Deep Learning for Biologists

EunAh Lee, PhD

Curriculum

- Introduction to BME
- Fundamentals of Biochemistry & Cell Biology
- Principles of Tissue Engineering & Regenerative Medicine
- Convergence Technology for Regenerative Medicine
  - Materials Science
  - AI Technology

DNN 응용 알고리즘

scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
Text Data
NOT WORKING
Linear SVC
YES
<100K samples
YES
get more data
NO
>50 samples
YES
predicting a category
YES
do you have labeled data
NO
NO
predicting a quantity
YES
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NO
<100K samples
YES
few features should be important
YES
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
YES
number of categories known
<10K samples
NO
NO
<10K samples
YES
NO
MiniBatch KMeans
MeanShift
VBGMM
tough luck
NO
just looking
NO
predicting structure
YES
Randomized PCA
NOT WORKING
YES
<10K samples
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
dimensionality reduction
Back
scikit learn

DNN 기반 응용 예시

영상인식

- MNIST 분류 문제
- CIFAR10 분류

DNN as Classifier
CNN
VGG16
GAN

영상 데이터

영상데이터 - 가로 세로로 배열된 픽셀의 값을 숫자로 표시한 데이터

- 영상은 픽셀로 구성됨
- 회색 영상은 픽셀당 밝기 값 1개로 나타냄
- 칼라 영상은 1 픽셀당 RGB 3개의 표시 값으로 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/

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

https://bwessay.tistory.com/90

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

DNN MNIST 분류문제

- "Hello World" in Machine Learning
- MNIST stands for Modified National Institute of Standards and Technology
- 입력 노드 784개 (input image: 28*28 pixel)
- 10개 카테고리 (0~9), 10개 출력
- 입력 feature 갯수가 많음

양의 가중치

음의 가중치

출력노드에 SoftMax 적용

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/mnist/beginners/

CIFAR10 분류

- **CIFAR** stands for **Canadian Institute For Advanced Research**
- The **CIFAR-10 dataset** is a collection of images that are commonly used to train machine learning and computer vision algorithms.
- It is one of the most widely used datasets for machine learning research.
- The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes.
  - The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.
- Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works.
- Various kinds of convolutional neural networks tend to be the best at recognizing the images in CIFAR-10.

Here are the classes in the dataset, as well as 10 random images from each:

https://www.cs.toronto.edu/~kriz/cifar.html

Research Papers Claiming State-of-the-Art Results on CIFAR-10

Research Paper	Error rate (%)	Publication Date
Convolutional Deep Belief Networks on CIFAR-10[6]	21.1	August, 2010
Maxout Networks[7]	9.38	February 13, 2013
Wide Residual Networks[8]	4.0	May 23, 2016
Neural Architecture Search with Reinforcement Learning[9]	3.65	November 4, 2016
Fractional Max-Pooling[10]	3.47	December 18, 2014
Densely Connected Convolutional Networks[11]	3.46	August 24, 2016
Shake-Shake regularization[12]	2.86	May 21, 2017
Coupled Ensembles of Neural Networks[13]	2.68	September 18, 2017
ShakeDrop regularization[14]	2.67	Feb 7, 2018
Improved Regularization of Convolutional Neural Networks with Cutout[15]	2.56	Aug 15, 2017
Regularized Evolution for Image Classifier Architecture Search[16]	2.13	Feb 6, 2018
Rethinking Recurrent Neural Networks and other Improvements for Image Classification[17]	1.64	July 31, 2020
AutoAugment: Learning Augmentation Policies from Data[18]	1.48	May 24, 2018
A Survey on Neural Architecture Search[19]	1.33	May 4, 2019
GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism[20]	1.00	Nov 16, 2018

cifar10

- **설명** :

CIFAR-10 데이터 세트는 10 개 클래스의 60000 32x32 컬러 이미지와 클래스 당 6000 개의 이미지로 구성됩니다. 50000 개의 훈련 이미지와 10000 개의 테스트 이미지가 있습니다.

- **홈페이지** : https://www.cs.toronto.edu/~kriz/cifar.html
- **소스 코드** : `tfds.image_classification.Cifar10`
- **버전** :
  - `3.0.2` (기본값) : 릴리즈 노트가 없습니다.
- **다운로드 크기** : `162.17 MiB`
- **데이터 세트 크기** : `132.40 MiB`
- **자동 캐시** (문서) : 예
- **스플릿** :

영상분류

- 대상을 미리 정해진 클래스로 분류하는 작업
- 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

분류기 DNN의 입력

- 입력은 영상 데이터 자체
- 입력 노드의 수는 영상 데이터의 값 개수와 같음

분류기 DNN의 출력

- 출력 노드는 클래스의 개수만큼
- 출력된 값은 0~1사이의 값을 가짐
- 가장 큰 값을 가지는 노드에 해당되는 클래스로 분류됨

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

2012년 AI 부활

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
: ImageNet data set을 가지고 겨루는 대회

* ImageNet: 일반적인 사물의 영상의 데이터 셋. 1,000개의 클래스 존재

Revolution of Depth

최신 ← 과거

ILSVRC 우승 모델들

GoogLeNet
ResNet
VGG16

- 모델 구조 뿐 아니라 학습된 모델도 공개되어 있음
- 기술적으로 완숙되어 있음
- Keras를 설치하면 VGG나 ResNet이 default로 포함되어 있음
- 그대로 가져다 사용가능 (Open Source License 준수)

VGG16

- 2014 ILSVRC에서 2등한 모델
- Convolution layer와 max pooling layer가 있는 일반 CNN
- 21개의 Layer를 가짐
- 학습 대상 weight는 대략 1억 4천만개
- 모델 파일 크기는 약 500M
- Pooling 레이어를 제외하고 모두 16개의 레이어 존재
- Image size는 224*224
- 처리시간 많이 소요하므로 ROI만 crop 해서 처리하기도 함

Layer (1-11)	Volume	Parameters	Layer (12-22)	Volume	Parameters
INPUT	224 × 224 × 3	0	CONV3-512	28 × 28 × 512	(3 * 3 * 256) * 512 = 1,179,648
CONV3-64	224 × 224 × 64	(3 * 3 * 3) * 64 = 1,728	CONV3-512	28 × 28 × 512	(3 * 3 * 512) * 512 = 2,359,296
CONV3-64	224 × 224 × 64	(3 * 3 * 64) * 64 = 36,864	CONV3-512	28 × 28 × 512	(3 * 3 * 512) * 512 = 2,359,296
POOL2	112 × 112 × 64	0	POOL2	14 × 14 × 512	0
CONV3-128	112 × 112 × 128	(3 * 3 * 64) * 128 = 73,728	CONV3-512	14 × 14 × 512	(3 * 3 * 512) * 512 = 2,359,296
CONV3-128	112 × 112 × 128	(3 * 3 * 128) * 128 = 147,456	CONV3-512	14 × 14 × 512	(3 * 3 * 512) * 512 = 2,359,296
POOL2	56 × 56 × 128	0	CONV3-512	14 × 14 × 512	(3 * 3 * 512) * 512 = 2,359,296
CONV3-256	56 × 56 × 256	(3 * 3 * 128) * 256 = 294,912	POOL2	7 × 7 × 512	0
CONV3-256	56 × 56 × 256	(3 * 3 * 256) * 256 = 589,824	FC-1	1 × 1 × 4096	7 * 7 * 512 * 4096 = 102,760,448
CONV3-256	56 × 56 × 256	(3 * 3 * 256) * 256 = 589,824	FC-2	1 × 1 × 4096	4096 * 4096 = 16,777,216
POOL2	28 × 28 × 256	0	FC-3	1 × 1 × 2622	4096 * 2622 = 10,739,712

https://file.scirp.org/Html/4-7800353_65406.htm

Hachim El Khiyari, Harry Wechsler (2016) Face Recognition across Time Lapse Using Convolutional Neural Networks. *Journal of Information Security*, **07**, 141-151. doi: 10.4236/jis.2016.73010

필터수

카테고리 개수: 1000개

CNN (Convolutional Neural Network)

CNN 주요 용어 정리

합성곱(Convolution)

채널(Channel)

필터(Filter)

커널(Kernel)

스트라이드(Strid)

패딩(Padding)

피처 맵(Feature Map)

액티베이션 맵(Activation Map)

풀링(Pooling) 레이어

Convolution Filter 적용 과정

Convolution (합성곱) → Feature map

1개의 필터로 합성곱 연산을 수행하는 과정

Filter, Kernel, & Stride

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
http://taewan.kim/post/cnn/

Conv/FC Filters. The second common strategy is to visualize the weights. These are usually most interpretable on the first CONV layer which is looking directly at the raw pixel data, but it is possible to also show the filter weights deeper in the network. The weights are useful to visualize because well-trained networks usually display nice and smooth filters without any noisy patterns. Noisy patterns can be an indicator of a network that hasn't been trained for long enough, or possibly a very low regularization strength that may have led to overfitting.

https://cs231n.github.io/understanding-cnn/

Channel (채널)

컬러 이미지는 3개의 채널로 구성됨

이미지 출처: https://en.wikipedia.org/wiki/Channel_(digital_image)

Padding

Pooling or MaxPooling

Feature Map or Activation Map

http://xujuefei.com/pnn.html

Feature Map or Activation Map

https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

CNN = feature extraction + classification

이미지 처리에 뛰어나며, 이미지 이외의 것에도 좋은 성능을 보임
같은 문제에 대해서 모델만 바꿔서 실행하고 결과를 비교하면 DNN보다 CNN의 acc가 더 높음 (0.92 vs 0.97)

MNIST 코드

mnist_normalization.ipynb

MNIST DNN Classification Template

```
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import optimizers
from tensorflow.keras.layers import Dense

import time


(train_x, train_y), (test_x, test_y) = tf.keras.datasets.mnist.load_data()

train_x = train_x/255
test_x = test_x/255

#(None,28,28)의 train_x는 그대로 사용, 모델의 처음에 Flatten() 레이어를 두어 입력 모양을 변경
model = keras.Sequential()
model.add(Flatten(input_shape=(28, 28)))#ADD
#model.add(Dense(10, activation-'relu', input_shape=(28*28,))) #COMMENT OUT
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.summary()


model.fit(train_x, train_y, epochs=5, verbose=1, batch_size=128)


loss, acc = model.evaluate(test_x, test_y)
print("loss=",loss)
print("acc=",acc)

y_ = model.predict(test_x)
predicted = np.argmax(y_, axis=1)

print(predicted)
```

Normalization Test

0~255값으로 학습 실행

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_21 (Dense)             (None, 10)                7850
_________________________________________________________________
dense_22 (Dense)             (None, 10)                110
_________________________________________________________________
dense_23 (Dense)             (None, 10)                110
=================================================================
Total params: 8,070
Trainable params: 8,070
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 1s 19us/sample - loss: 3.0471 - acc: 0.1124
Epoch 2/5
60000/60000 [==============================] - 1s 17us/sample - loss: 2.2535 - acc: 0.1387
Epoch 3/5
60000/60000 [==============================] - 1s 18us/sample - loss: 2.1214 - acc: 0.1904
Epoch 4/5
60000/60000 [==============================] - 1s 18us/sample - loss: 1.9593 - acc: 0.2617
Epoch 5/5
60000/60000 [==============================] - 1s 18us/sample - loss: 1.8325 - acc: 0.2973
elapsed : 5.841329097747803
10000/10000 [==============================] - 0s 35us/sample - loss: 1.7931 - acc: 0.3206
loss= 1.7930989349365234
acc= 0.3206
[1 0 1 ... 1 1 1]
```

0~1

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_5 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_24 (Dense)             (None, 10)                7850
_________________________________________________________________
dense_25 (Dense)             (None, 10)                110
_________________________________________________________________
dense_26 (Dense)             (None, 10)                110
=================================================================
Total params: 8,070
Trainable params: 8,070
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 1s 22us/sample - loss: 0.8692 - acc: 0.7350
Epoch 2/5
60000/60000 [==============================] - 1s 21us/sample - loss: 0.3617 - acc: 0.8978
Epoch 3/5
60000/60000 [==============================] - 1s 20us/sample - loss: 0.3028 - acc: 0.9143
Epoch 4/5
60000/60000 [==============================] - 1s 20us/sample - loss: 0.2748 - acc: 0.9215
Epoch 5/5
60000/60000 [==============================] - 1s 21us/sample - loss: 0.2569 - acc: 0.9269
elapsed : 6.673229932785034
10000/10000 [==============================] - 0s 36us/sample - loss: 0.2497 - acc: 0.9284
loss= 0.24970011441707612
acc= 0.9284
[7 2 1 ... 4 5 6]
```

-1~1

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_6 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_27 (Dense)             (None, 10)                7850
_________________________________________________________________
dense_28 (Dense)             (None, 10)                110
_________________________________________________________________
dense_29 (Dense)             (None, 10)                110
=================================================================
Total params: 8,070
Trainable params: 8,070
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 1s 21us/sample - loss: 1.7792 - acc: 0.3451
Epoch 2/5
60000/60000 [==============================] - 1s 20us/sample - loss: 1.3118 - acc: 0.5363
Epoch 3/5
60000/60000 [==============================] - 1s 19us/sample - loss: 1.1773 - acc: 0.5819
Epoch 4/5
60000/60000 [==============================] - 1s 19us/sample - loss: 1.1196 - acc: 0.6064
Epoch 5/5
60000/60000 [==============================] - 1s 19us/sample - loss: 1.0899 - acc: 0.6149
elapsed : 6.3451361656188965
10000/10000 [==============================] - 0s 36us/sample - loss: 1.0563 - acc: 0.6366
loss= 1.0562564421653748
acc= 0.6366
[7 2 1 ... 9 5 6]
```

-255~255

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_7 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_30 (Dense)             (None, 10)                7850
_________________________________________________________________
dense_31 (Dense)             (None, 10)                110
_________________________________________________________________
dense_32 (Dense)             (None, 10)                110
=================================================================
Total params: 8,070
Trainable params: 8,070
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 1s 20us/sample - loss: 3.1857 - acc: 0.1844
Epoch 2/5
60000/60000 [==============================] - 1s 18us/sample - loss: 1.9233 - acc: 0.2795
Epoch 3/5
60000/60000 [==============================] - 1s 19us/sample - loss: 1.5677 - acc: 0.4216
Epoch 4/5
60000/60000 [==============================] - 1s 19us/sample - loss: 1.2807 - acc: 0.5186
Epoch 5/5
60000/60000 [==============================] - 1s 19us/sample - loss: 1.1611 - acc: 0.5695
elapsed : 6.1723597049713135
10000/10000 [==============================] - 0s 36us/sample - loss: 1.1311 - acc: 0.5796
loss= 1.1311326502799988
acc= 0.5796
[7 2 1 ... 4 5 4]
```

CNN

MNIST Dataset

mnist_by_cnn.ipynb

Data 준비

```
#Data 준비
(raw_train_x, raw_train_y), (raw_test_x, raw_test_y) =
tf.keras.datasets.mnist.load_data()

train_x = raw_train_x/255 #Normalization
test_x = raw_test_x/255

# Conv2D는 width*height*channel을 입력으로 받음
train_x = train_x.reshape((60000, 28, 28, 1)) # ADDED
test_x = test_x.reshape((10000, 28, 28, 1)) # ADDED

train_y = raw_train_y
test_y = raw_test_y
```

Model

```
model = keras.Sequential()
# model.add(Input((28,28)))
model.add(Input((28,28,1)))
model.add(Conv2D(32, (3, 3))) # ADDED
model.add(MaxPooling2D((2, 2))) # ADDED
model.add(Conv2D(64, (3, 3))) # ADDED
model.add(MaxPooling2D((2, 2))) # ADDED
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
model.summary()


model.fit(train_x, train_y, epochs=5, verbose=1, batch_size=128) #'berbose=1'
진행 상태를 막대로 표시


loss, acc = model.evaluate(test_x, test_y)
print("loss=",loss)
print("acc=",acc)

y_ = model.predict(test_x)
predicted = np.argmax(y_, axis=1)

print(predicted)
```

Conv2D 레이어의 Param#는 다음에 의해 결정됨

Parameter # = (conv필터 가로 * conv필터 세포* 데이터 깊이+bias 1개)*필터 수

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_varial
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
=================================================================
Total params: 35,046
Trainable params: 35,046
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 40s 673us/sample - loss: 0.4828 - acc: 0.8499
Epoch 2/5
60000/60000 [==============================] - 40s 665us/sample - loss: 0.1434 - acc: 0.9583
Epoch 3/5
60000/60000 [==============================] - 40s 663us/sample - loss: 0.1036 - acc: 0.9692
Epoch 4/5
60000/60000 [==============================] - 40s 663us/sample - loss: 0.0828 - acc: 0.9756
Epoch 5/5
60000/60000 [==============================] - 40s 666us/sample - loss: 0.0688 - acc: 0.9793
10000/10000 [==============================] - 3s 313us/sample - loss: 0.0762 - acc: 0.9752
loss= 0.07622277518818155
acc= 0.9752
[7 2 1 ... 4 5 6]
```

CNN

CIFAR10 Dataset

CIFAR Dataset

CIFAR-10 dataset

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum
CIFAR-10 python version	163 MB	c58f30108f718f92721af3b95e74349a
CIFAR-10 Matlab version	175 MB	70270af85842c9e89bb428ec9976c926
CIFAR-10 binary version (suitable for C programs)	162 MB	c32a1d4ab5d03f1284b67883e8d87530

CIFAR-100 dataset

The CIFAR-100 dataset

This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).
Here is the list of classes in the CIFAR-100:

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Yes, I know mushrooms aren't really fruit or vegetables and bears aren't really carnivores.

cifar10.ipynb

Code

Data 준비

```
# (raw_train_x, raw_train_y), (raw_test_x, raw_test_y) =
tf.keras.datasets.mnist.load_data() # COMMENT OUT
(raw_train_x, raw_train_y), (raw_test_x, raw_test_y) =
tf.keras.datasets.cifar10.load_data()

print(raw_train_x.shape)
print(raw_train_y.shape)
print(raw_test_x.shape)
print(raw_test_y.shape)


train_x = raw_train_x/255
test_x = raw_test_x/255

# train_x = train_x.reshape((60000, 28, 28, 1)) # COMMENT OUT
# test_x = test_x.reshape((10000, 28, 28, 1)) # COMMENT OUT

train_y = raw_train_y
test_y = raw_test_y
```

출력내용

```
(50000, 32, 32, 3)
(50000, 1)
(10000, 32, 32, 3)
(10000, 1)
```

모델 (epochs=5)

```
model = keras.Sequential()
# model.add(Input((28,28,1)))
model.add(Input((32,32,3)))
model.add(Conv2D(32, (3, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accurac
y"])
model.summary()


model.fit(train_x, train_y, epochs=5, verbose=1, batch_size=128)


loss, acc = model.evaluate(test_x, test_y)
print("loss=",loss)
print("acc=",acc)

y_ = model.predict(test_x)
predicted = np.argmax(y_, axis=1)

print(predicted)
```

학습 결과

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_varial
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 10)                23050
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
=================================================================
Total params: 42,662
Trainable params: 42,662
Non-trainable params: 0
_________________________________________________________________
Train on 10000 samples
Epoch 1/5
10000/10000 [==============================] - 11s 1ms/sample - loss: 2.2019 - acc: 0.1683
Epoch 2/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 2.0426 - acc: 0.2273
Epoch 3/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 1.9005 - acc: 0.2980
Epoch 4/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 1.7785 - acc: 0.3458
Epoch 5/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 1.6797 - acc: 0.3848
10000/10000 [==============================] - 4s 409us/sample - loss: 1.6338 - acc: 0.4060
loss= 1.6337748119354247
acc= 0.406
[3 1 8 ... 5 7 7]
```

모델 (epochs=50)

```
model = keras.Sequential()
# model.add(Input((28,28,1)))
model.add(Input((32,32,3)))
model.add(Conv2D(32, (3, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accurac
y"])
model.summary()


model.fit(train_x, train_y, epochs=50, verbose=1, batch_size=128)


loss, acc = model.evaluate(test_x, test_y)
print("loss=",loss)
print("acc=",acc)

y_ = model.predict(test_x)
predicted = np.argmax(y_, axis=1)

print(predicted)
```

학습 결과

```
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.7637 - acc: 0.7385
Epoch 31/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.7433 - acc: 0.7397
Epoch 32/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.7077 - acc: 0.7528
Epoch 33/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.7095 - acc: 0.7540
Epoch 34/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.6813 - acc: 0.7629
Epoch 35/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.6603 - acc: 0.7693
Epoch 36/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.6393 - acc: 0.7783
Epoch 37/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.6212 - acc: 0.7868
Epoch 38/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5925 - acc: 0.7983
Epoch 39/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5745 - acc: 0.8055
Epoch 40/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5787 - acc: 0.7996
Epoch 41/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5549 - acc: 0.8075
Epoch 42/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5324 - acc: 0.8187
Epoch 43/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5264 - acc: 0.8196
Epoch 44/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.5097 - acc: 0.8238
Epoch 45/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.4876 - acc: 0.8311
Epoch 46/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.4675 - acc: 0.8395
Epoch 47/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.4610 - acc: 0.8440
Epoch 48/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.4412 - acc: 0.8513
Epoch 49/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.4216 - acc: 0.8593
Epoch 50/50
10000/10000 [==============================] - 10s 1ms/sample - loss: 0.4085 - acc: 0.8629
10000/10000 [==============================] - 4s 409us/sample - loss: 2.0022 - acc: 0.5149
loss= 2.002237993431091
acc= 0.5149
[3 8 0 ... 2 0 4]
```

영상 데이터 증강

ImageDataGenerator

imagedatagenerator.ipynb

ImageDataGenerator 사용

Data 준비

```
(raw_train_x, raw_train_y), (raw_test_x, raw_test_y) =
tf.keras.datasets.cifar10.load_data()

print(raw_train_x.shape)
print(raw_train_y.shape)
print(raw_test_x.shape)
print(raw_test_y.shape)

train_x = raw_train_x/255
test_x = raw_test_x/255

train_y = raw_train_y
test_y = raw_test_y

train_x = train_x[:10000] #Data중 일부만 사용
train_y = train_y[:10000]
```

ImageDataGenerator 옵션 설정

```
from tensorflow.keras.preprocessing.image
import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10, # 0 ~ 180
    width_shift_range=0.1,
    height_shift_range=0.1,
    fill_mode='nearest',
    horizontal_flip=True,
    vertical_flip=False
)
```

Model

학습결과 - Loss ↓, Acc ↑, elapsed time ↑

```
model = keras.Sequential()
model.add(Input((32,32,3)))
model.add(Conv2D(32, (3, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer="adam", loss="sparse_c
metrics=["accuracy"])
model.summary()


# model.fit(train_x, train_y, epochs=5, verbose=1,
model.fit_generator(datagen.flow(train_x, train_y,


loss, acc = model.evaluate(test_x, test_y)
print("loss=",loss)
print("acc=",acc)

y_ = model.predict(test_x)
predicted = np.argmax(y_, axis=1)

print(predicted)
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_varia
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 10)                23050
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
=================================================================
Total params: 42,662
Trainable params: 42,662
Non-trainable params: 0
_________________________________________________________________
Train on 10000 samples
Epoch 1/5
10000/10000 [==============================] - 11s 1ms/sample - loss: 2.2019 - acc: 0.1683
Epoch 2/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 2.0426 - acc: 0.2273
Epoch 3/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 1.9005 - acc: 0.2980
Epoch 4/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 1.7785 - acc: 0.3458
Epoch 5/5
10000/10000 [==============================] - 10s 1ms/sample - loss: 1.6797 - acc: 0.3848
10000/10000 [==============================] - 4s 409us/sample - loss: 1.6338 - acc: 0.4060
loss= 1.6337748119354247
acc= 0.406
[3 1 8 ... 5 7 7]
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                23050
_________________________________________________________________
dense_7 (Dense)              (None, 10)                110
_________________________________________________________________
dense_8 (Dense)              (None, 10)                110
=================================================================
Total params: 42,662
Trainable params: 42,662
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
391/391 [==============================] - 71s 183ms/step - loss: 1.9487 - acc: 0.2704
Epoch 2/5
391/391 [==============================] - 71s 182ms/step - loss: 1.6360 - acc: 0.3927
Epoch 3/5
391/391 [==============================] - 74s 190ms/step - loss: 1.5213 - acc: 0.4397
Epoch 4/5
391/391 [==============================] - 74s 188ms/step - loss: 1.4448 - acc: 0.4750
Epoch 5/5
391/391 [==============================] - 71s 181ms/step - loss: 1.4078 - acc: 0.4901
10000/10000 [==============================] - 4s 420us/sample - loss: 1.3679 - acc: 0.5083
loss= 1.3679221759796143
acc= 0.5083
[1 8 8 ... 5 1 7]
```

영상 데이터 커스텀 학습

CNN = feature extraction + classification

Conv layer + Pooling layer

Fully connected layer

ImageNet 우승모델 대부분 CNN 기반 알고리즘 사용
이미지 처리에 뛰어나며, 이미지 이외의 것에도 좋은 성능을 보임 (트질 추출 및 분류 기능 우수)
같은 문제에 대해서 모델만 바꿔서 실행하고 결과를 비교하면 DNN보다 CNN의 acc가 더 높음 (0.92 vs 0.97)

맨 마지막 layer에는
SoftMax 적용

DNN

Custom Data 학습을 위한 전이학습

Custom data

- 보통 실무에서 사용하는 data는 ImageNet의 클래스와 다르다
- 이런 이유로 DNN의 출력 층 노드 수가 다를 수 밖에 없음
- 1,000개의 클래스의 분류를 그대로 사용할 수 없음

우승 모델 재활용

- 우승한 모델의 우수한 특질 추출 기능을 재활용해서 사용
- 대신 분류 부분을 다시 학습 시켜야 함

전이학습

- 학습된 모델을 가져와 다시 학습한다
- 특질 추출부분은 고정, 분류하는 부분만 학습

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

VGG로 Custom Data 학습 - Dental Image Data

<u>원 VGG의 구조</u>

- 입력 데이터는 224(가로)*224(세로)*3(RGB)
- 출력은 1,000개(클래스 갯수)

<u>새로 학습할 VGG의 구조</u>

- 입력 데이터는 224(가로)*224(세로)*3(RGB)
- 출력은 3개 (목적에 맞게 변형)

<u>Labeled Data</u>

- 따로 준비하지 않고 keras의 ImageDataGenerator를 사용해서 폴더 이름을 레이블로 사용함(폴더로 클래스 구분)

<u>데이터 증강</u>

- Keras의 ImageDataGenerator에서 기능 제공
- 데이터 부족을 극복하기 위해 위치이동, 회전, 좌우반전 등의 변화를 줌

VGG16.ipynb

VGG16 커스텀 학습

모델 구조 정의

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers

from tensorflow.keras.applications import VGG16

# 모델 불러오기
conv_layers = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
conv_layers.summary()

# Convolution Layer를 학습되지 않도록 고정
for layer in conv_layers.layers:
    layer.trainable = False


# 새로운 모델 생성하기
model = models.Sequential()

# VGG16모델의 Convolution Layer를 추가
model.add(conv_layers)

# 모델의 Fully Connected 부분을 재구성
model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))

# 모델
model.summary()

model.save(vgg16_model_path)
```

CNN의 feature extraction 부분만 취하고
DNN 부분은 재구성
그 과정에서 convolution layer 학습되지 않도록 설정

Include_top 의 의미:
보통은 입력이 밑으로, 출력이 top으로 표현됨
Top 제외한다는 것은 DNN 부분 제외하고 로딩하라는 뜻

GPU_assing.ipynb
weight_initializer.ipynb

Etc

- GPU 사용
- Weight initializer

CNN 기반 기타 알고리즘

ResNet

값을 가져와서
더해버림

- 2015년에 ILSVRC에서 1등한 모델.
- convolution layer와 max pooling layer가 있는 CNN
- 많은 레이어가 있으며weight의 갯수는 약 2500만개
- 모델 파일의 크기는 약 100M
- 2개의 레이어 간에 링크가 있음
- 입력과 출력은 VGG16과 동일

GoogLeNet

- 2014년에 ILSVRC에서 1등한 모델 (Inception-v1)

https://ikkison.tistory.com/86

DenseNet

https://hoya012.github.io/blog/DenseNet-Tutorial-1/

U-Net

의학논문에서 Segmentation에 사용된 예가 있음

https://arxiv.org/pdf/1505.04597.pdf

RNN (Recurrent Neural Network)

RNN의 구조

- 출력 상태를 가지고 있고, 다음 연산 때 그 상태가 재입력되어 사용됨

RNN으로 해결하는 문제의 예

- [h, e, l, l, o]의 학습 예
  - h -> e
  - e -> l
  - l -> l
  - l - > o
- 첫번째 l과 두번째 l의 출력이 다름
- 단점 - 최근 값이 가장 크게 영향을 미치고 거리가 멀어질수록 영향이 희미해짐

LSTM (Long Short Term Memory)

- 최근 것이 가장 크게 영향을 미치고 거리가 멀어질수록 영향이 희미해지는 RNN의 단점을 방지하기 위해 보완된 모델

GAN (Generative Adversarial Network)

GAN – Generative Adversarial Network (상호경쟁 네트워크/생성적 적대 신경망)

- 대표적 비지도 학습. 월 500편의 논문이 발표되고 있음
- 위조범과 위조를 구분해 내려는 자의 경쟁이 GAN의 기본 아이디어

- GAN의 주요 구성요소: generator, discriminator 두개의 DNN을 엮은 구조의 모델
  - Generator (생성자) – 실제 이미지와 비슷한 fake image를 계속해서 생성
  - Discriminator (식별자) – 실제 이미지와 가짜 이미지의 판별을 학습

  → 궁극적인 목표는 실제 이미지와 구별할 수 없는 이미지를 생성해 내는 네트워크를 갖는 것

- 한 네트워크가 다른 네트워크와 겨루는 구조이기 때문에 Adversarial이라는 이름이 붙었음

GAN의 장점

- DNN 학습을 위한 labeled data가 없어도 됨
- DNN의 가장 큰 단점을 해결
- 없었던 이미지를 생성해 냄
- 학습에 사용된 data와 비슷한 data를 생성

- Conditional GAN: 데이터 생성 시 조건을 주어 생성되는 바를 제어할 수 있음

http://research.nvidia.com/sites/default/files/publications/karras2017gan-paper.pdf

GAN의 구조

Z
노이즈
Generator
: 노이즈에서 가짜 이미지를 생성
G
가짜 이미지
Discriminator
: 이미지가 진짜인지 가짜인지 판별
D_gene
가짜 이미지에 대한 판별값
이 부분을 학습시켜 real과 유사한 이미지를 생성해 냄
X
진짜 이미지
Discriminator
D_real
진짜 이미지에 대한 판별값

생성기의 목적

- 생성한 가짜 이미지G에 대한 결과(D_gene)은 1

생성기의 비용함수

- D_gene이 1이면 좋은 결과
- loss_G=log(D_gene)

판별기의 목적

- 진짜 이미지X에 대한 결과(D_real)은 1
- 가짜 이미지G에 대한 결과(D_gene)은 0

판별기의 비용함수

- D_real은 1이면 좋은 결과이고, D_gene은 0이면 좋은 결과
- loss_D=log(D_real)+log(1-D_gene)

loss_G와 loss_D를 교대로 학습하여 발달시킴

loss_D = log(D_real) + log(1-D_gene)

loss_G = log(D_gene)

학습이 완료되면

생성기는 판별기가 구별 못하는 가짜 이미지를 생성

https://tykimos.github.io/2018/10/10/Understanding_Generative_Adversarial_Nets/

https://medium.com/free-code-camp/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394

DCGAN_MNIST.ipynb

GAN의 의미

생성자 학습을 위한 식별자

- Generator의 목적은 Discriminator가 구별하지 못하게 하는것
- Discriminator 없이는 Generator도 학습하지 못함
- Generator의 출력을 Discriminator의 입력으로 사용

의미에 맞는 비용함수를 정의해서 사용하면 DNN은 해당 비용함수를 최소화 하도록 학습

- 판별 알고리즘은 feature를 label에 매핑
- 생성 알고리즘은 이와 반대로 작용
  (주어진 특정 label에서 feature를 예측)
- 판별 알고리즘은 클래스 간의 경계를 학습
- 생성 알고리즘은 각 클래스의 분포를 모델링

https://medium.com/free-code-camp/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394

https://www.youtube.com/watch?v=kSLJriaOumA

GAN 결과 예

http://research.nvidia.com/sites/default/files/publications/karras2017gan-paper.pdf

GAN 결과 예

https://arxiv.org/pdf/1511.06434.pdf

GAN을 사용한 noise 제거 (Inpaining)

Random Noise

Iteration :	150
Learning rate :	0.001
Momentum :	0.9
Lambda :	0.1

GAN을 사용한 inpainting

요약

GAN의 장점: 학습을 위한 레이블 데이터가 없어도 된다.
(DNN의 가장 큰 단점을 커버)

GAN의 결과물: 학습에 사용된 데이터와 비슷한 데이터를 생성한다.
(없었던 것을 생성한다)

Conditional GAN: 생성 시 조건(Condition)을 주어 생성되는 바를 제어한다.

숫자 이미지 좌측 반절을 조건으로 주고
우측 반절을 생성한 예

무수하게 많은 정상 case를 생성

VGG알고리즘 이용 얼굴 인식

실습 예정

face_recognition.ipynb

준비할 사항

Data 준비: '똑같은이름_#' 형식으로 이미지 저장

Data directory 구조

```
face_data/
    images/
        jobs_1.jpg
        jobs_2.jpg
        jobs_3.jpg
        ⋮
    test_images/
        jobs_1.jpg
        jobs_2.jpg
        jobs_3.jpg
        ⋮
```

얼굴 인식 과정

Data 준비
- 이미지 가져오기

얼굴 부분 crop
- DLIB Face Detector 로딩
- Images 폴더 파일들 crop
- Test_images 폴더 파일들 crop

VGG Face Model 다운로드

Define VGG_FACE_MODEL Architecture
- 학습모델 준비
- 모델 정의
- 모델 학습

얼굴인식 실행

VGG_FACE_Model Architecture

```
#Define VGG_FACE_MODEL architecture
model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(224,224, 3)))
model.add(Convolution2D(64, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Convolution2D(4096, (7, 7), activation='relu'))
model.add(Dropout(0.5))
model.add(Convolution2D(4096, (1, 1), activation='relu'))
model.add(Dropout(0.5))
model.add(Convolution2D(2622, (1, 1)))
model.add(Flatten())
model.add(Activation('softmax'))
```

Image Recognition

Image Recognition

- 2가지 작업 수행
  1. 물체의 위치 탐지 (Localization, Detection) - 결과를 박스로 표시
  2. 물체를 분류 (Classification)

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

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

영상 인식 현황

- 2가지 작업이 동시에 필요함 (탐지, 분류)
- 이로 인해 속도 또는 정확도가 낮아짐
- 동시 분류는 20개 정도의 소수 클래스 가능

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

- 분류: 영상내 대상의 class 확인
- 위치 탐지: 영상 내 대상의 위치 확인
- 영상 인식: 위치 탐지와 대상 분류

Image Classification
Classify an image based on the dominant object inside it.
datasets: MNIST, CIFAR, ImageNet

Object Localization
Predict the image region that contains the dominant object. Then image classification can be used to recognize object in the region
datasets: ImageNet

Object Recognition
Localize and classify all objects appearing in the image. This task typically includes: proposing regions then classify the object inside them.
datasets: PASCAL, COCO

Semantic Segmentation
Label each pixel of an image by the object class that it belongs to, such as human, sheep, and grass in the example.
datasets: PASCAL, COCO

Instance Segmentation
Label each pixel of an image by the object class and object instance that it belongs to.
datasets: PASCAL, COCO

Keypoint Detection
Detect locations of a set of predefined keypoints of an object, such as keypoints in a human body, or a human face.
datasets: COCO

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

영상 인식 모델

크게 두 종류로 나뉨

- 영역을 먼저 찾고, 찾은 영역을 분류하는 모델들
  - RCNN, Fast RCNN, Faster RCNN, Mask RCNN
- 영역찾기와 분류를 동시에 하는 모델들
  - YOLO, SSD

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

RCNN (Region-based CNN)

- 영역을 먼저 찾고 해당 영역에 대해 분류

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

Fig. 8 High level diagrams of the leading frameworks for generic object detection. The properties of these methods are summarized in Table 10.

영역찾기와 분류를 동시에 하는 모델들

- 두개의 작업을 동시에 처리
- 딥러닝 적용 이후 90% 초반의 정확도 구현

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

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

성능지표

IoU, mAP, AR 등

IoU (Intersection over Union)

- 정답과 탐지해낸 결과의 영역 일치 정도

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

mAP (mean Average Precision)

- 여러 개의 물체가 탐지된 경우 각 정확도의 평균

AR-Average Recall

- 탐지해야 할 물체 중 정확히 탐지한 개수의 비율의 평균

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

YOLO (You Only Look Once!)

Real-Time Object Detection

KHU

YOLO3

- Yoo Only Look Once 약자. 1회를 강조한 이름
- 영역 탐지와 분류를 동시에 실행
- 빠르다는 것이 가장 큰 장점
- 일반 CNN에 비교하면 정확도는 다소 낮음
- 성능
  - 빠르다(초당20번 정도) - 동영상 처리도 가능한 수준
  - C++로 개발한 Darknet 사용
  - Tensorflow, Keras 사용 안함
  - Keras로 컴파일 해서 사용 가능

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

YOLO의 탐지 방식

영역의 박스와 해당 박스의 분류를 동시에 여러 개 출력하는 네트워크
(오른쪽 예시)모두 98개 (7*7*2)가 제안되고 중복 없이 확신도가 높은 것만 추려 냄

S*S개의 셀 당 B개의 Boundary 박스를 탐지
→ 셀에 중심을 두는 박스를 탐지해서 각 박스별로 분류

Each grid cell predicts a fixed number of boundary boxes. In this example, the yellow grid cell makes two boundary box predictions (blue boxes) to locate where the person is.

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

YOLO의 구조

GoogLeNet을 기반으로 한 CNN을 사용
입력: 일반 이미지
출력: 98개 상자에 대한 정보(위치, 폭, 높이), 클래스, 확신도를 출력

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

YOLOv2 https://youtu.be/VOC3huqHrss

YOLOv3 https://youtu.be/MPU2HistivI

YOLOv4 https://www.youtube.com/watch?v=_JzOFWx1vZg

yolo3_object_dection_and_train_custom_data.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 변경사항을 저장할 수 없음

공유

목차

설치
- tensorlfow 1.x 사용 설정
- 프로젝트 다운로드
- 필요 라이브러리 설치
- 모델 파일 다운로드

탐지
- 대상 이미지 다운로드
- 버그 픽스
- 탐지 실행

학습 데이터 준비
- 학습 데이터셋 다운로드
- 디렉토리 생성

학습
- 학습된 모델파일 다운로드
- 설정 파일
- 학습 실행

섹션

+ 코드 + 텍스트 드라이브로 복사

RAM 디스크 수정 가능

```
!python yolo3_one_file_to_detect_them_all.py -w yolov3.weights -i dog.jpg
```

```
loading weights of convolution #56
loading weights of convolution #57
no convolution #58
loading weights of convolution #59
loading weights of convolution #60
no convolution #61
loading weights of convolution #62
loading weights of convolution #63
loading weights of convolution #64
no convolution #65
loading weights of convolution #66
loading weights of convolution #67
no convolution #68
loading weights of convolution #69
loading weights of convolution #70
no convolution #71
loading weights of convolution #72
loading weights of convolution #73
no convolution #74
loading weights of convolution #75
loading weights of convolution #76
loading weights of convolution #77
loading weights of convolution #78
loading weights of convolution #79
loading weights of convolution #80
loading weights of convolution #81
no convolution #82
no convolution #83
loading weights of convolution #84
no convolution #85
no convolution #86
loading weights of convolution #87
loading weights of convolution #88
loading weights of convolution #89
loading weights of convolution #90
```

→모델로딩 과정 (시간 걸린다)
프로세싱은 금방

```
no convolution #98
loading weights of convolution #99
loading weights of convolution #100
loading weights of convolution #101
loading weights of convolution #102
loading weights of convolution #103
loading weights of convolution #104
loading weights of convolution #105
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-pa

2020-06-24 04:38:40.249875: I tensorflow/stream_executor
2020-06-24 04:38:44.583512: I tensorflow/stream_executor
dog: 99.76491332054138%
dog: 99.8070478439331%
dog: 98.69624376296997%
```

강아지 3마리 위치도 표시

```
!cat raccoon_dataset/annotations/raccoon-100.xml
```

```
<annotation verified="yes">
	<folder>images</folder>
	<filename>raccoon-100.jpg</filename>
	<path>/Users/datitran/Desktop/raccoon/images/raccoon-100.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>960</width>
		<height>576</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>raccoon</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>548</xmin>
			<ymin>10</ymin>
			<xmax>954</xmax>
			<ymax>520</ymax>
		</bndbox>
	</object>
</annotation>
```

영상에서 x,y 좌표 방향은 반대

yolo3_object_dection_and_train_custom_data.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 변경사항을 저장할 수 없음

공유

목차

설치
tensorlfow 1.x 사용 설정
프로젝트 다운로드
필요 라이브러리 설치
모델 파일 다운로드
탐지
대상 이미지 다운로드
버그 픽스
탐지 실행
학습 데이터 준비
학습 데이터셋 다운로드
디렉토리 생성
학습
학습된 모델파일 다운로드
설정 파일
학습 실행
섹션

+ 코드 + 텍스트 드라이브로 복사

RAM 디스크 수정 가능

```
resizing:  384 384
resizing:  416 416
resizing:  288 288
resizing:  352 352
 - 29s - loss: 1.3291 - yolo_layer_1_loss: 0.7187 - yolo_layer_2_loss: 0.5020 - yolo_layer_3_loss: 0.1084

Epoch 00018: loss did not improve from 1.00744

Epoch 00018: ReduceLROnPlateau reducing learning rate to 9.999999939225292e-10.
Epoch 19/100
resizing:  416 416
resizing:  320 320
resizing:  320 320
resizing:  320 320
resizing:  416 416
resizing:  288 288
resizing:  416 416
resizing:  448 448
resizing:  448 448
resizing:  448 448
resizing:  288 288
resizing:  320 320
resizing:  320 320
resizing:  384 384
resizing:  288 288
resizing:  416 416
resizing:  448 448
resizing:  320 320
 - 30s - loss: 1.1222 - yolo_layer_1_loss: 0.7266 - yolo_layer_2_loss: 0.3289 - yolo_layer_3_loss: 0.0667

Epoch 00019: loss did not improve from 1.00744
Epoch 00019: early stopping
raccoon: 1.0000
mAP: 1.0000
```

Keras에서 제공하는 running rate 줄여주는 기능

100 개 epochs 중 19에서 early stopping
Accuracy 확보 됨

```
[ ] !nvidia-smi
```

Object Recognition by SSD

SSD (Single Shot Detector)

- Single Shot Detector. 역시 1회를 강조한 이름
- VGG16 기반
- Fully connected 레이어를 없애고, 각 conv 레이어의 값을 하나로 받는다.
- YOLO는 98개 박스를 출력하는데 비해, 5776개의 박스를 출력
- 구조는 YOLO에 비해 복잡함

KHU

SSD 구조

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

GAN을 사용한 제조설비 이상탐지

임도형, dh-rim@hanmail.net

제조설비 유지

제조 설비 센서 데이터

- 센서 데이터 : 온도, 진동, 압력, 서보모터의 전류값/각도
- 제어 데이터 : PLC, CNC 데이터

제조 설비의 이상 동작

- 물리적 장치의 미지 내적 값이 시간이 지나면 증가
- 특정 한계를 넘으면 외적인 현상이 나타난다.
  - 부러짐. 깨짐
- 이로 인해 장애나 불량품 발생

DL 모델의 적절성에 대한 고려

이상 탐지의 어려움

- 평소와 다른 데이터의 미묘한 변화를 사람이 탐지하기가 무척 어렵다.
- 기존 통계적인 방법이나 기타 ML 방법으로도 어렵다.

제조 설비의 DL 적용 어려움

- 실 제조설비에는 레이블링 데이터(이상 유무)가 거의 없다.
- 있다 하더라도 정상/비정상의 비율이 극단적이다.

Conditional GAN을 사용

데이터 구성

Hint
PLC, CNC 데이터
00:01
00:02.
00:03
센서 데이터
00:01
00:02.
00:03
00:04
00:05
Anchor
Follower

학습 결과

학습된 생성기(G)

hint, anchor는 과거의 데이터
hint, anchor에 부합하는 미래의 follower 생성
과거의 데이터에 대한 정상이라 할 수 있는 follower
GAN의 특징으로 무수한 follower의 생성이 가능하다.

정상 밴드 생성

이상 탐지

평소 데이터 학습

↓

현재 데이터로
미래 데이터 예측

↓

여러 개 예측

↓

중첩하여 밴드 생성

↓

밴드에서 벗어난 것이
비정상

- **정상 data 밖에 없는데**
- **현재의 data를 가지고 미래를 예측**
- **무수하게 많은 예측 가능**

결과

이상 탐지 결과

의미

비싼 레이블링 데이터가 필요하지 않다.
타 제어 데이터도 사용
복수의 센서 데이터가 동시에 사용

References

1. AI 마인드 (마틴 포드 2019. 07. 01 터닝포인트)
   http://aidev.co.kr/book/7875
2. https://www.tiobe.com/tiobe-index/
3. 하용호@kakao
   https://www.slideshare.net/yongho/ss-79607172
4. 인공지능 이해 및 활용사례 온라인 강의 -2020 OSSW 개발자대회 온라인 개론
5. https://www.di.ens.fr/~lelarge/dldiy/slides/lecture_5/index.html#213
6. https://cs231n.github.io/neural-networks-3/
7. https://www.wikipendium.no/TDT4137_Cognitive_Architectures
8. labelImg SW
   https://github.com/tzutalin/labelImg
9. 예제로 배우는 파이썬 프로그래밍
   http://pythonstudy.xyz/

YOLO를 이용한 커스텀 데이터 학습

VOC (Visual Object Classes) 데이터 준비

루트 폴더 아래 2개의 폴더와 1개 파일이 있다.

- Annotations : 레이블링 xml 파일이 있다 (파일명은 임의로 정할 수 있음)
- JPEGImages : 이미지 파일이 있다.
- classes.txt : 클래스 리스트를 담은 파일

<디렉토리 구조>

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

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

<Labeled data 파일의 예>

```
<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 - labelImg

https://github.com/tzutalin/labelImg

사용하기 위해 필요한 SW

Python
Anaconda
(Pip install)
  PyQt5
  lxml
  labelImg

https://github.com/tzutalin/labelImg

Labeled data 파일 예

train_all.txt (VOC 포멧의 Annotations밑의 xml파일의 내용을 이 파일에 담는다.)

```
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
```

Labeled data 포맷

레이블링 툴의 경우 일반적으로 VOC 포멧을 지원한다. 이 파일을 YOLO로 학습하기 위해서 Darknet 포맷으로 변환이 필요함

(파일이름) + (box1정보) + (box2정보) + …

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

파일이름 box1 정보 box2 정보

...path/301.jpg (103,33),(171,100),2 (133,109),(222,181),2

box 좌상단 좌표 box 우하단 좌표 클래스 인덱스

Hands-on Practice

Preparation for Hands-on Class

- **Installation**
  - Python (Python.com)
  - Anaconda
  - Git (https://git-scm.com/downloads)
- **Sign-in**
  - Google.com
  - Github.com

SW Installation

Python, Anaconda, PyQt5, lxml, labelImg

Anaconda

1. 아나콘다(Anaconda)란?
아나콘다(Anaconda)는 수학과 과학 분야에서 사용되는 여러 패키지들을 묶어 놓은 파이썬 배포판으로서 SciPy, Numpy, Matplotlib, Pandas 등을 비롯한 많은 패키지들을 포함하고 있다. Anaconda는 특히 최근에 데이타 사이언스와 머신 러닝 분야에서 파이썬을 사용하기 위해 기본적으로 설치하는 배포판이 되었다.
Anaconda를 설치하기 위해서는 https://www.anaconda.com/ 웹사이트에서 자신의 OS에 맞는 프로그램을 다운받아 설치하면 된다. 통상 Python 3.x 버전을 선택한다.
2. Anaconda 구성요소
Anaconda는 아래 그림과 같이 크게 4 부분으로 나뉘는데, 이 중 Data Science 라이브러리와 Anaconda Navigator를 주목할 만하다. Data Science 라이브러리는 (a) Jupyter와 같은 IDE 개발도구, (b) Numpy, SciPy 같은 과학 분석용 라이브러리, (c) Matplotlib 같은 데이타 시각화 (Data Visualization) 라이브러리, (d) TensorFlow 같은 머신 러닝(Machine Learning) 라이브러리 등을 포함하고 있다.

Anaconda Navigator는 UI 클라이언트로서 하부 컴포넌트를 쉽게 사용하도록 한 데스크탑 포털 기능을 담당한다. 예를 들어, Jupyter나 Spyder 같은 개발도구를 이곳에서 Launch할 수 있다.

Install Python

https://www.python.org/downloads/release/python-385/

Make sure the check box selected for adding to system PATH

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e2f52bcf531c8cc94732c0b6ff933ff0	24149103	SIG
XZ compressed source tarball	Source release		35b5a3d0254c1c59be9736373d429db7	18019640	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	2f8a736eeb307a27f1998cfd07f22440	30238024	SIG
Windows help file	Windows		3079d9cf19ac09d7b3e5eb3fb05581c4	8528031	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	73bd7aab047b81f83e473efb5d5652a0	8168581	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	0ba2e9ca29b719da6e0b81f7f33f08f6	27864320	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	eeab52a08398a009c90189248ff43dac	1364128	SIG
Windows x86 embeddable zip file	Windows		bc354669bffd81a4ca14f06817222e50	7305731	SIG
Windows x86 executable installer	Windows		959873b37b74c1508428596b7f9df151	26777232	SIG
Windows x86 web-based installer	Windows		c813e6671f334a269e669d913b1f9b0d	1328184	SIG

Install Anaconda

https://www.anaconda.com/

Pip install

1. pip 패키지 관리자

pip은 파이썬 패키지를 설치하고 관리하는 패키지 관리자(Package Manager)이다. pip은 "Pip Installs Packages"의 약자로서 재귀적인 약어이다.
pip은 Python 2.7.9+와 Python 3.4+에서 디폴트로 설치되어 있다. 만약 pip이 시스템에 설치되지 않은 경우는 다음과 같이 설치할 수 있다.

curl https://bootstrap.pypa.io/get-pip.py | python

2. pip 패키지 설치와 제거

pip을 이용하면 패키지를 쉽게 설치할 수 있다. pip으로 설치할 수 패키지들의 목록은 [Python Package Index (PyPI)]에서 찾아 볼 수 있다. 다음은 패키지를 설치하는 pip install 명령이다.

pip install 패키지명 ex) $ pip install django $ pip install numpy $ pip install matplotlib

설치된 패키지를 Uninstall하기 위해서는 아래와 같은 pip uninstall 명령을 사용한다.

pip uninstall 패키지명 ex) $ pip uninstall django

현재 pip으로 설치된 패키지들의 리스트를 보기 위해서는 pip list 명령을 사용한다.

$ pip list certifi (2017.7.27.1) chardet (3.0.4) cytoolz (0.8.2) Django (1.11.2) ethereum-abi-utils (0.4.3) ethereum-utils (0.5.1) idna (2.6) pip (9.0.1) pkg-resources (0.0.0) pysha3 (1.0.2) pytz (2017.2) requests (2.18.4) setuptools (20.7.0) toolz (0.8.2) urllib3 (1.22)

이미 설치된 패키지를 최신 버전으로 업그레이드하기 위해서는 아래와 같이 --upgrade 옵션을 지정한다.

$ pip install django --upgrade

Pip install

3. pip: requirements 파일 생성과 사용

pip을 이용하면 일련의 패키지 그룹을 묶어서 설치할 수도 있다. 즉, requirements.txt 파일에 설치할 패키지 리스트를 정의한 후, 아래와 같은 명령으로 전체를 한꺼번에 설치할 수 있다. 실제 "pip install 패지키명" 을 사용하여 패키지를 하나 하나 설치하는 것보다, requirements.txt를 만들어 패키지 리스트를 이 파일에서 관리하고 이 파일을 통해 pip 설치 수행하는 것이 더 좋은 방법이다.
$ pip install -r requirements.txt
하지만 경우에 따라 requirements.txt를 미리 만들지 않고, 개발서버의 가상환경에 여러 패키지를 이미 설치했을 수도 있다. 이런 경우는 아래와 같은 pip freeze 명령을 사용하여 requirements.txt 파일을 만들 수 있다. 이러한 기능은 개발서버의 가상환경을 Production 가상환경으로 Deployment 하는데 유용하게 쓰일 수 있다.
$ pip freeze > requirements.txt

4. 윈도우즈에서의 pip 사용

윈도우즈에서 pip 을 사용하기 위해서는 python 설치 폴더 밑의 Scripts 서브디렉토리에서 pip을 실행한다. 예를 들어, 아래 그림은 C:\Python3.5 폴더에 Python 3.5 가 설치되어 있을때, Scripts/pip 명령을 사용하여 numpy 패키지를 설치하는 예이다.

```
C:₩Python35₩Scripts>pip install numpy
Collecting numpy
  Downloading numpy-1.12.1-cp35-none-win32.whl (6.7MB)
    100% |################################| 6.7MB 61kB/s
Installing collected packages: numpy
Successfully installed numpy-1.12.1
You are using pip version 7.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:₩Python35₩Scripts>_
```

Install PyQt5

Search → cmd
"pip install PyQt5"

명령 프롬프트

```
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:₩Users₩user>pip install PyQt5
Collecting PyQt5
  Downloading PyQt5-5.15.0-5.15.0-cp35.cp36.cp37.cp38-none-win_amd64.whl (64.5 MB)
     |████████████████████████████████| 64.5 MB 6.4 MB/s
Collecting PyQt5-sip<13,>=12.8
  Downloading PyQt5_sip-12.8.0-cp38-cp38-win_amd64.whl (63 kB)
     |████████████████████████████████| 63 kB 4.8 MB/s
Installing collected packages: PyQt5-sip, PyQt5
Successfully installed PyQt5-5.15.0 PyQt5-sip-12.8.0

C:₩Users₩user>
```

conda install pyqt=5

Anaconda Prompt (Anaconda3)

```
(base) C:₩Users₩user>conda install pyqt=5
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:₩ProgramData₩Anaconda3

  added / updated specs:
    - pyqt=5


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.8.3                |           py37_0        2.8 MB
    ------------------------------------------------------------
                                           Total:        2.8 MB

The following packages will be UPDATED:

  conda                                     4.8.2-py37_0 --> 4.8.3-py37_0


Proceed ([y]/n)? y


Downloading and Extracting Packages
conda-4.8.3          | 2.8 MB    | ################################################################################ | 100%
Preparing transaction: done
Verifying transaction: failed

EnvironmentNotWritableError: The current user does not have write permissions to the target environment.
  environment location: C:₩ProgramData₩Anaconda3



(base) C:₩Users₩user>
```

Installing lxml

Pip install lxml

명령 프롬프트

```
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\user>pip install lxml
Collecting lxml
  Downloading lxml-4.5.2-cp38-cp38-win_amd64.whl (3.5 MB)
     |████████████████████████████████| 3.5 MB 1.7 MB/s
Installing collected packages: lxml
Successfully installed lxml-4.5.2

C:\Users\user>
```

Installing lxml

Contents

- Where to get it
- Requirements
- Installation
- Building lxml from dev sources
- Using lxml with python-libxml2
- Source builds on MS Windows
- Source builds on MacOS-X

» Where to get it

lxml is generally distributed through **PyPI**.

Most **Linux** platforms come with some version of lxml readily packaged, usually named `python-lxml` for the Python 2.x version and `python3-lxml` for Python 3.x. If you can use that version, the quickest way to install lxml is to use the system package manager, e.g. `apt-get` on Debian/Ubuntu:

pip install labellmg

```
(base) C:₩Users₩user>pip3 install labellmg
Collecting labellmg
  Downloading labellmg-1.8.3-py2.py3-none-any.whl (248 kB)
     |████████████████████████████████| 248 kB 930 kB/s
Requirement already satisfied: lxml in c:₩users₩user₩appdata₩local₩programs₩python₩python38₩lib₩site-packages (from labe
llmg) (4.5.2)
Requirement already satisfied: pyqt5 in c:₩users₩user₩appdata₩local₩programs₩python₩python38₩lib₩site-packages (from lab
ellmg) (5.15.0)
Requirement already satisfied: PyQt5-sip<13,>=12.8 in c:₩users₩user₩appdata₩local₩programs₩python₩python38₩lib₩site-pack
ages (from pyqt5->labellmg) (12.8.0)
Installing collected packages: labellmg
Successfully installed labellmg-1.8.3

(base) C:₩Users₩user>
```

labelimg C:₩Users₩user₩Desktop₩rbMSC₩train₩P1_eunah₩0114.jpg
File
Edit
View
Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
PascalVOC
Xml 파일로 저장하기 위해 선택 상태 변경
Create₩nRectBox
Duplicate₩nRectBox
Delete₩nRectBox
Zoom In
94 %
Zoom Out
Fit Window
100μm
224px
Box Labels
Edit Label
difficult
Use default label
medium-polygonal
dividing
File List
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0093.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0094.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0095.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0096.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0097.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0098.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0099.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0100.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0101.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0102.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0103.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0104.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0105.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0106.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0107.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0108.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0109.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0110.jpg
iser₩Desktop₩rbMSC₩train₩P1_eunah₩0111.jpg
Click & drag to move point
X: 880; Y: 304

알파고 학습 이해하기

(주)엑셈 / 임도형

신경망(Neural Network)

입출력의 함수를 근사화 할 수 있다.

입력과 출력을 반복적으로 주고, 함수를 근사화 해가는 과정

https://www.slideshare.net/yonghakim900/ss-60252533

틱텍토

모든 경우의 수 탐색가능

9 x 8 x… 2

http://blog.naver.com/PostView.nhn?blogId=best999&logNo=220680107733&parentCategoryNo=&categoryNo=&viewDate=&isShowPopularPosts=false&from=postView

체스

1996년 딥블루
경우의 수는 탐색할 만 하다.

https://www.youtube.com/watch?v=SUbqykXVx0A&feature=youtu.be

바둑

모든 경우의 수 탐색 불가.
361X 360 x … > 우주 원자 수

https://www.youtube.com/watch?v=SUbqykXVx0A&feature=youtu.be

Alphago를 요약하면

2개의 네트웍 + MCTS

Monte Carlos Tree Search 가 사용됨

Policy Network

Value Network

Policy Network

현재의 판에 대한 다음 수를 학습
프로기사의 기보를 사용.

19*19*13 이 입력

https://www.slideshare.net/ckmarkohchang/alphago-in-depth

Policy Network

CNN: channel이 13~12개 인것을 입력으로 하는

https://www.slideshare.net/ckmarkohchang/alphago-in-depth

Policy Network - 강화학습

프로기사들의 기보를 가지고 학습

학습이 된 Policy Network을 가지고 스스로 대국을 두게함.
그 결과를 가지고 네트웍을 업데이트.

바둑 프로그램 Pachi를 상대로 85% 승율

Policy Network의 역할

- 검색 폭을 줄인다.
- 출력된 수만을 고려한다.

https://gogameguru.com/i/2016/03/deepmind-mastering-go.pdf

Policy Network의 역할

http://dongguo.me/blog/2017/10/29/alphago-zero/

Policy Network의 역할

모든 가능한 수를 다 검색하지 않아도 된다.

http://dongguo.me/blog/2017/10/29/alphago-zero/

Value Network

현재의 판을 가지고 유리/불리를 학습
최종 결과를 출력으로 학습. 1:승리, 0: 패배

https://www.slideshare.net/ckmarkohchang/alphago-in-depth

Value Network

Policy Network으로 스스로 플레이한 기보 데이터를 사용

VS

Value Network

현재 판을 보고 유리/불리를 출력

Value Network의 역할 - 검색깊이를 줄인다

Value Network의 역할

양쪽의 수를 비교하고 유리한 개수를 셈

http://dongguo.me/blog/2017/10/29/alphago-zero/

검색의 폭을 줄이고 검색의 깊이를 확보
Tree 검색이 가능해지는 방법

MCTS

Monte-Carlose Tree Search
트리 검색 방법
이전 최강 프로그램 Pachi이 사용
Alphago의 경우 끝까지 가지 않고, 특정 깊이에서 Value Network의 값을 사용.

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

정리하면

Policy Network을 사용하여 검색 폭을 줄이고
MCTS를 사용하여 최선의 가지를 찾고
Value Network을 사용 하여 검색 깊이를 줄이고

- 알파고 - 중국식 기보로 학습 (그래서 이세돌씨가 중국식으로 두었음)
- Deep learning은 좁고 깊게 잘하는 방법
- 알파고는 narrow AI - Artificial General Intelligence (일반지능)의 구현은 아님

https://gogameguru.com/i/2016/03/deepmind-mastering-go.pdf

Reference

- Mastering the Game of Go with Deep Neural Networks and Tree Search https://gogameguru.com/i/2016/03/deepmind-mastering-go.pdf
- Google DeepMind's AlphaGo: How it works : https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works/
- Alphago in Depth :https://www.slideshare.net/ckmarkohchang/alphago-in-depth

알파고 이후 이야기

이후버전: 알파제로

- 데이터 없이 학습
- 알파제로끼리 대국
- 10% 추려서 학습 시킴
- 이과정을 반복

알파제로에 대해서는 발표된 내용이 많지 않음
기술 요약

- CNN
- MCTS

의학논문 리뷰

VGG16 사용 영상판독 실습

A deep learning model for the detection of both advanced and early glaucoma using fundus photography

PLoS One, 13 (11), e0207982 2018 Nov 27 eCollection 2018

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0207982

Jin Mo Ahn 1, Sangsoo Kim 1, Kwang-Sung Ahn 2, Sung-Hoon Cho 2, Kwan Bok Lee 3, Ungsoo Samuel Kim 3
1 Department of Bioinformatics and Life Science, Soongsil University, Seoul, Korea
2 Functional Genome Institute, PDXen Biosystems Inc., Seoul, Korea
3 Kim's Eye Hospital, Seoul, Korea

KHU

논문 개요

녹내장 여부 분류
망막 영상 데이터 사용
GoogleLenet 전이학습, CNN 두가지 사용
정확도 87.7%, AUROC 0.95

여기서의 결과: 전이학습이 더 결과가 안 좋았음
대상의 세기가 달라서 그런 듯
대상에 맞게 잘 학습된 것을 사용해야 함

기존 방법 한계

어려운 검사들

- optical coherence tomography (OCT)
- red-free retinal-nerve-fiber-layer (RNFL) photography
- visual field tests

데이터 수집과 전처리

총 1,542 영상 데이터
224 x 224로 5곳으로 짤라서 사용함.
좌우 반전하여 영상 1개당 10개로 증강시킴

	Advanced Glaucoma	Early Glaucoma	Normal	Total
Entire Data	467	289	786	1,542
Training Data	228	141	385	754
Validation Data	98	61	165	324
Test Data	141	87	236	464

2명의 전문가에 의해 레이블링 됨

모델 구조 1

GoogleNet Incetption V3를 기반함 CNN의 한 종류

모델 구조 2

일반적인 CNN

입력과 출력

입력

224 x 224의 영상

출력

3개 카테고리. 초기 녹내장, 진행된 녹내장, 정상

기존 방법과 결과 비교

LR(Logistic Regression)과 비교

- LR의 경우 77.2%
- DL의 경우 87.7%

결과

정확도 82.9%, AUROC 0.940

	Advanced and Early Glaucoma	Advanced Glaucoma only	Early Glaucoma only
Training Data	82.9%	99.7%	83.7%
Test Data	77.2%	98.6%	73%

고찰

여기서의 결과: 전이학습이 더 결과가 안좋았음

Discussion

- GoogleLenet 전이학습 시에 일반 CNN보다 성능이 낮다. 대상 데이터의 특성에 기인한 듯.
- trial 정도의 수준.

기타

- 전이학습을 사용함.
  - 사용 안했을 경우 정확도 87.9%, AUROC 0.94
  - 전이학습의 경우 정확도 84.5%, AUROC 0.93
- 학습 데이터에 대한 정확도와 AUROC를 99.7%, 0.99라고 명시함. 의미 없다.
- LR을 위해 영상 데이터를 1차원 데이터로 변환함.

기타

사용된 데이터를 공유하고 있다.

- https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/1YRRAC
- 110 M

Data 폴더 구조

```
glaucoma/
        train/
                normal/ 1~394
                early/ 1~144
                advanced/ 1~233
        test/
                normal/ 395~788
                early/ 144~289
                advanced/ 234~467
```

Glaucoma_classification.ipynb

실습

glaucoma_classification.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

https://olc.kr/wiset/

공유

파일

- ..
- drive
- glaucoma
- sample_data
- test
- train
- glaucoma.zip

디스크 36.40 GB 사용 가능

+ 코드 + 텍스트

RAM 디스크 수정 가능

```
[15] %tensorflow_version 1.x
```

드라이브 마운트 하고

```
[16] !cp '/content/drive/My Drive/tmp/glaucoma.zip' ./ #마운트 된 파일을 현재 디렉토리로 가져오는(./)
```

파일의 경로복사해서 가져오기

```
[17] !unzip glaucoma.zip
```

```
extracting: test/normal/671.png
extracting: test/normal/672.png
extracting: test/normal/673.png
extracting: test/normal/674.png
extracting: test/normal/675.png
extracting: test/normal/676.png
extracting: test/normal/677.png
extracting: test/normal/678.png
extracting: test/normal/679.png
extracting: test/normal/680.png
 inflating: test/normal/681.png
extracting: test/normal/682.png
extracting: test/normal/683.png
extracting: test/normal/684.png
extracting: test/normal/685.png
extracting: test/normal/686.png
extracting: test/normal/687.png
extracting: test/normal/688.png
extracting: test/normal/689.png
extracting: test/normal/690.png
extracting: test/normal/691.png
extracting: test/normal/692.png
extracting: test/normal/693.png
extracting: test/normal/694.png
extracting: test/normal/695.png
extracting: test/normal/696.png
extracting: test/normal/697.png
extracting: test/normal/698.png
extracting: test/normal/699.png
 inflating: test/normal/700.png
 inflating: test/normal/701.png
extracting: test/normal/702.png
extracting: test/normal/703.png
```

glaucoma_classification.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

파일

+ 코드 + 텍스트

RAM 디스크 수정 가능

- ..
- drive
  - My Drive
- glaucoma
- sample_data
- test
- train
- glaucoma.zip

디스크 36.40 GB 사용 가능

[18]
```
!ls -al #가져온 파일이 현재 디렉토리에 있는지 확인하려고..  없는것 같으니 디렉토리를 다시 만들어줌
```

```
total 121144
drwxr-xr-x 1 root root      4096 Jun 26 06:16 .
drwxr-xr-x 1 root root      4096 Jun 26 05:22 ..
drwxr-xr-x 1 root root      4096 Jun 19 16:15 .config
drwx------ 4 root root      4096 Jun 26 05:23 drive
drwxr-xr-x 4 root root      4096 Jun 26 05:28 glaucoma
-rw------- 1 root root 124016091 Jun 26 06:16 glaucoma.zip
drwxr-xr-x 1 root root      4096 Jun 17 16:18 sample_data
drwxr-xr-x 5 root root      4096 Jun 26 06:16 test
drwxr-xr-x 5 root root      4096 Jun 26 06:16 train
```

[20]
```
!mkdir glaucoma
!mv train glaucoma/
!mv test glaucoma/
```

```
mkdir: cannot create directory  'glaucoma' : File exists
mv: cannot move 'train' to 'glaucoma/train': Directory not empty
mv: cannot move 'test' to 'glaucoma/test': Directory not empty
```

[21]
```
!ls -al
```

```
total 121144
drwxr-xr-x 1 root root      4096 Jun 26 06:16 .
drwxr-xr-x 1 root root      4096 Jun 26 05:22 ..
drwxr-xr-x 1 root root      4096 Jun 19 16:15 .config
drwx------ 4 root root      4096 Jun 26 05:23 drive
drwxr-xr-x 4 root root      4096 Jun 26 05:28 glaucoma
-rw------- 1 root root 124016091 Jun 26 06:16 glaucoma.zip
drwxr-xr-x 1 root root      4096 Jun 17 16:18 sample_data
drwxr-xr-x 5 root root      4096 Jun 26 06:16 test
drwxr-xr-x 5 root root      4096 Jun 26 06:16 train
```

3일차 VGG16_classification_and_cumtom_data_training.ipynb.. 코랩에서 오픈하고 '디렉토리 구조의 데이터의 학습 최소코드 부분 전체 copy해
음

VGG16 최소코드 전체 copy

```
import os

import matplotlib.pyplot as plt
import numpy as np

from tensorflow.keras import optimizers
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array

from tensorflow.keras.applications import vgg16
from tensorflow.keras.applications import VGG16


conv_layers = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
conv_layers.trainable = False

model = models.Sequential()

model.add(conv_layers)
model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
#model.add(layers.Dense(10, activation='softmax'))
model.add(layers.Dense(3, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer="RMSprop", metrics=['acc'])
```

```python
data_aug_generator = ImageDataGenerator(
      rotation_range=10,
      width_shift_range=0.1,
      height_shift_range=0.1,
      horizontal_flip=True,
      vertical_flip=False,
      preprocessing_function=vgg16.preprocess_input
)

data_no_aug_generator = ImageDataGenerator(
      preprocessing_function=vgg16.preprocess_input
)

train_data_generator = data_aug_generator.flow_from_directory(
      "glaucoma/train",
      target_size=(224,224),
      batch_size=64,
      class_mode='sparse'
)

test_data_generator = data_no_aug_generator.flow_from_directory(
      "glaucoma/test",
      target_size=(224,224),
      class_mode='sparse'
)


model.fit_generator(
      train_data_generator,
      validation_data=test_data_generator,
      validation_steps=5,
      steps_per_epoch=train_data_generator.samples/64,
      epochs=10
)
```

```python
y_ = model.predict_generator(
      test_data_generator,
      steps=test_data_generator.samples/64
)

custom_labels = list(test_data_generator.class_indices.keys())
predicted = np.argmax(y_, axis=1)
print(predicted[0], custom_labels[predicted[0]])
```

결과

```
Found 771 images belonging to 3 classes.
Found 773 images belonging to 3 classes.
Epoch 1/10
12/12 [==============================>.] - ETA: 0s - loss: 105.8674 - acc: 0.5530Epoch 1/10
13/12 [==============================] - 12s 950ms/step - loss: 98.6731 - acc: 0.5720 - val_loss: 7.4250 - val_acc: 0.4938
Epoch 2/10
12/12 [==============================>.] - ETA: 0s - loss: 7.3394 - acc: 0.6973Epoch 1/10
13/12 [==============================] - 11s 848ms/step - loss: 7.0260 - acc: 0.7056 - val_loss: 3.7434 - val_acc: 0.7250
Epoch 3/10
12/12 [==============================>.] - ETA: 0s - loss: 2.5690 - acc: 0.7581Epoch 1/10
13/12 [==============================] - 11s 855ms/step - loss: 2.5615 - acc: 0.7588 - val_loss: 1.3485 - val_acc: 0.7125
Epoch 4/10
12/12 [==============================>.] - ETA: 0s - loss: 1.6250 - acc: 0.7610Epoch 1/10
13/12 [==============================] - 11s 851ms/step - loss: 1.6019 - acc: 0.7575 - val_loss: 1.1509 - val_acc: 0.6625
Epoch 5/10
12/12 [==============================>.] - ETA: 0s - loss: 0.9085 - acc: 0.7440Epoch 1/10
13/12 [==============================] - 11s 854ms/step - loss: 0.8892 - acc: 0.7484 - val_loss: 1.2355 - val_acc: 0.6000
Epoch 6/10
12/12 [==============================>.] - ETA: 0s - loss: 0.8314 - acc: 0.7638Epoch 1/10
13/12 [==============================] - 11s 862ms/step - loss: 0.8565 - acc: 0.7549 - val_loss: 1.6468 - val_acc: 0.5250
Epoch 7/10
12/12 [==============================>.] - ETA: 0s - loss: 1.3159 - acc: 0.7185Epoch 1/10
13/12 [==============================] - 11s 853ms/step - loss: 1.2813 - acc: 0.7250 - val_loss: 2.6224 - val_acc: 0.5500
Epoch 8/10
11/12 [==========================>...] - ETA: 0s - loss: 0.6482 - acc: 0.7841Epoch 1/10
13/12 [==============================] - 11s 863ms/step - loss: 0.6860 - acc: 0.7847 - val_loss: 1.1516 - val_acc: 0.7125
Epoch 9/10
12/12 [==============================>.] - ETA: 0s - loss: 0.9619 - acc: 0.7567Epoch 1/10
13/12 [==============================] - 11s 851ms/step - loss: 0.9501 - acc: 0.7575 - val_loss: 0.9998 - val_acc: 0.7000
Epoch 10/10
12/12 [==============================>.] - ETA: 0s - loss: 0.6909 - acc: 0.7963Epoch 1/10
13/12 [==============================] - 11s 855ms/step - loss: 0.6982 - acc: 0.7964 - val_loss: 1.1147 - val_acc: 0.6062
1 early
```