Tensorflow Introduction

최준명
Pusan National University
Vision & Intelligent system Lab
https://jimmylifestudy.blogspot.kr/
jmchoi@pusan.ac.kr

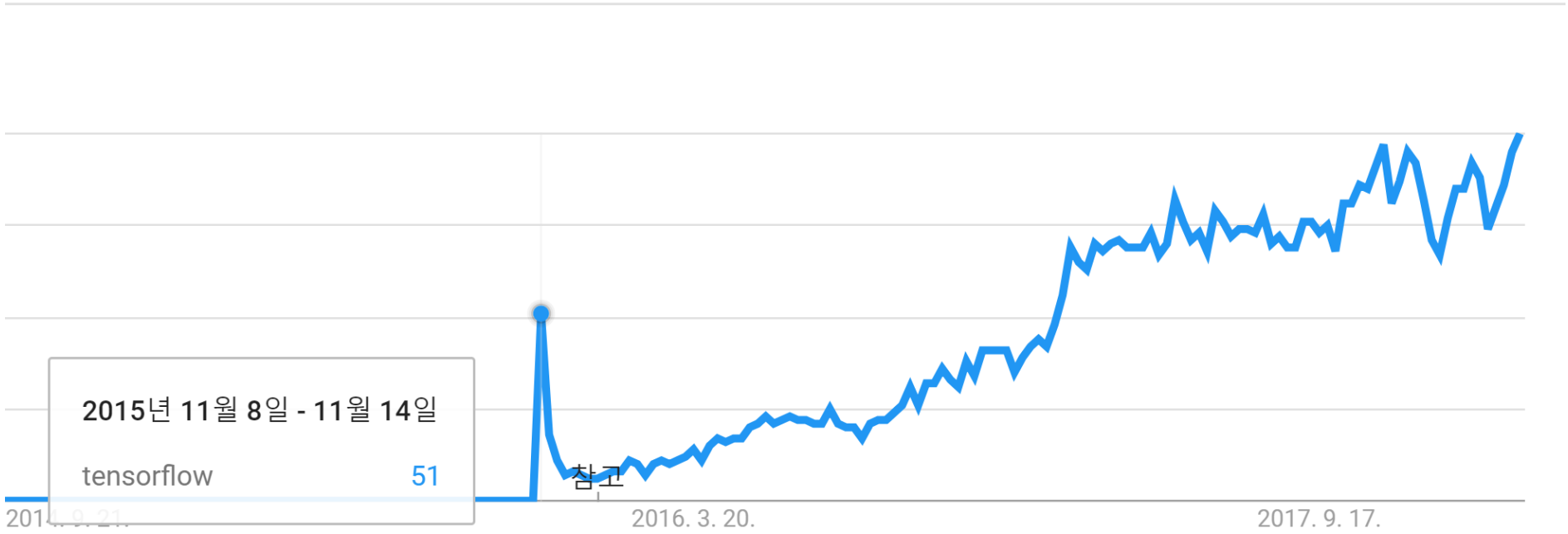
Tensorflow 란?

Dataflow Graph 기반의 수치 계산을 위한

오픈 소스 라이브러리

Tensorflow 란?

2015년 11월 처음 시작



Tensorflow 란?

하나의 언어가 아닌 여러 언어에서 쓸 수 있다.

Productivity = efficient + flexibility + Scalability

간단하지않게 코딩 할 수 있어 새로운 방법에 적용, 구현이 쉽다.

Tensorflow 설치

리눅스 기반의 컴퓨터는 블로그를 보고 따라하면 된다.
윈도우 기반의 컴퓨터는 공식 홈페이지가 보고 따라하면 된다.

설치 순서

1. NVIDIA 그래픽 카드 설치
2. CUDA 설치
3. CuDNN 설치
4. Anaconda 설치
5. Anaconda 가상환경 생성(옵션)
6. Pycharm 설치(옵션)
7. Tensorflow 설치

Tensorflow 배우기

강의 Resource

Stanford CS 20

공식 홈페이지 Tutorial

참고

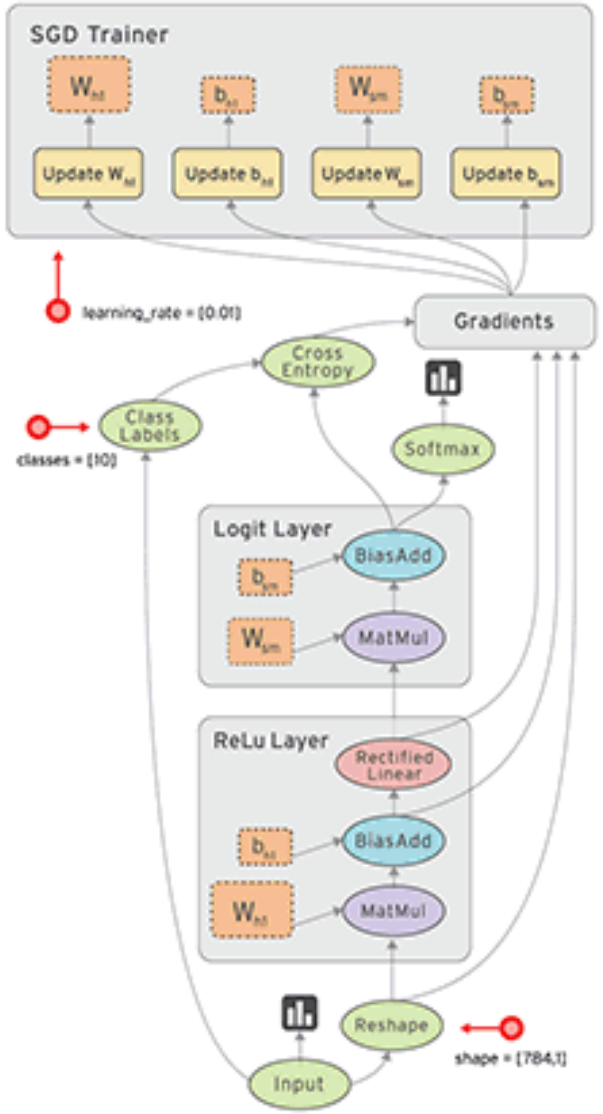
공식 홈페이지 API Document

GitHub

Tensorflow 배우기

Graph and Session

학습을 위한 모델을 쌓고 실행시킨다

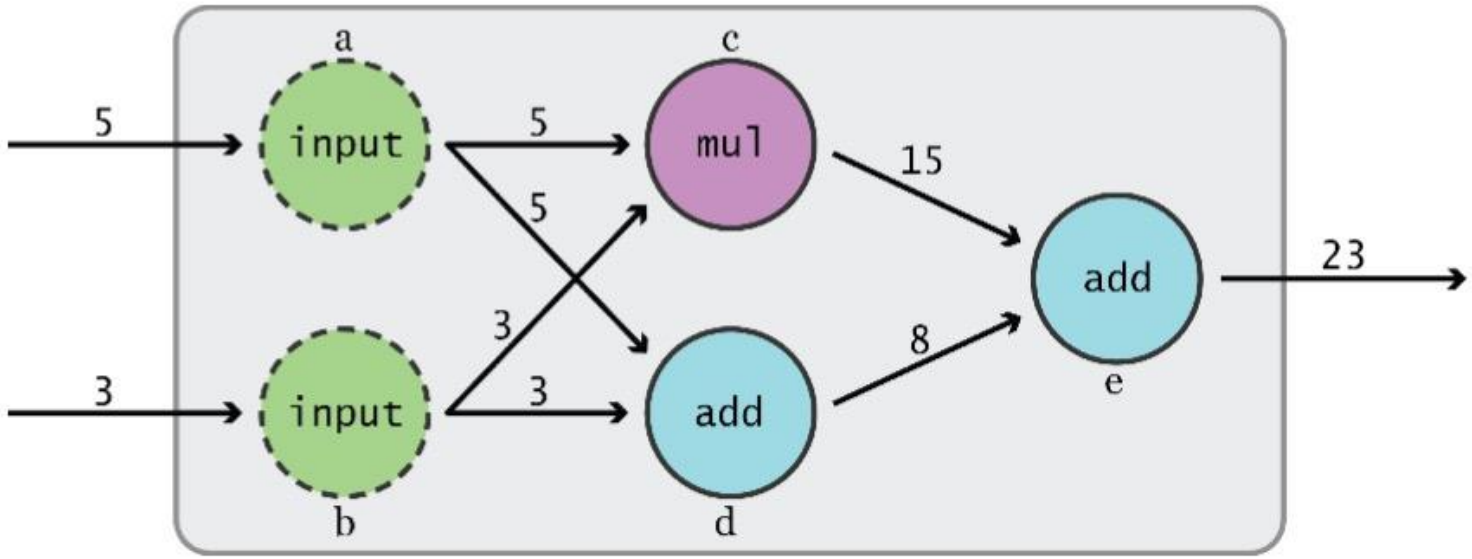


Tensorflow 배우기

Data Flow Graphs

Phase 1: assemble a graph
Phase 2: use a session to execute operations in the graph.

Edges: Tensor: N-dimensional array
Nodes: operators, variables, and constants



Graph from *TensorFlow for Machine Intellige*

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g2dfad34716_0_307

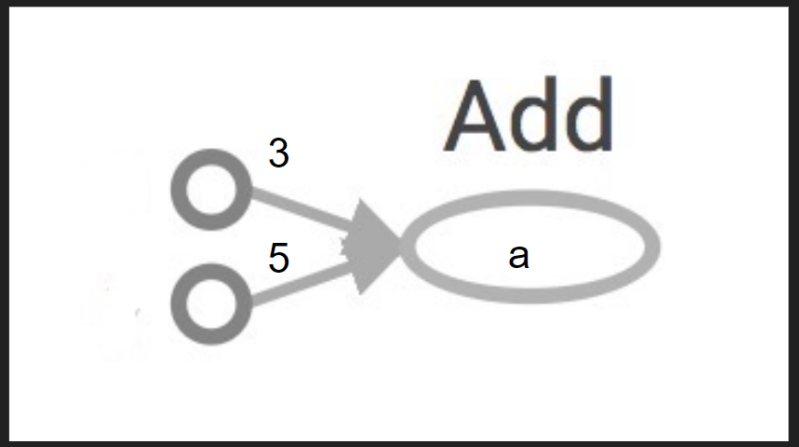
Tensorflow 배우기

변수 값 불러오기

```
import tensorflow as tf
a = tf.add(3, 5)
print(a)
```

```
>> Tensor("Add:0", shape=(), dtype=int32)
```

(Not 8)



Tensor의 값을 볼려고 print 하면 이렇게 나온다..

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g1bcfa4d819_0_210

Tensorflow 배우기

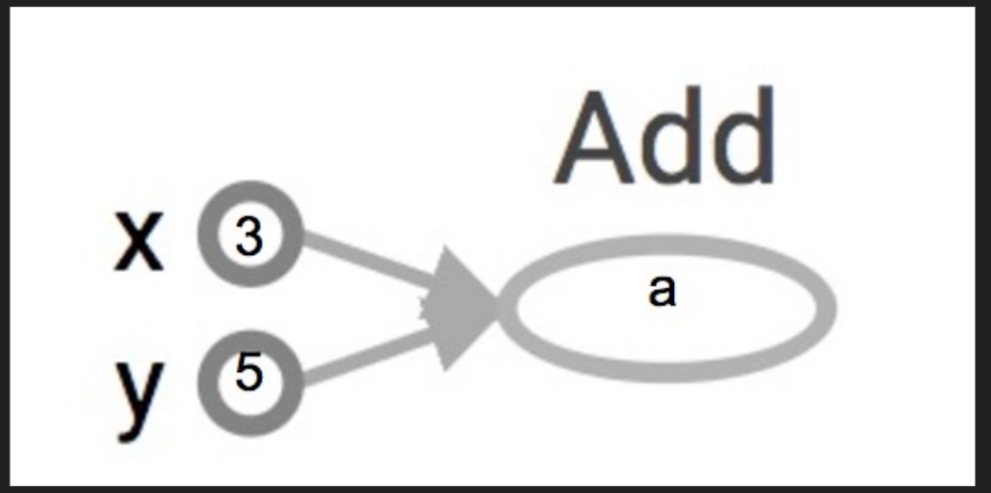
변수 값 불러오기

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

불러오다.

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print(sess.run(a))
sess.close()
```



Session을 열어서 Graph 안의 a라는 변수의 값을 불러와야함.

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g21a6bfb134_0_0

Tensorflow 배우기

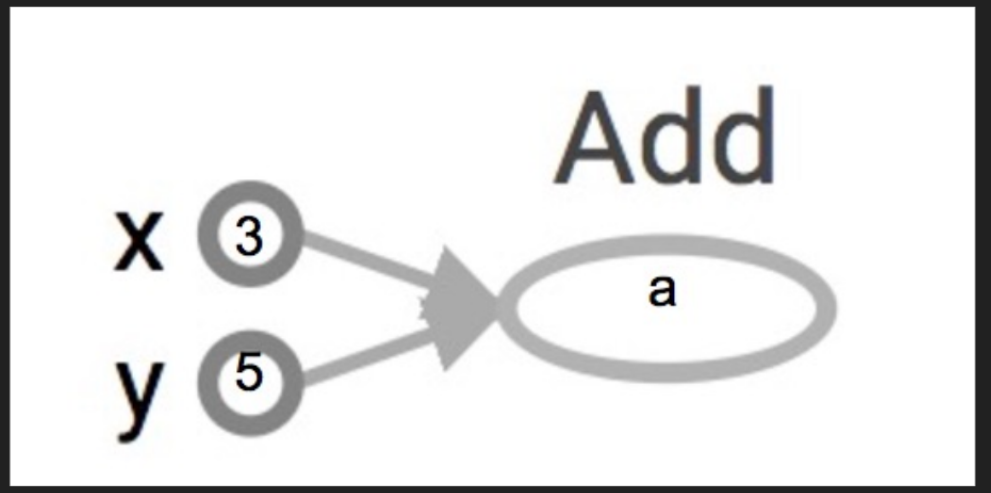
변수 값 불러오기

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of a

불러오다.

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
with tf.Session() as sess:
    print(sess.run(a))
sess.close()
```



Session을 열어서 Graph 안의 a라는 변수의 값을 불러야함.

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g21a6bfb134_0_0

Tensorflow | tf.Session( )

tf.Session()의 역할

tf.Session()

요약(압축)하다.

A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

Session will also allocate memory to store the current values of variables.

할당하다

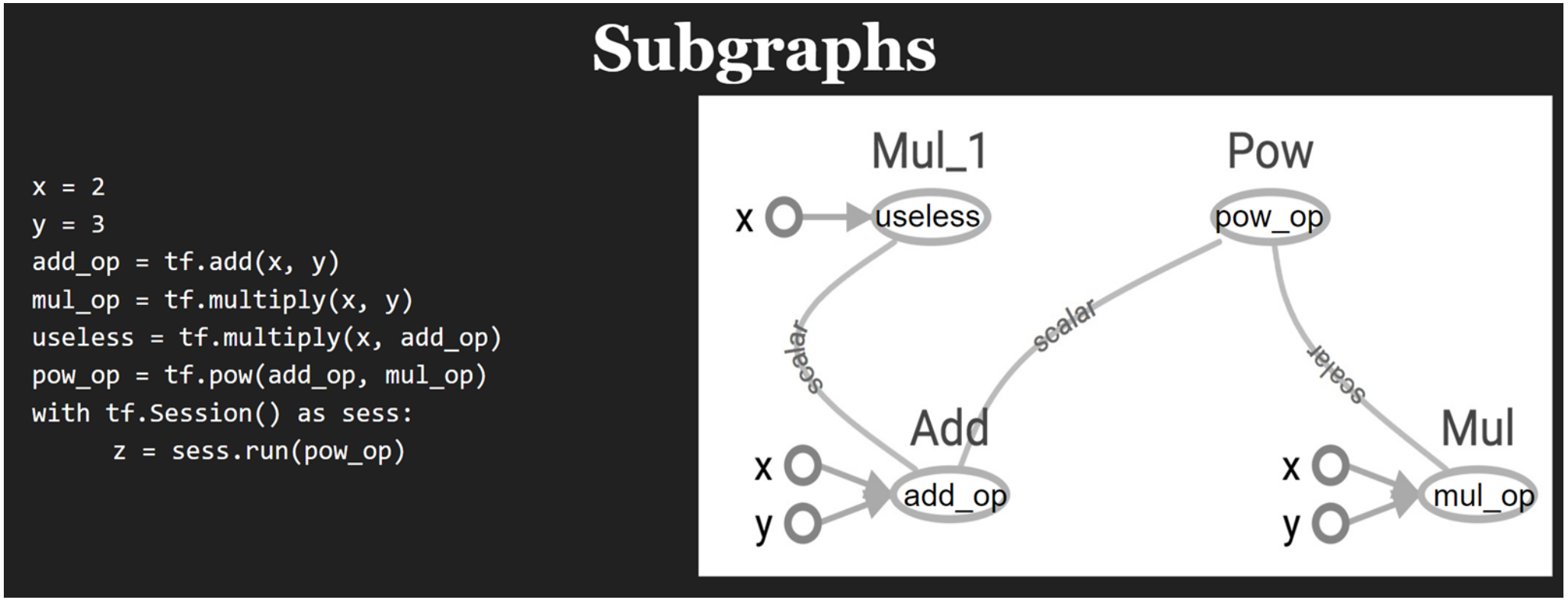
Session은 Operation과 Tensor objects를 담고 있는 환경.
각 변수들의 메모리를 할당하는 역할

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g2dfad34716_0_318

Tensorflow | tf.Session( )

tf.Session()으로 그래프 연산

```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
      z = sess.run(pow_op)
```



Useless라는 nodes는 pow_op을 계산하는데 쓰이지 않음. Session 역시 useless 변수를 계산 안함 –〉 계산량 감소!

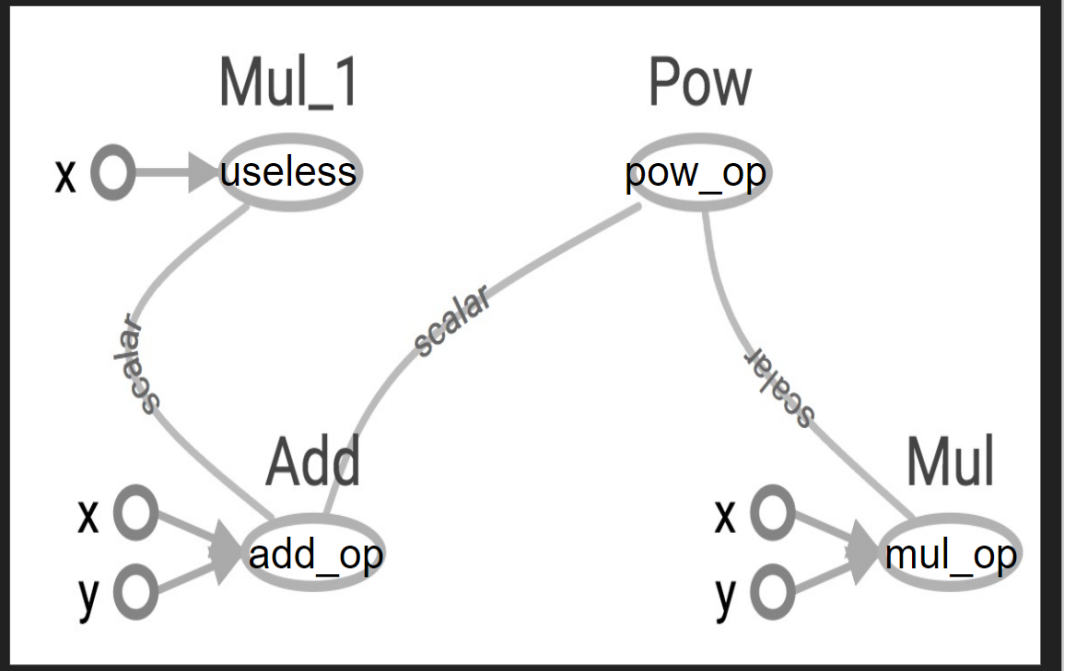
출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g1bcfa4d819_0_288

Tensorflow | tf.Session( )

tf.Session()으로 그래프 연산

Subgraphs

```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
      z, not_useless = sess.run([pow_op, useless])
```




```
tf.Session.run(fetches,
               feed_dict=None,
               options=None,
               run_metadata=None)
```

fetches is a list of tensors whose values you want

Fetches가 여러 개면 list형식으로 입력을 줘야함.

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g1bcfa4d819_0_314

Tensorflow | tf.device( )

CPU 혹은 GPU에 연산 할당하기

Distributed Computation

To put part of a graph on a specific CPU or GPU:

```
# Creates a graph.
with tf.device('/gpu:2'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='b')
  c = tf.multiply(a, b)

# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

# Runs the op.
print(sess.run(c))
```

With tf.device('/gpu:2'): 안의 연산은 gpu:2에서 진행됨
cpu:0, gpu:0 가 default

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g1bcfa4d819_0_585

Tensorflow | tf.Graph( )

tf.Graph()의 역할

그래프를 하나 이상으로 만들고 싶다?? **추천 안함.**

tf.Graph()

Do not mix default graph and user created graphs

```
g = tf.Graph()

# add ops to the default graph
a = tf.constant(3)

# add ops to the user created graph
with g.as_default():
      b = tf.constant(5)
```

Prone to errors

Graph는 가능한 하나만 생성하는게 좋다.

출처: https://docs.google.com/presentation/d/1dizKPtp9hkuTwVDzoGZdYQb_61ULSsSUvaFfDFuhIc4/edit#slide=id.g1bd9feb8dd_0_33

Tensorflow | tf.Graph( )

tf.Graph()를 하나만 써야하는 이유

- 멀티 그래프는 멀티 Session가 필요한데, 각 Session 마다 기본적으로 모든 리소스를 사용함.
- Data가 멀티 그래프간에 이동이 안됨. 따라서 pytho이나 numpy로 바꿔주거나 해야함.

Dataflow Graph 연산 방식이 좋은 이유

- 블로그를 참고하자.

Tensorflow | Tensorboard

```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a,b)

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

Tensorflow | Tensorboard

```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)
x = tf.add(a,b)
```

저장 위치는 원하는 경로 설정하면 됨

```
writer = tf.summary.FileWriter('./ckpt', tf.get_default_graph())
with tf.Session() as sess:
  print(sess.run(x))
writer.close()
```

기본 default graph를 저장

다 끝났으면 writer를 닫음

Tensorflow | Tensorboard

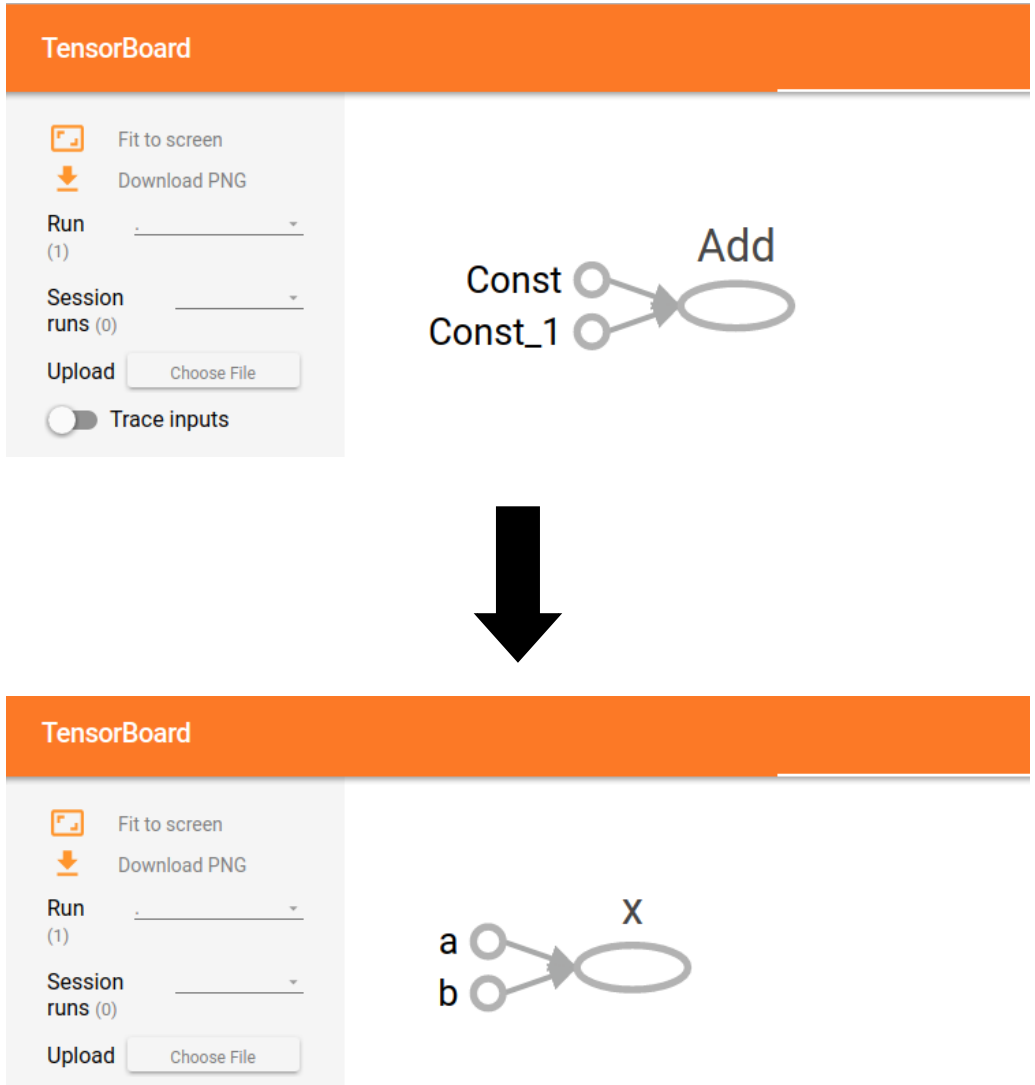
Tensorboard로 학습진행 정도, Graph 형태 등 Tensorflow의 모든 것을 확인 가능!

터미널(ctrl + alt + T)에서 실행

```
$python program_name.py
$tensorboard –logdir=./graphs
```

```
http://localhost:6006/
```

 # 여기 포트로 접속하면 Tensorboard 를 확인 할 수 있음.

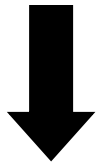



```
a = tf.constant(2, name='a')
b = tf.constant(3, name='b')
x = tf.add(a,b, name='x')
```

출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g4cb222212c3f275d_40

Tensorflow | tf.Variable( )

tf.constant의 단점은 Graph를 정의할 때 함께 저장된다.
즉, Graph를 loading하는게 시간이 오래걸린다..



constant는 꼭 써야하는 경우에만 쓰자!
메모리가 많이 차지하는 데이터의 경우 Varialbe이나 Reader 같은 걸 쓰자!

Tensorflow | tf.Variable( )

tf.Vairable은 Class이다. 따라서 다양한 ops가 있다.
x.value()는 x값인데 실제 TF에서는 .value()를 생략 가능 ex) tf.add(x.value(),…) –> tf.add(x,…)

tf.Variable class

```
# create variables with tf.get_variable
s = tf.get_variable("scalar", initializer=tf.constant(2))
m = tf.get_variable("matrix", initializer=tf.constant([[0, 1], [2, 3]]))
W = tf.get_variable("big_matrix", shape=(784, 10), initializer=tf.zeros_initializer())
```

tf.Variable holds several ops:

```
x = tf.Variable(...)

x.initializer # init op
x.value() # read op
x.assign(...) # write op
x.assign_add(...) # and more
```

출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g2ee508db2e_1_178

Tensorflow | tf.Variable( )

tf.Vairable 값을 읽기 위해서는 반드시 Initialization을 해야한다.

You have to initialize your variables

The easiest way is initializing all variables at once:

```
with tf.Session() as sess:
      sess.run(tf.global_variables_initializer())
```

Initialize only a subset of variables:

```
with tf.Session() as sess:
      sess.run(tf.variables_initializer([a, b]))
```

Initialize a single variable

```
W = tf.Variable(tf.zeros([784,10]))
with tf.Session() as sess:
      sess.run(W.initializer)
```

출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g1bf2cc2b44_0_15

Tensorflow | tf.Variable( )

tf.Vairable은 .eval()을 통해서 확인한다.

```
# W is a random 700 x 100 variable object
W = tf.Variable(tf.truncated_normal([700, 10]))
with tf.Session() as sess:
      sess.run(W.initializer)
      print(W)

>> Tensor("Variable/read:0", shape=(700, 10), dtype=float32)
```

```
# W is a random 700 x 100 variable object
W = tf.Variable(tf.truncated_normal([700, 10]))
with tf.Session() as sess:
      sess.run(W.initializer)
      print(W.eval())                    # Similar to print(sess.run(W))

>> [[-0.76781619 -0.67020458  1.15333688 ..., -0.98434633 -1.25692499
  -0.90904623]
 [-0.36763489 -0.65037876 -1.52936983 ...,  0.19320194 -0.38379928
   0.44387451]
 [ 0.12510735 -0.82649058  0.4321366  ..., -0.3816964   0.70466036
   1.33211911]
 ...,
 [ 0.9203397  -0.99590844  0.76853162 ..., -0.74290705  0.37568584
   0.64072722]
 [-0.12753558  0.52571583  1.03265858 ...,  0.59978199 -0.91293705
  -0.02646019]
 [ 0.19076447 -0.62968266 -1.97970271 ..., -1.48389161  0.68170643
   1.46369624]]
```

출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g4cb222212c3f275d_160

Tensorflow | tf.Variable( )

각 Session은 variable를 copy해서 가지기 때문에 독립적임.

Each session maintains its own copy of variables

```
W = tf.Variable(10)

sess1 = tf.Session()
sess2 = tf.Session()

sess1.run(W.initializer)
sess2.run(W.initializer)

print(sess1.run(W.assign_add(10)))          # >> 20
print(sess2.run(W.assign_sub(2)))           # >> 8

print(sess1.run(W.assign_add(100)))         # >> 120
print(sess2.run(W.assign_sub(50)))          # >> -42

sess1.close()
sess2.close()
```

출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g1c0aab257d_0_18

Tensorflow | tf.Variable( ) | tf.get_variable( )

tf.Variable()은 Class
tf.get_variable()은 wrapper

tf.Variable()로 변수를 정의하는 건 old way
tf.get_variable()로 변수를 정의하면 그뤠잇!!

가장 중요한건
variable sharing과 initialization이 가능

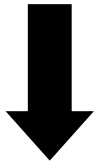
```
tf.get_variable(
    name,
    shape=None,
    dtype=None,
    initializer=None,
    regularizer=None,
    trainable=True,
    collections=None,
    caching_device=None,
    partitioner=None,
    validate_shape=True,
    use_resource=None,
    custom_getter=None,
    constraint=None
)
```

Tensorflow | tf.placeholder( )

Graph and Session

학습을 위한 모델을 쌓고 실행시킨다

그런데 Graph를 정의할 때 f(x,y) = a*x + y에서 x,y값은 나중에 정의하고 싶은데??



tf.Placeholder( )를 쓰자!

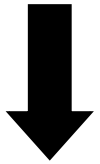
tf.placeholder( )는 Graph 정의할 때 말고 나중에 고유 데이터를 이용하고 싶을 때 feed 할 수 있게 해주는 노드

Tensorflow | tf.placeholder( )

Graph and Session

학습을 위한 모델을 쌓고 실행시킨다

그런데 Graph를 정의할 때 f(x,y) = a*x + y에서 x,y값은 나중에 정의하고 싶은데??



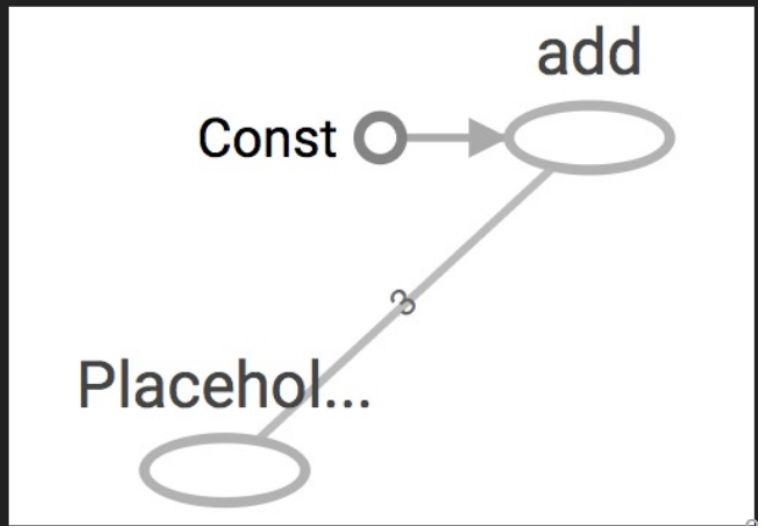
tf.data( )를 쓰자

~~다음 시간에 tf.data()에 대해서 알려 주겠습니다..... 저도 이거 넘나 새로운 기술…~~

Tensorflow | tf.placeholder( )

tf.placeholder( )를 쓸 땐 가능한 shape을 정의해주자. shape=None도 가능하긴함.
그리고 sess.run 과정에서 반드시 feed_dict에서 값을 넣어주자.

```
# create a placeholder of type float 32-bit, shape is a vector of 3 elements
a = tf.placeholder(tf.float32, shape=[3])

# create a constant of type float 32-bit, shape is a vector of 3 elements
b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable
c = a + b  # Short for tf.add(a, b)

with tf.Session() as sess:
      print(sess.run(c, feed_dict={a: [1, 2, 3]}))

# >> [6, 7, 8]
```

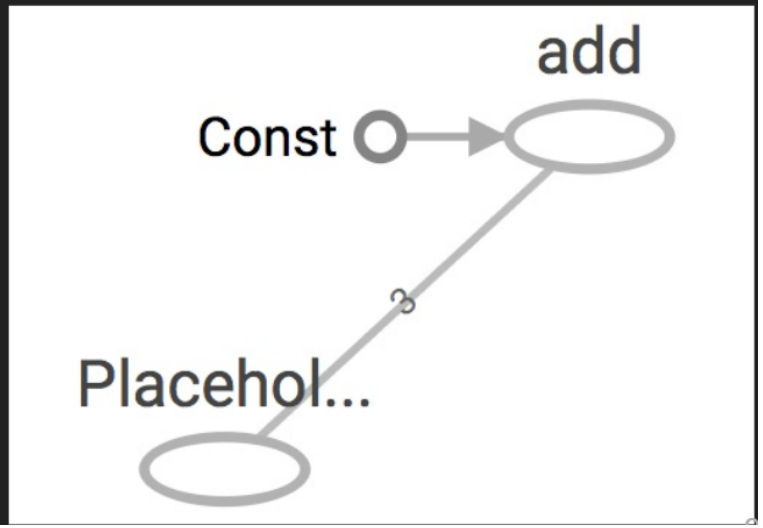


출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g1bd9898b6a_0_103

Tensorflow | tf.placeholder( )

Input 데이터가 하나가 아니라 여러 개일 경우!
for 문을 써서 하나씩 하나씩 feed 해주자!

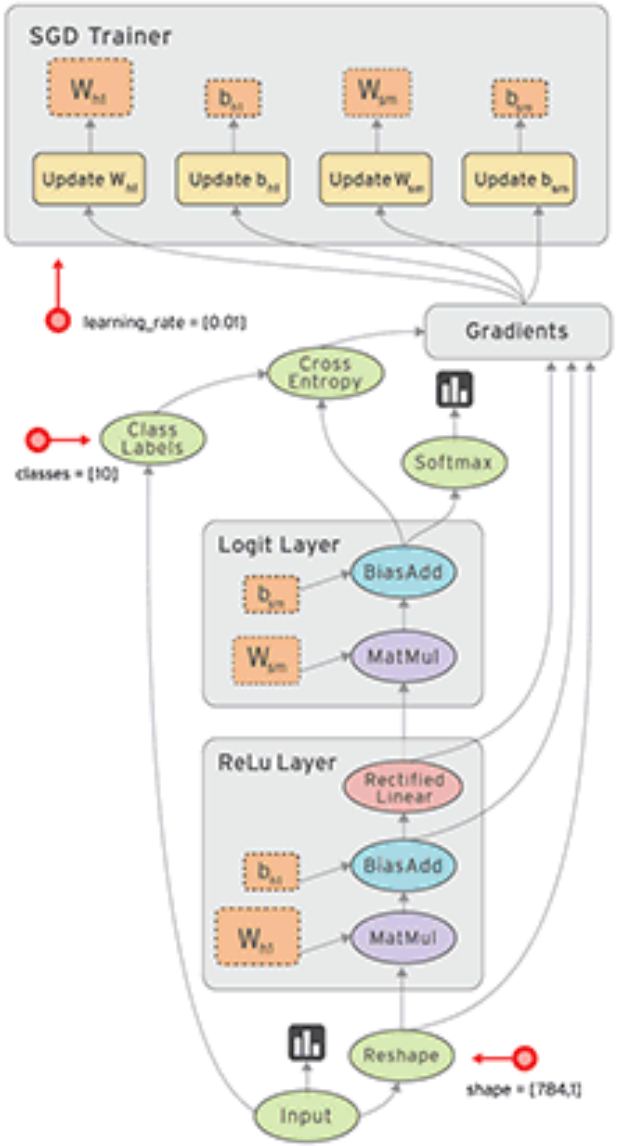
```
# create a placeholder of type float 32-bit, shape is a vector of 3 elements
a = tf.placeholder(tf.float32, shape=[3])

# create a constant of type float 32-bit, shape is a vector of 3 elements
b = tf.constant([5, 5, 5], tf.float32)

# use the placeholder as you would a constant or a variable
c = a + b  # Short for tf.add(a, b)

with tf.Session() as sess:
      for a_value in list_of_values_for_a:
            print(sess.run(c, {a: a_value}))
```



출처: https://docs.google.com/presentation/d/1iO_bBL_5REuDQ7RJ2F35vH2BxAiGMocLC6t_N-6eXaE/edit#slide=id.g4cb222212c3f275d_330

요약

- Tensorflow는 **Dataflow Graph** 기반의 딥러닝 프레임워크이다.
- 따라서 코딩도 **Graph 기반**으로 짜야 한다.
- 그때 필요한게 **tf.Session(), tf.Graph()** 등이 있다.
- 변수에는 **이름**을 붙여줘야 나중에 확인하기 편하다.
- **Tensorboard**로 학습을 모든 상황을 확인할 수 있다.
- 변수 설정은 Variable말고 **get_variable**을 쓰자.
- 내 데이터를 Input값으로 쓰고 싶다면 tf.placeholder() 나 tf.data()를 쓰자.



이제 옆에 그림처럼 모델을 정의만 하면 된다!!!

To be continued..