 더욱 깊게 쌓을 수 있음
  - ✓ 비선형성(nonlinearity) 증가

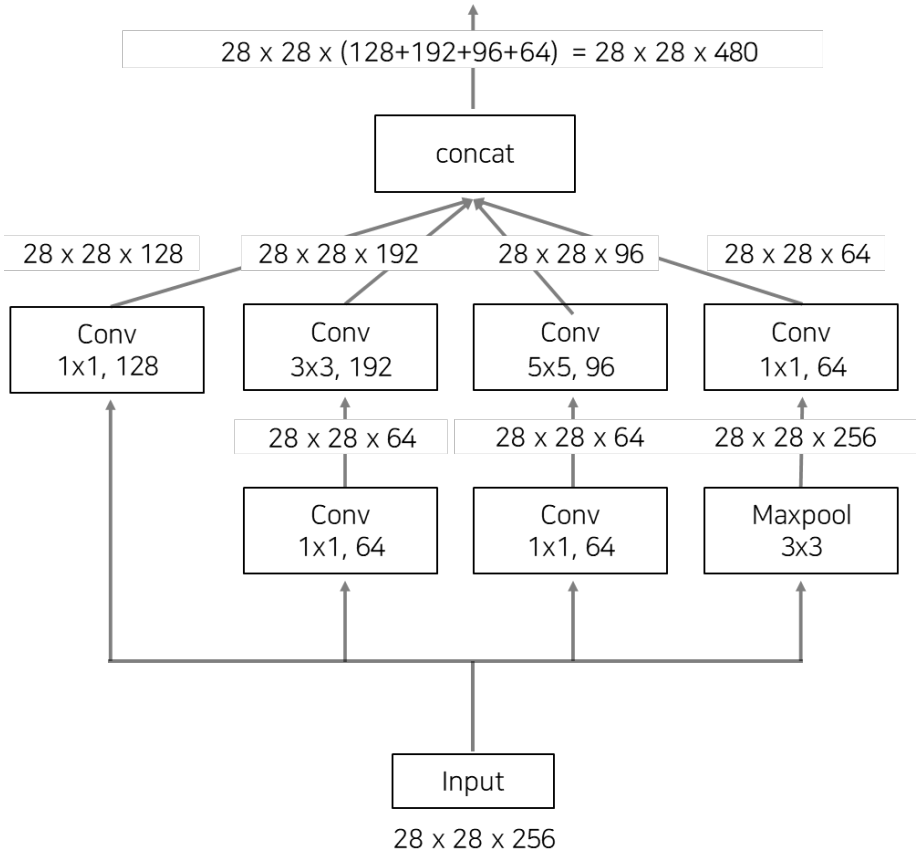
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

GoogLeNet – Memory of Inception Module

Naïve Inception



Inception with bottleneck structure

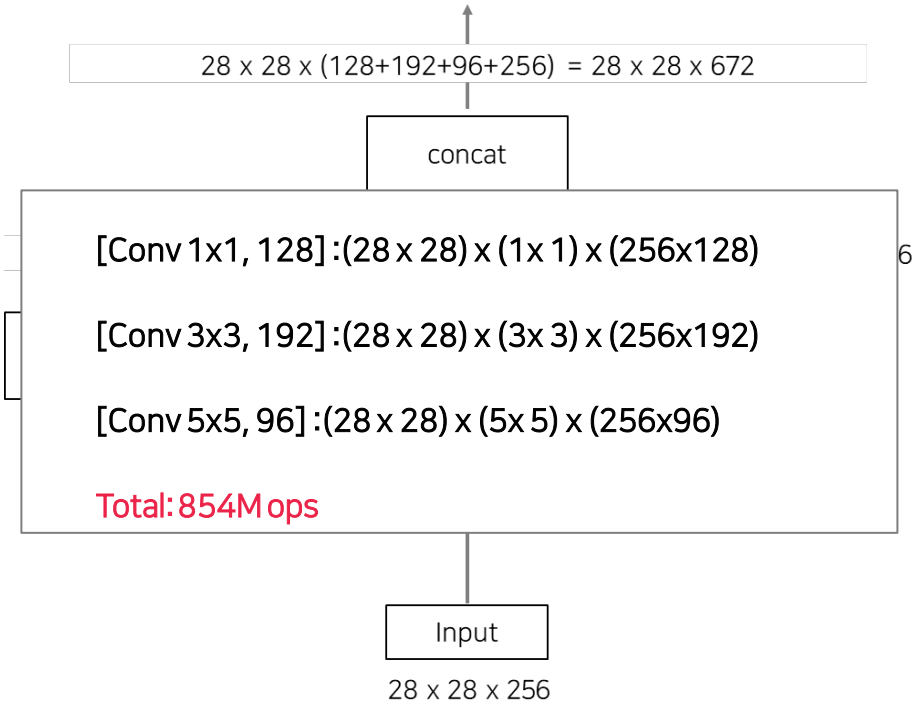


Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

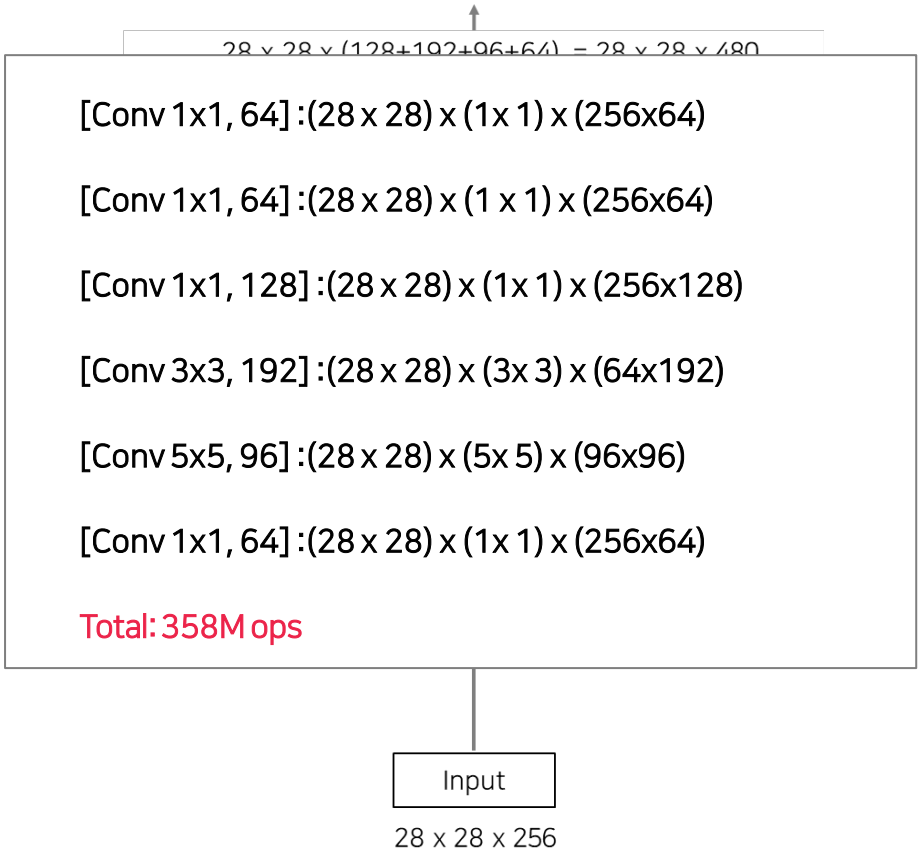
GoogLeNet – Memory of Inception Module

Conv Ops = $(k \times k) \times (\text{mi} \times mi) \times (\text{ci}-1 \times \text{ci})$

Naïve Inception

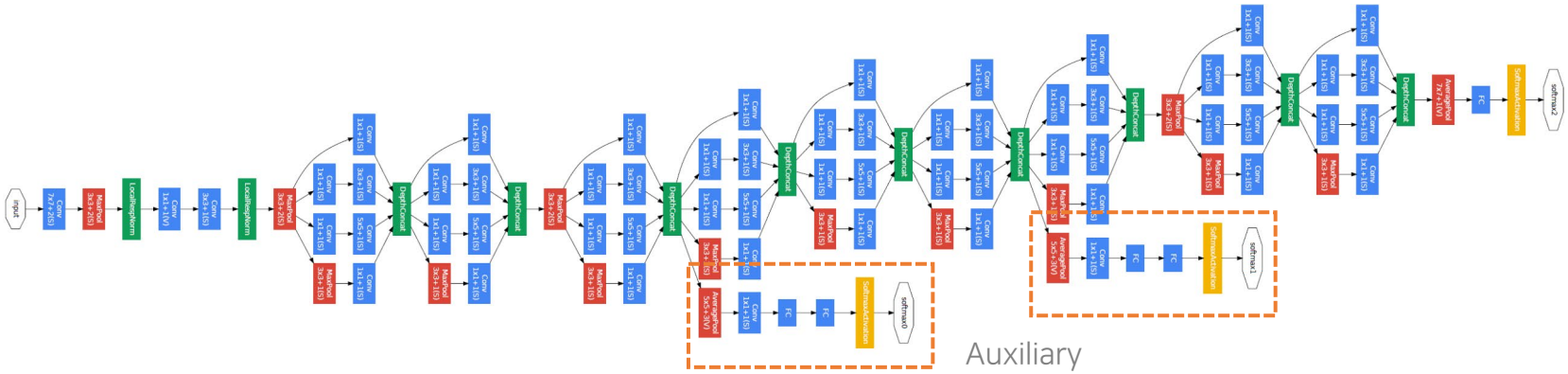


Inception with bottleneck structure



Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

GoogLeNet – Auxiliary Classifier



- Auxiliary Classifier: 역전파 신호를 잘 전달하기 위해 학습 시 사용하는 모듈
- 학습 이후 추론 단계에서는 사용하지 않음

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

GoogLeNet – Conclusions

- Inception
  - ✓ Parallel filters concatenation + Bottleneck architecture
- 가중치의 수를 줄임
- 비선형성을 증가시킴
- 모델 구성의 새로운 패러다임
  - ✓ 단순히 layer를 쌓아 올리는 구조가 아닌, 모듈/블록 구조를 사용

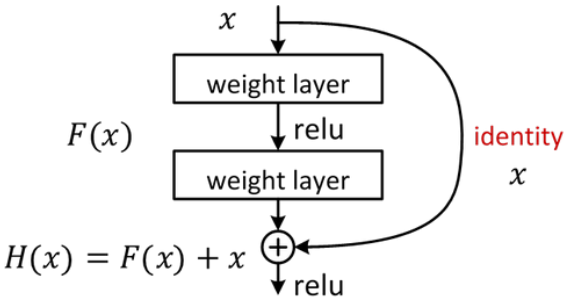
ResNet

CVPR 2016 open access

These CVPR 2016 papers are the Open Access versions, provided by the Computer Vision Foundation.

Except for the watermark, they are identical to the accepted versions; the final published version of the proceedings is available on IEEE Xplore.

Search

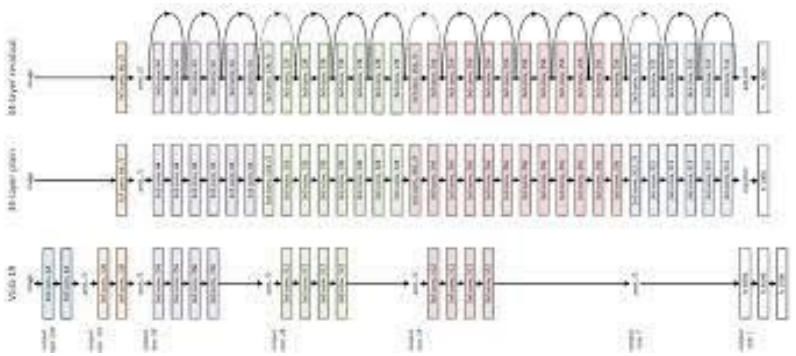


Deep Residual Learning for Image Recognition

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

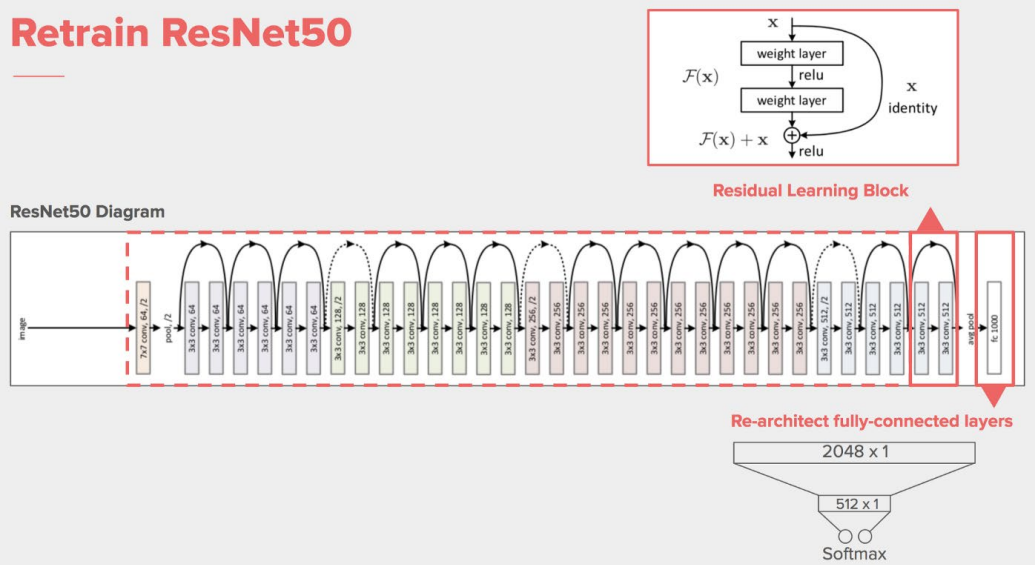
Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.



He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

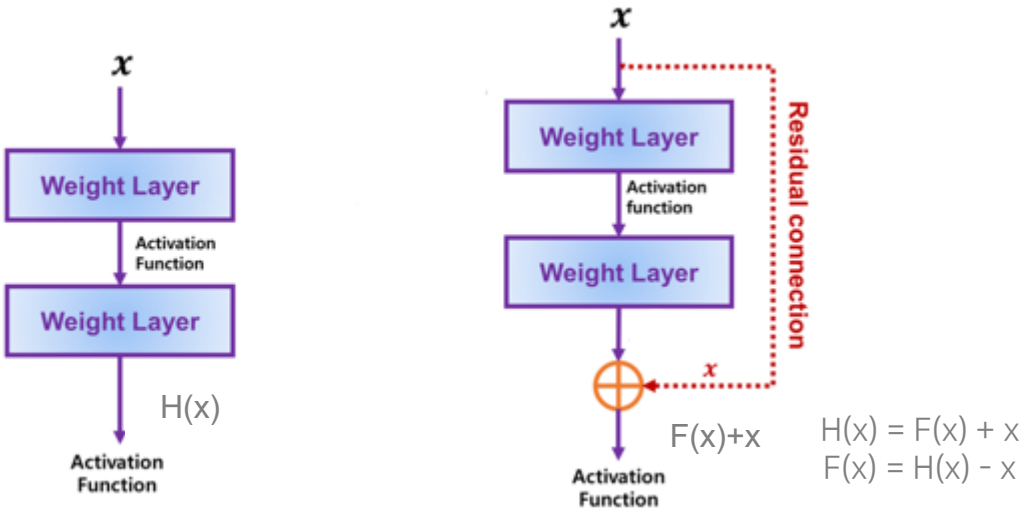
ResNet



- 2015년 ILSVRC 1위 모델
- 깊은 신경망(152 Layers) 학습이 가능하게 하는 Skip Connections 구조 제안
- Batch Normalization 적용
- 분류기 학습을 위한 Fully Connected를 사용하지 않고, GAP(Global Average Pooling)을 수행

https://medium.com/@wularitz/image-classifier-for-dish-classification-using-resnet50-with-pytorch-5d11c02067b5

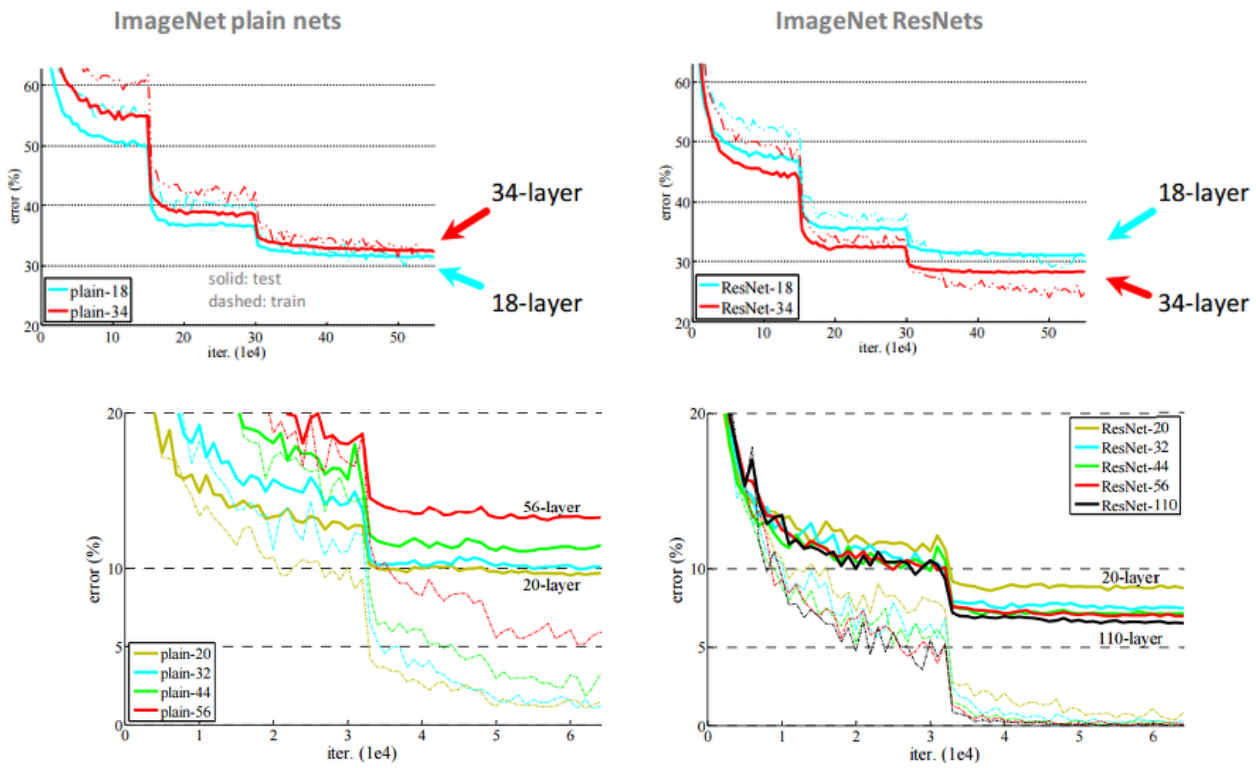
ResNet – Skip Connection



- Residual Network는 하나의 모델로 여러 모델을 훈련 시킨 것과 같은 효과를 얻는 Dropout과 유사 ➔ CNN 모델에 적용
- 일부는 레이어를 건너 뛰는 방식을 취하고, 일부는 그대로 넘어와서 두 값을 합치는 방식
- Residual 구조
  - 일반적인 신경망: H(x)를 얻기 위해 학습 수행
  - ResNet: 잔차(F(x) = H(x) – x)를 최소화하는 방향으로 수행
  - 깊은 층을 가지고 있지만 Gradient Vanishing이 발생하지 않고 학습이 잘 됨

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

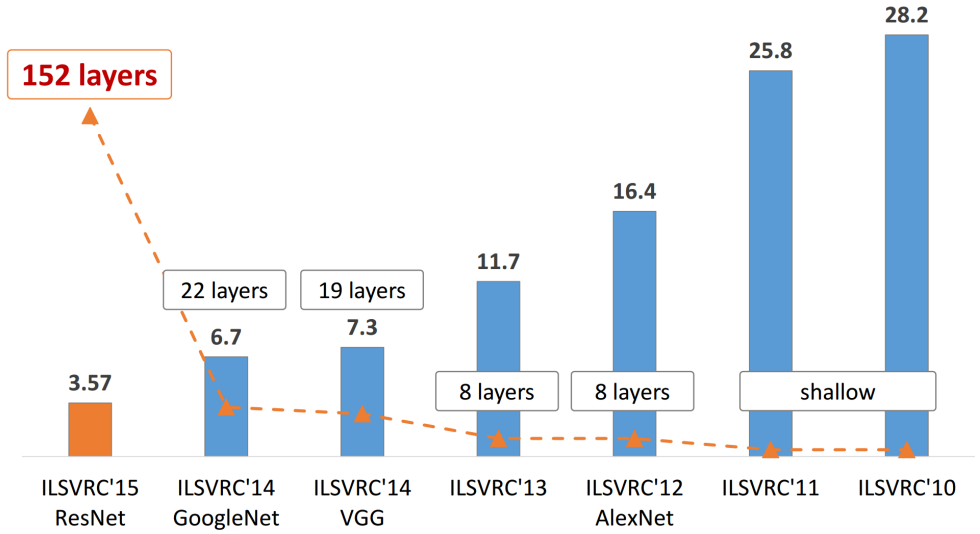
ResNet - Performances



- AlexNet(2012)의 경우 8개의 Layer를 사용, GoogLeNet(2014)은 22개의 Layer를 사용
- 레이어가 많다고 더 좋은 성능을 발휘하는 것은 아님
- 하지만, Residual Network를 사용하면 152개의 레이어를 사용해서 좋은 성능을 보여줌

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

ResNet - Performance

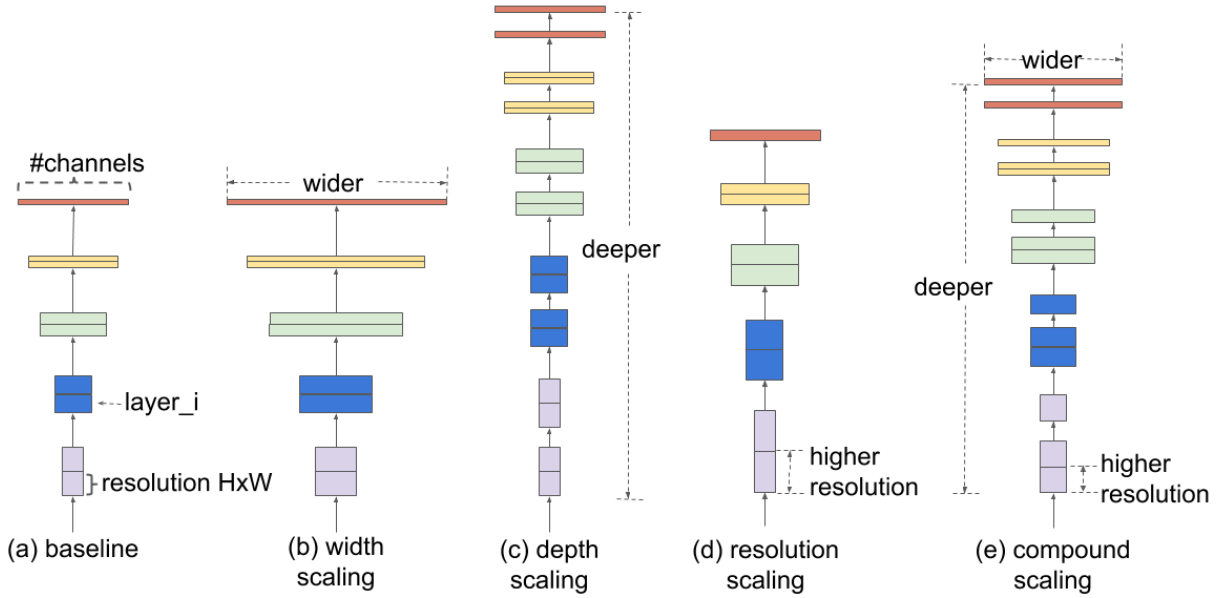


ImageNet Classification top-5 error (%)

- 1X1 연산을 전후로 더하는 Bottleneck 구조를 취해 퍼포먼스를 높임
- 3.57%의 top 5 error를 보임

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

EfficientNet



- ❖ Depth, Width, Resolution을 동시에 고려한 Compound Scaling을 통해 ConvNet기반 이미지 분류 성능을 개선한 모델
- ❖ 목적: 이미지 분류
- ❖ 개선방향 : Depth, Width, Resolution 동시에 키워서 성능 개선
  - ✓ Width(필터의 개수 늘림), Depth(레이어 수 늘림), Resolution(이미지 해상도 키움)
- ❖ 성과: 연산은 줄이고, 정확도는 높임

Tan, M. and Le, Q., 2019, May. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

Recent Developed Deep Learning Models

DenseNet (2017) ➔ SeNet(2017-2018) ➔ EfficientNet (2018-2019) ➔ Self-training & Noisy Student (2020) ➔ Meta Pseudo Labels (2021) ➔ ViT (2022)