딥러닝
영상분류/영상인식 입문

임도형, dh-rim@hanmail.net

진행 일정

13:00 – 13:30	딥러닝 소개	딥러닝 개요 및 딥러닝 동작방식 이해
13:30 – 14:10	영상인식 작업 소개	딥러닝을 사용한 영상관련 작업 소개
14:10 – 14:30	실습 환경 준비	실습 환경 설명
14:40 – 16:00	영상분류 실습	VGG16, ResNet을 사용한 영상분류 이해 및 실습
16:10 – 18:00	영상인식 실습	YOLO, SSD를 사용한 영상인식 이해 및 실습

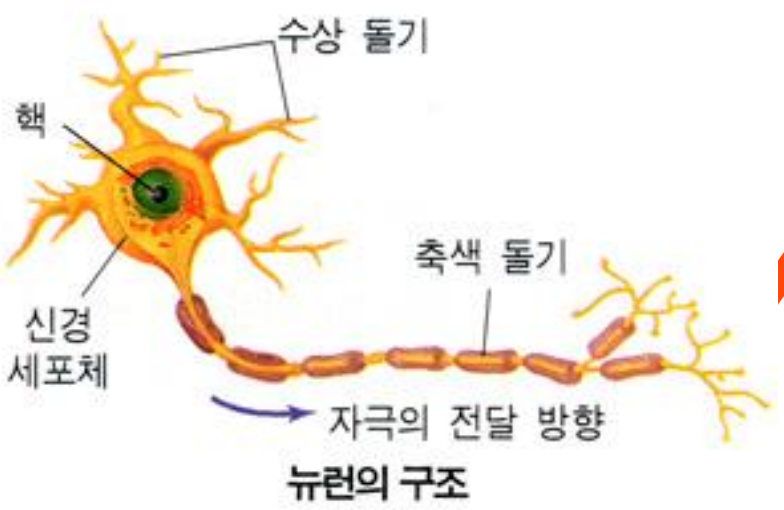
신경망

신경세포

여러 개의 수상돌기에서 자극이 합해져서

그 값이 어느 값 이상일 경우

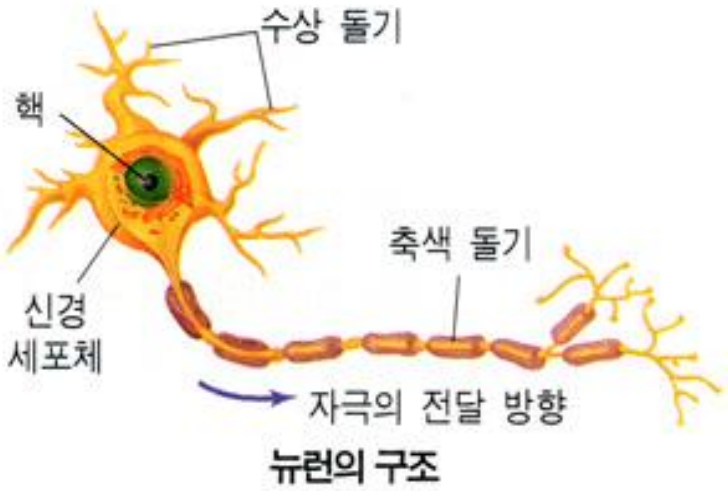
축색돌기로 자극을 발생시킨다.



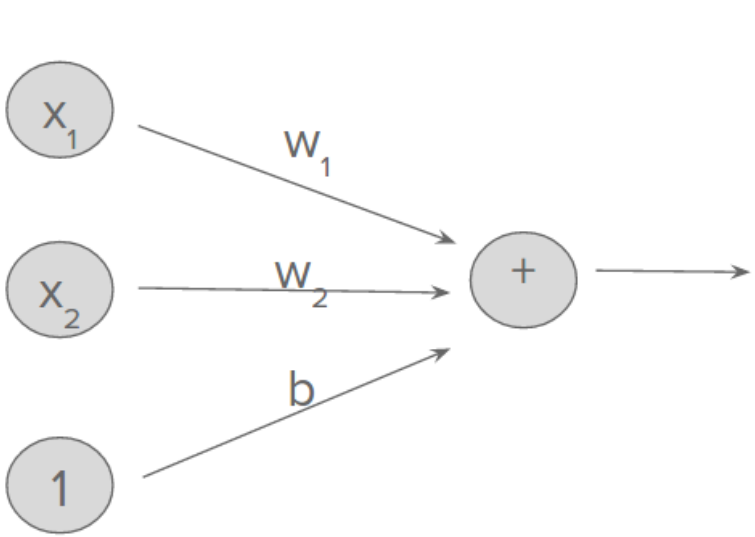
뉴런의 구조

http://m.blog.daum.net/lpds417/17453443

신경세포와 유사



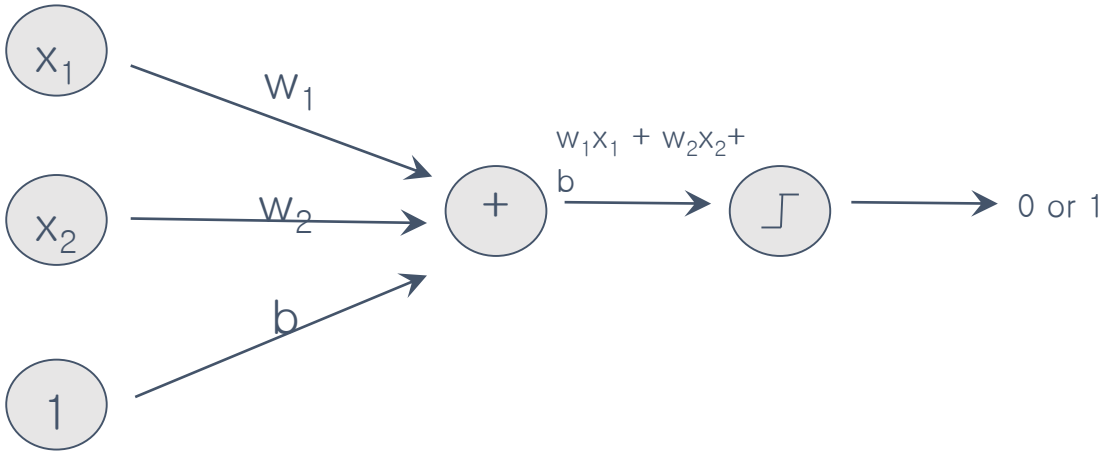
뉴런의 구조



http://m.blog.daum.net/lpds417/17453443

퍼셉트론(Perceptron)

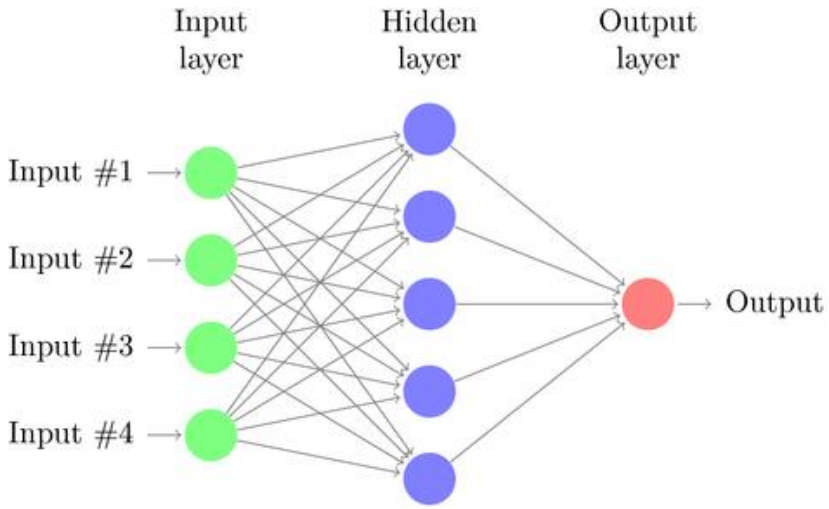
입력에 가중치를 곱하고 전부 더한 후 활성화 함수의 결과를 출력한다.



MLP(Multi Layer Perceptron)

입력과 출력 사이에 층이 더 있다.

개별 perceptron의 결과를 다음 층의 입력으로 사용한다.



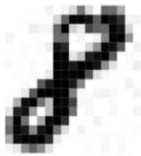
http://docs.opencv.org/2.4/modules/ml/doc/neural_networks.html

숫자 인식의 예

이미지를 구성하는 pixel의 각 값으로 구성된 입력벡터를 NN의 입력으로 한다.

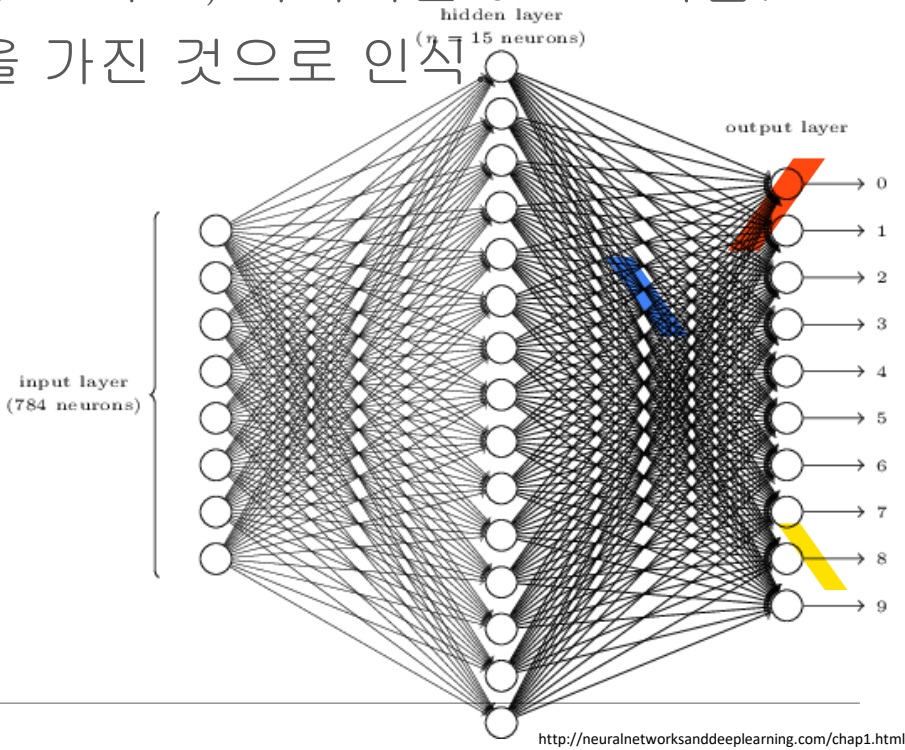
그리고 학습시 해당 출력 노드만 1로 하고, 나머지는 0으로 학습.

test 시에는 출력 노드중 최대 값을 가진 것으로 인식



=

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 12, 0, 11, 39, 137, 37, 0, 152, 147, 84, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 41, 160, 250, 255, 235, 162, 255, 238, 206, 11, 13, 0, 0, 0, 0, 16, 9, 9, 150, 251, 45, 21, 184, 159, 154, 255, 233, 40, 0, 0, 10, 0, 0, 0, 0, 0, 145, 146, 3, 10, 0, 11, 124, 253, 255, 107, 0, 0, 0, 0, 3, 0, 4, 15, 236, 216, 0, 0, 38, 109, 247, 240, 169, 0, 11, 0, 1, 0, 2, 0, 0, 0, 253, 253, 23, 62, 224, 241, 255, 164, 0, 5, 0, 0, 6, 0, 0, 4, 0, 3, 252, 250, 228, 255, 255, 234, 112, 28, 0, 2, 17, 0, 0, 2, 1, 4, 0, 21, 255, 253, 251, 255, 172, 31, 8, 0, 1, 0, 0, 0, 0, 0, 4, 0, 163, 225, 251, 255, 229, 120, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0, 21, 162, 255, 255, 254, 255, 126, 6, 0, 10, 14, 6, 0, 0, 9, 0, 3, 79, 242, 255, 141, 66, 255, 245, 189, 7, 8, 0, 0, 5, 0, 0, 0, 0, 26, 221, 237, 98, 0, 67, 251, 255, 144, 0, 8, 0, 0, 7, 0, 0, 11, 0, 125, 255, 141, 0, 87, 244, 255, 208, 3, 0, 0, 13, 0, 1, 0, 1, 0, 0, 145, 248, 228, 116, 235, 255, 141, 34, 0, 11, 0, 1, 0, 0, 0, 1, 3, 0, 85, 237, 253, 246, 255, 210, 21, 1, 0, 1, 0, 0, 6, 2, 4, 0, 0, 0, 6, 23, 112, 157, 114, 32, 0, 0, 0, 0, 2, 0, 8, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]



https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721#.ylp7hd52a

http://neuralnetworksanddeeplearning.com/chap1.html

함수 근사화 능력

DNN의 능력

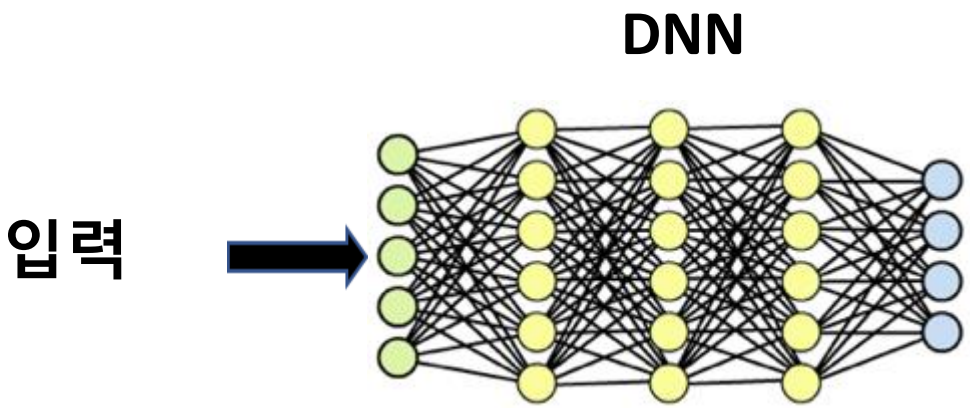
Universal Approximator

어떠한 함수도 근사화 할 수 있다.

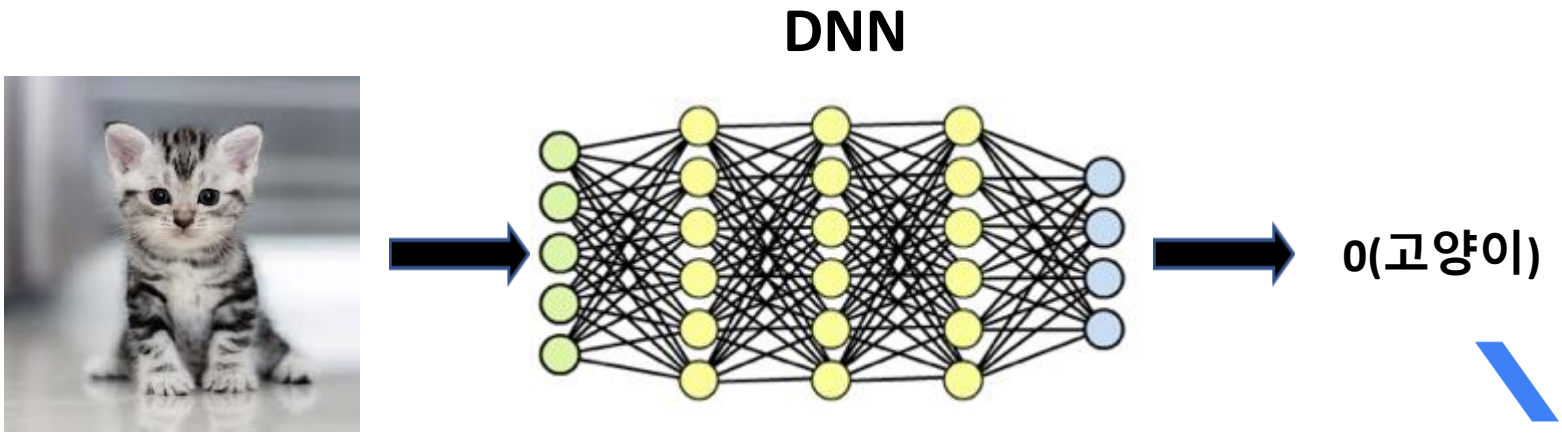
DNN의 함수 근사화 능력

- DNN은 임의의 함수를 근사화 할 수 있다.
- 함수의 내부를 모르더라도.
- 입력과 출력 데이터로.

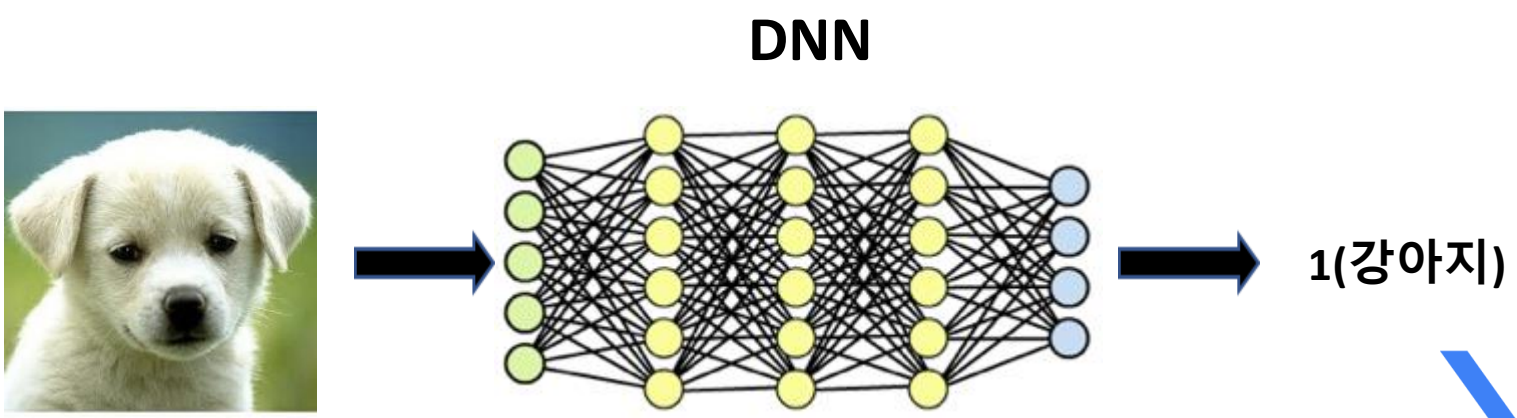
함수



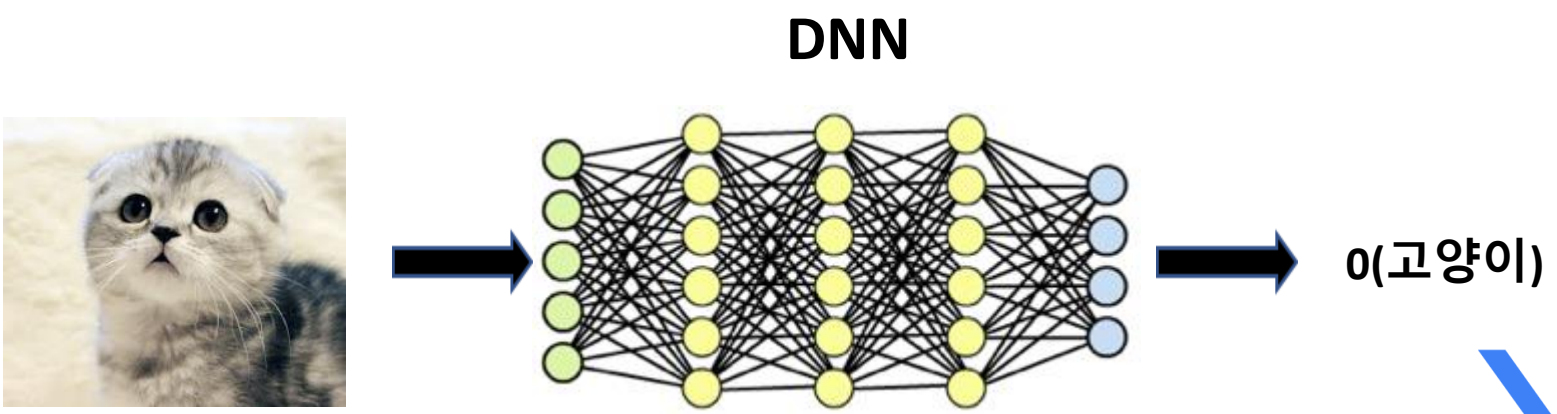
학습



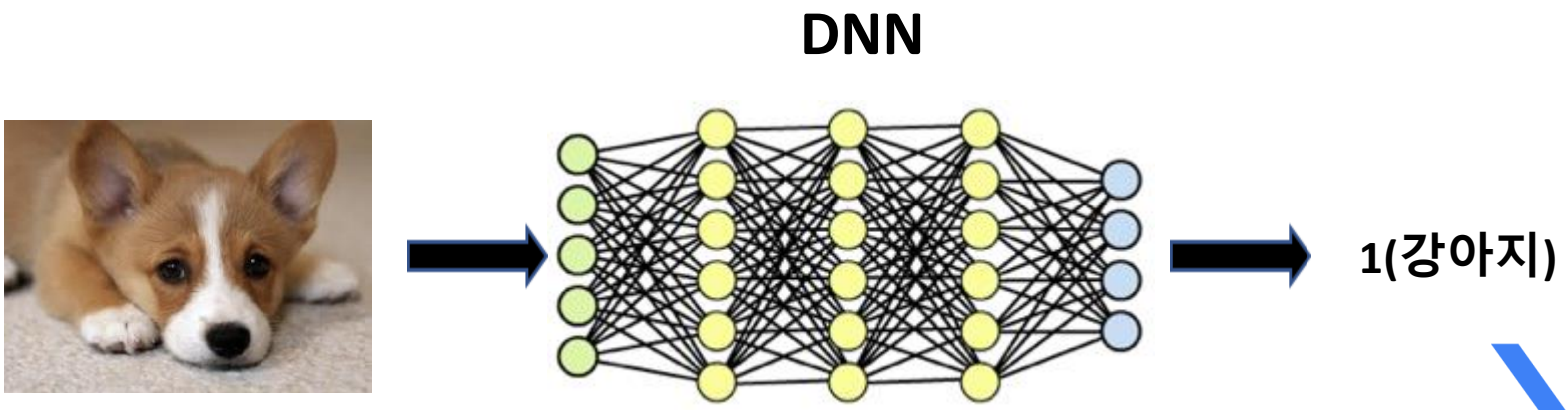
학습



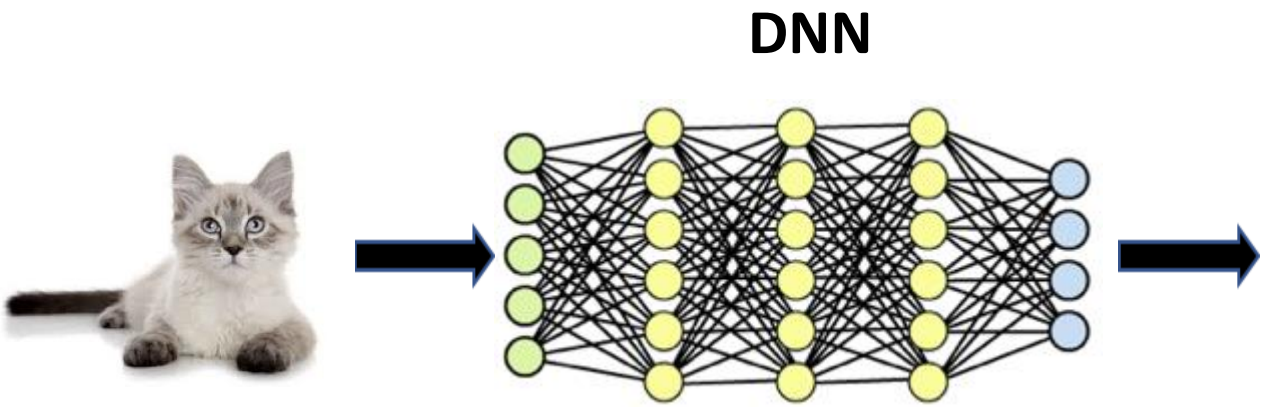
학습



학습



학습



처음 보는 입력이라도 출력을 낸다. 의미 있는.

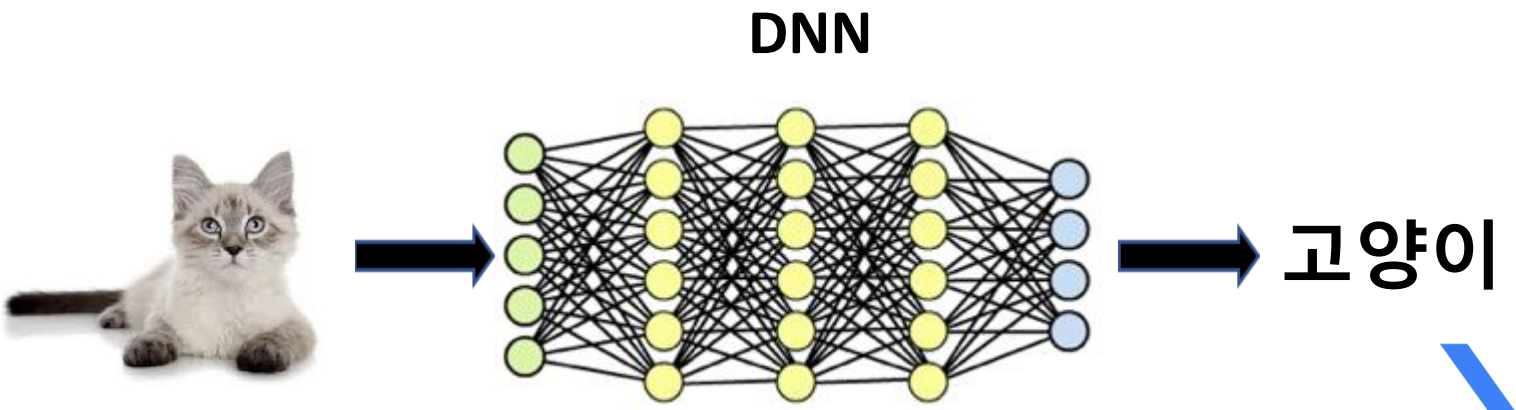
함수를 근사화 한 것이다

- 고양이, 강아지 구분 함수
- 어떻게 구분하는 지 정의하지 않았다.
- 정의하기는 힘들어도, 그 함수는 존재한다.
- 단순한 입출력 쌍 데이터로 그 함수를 근사화 하였다.

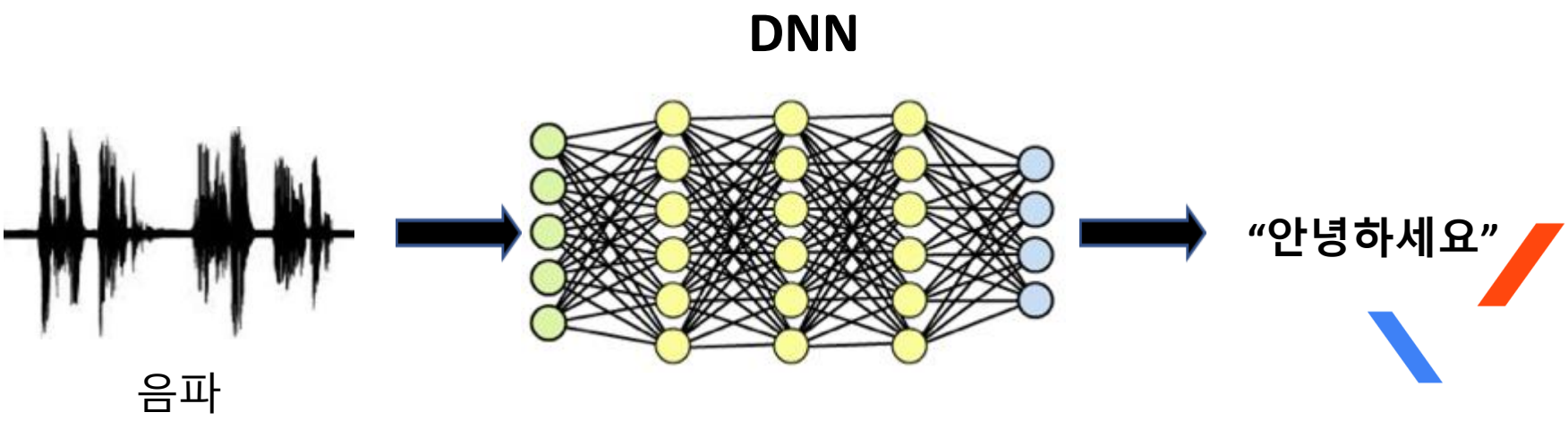
단순, 하지만 강력한 방법

- 단순히 입출력 쌍을 반복하여 학습시킨다.
- 하지만 로직을 찾아내기 어려운 문제에는 아주 효과적이다.
- 얼굴 인식, 물체 인식 같은.

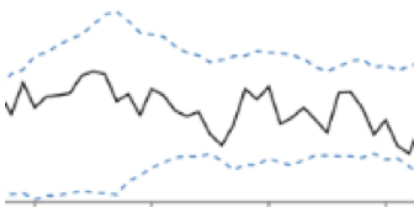
함수 예



함수 예

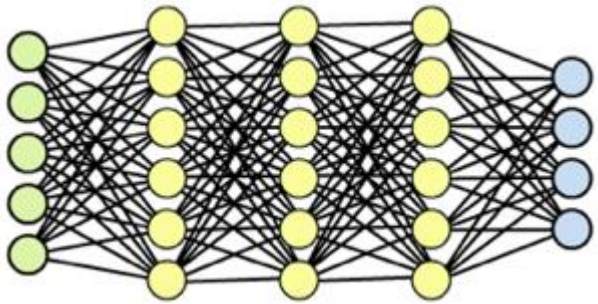


함수 예



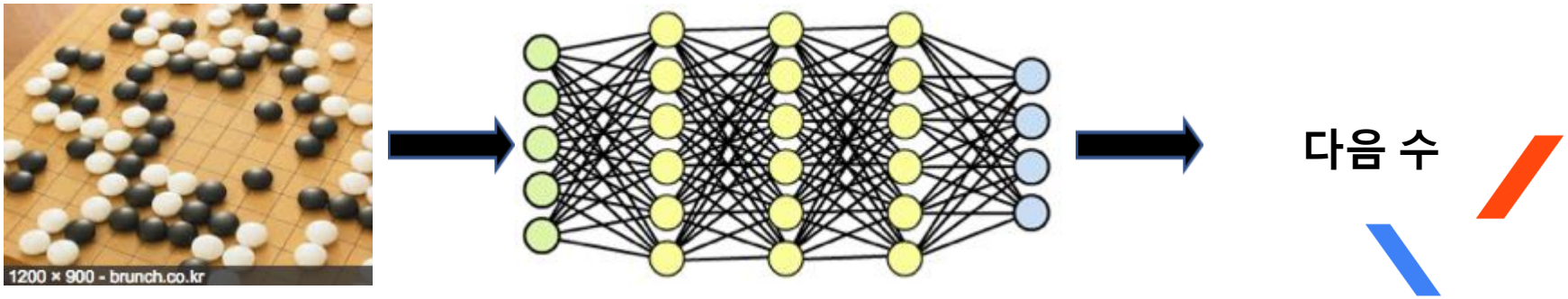
시스템
데이터

DNN

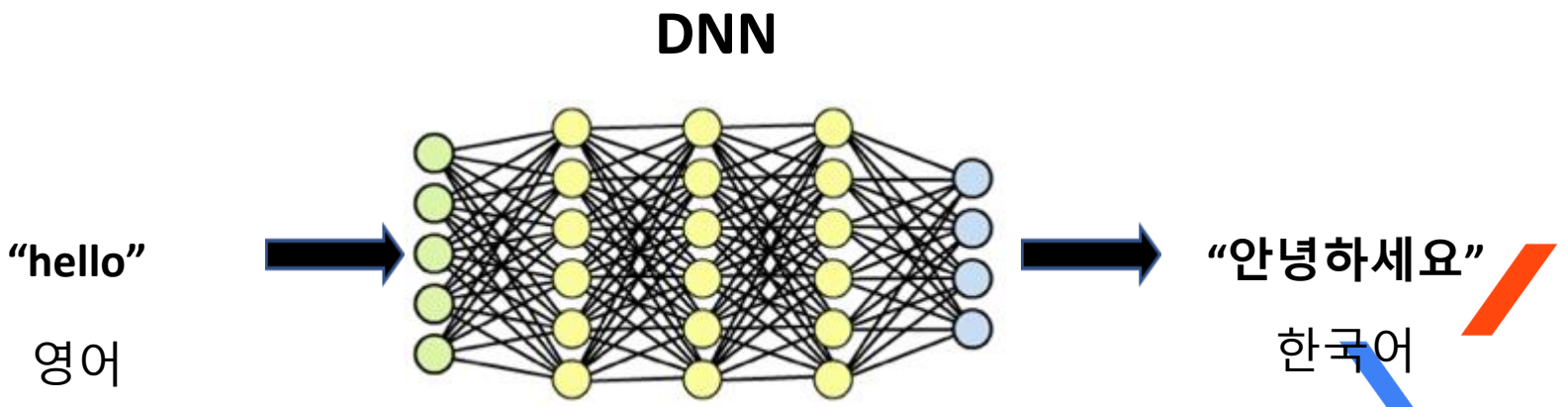


정상/비정상

함수 예



함수 예



딥러닝

- DNN은 함수 근사화 능력이 있다.
- 입출력 쌍을 계속 제공하여 DNN내부의 웨이트를 업데이트 한다.
- 이를 위해 내부적으로 BP와 GD 알고리즘을 사용.
- 충분한 입출력 데이터와 컴퓨팅 파워가 필요.

- 이를 반복하여 함수를 근사화하는 것을 DNN의 학습, Deep Learning이라 한다.

BP(Back Propagation), GD(Gradient Descent)

인공지능, 머신러닝, 딥러닝

인공지능(AI; Artificial Intelligence)

- 사람의 손이 아닌 기계가 알아서 하면 인공지능이다.

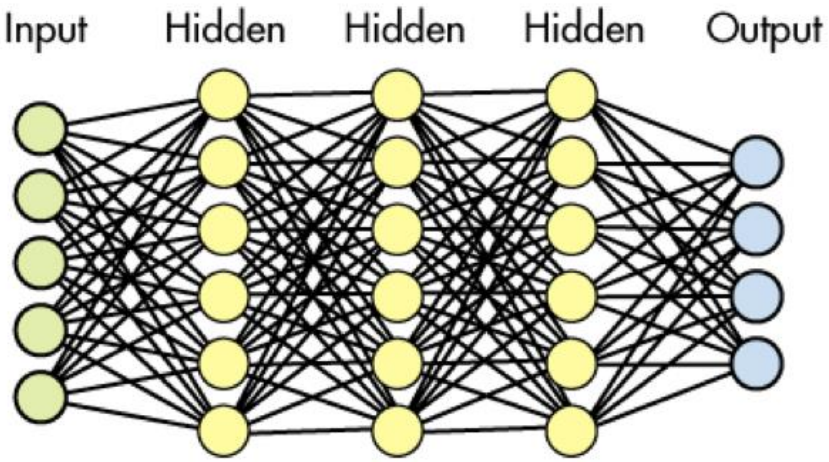
- 전문가의 지식을 하드코딩 할 수도 있고(전문가 시스템)
- 데이터에서 로직을 찾을 수도 있다.(머신러닝)

머신러닝(ML; Machine Learning)

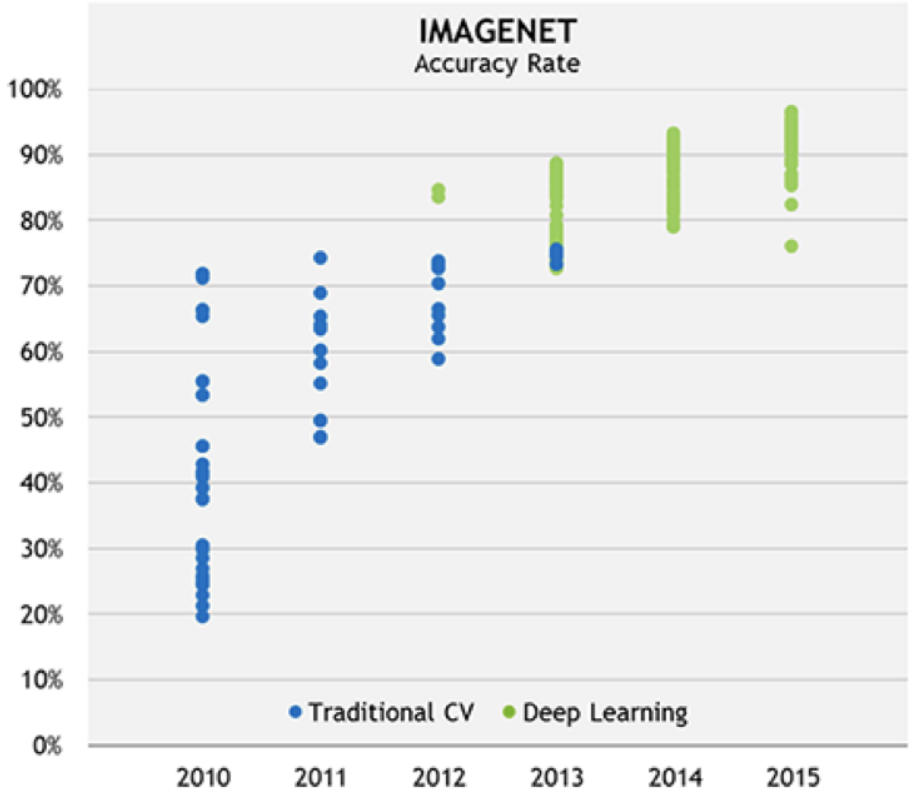
- 인공지능의 한 분야
- 데이터에서 가치를 찾아내는 것
- 아주 다양한 방법이 있다.
  - SVM, 의사결정트리, Random Forest, Bayesian, K-Means Clustering, K-NN, Neural Network

딥러닝(DL; Deep Learning)

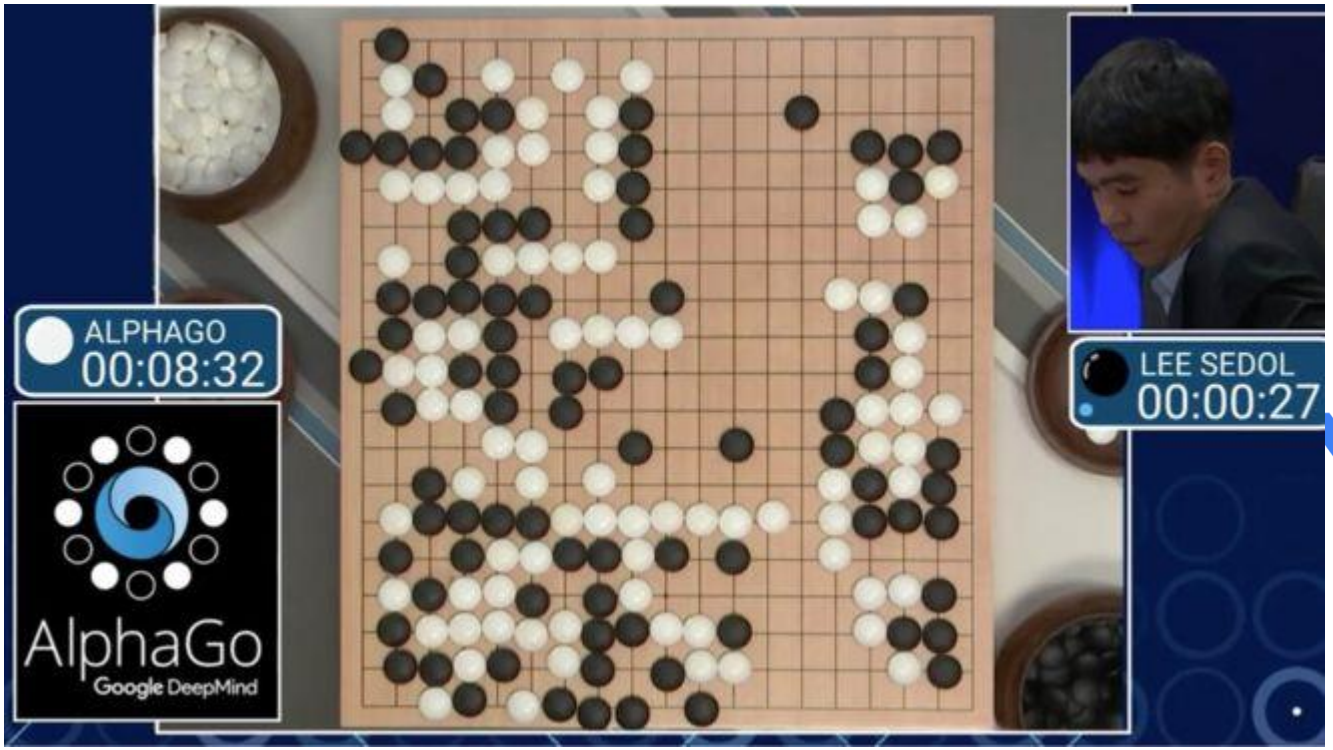
- 신경망(NN; Neural Network)을 사용한 머신러닝 방법
- 신경망의 은닉층이 많아서(deep) DNN(Deep NN)이라 부른다



2012년 AI 부활


IMAGENET
Accuracy Rate
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Traditional CV
Deep Learning
2010
2011
2012
2013
2014
2015

2016년 화려한 복귀



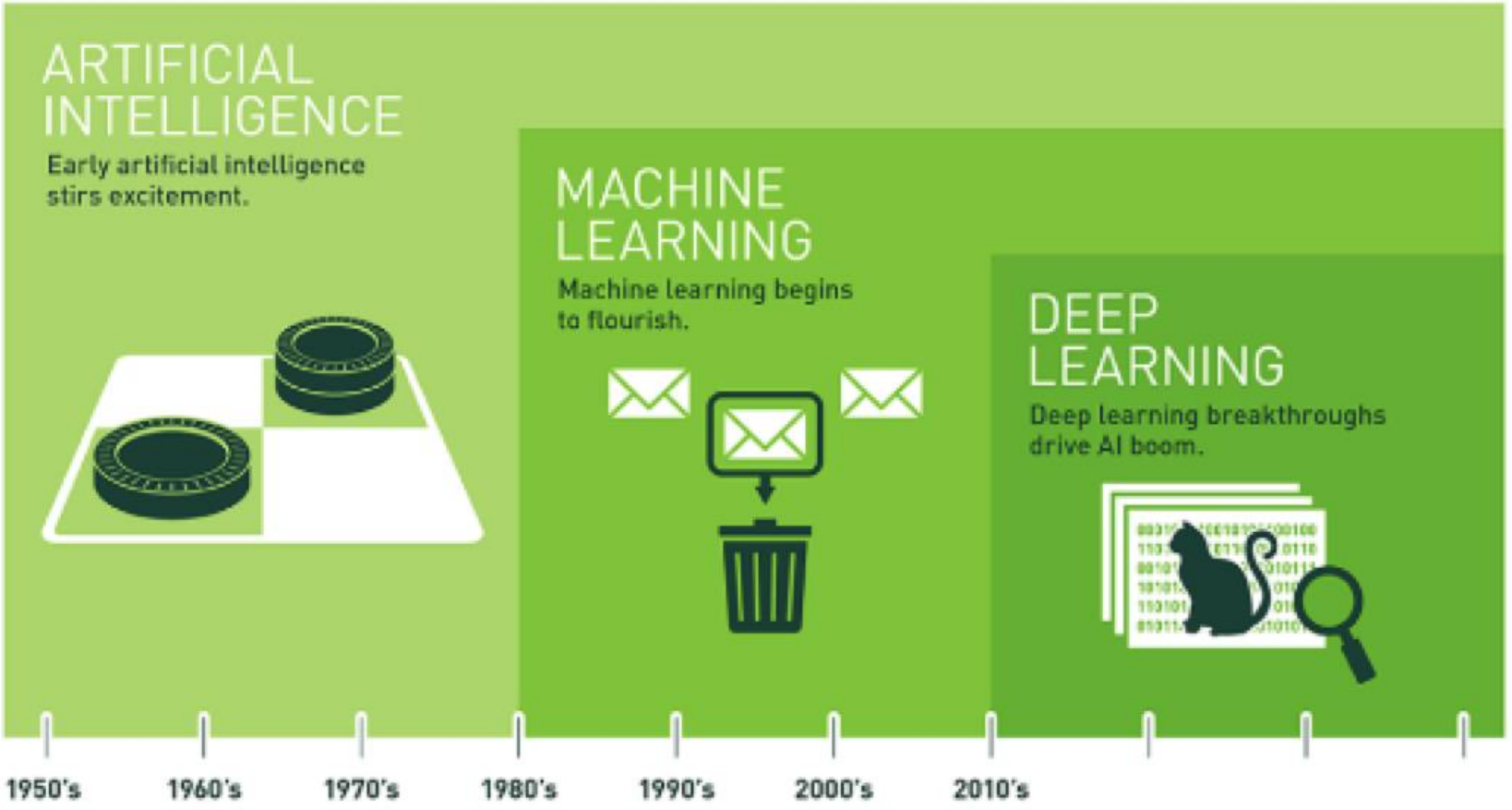
심층신경망과 신경망

- DNN과 NN은 별 차이 없다.
- NN: AI에 대한 엄청난 기대와 그 만큼의 실망.
- 최근 뚜렷한 성과를 보이면서 다시 큰 관심.
- DNN: 실망했던 용어를 대신하여 붐을 일으키기 위한 용어.

AI, ML, DL

- 인간이 고안한 알고리즘이건 기계가 학습한 알고리즘 이건, 기계가 스스로 처리하면 AI
- 기계가 학습하는 경우가 ML
- 그중 신경망을 사용하는 것이 DL
- 서로 다르지만, 그냥 AI = ML = DL 이라 부른다.

AI, ML, DL


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

딥러닝 조금 더

딥러닝 장단점

- 장점 : 대상 함수의 내부를 몰라도 된다.

- 단점 : 비싸다
  - 많은 데이터, 많은 연산량

데이터만 있으면 되지만, 데이터를 모으는 것이 어려운 일.

ML과 DL의 선택

- 기존의 방법으로 이미 풀린 문제는 ML
- 기존의 방법으로 못풀었는데 데이터가 있으면 DL
  - 바둑, 얼굴인식, 물체인식, 음성인식, 번역

딥러닝 실제 코드

```
model = build_model() // 모델 구성

for i in range(1000): // 모든 데이터에 대하여 1000번 반복
    model.fit(datas)

model.save('my_model.h5') // 저장. 파일 크기는 보통 100메가 단위

model = load('my_model.h5') // 저장된 모델 로딩
predicted = model.predict(input) // 예측 실행
```

딥러닝 트렌드

- 2대 적용 분야 : 자연어, 영상

- 일부 작업은 이미 안정화 단계에 있다.
  - 영상분류, 영상인식
  - Keras의 배포본에 포함

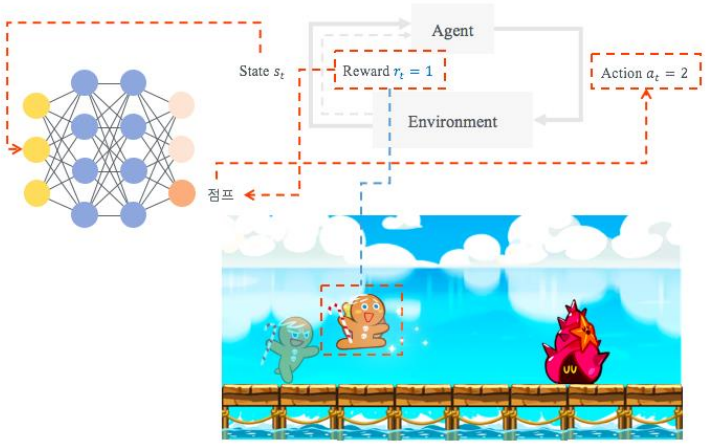
```
model = keras.vgg16.VGG16()
predicted = model.predict(image_data)
```

딥러닝의 큰 단점

- 입출력 데이터 쌍을 구하기 어렵다.
- 특히 출력 데이터. 레이블링 데이터(labeling data)

비지도 학습, 강화 학습

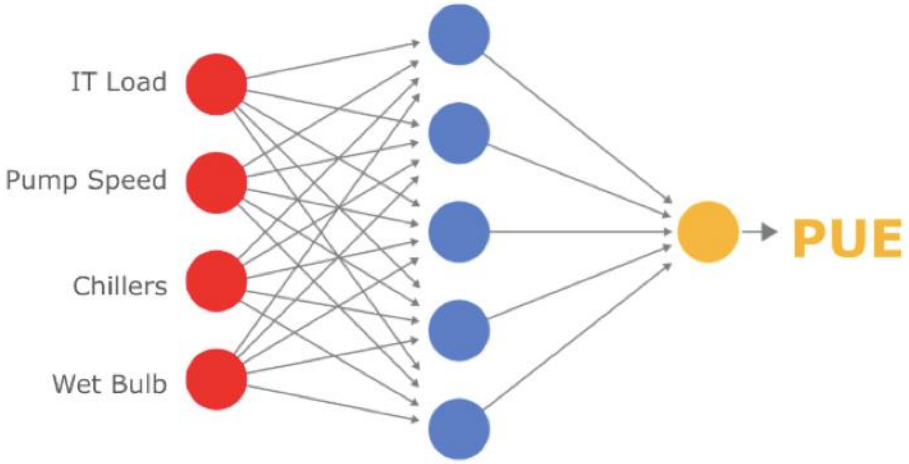
- 레이블링 데이터 문제를 해결하기 위한.
- 비지도 학습 : 모델 구조를 통해 레이블링 데이터 없이. GAN
- 강화 학습 : 환경과 동적으로 연동하여 레이블링 데이터를 취득.



예 – 구글 데이터 센터

DNN 적용

- 제어값을 입력으로 하고, PUE*를 출력으로 하여 학습
- 실제 데이터 사용

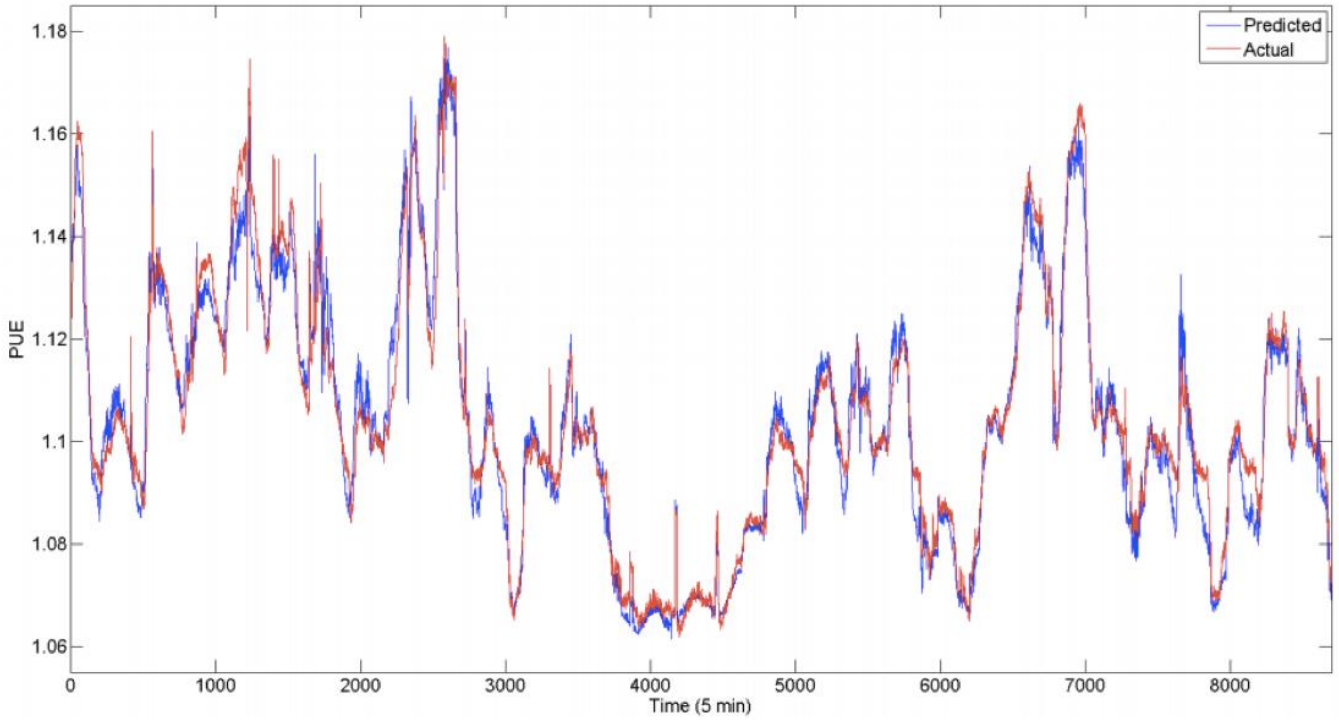


*PUE(Power Usage Effectiveness) : 에너지 사용 효율도

입력

1. Total server IT load [kW]
2. Total Campus Core Network Room (CCNR) IT load [kW]
3. Total number of process water pumps (PWP) running
4. Mean PWP variable frequency drive (VFD) speed [%]
5. Total number of condenser water pumps (CWP) running
6. Mean CWP variable frequency drive (VFD) speed [%]
7. Total number of cooling towers running
8. Mean cooling tower leaving water temperature (LWT) setpoint [F]
9. Total number of chillers running
10. Total number of drycoolers running
11. Total number of chilled water injection pumps running
12. Mean chilled water injection pump setpoint temperature [F]
13. Mean heat exchanger approach temperature [F]
14. Outside air wet bulb (WB) temperature [F]
15. Outside air dry bulb (DB) temperature [F]
16. Outside air enthalpy [kJ/kg]
17. Outside air relative humidity (RH) [%]
18. Outdoor wind speed [mph]
19. Outdoor wind direction [deg]

예측 결과


Predicted
Actual
PUE
1.18
1.16
1.14
1.12
1.1
1.08
1.06
0
1000
2000
3000
4000
5000
6000
7000
8000
Time (5 min)

학습된 DNN

- 제어값과 PUE의 관계를 학습
- 임의의 제어값에 대한 PUE를 알 수 있다.
- 학습된 DNN은 시뮬레이터로 사용할 수 있다.

시뮬레이션이 가능하면

- 임의의 제어값에 대한 PUE를 미리 알 수 있다.
- 다양한 입력에 대한 뮬레이션으로, 최선의 PUE를 찾을 수 있다.

딥러닝 기술 용어들

Cost Function 종류

- **MSE**(Mean Squared Error)
- **CE**(Cross Entropy)
- **KL-Divergence**
- MLE(Maximum Likelihood Estimation)

Optimizer 종류

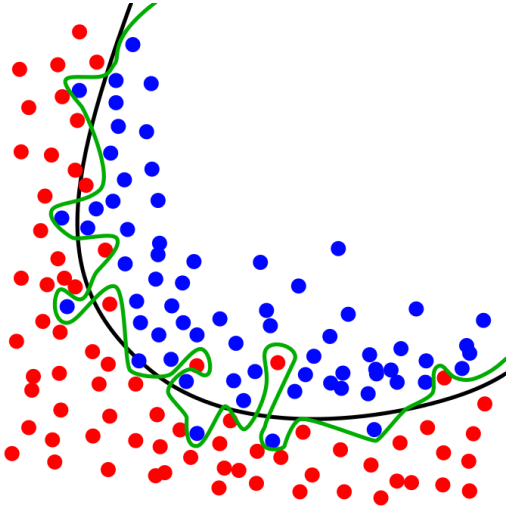
- 오차에 대하여 w를 업데이트 시키는 알고리즘들.

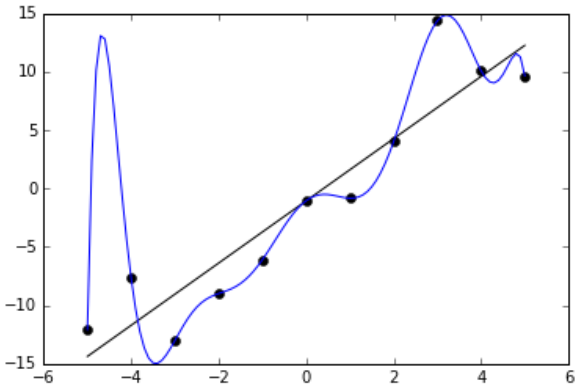
- GD(Gradient Descent)
- Batch GD
- **Mini-Batch GD**
- **SGD**(Stochastic GD)
- Momentum
- AdaGrad
- AdaDelta
- Adam
- RMSprop

Overfitting 방지법 종류

- DropOut
- BN(Batch Normalization)
- Regularization

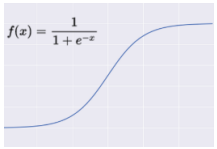
- Data Augmentati

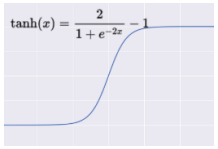


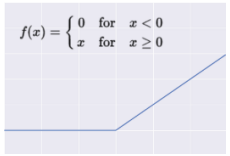


https://en.wikipedia.org/wiki/Overfitting

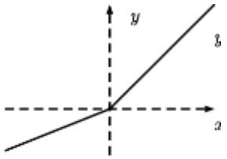
활성화 함수 종류

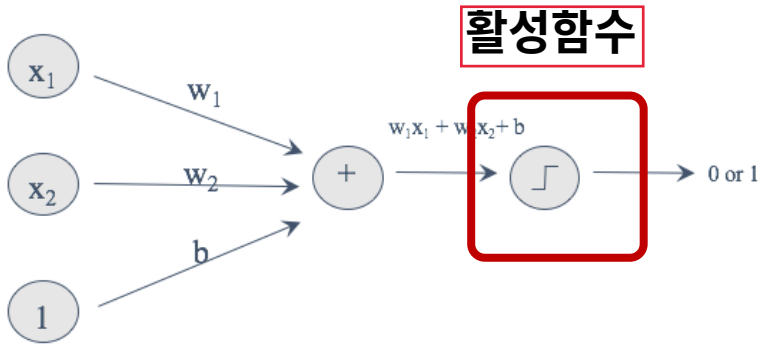
- sigmoid (= logistics) 

- Tanh 

- ReLU 

- Leaky ReLU

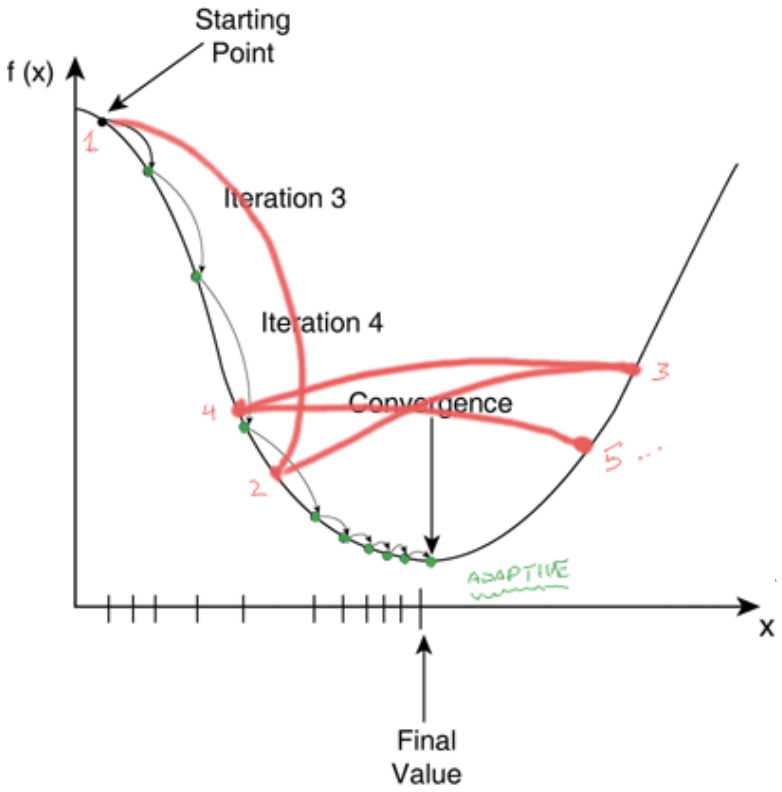




https://stats.stackexchange.com/questions/115258/comprehensive-list-of-activation-functions-in-neural-networks-with-pros-cons

학습율

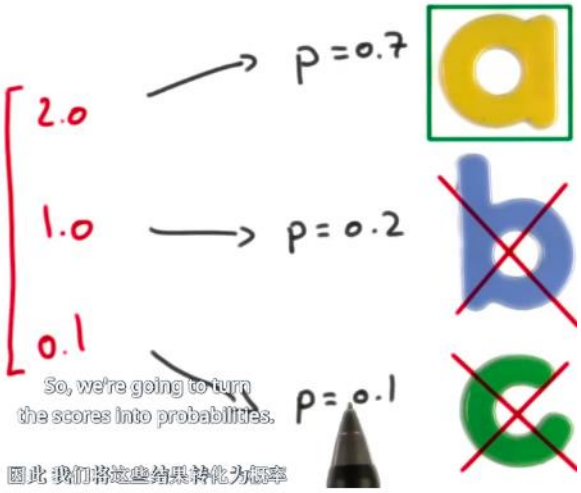
- 가중치가 변경되는 정도



https://www.quora.com/In-an-artificial-neural-network-algorithm-what-happens-if-my-learning-rate-is-wrong-too-high-or-too-low

SoftMax

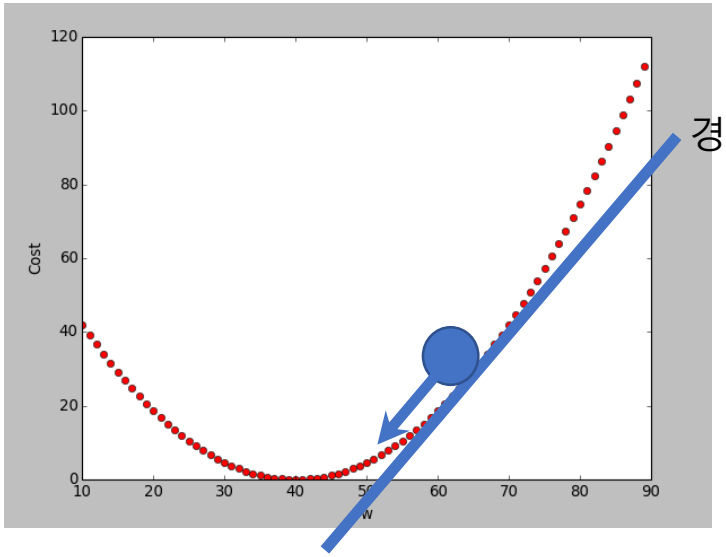
- activation function중의 하나.
- 최종 출력층에 사용되며, 여러개의 출력 노드의 합 중에 비중의 값으로 나타낸다.
- 확률 처럼 표현된다.



http://www.jianshu.com/p/cad5567aeb58

Gradient Descent

- 함수가 학습될 바를 정의한 비용함수의 값이 최소로 하기 위해 가중치를 업데이트 하기 위한 알고리즘



BackPropagation

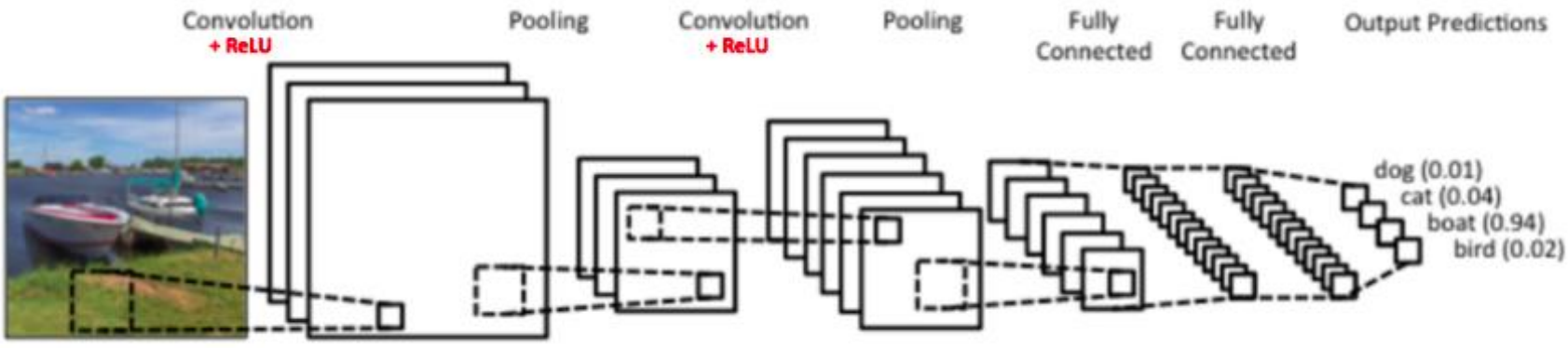
- 출력된 값과 원하는 값과의 차이를 가지고 그 전의 w 값들을 변경하는 알고리즘.
- 뒤에서부터 그 오차의 값이 전파된다는 이름.
- 실제 변경되는 값의 크기는 GD로 결정됨.

```
/*
에러를 뒤로 전파한다는 의미를 가짐.
코딩에서 접할 일은 거의 없음.
*/
```

CNN

CNN(Convolutional NN) 구조

- Convolution filter와 pooling을 반복.
- 이후 일반 DNN에 적용.

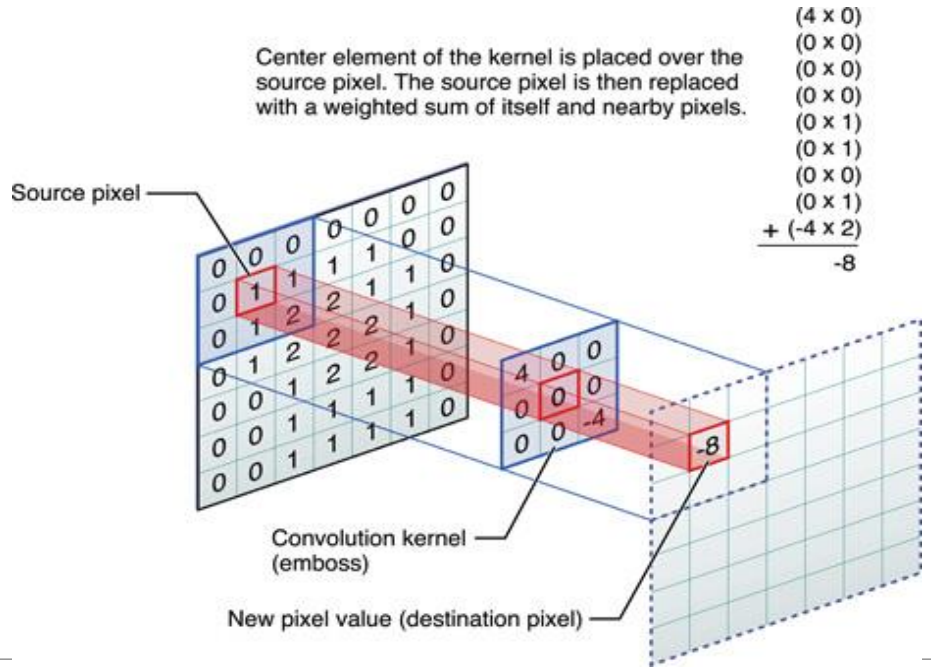


https://sites.google.com/site/nttrungmtwiki/home/it/deep-learning/convolution-neural-network/understanding-convolutional-neural-networks-for-nlp

Convolution Filter

사람눈에는 특정 모양에 반응하는 신경세포들이 있다.

이를 구현한 가장 간단한 방법이 convolution filter이다.



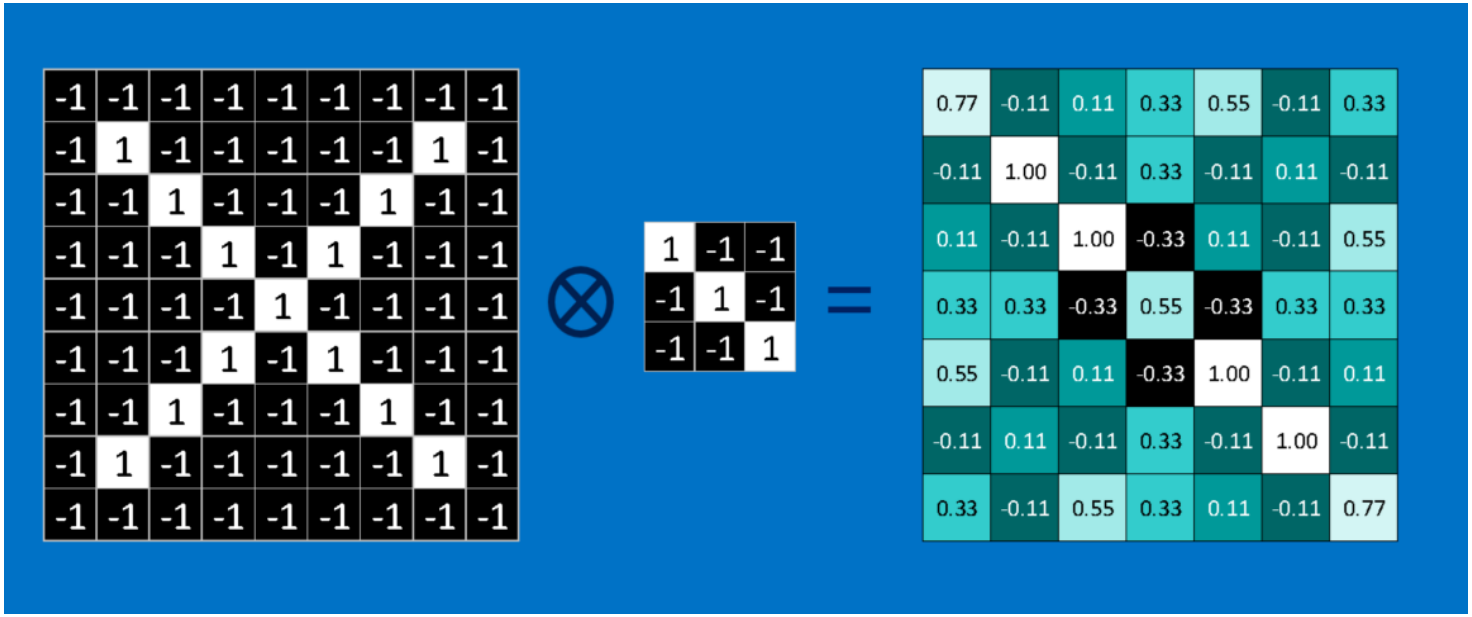
https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

단순하게 왼쪽으로 가는 직선 하나.

Convolution Filter – 의미

필터 모양의 자극이 있으면 그 결과가 최대가 된다.

필터 모양이 있는 지 찾아낸다.

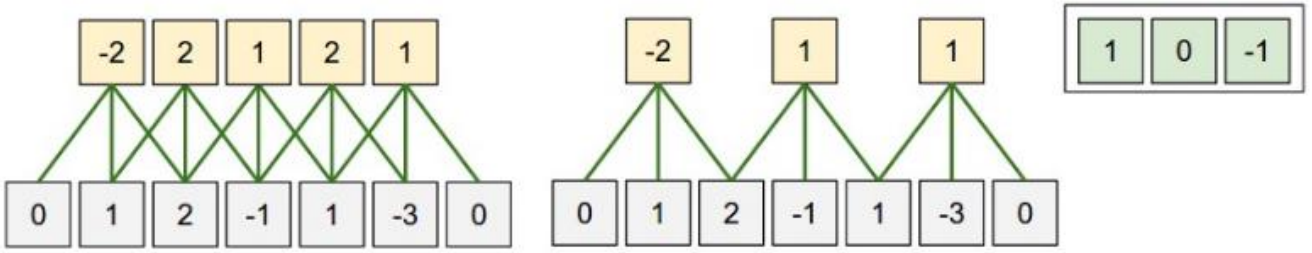


http://www.kdnuggets.com/2016/08/brohrer-convolutional-neural-networks-explanation.html

Convolution Filter – Stride

필터 적용 시의 이동 칸 수.

보통은 1.

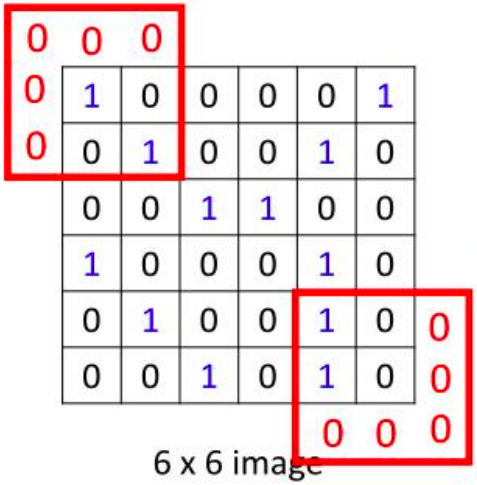


stride 1 stride 2

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

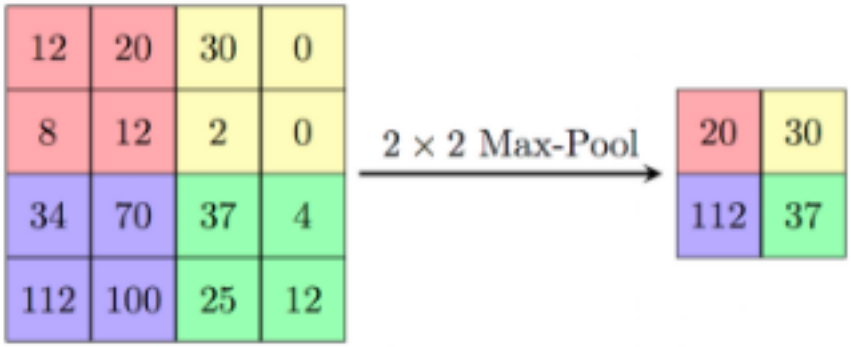
Convolution Filter – Padding

필터 특성상 이미지가 작아지는 것을 방지하기 위해 원 이미지를 키우는 것.



http://blog.csdn.net/tianzhaixing2013/article/details/64922396

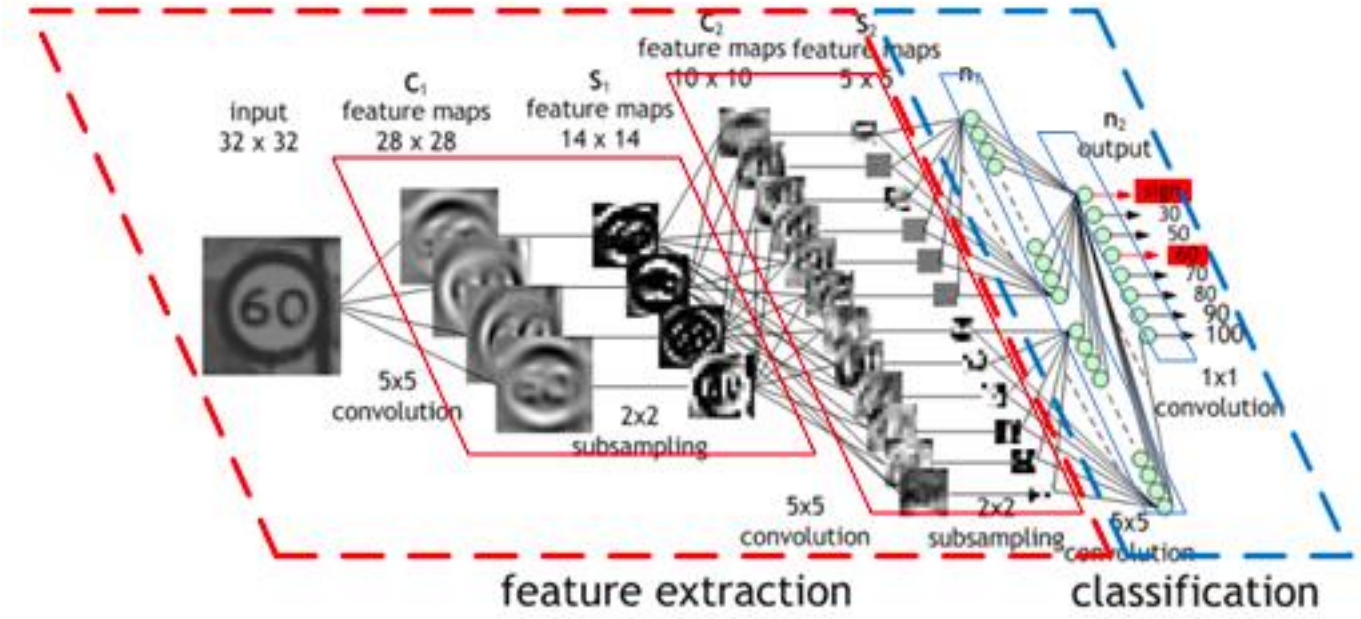
Max Pooling



https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

특징 추출과 분류

- convolution 필터와 pooling이 반복하여 feature 추출.
- 이후 일반 DNN로 분류.



http://parse.ele.tue.nl/education/cluster2

CNN 성능

- 이미지 처리에 뛰어남.
- 이미지 외의 것에도 좋은 성능을 보임.

영상 데이터

영상과 픽셀

- 영상은 픽셀로 구성됨

https://bwessay.tistory.com/90

픽셀의 값

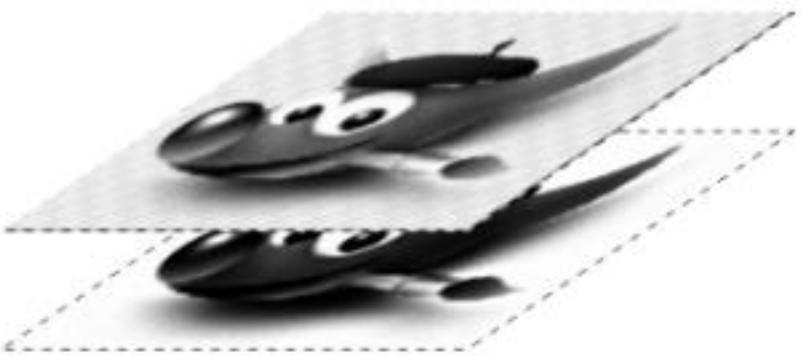
- 회색 영상의 경우 1개의 픽셀은 밝기의 값 1개로 나타냄.

187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

https://thebook.io/006796/ch01/02/01_01/

픽셀의 값

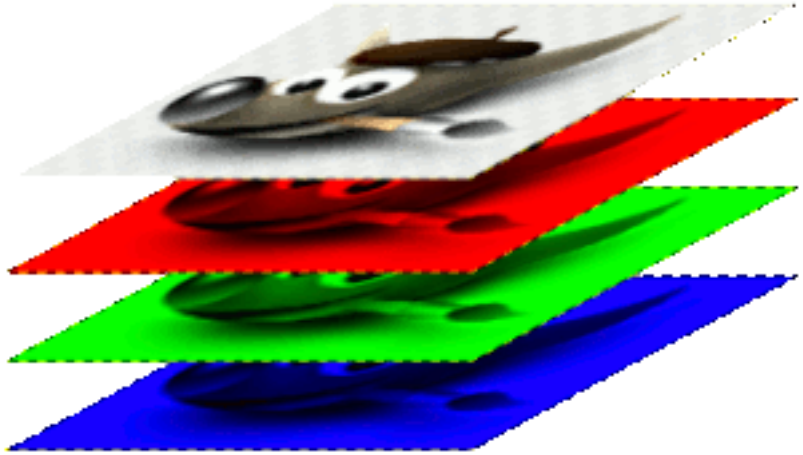
- 회색 영상의 경우 1개의 픽셀은 밝기의 값 1개로 나타냄.



https://docs.gimp.org/2.10/ko/gimp-images-in.html

픽셀의 값

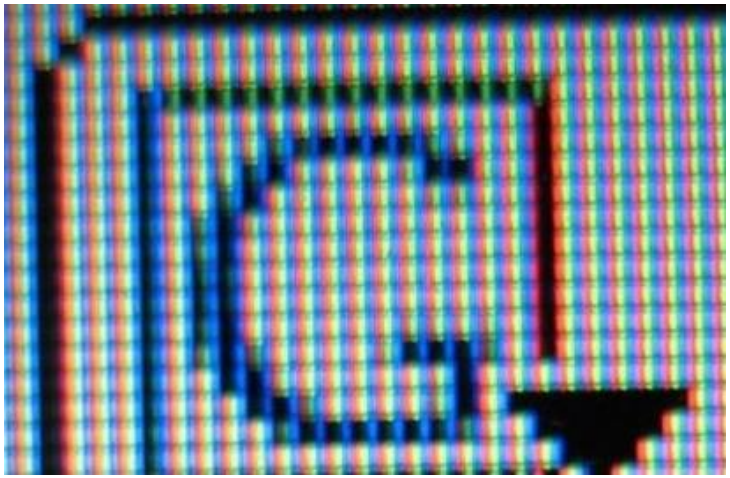
- 칼라 영상의 경우 1개의 픽셀은 3개의 색깔 Red, Green, Blue 을 표시한 3개의 값으로 나타냄.



https://docs.gimp.org/2.10/ko/gimp-images-in.html

영상 데이터

- 가로 세로로 배열된 픽셀의 값을 숫자로 표시한 데이터
- 단지 숫자들이다.
- 모니터에서는 픽셀의 값에 따라 밝기 혹은 색깔로 표시하여 보여준다.



https://www.howtogeek.com/130826/how-to-fix-a-stuck-pixel-on-an-lcd-monitor/

영상 파일 포멧

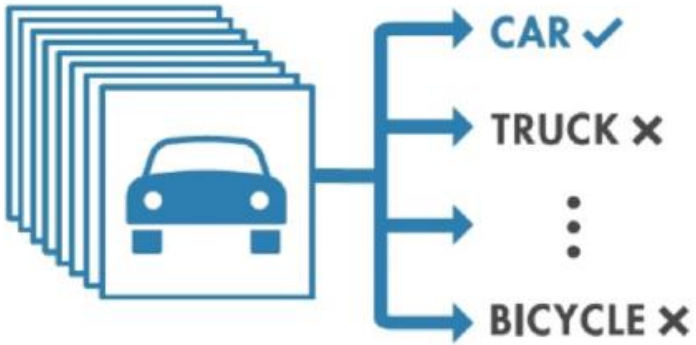
- bitmap은 pixel 값 그대로 저장
- gif, jpeg, png 등은 압축하여 저장한 것.

https://docs.gimp.org/2.10/ko/gimp-images-in.html

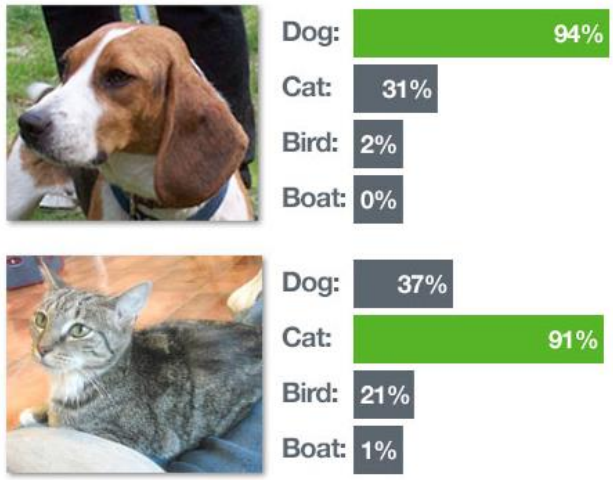
영상 분류

분류(Classification)

- 대상을 미리 정해진 클래스(class)로 분류하는 작업
- 영상의 경우 해당 영상이 무엇인지 인식하는 작업



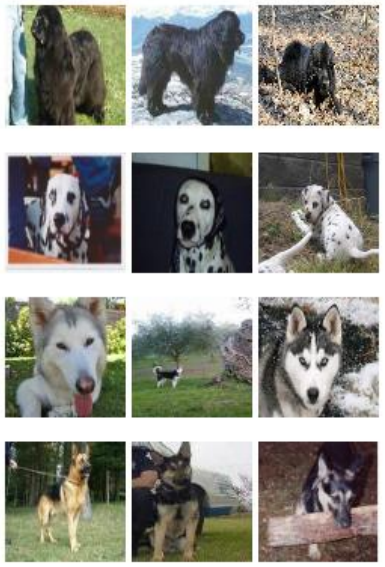
https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html



https://www.pyimagesearch.com/pyimagesearch-gurus/?src=post-deep-learning-libs

ImageNet

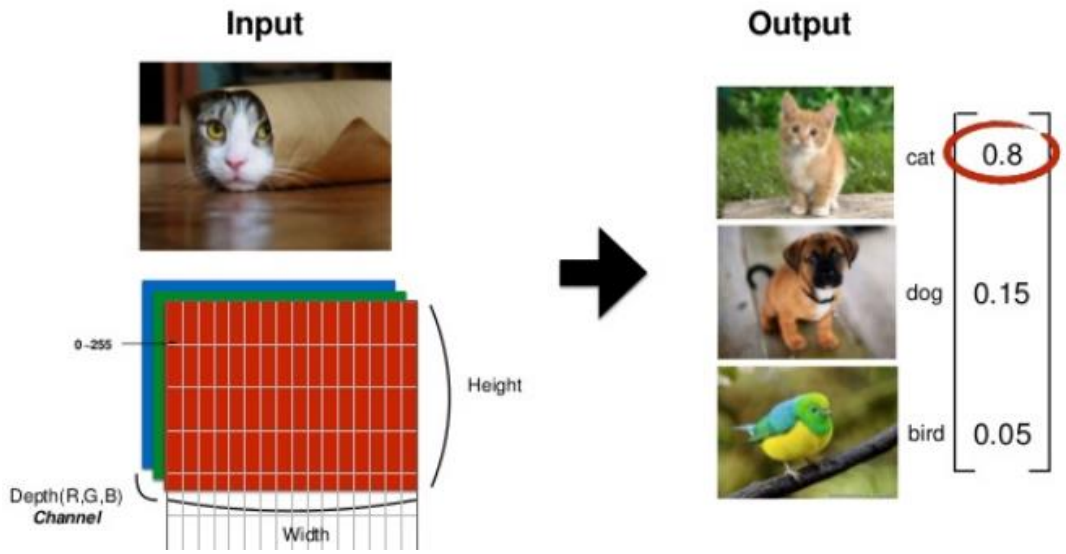
- 일반적인 사물의 영상의 데이터셋
- 1000개의 클래스



http://image-net.org

분류기 DNN의 입력

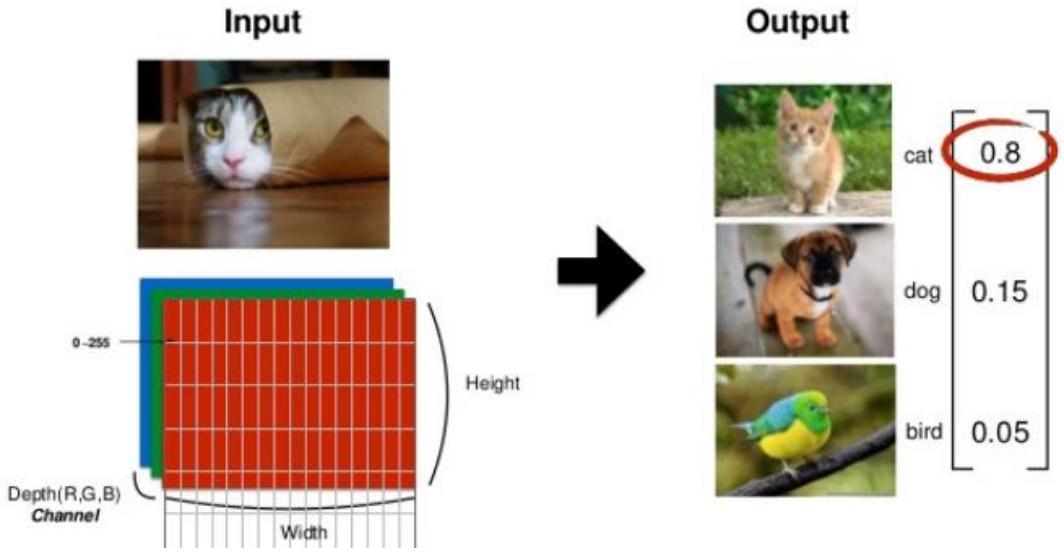
- 입력은 영상 데이터 자체.
- 입력 노드의 수는 영상 데이터의 값 갯수와 같다.



https://www.slideshare.net/leeseungeun/cnn-vgg-72164295

분류기 DNN의 출력

- 클래스의 갯수 만큼의 출력 노드
- 출력된 값은 0에서 1사이의 값을 가진다.
- 가장 큰 값을 갖는 노드에 해당하는 클래스로 분류된다.



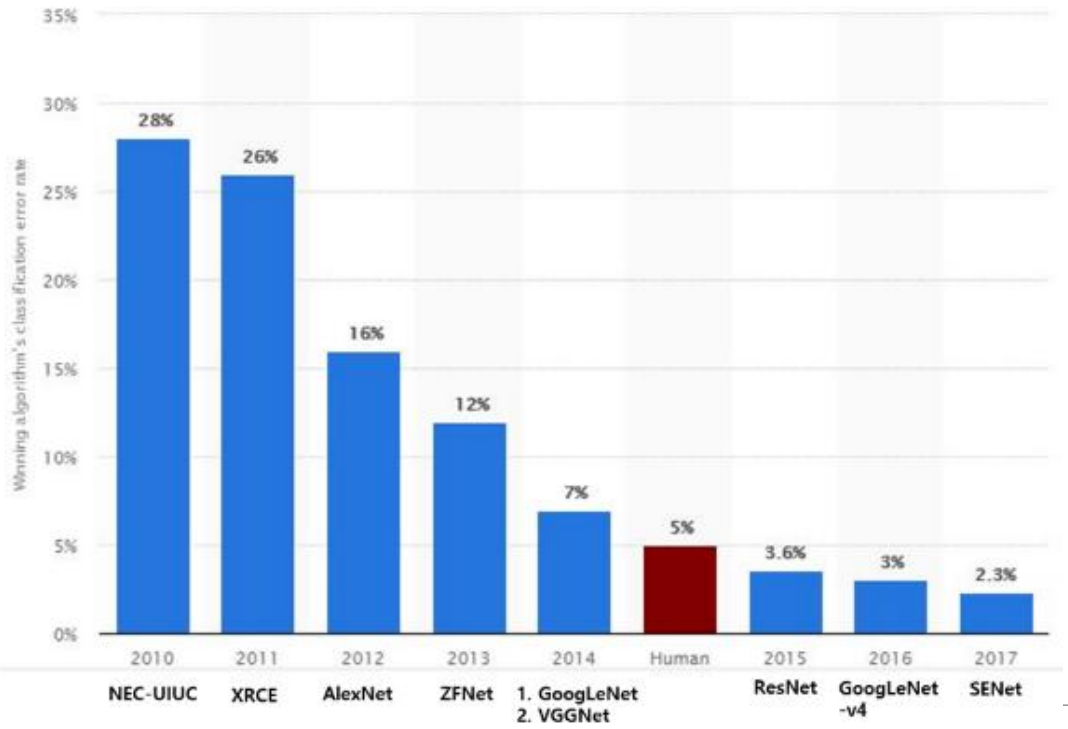
https://www.slideshare.net/leeseungeun/cnn-vgg-72164295

영상 분류 현황

- 완숙된 상태
- Keras를 설치하면 VGG나 ResNet이 default로 포함되어 있다.
- 그냥 가져다 사용하면 되는 수준

ILSVRC

- ImageNet 데이터셋을 가지고 겨루는 대회
- ImageNet Large Scale Visual Recognition Competition



공개된 모델

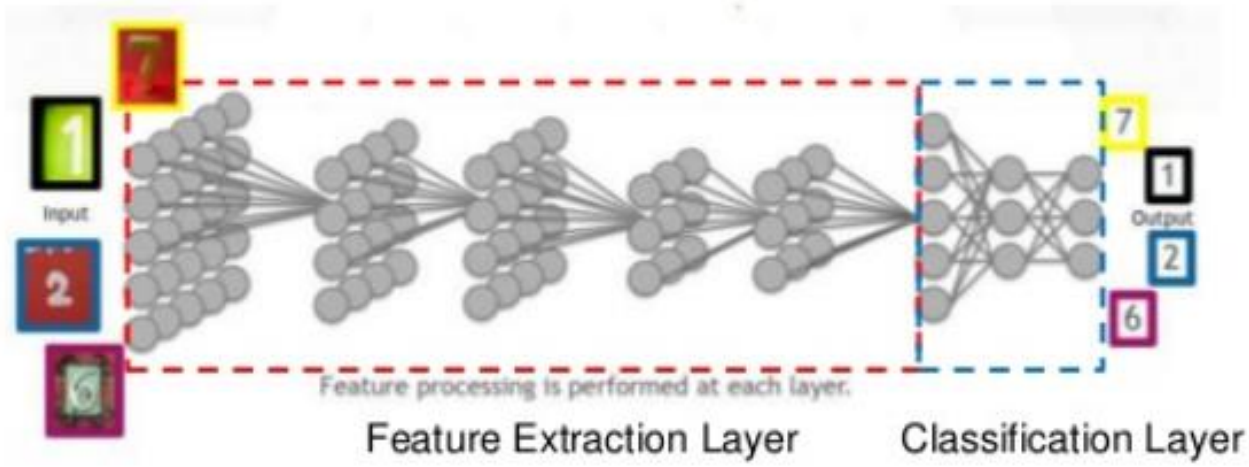
- ILSVRC에서 우승한 VGG, ResNet 등의 모델이 공개되어 있다.
- 구조 뿐 아니라 학습된 모델도 공개.

- 가져다 사용할 수 있다.

전이 학습

CNN의 구조

- CNN은 2개의 부분으로 구분된다.
  - feature extraction : conv layer + pooling layer
  - classification : fully connected layer



https://www.slideshare.net/ssuser06e0c5/convolutional-neural-networks-135496264

우승 모델

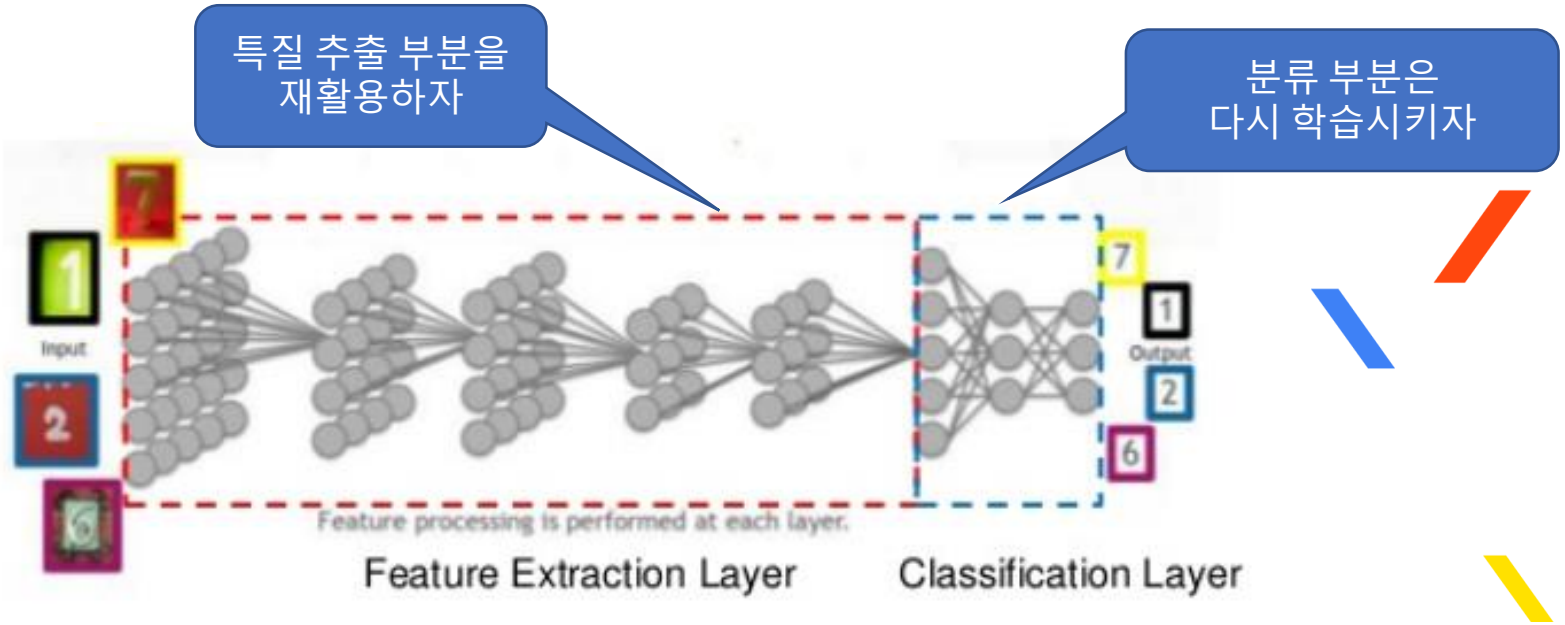
- 대부분은 CNN을 사용한다.
- 우승했던 만큼 CNN의 2개 기능이 좋다고 볼 수 있다.
- 특질 추출기능도 우수하고 분류 기능도 우수하고.

커스텀 데이타

- 보통 실무에서 사용하는 데이터는 ImageNet의 클래스와 다르다.
- 이런 이유로 DNN의 출력층의 노드 수가 다를 수 밖에 없다.
- 1000개의 클래스의 분류를 그대로 사용할 수 없다.

우승 모델 재활용

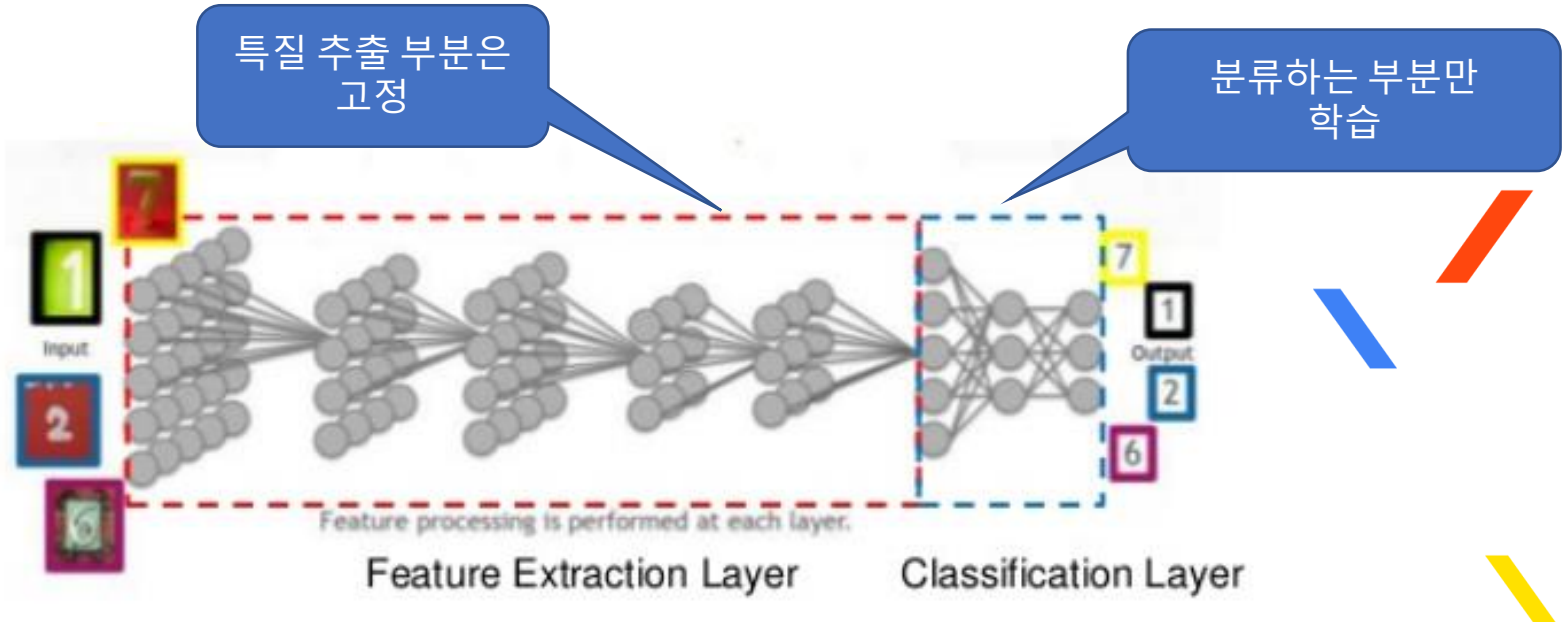
- 우승한 모델의 우수한 특질 추출 기능을 재활용 하자.
- 대신 분류 기능은 다시 학습시키자.



https://www.slideshare.net/ssuser06e0c5/convolutional-neural-networks-135496264

전이 학습

- Transfer Learning
- 학습된 모델을 가져와 다시 학습한다.

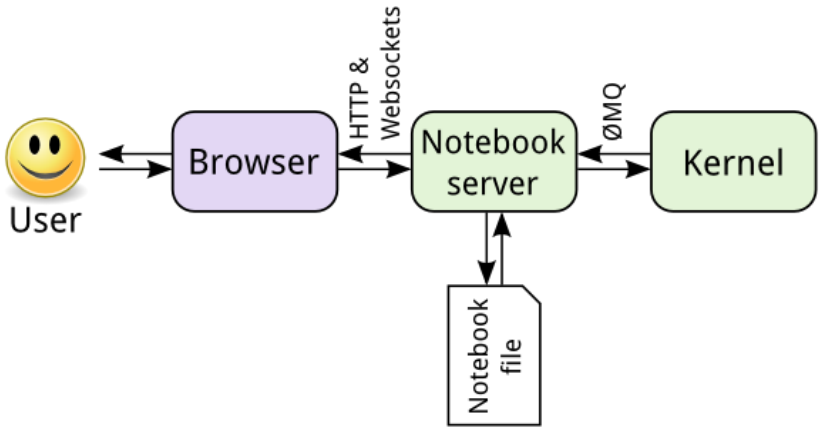


https://www.slideshare.net/ssuser06e0c5/convolutional-neural-networks-135496264

실습 환경

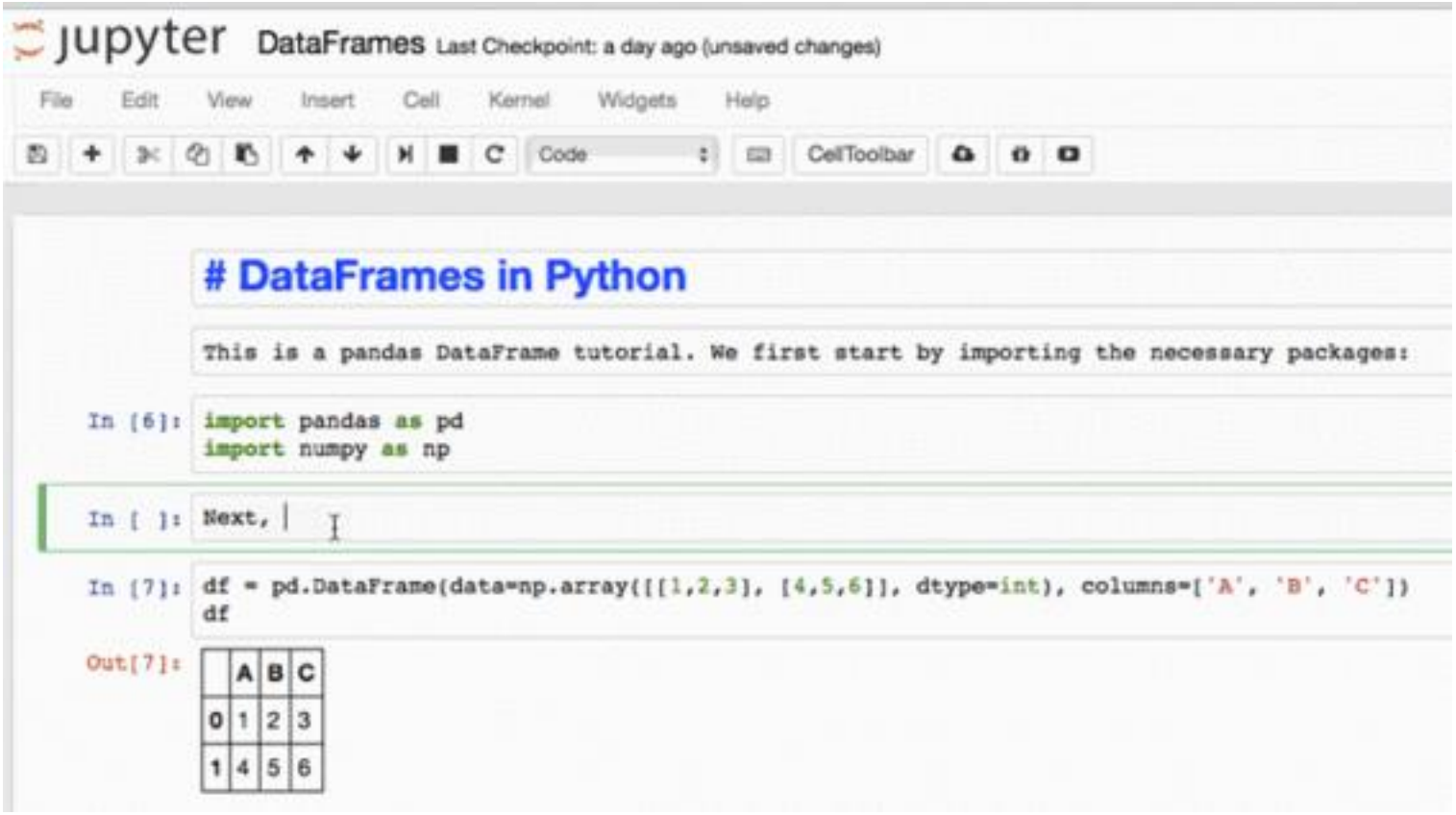
Jupyter Notebook

- 웹브라우저로 서버에서 python 실행
- 코드와 실행 결과, 글과 그림을 저장할 수 있다.



https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Jupyter Notebook



https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook

Google 클라우드의 Colab

- Google의 온라인 Jupyter Notebook 서비스

https://nholmber.github.io/2018/09/google-colab/

Google 계정 필요

- Google 계정이 생성되어 있다면, 별도의 설치나 작업 없이 Colab 사용 가능

https://nholmber.github.io/2018/09/google-colab/

Colab 첫 화면

- https://colab.research.google.com

실습 데이타

커스텀 데이타

- 치아 이미지
- 3개로 분류됨
  - 건강 치아, 충치 치아, 치료된 치아
  - 클래스가 3개.
  - 각 70개



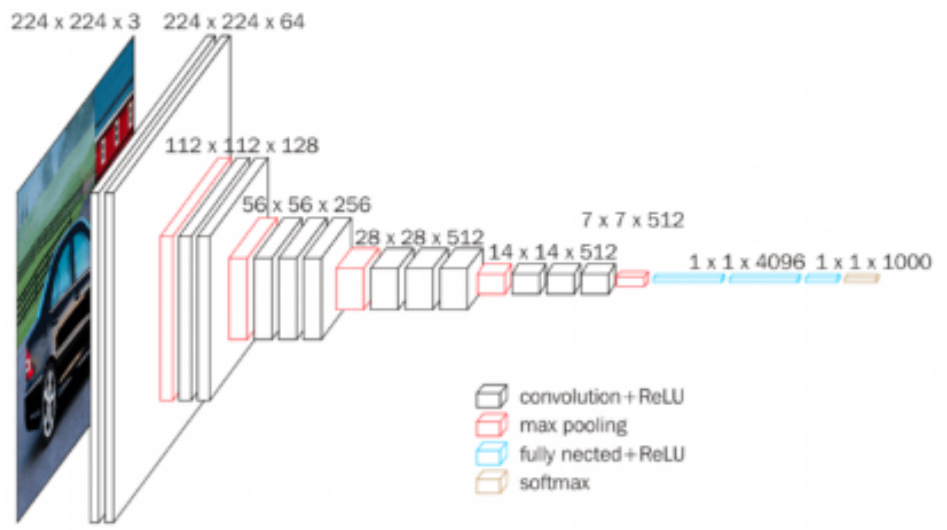
VGG로 분류 실행

VGG 16

- 2014년에 ILSVRC에서 2등한 모델.
- convolution layer와 max pooling layer가 있는 일반 CNN
- 21개의 layer를 갖는다. 학습 대상 weight는대략 1억 4천만개.
- 모델 파일 크기는 약 500M

- pooling 레이어를 제외하고 모두 16개의 레이어가 있다.

VGG16 구조

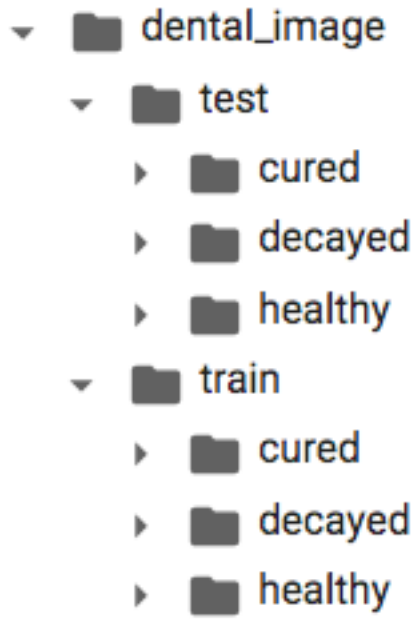

224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution + ReLU
max pooling
fully nected + ReLU
softmax


3x3 conv, 64
3x3 conv, 64
pool/2
3x3 conv, 128
3x3 conv, 128
pool/2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool/2
fc 4096
fc 4096
fc 4096
Size:224
Size:112
Size:56
Size:28
Size:14
Size:7

VGG로
커스텀 데이타 학습

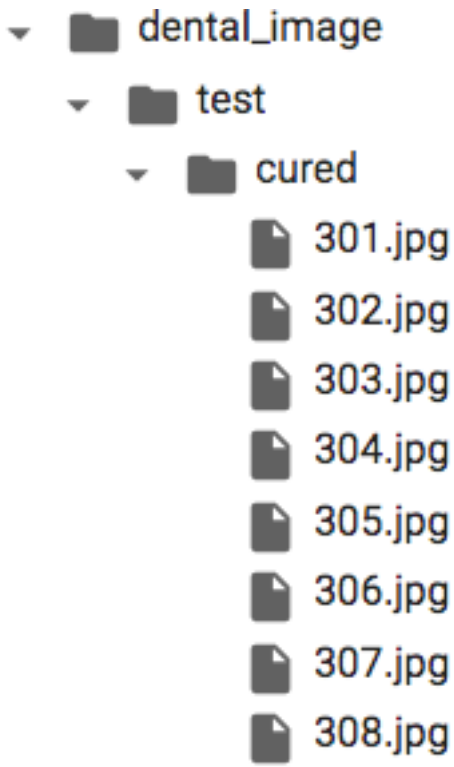
데이터 준비

- 디렉토리 구조가 레이블링을 대신함
- 데이터 홈 폴더 밑에 2개의 폴더. train, test
- train과 test 밑에 각각의 클래스 이름의 폴더



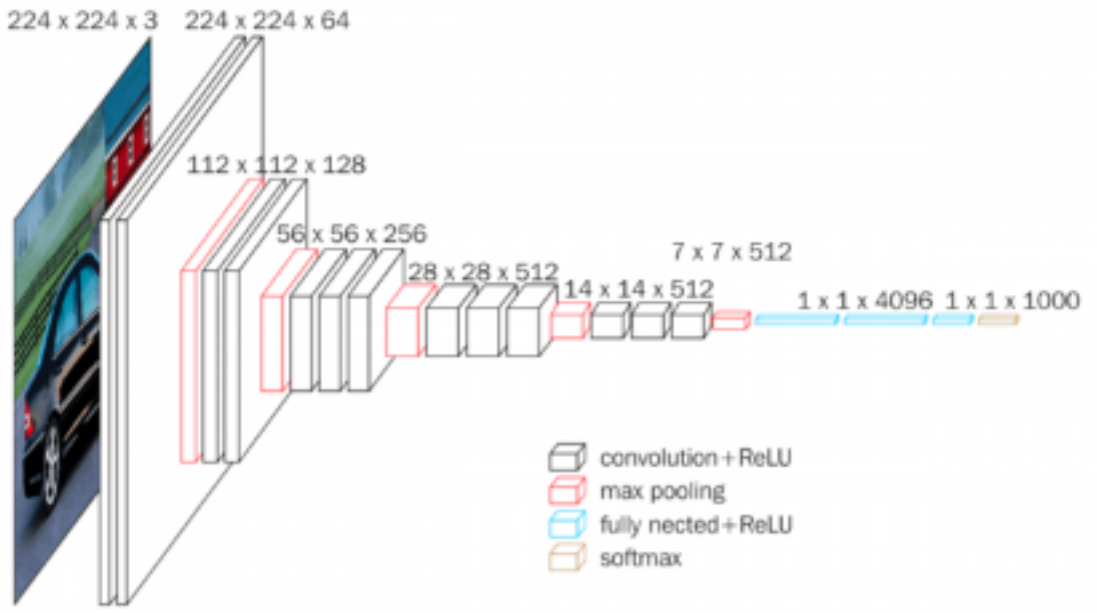
데이터 준비

- 각 클래스에 해당하는 폴더 안에 이미지 파일이 있다.



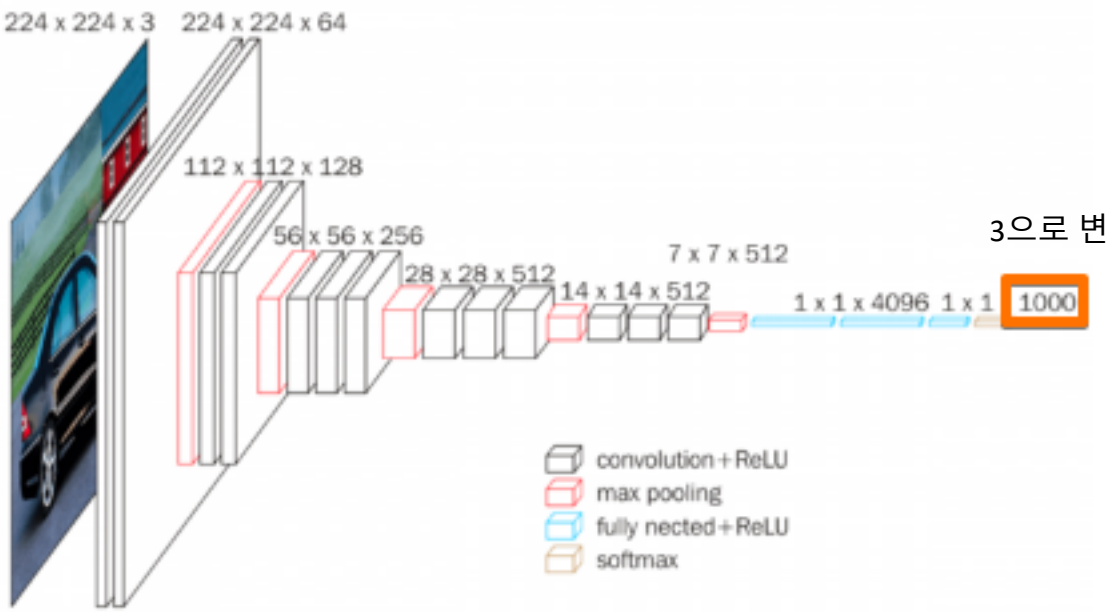
원 VGG 모습

- 입력 데이터는 224(가로)x224(세로)x3(RGB)
- 출력은 1000개(클래스 갯수)의 숫자



새로 학습 시의 VGG 모습

- 입력 데이터는 224(가로)x224(세로)x3(RGB)
- 출력은 3개(클래스 갯수)의 숫자



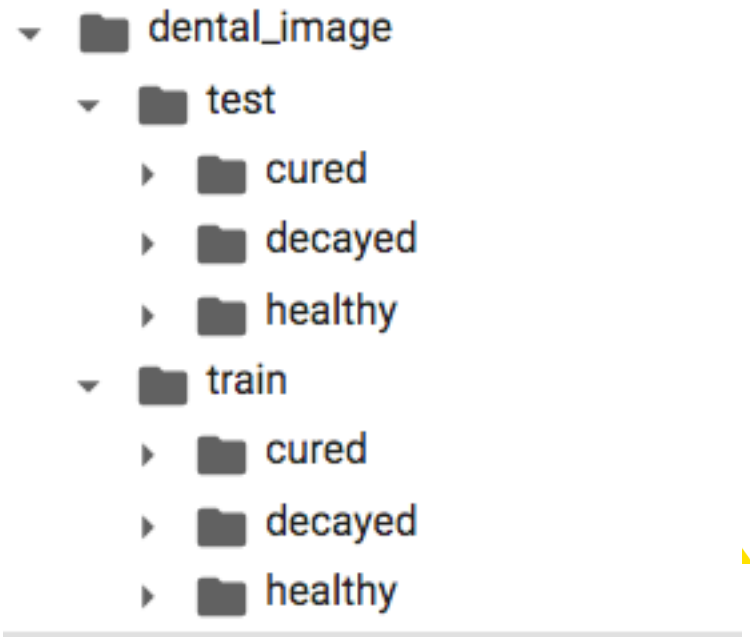
데이터 전처리

- VGG16의 입력은 224x224의 크기로 고정되어 있다.
- 이미지 파일을 로딩하고 이 사이즈로 크기 조절을 해야 한다.

레이블링 데이타

- 따로 준비하지 않는다.
- Keras의 ImageDataGenerator를 사용하면 폴더 이름을 레이블링으로 사용한다.

- train/cured 밑의 모든 파일은 'cured' 클래스에 속함
- train/decayed 밑의 모든 파일은 'decayed' 클래스에 속함



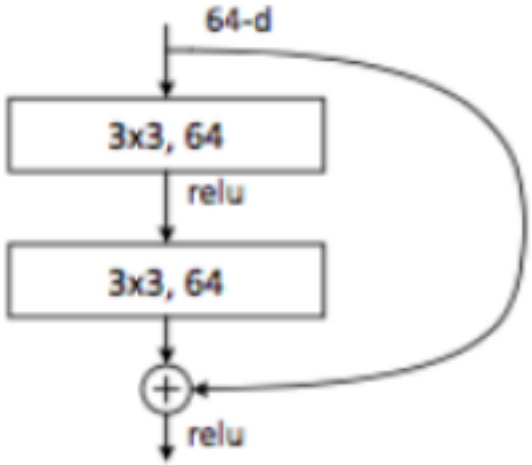
데이터 증강

- Keras의 ImageDataGenerator에서 기능 제공
- 데이터 부족을 극복하기 위해 위치 이동, 회전, 좌우반전등의 변화를 준다.

ResNet으로 분류 실행

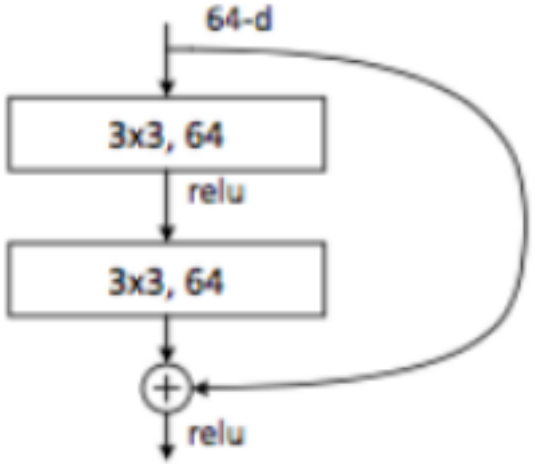
ResNet

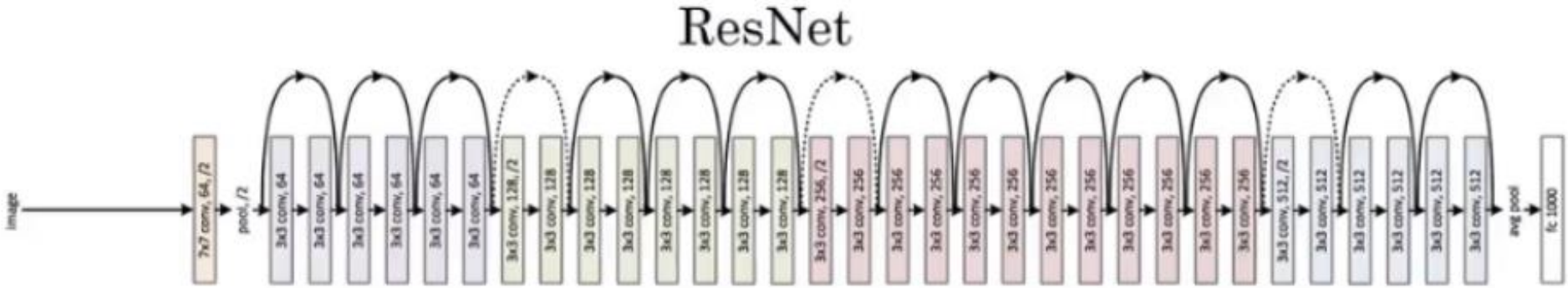
- 2015년에 ILSVRC에서 1등한 모델.
- convolution layer와 max pooling layer가 있는 CNN
- 많은 레이어가 있으며weight의 갯수는 약 2500만개
- 모델 파일의 크기는 약 100M



ResNet 구조

- 2개의 레이어 간에 링크가 있다.
- 입력과 출력은 VGG16과 동일

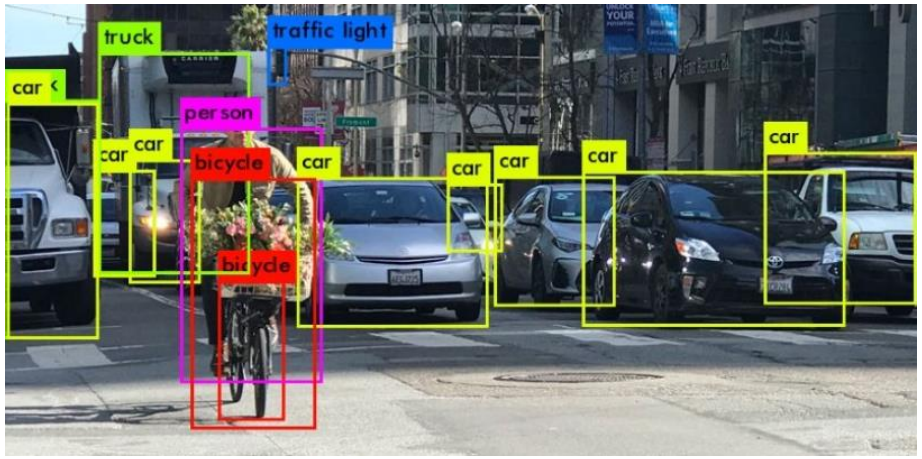




영상 인식

영상 인식(image recognition)

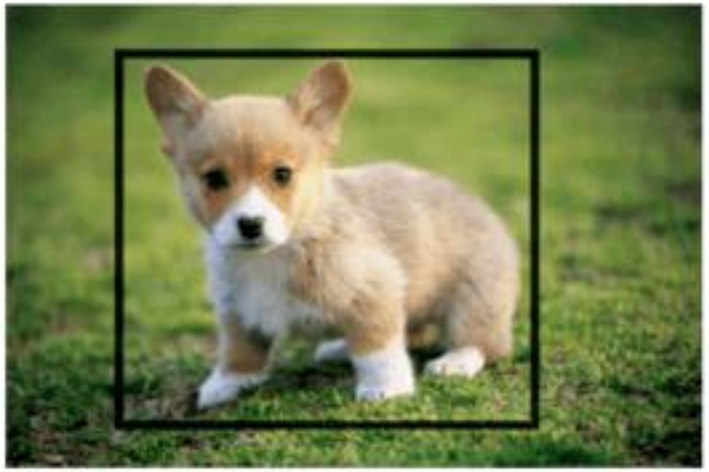
- 영상 내에서 대상을 탐지(detection)한다.
- 그리고 그 대상을 분류(classification)한다.
- 이를 영상 인식(recognition)이라 한다.



https://azati.com/image-detection-recognition-and-classification-with-machine-learning/

탐지(detection)와 분류(classification)

- 보통 탐지(detection)된 결과를 박스로 표시한다.
- 위치를 나타낸다고 해서 localization이라고도 부른다.
- 그리고 박스내의 영상을 분류(classification)한다.



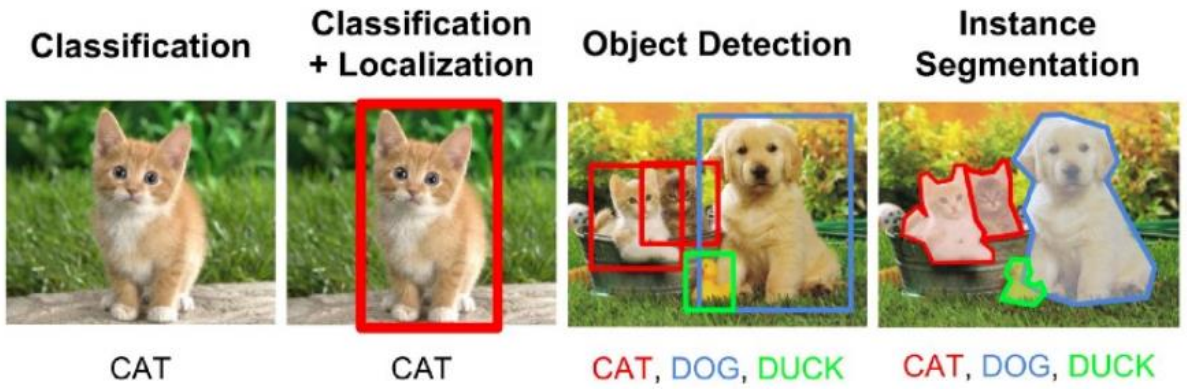
https://www.commonlounge.com/discussion/c9975025c9ff473c8f9ed2c4b1c3ea6a

영상 인식 현황

- 2개의 작업(탐지, 분류)이 동시에 필요.
- 이로 인해 속도가 느리거나 혹은 정확도가 떨어진다.
- 그리고 20개 정도의 소수 클래스.

영상인식 작업 요약

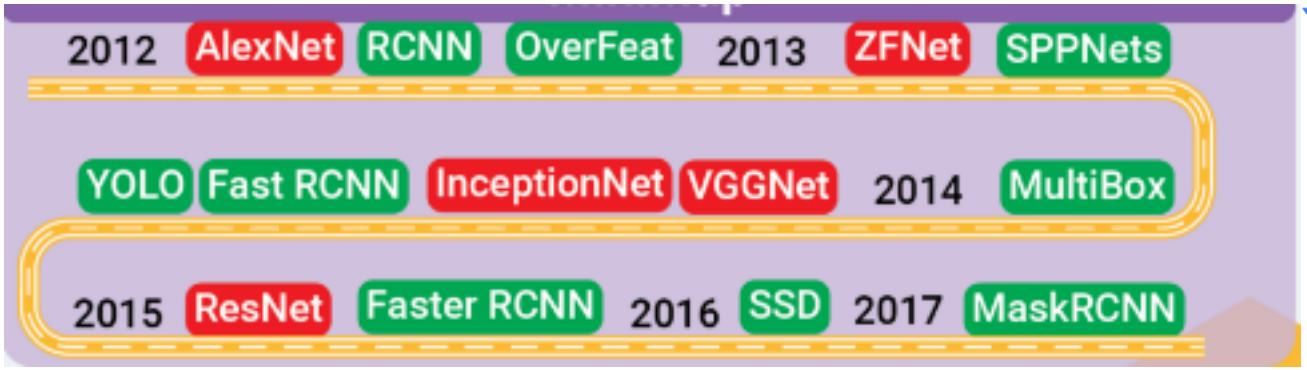
- 분류 : 영상 전체에 대한 분류
- 위치 탐지 : 영상내에서 대상의 위치 탐지
- 영상 인식 : 위치탐지와 대상의 분류



https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

영상 인식 모델

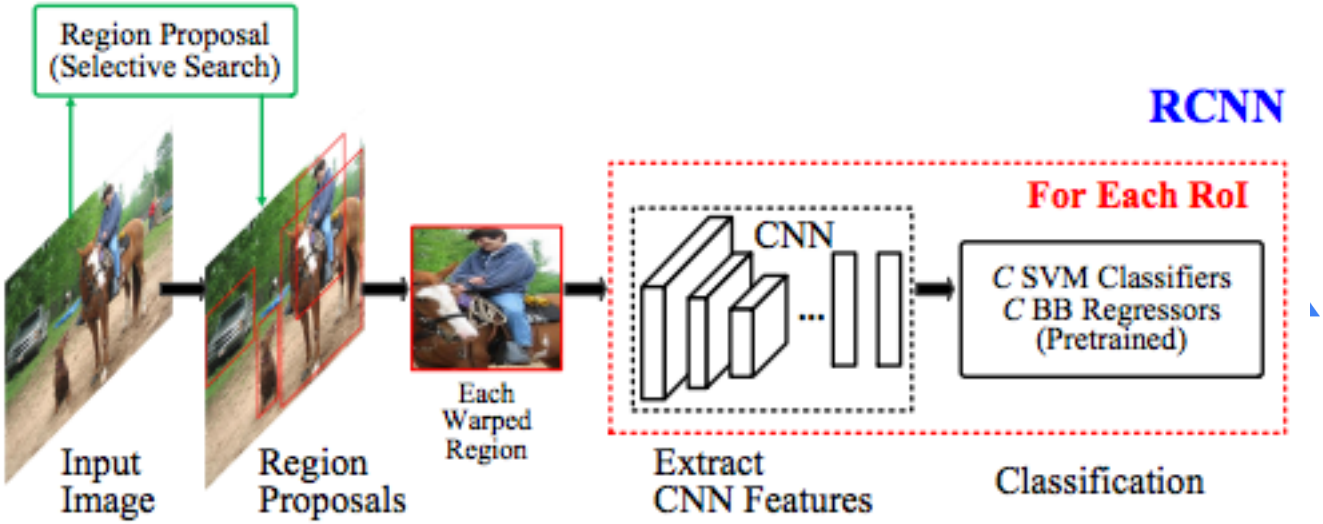
- 크게 두 종류로 나뉜다.
- 영역을 먼저 찾고, 그 영역을 분류하는 모델들
  - RCNN, Fast RCNN, Faster RCNN, Mask RCNN)
- 영역 찾기와 분류를 동시에 하는 모델들
  - YOLO, SSD



https://medium.com/@nikasa1889/the-modern-history-of-object-recognition-infographic-aea18517c318

영역 찾기와 분류를 순차적으로

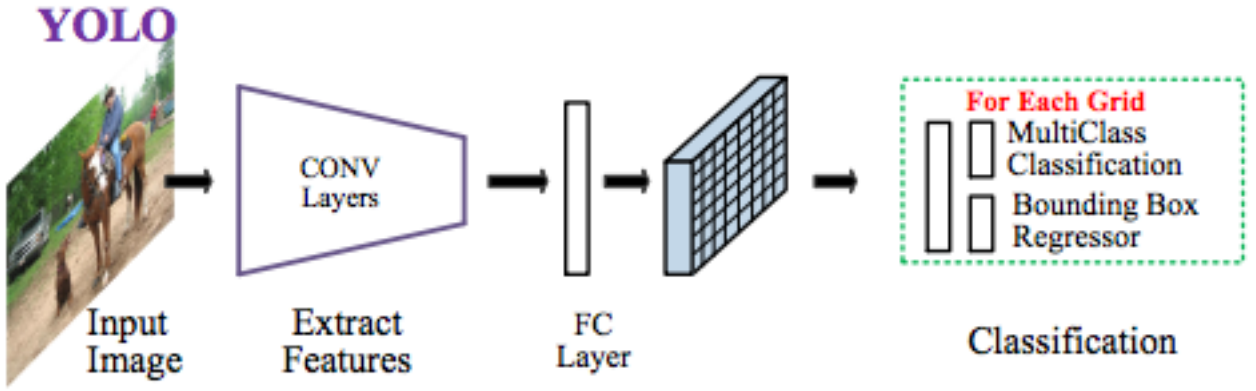
- *-RCNN 방법들
- 영역을 먼저 찾고, 해당 영역에 대하여 분류한다.



https://arxiv.org/pdf/1809.02165v1.pdf?fbclid=IwAR2_FnOpBS_scTRJv0EMR-dCPywWYLP3pTgXHoahtOjfp2XfiEH6eujmqKw

영역 찾기와 분류를 동시에

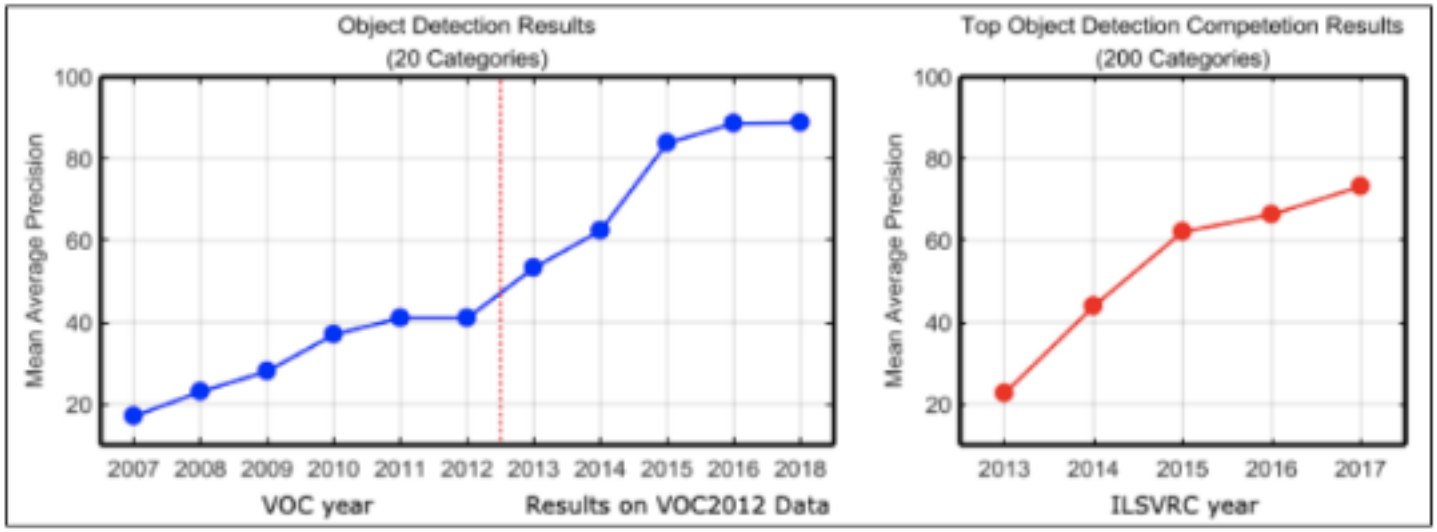
- YOLO, SSD의 방법
- 두개의 작업을 동시에 처리한다.



https://arxiv.org/pdf/1809.02165v1.pdf?fbclid=IwAR2_FnOpBS_scTRJv0EMR-dCPywWYLP3pTgXHoahtOjfp2XfiEH6eujmqKw

성능

- 딥러닝 적용 이후 90% 초반의 정도의 적확도



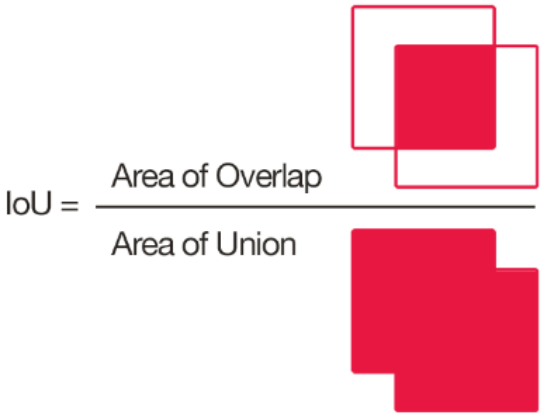
https://arxiv.org/pdf/1809.02165v1.pdf?fbclid=IwAR2_FnOpBS_scTRJv0EMR-dCPywWYLP3pTgXHoahtOjfp2XfiEH6eujmqKw

성능 지표

- IoU, mAP, AR이 많이 사용된다.

성능 지표 – IoU

- Intersection over Union
- 정답과 탐지해낸 결과의 영역 일치정도



https://www.hackerearth.com/blog/machine-learning/object-detection-for-self-driving-cars/

성능 지표 – mAP, AR

- mean Average Precision
- 여러개의 물체가 탐지된 경우 각 정확도의 평균

- Average Recall
- 탐지해야 할 물체 중 정확히 탐지한 갯수의 비율의 평균

YOLO로 물체 탐지

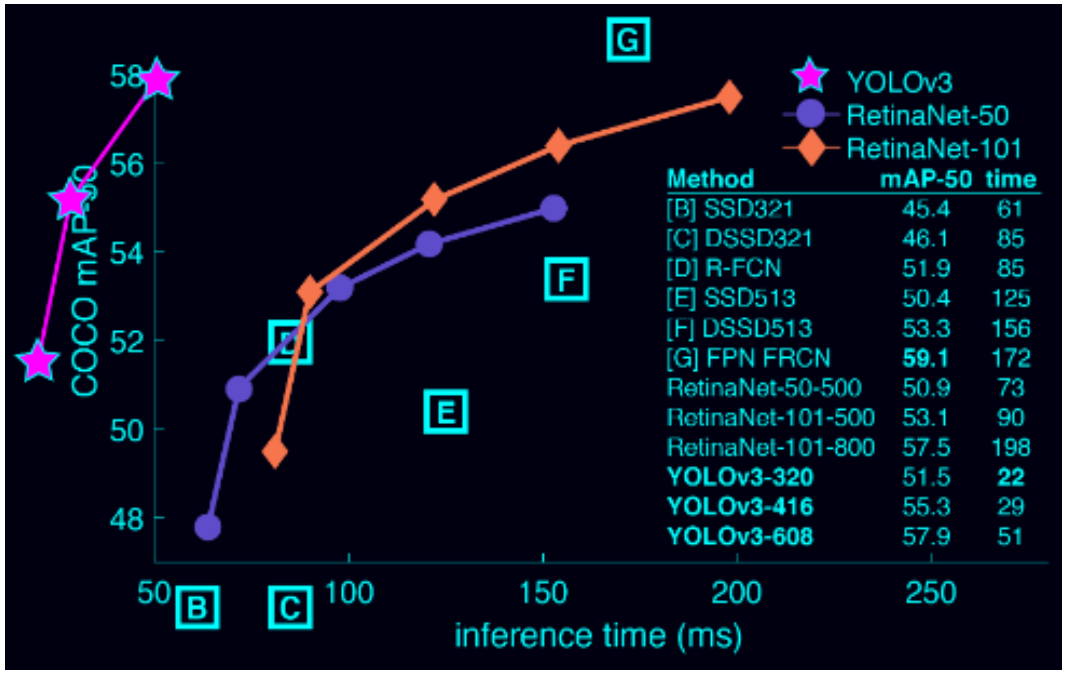
YOLO3

- Yoo Only Look Once 약자. 1회를 강조한 이름
- 영역 탐지와 분류를 동시에 실행
- 빠르다는 것이 가장 큰 장점
- 정확도는 다소 떨어진다.

https://www.slideshare.net/ssuser06e0c5/convolutional-neural-networks-135496264

성능

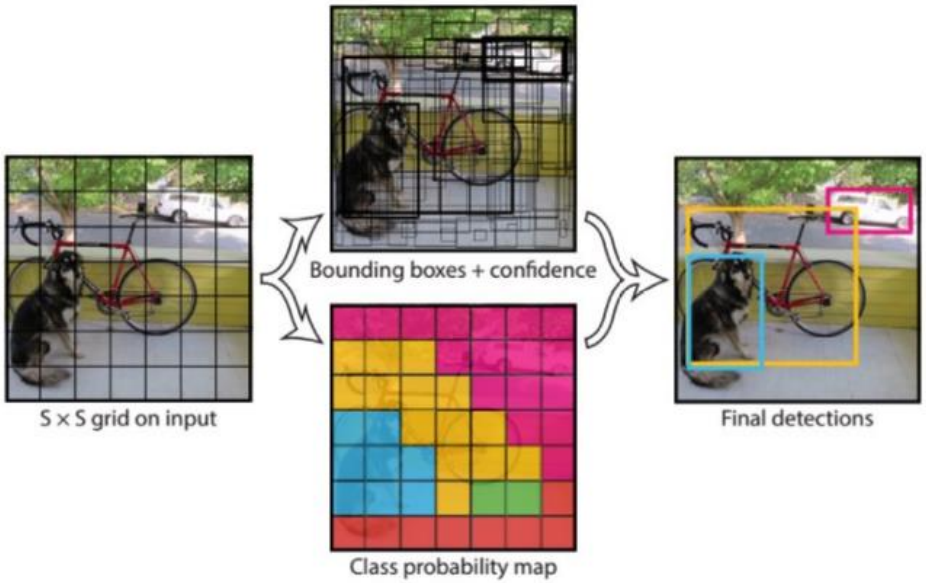
- 초당 20번 정도



https://pjreddie.com/darknet/yolo/

탐지 방식

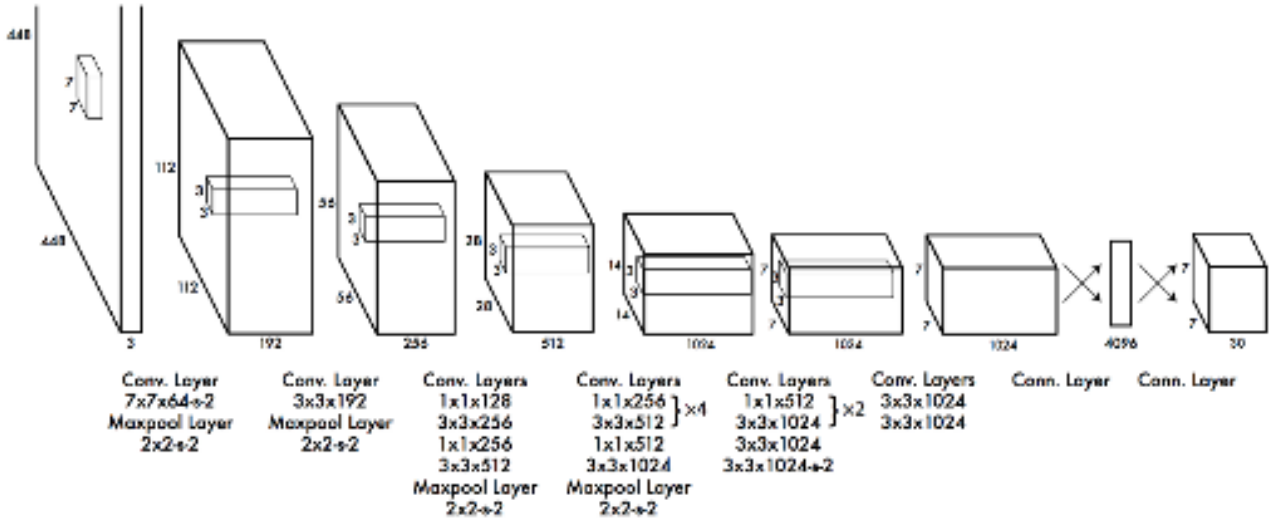
- 영역의 박스와 해당 박스의 분류를 동시에 복수개 출력하는 네트웍.
- 모두 98개(7x7x2)가 제안되고.
- 중복되지 않고 확신도가 높은 것만 추린다.



https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

구조

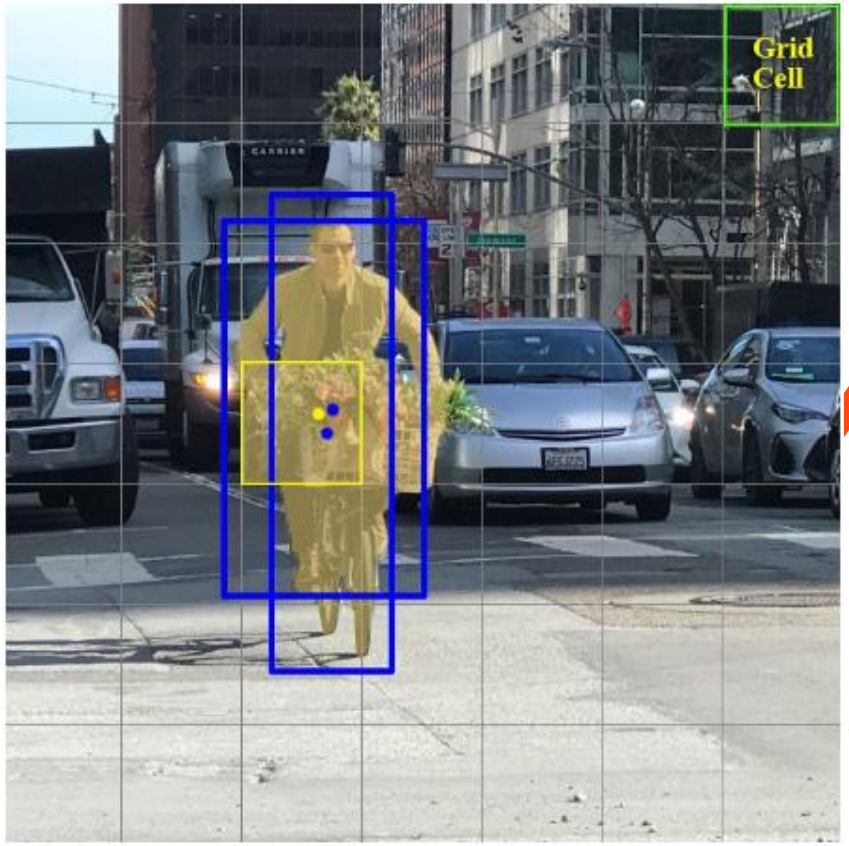
- GoogLeNet을 기반으로 한 CNN을 사용.
- 입력은 일반 이미지를 입력으로 받는다.
- 출력은 98개 상자에 대한 정보(위치, 폭과 높이)와 클래스와 확신도를 출력으로 한다.



https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

각 셀의 탐지

- 각 셀별 2개의 박스를 탐지
- 셀에 중심을 두는 박스
- 각 박스 별로 분류



https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

성

- |

https://www.hackerearth.com/blog/machine-learning/object-detection-for-self-driving-cars/

YOLO로
커스텀 데이타 학습

VOC 데이터 포멧

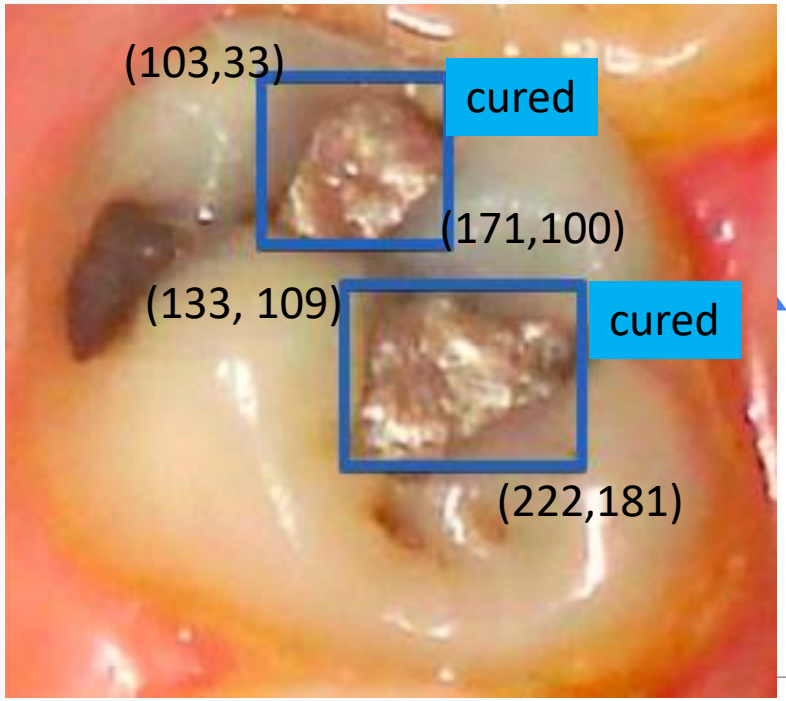
- 루트 폴더 아래 2개의 폴더와 1개 파일이 있다.
  - Annotations : 레이블링 xml 파일이 있다.
  - JPEGImages : 이미지 파일이 있다.
  - classes.txt :

```
dental_image_in_voc_format/
  classes.txt
  Annotations/
    301.xml
    302.xml
    ...
  JPEGImages/
    301.jpg
    302.jpg
```

레이블링 데이타 파일 예

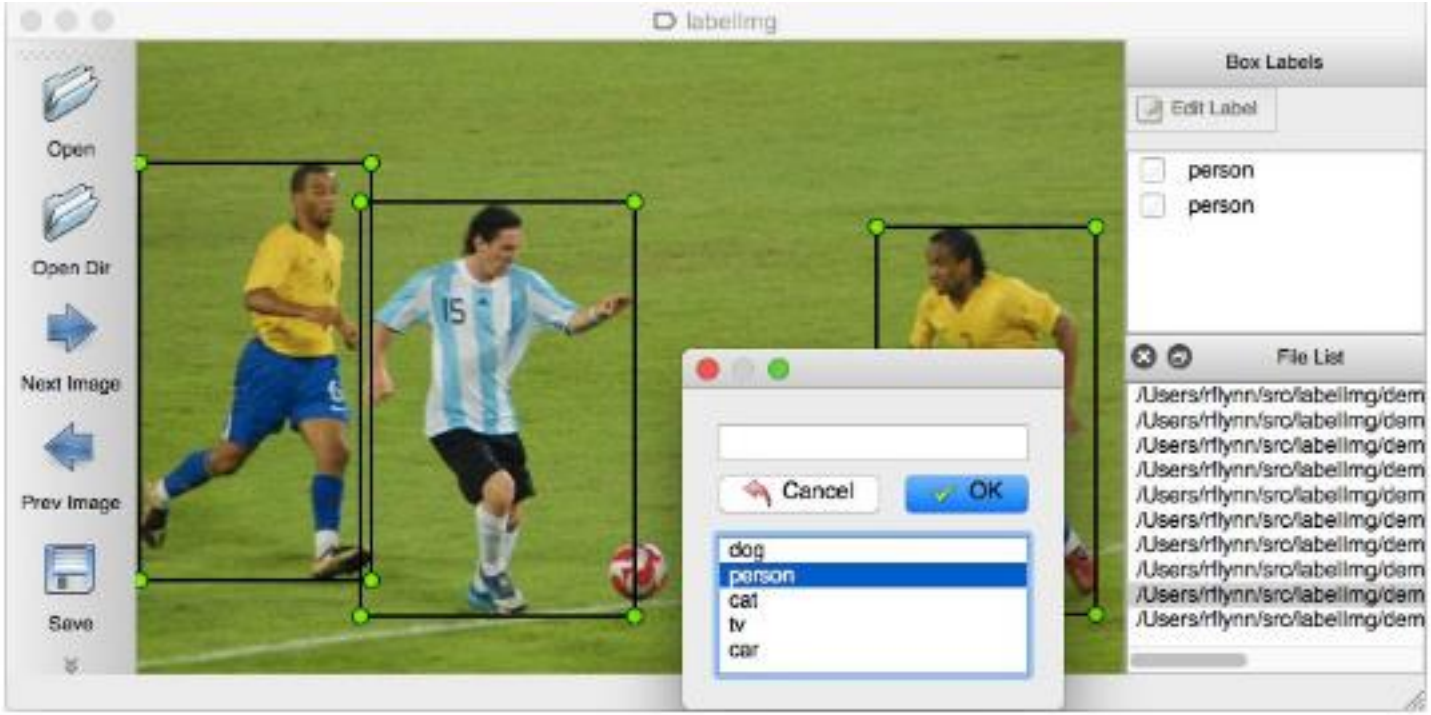
- Annotations/301.xml

```
<annotation>
  <path>dental_image_in_voc_format/JPEGImages/301.jpg</path>
  <object>
    <bndbox>
      <xmin>103</xmin>
      <ymin>33</ymin>
      <xmax>171</xmax>
      <ymax>100</ymax>
    </bndbox>
  </object>
  <object>
    <bndbox>
      <xmin>133</xmin>
      <ymin>109</ymin>
      <xmax>222</xmax>
      <ymax>181</ymax>
    </bndbox>
  </object>
</annotation>
```



레이블링 툴

- https://github.com/tzutalin/labelImg 를 사용.



darknet의 레이블링 데이터 포멧

- 2개의 파일과 1개의 폴더가 있다.
  - train_all.txt : 레이블링 데이터가 담긴 파일. 이름은 임의 가능
  - JPEGImages : 이미지 파일이 있다.
  - classes.txt : 클래스의 이름이 담긴 파일

```
train_all.txt
dental_image_in_voc_format/
  classes.txt
  JPEGImages/
    301.jpg
    302.jpg
    ...
```

https://www.hackerearth.com/blog/machine-learning/object-detection-for-self-driving-cars/

레이블링 데이터 파일 예

- train_all.txt

```
dental_image_in_voc_format/JPEGImages/301.jpg 103,33,171,100,2 133,109,222,181,2
dental_image_in_voc_format/JPEGImages/302.jpg 260,6,556,304,2
dental_image_in_voc_format/JPEGImages/303.jpg 16,4,541,518,2
dental_image_in_voc_format/JPEGImages/304.jpg 5,41,245,239,2
dental_image_in_voc_format/JPEGImages/305.jpg 45,25,83,82,2
dental_image_in_voc_format/JPEGImages/306.jpg 32,23,69,78,2
```

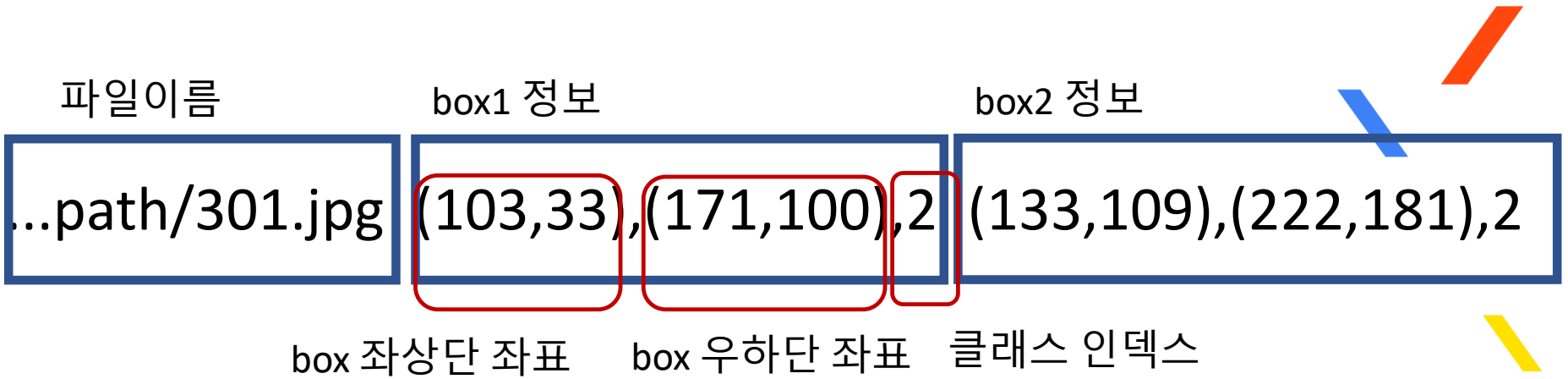
- VOC 포멧의 Annotations밑의 xml파일의 내용을 이파일에 담는다.

https://www.hackerearth.com/blog/machine-learning/object-detection-for-self-driving-cars/

레이블링 데이터 포멧

- (파일이름) + (box1정보) + (box2정보) + …

```
dental_image_in_voc_format/JPEGImages/301.jpg 103,33,171,100,2 133,109,222,181,2
```



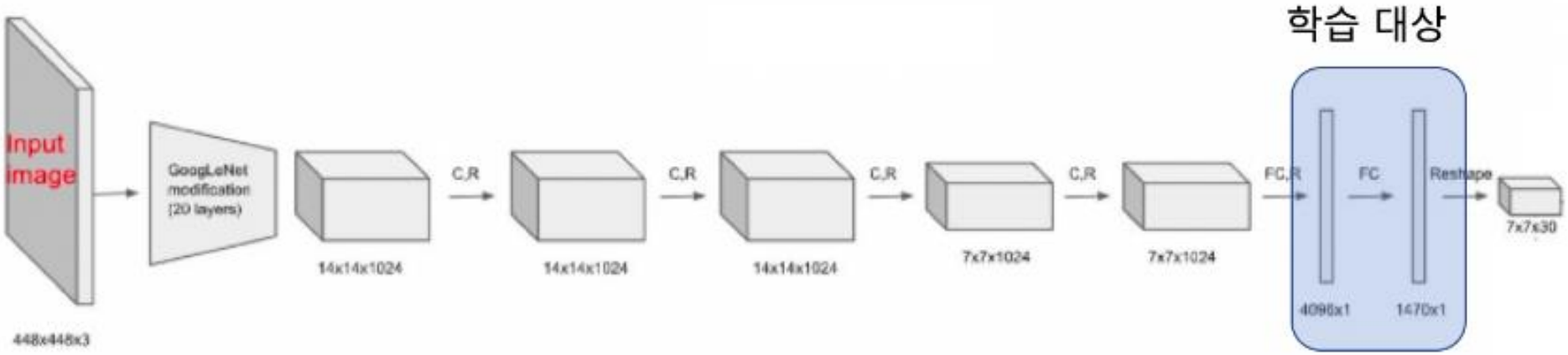
https://www.hackerearth.com/blog/machine-learning/object-detection-for-self-driving-cars/

VOC 포멧을 darknet 포멧으로 변환

- 레이블링 툴의 경우 일반적으로 VOC 포멧을 지원한다.
- YOLO로 학습하기 위해서 변환이 필요하다.

전이학습

- 원 모델에서 Fully Connted 레이버만 학습한다.

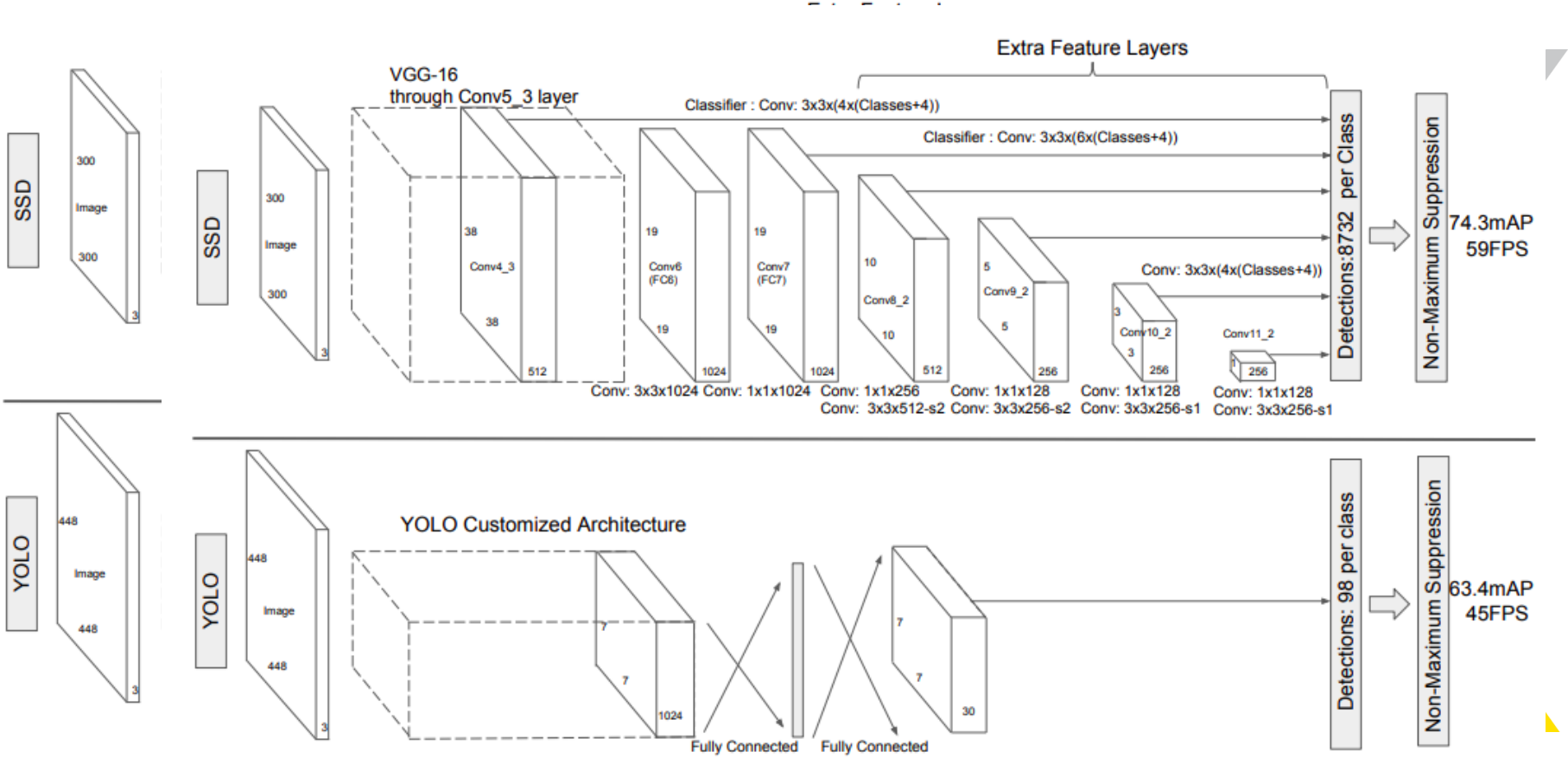


SSD로 물체 탐지

SSD

- Single Shot Detector. 역시 1회를 강조한 이름
- VGG16을 기반으로 한다.
- full conntected 레이를 없에고, 각 conv 레이어의 값을 하나로 받는다.
- YOLO는 98개 박스를 출력하는데 비해, 5776개의 박스를 출력한다.

SSD 구조



https://arxiv.org/pdf/1512.02325.pdf