SUN SOFTWARE MAESTRO

Python Programming Language

1991년 Guido Van Rossum이 발표

파이썬은 배우기 쉽고, 강력한 프로그래밍 언어

효율적인 고수준 데이터 구조

간단하지만 효과적인 객체 지향 프로그래밍 접근

우아한 문법과 동적 타이핑, 인터프리팅 환경

파이썬은 다양한 분야, 다양한 플랫폼에서 사용

Python Programming Language

1989년 크리스마스에 딱히 할 일이 없어 개발

BBC TV 프로그램 "Monty Python's Flying Circus"

에서 유래

Python은 그리스 신화에 나오는 거대한 뱀

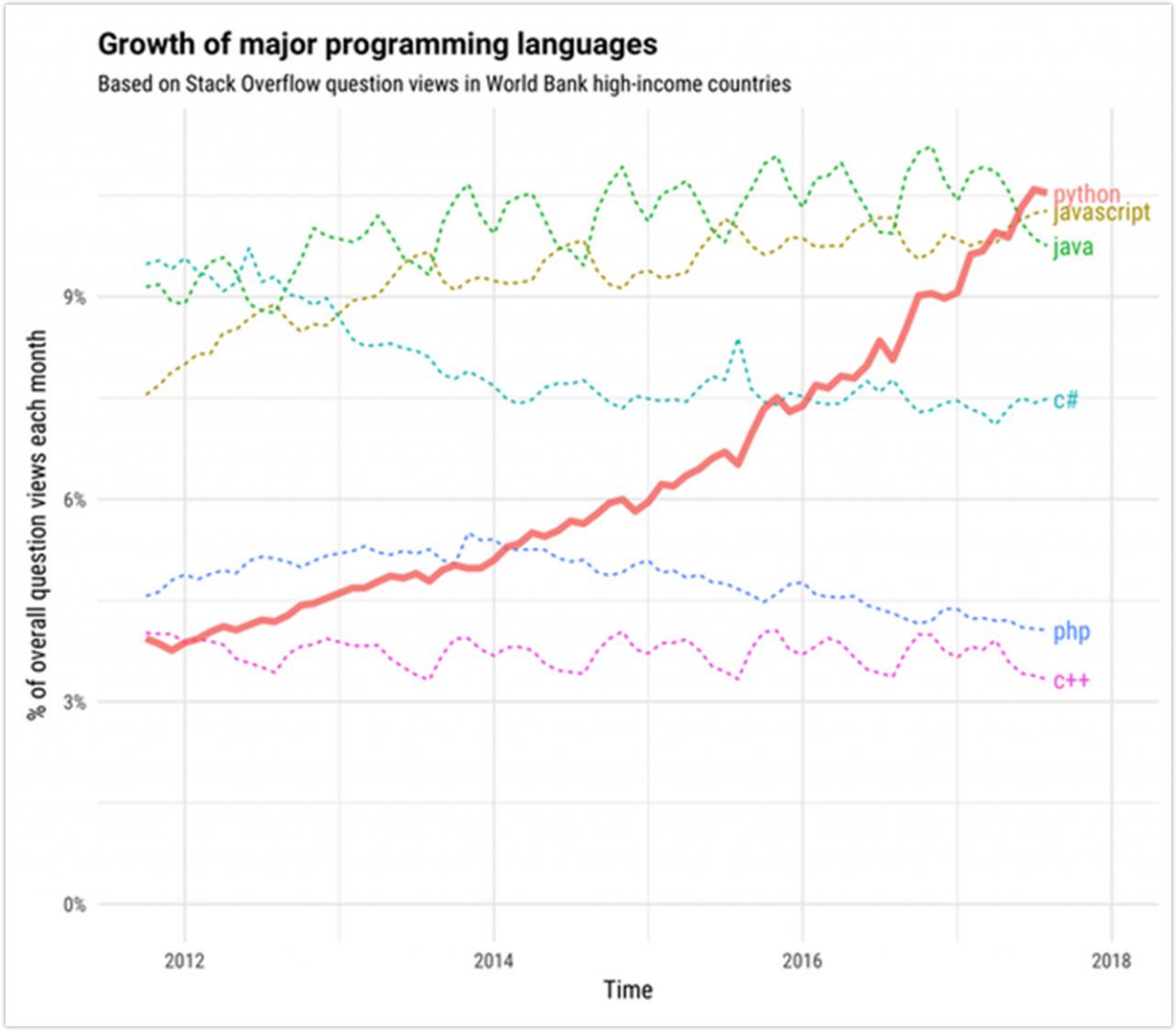
Guido는 구글에 근무한 적이 있으며,

현재는 DropBox에서 근무중

Python의 특징

- 단순하고 최소화된 언어, 쉬운 문법 체계, 가독성, 간결한 코드
- FLOSS (Free/Libre and Open Source Software, 자유/오픈 소스 소프트웨어)
- 메모리 관리 등이 불필요한 고수준 언어
- 플랫폼 독립적, 객체지향, 인터프리터 언어
- 유니코드, 동적 타이핑 지원 언어
- 방대한 규모의 라이브러리

Why Python



출처 : https://www.quora.com/What-are-the-programming-languages-most-popular-in-2019

Why Python

출처 : https://data-flair.training/blogs/why-should-i-learn-python/

Python 개발환경

- Python : https://www.python.org/downloads/

  Python 기본 개발환경

- Anaconda : https://www.anaconda.com/distribution/

  Python과 여러 패키지들을 쉽게 설치/관리할 수 있는 배포판
  Jupyter notebook을 이용한 대화형 개발/분석 도구 제공

- Pycharm : https://www.jetbrains.com/pycharm

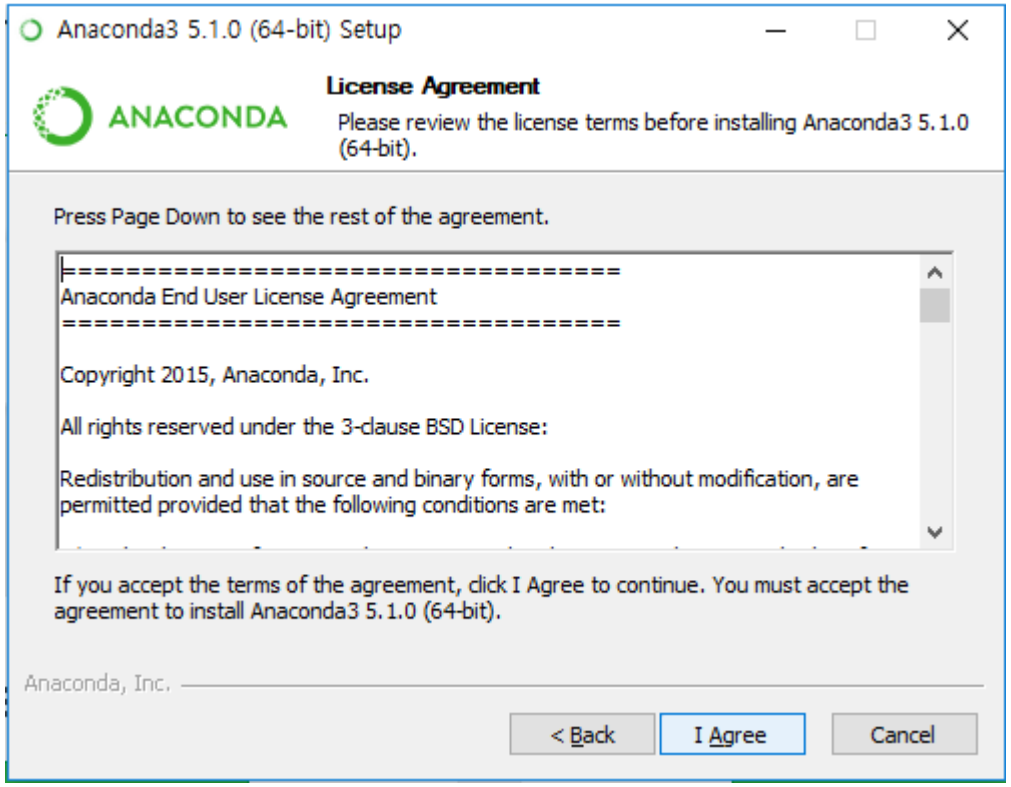
  디버깅 등의 기능을 제공하는 통합 IDE 개발환경
  Community 버전은 Free, Open Source로 제공

Python 개발환경 – Anaconda 설치

설치파일 다운로드 및 실행

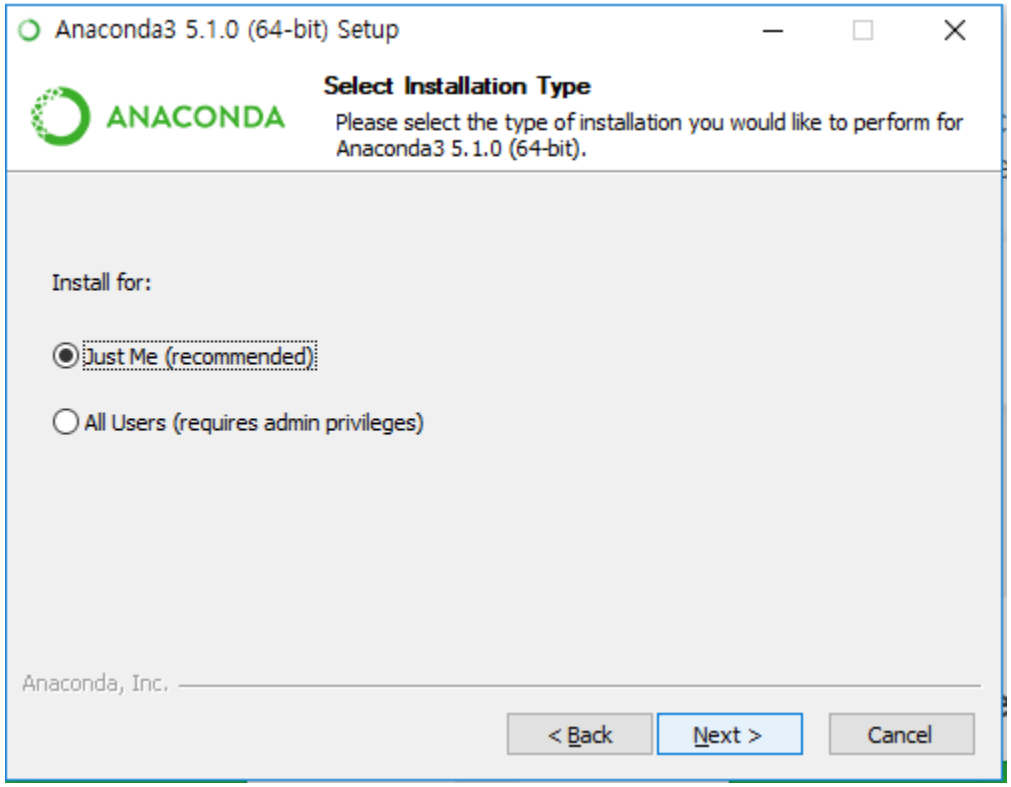
Python 개발환경 – Anaconda 설치

라이센트 동의 – I Agree 선택



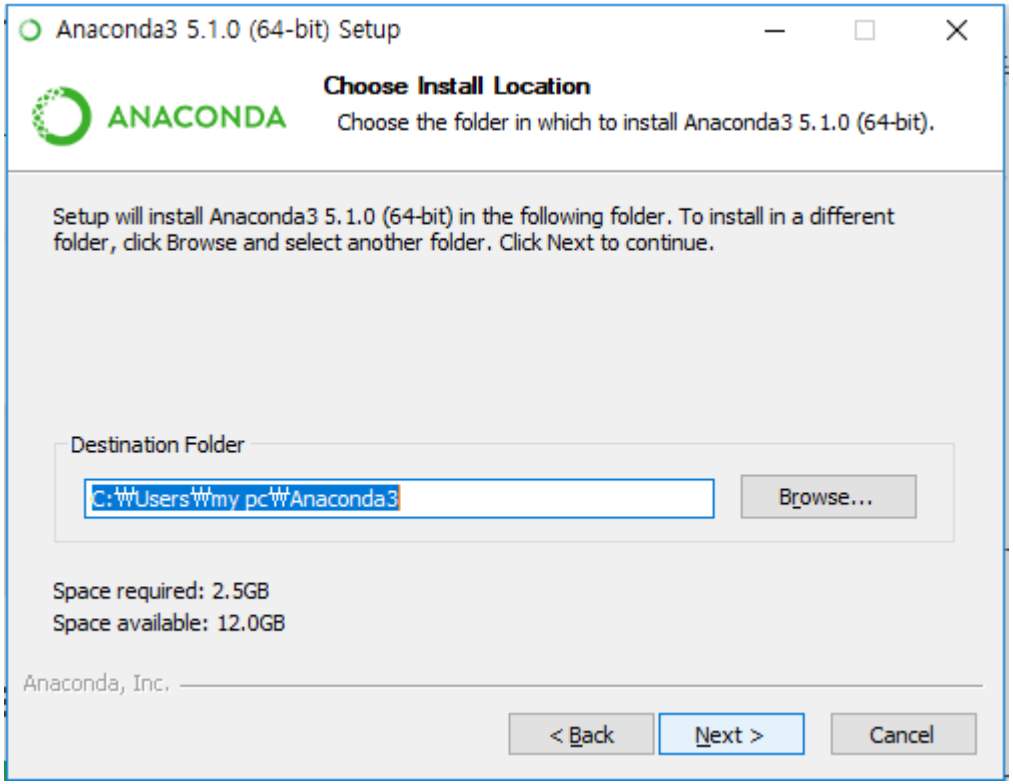
Python 개발환경 – Anaconda 설치

설치 유형 선택 – Just Me 선택 및 Next



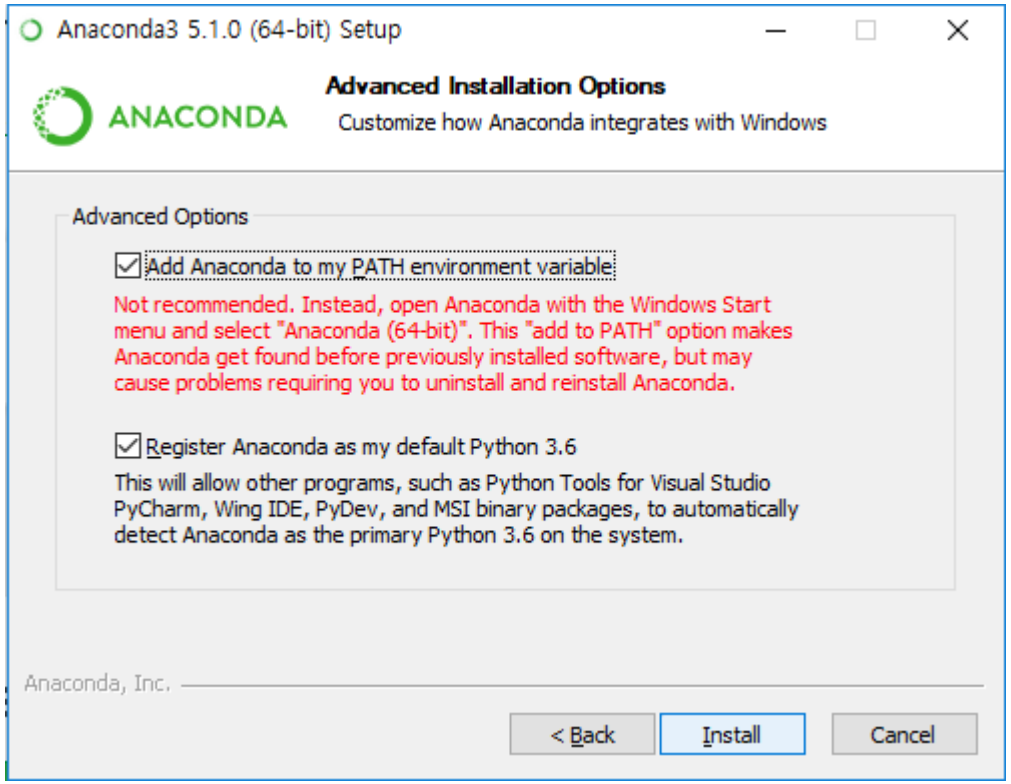
Python 개발환경 – Anaconda 설치

설치 위치 선택 – Next



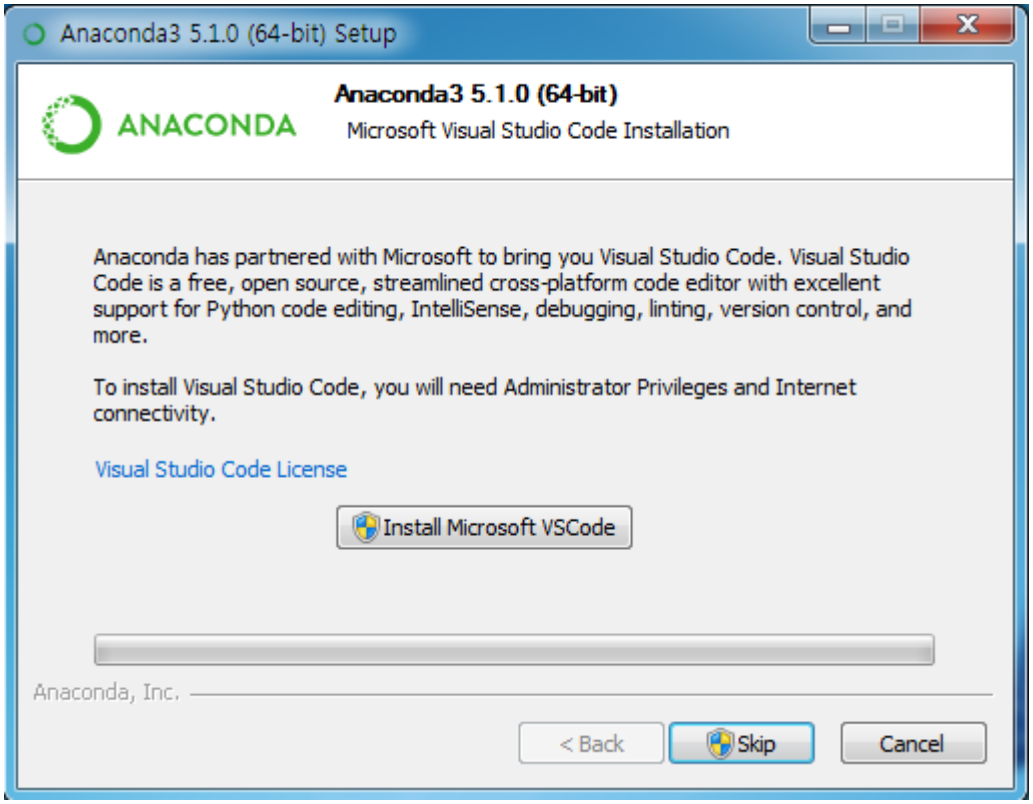
Python 개발환경 – Anaconda 설치

고급 설치 옵션 – PATH/default Python version – check 및 Install



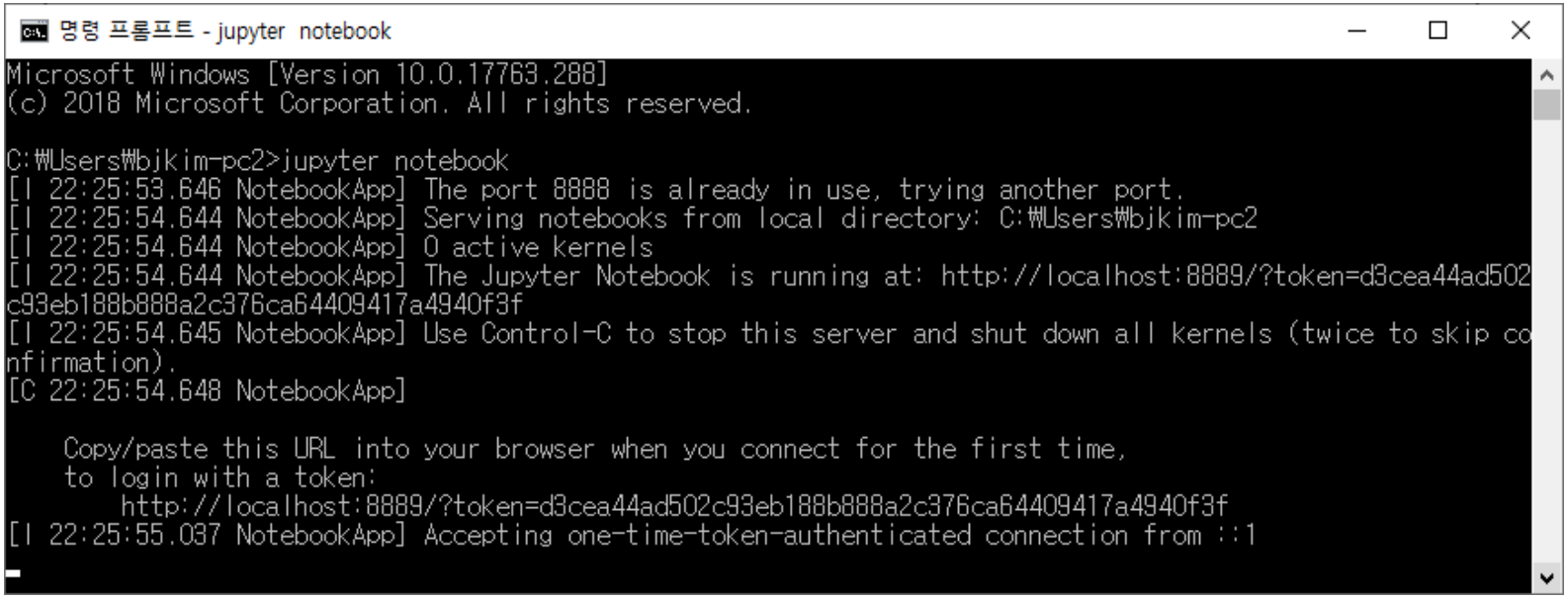
Python 개발환경 – Anaconda 설치

Install Microsoft VSCode – Skip

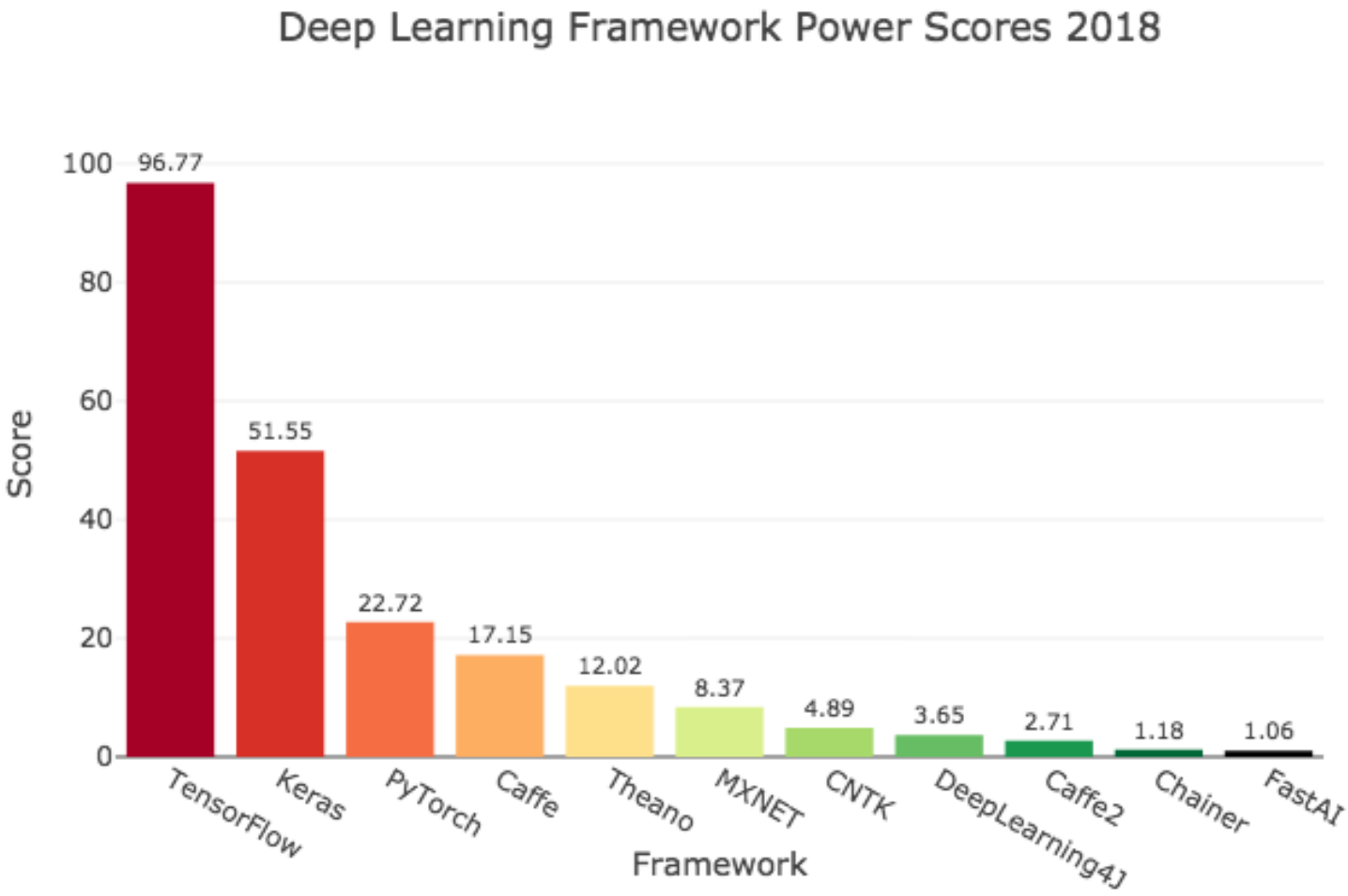


Jupyter Notebook

cmd 창에서 jupyter notebook 입력



딥러닝 framework



출처 : https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a

딥러닝 framework – Tensorflow

- 가장 인기있는 딥러닝 라이브러리 중 하나로 머신러닝과 딥 뉴럴 네트워크 연구 등을 위해 개발
- Google Brain 팀에서 머신러닝 인텔리전스 연구과정에서 개발하여 2015년 오픈소스로 공개
- Python 기반 라이브러리로 C++ 및 R과 같은 다른 언어도 지원
- CPU, GPU 및 TPU 환경과 데스크톱 및 모바일에서도 사용 가능
- 딥러닝 모델을 직접 작성하거나 Keras와 같은 rapper 라이브러리를 사용하여 작성
- 산술연산 최적화 기법을 적용해 다양한 수식을 쉽고 효율적으로 처리

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

딥러닝 framework – theano

theano

- 최초의 딥러닝 라이브러리 중 하나
- Python 기반이며 CPU 및 GPU의 수치 계산에 매우 유용
- Tensorflow와 같은 저수준 라이브러리로
- 딥러닝 모델을 직접 만들거나 Keras와 같은 rapper 라이브러리를 사용 가능
- 확장 학습 프레임워크와 달리 확장성이 뛰어나지 않으며 다중 GPU 지원이 부족
- 범용적 딥러닝 개발 목적으로 여전히 많은 개발자가 선택

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

딥러닝 framework – Keras

- Theano와 Tensorflow는 저수준 라이브러리로 접근이 용이하지 않음
- 단순화된 인터페이스로 효율적인 신경망 구축 가능
- Theano 또는 Tensorflow의 상위의 rapper로 구성
- Python으로 작성되었으며, 초보자의 접근이 용이함
- 풍부한 개발자 층을 확보하며 다양한 문서와 레퍼런스 제공
- tensorflow에서 상위 수준 API로 Keras를 적극 지원

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

딥러닝 framework – caffe

- 표현, 속도 및 모듈성을 염두에 두고 개발
- BVLC(Berkeley Vision and Learning Center) 에서 주로 개발
- Python 인터페이스를 가지고 있는 C++ 라이브러리
- CNN(Convolusional Neural Network) 모델 구축시 많이 사용
- Caffe Model Zoo에서 미리 훈련된 여러 네트워크를 사용 가능
- 페이스북은 최근 고성능 학습 모델을 구축 할 수 있는 Caffe2를 공개

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

딥러닝 framework – torch

- Lua 기반의 딥러닝 프레임워크
- 페이스 북, 트위터, 구글 등에서 개발
- GPU 처리를 위해 C/C ++ 라이브러리와 CUDA를 사용
- 유연한 모델을 간단하게 작성할 수있도록 하는 것을 목표
- PyTorch라고 불리는 Torch의 Python 구현 등장

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

딥러닝 framework – PyTorch

- 딥러닝 모델을 구축하고 복잡한 텐서 연산을 수행하기 위한 Python 기반의 라이브러리
- Torch는 Lua를 사용하지만 PyTorch는 Python을 사용
- 기본적인 Python 사용자의 용이한 접근 가능
- PyTorch는 Torch의 아키텍처 스타일을 개선
- 딥러닝 모델 구축 용이성과 투명성 제공

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

딥러닝 framework – ETC

출처 : https://hub.packtpub.com/top-10-deep-learning-frameworks/

Tensor & Flow

Tensor는

하나의 객체
쉽게 말하자면 list, queue 같은 자료구조라고 생각

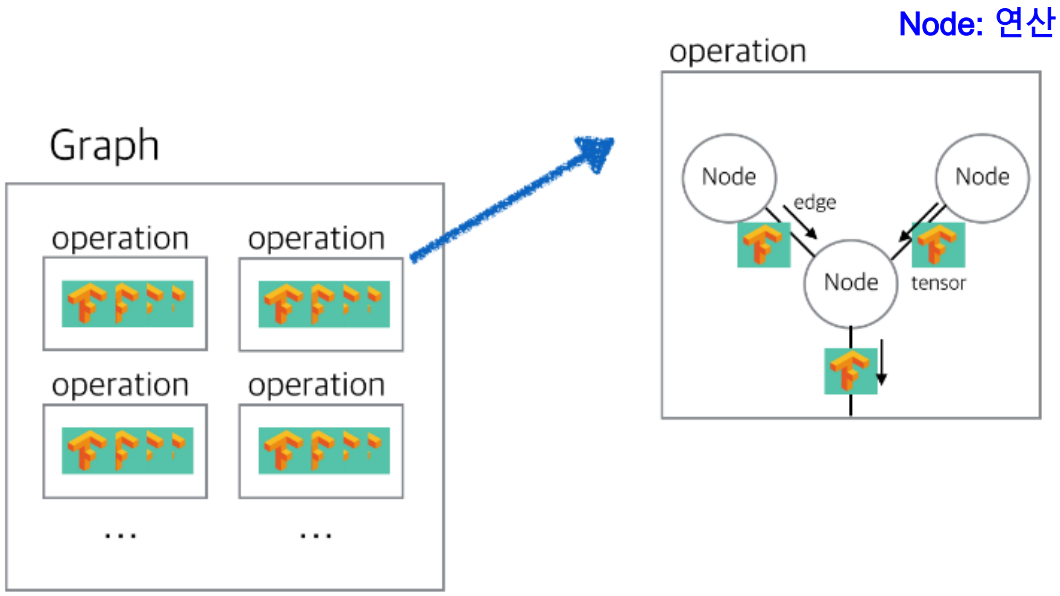
- 대량의 데이터를 효과적으로 처리하기 위해 numpy를 확장한 자료 구조

```
# What is tensor?
tensor1 = 7             # 0-dimensional
tensor2 = [7]           # 1-dimensional
tensor3 = [[1,2,3],     # 2-dimensional
           [4,5,6]]     ...
           ...
```

Tensor & Flow

Flow는

- 연산을 용이하게 하기 위해 각각의 연산을 작은 단위로 분해하여 Graph로 연결한 구조



❖ Tensor데이터 구조가 이동하면서 Operation에 의해 연산을 수행하는 구조

출처 : http://goodtogreate.tistory.com

TensorFlow Hello World

```
import tensorflow as tf

hello = tf.constant('Hello World, TensorFlow!')

sess = tf.Session()

print(sess.run(hello))

sess.close()
```

실습 : tensorflow_01.py

TensorFlow 실행구조

Python

```python
#-*- coding: utf-8 -*-
x = 1
y = x + 9
print(y) # Python 자료구조 : 10
```

TensorFlow 선언영역

```python
import tensorflow as tf

x = tf.constant(1, name = 'x')
y = tf.Variable(x + 9, name = 'y')
print(y) # Tensor 객체 : <tensorflow.python.ops.variables.Variable object at 0x7f4e49d8b610>

print(y.value()) # Tensor 객체 정보 출력 : Tensor("y/read:0", shape=(), dtype=int32)

model = tf.global_variables_initializer()
```

TensorFlow 실행영역

```python
with tf.Session() as sess:
    sess.run(model)
    print(sess.run(y)) # Tensor 값 출력 :10
```

❖ with 구문 사용시 session.clsoe() 생략 가능

실습 : tensorflow_02.py

TensorFlow 실행 원리

Building the Graph

```
import tensorflow as tf

mat1 = tf.constant([[3., 3.]])
mat2 = tf.constant([[2.], [2.]])

mat_mul = tf.matmul(mat1, mat2)
```

Launching the Graph

```
sess = tf.Session()

result = sess.run(mat_mul)
print(result)

sess.close()
```

$$\begin{bmatrix} 3.0, & 3.0 \end{bmatrix} \times \begin{bmatrix} 2.0 \\ 2.0 \end{bmatrix} = \begin{bmatrix} 12 \end{bmatrix}$$

1, 2 　　2, 1 　　1, 1

실습 : tensorflow_03.py

TensorFlow tensor – variable

variable
constant
placeholder

TensorFlow Variables는

- tf.Variable()를 이용해서 변수명 정의 및 초기화
- 변수는 그래프 실행 시 파라미터 저장 및 갱신에 사용
- 메모리에서 Tensor를 저장하는 버퍼 역할 수행
- 실행 시 변수 초기화 과정 필수

TensorFlow 변수화

```
import tensorflow as tf

x = tf.constant(10, name = 'x')
y = tf.Variable(x + 5, name = 'y')

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(y))
```

변수 초기화 필수

그냥 무조건 넣어줘라!

실습 : tensorflow_04.py

Tensor – Rank, Shape, Dtype

Tensor의 Rank는 차원 단위로 표현

Rank	Math entity	Python example
0	Scalar (magnitude only)	s = 483
1	Vector (magnitude and direction)	v = [1.1, 2.2, 3.3]
2	Matrix (table of numbers)	m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	3-Tensor (cube of numbers)	t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]
n	n-Tensor (you get the idea)	....

출처 : https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/resources/dims_types.html

Tensor – Rank, Shape, Dtype

```
import tensorflow as tf

s = tf.constant(483)
v = tf.constant([1.1, 2.2, 3.3])
m = tf.constant([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
t = tf.constant([[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]])

sess = tf.Session()

print(sess.run(tf.rank(s)))
print(sess.run(tf.rank(v)))
print(sess.run(tf.rank(m)))
print(sess.run(tf.rank(t)))

sess.close()
```

실습 : tensorflow_07.py

Tensor –Shape 늘 체크하기

Tensor의 차원 표현은 Rank, Shape, Dimension number로 표현

Shape는 Python 리스트, 정수형 튜플 또는 TensorShape class 로 표현

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

각 차원의 값을 None으로 지정하면 가변길이 적용 가능 -> 어떠한 값이든 들어올 수 있음

출처 : https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/resources/dims_types.html

Tensor -Shape

```
import tensorflow as tf

s = tf.constant(483)
v = tf.constant([1.1, 2.2, 3.3])
m = tf.constant([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
t = tf.constant([[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]])

print(s.shape)
print(v.shape)
print(m.shape)
print(t.shape)
```

실습 : tensorflow_08.py

Tensor – Dtype

Tensor의 Dtype은 데이터 타입을 의미

Data type	Python type	Description
DT_FLOAT	tf.float32	32 비트 부동 소수.
DT_DOUBLE	tf.float64	64 비트 부동 소수. 속도 느림
DT_INT8	tf.int8	8 비트 부호 있는 정수.
DT_INT16	tf.int16	16 비트 부호 있는 정수.
DT_INT32	tf.int32	32 비트 부호 있는 정수.
DT_INT64	tf.int64	64 비트 부호 있는 정수.
DT_UINT8	tf.uint8	8 비트 부호 없는 정수.
DT_STRING	tf.string	가변 길이 바이트 배열. Tensor의 각 원소는 바이트 배열.
DT_BOOL	tf.bool	불리언.

TensorFlow tensor - placeholder

TensorFlow placeholder는

- tf.placeholder(dtype, shape=None, name=None)
  (런타임 시에)
- **실행 시점에 데이터 제공**
- feed_dict를 이용해 데이터 feeding runtime assign
  dictionary type(json과 유사)

외부에서 파일을 읽어오거나 연산으로 나온 값을 넣을 때

placeholder 선언

```
import tensorflow as tf

x = tf.placeholder("int32")
y = tf.placeholder("int32")

z = tf.multiply(x, y)

sess = tf.Session()

result = sess.run(z, feed_dict = {x : 2, y : 5})
print(result)
sess.close()
```

feeding

실습 : tensorflow_05.py

TensorFlow tensor – sparse tensor

```
import tensorflow as tf
                        위치
st = tf.SparseTensor(indices=[[0, 0], [1, 2]], values=[1.0, 1.0], dense_shape=[3, 4])

dense = tf.sparse_tensor_to_dense(st)

c = tf.constant([[1.0, 2.0, 3.0, 4.0]], dtype=tf.float32)
print(c.shape)
tp = tf.transpose(c)  transpose: 행과 열을 바꿔줌
print(tp.shape)
dense_matmul = tf.sparse_tensor_dense_matmul(st, tp)

with tf.Session() as sess:
    result = sess.run(dense_matmul)
    print("dense \n", sess.run(dense))
    print("transpose \n", sess.run(tp))
    print("result \n", result)
```

```
[[1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
```

실습 : tensorflow_09.py

TensorFlow Interactive Session

TensorFlow Interactive Session는

- Jupyter와 같은 대화형 환경에서 Interactive Session을 이용
- Tensor.eval() 혹은 Operation.run() c

Interactive Session 선언

변수 초기화

Tensor.eval()

```
import tensorflow as tf

sess = tf.InteractiveSession()

x = tf.Variable([1.0, 2.0])
a = tf.constant([3.0, 3.0])

x.initializer.run()

sub = tf.sub(x, a)
print(sub.run())

sess.close()
```

TensorFlow TensorBoard

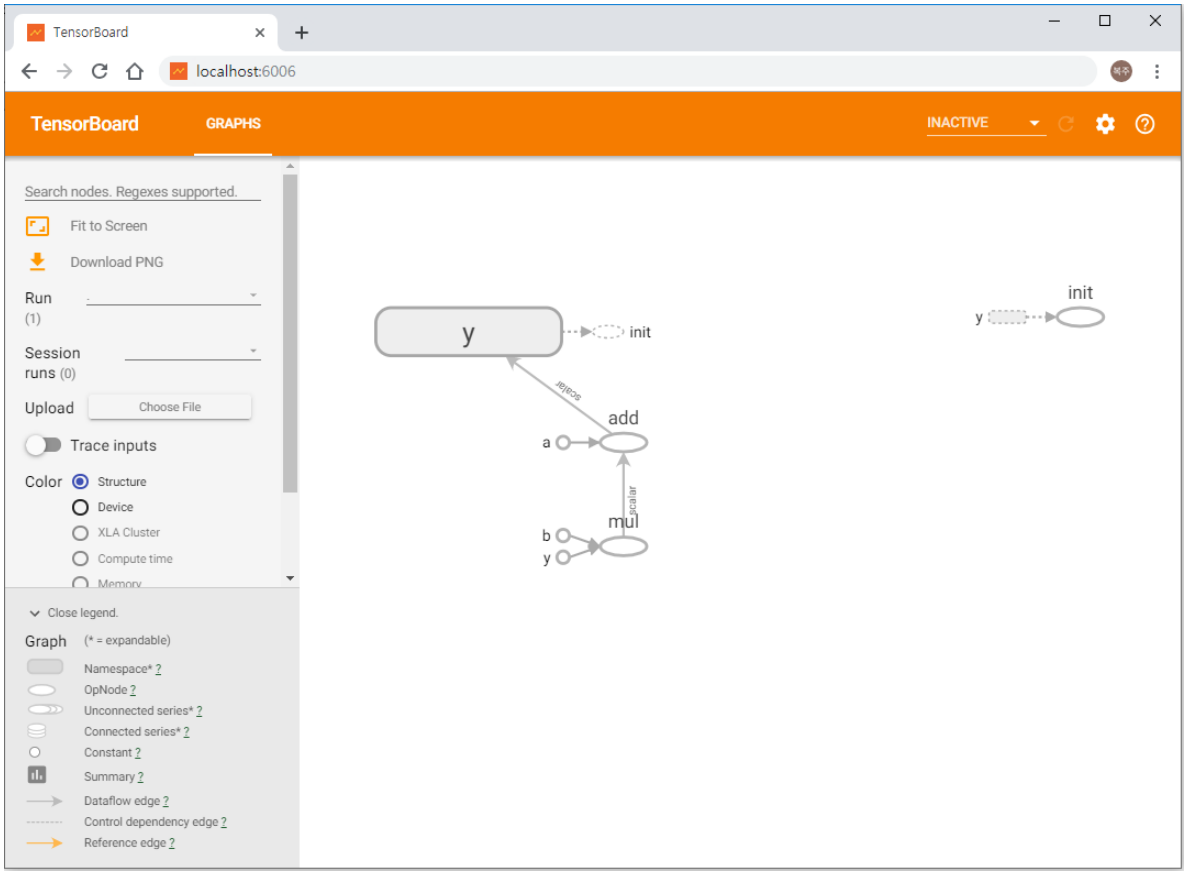
터미널에서

내 위치 C:\Temp/tensorflowlogs

```
(venv) G:₩sync₩Project₩Python₩pycharm₩DeepLearningZeroToAll-master>tensorboard --logdir=d:/temp/tensorflowlogs
TensorBoard 1.12.2 at http://bjkim-pc2:6006 (Press CTRL+C to quit)
```

http://localhost:6006

브라우저에서

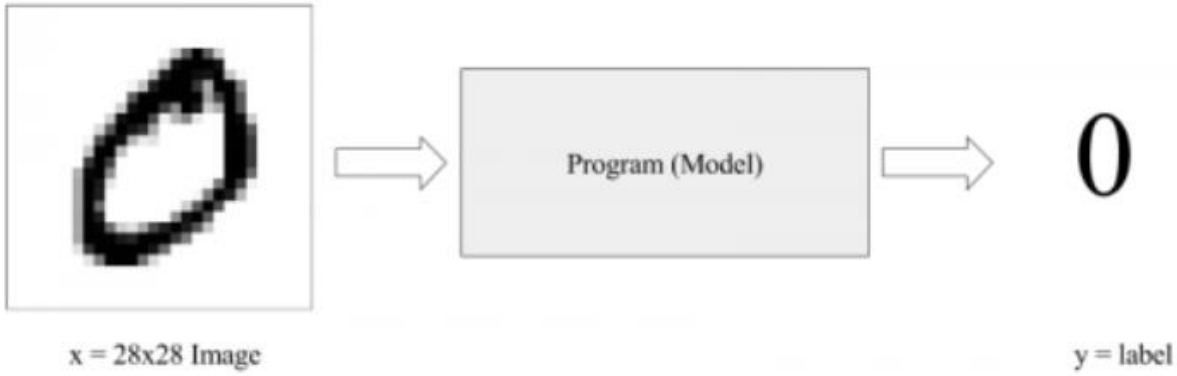


TensorFlow MNIST

MNIST 는

- MNIST는 간단한 컴퓨터 비전 데이터셋
- 머신러닝의 “Hello World!!!”
- 모델이 이미지에 대해 어떤 숫자인지 예측하는 모델을 훈련

(mnist_introduction.py)



TensorFlow MNIST

MNIST 데이터셋 다운로드

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

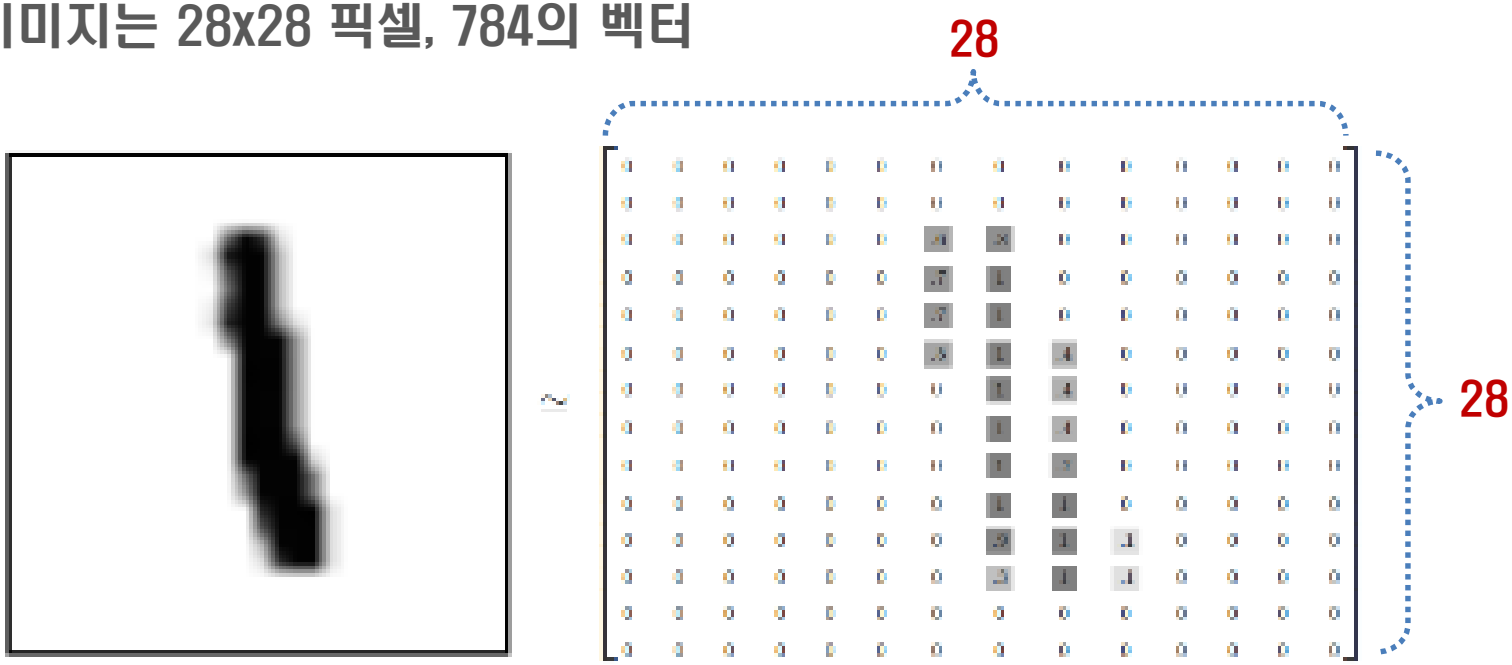
- 55,000개의 학습 데이터(mnist.train),
- 10,000개의 테스트 데이터(mnist.text)
- 5,000개의 검증 데이터(mnist.validation)
- 학습 이미지 : mnist.train.images
- 학습 라벨은 : mnist.train.labels

  머신러닝, 딥러닝에서 답 의미

TensorFlow MNIST

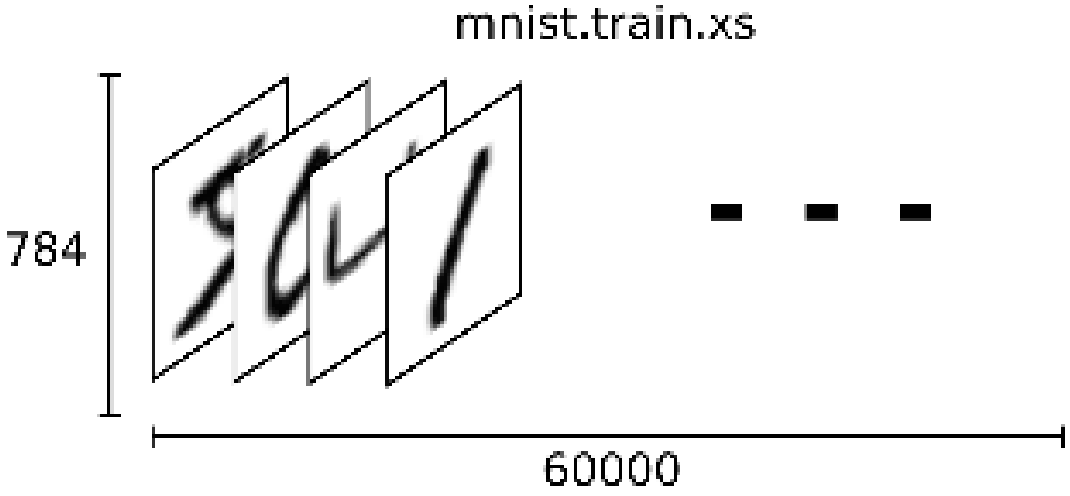
```
nb_classes = 10

# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])
```

MNIST 이미지는 28x28 픽셀, 784의 벡터



TensorFlow MNIST

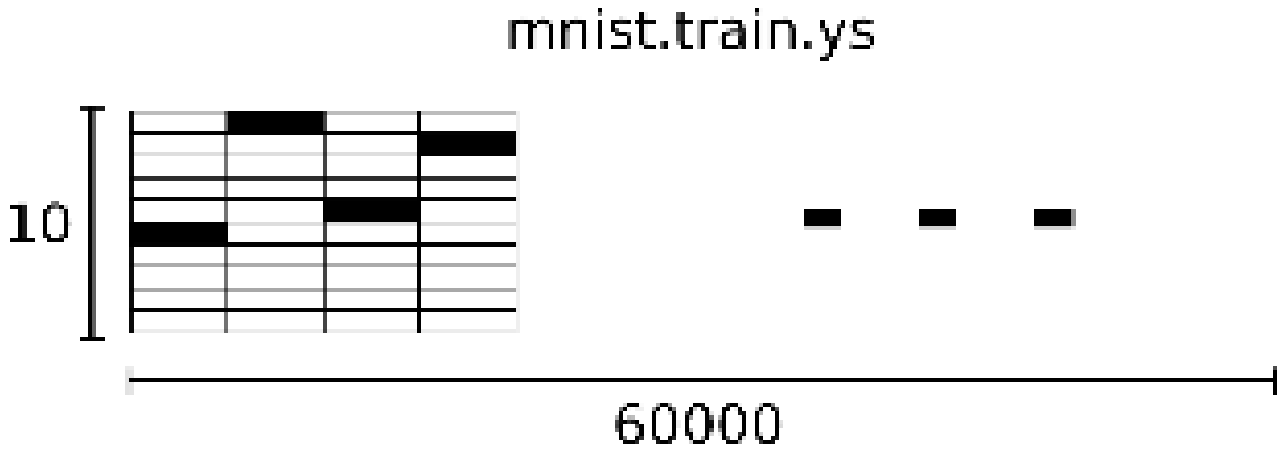
mnist.train.images는 [55000, 784]의 형태를 가진 텐서(n차원 배열)



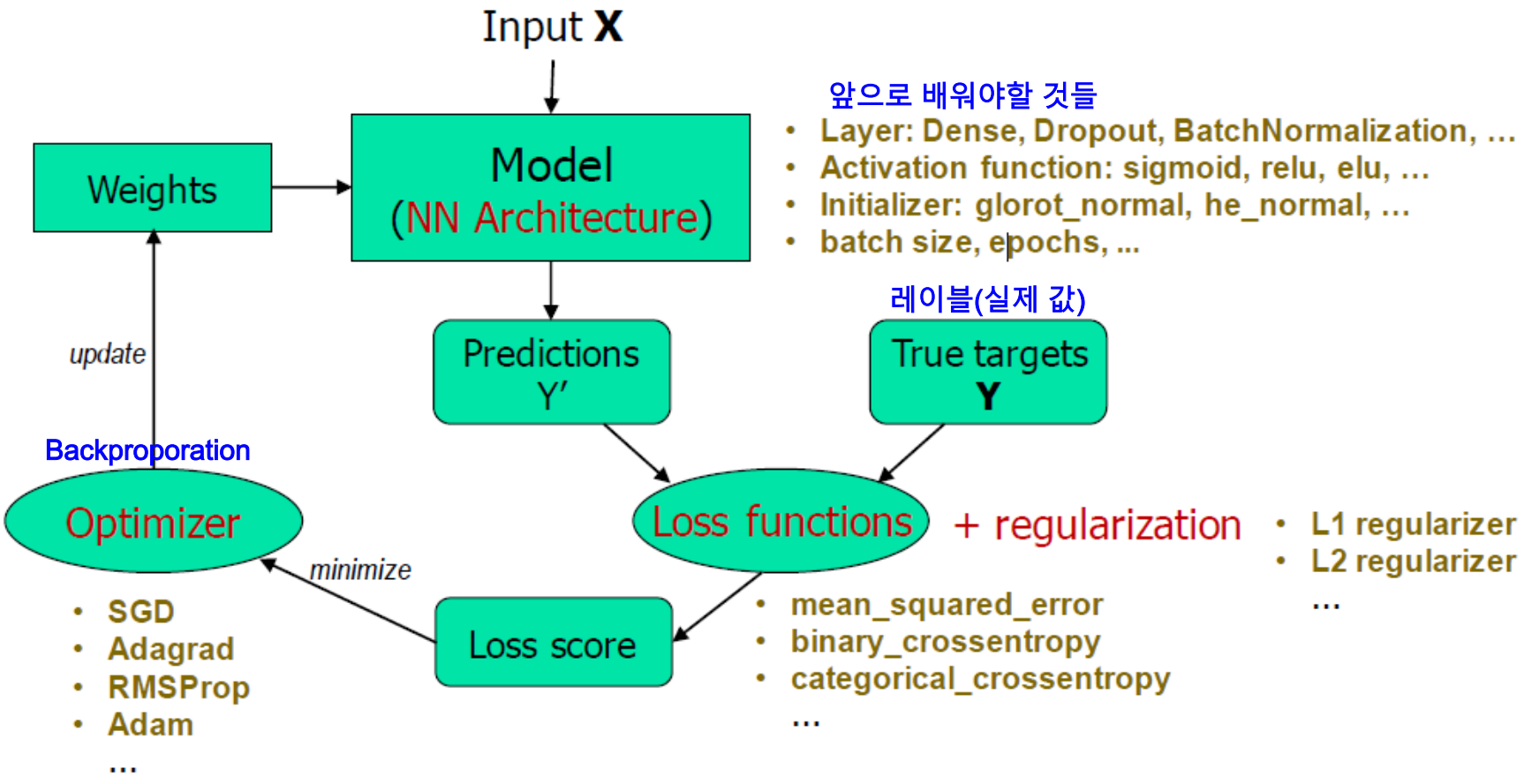
TensorFlow MNIST

mnist.train.labels는 [55000, 10]의 실수 배열

- 소프트맥스 회귀의 결과가 정수형이 아닌 실수형으로 산출



Deep Neural Network – Overview

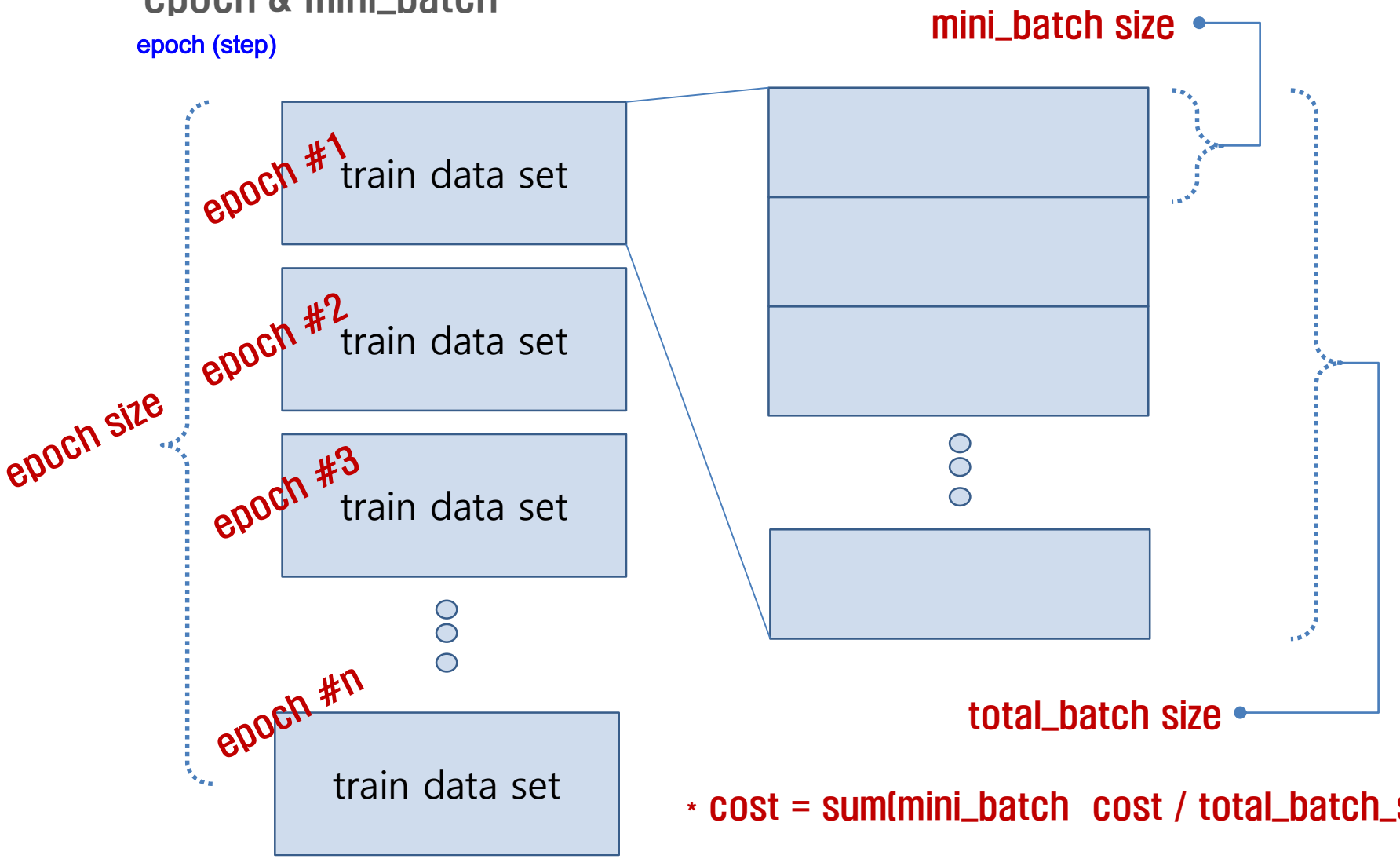


TensorFlow MNIST

6만개 전체를 다 돌리고 가중치를 업데이트 시키면
시간이 오래 걸리므로 쪼개서 가중치를 업데이트 한다.

epoch & mini_batch

epoch (step)



* cost = sum(mini_batch cost / total_batch_size)

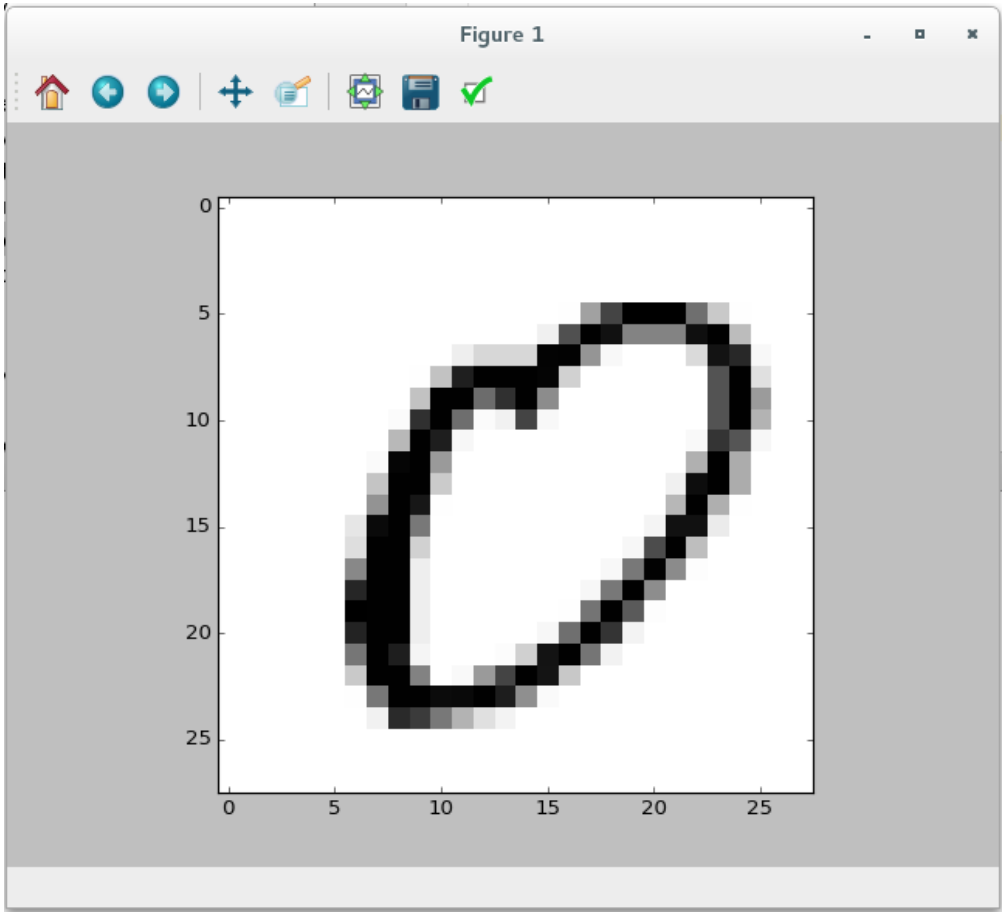
TensorFlow MNIST

epoch & mini_batch

```
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        c, _ = sess.run([cost, optimizer], feed_dict={
                         X: batch_xs, Y: batch_ys})
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1),
          'cost =', '{:.9f}'.format(avg_cost))

print("Learning finished")
```

TensorFlow MNIST

```
Epoch: 0001 cost = 2.827615561
Epoch: 0002 cost = 1.061584424
Epoch: 0003 cost = 0.837877559
Epoch: 0004 cost = 0.734011201
Epoch: 0005 cost = 0.670121456
Epoch: 0006 cost = 0.624561464
Epoch: 0007 cost = 0.590876855
Epoch: 0008 cost = 0.563826518
Epoch: 0009 cost = 0.541655943
Epoch: 0010 cost = 0.522702615
Epoch: 0011 cost = 0.506496948
Epoch: 0012 cost = 0.492143618
Epoch: 0013 cost = 0.480045103
Epoch: 0014 cost = 0.469025962
Epoch: 0015 cost = 0.458973900
Learning finished
Accuracy:  0.8961
Label:  [0]
Prediction:  [0]
```



실습 : tensorflow_mnist.py

TensorFlow Model Save

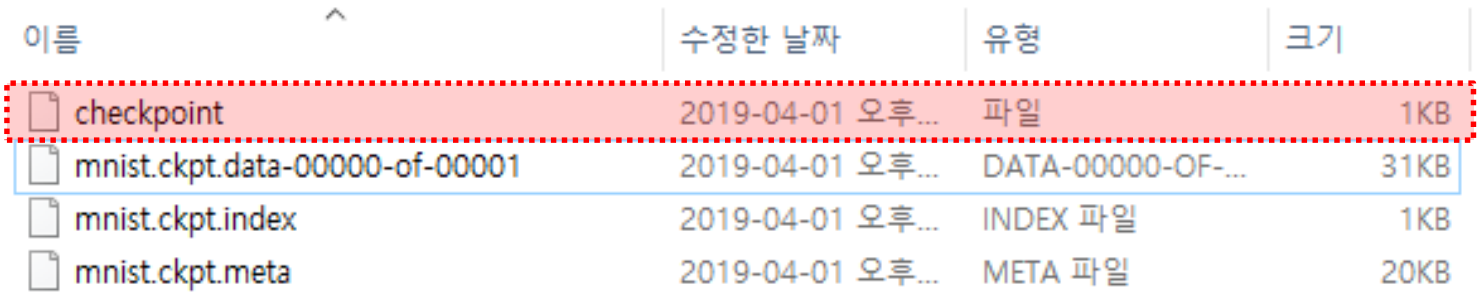
Saver 생성

```
saver = tf.train.Saver()

with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())
    # Training cycle
    for epoch in range(num_epochs):
        avg_cost = 0

        for i in range(num_iterations):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            _, cost_val = sess.run([train, cost], feed_dict={X: batch_xs, Y: batch_ys})
            avg_cost += cost_val / num_iterations

        print("Epoch: {:04d}, Cost: {:.9f}".format(epoch + 1, avg_cost))

    print("Learning finished")

    saver.save(sess, "model/mnist.ckpt")
```

모델 저장

실습 : tensorflow_model_save.py

TensorFlow Model Save



이름	수정한 날짜	유형	크기
checkpoint	2019-04-01 오후...	파일	1KB
mnist.ckpt.data-00000-of-00001	2019-04-01 오후...	DATA-00000-OF-...	31KB
mnist.ckpt.index	2019-04-01 오후...	INDEX 파일	1KB
mnist.ckpt.meta	2019-04-01 오후...	META 파일	20KB

all_model_checkpoint_paths : 저장한 모든 체크포인트 기록

model_checkpoint_path : 가장 최근에 저장된 체크포인트 이름 기록

.meta 파일은 모델의 그래프 구조를 저장

.data, .index 파일은 binary 형태로 학습된 파라미터를 저장

TensorFlow Model Restore

saver 생성

```
saver = tf.train.Saver()
```

모델 restore

```
with tf.Session() as sess:
    saver.restore(sess, 'model/mnist.ckpt')
    print(
        "Accuracy: ",
        accuracy.eval(
            session=sess, feed_dict={X: mnist.test.images, Y: mnist.test.labels}
        ),
    )

    # Get one and predict
    r = random.randint(0, mnist.test.num_examples - 1)
    print("Label: ", sess.run(tf.argmax(mnist.test.labels[r : r + 1], 1)))
    print(
        "Prediction: ",
        sess.run(tf.argmax(hypothesis, 1), feed_dict={X: mnist.test.images[r : r + 1]}),
    )
```

실습 : tensorflow_model_restore.py