# 의료영상전처리

## 데이터 정제

### 데이터 정제를 하는 이유?

1. 데이터 분석과 모델링 작업이라는 머신러닝 task 말고도 데이터를 불러오고 정제하고 변형하고 재정렬하는 준비 과정에 많은 시간이 소모된다.
2. 파이썬 라이브러리 Pandas를 사용하면 유연하고 고수준의 알고리즘과 처리기능을 제공한다.
3. 결측치, 중복데이터, 문자열 처리 등 다른 분석적 데이터 변환에 대한 도구들을 다룬다.

### NaN값

* NaN 값 확인하기 : isnull()
* NaN값 처리 method
  + dropna : 누락된 데이터가 있는 축(row,column)을 삭제. 어느정도 누락 데이터를 삭제할 것인지 지정할 수 있다. (thresh값 넣어주기)
    - df.dropna(axis=1, thresh=500) #누락 데이터가 500개 이상이라면 열 삭제
    - df.dropna(subset=[‘age’], how=’any’, axis=0) #age의 열에서 NaN값이 존재하면 삭제. how=’any’ 옵션은 NaN값이 하나라도 존재하면 삭제하라는 명령
  + fillna : 누락된 데이터에 대신할 값을 채우거나 ‘ffill’,’bfill’같은 보간 방법을 사용
    - df[‘age’].fillna(mean\_age, inplace=True) #누락값을 mean\_age로 대체하며 실행 즉시 저장
  + isnull : 누락되거나 NaN값을 알려주는 Boolean (True, False) 값이 저장된 같은 형의 객체로 알려줌.
  + notnull : isnull과 반대 method

### 중복 값 제거

* 모든 값이 중복된 row가 존재할 수 있다.
* duplicate로 확인하고 drop\_duplicates로 제거
  + df2 = df.drop\_duplicates() #데이터프레임에서 중복 행을 제거
  + df3 = df.drop\_duplicates(subset=[‘c2’,’c3’]) #c2,c3열을 기준으로 중복 행을 제거

### 특잇값 찾기/ 범주 만들기

1. outlier를 찾는 것은 모델의 정확도를 높이는데 매우 중요
2. 그 외 알고리즘을 수행하는데도 outlier의 존재는 그 성능을 크게 훼손시킴
3. 특잇값을 찾아서 다른 값으로 대체
4. 더미변수를 사용하여 값을 이산화(discretization) 시키는것도 학습에 효율적임
5. cut함수의 사용방법은 [데이터, 구간의 갯수, 레이블명]에 해당하는 인자 값을 지정

### 문자열 데이터 정제

대부분의 경우 문자열은 쉼표로 구분하고 공백을 제거해야 한다.

* count : 문자열에서 겹치지 않는 부분 문자열의 개수를 반환
* startswith : 문자열이 주어진 접두사로 시작할 경우 True
* endswith : 문자열이 주어진 접미사로 끝날경우 True
* join : 문자열을 구분자로 하여 다른 문자열을 순서대로 이어붙임
* find : 첫번째 부분 문자열의 첫번째 글자의 ‘위치’ 반환
* rfind : 마지막 부분 문자열의 첫번째 글자의 ‘위치’ 반환
* split : 문자열을 구분자를 기준으로 리스트로 분리
* lower, upper, caplitalize : 각각 소문자와 대문자 그리고 첫글자만 대문자로 변경

### OS 라이브러리

Dataset을 정리할 때 사용하는 강력한 모듈 중 하나. import os로 사용

* os.getcwd() : 현재 경로 구하기
* os.listdir() : 특정 경로에 존재하는 파일과 디렉터리 목록 구하기
* os.path.exists() : 파일 및 디렉토리의 존재유무 확인. return값은 boolean으로 나온다.
* os.path.basename() : 지정된 경로의 파일명만 나온다. os.path.dirname() : 지정된 경로의 경로명만 나온다.
* os.path.join() : 경로와 파일명, 경로와 경로를 결합 가능
* os.mkdir() : 디렉토리 생성
* os.rename() : 파일명 변경
* os.remove() : 파일 삭제
* os.walk(dir.topdown=True) : 경로의 파일과 디렉토리 목록 파일명까지 순서대로 구할 수 있다.

## 데이터 전처리

데이터 전처리를 하는 이유?

* 모든 데이터 분석 프로젝트에서 데이터 전처리는 반드시 거쳐야 하는 과정이다.
* 분석 결과/인사이트와 모델 성능에 직접적인 영향을 미치는 과정이기 때문에 중요하게 다루어지는 과정

### Resize

* 이미지의 크기를 변화시킨다.
* 픽셀 값을 구할 때는 interpolation을 사용하는데 1차원에서는 linear interpolation을, 2차원에서는 bilinear interpolation을 사용.

### Resample

* 이미지의 픽셀 간 물리적 거리를 변화시킨다.
* dicom file의 meta정보를 불러와 거리 정보를 확인
* 픽셀 간 간격에 따라서 이미지가 확대 또는 축소될 수 있다.

### Normalization

* 이미지의 픽셀 값들을 0과 1사이로 변경시킨다.
* 이미지의 픽셀 값 범위를 통일 시켜 신경망이 불필요한 feature를 학습하지 않게 함.
* 연산의 효율 향상
* 픽셀 값의 비율은 유지하기 때문에 보이는 이미지의 변화는 없다.

### Z-score normalization

* 이상치(outlier)에 영향을 덜 받도록 평균과 표준편차를 이용

### Gamma correction

* 이미지가 너무 어둡거나 밝은 경우 픽셀 값의 변화를 통해 밝기 보정
* 선형적 방법은 전체적인 밝기가 일괄적으로 변경되고
* 비선형적 방법은 파라미터에 따라 변화를 더 크게 만든다.

### Color to grayscale

* Color 이미지를 변환 식에 맞게 Grayscale 이미지로 변환
* 변환 식은 average 또는 파라미터 값을 설정 가능
* 각각 R,G,B 채널에 가중치를 곱함.

## 데이터 증강

데이터를 증강하는 이유?

* 전체 Data의 N수가 부족할 때
* 클래스 당 균형 잡힌 데이터를 학습 가능
* 데이터 수가 증가할 수록 핵심적인 특징 부각
* 의료분야의 데이터는 비정상 데이터가 정상 데이터에 비해 적은 경우가 많다.
* 중요한 클래스(비정상 데이터)의 데이터 수가 다른 클래스에 비해 매우 적을 경우 클래스 불균형 문제가 발생하며 결과에 영향을 크게 준다.

### Rotation

* 이미지 회전

### Crop

* 이미지 자르기

### Flip

* 이미지를 좌우 또는 상하 반전

### Shift

* 이미지를 수평 또는 수직이동
* 새로운 영역에 대한 파라미터 설정 가능

## 의료영상의 획득 원리

### General radiography

* 영상 획득의 기본원리는 물체의 밀도 차이를 이용
* 물체의 밀도가 커서 X-선의 투과력이 약해지면 영상에서는 밝게 보이고
* 물체의 밀도가 낮아서 X-선의 투과력이 강해지면 영상에서는 검게 보인다.
* X-선은 발생 후 방사형태로 퍼져나가며 디텍터에 도달하면 영상화 시킬 수 있습니다.
* 일반촬영 영상을 볼 때는 해부학적 구조의 위치와 모양을 파악하고
* 음영의 높낮이로 정상 부위와 병변 부위를 구분
* 흰색 영역(음영이 높다)은 고밀도 물체로써 뼈가 대표적
* 검은색 영역(음영이 낮다)은 저밀도 물체로써 공기가 대표적
* 일반촬영 영상은 여러 장기들이 겹쳐 보이는 단점이 있다.
* 골절, 관절강, 폐렴, 기흉, 해부학적 구조 상 어긋남
* 병변에 따라 고음영/저음영으로 표현
* 연구에 포함된 병변에 따라 선택 ex)초기 단계에서 예후가 어떻게 될 것인지 파악하는 연구
* 어떤 부분의 표현력을 높여서 신경망에 전달할 것인지

### Computed tomography

* 전산화 단층 촬영(CT)은 X선 튜브가 회전하면서 영상 획득
* 영상획득에는 일반촬영과 동일하게 X선을 사용하기 때문에 획득 원리는 동일
* object의 밀도차이를 이용하여 영상의 대조도 비교
* 인체의 체적에 대한 정보와 공간에 대한 정보도 얻기 때문에 이를 바탕으로 다양한 영상으로 변환 가능
* CT영상을 획득할 때는 조영제(contrast)를 사용하면 원하는 부위(병변,혈관,종양)의 더욱 큰 대조도를 나타낼 수 있다.
* CT에서 장기와 조영제의 주입에 따라 이미지가 어떻게 달라지는지 확인 가능
* 조영제를 주입한 뒤 시간의 경과에 따라 해부학적 구조물의 대조도 차이가 달라짐
* 병변의 종류에 따라 픽셀 값(Hounsfield Unit)이 달라짐
* 어떤 부분의 표현력을 높여서 신경망에 전달할 것인지

### Magnetic resonance imaging

* 1.5T 이상의 강자강 안에 놓인 수소핵의 자기모멘트를 사용
* 수소핵의 자기모멘트는 전자기유도 법칙에 의해 자기장을 형성하고 신호를 얻는다.
* 자기장 안에 놓인 수소핵의 세차주파수를 RF pulse로 컨트롤
* 수소핵의 밀도에 따라서 조직의 신호 강도가 달라짐.
* 물은 공명현상이(excitation) 더 느리게 풀리고 지방은 더 빨리 풀리는 현상 이용
* RF pulse를 컨트롤하여 대조도를 얻고 여러 종류의 영상 획득
* f-MRI와 같이 해부 병리 영상뿐 아니라 뇌의 기능과 관련된 영상도 얻을 수 있다.
* 지방 부위의 묘사가 뛰어나서 근육, 인대 부위의 표현력이 우수
* FLAIR(액체 감쇠 반전 회복 기법) : 뇌척수액(CSF)의 신호를 무효화시켜서 뇌실 주위와 척수의 병변을 좀 더 선명하게 볼 수 있다.
* MRA TOF(MRI angiography Time of Flight) : 혈액 속의 분자들을 excitation시킨 후 영상을 얻고 싶은 부위만 신호를 획득하면 검사되는 부위는 혈관만 밝게 보인다. 혈관 부위의 병변을 보는 데 이용
* DWI : 정상조직은 자유롭게 움직여서(확산하지만) 신호강도가 더 낮고 병변이 존재하는 조직은 확산이 제한된다.

## 의료영상 전처리

### Edge detection

* 의료영상에서 abnormal한 부분의 표현력을 높이기 위해서 사용
* OpenCV 라이브러리
* 일반촬영 영상에서 관절강의 길이는 병변 진단에 중요한 역할을 함.

### Patch Extraction

* 이미지의 일부분을 떼어서 사용하는 방법
* CT와 MRI 영상은 인체의 횡단면 영상을 얻는다. 이를 Z축으로 일정한 길이마다 슬라이싱해서 모델 학습에 이용 가능
* Segmentation 모델 학습에서는 lesion의 labeling data에 coordinate가 들어있다. 이를 이용해서 lesion의 extraction을 사용 가능
* 병변 중에는 입체적인 모양을 갖는 것도 존재하고 단면적인 패턴이 중요한 것도 존재
* 3D로 적용할지 2D로 적용할지 역시 연구자의 선택
* 병원에서 판독을 할 때 폐 실질 영역의 병변은 slice별로 확인하고 판단 > 2 dimension
* 전처리 및 신경망 구조를 설계할 때 임상에서 내리는 의사결정을 바탕으로 유사한 workflow를 설계
* 2D영상 및 3D영상으로 변환

### Mask Extraction

* 폐 영역 내의 폐 실질 영역의 학습이 필요

1. 폐 영역만 남길 수 있는 알고리즘 사용
2. 폐 흉곽 부분의 pixel value가 255로 학습 시에 신경망이 폐 실질 부분과 함께 흉곽 부분을 집중해서 학습
3. 잘라낸 영역을 원본 dicom 영상으로 대체해서 영상을 만듦

## 데이터 이해

### Hounsfield Unit

* CT imaging에 적합한 대상 : X선 흡수율 차이가 큰 경계면, 뼈-연조직, 공기-연조직, 조영증강 혈관 등
* 부적합한 대상 : 뇌의 백질-회백질, 연조직으로 이루어진 두 장기나 조직 간 경계면
* Solid tumor : 위치한 장기에 따라 X-ray contrast가 달라서 난이도가 다름

### Window setting

* parameter는 window level과 window width
* W/L은 보고자 하는 부위를 결정하고 W/W는 Window Level을 중심으로 얼마나 넓게 볼 수 있는지를 표현
* 두부, 흉부, 복부, 사지 등 스캔 부위와 보고자 하는 부위에 따라서 window setting은 다르게 표현
* 파이썬 코드 구현할 때는 min,max값을 고정하고 center값을 원하는 HU값에 고정

### Algorithm

|  | Hard | Soft |
| --- | --- | --- |
| parameter | Spatial resolution | low contrast |
| 차이가 큰 pixel value를 비교하기 위한 것 | pixel value의 차이가 적은 물체를 구분하는데 관련된 parameter |
|  | 좁은 범위의 gray scale을 비교할 때 사용 |

### CT slice thickness

* slice thickness를 결정하는 요인은 low contrast를 높일 것인지 high resolution을 높일 것인지에 의해 결정

# Introduction of AI

## 컴퓨팅사고와 인공지능

### 인공지능의 개념

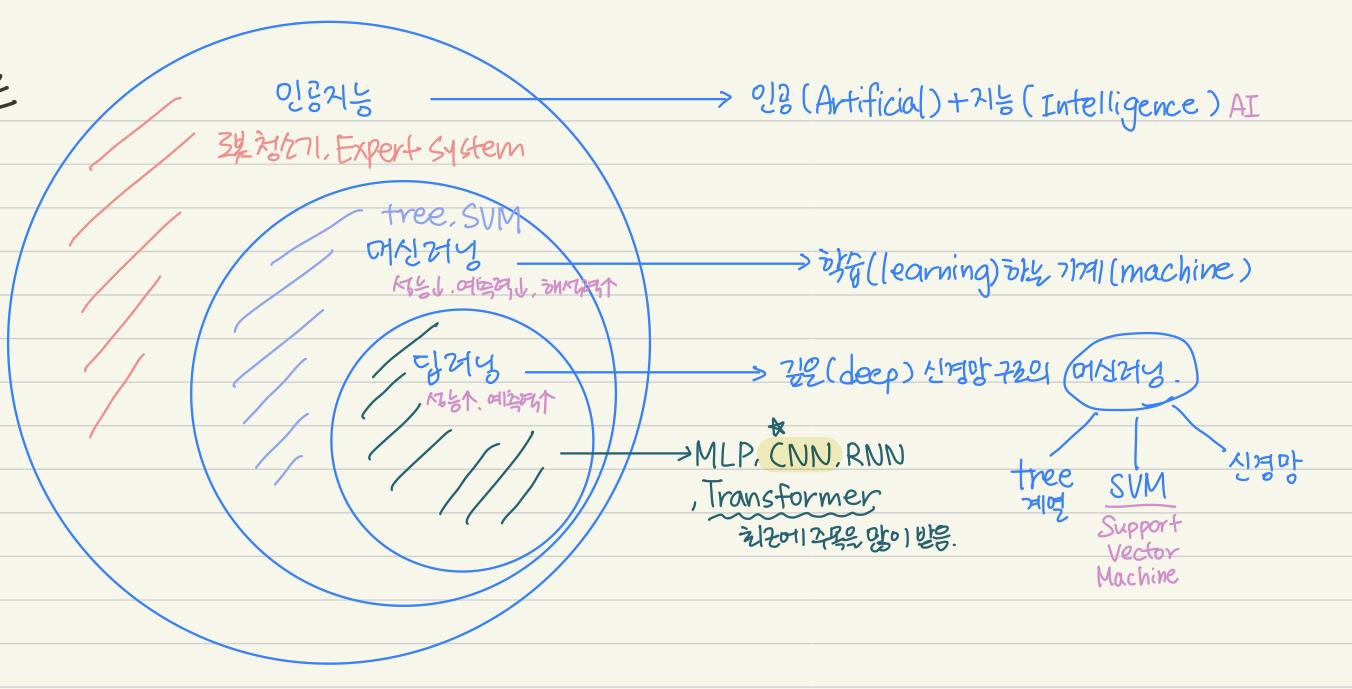
* 지능 : 본능적이나 자동적으로 행동하는 대신에, 생각하고 이해하여 행동하는 능력
* 인간이 어떤 사물을 이해하고 학습하는 능력(learning)
* 어떤 문제가 주어졌을 때, 합리적으로 사고하여 문제를 해결하는 능력(problem solving)
* 인간의 고유한 능력
* 인공지능 : 컴퓨터를 사용하여 인간의 지능을 모델링하는 기술
* 사람의 생각과 관련된 활동 예를 들면 의사결정, 문제해결, 학습 등의 활동을 자동화하는 것
* 인간의 인지적인 기능을 흉내 내서 문제를 해결하기 위하여 학습하고 이해하는 기계

### 데이터의 폭발적 증가

* 과거와 비교할 수 없는 속도로 데이터의 양이 증가하고 있다.
* 회로 안의 트랜지스터 개수가 2년에 한 번씩 exponential하게 증가한다. 이는 IT 기술의 빠른 발전 의미

### 인공지능의 종류

* Artificial Intelligence
* Machine Learning
* Deep Learning



* 비지도 학습 : 사람의 개입 없이 머신이 스스로 학습
* 지도학습
  + 정해둔 답(labeling file)이 존재
  + 오차를(Loss function) 최소화 시키도록 학습
  + semi-supervised learning, active learning
* 강화 학습
  + 환경과 agent가 존재하는 상태에서
  + 행동에 따른 보상을 최대화 하려는 방식

## 머신러닝이란?

### 머신러닝 용어

* Inductive bias : 딥러닝 모델이 데이터로부터 학습한 것
* Formulation : 딥러닝 모델에 학습 데이터를 입력하면 모델이 그 데이터의 전문가가 되는 것
* Training data
* Test data
* Loss function : 모델의 예측과 실제 정답과의 차이
* Parameter
* Loss
* Estimation, Evaluation, Inference

### Basic law

* No free lunch : 일반화의 문제
* Occam’s Razor : 복잡성의 문제
* Complexity : 모델 복잡성, 데이터 복잡성
* Generalization : 과대적합(overfitting)의 문제

### Overfitting

* 해결방안

1. 데이터셋 늘리기
2. 파라미터(feature)수 줄이기
3. 가중치 규제(regularization) 추가하기

### 정리

* input data x를 모델에 넣었을 때 모델의 파라미터에 의해 나온 output(예측 값)을 실제 정답과 비교
* 비교(Loss)를 최소화 하도록 하는 것이 학습
* 학습이 너무 학습데이터에만 잘 맞으면 과대적합. 반대 상황은 과소적합
* input data를 다루기 위해 행렬 형태로 변환해야 한다.
* 파라미터들이 input data의 행렬 연산이 필요하기 때문에 벡터의 성질이 필요
* 최소값을 찾기 위해 미분이 필요(벡터의 미분 값)
* 이러한 연산들을 반복하여 수행시키기 위해 파이썬 및 GPU가 필요

## 머신러닝 예시

### Supervised Learning\_지도학습

* Input data와 Input data에 대한 정답에 해당하는 Label(또는 class) 정보를 입력 받는다.
* 주어진 데이터와 label로부터 모델을 학습시켜 새로운 데이터를 입력으로 받았을 때 label을 예측
* 고양이 인지 아닌지를 분류하는 Binary-Classification

### Unsupervised Learning\_비지도 학습

* 오직 Input data를 기반으로 군집을 찾는 학습 진행
* 데이터 분석을 통해 unknown pattern을 학습할 수 있다. 그 결과로 새로운 데이터가 입력으로 주어졌을 때 분류할 수 있게 됨.
* 예를 들어, 데이터 간의 유사도(similarity), 패턴(pattern), 차이(difference) 등으로 데이터를 분류할 수 있는 학습 진행

### Linear Classifier\_선형 분류

* 주어진 속성의 선형결합을 바탕으로 분류를 수행 (직선으로 데이터를 두 그룹으로 분류)
* 개체의 속성은 feature값이라고 부른다. 보통 feature vector라는 벡터 형태로 제공
* 분류를 할 때 주로 Neural Network, SVM을 많이 사용

### Reinforcement Learning\_강화학습

* 기계 학습의 한 영역
* 어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동들 중 보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법

### Perceptron

* 다수의 신호(Input)을 입력받아서 하나의 신호(Output)을 출력
* weight가 신호를 전달하는 역할
* weight는 각각의 입력신호에 부여되어 입력신호와 계산을 하고 신호의 총합이 정해진 임계값(세타)을 넘었을 때 1을 출력. 넘지 못하면 0 또는 -1을 출력
* 각 입력신호에는 고유한 weight가 부여되며 weight가 클수록 해당 신호가 중요하다.

## Linear Classifier

### 기본 개념 및 용어

* Classification : 주어진 데이터를 분류하는 것
* feature vector : 이미지의 특성을 설명하는 정보
* transpose(T) : 벡터의 열과 행을 바꾸는 것(계산을 쉽게 하기 위해)
* Hypothesis : input data를 받아서 +-1을 예측해주는 모델(함수)
  + Model(=hypothesis,h)은 예측을 하고 이것이 label(정답)과 일치해야 한다.

### Loss function

* hypothesis의 성능을 확인하는 지표가 필요
* 이를 수치화 하기 위해 만든 것을 Loss function이라 하고 L이라 표기

### Perceptron algorithm

* 초기값 설정
* 설정된 iteration만큼 반복 수행
* 매 수행마다 loss function을 계산
* 값이 0보다 작거나 같을 때 weight와 bias를 갱신
* 반복 수행이 종료되면 최종적인 모델의 결과를 얻을 수 있다.

## Regression\_회귀

* 회귀분석은 독립변인이 종속변인에 영향을 미치는지 알아보고자 할 때 실시하는 분석 방법
* 단순 선형 회귀분석은 독립변수 X(설명변수)에 대하여 종속변수 Y(반응변수)들 사이의 관계를 수학적 모형을 이용하여 규명하는 것
* 규명된 함수식을 이용하여 설명변수들의 변화로부터 종속변수의 변화를 예측하는 분석
* 기본 원리는 선형 회귀모델의 직선과 실제 값 사이의 차를 뜻하는 residual error를 최소화 시키는 것

### Uncertainty

* Perceptron의 문제점 : +1 또는 -1로 판단하는 것에 대해 얼마만큼의 확신을 갖고 판단하는 가에 대한 정보가 없다.
* 데이터가 선형적으로 나뉘어 있지 않다면 좋은 성능을 내기 어렵다.
* 이를 극복하기 위해 모델에 uncertainty를 넣어준다
* sigmoid함수를 이용해 logistic function을 만든다.

## 

# Feed forward Neural network

## Gradient Descent

### 개념 및 용어 정리

* 경사하강법(Gradient descent) : Loss function을 수행할 때 이 함수를 미분하여 근사값(기울기)를 구하고 경사의 반대방향으로 계속 이동시켜 최소 값에 이를 때까지 반복하는 것이다.

### Learning rate

* Gradient Descent 방향으로 얼마나 움직일 지 정해준다.

## Regularizer

### 정규화가 필요한 이유

* 학습이 진행되면 어느 지점을 지나는 순간 오버피팅(overfitting)현상이 발생
* 딥러닝 모델이 training data를 적절히 학습하는 것도 중요하지만
* 모델이 test data에 대해 정확히 예측하는 것이 궁극적인 목표
* 베스트한 컨디션을 찾기 위해 regularization을 사용
* 또 다른 방법은 사용하는 feature를 줄이기 위해 training data의 수를 줄이거나 feature selection 알고리즘을 사용하지만 권장되지는 않는다.

### 개념 및 용어 정리

* Regularization : Loss function을 통해 학습이 진행될 때 계속 갱신되는 weight factor를 training data에 너무 피팅되지 않도록 weight factor의 변화를 억제한다.
* L2 regularization : L2 norm을 사용한 regularization로 유클리안 거리 공식 사용

## Feed forward NN

* classifier의 decision boundary가 더 복잡한 모형을 가진다면 복잡한 데이터를 잘 분류할 수 있을 것이다.
* classifier를 더 복잡한 형태로 만들기 위해서는 Input으로 더 많은 정보(feature)를 넣어주면 가능할 것이다.
* 각각의 layer에 더 많은 파라미터를 넣으면 더 좋은 classifier가 될 것이다.
* 가중치(w)의 반복적인 업데이트를 통하여 출력값의 에러를 최소화하는 것이 목표
* 가중치 업데이트는 경사하강법을 이용하여 에러를 역전파하여 구한다.
* 그러나, 이 경우 기울기 소실 문제가 발생할 수 있다.

### Activation function

* step function : 0과 1의 출력이 나오는 함수
* sigmoid function : linear classifier에 uncertainty를 추가
* 그 외 activation function : tanh, ReLU, Leaky ReLU

### Back propagation\_오차역전파

* 모두 한 개의 layer로만 구성된 network를 가정. 각각의 layer에 weight와 bias가 존재
* 왼쪽에서 오른쪽으로 연산이 진행되는 것을 순전파라 하며 output과 ground truth를 비교
* 학습을 위해 loss를 가지고 역전파 계산과정에 사용. 이를 위해 오차를 각 가중치로 미분
* 각 노드를 하나의 function이라 생각하고 편미분을 해준다.
* gradient를 구할 때 미분 값의 마이너스 부호를 붙여줬던 것처럼 Cost를 편미분한 값의 변화량으로 그 다음 derivative변화시킨다.
* 각각을 편미분하지 않고 한 번에 계산하여 다른 수식들은 모두 사라지게 만든다.
* Chain rule은 첫 변화량으로 연쇄적인 변화를 확인하는 것이다.
* Loss function을 미분한 값을 찾기 위해 chain rule을 이용
* 하나의 뉴런만 생각했을 때와 달라진 점은 이전 레이어에서 구한 하나의 노드의 출력 값이 다음 레이어의 여러 노드에 영향을 준다는 점입니다. 이것은 에버리지를 구하는 연산으로 처리하게 됩니다.

## Optimizer

### optimizer란

* 최적화(Optimization)은 손실함수의 결과값을 최소화하는 모델의 파라미터(가중치)를 찾는 것을 의미. 그리고 Optimizer는 이러한 Optimization의 알고리즘의 종류이다.
* gradient descent는 가장 기본적인 Optimizer의 하나로 경사를 따라 손실함수의 최소 방향을 찾아가면서 가중치를 업데이트합니다.
* SGD, Momentum, AdaGrad, AdaDelta, RMSProp, Adam등이 존재하며 가장 많이 사용되는 것은 Adam또는 SGD입니다.
* Adam은 어떤 상황에서든 global minimum에 빠르게 도달하는 만능 알고리즘이며 SGD는 개발자가 다른 조건들을 최대한 통제 가능한 상황에 두고 학습 변화를 면밀히 분석할 수 있어서 사용

### Stochastic Gradient Descent

* 확률적 경사하강법으로 최초의 방법인 Gradient Descent가 모든 데이터를 학습하는 것에 비해 random하게 추출된 일부 데이터에 대해 gradient를 확인하고 이동. 다시 데이터셋을 섞고 random하게 추출. 속도는 개선되지만 정확도는 낮은 방법으로 mini-batch와 함께 사용되면 시너지 효과가 발생
* Mini-batch : 전체 데이터 셋 N개에 대해 batch size로 나눠 각 배치 셋을 순차적으로 학습에 이용하는 것입니다. 미니 배치의 장점은 GPU의 병렬처리를 이용할 수 있다는 것

### SGD의 단점

* SGD에서는 도착한 순간에서 gradient를 구하기 때문에 고차원의 hypothesis안에서는 local minimum보다 saddle point에서 문제가 발생
* 안장점(saddle point)는 그래프에서 보이는 주변 부위가 평평한 영역인데 SGD는 이 포인트에서 변화량이 줄어들 수 있다.
* 또한 탐색경로가 비효율적이라는 문제도 있습니다. 지그재그로 움직이는 단순한 방식을 보완해야 함

### Optimizer

* Momentum은 ‘운동량'을 뜻하며 모멘텀 옵티마이저는 관성이라는 개념을 추가한 것으로 볼 수 있다. 기울기의 방향으로 속도가 생성됨. 초기에는 0일 것이고 그 방향으로 gradient이동이 지속되면 더 큰 운동량을 갖게 됨
  + 현재의 time stamp기울기만 고려하는 SGD보다 이전 time stamp까지 고려해서 누적된 값들을 Optimizer에 사용
* AdaGrad : 학습이 진행될 수록 학습률(lr)을 줄여주는 방식. 여기서 새로 등장하는 파라미터는 h로 기존에 기울기 값을 계속 더해주되 전체 gradient에는 그 제곱근으로 나눠줍니다. 즉, 학습이 거듭될수록 학습률이 감소
  + global minimum에서 더 이상 학습이 진행되지않고 왔다갔다 하는 경우가 발생할 때 도움이 됨
* RMSProp : AdaGrad에서 기울기를 추가로 반영하기 위해 지수 가중 이동평균을 넣는다. AdaGrad에서 학습이 진행될수록 lr이 0에 수렴하는 것을 방지
* Adam : Momentum과 RMSProp을 함께 적용한 방법. momentum의 관성을 이용하기 때문에 local minimum이나 saddle point에 빠지지 않고 RMSProp을 적용해서 lr이 자동으로 조절됨.

# CNN

## Convolutional Neural Networks

### 합성곱 신경망(CNN)이란?

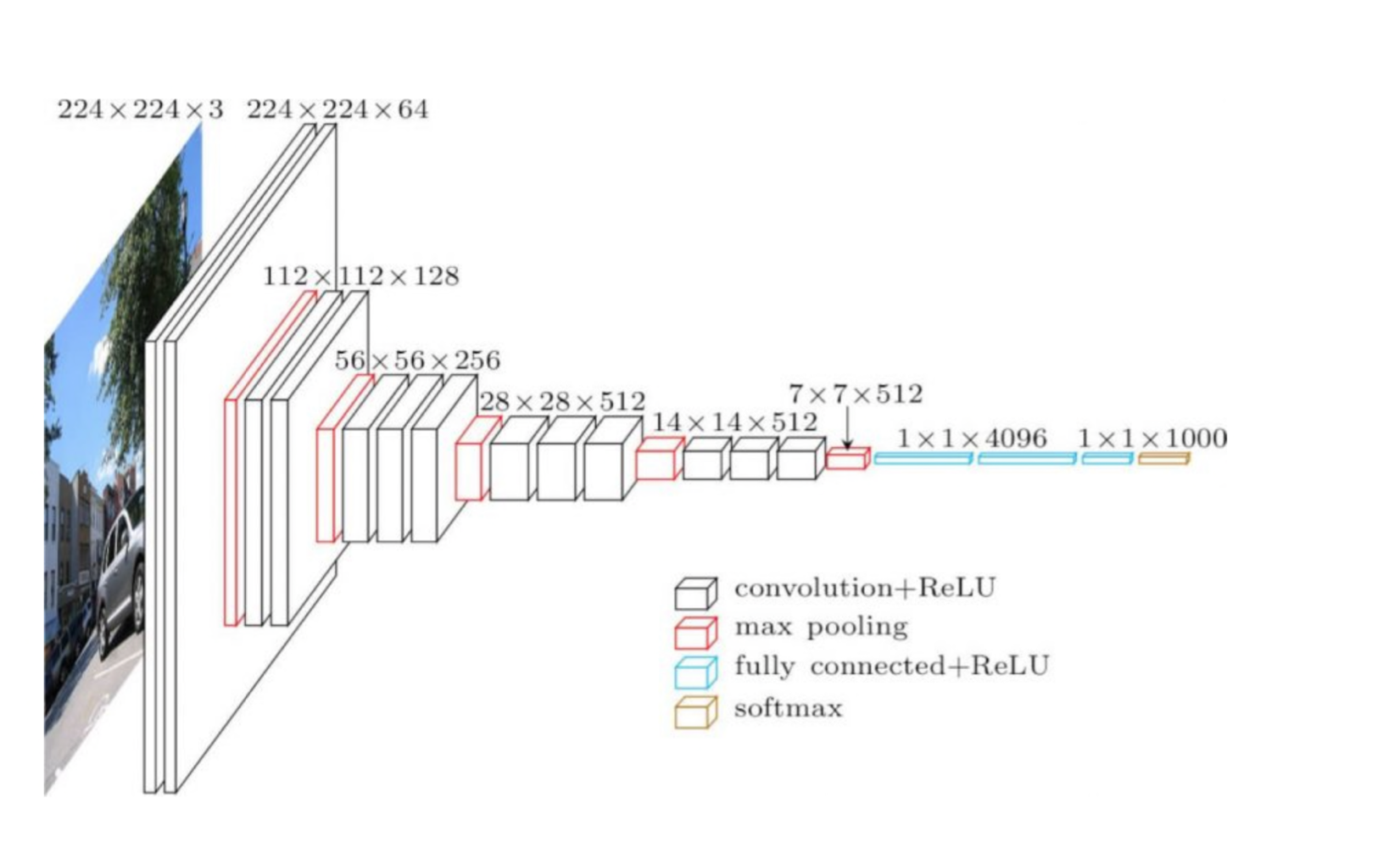
* 이미지의 특징을 추출하고 학습할 수 있는 layer
* 이미지 데이터의 패턴을 학습하고 이미지를 분류할 수 있다.
* 컴퓨터 비전 분야에서 다양하게 사용되고 있으며 feature map을 생성하는 필터까지도 학습이 가능
* 이미지의 공간 정보를 유지한 상태로 학습이 가능한 모델
* Fully connected layer : 모든 Input이 모든 node와 연결된 형태의 layer
  + 입력 데이터는 1차원(배열)형태로 한정된다.
* CNN : 하나의 node에 filter에서 추출된 정보가 담기며 kernel은 공간정보를 유지할 수 있다. filter의 size를 5x5x3이라 한다면 하나의 뉴런의 filter를 통과한 feature map의 정보가 담기며 5x5x3개의 weight가 존재. filter의 존재로 인해 local connectivity 특성을 갖는다.
* CNN과 기존 Fully Connected Neural Network의 차이점
  + 각 레이어의 입출력 데이터의 형상 유지
  + 이미지의 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식
  + 복수의 필터로 이미지의 특징 추출 및 학습
  + 추출한 이미지의 특징을 모으고 강화하는 Pooling 레이어
  + 필터를 공유 파라미터로 사용하기 때문에, 일반 인공 신경망과 비교하여 학습 파라미터가 매우 적음

## Filter

* 이미지의 특징을 찾아내기 위한 공용 파라미터
* filter를 kernel이라고 하기도 한다.
* 일반적으로 (4,4)이나 (3,3)과 같은 정사각 행렬로 정의된다.
* CNN에서 학습의 대상을 필터 파라미터이다.
* 필터는 입력 데이터를 지정한 간격으로 순회하면서 합성곱을 계산한다.
* 필터의 개수는 출력 데이터의 volume과 동일하다
* Filter의 파라미터
  + Receptive field : 필터가 한 번에 보는 영역
  + kernel size : 필터의 크기
  + Stride : 필터가 움직인 거리
  + Zeropadding : stride에 따라 input size가 맞지 않을 때 사용
    - 입력 데이터의 외각에 지정된 픽셀만큼 특정 값으로 채워 넣는 것을 의미
    - 합성곱 연산에서 출력차원(height&width)이 계속 감소하는 문제가 발생 ->Zeropadding을 사용하는 주된 이유
    - stride를 1로 학습할 때 데이터의 크기를 유지하기 위해서 사용

### Max pooling

* 이미지의 크기를 유지한 채 Fully connected layer로 간다면 연산량이 기하급수적으로 증가한다.
* kernel size를 작게 하면서도 더 넓은 영역의 이미지를 학습할 수 있다.(연산량 감소)
* 이미지의 크기를 줄이는 방법



CNN 구조

## Layer

### Batch Normalization layer\_배치 정규화

* 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이요해 정규화하는 것
* chain rule에 의해 파라미터 값의 작은 변화량이 전체 신경망의 출력에 영향을 주게 되는데 변화량이 매우 작아지거나 매우 커지는 문제를 gradient vanishing 또는 exploding이라 하고 학습에 매우 큰 영향을 준다.
* Sigmoid와 tanh등의 활성화 함수는 비선형적인 방식으로 입력 값을 받아 출력할 때 극히 작은 범위로 매핑하는데 이러한 비선형 레이어들이 여러 개 있을 경우 더 큰 효과를 발휘하여 학습이 악화된다.
* BN layer는 이러한 문제를 완화시키기 위해 사용되며 학습 전체 과정을 안정화 시킨다.
* Normalization의 방식과 유사하게 batch끼리 묶어서 normalization으로 수행
* x data의 분포를 학습이 잘되는 지점으로 이동시켜주는 것이다.
* 유사한 task로 whitening이 있는데 이는 신경망 전체의 비선형성을 훼손하기 때문에 BN이 추천된다.
* 학습의 속도를 증가시킨다.
* 전체 정확도의 영향은 크지 않다.

### Dropout layer

* 오버피팅을 방지하기 위해 사용된다.
* 노드를 random하게 삭제하는 기법이다
  + 노드를 삭제하면 연산량이 적어져 성능이 떨어진다고 생각할수도 있지만 Dropout한 것이 결과적으로 성능이 훨씬 좋게 나온다.
* 학습의 순전파 때 노드를 삭제하면 역전파 때도 동일한 노드의 전원을 꺼주며 테스트 때는 사용하지 않는다.
* Dropblock, Dropconnect도 유사한 기법이다.

### Flatten, Dense layer

* Flatten : 완전연결계층(Dense)과 연결하기 위해서 갖고 있는 feature의 shape을 변경해야 한다. 학습되는 파라미터는 없으며 입력 데이터의 shape만 변경한다.
  + 추출된 주요 특징을 전결합층에 전달하기 위해 1차원 자료로 바꿔주는 layer이다.
  + 이미지 형태의 데이터를 배열형태로 flatten하게 만들어준다.
  + 만약 input 데이터의 shape이 (batch,)로 되어있다면 flatten layer가 추가적으로 channel dimension을 추가하여 output shape이 (batch,1)이 된다.
* Dense layer : 노드들을 연결하고 학습하는 layer
  + keras에서 사용되는 레이어는 신경망 모델을 구성하는 주요한 요소이다.
  + 다층 perceptron 신경망에서 사용되는 레이어로 입력과 출력을 모두 연결해준다.
  + 각 연결선은 가중치를 포함하고 있는데 연결강도를 의미한다. 가중치가 높을수록 해당 입력뉴런이 출력뉴런에 미치는 영향이 크고, 낮을수록 미치는 영향이 작다.

### Softmax with loss

* Softmax : 입력받은 값을 정규화하여 출력. 입력값의 개수는 클래스의 개수에 따라 정해지며 출력 값의 합은1, 개수는 입력값의 개수와 동일하다.
  + 분류하고 싶은 클래스의 수 만큼 출력으로 구성
  + 가장 큰 출력값을 부여받은 클래스가 확률이 가장 높은것으로 이용된다.
* Crossentropy loss : 소프트맥스 함수의 입력 값을 받아 정답 레이블과 비교. 여기서 원핫 인코딩된 정답 데이터를 사용할 수 있다.

## Hyper parameter

### 용어 정리

* 딥러닝 모델안에서 학습되며 업데이트 되는 weight나 bias는 파라미터라고 부른다.
* hyper parameter라고 따로 부르는 이유는 사용자가 학습을 효율적으로 진행하기 위해 휴리스틱한 방식으로 모델에 설정하는 변수이기 때문이다.
* 학습률(learning rate), 배치크기(batch size), 훈련 횟수(epoch), layer설계 등이 있으며 학습 전 설정해준다.
* layer 설계
  + Dense layer 노드의 개수
  + Convolutional layer의 필터 크기, 개수, stack하는 방법
  + Dropout layer의 drop비율
  + Convolutional layer의 연결 방법 -> 다양한 CNN 신경망 방법론

# Image Classification

## 컴퓨터 비전과 신경망

* 컴퓨터 비전 : 디지털 이미지에서 정보를 자동으로 추출하는 것
  + 인공지능의 한 분야로서, 어떤 영상에서 장면이나 특징들을 ‘이해'하는 컴퓨터를 프로그램하는 것이 목적이다.
  + 영상에서 물체의 detection, segmentation, location, recognition
  + 결과의 평가
  + 같은 장면이나 물체에 대한 다른 관점의 등록
  + 연속 영상에서 물체를 추적
  + 어떤 장면을 3차원 모델로 mapping; 그 모델은 영상화된 장면을 돌아다니는 로봇에 의해 사용
  + 인간의 자세와 팔다리 움직임을 3차원으로 추정
  + 콘텐츠에 따라 디지털영상을 탐색

### Image Classification

* Computer Vision에서 핵심이 되는 task (이미지 분류 task)
* 입력 이미지가 들어왔을 때 해당 이미지가 정해진 Categories 중에서 어떤 Category에 속하는지 분류
* Viewpoint variation : 보이는 각도에 따라 달라질 수 있는 문제
* Illumination : 조명에 따라 픽셀 값의 변형
* Deformation : 객체의 형태가 변형될 때
* Occlusion : 객체가 숨어있어 일부분의 특징만 보임
* Background clutter : 주변 배경색과 객체가 비슷해 알아보기 힘들 때
* Intra-class variation : 고양이 클래스 내에 하위 분류는 어떻게 할 것인지(다양한 종류의 고양이)
* Scale variation : 객체들의 크기 변화로 인한 문제

### Image classification 성능지표

* Precision(정밀도)
  + 모델이 True라고 분류한 것 중에서 실제 True인 것의 비율
  + 정답률(Positive Predictive Value, PPV)라고도 불린다.
  + 만약 FP=0이면 결과가 무조건 1이 됨으로 꼭 1이라고 좋은 것은 아니다.
  + Precision이 0이라는 것은 False인 것을 False로는 잘 예측했지만, True인 것을 True로 잘 인식했다고 할 수는 없다.
* Recall(재현율)
  + 실제 True인 것 중에서 모델이 True라고 예측한 비율
  + Precision과 다르게 결과가 1인 나온다는 것은 True인 것은 모두 True로 잘 예측했지만 False인 것을 False로 잘 예측했다고 할 수 없다.
  + Precision은 모델의 입장, Recall은 실제 정답의 입장에서 본 관점
* Accuracy(정확도)
  + 가장 직관적으로 모델의 성능을 나타낼 수 있는 평가 지표
  + 이는 data의 bias이 발생할 경우 평가의 신뢰도가 낮아진다.
  + 다시 말해 잘못 예측한 것이 모든 데이터의 극소수라면 모델이 잘 만들어졌다고 보기 힘든 것
* F1 score
  + Precision과 Recall의 조화평균
  + 데이터 label이 불균형 구조일 때, 모델의 성능을 정확하게 평가할 수 있으며, 성능을 하나의 숫자로 표현 가능
  + precision과 recall이 0에 가까울수록 F1 score도 동일하게 낮은 값을 갖도록 함
* ROC curve
  + 모든 분류 임계값에서 분류 모델의 성능을 보여주는 그래프
  + 이 곡선 참 양성 비율(TPR)과 허위 양성 비율(FPR)을 표시
  + ROC 곡선의 점을 계산하기 위해 분류 임계값이 다른 로지스틱 회귀 모형을 여러번 평가할 수 있지만 이 방법은 효율적이지 않아 AUC라는 알고리즘이 나옴
* AUROC(=AUC)
  + ROC curver 그래프 아래의 면적값으로 최대값은 1이며 좋은 모델일수록 1에 가깝다.
  + AUC 값의 범위는 0~1
  + 척도 불변이여서 절대값이 아니라 예측이 얼마나 잘 평가되는지 측정한다.
  + 분류 임계값 불변이여서 어떤 분류 임계값이 선택되었는지와 상관없이 모델의 예측 품질을 측정
  + 하지만 척도 불변이 항상 이상적인 것은 아니다. 예를 들어 잘 보정된 확률 결과가 필요한 경우 AUC는 이 정보를 알 수 없음
  + 분류 임계값 불변의 경우 FN과 FP 비용에 큰 차이가 있는 경우 한 가지 유형의 분류 오류를 최소화하는 것은 위험할 수 있다.
  + 임상에서 사용할 수 있을 정도의 모델이라면 AUC가 적어도 0.8이상은 나와야 한다.
  + sklearn.metrics.auc(x,y)를 이용하여 계산 가능
* Fall-out
  + FPR(False Positive Rate)으로도 불리며, 실제 False인 data 중에서 모델이 True라고 예측한 비율
  + 1-Specificity -> TPR을 올리면 FPR도 증가

## CNN을 이용한 Image Classification

### Image classification Algorithm

* Image의 Feature(특징)을 찾고 Feature(특징)을 이용하여 명시적인 규칙을 만드는 방법으로 접근
* 문제점
  + 알고리즘이 강인(Robust)하지 않다. 즉, 확장성이 없어서 다른 이미지에 적용이 어려움
  + 새 이미지 분류가 아닌 다른 물체에 대해 분류할 때 해당 물체에 대한 특징을 다시 새로 입력해 줘야 함
* 문제 해결을 위해, 이 세상 존재하는 다양한 물체들에 대해 적용이 가능한 방식 필요

### Data-Driven Approach

* 모델에 방대한 양의 데이터를 입력해서 학습시키는 방법
* 즉, 새의 특징을 일일이 하나하나 입력하는 것이 아니라 방대한 양의 데이터를 입력하여 classifier가 스스로 학습하여 새임을 분류해낼 수 있게 하는 것

### CNN을 이용한 이미지 분류 방법

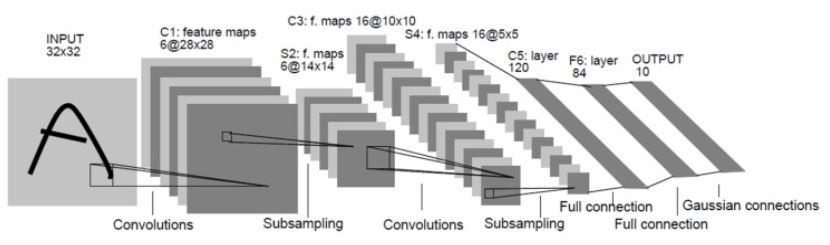
| 구분 | 설명 |
| --- | --- |
| 1. 사용자가 직접 CNN 생성 | * 직접 ConvNet2D를 여러 개 쌓아서 신경망 구축 * 단점 : 정교한 모형 구축하는데 많은 시간/데이터/컴퓨팅 파워가 필요 |
| 1. 사전학습 모델 사용 | * 사전 학습 모델 : 이미 방대한 양의 학습 데이터를 이용해서 학습을 완료한 모델 * 해당 모델의 구조와 파라미터 최적값이 학습을 통해 정해짐 * 성능이 좋은 정교한 모형을 사용할 수 있고, 오랜 시간과 컴퓨팅 파워를 사용해서 대용량 데이터를 학습한 모델을 효율적으로 사용 가능 |
| 2-1. 학습 모델 그대로 사용 | * 원래 모델을 그대로 사용하는 방식으로, 새로운 task가 사전 학습 모형의 task와 유사한 경우에 적용 가능 * 단점 : 사전학습 모형은 대부분 ImageNet대회에서 우승한 모델이기 때문에 대부분 종속변수 클래스가 동물 |
| 2-2. 전이학습(Transfer Learning) | * 사전학습 모델의 파라미터나 구조를 일부 변형해서 사용하는 방법 * 새로운 학습 데이터가 필요하지만, 완전히 모델을 새로 만드는 것 만큼 데이터가 필요하지 않음 |

### ImageNet

* 어떠한 사진을 보여주었을 때 이 사진이 무엇인지 맞출 수 있는 컴퓨터를 만드는 프로젝트
* 대표적인 대규모 데이터셋
* 전체 데이터셋에 포함된 이미지만 해도 1,000만개가 넘는다.

## CNN 모델

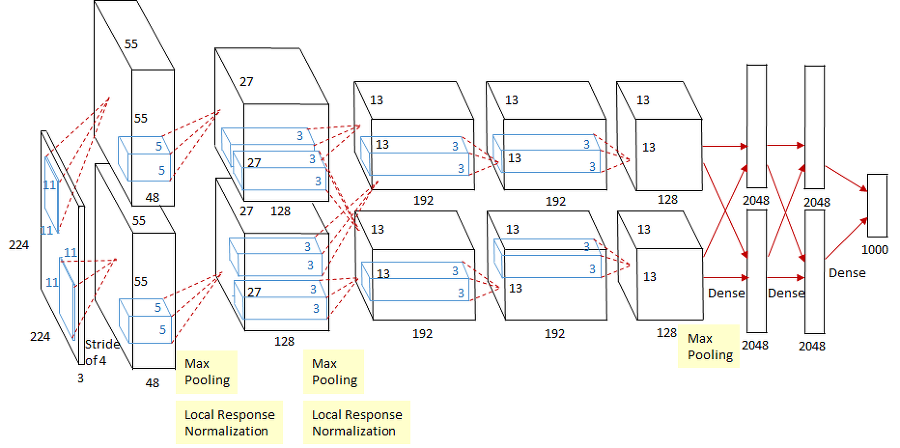
### LeNet5



* Input, 3개의 Convolution layer(C1,C3,C5), 2개의 subsampling(S2,S4), 1층의 full-connected레이어(F6), output layer로 구성되어 있다.
* 활성화함수로 tanh을 사용
* C1 레이어 : 입력 영상을 6개의 5x5 필터와 convolution 연산을 해준다. 그 결과 6장의 28x28 feature map을 얻게 된다.
* S2 레이어 : 6장의 28x28 feature map에 대해 subsampling을 진행. 결과적으로 28x28 사이즈의 feature map이 14x14 사이즈의 feature map으로 축소된다. (2x2 필터를 stride 2로 설정해서 subsampling해주기 때문) 사용하는 subsampling 방법은 평균 풀링(average pooling)이다.
* C3 레이어 : 6장의 14x14 feature map에 convolution 연산을 수행해서 16장의 10x10 feature map을 산출해낸다.
* S4 레이어 : 16장의 10x10 feature map에 대해서 subsampling을 진행해 16장의 5x5 feature map으로 축소
* C5 레이어 : 16장의 5x5 feature map을 120개 5x5x16 사이즈의 필터와 convolution 해준다. 결과적으로 120개 1x1 feature map 산출
* F6 레이어 : 84개의 유닛을 가진 feed forward NN이다. C5의 결과를 84개의 유닛에 연결
* output 레이어 : 10개의 Eucidean radial basis function(RBF) 유닛들로 구성
  + 최종적으로 class 10개를 분류하기 때문에 10개의 노드와 연결됨
  + 각각 F6의 84개 유닛으로부터 input을 받는다.
  + 최종적으로 이미지가 속한 클래스들을 알려준다.
  + 10개의 출력에서 각각의 특정 이미지일 확률을 나타낸다.

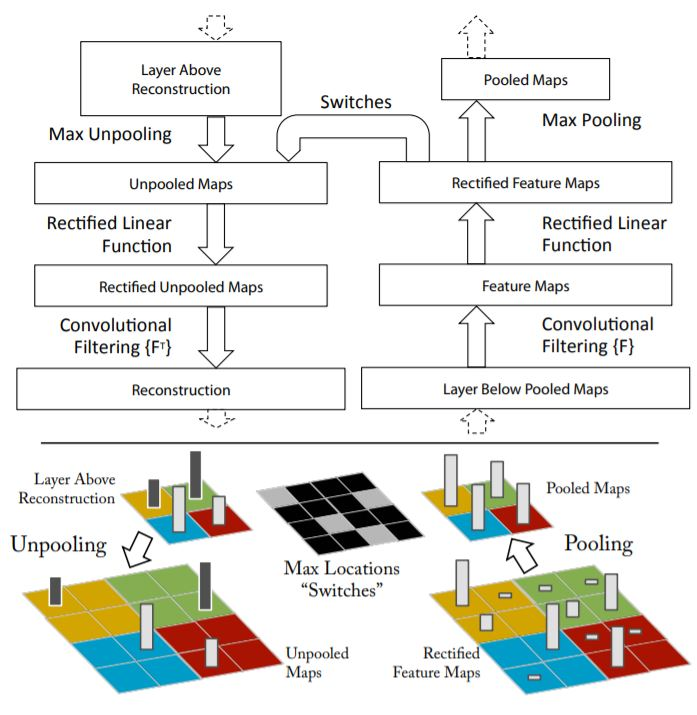
### AlexNet

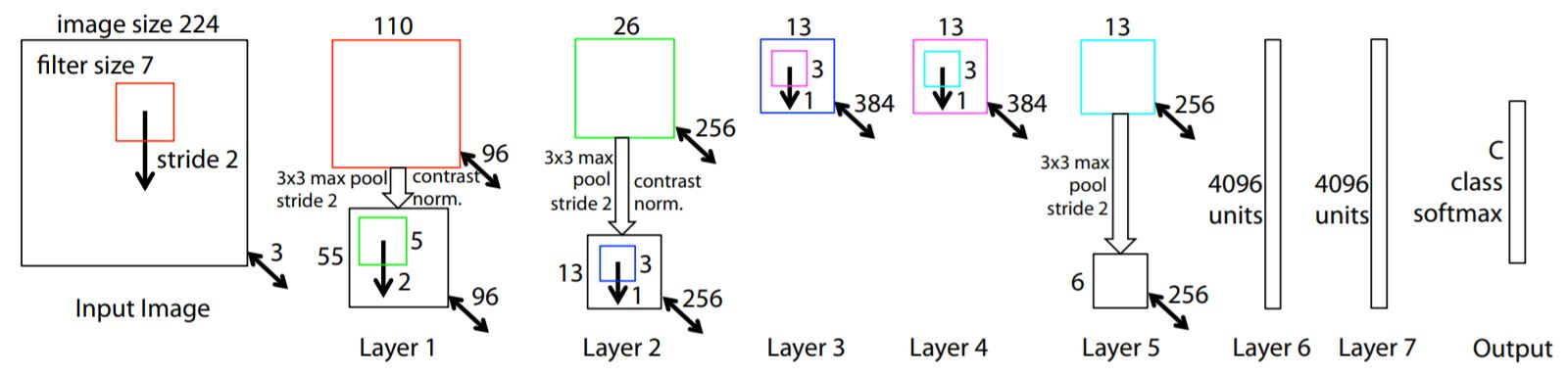
* 기본구조는 LeNet-5와 크게 다르지 않다.
* 2개의 GPU로 병렬연산을 수행하기 위해서 병렬적인 구조로 설계되었다는 점이 가장 큰 변화
* LeNet-5에서 사용되었던 tanh 함수 대신에 ReLU 함수 사용
  + ReLU : 같은 정확도를 유지하면서 tanh 사용 시보다 6배 빠름
* dropout 사용
* Overlapping pooling
* data augmentation
  + 과적합을 막기 위한 가장 좋은 방법 - 데이터 양을 늘리는 것
  + 훈련시킬 때 적은 양의 데이터를 가지고 훈련시킬 경우 과적합 될 가능성이 크기 때문



* 8개의 레이어로 구성
* 5개의 convolution 레이어와 3개의 full-connected 레이어로 구성
* 두번째,네번째,다섯번째 convolution레이어들은 전 단계와 같은 채널의 feature map들과만 연결되어 있는 반면, 세번째 convolution 레이어는 전 단계의 두 채널의 feature map들과 모두 연결되어 있다.
* 입력되는 이미지는 227x227x3 이미지 (227x227 사이즈의 RGB컬러 이미지를 뜻함)
* 첫번째 레이어(convolution layer) : 96개의 11x11x3 사이즈 필터커널로 입력 영상을 convolution해준다. convolution stride를 4로 설정했고 zero-padding은 사용하지 않았다. 결과적으로 55x55x96 feature map 산출. 그 다음 ReLU 함수로 활성화해준다. 이어서 3x3 overlapping max pooling이 stride 2로 시행된다. 그 결과 27x27x96 feature map을 갖게 된다. 그 다음에는 수렴 속도를 높이기 위해 local response normalization 시행
* 두번째 레이어(convolution layer) : 256개의 5x5x48 커널을 사용하여 전 단계의 feature map을 convolution해준다. stride는 1로, zero-padding은 2로 설정. 따라서 27x27x256 feature map을 얻게된다. 역시 ReLU 함수로 활성화한다. 그다음에 3x3 overlapping max pooling을 stride 2로 시행. 그 결과 13x13x256 feature map을 얻게된다. 그 후 local response normalization이 시행되고 feature map의 크기는 13x13x256으로 그대로 유지된다.
* 세번째 레이어(convolution layer) : 384개의 3x3x256 커널을 사용하여 전 단계의 feature map을 convolution해준다. stride와 zero-padding 모두 1로 설정한다. 따라서 13x13x384 feature map을 얻게된다. 역시 ReLU함수로 활성화한다.
* 네번째 레이어(convolution layer) : 384개의 3x3x192 커널을 사용하여 전 단계의 feature map을 convolution해준다. stride와 zero-padding 모두 1로 설정한다. 따라서 13x13x384 feature map을 얻게된다. 역시 ReLU함수로 활성화한다.
* 다섯번째 레이어(convolution layer) : 256개의 3x3x192 커널을 사용하여 전 단계의 feature map을 convolution해준다. stride와 zero-padding 모두 1로 설정한다. 따라서 13x13x256 feature map을 얻게된다. 역시 ReLU함수로 활성화한다. 그 다음에 3x3 overlapping max pooling을 stride 2로 시행. 그 결과 6x6x256 feature map을 얻게 된다.
* 여섯번째 레이어(Fully connected layer) : 6x6x256 feature map을 flatten해줘서 6x6x256=9216차원의 벡터로 만들어준다. 그것을 여섯번째 레이어의 4096개의 뉴런과 fully connected해준다. 그 결과를 ReLU함수로 활성화한다.
* 일곱번째 레이어(Fully connected layer) : 4096개의 뉴런으로 구성되어 있다. 전 단계의 4096개 뉴런과 fully connected되어 있다. 출력 값은 ReLU 함수로 활성화된다.
* 여덟번째 레이어(Fully connected layer) : 1000개의 뉴런으로 구성되어 있다. 전 단계의 4096개 뉴런과 fully connected되어 있다. 1000개 뉴런의 출력값에 softmax 함수를 적용해 1000개 클래스 각각에 속할 확률을 나타낸다.

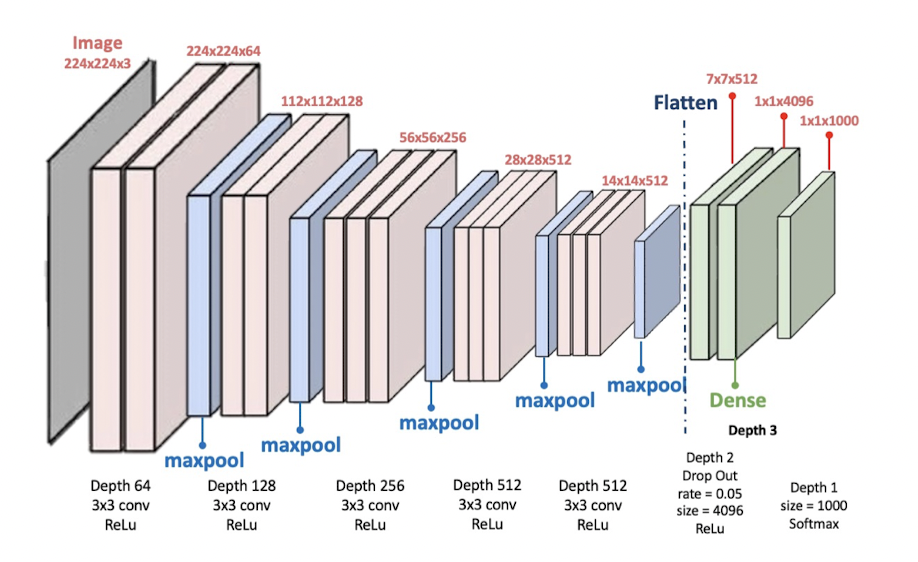
### ZFNet

* 딥러닝의 동작을 보다 더 잘 이해하기 위해서 visualizing기법을 사용했으며 network의 동작을 이해하고 성능을 개선하는데 큰 역할을 하였다.
* 일반적인 convolution 과정은 위 그림의 오른쪽과 같이 convolution filter를 거쳐 feature map을 형성하고 activation function과 pooling 과정을 거친 후 다음 convolution 과정으로 넘어가게 된다.
* 이렇게 pooling을 거친 layer를 다시 visualize 시키려면 역방향으로 똑같이 진행을 하면 되는데 문제는 unpooling시 현재 정보만을 가지고는 모든 data를 복원할 수 없다. ZFNet에서는 pooling과정에서 가장 강한 자극의 위치 정보를 switch라는 곳에 저장을하여 unpooling시 해당 위치에 대해서 왼쪽 그림과 같이 복원해낼 수 있게 구성하였다.



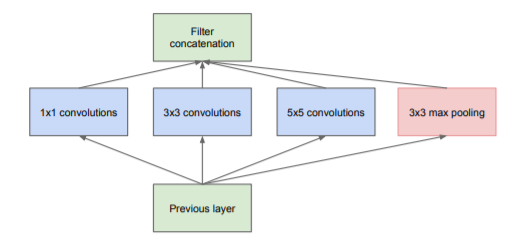
* AlexNet과 거의 동일한 구조
* 다른점
  + 첫번째 layer에서 filter size를 11에서 7로 줄였고 stride도 4에서 2로 줄였다.
  + 이 후 stride를 2로 3x3 overlapped max pooling을 적용
* 적절한 epochs 수 결정 -> 적절한 epochs에서 feature가 선명해지는 것을 확인
* 이동, 크기, 회전에 영향을 받는지에 대한 여부 확인
* 이미지에 특정 부분을 가릴 경우 인식률이 떨어지는지 확인
* 구조 변경 시에 성능 향상 여부 확인 -> feature 선명해지거나, anti-aliasing 현상 없어짐

### VGG

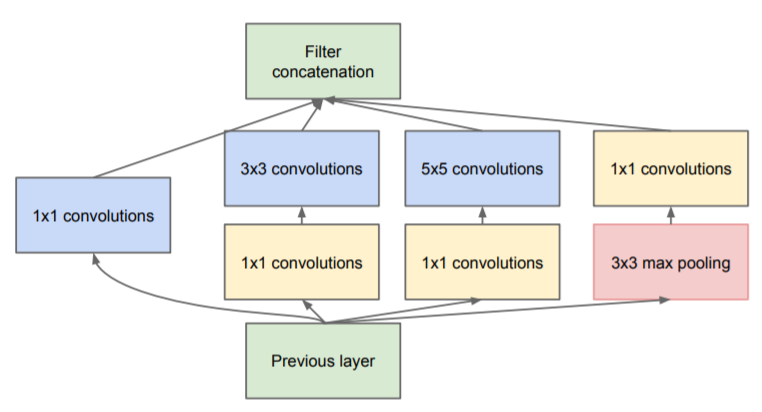


* 네트워크 깊이를 깊게 만드는 것이 성능에 어떤 영향을 미치는지 확인
* convolution layer와 pooling layer로 구성되는 기본적인 CNN
* convolution layer와 pooling layer 모두 16으로 심화한게 특징
* droupout 적용
* 깊이의 영향만을 최대한 확인하고자 convolution filter kernel 사이즈를 가장 작은 3x3으로 고정
  + 필터 커널 사이즈가 크면 이미지 사이즈가 금방 축소되기 때문에 네트워크 깊이를 충분히 깊게 만들기 불가능 하기 때문에 필터 사이즈를 작게함

### Inception(GoogleNet)



-> 초기모듈

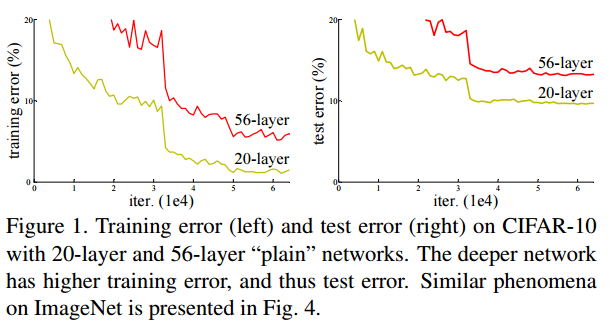


-> Inception module with dimension reductions

* 입력값에 대해 4가지 종류의 Convolution, Pooling을 수행하고, 4개의 결과를 채널 방향으로 합친다.
* feature map을 효과적으로 추출하기 위해 1x1, 3x3, 5x5의 convolution 연산을 각각 수행하며,
  + 1x1 convolution 연산을 넣어 channel을 줄였다가, 다음의 3x3, 5x5 convolution 에서 확장하는데, 이렇게 하여 필요한 연산량을 감소시킨다.
  + 1x1 합성곱 연산의 효과
    - 채널의 수 조절
    - 채널 감소 = > 파라미터 수 감소
* 3x3 Max pooling에서 입력과 출력 Matrix의 height, width를 같아야하므로 pooling연산에서 padding을 추가해준다.
  + height와 width는 모두 동일하다
* feature map을 추출하는 과정에서 최대한 Spare한 연결을 유지하고, Matrix 연산에서는 최대한 Dense하게 하고자 함

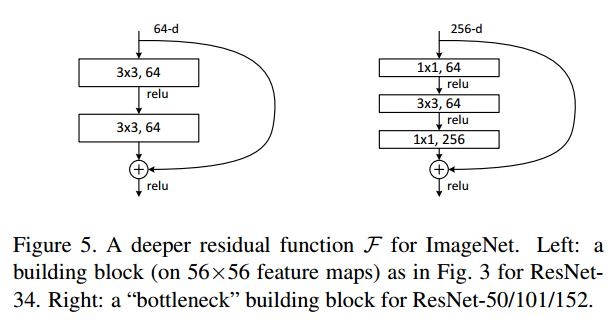
### ResNet

* skip connection을 사용하지 않은 일반적인 Plain network(CNN\_AlexNet, VGG)은 점점 깊어질수록 기울기 소실과 폭발 문제가 발생한다.
* 이를 해결하기 위해 만들어진 것이 ResNet(Residual neural Network)
* 신경망이 깊어질수록 더 정확한 예측을 할 것이라고 생각하지만 기울기 소실에 의해 error가 발생한다.



* skip connection
  + 기울기 소실/폭발 문제를 해결하기 위해 입력 x를 몇 layer 이후의 출력값에 더해줍니다.
  + 기존 신경망은 H(x) = x 로 학습
  + skip connection에 의해 output에 x를 더하고 H(x) = F(x) + x로 정의. F(x) = 0 이 되도록 학습하여 H(x) = 0 + x가 되도록 함
  + 이 방법이 최적화 하기 훨씬 쉽다.
  + 미분을 했을 때 더해진 x가 1이 되어 기울기 소실 문제가 해결된다.
  + 기울기 소실 문제가 해결되면 정확도가 감소되지 않고 신경망의 layer를 깊게 쌓을 수 있어 더 나은 성능의 신경망을 구축할 수 있다.



* 맨 아래 구조는 VGG-19. 중간 구조는 34-layer plain network. 맨 위의 구조는 ResNet이며, plain network에 skip connection이 추가되었다.
* skip connection을 추가하기 위해서는 더해지는 값 x와 출력값의 차원이 같아야한다.
* ResNet에서는 입력 차원이 출력 차원보다 작을 때 사용하는 3종류의 skip connection이 있다.
* skip connection은 증가하는 차원에 대해 추가적으로 zero padding을 적용하여 identity mapping을 수행. 따라서 추가적인 파라미터가 없다.
* Convolution layer는 3x3 kernel을 사용
* feature map 크기가 같을 경우, 모두 동일한 수의 filter로 설정
* feature map 크기가 절반이 될 경우, filter 개수 2배 증가
* Down sampling을 진행한다면 pooling 대신 stride 크기 2로 설정
* 2개의 convolution layer마다 skip connection 사용
* Bottleneck Design
* 
* 신경망이 깊어지면 학습하는데 소요되는 시간은 엄청 오래 걸린다.
* bottleneck design은 다음과 같이 신경망의 복잡도를 감소하기 위해 사용
* 1x1 conv layers는 오른쪽 그림과 같이 신경망의 시작과 끝에 추가된다.
  + 신경망의 성능을 감소시키지 않고 파라미터 수를 감소 시킨다.

# Image Detection

## Computer VIsion Task

### computer vision

* Image Classification
  + input image에 대해 label을 예측하여 분류
* Semantic Segmentation
  + input image에 대해 여러 개의 label을 예측하여 다중 분류
  + 모든 픽셀의 label을 예측
* Detection / Localization
  + 물체의 label을 예측하면서 해당 예측 값이 어디에 있는지 정보를 제공
  + object의 분류 뿐만 아니라 위치까지 탐색
  + classification과 다르게 translation variance
  + translation variance : Input의 위치가 달라져도 output이 동일한 값을 가짐

1. Max pooling
2. Weight sharing & Learn Local features
3. Compute Softmax probability

* Classification : 어떤 object를 대표하는지 분류
  + image feature를 분석하고 그에 해당되는 label에 따라 분류를 진행
  + - > translation invariance

## Image Detection

### Detection / Localization

* Semantic object instance detection
  + 찾고자하는 객체에 대한 특징(feature)를 추출하고 검출(detection)
  + 머신러닝 classifier를 사용하여 특징 추출 - > 특징 분포의 경계에서 구분
* Object classification and localization
  + 객체의 영역을 표시한 후 label로 분류된 값에 따라 classification
  + 이미지에 존재하는 좌표값(coordinates)를 구성하고 bounding box 생성
  + - > 객체의 중심을 잡은 후 가로와 세로의 길이(비율)을 생성
* Process

1. Pre-processing
2. Feature extraction
3. Classification

### Result Analysis

1. IoU

* Intersection over Union
* label box와 predicted box가 겹쳐진 정도를 토대로 detect 정도를 확인
* object detection model 훈련과정, 예측 과정 등 여러 곳에서 유용하게 쓰인다.
* R-CNN에서는 IoU가 labelling 과정에서 핵심적인 역할을 한다

1. AP

* Average Precision
* Recall(재현율)을 0부터 0.1단위로 증가시켜 1까지 증가시킬 때 Precision 값을 계산하여 평균 계산
* 0-1 사이의 모든 Recall에 대응하는 평균 Precision
* 2010년 이후 11-point interpolation에서 interpolating all data를 사용

1. mAP

* mean Average Precision
* 전체 클래스 개수에 대해 AP를 계산하여 평균값
* interpolating all data를 이용하여 AP를 구한 값의 평균
* 여러 Class에 대한 AP를 구해야 하므로 각 Class에 대해 AP를 구하고 평균을 산출

1. P-R(Precision-Recall) Curve : Confidence score를 조정하면서 얻은 Recall 값의 변화에 따른 Precision

* Recall이 높아져도 Precision이 유지되는 경우 - > 모델 성능이 좋은 것으로 평가
* Precision이 높은 경우도 좋은 모델로 평가
* class object를 모두 찾아내기 위해 Detection을 진행할 경우 일반적으로 Recall이 증가함에 따라 Precision 감소

## Detection Networks

### RCNN

* 이미지에서 1차적으로 cpu상에서 Selective Search를 진행
* 이 Search 과정을 통해 물체가 있을 법한 위치 약 2000개 정도를 찾게 되고, 이러한 2000개의 부분을 Cropping과 Resizing을 하여서 개별적으로 CNN network를 거치면서 feature vector를 추출한다.
* 이러한 feature vector들을 SVM을 통해서 Classification을 진행하고
* Regressor를 통해서 물체의 정확한 위치가 어딘지 bounding box를 예측

### Fast RCNN

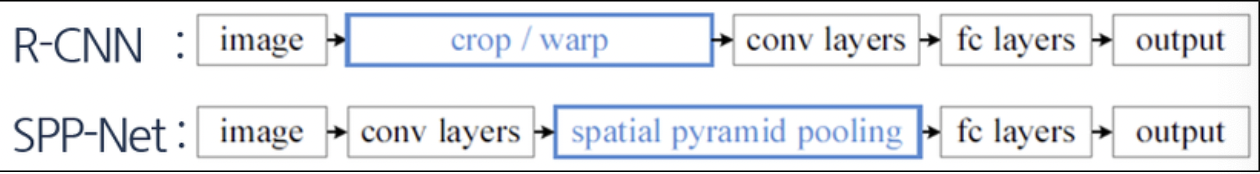
* RCNN보다 좀 더 빠른 성능
* 기존의 RCNN과 마찬가지로 Selective Search를 통해 Region Proposal을 찾는다.
* feature vector를 추출하기 위해서 CNN을 거친다.
* RCNN은 Selective Search된 이미지를 모두 CNN network를 거치지지만,
* Fast RCNN은 단 한 번만 거치면서 기존의 RCNN보다 빠른 성능을 가진다.
* ROI pooling을 통해 각각의 Region에 대해서 feature에 대한 정보를 추출한다.
* RCNN과 달리 Softmax layer을 거쳐서 각각의 class에 대한 확률 값을 구하고 이를 이용해 classification을 이뤄낸다.

### Faster-RCNN

* 기존의 RCNN과 Fast-RCNN은 Selective Search과정이 모두 cpu내에서 이뤄지기 때문에 속도가 느릴 수 밖에 없었다.
* Faster-RCNN은 GPU상에서 연산을 하기 위해 Region Proposal Network(RPN)을 제안한다. 이는 feature map을 보고 어느 곳에 물체가 있을지 예측
  + 즉, Selective Search의 시간적인 단점을 해결하는 부분
* RPN을 통해 한 번의 forwarding을 수행하면서 어느 곳에 물체가 있을 법 한지 예측이 가기 때문에 더 빠르고 정확하게 모델이 동작할 수 있다.
* 나머지는 Fast RCNN과 동일하게 Softmax 함수를 사용한다.

| RCNN | 장점 | CNN을 이용해 각 Region의 클래스를 분류할 수 있다. |
| --- | --- | --- |
| 단점 | 전체 framework를 End-to-End 방식으로 학습할 수 없다.  Global Optimal Solution을 찾을 수 없다. |
| Fast RCNN | 장점 | Feature Extraction, ROI pooling, Region Classification, Bounding box, Regression 단계를 모두 End-to-End로 묶어서 학습될 수 있다. |
| 단점 | 여전히 첫번째 Selective Search는 CPU에서 수행되므로 속도가 느리다. |
| Faster RCNN | 장점 | RPN을 제안하여, 전체 프레임워크를 End-to-End로 학습할 수 있다. |
| 단점 | 여전히 많은 컴포넌트로 구성되며, Region Classification 단계에서 각 특징 벡터(feature vector)는 개별적으로 FC layer로 Forward 된다. |

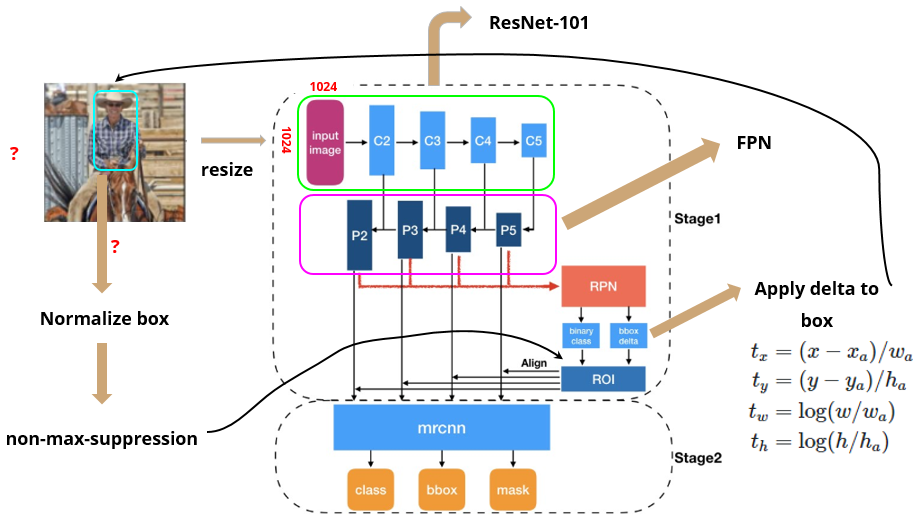
### SPP Net



* 먼저 SPP-Net은 warping으로 인한 distortion을 없애주고자 ‘spatial pyramid pooling’이라는 개념을 이용한다.
* RCNN과의 차이를 보면 warping하는 부분이 없어지고 spatial pyramid pooling이 추가 되었다.
* spatial pyramid pooling
  + First step : Conv layer 5까지 거친 feature map에 대해서 region proposal 방식(ex: selective search)을 적용하여 candidate bounding box(ROI)를 선별.
    - ROI : 여러 convolution과 max pooling layer가 있는 deep convolution network에서 얻는 고정 크기의 feature map
    - region of interest 목록을 Nx5행렬로 구성
    - 첫번째 열은 image index, 나머지 4개는 박스 모퉁이 좌표(coordinates)값
  + Second step : ROI영역에서 spatial pyramid pooling 알고리즘을 적용.
  + Third step : spatial pyramid pooling 알고리즘을 통해 1x1,2x2,4x4 spatial bin을 얻었으면 spatial bin들을 모두 flatten하게 된다. 총 16+4+1=21개의 feature들은 fc layer로 넘어간다.
* SPP-Net 장점
  + spatial pyramid pooling을 통해 RCNN에서 사용된 warping 작업을 없애 distortion을 피할 수 있었다.
  + RCNN에서는 CNN 연산을 2000번 한 것에 비해, SPP Net은 CNN 연산을 한 번만 수행하므로 training,test 시간을 굉장히 단축시킴

### Mask RCNN

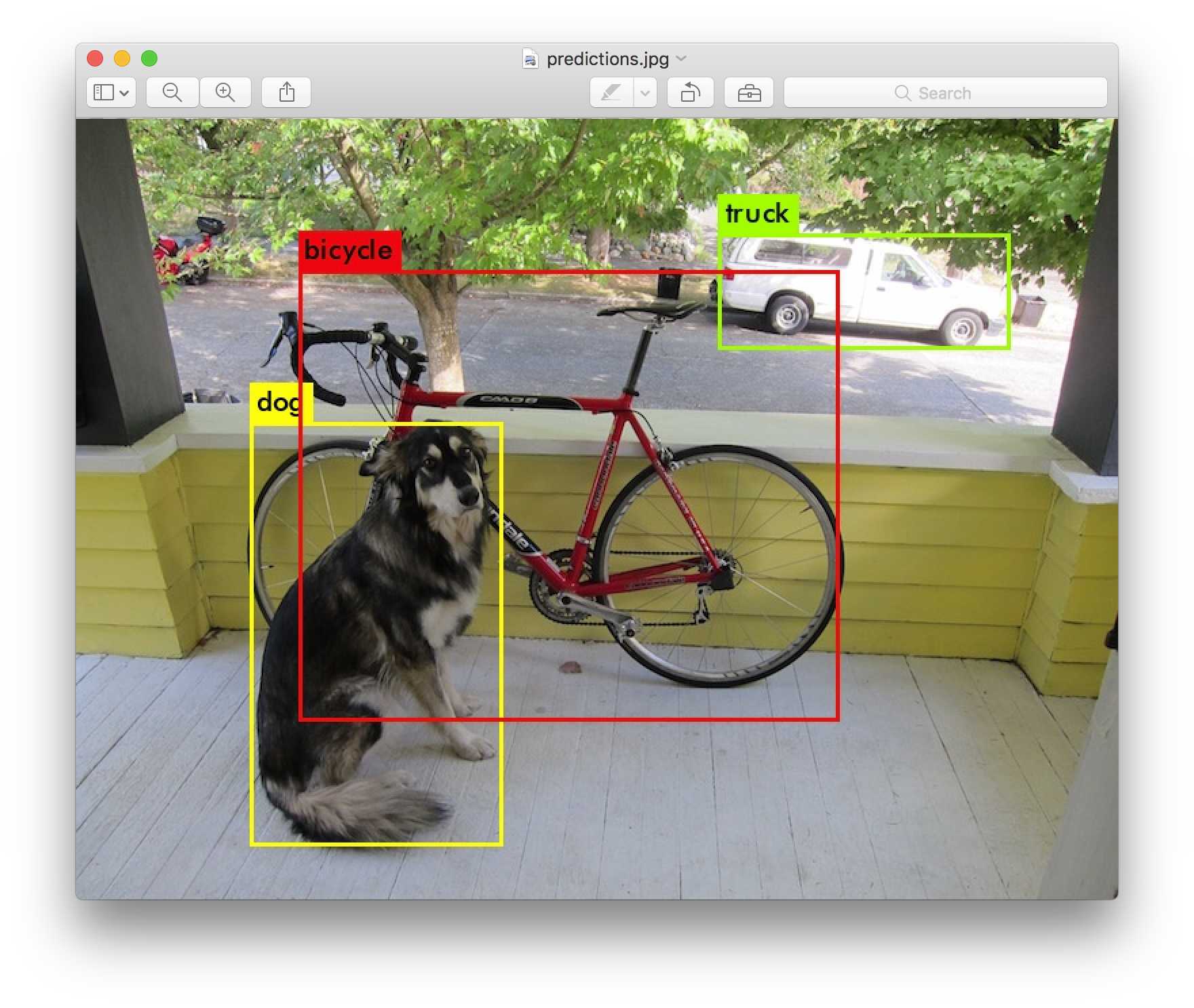
* Fast RCNN의 classification,localization(bounding box regression) branch에 새롭게 mask branch가 추가되었다.
* RPN 전에 FPN(feature pyramid network)가 추가되었다.
* Image segmentation의 masking을 위해 ROI align이 ROI pooling을 대신하게 됐다.



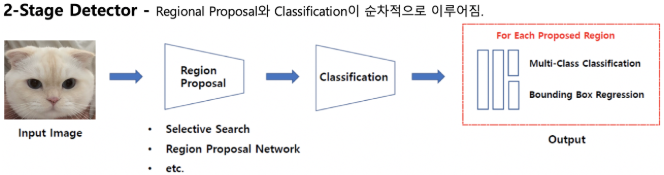
* NxN 사이즈의 input 이미지가 주어졌을때 Mask RCNN의 Process

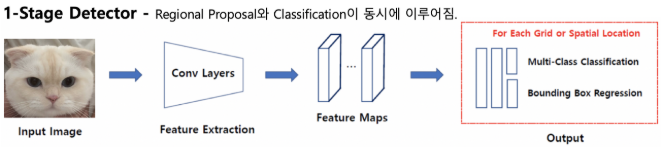
1. 800-1024 사이즈로 이미지를 resize
2. Backbone network의 input으로 들어가기 위해 1024x1024의 input사이즈로 맞춰준다.
3. ResNet을 통해 각 layer에서 feature map(C1,C2,C3,C4,C5)를 생성
4. FPN을 통해 이전에 생성된 feature map에서 P2,P3,P4,P5,P6 feature map 생성
5. 최종 생성된 feature map에 각각 RPN을 적용하여 classification,boundingbox regression output값을 도출
6. output으로 얻은 boundingbox regression값을 원래 이미지로 projection 시켜서 anchor box 생성
7. Non-max-suppression을 통해 생성된 anchor box 중 score가 가장 높은 anchor box를 제외하고 모두 삭제
8. 각각 크기가 서로 다른 anchor box들을 ROI align을 통해 size를 맞춰준다.
9. Fast RCNN에서의 classification, boundingbox regression branch와 더불어 mask branch에 anchor box값을 통과시킨다.

### YOLO



* 카메라에 비춰진 사물의 영역을 표시할 뿐만 아니라 인식된 사물이 어떤 것인지 까지 알려주는 모델
* YOLO( You Only Look Once)



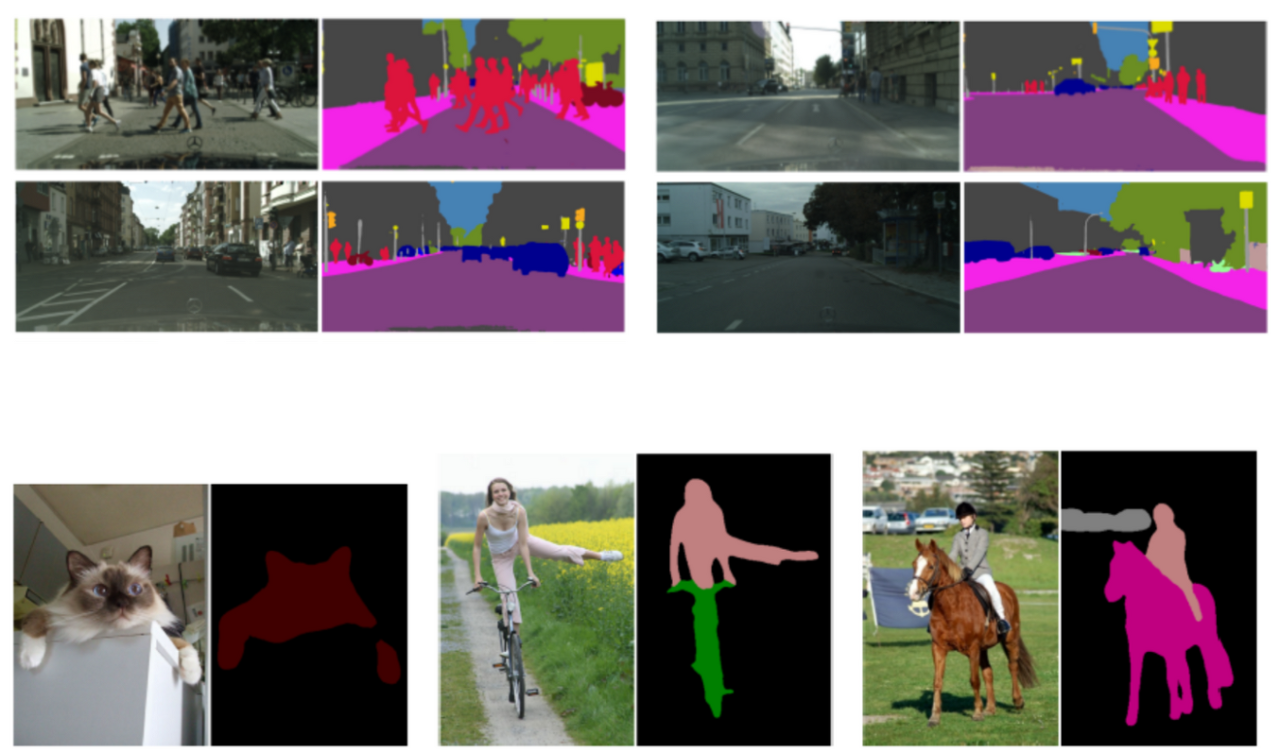


* object Detection은 classification, localization 두 가지를 합한 문제이다.
* 2-stage는 이 두 문제를 순차적으로 해결하고, 1-stage는 동시에 해결한다.
* 2-stage에 비해 1-stage가 속도면에서 더 우수한 성능을 보이지만, 정확도는 2-stage가 더 높다.
* YOLO는 1-stage에 해당된다.
* YOLO는 1-stage detector 방식을 처음으로 고안해서 실시간 객체 검출을 가능하게 한 모델
* YOLO 특징
  + 이미지 전체를 한 번만 본다. CNN처럼 이미지를 여러장으로 분할해 해석하지 않는다.
  + 통합된 모델을 사용해 간단하다.
  + 기존의 모델보다 빠른 성능으로 실시간 객체 검출이 가능하다
  + 빠르고 간단한 장점이 있지만, 작은 객체의 인식률이 떨어진다는 단점도 있다.
  + 각 그리드에 대해 그리드를 중심으로 미리 정의된 형태로 지정된 경계 박스의 개수(anchor boxes)를 예측하고 이를 기반으로 신뢰점수(confidence score)를 계산한다. 이때 높은 객체 신뢰도를 가진 위치를 선택해 객체 카테고리를 파악한다고 한다.
* YOLO 한계점
  + 각 cell은 2개의 bounding box와 하나의 class probability만 예측
    - 작은 물체들에 대해서는 예측률이 낮음
  + loss function이 작은 bounding box와 큰 bounding box에 대해 error를 동일하게 다룸
    - Scoring에 부적합
  + 몇 단계를 거친 feature map에 대해 예측
    - Localization이 부정확해질 수 있음

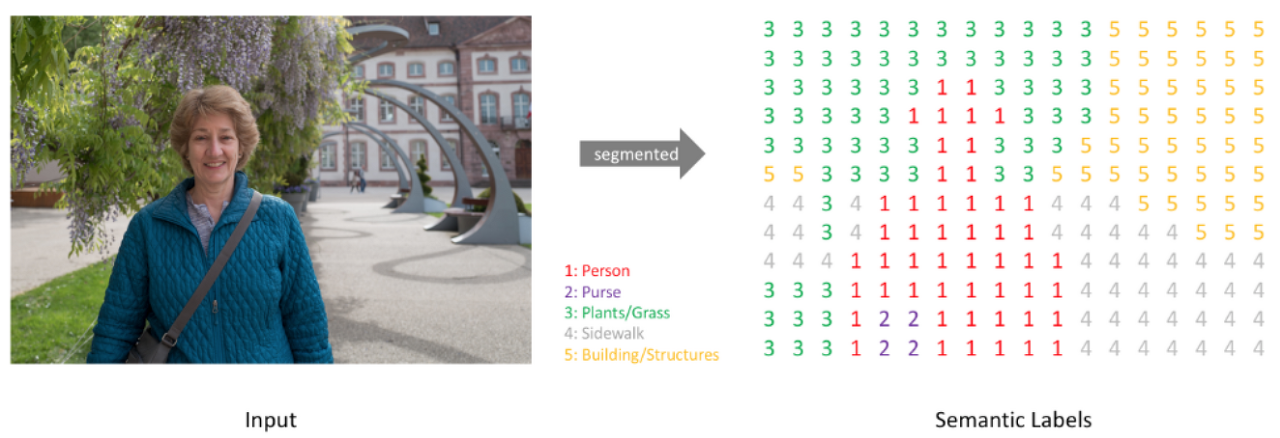
# Image Segmentation

## Semantic Segmentation

* 컴퓨터비전 분야에서 가장 핵심적인 분야



* 이미지 내에 있는 물체들을 의미 있는 단위로 분할해내는 것

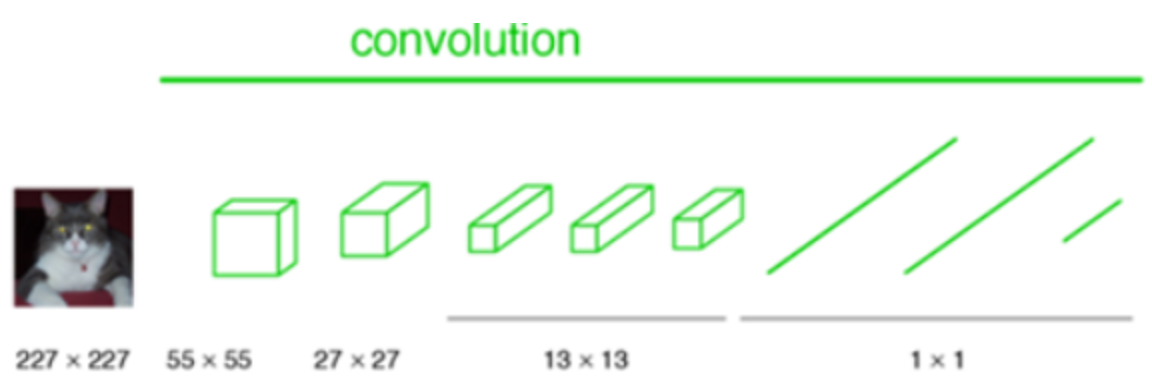


* 이미지의 각 픽셀이 어느 클래스에 속하는지 예측
* 목적 : 사진에 있는 모든 픽셀을 해당하는 class로 분류. 이미지에 있는 모든 픽셀에 대한 예측을 하는 것이기 때문에, dense prediction이라고도 불림
* Semantic Segmentation은 두가지로 나뉜다.
  + Semantic Segmentation : 픽셀 단위로 어떤 class인지만 구분
  + Instance Segmentation : 픽셀 단위로 어떤 class인지 구분한 이후, 동일한 class 내에서도 다른 Instance를 구분

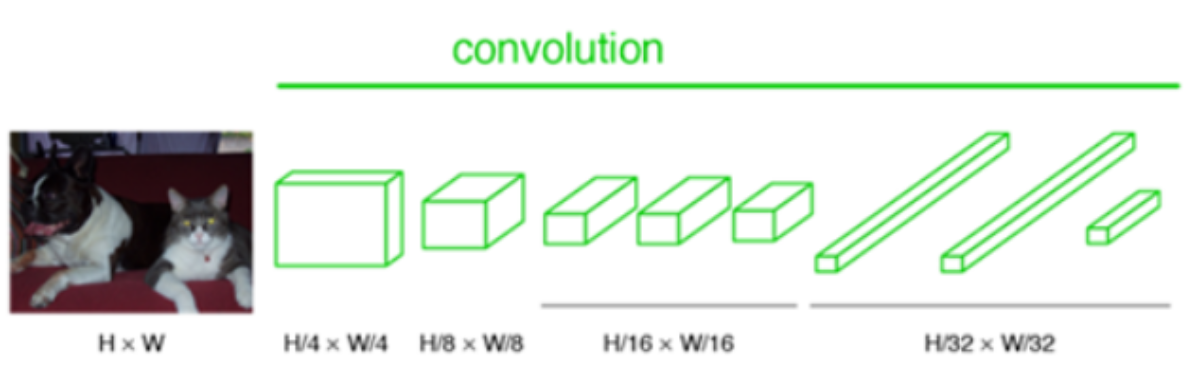
## Segmentation Networks

### Fully Convolutional Network

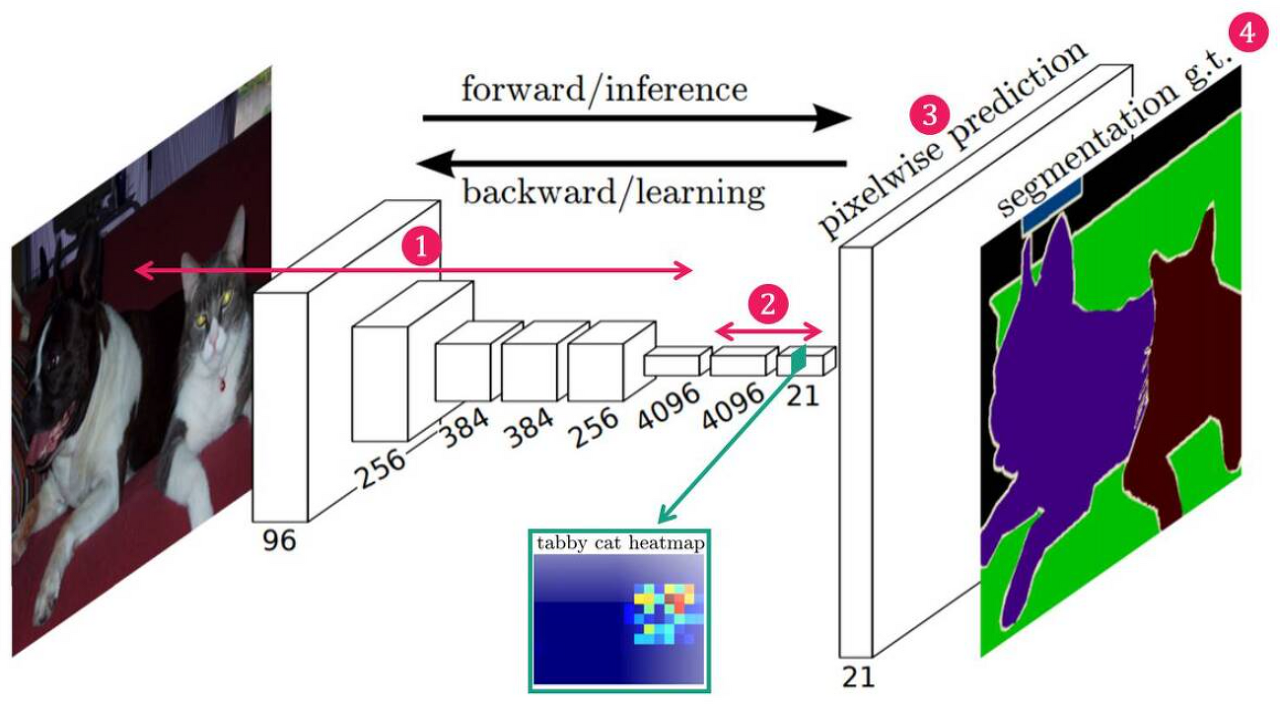
* FCN은 Semantic Segmentation의 대표적인 모델 중 하나
* 기존 Classification용 CNN 모델의 문제



* + AlexNet, VGG 등 분류에 자주 쓰이는 깊은 신경망들은, Semantic Segmentation을 하는데 적합하지 않다.
  + 이러한 모델들은 일반적으로 convolution층들과 fully connected 층들로 이루어져 있다.
  + 따라서 항상 입력이미지를 네트워크에 맞는 고정된 사이즈로 작게 만들어서 입력해줘야 한다.
  + 또한, 이러한 분류용 CNN 모델들은, 물체가 어떤 클래스에 속하는지는 예측해낼 수 있지만, parameter의 개수와 차원을 줄이는 layer들을 가지고 있어서, 자세한 위치정보를 잃게 된다.



* 이러한 문제를 해결하기 위해 FCN 모델은 fully connected 층을 1x1 convolution 층으로 바꿨다.
* 이를 통해, 네트워크 전체가 convolution층들로 이루어지게 되며,
* fully connected층들이 없어졌으므로, 더 이상 입력 이미지 크기에 제한을 받지 않게 된다.
  + 즉, 이제 어떠한 사이즈(HxW)의 이미지든 네트워크에 입력될 수 있습니다.
* 여러 층의 convolution층들을 거치고 나면, feature map의 크기가 H/32 x W/32가 되는데, 그 feature map의 한 픽셀이 입력이미지의 32x32 크기를 대표하게 된다.
  + 즉, 입력이미지의 위치 정보를 대략적으로 유지하고 있는 것
* 이 convolution 층들을 거치고 나서 얻게된 마지막 feature map의 개수는, 훈련된 class의 개수와 동일하다.
  + 5개의 클래스로 훈련된 네트워크라면, 5개의 heatmap을 산출
  + 각 feature map은 하나의 클래스를 대표



1. Convolution Layer을 통해 Feature 추출
2. 1x1 Convolution Layer을 통해, 낮은 해상도의 class presence Heat Map 추출
3. Transposed Convolution을 통해서, 이 낮은 해상도의 Heat Map을 Upsampling 한 뒤, input과 같은 크기의 map 생성
4. map의 각 pixel class에 따라 색칠 한 뒤, Segmentation 결과 반환

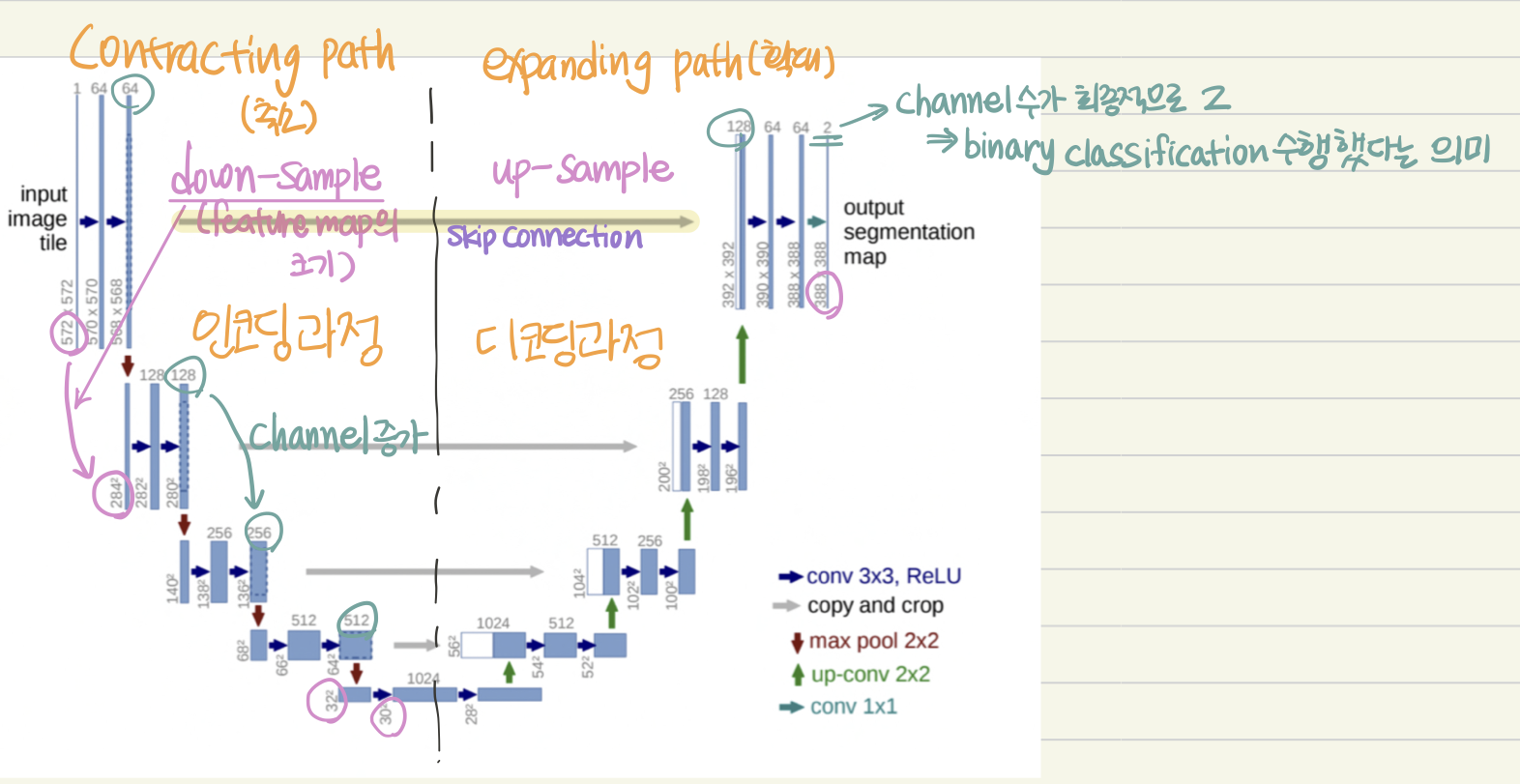
* 1.2번 과정은 downsampling 단계로, convolution을 통해 차원을 줄이는 단계
* 3번 과정은 upsampling 단계로, 1,2번 과정을 통해 만들어진 heatmap의 크기를 원래 이미지의 크기로 다시 복원해주는 단계
  + 이미지의 모든 픽셀에 대해서 클래스를 예측하는 것이, semantic segmentation의 목적이기 때문이다.
* 4번 과정은 upsampling된 heatmap들을 종합해서, 최종적인 segmentation map을 만드는 단계
  + 각 픽셀당 확률이 가장 높은 클래스를 선정해주는 것

### DeepLab

* DeepLab V1 : Atrous convolution을 처음 적용
* DeepLab V2 : multi-scale context를 적용하기 위한 Atrous spatial pyramid pooling(ASPP) 제안
* DeepLab V3 : 기존 ResNet 구조에 Atrous convolution을 활용
* DeepLab V3+ : Depthwise separable convolution과 Atrous convolution을 결합한 Atrous separable convolution을 제안
* Atrous convolution
  + classification이나 object detection에서 사용되는 신경망들은 semantic segmentation에 이용하는 경우, 여러 번의 convolution과 pooling을 거치면서 디테일한 정보가 줄어들고 특성이 점점 추상화되는 단점이 있다.
  + Atrous convolution을 이용하면 기존 convolution과 동일한 양의 파라미터와 계산량을 유지하면서도, receptive field는 커진다.
* ASPP
  + DeepLab V2에서 제안된 방법으로, feature map으로부터 확장비율(r)이 다른 atrous convolution을 병렬로 적용한 뒤 다시 합쳐주는 방법이다.
* Depthwise separable convolution
  + 기존 convolution 필터가 Spatial dimension과 channel dimension을 동시에 처리 하던 것을 따로 분리시켜 각각 처리하는 방법으로 해석 가능
  + 두 축을 분리시켜 처리를 하더라도, 최종 결과값은 두 축 모두를 처리한 것이므로 기존 convolution이 수행하던 역할을 대체할 수 있다.
  + 그러나 기존 convolution 연산과 비교했을 때 사용하는 파라미터 수와 연산량을 획기적으로 줄일 수 있기 때문에 이런 구조를 사용하고 있다.

### U-Net

* U-Net은 Semantic Segmentation Task 수행에 널리 쓰이는 모델 중 하나이다.



* U-Net의 장점
  + 기존의 Segmentation 모델의 단점이었던 느린 연산 속도를 개선
  + 속도 개선이 가능했던 이유는, 이미지를 인식하는 단위(Patch)에 대한 Overlap 비율이 적기 때문
  + 기존의 모델에서 많이 사용되었던 Sliding Window 방식의 경우, 이전 Patch에서 검증이 끝난 부분을 다음 Patch에서 다시 검증하게 됩니다. 이는 일종의 연산 낭비라고 볼 수 있다.
  + U-Net의 경우, 이전 Patch에서 검증이 끝난 부분을 다음 Patch에서 중복하여 검증하지 않기 때문에, 연산의 낭비가 없고 이로 인해 향상된 속도를 얻을 수 있다.
  + Segmentation Network는 클래스 분류를 위한 인접 문맥 파악(Context)과 객체의 위치 판단(Localization)을 동시에 수행해야 한다.
  + U-Net은 다층의 Layer의 Output을 동시에 검증해서 이러한 Trade-Off를 극복

# RNN

## 시계열 데이터 분석 방법론

### 전통적 시계열 데이터 분석 방법론

* MA(Moving Average)
* Exponential smoothing
* ARIMA(Auto Regressive Integrated Moving Average)
* S-ARIMA
* Trigonometric model
* Growth Curve Model
* Time series regression with autocorrelation

### 머신러닝 기반 시계열 데이터 분석 방법론

* Support Vector Machine(or Regression)
* Random, Forest
* Boosting
* Gaussian Process
* Hidden Markov model(HMM)

### AI 기반 시계열 데이터 분석 방법론

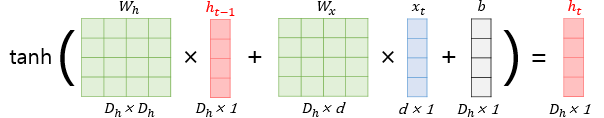
* RNN(Recurrent Neural Network)
* LSTM
* GRU
* Sequence-to-Sequence / Sequence-to-Sequence with Attention
* CNN and variants
* Transformer
* GPT-1/BERT/GPT-3

## RNN

* 입력과 출력을 시퀀스 단위로 처리하는 시퀀스 모델
* 단방향으로 진행되는 신경망이 아닌 재귀적으로 순환하여 출력 결과를 입력 값으로 다시 받아 사용할 수 있는 신경망

|  | CNN | RNN |
| --- | --- | --- |
| 특화된 사용처 | image와 같은 grid 형태의 데이터 | 시계열 같은 sequence 형태의 데이터 |
| 크기 | 넓고 긴 image | 긴 sequence |
| 연산 가능 | 다양한 크기의 image 데이터 | 다양한 길이의 sequence 데이터 |

* 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할을 하는 노드를 cell이라고 한다.
  + 이 셀은 이전의 값을 기억하려고 하는 일종의 메모리 역할을 수행하므로 이를 메모리 셀 또는 RNN cell이라고 표현
* 은닉층의 메모리 셀은 각각의 시점(time step)에서 바로 이전 시점에서의 은닉층의 메모리 셀에서 나온 값을 자신의 입력으로 사용하는 재귀적 활동을 하고 있다.
* 메모리 셀이 출력층 방향 또는 다음 시점인 t+1의 자신에게 보내는 값을 은닉 상태(hidden state)라고 한다.
  + 다시 말해 t 시점의 메모리 셀은 t-1 시점의 메모리 셀이 보낸 은닉 상태값을 t 시점의 은닉 상태 계산을 위한 입력값으로 사용
* RNN은 입력과 출력의 길이를 다르게 설계할 수 있으므로 다양한 용도로 사용 가능
  + 하나의 입력에 대해서 여러개의 출력을 의미하는 one-to-many 구조의 모델은 하나의 이미지 입력에 대해서 사진의 제목을 출력하는 image Captioning 작업에 사용 가능
  + 단어 시퀀스에 대해서 하나의 출력을 하는 many-to-one 구조의 모델은 입력 문서가 긍정적인지 부정적인지를 판별하는 감성 분류(sentiment classification), 또는 메일이 정상 메일인지 스팸 메일인지 판별하는 스팸 메일 분류 등에 사용
  + many-to-many 구조의 모델의 경우에는 사용자가 문장을 입력하면 대답 문장을 출력하는 챗봇과 입력 문장으로부터 번역된 문장을 출력하는 번역기, 또는 개체명 인식이나 품사 태깅과 같은 작업이 속한다.
* 배치 크기가 1이고, d와 RNN의 은닉층 연산



* 이때 를 계산하기 위한 활성화 함수로는 주로 tanh가 사용된다.
* 각각의 가중치 의 값은 하나의 층에서는 모든 시점에서 값을 동일하게 공유
* 하지만 은닉층이 2개 이상일 경우에는 각 은닉층에서의 가중치는 서로 다르다.
* 출력층은 결과값인 를 계산하기 위한 활성화함수로는 푸는 문제에 따라 다를텐데, 예를 들어
  + 이진 분류를 해야하는 경우라면 출력층에 로지스틱 회귀를 사용하여 sigmoid 함수를 사용
  + 다중 클래스 분류를 해야하는 경우라면 출력층에 softmax 회귀를 사용하여 softmax 함수를 사용

### BPTT

* RNN에서 계산되는 back propagation으로 sequential data의 특성으로 인해 발생하는 hidden state를 따라 역행하면서 전파되는 gradient의 계산 방법이다.
* Back Propagation Through Time의 약자이며 단순한 역전파 계산과 동일
* 긴 시계열 데이터를 학습할 때의 문제를 해결할 필요가 있다.
  + 시계열 데이터의 시간 크기가 커지는 것에 비례하여 BPTT 소비자원도 크게 소모 + 역전파 시의 기울기가 불안정(기울기 폭발, 기울기 소멸)
* Truncated BPTT
* 너무 길어진 신경망을 적당한 지점에서 잘라내어 작은 신경망 여러 개로 생성
  + Feed-Forward : 유지하여 전달
  + Backpropagation : 끊어서 학습
* 순전파가 연결되어 있기 때문에 데이터를 순서대로 입력

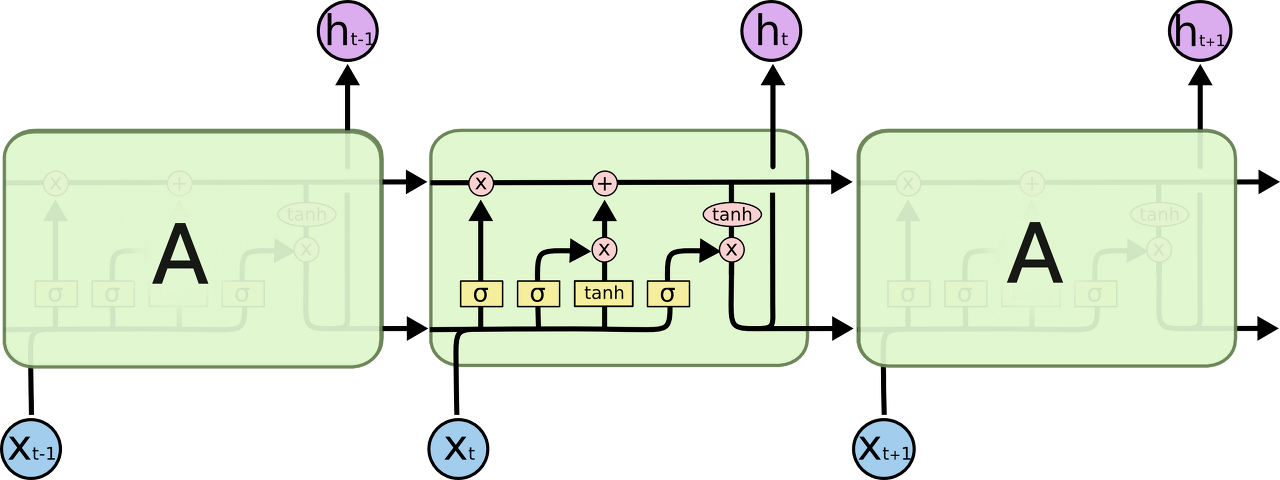
## Gated RNN

### RNN 장기 의존 관계

* 시계열 데이터에서 시간적으로 멀리 떨어진 장기 의존(long term) 관계를 학습하기 어려움
  + BPTT에서 기울기 소실 또는 기울기 폭발이 발생
  + 단순한 RNN 계층이 아닌 LSTM, GRU 계층을 이용하여 한계점 해결
* 기울기 소실 Gradient Vanishing
  + 역전파 과정에서 입력층으로 갈수록 기울기가 작아짐
  + 이로 인해 가중치가 제대로 업데이트 되지 않아 최적 값을 만들 수 없게 되는 현상
  + CNN의 경우 activate function을 ReLU or Leaky ReLU를 사용
  + RNN의 경우 Gradient Clipping과 게이트를 추가한 신경망 구성
* 기울기 폭발 Gradient Exploding
  + 기울기 소실과 반대로 점차 증가하여 비정상적으로 큰 값이 되어 발산
  + 마찬가지로 가중치 업데이트에 문제가 발생
  + 기울기 클리핑 - 기울기 자르기 : 임계치를 넘어가지 않도록 크기를 감소시키는 방법

### LSTM

* RNN의 특별한 한 종류로, 긴 의존 기간을 필요로 하는 학습을 수행할 능력을 갖고 있다.
* 모든 RNN은 NN모듈을 반복시키는 체인과 같은 형태를 하고 있다.
* LSTM도 똑같이 체인과 같은 구조를 가지고 있지만, 각 반복 모듈은 다른 구조를 갖고 있다. 단순한 neural network layer 한 층 대신에, 4개의 layer가 특별한 방식으로 서로 정보를 주고 받도록 되어 있다.

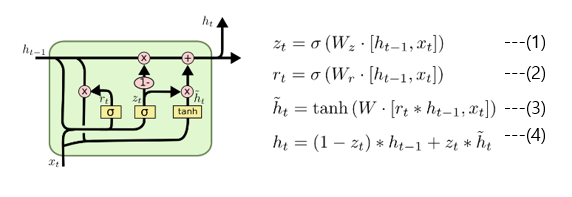


* LSTM의 핵심은 cell state인데, 모듈 그림에서 수평으로 그어진 윗 선에 해당한다.
* Cell state는 컨베이어 벨트와 같아서, 작은 linear interaction만을 적용시키면서 전체 체인을 계속 구동시킨다. 정보가 전혀 바뀌지 않고 그대로 흐르게만 하는 것은 매우 쉽게 할 수 있다.
* cell state에 뭔가를 더하거나 없앨 수 있는 능력이 있는데, 이 능력은 gate라고 불리는 구조에 의해서 제어된다.
  + gate는 정보가 전달될 수 있는 추가적인 방법으로, sigmoid layer와 pointwise 곱셈으로 이루어져 있다.
* LSTM은 3개의 gate를 가지고 있고, 이 gate들은 cell state를 보호하고 제어한다.
* LSTM의 첫 단계로는 cell state로부터 어떤 정보를 버릴 것인지를 정하는 것으로, sigmoid layer에 의해 결정된다. 그래서 이 단계의 gate를 “forget gate”라 부른다. 의 기억 중 불필요한 정보를 버릴 것인지 결정
* input gate : 새로운 정보가 입력된 셀 노드에서 각 원소의 가치가 얼마나 큰지 판단. 무비판적 정보 수용이 아닌 적절한 데이터를 선발
  + sigmoid layer가 어떤 값을 업데이트할지 정한다. 그 다음에 tanh layer가 새로운 후보 값들의 vector를 만들고, cell state에 더할 준비를 한다.
* output gate : tanh에 게이트를 적용하여 각 원소에 대해 다음 시각의 은닉 상태에 대한 중요도를 조정
* 행렬 곱 연산이 아닌 원소별 곱 연산이 이뤄지므로 역전파 시 곱셈 결과가 누락되지 않으므로 기울기 소실이 일어나지 않도록 함. 역전파가 진행될 때 연산을 forget 게이트에서 제어

## GRU

Gated Recurrent Unit

* LSTM에서는 출력, 입력, 삭제 게이트라는 3개의 게이트가 존재.
* 반면, GRU에서는 업데이트 게이트와 리셋 게이트 두 가지 게이트만 존재
* GRU는 LSTM보다 학습 속도가 빠르다고 알려져 있지만 여러 평가에서 GRU는 LSTM과 비슷한 성능을 보인다고 알려져 있다.

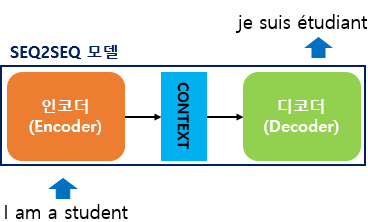


* Reset gate
  + (2)식에 해당. 이전 시점의 hidden state와 현 시점의 x를 활성화 함수 sigmoid를 적용하여 구하는 방식 .
  + 결과값은 0-1 사이의 값을 가질 것이며 이전 hidden state의 값을 얼마나 활용할 것인지에 대한 정보로 해석할 수 있다.
  + reset gate에서 나온 값은 그대로 사용되는 것이 아니라 (3)식으로 다시 활용. (3)식에서는 전 시점의 hidden state에 reset gate를 곱하여 계산
* Update gate
  + LSTM의 input,forget gate와 비슷한 역할을 하며 과거와 현재의 정보를 각각 얼마나 반영할지에 대한 비율을 구하는 것이 핵심
  + (1)식을 통해서 구한 결과 z는 현재 정보를 얼마나 사용할지 반영
  + 그리고 (1-z)는 과거 정보에 대해서 얼마나 사용할지 반영
  + 최종적으로 (4)식을 통해 현 시점의 출력값 hidden state를 구할 수 있다.

## Attention

### Sequence-to-Sequence

* 번역기에서 대표적으로 사용되는 모델



* 크게 인코더와 디코더라는 두 개의 모듈로 구성
* 인코더는 입력 문장의 모든 단어들을 순차적으로 입력받은 뒤에 마지막에 이 모든 단어 정보들을 압축해서 하나의 벡터로 만드는데, 이를 context vector라고 한다. 입력 문장의 정보가 하나의 context vector로 압축되면 인코더는 context vector를 디코더로 전송. 디코더는 context vector를 받아서 번역된 단어를 한 개씩 순차적으로 출력
* 디코더는 기본적으로 RNNLM(RNN Language Model)이다.
* 디코더는 인코더로부터 context vector를 넘겨받는다. 첫 입력으로 문장의 시작을 의미하는 <s>가 들어간다.
* 디코더의 첫번째 RNN셀은 context vector와 <s>, 이 2개의 입력을 바탕으로 새로운 Hidden State를 계산하고 이를 Affine 계층과 Softmax 계층을 거친다.
  + Affine 계층은 Hidden State를 입력으로 받아 분류 개수를 출력해주는 feed forward network이다.
* Sequence-to-Sequence의 한계점
  + 긴 sequence 입력 값에 대하여 context가 너무 작아 데이터를 모두 담는 것에 한계가 발생(차원의 문제)
  + 디코더에서 출력 단어를 예측하는 매 시점마다 인코더에서의 전체 입력 값을 다시 참고
* Attention mechanism 적용
  + fixed된 context vector를 사용하는 것이 아닌 variable-length sequence 형태로 context 전달
  + output의 길이와 같은 context vector를 생성
  + 학습하고자 하는 output의 출력값(t시점)에 초점을 맞춰서 주의를 기울이는 것(attention)이 목표

## Transformer

기존 seq2seq 모델의 한계

* 인코더는 입력 시퀀스를 하나의 벡터 표현으로 압축하고, 디코더는 이 벡터 표현을 통해서 출력 시퀀스를 만들어냈다.
* 하지만 이러한 구조는 인코더가 입력 시퀀스를 하나의 벡터로 압축하는 과정에서 입력 시퀀스의 정보가 일부 손실된다는 단점이 있었고, 이를 보정하기 위해 attention이 사용되었다.

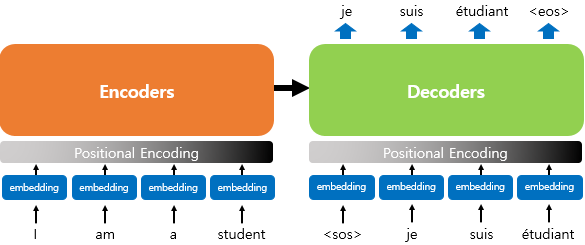
### Transformer의 주요 하이퍼파라미터

* = 512
  + 트랜스포머의 인코더와 디코더에서 정해진 입력과 출력의 크기를 의미함.
  + 임베딩 벡터의 차원 또한 이며, 각 인코더와 디코더가 다음 층의 인코더와 디코더로 값을 보낼 때에도 이 차원을 유지합니다.
* \_ = 6
  + 트랜스포머에서 하나의 인코더와 디코더를 층으로 생각하였을 때, 트랜스포머 모델에서 인코더와 디코더가 총 몇 층으로 구성되었는지를 의미.
* \_=8
  + 트랜스포머에서는 attention을 사용할 때, 한 번 하는 것보다 여러개로 분할해서 병렬로 attention을 수행하고 결과값을 다시 하나로 합치는 방식을 택함.
  + 이 때 이 병렬의 개수를 의미
* =2048
  + 트랜스포머 내부에는 feed forward 신경망이 존재하며 해당 신경망의 은닉층의 크기를 의미
  + feed forward 신경망의 입력층과 출력층의 크기는 입니다.

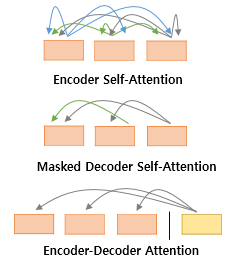
### Transformer



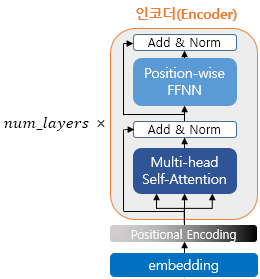
* 트랜스포머는 RNN을 사용하지 않지만 기존의 seq2seq처럼 인코더에서 입력 시퀀스를 입력받고, 디코더에서 출력 시퀀스를 출력하는 인코더-디코더 구조를 유지하고 있습니다.
* 이전 seq2seq 구조에서는 인코더와 디코더에서 각각 하나의 RNN이 t개의 시점(time step)을 가지는 구조였다면 이번에는 인코더와 디코더라는 단위가 N개로 구성되는 구조
* Positional Encoding
  + RNN이 자연어 처리에서 유용했던 이유는 단어의 위치에 따라 단어를 순차적으로 입력받아서 처리하는 RNN 특성으로 인해 각 단어의 위치 정보를 가질 수 있다는 점에서 있었다.
  + 하지만 트랜스포머는 단어 입력을 순차적으로 받는 방식이 아니므로 단어의 위치 정보를 다른 방식으로 알려줄 필요가 있다.
  + 트랜스포머는 단어의 위치 정보를 얻기 위해서 각 단어의 임베딩 벡터에 위치 정보들을 더하여 모델의 입력으로 사용하는데, 이를 positional encoding이라고 한다.



### Attention



* 첫번째 그림인 self- attention은 인코더에서 이루어지지만
* 두번째 그림인 self-attention과 세번째 그림인 인코더-디코더 attention은 디코더에서 이루어진다.
* self-attention은 본질적으로 Query, Key, Value가 동일한 경우를 말한다.
* 반면, 세 번째 그림 인코더-디코더 attention에서는 Query가 디코더의 벡터인 반면에 Key와 Value가 인코더의 벡터이므로 self-attention이라고 부르지 않는다.

 인코더의 구조

* 트랜스포머는 하이퍼파라미터인 \_ 개수의 인코더 층을 쌓는다.
* 인코더를 하나의 층이라는 개념으로 생각한다면, 하나의 인코더 층은 크게 총 2개의 sublayer으로 나뉘어진다.
* self-attention과 feed forward 신경망이다.
* Multi-head Self-Attention은 self-attention을 병렬적으로 사용하였다.
  + Multi-head Self-Attention : Attention 수행을 1회가 아닌 여러 번 시행
* 인코더의 self-attention
  + attention 함수는 주어진 쿼리에 대해서 모든 키와의 유사도를 각각 구한다. 그리고 구해낸 이 유사도를 가중치로 하여 키와 맵핑되어있는 각각의 값에 반영해준다. 그리고 유사도가 반영된 값을 모두 가중합하여 리턴한다.
  + Q = Query : t 시점의 디코더 셀에서의 은닉 상태
  + K = Key : 모든 시점의 인코더 셀의 은닉 상태들
  + V = Values : 모든 시점의 인코더 셀의 은닉 상태들
* self-attention은 입력 문장의 단어 벡터들을 가지고 수행한다고 하였는데, 사실 self-attention은 인코더의 초기 입력인 의 차원을 가지는 단어 벡터들을 사용하여 self-attention을 수행하는 것이 아니라 우선 각 단어 벡터들로부터 Q벡터, K벡터, V벡터를 얻는 작업을 거친다.
* 이때 이 Q벡터,K벡터,V벡터들은 초기 입력인 의 차원을 가지는 단어 벡터들보다 더 작은 차원을 가진다.
* Scaled dot-product Attention
  + Q벡터는 모든 K벡터에 대해서 attention score를 구하고, attention 분포를 구한 뒤에 이를 사용하여 모든 V벡터를 가중합하여 attention 값 또는 context벡터를 구하게 된다. 그리고 이를 모든 Q벡터에 대해서 반복한다.

# GAN

* 경찰
* 위조지폐범
* 최대한 진짜 같은 데이터를 생성하려는 생성 모데로가 진짜와 가짜를 판별하려는 분류모델이 각각 존재하여 서로 적대적으로 학습
* 적대적 학습에서는 분류 모델을 먼저 학습시킨후, 생성 모델을 학습시키는 과정을 서로 주고받으면서 반복.
* 분류 모델의 학습은 크게 두 가지 단계로 이루어져 있다.
* 하나는 진짜 데이터를 입력해서 네트워크가 해당 데이터를 진짜로 분류하도록 학습시키는 과정이고
* 두번째는 첫번째와 반대로 생성 모델에서 생성한 가짜 데이터를 입력해서 해당 데이터를 가짜로 분류하도록 학습하는 과정이다.
* 이 과정을 통해 분류 모델은 진짜 데이터를 진짜로, 가짜 데이터를 가짜로 분류할 수 있게 됩니다.
* 이와 같은 학습과정을 반복하면 분류 모델과 생성 모델이 서로 적대적인 경쟁자로 인식하여 모두 발전하게 된다.
* 결과적으로, 생성 모델은 진짜 데이터와 완벽히 유사한 가짜 데이터를 만들 수 있게 되고 이에 따라 분류 모델은 진짜 데이터와 가짜 데이터를 구분할 수 없게 된다.
* 즉, GAN은 생성 모델은 분류에 성공할 확률을 낮추려 하고, 분류 모델은 분류에 성공할 확률을 높이려 하면서 서로가 서로를 경쟁적으로 발전 시키는 구조를 이루고 있다.

# Reinforcement Learning

### 강화학습이란

* 강화는 시행착오를 통해 학습하는 방법 중 하나를 의미합니다. 이러한 강화를 바탕으로 강화학습은 실수와 보상을 통해 학습을 하여 목표를 찾아가는 알고리즘입니다.
* 기존의 신경망들이 라벨이 있는 데이터를 통해 가중치와 편향을 학습하는 것과 비슷하게 보상이라는 개념을 사용하여 가중치와 편향을 학습하는 것입니다.
* 목적은 최적의 행동양식 또는 정책을 학습하는 것이다.

### 특징

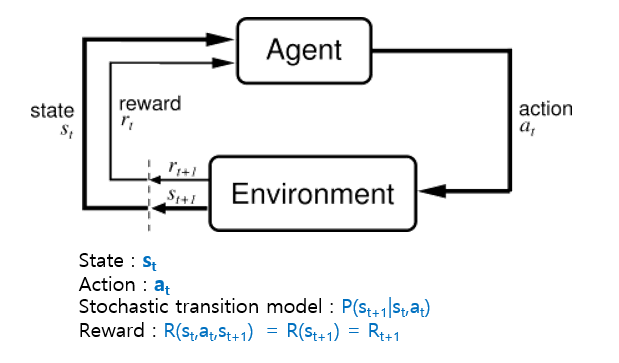
* 환경에 대한 사전지식이 없는 상태로 학습 진행
* 보상을 통하여 학습
* 컴퓨터가 선택한 행동에 대한 환경의 반응에 따라 보상이 주어진다.
* 행동의 결과로 나타나는 것이 보상이며 이것을 통하여 학습을 진행
* 보상을 최대한 많이 얻도록 하는 행동을 유도하도록 학습 진행

### 강화학습이 해결 가능한 문제

* 강화학습은 결정을 순차적으로 내려야 하는 문제에 적용을 합니다. 순차적으로 내려야 하는 문제를 정의하기 위해서는 MDP(Markov Decision Process)를 사용

1. 순차적 행동 문제를 MDP로 전환
2. 가치함수를 벨만 방정식
3. 최적 가치함수와 최적 정책을 찾는다
4. MP : MP는 이산 시간이 진행함에 따라 상태가 확률적으로 변화하는 과정을 의미. 시간 간격이 이산적이고 현재의 상태가 이전 상태에 영향을 받는 확률 과정이다. 이것을 기초로하여 MDP가 정의된다.

### MDP



* 상태(State) : 정적인 요소 + 동적인 요소를 의미
* 행동(Action) : 어떠한 상태에서 취할 수 있는 행동을 의미
* Stochastic transition model : 어떤 상태에서 특정 행동을 하여 다음 상태에 도달할 확률
* 보상(Reward) : Agent가 학습할 수 있는 유일한 정보를 의미. 어떤 상태에서 행동을 하여 다음 상태가 되고, 이때 받는 보상값은 다음 상태가 되는 것에 대한 보상
* 정책(Policy) : 순차적 행동 결정 문제(MDP)에서 구해야할 답을 의미. 모든 상태에 대해 Agent가 어떠한 Action을 해야 하는지 정해놓은 것을 의미
* 목표는 최적의 정책(Optimal Policy)을 찾는 것이다.

### 가치함수(Value Function)

* 순차적인 결정 문제를 풀기 위해서는 가치함수을 정의해야 한다. 가치함수는 현재 상태의 정책을 따라갔을 때 얻는 예측 보상의 총 합을 의미. 이때 현재 보상의 추세인 감가율을 고려하여 미래 보상을 예측한다.
* 에이전트는 가치함수를 통해서 보상의 합을 최대로 한다는 목표에 얼마나 다가갔는지를 판단.
* 가치함수에 대한 방정식은 벨만 방정식이다.

### 벨만 방정식(The Bellman Equation)

* 벨만 기대 방정식은 현재 상태와 다음 상태의 관계를 나타내는 방정식입니다. 두 가지 종류가 존재
* 벨만 기대 방정식 : 반복적으로 기대값을 업데이트하기 위해서, 현재와 다음의 가치함수 사이 관계를 정의합니다. 특정 정책일 때의 가치함수 사이의 관계를 의미. 모든 상태에 대한 가치함수를 반복적으로 업데이트하여 참 가치함수값이 나옵니다. 현 정책에 대한 수렴값을 구할 수 있다.
* 벨만 최적 방정식 : 가장 큰 가치함수를 구하는 정책을 최적 정책이라고 합니다. 벨만 기대 방정식에 맞춰 더 좋은 정책을 찾아내면 해당 정책이 바로 최적의 정책입니다. 이때, 최적의 가치함수 값을 내는 것을 최적 가치 함수라고 하며, 그 때의 정책이 최적정책입니다. 이때 가치함수들 사이의 관계식을 벨만 최적 방정식이라고 합니다.
* 벨만 방정식은 다이나믹 프로그래밍(Dynamic Programming, DP)으로 작은 문제들로 쪼개서 풀게 된다. 각 상태에서의 가치함수를 구하고 업데이트를 하고, 업데이트 이후의 가치함수를 다시 구하는 방식으로 반복합니다.

### 다이나믹 프로그래밍(DP)

* + 정책 이터레이션 (Policy Iteration) : 벨만 기대 방정식을 이용하여 문제를 푸는 방식
  + 초기에는 무작위로 지정된 정책을 시작으로, 업데이트를 반복하여 정책을 평가하고, 발전해나갑니다. 이를 반복하는 방법입니다.
  + 정책의 평가 방법은 가치함수를 이용하는 것인데, k번째 값을 이용하여 k+1번째 값을 업데이트 하는 방식입니다. 주변 상태들을 함께 살피고, 주변 상태의 가치함수를 사용하여 감가율을 곱하고 보상을 더해서 가능한 모든 행동에 대한 확률을 곱하고 더하는 순서로 업데이트를 진행
* 가치 이터레이션(Value Iteration) : 벨만 최적 방정식을 이용하여 문제를 푸는 방식
* 정책 이터레이션을 이용하여 최적의 가치함수를 얻을 때까지 반복하는 방식

### 정책(Policy)

모든 상태에서 에이전트가 할 행동을 의미. 최적의 정책은 부분 수열 상태의 기대값이 최대가 되는 정책입니다.

### Q-function

* 행동에 대한 가치를 나타낸 것
* 상태에 대한 가치가 아닌 상태에서 특정한 행동을 했을 때의 가치를 구하게 되는 것
* Q함수를 이용하면 행동에 대한 가치함수 값을 보고 어떤 행동을 할지 판단하면 되기 때문에. 상태들의 가치를 판단하고 어떤 행동을 했을 때 특정 다음 상태로 가게 될 확률도 고려해야 하는 번거로움이 없어진다.
* 즉, Q함수는 특정 상태 s에서 특정 행동 a를 취했을 때 받을 반환값에 대한 기댓값으로 특정 행동 a를 했을 때 얼마나 좋을 것인지에 대한 값이다.

## Monte Carlo Prediction

* 강화학습과 다이내믹 프로그래밍의 차이
  + 강화학습 : 환경의 모델을 모르는 상태에서도 환경과의 상호작용을 통해 최적 정책 학습
  + 강화학습의 학습 방법 : 시스템의 입력과 출력 사이의 관계를 학습. 사람이 학습하는 방법과 유사하게 강화학습도 진행
* 강화학습에서의 예측 종류
  + 몬테카를로 예측
  + 시간차 예측
* 강화학습에서의 제어 종류
  + 시간차 제어 : 살사
  + 오프폴리시 제어 : Q Learning

### 몬테카를로 예측

* 몬테카를로 근사
  + ‘몬테카를로'라는 말은 무엇인가를 무작위하게 측정하는 것을 의미한다. 통계적인 추정이며 sample을 통해 원래의 값에 대해 추정하는 것이다.
  + 원의 방정식을 사용하지 않고 원의 넓이를 구하는 방법
    - 종이 위에 무작위로 점을 계속 찍는다.
    - 그 점들 중에는 원 안에 찍힌 점도 있을 것이고 원 밖에 찍힌 점도 있을 것이다. 이 점을 찍는 행위를 무수히 많이 하여 빼곡하게 점이 찍혔다면 이 점들의 비율로 넓이를 추정할 수 있다. (원 안의 점 개수) / (전체 찍은 점 개수)를 통해 구할 수 있다.
* 몬테카를로 예측
  + 몬테카를로 근사를 이용해 참 가치함수를 추정하는 것이 몬테카를로 예측이다.
  + 즉, 샘플링을 통해서 얻은 샘플의 평균으로 참 가치함수 값을 추정한다. 위에서 예시로 든 원의 넓이 구하는 몬테카를로 근사에서 점을 찍는 행위가 샘플링이었다. 그렇다면 가치함수를 구하는 몬테카를로 예측에서 샘플링은 무엇일까?
  + 바로 에피소드를 한 번 진행하는 것이다.
  + 다시 말해 몬테카를로 예측은 에피소드를 여러 번 진행해서 얻은 샘플들의 평균을 통해 참 가치함수에 수렴해가는 과정이다.
  + DP의 정책 이터레이션에서 정책 평가는 벨만 기대 방정식을 이용해서 가치함수를 ‘계산'하는 과정이었다.
  + 벨만 기대 방정식에 포함되어 있는 기댓값을 계산하려면 환경 모델을 알아야 하기 때문에, 몬테카를로 예측에서는 환경을 몰라도 계산할 수 있는 반환값을 이용한다. 현재 정책에 따라 계속 행동을 하면 그에 따른 보상을 받고 반환값을 계산할 수 있다.
  + 따라서 몬테카를로 예측에서는 여러 에피소드를 통해 구한 반환값의 평균을 통해 가치함수를 추정한다.

## SARSA

SARSA란 일반화된 정책반복(Generalized Poliicy Iteration)과 시간차 학습(TD)가 접목된 알고리즘이다.

* 일반화된 정책반복이란, 정책향상을 하기 위해서 굳이 정책 평가를 끝까지 해보지 않고 적정양의 정책 평가만으로도 정책 향상을 할 수 있다는 개념이었다.
* 이 과정은 시간차 학습과 만나 더 강력해질 수 있다.
* 단 한 스텝의 정책 평가 만으로도 정책 향상을 이룰 수 있다.

### Expected SARSA

* SARSA 알고리즘은 다음 상태에서 취할 하나의 행동만을 샘플링 한다.
* 하지만 우리의 에이전트는 정책을 따라 행동하기 때문에 다음에 취할 행동들의 확률이 얼마인지 미리 알 수 있다.
* 다음 상태의 모든 행동들을 고려할 수 있다.
* SARSA는 다음 행동들 중 하나만 샘플링했지만, 다음의 모든 행동들을 고려한다.
* 따라서 SARSA보다는 비용이 크고 느리게 갱신되지만, 보다 안정적인 정책 갱신이 가능하다.

### Q-Learning

* 모델이 없이 학습하는 강화학습 알고리즘이다.
* 목표는 유한한 마르코프 결정과정에서 에이전트가 특정 상황에서 특정 행동을 하라는 최적의 정책을 배우는 것. 현재 상태로부터 시작하여 모든 연속적인 단계들을 거쳤을 때 전체 보상의 예측값을 극대화시킨다. 이것은 한 상태에서 다른 상태로의 전이가 확률적으로 일어나거나 보상이 확률적으로 주어지는 환경에서도 별다른 변형 없이 적용될 수 있다.
* 또한 Q라는 단어는 현재 상태에서 취한 행동의 보상에 대한 quality를 상징

### 