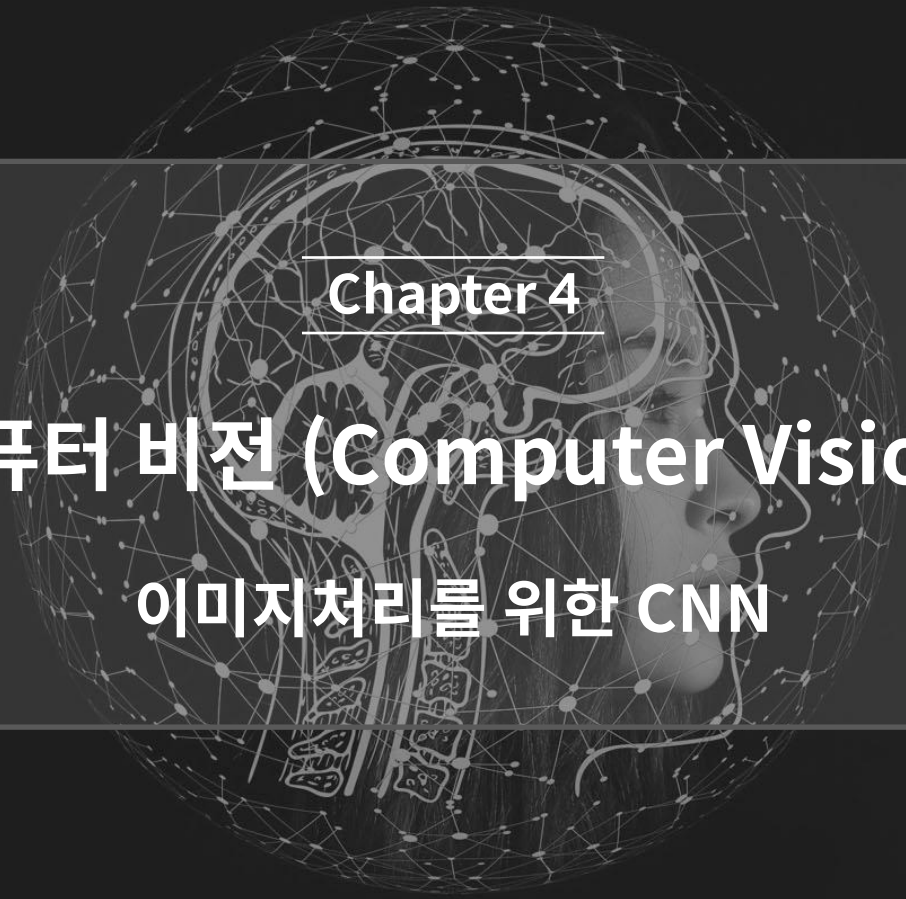


Chapter 4

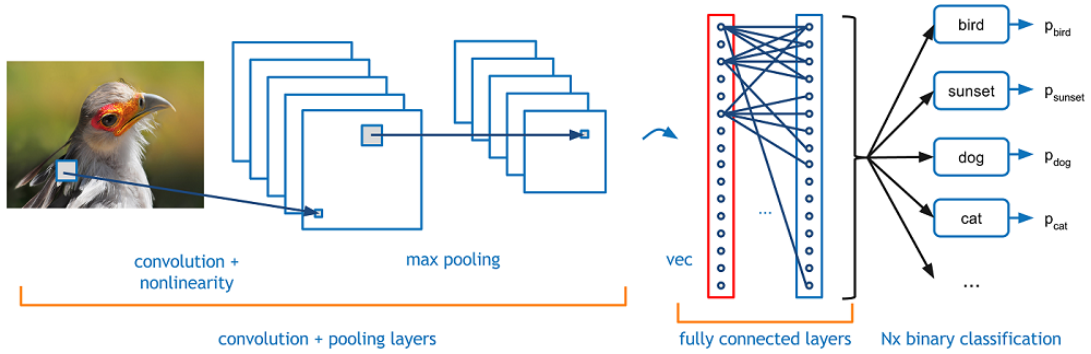
컴퓨터 비전 (Computer Vision)

이미지처리를 위한 CNN

Convolutional Neural Network

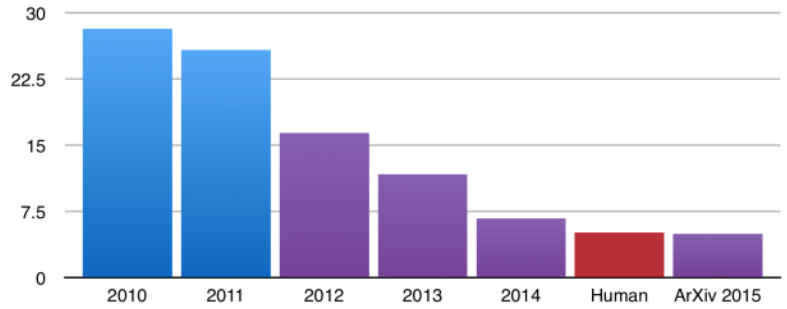
Deep Learning의 종류

- 이미지 처리에 많이 쓰이는 Convolutional Neural Network(CNN)



기본적인 CNN 구조

고양이와 강아지를 분류하는 모델



년도 별 ImageNet데이터 분류 모델 성능

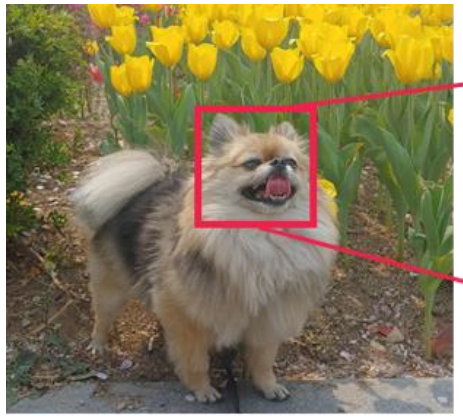
출처 : https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

출처 : https://excelsior-cjh.tistory.com/177

Convolutional Neural Network

이미지 분류에서 기존 Machine Learning 알고리즘의 한계

- 특정 이미지를 컴퓨터에게 보여주고, 이미지를 분류하는 모델을 만들 때에 우리는 이미지를 구성하는 Pixel값을 Input으로 사용.
- 일반적으로 Machine Learning을 할 때 에는 Input 변수들이 서로 독립이라는 가정이 있음. 그래서 독립변수 (Independent Variable)라고 부름.
- 하지만 이미지의 Pixel값이 서로 독립이 아니라는 것을 직관적으로 봐도 알 수 있음.
- 아래 그림과 같이 이미지 Pixel값들을 독립변수로 쭉 늘어뜨려서 Input으로 사용하는 것은 일반적인 Machine Learning의 가정에 위배되는 행위
- 현실적으로 이렇게 밖에 사용할 수 있는 방법이 없었고 어느정도 성능이 나오기는 하였기 때문에 이렇게 이미지의 2차원 Pixel값들을 Vector로 Flatten시켜 Input으로 사용 해왔음



$$\begin{matrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{11} & x_{12} & x_{13} \\ x_{11} & x_{12} & x_{13} \end{matrix}$$

Image1
Image2
…

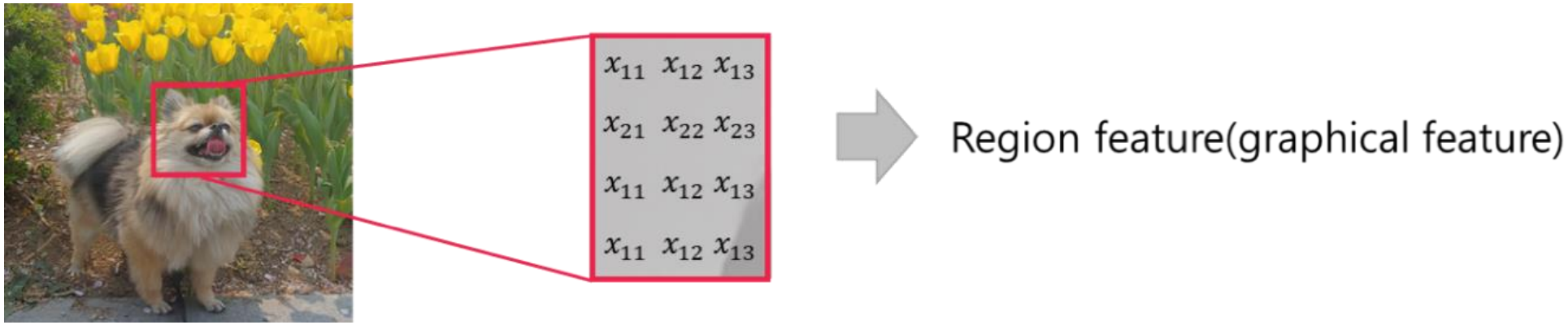
V1	V2	V3	V4	V5 …	V11	V12	V13	Y
x_{11}	x_{12}	x_{13}	x_{14}	x_{15} …	x_{11}	x_{12}	x_{13} …	개 고양이 사람

기존 Machine Learning

Convolutional Neural Network

Region Feature

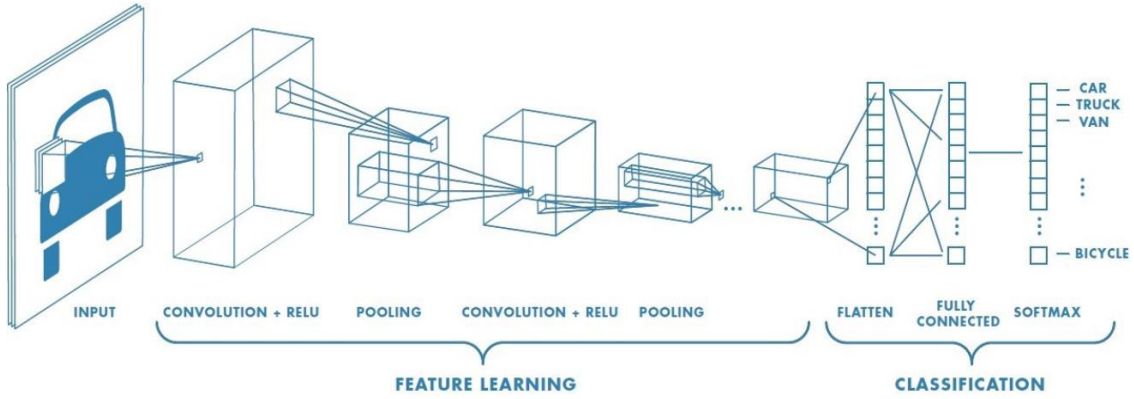
- x11과 x21 Pixel은 굉장히 가까운 위치에 있는 Pixel이지만 그러한 정보를 전혀 반영하지 못함.
- 이러한 지역 정보 (Region Feature) 를 학습할 수 있는 신경망 구조가 필요했고, 이러한 신경망 구조를 Convolutional Neural Network (CNN) 라고 함.
- x11과 가까운 x12, x21, x22 Pixel의 가까운 정보까지 함께 학습 시키고자 하는것.



Convolutional Neural Network

Convolutional Neural Network (CNN)

- CNN은 앞서 언급한 것처럼 Region Feature (Graphical Feature) 를 학습시키기 위한 신경망 모형으로서 Yann LeCun교수가 1998년에 제안.
- 당시 Yann LeCun교수는 이용하여 숫자 손 글씨를 분류하는 CNN모형을 제안하였으나, 학습시간이 너무 오래 걸리고 앞서서 언급했던 신경망 모형이 가지는 단점때문에 많이 부각 받지는 못했음.
- 그러나 2010년대에 들어서, 알고리즘과 하드웨어 발전으로 CNN이 이미지 처리하는 신경망 모형으로 급부상하게 되었음.
- 딥러닝하면 가장 먼저 떠오르는 CNN모형은 새롭게 개발된 모형이 아니라, 이전부터 있었던 모델이며 여러가지 요건들이 맞춰지면서 이제서 빛을 발하는 모델인 것.
- CNN은 기본적으로 Region Feature를 뽑아내는 Convolution Layer와, Feature Dimension을 위한 Pooling Layer 그리고 최종적인 분류를 위한 (일반적인 MLP구조를 가지는) Fully Connected Layer로 구성이 되어있음.

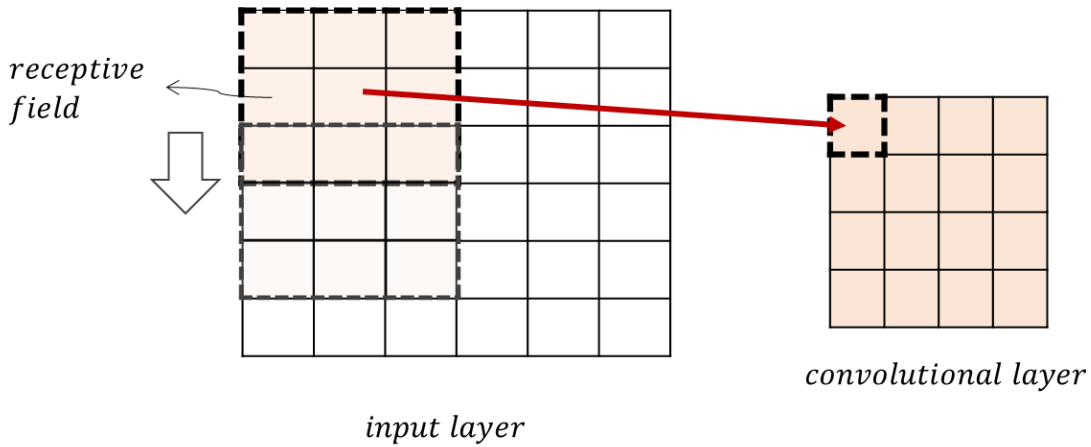


출처 : https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional Neural Network

Convolution Layer

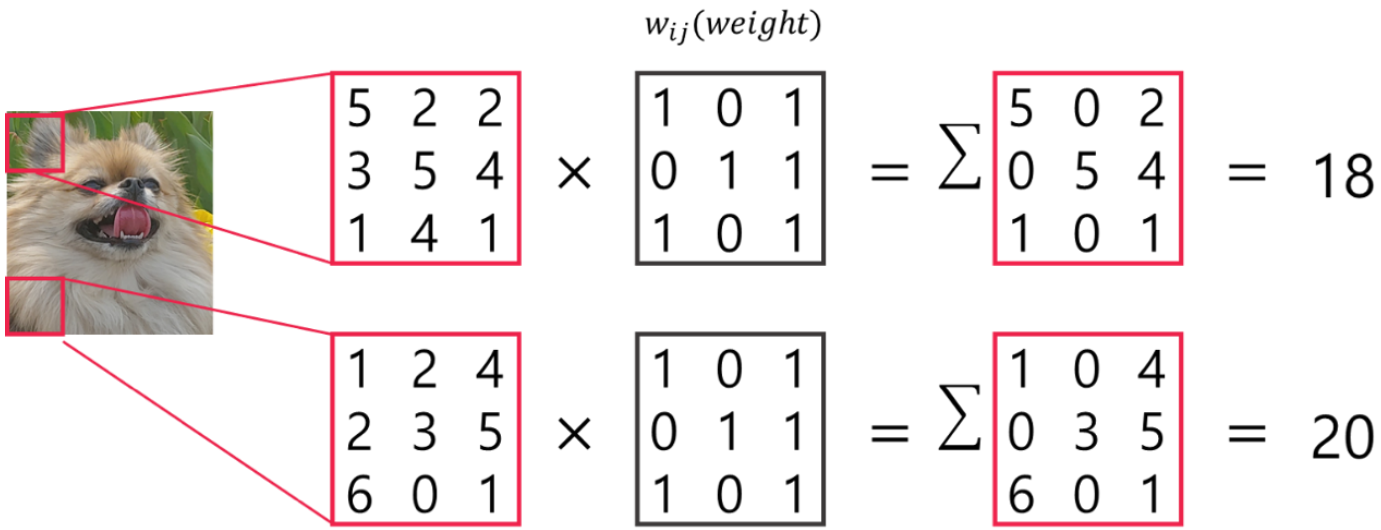
- Convolutional Layer는 Receptive Field를 정의하여 입력 층의 이미지의 Feature를 추출하는 역할을 담당
- 에서 이미지가 Input으로 들어왔을 때 사각형 모양의 Receptive Field가 이미지를 Scan하면서 이미지의 Region Feature를 추출.
- 이때 이미지 Pixel값과 Receptive Field의 Weight들과 선형 결합으로 하나의 값이 출력.



Convolutional Neural Network

Convolution Layer

- CNN의 학습은 이미지를 잘 분류 할 수 있도록 Feature를 추출해내는 Weight를 학습 시키는 것

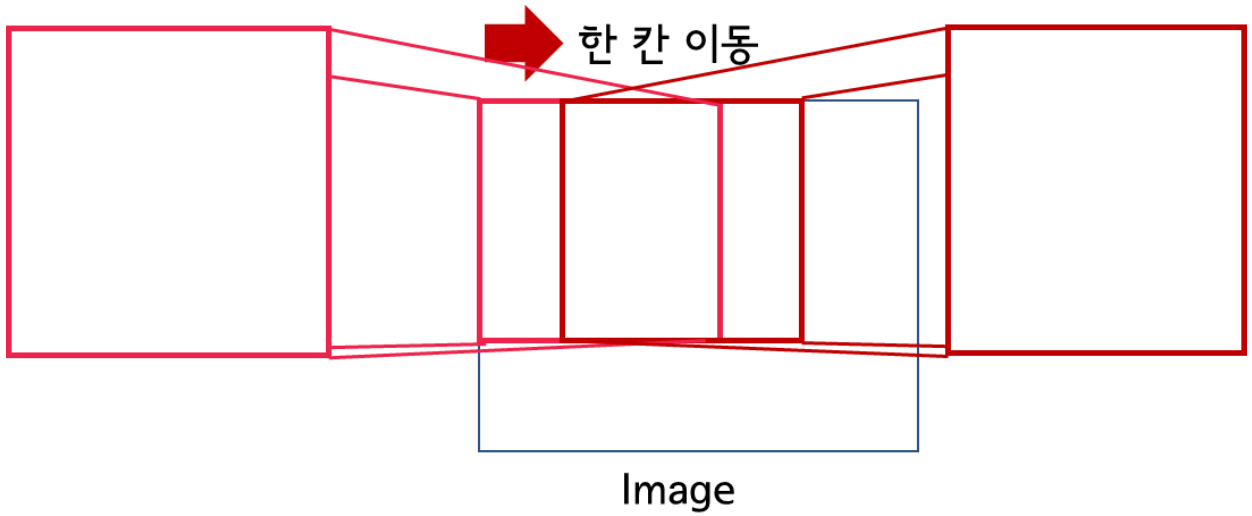


Convolutional Neural Network

Convolution Layer

- Stride

- Convolution Layer에서 Receptive Field가 이미지를 돌면서 Feature를 뽑을 때 아래 그림처럼 이동하는 칸 수를 의미
- 아래그림에서 Stride는 1

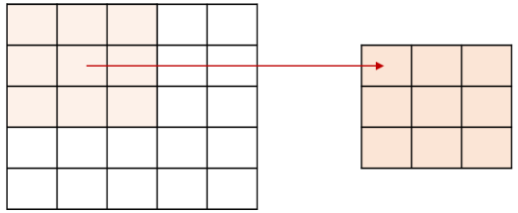


Convolutional Neural Network

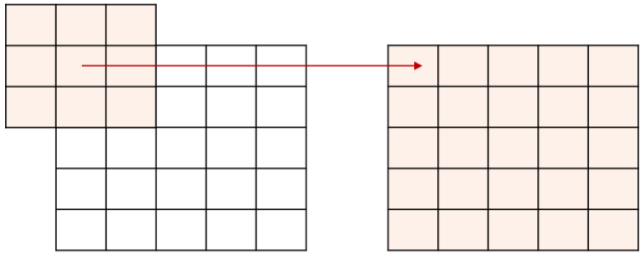
Convolution Layer

- Padding

- 아래 그림의 상단 그림처럼 5 x 5 Image에 3 x 3 Convolution을 적용하면 3 x 3 Image로 줄어 듦. 이와 같이 일반적인 Convolution을 적용하면 다음 Image또는 Feature의 Size가 줄어들게 되고, 가장 자리에 있는 Pixel값들은 안쪽에 있는 Pixel값 보다 적게 Convolution이 적용되는 단점이 있음.
- Image Size를 줄이지 않고 모든 Pixel값에 같이 Convolution을 적용하기 위해, 아래 그림처럼 Padding이라는 개념을 적용시킴.
- 기본 Image Size 테두리에 0값을 넣어, Image의 Size를 유지하고 테두리에 있는 Pixel값도 안에 있는 Pixel값과 똑같이 Convolution을 거치도록 하는 것.



Image



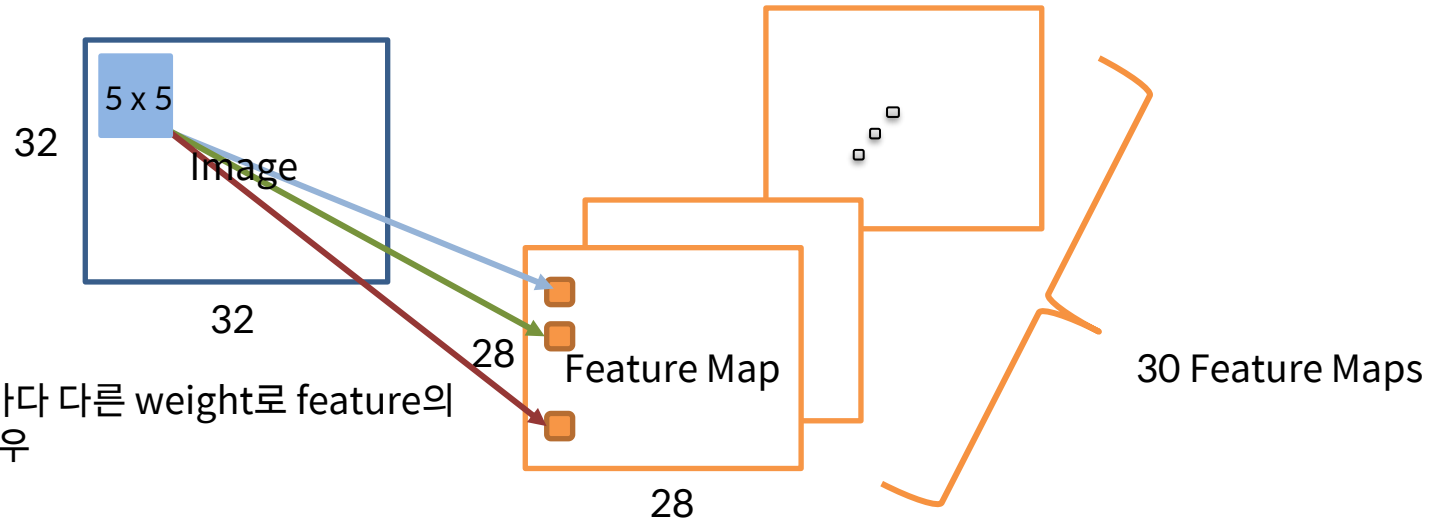
Image

Convolutional Neural Network

Convolution Layer

- Weight Sharing

- 하나의 feature map에 대하여 동일한 weight값을 적용



Receptive field가 slide할 때 마다 다른 weight로 feature의 노드에 연결되었다고 가정할 경우

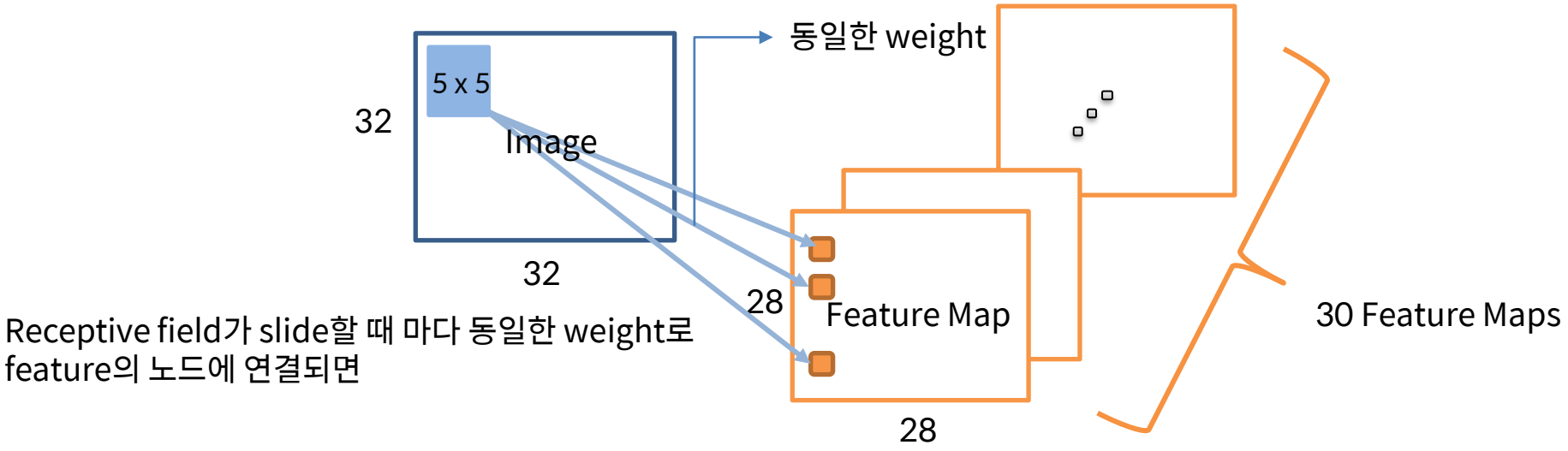
학습 해야 할 parameter 수((5 x 5) x (28 x 28 x 30) = 588,000)가 너무 많음

Convolutional Neural Network

Convolution Layer

- Weight Sharing

- 하나의 feature map에 대하여 동일한 weight값을 적용

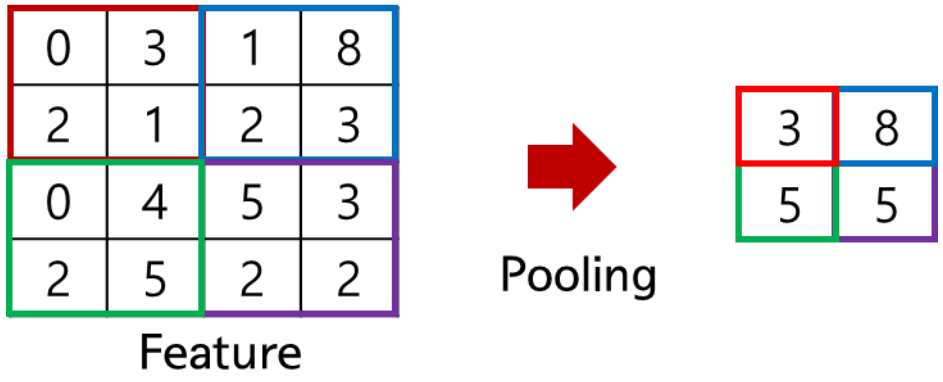


학습 해야 할 parameter 수 = (5 x 5) x (30) = 750
이전에 비해 학습 해야 할 parameter 수를 대폭 줄일 수 있음

Convolutional Neural Network

Pooling Layer

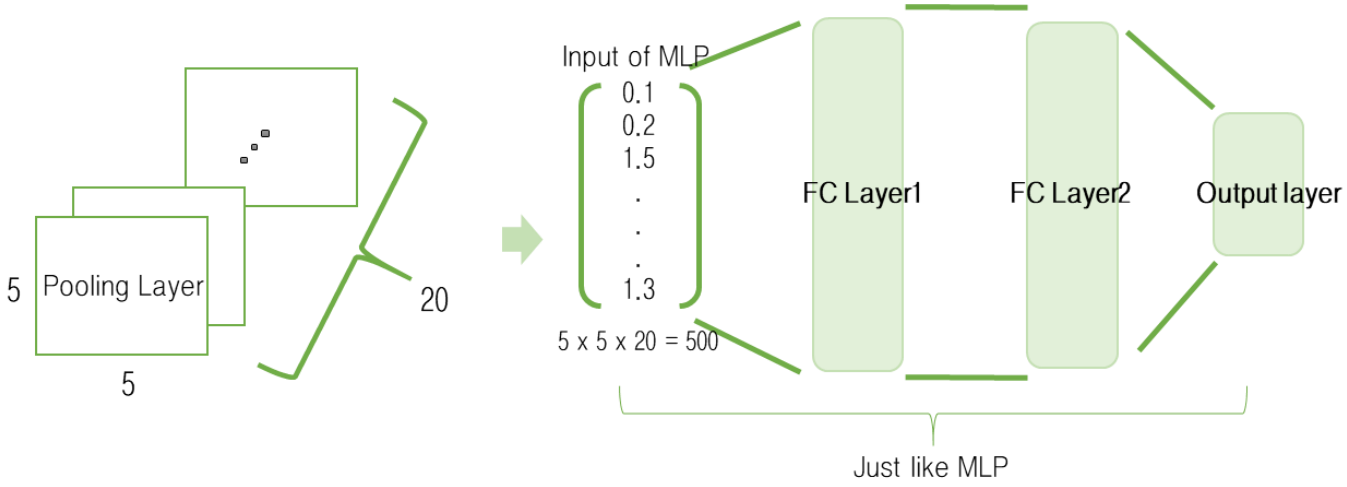
- 일반적으로 Image 또는 Feature에 Convolution을 거친 이후에 Pooling layer를 거침.
- Pooling Layer는 CNN의 학습속도를 향상 시키기 위해 Feature의 Dimension을 줄이는 개념. 사각형안의 Pixel값을 뽑으면 Max Pooling, 평균 Pixel값을 뽑으면 Average Pooling이라고 함
- Pooling의 목적은 CNN의 학습 속도를 높이기 위하여 Feature의 Dimension을 줄이는 것이기 때문에, 당연히 정보의 손실이 있을 수 밖에 없음.
- 최근에는, 최대한 정보를 활용하고 학습 속도를 높일 수 있는 알고리즘들이 많이 개발 되어서 상황에 따라서 Pooling Layer를 쓰지 않는 경우도 있음. 그렇지만, 일반적인 CNN모델을 디자인 할 때에 기본 적으로 Convolution - Pooling Layer를 번갈아 가면서 설계.



Convolutional Neural Network

Fully Connected Layer

- Fully Connected Layer는 일반적인 MLP구조와 동일.
- 아래 그림처럼 Pooling Layer에 나온 Feature들을 Flatten 시켜서 (풀어 헤쳐서) MLP의 Input으로 놓고 학습을 진행합니다.



Convolutional Neural Network

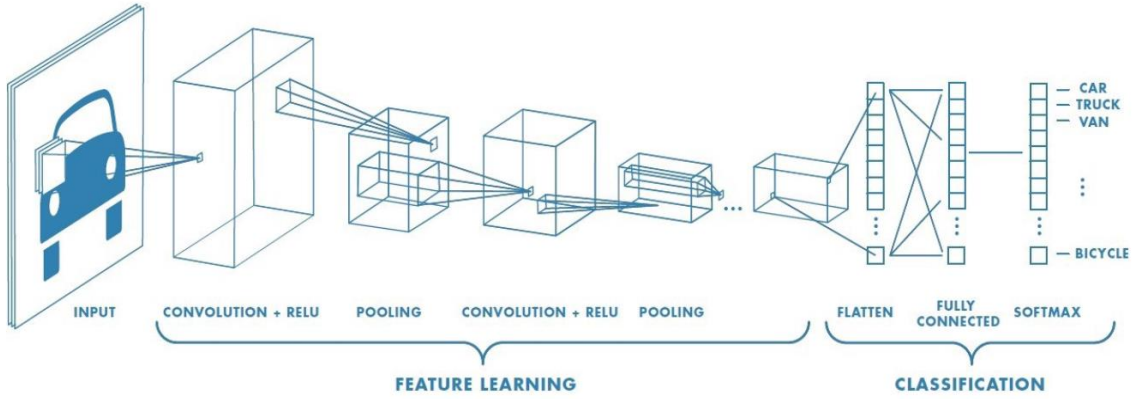
CNN의 모델 설계

- CNN의 Convolution의 Receptive의 Field의 크기와 Stride, Pooling의 종류, Layer를 쌓는 횟수 등 모두 사용자가 지정 해 주어야 하는 Hyperparameter임.
- '이렇게 Hyperparameter를 Tuning하면 잘된다'라는 것은 정해져 있지 않음.
- Parameter의 종류가 많아지면 과적합에 걸릴 확률 높고, 복잡한 모델에 적합하다라는 일반적인 머신 러닝의 개념을 가지고 디자인.
- '개와 고양이를 분류하는 단순한 모델이라면 Convolution Layer를 두개만 쌓으면 되고, Filter수는 많게 가져가지 않아도 되겠구나'라는 정도만 알고 디자인 하는 것이지, 최적의 Hyperparameter를 알기는 어려움. 다양한 시도를 통해 최적의 모델을 만들어가는 것이 중요.

Convolutional Neural Network

Convolutional Neural Network (CNN)

- CNN은 앞서 언급한 것처럼 Region Feature (Graphical Feature) 를 학습시키기 위한 신경망 모형으로서 Yann LeCun교수가 1998년에 제안.
- 당시 Yann LeCun교수는 이용하여 숫자 손 글씨를 분류하는 CNN모형을 제안하였으나, 학습시간이 너무 오래 걸리고 앞서서 언급했던 신경망 모형이 가지는 단점때문에 많이 부각 받지는 못했음.
- 그러나 2010년대에 들어서, 알고리즘과 하드웨어 발전으로 CNN이 이미지 처리하는 신경망 모형으로 급부상하게 되었음.
- 딥러닝하면 가장 먼저 떠오르는 CNN모형은 새롭게 개발된 모형이 아니라, 이전부터 있었던 모델이며 여러가지 요건들이 맞춰지면서 이제서 빛을 발하는 모델인 것.
- CNN은 기본적으로 Region Feature를 뽑아내는 Convolution Layer와, Feature Dimension을 위한 Pooling Layer 그리고 최종적인 분류를 위한 (일반적인 MLP구조를 가지는) Fully Connected Layer로 구성이 되어있음.

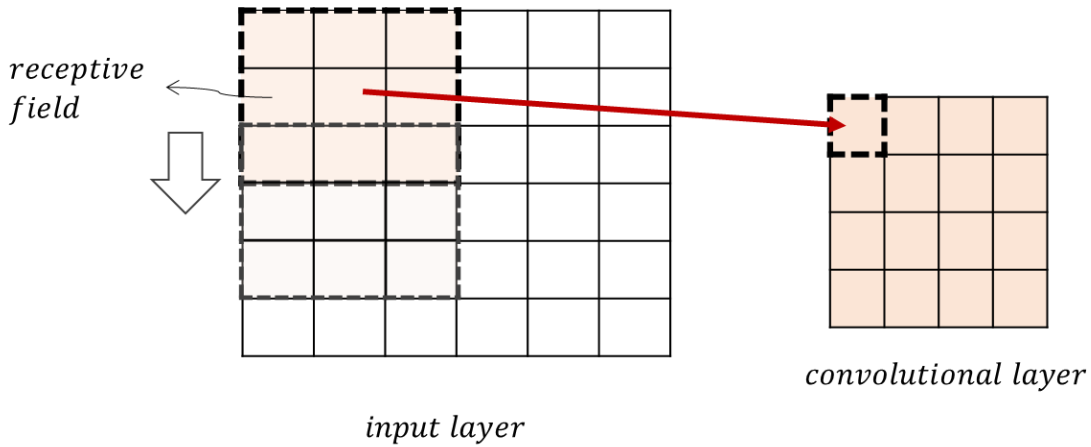


출처 : https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional Neural Network

Convolution Layer

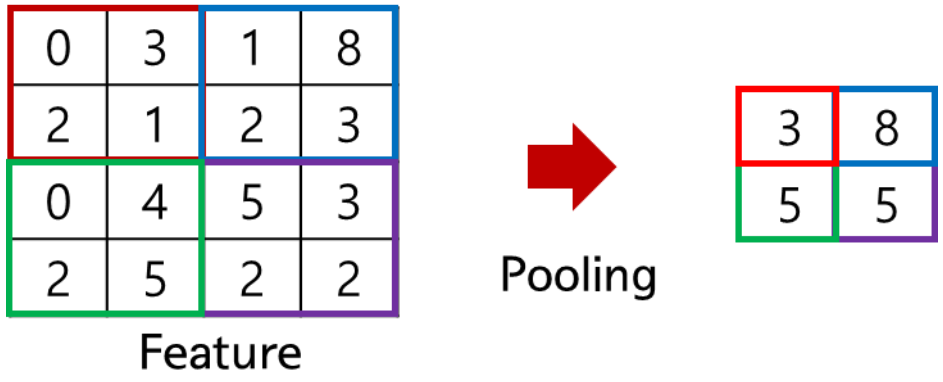
- Convolutional Layer는 Receptive Field를 정의하여 입력 층의 이미지의 Feature를 추출하는 역할을 담당
- 에서 이미지가 Input으로 들어왔을 때 사각형 모양의 Receptive Field가 이미지를 Scan하면서 이미지의 Region Feature를 추출.
- 이때 이미지 Pixel값과 Receptive Field의 Weight들과 선형 결합으로 하나의 값이 출력.



Convolutional Neural Network

Pooling Layer

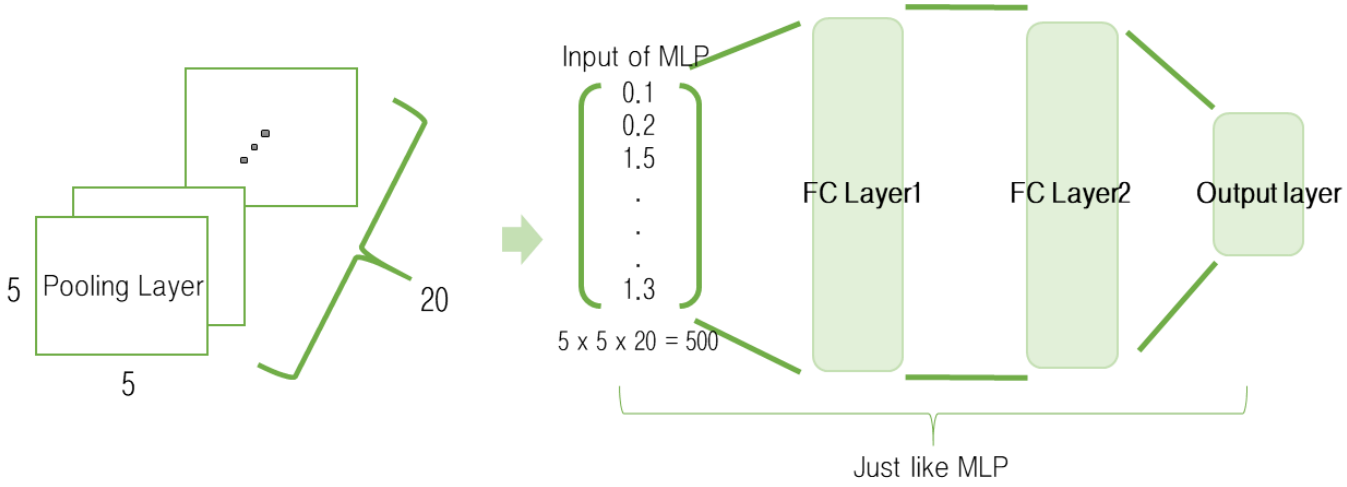
- 일반적으로 Image 또는 Feature에 Convolution을 거친 이후에 Pooling layer를 거침.
- Pooling Layer는 CNN의 학습속도를 향상 시키기 위해 Feature의 Dimension을 줄이는 개념. 사각형안의 Pixel값을 뽑으면 Max Pooling, 평균 Pixel값을 뽑으면 Average Pooling이라고 함
- Pooling의 목적은 CNN의 학습 속도를 높이기 위하여 Feature의 Dimension을 줄이는 것이기 때문에, 당연히 정보의 손실이 있을 수 밖에 없음.
- 최근에는, 최대한 정보를 활용하고 학습 속도를 높일 수 있는 알고리즘들이 많이 개발 되어서 상황에 따라서 Pooling Layer를 쓰지 않는 경우도 있음. 그렇지만, 일반적인 CNN모델을 디자인 할 때에 기본 적으로 Convolution - Pooling Layer를 번갈아 가면서 설계.



Convolutional Neural Network

Fully Connected Layer

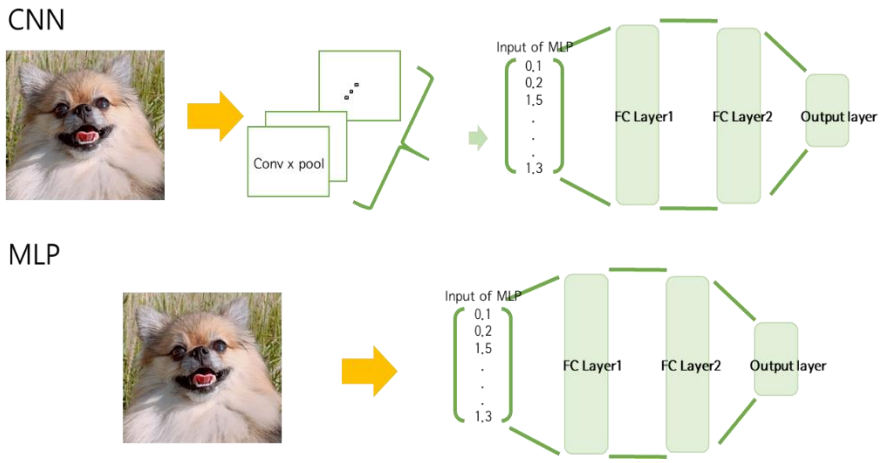
- Fully Connected Layer는 일반적인 MLP구조와 동일.
- 아래 그림처럼 Pooling Layer에 나온 Feature들을 Flatten 시켜서 (풀어 헤쳐서) MLP의 Input으로 놓고 학습을 진행합니다.



Convolutional Neural Network

CNN과 MLP

- 이미지를 분류하는 모델을 만들고자 할 때 CNN과 MLP의 가장 큰 차이점은 이미지의 Feature를 어떻게 추출 하느냐에 있음
- MLP는 이미지의 Pixel값을 바로 Network의 Input으로서 사용하는 것이고, CNN은 이미지의 Region Feature를 Convolution Layer와 Pooling Layer를 통해 추출하고 그 Feature들을 MLP의 Input으로 사용하는 것.
- CNN이 이미지 처리와 관련된 Vision 쪽에 성능이 좋은 이유는 단지 Region Feature를 추출 할 수 있기 때문.
- 여기서 알 수 있는 중요한 사실은 딥러닝이라고 하는 것은 새로운 모델이라는 개념보다 이렇게 Graphical Feature를 추출하는 것에 초점을 맞춘 모델이며, 모델을 만들 때에 Feature는 매우 중요하다 라는 사실.



Data Augmentation

Data Augmentation

- 복잡한 모델을 만들기 위해서는 다량의 데이터를 필요로 함
- CNN을 포함한 딥러닝 모델들은 Graphical Feature를 학습시키는 것이 주 목적이고 복잡한 문제를 풀기 위한 복잡한 모델이 대부분.
- 아래 그림은 같은 강아지 사진이지만, Convolution Layer의 Receptive Field를 생각해보면 같은 사진이지만 조금은 다르게 Feature가 뽑힐 거라는 것도 알 수 있음
- 데이터를 임의로 변형시켜 데이터의 수를 늘려 다양한 Feature를 뽑는 방법을 Data Augmentation이라고 함
- 일반적으로 이미지 분류 문제에서 Data Augmentation을 할 경우에 성능이 소폭 상승한다고 알려져 있음

Data Augmentation

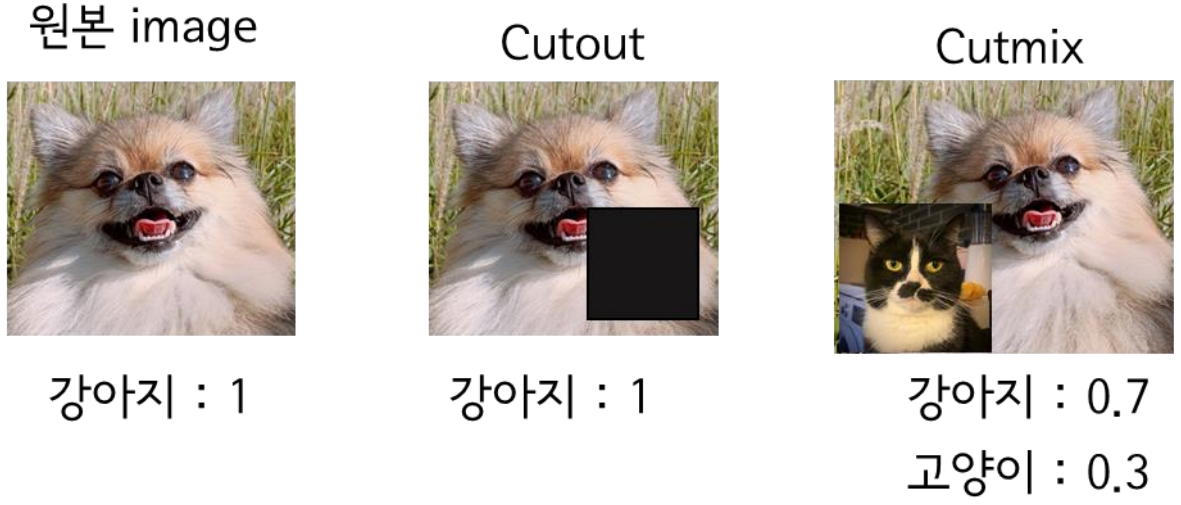
자주 사용하는 Data Augmentation

- Random Flip : Flip은 반전을 의미, 이미지를 랜덤하게 좌우 또는 상하 반전 시키는 기법
- Rotation : 이미지를 회전 시키는 기법
- Crop : 이미지의 일정 부분을 잘라버려서 사용하는 기법
- Scaling : 이미지를 확대 또는 축소 시키는 기법

Data Augmentation

자주 사용하는 Data Augmentation

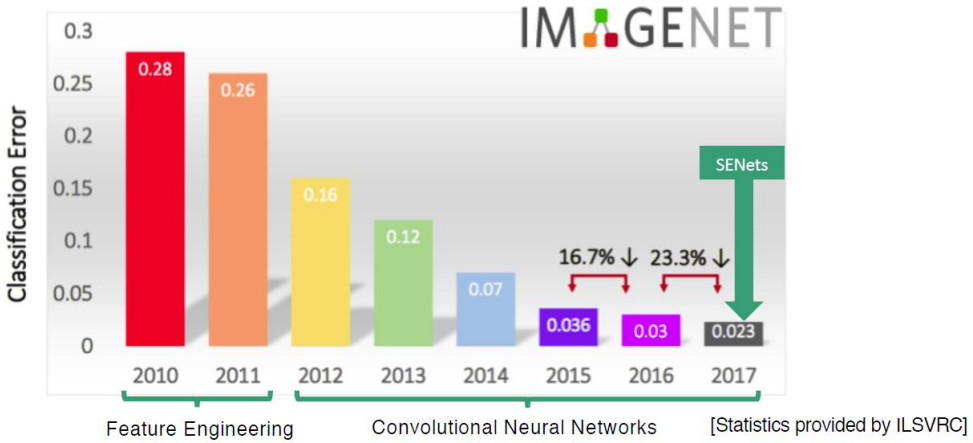
- Cutout : 아래 그림의 두번째 그림처럼 이미지의 일부를 사각형 모양으로 검은색을 칠해버리는 기법. 숫자로는 0을 채워 넣는 것이라고 생각 할 수 있음. 일종의 Input 데이터에 대해 Dropout을 적용한 기법이라고 이해 할 수 있음
- Cutmix : 아래 그림의 세번째 그림처럼 두 이미지를 합쳐 놓고, 이미지의 label을 학습 시킬 때 각각의 이미지가 차지하는 비율만큼 학습 시키는 방법. 아래 예시에는 전체 이미지에서 강아지가 차지하는 비율이 70%, 고양이가 차지하는 비율이 30%이기 때문에, 학습 시킬 때에 강아지는 0.7, 고양이는 0.3으로 Labeling하여 학습을 진행
- Cutout과 Cutmix 모두 일반적인 이미지 분류 문제에서 Data Augmentation보다 성능이 뛰어나다는 것이 논문을 통해 밝혀져 있음



다양한 CNN구조

Network Architecture

- Network의 성능을 높이기 위한 여러가지 방법(Overfitting이나 Gradient Vanishing을 방지하기 위하여, Activation Function, Batch Normalization, Drop Out, Initialization, Data Augmentation 등) 외에도 네트워크를 깊게 쌓으면서 Overfitting을 방지 시키고 성능을 높이기 위한 노력이 계속 되어옴
- ImageNet : 이미지 분류 모델을 측정하기 위한 데이터로 가장 많이 사용되는 데이터 셋. 학습 데이터만 총 138GB로서 총 2만개 이상의 클래스와 약 천 4백만장의 이미지로 구성되어 있음
- ImageNet 데이터를 가지고 이미지 분류 모델 대회를 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 라고 함.

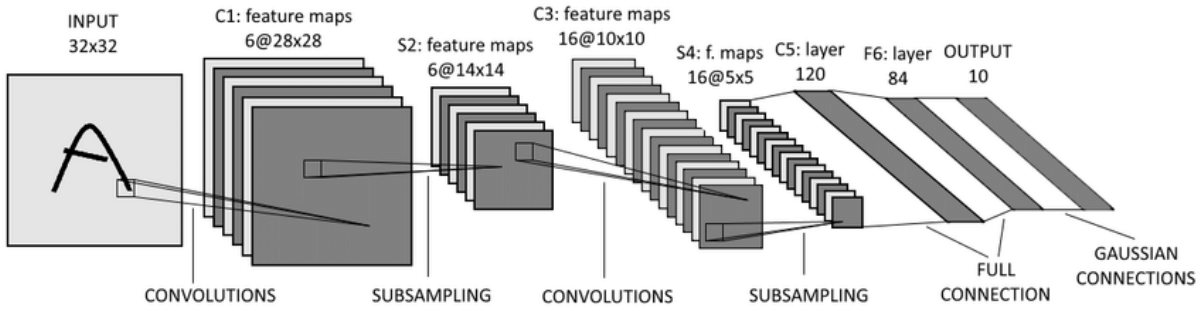


출처 : https://towardsdatascience.com/review-senet-squeeze-and-excitation-network-winner-of-ilsvrc-2017-image-classification-a887b98b2883

다양한 CNN구조

LeNet

- Yann LeCun 교수가 제안한 최초의 CNN모델. 1990년대 당시에는 컴퓨팅 문제가 있었기 때문에, 비교적 단순한 구조를 가지고 있음. 32 x 32 input을 가지고 있으며 Convolution layer 2개, pooling layer 2개, Fully Connected Layer 3개를 가지고 있음, 이 구조는 가장 기본적인 CNN구조로서 사용되고 있음.

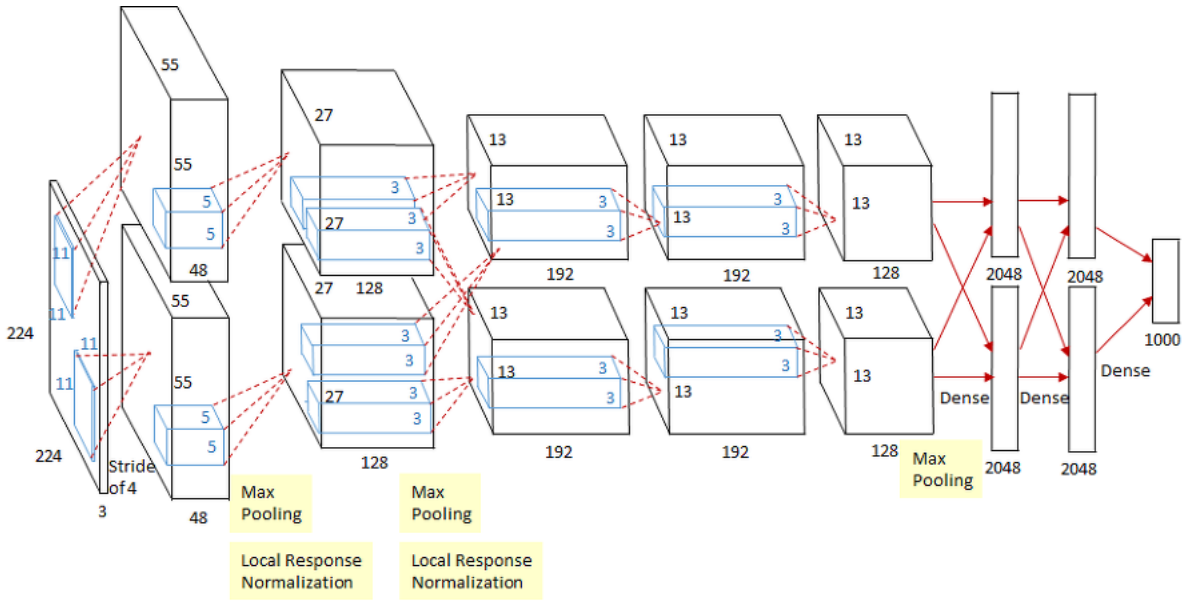


출처 : https://medium.com/deep-learning-g/build-lenet-from-scratch-7bd0c67a151e

다양한 CNN구조

AlexNet

- 2012 ILSVRC 대회 우승모델로서, 구조는 LeNet과 크게 다르지 않음. 224 x 224 크기의 RGB 3 Channel Image를 Input으로 사용하였으며, Activation Function은 ReLU를 사용하였으며, Dropout과 Data Augmentation등을 적용



출처 : https://bskyvision.com/421

다양한 CNN구조

VGG

- ILSVRC 2014대회에서 2위를 차지한 모델. AlexNet과 같은 이전의 모델과 다르게 처럼 3 x 3 convolution layer를 깊게 중첩 한다는 게 VGG의 큰 특징. Layer의 깊이에 따라 VGG16, VGG19등으로 불리우고 있음

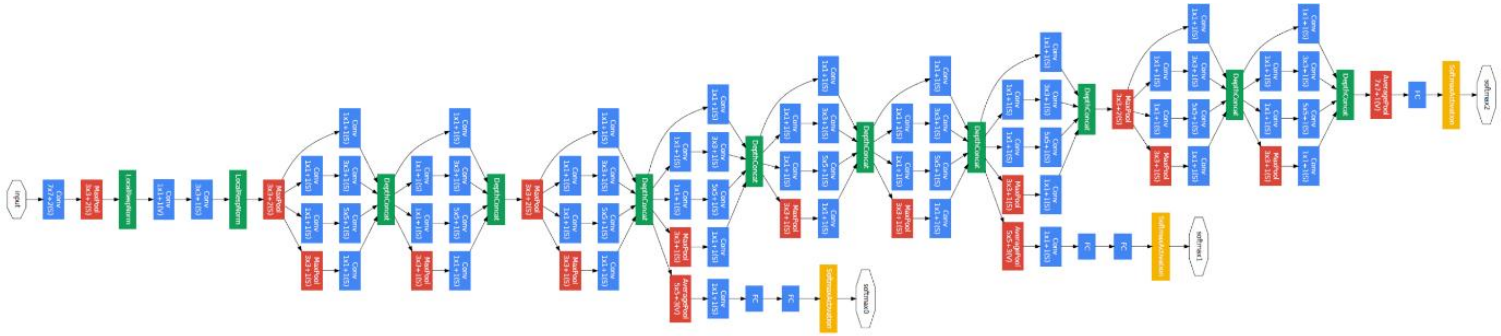
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

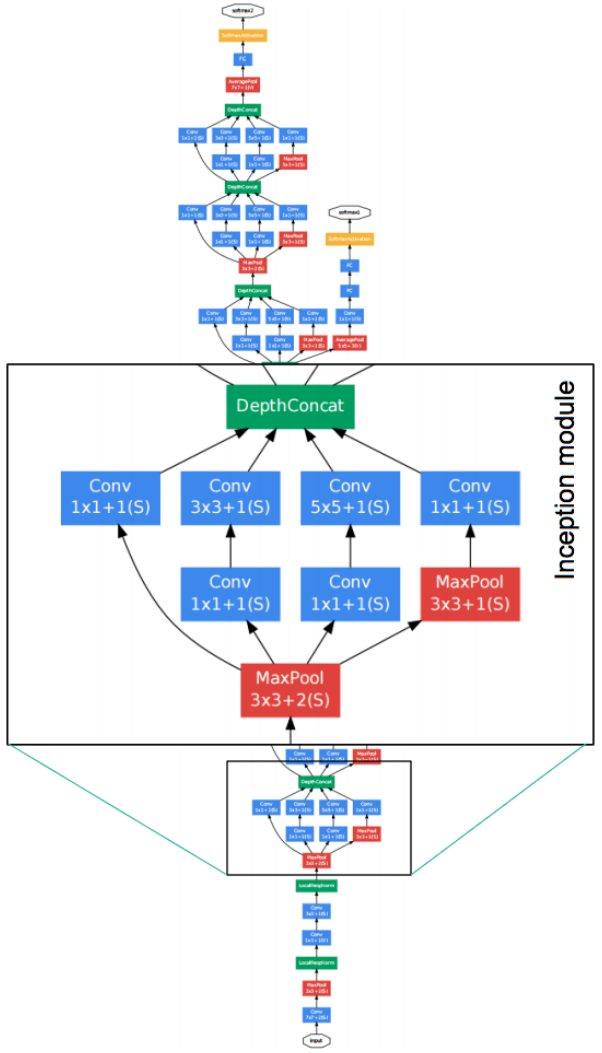
출처 : Very Deep Convolutional Networks for Large-Scale Image Recognition (https://arxiv.org/pdf/1409.1556.pdf)

다양한 CNN구조

GoogLeNet

- ILSVRC 2014 대회에서 1위를 한 모델로 Inception Model로도 불리움. Google 에서 제안한 모델로 Google + LeNet을 합쳐 이름이 지어짐
- 최초로 Inception Module제안



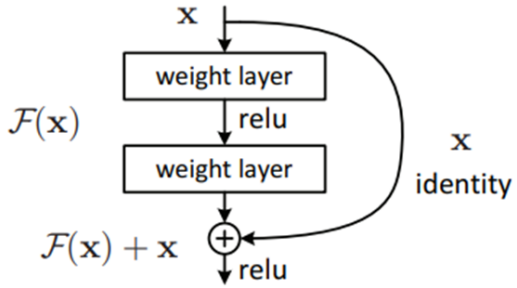


출처 : https://mc.ai/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet/

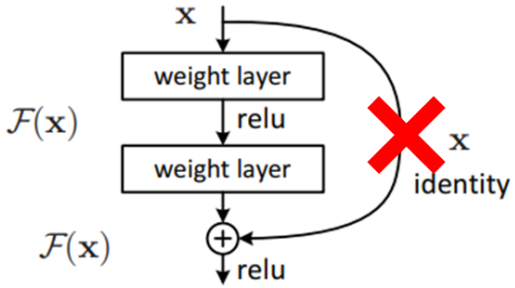
다양한 CNN구조

Residual Network(ResNet)

- ILSVRC 2015대회에서 1위를 차지한 모델이며, 지금까지도 이미지 분류의 정석 적인 기본 모델로서 널리 쓰이고 있음
- Residual Block이라는 개념을 도입하였으며, 아래 그림 처럼 이전 layer의 Feature Map을 다음 Layer의 Feature Map에 더해주는 개념. 이를 Skip Connection이라 함
- Network가 깊어지면 깊어짐에 따라서 앞 단의 Layer에 대한 정보는 뒤의 Layer에서는 희석 될 수 밖에 없음. 이러한 단점을 보완하기 위해 이전의 정보를 뒤에서도 함께 활용하는 개념이라고 이해할 수 있음
- ResNet의 구조는 널리 사용되고 있으며 많은 딥러닝 연구 논문에서도 기본 딥러닝 Frame으로서 활용되고 있음



Residual block



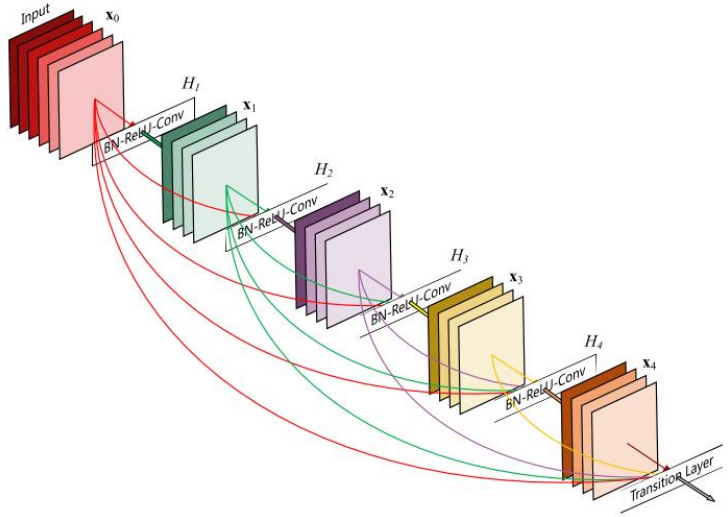
기본적인 network구조

출처 : Deep Residual Learning for Image Recognition (https://arxiv.org/pdf/1512.03385.pdf)

다양한 CNN구조

Dense Network(DenseNet)

- DenseNet은 ResNet의 확장된 version으로 ResNet은 이전 Layer와 다음 Layer에 대해 Skip Connection을 적용하는 모델이라면, DenseNet은 모든 Layer에 대해 Skip Connection을 적용하는 모델
- 첫번째 Layer에 대한 정보를 두번째, 세번째, 그리고 마지막 Layer에 대해서도 함께 학습을 진행 시켜주는 것.
- ResNet보다 DenseNet이 약간의 높은 성능을 기록하는 것으로 알려져 있음.
- 다만, DenseNet은 ResNet을 약간 변형한 모델이기 때문에, 학술적으로는 ResNet이 조 금더 가치가 있다고 여겨짐
- 이외에도 Network구조적으로 발전된 다양한 모델들이 많이 있으나, 가장 보편적으로 많이 사용하는 구조는 ResNet



출처 : Densely Connected Convolutional Networks (https://arxiv.org/pdf/1608.06993.pdf)

Transfer Learning

개와 고양이를 분류하는 모델

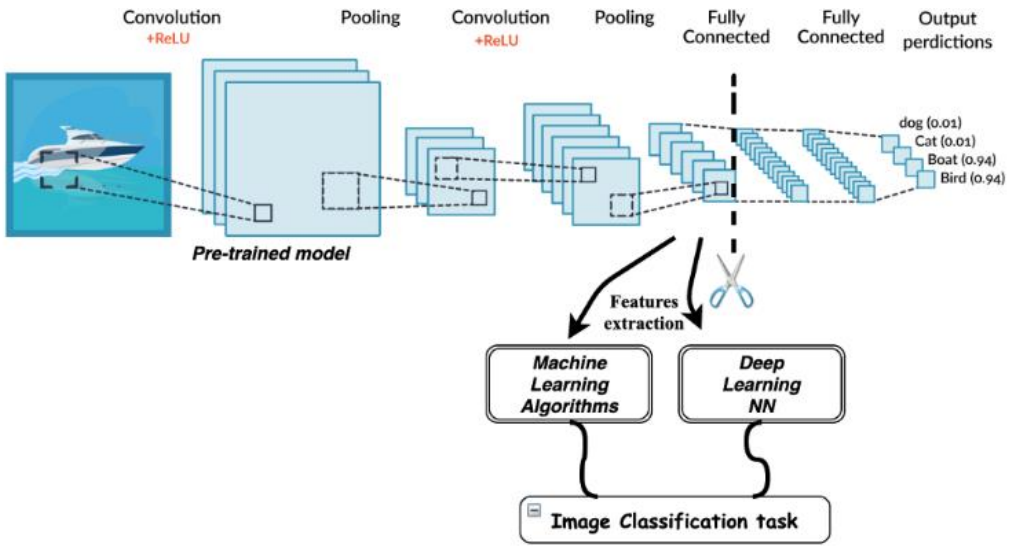
- 딥러닝을 구축하기 위해서는 수많은 데이터를 필요로 함.
- 그러나 우리가 가지고 있는 강아지와 고양이 사진은 약 100여장이 전부인 상황.
- 이러한 상황에서 딥러닝을 충분히 학습시키기는 쉽지 않음.
- 고양이와 강아지가 가지는 일반적인 Feature를 충분히 잘 학습하기가 어렵기 때문

VS

Transfer Learning

Transfer Learning

- 데이터가 적은 상황에서 ImageNet 데이터를 미리 학습해 놓은 딥러닝 모델 (Pre-Trained Model)을 가져와서 재학습 (Fine-tuning) 시키는 방법을 사용합니다. 이러한 방법은 전이 학습 (Transfer Learning) 이라고 함
- Pre-Trained Model을 Load한 다음 Fully Connected Layer 앞 단의 Network의 Weight를 가져오고 Fully Connected Layer는 우리가 디자인. 이때, Fully Connected Layer도 그대로 사용하고 Output Layer만 디자인 하기도 함

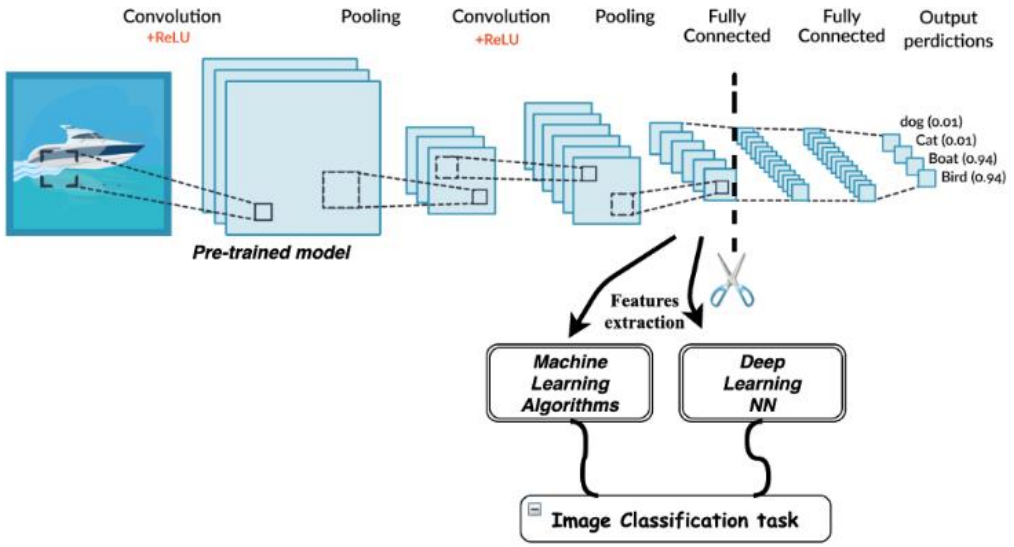


출처 : https://mc.ai/transfer-learning-with-deep-learning-machine-learning-techniques/

Transfer Learning

Transfer Learning

- Pre-Trained Model은 우리가 분류하고자 하는 문제 (개 vs 고양이)보다 훨씬 큰 문제를 푸는 모델이기 때문에 Output Layer의 Dimension을 수정 해 주어야 함.
- 우리가 보유한 데이터를 Input으로 하여 학습을 진행.
- 일반적으로, Pre-Trained Model의 Fully Connected Layer이전의 Weight는 학습을 시키지 않음. 이를 Weight를 Freezing한다라고 표현을 하며, 우리가 보유한 데이터를 가지고는 Fully Connected Layer 부분의 Weight만 학습을 진행을 하는 것. 이 과정을 Fine-tuning이라고 함.



출처 : https://mc.ai/transfer-learning-with-deep-learning-machine-learning-techniques/

Transfer Learning

Transfer Learning

- Transfer Learning은 Initialization의 개념으로 바라볼 수 도 있음.
- 앞서 Network의 높은 성능과 빠른 수렴을 위해서 Network의 Weight를 Initialization기법들이 연구되어 오고 있다고 서술한 바 있음.
- Transfer Learning은 결국 내가 학습하고자 하는 모델의 초기 Weight를 Pre-Trained Model의 Weight를 사용하는 것과 같기 때문에, Initialization기법으로 바라볼 수도 있음.
- 일반적으로, 데이터가 많지 않은 상황에서 보유하고 있는 데이터만을 가지고 학습 시켰을 때 보다 Transfer Learning을 시켰을 때 모델의 성능이 높음.