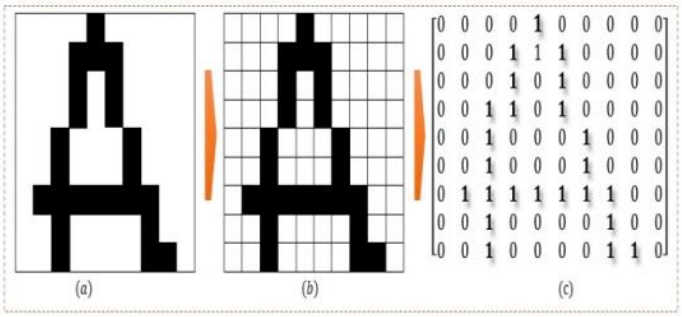
딥러닝팀 2 주차 클린업

1. 이미지 데이터의 특징

1.1. 컴퓨터에서의 이미지 데이터

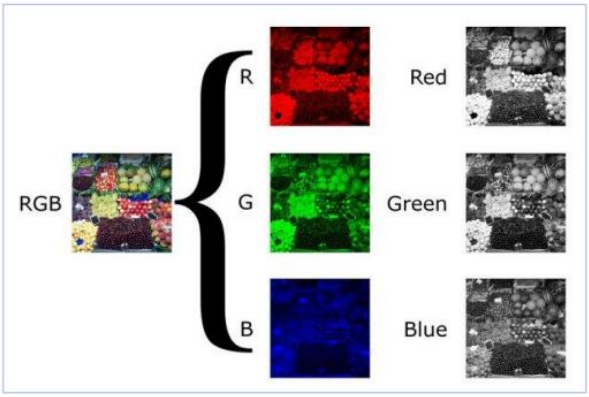


우리가 컴퓨터를 통해 보고 있는 모든 이미지는 픽셀(pixel)이라고 불리는 사각형의 점으로 이루어져 있습니다. 고화질의 이미지일수록 많은 픽셀로 이미지가 구성됩니다. 모니터에서 FHD는 1920*1080개의 픽셀을 가지고 있고, QHD는 2560*1440개의 픽셀을 가지고 있어 QHD가 FHD보다 좋은 화질을 가지고 있습니다.

만약 흑백 이미지라면 픽셀을 표현할 때 위 그림과 같이 해당 위치가 검정색이면 1, 흰색이면 0으로 나타낼 수 있습니다. 우리는 이제 숫자를 이용해 이미지를 행렬의 형태로 표현할 수 있습니다. 이런 식으로 표현한 이미지를 비트맵 이미지라고 합니다.

1.2. 채널

이미지 데이터는 빛의 3원색인 R, G, B 3개의 채널로 이루어져 있으며, 각 채널의 값들이 어떤 범위를 가지는지에 따라 색의 선명도가 결정됩니다.

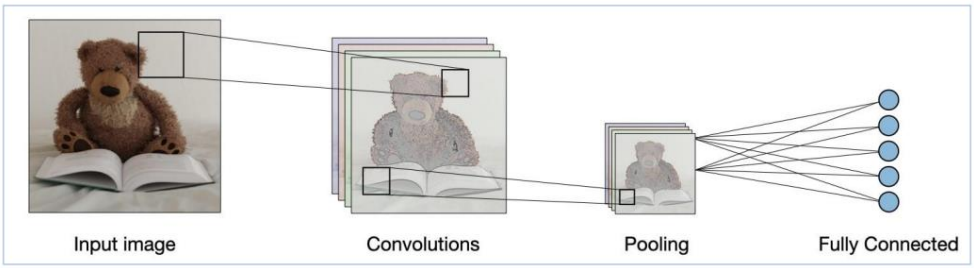


왼쪽의 과일 사진을 R채널, G채널, 그리고 B채널로 분리한다면 오른쪽과 같이 나타낼 수 있습니다. 과일 사진의 경우 대부분 빨간색과 초록색으로 이루어져 있는 것을 확인하실 수 있습니다. 따라서 B채널은 매우 어두운 반면, G채널은 상대적으로 밝은 것을 확인할 수 있습니다. 또한, Grayscale에서 R채널과 G채널이 밝은 것을 확인하실 수 있습니다.

이미지의 가로와 세로를 구성하는 픽셀의 수가 각각 256개이고 R, G, B 세개의 색으로 이루어진 경우 우리는 이 미지를 (256, 256)크기인 3개의 행렬로 표현할 수 있습니다. 이를 (Height, Width, Channel) 순으로 나열하여 (256, 256, 3)으로 표현했다고 말할 수 있습니다. Pytorch에서는 (C, H, W) 순으로 (3, 256, 256)이라고 표현합니다. 이렇게 컬러 이미지 역시 숫자를 이용해 행렬의 형태로 표현할 수 있습니다.

2. CNN (Convolutional Neural Network)

기존 신경망은 1차원 벡터만을 input으로 받아들입니다. 그래서 2차원 데이터인 흑백 이미지나 3차원 데이터인 컬러 이미지를 기존 신경망에서 사용하기 위해서는 1차원 벡터 형태로 변경해 주어야합니다. 하지만, 이 과정에서 이미지가 가지고 있는 공간적인 정보가 손실됩니다. 하지만 CNN 모델을 사용하게 되면 공간정보를 보존할 수 있습니다. 아래 그림을 통해서 그 의미를 파악해보도록 하겠습니다.

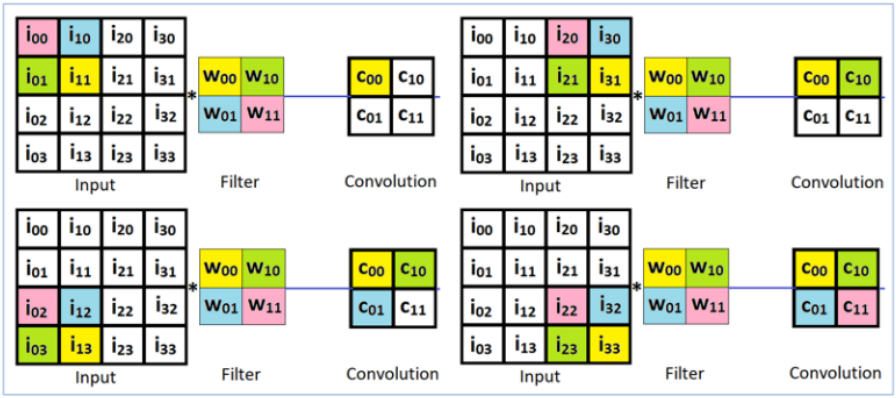


위 그림은 곰인형 사진을 CNN의 입력 데이터로 사용했을 때 데이터가 처리되는 과정을 도식화한 것입니다. 맨 왼쪽 Input image라고 되어있는 곰인형 사진에는 검은색 상자가 표시되어 있으며, 그 다음 Convolutions 라고 되어있는 단계에 있는 검은색 상자와 연결되어 있습니다. 여기서 Input image단계는 인공신경망의 입력층, Convolutions단계는 인공신경망의 은닉층의 하나에 해당한다고 볼 수 있습니다. 기존 인공신경망 모델에서는 입력이 가중치와의 선형결합을 통해 하나의 값으로 변환되었던 반면, 위 사진에서는 곰인형 사진이 입력층에서 다음 층으로 넘어갔음에도 검은색 상자의 위치가 다음 층에서도 그대로 유지되고 있음을 확인할 수 있습니다. 즉, 데이터의 공간 정보가 보존되고 있는 것입니다.

CNN 모델이 공간 정보를 보존할 수 있는 이유는 CNN 모델의 3가지 특별한 층(Layer)들 덕분입니다. 순서대로 Convolution Layer, Pooling Layer, 그리고 FC Layer(Fully Connected Layer)입니다. 결과적으로, CNN의 기본 아이디어와 CNN만의 특별한 층의 특성들을 고려했을 때, CNN은 공간 정보가 데이터의 값만큼 중요한 정보가 되는 형태의 데이터에 대해 아주 강력한 모델이 될 수 있습니다. 때문에, CNN은 위 예시와 같은 이미지 데이터, 그리고 이미지의 연속으로 볼 수 있는 영상 데이터 등에 활발하게 활용되고 있는 딥러닝 모델입니다. 지금부터는 Convolution Layer, Pooling Layer, 그리고 FC Layer를 순서대로 알아보면서 CNN 모델의 작동 방식을 구체적으로 살펴보겠습니다.

2.1. Convolutional Layer

Convolution Layer는 CNN의 가장 핵심적인 층입니다. 공간 정보의 보존이 CNN의 가장 큰 강점인 만큼, Convolution Layer는 Pooling Layer와 더불어 공간 정보를 보존하는 동시에 입력 데이터의 특징을 추출하는 역할을 합니다. 아래의 그림은 Convolution Layer의 작동 방식을 나타낸 그림입니다.



Convolution Layer에서는 필터라는 개념이 추가됩니다. 위 사진을 보면서 설명하도록 하겠습니다. 입력으로 이미지가 Convolution Layer에 들어오게 되면, 필터는 input에서 자신의 크기인 (2, 2) 사이즈만큼의 부분과 가중합하는 역할을 합니다. 위 사진에서는 같은 색으로 칠해진 i와 w가 곱해진 후 더해지는 연산입니다. 이를 수식으로 표현하면 다음과 같습니다.

$$i_{00} \times w_{11} + i_{10} \times w_{01} + i_{01} \times w_{10} + i_{11} \times w_{00} = c_{00}$$

그리고 이 (2, 2)사이즈의 필터는 (4, 4)사이즈의 input을 따라 오른쪽으로 움직이게 됩니다. 이때 움직이는 간격을 stride라고 부릅니다. 위 예시에서는 첫번째 그림에서 두번째 그림으로 넘어갈 때 필터가 2칸을 움직였습니다. 이것을 stride가 2라고 표현합니다.

이론적으로 Convolution연산은 위 수식과 같은 형태의 연산을 의미합니다. 하지만, pytorch나 tensorflow 등 실제 딥러닝 프레임워크에서의 Convolution 연산은 Cross-Correlation 연산으로 이루어집니다. 여기서 Cross-Correlation 연산은 두 행렬을 곱할 때 같은 위치에 있는 원소끼리 곱하는 것으로, 위 그림을 180 도 뒤집은 형태의 연산입니다. 그 이유는 우리가 모델을 구성할 때 가중치가 학습을 통해서 변화되기 때문에 어느 연산을 사용하든지 상관이 없지만, 합성곱 연산의 경우 필터를 뒤집어야 하는 추가적인 연산과 그에 따른 비용이 들어가기 때문입니다. 결과적으로 위 Convolution Layer예시에서는 (4, 4)의 input에 대해 (2, 2)의 필터가 stride=2 로 움직여 (2, 2)의 결과를 반환했습니다. 여기서 일차적으로 반환된 결과를 Feature Map이라고 부릅니다.

2.1.1 Feature Map

Convolution Layer의 출력에 해당하는 Feature Map은 크게 input의 channel, output의 channel, 필터의 사이즈, stride, 그리고 padding 등으로 결정됩니다. 아래의 표는 이 5가지 하이퍼 파라미터들의 역할을 간략하게 정리한 표입니다. 각각 Pytorch와 Tensorflow에서의 파라미터 표현들을 각각 작성했습니다.

하이퍼 파라미터	Pytorch 표현	Tensorflow 표현	역할
Input Channel	in_channels	미리 지정	입력 데이터의 채널 개수 지정
Output Channel	out_channels	filters	출력의 채널 개수 지정
필터 사이즈	kernel_size	kernel_size	필터의 크기 지정
Stride	stride	strides	필터의 이동 간격 지정
Padding	padding	padding	입력 데이터 주변에 붙일 padding 수 지정

- Input Channel

입력 이미지의 채널 개수를 의미합니다. 앞서 설명드린 것과 같이 일반적인 이미지 데이터는 R, G, B 3개의 채널을 가지고 있으므로 Input Channel은 3이 됩니다. 흑백사진인 경우는 1이 됩니다.

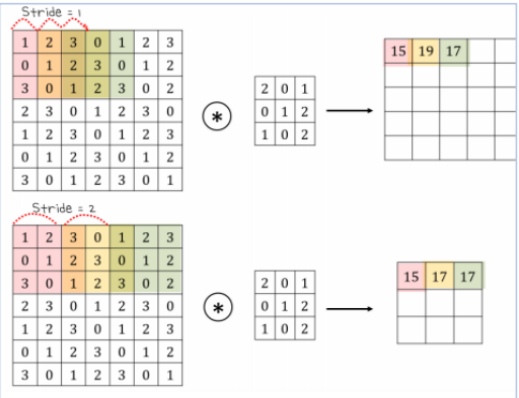
- Output Channel

Output Channel은 하나의 입력 데이터에 대해 Convolution Layer가 반환하는 channel의 수입니다. 다른 말로 하면 Convolution Layer의 필터의 개수라고도 볼 수 있습니다.

- 필터 사이즈

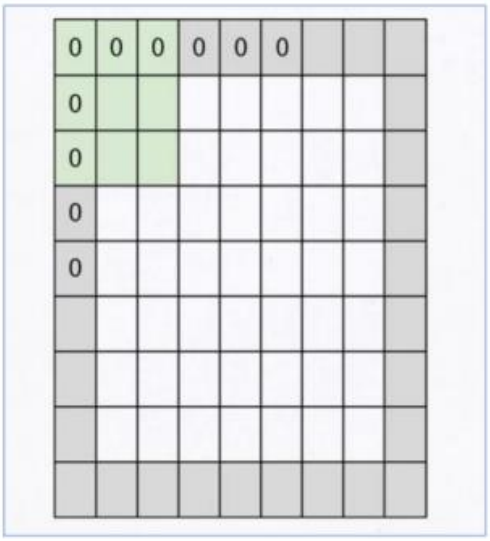
필터의 사이즈는 말 그대로 필터의 크기를 얼마나 크게 설정할 것인지를 의미합니다. 만약 (128, 128)로 정해진 크기의 input이 들어온다면, 필터의 크기가 클수록 그 필터를 거친 후 반환되는 Feature Map의 크기는 작아질 것입니다. 같은 stride에 대해서 (3, 3)가 (5, 5)보다 정해진 입력 내에서 움직일 수 있는 횟수가 더 많기 때문입니다. 필터의 크기는 일반적으로 홀수로 사용합니다.

- Stride



Stride는 필터가 이동하는 간격을 말합니다. Stride가 작으면 입력데이터에 대해 연산이 더 많이 이루어질 수 있기 때문에 Feature Map의 크기가 커지고, 크면 입력 데이터의 연산이 더 적게 이루어지기 때문에 Feature Map의 크기가 작아집니다. 왼쪽 사진을 보면 각각 stride가 1인 경우와 2인 경우가 나와있는데, 여기서 stride가 1인 경우의 Feature Map이 더 큰 것을 확인할 수 있습니다.

- Padding



Stride의 그림을 보면 입력데이터의 중심부 근처에 있는 값들은 필터에 여러 번 통과되는 반면, 가장자리에 있는 값들은 상대적으로 적게 통과됩니다. 이 문제점을 보완하기 위해서 Padding을 사용합니다. Padding은 위 그림과 같이 하얀색의 원본 데이터 주변 회색 칸에 새로운 값들을 넣어주어 가장자리의 값들도 필터를 여러 번 통과할 수 있게 해주는 방법입니다. 위 그림은 가장 많이 사용되는 zero padding으로 회색 칸에 모두 0을 대입하는 방법입니다. Padding은 stride와 마찬가지로 자연수를 대입하여야 하며, 위 그림은 Padding=1을 적용했을 때의 예시입니다.

최종적으로 위에 나온 파라미터들을 지정해주게 되면 합성곱 계층의 Feature Map의 크기를 계산해 줄 수 있습니다. 여기서 O_n은 출력의 가로 길이, I_n은 입력의 가로 길이, P는 padding의 크기, F는 필터의 크기, S는 stride의 크기입니다.

$$O_n = \frac{I_n + 2P - F}{S} + 1$$

예시를 통해 살펴보겠습니다. 아래의 그림에서 왼쪽은 입력 데이터, 오른쪽은 출력 데이터입니다. 입력 데이터의 사이즈는 (3, 32, 32)이며, 적용할 Convolution Layer 에는 (4, 4)사이즈의 필터가 10개가 있으며, stride=2, padding=1 인 경우, 출력 데이터, 즉 Feature Map의 사이즈를 구해보도록 하겠습니다.

가장 먼저 알 수 있는 것은, 필터가 총 10 개이기 때문에 output channel은 10이 된다는 점입니다. 이번에는 위 수식을 적용하여 Feature Map의 각 채널이 가지는 사이즈를 계산해보도록 하겠습니다.

$$O_n = \frac{I_n + 2P - F}{S} + 1 = \frac{32 + 2 - 4}{2} + 1 = 16$$

결과적으로 stride=2, padding=1, 그리고 필터 사이즈를 의미하는 kernel_size=4 인 경우 output channel의 사이즈가 input channel 사이즈의 절반이 되는 것을 알 수 있습니다. 그리고 필터의 개수가 10개로 output channel 이 10 이기 때문에 최종적으로 Feature Map의 사이즈는 (10, 16, 16)임을 알 수 있습니다. CNN 모델을 구성할 때 input 사이즈에 대한 output의 사이즈를 잘 계산할 줄 알아야 층이 깊은 모델을 구성할 수 있기 때문에, 이 식은 외우고 있는 것이 좋습니다.

2.2. Pooling Layer

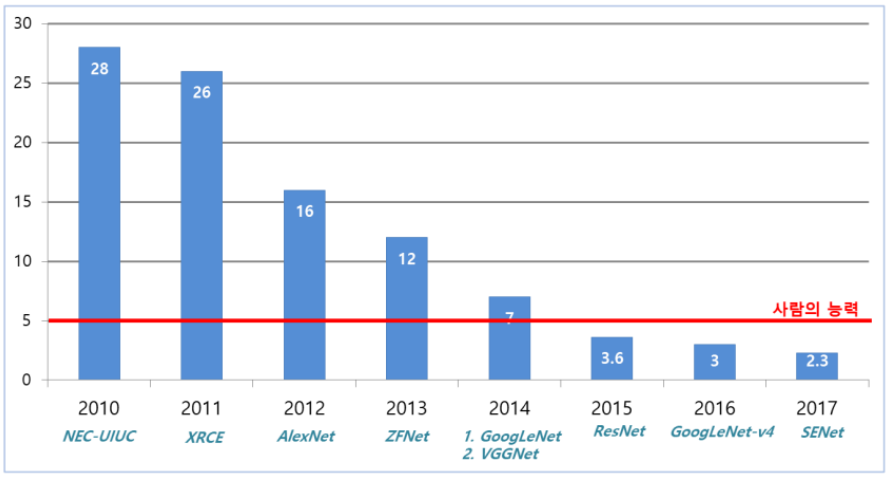
Pooling Layer는 Convolution Layer와 근본적으로 비슷한 역할을 하는 layer입니다. Convolution Layer의 경우 여러 개의 필터를 활용하여 입력 이미지의 중요한 특징들을 뽑아냈다면, Pooling Layer는 입력 이미지의 중요한 특징들을 더욱 강조하는 동시에 데이터의 크기를 줄이는 것에 더 중점을 둔 layer입니다. 하지만 Conv. Layer의 경우 필터 사이즈만큼의 가중치들이 존재하여 가중치와 입력의 연산을 통해 중요한 특징들을 뽑아냈던 반면, Pooling Layer는 가중치를 사용하지 않고 중요한 특징들을 강조한다는 차이점이 존재합니다.

Pooling Layer에는 Max Pooling, Average Pooling, Min Pooling 등이 있습니다. 이름처럼 Max Pooling은 필터가 덮고 있는 값 중 가장 큰 값을, Min Pooling은 가장 작은 값을 출력합니다. Average Pooling은 필터가 덮고 있는 값의 평균을 출력합니다. 이미지를 다룰 때는 일반적으로 Max Pooling을 사용합니다.

2.3. Fully Connected Layer

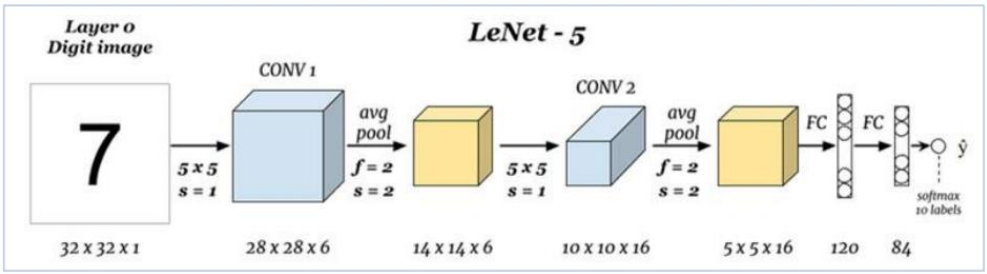
FC Layer는 1 주차 클린업 스터디에서 공부한 다층 퍼셉트론에서도 만난 적이 있는 개념입니다. 은닉층에서 출력층으로 넘어갈 때 여러 개의 퍼셉트론이 겹쳐 있는 형태와 동일한 형태의 layer입니다. 이전 두 layer와 달리 2 차원 이상의 데이터에 적용될 수 없기 때문에 FC Layer의 입력은 벡터로의 변환이 필수적입니다. 일반적으로 최종 단계에서 라벨 예측을 진행할 때 사용되는 layer입니다.

3. CNN의 발전 과정



위 사진은 대용량 이미지 데이터셋인 ImageNet을 이용해 이미지 분류 문제를 해결하는 대회(ILSVRC)에서 1등을 한 모델들입니다. 빨간색 선은 사람이 직접 이미지 분류 작업을 진행했을 때의 오류율로, ResNet부터는 사람의 오류율보다 인공지능 모델의 오류율이 더 낮아지기 시작했습니다. 위 사진의 모델들을 살펴보면서 모델이 어떻게 발전했는지 알아보겠습니다.

3.1. LeNet-5



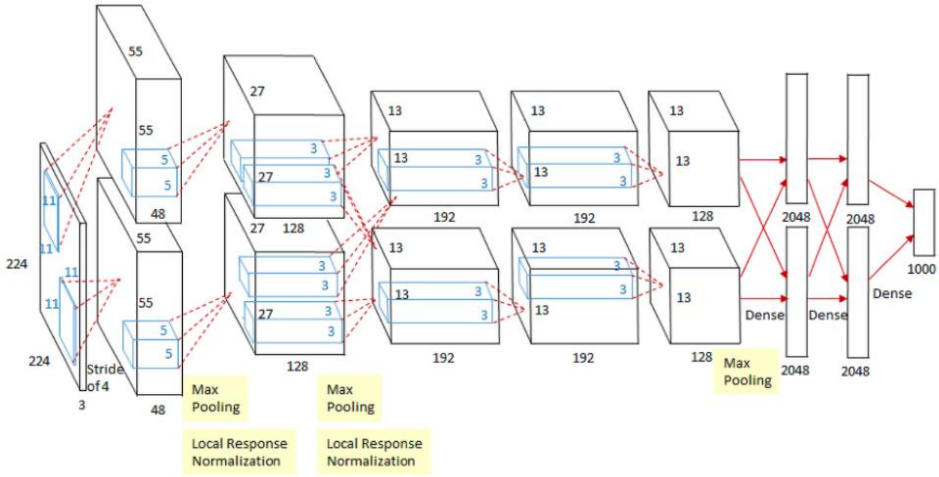
LeNet-5는 ILSVRC에서 우승한 모델은 아니지만, CNN의 태동과 같은 모델이기에 먼저 소개하도록 하겠습니다. LeNet-5는 현재 뉴욕대학교의 얀 르쿤 교수가 1998년 맨 처음 제안한 CNN 모델로 손으로 쓴 숫자를 구분하기 위한 모델입니다. LeNet-5는 (1, 32, 32)의 입력을 3개의 Convolution Layer와 2개의 Pooling Layer를 교차하여 통과시킨 후 2개의 FC Layer에 통과시켜 손글씨를 분류하는 모델이라고 볼 수 있습니다.

Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

FC Layer에 도달할 때까지 (1, 32, 32)의 입력 이미지가 (,120)으로 축소되었음을 확인할 수 있습니다. 이 때 LeNet-5에서 사용된 stride, 필터 사이즈, 필터의 개수, 그리고 각 Convolution Layer 혹은 Pooling Layer에 사용된 활성화 함수들은 아래의 표와 같습니다.

3.2. AlexNet

AlexNet 은 2012년 ILSVRC에서 큰 격차로 우승을 차지한 이미지 분류 모델입니다. AlexNet 이후부터는 ILSVRC에서 CNN이 주를 이뤘을 만큼 AlexNet의 성능이 매우 뛰어났으며, 그만큼 CNN 모델의 발전에 있어 중요한 위치를 차지합니다. 이러한 AlexNet의 구조는 아래와 같습니다.

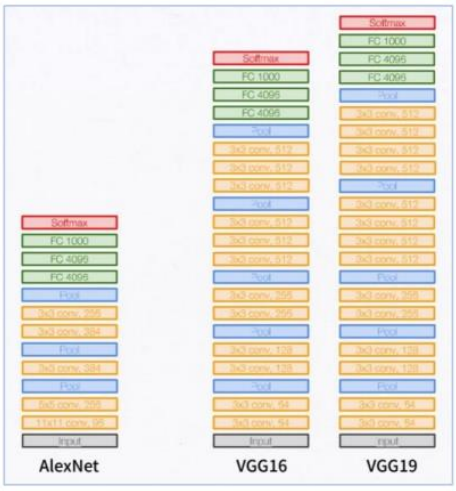


AlexNet에서 눈에 띄는 데이터의 병렬적 구조입니다. LeNet-5에서는 (1, 32, 32)의 작은 흑백 이미지를 사용했지만, AlexNet의 입력 이미지는 (3, 224, 224)의 컬러 이미지로 채널과 픽셀 수가 둘 다 늘었기 때문에 모델의 학습에 훨씬 더 많은 연산이 필요했습니다. 그래서 AlexNet은 2개의 GPU를 병렬적으로 이용하도록 구성되었습니다. ~~(이 시절 GPU의 성능의 한계 때문에 병렬적으로 이용한 것입니다.)~~

AlexNet에서 주목할 점은 ReLU 함수를 사용했다는 것입니다. LeNet-5에서는 모든 layer에서 tanh함수를 활성화 함수로 사용했던 반면, AlexNet에서는 ReLU 함수를 사용함으로써 연산을 획기적으로 줄이고 이미지의 특징을 효과적으로 뽑아낼 수 있었습니다. (1주차 클린업 내용 참고!) 또한, Average Pooling을 사용한 LeNet-5와 다르게 Max Pooling을 사용하여 이미지의 특징을 효과적으로 뽑아냈습니다.

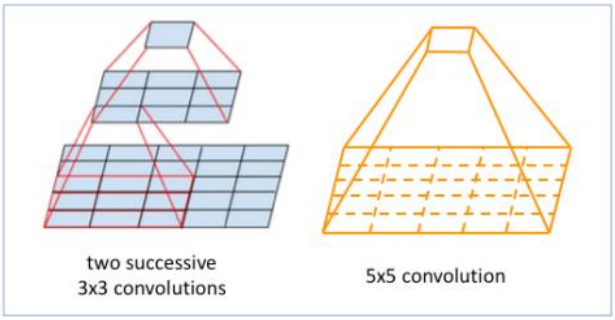
AlexNet의 다른 주목할 점은 1주차에서 성능향상 기법으로 배운 Normalization과 Dropout 기법을 사용하여 성능을 많이 향상시켰다는 것입니다. 여기서 사용한 Normalization은 local response Normalization입니다. 마지막으로 CNN 모델은 사물의 위치에 강건하지 못하기 때문에 이를 해결해주기 위해 Data Augmentation을 사용했습니다. Data Augmentation은 데이터 증강 방법으로 하나의 이미지를 사용해 여러 비슷한 이미지를 만들어내는 것을 뜻합니다. AlexNet에서는 데이터를 좌우반전 시키든지, 입력 이미지 크기인 (3, 244, 244)보다 조금 더 큰 이미지를 서로 다르게 잘라서 여러장을 만들어주었습니다. 이렇게 함으로써 과적합도 방지할 수 있습니다.

3.3. VGGNet



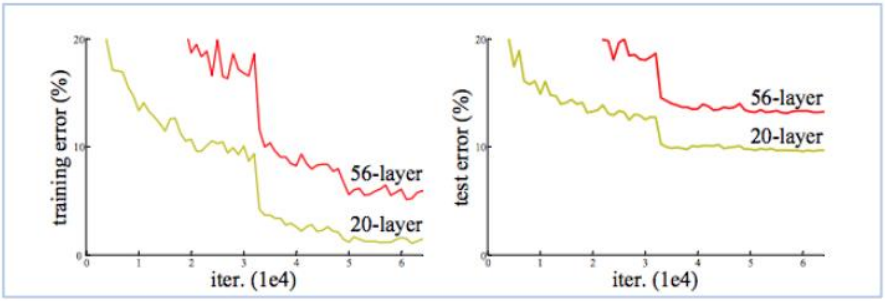
VGGNet은 2014년 ILSVRC에서 2등을 차지한 모델입니다. 1위는 GoogLeNet이 차지했지만, 구조가 매우 복잡했습니다. 하지만 VGGNet은 훨씬 더 간단한 구조임에도 불구하고 아주 좋은 결과를 보였기 때문에 GoogLeNet대신 VGGNet을 다뤄보겠습니다. VGGNet은 CNN에서 layer를 더 깊게 많이 쌓을 수 있다면 성능을 향상시킬 수 있음을 보여준 모델입니다. AlexNet에서는 8개의 layer 만을 사용했던 반면, VGG16Net과 VGG19Net은 각각 16개와 19개의 layer를 사용했음을 왼쪽 그림을 통해서 확인하실 수가 있습니다.

AlexNet에 비해 VGGNet이 더 많은 layer를 사용할 수 있었던 이유는 Convolution Layer의 필터 사이즈에 있습니다. AlexNet의 경우 (11, 11), (5, 5)를 거쳐 (3, 3)까지 필터의 사이즈가 줄어들도록 모델이 구성되어 있습니다. 반면 VGGNet은 (3, 3)의 필터(stride=1, pad=1)만을 처음부터 끝까지 사용했음을 확인할 수 있습니다. 마찬가지로 Pooling Layer역시 stride=2의 MaxPooling Layer만을 사용했습니다. 그렇다면 구체적으로 어떤 이유에서 작은 필터를 적용하여 모델을 더 깊게 쌓을 경우 성능이 올라가는 것인지 알아보겠습니다.

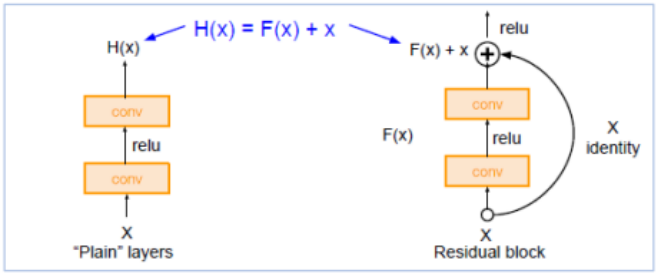


오른쪽은 (5, 5)의 데이터에 한 번의 (5, 5) Convolution Layer를 적용한 결과이며, 왼쪽은 같은 데이터에 두 번의 (3, 3) Convolution Layer를 적용한 결과입니다. 양쪽 모두 (5, 5)의 데이터를 (1, 1)로 변환해주었습니다. 결과는 동일하지만 왼쪽의 경우 2 번의 Convolution Layer를 통과하여 비선형성을 2번 추가해줄 수 있었던 반면, 오른쪽은 1번만 통과하여 비선형성을 1번밖에 추가하지 못했습니다. 즉, 왼쪽의 방법이 층의 중첩을 통한 비선형성의 추가라는 목적에 더 부합하는 것입니다. 게다가, 채널의 개수가 C개일 때 왼쪽에서 사용된 파라미터의 수는 $2 \times 3^2 \times C$개이며, 오른쪽에서 사용된 파라미터의 수는 $5^2 \times C$개로 왼쪽의 파라미터의 수가 더 적습니다. 결과적으로, 작은 사이즈의 필터를 가지는 Convolution Layer를 여러 번 사용할수록 비선형성을 더 부여할 수 있는 동시에 파라미터의 수를 줄일 수 있는 것입니다.

3.4. ResNet



ResNet은 기존의 CNN모델보다 훨씬 더 깊게 층을 쌓아 성능을 향상시킨 모델입니다. VGGNet에서 층을 깊게 쌓으면 성능이 더 좋아진다고 했으나, 층을 너무 깊게 쌓으면 오히려 성능이 더 떨어지는 문제를 확인할 수 있었습니다. 만약 모델의 깊이와 성능이 비례한다면 56-layer의 training error와 test error가 모두 20-layer보다 작아야 할 것입니다. 하지만, training error와 test error모두 순서가 역전되어 있습니다. 특히 training error조차 더 큰 것을 보았을 때 단순히 과적합의 문제가 아니라 성능이 떨어졌다는 것을 알 수 있습니다. ResNet을 개발한 연구자들은 이를 최적화의 문제로 파악했습니다. 이에 대한 대책으로 나온 것이 Residual Learning입니다.



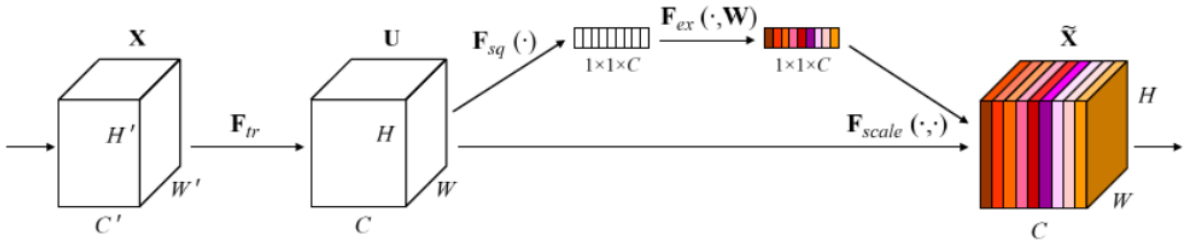
기존의 CNN 모델은 이전 layer의 결과가 입력으로 들어가며 Convolution Layer를 통과한 결과로써 $H(x)$라는 새로운 Feature Map이 반환됩니다. 이를 식으로 나타내면 $H(x) = f(x)$의 형태를 가지고 있습니다. 반면, ResNet은 이전 layer의 결과를 입력으로 사용함과 동시에 Convolution Layer 이후로 넘겨 그대로 더해줍니다. 즉, $H(x) = f(x) + x$의 형태를 가지고 있습니다. 이 경우 Convolution Layer의 역할은 이전 layer의 출력을 그대로 가져가되, 부가적으로 특징들을 학습해 주는 것이 됩니다. 이 식을 변형하면 $f(x) = H(x) - x$ 형태가 되고, 이는 $e = y - \hat{y}$ 와 같은 residual인 것을 확인할 수 있습니다. 따라서 이름을 ResNet이라고 합니다. 또한 역전파시 $H(x)$의 미분값이 어떻게 나타나는지 확인해 보겠습니다.

$$\frac{\partial H(x)}{\partial x} = (F(x) + x)' = F'(x) + 1$$

미분값이 위와 같이 나타난다는 것은 결국 Residual Block 은 이전 Feature Map에서 학습되지 못한 $F(x)$을 최적화하는 방향으로 학습이 진행된다는 것을 의미합니다. 동시에 기존 x의 미분값은 1이 되므로, 기울기 소실 문제 또한 예방하여 결과적으로 Residual Learning 은 매우 깊은 모델을 만들 수 있는 기반이 됩니다.

3.5. SENet

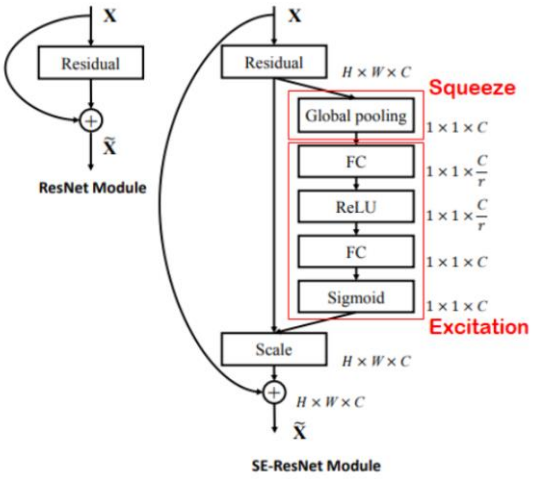
SENet 은 ILSVRC 2017 에서 1 등을 차지한 모델입니다. SENet 은 채널간의 상호작용에 집중하여 성능을 끌어올린 모델이라고 볼 수 있습니다. 여기서 SENet 은 SE Block 이라는 것을 제안했는데, 이는 CNN 의 어떤 모델에도 사용될 수 있습니다. 성능은 많이 향상되는 반면, 하이퍼 파라미터는 많이 늘지 않기 때문에 연산량 증가는 많지 않다는 장점이 존재합니다. 여기서 SE Block 이라는 것을 알아보도록 하겠습니다.



위 그림에서 X와 U는 Feature map, F_{tr}은 Convolution 입니다. $H' \times W' \times C'$크기의 Feature map X가 Convolution 을 통하여 $H \times W \times C$ 크기의 Feature map U로 변환됩니다. 그 이후 Squeeze 를 진행해줍니다. Squeeze 는 각 채널을 1 차원으로 만드는 역할을 합니다. Global Average Pooling (GAP)를 통해 각 2 차원 Feature map 을 평균 내어 하나의 값을 만들어준다고 생각하시면 됩니다.

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j)$$

여기서 Feature map은 C개의 채널을 갖고 있으므로 다 연결하면 $1 \times 1 \times C$가 됩니다. Squeeze 후에는 활성화(excitation) 작업에 돌입합니다. Excitation는 $1 \times 1 \times C$벡터를 정규화하여 가중치를 부여하는 역할을 합니다. Excitation은 FC1 - ReLU - FC2 - Sigmoid로 구성됩니다.



FC1에 $1 \times 1 \times C$ 벡터가 입력되어, C채널을 C/r개 채널로 축소합니다. 여기서 r은 하이퍼 파라미터입니다. C/r개 채널로 축소되어 $1 \times 1 \times C/r$ 가 된 벡터는 ReLU로 전달되고, FC2를 통과합니

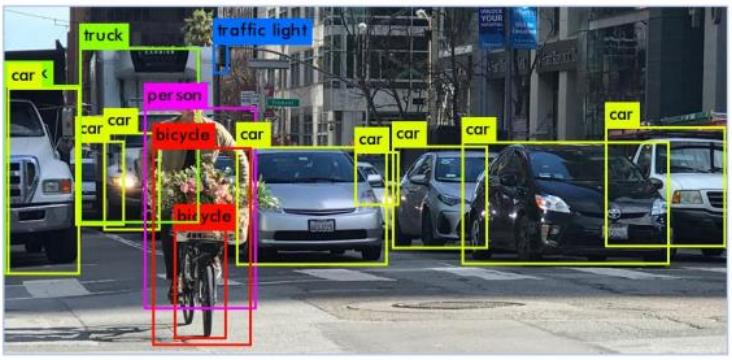
다. FC2는 채널 수를 다시 C 로 되돌립니다. 그리고 Sigmoid를 거쳐서 [0~1) 범위의 값을 지니게 됩니다. 마지막으로, Feature Map과 곱해져 Feature Map의 채널에 가중치를 주게 됩니다. SE Block의 목적을 간단하게 정리해보자면, Convolution을 통해 생성된 특성을 채널당 중요도를 고려해서 재보정한 것입니다. 이 SE Block을 기존의 CNN 모델에 붙여주어 성능향상을 도모한 것이라고 볼 수 있습니다.

위의 그림을 다시 한번 살펴보면 왼쪽은 기존의 ResNet구조이고, 오른쪽은 SE-ResNet구조입니다. SENet의 가장 큰 장점은 기존에 존재하는 CNN 모델들에 SE Block을 붙여주었지만, 연산량을 많이 증가시키지 않고 오류율을 많이 낮추었다는 점에 있습니다.

4. 컴퓨터 비전

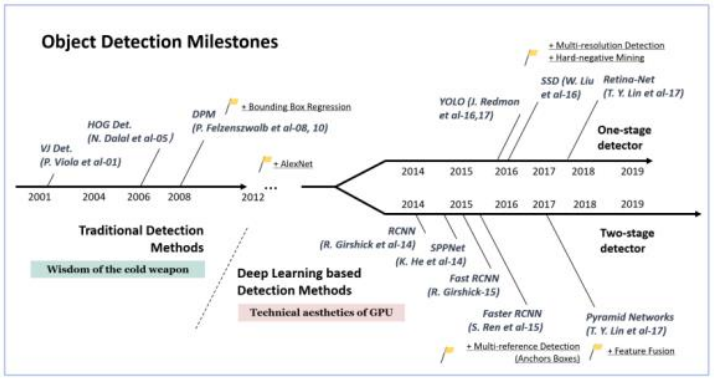
지금까지 CNN모델의 구조와 그 발전과정에 대해서 알아보았습니다. 이 때 CNN의 발전과정은 컴퓨터 비전 분야에서 이루어졌다고 볼 수 있습니다. 여기서 컴퓨터 비전이란 인공지능 분야 중 컴퓨터가 시각적인 체계를 이해하고 해석할 수 있도록 컴퓨터를 학습시키는 연구 분야입니다. 지금부터 딥러닝 모델이 컴퓨터 비전 분야에서 어떻게 활용되는지 알아보겠습니다.

4.1. Objection Detection (객체 탐지)



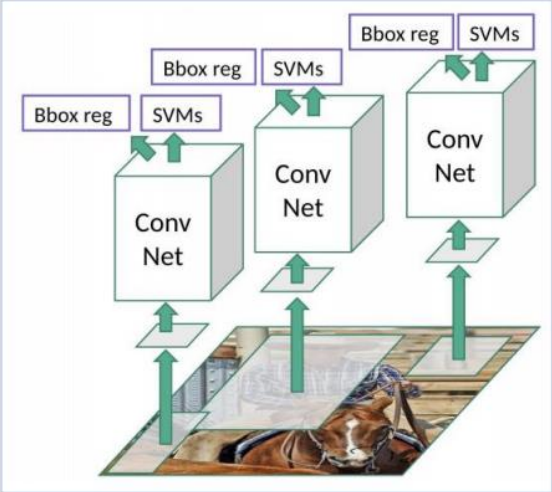
Objection Detection이란 이미지가 주어졌을 때, 어떤 물체가 어디에 있는지 탐지하는 과제를 의미합니다. 이 때 여러 객체가 존재할 때도 분류할 수 있게 합니다. 이 과정을 위해서는 어떤 범위에 객체가 존재하는지 탐지하는 Localization과 객체의 분류를 맡는 Classification이 동시에 일어나야 합니다. 여기서 사진을 보면 객체의 위치를 나타내는 경계 상자가 보이는데 이를 Bounding box(bbox)라고 합니다.

객체 탐지는 그 방법에 따라 크게 두 갈래로 나누어 생각할 수 있습니다. 첫 번째는 Localization과 Classification을 동시에 사용하는 1-Stage Detector입니다. 이는 두 작업을 동시에 사용하여 빠른 작업 수행이 가능하지만 상대적으로 정확도가 떨어진다는 단점이 있습니다. 두 번째는 순차적으로 진행하는 2-Stage Detector입니다. 2-Stage Detector는 상대적으로 느리지만 높은 정확도를 보인다는 장점이 있습니다.



위 사진은 Object Detection 모델들의 발전 과정을 나타낸 것입니다. 여기서 저희는 가장 1-Stage Detector와 2-Stage Detector의 대표적인 모델들을 살펴보겠습니다.

- RCNN (Regions with Convolutional Neuron Networks features)

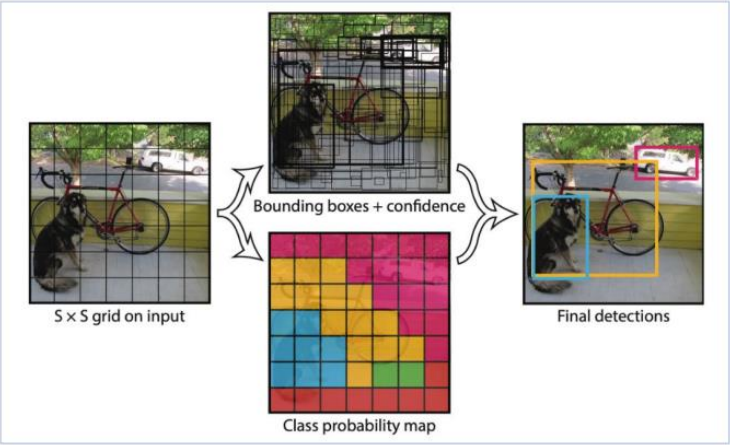


RCNN 모델은 가장 대표적인 2-Stage Detector 모델입니다. Object Detection의 발전 과정을 보면 알 수 있듯이 RCNN 모델은 2-Stage Detector 모델의 가장 기본적인 형태라고 볼 수 있습니다.

RCNN은 먼저 이미지에서 객체가 존재할 것으로 예상되는 위치인 RoI(Region of Interest)를 선별하며, 이 과정을 Region Proposal이라고 합니다. 이 때 RCNN은 Selective Search라는 알고리즘을 사용합니다. 이는 bbox를 랜덤하고 작게 많이 생성을 한 이후, 이것들을 계층적 그룹핑 알고리즘을 이용하여 조금씩 합쳐서 RoI를 만드는 것입니다. 그리고 이렇게 생긴 RoI를 CNN에 통과시킨 후 이 때 반환된 Feature Map을 바탕으로 회귀와 분류를 진행합니다. SVM을 통해 각 RoI의 라벨을 예측하고, localization error를 줄이기 위해 CNN feature을 이용하여 bbox regression model을 수정합니다.

하지만 RCNN은 여러 개의 CNN모델이 사용되기 때문에 매우 시간이 오래 걸린다는 단점이 존재합니다. 또한 CNN, SVM, Bbox regression이 한 번에 학습되지 않기 때문에 SVM, Bbox regression의 결과가 CNN을 업데이트 시키지 못합니다. 또한 회귀와 분류 문제가 모두 들어있어서 최적화가 어렵다는 단점도 존재합니다.

- YOLO

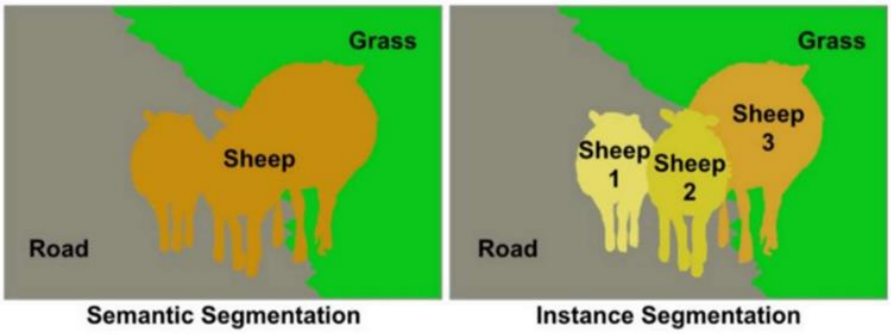


YOLO는 ~~YOU LIVE ONLY ONCE가 아니라~~ 1-Stage Detector의 가장 대표적인 모델입니다. 1-Stage Detector의 의미에 맞게 YOLO는 이미지 전체에 하나의 모델이 한 번의 연산을 통해 bbox와 객체의 라벨 확률을 동시에 반환하는 형태의 모델입니다. YOLO의 가장 대표적인 특징은 이미지를 여러 장으로 분할해 해석하는 CNN과 다르게 이미지 전체를 한 번만 보고 객체와 객체의 위치를 예측할 수 있다는 것입니다.

YOLO 알고리즘의 원리는 다음과 같습니다. 원본 이미지를 동일한 크기의 그리드(grid)로 나눈 후, 각 그리드에 대해 bbox와 Confidence Score을 계산하는 동시에 가장 높은 확률의 클래스를 선별합니다. 그리고 이를 바탕으로 확률이 일정 값 이상이 되는 셀들을 연결하여 최종적으로 bbox와 라벨을 반환하는 모델입니다.

YOLO는 통합된 모델을 사용하기 때문에 간단하고, 기존의 모델보다 빠르게 객체 검출이 가능하다는 장점이 있습니다. 하지만, 1-Stage Detector의 단점과 같이 2-Stage Detector보다 정확도가 떨어진다는 단점이 존재합니다. 특히, 작은 개체의 인식률이 떨어진다는 단점이 있습니다.

4.2. Image Segmentation



Semantic Segmentation　　Instance Segmentation

Image Segmentation이란 이미지를 구성하고 있는 모든 픽셀을 대상으로 어느 class에 속하는지 분류하는 작업입니다. 더 정확한 위치를 알아야 하기 때문에 object detection보다 더 복잡한 작업입니다. Image Segmentation에는 물체가 어디에 속하는지에 대해서만 분류를 진행하는 Semantic Segmentation과 같은 class 내에서도 더 세부적으로 분류를 해주는 Instance Segmentation이 있습니다. 그래서 겹쳐 있는 물체에 대해서도 Instance Segmentation은 더 세밀하게 문제를 해결할 수 있습니다.

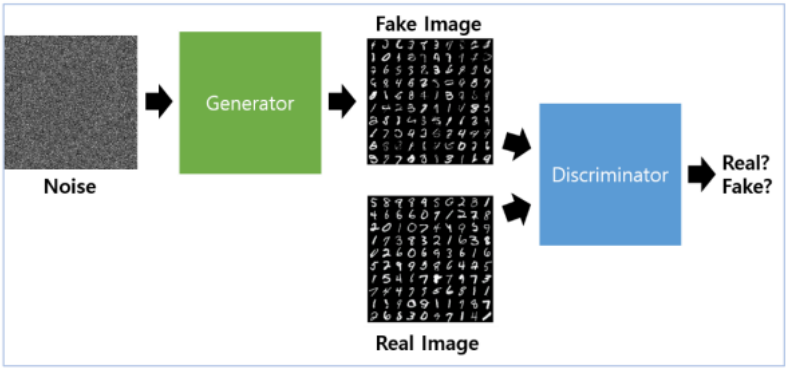
Input　　Semantic Labels

Image Segmentation은 이미지를 입력으로 넣어주면 위 사진처럼 각 픽셀이 어느 class에 속하는지가 출력으로 나오게 됩니다. 이때 입력 이미지의 크기와 출력의 크기는 같은 크기이고, 출력을 mask라고 부릅니다. 우리가 배웠던 일반적인 CNN같은 경우는 층을 거칠 때마다 이미지의 크기가 줄어들게 되기 때문에 그대로는 사용할 수 없습니다. 이에 대해서도 자세히 다루기엔 어려움이 있기 때문에 간단하게 설명하면 CNN에서 이미지의 크기를 줄이는 연산이었던 Convolution과 pooling을 역으로 연산해주어 다시 이미지의 크기를 키우는 방법을 사용합니다.

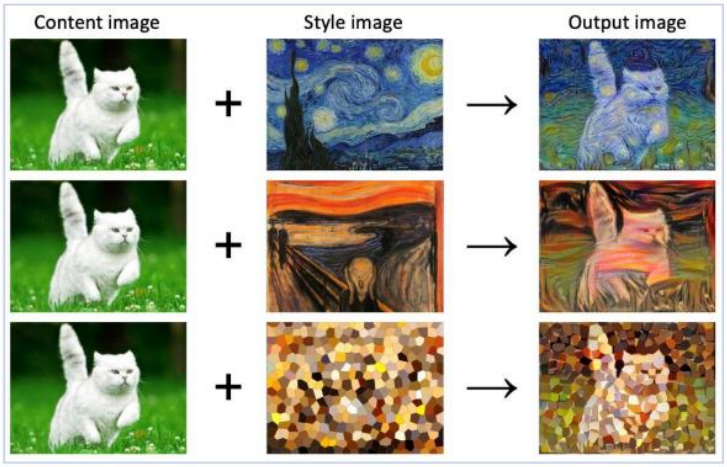
Image Segmentation은 자율주행, 의료 분야에서 주로 사용되는데, 그 외에도 다양한 분야에서 응용할 수 있습니다. 우리가 핸드폰으로 사진을 찍을 때 뒤의 배경을 흐리게 해주는 인물사진 같은 예시도 Image Segmentation을 사용했다고 볼 수 있습니다.

4.3. Image Generation

Image Generation 분야의 모델들은 주어진 이미지를 판단하는 형태가 아니라 주어진 이미지를 바탕으로 새로운 이미지를 만드는 모델이기 때문에 비지도학습에 속하는 딥러닝 모델들이 등장합니다. 몇 가지 예시를 간단하게 살펴보도록 하겠습니다.



GAN(Generative Adversarial Network)는 Image Generation 모델의 대표적인 사례입니다. GAN은 Generator와 Discriminator라는 모델로 구성되어 있으며, Generator는 가짜 이미지를 생성하는 모델이며, Discriminator는 실제 이미지와 가짜 이미지를 구별하는 모델입니다. 속이려는 쪽과 분류하려는 쪽은 서로를 적대적으로 학습하며 서로의 성능을 향상시키게 되어 궁극적으로 진짜 같은 가짜 이미지를 만들 수 있게 됩니다.



Style Transfer는 중간의 Style이미지의 특징을 추출하여 입력에 해당하는 Content Image에 적용하는 것을 목적으로 하는 알고리즘입니다. 두 이미지를 합성하는 것과 비슷하게 이해할 수 있기 때문에, Content Image와 새로 생성되는 이미지의 Content Difference, 그리고 Style Image와의 Style Difference를 모두 줄이는 것을 목적으로 손실함수를 구성합니다.

5. 마무리

2주차에서는 이미지 데이터에 대한 특징들을 알아보고 이미지 데이터를 CNN에서 처리하는 방식에 대해서 알아보았습니다. CNN모델은 가장 핵심이 되는 Convolutional Layer와 Fully Connected Layer 등을 알아본 후, 통시적으로 CNN 모델의 발전 과정을 살펴보았습니다. CNN 모델의 강점이 공간 정보를 활용할 수 있다는 것인만큼, 어떻게 층을 더욱 깊게 쌓는 동시에 공간 정보를 효과적으로 활용할 수 있는지에 대한 고민에 따라 LeNet부터 SeNet까지 발전해 왔는지 공부해 보았습니다. 또한 컴퓨터 비전 분야에서 이미지 데이터를 어떻게 활용하는지에 대해서 알아보며 Objection Detection, Image Segmentation, 그리고 Image Generator에 대해서 알아보았습니다.

딥러닝은 고전적 머신러닝과 다르게 비정형 데이터를 사용할 수 있다는 큰 장점이 있습니다. 하지만, 이를 이론적으로 알아보는 것도 중요하지만, 이를 실제로 구현해보는 것도 매우 중요합니다. 따라서 이번 코딩 실습 시간에는 pytorch를 활용한 이미지 분류 딥러닝 모델 구현 방법을 구체적으로 공부해보도록 하겠습니다. 그 후 3주차에서는 Sequential Data를 다룰 수 있는 RNN에 대해서 공부하도록 하겠습니다.