# Deep Learning

## 머신러닝, 딥러닝 실전개발 입문 Ch5
## 조남운

Namun Cho/ mailto:namun@snu.ac.kr

# 목차

- 딥러닝 개요
- TensorFlow
- TensorBoard
- Keras

# Deep Learning

- 머신러닝의 일종
  - 퍼렙트론 (수학적으로 구현한 뉴런 모형)을 여러 층으로 연결한 인공신경망을 이용한 기계학습 기법
- circle 없는 뉴럴 네트워크
  - 수리적 용이성
- “Deep Neural Network” (DNN): 3중 이상의 Hidden Layer

# DL이 주목받는 이유

- 압도적인 퍼포먼스

# Model of Neuron

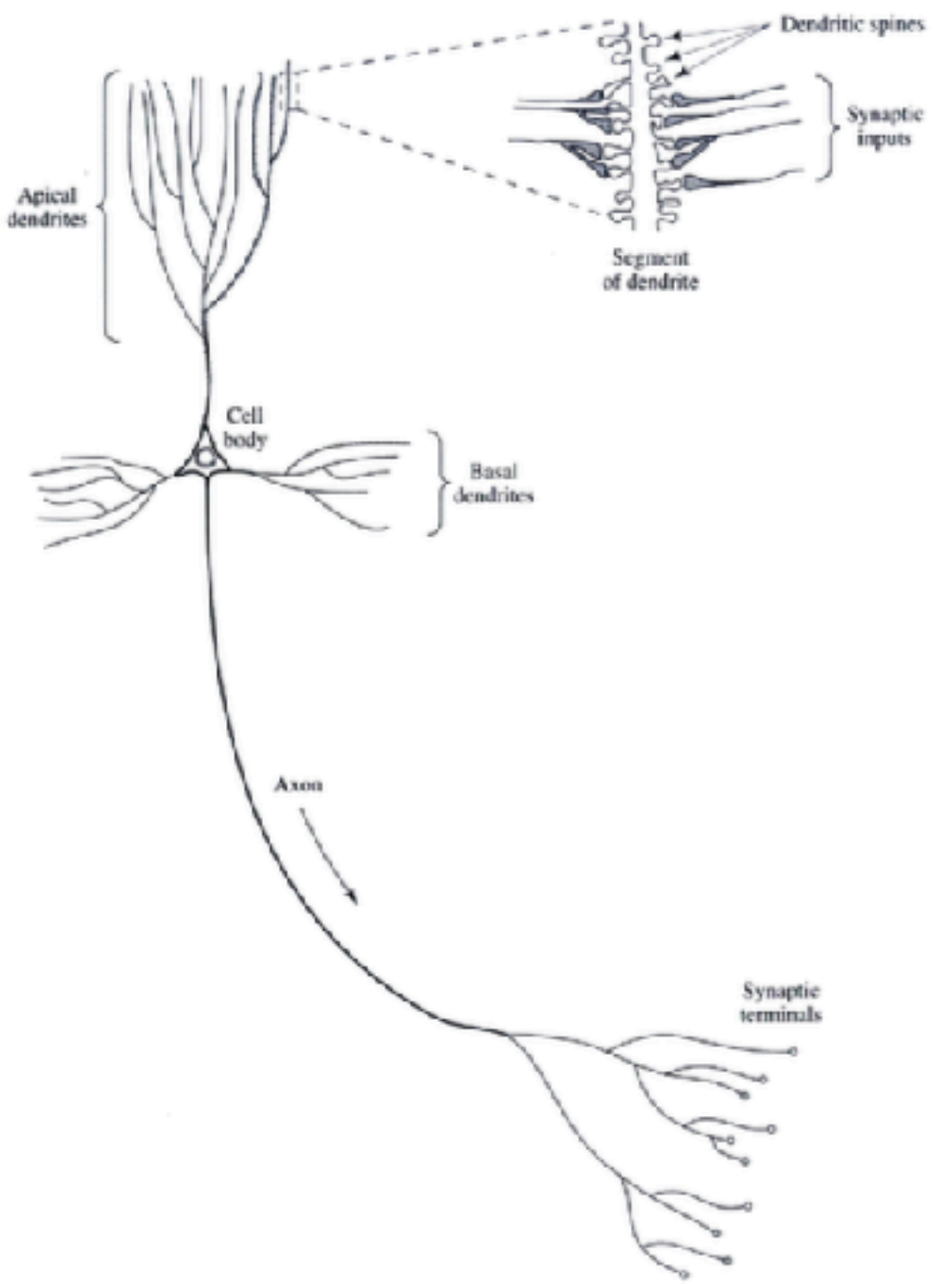
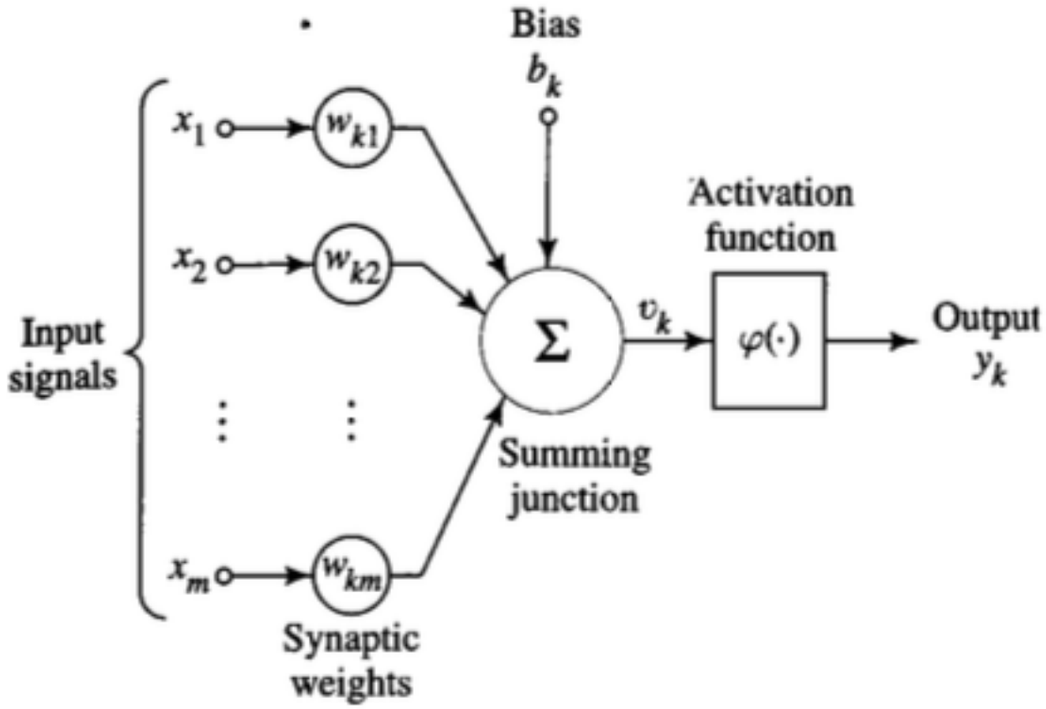



FIGURE 1.2 The pyramidal cell.



# TensorFlow

- Open Source Library for deep learning
  - https://github.com/tensorflow/tensorflow
  - https://www.tensorflow.org
    - $ sudo easy_install —upgrade pip
    - $ pip3 install tensorflow

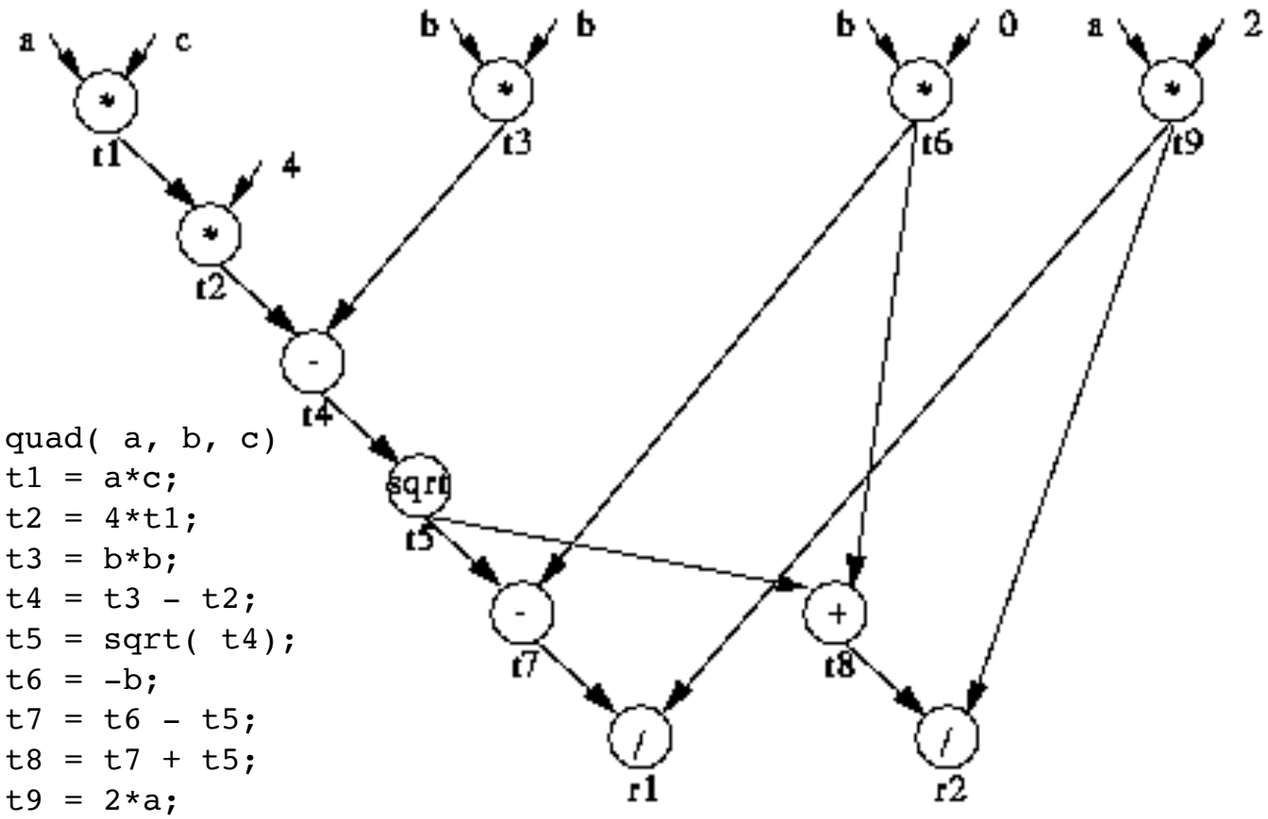
# Simple Calculation

- a+b 는 덧셈의 결과가 아니라 데이터 플로우 그래프라는 객체
- 텐서플로우 세션 (session)으로 데이터 플로우 그래프 객체를 실행하는 것

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

# Data-Flow Graphs (DFG)

- 연산들 사이에서의 자료 의존성을 그래프로 표현한 것
  - 예) 근의 공식
- t2는 t1에 의존적: t2는 t1이 계산된 후에만 계산 가능
- t3은 t1,t2와 무관하게 계산가능




```
quad( a, b, c)
t1 = a*c;
t2 = 4*t1;
t3 = b*b;
t4 = t3 - t2;
t5 = sqrt( t4);
t6 = -b;
t7 = t6 - t5;
t8 = t7 + t5;
t9 = 2*a;
r1 = t7/t9;
r2 = t8/t9;
```

http://bears.ece.ucsb.edu/research-info/DP/dfg.html

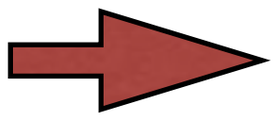
# Variable

- All seems to be DFG
- 텐서플로에서는 변수를 학습해야 할 파라미터들로 사용

```python
import tensorflow as tf

# define constants
a = tf.constant(120, name="a")
b = tf.constant(140, name="b")
c = tf.constant(199, name="c")

# define variable
x = tf.Variable(0, name="x")

# define data flow graph

calc_op = a + b + c
assign_op = tf.assign(x,calc_op)

# run session

sess = tf.Session()
sess.run(assign_op)

# output

print(sess.run(x))
```

# Placeholder

- Similar to Array?



```python
import tensorflow as tf

# define placeholder

a = tf.placeholder(tf.int32, [None])

# define some operation

b = tf.constant(2)
x2_op = a * b

# session start

sess = tf.Session()

# run & output

r1 = sess.run(x2_op, feed_dict = {a:[1,2,3]})
print(r1)

r2 = sess.run(x2_op, feed_dict = {a:[10,100,130,200,4002312]})
print(r2)
```

```
[2 4 6]
[     20     200     260     400 8004624]
```

# DL example

- height, weight, 그리고 비만도 (마름, 보통, 비만) 데이터를 학습시킨뒤, 5000개의 테스트셋으로 정확도를 판별
- data format: csv
- Procedure:
  - data process
  - making DFGs
  - define model learning
  - session run

```
import pandas as pd # to read csv
import numpy as np
import tensorflow as tf

# data read

csv = pd.read_csv("../_mainText_srcs/ch5/bmi.csv")

# data normalizing

csv["height"] /= 200
csv["weight"] /= 100

# label to array

bmi_class = {"thin": [1,0,0], "normal": [0,1,0], "fat": [0,
csv["label_pat"] = csv["label"].apply(lambda x: np.array(bm

# test set

test_csv = csv[15000:20000]
test_pat = test_csv[["weight","height"]]
test_ans = list(test_csv["label_pat"])

# making DFG

#1. declaring placeholder

x = tf.placeholder(tf.float32, [None,2])   # height, weight
y = tf.placeholder(tf.float32, [None,3])   # pattern

#2. declaring variables
```

# softmax regression

- softmax regression

```
#1. declaring placeholder

x = tf.placeholder(tf.float32, [None,2])   # height, we
y = tf.placeholder(tf.float32, [None,3])   # pattern

#2. declaring variables

W = tf.Variable(tf.zeros([2,3])) # weight
b = tf.Variable(tf.zeros([3])) # bias


#3. defining softmax regression

y_hat = tf.nn.softmax(tf.matmul(x,W)+b)
```

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

$$y = \sigma(W \bullet \mathbf{z} + b)$$

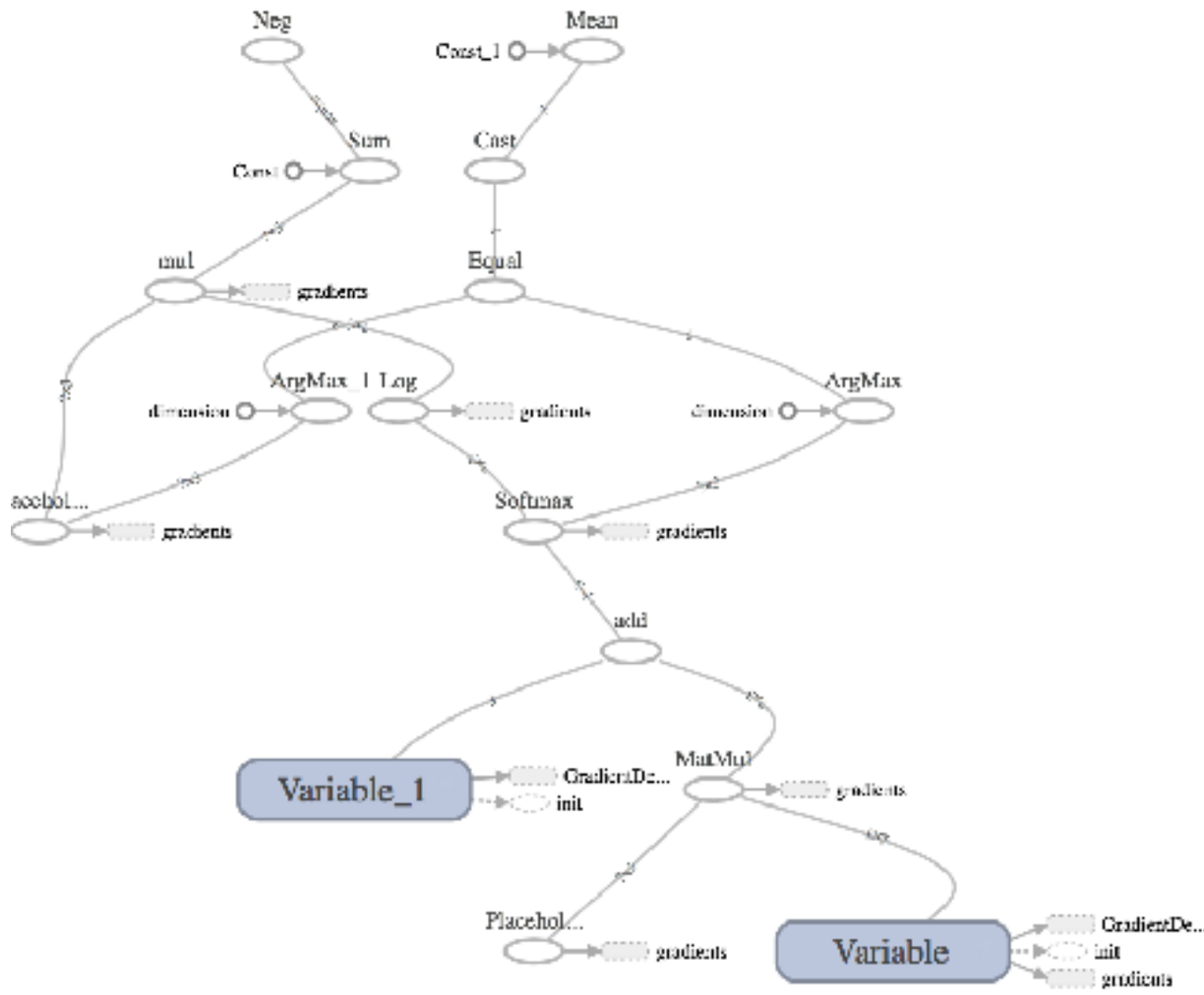
# learning

- train: cost function을 정의내리고 cost를 극소화하는 W,b를 찾는 것
  - OLS의 cost function: square sum of errors
  - 여기에서는 entropy 로 정의
- y: real value
- y_hat: predicted value

```
# model learning

cross_entropy = -tf.reduce_sum(y*tf.log(y_hat))
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(cross_entropy)
```

# TensorBoard

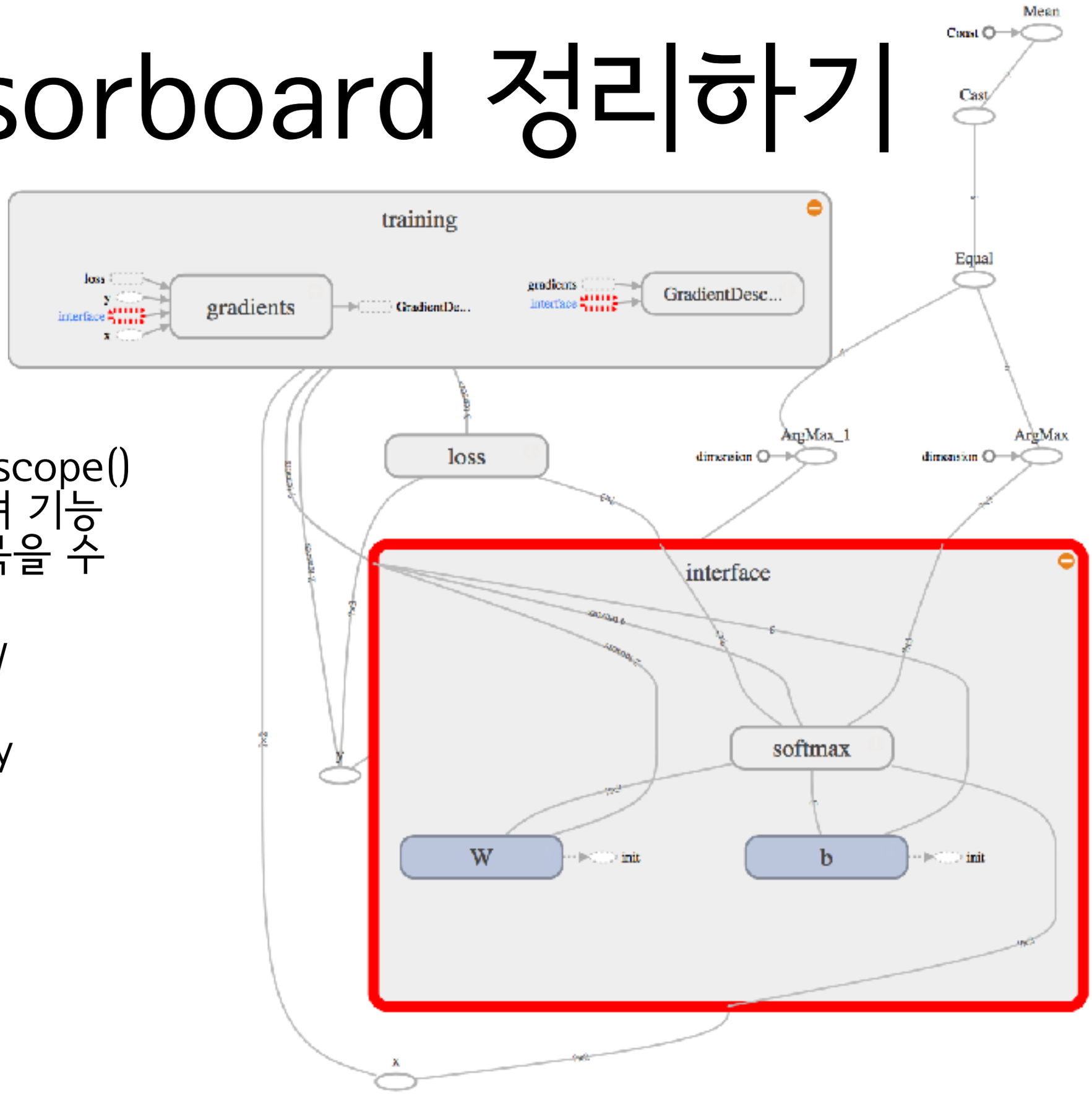
- 데이터 플로우 시각화 도구
- train 실행시 SummaryWriter를 병행 실행
- 첫번째 인자 디렉토리에 정보 저장
- tensorboard 명령어로 실행 → 브라우저상에서 확인
- tf 1.0 이후 부터는 tf.train.SummaryWriter 가 아니라 tf.summary.FileWriter임.




_sandbox/_ch05/bmi_tb.py

# Tensorboard 정리하기

- tb.name_scope()를 사용하여 기능 모듈별로 묶을 수 있음
- _sandbox/_ch05/bmi_tb2.py



# Keras

- https://keras.io/
- Deep Learning library for Theano and TensorFlow
  - $ pip3 install keras
  - ~/.keras/keras.json # 설정파일
- 좀 더 모델에 집중할 수 있도록 모듈화

```
{
    "image_dim_ordering": "tf",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "tensorflow"
}
```

# ex) keras-handwriting.py

- _sandbox/_ch05/keras-handwriting.py
- MNIST 손글씨 데이터 사용
  - 훈련 데이터#60000
  - 테스트 데이터#10000
  - http://yann.lecun.com/exdb/mnist/
- 핵심은 model
  - 각 층을 add로 추가 → compile → fit

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import Adam
from keras.utils import np_utils

# reading mnist data

(X_train, y_train), (X_test, y_test) = mnist.load_data()


# data postprocessing

X_train = X_train.reshape(60000,784).astype('float32')
X_test = X_test.reshape(10000,784).astype('float')
X_train /= 255
X_test /= 255

# converting label data to array type

y_train = np_utils.to_categorical(y_train,10)
y_test = np_utils.to_categorical(y_test,10)

# defining model structure

model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))


# making model

model.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(),
    metrics=['accuracy'],
    )

# training

hist = model.fit(X_train, y_train)

# testing

score = model.evaluate(X_test,y_test,verbose=1)
print('loss=',score[0])
print('accuracy=',score[1])
```